NM-SESES

Multiphysics Software for Computer Aided Engineering

Tutorial

Version September 2012

Written by:

J. Borth
T. Graf
T. Hocker
E. Lang
B. Neuenschwander
B. Ruhstaller
G. Sartoris
H. Schwarzenbach

NM Numerical Modelling GmbH
Böhnirainstrasse 12
CH-8800 Thalwil
Switzerland
http://www.nmtec.ch
mail://info@nmtec.ch

NM wishes to record its gratitude to the numerous individuals that contributed, either through discussions, proposals, support or testing, to the development of this project. In particular, we wish to thank Prof. Dr. Edoardo Anderheggen who has over many years supported the design and development of *SESES*. The following institutions supported the authors during the initial development of *SESES*: Swiss Federal Institute of Technology in Zürich, the Commission for Technology and Innovation of the Swiss Federal Government and Landis & Gyr Zug.

Contents

Introduction

Computer Aided Engineering (CAE) focuses on the application of physical-mathematical-numerical models to solve real-world multiphysics engineering problems. The past decade has been a period of rapid progress for CAE. The area of possible applications has been expanded and numerical methods have become increasingly sophisticated and adapted to exploit the available computational power of modern microprocessors. In particular, nowadays CAE-methods play a key role in the industrial product development process and help to

- Speed-up the development cycle of products.
- Optimize the product performance.
- Develop a thorough understanding of the underlying physical-chemical processes.
- Visualize device functionality not accessible through experiments.
- Support the decision making process at different product development stages.
- Eliminate potential failures and identify pitfalls.

In general, the design of a physical-mathematical-numerical model for the solution of an engineering multiphysics problem by CAE tools is a creative and often challenging task. For practical applications, simple recipes usually do not exist. Rather, it is the combination of physical insight, a sound basis in mathematics and last but not least comprehensive modeling experience that leads to the *correct* model and strategies for model validation, respectively. Often a problem can be described by a network model on a high level of abstraction. However, when it comes to understand the physical behavior and properties of system components, the modeling approach is often expressed by formulating *governing equations* consisting of the following parts.

- **Conservation Laws**: Exist for quantities such as mass, number of molecules, electrical charge, momentum and energy. Conservation laws are universal, i.e. they are independent of the materials considered and they always need to be satisfied. As an example, the conservation laws in fluid mechanics are the mass and momentum balances, of which former is also known as the continuity equation. In thermodynamics, the energy and entropy balances additionally come into play. These balance equations are also known as the first and second laws of thermodynamics.

- **Material Laws**: Balance equations by themselves are insufficient for a complete problem description, i. e. they usually contain more unknowns than available equations. Furthermore, they often contain unknowns such a flux-quantities that cannot be directly measured. However, it turns out that further relationships exist between these unknowns, i. e., they are not independent of each other. These relationships are called *material laws* or *constitutive equations*. As the name suggests, they do not hold universally, but only for particular materials or material classes, such as the class of ideal gases or the class of incompressible fluids.
- **Boundary Conditions**: Are required to *cut-off* a model, i. e. to separate the modeling domain from its surroundings. The number of boundary conditions required is determined by the degree (highest derivative) of the governing equations involved. Often, the solution strongly depends on the chosen boundary conditions and an improper choice may render the results useless, irrespective of the numerical accuracy. Using the wrong boundary conditions means that the model is incorrect or its connection to the exterior world is not correctly specified.

Combining these ingredients together, we obtain a coupled system of governing equations belonging almost exclusively to the class of partial differential equations (PDEs). In light of the specific geometries and boundary conditions that appear in realistic problems and the complexity of material laws, it is no surprise that analytical solutions to these problems can rarely be found. This in turn, motivates the use of numerical tools. In particular, the *SESES* software package allows for the numerical simulation of complex physical problems in various fields of applications as

- Micro-Opto-Electro-Mechanical systems.
- Magnetic field and eddy current simulations.
- Piezoelectric actuators and sensors.
- Electrochemical processes (fuel cells, batteries).
- Microfluidics (microreactors, membranes, flow sensors).
- Thermal transport by convection, conduction and radiation.
- Semiconductor devices.

Almost all of these equations can be coupled and solved together thus allowing the modeling and analysis of complex multiphysics problems. A rather complex example for an application of *SESES* is the numerical simulation of fuel cells, where the Navier-Stokes equations, the kinetics of the electro-chemical reactions and heat generation are all coupled together. The major effects here are the production and consumption of species and heat on the one hand and the separation of charges across the electrochemical double layer at the triple phase boundary on the other hand. This conversion process from chemical into thermal and electric energy strongly couples the different transport phenomena.

The syntax requirements for the input files, the numerical methods and the physical models for the different application fields are described in detail in the *SESES* user

manual. While the user manual serves as reference for everyday use, this tutorial represents a step-by-step introduction to solve design and optimization problems. The basic concepts are explained with the help of typical application examples. Several selected *SESES* features and postprocessing techniques, allowing the information obtained with the numerical simulation to be passed to other applications, such as Matlab or Excel, will also be illustrated. Thus, the reader of this tutorial will learn how to use *SESES* to solve engineering problems by describing them in mathematical terms (modeling), executing the numerical calculations (simulating), analyzing the data in a useful format (postprocessing) and how to generate new designs (optimizing).

Chapter 1

Getting Started

In this chapter, the reader will learn the basics of *SESES*. We will start by reviewing the *SESES* environment and after presenting some basic strategies for multiphysics modeling, a first example is presented to check a correct running environment. Although the example is not thoroughly explained, it serves as an introduction on how to use the *SESES* Front End program, how to run a numerical computation and how to visualize numerical results. Afterwards, we will present a second example in more detail by explaining the meaning of the input statements used to describe the modeling problem as well as the statements required to compute and display numerical solutions. We then continue to review the basics of *SESES* by presenting and explaining with simple examples the most common features that a user frequently applies when solving engineering problems. As an example, we review how to define algebraically a mesh of finite elements or how to solve non-linear governing equations. However, for a complete review of all available features and possibilities, the user manual should be consulted.

1.1 The *SESES* Environment

Fig. 1.1 shows a schematic overview of the simulation environment. The basic concepts, the Front End and the computational Kernel programs are thoroughly documented in the manual, here they will be briefly discussed and illustrated with an example. The `Seses` and `SesesSetUp` files are input files and for convenience, they may all be defined in a single container file. The modeling problem is specified inside the initial section of the `Seses` file, while numerical computations, data extraction, post-processing and similar tasks are specified in the command section. In the optional `SesesSetUp` file, settings of the Front End program are stored. The `Data` files contain the simulation data used to visualize numerical results. Please note, that the use of an input container file has the advantage that it can be either processed by the Front End program or executed as a script file or passed as argument to the Kernel program in a command-line fashion without invoking the Front End.

The Front End program is an interactive GUI program for the definition and visualization of simulation domains, finite element meshes and numerical results. Within this

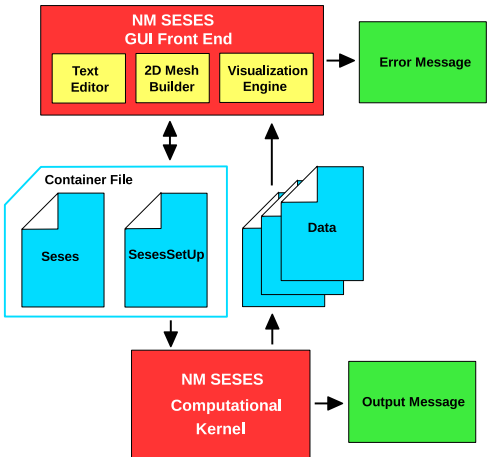

Figure 1.1: Schematic overview of the *SESES* simulation environment.

application, it is possible to edit the input file, to graphically construct a 2D mesh , to start a simulation in the background and to visualize numerical results, as shown in the Fig. 1.3, 1.4, 1.5. The computational Kernel program is a batch oriented program reading the input file and generating results by sending information to the output stream or by writing data files.

Modeling Strategy

We briefly discuss here the solution approach common to many design and optimization problems. The numerical modeling process can be subdivided into the following tasks:

- **Geometry Specification:** The geometrical domain relevant to the problem must be identified. Check if geometrical symmetries exist that allow a reduction of the computational effort.
- **Physical Model Specification:** Identify the governing equations relevant to the problem. Check if the governing equations and their variables are coupled among each other. If yes, decide on the solution algorithms.
- **Calculation:** Launch the calculation and inspect the convergence and accuracy of the results. For validation, check if similar but simpler problems with analytical solutions exist.
- **Postprocessing:** Post-process the numerical results for data extraction and visualization.
- **Optimization:** Modify the problem specification in order to find a design leading to improved device performances.

This solution approach is reflected in the structure of the input file. Namely, the initial section is structured in parts as follows:

- **Parameter definition:** Parameters can be defined for use in the material laws and geometry definitions.
- **Material definition:** Materials are defined by specifying material laws and parameters that are responsible for the physical properties.
- **Geometry definition:** The problem geometry and dimension is specified.
- **Material mapping:** Materials are assigned to local or global domains of the defined geometry.
- **Boundary condition (BC) definition:** The boundary conditions for the problem domain need to be specified which basically consider the interactions with the outside world i.e. where no modeling is performed.
- **Minimum Refinement Level:** The refinement level selects the number of refinement steps starting from the initially coarse mesh. A minimum refinement level can be set.

The command section contains the following important parts:

- **Solution procedure:** The computation of numerical solutions.
- **Accuracy:** Adaptively refine the computational mesh to obtain solutions with enough accuracy.
- **Post-processing:** Writing of computed fields into a graphics data file `Data` for visualization inside the Front End program and generation of relevant data for further post-processing tasks.

This typical structure of the input files is displayed with the help of comments in the tutorial examples included within the distribution. The input files are written in an input description language whose keywords are highlighted in blue and comments in violet inside the text editor. In case of syntax errors, the first invalid input line is highlighted in red. For detailed informations on the input syntax, please consult the manual explaining the syntax requirements with the help of railway diagrams.

Starting the Front End program

The Front End program can be launched by clicking the executables `g2d` or `g3d` inside a browser or by entering in a shell the command `g2d <file>` or `g3d <file>` with an optional file name. If the input file is not specified, you can open one by selecting the open button. However, if the associations for the file endings `.s2d`, `.s3d` and the executables `g2d`, `g3d` are set appropriately, the most common way to start the Front End program is by clicking on the input file inside a browser.

To illustrate the navigation, simulation and visualization, we shall next consider a simulation example of a lake with a dam. The situation is depicted in Fig. 1.2 and the input file is found at `example/Dam.s2d`. For this problem, we want to compute and visualize the mechanical stress and displacement in the 2D cross-section of the dam

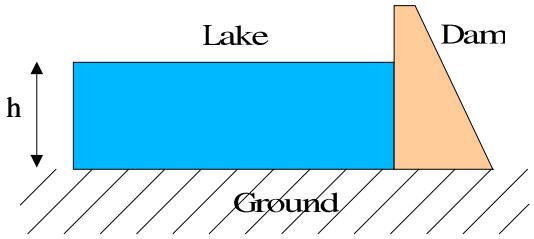

Figure 1.2: Schematic overview of the introductory example of a dam exposed to the water pressure of a lake.

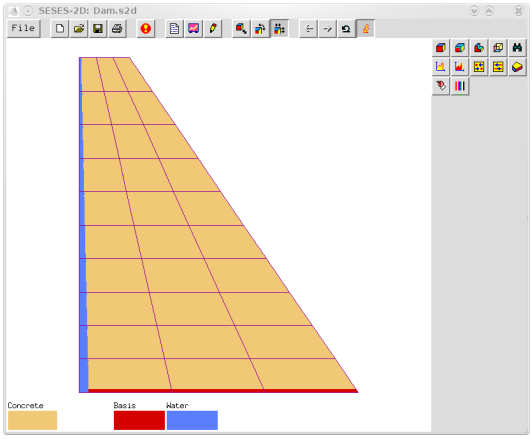

Figure 1.3: The Front End program showing the dam example `example/Dam.s2d`.

caused by the water pressure of the lake. Upon opening the input file, the Front End program will look like Fig. 1.3, showing the mesh geometry of the problem with a legend of assigned materials on the lower left corner.

Viewing and Editing the Input

We may want to inspect the content of the input file, namely the problem description and commands for the solution algorithm, by starting the text editor with the edit button has shown in Fig. 1.4. Within the editor, syntax highlighting causes syntax keywords to appear in blue, comments delimited by $(*$ and $*)$ in violet and the remaining text in black. The syntax examples in this tutorial, however, are reproduced without syntax highlighting. The input file of our first example is a container file starting with

```
cat > Seses <<EOF
```

ending with

```
Finish
EOF
```

and defining the input file. Optionally, a similar section defining the input `SesesSetUp` file may follow containing settings for the Front End program. The input container file ends with a section only used when the input file is used as a shell script. Here the Kernel `k2d` is started to run the simulation.

```
### Running Seses
k2d
```

Starting a Calculation and Viewing the Results

Rather than walking through all the details of this example, we proceed by commenting how to compute and display numerical results. Regardless of the active graphics or text mode, pressing the run button will start the calculation. We may now return

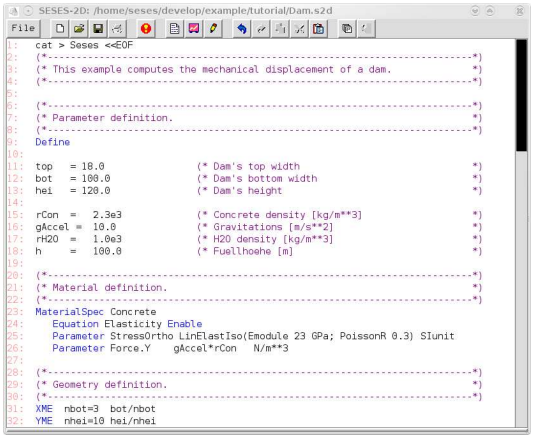

Figure 1.4: The text editor of the Front End program. The input file is displayed with syntax highlighting causing keywords to appear in blue and comments in violet.

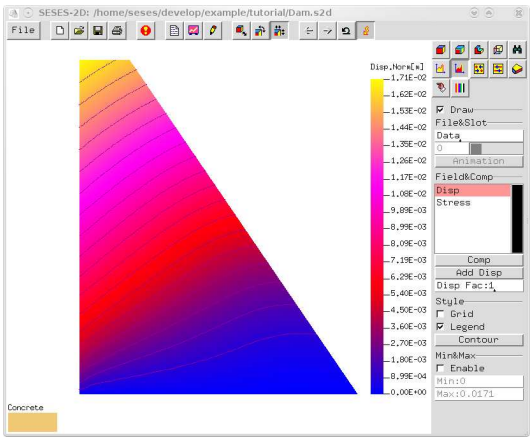

Figure 1.5: Graphical visualization within the Front End pogram showing the computed mechanical displacement.

to the graphics mode as displayed in Fig. 1.3, by pressing the graphics button. Here the geometry of the 2D problem is displayed. The boundary of the dam with the lake water is on the left-hand side, whereas the basis of the dam is at the bottom of the figure.

The content of the graphics window is updated manually by pressing the right mouse button or the graphics button or automatically by enabling automatic updates with the button. In the graphics mode, we can now inspect the computed fields by mapping the field values to the problem geometry. This visualization is launched with the field control, opening up a panel showing a list of computed fields available for display, see Fig. 1.5. Select the DRAW toggle and either the mechanical displacement DISP or the stress STRESS in the choice menu. After selecting the field DISP, the graphics window will look like Fig. 1.5. The cross-section of the dam is now colored according to the displacement values with the color legend given on the right side of the figure. Optionally, one can select the ADD DISP option causing the calculated displacement itself to be applied to the geometry. However, the mechanical displacement is negligible small relative to the dimensions of the dam so that to visualize the deformation, one has to choose a multiplicative factor of ca. 1000 with the DISP FAC: dialog. If the field STRESS is selected for visualization, then the graph reveals that the highest stress occurs at the lower left corner of the the dam, which faces the water.

You may wish to rerun this simulation with different parameters. For instance, find out which parameter values can be modified to give a mechanical displacement that is no longer negligible but leads to a significant distortion of the dam geometry due to the water pressure of the lake.

1.2 Building Your First Model

In the example of the previous section, the reader was introduced to navigating in the Front End program and running a simulation. In the next introductory example

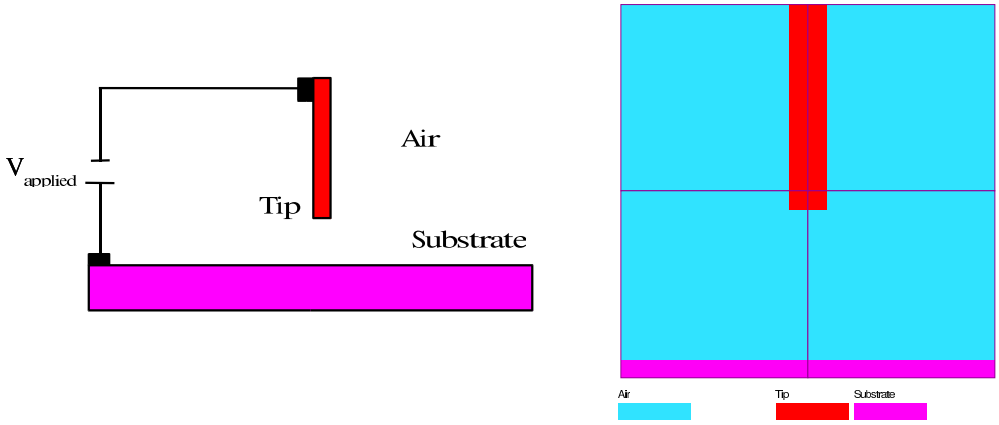

Figure 1.6: Illustration of the electrostatics problem for which a *SESES* simulation is being built and its representation in *SESES*.

found at `example/ElStatTip.s2d`, we will walk through the process of building a complete model, i.e. of writing an input file. We will document on how to implement the approach discussed in the section about the modeling strategy.

For this purpose, let us consider a 2D dielectric domain with two electrical contacts where an electrical bias is applied to. Our model shall correspond to the cross-section of a tip near a substrate surface at ambient air conditions, both contacted electrically as in the tunneling mode of a scanning probe microscopes, see Fig. 1.6. We wish to calculate the electrical potential and the corresponding electric field, denoted by the *SESES* keywords `Phi` and `Efield`, respectively. We start our problem specification by writing the initial section of input file, which has the same structure as explained in the previous example. Although the syntax statements may follow in arbitrary order, we will consistently use the sequence introduced in the earlier section about the modeling strategy. Since we will not define parameters for this example, the first statement is the definition of the material between the tip and the sample surface, namely air

```
MaterialSpec Air
  Equation   ElectroStatic
  Parameter  EpsIso 1
```

We name our material `Air` and specify it as a dielectric material with an isotropic dielectric permittivity `EpsIso` of value 1 as in vacuum and correspondingly enable the electrostatic equation computing the potential field `Phi`. The unknown `Phi` is called a degree-of-freedom field, in short a dof-field. Next, we define a 2D domain consisting of a mesh with 2×2 macro elements with the statements

```
QMEI 2 1E-6 QMEJ 2 1E-6
```

The `QMEI` and `QMEJ` keywords are followed by the number of elements in the rectangular mesh in x and y direction, respectively, and the size of each mesh element in these directions. The syntax is

```
QMEI n dx
```

with `n` being the number of elements and `dx` the size of each element. Frequently one defines the total length parameter `xlen` and substitutes `xlen/n` for `dx`. Moreover, the

rectangular mesh can be transformed to other geometries with the help of mathematical maps and the `Coord` statement. Since we defined a simple and coarse mesh with only 4 elements but expect the electrostatic potential to vary strongly, for instance near the tip, we choose a mesh refinement level of 3 with the statement

```
MinimumRL 3
```

Therefore, each macro element is subdivided 3 times such that each original element is replaced by 8×8 elements. In order to specify the electrical contacts, we define the mathematical boundary conditions (BC) with the `BC` statement as follows

```
BC Tip        1 0 JType 1 Dirichlet Phi 1 V
BC Substrate  0 2 IType 2 Dirichlet Phi 0 V
```

We name our boundary conditions `Tip` and `Substrate`. To define the BC geometry we use the keywords `JType` and `IType` to state that a mesh segment between 2 mesh nodes in j- and i-direction is specified, respectively. For instance for the horizontal `Substrate`, we specify the starting node $(0, 2)$ and a length of 2 in the i-direction thus defining a boundary segment between the nodes $(0, 2)$ and $(2, 2)$. Among the different types of BCs, we select here the `Dirichlet` BC allowing to prescribe the value of a dof-field on the boundary. For applying an electrical bias of $1\,\mathrm{V}$ between the contacts, we therefore use `Dirichlet Phi 1 V` at the `Tip` and `Dirichlet Phi 0 V` at the `Substrate`. The `Finish` statement tells the parser to stop reading the initial section of the input file.

We now proceed with the command section in which the solution method is specified. The statement

```
Solve Stationary
```

asks for the stationary rather than unstationary solution to be computed. Since we solve for the single dof-field `Phi` and the problem is linear no other statements are required to compute a solution. Lastly, we can export the computed dof-field `Phi` and the electrical field `Efield` obtained from the relation $\mathbf{E}^{\Phi} = -\nabla\Phi$ to the graphics data file `Data` with the statement

```
Dump Phi Efield
```

In summary, we have generated the following input container file and once read by the Front End program, it will display the picture in Fig. 1.6.

```
cat > Seses <<EOF
MaterialSpec Air
  Equation   ElectroStatic
  Parameter  EpsIso 1
QMEI 2 1E-6 QMEJ 2 1E-6
MinimumRL 3
BC Tip        1 0 JType 1 Dirichlet Phi 1 V
BC Substrate  0 2 IType 2 Dirichlet Phi 0 V
Finish
Solve Stationary
Dump Phi Efield
Finish
EOF

k2d
```

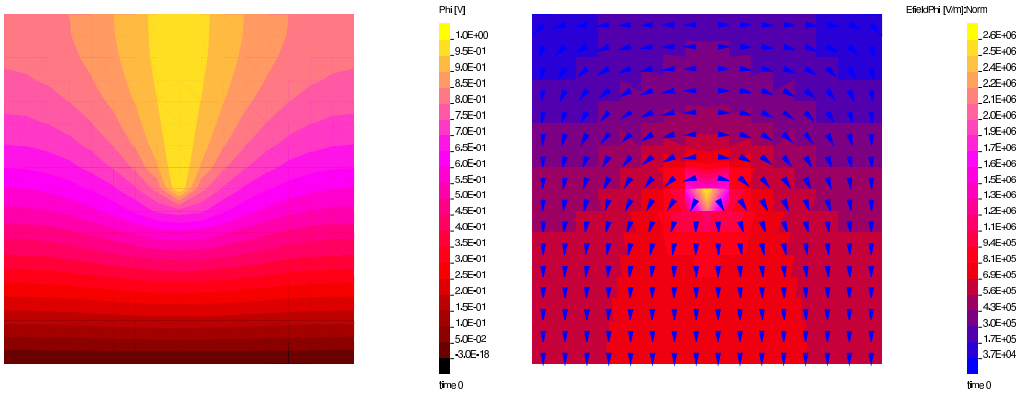

Figure 1.7: Calculated electrostatic potential and electric field upon application of a bias of 1 V between tip and substrate. The electric field is displayed with absolute values and arrows indicating the direction.

A simulation is started by pressing the run button. Runtime information will be displayed in the standard output and for this example the graphics data file `Data` will be generated. We may now investigate the simulation results by switching to the graphics mode with the graphics button and opening the field control panel with the button. Selecting the DRAW toggle and the electrostatic field `Phi` in the choice menu generates the content of Fig. 1.7. Obviously, the electrostatic potential transforms gradually between the two BC segments. In addition, we may also want to inspect the electric field by selecting `Efield`. The vector field direction can be visualized by selecting the ON ELEMENT option in the dot control. Note that the vector field direction is always perpendicular to the equipotential field lines and the highest electric field is located at the end of the tip facing the substrate surface. However, the value at the endpoint of the tip is strongly mesh dependent since for this simple model, the electric field is singular at this point.

1.3 Graphical Visualization

Although the *SESES* manual describes the Front End GUI program in details, for a beginner it is not always straightforward to find out the actions and doings required to visualize something. We have already presented some examples on how to display numerical fields so that here we mainly discuss how to choose a view, which is a task independent from the displayed content. For a 2D domain, this is not all that difficult since the default view shows the whole domain at once. However, for a 3D domain, this is not generally the case and one typically wishes to adjust the viewing angle and look from behind, look inside a device or limit the view for very complex structures. This will be explained in the following and the user is advised to open some 3D examples for practice and follow the instructions to quickly get acquainted with the tools.

For a 3D visualization, one of the most important tools is the capability to choose another viewing position i.e. to rotate or to move the displayed object. For this task one has to select the toggle button or. By moving the mouse while keeping the

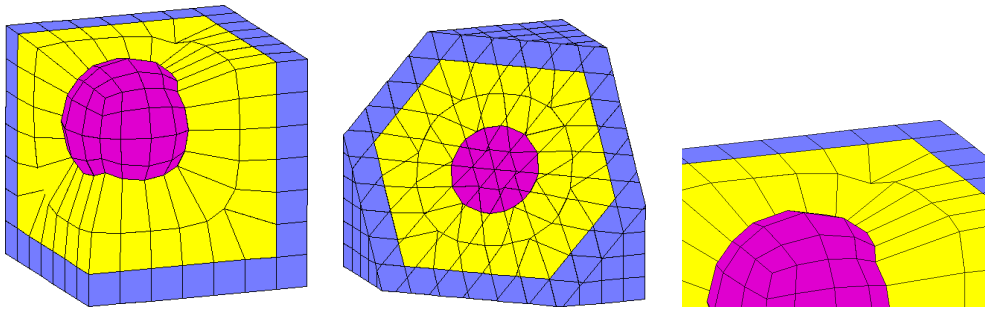

Figure 1.8: Illustration of some graphical visualization tools available in the Front End program. Rotating and reducing the mesh domain to be displayed (left), cutting through the mesh domain along an arbitrary plane (middle) and zooming (right).

first button pressed, you change either the viewing angle or the viewing position and you can quickly toggle between both choices by pressing the second button.

In order to look inside a 3D domain, there is the possibility to make a planar 3D-cut of the domain. This is done by selecting the toggle button and pressing the first mouse button to define the origin of the cutting plane. While keeping the mouse button pressed, you change the normal of the plane by moving the mouse and you shift the plane along its normal with the mouse wheel. To reset the 3D-cut view, just press the mouse button on a point not on the displayed object.

There are two possibilities to restrict the view. The first and most common one is to use the mouse wheel to zoom in and out. To reset the zoom, press once the reset button . A second possibility to restrict the view is given by selecting a subdomain of the macro element mesh. This is done by clicking on the domain button which opens a control panel to select combinations of user defined domains.

These tools for selecting a view are summarized in Fig. 1.8. Once you have found a proper view, you can save the settings by pressing the state button which opens up a control panel and then press the button STORE STATE. Each time you double click on the reset button , your default settings will be restored. If you press the button WRITE STATE, your settings will be written in the `SesesSetUp` file and restored each time you newly start the Front End program.

1.4 Animation

The Front End program offers the possibility to display animated sequences of numerical results. A prerequisite for an animation is a graphics file with multiple data slots. A slot is the data written in a single write operation and multiple slots are defined by appending data to a previously created file. Any type of numerical result can be animated supposed the data has been written to a graphics file. The choice of the independent parameter that effectively changes its value during animation is left to the user. For example, it may be the time or some other freely defined parameter.

Animation is typically used to quickly visualize the changes in the numerical solution due to different parameter values. In the example `example/ParamStudy.s2d` a

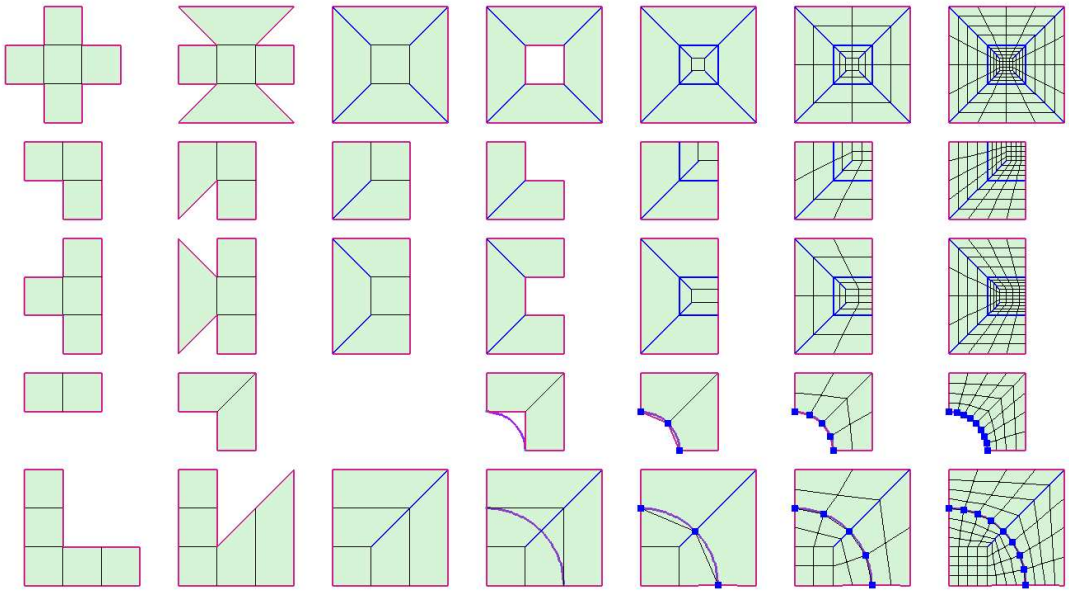

Figure 1.9: Construction process for meshes with local refinement and circular shapes.

sequence of calculations is performed using different values of the pressure from 0 to `PMax`. For each computed solution, a write operation is performed to write the values of temperature, displacement and stress fields to the graphics data file `Data`. The relevant input lines are the following

```
Define
Press = 1                    (* pressure (MPa)                          *)

Dump AtStep 1 Temp Disp Stress

Solve Stationary ForSimPar Press At 0 Step PMax/10#9
```

Note, that the sequence of these statements does matter. In particular, the statement `Dump AtStep 1 ...` to write the data for each computed solution must be defined prior computing solutions with the `Solve ...` statement. When viewing the results of our example in the draw window, an animation can be selected by choosing the ANIMATION option when one of the `Temp`, `Disp` or `Stress` field has been selected for visualization. A scrollbar is also available to go through the individual calculated data slots.

1.5 Preprocess Mesh Definition

Mesh generation is an important task and sometimes also a time consuming task, performed either with a preprocessing-static or algebraic-dynamic approach. For complex geometries and if the domain's shape is given and not part of the modeling problem, the preprocessing method is generally preferred. Here with the help of the built-in 2D mesh builder or another CAD and FE preprocessing tool, one creates a static mesh which is imported afterwards. For an external preprocessor, one can use the convenience routine `ReadMesh` to import unstructured meshes defined by the common

element-node data format. In this section, we shortly present useful tips when working with the built-in 2D mesh builder. You can experiment with its functionality and basic geometrical shapes with the example found at `example/MeshBuilder.s2d` and shown in Fig. 1.9. As a rule of thumb try to use a *divide and conquer* approach to construct the mesh, it is faster. As first example, since meshes are quickly refined, it is better to start working with coarse meshes. Secondly, since single mesh pieces are quickly joined together, try to exploit symmetries and start with one-quarter or one-half geometries. With cut&paste, mirroring and joining operations, the complete geometry is then quickly obtained.

We want to apply these simple rules for constructing meshes with local refinement and with a circular shape. The construction process is shown in Fig. 1.9 from left to right. As first, by selecting the *insert* task, a rectangular quad with just few macro elements is defined. Start drawing the quad with a double click defining the first boundary node, the other boundary nodes are defined by a single click and the last boundary node with a final double click. If something goes wrong, click on the *undo* button or press of the *ctrl-z* key combination and start again. As second, some nodes need to be moved at the correct position. Either with the *insert* task or with the *affine* task and option, click and hold down the first mouse button to select a node and drag it to its new position. As next, one has to join together macro element edges to define more generic shapes than simple quads. With the *join* task and the *join always* option, click and hold down the first mouse button on a edge and drag it over the edge to be joined until a snap takes place. Correctly joined edges are marked afterwards with a thick blue line. Be sure that edges drawn with a thick red line are just shown on the boundary and not inside the mesh. In this latter case, the two macro elements with a common thick red edge have not been joined and therefore the two macro elements are not topological neighbors, but there is a split hole in-between.

To create a mesh with local refinement as shown in the first three rows of Fig. 1.9, one applies recursively the constructed mesh. As first the mesh is duplicated by copy&paste. Within the *insert* task or the *affine* task and *move* option, select the mesh to be copied and either click the *copy* and *paste* buttons or press the *ctrl-c* and *ctrl-b* key combinations. The paste operation places the buffered object more or less at the actual pointer position. The copied mesh is automatically selected and you can move it at the correct position by picking and dragging one of its edges or nodes. As second, a macro element is deleted in the copied mesh and the original mesh is inserted in the created hole or indentation. Macro elements are deleted within the *insert* task, the *affine* task or whenever no task is enabled by selecting the macro element followed by a click of the *cut* button or by pressing the *ctrl-x* key combination. To insert the original mesh in the created hole or indentation, with the *join* task and *join fit* option, select the boundary to be joined, then pick one of its edge and drag it over the corresponding edge of the hole or indentation until a snap takes place. An automatic fitting of the inserted mesh is performed here. Again be sure that that just edges on the boundary are drawn with a thick red line are, otherwise click on the internal red edge to join it with its neighbor.

To create a mesh with a circular shape, one has first to define a NURBS curve with the *curve* task. This button is also a pulldown menu allowing the select the initial

drawing of a Bézier or NURBS curve. Due to symmetries, we just draw 90° arcs so that only two controlling points are necessary here. These are the starting and end arc points which are defined with a double click. By default these two points are connected by a straight line and to define a circular arc, open the *arc* panel and set an angle of 90°. Be sure you have selected a NURBS and not a Bézier curve, since this latter cannot exactly represents a circular arc. If not the case, select the two controlling points and apply the *rational segment* option. After a curve has been defined, one has to attach macro element nodes to the curve. With the *attach* task pick a macro element node, drag it over the curve until a snap takes place and move the node along the curve at the correct position. Attached nodes are marked afterwards with a thick blue point.

At the end one can quickly perform a uniform refinement with the *split* task. Select the whole mesh and pick a single edge of the selection. These simple shapes are easily combined together to form more complex meshes. Here, it is better to work with unrefined and coarse meshes and to perform the uniform refinement just as last operation on the whole mesh. The construction process is almost the same as for the basic shapes. With copy&paste operations, selections are duplicated and with the *join* task combined together. Here, you may first need to mirror the duplication along the x- or y-axis before joining, which is done by *affine* task and the *affine* panel.

1.6 Algebraic Mesh Definition

If the computational domain is non-constant and for example if its shape should be optimized, then working with a mesh constructed by a preprocessor is not well suited, since such meshes are hardly modified afterwards. For an automatic search of the optimum, better suited is to have a functional dependency of the domain's shape from some free parameters. For this algebraic method, the computational domain is constructed functionally with the help of the `QMEI`, `QMEJ` and `QMEK` statements defining an initial rectangular mesh and followed by a sequence of `Coord` statements defining geometrical maps for the node coordinates. The maps are defined by a functional expression of the latest node coordinates represented by the built-in variables `x,y,z` and the subdomain where the geometrical map is applied. Although this algebraic method is very flexible, one first needs to know the actual node coordinates and secondly the search of the proper geometrical map is not always a simple task. Here, several rectangular blocks of macro elements can be defined, individually transformed and later joined together with the statement `JoinME` to form a complex geometry. This is often more convenient than defining just one huge rectangular block and by deleting macro elements with the `DeleteME` statement.

Some 2D examples

In this section, we present some simple examples of defining 2D meshes with the help of geometrical maps. In the first example, we start with a rectangular mesh of dimension `lx*ly` and of `nx*ny` elements. We then translate a subdomain in the left half of our domain along the x-axis and rotate a subdomain of the right half about the

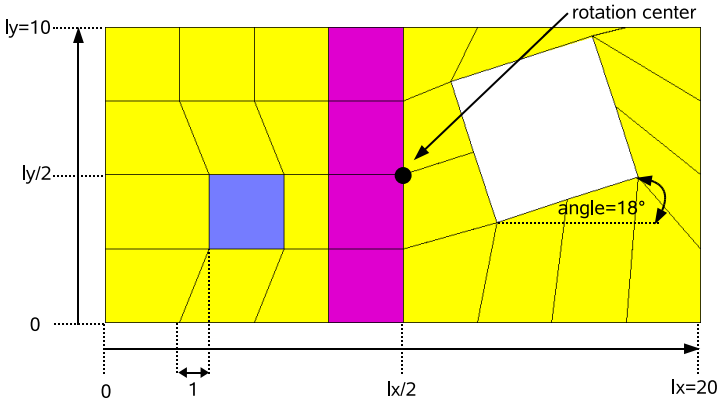

Figure 1.10: A mesh created with two `Coord` statements to perform a rotation and a translation and a `DeleteME` statement to create the void space.

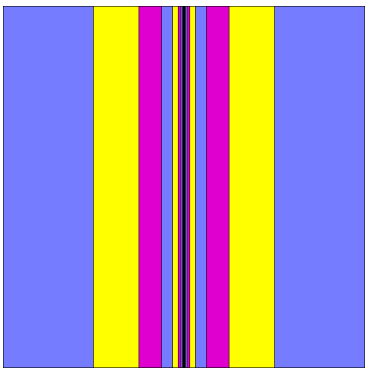

Figure 1.11: Manual mesh dimensioning according to a geometrical sequence.

angle `angle` with two `Coord` statements. A void space is created by deleting some elements with the `DeleteME` statement. Our final mesh is shown in Fig. 1.10 and it has been created with the statements

```
Define angle=PI/10
QMEI nx=8 (lx=20)/nx
QMEJ ny=4 (ly=10)/ny
Coord    x+1  y   block(1 2 1 2)
Coord    lx/2 + cos(angle)*(x-lx/2) - sin(angle)*(y-ly/2)
         ly/2 + sin(angle)*(x-lx/2) + cos(angle)*(y-ly/2)  block(5 7 1 3)
DeleteME block(5, 7, 1, 3)
```

Our next example illustrates a mesh defined using a function to adjust the width of neighboring elements according to a geometrical sequence. We first start by defining a 1D mesh $\mathsf{nx} \times 1$ with elements of side length 1. For domain x-coordinate $x \in (0, \mathsf{nx})$, we then change its value according to the function $\Phi(x)$

$$x := \Phi(x) = \begin{cases} 10(1 - \exp(-x \log(2)) & \text{for } x \geq \mathsf{nx}/2 \\ 10(1 - \exp((-\mathsf{nx} + x) \log(2)) & \text{for } x < \mathsf{nx}/2 \end{cases}$$

with the statements

```
QMEI nx=20 1 (* nx must be even *)
QMEJ ny=1  20
Coord 10*(1-exp(-x*log(2)))      y block(0   nx/2-1 0 ny)
Coord 10*(1+exp((-nx+x)*log(2))) y block(nx/2+1 nx  0 ny)
```

The resulting mesh is shown in Fig. 1.11. This type of mesh dimensioning with large variations in the element size is used when one knows in advance that a high density of elements is required somewhere to accurately compute solutions. Although an experienced user may easily locate these critical regions, as for example around corners, tips and boundary layers, this information is not always available or simply the task to devise suitable element meshes may be clumsy. Therefore one generally resorts to automatic procedures to create computational meshes as will be illustrated in some examples of this tutorial.

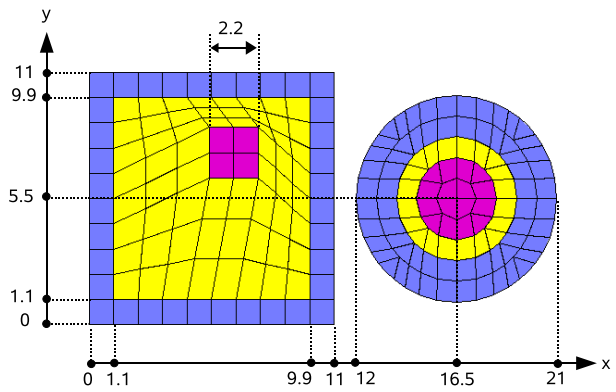

Figure 1.12: Example of the 2D geometrical maps with `bump` and `sphere`.

Predefined geometrical maps

The flexibility of algebraic approach defining geometrical maps to change the initial rectangular shape is counterbalanced by the fact that one has to find the proper map to obtain the shape of choice. To simplify this task and for commonly used geometries, the standard distribution provides the include file `Homotopic.sfc` defining several useful geometrical maps. For instance, this file contains the definition of a rotation function called `rot` and rotating mesh elements about the origin. Thus, in the input file of the example of Fig. 1.10, one may simply use the statements

```
Include "Homotopic.sfc"
Coord x+1  y                                block(1 2 1 2)
Coord (lx,ly)/2+rot(x-lx/2,y-ly/2,angle) block(5 7 1 3)
```

to obtain the same geometry. The next example is a little more involved and uses the function `bump` for graded translation on the first 10×10 block and the function `sphere` on the second 10×10 block to create the shapes of Fig. 1.12 with the following statements

```
Include "Homotopic.sfc"
QMEI  10 dx=1.1 QMEJ 10 dy=1.1
Coord coord+(1,2)*bump(x,1*dx,4*dx,6*dx,9*dx,1)
                 *bump(y,1*dy,4*dy,6*dy,9*dy,1)  block(Block 0)
QMEI Start 12 10 dx=0.9 QMEJ Start 1 10 dy=0.9
Coord sphere(coord,5*dx+12,5*dy+1,1) block(Block 1)
```

In the remainder of this section, we will describe the actions of some functions defined inside `Homotopic.sfc` and apply them to a 3D rectangular $10 \times 10 \times 10$ macro element mesh. These are the statements to start with

```
Include "Homotopic.sfc"
QMEI 10 1 QMEJ 10 1 QMEK 10 1
```

The mesh distortion is then performed with the `Coord` statement and the functions of `Homotopic.sfc`. It is surprising, that with only a few well defined functions combined together, many different geometries are easily obtained.

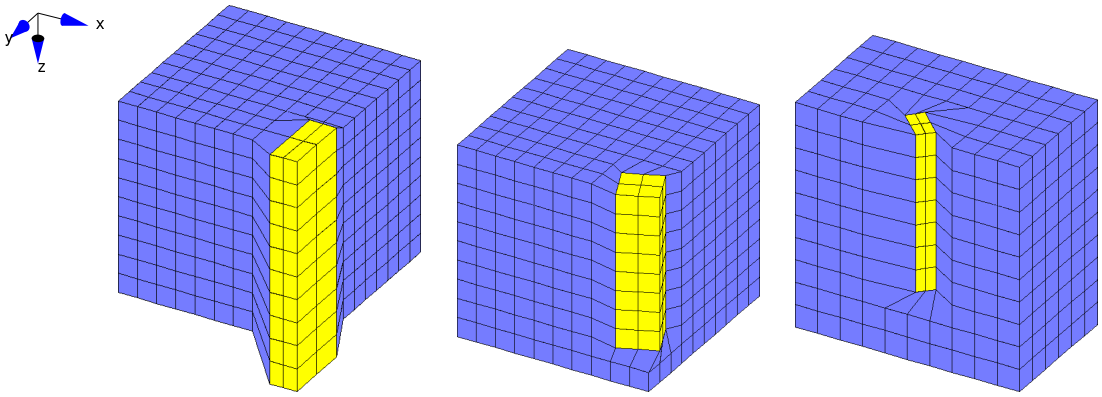

Figure 1.13: Direct scaling, rotation with `rot` and combination with `affine`.

Translating, rotating and scaling

These geometric maps are generally directly defined using the available vector algebra once the translation vectors, the rotation matrices and the scaling factors are known and so the role of `Homotopic.sfc` is limited to provide utility functions to easily define rotation matrices. In particular, the function

```
Routine double rot[T1](double val[T1],double alpha,double axis[T1])
```

defines a rotation of the vector `val` around the axis `axis` of angle `alpha` and returns the rotated vector. The rotation axis does not need to be a normalized vector. If the rotation is not with respect to the origin, then one calls `rot` by first subtracting the rotation point which is again added after the call. This is done by the function `affine` for the rotation center `cen` with an additional single scaling of the coordinates by `fac` as follows

```
Routine double affine[T1](double val[T1],double cen[T1],double fac[T1],
                          double alpha, double axis[T1])
{
   return cen+rot((val[0]-cen[0])*fac[0],
                  (val[1]-cen[1])*fac[1],
                  (val[2]-cen[2])*fac[2],alpha,axis);
}
```

Direct scaling and first example of Fig. 1.13:

```
Coord 10.0+(x-10.0)*0.7,8.5+(y-8.5)*1.8,+1.5+(z-1.5)*1.2 block(8 10 8 10 0 10)
```

Rotation with `rot` and second example of Fig. 1.13:

```
Coord rot(coord-(9,9,0),PI/8,0,0,1)+(9,9,0) block(8 10 8 10 0 8)
```

Affine map with `affine` and third example of Fig. 1.13:

```
Coord affine(coord,5,5,0.5, 0.4,0.8,1, PI/4,0,0,1) block(4 6 4 6 0 7)
```

Gradual mapping

It is sometimes useful to gradually apply a geometric map in order to obtain smooth results and for this purpose simple shape functions are required. We have defined the ramp function

```
Routine double ramp(double val,double type)
```

as the function $\mathcal{R}(\nu)$ with $\mathcal{R}((-\infty,0]) = 0$, $\mathcal{R}([1,\infty)) = 1$ and otherwise continuous between on $[0,1]$ with the degree of smoothness determined by the parameter `type`. With this function, we can define the bump function

```
Routine double bump(double val,double v[4],double type)
```

as $\mathcal{B}(\nu) = \mathcal{R}((\nu-\nu_0)/(\nu_1-\nu_0))(1-\mathcal{R}((\nu-\nu_2)/(\nu_3-\nu_2)))$ with values $\mathcal{B}((-\infty,\nu_0]\cup[\nu_3,\infty)]) = 0$, $\mathcal{B}([\nu_1,\nu_2]) = 1$ and the previous ramp behavior between $[\nu_0,\nu_1]$ and $[\nu_1,\nu_2]$. This bump function is typically used to apply a full map within the interval $[\nu_1,\nu_2]$ but only gradually in the intervals $[\nu_0,\nu_1]$, $[\nu_2,\nu_3]$ and not at all outside $[\nu_0,\nu_3]$. A last useful shape function is the following double triangle function, zero outside $[-1,1]$, antisymmetric with respect to the origin and with the maximum of 1 at `vmax`

```
Routine double triangle(double val,double vmax)
{
   double aval=abs(val);
   return (val<0?-1:1)*(aval<vmax?aval/vmax:aval<1?(1-aval)/(1-vmax):0);
}
```

which is used to define the function

```
Routine double dilat(double val,double c,double sc,double sm,double d)
{
   return val+(d-sc)*triangle((val-c)/sm,sc/sm);
}
```

The `dilat` function leaves the point c and region outside the interval $[c-sm, c+sm]$ invariant, it maps this interval onto itself and the points $c \pm sc$ are moved to $c \pm d$ and it is typically used to construct pyramidal geometries.

Shifting and stretching with `bump` and first example of Fig. 1.14:

```
Coord  coord+(1.5,1,-3)*bump(x,1,4,6,9,1)*bump(y,1,4,6,9,1)*
                        bump(z,-1,0,9,10,1)  block(1 9 1 9 0 10)
```

Shifting and stretching with `dilat` and second example of Fig. 1.14:

```
Coord  x,dilat(y,5,5,4,x/2+0.5),dilat(z,5,5,4,x/2+0.5) 1
```

Shifting and stretching with `dilat` and third example of Fig. 1.14:

```
Macro ZD { ((z-1)/8*3+2) }
Coord dilat(x,0, 5,4.9,@ZD),dilat(y,0, 5,4.9,@ZD),z block(0 4  0 4  1 9)
Coord dilat(x,0, 5,4.9,@ZD),dilat(y,10,5,4.9,@ZD),z block(0 4  6 10 1 9)
Coord dilat(x,10,5,4.9,@ZD),dilat(y,0, 5,4.9,@ZD),z block(6 10 0 4  1 9)
Coord dilat(x,10,5,4.9,@ZD),dilat(y,10,5,4.9,@ZD),z block(6 10 6 10 1 9)
DeleteME block(4,6,0,10,1,9)||block(0,10,4,6,1, 9)
```

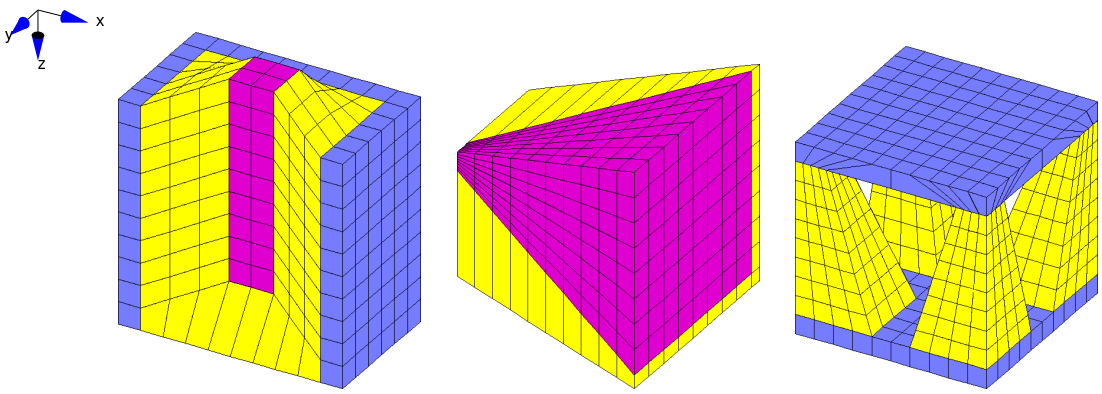

Figure 1.14: Gradual shifting and stretching.

Spherical geometries

The basic function used to define spherical geometries is given by

```
Routine double sphere[T1](double val[T1],double cen[T1],double mp)
```

which gradually maps any cube surface centered at `cen` into its in-sphere depending on the morphing parameter `mp`. If `mp` $= 0$ the cube surface is left unchanged and for `mp` $= 1$ we obtain the in-sphere. To obtain a ball centered at `cen`, just apply this function to a whole cube centered at the same point with `mp` $= 1$. The next function

```
Routine double spheremix[T1](double val[T1],double cen[T1],double rs,
                             double rq,double type)
```

uses the function `sphere` with a non-constant morphing parameter to gradually maps cube surfaces into in-spheres. For `rs`$<$`rq`, if the half-side of the cube is larger than `rq`, the cube surface is left invariant and for values less than `rs`, you will get exact spheres so that you actually obtain a ball in a cube. For `rs`$>$`rq`, the role is reversed and you obtain a cube in a ball. The parameter `type` determines the ramp used to gradually apply the morphing parameter.

A cube is gradually transformed with respect to the x-coordinate, first example of Fig. 1.15:

```
Coord  sphere(coord,5,5,5,x/10) 1
```

A ball in a cube, second example of Fig. 1.15:

```
Coord spheremix(coord,5,5,5,2,4,2) block(1 9 1 9 1 9)
```

A cube in a ball, third example of Fig. 1.15:

```
Coord spheremix(coord,5,5,5,3,2,1) block(1 9 1 9 1 9)
```

Cylindrical geometries

The cylindrical maps are equivalent to the 2D spherical maps and for each sphere function there are three `x`, `y`, or `z` versions corresponding to the yz-, xz-, xy-planes

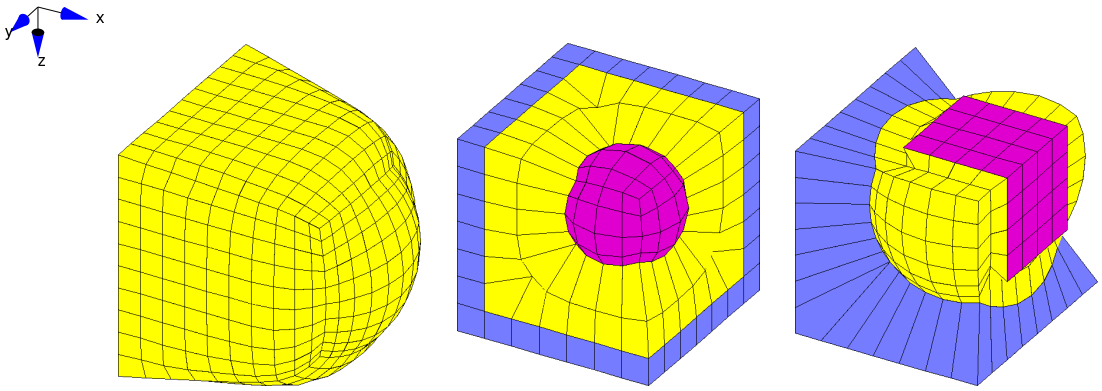

Figure 1.15: Sphere examples with `sphere` and `spheremix`.

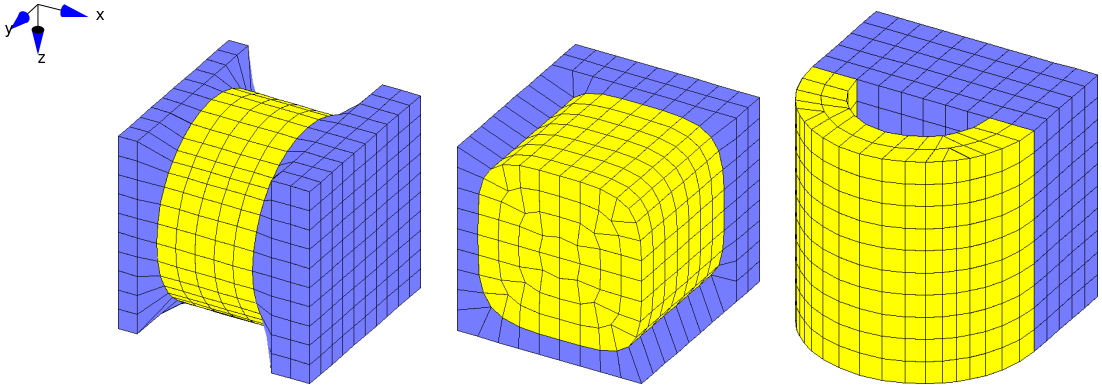

Figure 1.16: Cylindrical geometries with `cylx,cyly,cylz`.

where the 2D sphere functions are applied. Since no new concepts are meet here, we limit ourself in giving application examples with function calls and graphical output.

First example of Fig. 1.16:

```
Coord cylx(coord,5,5,1) block(2 7 0 10 0 10)
```

Second example of Fig. 1.16:

```
Coord cyly(coord,5,5,0.5) block(1 9  3 10 1 9)
```

Third example of Fig. 1.16:

```
Coord cylz(coord,5,5,1) block(0 10 5 10 0 10)
DeleteME block(2, 8, 5, 8, 0, 10)
```

First example of Fig. 1.17:

```
Coord cylxmix(coord,5,5,3,5,2) 1
```

Second example of Fig. 1.17:

```
Coord cylymix(coord,5,5,3,5,2) 1
```

Third example of Fig. 1.17:

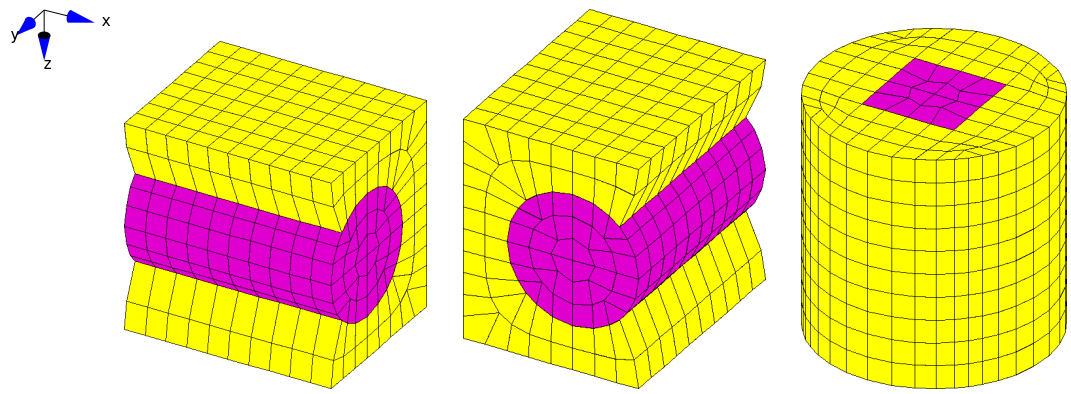

Figure 1.17: Cylindrical geometries with `cylxmix,cylymix,cylzmix`.

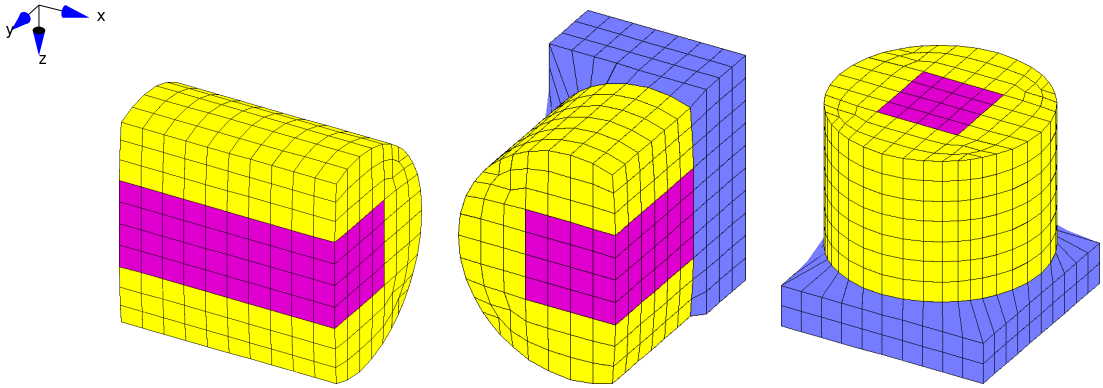

Figure 1.18: Cylindrical geometries with `cylxmix,cylymix,cylzmix`.

```
Coord cylzmix(coord,5,5,1,2,2) 1
Coord cylzmix(coord,5,5,5,2,2) block(0 2  0 10 0 10)
Coord cylzmix(coord,5,5,5,2,2) block(8 10 0 10 0 10)
Coord cylzmix(coord,5,5,5,2,2) block(3 7  0 2  0 10)
Coord cylzmix(coord,5,5,5,2,2) block(3 7  8 10 0 10)
```

First example of Fig. 1.18:

```
Coord cylxmix(coord,5,5,5,2,1) 1
```

Second example of Fig. 1.18:

```
Coord cylymix(coord,5,5,5,2,1) block(0 10 4 10 0 10)
```

Third example of Fig. 1.18:

```
Coord cylzmix(coord,5,5,5,2,1) block(0 10 0 10 0 7)
```

1.7 Non-Linear Algorithms

This section illustrates some of the available features when solving non-linear problems. A linear problem is solved in one single step by finding the solution to the linear system of equations obtained when discretizing the governing equations. Non-linear

problems, instead, are generally solved iteratively based on the solutions of single linear steps. Although for very large linear systems, iterative methods may as well be used to find their solutions, in the sequent we will consider the linear step as a black box and we will focus on the non-linear iteration. We start the discussion with the example `example/NonLin.s2d` and the simple linear 1D thermal problem for the temperature $T(x)$

$$\nabla.(\kappa \nabla T) = 1 \text{ with } x \in (0,1) \text{ and } T(0) = T(1) = 0\,.$$

The analytical solution for a heat conductivity of $\kappa = 1$ is simple $T = x^2/2 - x/2$. A solution to this problem is computed with the statement `Solve Stationary`. Before computing the solution, we have added a print statement to improve the understanding of the textual output.

```
Write "@@@ Solving for the first  time:"
Solve Stationary
```

The output for this solution step will be something similar to

```
Solving for the first  time:  AbsResid.Temp=9.64e-02
```

After printing our label, during the solution step the program writes the L^2-norm of the residuals or out-of-balance values for the temperature equation we are solving for. Since *SESES* identifies our thermal equation as linear, a single linear step is performed. By repeating the solution step a second time, one obtains

```
Solving for the second time:  AbsResid.Temp=1.71e-16
Solving for the third  time:  AbsResid.Temp=2.10e-16
```

The values of the residual norm depends on internal scaling factors of the governing equations and so a priori its absolute value may not say much. However, in this example, we see that when solving the for second and third time, the value is almost 15 orders of magnitude smaller than the first time and it does not change much with the third step. With the second and third step, the temperature value is already the solution of the governing equation and so the residuals are numerically zero. Due to the limited size of the number representation in computer memory, a numerical zero is almost never an algebraic zero and must always be interpreted with respect to numerical values being computed. The machine precision ϵ is defined as the largest number such that $1 + \epsilon = 1$ holds and for common applications we have $\epsilon \approx 10^{-15}$ which explains the drop of 15 orders of magnitude in the residual norm between the first and second solution step.

The convergence behavior can also be characterized with respect to increment norms i.e. the change in the numerical solution. Increment norms have the advantages to be easier to interprete since their reflect the change of the independent variables solved for and not the dependent out-of-balance values of the governing equations. We therefore repeat the previous example, by printing the L^2-increment norm.

```
Solve Init
Convergence { write("  AbsIncr.Temp=%.2e\n",AbsIncr.Temp); return 1; }
Write "@@@ Solving for the first  time:"
Solve Stationary
Write "@@@ Solving for the second time:"
Solve Stationary
Write "@@@ Solving for the third  time:"
Solve Stationary
```

As first step, we initialize again to zero the solution in order to destroy the previously computed one. Then we substitute the default printing statement displaying the residual norm with a statement printing the incremental norm of the temperature. This is done by defining a user convergence criterion and by embedding a call to the `write` function. Since the problem is still linear, the criterion always returns 1 meaning unconditional convergence. As last, we solve as before three times and obtain the following output.

```
Solution newly initialized
@@@ Solving for the first  time:  AbsIncr.Temp=1.86e+00
@@@ Solving for the second time:  AbsIncr.Temp=6.60e-16
@@@ Solving for the third  time:  AbsIncr.Temp=1.77e-16
```

The numerical behavior is the same as before, but now we can state that the temperature is computed up to a precision of 10^{-15}. This is the precision of the numerical solution when solving the discretized governing equations, which is by far much smaller than the precision with respect to the exact analytical solution of the governing equation.

Now that we have acquainted some feeling on norm values for numerical solutions of the linear thermal problem, we change its character and turn it into a non-linear problem. Here, we define the thermal conductivity to be a function of the temperature $\kappa = \kappa(T) = 1 + T$ and try to compute a solution.

```
MaterialSpec Silicon Parameter KappaIso  1+Temp W/(m*K)
Convergence  (* restore default *)
Write "@@@ Solving the non-linear problem:\n"
Solve Stationary
```

After restoring the default convergence criterion, this time the output will be something similar to

```
@@@ Solving the non-linear problem:
  AbsResid.Temp=8.73e-01
  AbsResid.Temp=5.52e-02
              .
              .
              .
  AbsResid.Temp=1.26e-04
  AbsResid.Temp=6.58e-05
  AbsResid.Temp=3.41e-05
  AbsResid.Temp=1.77e-05
@@@@ SESES SOFT ERROR: Coupled (Newton-Raphson) loop did not converge
```

SESES recognizes the new problem to be non-linear and so it automatically starts a Newton's iteration consisting of linear solution steps. The convergence of the residual norm is low and since after 15 steps the convergence criterion is not fulfilled, the Newton's algorithm gives up and returns. The coupled Newton's algorithm, however, converges in the vicinity of a solution quadratically but only if the derivatives of the residual equations with respect to the solution are available. For this problem, we have specified the heat conductivity to be a function of the temperature, but *SESES* is not designed to figure out the derivative. Actually with some programming effort it would be possible to obtain the derivative either in analytical form or numerically by difference. However, the former method is rather complex and may result in long and complex analytical expressions to evaluate while the latter may be numerically unstable. For these reasons, we do not try to obtain the derivative from the function

itself but the user should specify the derivative in order to get the correct convergence behavior. We thus provide the derivative and try to solve again the non-linear thermal problem.

```
MaterialSpec Silicon Parameter KappaIso D_Temp 1+Temp,1 W/(m*K)-W/(m*K*K)
Write "@@@ Solving with exact derivative:\n"
Solve Stationary
```

This time we obtain a quadratic convergence behavior.

```
@@@ Solving with exact derivative:
  AbsResid.Temp=9.64e-02
  AbsResid.Temp=8.73e-01
  AbsResid.Temp=2.06e-01
  AbsResid.Temp=3.42e-02
  AbsResid.Temp=1.61e-03
  AbsResid.Temp=3.74e-06
  AbsResid.Temp=1.56e-11
```

The default convergence criterion is a value of 10^{-6} for the residual norm and whenever the norm value is less than this tolerance, the Newton's iteration is successfully ended. This convergence criterion, however, can be specified as a function of any computable norm and the actual iteration number `nIter`. Since for our thermal problem an accuracy of $\approx 10^{-2}\,\mathrm{K}$ for the temperature may be considered as acceptable, we solve again by setting the new convergence criterion. We also visualize the convergence rate by using the built-in function `quot` to obtain the quotient of the function value in the previous iteration

```
Write "@@@ Solving with reduced accuracy:\n"
Convergence { write("  Step %2.0f AbsIncr.Temp=%e Rate=%e\n"
  nIter,AbsIncr.Temp,quot(AbsIncr.Temp)); return AbsIncr.Temp<1.E-2; }
Solve Stationary
```

For our reduced accuracy requirement, only 5 solution steps are now required.

```
@@@ Solving with reduced accuracy:
  Step  1 AbsIncr.Temp=1.855581e+00 Rate=inf
  Step  2 AbsIncr.Temp=8.451505e-01 Rate=4.554640e-01
  Step  3 AbsIncr.Temp=3.025883e-01 Rate=3.580289e-01
  Step  4 AbsIncr.Temp=5.419181e-02 Rate=1.790942e-01
  Step  5 AbsIncr.Temp=1.994451e-03 Rate=3.680355e-02
```

The effort required to solve a non-linear problem based on the full solution of linearized problems is the number of iteration times the almost constant effort to solve a single linear problem. Even for weak non-linear problems, the number of iterations required is generally in the order of 5 to 10 which makes the solution of non-linear problems computationally much more expensive. However, by changing the approach of solving the underlying linear problems, one can effectively solve at least weakly non-linear problems with little more effort than a single linear one. For optimized algorithms, the total effort is then determined by the character of the non-linearity. Devising optimized algorithms principally differs whether we are using a direct or an iterative linear solver. We present here a speed-up method based on the utilization of a direct linear solver which is also the solver used per default. However, methods are also available for iterative linear solvers. The general property of direct solvers is that the cost of solving a second linear system with the same matrix as the first system but a different right-hand-side vector can be done with much less effort then when

solving for the first time. This is because the numerical factorization of the system matrix is by far much more expensive than the final backward-forward substitution which has to be done for each different right-hand-side vector. In the first computed non-linear example, we have seen that with a wrong derivative the convergence rate of Newton's algorithm is slow and generally linear but the iteration may soon or later converges towards the convergence criterion. Fast algorithms are therefore obtained by updating the derivative and its inverse just when the convergence rate is slow and this implies avoiding factorizing the system matrix. We just keep the matrix computed in one of the previous steps. At each linear iteration step, we just update the residual values and this only implies changing the right-hand-side vector of the linear system to be solved. In *SESES*, this behavior is achieved by defining fast-increments meaning that the linear system matrix of the previous step is not to be changed.

In order for this example to converge by computing the derivative just in the first step, we need to cut the increments in the first step. In general, the increments computed by the Newton algorithm far from the solution are overestimated and can easily lead to divergence. In this example, by taking just one third of their values with `setincr(Incr/3)`, divergence in the following steps is avoided. After the first step, we specify for 100 times the computation of fast increments with `ReuseFactoriz` and again we are forced to divide their values by 4 in order to avoid divergence. Although some tricks were necessary to avoid divergence, the algorithm converges with a slow rate but each single step is fast since the linear system matrix is factorized just in the first step.

```
Write "@@@ Solving with fast increments:\n"
Convergence  { write("  Step %2.0f AbsIncr.Temp=%e Rate=%e\n"
  nIter,AbsIncr.Temp,quot(AbsIncr.Temp)); return AbsResid.Temp<1.E-15; }
Increment    setincr(Incr/3) ReuseFactoriz setincr(Incr/4) #100
Solve Stationary
```

For this small 1D example running very fast, the speed-up when using fast increments is not clearly identifiable, but for 2D and 3D large problems it will.

```
@@@ Solving with fast increments:
  Step  1 AbsIncr.Temp=1.855581e+00 Rate=inf
  Step  2 AbsIncr.Temp=2.489984e-01 Rate=1.341889e-01
  Step  3 AbsIncr.Temp=5.581262e-02 Rate=2.241485e-01
            .
            .
            .
  Step 25 AbsIncr.Temp=4.052745e-15 Rate=3.768345e-01
  Step 26 AbsIncr.Temp=2.515550e-15 Rate=6.207028e-01
  Step 27 AbsIncr.Temp=1.509828e-15 Rate=6.001977e-01
```

From the output, we see that the derivative computed in the first step is good enough to reach convergence although with a slow rate and by cutting the increments. If the convergence rate is too slow or if divergence shows up, then one has to update again the derivative. It is possible to do it on a static base by specifying the computation of, for example, once standard increment followed by three times fast-increments and by repeating this cycle. However, it is also possible to do it dynamically with the help of increment control.

```
Write "@@@ Solving with increment control:\n"
Increment   Standard      setincr(Incr/3)
            ReuseFactoriz Control quot(AbsResid.Temp)>0.1?1:0
            Standard      Control -1
Solve Stationary
```

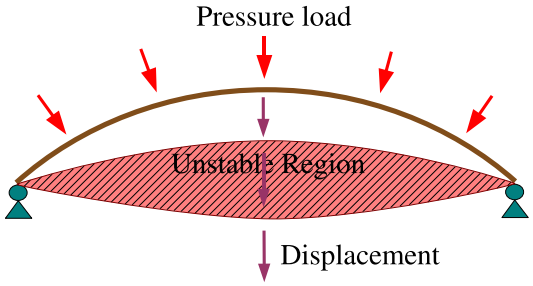

Figure 1.19: A cylindrical roof under pressure.

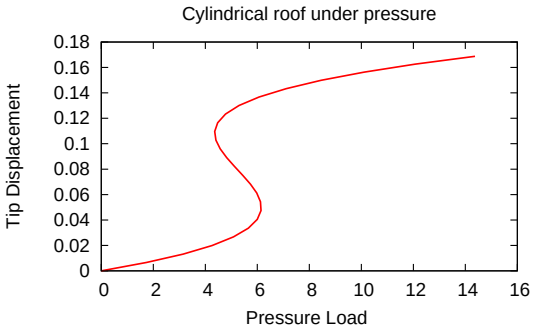

Figure 1.20: Load-displacement curve for the cylindrical roof under pressure load.

In the first step of this example, we compute standard increments and divide them by 3 to reduce initial overshooting. We then define fast increments and if the convergence rate falls below `0.1`, we update again the derivative by performing a full linear step. This is achieved with the `Control` statement allowing to jump back and forth between the different increment directives. The first increment directive `Standard setincr(Incr/3)` is executed just once at the beginning, then the second directive `ReuseFactoriz` takes place and the control function is evaluated to eventually jump to another directive. Here, if the convergence rate is below 0.1, we jump to the next directive `Standard` which performs a full linear step and jumps back to the second directive.

```
@@@ Solving with increment control:
  Step  1 AbsIncr.Temp=1.855581e+00 Rate=inf
  Step  2 AbsIncr.Temp=2.489984e-01 Rate=1.341889e-01
  Step  3 AbsIncr.Temp=1.545840e-01 Rate=6.208232e-01
  Step  4 AbsIncr.Temp=1.635748e-02 Rate=1.058161e-01
  Step  5 AbsIncr.Temp=5.149250e-03 Rate=3.147948e-01
  Step  6 AbsIncr.Temp=3.002182e-05 Rate=5.830330e-03
  Step  7 AbsIncr.Temp=3.765816e-07 Rate=1.254360e-02
  Step  8 AbsIncr.Temp=4.923727e-09 Rate=1.307479e-02
  Step  9 AbsIncr.Temp=6.560768e-11 Rate=1.332480e-02
  Step 10 AbsIncr.Temp=8.815400e-13 Rate=1.343654e-02
  Step 11 AbsIncr.Temp=1.189537e-14 Rate=1.349385e-02
  Step 12 AbsIncr.Temp=5.359217e-16 Rate=4.505298e-02
```

From the output, we can see the speed-up in the convergence rate as soon as we perform a full linear step updating the derivative and its inverse.

1.8 Continuation Methods

This section illustrates some of features available in *SESES* when computing families of solutions depending on user parameter. We will go through the discussion by presenting the example `example/ContMethod.s2d` of a cylindrical roof under a pressure load from above as displayed in Fig. 1.19. We will compute different solutions for different values of the applied pressure on the roof, i.e. a family of solutions parameterized by the pressure representing our simulation parameter. We assume longitudinal invariance of the mechanical structure so that beam elements can be used and we perform the analysis in a small strains but large displacements framework. This results in a non-linear problem that must be solved for each applied load. By zero load,

we have zero mechanical displacements and by increasing the load the roof starts to bend inwards. By further increasing the load, we reach the snap-through point characterized by the fact that locally no other solution exists for higher values of the load. If we want to find solutions with still larger displacements, we have to reduce the load but we also have to avoid turning back on our solution path. At this limit point, the mechanical structure becomes instable and will snap-through the next stable solution with the same load value. We will not perform here a dynamic analysis, but limit ourself to compute static solutions, although some of them may be unstable. We will see how limit points are hard to pass by and present methods to successfully compute the load-displacement curve.

Without discussing the details of non-linear mechanical models, not so important for an understanding of this example, the input file defines the roof geometry, material properties and models to enable a non-linear mechanical analysis and the pressure load as a boundary condition. To compute a family of solutions depending on the pressure load, we define the global variables `load` and `control`, but for the moment we will just consider the first one in order to specify a pressure load with the help of the `Pressure` BC. The definition of a global variable, let us choose its value before computing a solution. Therefore we start computing some solutions for values of `load`=1,2,3,4,5.

```
Write "@@@ Computing solutions for different loads\n"
Convergence (AbsIncr.Disp<1.E-12)?1:
            (nIter>=15)?failure("Too many iterations"):0
Solve Stationary ForSimPar load At 0 Step 1#5
```

Since the mechanical displacement is mainly in the vertical direction and the problem is non-linear, we have chosen the convergence criterion solely based on the convergence of the vertical displacement and a maximal number of 15 iterations. In the `Solve Stationary` statement, we select the simulation parameter `load` and define 5 steps with an increment of one. Six different solutions will be computed starting with a null value of the pressure load.

For this non-linear problem, the derivative of the residual equations with respect to the unknowns representing the displacement field are computed exactly, so that the Newton iteration converges quadratically. We may apply the techniques presented in the previous section using fast-increments to speed-up the computation, but here we present another method using a predictor-corrector technique. When a family of solution is computed with respect to a parameter and the dependency is smooth, one can use a Taylor expansion to approximate a solution for a new value of the parameter. This predicted solution is then corrected by starting the non-linear solution algorithm. Since we are closer to the solution less iterations are required and a speed-up is obtained. Actually Taylor expansions are not practical to use since they require the derivatives of the solution with respect to the parameter. Therefore, they are commonly replaced by polynomial or rational extrapolation close related to Taylor expansions and continued fractions. By repeating the previous solution and enabling extrapolation we effectively perform less iterations.

```
Write "@@@ Computing predicted solutions\n"
Solve Init
Extrapolation Rat_2_2
Solve Stationary ForSimPar load At 0 Step 1/2#10
```

In this example, we have used quite a high degree of rational extrapolation in order to reduce the computational cost at the expense of increased memory requirements needed to store the computed solutions. However, there are many examples, where high degree extrapolation is not better than low degree and can even be detrimental and this is especially true for polynomial extrapolation.

The pressure load of `load`=5 is not large enough to reach the snap-through state, therefore we may decide to increase further the load and compute additional steps.

```
Write "@@@ Computing further solutions\n"
Extrapolation NoExtrapolation
Solve Stationary Skip Step 1/2#10
```

Since at the actual value of the simulation parameter `load`=5, the solution has already been computed in the last solution step, we use the `Skip` option to skip computing again this solution. At the pressure load of `load`=7, Newton's algorihtm slows down and needs much more initial iterations to reach the basin of quadratic convergence. At the load of `load`=8, it does not converge anymore and the solution process is stopped. As discussed previously, no solutions exist for this pressure load close to the actual computed solution. Since many mechanical structures, due to their limited elasticity, do not survive a dynamical snap-through process, knowing the maximal value of the pressure is equivalent to determine the ultimate collapse point. We may combine here the impossibility to find solutions with a try&error technique to determine the maximal load. This method is expensive and not optimal from a computational point of view, but it is easy and simple to use. We approach the maximal load by below with an adaptive incremental procedure. As soon as a solution is successfully computed we increment the load, otherwise we restart the computation with a smaller incremental step. The step length will converges toward zero and when small enough we stop the computation.

```
Write "@@@ Computing limit point by try&error\n"
Convergence (AbsIncr.Disp<1.E-12)?1:
            (nIter>5||isnan(AbsIncr.Disp))?failure(""):0
Solve Stationary At 6 Step 0.5, step#100 Failure step<1E-3?0:step/2
```

Within the convergence criterion, we set the failure criterion for Newton's algorithm as a maximal of 5 iterations and at the same time we catch NaN values showing up in the displacement norm. With the `Solve Stationary` statement and the `Failure` option, we set the response directive whenever a failure condition is meet inside Newton's algorithm. If the case, the incremental step of the pressure load is halved and if less than 10^{-3} we stop.

By conveniently changing the simulation parameter and predicting the solution, it may be possible to pass the limit point and reaches the snap-through point. However, it is not easy to follow the path in the unstable region since by reducing the simulation parameter one generally turns back on the solution path. In order to easily obtain the displacement-load curve, one has to let making the changes of the simulation parameter to load control algorithms. Here, the simulation parameter becomes a dependent and thus controlled variable and a new independent variable must be introduced together with an additional equation connecting the solution, the load and the new independent controlling variable. The additional equation determines the type of load control applied and a common choice is spherical or cylindrical load control. The

change of solution and load together with a proper norm definition must be the same as the change of the new controlling variable. Load control is not only used to compute and pass limit points, but also in general as replacement for simple continuation methods. Load control has the property to stabilize the continuation methods and so larger steps can be taken. Load control methods require the derivative of the residual equations with respect to the simulation parameter begin controlled. In *SESES*, this derivative is by default computed by numerical difference, it can be computed exactly but then must be defined explicitely. For this example, the default difference algorithm works nicely, but there may be times where an exact definition is necessary. In order to use load control, a second simulation parameter, playing the role of the new independent variable, must be defined and used similarly and in replacement of the simulation parameter without load control. In our example, the new simulation parameter is `control`. This parameter will control the length of each new step by considering displacement and load together.

```
Write "@@@ Computing with load control\n"
Solve Init
Dump  AtStep 1 Disp
Write AtStep 1 File "plot" Text "%f " load Lattice point "%f\n" -Disp.Y
Convergence { double ok=(AbsIncr.Disp<1.E-12);
              write("  AbsIncr.Disp=%.2e\n",AbsIncr.Disp);
              if (ok) write(" @load %f control %f\n",load,control);
              return ok; }
Increment   Standard ReuseFactoriz #3
            LoadControl Spherical(FacIncr 1E3/sqrt(2); Predic step; load)
Extrapolation Quadratic
Define load=0
Solve Stationary ForSimPar control Step 5#25
```

By running this example, we first start all over again and so we reinitialize the displacement to zero with `Solve Init` and the pressure load with `Solve Define load=0`. We write data for graphical visualization and to display the displacement-load curve of Fig. 1.20. Under load control, the user specifies the controlling variable `control`, whereas the changes of the controlled pressure load `load` are made by the load control algorithm. In order to follow the changes of the pressure load, we print its value at the end of each solution step together with the `Convergence` statement. The load control algorithm is specified with the `Increment LoadControl` statement and the `Spherical` function corresponds to spherical load control. This function requires the specification of the parameter `load` to be controlled, a scaling factor `FacIncr` for weighting the contribution of displacement and load with respect to the step length of `control` and a prediction function `Pred` for the controlled parameter used to speed-up the computation.

1.9 External Post Processing and Controlling

This section illustrates how *SESES* can be interfaced with other software tools to analyze or visualize simulation data or to control numerical computations. First, a general method for exporting the numerical data is presented and then examples for postprocessing the data with mathematical and plotting tools, including *Mathematica* and *Gnuplot* are given. We also discuss how to create an external control system with *Mat-*

Lab that uses feedback from simulation results and launches *SESES* in batch mode in order e.g. to automate the search for an optimized design.

Let us first discuss how to store simulation data in a multicolumn ASCII format for subsequent import in other software tools. We shall illustrate this with the cantilever microactuator example `example/ElThermMech.s2d`. Analogous to the previous example discussed in the context of animation, we run a `Solve Stationary` statement with the sweep specification for the parameter `cur` according to

```
Solve Stationary ForSimPar cur At 0.005 Step 0.005#4
```

Here we compute five solutions starting with a current of $5\,\mathrm{mA}$ and with increments of $5\,\mathrm{mA}$.

For plot post-processing, we want to create a column table with the current values in the first column, the x- and y-displacement of the cantilever tip in the second and third column and in the last column the average temperature of the device. The first table row is reserved for a comment and the vertical size of the table is given by the number of solution steps i.e. the number of different current values. These are the statements used to create our table and they must be defined before the `Solve` statement.

```
Lattice tip node(nx ny/2)

Write  File "curve"
  Text  "#current (A)\tx-displacement (m)\ty-displacement (m) temp (K)\n"

Write AtStep 1 File "curve" Append
  Text "%.3f\t" cur
  Lattice tip "%.3e\t%.3e\t" Disp.X Disp.Y
  Text "%.3f\n" integrate(Temp)/integrate(1)

Solve Stationary ForSimPar cur At 0.005 Step 0.005#4
```

In order to obtain the displacement values at the tip, we have first to define a lattice where numerical fields can be evaluated. We do this with the `Lattice` statement followed by the lattice name and the lattice points, here a single point located at the cantilever tip. To write the first comment line to our data file named `curve`, we use the `Write` statement followed by a format string and the output file specification. Within the string, we use the tabulator `\t` and the newline character `\n`. At each value of the swept current, we append to the output file the value of the current, of the displacements `Disp.X` `Disp.Y` at the tip and the averaged temperature. The average value is computed with the built-in function `integrate`. The format strings `"%.3f\t"` and `"%.3e\t"` following the `Text` and `Lattice` keywords specify floating point (f) and engineering (e) notation, respectively, with a precision of three digits. As column separator, we choose the tabulator `\t`. The generated file `curve` reads

```
#current (A)    x-displacement (m)    y-displacement (m) temp (K)
0.005   3.879e-08       1.316e-06       303.281
0.010   1.552e-07       5.265e-06       313.125
0.015   3.491e-07       1.185e-05       329.531
0.020   6.206e-07       2.106e-05       352.499
0.025   9.697e-07       3.290e-05       382.030
```

and may now be used for plotting with third-party software. Another popular file format for data is the comma-separated format with the suffix `.csv` for the *MS Excel* application. To generate such a format, one just needs to use a comma instead of a tabulator inside the format string.

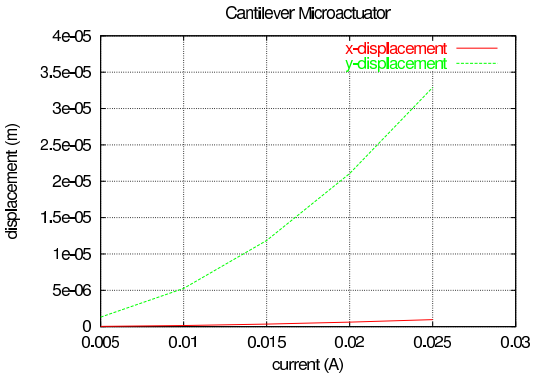

Figure 1.21: Plot of the microactuator tip displacement data generated by *Gnuplot*.

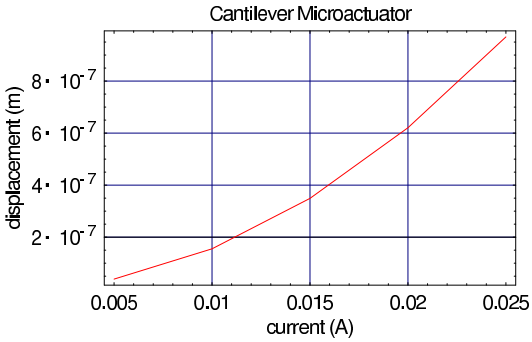

Figure 1.22: Plot of the microactuator tip displacement generated by *Mathematica*.

Plotting with *Gnuplot*

Gnuplot is a freely distributed scientific plotting software available for Linux and Windows. The following commands can be entered interactively in the *Gnuplot* mode or be pasted in the settings file `script.gnu`.

```
set terminal postscript color 20
set output "curveplot.ps"
set title "Cantilever Microactuator"
set xlabel "current (A)"
set ylabel "displacement (m)"
set xzeroaxis
set yrange [0:4e-5]
set grid
plot "curve" using 1:2 title "x-displacement" with lines,\
     "curve" using 1:3 title "y-displacement" with lines
```

A PostScript file named `curveplot.ps` is then created by running the command `gnuplot script.gnu`. Here the second and third column of the data file `curve` is plotted against the first column. The resulting graph is shown in Fig. 1.21.

SESES allows to specify the content of a command line to be executed in line with the numerical computations so that it is possible to launch postprocessing tasks as soon as the numerical data is available. The command line can extend over multiple lines with the help of the backslash escape character. To launch the previous plotting command within *SESES*, one uses the statement

```
System gnuplot script.gnu
```

Plotting with *Mathematica*

Mathematica is a mathematical software by Wolfram Research Inc. useful for linear algebra, analysis, graphical visualization and programming. The content of our data file `curve` can be read with the `ReadList` command and stored in a *Mathematica* variable of the same name and plotted with the following statements.

```
file = OpenRead["curve"];
Skip[file, String];
```

```
curve = ReadList[file, Number, RecordLists -> True];
Close[file];

ListPlot[Map[{#[[1]], #[[2]]} &, curve], Frame -> True, PlotJoined -> True,
  FrameLabel -> {"current (A)", "displacement (m)"},
  PlotLabel -> "Cantilever Microactuator", PlotStyle -> Hue[1],
  GridLines -> Automatic, DefaultFont->"Helvetica"]
```

The `Skip` command skips the first line of the data file that contains comments, i.e. the columns labels. With the `ListPlot` command, the plot shown in Fig. 1.22 is generated.

Controlling with *Matlab*

Matlab is a mathematical software tool by MathWorks Inc. addressed to scientists and engineers for both mathematical calculations and communication to measurement instrumentation. Here, we shall discuss how to create an external control system with *Matlab* that uses simulation results obtained by launching the *SESES* kernel in batch mode. The most straight forward approach is to write to and read from intermediate *SESES* files. Within *Matlab* we can generate a file named `MatLab_2_SESES` to be loaded by *SESES* with an `Include` statement, afterwards we run the *SESES* calculation and let *SESES* write a result file `SESES_2_MatLab` to be read by *Matlab*. The *Matlab* code for our simple interaction example reads as follows.

```
clear all; close all;

% Transferring some data to SESES
par=-0.5;
fid=fopen('MatLab_2_SESES','w');
   fprintf(fid,'Define par = %20.12e \n',par);
fclose(fid);

% Invoke SESES
dos ('.../k2d example.s2d');

% Read the SESES results
fid1=fopen('SESES_2_MatLab','r')
res = fscanf(fid1,'%s')
fclose(fid1)
eval(res);
```

Comment lines in *Matlab* start with `%`. To invoke a command line statement, *Matlab* provides the command `dos`. At the end of the above example, the simulation results are read from the file `SESES_2_MatLab` and evaluated within *Matlab*. With the help of such an approach it is possible, for instance by using optimization code available within *Matlab*, to implement a search for input values producing an optimized result. Such optimization algorithms may be helpful whenever systematic parameter sweeps are too time consuming and possibly not independent of each other. This simple approach has the drawback that a new instance of *SESES* is stared each time. If may not be so and with the help of e.g. named pipes, it is possible to write some simple code implementing a master-slave communication channel to be compiled into dynamical libraries and to be loaded both by *Matlab* and *SESES*.

Chapter 2

Application Examples

In this chapter, we present collected *SESES* examples ranging from single field linear problems to more complex multiphysics non-linear coupled problems. All examples discussed here are found in the `examples` directory of the distribution and can be run without any *SESES* license. This allows the user to visualize all numerical results by its own and to verify or eventually criticize our assertions. All examples are explicitely kept small in size in order to run on commonly used PC hardware.

2.1 Thermal strain

In this example `example/ThermStrain.s2d`, we present a 2D thermal induced mechanical displacement by focusing on setting up the boundary conditions. An isotropic material with induced thermal strain is characterized by the Poisson ratio ν, Young modulus E and the coefficient of thermal expansion α. In this example, a cube of side length $\ell = 1\,\mathrm{m}$ with $\alpha = 17{\times}10^{-6}$, $\nu = 0.3$, $E = 200{\times}10^{9}\,\mathrm{N/m^2}$ will be subjected to a temperature change of $dT = 14\,\mathrm{K}$. By restricting the free expansion on portion of the boundaries, a stress s will be produced in the material. For the unidirectional case, this behavior can be characterized by the following equations

$$e = \frac{\Delta\ell}{\ell} = \alpha\, dT\,, \quad s = E\frac{\Delta\ell}{\ell} = Ee\,.$$

In our example, we constrain the displacement on three side faces of the cube as shown in Fig. 2.1. The material of the left edge is not allowed to move up or down i.e. the y-movement is clamped, the bottom edge is not allowed to move in the x-direction and the right edge cannot move in the y-direction. These boundary conditions have the following consequences. The top left corner is free to move in the x-direction and the top right corner can freely move in the y-direction. The lower left corner is very limited in its movement in both directions. In this corner, we expect the largest total stress. Fig. 2.2-2.3 show the x- and y-displacement, respectively, whereas Fig. 2.4 shows the absolute displacement. An enlargement factor, in our case 200, needs to be set to visualize the small displacements. When displaying the stress field, one observes the largest stress in the lower left corner as predicted. The stress field matches the strain field in the following sense: small strain $\equiv$ large restriction $\equiv$ large stress.

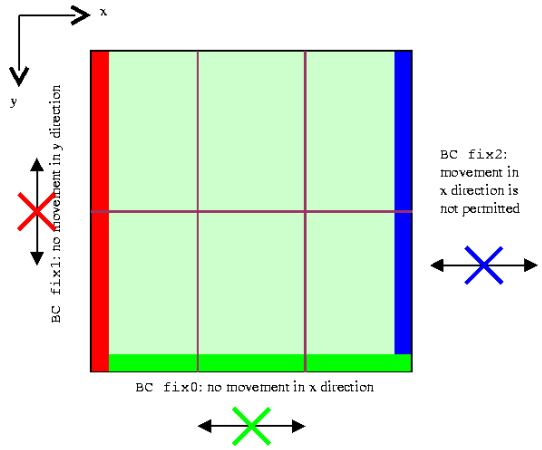

Figure 2.1: Three boundary conditions for thermal expansion.

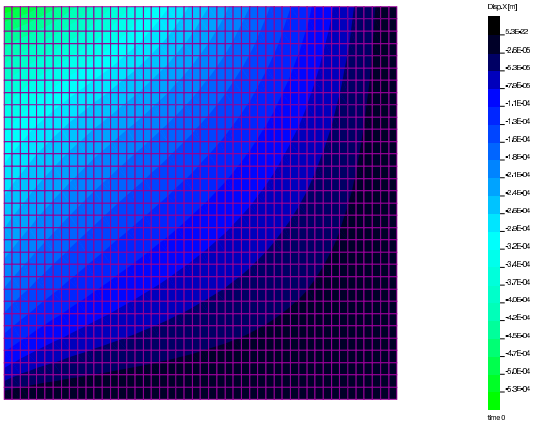

Figure 2.2: The x-displacement. The values are negative everywhere and the top left corner shows the largest value.

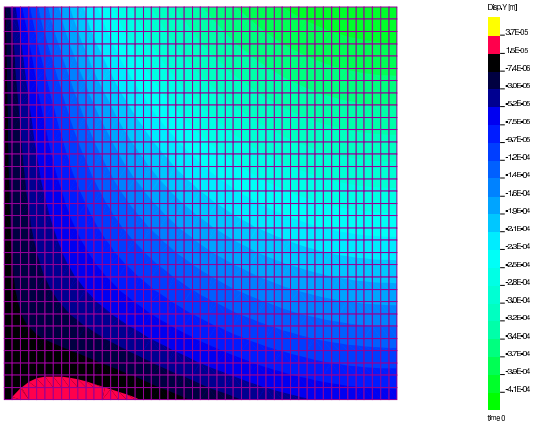

Figure 2.3: The y-displacement. Most values are negative, which corresponds to an upward displacement. The top right corner shows the largest value.

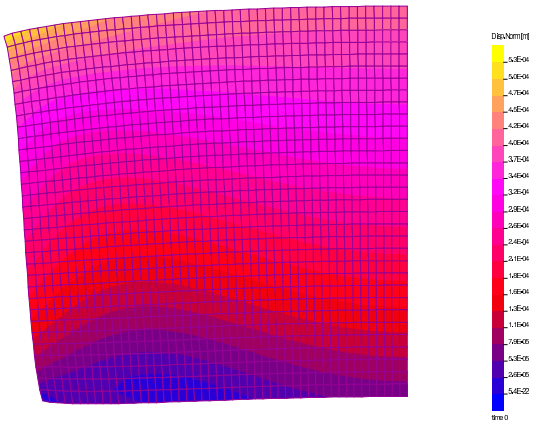

Figure 2.4: The displacement visualized in sito using an enlargement factor of 200.

Alternatively one can select other boundary conditions. As an example one can fix the cube so that it can freely expand and no stress is induced. In conclusion, starting the simulation with very simple, preliminary models helps to get an intuitive understanding of the physics behind the problem considered.

2.2 Designing a Hall Sensor

This example presents the design and optimization of a Hall sensor device. The Hall sensor consists of a rectangular silicon Hall sensor element as displayed in Fig. 2.5. The sensor is manufactured with a 3 micron CMOS process. Two voltage contacts are placed on two opposite sides and an external voltage is applied to induce a current flow. The silicon device is properly doped to act as a resistor. On their way through the device, the charge carriers are deflected in an external magnetic field by

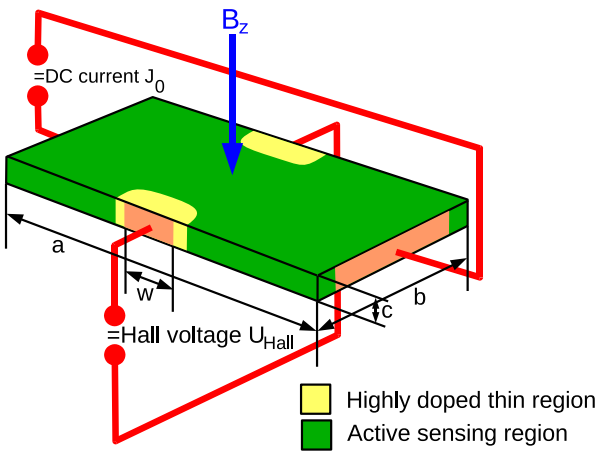

Figure 2.5: Hall sensor layout with J_0: sensor current, B_z: outer magnetic field in z-direction and U_{Hall}: measured Hall voltage.

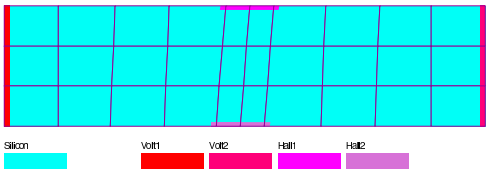

Figure 2.6: Geometry and contacts of the modeled Hall sensor.

the Lorentz force and accumulate on the lateral sides. By placing two contacts on the lateral sides, a potential difference can be measured called Hall voltage. The lateral Hall contacts are usually realized with highly doped thin adjacent regions to achieve optimum Ohmic contact behavior. The modeling of our Hall device starts by posing the problem in mathematical terms. Physical models of varying complexity and degree of sophistication are available. Here we use a simple single carrier model able to correctly characterize the device electrically. The semiconductor drift-diffusion model or even more complex models may be used but do not necessarily improve the results for this Hall device. The single carrier model is based on the solution of the current conservation law for the current density $\mathbf{J}$

$$\nabla \cdot \mathbf{J} = 0 \text{ for } \mathbf{x} \in \Omega \,. \tag{2.1}$$

The material law for charge transport is given by

$$\mathbf{J} = \frac{q_0 \mu_e n_e}{1 + {\mu_e}^2 B^2} (-\nabla\phi - \mu_e \nabla\phi \times \mathbf{B}) \,, \tag{2.2}$$

with q_0 the elementary charge, ϕ the electric potential, $\mathbf{B}$ the outer magnetic field, μ_e the charge carrier mobility and n_e the carrier density. We assume here the carrier density to be equal to the doping concentration.

For our Hall device, the domain boundary $\partial\Omega$ is partitioned according to the electrical contacts

$$\partial\Omega = \partial\Omega_N \cup \partial\Omega_{H_1} \cup \partial\Omega_{H_2} \cup \partial\Omega_{C_1} \cup \partial\Omega_{C_2} \,,$$

with $\partial\Omega_{C_1,C_2}$ the two current contacts, $\partial\Omega_{H_1,H_2}$ the two Hall contacts and $\partial\Omega_N$ the remaining boundary. On $\partial\Omega_N$ no electric current is flowing out of the device, on $\partial\Omega_{C_1,C_2}$ the potential is given and on the Hall contact $\partial\Omega_{H_1,H_2}$ the potential is floating, constant and no current is flowing. This physical situation is represented mathematically by the following BCs

$$\begin{aligned} &\mathbf{J} \cdot \mathbf{n}|_{\partial\Omega_N} = 0 \,, \\ &\phi|_{\partial\Omega_{C_1}} = 0 \,, \\ &\phi|_{\partial\Omega_{C_2}} = V_{\text{applied}} \,, \\ &\textstyle\int_{\partial\Omega_{H_1,H_2}} \mathbf{J} \cdot \mathrm{d}\mathbf{n} = 0 \text{ and } \phi|_{\partial\Omega_{H_1,H_2}} = \text{const} \,. \end{aligned} \tag{2.3}$$

From the theory of boundary value problems, the governing equation (2.1), together with the material law (2.2) and the BCs (2.3) is a well posed problem with a unique solution for the electric potential ϕ.

For an ideal Hall plate of infinite length with a vanishing Hall contact width, the Hall voltage is a function of the current density J_0 and the magnetic field B_z according to

$$U_{\text{Hall}} = \frac{J_0 B_z b}{q_0 n_e} = \frac{\mu_e B_z V_{\text{applied}} b}{a} \,. \tag{2.4}$$

For a real device, we have to consider the finite device length a and contact width w as given in Fig. 2.6. A possible optimization goal would be the requirement that the discrepancy of the measured Hall voltage be less than 10% of the ideal case by choosing the device length a as small as possible. Another important optimization step is the minimization of the offset Hall voltage relevant for the device precision. This is the Hall voltage measured when the external magnetic field is zero $B_z = 0$ and is determined by slightly misaligned Hall contacts introduced by the CMOS fabrication process. Other studies requiring a 3D model may include the influence of the device thickness or a magnetic field $\mathbf{B}$ not orthogonal to the sensor plane.

In the following, we will limit ourselves to a 2D model and obtain the device characteristics as function of the geometric parameters as device length, device width, contact dimension and contact misalignment.

The *SESES* input file for this Hall sensor problem can be found at `example/Hall Sensor.s2d`. The geometry of the device and the definition of the contacts is shown in Fig. 2.6. The device is fully parameterized with the help of user variables.

```
Define
  width    = 5      (* um *)
  length   = 20.0   (* um *)
  cont_w   = 2.0    (* um *)
  misalign = 0.2    (* um *)
  bfield   = 1      (* Tesla *)
  mobil    = 0.1265 (* m**2/(V*s) *)
  vapplied = 1      (* V *)
```

The governing equation (2.1) and the material law (2.2) are selected by defining material `Silicon` as follows

```
MaterialSpec Silicon
  Equation  OhmicCurrent Enable
  Parameter SigmaUns  HallRes(SigmaIso Q0*1.0e24*mobil S/m;
                              BfieldZ  bfield T;
                              MobHall  mobil m**2/(V*s)) SIunit
```

The BCs for the voltage and Hall contacts (2.3) correspond to `Dirichlet` and `Floating` BCs.

```
BC Volt1  0    0  JType ny  Dirichlet Psi 0 V
BC Volt2  nx   0  JType ny  Dirichlet Psi vapplied V
BC Hall1  nx0  0  IType nx1 Floating  Psi 0 A
BC Hall2  nx0  ny IType nx1 Floating  Psi 0 A
```

This single field boundary value problem is linear with respect to the dof-field `Psi` and a solution is computed by selecting an automatic mesh refinement as shown in Fig. 2.7. For a non-vanishing magnetic field or a misalignment, the solution is singular at the corner of the Hall contacts. Here the automatic mesh refinement would

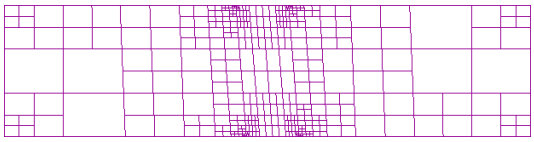

Figure 2.7: Adaptively refined computational mesh showing field singulariries at the contact's corners.

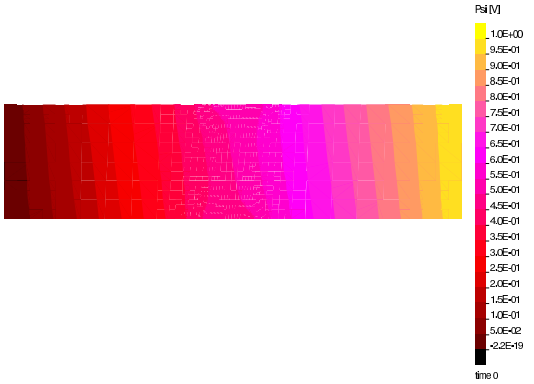

Figure 2.8: Example of a calculated electrostatic potential field in the Hall sensor.

never stop to refine due to the singularities. However, the local singular behavior has little influence on the overall device performances and we can stop refining around the singularities. In practice, refinement of singularities is characterized by a small number of new elements each time we refine and the parameter `MinFractionElmt` is used to stop refinement when the fraction of new and actual elements undergoes the given limit.

```
Store  AtStep 1 CurrentAver(Continuous 1;FreeOnRef)=Current
Remesh AtStep 1 MaxNoElmt 20000 MinFractionElmt 0.05
       Refine ErrPsi(epsilon) Global
Solve Stationary
```

The figure of merit for the Hall sensor is the Hall-voltage which is written to the output together with some model parameters.

```
Write
  "Magnetic field %e T\n"  bfield
  "Misalignment   %e um\n" misalign
  "Hall ideal     %e V\n"  mobil*bfield*vapplied*width/length
  "Hall voltage   %e V\n"  Hall1.Psi.Shift-Hall2.Psi.Shift
```

Here are some results obtained by running *SESES* for different parameter values. The discrepancy of the Hall voltage with the ideal case is of ca. 25% which can be reduced by augmenting the length-width ratio. With a misalignment of just $0.2\,\mu\mathrm{m}$, we have an offset Hall voltage almost as large as for no misalignment and a magnetic field of $1\,\mathrm{T}$.

```
Magnetic field 1.000000e+00 T
Misalignment   0.000000e+00 um
Hall ideal     3.162500e-02 V
Hall voltage   2.455195e-02 V

Magnetic field 0.000000e+00 T
Misalignment   2.000000e-01 um
Hall ideal     0.000000e+00 V
Hall voltage   1.880438e-02 V

Magnetic field 1.000000e+00 T
Misalignment   2.000000e-01 um
Hall ideal     3.162500e-02 V
Hall voltage   4.333158e-02 V
```

The electrostatic potential field `Psi` corresponding to the calculation with the third parameter set is plotted in Fig. 2.8.

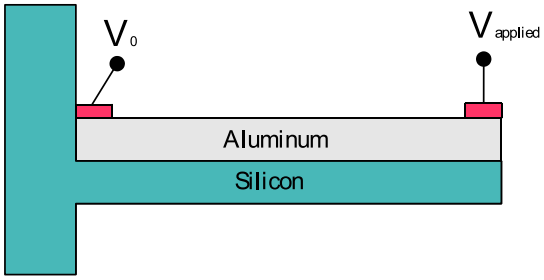

Figure 2.9: A double layer cantilever heated by a current flow.

Figure 2.10: Computed temperature with the hottest spot at the cantilever's tip.

2.3 Electrothermally Driven Cantilever Microactuator

This example presents the modeling of a cantilever heated by a resistor as displayed in Fig. 2.9. By applying a voltage between the contacts V_0 and V_{applied} a current will flow in the aluminum layer and Joule's heat dissipation will heat up the cantilever. Due to mismatch in the thermal expansion coefficients of the aluminum and silicon layer, the thermal induced strain will bend the cantilever. This example is not a proper real example, but serves the purposes of explaining the modeling of a multiple field coupled problem. Is it kept as simple a possible, so that we just perform a 2D simulation and use the bare minimum of details.

The *SESES* input file for this cantilever problem can be found at `example/ElTherm Mech.s2d`. In a first step we define the material properties for bulk silicon. For this material we are going to compute the temperature and the mechanical displacement but since bulk silicon is an insulator no current flow is computed. The isotropic thermal induced strain is defined with the parameter `AlphaIso` of the the built-in `LinElastIso` defining a linear isotropic elastic law. The thermal strain is made a function of the temperature, we assume it vanishes at the environment temperature of $300\,\mathrm{K}$ and is a linear function of the temperature, i.e. we use a first order approximation for the thermal strain.

```
MaterialSpec Silicon
  Equation   ThermalEnergy Elasticity Enable
  Parameter  KappaIso  150.0   W/(m*K)
  Parameter  StressOrtho LinElastIso(Emodule 190 GPa; PoissonR 0.3;
                                     AlphaIso 2.3e-6*(Temp-300)) SIunit
```

The cantilever is a thin structure and care must be taken when computing the mechanical displacement since low order finite elements do not perform well. Two options are available here, either one uses higher order finite elements or special structural elements for beams and shells. We follow the second approach and define a new material `SiliconShell` inheriting from `Silicon` all physical properties but using shell elements of type `ElasticShellJ` to compute the mechanical displacement. Actually since we are solving a 2D problem, the elements will be beam elements.

```
MaterialSpec SiliconShell From Silicon
  Equation ElasticShellJ Enable
```

Next we define the material properties of the aluminum layer. For this conducting material we solve also for the electric field responsible for heating the cantilever. The dissipated Joule's is defined as the source of thermal heat with the statement `Parameter Heat JouleHeat`. The mismatch in the thermal expansion coefficient with respect to silicon of one order of magnitude induces a large inhomogeneous thermal strain causing the cantilever to bend if the temperature is not $300\,\mathrm{K}$. Since the aluminum layer is a thin layer, we use here the same finite elements used for the thin silicon layer. In a real structure, we may not use aluminum as resistor, otherwise large dissipation will be found at the connecting wires. We may still use aluminum for its large expansion coefficient but electrically insulated from a doped silicon channel acting as resistor.

```
MaterialSpec AluShell From Silicon
  Equation   ThermalEnergy OhmicCurrent ElasticShellJ Enable
  Parameter  Heat      JouleHeat() W/m**3
  Parameter  KappaIso  235.0     W/(m*K)
  Parameter  SigmaIso  2.1e5     S/cm
  Parameter  StressOrtho LinElastIso(Emodule 69 GPa; PoissonR 0.333;
                                     AlphaIso 2.3E-5*(Temp-300)) SIunit
```

The device is defined as a stack of two thin layers made of silicon and aluminum attached to a bulk of silicon material.

```
Define
  cur=0.01 (* for one um depth *)
  nx0=4  nx1=4  nx=nx0+nx1
  ny0=4  ny1=4  ny=ny0*2+ny1
QMEI  nx0  50/nx0  nx1  300/nx1
QMEJ  ny0  50/ny0  ny1  10/ny1   ny0  50/ny0
Coord coord*1E-6 1
DeleteME              block(nx0, nx, 0,      ny0)
DeleteME              block(nx0, nx, ny-ny0, ny)
Material AluShell     block(nx0, nx, ny0,    ny0+1)
Material SiliconShell block(nx0, nx, ny0+1,  ny-ny0)
```

The BCs fix mechanically the bulk silicon material and embed it in a thermal bad at $300\,\mathrm{K}$. Two electrical contacts are defined at the extremes of the aluminum layer and the total amount of current is prescribed. This value is to be considered for a default device's depth of $1\,\mathrm{m}$. For this device, the source of heat is the dissipated Joule's heat in the aluminum layer and the sink of heat is represented by the thermal bad attached to the bulk silicon. Black body radiation all along the device's boundary is also a sink of heat and may be defined with a Neumann BC, but due to the limited temperature variation can be neglected.

```
BC Plus    nx  ny0 JType 1 Floating  Psi -cur*1E6 A
BC Minus   nx0 ny0 JType 1 Dirichlet Psi  0       V
BC Support 0 0 JType ny  0 0 IType nx  0 ny IType nx
  Dirichlet Disp 0,0 m
  Dirichlet Temp 300 K
```

Although this example involves a coupled computation with the thermal, electric and mechanical fields, a closer analysis shows that the coupling scenario is unidirectional and the fields can be computed sequentially. If the electric conductivity of aluminum is considered independent from the temperature and the mechanical displacement, the electric potential can be exactly computed in a first step without knowledge of the other fields. Note that if this assumption is reasonable for aluminum, it may not hold for semiconductors where the electric conductivity can be strongly temperature's dependent. Analogously, if the thermal conductivity is independent from the displacement, the temperature can be exactly computed as a second step. Here, we use the

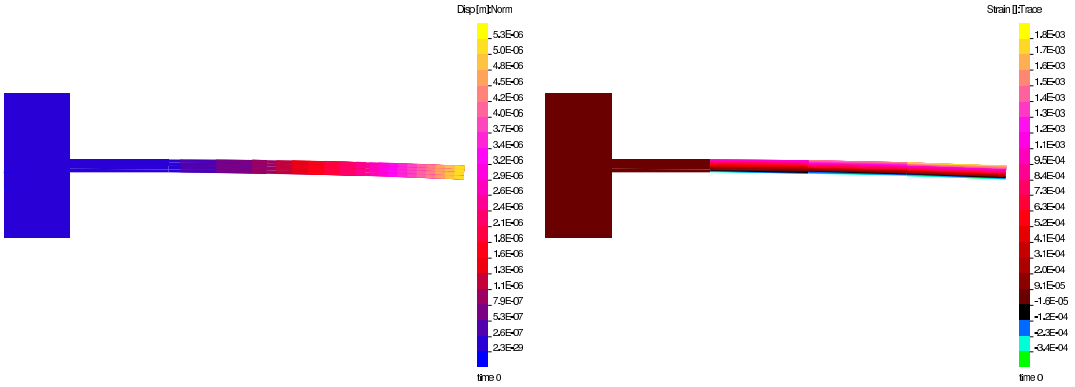

Figure 2.11: Computed mechanical displacement (left) and strain (right) in the cantilever.

electric field computed in the first step to evaluate the Joule's heat. Electric potential and temperature are then used in a third step to compute the mechanical displacement. In this problem, however, we just use the temperature to evaluate the thermal strain, the electric potential is only used in the second step. To compute a solution with the least amount of time, we therefore define three blocks solving for the potential, temperature and displacement fields. The system automatically detects that all three blocks define linear equations and so no block convergence criteria must be defined and a single block iteration will be done. When multiple blocks are defined, the default solution behavior is to solve each block once and then exit which will correctly solve our unidirectional coupling problem.

```
BlockStruct Block Psi Block Temp Block Disp
Solve Stationary
```

When solving for the mechanical displacements, *SESES* will additionally solves for the mechanical rotations within the shell elements defined for the thin cantilever, whereas for the bulky part, we use standard displacement volume elements. At the interface between these two types of elements, additional kinematical constraint equations must be satisfied among the various mechanical dofs, but they are set automatically so that the mechanical rotations are somewhat hidden to the user except for the specification of Dirichlet BCs. For illustration of the results, the temperature field `Temp` can be displayed together with the mechanical displacement as shown in Fig. 2.10. The hottest spot is at the outermost region of the cantilever, while the greatest heat flux is located at the anchor of the cantilever. The displacement and strain fields are shown in Fig. 2.11.

2.4 Heat Conduction in a Cylindrical Stick

Heat issues are of common interest since they occur in many distinct application fields and in general are the result of coupled effects. Thus it is worthwhile to consider a basic heat problem here in order to get familiar with the relevant simulation concepts. Let us consider a cylindrical aluminum stick of total length $125\,\mathrm{cm}$ and a diameter of $4\,\mathrm{cm}$ being cooled at one end and heated by a gas burner at the other end. For an overview of the experimental setup see Fig. 2.12. Essentially, the elongated geometry

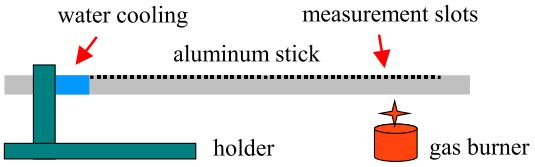

Figure 2.12: Sketch of the horizontal aluminum stick heated on the right and water cooled on the left side.

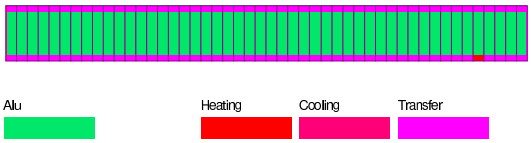

Figure 2.13: 2D model for the heated stick with cooling on the left and heating on the right.

of the stick suggests a simplification to a 1D problem and so we will first present an analytical solution to a simplified 1D problem. As a next step, heat radiation loss at the stick surface is considered and an analytical solution method is proposed. Then a model for a 2D case and lastly a 3D case are calculated in order to refine the simulation results and make them comparable to experimental data of the stick in transient and steady-state.

Analytical model in 1D

In one dimension, the heat equation reads

$$c_p\,\rho\frac{\mathrm{d}T}{\mathrm{d}t} - \kappa\frac{\mathrm{d}^2T}{\mathrm{d}x^2} = q\,,$$

with T the temperature, κ the heat conductivity, q the heat density rate, c_p the specific heat capacity and ρ the specific density. As a first approach let us consider the stationary equation with a heat loss proportional to the local temperature

$$-\kappa\frac{\mathrm{d}^2T}{\mathrm{d}x^2} = -\alpha(T - T_{\mathrm{ref}})\,, \tag{2.5}$$

with α a phenomenological heat loss coefficient. As boundary conditions, we choose

$$T(0) = T_{\mathrm{ref}} \quad \text{and} \quad F(L) = h\,, \tag{2.6}$$

with T_{ref} the temperature of the cooling water at $x = 0$, $\mathbf{F} = -\kappa\nabla T$ the heat flux and h a measure of the heating power at $x = L$. The general solution is a particular solution to the inhomogeneous equation that we simply choose as $T_I = T_{\mathrm{ref}}$ plus the homogeneous solution which can be found with the help of the Ansatz $T_H(x) = Ce^{\lambda x}$. Back substitution yields the characteristic polynomial and the general form

$$\kappa\lambda^2 - \alpha = 0 \;\rightarrow\; \lambda_\pm = \pm\sqrt{\frac{\alpha}{\kappa}} \;\rightarrow\; T_H(x) = C_1e^{\lambda_+ x} + C_2e^{\lambda_- x}\,, \tag{2.7}$$

with the two coefficients C_1 and C_2 determined by the boundary conditions (2.6). At $x = 0$, we obtain $C_2 = -C_1 = -C$ and since $\mathrm{d}T/\mathrm{d}x = h/\kappa$ at $x = L$, we find $C = h/(\kappa\lambda 2\cosh(\lambda L))$ with $\lambda = \sqrt{\alpha/\kappa}$. The solution to (2.5)-(2.6) is therefore

$$T(x) = T_I(x) + T_H(x) = T_{\mathrm{ref}} + \frac{h}{2\kappa\lambda\cosh(\lambda L)}\sinh(\lambda x)\,. \tag{2.8}$$

The larger the heat transfer coefficient α the stronger the temperature profile deviates from the linear solution obtained for $\alpha \to 0$. In other words, α determines the curvature of the temperature profile.

Building a 2D model for the heated stick

We shall start with a 2D model of the heated stick which will be used to calculate the time-dependent temperature at selected points as well as complete temperature profiles. The input file can be found at `example/HeatStick.s2d`. These calculations will be compared to measurement data acquired at the slots indicated in Fig. 2.12. The mesh of the 2D model is shown in Fig. 2.13. In order to get accurate simulation results, one needs to consider heat loss by radiation and convective transport apart from the cooling and heating boundary conditions mentioned above in (2.6). In heat problems, it is common to introduce a phenomenological heat transfer coefficient α as the proportionality factor between the heat change and the temperature difference with respect to a reference point of temperature T_{ref}

$$\frac{\mathrm{d}Q}{\mathrm{d}t} = \alpha A(T - T_{\mathrm{ref}})\,, \tag{2.9}$$

with A being the surface area. The transfer coefficients for convection can be derived tabulated figures of merit that depend on the specific geometry and surrounding material. In the section discussing a 3D model, we will also introduce heat radiation and consider a realistic value for the heat transfer coefficient. In our model heat transfer is defined with the following statements

```
BC Transfer  OnChange 1 OnBC Heating Disable OnBC Cooling Disable
   Neumann Temp D_Temp tcoeff*(-refTmp+Temp),tcoeff  W/m**2-W/(m**2*K)
```

with the reference temperature `refTmp`. We note that since the Neumann BC depends on the temperature itself, one needs to supply the derivative of the temperature `dof` field with the `D_Temp` statement in order to enable a successful Newton's algorithm. The cooling conditions enforced with water flow through the left end of the heated stick is implemented with a simple Dirichlet boundary condition

```
BC Cooling  0 0 JType 1
   Dirichlet Temp refTmp K
```

The heat source located at the right end of the stick is defined as follows

```
BC Heating nx-5 1 IType 1
   Neumann Temp -(time<=toff?heaterflux:0) W/(cm**2)  (* time-dep. heating *)
```

where a `(test?true:false)` statement was constructed to implement switching off the heat source after the time `toff`. The duration of the time-dependent simulation is set with the parameter `period` and the turn-off time `toff` is set to half this time interval. The time-dependent simulation is defined with the statement

```
Dump AtStep 1 Temp(Title ("TIME=%g\n",time)) TempDiff(Title ("TIME=%g\n",time))
Solve Unstationary At 0 Step period/npts Until period
```

The first line above makes sure that the temperature and heat flux fields are stored in the `Data` file at each time step. For a heating parameter of $h = 160\,\mathrm{W/cm^2}$ some time-dependent temperature fields during heating are shown in Fig. 2.14. The lowermost distribution corresponds to the steady state. In Fig. 2.15 some temperature fields during cooling $t > t_{\mathrm{off}}$ are shown. Note that the color scale is normalized for each instance in time and therefore cannot be compared directly.

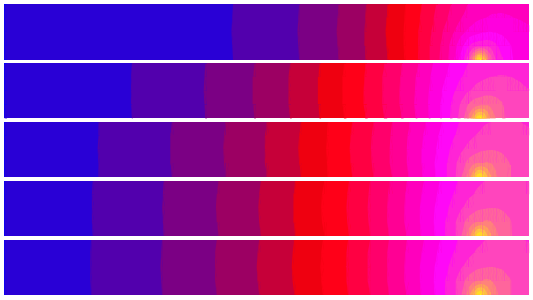

Figure 2.14: Temperature fields at different times after turn-on during heating (temperature scale not shown).

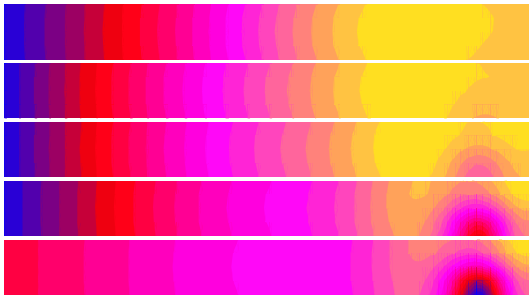

Figure 2.15: Temperature fields at different times after turn-off during cooling (temperature scale not shown).

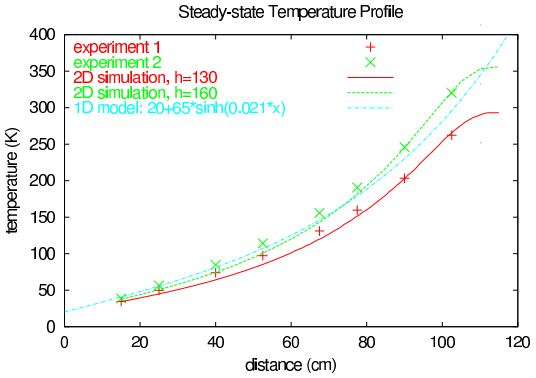

Figure 2.16: Steady-state temperature profile in the stick for two distinct heating powers and simulations with adjusted heating parameter $h\,(\mathrm{W/m^2})$. The prediction by the 1D analytical model is shown for comparison.

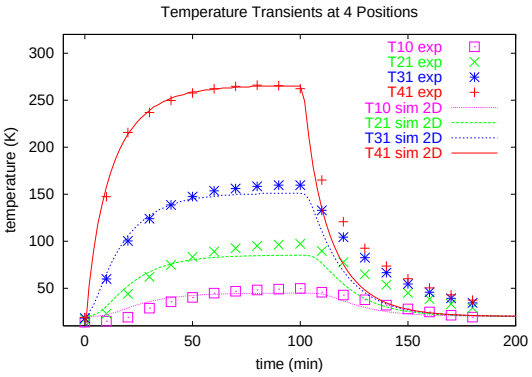

Figure 2.17: Time-dependent temperature at different positions on the aluminum stick.

Let us now compare the simulation results with measured data sets of temperature profiles and transients. For this purpose four points along the aluminum stick were defined with `Lattice` statements whose temperature is written to a file during the calculation. Moreover a line was defined along the stick along which the temperature profile was recorded. In Fig. 2.16, two measurements with distinct heating intensities are shown that can be reproduced successfully by our 2D simulation by adjusting the heating parameter h, denoted by `heaterflux`. For the simulation shown in this figure, the heat transfer coefficient `tcoeff` was set to $100\,\mathrm{W/(m^2\,K)}$. This value seems quite large but includes a correction factor due to the reduced surface area in the 2D model. The reduced surface area results from the 2D model dimensions that were determined by the requirement of identical volume as in the real geometry, thus leading to a surface area smaller by a factor of 11 with respect to the real cylinder surface. The exact value of `tcoeff` will be discussed in more detail in the next section on a 3D model of the heated stick. In addition, the analytical solution of (2.8) is also plotted. The factor of sinh representing the heating intensity was adjusted. By contrast, the prefactor in the argument, $\lambda = \sqrt{\alpha/\kappa}$ where α corresponds to our parameter `tcoeff`, was calculated as $0.021\,\mathrm{cm^{-1}}$ using $\alpha = 10\,\mathrm{W/(m^3K)}$ and $\kappa = 235\,\mathrm{W/(m\,K)}$.

Keeping the simulation parameters fixed, we can now consider the time-dependent

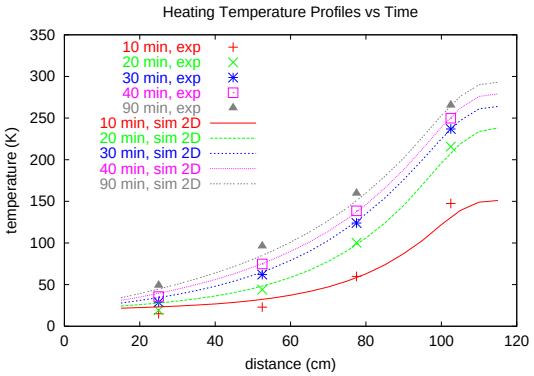

Figure 2.18: Time-dependent temperature profiles during heating.

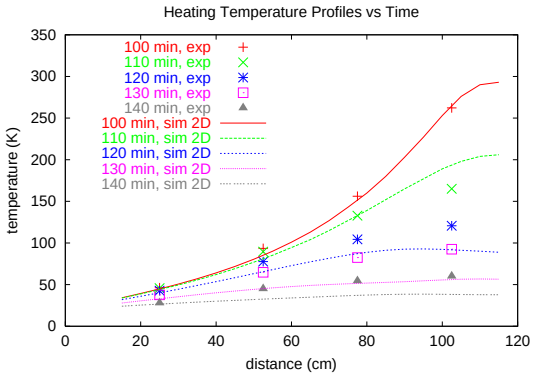

Figure 2.19: Time-dependent temperature profiles during cooling.

temperature at 4 selected points on the stick, denoted by T10, T21, T31, T41. The transient behavior is shown in Fig. 2.17 during a 200 minute time window. The heat source was switched off after 100 minutes. Fig. 2.17 exhibits good agreement between measurement and simulation during the heating cycle. However, after turn-off the temperature drops significantly faster in the simulation. In addition we can compare the time-dependent temperature profiles during heating as well as cooling. The former is shown in Fig. 2.18. The data plotted in constant time increments demonstrates that the heat conduction is fast initially but slows down when approaching steady state.

Building a 3D Model for the heated stick

In the previous 2D approach, we have achieved satisfactory agreement with measurement data by assuming convective heat transfer only and by adjusting the two parameters heating intensity `heaterflux` and the transfer coefficient `tcoeff`. The former adjusts the resulting temperature scale and the latter the curvature of the temperature profile. With the optimum value `tcoeff=100`, the data is reproduced nicely. However, a closer look into the heat transfer literature reveals that typical values for α are around 5 to $20\,\mathrm{W/(m^2K)}$. In particular, the convective heat transfer coefficient is determined by the Nusselt number Nu through $\alpha = \mathrm{Nu}\,\kappa/d$ where d is the characteristic length scale, in our case the cylinder diameter. For horizontal cylinders, the Nusselt number is given by

$$\mathrm{Nu} = \left[0.6 + \frac{0.387\,\mathrm{Ra}^{0.167}}{(1 + (\frac{0.559}{\mathrm{Pr}})^{0.563})^{0.296}}\right]^2 , \tag{2.10}$$

where both the Prandtl number Pr and the Rayleigh number Ra depend on material parameters of the surrounding air. For our geometry the convective heat transfer coefficient is calculated as $10.1\,\mathrm{W/(m^2K)}$. We note that this value is in good agreement with the value of $100\,\mathrm{W/(m^2K)}$ used in the 2D model if the surface reduction factor is considered. We shall now keep this value fixed and in addition consider heat transfer by radiation. The Stephan-Boltzmann radiation law states

$$\frac{\mathrm{d}Q}{\mathrm{d}t} = \epsilon\sigma A(T^4 - T_{\mathrm{ref}}^4)\,, \tag{2.11}$$

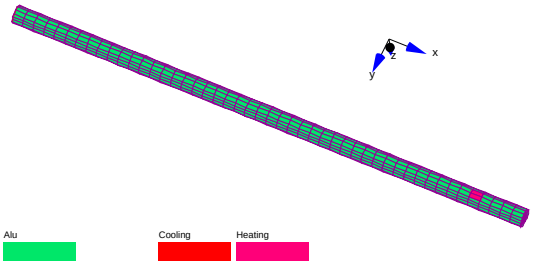

Figure 2.20: 3D *SESES* Model of the heated stick shown from below with cooling on the right and heating on the left.

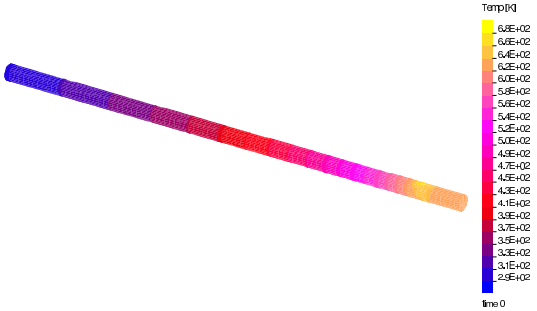

Figure 2.21: Example of calculated temperature distribution in 3D assuming both convective and radiative heat transfer.

where we have introduced the Stephan-Boltzmann constant σ and the emissivity ϵ. The emissivity is a unitless material parameter between 0 and 1 and for a given material varies depending on the surface quality such as the roughness. For aluminum a typical value above room temperature is 0.25. The new heat transfer boundary condition considering both convective and radiative loss thus reads

```
BC Transfer OnChange 1 Neumann Temp
  D_Temp tcoeff*(-refTmp+Temp)+sigma*eps*(-(refTmp**4)+Temp**4),
                 tcoeff+sigma*eps*4*Temp**3 W/m**2-W/(m**2*K)
```

As for the geometry, a 3D model is derived with only minor corrections from the 2D one. For our cylindrical stick, we keep the x-axis parallel to the stick and choose the z-axis perpendicular to x and y. By use of the cylinder homotopy routine we can thus define the 3D stick geometry and the resulting geometry is illustrated in Fig. 2.20.

```
QMEI nx xlen/nx
QMEJ ny ylen/ny
QMEK nz=ny (zlen=ylen)/ny

Coord  cylx(coord,ylen/2,zlen/2,1)*1E-2 1
```

With the heat flux now chosen as $60\,\mathrm{W/m^2}$, we calculate the stationary distribution shown in Fig. 2.21. The heated spot on the stick is pointing upward in this figure. With the refined 3D model, we expect the data to be reproduced more accurately which is confirmed in the comparison of Fig. 2.22. The curvature of the temperature profile is now matched more closely to the measured data. Note that the temperature drops at distances exceeding $102.5\,\mathrm{cm}$ since the stick is heated at this location. As for the transient behavior, the 3D model gives comparable results as the 2D model, see Fig. 2.23. With the 3D model the temperature decay is matched somewhat closer.

The last simulation aspect to be discussed here is the thermal expansion resulting from the heat distribution calculated above. For this purpose the thermal expansion coefficient `AlphaIso` and the displacement field `Disp` needs to be defined and enabled, respectively. The following statement inserted in the command section defining the calculation sequence is also required.

```
BlockStruct Block Temp Block Disp
```

To fix the cooled left end of the aluminum stick, we set the x-component of the displacement field to zero in the boundary condition

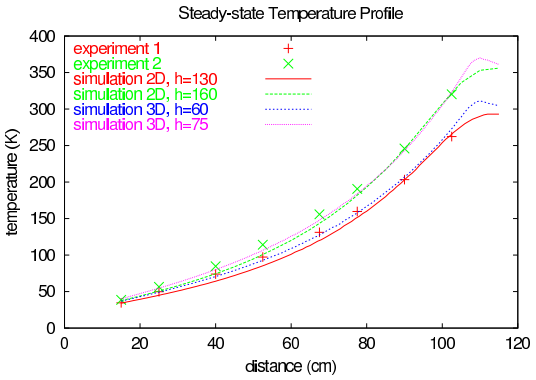

Figure 2.22: Comparison of the 3D simulation results with the previously shown temperature profile data.

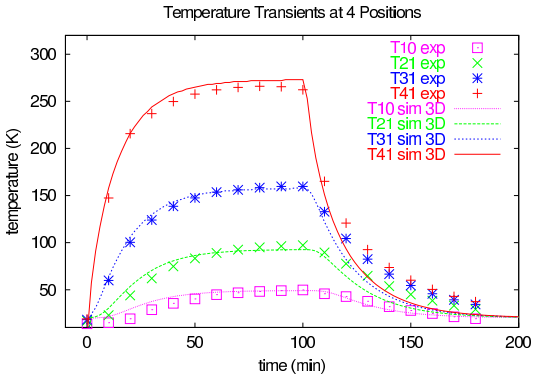

Figure 2.23: Comparison of the 3D simulation results with the previously shown transient data.

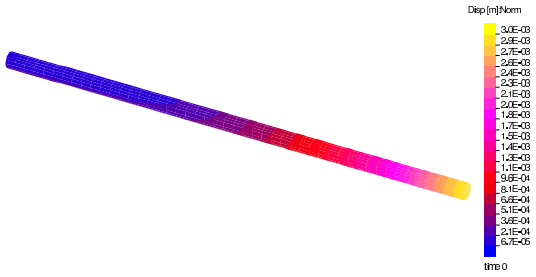

Figure 2.24: Calculation of the displacement due to thermal expansion in the aluminum stick.

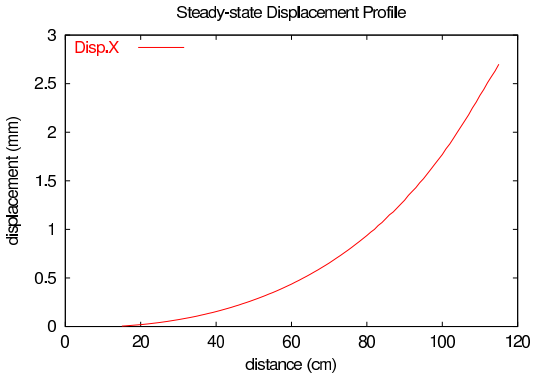

Figure 2.25: Calculation of the displacement profile in x-direction due to thermal expansion.

```
BC Cooling  0 0 0 JKType ny nz
   Dirichlet Temp   refTmp K
   Dirichlet Disp.X 0 m
```

The resulting total elongation of the stick is the value of the displacement field x-component at the right end of the stick. For our example with $h = 60\,\mathrm{W/(m^2\,K)}$ and an expansion coefficient of `AlphaIso` $= 2.310^{-5}\,\mathrm{K}^{-1}$, we get an elongation of $3\,\mathrm{mm}$. The displacement profile is shown in Fig. 2.25 and is essentially the integral of the temperature profile times the expansion coefficient. From the solution of the simple 1D analytical model (2.8), one would thus predict a hyperbolic cosine function for this x-displacement profile.

In conclusion we have demonstrated the use of *SESES* to model the temperature distribution and thermal expansion in a heated aluminum stick by considering both convective and radiative heat loss. The simulation results were compared successfully with experimental data in transient and steady state.

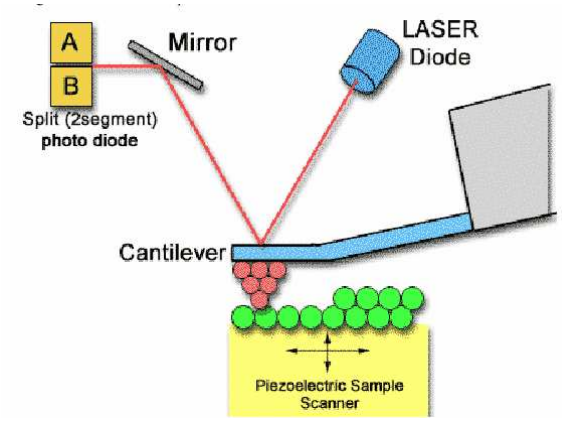

Figure 2.26: Schematic overview of a scanning probe microscope (SPM) showing the detection principle for measuring the tip sample interaction.

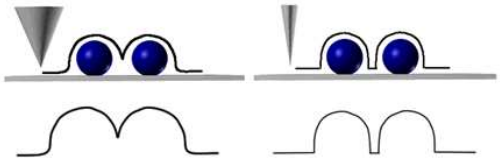

Figure 2.27: Illustration of image acquisition artefact in scanning probe microscopy.

2.5 Image Acquisition in Scanning Probe Microscopy

In scanning probe microscopy (SPM) a tiny and sharp tip scans across a surface while the tip-sample interaction is monitored. The signal of this interaction allows the sequential construction of an image of the sample surface to be investigated. Two distinct operating principles for scanning are used: constant-height and constant-distance. In the former principle, the tip height is kept constant and the varying signal is recorded. In contrast, for the constant-distance mode an electronic feedback loop adjusts the z-position to ensure constant tip-sample distance and the z-position is recorded. The most common measurement principles of the tip-sample interaction are the atomic force employed in atomic force microscopy (AFM) and the electronic tunneling current used in scanning tunneling microscopy (STM).

In this tutorial example, the operation of SPMs shall be simulated in order to illustrate the different operating principles and the limitations for the lateral resolution. In particular, a virtual experiment with a sample surface shall be carried out in order to study the acquired surface image with regards to the tip geometry. In mathematical terms, the acquired image is the convolution of the tip geometry and the surface topography. With simplified models we shall calculate the electrostatic field between a metallic tip and a conducting sample surface and interprete this quantity as a measure of the resulting tunneling current. In a real STM experiment the geometry of the tip is not that critical since the tunneling current depends exponentially on the tip-sample distance. Thus, the outermost atoms of the metallic tip maintain the tunneling current and are thus responsible for the lateral resolution. A schematic overview of the scanning probe setup is shown in Fig. 2.26. In most commercial SPM setups, the tip holder is kept fixed while a piezo moves the sample relative to the tip in order to scan the sample surface. A method for simulating the surface scan and the dependence of the tip geometry will be discussed next.

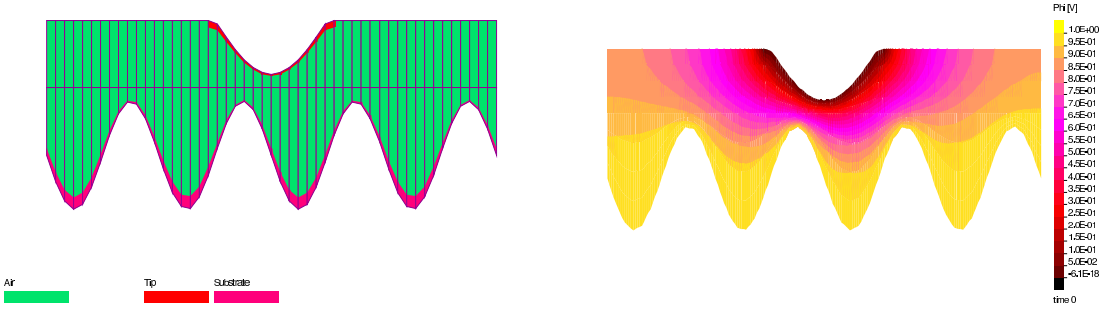

Figure 2.28: Left: 2D simulation domain of sinusoidal tip and surface. Right: Calculated electrostatic potential distribution between tip and sample.

Numerical model of a tip-sample geometry

In the introductory example of the first chapter Chapter *1 Getting Started* a simplified SPM geometry with a vertical line segment as tip and a flat sample surface, both defined as boundary condition of an electrostatics problem, was presented. As a first refinement to that two-dimensional model, the example `example/SpmSine.s2d` defines the tip geometry as well as the sample surface with a sinusoidal function. Since both the tip and the surface contour are assumed perfectly conductive, it suffices to implement the medium in between them as the modeling domain. In particular, we may start with a single rectangular macro element and just change the upper (tip) and lower (substrate) domain boundary by applying a sine function to distort the regular geometry. The result is shown in Fig. 2.28 together with an example of a computed electrostatic potential.

Let us now consider a unidirectional scan across the surface during which we shall record the electric field at the tip end. We can perform such a scan by moving the substrate in x-direction relative to the tip. A simulation parameter `xoff` was introduced for this purpose. The command section of the input file thus reads as follows.

```
Dump AtStep 1 Phi Efield

Write File "tip_1Dscan_s"@@{tshape}"n"@@{nsteps}".txt"
  Text "speriod=%.2f sheight=%.2f tshape=%.2f theight=%.2f\n"
    speriod sheight tshape theight
  "xoff (nm) Efield (V/m) Phi (V)\n"

Write AtStep 1 File "tip_1Dscan_s"@@{tshape}"n"@@{nsteps}".txt" Append
  Lattice tip "%.3e %.3e %.3e\n" xoff*lx Efield Phi

Solve Stationary ForSimPar xoff At 0 Step 0.01#nsteps
```

The file name contains the tip shape parameter `tshape` and the number of incremental scan steps `nsteps`. The 3-column data file obtained can now be plotted with *Gnuplot*.

In scanning probe experiments the acquired images are subject to artefacts caused by the non-ideal tip geometry. In general the acquired signal is a convolution of the tip geometry and the substrate topography. This effect is schematically shown in Fig. 2.27 in a constant-distance acquisition mode. Depending on the sharpness of the tip, the acquired image reveals more or less structure of the surface topography. The simplest simulation scan is performed in constant-height mode. The shape of the tip can be

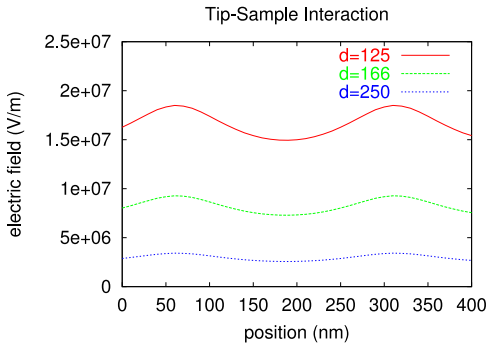

Figure 2.29: Simulation of the image acquisition artefact caused by non-ideal tip shape with `d` the width of the tip.

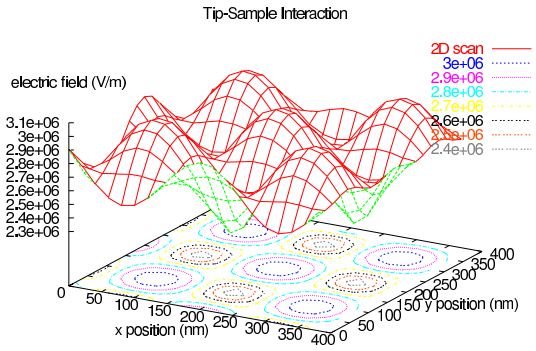

Figure 2.30: Plot of the calculated tip-sample interaction using exported 2D simulation data. Contour lines are drawn in the xy-projection plane.

varied by the `tshape` parameter. For a small `tshape` parameter the tip is sharp and the resulting field correspondingly high. Thus, if the electric field is taken as a measure for the substrate topography, one gets a stronger apparent topography modulation. This effect is shown in Fig. 2.29 for three different `tshape` values.

As a next step, we discuss the extension `example/SpmSine.s3d` of the above model to 3D in order to perform a 2D surface scan in constant-height acquisition mode. The 2D geometry has been extended to 3D in a simple way. The shape of the tip is chosen rotation symmetric around the z-axis, its radial shape is determined by a cosine function and its lateral xy-position has been parameterized with two user parameters `xoff` and `yoff`. The solution procedure is as before, but since with the `ForSimPar` parameter sweep option of the `Solve Stationary` statement only one parameter can be swept, we have used two preprocessor nested loops and the statement `Define` to update the simulation variables `xoff` and `yoff` as follows

```
For i From 0 To nsteps
{
   For j From 0 To nsteps
   {
     Define xoff=@i*sweep*lx/nsteps, yoff=@j*sweep*ly/nsteps
     Solve Stationary

     Write File "tip_2Dscan.txt" Append
        Lattice tip "%.3e %.3e %.3e\n" xoff yoff Efield
  }
  (* separates data blocks for gnuplot *)
  Write File "tip_2Dscan.txt" Append "\n"
}
```

At the end of the inner loop we add a line feed to the data file in order to separate row data. The exported simulation data is easily visualized using the *Gnuplot* software and the necessary files and commands are also included in the example. The resulting graph for a 2D scan within a $400 \times 400\,\mathrm{nm}^2$ region is shown in Fig. 2.30.

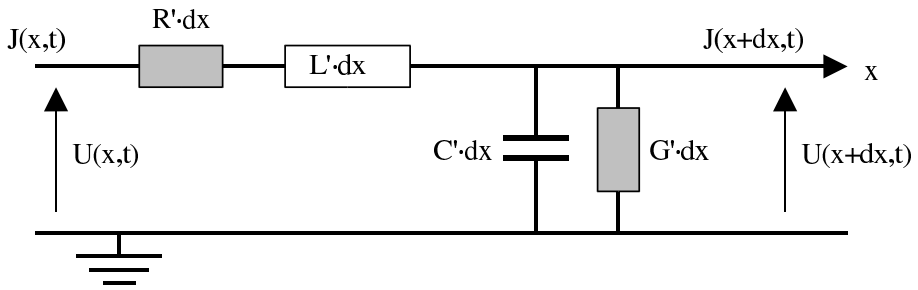

Figure 2.31: Circuit diagram of the voltage $U(x,t)$ and current $J(x,t)$ along a wire.

2.6 Cross Talk and Telegraphy

Electric signals running along parallel wires cross talk to each other, i.e. part of the signal in one wire is transferred to the other wire. The phenomenon might be conductive, capacitive or inductive in nature and is treated in text books under the name of *equation of telegraphy*. Fig. 2.31 shows the situation and the analytical description is summarized by the following equations

$$\begin{aligned} U(x,t) - U(x+\mathrm{d}x,t) &= RJ(x,t)\mathrm{d}x + L\tfrac{\partial J(x,t)}{\partial t}\mathrm{d}x \to -\tfrac{\partial U}{\partial x} = RJ + L\tfrac{\partial J}{\partial t}\,, \\ J(x,t) - J(x+\mathrm{d}x,t) &= GU(x,t)\mathrm{d}x + C\tfrac{\partial U(x,t)}{\partial t}\mathrm{d}x \to -\tfrac{\partial J}{\partial x} = GU + C\tfrac{\partial U}{\partial t}\,, \end{aligned} \tag{2.12}$$

with U, J the voltage and current along the wire and R, L, G, C the resistance, inductance, conductance and capacity pro unit of length of the wire. Combining (2.12) together, we arrive at the following equation of telegraphy for the voltage U

$$\frac{\partial^2 U}{\partial x^2} = (RG)U + (RC+LG)\frac{\partial U}{\partial t} + (LC)\frac{\partial^2 U}{\partial t^2}.$$

The solutions are damped, harmonic waves propagating along the cable with the damping constant $\gamma^2 = (R+i\omega L)(G+i\omega C)$ and impedance

$$Z(\omega) = U(\omega)/J(\omega) = \sqrt{(R+i\omega L)/(G+i\omega C)}\,. \tag{2.13}$$

In this tutorial example `example/CrossTalk.s2d`, we are going to compute the capacity C and the inductance L for two ideal cylindrical parallel wires and compare the values with analytical results. In doing this, we also briefly present the theory of transversal modes in transmission lines. The conductive cross talk is not considered, since we assume perfect electrical insulation between the wires, i.e. $G = 0$. The ideal case is characterized by ideal conductors with $R = 0$, where the current is flowing over the surface of the wires. Therefore the current flow in the wires does not need to be considered and the damping caused by the physical non-zero value of R does not enter the computation of L and C. For this ideal case, we are therefore left to solve the Maxwell's eqs. in the free space surrounding the wires where all currents and charges are zero. Non-zero currents and charges will just show up just at the surface of the wires. In signal theory and for linear systems, it is customary to perform a harmonic analysis, where the participating fields have complex values and the time dependency is taken to be of the form $(\bullet)(t) = (\bullet)\exp(i\omega t)$. For this setup, the Maxwell's Eqs. read

$$\nabla.\mathbf{B} = 0\,,\quad \nabla.\mathbf{E} = 0\,,\quad \nabla\times\mathbf{E} + i\omega\mathbf{B} = 0\,,\quad \nabla\times\mathbf{B} - i\omega\mu_0\epsilon_0\mathbf{E} = 0\,. \tag{2.14}$$

The particularity of transmission lines with two or more wires compared to waveguides with one single wire, is that for the former a signal can be transmitted at any frequency, whereas for the latter there is a cutoff frequency below the one no signal can pass through. The modes with zero cutoff frequency are transversal modes, so called TEM modes, where the longitudinal components of $\mathbf{E}$ and $\mathbf{B}$ are zero i.e. $E_z = B_z = 0$ and just for these modes we seek here solutions of (2.14). Because of the z-invariance, we may as well consider the z-dependency to be of the form $(.)(z) = (.)\exp(ikz)$ and construct general z-dependent solutions by superposition of these spatial modes. With these assumptions and the property $\mu_0\epsilon_0 = 1/c^2$, the Maxwell's Eqs. (2.14) yield the dispersion law $k = w/c$ and result in the equations [1, 2]

$$\frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} = 0\,,\ \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} = 0\,,\ cB_y = -E_x\,,\ cB_x = E_y\,, \tag{2.15}$$

i.e. the magnetic field $\mathbf{B}$ is given by the rotating clockwise the electric field $\mathbf{E}$ by $90°$, the electric field $\mathbf{E} = -\nabla\Phi$ is given by solving the 2D Poisson equation $\nabla.\nabla\Phi = 0$ and the signals are transmitted with light velocity c.

We now want to compute the inductance L and capacity C for our transmission from the solution of (2.15) in order to obtain the impedance (2.13). From the definition of C and L we have $Q = CV$ and $\partial V/\partial z = L\partial J/\partial t$ resulting in $V = cLJ$ with Q the total charge, J the total current and V the potential on the surface of one wire when the second one is grounded. The total charge Q and current J are given by the surface integrals

$$Q = \epsilon_o \int_{\partial\Omega_{\text{wire}}} \mathbf{E}.\mathbf{n}\,\mathrm{d}s\,,\quad J = \mu_o^{-1} \int_{\partial\Omega_{\text{wire}}} \mathbf{B}.\mathbf{t}\,\mathrm{d}s\,, \tag{2.16}$$

with $\mathbf{n}$ the outward normal and $\mathbf{t}$ the unit tangent to the boundary $\partial\Omega_{\text{wire}}$. But since the magnetic field is a rotation of the electric field times c, we actually have $J = cQ$. The product of inductance and capacity is therefore a constant $LC = 1/c^2$ independent from the wire's shape and the impedance is given by

$$Z(\omega) = \sqrt{\frac{RC + i\omega/c^2}{i\omega C^2}}\,.$$

Following [3], for two parallel cylindrical wires, the exact value of the capacity is given by

$$C = \frac{\pi\ell\epsilon_o}{\log\frac{d}{r}}\,, \tag{2.17}$$

with $\epsilon_o = 8.85{\times}10^{-12}\,\mathrm{C}^2/(\mathrm{Nm}^2)$, ℓ the length of wire, d the distance between the wires and r the wire's radius.

Results and Conclusion

In the initial section of the input file, we define the two wires of radius $r = 0.1\,\mathrm{m}$ separated by a distance of $d = 2\,\mathrm{m}$ together with the surrounding air. To solve for the electro static problem (2.15) , we enable the equation `ElectroStatic` and set the value of the electric potential to $0\,\mathrm{V}$ and $1\,\mathrm{V}$ on both wires. The value of the potential

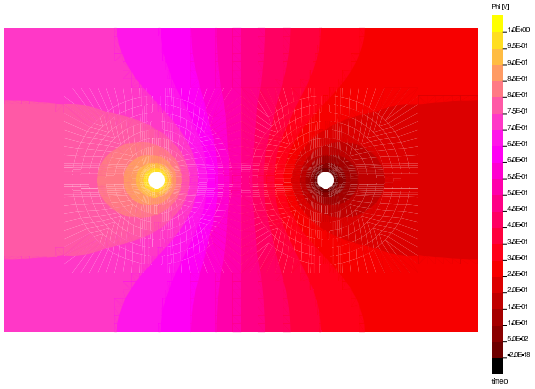

Figure 2.32: Electro scalar potential `Phi` between two charged cables.

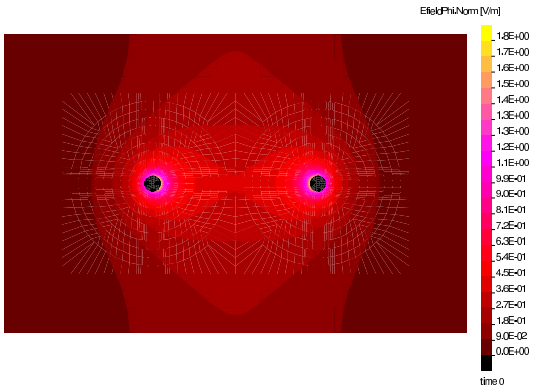

Figure 2.33: Electric field absolute value $|\mathbf{E}|$.

on the exterior boundary should be set to represent an infinite domain and a homogeneous Neumann BC is the preferred choice. Fig. 2.32-Fig. 2.33 show the computed potential and electric field distribution. The total charge (2.16) on the wire is computed with the statement

```
Write "Charge %e (%e)\n"
  Wire1.Phi.Flux,-integrate(Bound Wire1; Dfield.X*Normal.X+Dfield.Y*Normal.Y)
```

whereas the first value being computed internally over residual assembling is generally more precise than the second one computed by direct integration. From the charge, we obtain a capacity of $C = 8.08\,\mathrm{pF}$ compared to the analytical value (2.17) of $C = 9.3\,\mathrm{pF}$.

It is possible to compute the inductance and capacitance of more complex structures following the procedures discussed above and so the impedance Z according to (2.13). Using the `EddyHarmonic` model, it is also possible to include the effects of frequency dependency of material laws and thus to model non-ideal transmission lines. Once the dispersion relation is know for the full spectrum, we may as well perform a spectral analysis on e.g. rectangular shaped signals and follow their dispersive evolution when traveling along the line.

References

[1] D. J. GRIFFITHS, *Introduction to electrodynamics*, Prentice-Hall Inc., 1989.

[2] J. D. JACKSON, *Classical Electrodynamics*, John Wiley & Sons, 1999.

[3] HORST KUCHLIN, *Taschenbuch der Physik*.

2.7 Variable Gap Sensor

The physical principle of variable gap magnetic sensors is based on the variation of an air gap within a magnetic circuit. These sensors may be realized by winding a coil on

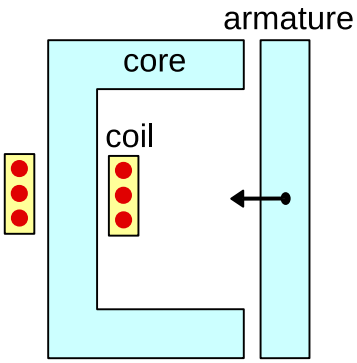

Figure 2.34: Example of a variable gap sensor.

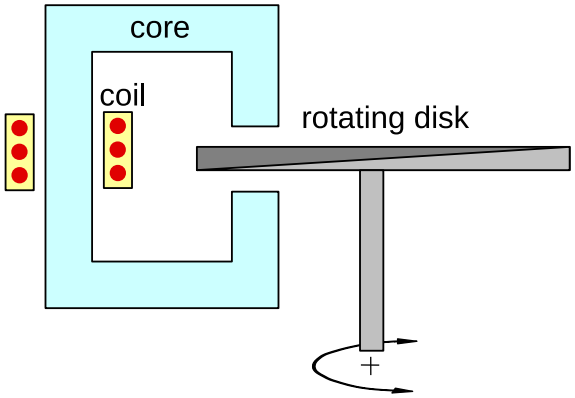

Figure 2.35: Angle sensor system.

a C-shaped magnetic core as shown in Fig. 2.34. A displacement change of a magnetic armature leads to a change in the inductance L of the coil and therefore the electric impedance can be used as a measure for the armature displacement. Variable gap sensors show some advantages in the measurement of small displacements. They are highly sensitive with a resolution of less than $1\,\mathrm{nm}$ [1] and the maximum non-linearity typically is given as 0.5% [2]. They are used for non-contact applications and can be used to measure angles in a form as shown in Fig. 2.35 with a rotating disk of variable thickness. In this example, simple formulas for estimating the inductance of the angle sensor as a function of design parameters are derived followed by a comparison with more precise finite element models considering scattering effects within the air gap.

Estimating the magnetic inductance

For induction phenomena, the displacement current term $\partial_t \mathbf{D}$ within the Maxwell' equations can be neglected, hence to estimate the inductance L, we can work with the Maxwell' equations $\nabla \times \mathbf{H} = \mathbf{j}_{\mathrm{cond}}$ and $\nabla \times \mathbf{E} + \partial_t \mathbf{B} = 0$. By considering an oriented surface F with boundary ∂F and normal $\mathbf{n}$, their integral form after application of Stoke's theorem reads

$$\oint_{\partial F} \mathbf{H} \cdot \mathrm{d}\mathbf{s} = \int_F \mathbf{j}_{\mathrm{cond}} \cdot \mathrm{d}\mathbf{n} \quad \text{and} \quad \oint_{\partial F} \mathbf{E} \cdot \mathrm{d}\mathbf{s} = -\frac{\partial}{\partial t} \int_F \mathbf{B} \cdot \mathrm{d}\mathbf{n}\,. \tag{2.18}$$

By considering the surface $F = A$ as shown in Fig. 2.36, from the first integral in (2.18), we obtain

$$N\,I = \oint_{\partial A} \mathbf{H} \cdot \mathrm{d}\mathbf{s} = \oint_{\partial A} \mu^{-1} \mathbf{B} \cdot \mathrm{d}\mathbf{s}\,, \tag{2.19}$$

with N the number of coil windings and I the coil's current. The normal component of the magnetic induction $\mathbf{B}$ is continuous across the boundary between air and magnetic material, hence to a first order we can assume $|\mathbf{B}|$ to be constant along the closed path ∂A resulting in

$$N\,I \approx |\mathbf{B}|_{\mathrm{core}} \left(\frac{l_{\mathrm{core}}}{\mu_{\mathrm{core}}} + \frac{d(\alpha)}{\mu_{\mathrm{disk}}} + \frac{h - d(\alpha)}{\mu_0} \right) .$$

with h , $d(\alpha)$ as defined in Fig. 2.36 and l_{core} the lenght of the magnetic core with large permittvity μ_{core}. By considering the surface $F = S_{\mathrm{core}}$ to be a section of the core and

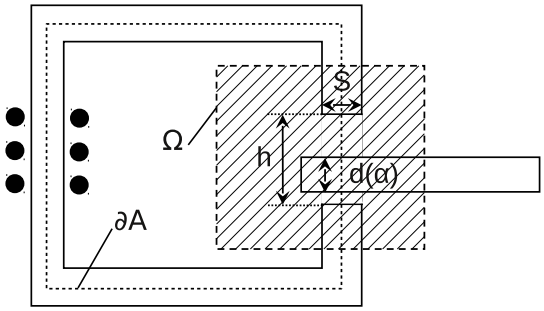

Figure 2.36: Simply connected simulation domain Ω without currents.

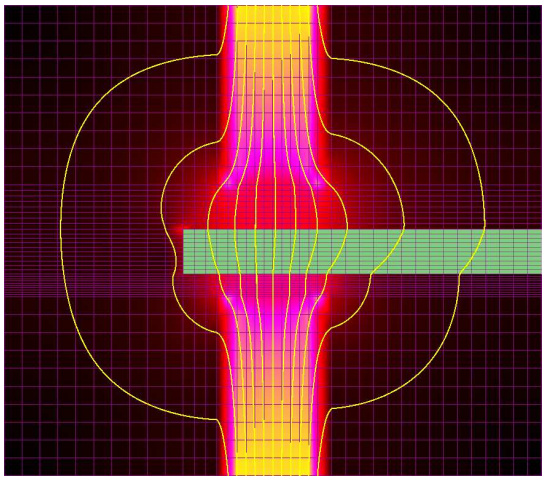

Figure 2.37: Magnetic field around disk.

that here the induction field $\mathbf{B}$ due to the high permittivity is almost constant, from the second integral in (2.18), we obtain

$$U = N \oint_{\partial S_{\text{core}}} \mathbf{E} \cdot \mathbf{ds} \approx -N S_{\text{core}} \frac{\partial \mathbf{B}}{\partial t} \approx -N^2 S_{\text{core}} \left(\frac{l_{\text{core}}}{\mu_{\text{core}}} + \frac{d(\alpha)}{\mu_{\text{disk}}} + \frac{h - d(\alpha)}{\mu_0} \right)^{-1} \frac{dI}{dt},$$

with U the induced coil's voltage. Since the inductance L is defined by the relation $U = -L\, dI/dt$, we have

$$L \approx N^2 S_{\text{core}} \left(\frac{l_{\text{core}}}{\mu_{\text{core}}} + \frac{d(\alpha)}{\mu_{\text{disk}}} + \frac{h - d(\alpha)}{\mu_0} \right)^{-1}. \tag{2.20}$$

Numerical model

The hand formula (2.20) for the inductance L has been derived by considering the induction field $|\mathbf{B}|$ to be constant along the integration path ∂A, see Fig. 2.36. Within the air gap, see Fig. 2.37, this may be a crude approximation and therefore for comparison purposes, a numerical model is developed including 2D scattering effects within the air gap. The input file for this example can be found at `example/AngSens.s3d`. The shaded simply connected 2D computational domain $\Omega = \Omega_{\text{core}} \cup \Omega_{\text{disk}} \cup \Omega_{\text{air}}$ shown in Fig. 2.36 is chosen such that there is no current $\mathbf{j} = 0$ and therefore $\nabla \times \mathbf{H} = 0$ holds in Ω. This condition implies the existence of a scalar magnetic potential ϕ in Ω with $\mathbf{H} = \nabla\phi$. Together with the Maxwell's equation $\nabla.\mathbf{B} = 0$, we obtain the Poisson equation $\nabla.(\mu\nabla\phi) = 0$ to be solved with $\mu = \mu_{\text{core}}$ in core Ω_{core}, $\mu = \mu_{\text{disk}}$ in the disk Ω_{disk} and $\mu = \mu_0$ in the air Ω_{air}. On the two surfaces $\partial\Omega \cap \Omega_{\text{core}} = \partial\Omega_0 \cup \partial\Omega_1$, we set Dirichlet BCs with constant values of $\phi = 0$ and $\phi = \Delta$ a free parameter. On $\partial\Omega - (\partial\Omega_0 \cup \partial\Omega_1)$ we use homogenous Neumann BCs with $\mathbf{B} \cdot \mathbf{n} = 0$. This problem is solved with 2nd order elements by enabling the equation `MagnetoStatic2` requiring the definition of the material parameter `Mue.Val` defining the relative permittivity μ/μ_0 and by setting the above Dirichlet BCs for the dof-field `MagnPot`.

By splitting the evaluation of (2.19) in two parts, with the contribution from the core as before, but now with a numerical contribution from our computational domain Ω,

we have

$$N\,I = \int_{\partial A\cap\Omega^c} \mathbf{H}\cdot \mathrm{d}\mathbf{s} + \int_{\partial A\cap\Omega} \nabla\phi\cdot d\mathbf{s} = |\mathbf{B}|_{\mathrm{core}}\frac{l^c_{\mathrm{core}}}{\mu_{\mathrm{core}}} + \Delta\,.$$

with l^c_{core} the length of the core outside the computational domain. The value $|\mathbf{B}|_{\mathrm{core}}$ can by approximated by the average

$$|\mathbf{B}|_{\mathrm{core}} \approx S^{-1}_{\mathrm{core}} \int\limits_{\partial\Omega_1} \mathbf{B}\cdot \mathrm{d}\mathbf{n} = S^{-1}_{\mathrm{core}}\mathcal{F}(\Delta)\,,$$

with $\mathcal{F}(\Delta)$ the BC flux directly available as *SESES* output. Due to the linearity of the problem, we have $\mathcal{F}(\Delta) = \mathcal{F}(1)\Delta$ so that $NI = (S^{-1}_{\mathrm{core}}\mu^{-1}_{\mathrm{core}}l^c_{\mathrm{core}}\mathcal{F}(1)+1)\Delta$ and evaluation of the second integral in (2.18) now yields

$$U = -N\frac{d}{dt}\int_{S_{\mathrm{core}}} \mathbf{B}\cdot \mathrm{d}\mathbf{n} = -N\mathcal{F}(1)\frac{d\Delta}{dt} = -N^2\mathcal{F}(1)\left(\frac{l^c_{\mathrm{core}}\mathcal{F}(1)}{S_{\mathrm{core}}\mu_{\mathrm{core}}}+1\right)^{-1}\frac{dI}{dt}\,,$$

resulting in the inductance

$$L = N^2 S_{\mathrm{core}}\left(\frac{l^c_{\mathrm{core}}}{\mu_{\mathrm{core}}} + \frac{S_{\mathrm{core}}}{\mathcal{F}(1)}\right)^{-1}\,.$$

It is apparent that the relative error between both approximations of L tends to zero with $l_{\mathrm{core}} \to \infty$ and so we limit ourselves by comparing the different terms within the denominator

$$\mathcal{R} = \left(\frac{S_{\mathrm{core}}}{\mathcal{F}(1)}\right) \Big/ \left(\frac{(l_{\mathrm{core}} - l^c_{\mathrm{core}})}{\mu_{\mathrm{core}}} + \frac{d(\alpha)}{\mu_{\mathrm{disk}}} + \frac{h-d(\alpha)}{\mu_0}\right)\,,$$

where a value of $\mathcal{R} = 1$ implies equivalency. From the textual output generated by running the example, it is apparent that with a value of $\mathcal{R} \approx 0.5$ the scattering of the magnetic field in the air cannot be neglected and that by rotating the disk, this large error is also not constant. Formula (2.20) is therefore a crude approximation and it is also expected that a 3D simulation will further improve the value of L considerably.

References

[1] P. KRITSCHKER, T. GAST, *Inductive Cross-Anchor Detector with High Resolution*, Technisches Messen 49, No. 2, pp. 43-49, 1982.

[2] J. P. BENTLEY, *Principles of Measurements Systems*, New York Longman Inc. 1983.

2.8 Modeling of a micro-reed switch

Micro-reed relays with a magnetostatic operating principle were further miniaturized during the last years. These electrical switches provide the galvanic separation of the control and load circuit, so they are of special interest in micro system engineering

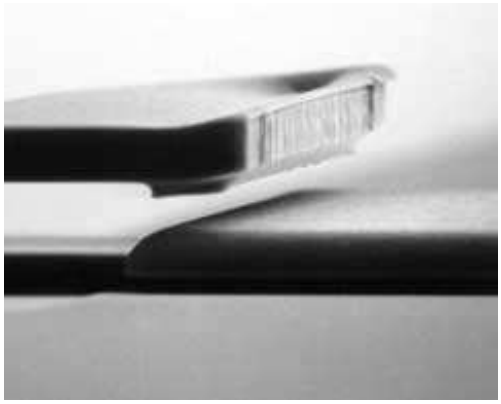

Figure 2.38: Detail view of a micro-reed relay. Components manufactured according to this process are $2 \times 1.4 \times 0.75\,\mathrm{mm}^3$ in size.

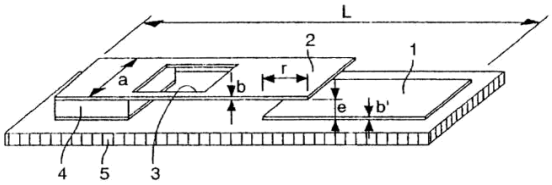

Figure 2.39: Schematic drawing of a micro-reed relay, [3].

applications [1]. Magnetostatic sensors are particularly valuable for portable applications because they require little energy or space, they are sealed and can withstand high mechanical stresses. The reed switches currently found on the market are made of two ferromagnetic overlapping metal blades placed in a glass tube [2]. One of the blades is fixed to the substrate and the other one is free to bend down and to touch the fixed blade in order to close the contact, see Fig. 2.38-2.39. The magnetic and mechanical forces acting on the bendable lamella are shown in Fig. 2.40. The mechanical force F_{mech} is represented by the flat surface, it is a linear function of the contact distance and it does not depend on the magnetic field. The magnetic force F_{mag} is a nonlinear function of the contact distance and the magnetic field. Both forces are at the intersection of both surfaces in equilibrium. If the magnetic force drops below the value at the point 3, then the relay opens and enters the new equilibrium state as indicated by point 4. If the external magnetic field is increased and exceeds the value at point 1, then the relay closes again. The mobile lamella snatches down and the new equilibrium state is indicated by point 2. An optimal relay is characterized by a low value of the magnetic field necessary for closing the contact. The switching hysteresis is represented by the path along the points 1, 2, 3 and 4 and for a reliable operation of the relay, it is important for the hysteresis not to be too small in order to prevent a flutter of the contact at the time of switching.

Mathematical model

For this micro-reed relay, the mathematical model is represented by the magnetostatic equations obtained from the Maxwell's equations with the assumptions of stationarity, zero charges and zero electrical fields. The equations to be solved for the magnetic field $\mathbf{B}$ and $\mathbf{H}$-field are given by

$$\nabla.\mathbf{B} = 0\,, \quad \nabla \times \mathbf{H} = \mathbf{J}_0\,,$$

with $\mathbf{J}_0$ a given external current. If we assume that no electrical current flows in the lamellas, then $\mathbf{J}_0 = 0$ since no current flows in vacuum. The relation $\nabla \times \mathbf{H} = 0$

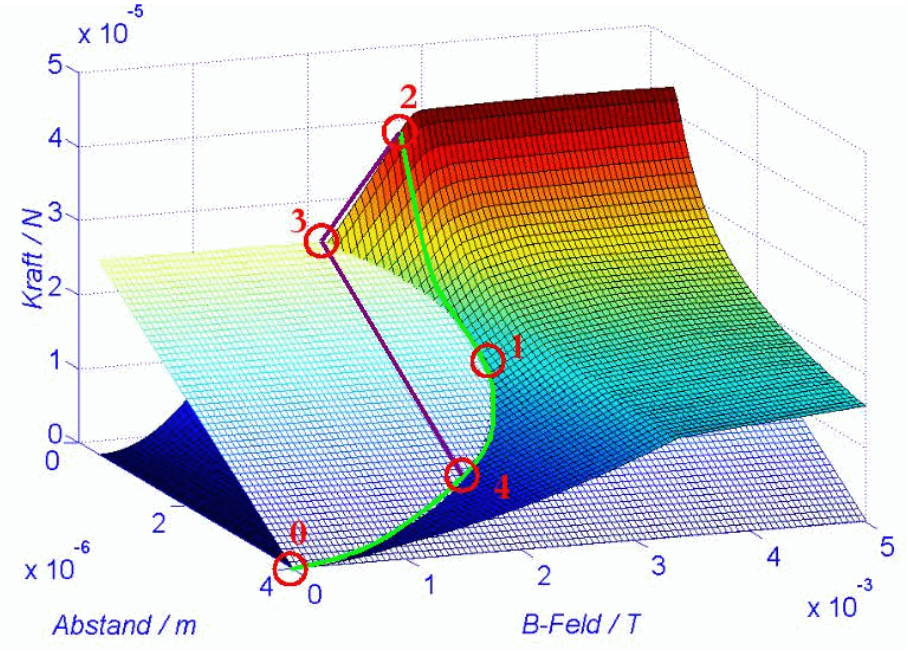

Figure 2.40: Magnetic force and mechanical force as function of the contact distance and the magnetic field B.

expresses the fact that the **H**-field can now be given as the gradient of a magnetic potential $\mathbf{H} = -\nabla\Theta$ and by using the constitutive relation $\mathbf{B} = \mu\mathbf{H}$, a Poission equation for the magnetic potential Θ is left to be solved

$$\nabla.(\mu_0\,\mu_r\,\nabla\Theta) = 0\,, \tag{2.21}$$

with $\mu = \mu_0\mu_r(H)$, μ_0 the vacuum permeability and $\mu_r = \mu_r(H)$ the relative permeability in general a non-linear function of $H = |\mathbf{H}|$. For this switch, we are mainly interested in the magnetostatic forces acting on the lamella responsible for turning on/off the switch and two methods are available here. The first method is based on computing the magnetic energy inside in the domain Ω given by

$$W = \int_\Omega E(B)\mathrm{d}V\,, \tag{2.22}$$

with $E(B)$ the magnetic energy density [4]

$$E(B) = \int_0^B \mathbf{H}(B')\cdot\mathrm{d}\mathbf{B}'\,. \tag{2.23}$$

Let d represents the distance between the lamellas and view this value as an independent parameter. The energy becomes a function of this parameter $W = W(d)$ and for small changes $d' = d + \delta d$, we can write to a first order $W(d') = W(d) + (\partial W)/(\partial d)\delta d$. The change in energy $(\partial W)/(\partial d)\delta d$ can now be interpreted as the work done by the total force acting on the lamellas times the displacement and so

$$F = -\frac{\partial W}{\partial d}\,, \tag{2.24}$$

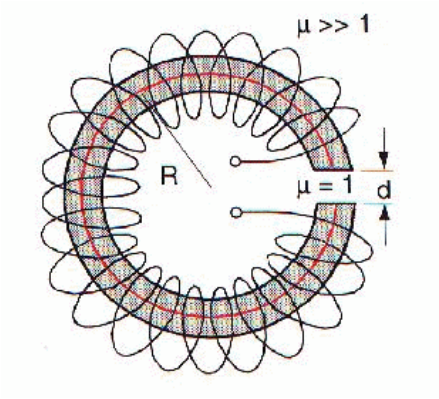

Figure 2.41: Ring coil with air gap.

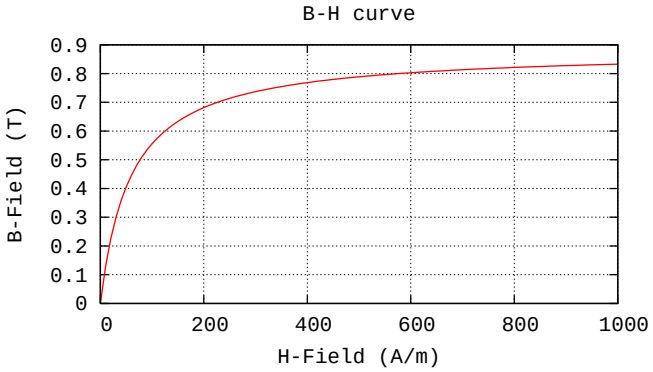

Figure 2.42: Saturation of the magnetic field B for large values of the H-field.

can be interpreted as the component of this force along the displacement. This method requires to compute two solutions for small different values of d and obtains the force by numerical difference of the magnetic energy. If the magnetic energy density $E(B)$ is too complex to compute, there is an alternative based on computing the $\mathbf{B}$- and $\mathbf{H}$-field only. Taking the derivative of (2.22)-(2.23), we obtain

$$\begin{aligned} \Delta W = & \int_\Omega \mathbf{H}(B) \cdot \Delta\mathbf{B}\,\mathrm{d}V + \int_\Omega E(B)\,\Delta\mathrm{d}V & (2.25)\\ = & \int_{\Omega_L} \mathbf{H}(B) \cdot \Delta\mathbf{B}\,\mathrm{d}V + \Delta \int_{\Omega/\Omega_L} \tfrac{1}{2}\mathbf{H}\cdot\mathbf{B}\,\mathrm{d}V\,. & (2.26)\end{aligned}$$

Since the parameter d changes the domain's shape, we have to consider also the possible change in the integration domain, formally denoted by $\Delta\mathrm{d}V$. It seems that still we need to know the energy density $E(B)$, however, we may well assume that the ferromagnetic material just undergoes rigid body deformations without mechanical strain and therefore $\Delta\mathrm{d}V = 0$ on the lamella's domain Ω_L. Just in vacuum Ω/Ω_L there is a change in the domain, but there we have $E(B) = \mathbf{H}\cdot\mathbf{B}/2$.

The second method is based on Maxwell's stress tensor, it is more general and allows to compute the mechanical tractions on any portion of the lamella boundary whereas the first method just yields total forces for a given and prescribed displacement. It is therefore possible to perform a coupled magnetostatic-mechanical analysis and to follow in details the dynamic of the switch process. The magnetic force per unit volume in the ferromagnetic material is $\mathbf{J}\times\mathbf{B} = (\nabla\times\mathbf{H})\times\mathbf{B}$, integration over the lamella's volume Ω_L, application of Gauss theorem and after some algebra, we obtain

$$\mathbf{F} = \int_{\Omega_L} \mathbf{J}\times\mathbf{B}\,\mathrm{d}V = \int_{\partial\Omega_L} \tau\cdot\mathrm{d}\mathbf{n}\,, \qquad (2.27)$$

with $\tau_{ij} = (B_iH_j - 1/2\delta_{ij}\mathbf{B}\cdot\mathbf{H})$ the Maxwell stress tensor for the magnetostatic case and a scalar permeability. The total force acting on a lamella is now given by a surface integral over the lamella boundary $\partial\Omega_L$ and therefore the term $\tau\cdot\mathbf{n}$ with $\mathbf{n}$ the outward normal can be interpreted as the mechanical tractions acting locally on the surface. It is to be noted that if only the total force is of interest, then the volume Ω_L can be taken as any volume around the lamella, since in vacuum $\mathbf{J} = 0$ and $\nabla.\tau = 0$ and therefore the integral's value is invariant.

It is possible to get a rough estimate of the magnetic force by considering a magnetic ring core of cross-section A, length L and with an air gap of d, see Fig. 2.41. By dis-

carding all volume effects and by considering just the fields along the centerline of the torus, from the relation $\nabla.\mathbf{B} = 0$ follows a constant magnetic field $B = |\mathbf{B}|$ along this line. In order to compute the magnetic force, we need to consider the magnetic energy as function of the air gap d. The volume of the ferromagnetic material is constant and since B is constant as well, the magnetic energy stored in the ferromagnetic material does not depend on d. The only contribution stems from the magnetic energy stored in the air gap which is given by $W_{\text{gap}} = d\,A\,B^2/(2\mu_0)$ and derivative with respect to d as of (2.22) yields the force $F_{\text{mag}} = AB^2/(2\mu_0)$. This force grows quadratically in B but just for small B values since afterwards volume and saturation effects will limit the growth, as schematically shown in Fig 2.40.

Numerical model

We present here some numerical results for the micro-reed relay and the *SESES* input file can be found at `example/MicroRelay.s3d`. Most of the `Initial` section is concerned with the construction of the macro element mesh for the relay and vacuum region. For the ferromagnetic material, we use the material law

$$B(H) = H\frac{H\mu_{\text{asim}}\left(\mu_{\text{init}} - \mu_{\text{asim}}\right) + B_{\text{sat}}\mu_{\text{init}}}{H\left(\mu_{\text{init}} - \mu_{\text{asim}}\right) + B_{\text{sat}}}\,, \tag{2.28}$$

with B_{sat} the magnetic field value where saturation starts to act and μ_{init}, μ_{asim} the slopes of the B-H-curve for zero and ∞-values

$$\lim_{H\to 0}\frac{\partial B(H)}{\partial H} = \mu_{\text{init}} \text{ and } \lim_{H\to\infty}\frac{\partial B(H)}{\partial H} = \mu_{\text{asim}}\,.$$

For this model shown in Fig. 2.42 for $B_{\text{sat}} = 0.88\,\text{T}$, $\mu_{\text{init}} = 12000$ and $\mu_{\text{asim}} = 1$, we can easily obtain the inverse relation $H = H(B)$ and the energy density integral (2.23) in analytical form. In *SESES* this material law is defined through the material parameter `Mue`, requiring the relative permeability $\mu_r = \mu_0^{-1}H^{-1}B(H)$ and its logarithmic derivative $H^{-1}\partial\mu_r/\partial H$. The relay is embedded in a homogeneous external magnetic field with only a z-component and so on the bottom face of the rectangular domain we set the magnetic potential Θ to zero and on the top face we prescribe the magnetic field B_z which may be done with either a `Neumann` or a `Floating` BC. To compute the force on the relay with the surface integral (2.27), we define the boundary of the bendable lamella with the statements

```
BC SurfaceReed  Restrict material(Vacuum) OnChange  material(ReedL)
BC SurfaceCheck Restrict material(Vacuum) OnChange  material(Vacuum)
```

It is important to use the `Restrict` option to only select the boundary part external to the lamella, since it is in the vacuum region that we want to perform the numerical integration. The second boundary `SurfaceCheck` selects the whole vacuum's boundary and it is used to check the accuracy of the surface integral. We end up the `Initial` section by defining postprocessing routines for computing the Maxwell's stress tensor and magnetic energy density.

In the `Command` section we define the solution algorithm and post-processing tasks. The ferromagnetic material law is non-linear and so we set up a non-default convergence criterion to stop Newton's algorithm and to speed-up the computation we reuse

the factorized linear system for six consecutive linear steps. Fig. 2.43 shows a solution of the magnetic potential and the magnetic field is shown in Fig. 2.44. Afterwards we compare several methods to obtain the magnetic force acting on the bendable lamella. The first method is the Maxwell's stress tensor method of (2.27), the second method is a numerical improvement of the first one and can be used whenever the integrand function is divergence free, which is always the case for the Maxwell's stress tensor in vacuum. The third method is the energy method of (2.24) requiring the magnetic energy density (2.23) and the fourth and fifth methods are the energy methods of (2.25)-(2.26). All energy methods require a second solution computed for a slight change of the distance parameter. The fourth method requires to backup the magnetic field and volume weights used for numerical integration and the numerical integration must be self-programmed with the help of the built-in routine `traverse`. The fifth method is a hybrid between the third and fourth methods and does not require to backup the volume weights since in vacuum we work with the magnetic energy density.

```
Solve Stationary
Define Force0=integrate(Bound SurfaceReed;  MagnTraction())
       Force1=residual( Bound SurfaceReed;  MagnStress())
       ZeroC0=integrate(Bound SurfaceCheck; MagnTraction())
       ZeroC1=residual(Bound SurfaceCheck;  MagnStress())
       Energy0=integrate(MagnEnergy())
       Energy1=integrate(Domain material(Vacuum); MagnEnergyLin())
Store BfieldBack=Bfield, WeightBack=VolWeight
Solve Stationary ForSimPar LDisp At LDisp+LEps
Write
  "  Bexternal = %e\n  LDisp = %e um\n" BExt LDisp
  "  ForceStress0 = %E N Accuracy = %E\n" Force0[2], ZeroC0[2]
  "  ForceStress1 = %E N Accuracy = %E\n" Force1[2], ZeroC1[2]
  "  ForceEnergy0 = %E N\n"  1E6*(Energy0-integrate(MagnEnergy()))/LEps
  "  ForceEnergy1 = %E N\n"
  { ForceZ=traverse(MagnDeltaEnergy()); return 1E6*ForceZ/LEps; }
  "  ForceEnergy2 = %E N\n"
  {  ForceZ=Energy1-integrate(Domain material(Vacuum); MagnEnergyLin())
                   +traverse(Domain !material(Vacuum); MagnDeltaEnergyMat());
     return 1E6*ForceZ/LEps; }
```

The accuracy of the stress tensor methods can be approximately obtained by integrating the divergence free integrand over the vacuum surface `SurfaceCheck`. The results for a single run are as follows

```
ForceStress0 = 1.073258E-06 N Accuracy = 7.838263E-08
ForceStress1 = 1.402041E-06 N Accuracy = -6.623098E-08
ForceEnergy0 = 1.432143E-06 N
ForceEnergy1 = 1.432908E-06 N
ForceEnergy2 = 1.428980E-06 N
```

The most inaccurate method is generally the first one, whereas the third, fourth and fifth methods are in general the most accurate and stable ones, they all should give the same results up to some numerical cancellation. The discrepancy here is caused by the fact the our deformation is not actually a truly rigid body deformation since some lamella's elements are actually streched. The second method is quite close to the energy methods but in general tends to give lower force values.

References

[1] J. SCHIMKAT, *Grundlagen und Modelle zur Entwicklung und Optimierung von*

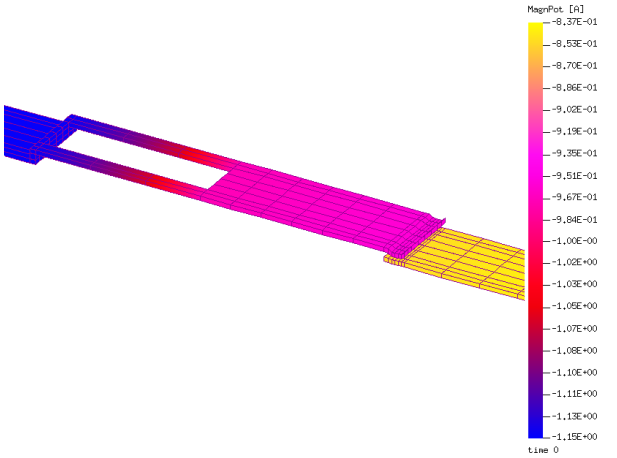

Figure 2.43: Numerical solution for the magnetic scalar potential Θ.

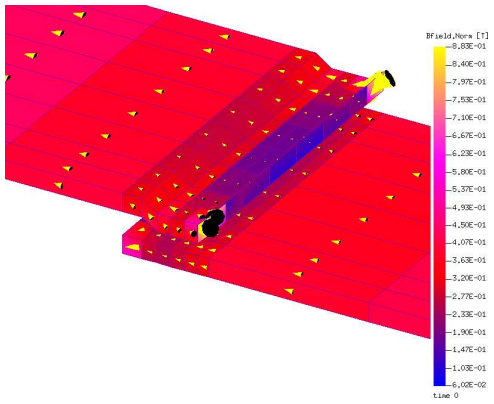

Figure 2.44: Computed magnetic field $\mathbf{B}$ with arrows indicating the direction and intensity around the lamellas.

Silizium-Mikrorelay, Dissertation, Technische Universität Berlin, FB Maschinenbau und Produktionstechnik, 1996.

[2] URL: http://www.asulab.ch/de/composants_de.html

[3] F. GUEISSAZ, D. PIGUET, *The microreed, an ultra-small passive MEMS magnetic proximity sensor designed for portable applications*, Proceedings of the 14th IEEE International Conference on Micro Electro Mechanical Systems, MEMS 2001, Interlaken, Switzerland, January 2001.

[4] J. D. JACKSON, *Classical Electrodynamics,* John Wiley & Sons, 1999.

2.9 Skin and Proximity Effect

Skin and proximity effect belong to the class of eddy current problems in electromagnetism where time dependent magnetic fields induce electrical currents in conducting wires. For an AC driven current in a wire, the harmonic magnetic field generates eddy currents but with opposite direction of the driving one so that the current tends to flow within a thin layer at the surface of the wire. This is the classical skin effect and it is dissipative in nature so that the wire effective resistance is larger than the one for a DC current and grows with the square root of the frequency. Due to the reduced current flow, the inductance of the wire will be smaller at higher frequencies. When multiple wires are close together as for example in a coil, on each wire acts the skin effect, but it is also true that the magnetic field created by each wire will reduce the current flow in the other wires. This disturbance is called proximity effect due to the fact that the wires are close together and they disturb each other. The proximity effect is also dissipative in nature, but differently from the skin effect the effective resistance of the wire grows quadratically with the frequency and it is dominant at low frequencies. As an example, already at audio frequencies, resonant circuits with high quality factors need to consider effective resistance values for coils. In this example, we are

going to investigate the skin and proximity effect in wires and coils and from finite element computations derive effective values for resistance and inductance. For a single straight wire, a 2D computation is well suited, but for coils an exact analysis would require a 3D computation. This is still a very expensive operation, but if one is ready to accept a small analysis error then valid 2D formulations are available and will be presented.

General Theory

Starting from the Maxwell's equations, one arrives at the eddy current equations by assuming zero displacement currents $\partial_t \mathbf{D} = 0$, which for good conductors are typically small and therefore can be neglected. For an harmonic analysis at a fixed frequency, we may use the complex notation with a time dependency of the form $(.)(t) = (.)e^{i\omega t}$, which save us from directly integrating the governing equation with respect to time. Within the conducting wire, we have the material law $\mathbf{J} = \sigma\mathbf{E}$ and by considering a homogeneous conductivity σ, from the property $\nabla.\mathbf{J} = 0$, we obtain $\nabla.\mathbf{E} = 0$. With these assumptions and for a 2D domain, one then derives the following governing equations

$$\nabla\mu^{-1}\nabla A_z - i\omega\sigma A_z + J_{0z} = 0\,, \tag{2.29}$$

with $\mathbf{A} = (0, 0, A_z)$ the vector potential to be computed and $\mathbf{J}_0 = (0, 0, J_{0z})$ the driving external current both complex quantities. The resulting magnetic field is given by $\mathbf{B} = \nabla \times \mathbf{A}$ and the electric field by $\mathbf{E} = -\partial_t \mathbf{A} = -i\omega\mathbf{A}$. The effective current has only a non-zero z-component $\mathbf{J} = (0, 0, J_z)$ given by

$$J_z = J_{0z} - i\omega\sigma A_z\,. \tag{2.30}$$

Let us assume we want to compute the skin effect on a straight circular wire of radius r_0, a problem with a known analytical solution, see [1]. We start by fixing the frequency ω, choose a large enough 2D domain Ω to represent the whole plane and define in the middle the cross section of the wire with a constant conductivity of σ. On the boundary $\partial\Omega$, we apply BCs that best represent an infinite domain and on the wire we apply a driving harmonic electrical field $E_0 e^{i\omega t}$ constant over the cross section. Without loss of generality, we may assume E_0 to be real. For a wire length of ℓ, the applied voltage at the wire ends will be $V = \ell E_0 e^{i\omega t}$ which is also constant over the cross section. The impedance of the wire is given by $Z = V/I$ with I the total wire current computed by the integrals

$$I = \int_{\text{wire}} J_z \mathrm{d}A = \int_{\partial\text{wire}} \mu^{-1}\mathbf{B}\cdot\mathrm{d}\mathbf{l} = \int_{\partial\text{wire}} \mu^{-1}\nabla A_z \cdot \mathrm{d}\mathbf{n}\,. \tag{2.31}$$

By splitting the real and imaginary parts of $Z(\omega) = R(\omega) + i\omega L(\omega)$, we obtain the effective resistance $R(\omega)$ and inductance $L(\omega)$ both depending on the frequency ω

$$R = \frac{\ell\, E_0 \Re(Ie^{-i\omega t})}{|I|^2}\,,\quad \omega L = -\frac{\ell\, E_0 \Im(Ie^{-i\omega t})}{|I|^2}\,. \tag{2.32}$$

For a single wire, one knows in advance that the inductance per unit of length is infinite and a computation just yields the value of R whereas L grows with the logarithm

of the computational domain size. In general and for 2D computations, just for a system of straight wires with a total zero current is the inductance finite. Therefore for a single wire, one is limited at computing the internal inductance by excluding the contribution from the free space. This is also easily done analytically. Substitution of (2.30) into (2.29) and by considering rotational symmetry, we obtain the Bessel's eq. $(\partial_r^2 + r^{-1}\partial_r + \alpha^2)J_z = 0$ with $\alpha^2 = -i\omega\mu_0\sigma$. The solutions are Bessel's functions and with the BC $J_z(r_0) = \sigma E_0 e^{i\omega t}$, we have

$$J_z = \sigma E_0 \frac{J_0(\alpha r)}{J_0(\alpha r_0)} e^{i\omega t} \text{ with } J_0(x) = \sum_{m\geq 0} \frac{(-1)^m (\frac{x}{2})^{2m}}{(m!)^2} .$$

Computing the integral (2.31) and insertion into (2.32) yields the values

$$R = -\frac{\ell}{\sqrt{2}\pi r_0 \sigma\delta} \Re\left[\sqrt{-i}\frac{J_0(\alpha r_0)}{J_0'(\alpha r_0)}\right] , \quad \omega L = -\frac{\ell}{\sqrt{2}\pi r_0 \sigma\delta} \Im\left[\sqrt{-i}\frac{J_0(\alpha r_0)}{J_0'(\alpha r_0)}\right] , \tag{2.33}$$

with $\delta = \sqrt{2/(\omega\mu_0\sigma)}$ the skin depth and J_0' the derivative of J_0. The DC values are $R = \ell/(\pi\sigma r_0^2)$ and $L = \mu_0/(8\pi)$.

Now that we have seen how to compute resistance and inductance values of a straight wire, we want to do the same for an air coil with N-windings. Here the matter is a little bit more involved since only 3D computations can truly compute the magnetic field in a coil, however, these computations are too expensive for our purposes and we therefore look for valid 2D alternatives. The main idea is to consider the coil as a rotational symmetric structure by neglecting the spiral nature of the wire and by considering the windings as separate independent wires. In doing this, we assume each wire to be a ring carrying the same total current $I = I_0 e^{i\omega t}$. The solution of our coil problems then consists in determining which cross sectional electric field $E_{0i}e^{i\omega t}$ need to be applied in each wire $i = 1 \dots N$ so that the wire current is equal to I. Differently from before, we now prescribe the driving current instead of the voltage and without loss of generality we may assume I_0 to be real. If we assume the driving electric field E_{0i} to be constant over the i-th wire cross section then to a first order approximation the total voltage can be expressed as a sum of path integrals along the center of the rings with radius R_i

$$V = 2\pi \sum_{i=1}^{N} R_i E_{0i} e^{i\omega t} . \tag{2.34}$$

Clearly by taking another path within the rings other results are obtained since the path length is different. This is slightly incorrect and a better model is to consider the driving voltage $V_{0i}e^{i\omega t}$ constant on a wire cross section instead of the electric field $E_{0i}e^{i\omega t}$ which results in an electric field $E_{0i} = V_{0i}/(2\pi r)$ inversely proportional to the radius coordinate r. So the native variables to be determined are now the wire driving potentials V_{0i} and the question remains how to compute them ? Since the whole formulation is linear, the eddy currents induced by a wire into the other ones add linearly with all other contributions and the fields V_{0i} can be obtained by solving the following N-problems. For each i-th problem with $i = 1 \dots N$, set the following probe voltage $V_{0ij} = \delta_{ij}$ on each wire $j = 1 \dots N$ and compute the resulting wire current G_{ij} for $j = 1 \dots N$. The potential fields V_{0j} to be applied to obtain the current

I_0 in each wire are then given by solving the complex linear system of equations with the conductance matrix G

$$\sum_{j=1}^{N} G_{ij} V_{0j} = I_0 \text{ for } i = 1 \dots N \,, \tag{2.35}$$

and effective resistance and inductance are computed as

$$R = I_0^{-1} \sum_{i=1}^{N} \Re(V_{0i}) \,, \quad \omega L = I_0^{-1} \sum_{i=1}^{N} \Im(V_{0i}) \,. \tag{2.36}$$

Numerical Results

In a first example `example/Skin.s2d`, we compute the internal resistance and inductance of a straight circular copper wire for a frequency range of $0 - 25\,\mathrm{kHz}$. The skin effect will limit the current approximately to a layer of thickness δ which need to be resolved by the computational mesh. For copper, we have $\sigma = 56{\times}10^6\,\mathrm{s/m}$, resulting in $\delta \approx 0.4\,\mathrm{mm}$ at $25\,\mathrm{kHz}$. Having defined the geometry and a fine enough mesh near to the boundary, we enable the equation `EddyHarmonic` for eddy harmonic analysis. The driving voltage is applied by defining the external current `Current0.Z` as $J_0 = \sigma E_0$ within the wire. At the wire boundary, the current should be J_0 at any frequency resulting in homogeneous Dirichlet BCs for the dof-fields `Az` and `iAz`. We define the frequency as a simulation parameter `frequency` and compute several solutions in our frequency range with the statement

```
Solve Stationary ForSimPar Omega Skip Step 1 Omega**0.95 Until 25000*2*PI
```

At each solution step, we compute the total current (2.31), resistance and inductance after (2.32) which are plotted in Fig. 2.45 together with the analytical values (2.33).

```
Write AtStep 1
  "f=%e Hz, I=(%9.2e,%9.2e)=%9.2e A, R=%9.2e Ohm, L=%9.2e H\n"
  {
     double c[2]=integrate(Current.Z,iCurrent.Z);
            c2=(c[0]*c[0]+c[1]*c[1]);
     return Omega/(2*PI),c,sqrt(c2),c[0]/c2,-c[1]/(c2*Omega);
  }
```

In a second example `example/Proxi.s2d`, we compute an air coil with N-windings at the single frequency $\omega = 2\pi \times 100\,\mathrm{kHz}$. The finite element mesh is constructed automatically starting from generic parameters like the number of wires, the wire diameter and the free space between the wires. Because the wire diameter is small with respect to the skin depth, we do not need to construct a finer mesh around the inner wire boundary. The construction of the mesh is the most complex part of this example and the other definitions are similar to the previous example, except that here we have to define the rotational symmetry. Around the coil, we have a cladding of free space which ideally should extent to infinity and on its boundary we have to set BCs for the dof-fields `Az` and `iAz`. Here we use the built-in function `Infinity` as replacement for homogeneous Neumann or Dirichlet BCs which better models an infinite domain. For this built-in to work properly, we have defined the coordinate origin in the middle

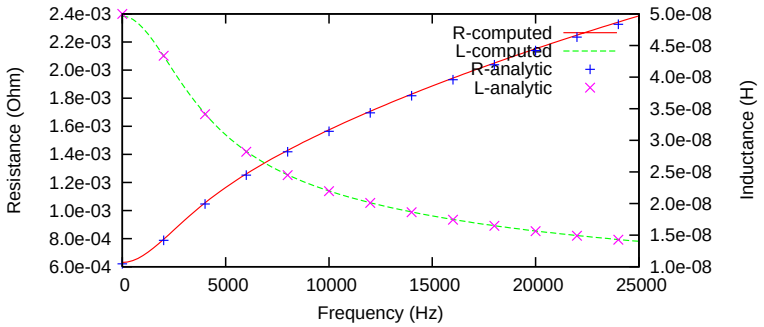

Figure 2.45: Internal resistance and inductance values of a straight circular wire compared with analytical values.

of the left side. On the symmetry axis, the proper BCs are homogeneous Dirichlet. In a series of N-linear problems, we set up the complex conductance matrix G_{ij} in order to compute the driving voltages V_{0j}. In order to easily apply the driving voltages, we have defined for each wire a corresponding domain of definition `Wire<i>` with $i = 0 \ldots N-1$ which are also used to compute the integral (2.31) for the wire current.

```
For j To nWind-1
{
   MaterialSpec Coil Parameter Current0.Z domain(Wire@j)*sigma/(2*PI*x) A/m**2
   Solve Stationary
   Define {  For i To nWind-1
   {  VAL=integrate(Domain domain(Wire@{i}); Current.Z,iCurrent.Z);
      MAT[@{2*nWind*(2*i+0)+2*j+0}]= VAL[0];
      MAT[@{2*nWind*(2*i+0)+2*j+1}]=-VAL[1];
      MAT[@{2*nWind*(2*i+1)+2*j+0}]= VAL[1];
      MAT[@{2*nWind*(2*i+1)+2*j+1}]= VAL[0];
   }  }
}
```

From the library `NmBlas.dll`, we use the routine `InvMatDotVec` to solve the linear system (2.35) with the conductance matrix stored in the global variable `MAT` and to obtain the driving voltages stored in `SOL`. This library is supplied together with the *SESES* binaries and its source code is available for reference in the *include* directory. Since *SESES* just works with real numbers, the conductance matrix `MAT` has actually a dimension of $2N \times 2N$ and solution `SOL` and right-hand side vector `RHS` a dimension of $2N$. The complex matrix inversion is simply obtained by storing instead of the single complex coefficient G_{ij}, the 2×2 real matrix

$$\begin{pmatrix} \Re G_{ij} & -\Im G_{ij} \\ \Im G_{ij} & \Re G_{ij} \end{pmatrix}$$

and for a complex vector component v_i, the two real components $(\Re v_i, \Im v_i)$.

```
Define
{  double R=0, wL=0;
   InvMatDotVec(MAT,RHS,SOL);
   For i To nWind-1
   {  write("Driving voltage[%.0f]=(%e %e) V\n",
      @i,SOL[@{2*i}],SOL[@{2*i+1}]); R=R+SOL[@{2*i}]; wL=wL+SOL[@{2*i+1}]; }
   write("R=%e Ohm, L=%e H\n",R, wL/(2*PI*frequency));
}
```

The effective resistance and inductance are given by (2.36) as the sum of the real and imaginary part of the driving voltages, since our computations are done with $I_0 = 1$.

These values are written to the output together with the single driving voltages for each wire and the following is the result.

```
Driving voltage[0]=(2.006826e-02 9.018853e-03) V
Driving voltage[1]=(2.001835e-02 9.859638e-03) V
Driving voltage[2]=(2.006826e-02 9.018853e-03) V
Driving voltage[3]=(2.214240e-02 1.120095e-02) V
Driving voltage[4]=(2.214240e-02 1.120095e-02) V
Driving voltage[5]=(2.431755e-02 1.057987e-02) V
Driving voltage[6]=(2.425840e-02 1.153813e-02) V
Driving voltage[7]=(2.431755e-02 1.057987e-02) V
R=1.773332e-01 Ohm, L=1.320940e-07 H
```

For visualization purposes in a final linear step, we compute the solution when all driving voltages V_{0j} act together.

```
MaterialSpec Coil
  Parameter Current0.Z  ( For i To nWind-1 { SOL[@{2*i+0}]*domain(Wire@{i})+ }
                          0)*sigma/(2*PI*x) A/m**2
  Parameter iCurrent0.Z ( For i To nWind-1 { SOL[@{2*i+1}]*domain(Wire@{i})+ }
                          0)*sigma/(2*PI*x) A/m**2
Solve Stationary
```

Although we solve several linear problems, the frequency is constant and so the linear system matrix. The linear problems just differ in a different right-hand side vector so that if we use a direct solver the linear matrix need to be factorized just once at the beginning. Keeping the factorized matrix in memory is done by defining the following statement at the beginning.

```
Increment ReuseFactoriz
```

As result, the time expenditure required to solve the N-linear problems is almost the same as to solve a single problem. Clearly if the number N of windings is large, then a large amount of finite elements are required to spatially resolve each wire which therefore is the major limiting factor.

References

[1] S. RAMO, J. R. WHINNERY, T. VAN DUZER, *Fields and waves in communication electronics*, Wiley, 1994.

2.10 Steel Hardening

Steel can be thermally treated to change and give it important properties as hardness, softness or to remove internal stresses generated by fabrication processes. Many treatments are available which include normalizing, anealing, hardening and tempering. Normalizing and anealing involve heating the steel above its critical temperature value, followed by holding this temperature for some time and a final cooling. This is done to remove internal stresses or to soften the steel. By anealing, the steel is slowly cooled in the furnace, whereas by normalization the cooling process takes place in the

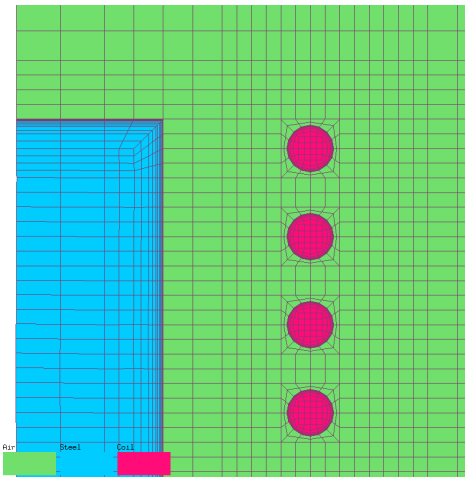

Figure 2.46: Computational mesh locally adapted to resolve the boundary layer.

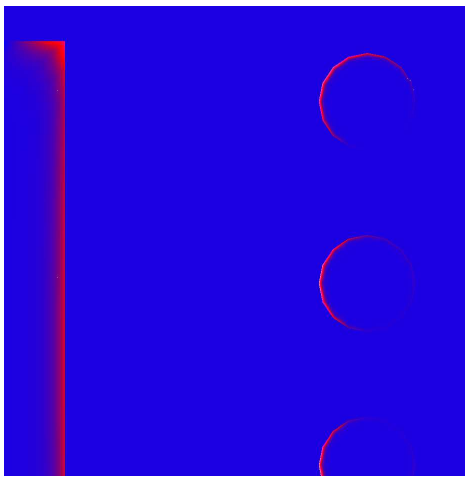

Figure 2.47: Heat boundary layer induced by the skin effect.

air. Hardening involves heating the steel to its normalizing temperature followed by a rapid cooling in a fluid as oil or water. In this way, the steel becomes very hard but also brittle. By tempering the steel is reheated to remove the brittleness created by the hardening process. If a furnace with controlled temperature is generally used for these thermal treatments, for small volumes an alternative is represented by a coil with AC current surrounding the steel object to be treated. By induction, eddy currents are generated and their power dissipation will heat up the steel.

From a numerical model as presented here, one can expect to answer questions as the temperature distribution in the steel, the dynamic behavior or the ability to optimize the coil geometry towards some given design goals. In this example, in order to reduce the complexity and to avoid 3D computations, we are going to assume rotational symmetry of the steel object and steady state conditions. Due to the elicity of the wire windings, this symmetry never holds for the coil, so that we make use of the proximity approximation as discussed and applied in the previous example. The computation of the eddy currents is done in the same way, hence we quickly review this matter and mainly present the thermal problem.

To solve the eddy current problem, we assume an harmonic excitation and a linear system, therefore allowing us to work with complex variables and fields all with an implicit time dependency of $e^{i\omega t}$ with ω the harmonic frequency. By the rotational symmetry, we use cylinder coordinates (ρ, ϕ, z) and the problem consists in computing the azimuthal component A_ϕ of the vector potential $\mathbf{A}$ given a driving electric field E^0_ϕ. The current is then given by $j_\phi = \sigma E_\phi = \sigma(E^0_\phi + E^{\text{ind}}_\phi)$ with $E^{\text{ind}}_\phi = -\partial_t A_\phi = -i\omega A_\phi$ the induced electric field and σ the electric conductivity. By symmetry, the other vector components of $\mathbf{A}$, $\mathbf{E}$, $\mathbf{j}$ are all zero. For a coil with N-winding, by neglecting their elicity, we consider separately the N-rings and for each ring we assume a constant voltage on each radial cross section. Hence for generic applied voltages V_i along the whole circumference with $i = 1 \dots N$ the wire index, approximately on each wire cross-section we have the driving electric field $E^0_{\phi i} = V_i/(2\pi\rho)$. These fields induce currents and the wire currents are given by evaluating the integrals $I_i = \int_{\Omega_i} j_\phi \mathrm{d}A$ over the wire

cross-sections Ω_i. However, a consistent solution is first given when there is the same current in each wire thus constraining the possible values of V_i. Such a solution can be computed by first solving N-problems with $j = 1 \dots N$ the problem index and driving potentials $V_{ij} = \delta_{ij}$ where δ_{ij} is the Kronecker's symbol. By post evaluating the wire currents G_{ij}, the driving potentials V_j to be applied for an overall consistent current in each wire of I are given by solving the linear system $\sum_{j=1}^{N} G_{ij} V_j = I$.

By assuming the electric conductivity σ to be independent from the temperature, once we have solved the eddy current problem, we can compute the temperature by defining the heat source as the Joule's dissipated heat. For a time harmonic analysis, this heat source is given by time averaging the real parts of current density $j_\phi e^{i\omega t}$ and electric field $E_\phi e^{i\omega t}$ over a harmonic cycle $T = 2\pi\omega^{-1}$. By considering that $T^{-1}\int_0^T \cos(\omega t)\sin(\omega t)\mathrm{d}t = 0$ and $T^{-1}\int_0^T \cos(\omega t)^2\mathrm{d}t = T^{-1}\int_0^T \sin(\omega t)^2\mathrm{d}t = 1/2$, we obtain

$$\begin{aligned} q &= T^{-1}\int_0^T \Re(j_\phi e^{i\omega t})\cdot\Re(E_\phi e^{i\omega t})\,\mathrm{d}t \\ &= T^{-1}\int_0^T (\Re j_\phi\cos(\omega t) - \Im j_\phi \sin(\omega t))\cdot(\Re E_\phi\cos(\omega t) - \Im E_\phi\sin(\omega t))\mathrm{d}t \\ &= (\Re j_\phi\cdot\Re E_\phi + \Im j_\phi\cdot\Im E_\phi)/2 = \langle j_\phi, E_\phi\rangle\,, \end{aligned}$$

where as last we have used the definition $\langle a, b\rangle = (\Re a\Re b + \Im a\Im b)/2$. As one would expect, the total dissipated heat in the wires and in the steel is the nothing else than the supplied AC power $W = \int_{\Omega_{3D}} q\mathrm{d}V = 2\pi\int_\Omega q\rho\,\mathrm{d}A \stackrel{!}{=} \langle I, V\rangle$ with $V = \sum_{i=1}^{N} V_i$. In fact, the solution of the eddy problem $\nabla\times\mu^{-1}\nabla\times\mathbf{A} = \mathbf{j}$ with Coulomb gauge $\nabla.\mathbf{A} = 0$ and rotational symmetry implies solving the scalar equation $\nabla_2\cdot\rho^{-1}\mu^{-1}\nabla_2\rho A_\phi = -j_\phi$ with $\nabla_2 = (\partial_\rho, \partial_z)$, so that for the induced electric field E_ϕ^{ind} we have

$$\begin{aligned} W = \quad & 2\pi\int_\Omega \langle j_\phi, E_\phi^{\mathrm{ind}}\rangle\rho\,\mathrm{d}A \\ = \quad & -2\pi\omega\int_\Omega \langle\nabla_2\cdot\rho^{-1}\mu^{-1}\nabla_2\rho A_\phi, iA_\phi\rangle\rho\,\mathrm{d}A \\ = \quad & -2\pi\omega\int_\Omega[(\nabla_2\cdot\rho^{-1}\mu^{-1}\nabla_2\rho\Re A_\phi)(\rho\Im A_\phi) - (\nabla_2\cdot\rho^{-1}\mu^{-1}\nabla_2\rho\Im A_\phi)(\rho\Re A_\phi)]\mathrm{d}A \\ = \quad & -2\pi\omega\int_{\partial\Omega}\mu^{-1}[(\nabla_2\rho\Re A_\phi)(\Im A_\phi) - (\nabla_2\rho\Im A_\phi)(\Re A_\phi)]\cdot\mathrm{d}\mathbf{n} \\ & +2\pi\omega\int_\Omega\rho^{-1}\mu^{-1}[(\nabla_2\rho\Re A_\phi)(\nabla_2\rho\Im A_\phi) - (\nabla_2\rho\Im A_\phi)(\nabla_2\rho\Re A_\phi)]\mathrm{d}A \\ = \quad & -2\pi\omega\int_{\partial\Omega}\mu^{-1}[(\nabla_2\rho\Re A_\phi)(\Im A_\phi) - (\nabla_2\rho\Im A_\phi)(\Re A_\phi)]\cdot\mathrm{d}\mathbf{n} = 0\,, \end{aligned}$$

since the last boundary integral is zero by the symmetric choice of boundary conditions. Hence for the total dissipation, we obtain

$$\begin{aligned} W = \quad & \int_\Omega\langle j_\phi, E_\phi\rangle(2\pi\rho)\mathrm{d}A \\ = \quad & \int_\Omega\langle j_\phi, (E_\phi^0 + E_\phi^{\mathrm{ind}})\rangle(2\pi\rho)\mathrm{d}A \\ = \quad & \int_\Omega\langle j_\phi, (E_\phi^0 2\pi\rho)\rangle\mathrm{d}A \\ = \quad & \sum_{i=1}^{N}\int_{\Omega_i}\langle j_\phi, V_i\rangle\mathrm{d}A = \sum_{i=1}^{N}\langle I, V_i\rangle = \langle I, V\rangle\,. \end{aligned}$$

A steady thermal problem is solved to compute the temperature of the steel. The temperature of the surrounding air and wires is not considered, instead at the interface between steel and air we use the linear mixed boundary condition $\mathbf{F}\cdot\mathbf{n} = \alpha(T - T_{\mathrm{air}})$ with $\mathbf{F}$ the thermal flux and $\mathbf{n}$ the normal to the boundary. The coefficient α includes here thermal radiation and heat transfer to the surrounding air with an ambient temperature of T_{air}. This mixed BC is enough to stabilize the thermal problem thus making it unique solvable.

This example is found at `example/Hardening.s2d`. The computational mesh has been constructed with the built-in mesh generator and is shown in Fig. 2.46. Due to the

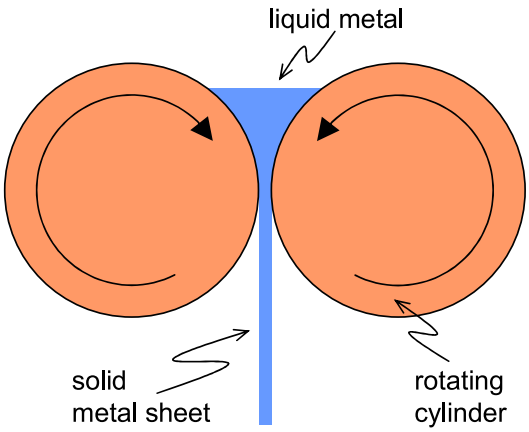

Figure 2.48: Drawing of the metal sheet fabrication.

Figure 2.49: $40\,\mathrm{kHz}$ alternating current of $50\,\mathrm{A}$ in the four copper turns give rise to a ditch in the liquid metal. The metal ditch radius is $0.5\,\mathrm{cm}$ deep, to be compared with (2.37).

rotational symmetry, the left edge of the domain must be located at $x = 0$. Further, it is important to correctly resolve the boundary layer caused by the skin effect as shown in Fig. 2.47 for the dissipated heat. The problem is defined in a way that given the coil current $I \in \mathbb{C}$, we solve the consistent eddy problem to compute the coil voltage $V \in \mathbb{C}$ which is followed by the thermal computation. On the output, as show below,

```
CURRENT-WIRE0=(3.000000e+02,1.439329e-11) A
CURRENT-WIRE1=(3.000000e+02,2.026934e-11) A
CURRENT-WIRE2=(3.000000e+02,2.239964e-11) A
CURRENT-WIRE3=(3.000000e+02,2.508244e-12) A
TOT-DISSP-HEAT-0=7.8354381554594045e+02 W
TOT-DISSP-HEAT-1=7.8354381554478482e+02 W
```

we check the consistency condition for the current I on each wire and that the computation of the total dissipated heat can indeed be computed either by the relation $W = 2\pi \int_\Omega q\rho \mathrm{d}A$ or $W = \langle I, V \rangle$ which are equivalent to machine accuracy.

2.11 Eddy Current Repulsion

The fabrication of thin metal sheets by solidification of the liquid metal between rotating cylinders has not attained industrial production yet. Fig. 2.48 shows the idea of the sheet fabrication. For a successful industrial production, several technical problems have to be surmounted. One such problem is the confinement of the liquid metal at both ends of the cylinders. Eddy current forces have been suggested to prevent the leak of the liquid metal at the end of the cylinders. Experimental studies indicate

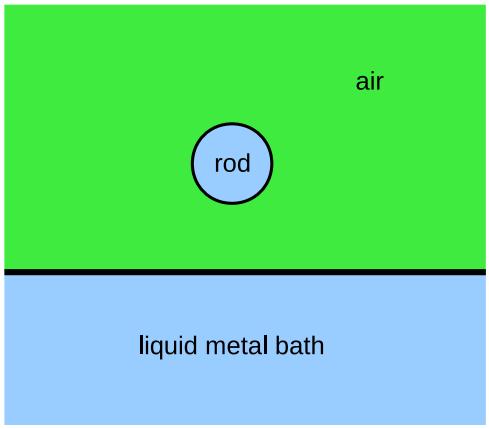

Figure 2.50: Two dimensional model of the experimental situation.

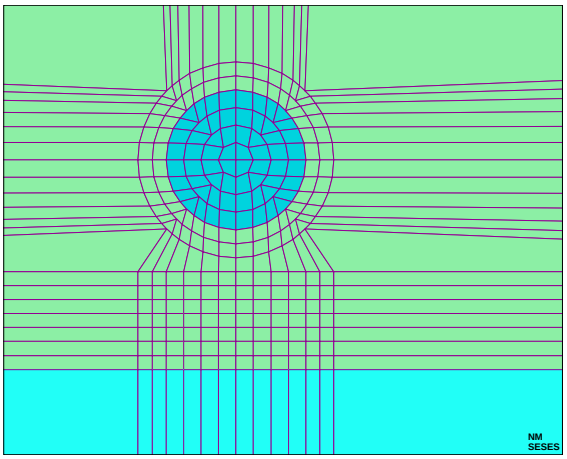

Figure 2.51: Finite element mesh for the circular rod and surrounding.

large repulsion forces between a liquid metal bath and a copper rod subject to alternating current. Fig. 2.49 demonstrates the electromagnetic repulsion between liquid bismuth-tin and a copper rod carrying $40\,\mathrm{kHz}$ AC current. The ditch in the liquid is about $0.5\,\mathrm{cm}$ deep. For a half cylinder shaped ditch in the liquid with the dimensions $r = 0.5\,\mathrm{cm}$ and $\ell = 1\,\mathrm{m}$ and the density $\rho = 8700\,\mathrm{kg/m^3}$ we calculate after Archimedes principle a buoyancy force of

$$F_{\mathrm{buoyancy}} = \rho\,(\frac{1}{2}r^2\pi\ell)\,9.81\,\frac{\mathrm{m}}{\mathrm{s}^2} = 3.4\,\mathrm{N}\,. \tag{2.37}$$

To expand the experimental data to different excitation frequencies and currents, the following model calculations have been performed. The result of (2.37) will serve for validation of the computations. The repulsion principle and its *SESES* model could then be applied to design the turns of a coil, which prevents the leak of the liquid metal at the end of the rotating cylinders in Fig. 2.48.

Implementation of the Model

A two dimensional model is used to describe the experimental situation of Fig. 2.50. The input file `example/Repulsion.s2d` is divided in four parts. The first part comprises the definition of parameters and dimensions, as well as the material properties. In particular, the magnitude of the electrical field in the rod along the z-axis is specified with the copper properties. The electrical field is given as the quotient of the maximum DC current, that copper can stand and the DC conductivity of copper. To vary the total current, the value of the electrical field can be increased by a factor `CurrCorr`. In order to resolve details of the conductor and its close surrounding, the mesh has been additionally refined in the central region via the parameter `nn`. Note the distinction between logic coordinates (integer numbers) and absolute coordinates (real dimensions). This distinction is important for the geometrical transformation in part two. Part two defines the mesh and the geometry, as well as the minimum refinement level for calculation. The homotopic transformation of the rectangular geometry

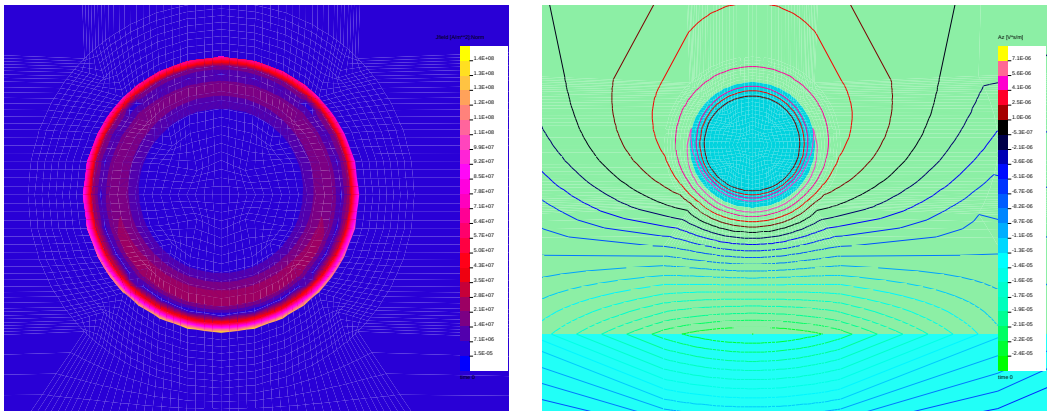

Figure 2.52: Left: skin effect at $40\,\mathrm{kHz}$ AC excitation. Only the outer shell of the copper rod carries current. Right: magnetic field lines in the vicinity of the copper rod at $40\,\mathrm{kHz}$ alternating electrical field.

of the copper rod and its close surrounding to a spherical geometry is optional. The effect of the statements

```
Include "Homotopic.sfc"
Coord sphere(coord,(ax2+ax3)/2,(ay2+ay3)/2,1) block(nx1 nx4 ny1 ny4)
```

is shown in Fig. 2.51. In part three, the space is filled with material and the boundary conditions are defined. The vector potential $\mathbf{A}$ is chosen to be zero at the domain's boundary. This implies that no magnetic field lines cross the boundary. Finally part four is the command section and performs the FEM computations and prepares the graphical visualization of the results. The total force on the rod as well as the total electrical current in the rod are computed as integral of output fields and displayed.

Discussion and Validation of the Data

The left of Fig. 2.52 shows the skin effect of $40\,\mathrm{kHz}$ alternating current through the rod. A total current of $50\,\mathrm{A}$ yields the magnetic field lines in the right of Fig. 2.52. *SESES* calculates an induced electromagnetic force of $\approx 1.28\,\mathrm{N}$ on the rod. This value is smaller than the estimated buoyancy force of $3.4\,\mathrm{N}$ given by (2.37). However nor the exact current through the rod, nor the detailed shape and dimensions of the ditch in the liquid metal are well known. The difference between the computed and estimated repulsion force is therefore not surprising. Fig. 2.53 plots the exponential current decay as a function of the rod penetration depth. The penetration length has been calculated with (2.38). *SESES* also calculates an exponential current decay, however with a superimposed oscillation, see Fig. 2.53. The origin of this oscillation is not yet clear. For sure it is not influenced by mesh refinement.

$$d(\omega) = \sqrt{\frac{2}{\mu_0\,\sigma\,\omega}} = \sqrt{\frac{2}{1.2566{\times}10^{-6}\cdot 5.7{\times}10^{7}\cdot 40{\times}10^{3}\cdot 2\pi}} = 0.33\,\mathrm{mm}\,. \qquad (2.38)$$

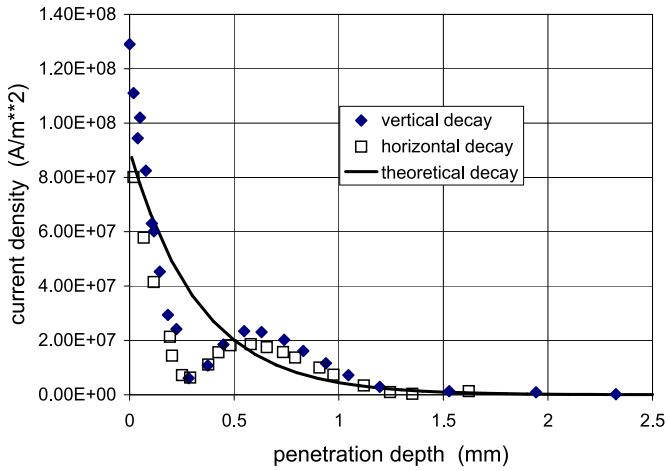

Figure 2.53: Skin effect for $40\,\mathrm{kHz}$ AC current in the copper rod is plotted as a function of the penetration depth. The computed data is compared with the analytical expression of the skin effect.

2.12 The total and reduced formulation of magnetostatic

In electrical machineries as well as in many sensors and micro devices one is often interested in computing magnetostatic forces caused by the flowing of time-constant electrical currents $\mathbf{J}_0$ which by the Biot-Savart's law generate an induction field $\mathbf{B}$ and by Lorentz's law forces on moving charged particles. The starting point for computing magnetostatic forces is therefore the knowledge of the magnetic induction $\mathbf{B}$. For efficiency reasons, this magnetostatic problem is generally solved numerically by computing either a total or a reduced magnetostatic potential Θ determining the $\mathbf{B}$-field by differentiation. The most generic formulation is based on computing the reduced potential Θ_{red}, however, it may be that the problem to be solved is numerically unstable and strongly affected by numerical cancellation so that one has to resort to a formulation combining together both the reduced and total magnetic potential. It is exactly this combination of potentials which makes the problem a little bit involved and this contribution aims to clarify this situation. We start with the formulation of the magnetostatic problem characterized by solving the subset of the Maxwell equations

$$\nabla.\mathbf{B} = 0, \quad \nabla \times \mathbf{H} = \mathbf{J}_0\,, \tag{2.39}$$

for the induction field $\mathbf{B}$, the magnetic field $\mathbf{H}$ and for a given external time-constant current $\mathbf{J}_0$ having the property $\nabla.\mathbf{J}_0 = 0$. Induction and magnetic fields are assumed to be related by the constitutive relationship $\mathbf{B} = \mu\mathbf{H}$ with μ the permeability. In order to have a numerically attractive formulation, we would like to solve this problem by computing a single scalar potential, which however, requires an irrotational i.e. curl-free governing equation which is not yet the case for (2.39). For the moment, lets assume we are working in empty space $\Omega = \mathbb{R}^3$ with $\mu = \mu_0 = 4\pi \times 10^{-7}\,\mathrm{Vs/m}$. Then the solution for the $\mathbf{H}$-field in (2.39) is well known and given by the Biot-Savart integral

$$\mathbf{H}_{\mathrm{BS}}(\mathbf{x}) = \frac{1}{4\pi}\int_{\mathbb{R}^3} \frac{\mathbf{J}_0(\mathbf{y}) \times (\mathbf{x}-\mathbf{y})}{|\mathbf{x}-\mathbf{y}|^3}\mathrm{d}V_y\,. \tag{2.40}$$

The major idea behind the magnetostatic reduced formulation is to split the $\mathbf{H}$-field in two parts $\mathbf{H} = \mathbf{H}_0 + \mathbf{H}_1$, one consisting of the known Biot-Savart integral $\mathbf{H}_0 = \mathbf{H}_{\rm BS}$ having the property $\nabla \times \mathbf{H}_0 = \mathbf{J}_0$ and the irrotational part $\mathbf{H}_1$ to be computed by a scalar formulation and taking into account material and bounded domains effects. In fact, for the $\mathbf{H}_1$-field we now have $\nabla \times \mathbf{H}_1 = 0$ so that a reduced magnetic scalar potential $\Theta_{\rm red}$ can be defined with $\mathbf{H}_1 = -\nabla\Theta_{\rm red}$ and insertion into $\nabla.\mathbf{B}$ yields a Laplace equation

$$\nabla.(\mu\nabla\Theta_{\rm red} - \mu\mathbf{H}_0) = 0\,, \tag{2.41}$$

to be solves for $\Theta_{\rm red}$ on a domain Ω. The term *reduced formulation* originates from the fact that the magnetic potential $\Theta_{\rm red}$ only determines the irrotational part $\mathbf{H}_1 = -\nabla\Theta_{\rm red}$ of the $\mathbf{H}$-field and a *total formulation* is present whenever the magnetic potential fully determine the magnetic field $\mathbf{H} = -\nabla\Theta_{\rm tot}$ which automatically implies $\mathbf{J}_0 = 0$ within Ω. In this latter case, the excitation of external currents must be included within the BCs on $\partial\Omega$ when solving for $\Theta_{\rm tot}$. A total formulation can always be turned into a reduced one by defining $\mathbf{H}_0$ as the Biot-Savart field of the current $\mathbf{J}_0$ external to the domain Ω thus eventually simplifying the setting of BCs.

Let us get acquainted a little bit with both the reduced and total formulations and for a case with $\mathbf{J}_0 = 0$ let present an analytical solution. The problem we are going to investigate consists of a ball of radius r_b and permeability $\mu = \mu_0\mu_{\rm rel}$ immersed in a constant magnetic field of $\mathbf{H}_0 = \mathbf{e}_z \times 1\,{\rm A/m}$ generated by external currents. The solution for either the total $\tilde\Theta_{\rm tot}$ or reduced $\tilde\Theta_{\rm red}$ potential is given by

$$\tilde\Theta_{\rm red} = \begin{cases} \dfrac{\mu_{\rm rel}-1}{\mu_{\rm rel}+2}\, z\, r_b^3 |\mathbf{x}|^{-3} & \text{for} \quad |\mathbf{x}| \geq r_b \\ \dfrac{\mu_{\rm rel}-1}{\mu_{\rm rel}+2}\, z & \text{for} \quad |\mathbf{x}| \leq r_b \end{cases} \quad \text{and } \tilde\Theta_{\rm tot} = \tilde\Theta_{\rm red} - z\,. \tag{2.42}$$

Since $\nabla.\nabla z = 0$ and $\nabla.\nabla z|\mathbf{x}|^{-3} = 0$, we have $\nabla.\nabla\tilde\Theta_{\rm tot} = 0$ and $\tilde\Theta_{\rm tot}$ is indeed a magnetostatic solution if the normal component of $\mathbf{B}$ and the tangential component of $\mathbf{H}$ are continuous on the sphere $|\mathbf{x}| = r_b$. Using spherical coordinates with $|\mathbf{x}| = \rho$ and $z = \rho\cos(\theta)$, for a generic function $g = f(\rho)\cos(\theta)$ we have $\nabla.g = \partial_\rho f(\rho)\cos(\theta)\mathbf{e}_\rho + f(\rho)\rho^{-1}\sin(\theta)\mathbf{e}_\theta$. With $\mathbf{H} = -\nabla\tilde\Theta_{\rm tot}$, we see that the tangential component $\mathbf{H}\times\mathbf{n} = \mathbf{H}\cdot\mathbf{e}_\theta$ is continuous if $\tilde\Theta_{\rm tot}$ is continuous and the normal component $\mathbf{B}\cdot\mathbf{n} = \mathbf{B}\cdot\mathbf{e}_\rho$ is continuous if $\mu\partial_\rho\tilde\Theta_{\rm tot}$ is continuous which is indeed the case. Since within the ball $|\mathbf{x}| \leq r_b$ both the $\mathbf{B}$- and $\mathbf{H}$-fields are constant, even for a non-linear permeability $\mu_{\rm rel} = \mu_{\rm rel}(|\mathbf{H}|)$ is $\tilde\Theta_{\rm tot}$ a solution provided $|\mathbf{H}|$ is a solution of equation $|\mathbf{H}| = (\mu_{\rm rel}(|\mathbf{H}|) - 1)/(\mu_{\rm rel}(|\mathbf{H}|) + 2)$.

This solution is rotational symmetric and the accompagning *SESES* example can be found at `example/TotRedFormul.s2d`. In order to compare the numerical solution with the exact analytical one, we use exact BCs. For the total formulation, we just set a Dirichlet value of $\tilde\Theta_{\rm tot}$ on $\partial\Omega$ and for the reduced formulation, we set the Dirichlet value $\tilde\Theta_{\rm red}$ and define the $\mathbf{H}_0$-field as $\mathbf{e}_z \times 1\,{\rm A/m}$ by specifying the material parameter `Hfield0`. For moderately values of $\mu_{\rm rel} \geq 1$ both formulations yields the same results and convergence with respect to mesh refinement is according to the theory i.e. for first and 2nd order when using $H^1(\Omega)$ conformal elements selected with the equation statement `MagnetoStatic` or `MagnetoStatic2`.

Even for very large values of $\mu_{\rm rel}$, the total formulation is very stable, however, not the reduced one. The problem lies in the cancellation of the terms $\mu\nabla\Theta_{\rm red} - \mu\mathbf{H}_0$ within

the divergence operator of (2.41) and this shortcoming will be even more pronounced for non-linear problems with $\mu = \mu(|\mathbf{H}|)$. For computing this analytical solution, the obvious result is to use the total formulation, however, in practice seldomly we have $\mathbf{J}_0 = 0$ on all Ω and a reduced formulation is therefore required. The solution to this dilemma is given by combining both the reduced and total formulations and by noting that in empty space and generally also in conductors we have $\mu_{\rm rel} = 1$ whereas for the magnetic materials with $\mu_{\rm rel} \gg 1$, we generally have $\mathbf{J}_0 = 0$. However, the reduced formulation with $\mu_{\rm rel} = 1$ is not at all critical and yields stable numerical results so that we can continue to use it, just within the magnetic material we have to use the total formulation thus avoiding the before mentioned problem related to numerical cancellation. In order to proceed, we split our domain in two parts $\Omega = \Omega_{\rm red} \cup \Omega_{\rm tot}$, assume $\mathbf{J}_0 = 0$ on $\Omega_{\rm tot}$ and first compute the reduced magnetic potential $\Theta_{\rm red}$ in all Ω. Within $\Omega_{\rm tot}$ and because $\mathbf{J}_0 = 0$, the $\mathbf{H}_0$-field is a gradient field and so locally there exists a potential field Θ_0 with $\mathbf{H}_0 = -\nabla\Theta_0$. If we additionally assume the domain $\Omega_{\rm tot}$ to be simply connected or has simply connected components $\Omega^i_{\rm tot}$ with $\Omega_{\rm tot} = \bigcup_i \Omega^i_{\rm tot}$, we can write

$$\Theta_0(\mathbf{x}) = \int_{\boldsymbol{\gamma}(\mathbf{x}\rightsquigarrow\mathbf{x_i})} \mathbf{H}_0 \cdot \mathrm{d}\boldsymbol{\gamma}\,, \tag{2.43}$$

with $\mathbf{x}_i, \mathbf{x} \in \Omega^i_{\rm tot}$ and $\boldsymbol{\gamma}(\mathbf{x} \rightsquigarrow \mathbf{x}_i)$ any path on $\Omega^i_{\rm tot}$ joining $\mathbf{x}$ to the fixed point $\mathbf{x_i}$. In summary, we can write

$$\begin{aligned} \nabla.(\mu\nabla\Theta_{\rm red} - \mu\mathbf{H}_0) = \nabla.(\mu\nabla\Theta_{\rm red} + \mu\nabla\Theta_0) = \nabla.\mu\nabla\Theta_{\rm tot} = 0 \quad &\text{for} \quad \mathbf{x} \in \Omega_{\rm tot}\,, \\ \nabla.(\mu\nabla\Theta_{\rm red} - \mu\mathbf{H}_0) = 0 \quad &\text{for} \quad \mathbf{x} \in \Omega_{\rm red}\,, \end{aligned}$$

and since the $\mathbf{H}_0$-field is generally the Biot-Savart field of (2.40), the potential Θ_0 is also a well-known function of the problem. Now let us drop the subscripts $\rm red$ and $\rm tot$ on the magnetic potential Θ, then we have to solve the equations

$$\begin{aligned} \nabla.\mu\nabla\Theta = 0 \quad &\text{for} \quad \mathbf{x} \in \Omega_{\rm tot}\,, \\ \nabla.(\mu\nabla\Theta - \mu\mathbf{H}_0) = 0 \quad &\text{for} \quad \mathbf{x} \in \Omega_{\rm red}\,, \end{aligned}$$

which are equivalent to the previous ones if at the interface boundary $\partial\Omega_{\rm inter} = \overline{\Omega}_{\rm red} \cap \overline{\Omega}_{\rm tot}$, we assume the magnetic potential Θ is a double valued function and undergoes a jump of Θ_0 expressing the difference between a total and reduced formulation. The point here is that from a numerical point of view is quite simple to apply a well defined jump on an internal boundary when computing a scalar potential and in doing so we can easily combine the total and reduced magnetostatic formulation. For our analytical example, we have $\Theta_0 = -z$ and this value can be used as a jump value on the sphere $|\mathbf{x}| = r_b$ if one want to solve for the total formulation within the ball and a reduced one outside. When defining a jump BC, we can decide on which side of the jump the dof-values are defined, so that on the other side their values are obtained by adding the jump value. Since within the sphere the magnetic potential tends to be flat, it is better to define here the dofs which in our example is done by

```
BC jump Restrict material(Sphere) OnChangeOf 3 4*material(Sphere)+material(Air)
```

In a general case, the value of the jump Θ_0 to be prescribed at the interface boundary $\partial\Omega_{\rm inter}$ is not known analytically but must be evaluated numerically. We can directly

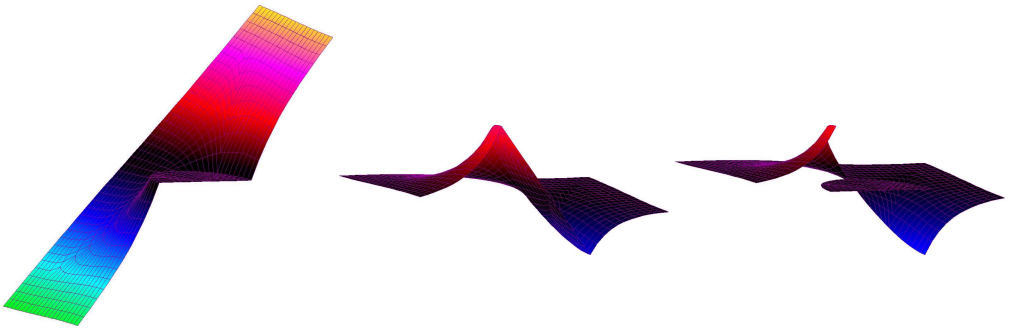

Figure 2.54: The magnetic potential for the analytic solution of (2.42) computed with the total, reduced and total-reduced formulation.

arrive at an expression for Θ_0 by inserting (2.40) into (2.43) and thus obtaining an integral expression for this Biot-Savart potential as a function of the external current $\mathbf{J}_0$. This is indeed possible and the resulting expression is given in the *SESES* manual. However, its numerical evaluation is rather delicate and error prone and therefore its usage is discouraged. A better approach is obtained by considering that the value of the jump Θ_0 on $\partial\Omega_{\text{inter}}$ is a potential field on Ω_{tot} satisfying the Laplace equation $\nabla.\mathbf{H}_0 = -\nabla.\nabla\Theta_0 = 0$ and having the Neumann BCs $\mathbf{H}_0 \cdot \mathbf{n}$ on $\partial\Omega_{\text{inter}}$. Fixing the value of Θ_0 on a single point of each simply connected component of Ω_{tot} makes the solution of this problem unique. Therefore in practice the solution of the magnetostatic problem with a reduced and total formulation is generally preceeded by the computation of the jump Θ_0 by solving a linear Laplace problem. The solution of Θ_0 at $\partial\Omega_{\text{inter}}$ is then stored in memory with the help of a boundary user field `jumppot` and used as a jump value when solving for the reduced-total formulation which is done in our example by

```
BC jump Neumann   MagnPot  Normal.Y*MUE0*mue T
BC fix  Dirichlet MagnPot  0 A
Solve Init
Solve Stationary
Store jumppot=MagnPot
BC jump Jump      MagnPot -jumppot  A
```

All these four approaches, i.e. total, reduced, total-reduced with Biot-Savart potential as jump and total-reduced with jump computation by a Laplace problem are computed by the given *SESES* example. The magnetic potential solution is displayed in Fig. (2.54) and the example reports the maximal numerical error of the solution. The magnetic potential is computed accurately by all formulations, however, for values of $\mu \geq 10^{15}$, the reduced formulation wrongly computes the $\mathbf{H}$- and $\mathbf{B}$-field within the ball.

```
Rel. Perm. 1.000000e+25
Total     MagnPot-error 5.581309e-02 Bfield-error 2.474931e-01 3.233611e+00
Reduced   MagnPot-error 8.527405e-01 Bfield-error 1.081236e+11 9.095459e+10
Tot-Red-0 MagnPot-error 5.581309e-02 Bfield-error 2.474931e-01 3.233611e+00
Tot-Red-1 MagnPot-error 6.601821e-01 Bfield-error 2.805619e+00 5.698386e+00
```

One can argue that so large permeability values never show up in practice and indeed this total-reduced formulation is generally not required in the case of a large but constant permeability. However, this may be a necessity whenever we have a non-linear material law $\mu = \mu(|\mathbf{H}|)$.

For large problems, even when the permeability μ does not cause numerical instabilities, for practical reasons one also resorts to this reduced-total formulation. The reason is that for a given current $\mathbf{J}_0$, the computation Biot-Savart integral (2.40) has to be performed at each integration point of each element belonging to the meshing of Ω and this is a very CPU intensive computation which is better avoided. And indeed this is possible if one chooses $\Omega_{\rm red}$ as the minimal domain containing the support of $\mathbf{J}_0$ and so that $\Omega_{\rm tot} = \Omega/\Omega_{\rm red}$ is simply connected. Then $\mathbf{H}_0$ and thus the Biot-Savart integral must only be evaluated within $\Omega_{\rm red}$ and on $\partial\Omega_{\rm inter}$.

2.13 The scalar and vector formulation of magnetostatic

The magnetostatic solution of Maxwell's equation is an important topic in the design and development of electrical machineries driven by magnetostatic forces. Basically this problem formulation computes on a domain $\Omega \subset \mathbb{R}^3$ the magnetic induction $\mathbf{B}$ and the magnetic field $\mathbf{H}$ starting from a given time-constant current $\mathbf{J}_0$ having the property $\nabla.\mathbf{J}_0 = 0$ and a given permeability μ specifying the material law $\mathbf{B} = \mu\mathbf{H}$. It is interesting to know, that this static model may also be used to find low frequency approximations of the computational more expensive time-dependent eddy current model and therefore good optimized solution algorithms are of large interest. The magnetostatic model can either be solved by computing a scalar or a vector potential, both determining the magnetic field by differentiation and both having their drawbacks and advantages. However, a considerable increase in flexibility is first obtained by combining both formulations thus allowing to devise better optimized solution algorithms. Here, the price to pay is a more complex problem specification and this contribution aims at explaining the different available choices. The scalar formulation is generally the preferred one, but has the drawback that one cannot directly specify the current source $\mathbf{J}_0$. Instead, one has first to define the associated Biot-Savart field

$$\mathbf{H}_{\rm BS}(\mathbf{x}) = \frac{1}{4\pi}\int_{\mathbb{R}^3} \frac{\mathbf{J}_0(\mathbf{y}) \times (\mathbf{x}-\mathbf{y})}{|\mathbf{x}-\mathbf{y}|^3} {\rm d}V_y \,, \tag{2.44}$$

and then solve the equation

$$\nabla.(\mu\nabla\Theta_{\rm red} - \mu\mathbf{H}_{\rm BS}) = 0 \,, \tag{2.45}$$

for the reduced scalar potential $\Theta_{\rm red}$ with $\mathbf{B} = \mu(\mathbf{H}_{\rm BS} - \nabla\Theta_{\rm red})$. Rare are the circumstances when the Biot-Savart field $\mathbf{H}_{\rm BS}$ is known analytically and so one has to evaluate the integral (2.44) numerically which may be critical and imprecise for $\mathbf{x} \in {\rm supp}(\mathbf{J}_0)$. Also this integral over the volume of ${\rm supp}(\mathbf{J}_0)$ has to be evaluated on many points of Ω, generally taken as the element integration points of the mesh, which therefore can be computationally quite expensive. Another drawback of the scalar formulation is its instability for non-linear material laws $\mu = \mu(|\mathbf{H}|)$ stemming from the subtraction of the term $\mu\mathbf{H}_{\rm BS}$ in (2.45), forcing the user to combine a total and reduced potential formulation. These drawbacks are partially eliminated by the vector formulation solving the equation

$$\nabla \times \mu^{-1}\nabla \times \mathbf{A} = \mathbf{J}_0 + \nabla \times \mathbf{H}_0 \,, \tag{2.46}$$

for the vector potential $\mathbf{A}$ with $\mathbf{B} = \nabla \times \mathbf{A}$. The additional numerical work required to compute the vector field $\mathbf{A}$ instead of a scalar field $\Theta_{\rm red}$ is compensated by the fact that one has more flexibility and one can directly specify the current $\mathbf{J}_0$. This requires an element mesh fine enough to have a good approximation of the current $\mathbf{J}_0$ within Ω, but the previous Biot-Savart approach by defining $\mathbf{J}_0 = 0$ and $\mathbf{H}_0 = \mathbf{H}_{\rm BS}$ is still available. This vector formulation is also the one required when computing approximations of the time-dependent eddy-current model, since it requires the $\mathbf{A}$-field on $\mathrm{supp}(\mathbf{J}_0)$.

Mixing the scalar and vector formulation

By choosing a domain $\Omega_{\rm vec}$ with $\mathrm{supp}(\mathbf{J}_0) \subset \Omega_{\rm vec}$ and assuming $\Omega_{\rm vec} \subset\subset \Omega$ then on $\Omega_{\rm free} = (\Omega/\Omega_{\rm vec})^\circ$ where the current is vanishing $\mathbf{J}_0 = 0$, we do not necessarily have to compute the Biot-Savart field $\mathbf{H}_{\rm BS}$. In fact, the reduced formulation can be replaced by computing the total scalar potential $\Theta_{\rm tot}$ with $\mathbf{H} = -\nabla\Theta_{\rm tot}$ and by solving the Laplace equation

$$\nabla.\mu\nabla\Theta_{\rm tot} = 0\,. \tag{2.47}$$

Both the reduced formulation in $\Omega_{\rm vec}$ and total formulation in $\Omega_{\rm free}$ can be coupled and solved together by defining proper interface BCs on $\partial\Omega_{\rm vec}$. Here however our major intent is to get rid of the Biot-Savart field and therefore on $\Omega_{\rm vec}$ we do not solve for the reduced potential $\Theta_{\rm red}$ but for the vector potential $\mathbf{A}$. In doing this several questions arise.

- We need to define interface BCs on $\partial\Omega_{\rm vec}$ connecting the scalar formulation on $\Omega_{\rm free}$ and the vector formulation on $\Omega_{\rm vec}$.
- Since the domain $\Omega_{\rm free}$ is generally not simply connected, the potential $\Theta_{\rm tot}$ is generally not a function but a multi-valued function.
- What about uniqueness and accuracy of the coupled solution ?

The answer to the first question is obtained from the Maxwell's equations requiring the normal component of $\mathbf{B}$ and the tangential component of $\mathbf{H}$ to be continuous. Let $\mathbf{n}$ be the normal to the boundary $\partial\Omega_{\rm vec}$ between the current domain $\Omega_{\rm vec}$ with $\mathbf{B} = \nabla \times \mathbf{A}$ and $\mathbf{H} = \mu^{-1}\nabla \times \mathbf{A}$ and the free space domain $\Omega_{\rm free}$ with $\mathbf{B} = \mu\mathbf{H}$ and $\mathbf{H} = \mathbf{H}_0 - \nabla\Theta_{\rm tot}$. The equality of the normal component $\mathbf{B}\cdot\mathbf{n}$ of $\mathbf{B}$ gives $\mu(\mathbf{H}_0 - \nabla\Theta_{\rm tot})\cdot\mathbf{n} = (\nabla\times\mathbf{A})\cdot\mathbf{n}$ and the one of the tangential component $\mathbf{H}\times\mathbf{n}$ of $\mathbf{H}$ gives $(\mathbf{H}_0 - \nabla\Theta_{\rm tot})\times\mathbf{n} = (\mu^{-1}\nabla\times\mathbf{A})\times\mathbf{n}$ or $(\mu^{-1}\nabla\times\mathbf{A} - \mathbf{H}_0)\times\mathbf{n} = -\nabla\Theta_{\rm tot}\times\mathbf{n}$. Following the *SESES* manual, these conditions are satisfied by setting the following Neumann BCs

$$\begin{array}{ll} (\nabla\times\mathbf{A})\cdot\mathbf{n} & \text{for } \Theta_{\rm tot}\,, \\ -\nabla\Theta_{\rm tot}\times\mathbf{n} & \text{for } \mathbf{A}\,. \end{array} \tag{2.48}$$

In order to solve our linear problem with a single linear step, also the correct derivatives should be specified for these interface BCs.

It is interesting to note that both Neumann BCs together may result in a global symmetric contribution. Let $H_i(\mathbf{x})$ and $\mathbf{H}_i(\mathbf{x})$ be the shape functions associated with the

dof-fields Θ and $\mathbf{A}$, i.e. we have $\Theta = \sum_i H_i \Theta_i$ and $\mathbf{A} = \sum_i \mathbf{H}_i A_i$ with Θ_i and A_i the unknowns of the problem to be solved. The residual contributions of both Neumann BCs (2.48) are given by

$$\begin{aligned} R_i^{\mathbf{A}} &= -\int_{\partial\Omega_{\text{vec}}} \mathbf{H}_i \cdot (\nabla\Theta \times \mathbf{n})\,\mathrm{d}A \\ &= -\int_{\partial\Omega_{\text{vec}}} (\mathbf{H}_i \times \nabla\Theta) \cdot \mathbf{n}\,\mathrm{d}A \end{aligned}$$

and

$$\begin{aligned} R_i^{\mathbf{\Theta}} &= \int_{\partial\Omega_{\text{vec}}} H_i(\nabla \times \mathbf{A}) \cdot \mathbf{n}\,\mathrm{d}A \\ &= \int_{\partial\Omega_{\text{vec}}} [\nabla \times (H_i\mathbf{A}) - \nabla H_i \times \mathbf{A}] \cdot \mathbf{n}\,\mathrm{d}A \\ &\overset{\text{(Stoke)}}{=} \int_{\partial\partial\Omega_{\text{vec}}} (H_i\mathbf{A}) \cdot d\mathbf{s} - \int_{\partial\Omega_{\text{vec}}} \nabla H_i \times \mathbf{A} \cdot \mathbf{n}\,\mathrm{d}A \\ &= -\int_{\partial\Omega_{\text{vec}}} \nabla H_i \times \mathbf{A} \cdot \mathbf{n}\,\mathrm{d}A \end{aligned}$$

since the condition $\Omega_{\text{vec}} \subset\subset \Omega$ implies $\partial\partial\Omega_{\text{vec}} = \emptyset$. Therefore these two linear contributions are given by

$$R_i^{\mathbf{A}} = -\sum_j \Theta_j \int_{\partial\Omega_{\text{vec}}} (\mathbf{H}_i \times \nabla H_j) \cdot \mathbf{n}\,\mathrm{d}A \tag{2.49}$$

and

$$R_i^{\mathbf{\Theta}} = -\sum_j A_j \int_{\partial\Omega_{\text{vec}}} (\nabla H_i \times \mathbf{H}_j) \cdot \mathbf{n}\,\mathrm{d}A \tag{2.50}$$

and are seen to be anti-symmetric. This fact may be used to symmetrize the linear system, however, if iterative solvers are used, the convergence rate may be slower than solving non-symmetric matrices which, however, per default have positive semi-definite diagonal blocks.

The answer to the second question is two folds. The most simple one is to avoid at the beginning a non-simply connected domain by choosing an enlarged domain Ω_{vec} and thus getting closer to a full vector formulation with $\Omega_{\text{vec}} = \Omega$. The second one is given by selecting surface cuts C_i, $i = 1 \ldots N$ in Ω_{free} so that the domain $\Omega_{\text{simple}} = \Omega_{\text{free}}/\cup_i C_i$ is simply connected and therefore Θ_{tot} is a simple function on Ω_{simple}. On each cut C_i, one then defines a possible constant jump Δ_i for the potential Θ_{tot} determined as follow. For a point on C_i, let be γ_i a path in Ω_{simple} joining this point. By applying Stoke's theorem on any surface S_i with boundary $\partial S_i = \gamma_i$, we have

$$\int_{S_i} \mathbf{J}_0 \cdot \mathrm{d}\mathbf{n} = \int_{S_i} \nabla \times \mathbf{H} \cdot \mathrm{d}\mathbf{n} \overset{\text{(Stoke)}}{=} \int_{\gamma_i} \mathbf{H} \cdot \mathrm{d}\mathbf{l} = -\int_{\gamma_i} \nabla\Theta_{\text{tot}} \cdot \mathrm{d}\mathbf{l} = -\Delta_i\,,$$

and so the constant Δ_i is given by the total current crossing the surface $S_i \cap \Omega_{\text{vec}}$. Since $\nabla.\mathbf{J}_0 = 0$ in Ω and $\nabla \times \mathbf{H} = 0$ in Ω_{free}, we see that the jump value Δ_i is actually invariant with respect to the choice of C_i, γ_i and S_i. Further, although the potential has a jump, the physical field $\mathbf{H} = -\nabla\Theta_{\text{tot}}$ is well defined on Ω_{free}.

Since this example is not meant to be a mathematical treatise on existence, uniqueness and approximation error of solutions, the third question is answered by doing numerical experiments.

Preprocessing

Basically, we want to compute solutions without the need to evaluate the Biot-Savart integral (2.44) and therefore on Ω_{vec} we solve for the vector formulation (2.46) and the current $\mathbf{J}_0$ needs to be specified fulfilling the solenoidal condition $\nabla.\mathbf{J}_0 = 0$. For simple geometrical shapes, one may look for analytical expressions, but this approach is generally avoided for the following reasons. As first, the user needs to construct a FE mesh with good approximation properties with respect to $\mathbf{J}_0$ and at the end one does not really know which are the discretized current values. Secondly, nothing can be said about the discretized solenoidal property which generally does not hold anymore. Now since the mesh must be in anyway adapted to the geometrical shape of $\text{supp}(\mathbf{J}_0)$, on this mesh the preferred way is to compute the current numerically instead of specifying it analytically. The first reason is that this preprocessor step is generally much faster then the magnetostatic solution itself, since one just needs to solve the current flow model on the domain $\text{supp}(\mathbf{J}_0)$. Secondly, since the current is computed by a numerical model, the meaning of the property $\nabla.\mathbf{J}_0 = 0$ at the discrete level is well defined, although it only holds in a weak form. Therefore the problem specification starts by defining the domain $\text{supp}(\mathbf{J}_0)$ to compute the current defined by the material law $\mathbf{J}_0 = \sigma\mathbf{E}$ with σ the electric conductivity and $\mathbf{E}$ the static electric field. In the electrostatic, we have $\nabla \times \mathbf{E} = 0$ and so we can define the electric potential Ψ with $\mathbf{E} = -\nabla\Psi$ and from $\nabla.\mathbf{J}_0$ we are left to solve the Laplace equation $\nabla.(\sigma\nabla\Psi) = 0$ by choosing BCs for Ψ i.e. the applied voltage at the electrical contacts.

Model specification

When solving the vector formulation (2.46) in *SESES*, we have the choice of either using the nodal elements of $H(\Omega)^3$ or the edge elements of $H(\text{curl}, \Omega)$. Edge elements directly discretize the curl-curl operator of (2.46) which has a large null-space and so without an additional fixing reducing the number of redundant dofs, the discretized linear system is singular. In order to have solutions, the right-hand-side must be consistent and must lie in the image space of the linear map, a failure condition detected e.g. by the non-convergence of iterative linear solvers. For edge elements, this implies that the current computed by the preprocessing step must be first projected on a well-defined functional space and at the present this rather technical step must be done by the user. On the other side, the usage of nodal elements with the correct choice of BCs results in a regular linear system. These elements should not be used when on some regions the permeability is different from μ_0, since they enforce an unphysical continuity of $\mathbf{A}$. This will not be the case for our example and also we like the possibility to have unique solutions in order to work with a robust direct linear solver, an appreciated feature when doing numerical experiments on small sized problems. Therefore in this example, we are going to use nodal elements for the discretization of $\mathbf{A}$ and a direct solver, but we will notice the necessary changes when using an iterative solver.

The *SESES* input file for this example can be found at `example/ScalVecFormul.s3d`. The `Initial` section is divided in two parts, the first one defining the geometry. The user has to define the ME domain `VectorJ` representing $\text{supp}(\mathbf{J}_0)$ and where the electrical current $\mathbf{J}_0$ is computed by the electrostatic model. In addition, the user

defines the ME domain `Vector` used either to make the remaining domain Ω_{free} simply connected or as a cladding layer around $\text{supp}(\mathbf{J}_0)$. This domain is then added to `VectorJ` to form the domain Ω_{vec} where we solve for the vector formulation (2.46) with nodal elements. For large problems requiring an iterative solver, you should use edge elements instead. With the macro `NWIRE`, the user defines the number of closed wires and for each i-wire, $i = 1 \ldots$ `NWIRE` one needs to define four ME domains `CutLJ<i>`, `CutRJ<i>`, `CutLP<i>`, `CutRP<i>`. The intersection of the first two domains defines the wire's cut-section where we apply the driving electrostatic voltage and measure the current, the intersection of last two must defines the cut surface used to apply a possible magnetostatic potential jump. Finally the macro `bcGround` must specify a grounding point for each wire. If other current configurations are chosen, then just this geometry section needs to be changed.

In the second part of the `Initial` section, we define the three materials `VectorJ`, `Vector`, `Scalar`, we declare the element field `current0` to store the current computed in the preprocessing step and define the BCs. The material `VectorJ` is used to compute the electrostatic current $\mathbf{J}_0$ and has a non-zero conductivity of $\sigma = 5.8{\times}10^7$ $\text{A}/(\text{V} * \text{m})$. In the materials `VectorJ`, `Vector` we will use the vector formulation (2.46) and in `Scalar`, the scalar one (2.47). On the domain boundary $\partial\Omega$, we define both the scalar and vector potential to be zero. With the macro `bcGround`, we define all wire's grounding points and for each wire together with some logical ME operations, we define a one-sided boundary to apply a jump BC of $1\,\text{V}$ for the electrostatic potential Ψ. Again with logical ME operations, we define the two one-side boundaries of $\partial\Omega_{\text{vec}}$ used to define the interface Neumann BCs of (2.48). For the magnetostatic potential Θ, we have

```
Macro A_DOT_N__DA(A)
{
   ((@A[0]*Normal[0]+@A[1]*Normal[1]+@A[2]*Normal[2]),Normal)
}
BC bc1  Restrict material(Vector)+material(VectorJ)
        OnChangeOf 2 3*material(Scalar)+material(Vector)+material(VectorJ)
  Neumann MagnPot  D_Bfield SetAlways
  { return @A_DOT_N__DA( Bfield)*(-1); } SIunit Symmetric Linear HideFlux
```

The value of $\nabla \times \mathbf{A}$ is not available as a model parameter to specify the BC value but this is nothing else as $\mathbf{B}$ which is therefore used for the specification. This interface BC is a linear function of the computed primary field $\mathbf{A}$ and therefore it is important to correctly specify the field-derivative in order to get the solution in a single linear step. So for the numerical BC, we specify the value $\mathbf{B}{\cdot}\mathbf{n}$ together with the $\mathbf{B}$-field derivatives given by $\mathbf{n}$. The additional minus sign with respect to (2.48) has to do with a change of sign of the normal $\mathbf{n}$ and the following technicality. For a boundary surface defined inside the computational domain Ω, we have the choice on which side we want to evaluate the BC value which is done by defining a one-sided BC boundary. Since the vector potential $\mathbf{A}$ is not defined within the domain Ω_{free} of the scalar formulation, we are forced to define and evaluate the BC on the other side inside Ω_{vec}. However, here the magnetostatic potential Θ is undefined and the default behavior will not set this BC value expect if using the additional option `SetAlways`. Generic Neumann BCs with derivatives are considered as generic non-linear functions of the primary fields yielding non-symmetric matrices to be solved. The `Linear` option just asserts the linearity of the BC thus helping *SESES* to select the correct solution algorithm and the `Symmetric` option asserts the symmetry of the resulting matrix. From (2.49)-(2.50),

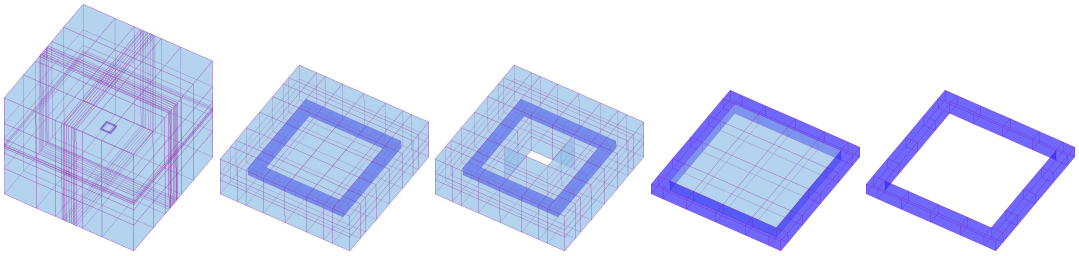

Figure 2.55: The computational domains Ω_{vec} solving for the vector potential $\mathbf{A}$.

we know that the overall contributions from the interface BCs are anti-symmetric, although the single element contributions are not. In order to have an overall symmetric system and thus less computational work, the following symmetrization is possible. The linear system arising from this scalar-vector formulation can be characterized by the following block structure

$$\begin{array}{c} \\ \mathbf{A} \\ \Theta \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{A} & \Theta \end{array} \\ \begin{pmatrix} A & D \\ -D^{\mathrm{T}} & B \end{pmatrix} \end{array}. \tag{2.51}$$

The diagonal blocks A, B are symmetric positive definite matrices stemming from the discretization of the vector and scalar potentials $\mathbf{A}$, Θ. The coupling between both formulations is uniquely defined by the D matrix stemming from the interface BCs (2.49)-(2.50). By changing the sign of the second block row, we obtain a symmetric matrix and this is accomplished with the following statement

```
GlobalSpec  Parameter MagnPotEqScaling -1
```

This symmetrization just works for a direct solver, but may lead to an unacceptable slow down of an iterative solver and therefore is not used per default and must to be enabled. In a similar way, we define the second interface Neumann BCs for $\mathbf{A}$

```
Macro A_CROSS_N__DA(A)
{
   ((@A[1]*Normal[2]-@A[2]*Normal[1]),          0, Normal[2],-Normal[1],
    (@A[2]*Normal[0]-@A[0]*Normal[2]),-Normal[2],          0, Normal[0],
    (@A[0]*Normal[1]-@A[1]*Normal[0]), Normal[1],-Normal[0],          0)
}
BC bc2 Restrict material(Scalar)
       OnChangeOf 2 3*material(Scalar)+material(Vector)+material(VectorJ)
  Neumann Afield  D_MagnPotGrad SetAlways
  @A_CROSS_N__DA(MagnPotGrad) SIunit Symmetric Linear HideFlux
```

The `Command` section solves the preprocessing step to compute the current $\mathbf{J}_0$ and various magnetostatic formulations to compute the magnetic induction $\mathbf{B}$.

Numerical results

In order to compare the numerical solutions, we use a large empty space $\mu = \mu_0$ as our domain Ω so that for given a current $\mathbf{J}_0$, the solution is given by the Biot-Savart integral (2.44). The integral is computed by the built-in routine `BiotSavart` by evaluating the integrand at the integration points of $\mathrm{supp}(\mathbf{J}_0)$ so that the result is just accurate for points outside $\mathrm{supp}(\mathbf{J}_0)$. Fig. 2.55 shows the five computational domains Ω_{vec} used

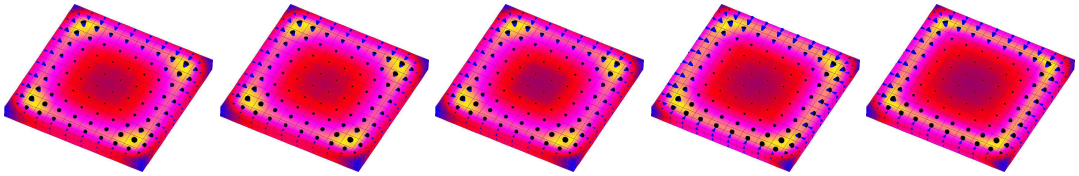

Figure 2.56: The magnetic field Euclidian norm $|\mathbf{B}|$ at the wire's boundary and on its interior together with directional cones.

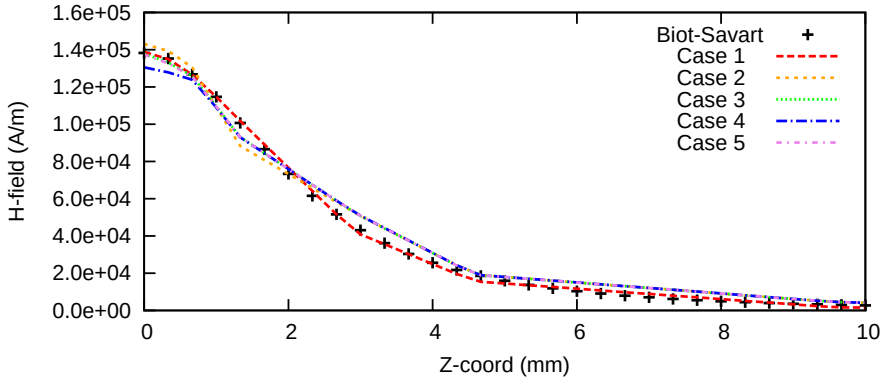

Figure 2.57: The z-component of the magnetic induction $\mathbf{H}$ along the symmetry axis.

for the vector formulation. The domain size for the vector formulation only decreases from left to right and so the computational cost, but the whole domain Ω as well as the finite mesh and $\mathbf{J}_0$ are identical in all five cases. The first case uses a full vector formulation $\Omega = \Omega_{\text{vec}}$ without scalar magnetic potential. The second and third cases use a cladding layer around $\text{supp}(\mathbf{J}_0)$. The second and the fourth cases enlarge the domain Ω_{vec} to be simply connected. The third and fifth cases do not have a simply connected domain Ω_{free} and require a jump definition for the magnetostatic potential. The value is available as BC characteristics from the solution of the electrostatic problem and is stored in the global variable `bcCurrent@i` specifying the jump BC value by the statement

```
For i From 1 To @NWIRE
{
   Write "Current%.0f=%e A\n" @i, bcCurrent@i=bcOne@i.Psi.Flux
}
```

Fig. 2.56 shows the magnetic induction around $\partial\Omega_{\text{vec}}$ and on the wire's interior. Since we are in empty space, the magnetic induction is continuous and so for a more precise representation, we have used a smoothing procedure averaging the contribution from neighbor elements. All five cases yield very similar results. Fig. 2.57 shows the z-component of $\mathbf{H}$ along the symmetry axis through the wire's center. Here one sees that the most precise formulation is the full vector formulation which is also the most expensive to compute when using a direct solver.

We end these numerical experiments by noting, that among all five models just the first one results in a regular linear system. You may check this fact by setting the variable `displaymatcond` and thus running some Maple code on the linear system matrix written to the file `/tmp/mat` by the statement

```
Info LinearMatrix Format "%.17e " MatrixFile "/tmp/mat" HideBC
```

When using both the scalar and vector formulation, there are excess of dofs at the interface $\partial\Omega_{\rm vec}$ which are not eliminated internally. In this case, the direct solver is still capable of computing the correct solution, but attention since this may not always be the case as for example when just computing an half, quarter or eighth of this symmetric solution. Also do not except the same symmetry for the vector potential $\mathbf{A}$, which however must be the case for $\mathbf{B}$ and $\mathbf{H}$.

2.14 Eddy currents in a linear system

The numerical characterization of a system of conductors, wires or coils with time-varying currents is in the industrial developing cycle an important topic and may be used to characterize e.g. inductive proximity sensors. For many problems of interest, the geometry is truly 3D so that 2D models cannot be used and numerical expensive 3D computations have to be done. However, by assuming a system with a linear response, the analysis performed within the context of linear theory simplifies the problem at hand. Further, if just the low frequency range is of interest, we will show some valid approximations to additionally reduce the computational work. Therefore let Ω be a domain of $\mathbb{R}^3$ and assume in Ω we have N closed and disjoint conductors $\Omega_m \subset \Omega$, $\Omega_m \cap \Omega_n = \emptyset$ for $m \neq n$ and $m, n = 1 \dots N$. Just within the conductor domain $\Omega_{\rm wire} = \cup_m \Omega_m$, we have a non-zero conductivity. For time-varying currents up to some upper limit frequency, the eddy current model represents a valid submodel of Maxwell's equation, see [1] for a mathematical background, which is given by solving the following equations in Ω

$$\nabla.\mathbf{B} = 0\,, \quad \nabla \times \mathbf{E} + \partial_t \mathbf{B} = 0\,, \quad \nabla \times \mathbf{H} = \mathbf{J} = \sigma \mathbf{E}\,, \tag{2.52}$$

with $\mathbf{E}$, $\mathbf{H}$, $\mathbf{B}$ the electric, magnetic and induction field, σ, μ the conductivity and permeability and the material laws $\mathbf{B} = \mu\mathbf{H}$, $\mathbf{J} = \sigma\mathbf{E}$. By taking the divergence of $\nabla \times \mathbf{H} = \mathbf{J}$, we also have $\nabla.\mathbf{J} = 0$. In order to simplify a little bit the matter and so the computational work too, we assume a linear dynamical system. Since the above equations are all linear with respect to the electro-dynamical fields, this implies linear material laws and thus conductivity σ and permeability μ can only be functions of the coordinates. For the numerical solution of the above equations, several formulations are available and the one adopted in *SESES* is based on the computation of the vector potential $\mathbf{A}$ and the time-integral $\overline{\Psi} = \int \Psi\, dt$ of the scalar potential Ψ as primary fields. In this case, we have $\mathbf{B} = \nabla \times \mathbf{A}$, $-\nabla\Psi = \mathbf{E} + \partial_t \mathbf{A}$ and the following equations are left to solve

$$\nabla \times \mu^{-1} \nabla \times \mathbf{A} + \partial_t(\sigma\mathbf{A} + \sigma\nabla\overline{\Psi}) = 0\,, \quad \partial_t \nabla.(\sigma\mathbf{A} + \sigma\nabla\overline{\Psi}) = 0\,, \tag{2.53}$$

although the second one is implied by the first one by taking the divergence. In practice, when one has to apply voltages and measuring currents or doing the reverse, the wires are not closed in Ω since they need to be connected to external measuring equipment. From a numerical point of view, however, we may assume ideal contacts where the measuring equipment has no influence on the system. In order to define ideal contacts, for each conductor Ω_m, $m = 1 \dots N$, we select a generic cross-section C_m with C_m^+, C_m^- being the two distinct oriented surfaces of C_m. We then allow the

potential Ψ to have a constant jump in its value when crossing this surface. The jump value is given by the path integral

$$V_m = \Psi(C_m^+) - \Psi(C_m^-) = -\int_{\gamma_m} \nabla\Psi d\mathbf{l} = \int_{\gamma_m} (\mathbf{E} + \partial_t \mathbf{A})\mathrm{d}\mathbf{l}\,, \tag{2.54}$$

with γ_m any directional path in Ω_m joining the surfaces $C_m^+ \rightsquigarrow C_m^-$ at the same point. The invariance of V_m from the choice of C_m is given by the fact that a constant jump of Ψ does not change the gradient $\mathbf{E} = -\nabla\Psi$. Since there is no current flowing through the conductor boundary $\partial\Omega_m$ and by considering $\nabla.\mathbf{J} = 0$ together with Gauss's theorem, we see that the computation of the conductor current I_m is also invariant from the choice of C_m. However, we have to consider the correct side in order to get the correct sign and by the above choice of V_m, we have

$$I_m = \int_{C_m^+} \mathbf{J} \cdot \mathrm{d}\mathbf{n}\,. \tag{2.55}$$

It is customary to study the dynamic of a linear system by using the complex notation for all involved fields, with the real part representing the physical value and by assuming a harmonic time-dependency of the form $\exp(i\omega t)$ which is always implied and for conciseness not written explicitely. At each harmonic frequency ω, our linear system is fully characterized by the impedance matrix $Z(\omega)$ with $\vec{V} = Z(\omega)\vec{I}$ and $\vec{V}, \vec{I} \in \mathbb{C}^N$ being the voltage and current values at the contacts. From (2.54), we see that at low frequencies $\omega \to 0$, the impedance has a constant real part and an imaginary part linear in ω, showing the inductive character of the system. Therefore the complex impedance is also written in the following form, splitting the real and imaginary parts into a resistance matrix R and inductance matrix L

$$Z(\omega) = R(\omega) + i\omega L(\omega)\,. \tag{2.56}$$

Since the equations (2.52) are uniquely solvable, the impedance matrix is regular and can be inverted. Therefore for numerical computations, we have the free choice in selecting either $\vec{V}$ or $\vec{I}$ as independent variables but even a mix would be possible. Because we solve for the primary fields $\mathbf{A}$ and $\overline{\Psi} = \Psi/(i\omega)$, here the most convenient way is to compute the admittance matrix $Y(w) = Z(w)^{-1}$ by choosing the contact voltages as independent variables which are specified by setting Dirichlet and jump BCs for $\overline{\Psi}$. The choice of the contact currents as independent variables is not so convenient in this formulation, although possible. We then solve N-problems $m = 1 \ldots N$ by selectively setting the value $V_m = 1$ on each conductor Ω_m and $V_n = 0$ on all other ones Ω_n, $n = 1 \ldots N, m \neq n$ and then compute the N-current values in the conductors by (2.55), yielding each time the m-column of $Y(w)$.

In general the computation of eddy currents is delicate, although the equations to be solved are linear. In particular, at high frequencies one has to pay attention to have a good mesh resolution in the conductors, since by the skin effect, the current tends to flow just beneath the conductor surface. For a planar conductor, the current skin depth is given by $\delta = \sqrt{2/\mu\omega\sigma}$ and since in our example we are going to use a copper wire cross-section of $1\,\mathrm{mm}$ with $\sigma = 5.8{\times}10^7\ \mathrm{A/(V\,m)}$ and just few elements, we see that our numerical example will work up to $\omega_{\max} \approx 10\,\mathrm{kHz}$.

Low frequency approximation

For our system of conductors, the time-averaged energy dissipation $W =< \Re\vec{V} \cdot \Re\vec{I} >_t = |\vec{I}| \cdot \Re Z(\omega)|\vec{I}|/2 = |\vec{V}| \cdot \Re Y(\omega)|\vec{V}|/2$ is just determined by the real part of the impedance or admittance matrix. Since by a Taylor expansion of $Z(\omega)$ or $Y(\omega)$ at $\omega = 0$, the real linear term in ω is missing, at low frequencies the energy dissipation is a quadratic function of ω. If one is just interested in this quadratic term or in general, in the first few Taylor coefficients of $Z(\omega)$ or $Y(\omega)$, these values may be computed in a less expensive way than using the classical approach of computing several eddy model solutions. We first note that for $\omega = 0$, the eddy model (2.52) decouples into the electrostatic and magnetostatic formulations of Maxwell's equation and as a consequence, the computation of the Taylor coefficients, always involves real and smaller linear problems to be solved with respect to a complex coupled solution of (2.53). For our formulation, the most convenient approach is to compute the admittance matrix $Y(w)$ and so we are looking for its Taylor's approximation

$$Y(\omega) = \sum_{p\geq 0} \partial_\omega^p Y(0) \frac{\omega^p}{p!} = Y_0 + Y_1\omega + Y_2\frac{\omega^2}{2} + \dots \tag{2.57}$$

with $Y_p = \partial_\omega^p Y(0)$ and which is based on computing the Taylor's coefficients of the conductor current (2.55)

$$\partial_\omega^p I_m(0) = \int_{C_m^+} \partial_\omega^p \mathbf{J}(0) \cdot \mathrm{d}\mathbf{n} = \int_{C_m^+} \sigma \partial_\omega^p \mathbf{E}(0) \cdot \mathrm{d}\mathbf{n}\,. \tag{2.58}$$

Up to the second order, the impedance $Z(\omega) = Y(\omega)^{-1}$ is related to (2.57) by

$$Z(\omega) = Y_0^{-1} - Y_0^{-1}Y_1Y_0^{-1}\,\omega + [2Y_0^{-1}(Y_1Y_0^{-1})^2 - Y_0^{-1}Y_2Y_0^{-1}]\frac{\omega^2}{2} + \dots \tag{2.59}$$

The governing equations determining the derivatives $\partial_\omega^p \mathbf{E}(0)$ in (2.58) are obtained by taking the ω-derivatives in (2.52)

$$\nabla \times (\partial_\omega^p \mathbf{E}(0) + i\,p\,\partial_\omega^{p-1}\mathbf{A}(0)) = 0\,,\ \ \nabla \times \mu^{-1}\nabla \times \partial_\omega^p \mathbf{A}(0) = \sigma \partial_\omega^p \mathbf{E}(0)\,. \tag{2.60}$$

These equations build-up a recursion of linear problems with the property that the two linear matrices involved are always the same, we just have different right-hand sides. The recursion starts by computing $\mathbf{E}(0)$ and solves the electrostatic problem

$$\nabla \times \mathbf{E}(0) = 0\,, \nabla.\sigma\mathbf{E}(0) = 0 \quad \text{or} \quad \nabla.\sigma\nabla\Psi(0) = 0\,, \tag{2.61}$$

for the potential $\Psi(0)$. Since the domains Ω_n, $n = 1 \dots N$ are disjoint, we actually have to solve N-linear problems for the scalar potential $\Psi_n(0)$ in each domain Ω_n. From (2.58), the $Y(0)$ matrix of (2.57) is a diagonal matrix given by

$$Y(0)_{nn} = \int_{C_n^+} \sigma \mathbf{E}_n(0) \cdot \mathrm{d}\mathbf{n}\,. \tag{2.62}$$

The next step in the recursion computes $\mathbf{A}(0)$ and solves the magnetostatic problem

$$\nabla \times \mu^{-1}\nabla \times \mathbf{A}(0) = \mathbf{J}(0)\,,\ \nabla.\mathbf{A}(0) = 0\,, \tag{2.63}$$

with the current $\mathbf{J}(0) = \sum_n \mathbf{J}_n(0) = \sum_n \sigma \mathbf{E}_n(0)$ computed by (2.61). By linearity, we have $\mathbf{A}(0) = \sum_n \mathbf{A}_n(0)$ with $\mathbf{A}_n(0)$ the solution of (2.63) with source term $\mathbf{J}_n(0)$ instead of $\mathbf{J}(0)$. The major difference with (2.61) is that each singular problem for $\mathbf{A}_n(0)$ has to be solved in Ω and not just in Ω_n.

Without giving a proof, this magnetostatic solution can be used to compute the inductance matrix $L(\omega)$ of (2.56) at $\omega = 0$ which is actually the magnetostatic inductance L determining the energy W. By assuming $\mathbf{B} = 0$ on $\partial\Omega$, we have

$$\begin{aligned} W &= \frac{1}{2}\int_\Omega \mathbf{H}\cdot\mathbf{B} = \frac{1}{2}\int_\Omega \mathbf{H}\cdot\nabla\times\mathbf{A} = \frac{1}{2}\int_\Omega \nabla\times\mathbf{H}\cdot\mathbf{A} = \frac{1}{2}\int_\Omega \mathbf{J}\cdot\mathbf{A} \\ &= \frac{1}{2}\sum_{m,n}\int_{\Omega_m} \mathbf{J}_m\cdot\mathbf{A}_n = \frac{1}{2}\sum_{m,n} L_{mn} I_m I_n\,, \end{aligned} \tag{2.64}$$

with

$$L_{mn} = \frac{1}{I_m I_n}\int_{\Omega_m} \mathbf{A}_n(0)\cdot\mathbf{J}_m(0)\,\mathrm{d}V\,. \tag{2.65}$$

The next term in the recursion (2.60) computes $\partial_\omega\mathbf{E}(0)$ and solves

$$\nabla\times(\partial_\omega\mathbf{E}(0) + i\mathbf{A}(0)) = 0\,,\ \nabla.\sigma\partial_\omega\mathbf{E}(0) = 0\,,$$

with the vector potential $\mathbf{A}(0) = \sum_n \mathbf{A}_n(0)$ computed in (2.63). The source term $i\mathbf{A}(0)$ is pure imaginary and so the solution $\partial_\omega\mathbf{E}(0)$ as well. Therefore, we define the real scalar field Φ with $i\partial_\omega\mathbf{E}(0) = -\nabla\Phi + \mathbf{A}(0)$ determined by solving in each conductor Ω_n the real equation

$$\nabla.(\sigma\nabla\Phi - \sigma\mathbf{A}(0)) = 0\,. \tag{2.66}$$

By linearity, we have $\partial_\omega\mathbf{E}(0) = \sum_n \partial_\omega\mathbf{E}_n(0)$ with $\partial_\omega\mathbf{E}_n(0)$ given by solving (2.66) with source term $\sigma\mathbf{A}_n(0)$ instead of $\sigma\mathbf{A}(0)$. The solution of this problem is very close to the electrostatic problem (2.61) and in fact we have the same linear matrices. From (2.58), the $\partial_\omega Y(0)$ matrix of (2.57) is pure imaginary and reads

$$\partial_\omega Y(0)_{mn} = \int_{C_m^+} \sigma\partial_\omega\mathbf{E}_n(0)\cdot\mathrm{d}\mathbf{n}\,.$$

To compute the next coefficient $\partial_\omega^2 Y(0)$, we need to solve for $\partial_\omega\mathbf{A}(0)$, a problem similar to the magnetostatic problem (2.63) and for $\partial_\omega^2\mathbf{E}(0)$, a problem similar to (2.66) but just with other source terms. And so on for each additional Taylor's coefficient of $Y(\omega)$ which are alternatively real and imaginary.

Model specification

When solving the eddy current model (2.53) in *SESES*, we have to take a decision, either to use $H(\Omega)^3$ or $H(\mathrm{curl},\Omega)$ conformal elements, the former also known as nodal elements and the latter as edge elements. The edge elements are the natural framework when working with the curl-curl operator of (2.53), since they do not enforce the normal component of the vector field to be continuous and differently from nodal elements, they are well suited to model sharp corners and sharp discontinuous material laws, see [2, 4]. However, the eddy current formulation cannot be gauged with

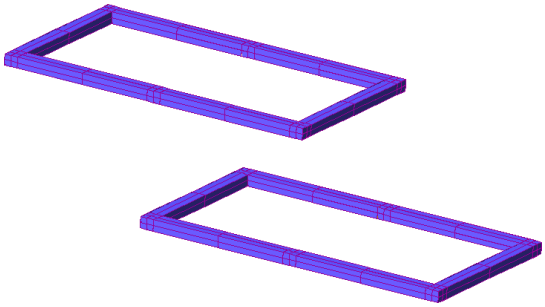

Figure 2.58: View of the two coils geometry.

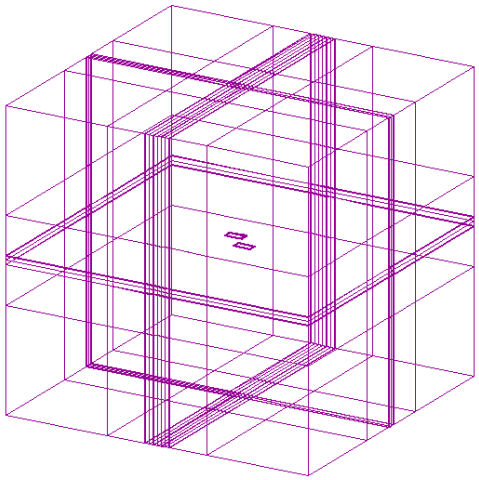

Figure 2.59: The computational domain showing the finite elements just on material boundaries.

edge elements and the linear system to be solved is singular. If the equations are assembled in a consistent manner, this is not a drawback for iterative solvers. However, the convergence rate is strongly influenced by the frequency, the electric conductivity, the shape of the elements and optimization techniques used to reduce the number of unknowns, e.g. symmetrization and the removal of the electric potential in the conductors with constant conductivity. Therefore when using edge elements, the success of the method is basically given by the convergence rate of the iterative solver and the availability of a good preconditioner. On the other side, nodal elements suffer from an inferior matrix conditioning, but the formulation based on nodal elements is gauged and so the linear system to be solved is regular. Here a robust direct solver can eventually be used for not too large problems with the advantage that optimizations reducing the number of unknowns but not the accuracy will directly result in a speed-up and that the time-expenditure does not change by varying the frequency ω. For this example, we have chosen a numerically robust formulation with nodal elements and a direct solver so that one has to be careful when extending this problem with regions of high permeability $\mu \gg \mu_0$.

When solving the eddy model (2.53) for a constant conductivity, we see that $\overline{\Psi} \equiv 0$ is a valid solution enforcing the Coulomb gauge $\nabla.\mathbf{A} = 0$ and so removing the electric potential $\overline{\Psi}$ from the solution process will speed-up the computation. Although it is not possible anymore to specify the driving voltage at the wire contact, an alternative is available. In a preprocessor step, one solves the electrostatic problem (2.61) by applying the driving voltage and the computed current $\mathbf{J}_0$ is used to define a stiff-contribution to the current $\mathbf{J} = \sigma\mathbf{E} + \mathbf{J}_0$. If the electric potential is removed from the solution, then it is also possible to symmetrize the linear system, thus obtaining a further speed-up. Given the field approximation $\mathbf{A}_h = \sum_i \mathbf{H}_i A_i$ with $\mathbf{H}_i$ the real shape functions and $A_i \in \mathbb{C}$ the complex unknowns, for nodal shape functions one solves

the complex linear equations

$$R_i = \int_\Omega [(\nabla \times \mathbf{H}_i)\mu^{-1} \cdot (\nabla \times \mathbf{A}_h) + \mu_*^{-1}(\nabla.\mathbf{H}_i)(\nabla.\mathbf{A}_h) + i\sigma\omega \mathbf{H}_i \cdot \mathbf{A}_h]\mathrm{d}V = 0\,, \quad (2.67)$$

with μ_* a constant average of μ on Ω. By splitting real and imaginary parts and by assuming μ is real, we have the following real linear system

$$\begin{array}{c} \\ \Re\mathbf{A} \\ \Im\mathbf{A} \end{array} \begin{array}{c} \begin{array}{cc} \Re\mathbf{A} & \Im\mathbf{A} \end{array} \\ \begin{pmatrix} A & B \\ -B & A \end{pmatrix} \end{array}, \quad (2.68)$$

with the blocks $A_{ij} = \int_\Omega [(\nabla \times \mathbf{H}_i)\mu^{-1} \cdot (\nabla \times \mathbf{H}_j) + \mu_*^{-1}(\nabla.\mathbf{H}_i)(\nabla.\mathbf{H}_j)]\mathrm{d}V$ and $B_{ij} = \int_\Omega \sigma\omega \mathbf{H}_i \cdot \mathbf{H}_j \mathrm{d}V$ being symmetric matrices. This global non-symmetric matrix with positive-definite diagonal blocks is the default assembled matrix, but we see that if we change the sign of the second row block, a symmetric matrix is left to be solved. Although we can apply these two optimization techniques to our example, our advice is to keep the electric potential, otherwise there is no exact current conservation within the conductors and the computed admittance or impedance matrices are in general far less precise. In particular, the low frequency approximation does not agree with a Taylor expansion of the discrete eddy current model.

The input file for this model can be found at `example/Eddy.s3d`. The initial section is divided in two parts, in the first one we have the geometry definition. One defines here the number of wires with the macro `NWIRE` and for each i-wire, the three domains `Wire<i>`, `CutLeft<i>`, `CutRight<i>` and a macro `Corner<i>`. The domain `Wire<i>` defines the wire itself. The domains `CutLeft<i>`, `CutRight<i>` are used to define a wire cut-section in order to apply a potential jump and to compute the current. The cut-section must be defined by the intersection of the two domains. The macro `Corner<i>` must define a point within the wire and is used to ground the potential at that point. This geometry section is the only one which needs to be changed in case of other geometrical choices. For this example, we have chosen a simple geometry with two coils as displayed in Fig.2.58-2.59.

The second part of the initial section constructs the computational domain and define two materials `Wire`, `AirEddy`. The `Wire` material is the copper conductor with a conductivity of $\sigma = 5.8{\times}10^7$ A/(V m). Solving the harmonic eddy formulation (2.53) with nodal elements is done by defining the material equation `EddyCurrentHarmonicNodal`. The `AirEddy` material is defined for solving (2.53) outside the conductors, where we have $\sigma = 0$. Here, we can assume $\overline{\Psi} \equiv 0$ and therefore we select the material equation `EddyFreeHarmonicNodal` not solving for the electric potential $\overline{\Psi}$. The problem description is concluded by defining BCs. To compute the electric potential, we define for each wire a point BC to fix and ground the potential and a section where we apply a potential jump. This is the most neutral method to apply a driving voltage, since it does not define any equipotential surface within the wire. At the exterior boundary $\partial\Omega$, we define the Dirichlet BC $\mathbf{A} = 0$ representing an infinitely extended domain.

The command section solves several linear problems in order to compute the impedance matrix (2.56) and the low frequency approximation (2.59). Coefficients of the linear system matrix (2.68) which are known in advance to be always zero are not defined

within the sparse matrix format used by the linear solver. However, the zeroness of many matrix coefficients actually depend on the user specifications in which case they are always defined independently from their numerical value. In our case, the material `AirEddy` has a zero conductivity $\sigma = 0$, so that the off-diagonal block matrix B of (2.68) has many zero entries. This default behavior is important for non-linear problems, since the sparse format is build once at the beginning of the non-linear block solution and here it is inconvenient to make the format dependent on the zeroness of the coefficients in the first iteration, which may not be zero in later iterations. For our example, however, we just solve linear problems and so it is safe to remove all zero or numerical close-to-zero coefficients from the matrix which are detected when building the sparse matrix. This non-default behavior is obtained with the statement

```
LinearSolver CoeffCutOff 0
```

which results in some speed-up. However, the most important speed-up is obtained by keeping the factorized matrix when solving a sequence of linear problems with an invariant matrix. The admittance matrix is computed by solving N-problems and by applying separately in each wire a driving voltage. For the same frequency ω, a closer look shows that the linear matrix obtained by discretizing (2.53) is unchanged and we just have different right-hand side vectors. Therefore, the most expensive numerical operation of factorizing the system matrix must be done only once and the N-problems are then solved by performing N-backward-forward substitutions which are by far less expensive, see [5]. This non-default behavior is enabled with the statement

```
Increment ReuseFactoriz
```

which tries, whenever meaningful, to reuse a previously factorized matrix.

As first, we compute the low frequency approximation by solving an alternate sequence of electrostatic (2.66) and magnetostatic (2.63) problems with the help of the following command procedures.

```
Procedure double SolveElectroStatic()
{
   MaterialSpec Wire Equation EddyFreeNodal Disable OhmicCurrent Enable
   BlockStruct Block OhmicCurrent ReuseMatrix electromat
   Solve Stationary
   For m From 1 To @NWIRE { Define CUR[@m-1]=v@m.Psi.Flux }
   Store current1(DofField Afield Psi; IpFrom FIELD)=Current
}


Procedure double SolveMagnetoStatic()
{
   MaterialSpec Wire    Equation EddyFreeNodal Enable OhmicCurrent Disable
   MaterialSpec AirEddy Equation EddyFreeNodal Enable
   BlockStruct Block EddyFree ReuseMatrix magnetomat
   Solve Stationary
   Store current1(DofField Afield Psi; IpFrom FIELD)=Afield*SigmaIso
}
```

For both problems, the source term is an initial current and is defined by the material parameters

```
MaterialSpec Wire
  Parameter Current0  current1 A/m**2
  Parameter Current00 current1 A/m**2
```

We use here the element field `current1` as link between both formulations. At the end of the electrostatic problem, we store in `current1` the computed current $\mathbf{J}$ and at the end of the magnetostatic problem, we store the value of $\sigma\mathbf{A}$. This is a necessity, since when solving e.g. the electrostatic problem by enabling the material equation `OhmicCurrent`, afterwards the value of $\mathbf{A}$ from the previous magnetostatic solution is lost. Since we solve alternatively two different linear problems, the option `ReuseFactoriz` discussed above will not work here, since it requires the same block matrix. However, we can use the back-storage of block matrices enabled with the option `ReuseMatrix <name>`. Here at the beginning of a block solution, the back stored matrix `<name>` is loaded in memory and stored back at the end of the block solution so that the `ReuseFactoriz` option will work again. The matrix back-storage can be released with the statement

```
Misc ReleaseMatrix
```

The sequence of linear problems starts with the electrostatic problem (2.61) computing in one single step all wire currents $\mathbf{J}_m(0)$, $m = 1 \ldots N$. All these initial currents are required later on to compute the inductance matrix $L(0)$ by the energy method (2.65) and therefore they are stored in the element field `current0`. Afterwards, for each single wire, we solve the sequence of electro- and magneto-static problems. We compute here the admittance matrix (2.57) up to the second order and from (2.59) we obtain the resistance $R(0)$ and inductance $L(0)$ matrices at $\omega = 0$. After the first magnetostatic solution, we can compute $L(0)$ by the energy method (2.65) but the volume integral (2.64) over Ω may be used as well. From the residual equations (2.67) together with $\mathbf{A}_h = 0$ on $\partial\Omega$, it is possible to show that the same numerical values are obtained, but just if we exactly have $\nabla.\mathbf{A}_h = 0$, which is not generally the case. These resistance and inductance matrices can also be obtained by solving the eddy current model (2.53) at very low frequencies. A frequency value of $\omega = 0$ cannot be used, but the problem is stable with respect to $\omega \to 0$ and so we use $\omega = 2\pi\times10^{-15}$ Hz. The result is that up to machine precision, the two computed resistance matrices are the same as well as the three inductance matrices.

The eddy current problem is solved with the statements

```
For n From 1 To @NWIRE
{
   BC v@n Jump iVint -1/Frequency V*s Natural Vint
   Solve Stationary
   Define For m From 1 To @NWIRE
   {
      (ADM[@{2*@NWIRE*(m-1)+2*(n-1)+0}],ADM[@{2*@NWIRE*(m-1)+2*(n-1)+1}])=
          (* residual(Bound v@m;Current,iCurrent) *)
          (-v@m.iVint.Flux,v@m.Vint.Flux)
   }
   BC v@n Natural iVint
}
```

For each wire, we set the driving voltage of $1\,\mathrm{V}$, solve the linear system and compute the N-wire currents yielding each time, one column of the admittance matrix. Since we solve for the time-integrated electric potential $\overline{\Psi} = \Psi/(i\omega)$, a jump BC value of $-1/\omega$ is used for the dof-field `iVint`. It is a must to use the BC characteristics for the dof-fields `Vint`, `iVint` to compute the conductor currents (2.55) or a valid alternative is given by the built-in function `residual`, however, one should not directly evaluate boundary integrals. This is the only available method, if one removes the electric

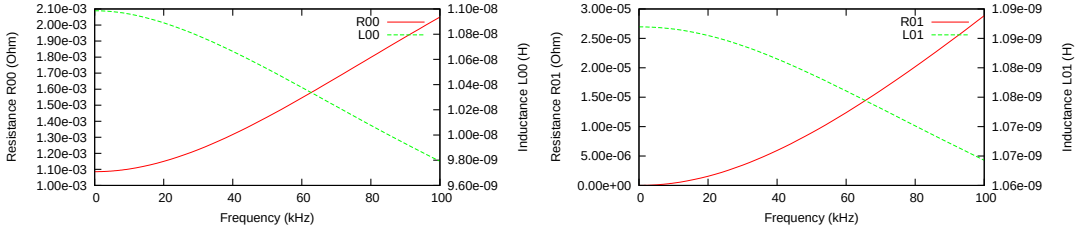

Figure 2.60: Resistance and inductance matrix coefficients up to $\omega = 2\pi\times10^5$ Hz.

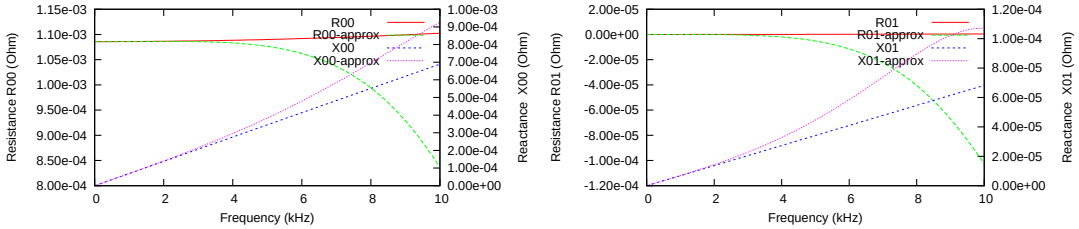

Figure 2.61: Resistance and reactance matrix coefficients compared with a second order low frequency approximation.

potential $\overline{\Psi}$ from the solution process and the disappointing observation is that e.g. the three presented methods to compute the inductance matrix at $\omega = 0$ will then give different results with up to 20% discrepancies.

Up to machine precision, the computed resistance R and inductance L matrices of the impedance (2.56) are symmetric and Fig. 2.60 shows the R_{00}, R_{01} and L_{00}, L_{01} coefficients for a frequency sweep up to $\omega_{\max} = 2\pi\times10^5$ Hz. Fig. 2.61 shows a perfect agreement with the resistance $R_{00} = \Re Z_{00}$, $R_{01} = \Re Z_{00}$ and reactance $X_{00} = \Im Z_{00}$, $X_{01} = \Im Z_{01}$ coefficients compared with the values obtained from a second order approximation of the admittance $Y(\omega) = Z(\omega)^{-1}$.

References

[1] H. AMARI, A. BUFFA, J. C. NÉDÉLEC, *A justification of eddy currents model for the Maxwell equations*, SIAM J. Appl. Math., Vol. 60, No. 5, pp. 1805-1823, 2000.

[2] O. BÍRÓ, *Edge element formulations of eddy current problems*, Comput. Methods. Appl. Mech. Engrg., No. 169, pp. 391-405, 1999.

[3] O. BÍRÓ, A. VALLI, *The Coulomb gauged vector potential formulation for the eddy-current problem in general geometry: Well-posedness and numerical approximation*, Comput. Methods. Appl. Mech. Engrg., No. 196, pp. 1890-1904, 2007.

[4] P. MONK, *Finite element methods for Maxwell's equations*, Oxford University Press, 2003.

[5] G. H. GOLUB, C. F. VAN LOAN, *Matrix computations*, 2nd edition, The John Hopkins University Press, 1989.

2.15 Preconditioning eddy current systems

The modeling of multiphysics systems leads to the successive solution of generally large but sparsely filled linear systems of equations. To solve these linear equations, one has the choice between a direct solver based on a LU-factorization and requiring a fixed amount of memory and floating point operations or an iterative solver requiring a fixed amount of memory but running an iteration loop up to a satisfactory convergence criterion is reached. Here the convergence rate depends on the condition number $\rho = |\lambda_{\max}/\lambda_{\min}|$, with $\lambda_{\min}$ and $\lambda_{\max}$ the extreme eigenvalues of the system matrix. For $\rho = 1$, we have the identity matrix and our linear system is easily solved, but for increasing values $\rho > 1$, the matrix departs from the identity matrix and the convergence rate slows down. Iterative solvers requires the storage of the system matrix and some additional working vectors to run, whereas direct solvers requires much more memory since many zero coefficients of the initial sparse system matrix become non-zero during the factorization. For 3D applications, the memory and computational requirements are generally too large for direct solvers to be used and so one has to resort to iterative solvers. Quite some iterative solvers can be found in numerical libraries and used out of the box, however, they do not generally converge at a satisfactory rate or may even diverge and therefore are not robust as direct solvers. This problem can be partially overcome by preconditioning the linear system, i.e. by solving a similar system much closer to the identity matrix in order to increase the convergence rate and thus the robustness of the solution process. Clearly in order to be effective, the associated similar system must be computed in a cheap way and one major drawback is that optimized and very effective preconditioners are problem dependent and need to be investigated and found for each application. But also within a single application, the best known preconditioner may not be very robust and the convergence rate may heavily depend on material parameters and the shape of the elements both having a strong impact on the condition of the linear system.

In this example, we are going to study preconditioning techniques available for solving the harmonic eddy current problem both with nodal and edge elements. This model is obtained from the Maxwell's equations by dropping the displacement current term, by using the time dependency $(\bullet)(t) = (\bullet)e^{i\omega t}$, complex field values, the linear material laws $\mathbf{J} = \sigma\mathbf{E}$ and $\mathbf{B} = \mu\mathbf{H}$ and is given by

$$\nabla.\mathbf{B} = 0\,, \quad \nabla \times \mathbf{E} + i\omega\mathbf{B} = 0\,, \quad \nabla \times \mu^{-1}\mathbf{B} = \mathbf{J}_0 + \sigma\mathbf{E}\,, \tag{2.69}$$

with σ the electric conductivity, μ the permeability, ω the frequency and $\mathbf{J}_0$ a possible stiff driving current. From the property $\nabla.\mathbf{B} = 0$, we can define the vector potential $\mathbf{A}$ with $\mathbf{B} = \nabla \times \mathbf{A}$. From the second equation in (2.69), we obtain the integrability condition $\nabla \times (\mathbf{E} + i\omega\mathbf{A}) = 0$ for the scalar potential Ψ with $-\nabla\Psi = \mathbf{E} + i\omega\mathbf{A}$. For a piecewise constant conductivity σ, one can show that it makes sense to assume a globally zero scalar potential $\Psi = 0$ so that one is left with the single linear equation

$$\nabla \times \mu^{-1}\nabla \times \mathbf{A} + i\omega\sigma\mathbf{A} = \mathbf{J}_0\,, \tag{2.70}$$

to be solved for the complex vector potential $\mathbf{A}$. Since we have dropped the computation of the scalar potential Ψ, the stiff current $\mathbf{J}_0$ is generally used as the driving source for the magnetic field. On a domain $\Omega \subset \mathbb{R}^3$ with conductors Ω_m, $m = 1 \ldots N$

where $\sigma(\Omega_m) = \text{const}$ and $\sigma(\Omega/\cup_m \Omega_m) = 0$, in a preprocessing step we apply voltages to the conductors resulting in quasi-static stiff currents $\mathbf{J}_0$. For $\omega > 0$, we then have additional eddy currents $\mathbf{J} = \mathbf{J}_0 + \sigma\mathbf{E} = \mathbf{J}_0 - i\omega\sigma\mathbf{A}$ induced by the time-dependent magnetic field $\mathbf{B} = \nabla \times \mathbf{A}$.

When solving the eddy current problems in *SESES*, we have to take a decision, either to use $(H^1(\Omega))^3$ or $H(\text{curl}, \Omega)$ conformal elements to approximate the vector potential $\mathbf{A}$, the former also known as nodal elements and the latter as edge elements. Then, from a weak formulation associated with the PDEs (2.70) and by plugging nodal or edge shape functions, we obtain a complex symmetric linear system of equations to be solved whose solution give us $\mathbf{A}$. The pro and contra of both finite element types have already been discussed in the Example 2.14 and here we just remind that with nodal elements, the system matrix is regular, but singular for edge elements if somewhere in the domain Ω we have $\sigma = 0$. Although it would be natural to use a complex linear solver to solve for the complex vector potential $\mathbf{A}$, in a multiphysics environment where several real and complex dof-fields may be computed simultaneously in a coupled manner, the most general solver is a real solver and complex dof-fields are simply split in a real and imaginary part. This approach can double the memory usage but from a numerical point of view is equivalent to a complex solution. In particular, since the linear system obtained is complex symmetric, as explained in the Example 2.14, it is possible to obtain a symmetric real system of the form

$$S = \begin{pmatrix} A & B \\ B & -A \end{pmatrix}, \tag{2.71}$$

with A, B symmetric real matrices. For symmetric matrices, the iterative solver of choice is the conjugate gradient method. In exact arithmetic and for positive definite matrices, it converges in at most N-step with N the size of the system [1, 2, 3]. Our real symmetric matrix (2.71) is not positive definite, but has pairs of real eigenvalues of opposite sign, since if $(\lambda, (x, y))$ is an eigenpair of S, then also $(-\lambda, (y, -x))$. Despite this fact, for our application the conjugate gradient generally converges smoothly, but stalling cannot be excluded and in general the chances of failure can be reduced with a good preconditioner.

We document here some numerical tests done with the model found at `example/EddyPrecond.s3d`, a simple voltage-driven rectangular coil as displayed in Fig. 2.62. The electrical conductivity is $\sigma = 5.8{\times}10^7\,\text{A}/(\text{Vm})$ and the driving voltage is applied as stiff current $\mathbf{J}_0$ computed in a preprocessing step with the `OhmicCurrent` model. The current is stored in a user field `cur0` used to define the material parameter `iCurrent0`. The eddy current problem is then solved using the equations `EddyFreeHarmonicNodal` or `EddyFreeHarmonic` for nodal and edge elements. The typical streamlines for the computed magnetic field are given in Fig. 2.63. In this example, we want to study the impact of the frequency and the element shape on the performance of the iterative solver. Therefore, the example is parameterized so as to freely change the frequency w and the slenderness $\alpha = \text{length}/\text{width}$ of the rectangular elements forming the coil.

Iterative solvers generally work both for regular and singular systems, if these latter have a solution. For regular matrices, the condition number ρ gives us a first impression how good iterative solvers will work, not however for singular matrices since

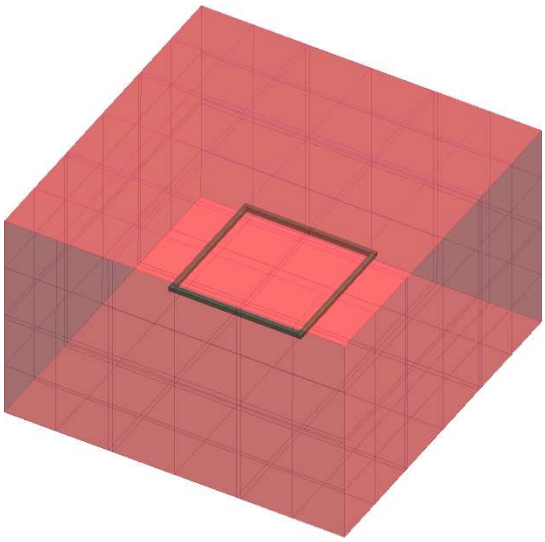

Figure 2.62: View of the single coil geometry.

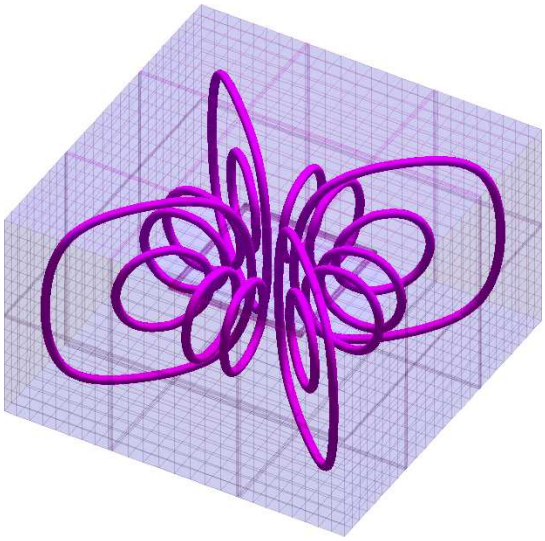

Figure 2.63: Streamlines of the real part of the magnetic field $\Re\,\mathbf{B}$.

Frequency ω	Slenderness α	# Iter-No P.	# Iter-Diag P.	#Iter-ILU(0) P.
1	1	86	86	14
1	100	21540	4554	81
10^2	1	92	92	16
10^2	100	29591	5235	84
10^4	1	162	160	10
10^4	100	31597	5887	84

Table 2.1: Number of solver iterations to reduce the residual by a factor of 10^{-8}, as function of the frequency ω, slenderness factor α and for no, diagonal or the ILU(0) preconditioner. The system matrix of size 19074 is for first order nodal elements.

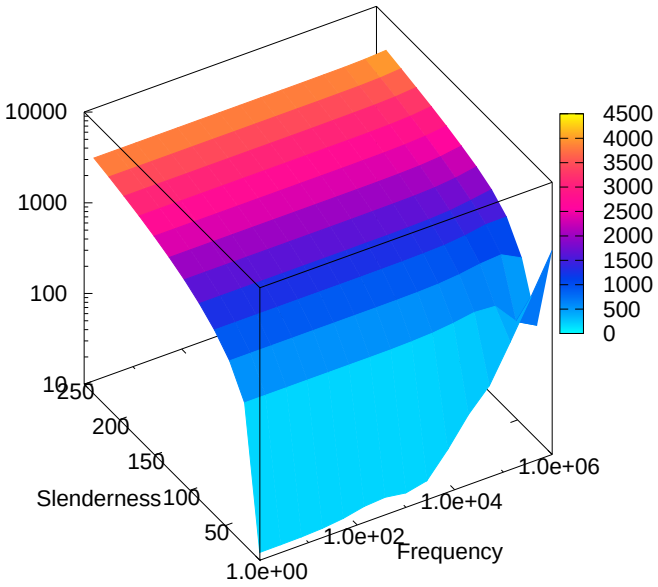

Figure 2.64: Condition number of the system matrix as function of the frequency w and slenderness factor α.

here we always have $\rho = \infty$. For not too many elements, the condition number can be computed in reasonable time so that for this example its computation is valuable in determining how strongly badly shaped elements and the frequency can impact on the convergence rate. Fig. 2.64 shows the condition number as function of these two parameters, one sees that badly shaped elements strongly worsen the condition number as well as the frequency for good shaped elements. Table 2.1 lists in the third column, the number of iterations required for the conjugate gradient to converge when working on the unamended system matrix of dimension 19074 and by requiring a reduction of the initial residual by a factor of 10^{-8}. One can see a close correlation between the condition number and the number of required iterations to converge. For large linear systems, a number of iteration larger than a few percent of the matrix dimension can be considered as a slow convergence.

The problem of the slow convergence for large condition numbers can be overcome by preconditioning the original linear system $Sx = b$ which is multiplied left and right with the preconditioning matrices P_{left}, P_{right} yielding the new system

$$S'(P_{\text{right}}^{-1}x) = b' \text{ with } \quad S' = P_{\text{left}}SP_{\text{right}} \text{ and } b' = P_{\text{left}}b\,. \tag{2.72}$$

The idea is for the preconditioning matrices to build a good approximation of the inverse, i.e. to have $S' \approx \mathrm{Id}$. We then apply the iterative solver to the system $S'x' = b'$ and since for the new condition number we have $\rho(S') \approx 1$, we expect a fast convergence. Since iterative solvers are generally initialized with $x = 0$, we do not have to perform the initial operation with the inverse $x' = P_{\text{right}}^{-1}x$ and at the end after having computed the solution x', we obtain our solution as $x = P_{\text{right}}x'$. This preconditioning technique is quite general, but clearly the success lies in the availability of cheap preconditioner matrices P_{left}, P_{right} and a good approximation property $S' \approx \mathrm{Id}$.

Running an iterative solver directly on a linear system obtained by the discretization of some PDES should never be done. Instead, one should always use a diagonal preconditioner which is very cheap to compute, does not require additional memory and is always superior than no preconditioner at all. The preconditioner is given by

$P_{\text{left}} = P_{\text{right}} = D$ with D a diagonal matrix defined by $D_{ii} = \sqrt{|S_{ii}|^{-1}}$ and has the only requirement $|S_{ii}| \neq 0$ which is always satisfied when discretizing PDEs. In our example, Table 2.1 shows quite an improvement for this diagonal preconditioner, but although this preconditioner can help, it is still not very robust and we have a large variation on the number of iterations necessary to reach convergence.

Another very popular preconditioner is the incomplete LU factorization without fill-in, in short notation $\text{ILU}(0)$. It is defined as left preconditioner $P_{\text{left}} = \text{ILU}(0)$ with $P_{\text{right}} = \text{Id}$. On a copy of the system matrix S, one performs the LU-factorization by discarding the fill-in, i.e. if a coefficient of the spare matrix S is zero, it will be zero also in the factorized form $S = \text{LU} \approx \text{ILU}(0)$. One knows that for an M-matrix, the incomplete LU-factorization cannot fail [1], i.e. we will never get a zero pivot, but the discretization of PDEs does not always yield M-matrices. Since the amount of fill-in in the LU-factorization is dependent on the ordering of the equations, permutation of the matrix can have a strong impact on the effectiveness of the $\text{ILU}(0)$ preconditioner.

In our example, for nodal elements the system matrix is not an M-matrix, but we are lucky and the $\text{ILU}(0)$ preconditioner works quite well for first order elements with good convergence rates for reasonable slenderness factors and frequencies, but only if the real and imaginary dofs of a complex dofs are clustered together in the global ordering of the dofs. Without this clustering, the $\text{ILU}(0)$ preconditioner easily fails and this is also the case when starting e.g. from an initial Cuthill McKee ordering and by widely and randomly permuting the dofs but by keeping the pairs together. In summary for first order nodal elements we have found a robust preconditioner, it is very good in reducing the convergence's dependency on the frequency and we just have a noticeable dependency from the slenderness factor, see Table 2.1. The convergence rate is so good that the problems to be solved are generally bounded by the large memory requirements and not by the computational time. This is also true in view of the fact that the solution time can be further reduced by parallelization and multigrid methods, technologies not discussed here.

For nodal elements of second order and the $\text{ILU}(0)$ preconditioner, the dependency on the convergence rate from the frequency is again low but much more pronounced for the slenderness factor with an impressive slow down, here an example for $\omega = 1$.

α	1	10	20	30	40	50	60	70
# Iter.	17	29	34	46	77	263	643	1838

As next we present a speed-up method similar to a two-cycle multigrid method, based on the robustness of the $\text{ILU}(0)$ preconditioner for first order nodal elements. If we have some additional memory at our disposal, we can apply a technique known as *auxiliary preconditioning*, where we use the solution obtained with first order elements to precondition the system of second order elements. On the same mesh and with the same BCs, one can show that the system matrices S_{first}, S_{2nd}, obtained with element of first and second order are related by $S_{\text{2nd}} = \Pi S_{\text{first}} \Pi^T$ with Π a *prolongation matrix* which is easily computed numerically. The left preconditioner is then defined as $P_{\text{left}} = \Pi S_{\text{first}}^{-1} \Pi^T$ and its application yields

$$P_{\text{left}} S_{\text{2nd}} = \Pi S_{\text{first}}^{-1} \Pi^T \Pi S_{\text{first}} \Pi^T \,. \tag{2.73}$$

Freq. ω	Slend. α	Pre 1-Post 0	Pre 0-Post 1	Pre 1-Post 1	Pre 2-Post 2
1	1	24(>500)	8(13)	6(12)	5(12)
1	100	>500(>500)	94(125)	67(93)	48(75)
10^2	1	22(272)	8(15)	6(15)	5(14)
10^2	100	>500(>500)	94(125)	67(93)	48(75)
10^4	1	>500(>500)	>500(>500)	18(17)	13(15)
10^4	100	>500(>500)	94(125)	67(95)	48(80)

Table 2.2: Number of solver iterations to reduce the residual by a factor of 10^{-8}, as function of the frequency ω, slenderness factor α and number of pre- and post-smoothing cycles. The system matrix of size 72318 is for second order nodal elements and it is preconditioned by the auxiliary preconditioner based on first order nodal elements. The auxiliary system of size 19074 is solved either with an exact LU or an $\mathrm{ILU}(0)$ preconditioner (value in parenthesis).

By considering the fact $\Pi^T\Pi \approx \mathrm{Id}$, we then have $P_{\mathrm{left}}S_{\mathrm{2nd}} \approx \mathrm{Id}$. The theory is actually a little bit more involved and in this form the method does not yet work. Before or after applying P_{left}, one has to smooth the high frequencies of the numerical error in the solution not seen by the element of first order and here the numerical approach is exact the same as for multigrid methods. The classical smoother is the Gauss-Seidel method, an iterative solver with less performance than the conjugate gradient but with excellent smoothing properties [1]. In general, the smoother should also converge if used as an iterative solver, but for this application where the system matrix is not positive definite nor an M-matrix, the Gauss-Seidel solver promptly diverges. As smoother cannot be used either and the fix is to apply the Gauss-Seidel smoother to the original complex symmetric system. This can also be done on the real system by applying a 2×2 block Gauss-Seidel smoother mimicking the complex algebra.

The *SESES* approach to auxiliary preconditioning for this example is as follows. One first assemble and backstore the system matrix for first order elements S_{first} under the name `firstorder` as well as the prolongation matrix Π with name `proj`.

```
MaterialSpec Wire    Equation EddyFreeHarmonicNodal Enable
MaterialSpec AirEddy Equation EddyFreeHarmonicNodal Enable
BlockStruct Block EddyFreeHarmonicNodal ReuseMatrix nodal1
LinearSolver None Reordering ComplexCluster
Solve Stationary
Misc Matrix(Afield; FirstTo2ndOrder; proj)
```

In a second step, one assembles and solves the system for the second order elements by enabling the auxiliary preconditioner composed by some pre- and post- complex Gauss-Seidel smoothing cycles, the auxiliary system matrix `firstorder`, prolongation matrix `proj` and the choice of a solver for the auxiliary system.

```
MaterialSpec Wire    Equation EddyFreeHarmonicNodal2 Enable
MaterialSpec AirEddy Equation EddyFreeHarmonicNodal2 Enable
BlockStruct Block EddyFreeHarmonicNodal
LinearSolver CG InfoIter 0 DiagPrecond Disable Reordering ComplexCluster
  ConvergSolver (SITER=nSolverIter)*0+NormR<1.0E-8*NormR0 MaxIter 500
  Preconditioner Auxiliary(PreSmooth 2; PostSmooth 2; ComplexSmoother;
    Projector Afield.proj,iAfield.proj; Matrix nodal1;
    Precond ILU(*ILU LU*))
Solve Init Solve Stationary
```

The auxiliary problem S_{first}^{-1} can be solved either exactly, inexactly with a nested iterative solver and its own preconditioner or inexactly by just applying once the auxil-

iary preconditioner. We point out that if a nested iterative solver is used, the auxiliary solution is inexact and non-constant and the CG algorithm may not work well anymore. A generalized CG algorithm with additional orthogonalization steps for a non-constant preconditioner matrix can be used to fix the problem, but by our experience, this is not generally required. Smoothing plays an important role as given in Table 2.2, where we have solved the auxiliary problem both with an exact LU and (inexact) $\mathrm{ILU}(0)$ preconditioner and a variable number of pre- and post-smoothing cycles performed before and after solving the auxiliary problem. By applying a sufficient number of pre- and post-smoothing cycles, the number of iterations needed to reach convergence is similar as for first order nodal elements. However, smoothing is quite an expensive numerical operation with each symmetric Guass-Seidel cycle being equivalent to two matrix-vector multiplications but not easily parallelized. A single pre- and post-smoothing cycle looks to be an optimum.

For edge elements we have not yet found a robust preconditioner as for nodal elements of first order. The major reason is that since the system matrix S is singular, the incomplete factorization generally fails. This unfavorable situation can be in part saved by adding a small constant in the diagonal before performing the incomplete factorization. This amendment, should be done on the diagonally scaled matrix i.e. after diagonal preconditioning and the value of 0.05 seems to be an optimum. However, the overall performance and robustness is poor, but get better with increasing frequency. The really bad performance at low frequencies may have its origin on a light inconsistent right-hand side, which is generally the case for $\omega = 0$ and a given stiff current. This failure should be further investigated and a consistent right-hand side should be enforced by finding a field $\mathbf{H}_0$ with $\nabla \times \mathbf{H}_0 = \mathbf{J}_0$ and using the term $\nabla \times \mathbf{H}_0$ in the dicretization instead of the stiff current $\mathbf{J}_0$, [4]

The good numerical results obtained for first order nodal elements and the $\mathrm{ILU}(0)$ preconditioner can be again used to precondition the system matrix obtained for edge elements with the auxiliary preconditioning method exposed above. The only difference is that we do not change the order of the finite element approximation, but the type i.e. from edge elements to nodal elements. The ingredients are all the same, just the prolongation matrix is different and here Π represents the interpolation between nodal and edge elements. This auxiliary preconditioner suffers as the previous auxiliary one whenever we have badly shaped elements and has the same poor performance at low frequencies as the above $\mathrm{ILU}(0)$ preconditioner with diagonal shift. We have used here a variant of the auxiliary preconditioner presented in [5, 6], since this latter fails for the case $\sigma = 0$ and although mathematically quite reliable, on the overall these two auxiliary preconditioners are generally inferior to the simple $\mathrm{ILU}(0)$ preconditioner with diagonal shift.

References

[1] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

[2] W. HACKBUSCH, *Iterative Solution of Large Sparse Systems of Equations*, Applied Mathematical Sciences, 95, Springer Verlag, 1994.

[3] H. A. VAN DER VORST, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, 2003.

[4] Z. REN, *Influence of the R.H.S. on the Convergence Behaviour of the Curl-Curl Equation*, IEEE Trans. Magn, Vol. 32, No. 3, pp 655-658, 1996.

[5] R. HIPTMAIR, J, XU, *Nodal auxiliary space preconditioning in $H(\mathrm{curl})$ and $H(\mathrm{div})$ spaces*, SIAM J. Numer. Anal., Vol. 45, No. 6, pp 2483-2509, 2007.

[6] R. HIPTMAIR, J, XU, *Auxiliary space preconditioning for edge elements*, IEEE Trans. Magn, Vol. 44, No. 6, pp 938-941, 2008.

2.16 Harmonic Analysis of a Piezoelectric Crystal

Piezoelectric crystals find a broad application in technology, from LC oscillators to sensors for pressure, force, acceleration as well as transducers and actuators for ultrasonic applications. The piezoelectric effect is a small physical effect coupling together electric field and mechanical strain. Due to its small perturbation of the state at rest, an analysis based on linear system theory is a good framework which is generally performed in the time Fourier space i.e. the system is fully characterized by determining its response to an external harmonic excitation over the full spectral range. In this example, we present the fundamentals of the impedance theory for piezoelectric crystals together with a computational example.

Impedance of a piezoelectric crystal

Let us assume the crystal to be at rest and in a unstressed state. By applying a pressure somewhere on the crystal surface, a stress state is induced, mechanical displacements are generated and charges are transported to the contacts. Due to the small magnitude of the displacements, a linear theory for the governing equations can be used. If $\mathbf{u}$ is the mechanical displacement and ϕ the electric potential in the crystal, then we have the following local relations between mechanical and electrical fields

$$\begin{array}{rcrcl} \mathbf{s} & = & \mathbf{C}\cdot\boldsymbol{\varepsilon}(\mathbf{u}) & - & \mathbf{g}^{\mathrm{T}}\cdot\mathbf{E}\,, \\ \mathbf{D} & = & \mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}) & + & \boldsymbol{\epsilon}\cdot\mathbf{E}\,, \end{array} \tag{2.74}$$

with $\mathbf{E}=\nabla\phi$ the electric field, $\mathbf{D}$ the dielectric displacement field, $\boldsymbol{\varepsilon}(\mathbf{u})=((\nabla\mathbf{u})^{\mathrm{T}}+\nabla\mathbf{u})/2$ the linearized strain tensor, $\mathbf{u}$ the displacement, $\mathbf{s}$ the stress tensor, $\boldsymbol{\epsilon}$ the dielectric permittivity tensor, $\mathbf{C}$ the elasticity tensor and $\mathbf{g}$ the piezoelectric tensor. The governing equations of a dynamical mechanical system are in the linear theory of small displacements given by $\rho\ddot{\mathbf{u}}+\gamma\dot{\mathbf{u}}=\nabla.\mathbf{s}+\mathbf{f}$, with ρ the mass density, $\mathbf{f}$ the body force and γ a phenomenological damping factor proportional to the velocity $\dot{\mathbf{u}}$. By considering a vanishing body force and electric bulk charge, we have to solve the coupled equations

$$\begin{pmatrix} \rho\ddot{\mathbf{u}}+\gamma\dot{\mathbf{u}} \\ 0 \end{pmatrix} = \begin{pmatrix} \nabla.\mathbf{s} \\ \nabla.\mathbf{D} \end{pmatrix} \text{ or } \begin{pmatrix} \rho\ddot{\mathbf{u}}+\gamma\dot{\mathbf{u}} \\ 0 \end{pmatrix} = D_2 \begin{pmatrix} \mathbf{u} \\ \phi \end{pmatrix}, \tag{2.75}$$

with D_2 a differential second order operator acting on the solution $(\mathbf{u}, \phi)$. In order to get a solution of (2.75) for the material laws (2.74) and a harmonic voltage applied to an electric contact, we use a technique based on the decomposition into homogeneous eigen solutions and an inhomogeneous stationary solution, which prevent us from explicitly integrating (2.75) with respect to the time. As first step, we look for solutions of (2.75) with a time harmonic dependency of the form $\exp(i\omega_n t)$ by using uniquely homogeneous BCs for the mechanical and electrical problem and by considering $\gamma = 0$. Insertion of the generic solution $(\mathbf{u}_n(\mathbf{x}), \phi_n(\mathbf{x}))^{\mathrm{T}} \exp(i\omega_n t)$ into (2.75) leads to the solution of the following eigen problem

$$D_2 \begin{pmatrix} \mathbf{u}_n \\ \phi_n \end{pmatrix} + \omega_n^2 \begin{pmatrix} \rho \mathbf{u}_n \\ 0 \end{pmatrix} = 0\,, \tag{2.76}$$

which has eigen solutions $(\mathbf{u}_n, \phi_n, \omega_n^2)$, $i = 1 \ldots \infty$ with positive eigenvalues ω_n^2 forming a base in the solution space of D_2. By choosing the orthogonality relation with the density ρ as weight, we can therefore write

$$< \mathbf{u}_m, \mathbf{u}_n >= \int_\Omega \rho \mathbf{u}_m \cdot \mathbf{u}_n \mathrm{d}V = \delta_{mn}\,. \tag{2.77}$$

It is to be noted, that the electric eigen solutions do no enter this orthogonality relation. As second step, les us compute the following inhomogeneous solution

$$D_2 \begin{pmatrix} \mathbf{u}_0 \\ \phi_0 \end{pmatrix} = 0\,, \tag{2.78}$$

with the same type of BCs as before but with possibly inhomogeneous values. In particular, we assume the driving voltage to have a value of $1\,\mathrm{V}$. A solution proposal satisfying the same BCs of the inhomogeneous solution, but now with a driving voltage of $V_0 \exp(i\omega t)$ can be given in the form

$$\begin{pmatrix} \mathbf{u} \\ \phi \end{pmatrix} = \begin{pmatrix} \mathbf{u}_0 \\ \phi_0 \end{pmatrix} V_0 e^{i\omega t} + \sum_n x_n(t) \begin{pmatrix} \mathbf{u}_n \\ \phi_n \end{pmatrix}\,, \tag{2.79}$$

with the time-dependent modal participation coefficients x_n to be determined. After inserting the proposal (2.79) into (2.75), taking the scalar product with $\mathbf{u}_m$, assuming the cross coupling coefficients $\int_\Omega \gamma \mathbf{u}_n \cdot \mathbf{u}_m \mathrm{d}V$ to be zero for $m \neq n$, we arrive at the equations for the x_n

$$\ddot{x}_n + \gamma_n \dot{x}_n + \omega_n^2 x_n = V_0 e^{i\omega t}(\Lambda_n \omega^2 - i\gamma_n \omega)\,, \tag{2.80}$$

with

$$\Lambda_n =< \mathbf{u}_0, \mathbf{u}_n >= \frac{\omega^2}{\omega_n^2} \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}_0) \cdot \mathbf{s}(\mathbf{u}_n) \mathrm{d}V \quad \text{and} \quad \gamma_n = \int_\Omega \gamma \mathbf{u}_n \cdot \mathbf{u}_n \mathrm{d}V\,. \tag{2.81}$$

The time-dependent solutions for the coefficients x_n are then given by

$$x_n(t) = V_0(\Lambda_n \omega^2 - i\gamma_n \omega) y_n(\omega) e^{i\omega t} \quad \text{with } y_n(\omega) = \frac{(\omega_n^2 - \omega^2) - i\omega\gamma_n}{(\omega_n^2 - \omega^2)^2 + (\omega\gamma_n)^2}\,. \tag{2.82}$$

The piezo crystal is generally characterized by its impedance $Z(\omega) = V(\omega)/I(\omega) = V(\omega)/\dot{Q}(\omega)$ with V the voltage, I the current and Q the charge at the contact which is given by the surface integral

$$Q = -\int_{\partial\Omega_0} \mathbf{D}\cdot \mathrm{d}\mathbf{n} = -\int_{\partial\Omega_0} (\boldsymbol{\epsilon}\cdot\mathbf{E} + \mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}))\cdot \mathrm{d}\mathbf{n}\,, \tag{2.83}$$

with $\mathbf{n}$ the outward normal to the surface. If we denote by α_0 and α_n the charge of the inhomogeneous stationary solution and of the eigen solutions, then the contact's charge is given by

$$Q = V_0\alpha_0 e^{i\omega t} + \sum_{n=1}^{\infty} \alpha_n x_n(t) = V_0\alpha_0 e^{i\omega t} + V_0 \sum_{n=1}^{\infty} \alpha_n(\Lambda_n\omega^2 - i\gamma_n\omega)y_n(\omega)e^{i\omega t}\,.$$

In the appendix, we prove the equivalency $\alpha_n = \omega_n^2\Lambda_n$ and therefore the impedance for the piezo crystal is given by

$$Z(\omega)^{-1} = i\omega(\alpha_0 + \sum_{n=1}^{\infty}(\frac{\alpha_n^2\omega^2}{\omega_n^2} - i\gamma_n\alpha_n\omega)y_n(\omega))\,. \tag{2.84}$$

In summary, in order to compute the impedance for the piezo crystal, one has first to compute the inhomogeneous solution yielding the charge α_0. Then one computes a representative number of eigenpairs of the coupled system (2.76) and for each pair one evaluates either the volume integrals Λ_n or the surface integral α_n. With the specification of the empirical damping constants γ_n, the impedance (2.84) is fully specified and can be plotted.

Near to a resonance frequency $\omega \approx \omega_n$, in (2.84) we may just keep the dominating n-th term. By considering $\omega/\omega_n \approx 1$ and a zero damping $\gamma_n = 0$, the electric impedance is given by an equivalent circuit with an LC series in parallel with a capacitor C_0 and values $C_0 = \alpha_0$, $L = \alpha_n^{-2}$, $LC = \omega_n^{-2}$. The impedance is therefore given by

$$Z(\omega) = \frac{1}{i\omega C_0}\,\frac{(LC)^{-1} - \omega^2}{\frac{C+C_0}{LCC_0} - \omega^2} = \frac{1}{i\omega\alpha_0}\,\frac{\omega_n^2 - \omega^2}{\omega_n^2 + \frac{\alpha_n^2}{\alpha_0} - \omega^2}\,.$$

Computational example

In this example, we are going to compute the resonance frequency spectra of a PZT-5A piezoceramic disk as published in [1] and the *SESES* input file can be found at `example/PiezoDisk.s2d`. The paper also presents the same impedance theory, although from the computational finite element point of view. In a first step, we define the material `PZT5A`. For this material, we have to enable the computation of the displacement field and electric field, as well as the activation of the piezoelectric model. For this analysis quadratic elements are best suited. In the definition of the material parameters from the literature, one has to pay attention at the different direction of the polarization axis. Since we are performing a rotational symmetric computation, we have also to specify the material parameters for the azimuthal direction, i.e. the ones normal to the computational domain.

```
MaterialSpec PZTA5
  Equation  Elasticity2 ElectroStatic2 Enable
  Model     PiezoElectric Enable
```

In a second step, we define a simple geometry, fix the structure mechanically and define two electric contacts where the external voltage is applied. Because the electric contacts are placed at the top and bottom of the disk, the disk is symmetric with respect to the middle plane and only such modes can be exited. Therefore is enough to compute half of the disk. The modeling starts by defining a coupled block structure both for the inhomogeneous and the eigen solutions

```
BlockStruct Block Phi Disp
```

Afterwards we compute the coupled inhomogeneous solution with the potential set to zero on one contact and to $1\,\mathrm{V}$ on the other one

```
Solve Stationary
Write "alpha0=%e\n" apply.Phi.Flux
```

and compute the total charge α_0 at the contact. This charge is nothing else than the *SESES* contact characteristic and can be accessed with the built-in symbol `apply.Phi.Flux`. To compare the values of the α_n and Λ_n constants, we store the displacement $\mathbf{u}_0$ and strain $\boldsymbol{\varepsilon}(\mathbf{u}_0)$ fields under the name of `Disp0` and `Strain00`, that otherwise would be lost by computing subsequent solutions.

```
Store Disp0=Disp
Store Strain00=Strain
```

Because the electric solution does not participate in norming the solution, it is important to start the eigen solver with a zero electric solution and therefore we initialize everything to zero after computing the inhomogeneous solution.

```
BC apply  Dirichlet Phi  0  V
Solve  Init
```

Afterwards one computes a representative number of eigenpairs

```
Solve EigenProblem nPair npair=48 Precision 1E-30
```

and for each computed eigenpair, we write to the output the value of the mechanical frequency and the α_n values computed with three different integral methods. Note that in computing the domain integral, we have to consider the rotational symmetry and that this postprocessing related to the mechanical modes, must be defined before computing the eigenpairs and it will be applied to each computed eigenpair.

```
Write  AtStep 1
  Text "[%-2.0f] frequency=%10.3e Hz alpha=%10.3e %10.3e %10.3e\n"
    eigennum+1, sqrt(eigenvalue)/2/PI apply.Phi.Flux
    2*PI*integrate(@MechEne(Strain00,Stress)*x)
    2*PI*integrate(@ScalP(Disp0,Disp)*x)*eigenvalue
```

The results are obtained in the form

```
[1 ] frequency= 4.956e+04 Hz alpha= 2.282e+01  2.282e+01  2.282e+01
[2 ] frequency= 1.281e+05 Hz alpha= 2.076e+01  2.076e+01  2.076e+01
[3 ] frequency= 2.016e+05 Hz alpha= 2.069e+01  2.069e+01  2.069e+01
...
[46] frequency= 1.184e+06 Hz alpha= 9.316e+00  9.316e+00  9.315e+00
[47] frequency= 1.211e+06 Hz alpha= 3.597e+00  3.597e+00  3.596e+00
[48] frequency= 1.222e+06 Hz alpha= 1.326e+00  1.326e+00  1.325e+00
```

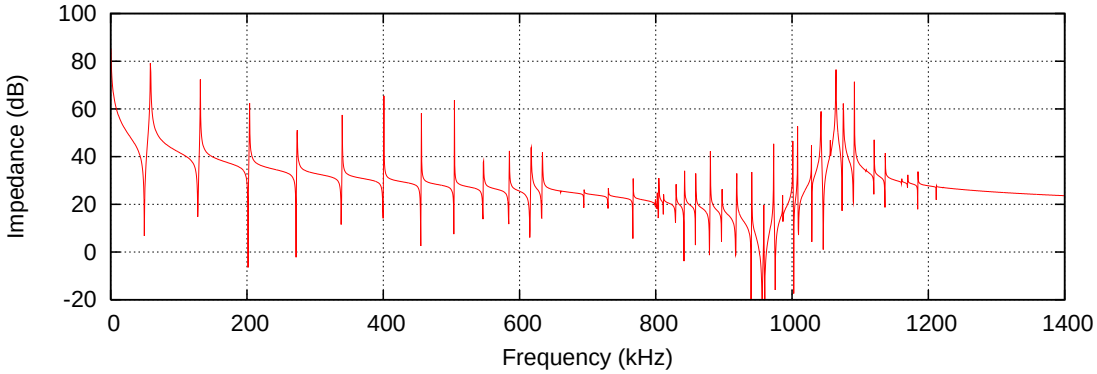

Figure 2.65: The impedance $|Z(\omega)|$ for the PZT5A disk of [1] without damping.

The first 10 frequencies well agree with the ones published in [1], with the higher ones being close within a few percent. The cause is to be found in a too coarse mesh used by [1] and our results well agree with the precise computations done by [2]. The first two versions of the α_n values are the same but just if a small enough precision `Precision 1E-30` is selected when computing the eigenpairs. This value is generally smaller than the one required to achieve the same precision within the eigen frequencies and with respect to this precision the more stable value is the second one. The small discrepancy of the third version for high eigenvalues has origin in a low order integration scheme not suitable for quadratic elements and all three values are the same by increasing the accuracy of the integral's evaluation. Simultaneously to the generation of the above output, we also create the plot file `Plot` with the analytical value of the impedance (2.84) for zero damping used to create Fig.2.65.

Analytical solution

In this section we present a 1D analytical solution and assume the piezo disc to have a thickness L along the z-axis and use the material laws $s = E\,\partial_z u + g\,\partial_z \phi$ and $D = g\,\partial_z u - \epsilon\,\partial_z \phi$ with E the Young's module, g the piezoelectric coupling coefficient and ϵ the dielectric constant. The governing equations (2.77) reduce now to the simple form $\partial_z s = \rho \ddot{u}$ and $\partial_z D = 0$ and we look for a solution with a clamped mechanical BC $u(0) = 0$ at $z = 0$, a stress free BC $s(L) = 0$ at $z = L$, a grounded potential $\phi(0) = 0$ at $z = 0$ and a time harmonic driving potential $\phi(L) = V_0 e^{i\omega t}$ at $z = L$. A straightforward computation shows that the solution is given by

$$\begin{pmatrix} u \\ \phi \end{pmatrix} = \begin{pmatrix} \frac{\epsilon}{g}\,\frac{\sin(\lambda z)}{\sin(\lambda L)+\nu L} \\ \frac{\sin(\lambda z)+\nu z}{\sin(\lambda L)+\nu L} \end{pmatrix} V_0 e^{i\omega t}\,, \tag{2.85}$$

with $\lambda^2 = (\rho/\alpha)\omega^2$, $\alpha = E + g^2/\epsilon$, $\nu = -\lambda\cos(\lambda L)\,(1 + \epsilon E/g^2)$ and the electric impedance reads

$$Z(\omega) = \frac{V_0 e^{i\omega t}}{-i\omega\, D} = \frac{L\,\lambda\,(1+\frac{\epsilon E}{g^2}) - \tan(\lambda L)}{i\omega\,\epsilon\,\lambda\,(1+\frac{\epsilon E}{g^2})}\,. \tag{2.86}$$

We may of course compute the same solution using the inhomogeneous and eigen solutions decomposition as described previously. The eigen solutions $u_n(z)e^{i\omega_n t}$ for the

homogeneous BCs $u(0)=0$, $s(L)=0$ and $\phi(0)=\phi(L)=0$ are computed by solving the equation $\alpha\partial_z^2 u_n+\omega_n^2\rho u_n=0$. The generic solution reads $u_n=k_0\cos(\lambda_n z)+k_1\sin(\lambda_n z)$ with $\lambda_n^2=(\rho/\alpha)\omega_n^2$. The potential ϕ_n is found from the condition $D=\text{const}$ and reads $\phi_n=g/\epsilon u_n+k_2 z+k_3$. By applying the homogeneous BCs, we obtain

$$\begin{pmatrix} u_n \\ \phi_n \end{pmatrix} = A_n \begin{pmatrix} \sin(\lambda_n z) \\ \frac{g}{\epsilon}(\sin(\lambda_n z)-\frac{z}{L}\sin(\lambda_n L)) \end{pmatrix},$$

with λ_n, $n=1\ldots\infty$ the positive solutions of the equation

$$(1+\frac{\epsilon E}{g^2})\lambda_n L-\tan(\lambda_n L)=0\,.$$

The constants A_n are choosen so that the functions are orthonormal $<u_n,u_m>=\delta_{nm}$ and so $A_n=1/\sqrt{<\sin(\lambda_n z),\sin(\lambda_n z)>}$. The inhomogeneous solution with BCs $u(0)=0$, $s(L)=0$, $\phi(0)=0$ and $\phi(L)=1$ is given by

$$\begin{pmatrix} u_0 \\ \phi_0 \end{pmatrix} = \frac{z}{L}\begin{pmatrix} -g/E \\ 1 \end{pmatrix}, \tag{2.87}$$

so that by considering zero damping, the solution (2.79) is given by

$$\begin{pmatrix} u \\ \phi \end{pmatrix} = \left[\begin{pmatrix} u_0 \\ \phi_0 \end{pmatrix} + \sum_{n=1}^{\infty}\frac{\omega^2<u_0,u_n>}{\omega_n^2-\omega^2}\begin{pmatrix} u_n \\ \phi_n \end{pmatrix}\right] V_0 e^{i\omega t}\,, \tag{2.88}$$

and the impendance by

$$Z(\omega)=\frac{V_0 e^{i\omega t}}{-i\omega D}=\frac{L}{i\omega}\left(\frac{g^2}{E}+\epsilon-g\sum_{n=1}^{\infty}\frac{\omega^2}{\omega_n^2-\omega^2}<u_0,u_n>A_n\sin(\lambda_n L)\right)^{-1},$$

which is the same as (2.86) although this is not easily seen except for the case $\omega=0$.

Appendix

We want to prove here the equivalency of $\alpha_n=\omega_n^2\Lambda_n$. We first note that if we solve for $\nabla.\mathbf{F}=0$ in Ω using a Dirichlet BC on $\partial\Omega_0$ and natural BC otherwise, then we have $\int_\Omega \nabla f\cdot\mathbf{F}\mathrm{d}V=0$ for any function f zero on $\partial\Omega_0$ since $\int_{\partial\Omega} f\mathbf{F}\cdot\mathrm{d}\mathbf{n}=0$. From (2.78), we are actually solving $\nabla.(\mathbf{C}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_0)-\mathbf{g}^{\mathrm{T}}\cdot\mathbf{E}_0)=0$ and $\nabla.(\boldsymbol{\epsilon}\cdot\mathbf{E}_0+\mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_0))=0$ and therefore for $f=\mathbf{u}_n$ we have $\int_\Omega\boldsymbol{\varepsilon}(\mathbf{u}_n)\cdot(\mathbf{C}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_0)-\mathbf{g}^{\mathrm{T}}\cdot\mathbf{E}_0)\mathrm{d}V=0$ and for $f=\phi_n$ we obtain $\int_\Omega\mathbf{E}_n\cdot(\boldsymbol{\epsilon}\cdot\mathbf{E}_0+\mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_0))\mathrm{d}V=0$. Because $\phi_0=1$ on $\partial\Omega_0$ and $\nabla.((\boldsymbol{\epsilon}\cdot\mathbf{E}_n+\mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_n))=0$ on Ω, we can write $\alpha_n=-\int_{\partial\Omega_0}(\boldsymbol{\epsilon}\cdot\mathbf{E}_n+\mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_n))\phi_0\cdot\mathrm{d}\mathbf{n}=-\int_\Omega\nabla.((\boldsymbol{\epsilon}\cdot\mathbf{E}_n+\mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_n))\phi_0)\mathrm{d}V=\int_\Omega(\boldsymbol{\epsilon}\cdot\mathbf{E}_n+\mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_n))\cdot\mathbf{E}_0\mathrm{d}V=\int_\Omega(-\mathbf{E}_n\cdot\mathbf{g}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_0)+\boldsymbol{\varepsilon}(\mathbf{u}_n)\cdot\mathbf{C}\cdot\boldsymbol{\varepsilon}(\mathbf{u}_0))\mathrm{d}V=\int_\Omega\mathbf{s}(\mathbf{u}_n)\cdot\boldsymbol{\varepsilon}(\mathbf{u}_0)\mathrm{d}V=\omega_n^2<\mathbf{u}_n,\mathbf{u}_0>=\omega_n^2\Lambda_n$.

References

[1] N. Guo, P. Cawley, D. Hitchings, *The finite element analysis of the vibration characteristics of piezoelectric discs*, Journal of Sound and Vibration, Vol. 159, pp 115-138, 1992.

[2] J. KOCBACH, P. LUNDE, M. VESTRHEIM, *Resonance frequency spectra with convergence tests for piezoceramic disks using the Finite Element Method*, Acustica, Vol. 87, pp. 271-285, 2001.

2.17 Longitudinally Diode Pumped Composite Laser Rod

The following examples deal with solid state laser devices. A good introduction into this field including set ups and pumping techniques is e.g. given in [1, 2, 3]. When a solid state laser crystal is optically pumped by flash lamps or diode lasers only a part of the pump light is converted to laser- or other radiation. The other parts acts as a veritable space resolved heat source leading to a spatially varying temperature distribution in the crystal. This temperature distribution will influence the laser behavior. The three main effects are

- *Thermal Dispersion*: Due to the dependency of the refractive index of a material on the temperature, the laser beam may be disturbed by its travel through the crystal.
- *Deformation*: Due to the temperature raise, the crystal will expand locally leading to deformation of the crystal surface. The deformed surfaces will have a lens like influence onto the laser beam traveling through the crystal.
- *Stress and strain*: Based on the local deformation stress and strain will occur in the crystal. This may lead on the one hand to mechanical cracking of the crystal and on the other hand to stress induced birefringence.

All these effects have a strong influence onto the behavior of a laser device by reducing the output power and the beam quality. Methods have to be found to reduce these effects as e.g. adequate resonator design, improved cooling techniques or compensation by external elements. Therefore a laser engineer must have the possibility to quantify the influence of different technical improvements onto these effects for estimating the laser performance during the development process.

Unfortunately analytical solutions only exist for some special cases as uniform pumping [1]. For real devices, we have to resort to numerical methods. With *SESES* its possible to deduce both the distribution of temperature, stress, strain and crystal deformation and to quantify its influence by calculating the Optical Path Difference (OPD) and the birefringence.

To start a numerical simulation, we have first to know the pump power distribution $p(\mathbf{x})$ which acts at the same time as heat source in the crystal. To get the temperature distribution, the steady state heat transfer equation has to be solved by considering the cooling boundary conditions at the crystals surface. Here we are concerned with Newton's law of heat transfer

$$\kappa \left.\frac{\partial T}{\partial n}\right|_S = h\left(T_C - T|_S\right) , \tag{2.89}$$

with κ the heat conductivity, T_C the coolant temperature assumed to be constant, $T|_S$ the local crystal temperature on the surface S, $\mathbf{n}$ the local normal to the surface and h

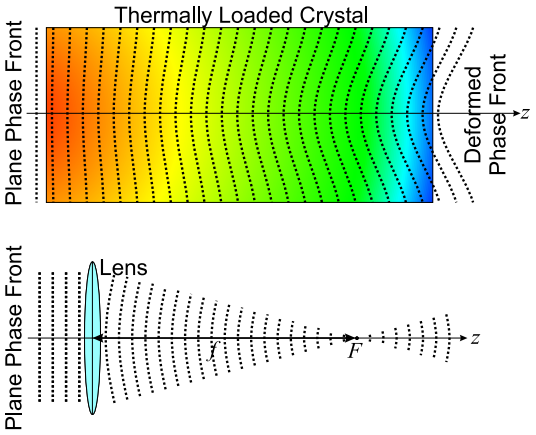

Figure 2.66: Lens like influence of a thermally loaded laser crystal onto a traveling plane phase front.

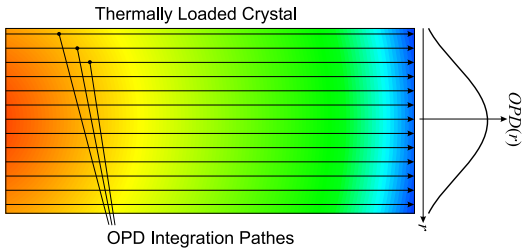

Figure 2.67: $\mathrm{OPD}_{\mathrm{th}}$ integrated along different paths parallel to the laser beam axes as a function of the distance r to this axes.

the heat transfer coefficient [4]. This boundary condition includes the cases of constant boundary temperature ($h \to \infty$), insulated boundaries ($h \to 0$) and convective heat transfer (finite h). Based on the computed temperature distribution $T(\mathbf{x})$, the local displacement field $\mathbf{u}(\mathbf{x})$ together with the corresponding stress tensor is calculated. As the local deformations are expected to be small, the two fields $T(\mathbf{x})$ and $\mathbf{u}(\mathbf{x})$ can be treated as uncoupled.

Optical Path Difference (OPD)

The refractive index of a material depends on its temperature. This effect is known as thermal dispersion and the change of the refractive index Δn as a function of the local temperature can be written as

$$n - n_0 = \Delta n = \int_{T_0}^{T} \frac{\partial n}{\partial T}\, \mathrm{d}T\,, \tag{2.90}$$

with T_0 the reference temperature where the refractive index n_0 of the material is defined and $\partial n/\partial T$ the thermal dispersion coefficient. Because the speed of light in a material directly scales with the refractive index following $c' = c/n$, a plane wave entering a thermally loaded crystal is disturbed during its travel through the crystal as shown in the upper part of Fig. 2.66. This situation can be compared to the phase front deformation caused by an ideal thin lens shown in the lower part of Fig. 2.66.

A valuable means to quantify this phase front deformation is the OPD. Let us assume a homogenous material of length ℓ with the refractive index n_0. The time a light ray needs to travel through the crystal is given by

$$\Delta t = \frac{\ell\, n_0}{c}\,.$$

Within the same time, the light would travel a distance of $c\,\Delta t = n_0\,\ell$ in free space. We therefore call $\ell_{\mathrm{opt}} = n_0\,\ell$ the optical length of the material. In our case the refractive

index of the crystal depends on the local temperature and therefore varies along the beam path following (2.90). We call this the disturbed case and the corresponding optical length ℓ'_{opt}. The difference between the disturbed and the undisturbed case $\ell'_{\mathrm{opt}} - \ell_{\mathrm{opt}}$ is the so called thermal OPD given by

$$\mathrm{OPD}_{\mathrm{th}} = \ell'_{\mathrm{opt}} - \ell_{\mathrm{opt}} = \int_0^{\ell} \Delta n(z)\,\mathrm{d}z = \int_0^{\ell} \left(\int_{T_0}^{T(z)} \frac{\partial n}{\partial T}\, dT \right) \mathrm{d}z\,. \tag{2.91}$$

This integral is evaluated along different paths parallel to the laser beam axes in the thermally loaded crystal, as shown in Fig. 2.67. This leads to the $\mathrm{OPD}_{\mathrm{th}}(r)$ as a function of the distance r to this axes.

Another lens like effect, the end effect, is introduced by the bending of the front and back surface of the crystal. This influence is summarized in the end effect optical path difference $\mathrm{OPD}_{\mathrm{end}}$ and it is given by

$$\mathrm{OPD}_{\mathrm{end}}(r) = \int_0^{\ell} (n_0 - 1)\sqrt{\mathbf{n}\cdot\boldsymbol{\varepsilon}\cdot\mathbf{n}}\,\mathrm{d}z\,, \tag{2.92}$$

with $\boldsymbol{\varepsilon}$ the mechanical strain tensor and $\mathbf{n}$ the normal vector tangent to the integration path. In the case where the crystal surface acts at the same time as resonator mirror, the factor $(n_0 - 1)$ in (2.92) has to be replaced by n_0.

The sum $\mathrm{OPD}_{\mathrm{tot}} = \mathrm{OPD}_{\mathrm{th}} + \mathrm{OPD}_{\mathrm{end}}$ can be compared to the $\mathrm{OPD}_{\mathrm{lens}}$, produced by an ideal thin lens of focal length f_{lens} given by

$$\mathrm{OPD}_{\mathrm{lens}}(r) = \mathrm{OPD}_0 - \frac{r^2}{2f_{\mathrm{lens}}}\,. \tag{2.93}$$

The thermal lensing effect including both the thermal dispersion and end effect OPD can in first order be described by the focal length of an averaged thermal lens. The value of the focal length is obtained from a parabolic least square fit, using (2.93), to the calculated $\mathrm{OPD}_{\mathrm{tot}}(r)$.

Depending on the shape of the pump distribution, almost arbitrary OPD profiles are possible. Especially for longitudinal pumping the thermal load is very inhomogeneous and the radial dependence of the OPD is far from being parabolic. The focal length of the fitted lens is therefore not the same, whether the fit is performed over the whole rod radius or just over a part of it. For the reminder of this example, the fit was extended over one pump spot radius, as the laser beam is usually adjusted to meet its size. Fig. 2.68 shows this situation.

Laser set up

This example presents the simulation of a longitudinally diode pumped laser rod with undoped end-caps, used in novel laser devices. The *SESES* input file can be found at `example/LongRod.s2d`. Fig. 2.70 shows a X-folded resonator as it is used to realize diode pumping from both sides. The pumping radiation is focused through the two

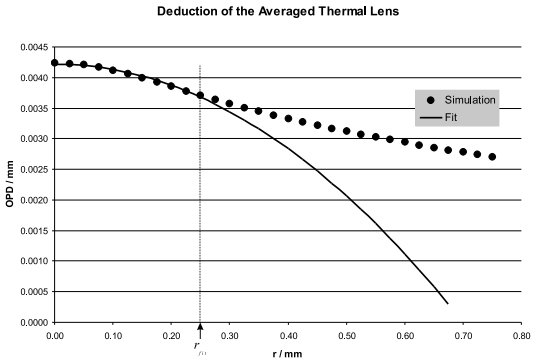

Figure 2.68: Parabolic fit to the deduced $\mathrm{OPD}_{\mathrm{tot}}$ over the extent r_{fit}.

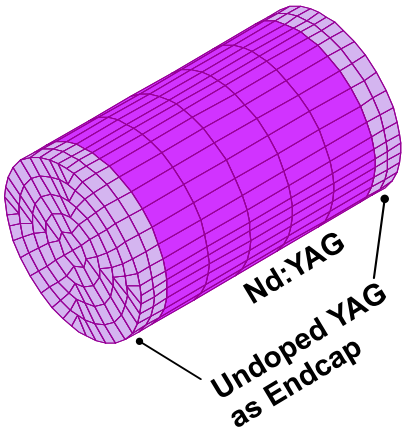

Figure 2.69: Nd:YAG laser rod with two undoped end caps.

folding mirrors directly into both sides of the laser rod as shown in Fig. 2.71. The folding mirrors have to be dichroic coated, e.a. highly reflective for the laser wavelength and highly transmittive for the pump wavelength. The laser rod consists of a 1.1% Nd doped YAG center part and two undoped end-caps as shown in Fig. 2.69. Without end-caps, longitudinal pumping leads to high temperatures and stresses on the pumped surface going with a significant bending of it [5]. Undoped end-caps significantly reduce temperatures and stresses on the pumped surface and prevent surface bending.

The heat source $q(r,z)$ directly scales with the normalized but arbitrary shaped pump distribution $p(r,z)$. In the present case of longitudinal pumping, the pump radiation is absorbed along the optical z-axes in the rod. Further, assuming the paraxial approximation, the absorbed part only depends on the distance z to the rod surface and the absorption coefficient α of the rod. These simplifications lead to the heat source

$$q(r,z) = \eta\, P_0\, p(r,z)\, \alpha\, e^{-\alpha z}\,, \tag{2.94}$$

with P_0 the total incident pump power and η the fraction of pump power converted to heat here. In this example we consider $\eta = 0.4$. For a focused laser beam, the radius at the distance z from the beam waist w_0 reads

$$w(z) = w_0 \sqrt{1 + \left(M^2 \frac{\lambda}{\pi\, w_0^2} \frac{z}{n} \right)^2}\,, \tag{2.95}$$

with λ the pump beam wavelength, M^2 the beam quality factor and n the refractive index of the crystal. For simplicity, we assume the pump distribution to be Gaussian shaped so that $p(r,z)$ reads

$$p(r,z) = \frac{2}{\pi\, w(z)} \exp\left(-\frac{2r^2}{w(z)^2} \right)\,, \tag{2.96}$$

with $w(z)$ defined by (2.95).

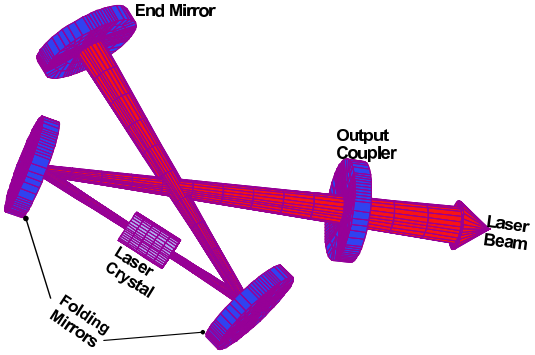

Figure 2.70: Laser rod in a x-folded resonator.

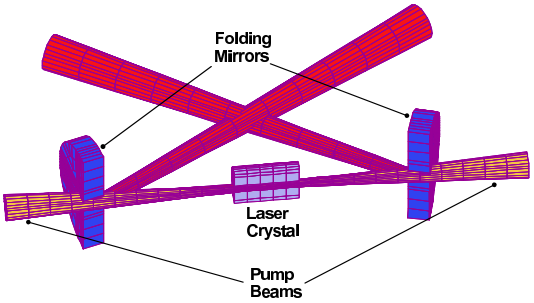

Figure 2.71: Pumping of the laser rod through the two folding mirrors.

Model Specification

To keep the example as simple as possible, we perform a 2D rotational symmetric simulation which is enabled with the statement

```
GlobalSpec Model AxiSymmetric Enable
```

Afterwards some geometrical parameters for the laser rod are defined followed by the relevant pump beam parameters relating to the models (2.94), (2.95) and (2.96).

```
Define
P      = 20                          (* pump power / W                    *)
w0     = 0.250*mm                    (* pump beam focus radius / m        *)
M      = 55                          (* pump beam quality factor          *)
lambda = 0.000809*mm                 (* pump beam wavelength / m          *)
ref    = 1.82                        (* crystal refractive index          *)
alpha  = 0.35/mm                     (* crystal abs. coefficient / 1/m    *)
P_Max  = 20                          (* maximum pump power / W            *)
P_n    = 10                          (* pump power steps                  *)
conv   = 0.4                         (* fraction converted to heat        *)
```

It is mentioned here, that the pump power P denotes the power per pumped side of the crystal. The total pump power double side pumping therefore amounts $2P$. In the present example, we will perform a parameter study with increasing pump power. The front and the back side of the rod are in contact with air, whereas the cylindrical surface is assumed to be directly cooled by water. The heat flow through the surface S of the rod defines the boundary condition. It is given by Newton's law of heat transfer (2.89) and the corresponding parameters h and T_C for water and air are defined as

```
hair = 0.0015                        (* heat tr. coefficient to air       *)
                                     (* in W/(cm**2*K)                    *)
hwat = 1.0                           (* heat tr. coefficient to water     *)
                                     (* in W/(cm**2*K)                    *)
Tair = 293                           (* temperature of the surrounding air *)
Twat = 290                           (* temperature of the cooling water  *)
```

The section `OPD calculation` concerns the parameters for the OPD evaluation. As shown in Fig. 2.67 the OPD has to be evaluated along predefined paths by calculating the integral (2.91) and equation (2.92). The number of paths parallel to the optical axis of the crystal is defined with the parameter `nrOPD`. These paths are equally spaced perpendicular to the axis from it to the distance `rOPD`. Each path starts at the longitudinal coordinate `lOPD0`, has a length of `lOPD` and contains `nlOPD` integration points for the OPD evaluation.

Next the material parameters of Nd:YAG (`NdYAG`) and undoped YAG (`Cap`) are specified.

```
MaterialSpec Cap
  Equation ThermalEnergy Elasticity Enable
  Parameter StressOrtho    LinElastIso(Emodule 307 GPa; PoissonR  0.3;
    AlphaIso -1.78e-6*(Temp-Tair)+1.65e-8*(Temp*Temp-Tair*Tair)) SIunit
  Parameter Refrac         1.82
  Parameter KappaIso D_Temp  1.9e8/pow(log(5.33*Temp),7.14)-331e2/Temp,
                             331e2/(Temp*Temp)-(1.9e8*7.14)/
                             (Temp*pow(log(5.33*Temp),8.14)) W/(m*K)-W/(m*K**2)

MaterialSpec NdYAG From Cap
  Parameter Heat conv*P*(Pump(x,y,w0,M,lambda,alpha,ref)
                       + Pump(x,-y+l,w0,M,lambda,alpha,ref)) W/m**3
```

We start with YAG and since we model both for the temperature and the mechanical displacement, we have to enable their computation with the `Equation` statement. The heat conductivity κ and the thermal expansion $\delta\ell/\ell_0$ are functions of the temperature T and following [6] we use the models

$$\kappa = \frac{1.9\times10^8}{(\log(5.33\,T))^{7.14}} - \frac{331\times10^2}{T}\,\frac{\mathrm{W}}{\mathrm{mK}^2}\,, \tag{2.97}$$

and

$$\frac{\delta\ell}{\ell_0} = -1.78\times10^{-6}(T-T_0) + 1.65\times10^{-8}\,(T^2-T_0^2)\,, \tag{2.98}$$

with $[T] = K$. The parameters `Emodule`, `PoissonR`, and `Refrac` denote Young's modulus E, the Poisson's ratio ν and the refractive index n_0. The parameter `DRefrac` denotes the change of the refractive index Δn as a function of the temperature following (2.90). Here, we assume the thermal dispersion $\partial n/\partial T$ to be a constant so that (2.90) can be rewritten as $\Delta n = (T-T_0)\partial n/\partial T$. Next we define the Nd:YAG material as inheriting all properties from YAG and by additionally defining the heat source. The routine `Pump` is a user routine defined following (2.94), (2.95) and (2.96). The first call is for the pumping from the left side and the second call for the pump beam from the right side of the rod. We then build the macro element mesh and map two materials onto the macro element mesh.

The thermal and mechanical boundary conditions are then next. The front and the back surface of the rod are in contact with air, defined by the boundary condition `BC Air`. The cylindrical surface of the rod is assumed to be in direct contact with the cooling water define by the boundary condition `BC Water`. As we have assumed a radial symmetric situation, the longitudinal axis of the crystal is automatically fixed for deformations in the radial direction. Further we assume the rod to be free to expand. Therefore we have only to fix one point of the rod for displacements in the longitudinal direction. Therefore we fix the point in the center of the rod by the boundary condition

```
BC Fixed  0 nytot/2 JType 0 Dirichlet Disp.Y 0 m
```

The command section of the input file starts with some settings concerning the used solver and desired information on the output stream during execution of the kernel program. Then several one-dimensional lattices for the OPD evaluation are defined with a `For` loop statement. Here we define straight lines along the pump lasing direction where the local OPD will be integrated.

```
For k From 0 To nrOPD
{
   Lattice OPD_@k Index i=0..nlOPD (@k*rOPD/nrOPD,lOPD0)+(0,lOPD)*i/nlOPD
}
```

The lattice names are numbered with the loop index `k` by `OPD_@k`, afterwards follows the definition of the lattice index used to define the lattice points as equally distributed along the lattice line. The corresponding parameters `nrOPD` and `nlOPD` were previously defined in the initial section. Before starting the simulation, the field `Temp` has to be set to the temperature value of the surrounding air. Without the statement

```
Solve Init Temp=Tair
```

the *SESES* kernel will initialize the temperature to $0\,\mathrm{K}$ which is disadvantageous for the simulation time and convergence. The next section specifies the solution algorithm. In this example, we perform a parameter study for different pump powers and the solution procedure looks as follows

```
For i From 1 To P_n
{
   while (1)
   {
      Solve Stationary ForSimPar P At @i*P_Max/P_n
      Store StressAver(Continuous 1;FreeOnRef)=Stress,
            TempDiffAver(Continuous 1;FreeOnRef)=TempDiff
      Remesh Refine ErrTemp(TErr)  Global Refine ErrDisp(MErr)  Global
      if (REMESH) break;
   }
   Write File "OPD_"@@{i}".dat" Text
   For k From 0 To nrOPD
   {
      "X-coord %e OPDtherm %e OPDexpan %e OPDtot %e\n" @k*rOPD/nrOPD,
        res0=integrate(Lattice OPD_@k; 7.3e-6*(Temp-Tair)),
        res1=integrate(Lattice OPD_@k; Refrac*sqrt(Strain.YY*Strain.YY)),
        res0+res1
   }
   Dump Temp Disp Stress If @i>1 { Append }
}
```

The two fields `Temp` and `Disp` are assumed bo be uncoupled and therefore they can be computed separately. For each field the corresponding `BlockStruct` and the automatic mesh refinement is set. Afterwards the field is calculated with the `Solve Stationary` statement. The corresponding pump power is set in the first `Solve` statement by the expression `ForSimPar P`. For each computed solution, we then integrate the local OPD on the defined lattices by computing the contributions $\mathrm{OPD}_{\mathrm{th}}$, $\mathrm{OPD}_{\mathrm{end}}$ and their sum $\mathrm{OPD}_{\mathrm{tot}} = \mathrm{OPD}_{\mathrm{th}} + \mathrm{OPD}_{\mathrm{end}}$. Finally, the mesh, the computed fields `Temp` and `Disp` and the derived field `Stress` are written to a data file for graphical visualization.

Numerical Results

We have performed a parameter study for pump powers up to $20\,\mathrm{W}$ per pumped side of the crystal in ten steps, with a pump beam waist radius of $w_0 = 250\,\mu\mathrm{m}$ and a M^2 factor of 55. The pump beam wavelength amounts to $809\,\mathrm{nm}$, the absorption coefficient of the YAG crystal is $3.5\,\mathrm{cm}^{-1}$ and 40% of the pump power is assumed to be converted to heat. The corresponding heat distribution for a pump power of $20\,\mathrm{W}$

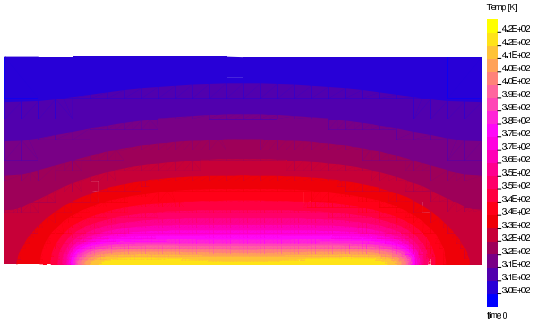

Figure 2.72: Temperature distribution in the laser rod at a pump power of 20 W per side. The temperature scale is given in K.

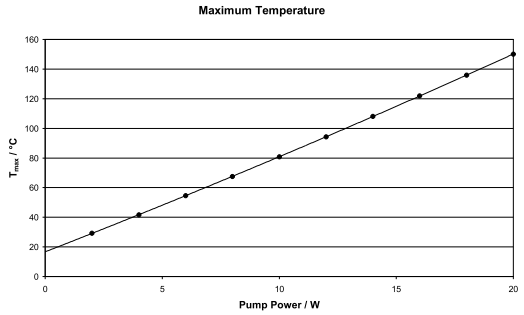

Figure 2.73: Maximum temperature in °C for the pump power range per side from 2 to 20 W.

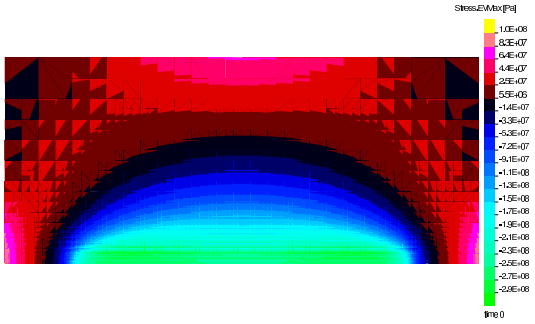

Figure 2.74: Distribution of the highest principal stress component.

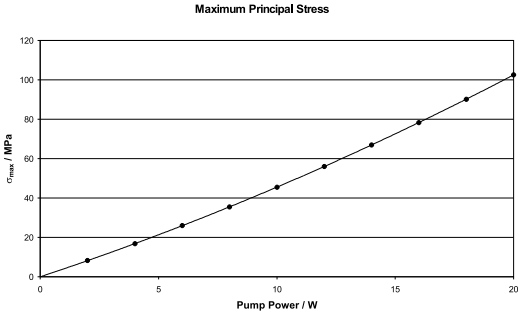

Figure 2.75: Maximum tensile stress in dependence of the pump power per side.

per side is shown in Fig. 2.72. The maximum temperature is located inside the crystal and amounts to 150 °C. Fig. 2.73 shows this maximum temperature for the whole pump power range from 2 to 20 W. Despite the non-linear material parameter (2.97) the raise of the maximum temperature goes linear with the pump power.

As mentioned at the beginning of this example, a too high stress may mechanically crack the crystal. The stress forms a second rank tensor. With suitable coordinate transformations the tensor can be diagonalized, i.e. the stress components are reduced to the principal stress components. Positive and negative values denote tensile and compressive stress, respectively. The largest positive value of the three remaining diagonal elements therefore denotes the maximum tensile stress, which is a valuable indicator for thermal crystal damage. Typical fracture limits for Nd:YAG are reported in the literature and confirmed by the author to range from 130 to 260 MPa. Fig. 2.74 shows the distribution of the highest principal stress component. It is obvious the the inner part of the rod is under compression whereas the outer part is under tension. The maximum tensile stress is located on the rod axes at the surfaces of the crystal. Fig. 2.75 shows this maximum tensile stress as a function of the pump power. It raises with the pump power up to about 100 MPa at 20 W per side, which is below the fracture limit. Due to the non linear thermal expansion (2.98) the maximum tensile stress shows a small nonlinearity with respect to the pump power per side.

The focal length of the averaged thermal lens, following (2.93), is deduced from the

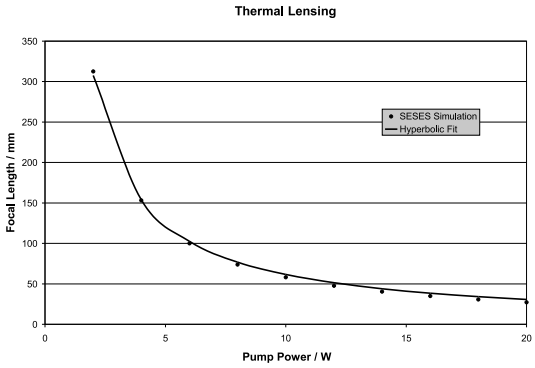

Figure 2.76: Focal length of the averaged thermal lens as a function of the pump power per side. Dots indicates the values obtained from the OPD data whereas the solid line represents the hyperbolic least quare fit to this data.

OPD-data and plotted in Fig. 2.76. The figure indicates that the focal length of the averaged thermal lens has a hyperbolic dependence on the pump power of the form $f_{\mathrm{av}} = F_{\mathrm{sp}}/P_{\mathrm{pump}}$ with F_{sp} the specific focal length. For this example, the specific focal length is $F_{\mathrm{sp}} = 615\,\mathrm{mmW}$.

References

[1] W. KOECHNER, *Solid-State Laser Engineering*, Springer-Verlag, 5^{th} Ed., pp. 407-468, 1999.

[2] TH. HALLER, H.U. SCHWARZENBACH, G. STEINER, *Finite Element Model Based Product Development with NM* SESES, Orell Füssli Verlag AG, pp. 71-102, 2002.

[3] J. EICHLER, H.J. EICHLER, *Laser*, Springer-Verlag, 3^{rd} Ed., 1998.

[4] W. KOECHNER, *Absorbed Pump Power, Thermal Profile and Stresses in a cw Pumped Nd:YAG Crystal*, Appl. Opt., Vol. 9, No. 6, pp. 1429-1434, 1970.

[5] R. WEBER, B. NEUENSCHWANDER, M. MAC DONALD, M.B. ROOS, H.P. WEBER, *Cooling Schemes for Longitudinally Diode Laser-Pumped Nd:YAG Rods*, IEEE J.Quantum Electron., Vol. 34, No. 6, pp. 1046-1053, 1998.

[6] D.C.BROWN, *Nonlinear Thermal and Stress Effects and Scaling Behavior of YAG Slab Amplifiers*, IEEE J. Quantum Electron., Vol. 34, No. 12, pp. 2393-2402, 1998.

2.18 Thin-Disk Laser

Most diode pumped solid state lasers (DPSSL) suffer from strong thermal lensing effects, especially at high pump powers. The strong thermal lens is mainly induced by high radial temperature gradients normal to the optical axis in the laser crystal. A very efficient way to overcome this drawback is offered by the thin-disk laser concept.

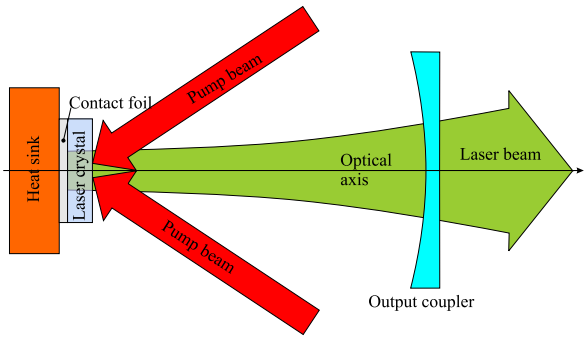

Figure 2.77: Optical scheme of a simple thin-disk laser with quasi longitudinal pumping of the crystal.

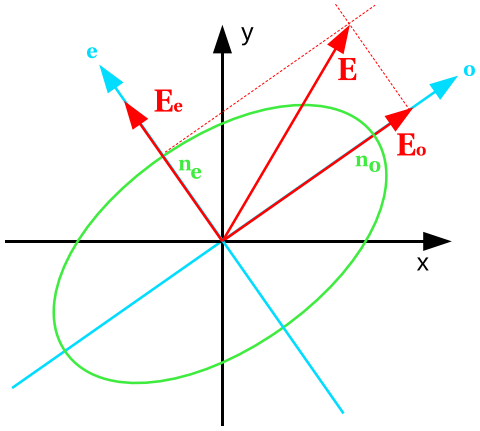

Figure 2.78: Representation of the electric field $\mathbf{E}$ in direction of the principal axes of the refractive index ellipsoid $\mathbf{E}_o$ and $\mathbf{E}_e$.

There, a crystal disk with a thickness smaller than the diameter of the disk is mounted with one of its faces on a heat think, see Fig. 2.77 which is high-reflectivity (HR) coated for both the laser and the pump wavelength. Therefore the excess heat is removed via this face. If we assume a large heat transfer coefficient over the hole area, a temperature field will be established in the crystal with the isotherms essentially normal to the optical axis resulting in significantly reduced temperature gradients in this direction. The disk can efficiently be pumped by laser diodes in a quasi-longitudinal scheme. To increase the absorbtion of the pump radiation for a given thickness, multiple passes of pump radiation through the disk must be used. This disk fits into the resonator as an end mirror or as folding mirror. The design is suitable for lasers in the power range of several watts to kilowatts. The power can be scaled by increasing the pumped diameter of the disk at constant pump power density and/or by using more than one disk. By employing the appropriate resonators and efficient cooling technologies, it is possible to achieve a high beam quality and a high efficiency simultaneously. This design is particularly well suited for quasi-three-level systems such as Yb:YAG, because a low mean temperature and a high pump power density are necessary for efficient operation. Therefore Yb:YAG was chosen as the preferred active medium but the thin-disk concept is also suitable for a variety of other laser materials as e.g. Nd:YAG, Nd:YVO_4 and Tm:YAG. Recently the thin-disk lasers of very high beam quality band high output powers became industrial available. More literature about this concept can be found in [1, 2, 3]. In the present example, we will explore the benefits of the thin-disk laser concept and will analyze the stress induced birefringence called photo-elastic-effect.

Birefringence

Birefringence is an involved matter and therefore we abstain from a detailed description, an introduction into this subject can be found in [4, 5]. Here we focus mainly on the effect of birefringence induced depolarization. Fig. 2.78 shows the situation of a linear polarized plane wave entering a depolarizing element. The refractive index

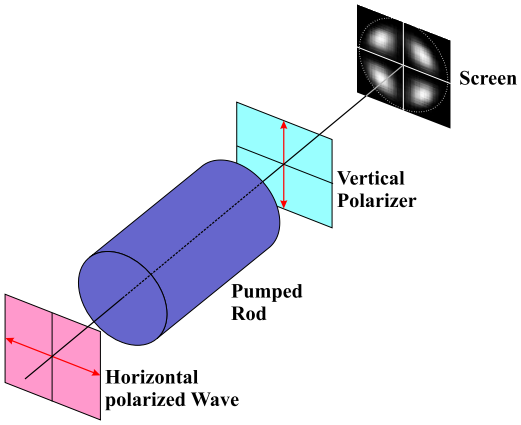

Figure 2.79: Set-up to analyze depolarization.

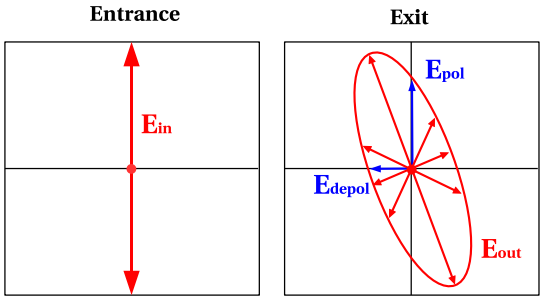

Figure 2.80: Depolarization by birefringence. The entrance wave is linear polarized whereas the exit wave shows elliptic polarization.

is not uniform and is represented by an ellipsoid with the principle axes o and e. In general, these principle axes do not coincide with the x- and y- axis of the laboratory system. The electric field vector $\mathbf{E}$ can be represented by its components $\mathbf{E}_o$ and $\mathbf{E}_e$ in directions of the principle axes of the ellipsoid. The refractive indices n_o and n_e for these two components are different, and one component of the field vector $\mathbf{E}$ is retarded with respect to the other when passing through this element. Therefore, depending on n_o and n_e and the thickness of this element the polarization of this wave is changed. When not a special thickness is chosen ($\lambda/2$ or $\lambda/4$ plate), the wave is elliptically polarized after this element, see Fig. 2.80. The field vector of this exit wave decomposes into a component in the primary direction E_{pol} and into the perpendicular component E_{depol}. The latter gives the amount of the wave which is depolarized by the element. Anisotropic laser crystals as e.g. Nd:YLF show a natural birefringence, in contrast to isotropic crystals as Nd:YAG. This situation changes when the crystal is heated. The thermal expansion leads to stresses and a local deformation of the crystals lattice which leads to spatially varying stress induced birefringence, called photoelastic effect [6]. The optical anisotropic activity of a material is generally described by the dielectric impermeability tensor $\mathbf{B}^0$ which is the inverse of the permittivity tensor ϵ. If the medium is isotropic, we then have $\mathbf{B}^0 = n_0^{-2}\mathrm{Id}$ with n_0 the refractive index. Under a deformation given by the strain $\boldsymbol{\varepsilon}$, the dielectric impermeability is changed due to the photoelastic effect according to [6]

$$\mathbf{B} = \mathbf{B}^0 + \mathbf{P} \cdot \boldsymbol{\varepsilon}\,, \tag{2.99}$$

with $\mathbf{P}$ the photoelastic tensor. To investigate the depolarization effects of strain induced birefringence, we follow the change of a plane wave $\mathbf{E}_0(z)\exp(-i(\omega t - kz))$ traveling in the z-direction which to a first order is given by

$$\frac{\partial \mathbf{E}_0(z)}{\partial z} = \frac{2\pi i}{\lambda}(\mathbf{B}^{-1/2} - n_0\,\mathrm{Id})\,. \tag{2.100}$$

This equation is then integrated along the ray path and this is done by the post-processing routine `ApplyPol` for a small increment of dz.

The observation of the depolarized part of a linear polarized plane wave, traveling through a pumped rod, is one possibility to analyze stress induced birefringence. The

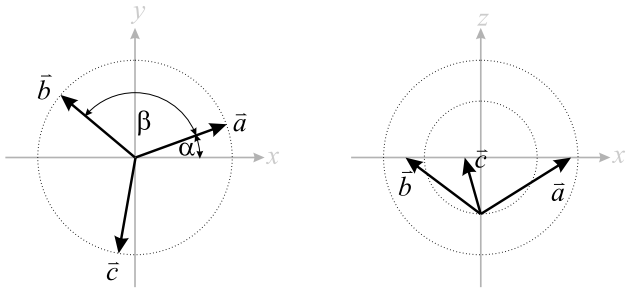

Figure 2.81: Top and side view of the crystal lattice axis (**a**, **b**, and **c**) in case of a [1,1,1] cut cubic crystal.

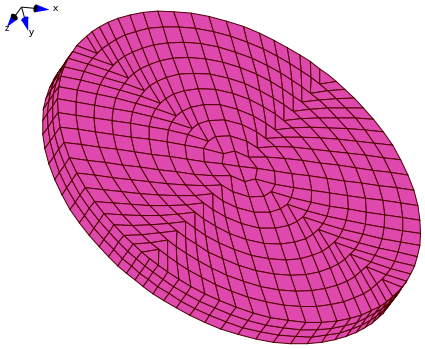

Figure 2.82: Macro element mesh after application of the cylz routine.

set-up is schematically shown in Fig. 2.79. A horizontal polarized plane wave enters the pumped rod. Inside the rod the polarization is locally disturbed. After passing the rod, a vertical polarizer transmits only the depolarized part of the wave, which is observed on the screen. With *SESES*, the photo-elastic effect can be calculated and an additional post-processing leads to the pattern of depolarized light as observed on the screen in Fig. 2.79. To quantify this effect, its useful to calculate the relative amount of depolarized light in a circle of radius r whose center is on the axis.

Model Specification

The file `example/ThinDisk.s3d` starts by defining some geometry parameters of the thin-disk device with Yb:YAG as active material. We then define the cooling parameters represented by the heat transfer coefficients to air and water h_{air} and h_{wat} as well as the absolute temperatures T_{air} and T_{wat}. As we have chosen Yb:YAG, a quasi-three-level system, as active material, only $15\,\%$ of the pump power is assumed to be converted to heat. Furthermore, since we use a thin-disk, we take benefit of the approximation that the pump power density is constant along the direction of the optical axis. As we assume a radial symmetric pump distribution here, we only have to specify the total absorbed pump power and the pump spot radius

```
w0     = 1.0E-3                    (* pump beam focus radius / m        *)
P      = 100                       (* Pump power (W)                    *)
conv   = 0.15                      (* fraction converted to heat        *)
```

For isotropic material behavior and for the photo-elastic effect, we have to specify the orientation of the crystal lattice with vectors **a**, **b** and **c** relative to the lab system. Yb:YAG is a cubic crystal and is normally [1,1,1] cut. With z being the optical axis, we will obtain the situation as shown in Fig. 2.81 in top- and side-view. In *SESES* we only have to specify the x-, y- and z-components of the lattice vectors **a** and **b**, the vector **c** is computed internally. From Fig. 2.81, we see that the z-component of **a** and **b** are equal and given by

$$a_z = b_z = \sqrt{\frac{1}{3}}\,. \tag{2.101}$$

The length of all axes in the top view of Fig. 2.81 equals $\sqrt{2/3}$ and we obtain for the x-

and y-components

$$\begin{aligned} a_x &= \sqrt{\frac{2}{3}}\cos(\alpha)\,, & b_x &= \sqrt{\frac{2}{3}}\cos(\alpha+\beta)\,, \\ a_y &= \sqrt{\frac{2}{3}}\sin(\alpha)\,, & b_y &= \sqrt{\frac{2}{3}}\sin(\alpha+\beta)\,. \end{aligned} \tag{2.102}$$

For simplicity, we chose $\alpha = 0$ and define

```
alpha   = 0*PI/180                    (* angle between Projection of a and x*)
beta    = 120*PI/180                  (* Angle between Projection of a and b*)
ax      = sqrt(2.0/3.0)*cos(alpha)
ay      = sqrt(2.0/3.0)*sin(alpha)
az      = sqrt(1.0/3.0)
bx      = sqrt(2.0/3.0)*cos(alpha+beta)
by      = sqrt(2.0/3.0)*sin(alpha+beta)
bz      = sqrt(1.0/3.0)
```

After the defintion of the relevant problem parameters, we construct the mesh shown in Fig. 2.82 and define the material properties. For Yb:YAG the photo-elastic coefficients are not reported in the literature. We therefore use the values of the well probed material Nd:YAG because the coefficients will mainly be given by the host material YAG. These values are defined with respect to the crystal lattice whose orientation is given by the vectors (2.101) and (2.102) and are to be transformed to the global system with the built-in function `TransformT4`.

```
Parameter Photo TransformT4(
  T.XXXX -0.029; T.YYYY -0.029;  T.ZZZZ -0.029; T.XXYY  0.0091;
  T.YYZZ  0.0091;T.XXZZ  0.0091; T.XYXY -0.0615;T.YZYZ -0.0615;
  T.XZXZ -0.0615; aDir ax,ay,az;  bDir bx,by,bz)
```

The laser is pumped by choosing a homogeneous pumped cylinder distribution with radius w_0

```
Parameter Heat      conv*P/(w0*w0*PI*d_Disk)*(sqrt(x*x+y*y)<w0?1:0) W/m**3
```

As boundary condition, we assume the back surface of the copper heat sink to be in contact with the cooling water and fixed to mechanical elongations; all other surfaces are in contact with air and are free to expand. As final step of the initial sectin, we define a routine to integrate eq. (2.100). The routine's input is a polarization state `RePol`, `ImPPol` and the output is this state after the wave has traveled from the points `p0` to `p`. What we are actually doing is the computation of the exponential function

$$\mathbf{E}(z+dz) = \exp\left(\frac{2\pi i}{\lambda}(\mathbf{B}^{-1/2} - n_0\mathrm{Id})dz\right)\mathbf{E}(z)$$

with the help of a spectral decomposion of $\mathbf{B}$.

In the first part of the command section, we define the lattices on which the OPD and the birefringence has to be determined. The lattices are equally spaced in a square with the lower left corner at $(-r_{\mathrm{opd}}, -r_{\mathrm{opd}})$ and the upper right corner at $(r_{\mathrm{opd}}, r_{\mathrm{opd}})$. Every lattice starts on the back surface of the disk at the z-coordinate $d_{\mathrm{copper}} + d_{\mathrm{indium}}$ and ends on the front surface of the disk with its direction parallel to the optical axis. The number of integration points for all lattices is given by n_{opd}. The used parameters `nrOPD`, `rOPD` and `nOPD` are defined in the initial section.

```
For i From 0 To nrOPD
{  For j From 0 To nrOPD
   { Lattice OPD_@{i}_@{j} Index m=0..nOPD
             (rOPD*(@i*2/nrOPD-1),rOPD*(@j*2/nrOPD-1),d_copper+d_indium)
             +(0,0,d_Disk)*m/nOPD
   }
}
```

The next block defines the solution procedure where we first solve for the temperature `Temp` and then for the displacement `Disp`. Finally we write out the temperature-, the heat-flux, the displacement- and the stress-field.

```
Convergence 1
BlockStruct Block Temp Block Disp
Solve Stationary
Dump Temp TempDiff Disp Stress
```

In the last part of the command section, we compute for all lattices the optical path difference (OPD) and the stress induced depolarization.

```
Write
  Text "OPD    X-coord   Y-coord   OPDtherm  OPDend    OPDtot   "
  Text "Re(Ex)     Re(Ey)    Im(Ex)    Im(Ey)\n"
For i From 0 To nrOPD {  For j From 0 To nrOPD
{
  Text "" Pol=(1,0,0,0), f=0, p0=zero
  Lattice OPD_@{i}_@{j} "" f?Pol=ApplyPol(p0,coord,Pol):Pol,p0=coord,f=1

  Text "(%.0f,%.0f) " @i,@j
  Text "%9.2e %9.2e %9.2e %9.2e %9.2e "
    rOPD*(@i*2/nrOPD-1),rOPD*(@j*2/nrOPD-1),integrate(Lattice OPD_@{i}_@{j};
      DRefrac,Refrac*sqrt(Strain.ZZ*Strain.ZZ),
      DRefrac+Refrac*sqrt(Strain.ZZ*Strain.ZZ))

  Text "%9.2e %9.2e %9.2e %9.2e\n" Pol
}  }
```

Numerical results

For this simulation, we have chosen an absorbed pump power of $100\,\mathrm{W}$ and a pump spot radius of $1\,\mathrm{mm}$. The temperature distribution in the thin-disk is shown in Fig. 2.83. The maximum temperature, localized in the center of the disks front surface, amounts about $92\,^0\mathrm{C}$. As we can easily see, the temperature gradients in the center part of the disk are really into the direction of the optical axis. The temperature gradient becomes strongly radial in the border range of the pumped region. This fact is also confirmed by the direction of the heat flow, shown in Fig. 2.84. In the center part of the disk the direction of the heat flow is parallel to the optical axis and in the border range the direction of the heat flow becomes more and more radially. The temperature raise on a radial straight line, starting in the center of the disk and located on the front and back surface and in the middle of the disk, are shown in Fig. 2.85. The corresponding curves are very flat up to a radius of about $0.75\,\mathrm{mm}$ which corresponds to $3/4$ of the pump spot radius. From a radius of $0.75\,\mathrm{mm}$ to $1.5\,\mathrm{mm}$ they drop significantly down and are again flat for higher radii. Therefore we can expect a very weak thermal lensing in the central part of the disk up to a radius of $0.75\,\mathrm{mm}$.

The optical path difference (OPD) induced by the thermal dispersion and the end effect is shown in Fig. 2.86. As expected, the OPD is very flat up to $r = 0.75\,\mathrm{mm}$.

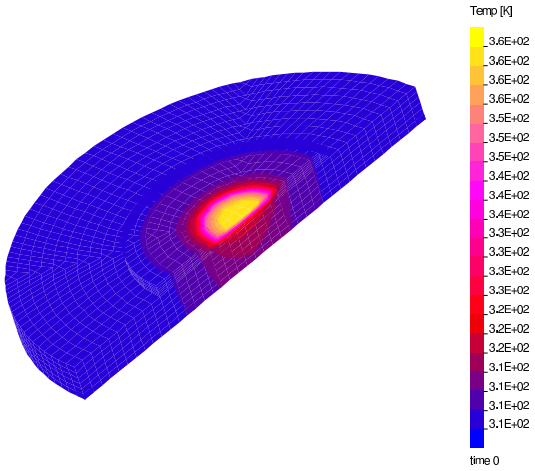

Figure 2.83: Temperature distribution in the disk and its central, pumped with 100 W. The maximum temperature amounts about 92 °C.

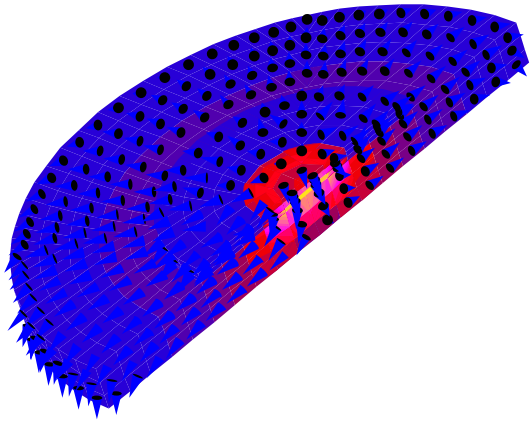

Figure 2.84: Heat flow in the central part of the disk. The cones denote the direction of the heat flow.

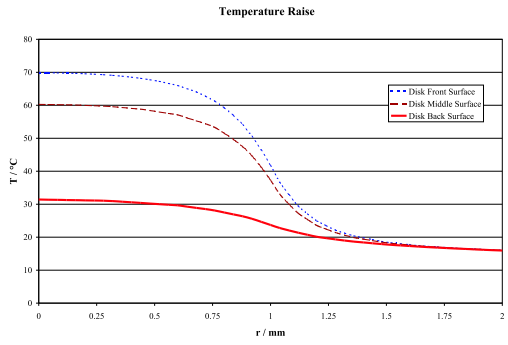

Figure 2.85: Temperature raise on radial straight lines starting in the center of the disk. The lines are located on the front and back surface and in the middle of the disk.

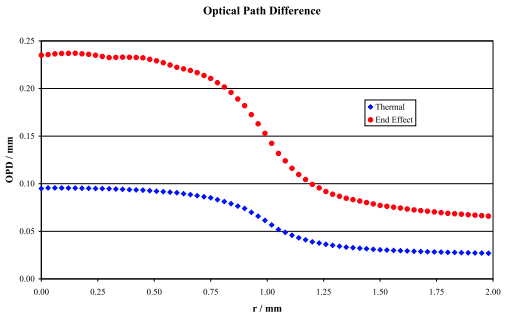

Figure 2.86: Optical path difference (OPD) induced by the thermal dispersion (blue) and the end effect (red).

The corresponding focal lengths of the averaged thermal lens are $f_{\mathrm{th}} = 29.1\,\mathrm{m}$ and $f_{\mathrm{end}} = 11.7\,\mathrm{m}$ which is definitively extremely weak. Even when we deduce the focal length of the averaged thermal lens over a radius of $1.25\,\mathrm{mm}$ the values rest weak with $f_{\mathrm{th}} = 12.6\,\mathrm{m}$ and $f_{\mathrm{end}} = 5.2\,\mathrm{m}$.

Lets last have a look onto the stress induced birefringence by checking the amount of local depolarization of a light ray following the corresponding lattice. As the lattices are equally spaced in a square we get from this depolarized part the information about the local depolarization of a homogenous beam traveling through the disk, see Fig. 2.78. The result, with the well known Maltese cross, is shown in Fig. 2.87. We can see, that depolarization is almost not existing inside the pump spot area and becomes much higher outside. We can determine the relative amount of depolarization in a homogenous beam of radius r as follows. We summarize the relative depolarization of each lattice inside the beam cross section and divide the obtained result by the number of lattices inside this cross section. When we perform this for a varying radius r we obtain the relative amount of depolarization as a function of the beam radius r. A laser beam in the resonator passes the disk twice, because the back surface of the disk acts as high reflecting mirror. Therefore we have to use a double forth and back path to estimate the real depolarization of the disk. If the resonator contains polarizing elements, this relative amount of depolarization can be interpreted as the depolarization loss of the disk for this kind of resonators. The obtained result is shown in Fig. 2.88. As we can see, the depolarization losses are extremely low in the range of $0.01\,\% - 0.05\,\%$. When we chose a beam radius smaller than $0.75\,\mathrm{mm}$, the depolarization loss only amounts to $0.002\,\%$ whereas the highest loss of $0.05\,\%$ is obtained for a beam with radius $r = 1.5\,\mathrm{mm}$.

In summary we can say, that the thin-disk-laser represents a laser concept which can really overcome the drawback of high radial temperature gradients. We have obtained an extremely weak thermal lensing and almost no depolarization losses. The thin-disk-laser concept is therefore very promising for the future.

References

[1] K. CONTAG, M. KARSZEWSKI, CHR. STEVEN, A. GIESEN, H. HÜGEL, *Theoretical modeling and experimental investigations of the diode-pumped thin-disk Yb:YAG laser*, Quantum Electronics 29, pp. 697-703, 1999.

[2] A. GIESEN, H. HÜGEL, A. VOSS, K. WITTIG, U. BRAUCH, H. OPOWER, *Scalable concept for diode-pumped high-power solid-state lasers*, Appl. Phys. B 58, pp. 365-372, 1994.

[3] C. STEWEN, M. LARIONOV, A. GIESEN, K. CONTAG, *Yb:YAG thin disk laser with 1 kW output power*.

[4] E. HECHT, *Optics*, Addison-Wesley publishing company, pp. 282-26, 2nd Ed., 1987.

[5] B. E. A. SALEH, *Fundamentals of Photonics*, pp. 210-234, John Wiley & SonsInc., 1991.

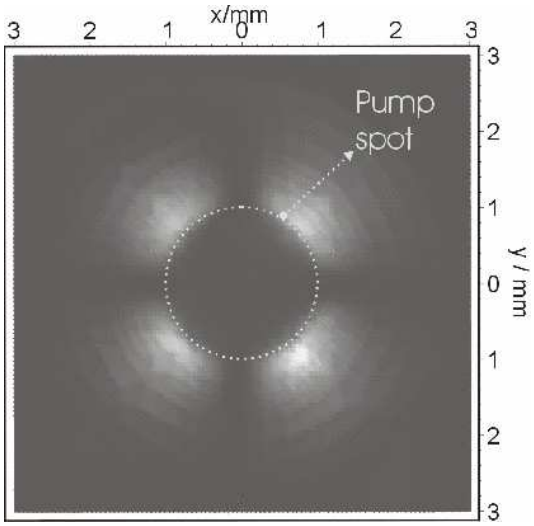

Figure 2.87: Depolarized part of a plane wave traveling through the disk. The grey dotted circle indicates the pump spot.

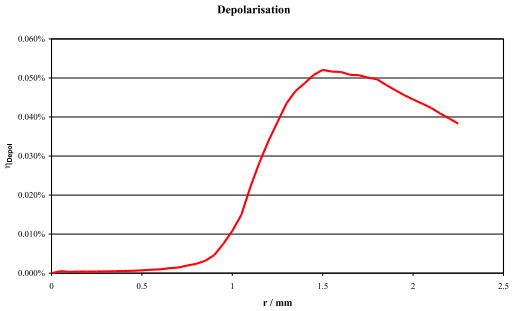

Figure 2.88: Depolarization loss of the disk for a homogenous beam of radius r passing the disk forth and back.

[6] J. F. NYE, *Physical properties of crystals*, Clarendon Press, Oxford, 1993.

2.19 Thermo Optically Self-Compensated Amplifier

The thermally induced lens in a laser rod is by far the most critical issue in the development of high-power solid state lasers and limits the output power of lasers with good beam quality ($M^2 \approx 10 \ldots 20$) to a narrow power range. In the last example, we have presented the thin-disk-laser design which is well suited to overcome this drawback. But this design is limited to quasi-longitudinal-pumping requiring high quality diode pump modules and beam shaping optics. The pump radiation has to pass the disk several times to guarantee a high absorbed part of the pump power in the thin disk. In practice this is realized by a special and costly pump optics. Furthermore the maximum pump power per disk is limited and for high power lasers two ore more disks have to be used in one and the same resonator. The scaling of this design to higher output powers requires therefore an additional effort. In contrast the scalability of transversal diode pumped rods is much easier. In principal we only have to increase the length of the laser rod and the pump modules. Also the requirements to the beam quality of the pump modules and the beam shaping optics are much lower. Therefore transversal diode pumping represents still attractive design for high power solid state lasers.

In the present example we will introduce a new and very promising transversally diode pumped laser design with an internal compensation of the thermally induced lens. The design bases on the idea to take advantage of the thermal effects themselves and to use a heated optical element as a compensating element [1, 2]. The thermal lens in a rod is evoked by the temperature dependent refractive index of the laser gain medium which mainly shows a positive thermal dispersion. In contrast some selected

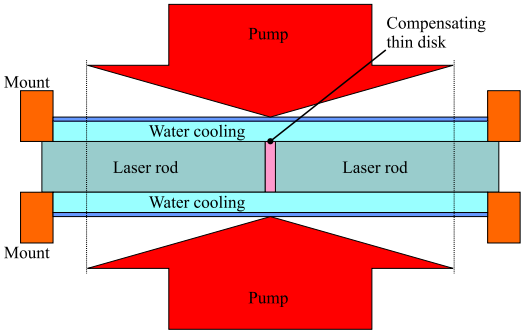

Figure 2.89: Scheme of a thermo optically self-compensating amplifier with a compensating thin disk placed between two transversally pumped laser rods.

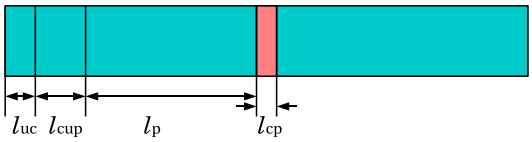

Figure 2.90: Simulated device consisting of two laser rods and the compensating thin disk.

glasses, liquids and curing gels show a negative thermal dispersion, whose modulus may be two magnitudes higher than for the laser gain medium. Let us cut a transversally pumped laser rod into two parts and place between them a thin disk of such a material, then this disk will quite exactly adopt the radial temperature distribution of the rod. The negative thermal dispersion in the disk leads to a negative thermal lens which will be much stronger than the ones in the two parts of the rod. Therefore such a thin disk of suitable thickness should be able to mainly compensate the thermally induced lens of the pumped rod. To guarantee a good contact of the thin disk and the two parts of the laser rod on the whole surfaces, the most advantageous materials are liquids or curing gels. Fig. 2.89 shows a scheme of this set up, called Thermo Optically Self-Compensating Amplifier (TOSCA). The thin compensating disk is placed adjacent between two laser rods. The main part of the rods and the compensating disk, in case of a non hygroscopic material, are in direct contact with the flowing cooling water and are transversally pumped pumped through this cooling water. For simplicity, we will assume that the compensating disk shows no absorption at the pump wavelength.

Model Specification

With the present example `example/Tosca.s2d`, we will perform a parameter study to obtain the optimal thickness of the thin compensating disk for a set of given geometrical and pump parameters, where for simplicity we will perform just a rotational symmetric simulation. Fig. 2.90 shows the simulated device consisting of two rods and the compensating thin disk. The outer rod parts of the length $\ell_{\rm uc}$ are in contact with the mount, see Fig. 2.89 and are therefore not directly cooled by water. Adjacent, we have two parts of length $\ell_{\rm cup}$ which are directly cooled by water but not pumped by diode radiation. The disk with thickness $l_{\rm cp}$ is located between the two pumped rod parts of length $\ell_{\rm p}$. These geometrical parameters together with the rod radius r are defined at the beginning of the input file.

```
Define
l_uc  =  1.0E-3   nl_uc   = 1          (* uncooled length / m               *)
l_cup =  2.0E-3   nl_cup  = 1          (* cooled and unpumped length / m    *)
l_p   = 20.0E-3   nl_p    = 10         (* pumped length / m                 *)
r     =  2.0E-3   nr  = 5              (* rod radius / m                    *)
nl_cp    = 4
```

```
l_cpmin = 0.6E-3                        (* minimum length of compensator / m *)
l_cpmax = 1.3E-3                        (* maximum length of compensator / m *)
n_l_cp   = 3                            (* #steps in length of compensator   *)
```

For the parameter study, we have also to define the minimum and maximum thickness of the compensating disk $\ell_{\mathrm{cp\_min}}$ and $\ell_{\mathrm{cp\_max}}$, as well as the number of parameter steps n_{ℓ_cp}. This section is followed by the definition of the element dimensions, coordinates, node numbers of the mesh in the direction of the optical y-axis. As default value for the thickness of the compensating thin disk we set $1\,\mathrm{mm}$. This value will be automatically changed during the parameter study. After the definition of the cooling, refinement and OPD parameters, we start with the specification of the pump distribution. In the case of transversal pumping, we usually specify the total absorbed pump power per length $\mathrm{d}P/\mathrm{d}y$ from which the fraction $conv$ (40% in case of Nd:YAG) is converted to heat. Inside the rod, we assume a parabolic shaped pump distributionsee of the type

$$\frac{\mathrm{d}P}{\mathrm{d}V} = p_c(1 - a\,r^2)\,,$$

with p_c the pump power density on the optical axis and a a decreasing factor, see Fig. 2.91. If at the distance R along the rod's radius, the power density drops to $\eta\,p_c$ (we use here $\eta = 0.5$), then we have $a = (1-\eta)/R^2$. The integrated pump power density in a cross-section of the rod equals the pump power per length dP/dy so that we have

$$\int_0^{2\pi}\int_0^{R} p_c\,\frac{\mathrm{d}P}{\mathrm{d}A}\,r\,\mathrm{d}r\,\mathrm{d}\varphi = \frac{\mathrm{d}P}{\mathrm{d}y} \quad \Rightarrow \quad p_c = \frac{\frac{\mathrm{d}P}{\mathrm{d}y}}{\pi\,R^2\,\frac{1+\eta}{2}}\,.$$

The corresponding laser pump values are then defined as parameters.

```
dP_dy = 10                              (* pump power per length / W/mm       *)
conv  = 0.4                             (* fraction of power converted to heat*)
eta   = 0.5                             (* power drop at radius               *)
p_c   = dP_dy*2/(PI*r*r*(1+eta))        (* power density on rod axes          *)
```

As we have a unpumped- and pumped Nd:YAG part, we define the two materials `UP_NdYAG` and `P_NdYAG`.

```
MaterialSpec UP_NdYAG
  Equation ThermalEnergy Elasticity Enable
  Parameter StressOrtho   LinElastIso(Emodule 307 GPa; PoissonR 0.3;
                                      AlphaIso alpha(Tair)) SIunit
  Parameter Refrac          1.82
  Parameter DRefrac         (7.3e-6)*(Temp-Tair)
  Parameter KappaIso D_Temp kappa() Dkappa()  W/(m*K)-W/(m*K**2)

MaterialSpec P_NdYAG From UP_NdYAG
  Equation ThermalEnergy Elasticity Enable
  Parameter Heat         conv*p_c*(1-(1-eta)*x*x/(r*r)) W/m**3
```

For the pumped part, we only have to add the heat distribution, therefore we derive the material `P_NdYAG` from `UP_NdYAG` and additively define the heat distribution. We further assume Sylgard 184 as compensator material and dedfine the following material properties.

```
MaterialSpec Compensator
  Equation ThermalEnergy Elasticity Enable
  Parameter Refrac      1.43
  Parameter DRefrac     (-3.0e-4)*(Temp-Tair)
  Parameter KappaIso    0.15 W/(m*K)
  Parameter StressOrtho LinElastIso(Emodule 20 kPa; PoissonR 0.45;
                                    AlphaIso 0.25e-4*(Temp-Tair)) SIunit
```

After the definition of the material properties, we define the mesh and maps the materials to to mesh.

```
QMEI nr (r/nr)
QMEJ ny1 (dy1/ny1)  ny2 (dy2/ny2)  ny3 (dy3/ny3)  ny4 (dy4/ny4)
     ny5 (dy5/ny5)  ny6 (dy6/ny6)  ny7 (dy7/ny7)

CoordNonConst
Coord x y+(y-y3)*(l_cp-1) block(0 nxtot   nny3 nny4-1)
Coord x y+l_cp-1          block(0 nxtot   nny4 nytot)
```

The `Coord` statements are introduced due to the varying thickness ℓ_{cp} of the compensating disk which enforces a redefinition of the mesh for each solution step. As thermal boundary condition, we assume the uncooled part as well as the front and back surface of the rod to be in contact with air. The cooled part is of course in contact with water. For the mechanical fixation, we assume the uncooled part, where the mount is located, to be fixed for radial elongations and due to symmetry, we fix longitudinal elongations in the center of the rod, see Fig. 2.89. As last part of the initial section, we define the rotational symmetry with the statement

```
GlobalSpec Model AxiSymmetric Enable
```

We start the command section with the definition of the information in the output stream and the used solver

```
Info Refine HideBC
LinearSolver NonSymmetric
```

and define the lattices on which the OPD has to be calculated.

```
For k From 0 To nr_OPD
{  Lattice OPD_@k Index m=0..d_OPD (@k*r_OPD/nr_OPD,0)+(0,@l_OPD)*m/d_OPD
}
```

We then initialize the temperature field with the air temperature.

```
Solve Init Temp=Tair
```

We ingrain the solution procedure into a loop to perform the parameter study. For each solution step, the automatic mesh refinement algorithm is used for the temperature and the displacement field. At the first `Solve Stationary` statement the corresponding thickness of the compensating disk is set by the `ForSimPar` statement. Afterwards the fields `Temp` and `Disp` are calculated using the adaptive mesh refinement algorithm and written out together with the `Stress` field. Finally the OPD is calculated and written out to the file `OPD_i.dat`.

```
   while (1)
   {
      Store StressAver(Continuous 1;FreeOnRef)=Stress,
            TempDiffAver(Continuous 1;FreeOnRef)=TempDiff
      Remesh Refine ErrTemp(TErr)  Global Refine ErrDisp(MErr)  Global
      Solve Stationary ForSimPar l_cp At l_cpmin+@i*(l_cpmax-l_cpmin)/n_l_cp
      if (REMESH) break;
   }
   Dump Temp Disp Stress If @i>0 { Append }
   Write File "OPD_"@@i".dat" Text "r\t OPDtherm\t OPDend\t OPDtot\n"
   For k From 0 To nr_OPD
   {
      "%12.8e\t %12.8e\t %12.8e\t %12.8e\n" @k*r_OPD/nr_OPD,
      integrate(Lattice OPD_@k; DRefrac,(Refrac-1)*sqrt(Strain.YY*Strain.YY),
                                DRefrac+(Refrac-1)*sqrt(Strain.YY*Strain.YY))
   }
```

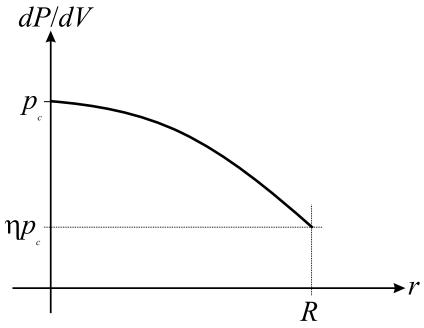

Figure 2.91: Radial pump power density profile.

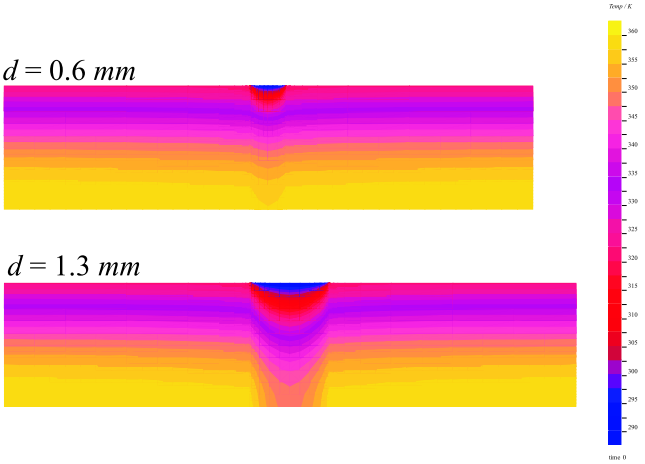

Figure 2.92: Temperature distribution in the center region of the device. The temperatures decreases in the compensating disk, especially in the outer part. This decrease is more pronounced in the case of thicker disks.

Numerical results

We perform the simulation for a Nd:YAG rod of $r = 2\,\mathrm{mm}$ radius, a pumped length of $\ell_p = 20\,\mathrm{mm}$, a cooled and unpumped length of $\ell_{\mathrm{cup}} = 2\,\mathrm{mm}$ and an uncooled length of $\ell_{\mathrm{uc}} = 1\,\mathrm{mm}$. We vary the thickness of the compensating disk in $0.1\,\mathrm{mm}$ steps from $0.6\,\mathrm{mm}$ up to $1.3\,\mathrm{mm}$. The absorbed pump power per length amounts $10\,\mathrm{W/mm}$ which corresponds to total pump power of $400\,\mathrm{W}$. Fig. 2.92 shows the temperature distribution in the center part of the device for thicknesses of the compensating disk of $0.6\,\mathrm{mm}$ and $1.3\,\mathrm{mm}$. We can see that, especially in the outer parts of the rod, the temperature distribution of the laser rods is not exactly transferred into the compensating disk and decreases in the disk. This deviation becomes more pronounced for thicker disks. From this we can expect, that the compensation of the thermal lens will not be optimal in the outer part of the device. This assumption is confirmed by the OPD's, as shown in Fig. 2.93. The curvature of the OPD near the optical axis is concave for small thicknesses of the disk and becomes convex when the thickness is increased. Therefore we have first a positive focal length of the thermal lens which becomes subsequent negative with increasing thickness of the disk. For suitable thicknesses of the disk, the OPD near the optical axis is almost constant and the thermal lens in this range therefore vanishes. The OPD in the outer part of the device is not influenced by the compensating disk, this is due to the fact the the outer part of the disk adopts the temperature of the cooling water as we have previously seen in Fig. 2.92. From the OPD data we can determine the focal length of the averaged thermal lens by a parabolic fit. The fit was performed over the extension from $r = 0$ to $r = 1.5\,\mathrm{mm}$ and the corresponding results are shown in Fig. 2.94. The dioptric power of the averaged thermal lens has a linear dependence from the thickness of the compensating disk. The linear regression to this data leads to the optimal thickness of the compensating element of $d_{\mathrm{opt}} = 0.86\,\mathrm{mm}$, for a dioptric power of 0.

In summary, the thermo optically self-compensated amplifier is a very promising de-

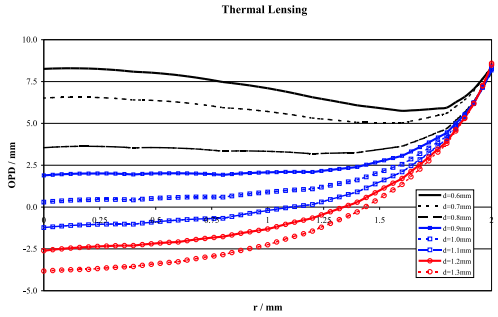

Figure 2.93: OPD distributions as a function of the thickness of the compensating disk.

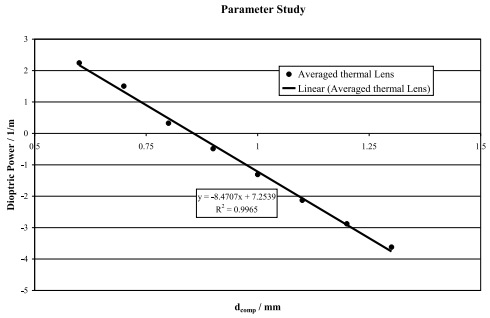

Figure 2.94: Dioptric power of the averaged thermal lens as a function of the thickness of the compensating disk (dots) and the corresponding linear fit (solid line).

sign. It overcomes the drawback of strong thermal lensing and takes benefit of the easy to manage and not costly technique of transversal diode pumping. In addition the design rests easy scalable for higher output powers and is therefore very interesting for industrial application.

References

[1] TH. GRAF, E. WYSS, M. ROTH, H.P. WEBER, *Laser resonator with balanced thermal lens*, Optics Communication, Vol. 190, pp. 327-331, 2001.

[2] E. WYSS, M. ROTH, TH. GRAF, H.P. WEBER, *Thermo Optical Compensation Methods for High-Power Lasers*, IEEE J.Q.E, Vol. 38, No. 12, pp. 1620-1628, 2002.

2.20 Shells or Thin mechanical structures

A shell is a curved thin-walled mechanical structure capable of supporting large loads. This property stems from its curved nature and the ability to distribute internal transverse loads towards in-plane loads at the edges of the shell. Shells are pervasive in nature and as artefacts and include skulls, aortic valves, blood vessels, domes, silos, tubes, cans, car bodies, balloons, etc. Because of their overall presence, many mechanical simulations will then actually deal with shell structures. A priori one may think this task being as simple as performing a mechanical simulation based upon the discretization of the 3D governing equations of continuum mechanics by some standard finite elements. However, this approach does not always work well when the mechanical structure is shallow in one of the three directions, since low order finite elements are numerically unstable here and solutions are affected by large errors, an effect known as *numerical locking*. When numerical locking appears, the displacement are underestimated and the stress is overestimated, i.e. the mechanical structure is by far more stiff than in reality. The main reason of this failure is that thin mechanical structures tend to bend and the resulting displacement is a third order function of

the coordinates, badly approximated by first or low order finite elements.

Because the analytical solution of the 3D continuum mechanical equations is seldom possible, except for some trivial cases, there has always been interest in obtaining more tractable governing equations by model reduction. For shell structures and based on physical sounded arguments stemming from the shallowness hypothesis, one can make kinematic assumptions on the solution. The most prominent one is that straight vertical fibers in the undeformed configuration, remain straight during any deformation. By considering the now constrained displacement, integration of the balance laws in the transverse direction yields shell models and as particular cases plate and beam models. With this approach, many analytical solutions have been obtained, see for example [10] for linear axis-symmetric shells of revolution.

Numerical methods for shells may use the underlying shell models for their discretization. At the present time, these shell models are mostly linear models where the displacement just enters linearly into the governing equations. Among the most prominent models, we have the linear model of Koiter for thin shells with zero transverse shear strain and the linear Nagadi model for thick shells, [2, 5, 6]. For numerical applications, the Koiter model is used to a less extent, since it requires finite elements for the Sobolov space $(H^1(\Omega))^2 \times H^2(\Omega)$. Here the main drawback is that finite elements with continuous derivatives as required by $H^2(\Omega)$ are not easily constructed. Instead, the Nagadi model requires finite element approximations in the standard Sobolov space $(H^1(\Omega))^5$. Here similarly as for the discretization of the native 3D mechanical formulation, standard finite elements for $H^1(\Omega)$ are subject to numerical locking. However, stable and locking free finite elements have been discovered for the particular cases of plates [7, 3] and shells in the bending dominated state [4]. This stability is asserted by proving the convergence rate to be independent from the thickness of the shell.

Even assuming that stable finite element based on the Nagadi shell model will be sometimes discovered thus paving the way for robust numerical simulations of shells, their usage for practical applications is somewhat limited. As first, just the geometric linear case is generally considered, so that for large displacements, these linear shell elements need to be amended e.g. by a corotational formulation not free of disadvantages, see [8]. Secondly the kinematic behind the Nagadi model assumes a vanishing transverse normal stress and no change in the shell thickness during the deformation. As a result generic 3D material laws need to be amended for shells and mechanical contacts on the upper and lower shell surfaces are not easily modeled numerically.

The engineering community is therefore pursing another approach to shell modeling than the applied mathematical community which uses shell kinematic models. The approach is based on the discretization of the native 3D non-linear continuum formulation and by trying to fix first or low order finite elements to avoid numerical locking. This trend started with the reduced integration approach [11] and the mixed MITC plate elements [1] and has been further developed into what has become a solid-shell approach. These finite elements just use displacement degree-of-freedoms at the top and bottom of the shell, from the external point of view they are the same as standard $(H^1(\Omega))^3$ solid elements but they are inherently anisotropic in the transverse direction which is assumed to be shallow. The shallowness hypothesis is used to fix the various locking numerical behaviors. The main advantage of this approach is that we can use the same 3D material laws and the same boundary conditions as for solid elements.

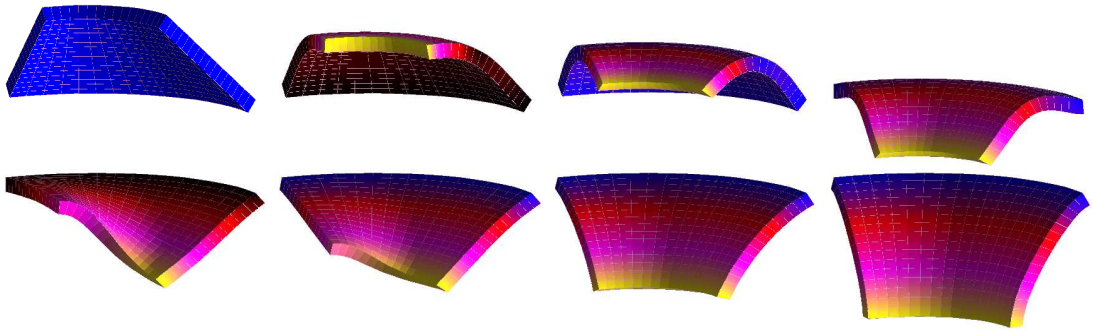

Figure 2.95: Shell displacements for snap-through and warp solutions.

Numerical example

For the solution of mechanical problems, *SESES* offers the choice between first and 2nd order solid elements and the solid-shell elements of [9], all ones either for the geometric linear and non-linear formulation and with generic 3D material laws. For a practitioner is therefore of interest to know the range of applicability and properties of the various finite elements, topics discussed together with the *SESES* file found at `example/ConicalShell.s3d` modeling a conical shell. The shell is an axis-symmetric shell of revolution, with isotropic linear material laws and with loads applied symmetrically. We use the geometric non-linear model allowing large displacements to be computed. The shell is shallow in the topological `K`-direction so that the three type of finite elements are easily selected by specifying either the `Elasticity`, `Elasticity2` or `ElasticShellSingleK` equation model.

Although the problem formulation is fully symmetric with respect to the axis of rotation, non-linear solutions may not be necessarily symmetric as for warp deformations breaking the symmetry, so that a reduced 2D axis-symmetric approach only computing symmetric solutions is not followed here. However, to reduce a little bit the numerical work, we just compute a quarter of the 3D shell by applying symmetric BCs on the virtual cuts. In doing so, we cannot compute solutions with half-symmetry. The bottom of the shell is fixed but can freely rotate on the bottom edge and on the top we apply an increasing vertical downward traction. All three types of finite elements just use Lagrange displacement degree-of-freedoms, therefore the BCs are independent of the element type.

The conical shell starts to deform inward and after a while it approaches an unstable snap-through region, see first row of Fig. 2.95. In a dynamical simulation, the shell will snap-through and will start oscillating around the next stable solution. For stationary computations, however, solutions around this turning point can only be computed with the help of a load control algorithm. The particularity of this instability is that increasing displacements are characterized by decreasing loads. By just changing the load, we cannot pass the turning point, since by decreasing the load we go back on the path of already computed solutions. Load control algorithms are devised to pass these points, since they use another solution's parameter than the load, typically something related to the norm of the computed displacement. A brief overview is given in the Section *1.7 Non-Linear Algorithms*. By further increasing the displacement, we arrive

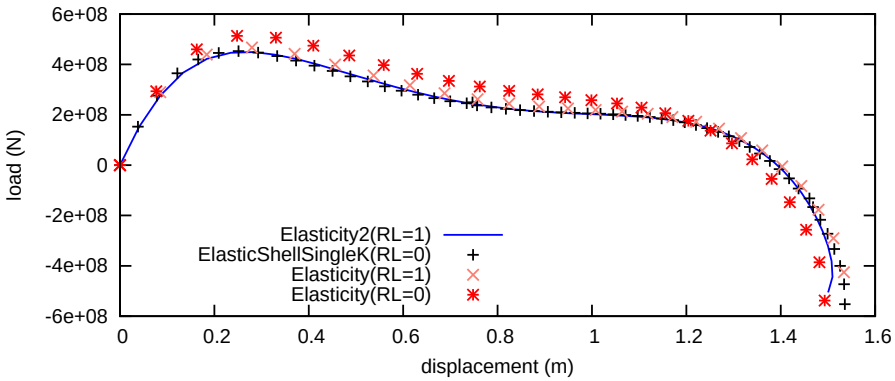

Figure 2.96: Applied load versus displacement L^∞-norm for various finite elements and refinement levels.

at a bifurcation point where warp deformations show up, see second row of Fig. 2.95. At the present time, no robust algorithms are implemented to pass bifurcation points and so it is a matter of luck if we can pass these singular points and on which solution branch we land afterwards. In particular, the warp solutions of Fig. 2.95 have been computed with first order solid elements. As a rule of thumbs, low order elements with few degree-of-freedoms are more robust when solving non-linear problems and in this case they can easily jump over the singularity. This is not the case for the 2nd order solid elements or the solid-shell elements with many internal degree-of-freedoms.

The geometry of the conical shell is easily defined as well as the definition of the linear isotropic elastic law and BCs. To use a load control algorithm, we need to specify the derivative of the global residuals with respect to the load parameter, in our case the pressure load. By default, a difference method is implemented to compute this derivative, however, in our example the load is applied as a Neumann `Pres` BCs which enters linearly into the formulation. In this special case, as alternative to the numerical difference, there is the possibility to exactly specify the derivative. One has to define a second Neumann `Pres1` BCs, with a value representing the derivative with respect to the load parameter. The boundary for this second BC is not used and can be empty.

```
BC Press 0 0 0 IKType nx nz Neumann Disp (0,0,load) Pa
BC Press1                   Neumann Disp (0,0,1) Pa Disable
```

The declaration of the exact derivative is done by associating both BCs with the statement

```
Increment DR_DSimPar Linear(BC Press.Press1)
```

The family of solutions computed by the load control algorithm is shown in Fig. 2.96 up to the first warp solution with the applied load as function of the displacement L^∞-norm. A reference solution has been computed with second order elements and a homogeneous refinement level of 1 and the other curves are for solid-shell and first order solid elements with a homogeneous refinement level of 0 and 1. The solid-shell elements are in very good agreement with the most precise solution and the first order

solid elements are not so dramatically worse. This is not so a critical example for numerical locking since the elements are well shaped and the shell is rather thick. However, first order elements are by far less performant as soon as the thickness/size ratio of the elements is getting worse. The solid-shell elements specified by the equation `ElasticShellSingleK` always use displacement degree-of-freedom defined just at the top and bottom of the shell independently from the number of elements defined through the thickness. Therefore, the time spent within the linear solver is the same as when using first order solid elements with a single element through the thickness, which is the cheapest way to compute a solution. However, the overall solution time is larger since solid-shell elements are more expensive to assemble. If not tremendously slow, the time spent to compute elements is in general of little concern since it grows linearly with the number of elements and therefore for large problems, the linear solver time will always dominate the overall solution time.

References

[1] K. J. BATHE, *Finite Element Procedures*, Prentice Hall-International, 1996.

[2] M. BERNADOU, *Finite Element Methods for Thin Shell Problems*, Wiley, 1996.

[3] D. CHAPELLE, R. STENBERG, *An optimal low-order locking free finite element method for Reissner-Mindlin plates*, Mathematical Models and Methods in Applied Science, Vol. 8, No. 3, pp. 407-430, 1998.

[4] D. CHAPELLE, R. STENBERG, *Stabilized finite element formulations for shells in a bending dominated state*, SIAM J. Numer. Anal., Vol. 36, No. 1, pp. 32-73, 1998.

[5] D. CHAPELLE, K. J. BATHE, *The Finite Element Analysis of Shells-Fundamentals*, Springer, 2003.

[6] P. G. CIARLET, T. LI, EDS., *Differential geometry : theory and applications*, Series in contemporary applied mathematics CAM 9, World Scientific, 2008,

[7] R. DURAN, A. GHIOLDI, N. WOLANSKI, *A finite element method for the Mindlin-Reissner plate model*, SIAM J. Numer. Anal., Vol. 28, No. 4, pp. 1004-1014, 1991.

[8] C. A. FELIPPA, B. HAUGEN, *A unified formulation of small-strain corotational finite elements: I. Theory*, Comput. Methods Appl. Mech. Engrg. Vol. 194, pp. 2285-2335, 2005.

[9] S. KLINKEL, F. GRUTTMANN, W. WAGNER, *A robust non-linear solid shell element based on a mixed variational formulation,* Comput. Methods Appl. Mech. Engrg., 195, pp. 179-201, 2006.

[10] G. MÁRKUS, *Theorie und Berechnung rotationssymmetrischer Bauwerke*, Werner-Verlag, 1976.

[11] O. C. ZIENKIEWICZ, R. L. TAYLOR, J. M. TOO, *Reduced integration technique in general analysis of plates and shells*, Int. J. Numer. Meth. Engng., Vol. 3, pp. 275-290, 1971.

2.21 Plasticity Models

Plasticity theory plays an important role in metal forming and geophysics, where simple linear elastic laws cannot correctly reproduce mechanical deformations. Even for infinitesimal strains, plastic models are non-linear, they depend on the history of the deformations and may be on the overall quite involved, for an overview see e.g. [1]-[2]. Since *SESES* accepts general non-linear elastic laws with the possibility to use user defined element fields to implement the history dependency, a possible approach to plasticity is to define models of interest within template files and use them on demand. Although the generic user does not need to understand the model set-up and the details of plasticity theory, this approach has the advantage that one clearly see what the models are doing and how they are written so that small adaptations and extensions are readily implemented. In this example, we will discuss such an approach and comment the steps required to define commonly used plasticity models for infinitesimal strains. In particular, we will compute the 2D example of J_2-flow with plane strain presented in [3]. Because the *SESES* input is very basic, actually one has to specify the stress and its derivatives with respect to the infinitesimal strain, there is quite some work involved before ending up modeling a plastic flow, which however shows the flexibility of such an approach.

General Theory

Plastic models are a particular class of hypoelastic models which are an extension of elastic models. Within an infinitesimal theory of small displacements, an elastic model consists in defining the stress s as a function of the linearized strain $\boldsymbol{\varepsilon} = (\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}})/2$ with $\mathbf{u}$ the mechanical displacement. The extension to a hypoelastic model consists in specifying a stress flow $\dot{\mathbf{s}}$ instead of $\mathbf{s}$ which introduces a time dependency and the need for a time integration algorithm to be performed locally at each point. The time, however, it is a pseudo time and intended is a slow mechanical deformation where all inertial forces can be neglected. In this contribution, we will not discuss details related to a particular choice of the time integration algorithm nor of an automatic step selection, but we will use the implicit backward Euler algorithm with a fixed time step to be defined by the user. This algorithm is first order accurate, absolute stable and has been shown to be superior with respect to second order ones for several plasticity models.

The basic assumptions used by phenomenological models of plasticity are that the strain $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p$ can be decomposed in an elastic $\boldsymbol{\varepsilon}^e$ and plastic $\boldsymbol{\varepsilon}^p$ part, there is an energy function $\Psi(\boldsymbol{\varepsilon}^e, \xi)$ depending on the elastic strain $\boldsymbol{\varepsilon}^e$ and some additional internal strain variables ξ specifying the stress $\mathbf{s} = \partial_{\boldsymbol{\varepsilon}^e}\Psi$ and the internal stress variables $\mathbf{q} = -\partial_{\xi}\Psi$ together with a yield function $f(\mathbf{s}, \mathbf{q}) \leq 0$ constraining the stress variables $(\mathbf{s}, \mathbf{q})$ and where the flow of the plastic strain variables $(\dot{\boldsymbol{\varepsilon}}^p, \dot{\xi})$ is defined as follow. If $f(\mathbf{s}, \mathbf{q}) < 0$ holds, we are in the elastic region and there is no plastic flow $(\dot{\boldsymbol{\varepsilon}}^p, \dot{\xi}) = 0$, otherwise the plastic flow must maximize the plastic dissipation $\mathbf{s} \cdot \dot{\boldsymbol{\varepsilon}}^p + \mathbf{q} \cdot \dot{\xi}$. This maximum principle results in a yield function f being necessarily convex. Although quite abstract, this problem formulation is mathematically well sounded and existence/uniqueness properties of solutions can be proven for some models, [5]. The

above formulation is equivalent to the following Kuhn-Tucker equations where the independent variables are $(\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}^p, \xi)$

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\partial_{\mathbf{s}} f\,, \quad \dot{\xi} = \dot{\lambda}\partial_{\mathbf{q}} f \text{ with } \; f\dot{\lambda} = 0\,, \quad \dot{\lambda} \leq 0\,. \tag{2.103}$$

This is clear since if $f < 0$ we must have $\dot{\lambda} = 0$ and so there is no plastic flow, otherwise if $f = 0$ the above equations are just the classical Lagrange equations for the maximum of a constrained functional.

Let us apply the backward Euler algorithm to the eqs. (2.103) by performing a single step. The starting point is therefore a time t_n with a know configuration of the independent variables $(\boldsymbol{\varepsilon}_n, \boldsymbol{\varepsilon}^p_n, \xi_n)$ and a time step Δt where we use the subscript n to specify a time function evaluated at t_n. With the backward Euler algorithm, the solution for $(\dot{\boldsymbol{\varepsilon}}^p_{n+1}, \dot{\mathbf{q}}_{n+1})$ at $t_{n+1} = t_n + \Delta t$ is computed by performing a single time step of length Δt and using the time derivative evaluated at t_{n+1}. From (2.103), we therefore obtain the system of equations

$$\begin{array}{rcl}
\boldsymbol{\varepsilon}^p_{n+1} & = & \boldsymbol{\varepsilon}^p_n + \Delta\lambda\partial_{\mathbf{s}} f_{n+1}\,, \\
\xi_{n+1} & = & \xi_n + \Delta\lambda\partial_{\mathbf{q}} f_{n+1}\,, \\
\mathbf{s}_{n+1} & = & \partial_{\boldsymbol{\varepsilon}^e}\Psi(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1}, \xi_{n+1})\,, \\
\mathbf{q}_{n+1} & = & -\partial_{\xi}\Psi(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1}, \xi_{n+1})\,, \\
f_{n+1} & \leq & 0\,, \\
f_{n+1}\Delta\lambda & = & 0\,, \\
\Delta\lambda & \geq & 0\,,
\end{array} \tag{2.104}$$

with $f_{n+1} = f(\mathbf{s}_{n+1}, \mathbf{q}_{n+1})$ and $\Delta\lambda = \dot{\lambda}\Delta T$. The value of the strain $\boldsymbol{\varepsilon}_{n+1}$ at the new time t_{n+1} is considered know and determined by incremented solution $\mathbf{u}_{n+1}$. The common procedure used to solve the system (2.104) is first to assume that the plastic flow does not change, i.e. $\Delta\lambda = 0$ and from (2.104) we obtain the trial solution

$$\boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^p_n\,, \; \xi_{n+1} = \xi_n\,, \; \mathbf{s}_{n+1} = \partial_{\boldsymbol{\varepsilon}^e}\Psi(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_n, \xi_n)\,, \; \mathbf{q}_{n+1} = \mathbf{q}_n\,. \tag{2.105}$$

The important point here is that this solution satisfies $f_{n+1} \leq f_{\text{trial}}$ which is a property following from the convexity of the yield function f, see [3, 4]. Therefore since $f_{n+1} \leq 0$, if $f_{\text{trial}} \leq 0$, we must necessarily have $\Delta\lambda = 0$ and our trial solution (2.105) is actually the right solution at t_{n+1}, otherwise we know that $\Delta\lambda > 0$ and the unilateral constraint of (2.104) turns into simple a constraint and we have to find $(\boldsymbol{\varepsilon}^p_{n+1}, \xi_{n+1}, \mathbf{s}_{n+1}, \mathbf{q}_{n+1}, \Delta\lambda)$ so that

$$\begin{array}{rcl}
\boldsymbol{\varepsilon}^p_{n+1} & = & \boldsymbol{\varepsilon}^p_n + \Delta\lambda\partial_{\mathbf{s}} f_{n+1}\,, \\
\xi_{n+1} & = & \xi_n + \Delta\lambda\partial_{\mathbf{q}} f_{n+1}\,, \\
\mathbf{s}_{n+1} & = & \partial_{\boldsymbol{\varepsilon}^e}\Psi(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1}, \xi_{n+1})\,, \\
\mathbf{q}_{n+1} & = & -\partial_{\xi}\Psi(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1}, \xi_{n+1})\,, \\
f_{n+1} & = & 0\,.
\end{array} \tag{2.106}$$

For general plasticity models, this system of equations is solved with an iterative Newton's algorithm generally started with the trial solution (2.105).

J_2-Flow with isotropic/kinematic linear hardening

For particular models, we can simplify the equations (2.106) and for linear models we can find a closed solution as done for the following J_2-flow example with hardening

which is valid for a 2D plane strain, 2D rotational symmetric and 3D analysis, not however for a 2D plane stress analysis. The energy function $\Psi(\boldsymbol{\varepsilon}^e, \xi) = \Psi_m(\boldsymbol{\varepsilon}^e) + \Psi_p(\xi)$ generally splits in a mechanical and plastic part. For the mechanical part, we assume the most simple linear isotropic strain-stress law given by

$$\mathbf{s} = \partial_{\boldsymbol{\varepsilon}} \Psi_m(\boldsymbol{\varepsilon}) = k\,\mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{Id} + 2\mu \mathrm{dev}(\boldsymbol{\varepsilon})\,, \tag{2.107}$$

with k, μ the bulk and shear modulus, $\mathrm{tr}(\boldsymbol{\varepsilon}) = \varepsilon_{ii}$ the trace of a tensor and $\mathrm{dev}(\boldsymbol{\varepsilon}) = \boldsymbol{\varepsilon} - \mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{Id}/3$ the deviator operator. For the plasticity model, we use the classical J_2-flow model with the von Mises yield function and isotropic/kinematic linear hardening

$$\begin{aligned} f(\mathbf{s}, \mathbf{q}) &= |\mathrm{dev}(\mathbf{s}) - \mathbf{q}_1| - \sqrt{\tfrac{2}{3}}(\sigma_Y - q_2)\,, \\ \Psi_p(\xi) &= \tfrac{1}{3}\bar{H}\xi_1^2 + \tfrac{1}{2}\bar{K}\xi_2^2\,, \end{aligned}$$

with $\xi = (\xi_1, \xi_2)$ and $\mathbf{q} = (\mathbf{q}_1, q_2), \mathrm{dev}(\mathbf{q}_1) = 0$ the strain-stress plastic variables and $\sigma_Y, \bar{H}, \bar{K}$ three model's constants. The von Mises yield function has the properties

$$\partial_{\mathbf{q}} f = (-\mathbf{n}, \sqrt{\tfrac{2}{3}})\,, \quad \partial_{\mathbf{s}} f = \mathbf{n}, \text{ with } \mathbf{n} = \frac{\mathrm{dev}(\mathbf{s}) - \mathbf{q}_1}{|\mathrm{dev}(\mathbf{s}) - \mathbf{q}_1|}\,,$$

which are readily verified and substitution into (2.104) yields the following equations

$$\begin{aligned} \boldsymbol{\varepsilon}^p_{n+1} &= \boldsymbol{\varepsilon}^p_n + \Delta\lambda \mathbf{n}_{n+1}\,, \\ \xi_{n+1} &= \xi_n + \Delta\lambda(-\mathbf{n}_{n+1}, \sqrt{\tfrac{2}{3}})\,, \\ \mathbf{s}_{n+1} &= k\,\mathrm{tr}(\boldsymbol{\varepsilon}_{n+1})\mathbf{Id} + 2\mu\,\mathrm{dev}(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1})\,, \\ \mathbf{q}_{n+1} &= -(\tfrac{2}{3}\bar{H}\xi_{1,n+1}, \bar{K}\xi_{2,n+1})\,, \\ f_{n+1} &= 0\,. \end{aligned}$$

In order to obtain a solution to these equations, we note that the normal vector $\mathbf{n}$ to the yield surface f does not depend on any values at t_{n+1} except for the strain $\boldsymbol{\varepsilon}_{n+1}$ which is a know quantity

$$\mathbf{n}_{n+1} = \frac{2\mu\,\mathrm{dev}(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_n) - \mathbf{q}_{1,n}}{|2\mu\,\mathrm{dev}(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_n) - \mathbf{q}_{1,n}|} = \frac{\mathrm{dev}(\mathbf{s}_{\mathrm{trial}}) - \mathbf{q}_{1,n}}{|\mathrm{dev}(\mathbf{s}_{\mathrm{trial}}) - \mathbf{q}_{1,n}|}\,,$$

with

$$\mathbf{s}_{\mathrm{trial}} = k\,\mathrm{tr}(\boldsymbol{\varepsilon}_{n+1})\mathbf{Id} + 2\mu\,\mathrm{dev}(\boldsymbol{\varepsilon}_{n+1}) - 2\mu\,\boldsymbol{\varepsilon}^p_n\,,$$

the stress of the trial solution (2.105). This decoupling is a property of the von Mises yield function and does not hold for general models. A final evaluation of $f_{n+1} = 0$ yields the value of $\Delta\lambda$

$$\Delta\lambda = \frac{|\mathrm{dev}(\mathbf{s}_{\mathrm{trial}}) - \mathbf{q}_{1,n}| + \sqrt{\frac{2}{3}}(\sigma_Y - q_{2,n})}{2\mu + \frac{2}{3}\bar{H} + \frac{2}{3}\bar{K}} = \frac{f_{\mathrm{trial}}}{2\mu + \frac{2}{3}\bar{H} + \frac{2}{3}\bar{K}}\,.$$

In summary, the stress to be defined as input to *SESES* is given by $\mathbf{s}_{n+1} = \mathbf{s}_{\mathrm{trial}}$ if $f_{\mathrm{trial}} < 0$, otherwise by

$$\mathbf{s}_{n+1} = \mathbf{s}_{\mathrm{trial}} - 2\mu \frac{f_{\mathrm{trial}}\mathbf{n}_{n+1}}{2\mu + \frac{2}{3}\bar{H} + \frac{2}{3}\bar{K}}\,.$$

To obtain an optimal convergence behavior, we need further the derivative of the stress $\mathbf{s}_{n+1}$ with respect to the strain $\boldsymbol{\varepsilon}_{n+1}$. In order to keep the same notation, we compute here directional derivatives $D(\bullet)[\mathbf{a}]$ with respect to a vector $\mathbf{a}$. By noting the properties $\mathrm{tr}(\mathrm{dev}(\mathbf{a})) = 0$ and $\mathbf{n}_{n+1} \cdot \mathrm{dev}(\mathbf{a}) = \mathbf{n}_{n+1} \cdot \mathbf{a}$, after some algebra we obtain

$$\begin{array}{rcl} D_{\boldsymbol{\varepsilon}_{n+1}} f_{\mathrm{trial}}[\mathbf{a}]) & = & 2\mu\, \mathbf{n}_{n+1} \cdot \mathbf{a}\,, \\ D_{\boldsymbol{\varepsilon}_{n+1}} \mathbf{s}_{\mathrm{trial}}[\mathbf{a}]) & = & k\,\mathrm{tr}(\mathbf{a})\mathbf{Id} + 2\mu\,\mathrm{dev}(\mathbf{a})\,, \\ D_{\boldsymbol{\varepsilon}_{n+1}} \mathbf{n}_{n+1}[\mathbf{a}]) & = & 2\mu \frac{\mathrm{dev}(\mathbf{a}) - \mathbf{n}_{n+1}\cdot(\mathbf{n}_{n+1}\cdot\mathbf{a})}{|\mathrm{dev}(\mathbf{s}_{\mathrm{trial}}) - \mathbf{q}_{1,n}|}\,, \\ D_{\boldsymbol{\varepsilon}_{n+1}} \mathbf{s}_{n+1}[\mathbf{a}]) & = & k\,\mathrm{tr}(\mathbf{a})\mathbf{Id} + 2\mu\,(1 - \frac{2\mu\Delta\lambda}{|\mathrm{dev}(\mathbf{s}_{\mathrm{trial}}) - \mathbf{q}_{1,n}|})\mathrm{dev}(\mathbf{a}) \\ & & -\frac{(2\mu)^2}{2\mu + \frac{2}{3}\bar{H} + \frac{2}{3}\bar{K}}(1 - \frac{f_{\mathrm{trial}}}{|\mathrm{dev}(\mathbf{s}_{\mathrm{trial}}) - \mathbf{q}_{1,n}|})\mathbf{n}_{n+1} \cdot (\mathbf{n}_{n+1} \cdot \mathbf{a})\,. \end{array} \tag{2.108}$$

Defining the model

Let us compute the plane strain perfect plastic example presented in [3]. The material parameters are given by Young's module $E_{\mathrm{mod}} = 7{\times}10^7$ Pa, Poisson's ratio $\nu = 0.2$, yield stress $\bar{\sigma}_Y = \sqrt{2/3}\,\sigma_Y = \sqrt{2/3} \times 0.24{\times}10^6$ Pa and zero hardening $\bar{H} = \bar{K} = 0$ so that we also have $q = \xi = 0$. The bulk and shear modulus are defined by $k = E_{\mathrm{mod}}/(3(1-2\nu)) = 3.888{\times}10^7$ Pa and $\mu = E_{\mathrm{mod}}/(2(1+\nu)) = 2.916{\times}10^7$ Pa. The *SESES* input file for this problem can be found at `example/J2Flow.s2d`. In a first step, we define an element field `StrainP` to store the plastic strain $\boldsymbol{\varepsilon}^p$ at each integration point and the global parameters `BULK_MOD`, `LAME_MUE` to define the bulk and shear modulus k, μ.

```
ElmtFieldDef StrainP(DofField Disp)[T2Z]
GlobalDef Const BULK_MOD  Pa
GlobalDef Const SHEAR_MOD Pa
GlobalDef Const YIELD     Pa
```

We then define a routine `PlasticStress` to specify the material parameter `Stress Law` defining both the local stress and the derivative with respect to the strain. For this perfect plasticity example and $f_{\mathrm{trial}} > 0$, the stress is given by

$$\mathbf{s}_{n+1} = k\,\mathrm{tr}(\boldsymbol{\varepsilon}_{n+1})\mathbf{Id} + \bar{\sigma}_Y \mathbf{n}_{n+1}\,,$$

and from (2.108) we have the derivative

$$D_{\boldsymbol{\varepsilon}_{n+1}} \mathbf{s}_{n+1}[\mathbf{a}] = k\,\mathrm{tr}(\mathbf{a})\mathbf{Id} + 2\mu\bar{\sigma}_Y \frac{\mathrm{dev}(\mathbf{a}) - \mathbf{n}_{n+1} \cdot (\mathbf{n}_{n+1} \cdot \mathbf{a})}{|\mathrm{dev}(\mathbf{s}_{\mathrm{trial}}) - \mathbf{q}_{1,n}|}\,.$$

At the end of this return mapping algorithm, we have to update the value of the plastic strain according to

$$\boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^p_n + \frac{f_{\mathrm{trial}}\mathbf{n}_{n+1}}{2\mu}\,.$$

If the step is elastic $f_{\mathrm{trial}} \leq 0$, we just define the isotropic and linear elastic law (2.107). The geometry and BCs are defined according to the reference.

Numerical Results

Before computing a solution, we must allocate the plastic strain field with the statement

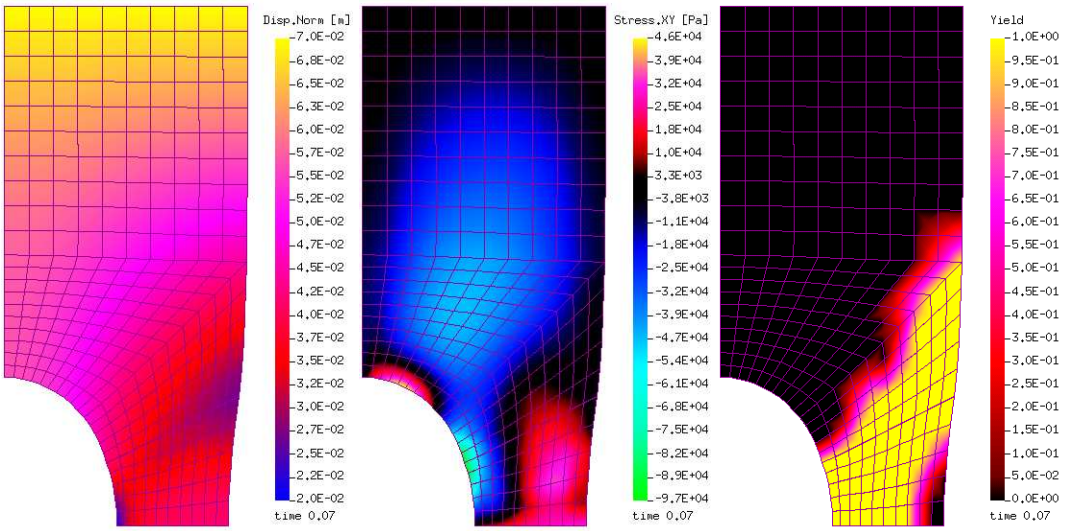

Figure 2.97: Displacement, shear stress and yield region for a vertical displacement of $0.007\,\mathrm{m}$.

```
Store StrainP(RestoreOnFailure)=zero
```

otherwise its value will always be zero. The BCs have been defined to be time dependent and the solution is computed in a series of stationary steps. The convergence of Newton's algorithm is optimal, but differently from the reference we need far more steps to avoid divergence. However, by using linear extrapolation of the solutions with respect to the pseudo time

```
Extrapolation  Linear
```

the number of steps can be reduced to 10. Fig. 2.97 shows the displacement $\mathbf{u}$, shear stress $\mathbf{s}_{xy}$ and yield region where $f(s) = 0$ holds for the solution at time $t = 0.07$ corresponding to a vertical displacement of the upper edge of $0.07\,\mathrm{m}$ with the bottom one being clamped in the vertical direction. An amplification factor of 20 has been used to visualize the displacement on the computational domain.

References

[1] J. LUBLINER, *Plasticity Theory*, Dover Publications, 2008.

[2] A. S. KHAN, S. HUANG, *Continuum Theory of Plasticity*, John Wiley & Sons, 1995.

[3] J.C. SIMO, T.J.R. HUGHES, *Computational Inelasticity*, Springer Verlag, 1998.

[4] J.C. SIMO, *Numerical Analysis and Simulation of Plasticity*, Handbook of Numerical Analysis by P. G. Ciarlet and J. L. Lions, Vol. VI, North-Holland, 1998.

[5] W. HAN, B.D. REDDY, *Plasticity, Mathematical Theory and Numerical Analysis*, Springer-Verlag, 1999.

2.22 Necking of a circular bar

In this example, we extend the previous infinitesimal elasto-plastic model to the finite strain case by considering the necking of a rod subjected to a large elongation. For deformations larger than $10-20\%$, the modeling of anisotropic plasticity is still an active research field and open questions still remain, whereas the isotropic case is much less controversial and is considered with this example. Here the theoretical framework may get quite involved and the reader is referenced to [1]-[3]-[4] for additional background informations. When modeling elasto-plasticity at finite strain, basically two different approaches are available. The first one is based on a priori additive decomposition of the strain rate into a plastic and elastic part combined with a hypoelastic law, whereas the second one is based on a multiplicative decomposition of the deformation gradient into a plastic and elastic part combined with a hyperelastic law. In the case of isotropy, both formulations can be shown to be equivalent. Historically the additive decomposition was the first to be developed and it is still widely used in commercial codes, whereas the multiplicative decomposition is considered more precise and better supported by physical arguments. In the following, we present the multiplicative approach for quasistationary elasto-plasticity in the form proposed by [5], where infinitesimal plastic models can be reused without changes and just some wrapper code interfacing the numerical solver is different. However, the important assumption of isotropy has to be made here, both for the elastic and plastic laws. In addition, we have a restriction on the form of the elastic law, which however does not hold, if one is ready to rewrite the infinitesimal plastic models.

Isotropic plasticity at finite strain

Let $\Omega \subset \mathbb{R}^3$ be the domain of a body at rest. Under actions of some quasi-static external forces, the body deforms and so let $\phi : \Omega \to \Omega^\phi = \phi(\Omega, t)$ be the unknown motion to be computed, $\mathbf{F} = \partial \mathbf{x}/\partial \mathbf{X}$ the deformation gradient with $\mathbf{X} \in \Omega$ a material point and $\mathbf{x} = \phi(\mathbf{X}, t) \in \Omega^\phi$ a spatial point. In the multiplicative decomposition, one postulates $\mathbf{F} = \mathbf{F_e} \cdot \mathbf{F_p}$ with $\mathbf{F_e}$ and $\mathbf{F_p}$ the elastic and plastic parts of the deformation gradient. The idea behind this decomposition is that for any small neighborhood U_x of $\mathbf{x} \in \phi(\mathbf{X})$, by removing all tractions acting on ∂U_x we have an elastic relaxation into a stress-free configuration with a remaining irreversible plastic deformation $\mathbf{F_p}$, a physical effect known as spring-back. The stress-free intermediate configuration is labeled $\bar{\Omega}$, however, since in general $\mathbf{F_p}$ nor $\mathbf{F_e}$ are gradients, we cannot define the stress-free configuration as $\bar{\Omega} = \phi^p(\Omega)$ with $\phi = \phi^e \circ \phi^p$ but just through the action of $\mathbf{F_p}$ or $\mathbf{F_e}$. Let for the moment assume, there is no plastic deformation $\mathbf{F_p} = 0$, then the whole elasto-plastic formulation should turn into an elastic formulation with $\bar{\Omega} = \Omega$. We use here a generic hyperelastic law with the Cauchy stress $\mathbf{s}$ given by the derivative of an elastic potential. Although material and spatial points are connected by the diffeomorphism ϕ, the material description is required to formulate anisotropic material laws which however is not the case in our example. Here, the second Piola-Kirchhoff (P2K) stress $\bar{\mathbf{S}} = \bar{J}\, \mathbf{F_e}^{-1} \cdot \mathbf{s} \cdot \mathbf{F_e}^{-\mathrm{T}}$ with $\bar{J} = \det \mathbf{F_e}$ is expressed by the derivative of an elastic potential $w(\mathbf{F_e})$ with respect to the elastic strain $\mathbf{F_e}$. By considering objectivity, one necessarily has a dependency of the potential from the symmetric right Cauchy strain $\bar{\mathbf{C}} = \mathbf{F_e}^{\mathrm{T}} \cdot \mathbf{F_e}$ and so we have $\bar{\mathbf{S}} = 2\partial w/\partial \bar{\mathbf{C}} = \partial w/\partial \bar{\mathbf{E}}$ with the Green-Lagrange strain defined by

$\bar{\mathbf{E}} = (\bar{\mathbf{C}} - \mathbf{Id})/2$. If we further restrict our attention to isotropic materials, then the potential w can only be a function of the invariants of $\bar{\mathbf{C}}$ which for this example are taken to be their eigenvalues. Since $\bar{\mathbf{C}}$ is positive definite, the eigenvalues are real and positive and we can write the spectral decomposition in the form $\bar{\mathbf{C}} = \sum_i \lambda_i^2 \bar{\mathbf{N}}_i \otimes \bar{\mathbf{N}}_i$ with $\lambda_i \in (0, \infty)$, λ_i^2 the eigenvalues and $\bar{\mathbf{N}}_i$ the orthonormal eigenvectors. The potential can now be expressed in the functional form $w = w(\lambda_i)$. In order to evaluate material laws expressed with respect to principal values, quite some numerical work is involved in computing the eigenpair decomposition at every integration point and so it may seem over killing to use the spectral formulation for isotropic materials. However, as we will seen, this is the only additional step required to upgrade an infinitesimal plastic model to the finite strain case. For this goal, we introduce the log-strains $\varepsilon_i = \log(\lambda_i)$ so that the stress is expressed by

$$\bar{\mathbf{S}} = 2\frac{\partial w(\lambda_i)}{\partial \bar{\mathbf{C}}} = \sum_i \frac{1}{\lambda_i}\partial_{\lambda_i} w \frac{\partial \lambda_i^2}{\partial \bar{\mathbf{C}}} = \sum_i \frac{\tau_i}{\lambda_i^2} \bar{\mathbf{N}}_i \otimes \bar{\mathbf{N}}_i \,, \tag{2.109}$$

with $\tau_i = \partial_{\varepsilon_i} w$. The push-forward of $\bar{\mathbf{S}}$ on Ω^ϕ defines the Kirchhoff stress $\boldsymbol{\tau} = \mathbf{F_e} \cdot \bar{\mathbf{S}} \cdot \mathbf{F_e}^{\mathrm{T}} = J\,\mathbf{s}$ given by $\boldsymbol{\tau} = \sum_i \tau_i \mathbf{n}_i \otimes \mathbf{n}_i$ with $\mathbf{n}_i = \mathbf{F_e} \cdot \bar{\mathbf{N}}_i / \lambda_i$ the orthonormal eigenvectors of $\boldsymbol{\tau}$. For a material description used here by the numerical algorithms, the Kirchhoff stress $\boldsymbol{\tau}$ or the intermediate P2K stress $\bar{\mathbf{S}}$ is pulled-back on Ω from Ω^ϕ or $\bar{\Omega}$ resulting in

$$\mathbf{S} = \sum_i \frac{\tau_i}{\lambda_i^2} \mathbf{N}_i \otimes \mathbf{N}_i \,, \tag{2.110}$$

with vectors $\mathbf{N}_i = \lambda_i \mathbf{F}^{-1} \cdot \mathbf{n}_i = \mathbf{F_p}^{-1} \cdot \bar{\mathbf{N}}_i$ not generally orthonormal anymore. In addition, we need a material description of the elasticity tensor $\partial\mathbf{S}/\partial\mathbf{E}$, with respect to the material Green-Lagrange strain $\mathbf{E}$. As shown in the appendix, for the intermediate configuration $\bar{\Omega}$, we have

$$\begin{aligned} \frac{\partial \bar{\mathbf{S}}}{\partial \bar{\mathbf{E}}} = & \sum_{i \neq j} \frac{\tau_i \lambda_j^2 - \tau_j \lambda_i^2}{\lambda_i^2 \lambda_j^2 (\lambda_i^2 - \lambda_j^2)} \bar{\mathbf{N}}_i \otimes \bar{\mathbf{N}}_j \otimes (\bar{\mathbf{N}}_i \otimes \bar{\mathbf{N}}_j + \bar{\mathbf{N}}_j \otimes \bar{\mathbf{N}}_i) + \\ & \sum_{i,j} \frac{\partial_{\varepsilon_j} \tau_i - 2\tau_i \delta_{ij}}{\lambda_i^2 \lambda_j^2} \bar{\mathbf{N}}_i \otimes \bar{\mathbf{N}}_i \otimes \bar{\mathbf{N}}_j \otimes \bar{\mathbf{N}}_j \,. \end{aligned} \tag{2.111}$$

By considering that the plastic strain $\mathbf{F_p}$ is frozen during the solution of a single time-step and does not depend upon $\mathbf{E}$, the relation $\partial\mathbf{S}/\partial\mathbf{E}$ follows from the above one by a simple pull-back, i.e. by replacing the vectors $\bar{\mathbf{N}}_i$ with pulled-back ones $\mathbf{N}_i = \mathbf{F_p}^{-1} \cdot \bar{\mathbf{N}}_i$.

We now need to identify a quantity representing the plastic flow and to formulate a rate equation in the stress-free configuration $\bar{\Omega}$. By writing the spatial velocity gradient in the form

$$\mathbf{l} = \frac{\partial \mathbf{v}}{\partial \mathbf{x}} = \dot{\mathbf{F}} \cdot \mathbf{F}^{-1} = \dot{\mathbf{F}}_\mathbf{e} \mathbf{F}_\mathbf{e}^{-1} + \mathbf{F_e} \cdot \dot{\mathbf{F}}_\mathbf{p} \cdot \mathbf{F_p}^{-1} \cdot \mathbf{F_e}^{-1} = \mathbf{l_e} + \mathbf{l_p} \,,$$

with $\mathbf{l_e} = \dot{\mathbf{F}}_\mathbf{e} \mathbf{F}_\mathbf{e}^{-1}$ and $\mathbf{l_p} = \mathbf{F_e} \cdot \dot{\mathbf{F}}_\mathbf{p} \cdot \mathbf{F_p}^{-1} \cdot \mathbf{F_e}^{-1}$, we may now identify the pull-back of $\mathbf{l_p}$ given by $\bar{\mathbf{L}}_\mathbf{p} = \dot{\mathbf{F}}_\mathbf{p} \cdot \mathbf{F_p}^{-1}$ as the plastic velocity gradient in $\bar{\Omega}$ where we have

$$\bar{\mathbf{L}} = \mathbf{F_e}^{-1} \cdot \mathbf{l} \cdot \mathbf{F_e} = \mathbf{F_e}^{-1} \cdot \dot{\mathbf{F}}_\mathbf{e} + \dot{\mathbf{F}}_\mathbf{p} \cdot \mathbf{F_p}^{-1} = \bar{\mathbf{L}}_\mathbf{e} + \bar{\mathbf{L}}_\mathbf{p} \,,$$

with $\bar{\mathbf{L}}_\mathbf{e} = \mathbf{F}_\mathbf{e}^{-1} \cdot \dot{\mathbf{F}}_\mathbf{e}$ the elastic velocity gradient in $\bar{\Omega}$. The rate of mechanical work or stress power in the spatial configuration is known to be $\mathcal{P} = \mathbf{s} : \mathbf{l} = \mathbf{s} : \mathrm{sym}(\mathbf{l})$. By just considering the plastic part with respect to the volume of Ω, we have

$$\mathcal{P}^p_\Omega = J\mathbf{s} : \mathbf{l}_\mathbf{p} = J\mathrm{tr}(\mathbf{F}_\mathbf{e} \cdot \bar{\mathbf{S}} \cdot \mathbf{F}_\mathbf{e}^{\mathrm{T}} \cdot \mathbf{F}_\mathbf{e} \cdot \bar{\mathbf{L}}_\mathbf{p} \cdot \mathbf{F}_\mathbf{e}^{-1}) = \bar{\mathbf{S}} : (\bar{\mathbf{C}} \cdot \bar{\mathbf{L}}_\mathbf{p}) = (\bar{\mathbf{C}} \cdot \bar{\mathbf{S}}) : \bar{\mathbf{L}}_\mathbf{p} \, .$$

By the assumption of isotropy, from (2.109) we see that the tensors $\bar{\mathbf{C}}$ and $\bar{\mathbf{S}}$ are coaxial, therefore they commutes and the matrix $\bar{\mathbf{C}} \cdot \bar{\mathbf{S}} = \bar{\mathbf{S}} \cdot \bar{\mathbf{C}} = (\bar{\mathbf{C}} \cdot \bar{\mathbf{S}})^{\mathrm{T}}$ is symmetric. So for isotropy, the asymmetric part $\mathrm{asym}(\bar{\mathbf{L}}_\mathbf{p}) = (\bar{\mathbf{L}}_\mathbf{p} - \bar{\mathbf{L}}_\mathbf{p}^{\mathrm{T}})/2$ does not dissipate and does not need to be specified. At this point we proceed as for the infinitesimal case, by expressing the plasticity law with respect to the stress-free configuration $\bar{\Omega}$. We assume there are some internal variables q and ξ related by a phenomenological law $q = q(\xi)$ and that together with $\bar{\mathbf{S}}$, they uniquely determine the yield condition $f(\bar{\mathbf{S}}, q) = 0$ with f a phenomenological convex yield surface. By considering the model of associative plasticity based on the principle of maximal plastic dissipation, which is a generally accepted model for hardening materials without softening, the evolution of the plastic strain $\mathrm{sym}(\bar{\mathbf{C}} \cdot \bar{\mathbf{L}}_\mathbf{p})$ and internal variables $\dot{\xi}$ is normal to the yield surface and we have the Kuhn-Tucker equations

$$\mathrm{sym}(\bar{\mathbf{C}} \cdot \bar{\mathbf{L}}_\mathbf{p}) = \dot{\lambda} \frac{\partial f}{\partial \bar{\mathbf{S}}} \, , \quad \dot{\xi} = \dot{\lambda} \frac{\partial f}{\partial q} \, , \quad \dot{\lambda} \geq 0 \, , \quad f(\bar{\mathbf{S}}, q) \leq 0 \, , \quad \dot{\lambda} f(\bar{\mathbf{S}}, q) = 0 \, , \tag{2.112}$$

with $\dot{\lambda}$ the plastic multiplier to be determined by the additional consistency condition $\dot{\lambda} \dot{f}(\bar{\mathbf{S}}, q) = 0$. Since for isotropy $\bar{\mathbf{C}}$ and $\bar{\mathbf{S}}$ commutes, similarly we can obtain $\mathrm{sym}(\bar{\mathbf{C}} \cdot \bar{\mathbf{L}}_\mathbf{p}^{\mathrm{T}}) = \dot{\lambda} \partial_{\bar{\mathbf{S}}} f$ and since $\mathrm{asym}(\bar{\mathbf{L}}_\mathbf{p})$ is unused, we can choose $\mathrm{asym}(\bar{\mathbf{L}}_\mathbf{p}) = 0$ in order to get $\mathrm{sym}(\bar{\mathbf{C}} \cdot \bar{\mathbf{L}}_\mathbf{p}) = \bar{\mathbf{C}} \cdot \bar{\mathbf{L}}_\mathbf{p}$ and the property of $\bar{\mathbf{C}}$, $\bar{\mathbf{L}}_\mathbf{p}$, $\partial_{\bar{\mathbf{S}}} f$ being three coaxial matrices.

The push-forward of (2.112) gives $\mathbf{l}_\mathbf{p} = \dot{\lambda} \partial f / \partial \boldsymbol{\tau}$ and if the yield surface is just a function of the deviatoric stress $\mathrm{dev}(\boldsymbol{\tau})$ with $\mathrm{dev}(\bullet) = (\bullet) - \mathbf{Id}\mathrm{tr}(\bullet)/3$ the deviator operator, then we have $\partial f / \partial \boldsymbol{\tau} = \mathrm{dev}[\partial f(\mathrm{dev}(\boldsymbol{\tau}))/\partial \mathrm{dev}(\boldsymbol{\tau})]$ and so $\mathrm{tr}(\mathbf{D}_\mathbf{p}) = \dot{\lambda} \mathrm{tr}(\partial f / \partial \boldsymbol{\tau}) = 0$. For a matrix $\mathbf{A}$, we have the relation $\partial \det(\mathbf{A}) / \partial \mathbf{A} = \det(\mathbf{A}) \cdot \mathbf{A}^{-\mathrm{T}}$ and so the rate of plastic volume is given by $\dot{J}^p = (d/dt) \det \mathbf{F}_\mathbf{p} = J^p \mathrm{tr}(\dot{\mathbf{F}}_\mathbf{p} \mathbf{F}_\mathbf{p}^{-1}) = J^p \mathrm{tr}(\bar{\mathbf{L}}_\mathbf{p}) = J^p \mathrm{tr}(\mathbf{F}_\mathbf{e} \cdot \bar{\mathbf{L}}_\mathbf{p} \cdot \mathbf{F}_\mathbf{e}^{-1}) = J^p \mathrm{tr}(\mathbf{l}_\mathbf{p}) = J^p \mathrm{tr}(\mathbf{D}_\mathbf{p}) = 0$ showing that yield surfaces of the form $f(\mathrm{dev}(\boldsymbol{\tau}))$ have an isochoric plastic flow which is typical for metal plasticity.

Before proceeding further, we have to decide which internal variables to use and as first we note that since the implementation is working with the displacement $\mathbf{u} = \mathbf{x} - \mathbf{X}$ as unknown, the total deformation $\mathbf{F} = \mathbf{F}_\mathbf{e} \cdot \mathbf{F}_\mathbf{p} = \partial \mathbf{x} / \partial \mathbf{X}$ is given as well as $\mathbf{C} = \mathbf{F}^{\mathrm{T}} \cdot \mathbf{F}$. In (2.112), the time derivative is directly associated with the internal variables ξ, so that it is natural to store and update ξ instead of q. For the plastic flow, the natural choice is to use $\mathbf{F}_\mathbf{p}$ so that together with $\bar{\mathbf{L}}_\mathbf{p} = \dot{\mathbf{F}}_\mathbf{p} \cdot \mathbf{F}_\mathbf{p}^{-1}$, we have

$$\dot{\mathbf{F}}_\mathbf{p} = \dot{\lambda} \bar{\mathbf{C}}^{-1} \cdot \frac{\partial f}{\partial \bar{\mathbf{S}}} \cdot \mathbf{F}_\mathbf{p} \, . \tag{2.113}$$

By considering the pull-back $\partial f / \partial \bar{\mathbf{S}} = \mathbf{F}_\mathbf{e}^{\mathrm{T}} \cdot \partial f / \partial \boldsymbol{\tau} \cdot \mathbf{F}_\mathbf{e}$ and the relation $\dot{\mathbf{F}}_\mathbf{p} = -\mathbf{F}_\mathbf{p} \cdot (d/dt) \mathbf{F}_\mathbf{p}^{-1} \cdot \mathbf{F}_\mathbf{p}$ we also have

$$(d/dt) \mathbf{F}_\mathbf{p}^{-1} = -\dot{\lambda} \mathbf{F}^{-1} \cdot \frac{\partial f}{\partial \boldsymbol{\tau}} \cdot \mathbf{F} \cdot \mathbf{F}_\mathbf{p}^{-1} \, .$$

Another possible choice as internal variables for the plastic flow are the symmetric tensors $\mathbf{C}_\mathbf{p} = \mathbf{F}_\mathbf{p}^{\mathrm{T}} \cdot \mathbf{F}_\mathbf{p}$ or $\mathbf{C}_\mathbf{p}^{-1}$ which are independent from any rotation associated

with $\mathbf{F_p}$. By considering the symmetry of $\bar{\mathbf{C}}$ and $\partial f/\partial \bar{\mathbf{S}}$, together with $\dot{\mathbf{C}}_\mathbf{p} = -\mathbf{C_p} \cdot (d/dt)\mathbf{C}_\mathbf{p}^{-1} \cdot \mathbf{C_p}$, from (2.113) we have

$$(d/dt)\mathbf{C}_\mathbf{p}^{-1} = -2\dot{\lambda}\mathbf{F}_\mathbf{p}^{-1} \cdot \mathrm{sym}(\bar{\mathbf{C}}^{-1} \cdot \frac{\partial f}{\partial \bar{\mathbf{S}}}) \cdot \mathbf{F}_\mathbf{p}^{-\mathrm{T}} \stackrel{!}{=} -2\dot{\lambda}\mathbf{F}^{-1} \cdot \frac{\partial f}{\partial \boldsymbol{\tau}} \cdot \mathbf{F} \cdot \mathbf{C}_\mathbf{p}^{-1}\,, \qquad (2.114)$$

where the last equality just holds if we have $\mathrm{sym}(\bar{\mathbf{C}}^{-1}\cdot\partial f/\partial\bar{\mathbf{S}}) = \bar{\mathbf{C}}^{-1}\cdot\partial f/\partial\bar{\mathbf{S}}$ or equivalently $\bar{\mathbf{C}}^{-1} \cdot \partial f/\partial\bar{\mathbf{S}} = \partial f/\partial\bar{\mathbf{S}} \cdot \bar{\mathbf{C}}^{-1}$. This is indeed the case for isotropy, where $f(\bar{\mathbf{S}})$ can only depend on the eigenvalues $\mu_i = \tau_i/\lambda_i^2$ of the spectral representation (2.109) of $\bar{\mathbf{S}}$. The relation $\partial f/\partial\bar{\mathbf{S}} = (\partial f/\partial\mu_i)\bar{\mathbf{N}}_i \otimes \bar{\mathbf{N}}_i$ shows that $\bar{\mathbf{S}}$ and $\partial f/\partial\bar{\mathbf{S}}$ are coaxial and from (2.109), this is also the case for $\bar{\mathbf{C}}^{-1}$ and $\bar{\mathbf{S}}$, therefore $\bar{\mathbf{C}}^{-1}$ and $\partial f/\partial\bar{\mathbf{S}}$ are simultaneously diagonalizable and they commute.

Several numerical algorithms are available to integrate (2.114) and it is important to note that they must all be formulated with respect to the fixed stress-free configuration $\bar{\Omega}$. By applying the first order implicit solution $z_{n+1} = \exp(a_{n+1})z_n$ of $\dot{z}(t) = a(t)z(t)$ on $t \in [t_n, t_{n+1}]$ to (2.114), we obtain

$$\mathbf{C}_{\mathbf{p},n+1}^{-1} = \mathbf{F}_{n+1}^{-1} \cdot \exp(-2\Delta\lambda\partial_{\boldsymbol{\tau}} f_{n+1}) \cdot \mathbf{F}_{n+1} \cdot \mathbf{C}_{\mathbf{p},n}^{-1}\,. \qquad (2.115)$$

Since $\det(\exp(\mathbf{A})) = \exp(\mathrm{tr}(\mathbf{A}))$ holds for any matrix $\mathbf{A}$, see [2], if the plastic flow is pressure insensitive $f(\boldsymbol{\tau}) = f(\mathrm{dev}(\boldsymbol{\tau}))$, then $\partial_{\boldsymbol{\tau}} f = \mathrm{dev}(\partial_{\boldsymbol{\tau}} f)$ and $\mathrm{tr}(\mathrm{dev}(\partial_{\boldsymbol{\tau}} f)) = 0$, so that there is exact numerical conservation of the plastic volume $\det(\mathbf{F_p}) = \det(\mathbf{C_p})^{-1/2}$. This condition is generally required on the onset for metal plasticity and it is the reason for using here the exponential approximation. Now that we have performed the time-integration, we can change the formulation to a spatial description, which is the preferred one to formulate the plastic laws. From the relation $\lambda_i^2\mathbf{F_e}\cdot\bar{\mathbf{N}}_i = \mathbf{F_e}\cdot\bar{\mathbf{C}}\cdot\bar{\mathbf{N}}_i = (\mathbf{F_e}\cdot\mathbf{F_e^{\mathrm{T}}})\cdot\mathbf{F_e}\cdot\bar{\mathbf{N}}_i$, we note that the elastic left Cauchy strain $\mathbf{b_e} = \mathbf{F_e}\cdot\mathbf{F_e^{\mathrm{T}}} = \mathbf{F}\cdot\mathbf{C_p^{-1}}\cdot\mathbf{F^{\mathrm{T}}} = \sum_i \lambda_i^2 \mathbf{n}_i \otimes \mathbf{n}_i$ has the same eigenvalues of $\bar{\mathbf{C}}$ and is coaxial with $\boldsymbol{\tau}$. As for the infinitesimal case, for the time integration of the Kuhn-Tucker equations (2.112), we apply an operator split based on a first trial step followed by a return-to-yield step, for details see [5]. Let the deformation together with the internal variables $(\mathbf{F}_n, \xi_n, \mathbf{C}_{\mathbf{p},n}^{-1})$ be given at time t_n together with a time step Δt. At time $t_{n+1} = t_n + \Delta t$, the deformation $\mathbf{F}_{n+1}$ is known and we are therefore seeking $(\xi_{n+1}, \mathbf{C}_{\mathbf{p},n+1}^{-1})$. If the trail principal stresses $\boldsymbol{\tau}_{\mathrm{trail},i}$ computed with the eigenvalues $\lambda_{\mathrm{trial},i}^2$ of the trial strain $\mathbf{b}_{\mathbf{e},\mathrm{trial}} = \mathbf{F}_{n+1}\cdot\mathbf{C}_{\mathbf{p},n}^{-1}\cdot\mathbf{F}_{n+1}^{\mathrm{T}}$ are in the elastic range $f(\boldsymbol{\tau}_{\mathrm{trail}}) \le 0$, then $(\mathbf{F}_{n+1}, \xi_{n+1} = \xi_n, \mathbf{C}_{\mathbf{p},n+1}^{-1} = \mathbf{C}_{\mathbf{p},n}^{-1})$ is the solution at t_{n+1}. Otherwise we have to solve the following system to determine $\mathbf{b}_{\mathbf{e},n+1}$, ξ_{n+1} and the plastic multiplier $\Delta\lambda = \dot{\lambda}\Delta t$

$$\begin{aligned}
\mathbf{b}_{\mathbf{e},n+1} &= \exp(-2\Delta\lambda\partial_\tau f_{n+1}) \cdot \mathbf{b}_{\mathbf{e},\mathrm{trial}}\,,\\
\xi_{n+1} &= \xi_n + \Delta\lambda\partial_q f_{n+1}\,,\\
f_{n+1} &= f(\boldsymbol{\tau}(\mathbf{b}_{\mathbf{e},n+1}), q(\xi_{n+1})) = 0\,,
\end{aligned}$$

and update the new values of ξ_{n+1} and $\mathbf{C}_{\mathbf{p},n+1}^{-1} = \mathbf{F}_{n+1}^{-1}\cdot\mathbf{b}_{\mathbf{e},n+1}\cdot\mathbf{F}_{n+1}^{-\mathrm{T}}$. At this point, we use the coaxiality of $\mathbf{b_e}$, $\boldsymbol{\tau}$ and $\partial_\tau f$ stemming from the isotropy assumption to proceed further in the solution of the above equations. Let use the *unbold* notation τ, ε for the vectors τ_i and $\varepsilon_i = \log(\lambda_i)$, $i = 1\ldots 3$. From the first equation, we see that $\mathbf{b}_{\mathbf{e},n+1}$ is coaxial with $\mathbf{b}_{\mathbf{e},\mathrm{trial}}$ and by taking the log on both sides, we are left with

$$\begin{aligned}
\varepsilon_{n+1} &= -\Delta\lambda\partial_\tau f_{n+1} + \varepsilon_{\mathrm{trial}}\,,\\
\xi_{n+1} &= \xi_n + \Delta\lambda\partial_q f_{n+1}\,,\\
f_{n+1} &= f(\tau(\varepsilon_{n+1}), q(\xi_{n+1})) = 0\,,
\end{aligned}$$

which are formally equivalent to the equations of the infinitesimal case. If we do not consider the trivial equation for ξ_{n+1}, we have to solve four equations for the unknowns $\Delta\lambda$ and ε_{n+1}, generally solved by a Newton-Raphson algorithm. In order to compute the elasticity tensor in material coordinates (2.111), we need the derivatives $\partial_\varepsilon \tau$ which in our case are actually $\partial_{\varepsilon_{\rm trial}} \tau$ and are computed as follows. Let η be a component of $\varepsilon_{\rm trial}$, by defining $\Pi = (\tau_{n+1}, q_{n+1})$, $\Sigma = (-\varepsilon_{n+1}, \xi_{n+1})$, $\Sigma_0 = (-\varepsilon_{\rm trial}, \xi_n)$, $\partial f = (\partial_\tau, \partial_q) f$ and the matrix G by the relation $\partial_\eta \Pi = -G \partial_\eta \Sigma$, the first two equations are given by $\Sigma = \Delta\lambda (\partial f)^{\rm T} + \Sigma_0$ and by taking the η-derivative, we have $G\partial_\eta \Sigma = Z((\partial f)^{\rm T} \partial_\eta \Delta\lambda + \partial_\eta \Sigma_0)$ with $Z = G(1+\Delta\lambda \partial^2 f G)^{-1} = (G^{-1} + \Delta\lambda \partial^2 f)^{-1}$. From $f_{n+1} = 0$, we obtain $\partial_\eta f_{n+1} = \partial f G \partial_\eta \Sigma = 0$ and by contracting the previous value of $G\partial_\eta \Sigma$ on the left with ∂f yields an equation for $\partial_\eta \Delta\lambda$ and the final result is

$$\partial_\eta \Pi = -G \partial_\eta \Sigma = \frac{Z (\partial f)^{\rm T} (\partial f) Z}{(\partial f) Z (\partial f)^{\rm T}} \partial_\eta \Sigma_0 - Z \partial_\eta \Sigma_0 \,,$$

with the derivatives $\partial_\eta \tau_{n+1}$ found in the first row. Since the relation $q = q(\xi)$ is generally derived by a potential through differentiation, the matrix $\partial_\xi q$ is symmetric and so the matrices G and Z as well. The matrix G has a block diagonal format and if this is also the case for $\partial^2 f$, then Z is block diagonal too. By letting Z_τ, Z_q be the two diagonal blocks of Z and since $\partial_\eta \Sigma_{0q} = \partial_\eta \xi_n = 0$, we arrive at

$$\frac{\partial \tau_{n+1}}{\partial \varepsilon_{\rm trial}} = -\frac{(\partial_\tau f)^{\rm T} Z_\tau^2 (\partial_\tau f)}{(\partial_\tau f) Z_\tau (\partial_\tau f)^{\rm T} + (\partial_q f) Z_q (\partial_q f)^{\rm T}} + Z_\tau \,. \tag{2.116}$$

For our strain-driven formulation, we assume that at the beginning of a time-step t_{n+1}, the data is given $(\mathbf{F}_{n+1}, \xi_n, \mathbf{C}_{\mathbf{p},n}^{-1})$ and one looks for $(\mathbf{F}_{n+1}, \xi_{n+1}, \mathbf{C}_{\mathbf{p},n+1}^{-1})$. However, we know from objectivity arguments, that material laws can only depends from right Cauchy strain $\mathbf{C} = \mathbf{F}^{\rm T} \cdot \mathbf{F}$ and therefore a formulation working with the data $(\mathbf{C}_{n+1}, \xi_n, \mathbf{C}_{\mathbf{p},n}^{-1})$ must be equivalent. The point here is that the strain $\mathbf{C}$ or $\mathbf{E} = (\mathbf{C} - \mathbf{Id})/2$ is always given when evaluating material laws, not necessarily the deformation gradient $\mathbf{F}$, since numerically the symmetrized strain forms may not be directly obtained from $\mathbf{F} = \partial\phi/\partial\mathbf{X}$, but instead from internal variables. The eigen problem $\mathbf{F} \cdot \mathbf{C}_\mathbf{p}^{-1} \cdot \mathbf{F}^{\rm T} \cdot \mathbf{n}_i = \lambda_i^2 \mathbf{n}_i$ is equivalent to $\mathbf{C}_\mathbf{p}^{-1} \cdot \mathbf{F}^{\rm T}\mathbf{F} \cdot \mathbf{F}^{-1} \cdot \mathbf{n}_i = \lambda_i^2 \mathbf{F}^{-1} \cdot \mathbf{n}_i$ or $\mathbf{C}_\mathbf{p}^{-1} \cdot \mathbf{C} \cdot \mathbf{N}_i = \lambda_i^2 \mathbf{N}_i$. Indeed the eigenvalues λ_i^2 and eigenvectors $\mathbf{N}_i$ is all what is needed to evaluate any isotropic plastic laws, but now they are determined by a non-symmetric eigen problems. By considering that $\mathbf{C}_\mathbf{p}^{-1}$ is positive definite, the numerical work is at most doubled. One can compute $\mathbf{C}_\mathbf{p}^{-0.5}$ by spectral decomposition by running a first symmetric eigen-problem and then use $\mathbf{C}_\mathbf{p}^{-0.5}$ to evaluate the second symmetric eigen-problem $\mathbf{C}_\mathbf{p}^{-0.5} \cdot \mathbf{C} \cdot \mathbf{C}_\mathbf{p}^{-0.5} \cdot \breve{\mathbf{N}}_i = \lambda_i^2 \breve{\mathbf{N}}_i$ with $\mathbf{N}_i = \mathbf{C}_\mathbf{p}^{-0.5} \cdot \breve{\mathbf{N}}_i$. The first eigen-problem can also be replaced by a quadratically convergent Denman-Beavers iteration directly computing $\mathbf{C}_\mathbf{p}^{-0.5}$ and running a little bit faster.

Numerical model

Up to now the model has been kept quite general, the only assumptions have been made are that of a multiplicative decomposition of the deformation, isotropic material laws and the special choice of a first-order time-integration algorithm involving an exponential map. The plastic models need now to be implemented in *SESES* as a

material law for non-linear elasticity where one has to define the P2K stress $\mathbf{S}$ as function of the Green-Lagrange strain $\mathbf{E}$ together with the symmetric derivatives $\partial\mathbf{S}/\partial\mathbf{E}$. Since the relations (2.110)-(2.111) together with the spectral decomposition of $\mathbf{b_e}$ are model independent, we see that each isotropic plastic model just needs to define the principal stresses τ together with the derivatives $\partial\tau/\partial\varepsilon$ as function of the log-strains ε. For our example, we use the following quadratic potential

$$w = \frac{1}{2}\lambda(\sum_i \varepsilon_i)^2 + \mu(\sum_i \varepsilon_i^2)\,, \tag{2.117}$$

with λ, μ the Lame's constants of the infinitesimal theory. For the stress, we have

$$\tau = \partial w/\partial\varepsilon = \lambda(\sum_i \varepsilon_i)(1,1,1)^{\mathrm{T}} + 2\mu\varepsilon = (\kappa\,\mathrm{ONE} + 2\mu\,\mathrm{DEV})\varepsilon\,, \tag{2.118}$$

with $\kappa = \lambda + 2/3\mu$ the bulk modulus, $\mathrm{ONE} = (1,1,1)(1,1,1)^{\mathrm{T}}$ and $\mathrm{DEV} = \mathrm{Id} - (1/3)\mathrm{ONE}$ or in tensor notation

$$\boldsymbol{\tau} = \sum_i \tau_i \mathbf{n}_i \otimes \mathbf{n}_i = \mathbf{Id}\kappa\mathrm{tr}(\boldsymbol{\varepsilon}) + 2\mu\mathrm{dev}(\boldsymbol{\varepsilon})\,, \tag{2.119}$$

with the log-strain tensor $\boldsymbol{\varepsilon} = \sum_i \varepsilon_i \mathbf{n}_i \otimes \mathbf{n}_i$. For small strains $\bar{\mathbf{C}} \approx \mathbf{Id}$, the log-strain are $\varepsilon_i = \log(\lambda_i) = \log(\sqrt{2\nu_i + 1}) \approx \nu_i$ with $\nu_i = (\lambda_i^2 - 1)/2$ the eigenvalues of the Green-Lagrange strain $\bar{\mathbf{E}} = (\bar{\mathbf{C}} - \mathbf{Id})/2$ which is coaxial with $\bar{\mathbf{C}}$. Since $\lambda_i \approx 1$, we have $\bar{\mathbf{S}} = \mathbf{Id}\lambda\mathrm{tr}(\bar{\mathbf{E}}) + 2\mu\bar{\mathbf{E}}$ which is the standard St. Venant-Kirchhoff isotropic material law at infinitesimal strains. If a correct behavior is given at small strains, for extremely large ones, the choice of (2.117) is questionable, however, we have to consider that the appearance of yield will limit the elastic range. Because the stress (2.119) has the same form of the infinitesimal case, previous infinitesimal plastic models can be reused here without changes by just considering the principal log-strains ε instead of the infinitesimal ones. In order to quickly comes to an end, we reuse here the von Mises J_2-flow model of the previous example just accounting for linear isotropic hardening

$$f(\tau, q) = |\mathrm{DEV}(\tau)| - \sqrt{\tfrac{2}{3}}(\sigma_Y - q)\,,\quad q = -\bar{K}\xi\,,$$

with σ_Y, $\bar{K}$ two material parameters. If for the trial stress $\tau_{\mathrm{trial}} = \tau(\varepsilon_{\mathrm{trial}})$ computed by (2.118), we have $f_{\mathrm{trial}} = f(\tau_{\mathrm{trial}}, q) \le 0$, then we return $\tau_{n+1} = \tau_{\mathrm{trial}}$ and $\partial\tau_{n+1}/\partial\varepsilon_{\mathrm{trial}} = \kappa\,\mathrm{ONE} + 2\mu\,\mathrm{DEV}$. Otherwise, we return

$$\begin{aligned}
\tau_{n+1} &= \tau_{\mathrm{trial}} - 2\mu\Delta\lambda n\,,\\
\frac{\partial\tau_{n+1}}{\partial\varepsilon_{\mathrm{trial}}} &= \kappa\,\mathrm{ONE} + 2\mu\,\mathrm{DEV} - (2\mu)^2\frac{\Delta\lambda}{\bar{\tau}}(\mathrm{DEV} - n\,n^{\mathrm{T}}) - \frac{(2\mu)^2}{2\mu + \frac{2}{3}\bar{K}}n\,n^{\mathrm{T}}\,,
\end{aligned}$$

with $\Delta\lambda = f_{\mathrm{trial}}/(2\mu + \frac{2}{3}\bar{K})$, $\bar{\tau} = |\mathrm{DEV}(\tau_{\mathrm{trial}})|$ and $n = \mathrm{DEV}(\tau_{\mathrm{trial}})/\bar{\tau}$. Clearly, the relation $\partial\tau_{n+1}/\partial\varepsilon_{\mathrm{trial}}$ may also be obtained by the general method of (2.116), but the algebra get a little bit involved. By noting that $G_\tau = \kappa\,\mathrm{ONE} + 2\mu\,\mathrm{DEV}$ and $\mathrm{DEV}(\tau_{\mathrm{n+1}})$ is parallel to $\mathrm{DEV}(\tau_{\mathrm{trial}})$ with $|\mathrm{DEV}(\tau_{\mathrm{n+1}})| = \bar{\tau} - 2\mu\Delta\lambda$, we have $\partial_\tau f = \mathrm{DEV}(\tau_{\mathrm{n+1}})/|\mathrm{DEV}(\tau_{\mathrm{n+1}})| = n^{\mathrm{T}}$ and $\partial_\tau^2 f = |\mathrm{DEV}(\tau_{\mathrm{n+1}})|^{-1}(\mathrm{Id} - n\,n^{\mathrm{T}})\mathrm{DEV} = (\bar{\tau} -$

$2\mu\Delta\lambda)^{-1}(\mathrm{DEV} - n\,n^{\mathrm{T}})$ giving

$$\begin{aligned} Z_\tau &= G_\tau(\mathrm{Id} + \Delta\lambda\partial^2_\tau f\, G_\tau)^{-1} = G_\tau\left(\mathrm{Id} + \frac{2\mu\Delta\lambda(\mathrm{DEV} - n\,n^{\mathrm{T}})}{\bar\tau - 2\mu\Delta\lambda}\right)^{-1} \\ &\stackrel{!}{=} G_\tau\left(\mathrm{Id} - \frac{2\mu\Delta\lambda}{\bar\tau}(\mathrm{DEV} - n\,n^{\mathrm{T}})\right) \\ &= \kappa\,\mathrm{ONE} + 2\mu\,\mathrm{DEV} - \frac{(2\mu)^2\Delta\lambda}{\bar\tau}(\mathrm{DEV} - n\,n^{\mathrm{T}})\,, \end{aligned}$$

where we have used a non-trivial matrix identity to compute the inverse just holding for a vector n with the properties $|n| = 1$ and $\mathrm{DEV}n = n$ as here the case. Together with $Z_\tau\partial_\tau f^{\mathrm{T}} = 2\mu n$, $\partial_\tau f Z_\tau \partial_\tau f^{\mathrm{T}} = 2\mu$ and $\partial_q f Z_q \partial_q f^{\mathrm{T}} = 2/3\bar K$, the formula (2.116) yields the above result.

This elastic-plastic law together with the computations of an axis-symmetric rod subjected to a large elongation and showing a typical necking behavior can be found at `example/Necking.s2d`. The value of the plastic deformation $\mathbf{F_p}$ is stored in the element field `CpInvMOneEF` in the form of $\mathbf{C_p^{-1}} - \mathbf{Id}$, the shift with the unit matrix is used to avoid numerical cancellation at infinitesimal strains. The value of the equivalent plastic strain ξ is stored in the element field `XiEF`. The above relations τ_{n+1}, $\partial\tau_{n+1}/\partial\varepsilon_{\mathrm{trial}}$ together with the conditional update of the internal variables are coded in the input routine `J2FlowPrincipal`, passed to the built-in routine `ElastLogStrain` taking care of the interfacing relationships (2.110)-(2.111). To this routine, we also pass the element field `CpInvMOneEF` used to compute the logarithmic strain $\varepsilon_{\mathrm{trial},i} = \log(\lambda_i)$ before calling our plastic model. The update of the plastic variables is done in the routine `J2FlowPrincipal` in the form of $\mathbf{C}^{-1}_{\mathbf{p},n+1} - \mathbf{Id} = \mathbf{F}^{-1}_{n+1}\cdot\mathbf{b}_{\mathbf{e},n+1}\cdot\mathbf{F}^{-\mathrm{T}}_{n+1} - \mathbf{Id} = \sum_i \exp(2\varepsilon_{n+1,i})/\lambda_i^2\,\mathbf{N}_i\otimes\mathbf{N}_i - \mathbf{Id}$ and $\xi_{n+1} = \xi_n + \Delta\lambda$. The updates are first stored in additional buffer element fields with suffix `u`, which are first copied into the permanent ones at the end of a successful solution step.

In addition, we supply the routine `J2FlowFiniteStrain` which may be used as a replacement to the built-in call of `ElastLogStrain`. It shows in details all numerical operations done by `ElastLogStrain` and it is completely equivalent. This routine computes the spectral decomposition of $\mathbf{b}_{\mathbf{e},\mathrm{trial}}$, performs a call of our plastic model `J2FlowPrincipal` and implements the relations (2.110)-(2.111). The numerical point of interest is the possible numerical cancellation in (2.111) when computing the term $(\tau_i\lambda_j^2 - \tau_j\lambda_i^2)/(\lambda_i^2 - \lambda_j^2)$ with $\lambda_i \to \lambda_j$. In this case, we use a symmetrized Hôpital's rule given by

$$\frac{\lambda_i\partial_{\varepsilon_j}\tau_j}{4\lambda_j} + \frac{\lambda_j\partial_{\varepsilon_i}\tau_i}{4\lambda_i} - \frac{\partial_{\varepsilon_i}\tau_j + \tau_i + \tau_j}{2}\,.$$

The geometrical setup of this example is quite simple, we define an axis-symmetric problem with a rectangular domain and set Dirichlet BCs on both ends of the rod to prescribe the elongation. We then compute solutions incrementally by increasing the displacement at the BCs with respect to a pseudo time and since the integration process is quite stable, small or large steps yield similar results. The necking breakdown is approached with a *trial&error* algorithm where the failure criterion is given by a slow convergence rate. If failure occurs, the step is repeated with a lower increment until the minimal value of 1×10^{-10}.

```
Convergence  {  if (!finite(AbsIncr.Disp) || nIter>5) failure("");
```

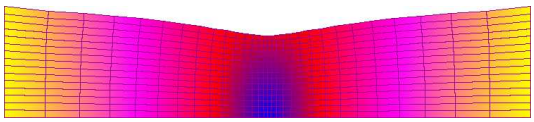

Figure 2.98: The necking of a circular bar under large elongation, shown is one-half of the last converged solution.

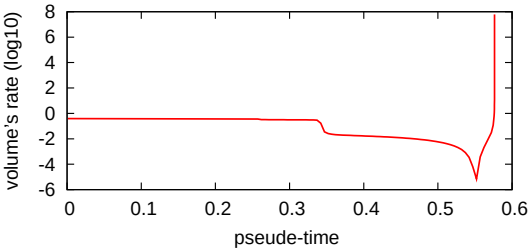

Figure 2.99: The rate of volume's change in logarithmic scale.

```
                else if (AbsIncr.Disp<1E-15) return 1;
                else return 0; write(""); }
Solve Stationary
  Step 0.001,0.256,maxs=0.005, (step*1.2>maxs?maxs:step*1.2)
  Until 1
  Failure { return step<1E-10?0:step/2; }
```

The necking breakdown appears suddenly and without some sort of load control, numerical instabilities prevent any further computations. The instability is due to the rapid change in volume as shown by Fig. 2.99 where the displacement for the last converged solution is shown in Fig. 2.98.

Appendix

To avoid an overloaded bar notation, let consider $\mathbf{F_p} = \mathbf{Id}$ and so $\mathbf{N}_i = \bar{\mathbf{N}}_i$. In order to compute $\partial\mathbf{S}/\partial\mathbf{E}$, we first note that by coaxiality of $\mathbf{S}$ and $\mathbf{C} = \mathbf{Id} + 2\mathbf{E}$, we have $\mathbf{S} = \sum_i S_i \mathbf{N}_i \otimes \mathbf{N}_i$ and $\mathbf{C} = \sum_i \lambda_i^2 \mathbf{N}_i \otimes \mathbf{N}_i$ with $S_i = \tau_i/\lambda_i^2$ and $\sum_i \mathbf{N}_i \otimes \mathbf{N}_i = \mathbf{Id}$. The chain rule gives $\delta\mathbf{S} = (\partial\mathbf{S}/\partial\mathbf{E}) : \delta\mathbf{E} = (\partial\mathbf{S}/\partial\mathbf{E}) : \delta(\mathbf{C}/2)$ so that by expressing the variations $\delta\mathbf{S}$, $\delta\mathbf{C}$ with respect to the base $\mathbf{N}_i \otimes \mathbf{N}_j$, we obtain $\partial\mathbf{S}/\partial\mathbf{E}$ with respect to the base $\mathbf{N}_i \otimes \mathbf{N}_j \otimes \mathbf{N}_k \otimes \mathbf{N}_l$. Let $\mathbf{e}_i$ be an orthonormal constant basis and write $\mathbf{N}_i = \mathbf{Q} \cdot \mathbf{e}_i$ with $\mathbf{Q}$ orthogonal, then $\mathbf{Q} \cdot \mathbf{Q}^{\mathrm{T}} = \mathbf{Id}$ or $\delta\mathbf{Q} \cdot \mathbf{Q}^{\mathrm{T}} + \mathbf{Q} \cdot \delta\mathbf{Q}^{\mathrm{T}} = 0$ and $\delta\mathbf{Q} = -\mathbf{Q} \cdot \delta\mathbf{Q}^{\mathrm{T}} \cdot \mathbf{Q}$. As first, we develop $\delta\mathbf{N}_i = \delta(\mathbf{Q} \cdot \mathbf{e}_i) = \delta\mathbf{Q} \cdot \mathbf{e}_i = -\mathbf{Q} \cdot \delta\mathbf{Q}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{e}_i = \mathbf{\Omega} \cdot \mathbf{N}_i = \sum_j \mathbf{N}_j \otimes \mathbf{N}_j \cdot \mathbf{\Omega} \cdot \mathbf{N}_i = \sum_j \Omega_{ji} \cdot \mathbf{N}_j$ with $\mathbf{\Omega} = -\mathbf{Q} \cdot \delta\mathbf{Q}^{\mathrm{T}}$ and $\Omega_{ji} = \mathbf{N}_j \cdot \mathbf{\Omega} \cdot \mathbf{N}_i$ and note that $\mathbf{\Omega}$ is anti-symmetric since $\mathbf{\Omega}^{\mathrm{T}} = -\delta\mathbf{Q} \cdot \mathbf{Q}^{\mathrm{T}} = \mathbf{Q} \cdot \delta\mathbf{Q}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \mathbf{Q}^{\mathrm{T}} = \mathbf{Q} \cdot \delta\mathbf{Q}^{\mathrm{T}} = -\mathbf{\Omega}$. As second, for some values α_i, we develop $\sum_i \alpha_i \delta(\mathbf{N}_i \otimes \mathbf{N}_i) = \sum_i \alpha_i (\delta\mathbf{N}_i \otimes \mathbf{N}_i + \mathbf{N}_i \otimes \delta\mathbf{N}_i) = \sum_{i,j} \alpha_i \Omega_{ji} (\mathbf{N}_j \otimes \mathbf{N}_i + \mathbf{N}_i \otimes \mathbf{N}_j) = \sum_{i,j} \alpha_j \Omega_{ij} \mathbf{N}_i \otimes \mathbf{N}_j + \alpha_i \Omega_{ji} \mathbf{N}_i \otimes \mathbf{N}_j = \sum_{i,j;i\neq j} \Omega_{ij} (\alpha_j - \alpha_i) \mathbf{N}_i \otimes \mathbf{N}_j$. We therefore have $\delta\mathbf{C} = \delta \sum_i \lambda_i^2 \mathbf{N}_i \otimes \mathbf{N}_i = \sum_i (2\lambda_i \delta\lambda_i) \mathbf{N}_i \otimes \mathbf{N}_i + \sum_{i,j} \Omega_{ij} (\lambda_j^2 - \lambda_i^2) \mathbf{N}_i \otimes \mathbf{N}_j$ and $\delta\mathbf{S} = \delta \sum_i S_i \mathbf{N}_i \otimes \mathbf{N}_i = \sum_{i,j} (\partial_{\lambda_j} S_i \delta\lambda_j) \mathbf{N}_i \otimes \mathbf{N}_i + \sum_{i,j} \Omega_{ij} (S_j - S_i) \mathbf{N}_i \otimes \mathbf{N}_j$. By noting that $\mathbf{N}_j \otimes \mathbf{N}_j : \delta\mathbf{C} = 2\lambda_j \delta\lambda_j$ and for $i \neq j$ we have $(\mathbf{N}_i \otimes \mathbf{N}_j + \mathbf{N}_j \otimes \mathbf{N}_i) : \delta\mathbf{C} = 2\Omega_{ij}(\lambda_j^2 - \lambda_i^2)$, we see that

$$\frac{\partial\mathbf{S}}{\partial\mathbf{E}} = \sum_{i,j} \frac{\partial_{\lambda_j} S_i}{\lambda_j} \mathbf{N}_i \otimes \mathbf{N}_i \otimes \mathbf{N}_j \otimes \mathbf{N}_j + \sum_{i \neq j} \frac{S_j - S_i}{\lambda_j^2 - \lambda_i^2} \mathbf{N}_i \otimes \mathbf{N}_j \otimes (\mathbf{N}_i \otimes \mathbf{N}_j + \mathbf{N}_j \otimes \mathbf{N}_i) \,,$$

resulting in (2.111) by considering $S_i = \tau_i/\lambda_i^2$.

Figure 2.100: Blisters created by deep-drawing of a laminate.

References

[1] T. BELYTSCHKO, W.K. LIU, B. MORAN, *Nonlinear Finite Elements for Continua and Structures*, John Wiley & Sons, 2000.

[2] R. A. HORN, C. R. JOHNSON, *Topics in Matrix Analysis*, Cambridge University Press, 1991.

[3] J.C. SIMO, T.J.R. HUGHES, *Computational Inelasticity*, Springer Verlag, 1998.

[4] J.C. SIMO, *Numerical Analysis and Simulation of Plasticity*, Handbook of Numerical Analysis by P. G. Ciarlet and J. L. Lions, Vol. VI, North-Holland, 1998.

[5] J.C. SIMO, *Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory*, Comput. Methods in Appl. Mech. Engrg., Vol. 99, pp. 61-112, 1992.

2.23 Stamping and mechanical contacts

There are several methods to shape an industrial mass product, one of them is by stamping a laminate made of a material showing a plastic behavior. The method is used to give cars their body shape, for the production of cooking pans, beer cans, blisters enclosing pharmaceutical pills as shown in Fig. 2.100, etc. Modeling the stamping or drawing process is an important task allowing precise statements to be made about the final product as the shape, thickness, stresses, points of rupture or delamination and so on. Important aspects of the modeling are the correct definition of the plastic material laws and of the mechanical contacts between the stamp and the underlying material. Since plastic laws are already discussed in other contributions, we focus here on the modeling of the mechanical contacts and start with a clamped thin laminate modeled by non-linear finite strain solid-shell elements together with a simple isotropic plastic law. To this initial configuration, we then add a moving stamp impressing a shape to the laminate. We assume here the drawing process does not

deform the stamp itself, the mechanical contacts just show up between the laminate and the stamp and the process is so slow that inertial forces can be neglected. These assumptions greatly simplify the solution of the problem, since the stamp's rigid body geometry is a mere input parameter known at any time, there are no self-contacts to be considered and we need to solve a series of stationary solutions parameterized with respect to a pseudo-time parameter determining the movement of the stamp.

The physics of mechanical contacts may be a complex matter and *tribology* is the science describing the interactions at the contact surface. However, for the ideal cases of a friction-less and sticky contact, the underlying physics is completely blended out and replaced by pure kinematic conditions. There are some practical interests in these two ideal situations, since they represent extreme cases with real contacts lying in-between. For both cases, we have first the impenetrability condition of the bodies involved. Then, if this condition occurs, for the friction-less case both bodies at the contact surface glide without friction, for the sticky case they are as glued together. No other physical laws are used here and the simple specification combined with robust solution methods makes these ideal contacts appealing for first quantitative estimations. However, since in general the contact surface is unknown, its determination is part of the problem solution which turns even a linear mechanical problem into a non-linear one and generally it further slows down the convergence rate of non-linear problems. It is important to note that the usage of sticky contacts always results in a path dependent problem, not however so for friction-less contacts. In the following, we will just consider the more common case of friction-less contacts which are also more complex to implement numerically.

At the base of any contact algorithms, there is a fast algorithm checking the impenetrability condition with the rigid-body, for every point $\mathbf{x}$ on the computational boundary where a mechanical contact has been enabled. If this condition is violated, the contact algorithm will then force the point $\mathbf{x}$ back on the rigid-body surface. To accomplish this task, one first computes the closest point projection on the rigid-body surface $\mathbf{x}_0$ formally defined so that the vector $\mathbf{x} - \mathbf{x}_0$ is parallel to the surface normal $\mathbf{n}$ at $\mathbf{x}_0$. In general this projection is not unique, however, it will be for points close enough to the surface, with an upper bound for the distance depending on the surface curvature. During the solution process and at each linear step, the friction-less contact algorithm will then force any invalid movement $\mathbf{x}$ to stay on the plane having normal $\mathbf{n}$ and passing through the point $\mathbf{x}_0$ which is represented by the linear kinematic constraint

$$\mathbf{n} \cdot (\mathbf{x} - \mathbf{x}_0) = 0\,. \tag{2.120}$$

Since the projection values $\mathbf{n}$ and $\mathbf{x}_0$ will generally change at each iteration step, in order to compute correct derivatives, we also require from the closest point projection, the surface characterization at $\mathbf{x}_0$ in form of two orthonormal tangent vectors $\mathbf{t}_\alpha$, $|\mathbf{t}_\alpha| = 1$, $\alpha = 1, 2$ with $\mathbf{n} = \mathbf{t}_1 \times \mathbf{t}_2$ and the two main curvature coefficients k_α with respect to the tangent vectors.

An exact algebraic or inexact penalty method are two possible choices to enforce these linear kinematic constraints. The algebraic method is more complex to implement than the penalty one, however, the penalty method is ill-conditioned in its solution is far less precise than the one of the exact approach. Both problems can be elegantly studied within a continuum formulation, without yet considering any numerical dis-

cretization of the governing equations, see [1, 2]. However, for simplicity and succinctness, we will assume the governing equations have already been discretized and we are looking for methods applying the contact conditions directly on the discretized form. Many discretizations of the governing equations of elasticity use pure displacement Lagrange FEs, where at some selected sampling points $\mathbf{X}_i$, $i = 1 \dots N$, three degree-of-freedoms $\mathbf{u}_i = \mathbf{u}(\mathbf{X}_i) \in \mathbb{R}^3$ determine the displacement $\mathbf{u}(\mathbf{X})$ at $\mathbf{X}_i$. In turn, the set of sampling points is determined by the finite element mesh and the chosen degree of the approximation for $\mathbf{u}(\mathbf{X})$. Other FEs may add other dofs to determine the displacement, as for example rotational dofs for shell elements. Here the numerical implementation of contact algorithms will be more complex and not so general, so that it will be not considered here.

Penalty method

One advantage of the penalty approach to contact mechanics is that it does not necessarily require any modifications of the underlying finite element implementation and it is solely based on the specification of contact forces in form of tractions. By comparing the known stamp's position with the actual computed displacements, as soon the impenetrability condition of both bodies is violated, the penalty method adds stiff forces $\mathbf{T}$ in the direction of the rigid-body normal $\mathbf{n}$, in order to restore the impenetrability condition. Actually this latter condition can only be satisfied approximately and the degree of approximation is controlled by a penalty parameter. The dilemma is that the better the impenetrability approximation is, the worse is the ill-conditioness of the system to be solved and computing solutions will be slower and harder. For a generic boundary point $\mathbf{x} = \mathbf{X} + \mathbf{u}$ with rigid-body projection $\mathbf{x}_0$, let δ be the actual value of penetration defined by $\delta = (\mathbf{x}_0 - \mathbf{x}) \cdot \mathbf{n}$. The penalty method defines the tractions as $\mathbf{T} = h(\delta)\mathbf{n}$ with $h(\delta)$ a stiff function having the properties $h(\delta \leq 0) = 0$ and $h(\delta \geq 0) \geq 0$. A good working example is $h(\delta) = a\delta^b$ with $a \gg 1$ and $b = 2$.

In order not to further slow down the non-linear solution process, we need to correctly specify the traction's derivatives $\partial_{\mathbf{u}}\mathbf{T}$ with respect to the displacements $\mathbf{u}$ in order to assemble a global exact Jacobi matrix and to guarantee a quadratic convergence rate of the Newton iteration close to the solution. For this task, we need a second order approximation of the surface at $\mathbf{x}_0$ given by

$$\mathbf{s}(p_1, p_2) = \mathbf{x}_0 + \mathbf{t}_\alpha p_\alpha + \mathbf{n} k_\alpha p_\alpha^2/2 + \mathcal{O}(|(p_1, p_2|^3)\,,$$

with free parameters p_α and an implied sum over the indices α. The closest point projection of any point $\mathbf{x}$ close to $\mathbf{x}_0$ is approximately given by $\bar{\mathbf{x}}_0(p_1, p_2) = \mathbf{s}(p_1, p_2)$ with $p_\alpha = (\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{t}_\alpha (1 + \delta k_\alpha)^{-1} + \mathcal{O}(|\mathbf{x} - \mathbf{x}_0|^2)$. By expressing the surface normal as

$$\begin{aligned}
\bar{\mathbf{n}}(p_1, p_2) &= \frac{\partial_{p_1}\mathbf{s} \times \partial_{p_2}\mathbf{s}}{|\partial_{p_1}\mathbf{s} \times \partial_{p_2}\mathbf{s}|} = \frac{(\mathbf{t}_1 + p_1 k_1 \mathbf{n}) \times (\mathbf{t}_2 + p_2 k_2 \mathbf{n})}{|\partial_{p_1}\mathbf{s} \times \partial_{p_2}\mathbf{s}|} \\
&= \frac{\mathbf{t}_1 \times \mathbf{t}_2 + p_2 k_2 \mathbf{t}_1 \times \mathbf{n} + p_1 k_1 \mathbf{n} \times \mathbf{t}_2 + p_1 p_2 k_1 k_2 \mathbf{n} \times \mathbf{n}}{|\partial_{p_1}\mathbf{s} \times \partial_{p_2}\mathbf{s}|} = \frac{\mathbf{n} - p_\alpha k_\alpha \mathbf{t}_\alpha}{\sqrt{1 + p_\alpha^2 k_\alpha^2}}\,,
\end{aligned}$$

we have $\partial_{p_\alpha}\bar{\mathbf{n}}|_{p_\alpha = 0} = -k_\alpha \mathbf{t}_\alpha$ and by the chain rule $\partial_{\mathbf{x}}\bar{\mathbf{n}} = (\partial_{p_\alpha}\bar{\mathbf{n}})(\partial_{\mathbf{x}} p_\alpha) = -k_\alpha/(1 + \delta k_\alpha)\mathbf{t}_\alpha \otimes \mathbf{t}_\alpha$. In a similar way, we obtain $\partial_{\mathbf{x}}\bar{\mathbf{x}}_0 = (\partial_{p_\alpha}\bar{\mathbf{x}}_0)(\partial_{\mathbf{x}} p_\alpha) = -k_\alpha \mathbf{t}_\alpha \otimes \mathbf{t}_\alpha$. It follows

that $\bar{\mathbf{n}} \otimes \partial_{\mathbf{x}}(\bar{\mathbf{x}}_0, \bar{\mathbf{n}}) = 0$ and therefore

$$\begin{aligned} \frac{\partial \mathbf{T}}{\partial \mathbf{u}} = \frac{\partial \bar{\mathbf{n}} h(\delta)}{\partial \mathbf{x}} &= \frac{\partial \bar{\mathbf{n}}}{\partial \mathbf{x}} h(\delta) + \frac{\partial h}{\partial \delta} \mathbf{n} \otimes \frac{\partial \delta}{\partial \mathbf{x}} = \frac{\partial \bar{\mathbf{n}}}{\partial \mathbf{x}} h(\delta) + \frac{\partial h}{\partial \delta} \mathbf{n} \otimes \frac{\partial (\bar{\mathbf{x}}_0 - \mathbf{x}) \cdot \mathbf{n}}{\partial \mathbf{x}} \\ &= -\frac{\partial h}{\partial \delta} \mathbf{n} \otimes \mathbf{n} - h(\delta) \frac{k_\alpha}{1 + \delta k_\alpha} \mathbf{t}_\alpha \otimes \mathbf{t}_\alpha \,. \end{aligned} \tag{2.121}$$

This short presentation of the penalty method uses a continuum approach so that the tractions $\mathbf{T}$ actually define Neumann BCs for the elasticity equations. The underlying FE software will then integrate these tractions over the contact boundary and add the contributions to the dof-residual equations and map the traction derivatives to dof-derivatives. This latter task will be more complex, if the FE discretization does not solely use displacement dofs. For FEs with just nodal displacement dofs, the above relations can be directly used to amend the nodal values at $\mathbf{X}_i$. Here the tractions $\mathbf{T}_i = \mathbf{T}(\mathbf{x}_i)$ with $\mathbf{x}_i = \mathbf{X}_i + \mathbf{u}_i$ will be added to the right-hand side vector and the derivatives $\partial \mathbf{T}_i / \partial \mathbf{u}_i$ to the matrix of the linearized system to be solved. The two methods yield slight different numerical results, which however can be compensated by a little change of the penalty parameter.

To explain the ill-conditioness of the penalty approach, for simplicity let us consider the direct nodal approach for the boundary node $\mathbf{X}_i$ and associated three degree-of-freedom $\mathbf{u}_i$ with zero curvatures $k_\alpha = 0$. If $\delta > 0$, in practice one adds the 3×3 matrix $-\mathbf{n} \otimes \mathbf{n}(\partial_\delta h)$ as diagonal subblock at the row and columns associated with $\mathbf{u}_i$ and $h(\delta)\mathbf{n}$ on the right-hand side. If the normal $\mathbf{n}$ has just a single non-zero component e.g. $\mathbf{n} = \mathbf{e}_x$, by just considering large values we have $u_{ix}(\partial_\delta h) = h(\delta)$. This linear system with a single unknown is regular and its solution $u_{ix} = -h(\delta)/\partial_\delta h$ dominates the global solution. Here the problem is not ill-conditioned at all since the penalty method is equivalent to setting a priori Dirichlet BCs for u_{ix} so that the penalty parameter can be chosen very very large resulting in an exact Dirichlet BCs. However, when $\mathbf{n}$ has at least two non-zero components, by considering large values in the global system, we have the singular linear system $-\mathbf{n} \otimes \mathbf{n}(\partial_\delta h)\mathbf{u}_i = h(\delta)\mathbf{n}$. It cannot be solved singularly, but has to be solved together with the global system. In this case, the problem is ill-conditioned and the value of the penalty cannot be chosen to large, otherwise the global solution is dominated by a singular system with infinitely many solutions.

Algebraic method

We shortly present an algebraic method to contact mechanics directly working on the nodal displacement dofs $\mathbf{u}_i$ of the discretized equations. When the nodal movement $\mathbf{x}_i = \mathbf{X}_i + \mathbf{u}_i$ violates the impenetrability condition, the constraint equation (2.120) is added to the system and of the three degree-of-freedoms $\mathbf{u}_i$, one picks a slave and two masters. The slave is generally the one with the absolute largest coefficient of the normal $\mathbf{n}$, we assume here $|n_z| \geq |n_x|, |n_y|$ thus u_{iz} is the slave degree-of-freedom and we have the additional equation $u_{iz} = n_z^{-1}(\mathbf{n} \cdot \mathbf{x}_0 - \mathbf{n} \cdot \mathbf{X}_i - n_x u_{ix} - n_y u_{iy})$. Clearly adding any constraint equation to an already uniquely solvable linear system, make the system unsolvable. However, by considering that we are actually looking for solutions which are saddle points of a functional, adding constraint equations is perfectly legal and the method of the Lagrange multiplier tells us how to solve the new constrained

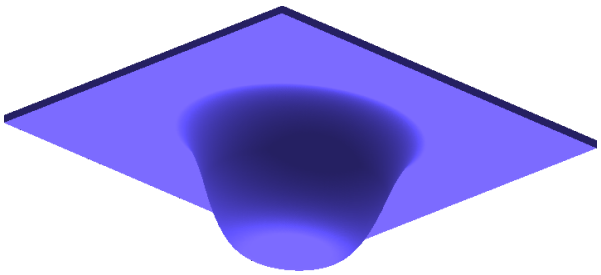

Figure 2.101: Rotational symmetric shape of the upper and lower stamps.

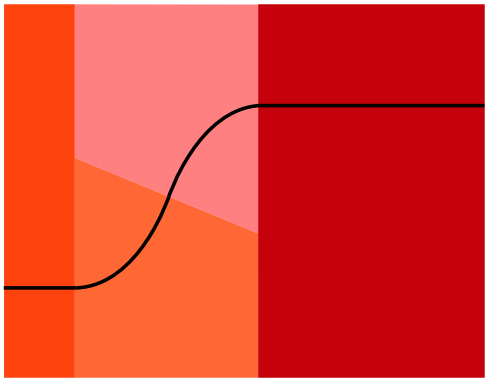

Figure 2.102: Radial shape consisting of two lines and two parabolae and the four regions for point projection onto the basic shapes.

system. This can be done by amending the native system with simple algebraic operations performed on the rows and columns of the assembled matrix. After solving the amended system, the linear constraint equations are fulfilled exactly. The solution of this exact formulation is robust, however, it suffers from the fact that once applied, we cannot say for sure if we have a contact point or not, i.e. if by releasing the glide constraint, the boundary point will violate or not the impenetrability condition. A brute force method will combinatorially turn-on and off all glide constraints upon reaching consistency with the impenetrability condition, an unfeasible approach. Much better is to check if the rigid-body sets the contact point under tension or compression. Just for compressive points we continue to apply the constraints, whereas for tensile ones we release them. At the end of the non-linear solution, all contact points must be compressive points and all other boundary points must not violate the impenetrability condition. The question arises how to know if a contact point is under compression or not, since at the discretization level the numerical stress is not at our disposal. However, we have the residual values $\mathbf{R}_i$ associated with the degree-of-freedoms $\mathbf{u}_i$. After a solution has been computed and thus after applying the contact constraints, the residual values are always zero $\mathbf{R}_i = 0$. Therefore before amending the discretized equations for the glide constraints, the residual values represent the sum of all forces acting on the boundary point without considering the contacts. These residual values are concentrated force resulting from integrating over the domain the governing equations with various a priori unknown weighting factors. The absolute value is therefore of no meaning to us, not so its direction specifying the direction of the contact force acted by the rigid body.

Closest Point Projection

A general approach to rigid-body contacts consist in approximating the contact surface by low order multivariate polynomials which are then used to evaluate the closest point projection. If the normal should be a continuous function, the global approximation must be C^1 and for continuous curvatures C^2-smoothness is required. This general approach is the simplest to apply from the user point of view, but a good ap-

proximation is memory intensive and requires well designed search algorithms. In *SESES*, a C^2-interpolation on a 2D tensor grid of 3D sampling points is available, but even with a very large number of points, the curvature values may not be very close to the exact values of an interpolated analytical function. This is due to the interpolating character which must be C^2 and exact at the sampling points. If the rigid-body surface is simple, then an analytical projection together with some well designed *if-else* statements can be a valid alternative providing exact values of the projection and surface characterization. In this contribution, we will present both approaches for a surface of revolution consisting of two piecewise linear segments and two parabolae with global C^1-continuity as displayed in Figs. 2.101-2.102.

The most difficult part of the analytical projection algorithm consists in computing the orthogonal projection to a generic 2D parabola $\mathbf{F} = (x, ax^2)$ with $x \in \mathbb{R}$, $a > 0$ and a generic point $\mathbf{P} = (P_x, P_y)$. By symmetry, we can assume $P_x \geq 0$ and since for $P_x = 0$ we always have the projection $(0, 0)$, we can further assume $P_x > 0$. The orthogonal projection is stated here as an orthogonality condition between the segment $\mathbf{P} - \mathbf{F}$ and the tangent $(d/dx)\mathbf{F}$ as $(\mathbf{P} - \mathbf{F}) \cdot (d/dx)\mathbf{F} = 0$ and resulting in the cubic equation $f(x) = 2a^2x^3 + (1 - 2aP_y)x - Px = 0$. Since $f(0) < 0$ and $f(+\infty) = +\infty$, we always have a solution for $x > 0$. For $(1 - 2aP_y) \geq 0$, we have $(d/dx)f(x) \geq 0$ so that f is monotone and just a single solution exists. For $(1 - 2aP_y) < 0$, we have $(d/dx)f(0) < 0$ so that the other two solutions, if they exist, they are necessarily negatives. Therefore the projection for $P_x > 0$ is unique. To find the single positive solution of $f(x) = 0$, we may use a simple Newton-Raphson iteration, but we have to avoid getting close to a possible local minimum $(d/dx)f(x) = 0$ for $x > 0$. Initialization with $x_0 = \sqrt{P_y/a}$ for $P_y > aP_x^2$ and $x_0 = P_x$ otherwise, always avoids this critical region and results in a robust monotone decreasing sequences $x_{n+1} = x_n - f(x_n)/(d/dx)f(x_n)$ with $f(x_{n+1}) > 0$, $x_{n+1} < x_n$ and $x_{n+1} > 0$.

For this analytical example, we can define a unique projection for any point $\mathbf{x} = (x, y, z)$. By computing the radial value $r = (x^2 + y^2)^{0.5}$ and depending on which region displayed in Fig. 2.102 the point (r, z) lies into, the point (r, z) is orthogonally projected onto the radial shape, either trivially for the line segments or with the algorithm presented above for the two parabola segments. The projection is always unique but is C^1 just for points close enough to the contact surface. Once the radial projection has been computed, we additionally need to provide the orthonormal triads of tangent vectors and normal, as well as the main curvatures. By parametrizing the surface as $\mathbf{s}(r, \phi) = (r\cos(\phi), r\sin(\phi), g(r))$ with $g(r)$ the radial shape of Fig. 2.102, the two unnormalized tangent vectors are given by $\mathbf{t}_1 = \partial_r\mathbf{s} = (\cos(\phi), \sin(\phi), \partial_r g)$ and $\mathbf{t}_2 = \partial_\phi\mathbf{s} = (-r\sin(\phi), r\cos(\phi), 0)$. The normal is given by $\mathbf{n} = (\mathbf{t}_1 \times \mathbf{t}_2)/|\mathbf{t}_1 \times \mathbf{t}_2| = (-\cos(\phi)\partial_r g, -\sin(\phi)\partial_r g, 1)/\sqrt{1 + \partial_r g^2}$. The tangent vectors are orthogonal and since $(\partial_r\partial_\phi\mathbf{s}) \cdot \mathbf{n} = 0$, the main curvatures with respect to the normalized tangent vectors are given by $k_1 = \partial_r^2\mathbf{s} \cdot \mathbf{n}/|\mathbf{t}_1|^2 = \partial_r^2 g/(1 + (\partial_r g)^2)^{1.5}$ and $k_2 = \partial_\phi^2\mathbf{s} \cdot \mathbf{n}/|\mathbf{t}_2|^2 = (\partial_r g)r^{-1}/(1 + (\partial_r g)^2)^{0.5}$.

Numerical model

The numerical example of a blister forming process by deep-drawing of a elasto-plastic laminate can be found at `example/Laminate.s3d`. Since the geometry of

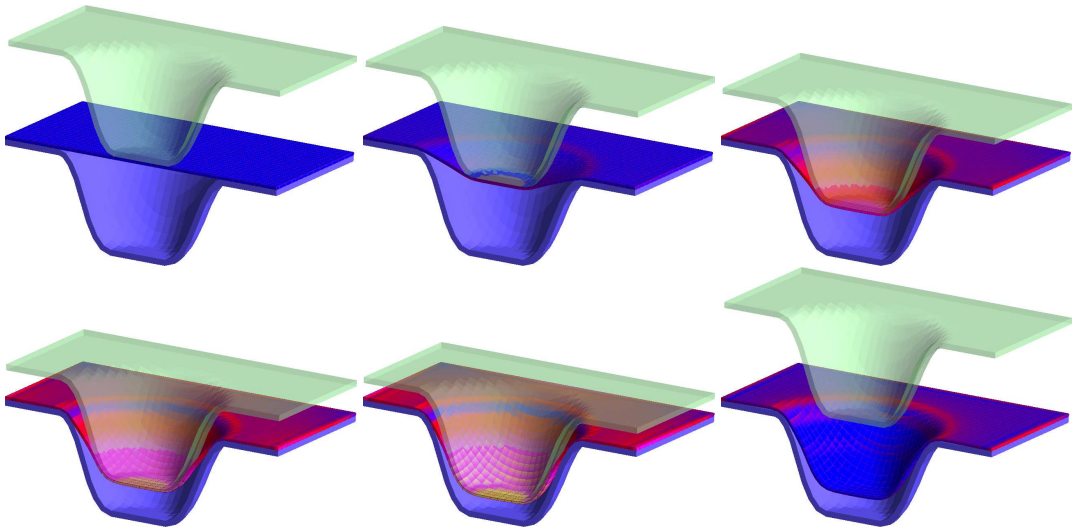

Figure 2.103: Initial and intermediary deformations with final state showing the spring-back effect. Just half of the structure is shown.

the model is rotational symmetric and the material laws used in this example are isotropic, a 2D rotational symmetric solution may do the job. However, we present a 3D problem formulation, since in practice anisotropic plastic laws are needed thus requiring 3D computations even for an initial symmetry of revolution. In addition realistic computations will also require contact with frictions, instead of friction-less ones as done in this example.

An initial rectangular laminate clamped at its border is all what is needed as initial geometry, if the closest point projection is computed by the user. However, for a visual aid in the graphical representation, it is also useful to define the geometry of the rigid-body stamps by dummy MEs where no numerical equation is ever solved, see Fig. (2.103). An additional advantage is that one can right use the geometry of these dummy MEs to compute the closest point projection numerically. This projection is computed by the routine `RigidIntersection`, however, before calling it, one has to declare the rigid-bodies with the statement

```
Misc RigidBody(StampDown Down, StampUp Up; Smooth)
```

The values `StampDown-StampUp` are the block ME numbers for the two stamps and the `Down-Up` specifiers determine the contact surface of the hexahedral ME block. The `Smooth` option activates cubic interpolation so that normal, tangents and curvatures are continuous functions. In this example both the analytical and numerical approach are used and compared. For the former, a series of input routines `ParabolaProj`, `Profile`, `StampCoord`, `StampProj` is defined evaluating the projection as described previously. Typo errors in these user routines are not so easily discovered and within comments some additional testing code that has been used is provided. The selection of one of the two approaches is simply determined by the setting of a global variable.

At the input's beginning, we have defined routines to be used by the penalty and algebraic contact approach. Since they are pretty generic, they can be placed once inside some input files to be included. For the penalty method, the routine `PenaltyContact` is used to define the Neumann BCs and takes as input the data structure `CONTACT`

specifying the closest point projection. This data needs to be set by the user or by calling the routine `RigidIntersection`. Other ramp functions than the quadratic one do not in general perform as well, so that the only free parameter for the penalty method is the rigidity factor defined by the constant `RIGIDITY`. For the algebraic method, we use the built-in BC routine `Glide` which applies the constraint equation (2.120) at any displacement dofs found at the BC surface. At the present time, the routine just implements constant constraints so that if the values of the normal $\mathbf{n}$ and $\mathbf{n} \cdot \mathbf{x}_0$ depends upon $\mathbf{x}$, quadratic convergence of the Newton iteration cannot be guaranteed. Actually the routine applies the constraint just if one passes a non-zero value of the normal and this decision is taken by the routine `AlgebraicContact`, which similarly to `PenaltyContact` takes as input the data structure `CONTACT`. We enable the constraint whenever the condition $|\delta_i| \leq 1{\times}10^{-14}$ **AND** $\mathbf{n} \cdot \mathbf{R}_i \geq 0$ **OR** $\delta_i \leq -1{\times}10^{-14}$ is fulfilled with δ_i the distance of the boundary point $\mathbf{x}_i$ to the contact surface and $\mathbf{R}_i$ the assembled residual forces at $\mathbf{x}_i$. If the boundary point was clamped on the contact surface on the previous iteration, then we have $|\delta_i| \leq 1{\times}10^{-14}$ and we release the lock just if at $\mathbf{x}_i$, we have a tensile residual force $\mathbf{n} \cdot \mathbf{R}_i \geq 0$. Further, we always enable the constraint, if the boundary point violates the impenetrability condition $\delta_i \leq -1{\times}10^{-14}$.

We use here the same isotropic plastic law of the example *Necking of a circular bar* which we refer for further explanations. For the discretization, we use non-linear solid-shell elements enabled with the equation `ElasticShellSingleK` and the model `MechNonLin`. Since these mixed FEs use internal variables and the Green-Lagrange strain used to evaluate material laws is not directly computed from the deformation gradient, the quasi-positivity condition > -1 of the Green-Lagrange strain eigenvalues is not given. The principal log-strain values cannot always be computed and `NaN` values may be returned. In general this just happen by taking too large steps, so that the condition of invalid log-strain values is checked within our material routine `J2FlowPrincipal`

```
if (isnan(STRAIN[0])||isnan(STRAIN[1])||isnan(STRAIN[2]))
  failure("LOGSTRAIN");
```

For `Nan` log-strain values, a failure message is fired and the solution step will be repeated with a smaller step value.

Friction-less contacts are path-independent and therefore do not destroy the symmetry of the system. In particular, the derivative of the traction (2.121) is symmetric so that the global system is symmetric as well. This condition is not detected automatically and therefore we use the declaration `LinearSolver Symmetric` to force a symmetric linear solver. We then solve a series of stationary problems by changing the position of one stamp and at each step a Newton iteration is performed up to convergence of the displacement's increments, see Fig. (2.103). The step length is chosen adaptively based on the convergence rates of the Newton iteration. At the end of the inward movement, the stamp is retracted to its original position in order to observe the spring-back effect. We note that for this particular geometry and depending on the yield value for the plastic flow, the structure may buckle and our simple solution method may not be able to pass this singular point.

For the penalty method, we have quadratic convergence, both with a closest-point projection computed analytically or by interpolation. In the latter case, a slight degra-

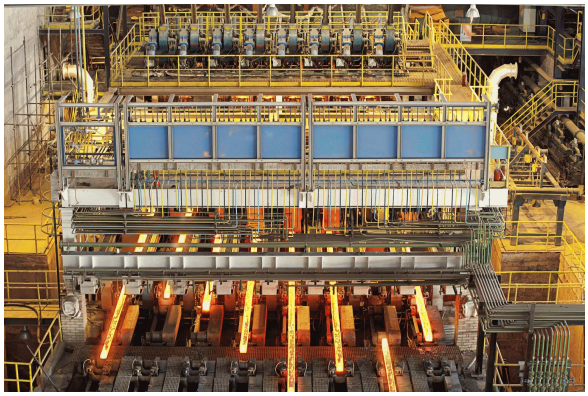

Figure 2.104: Steel casting machine made by *SMS Concast* Zürich.

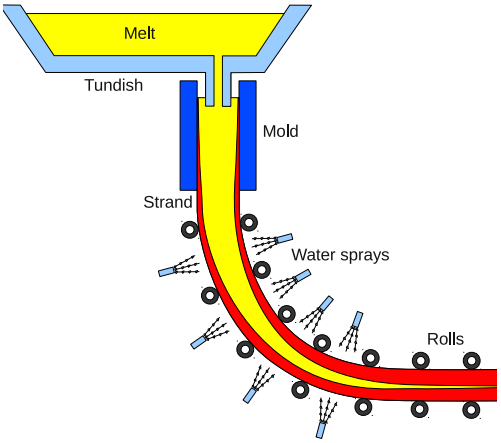

Figure 2.105: Schematic view of steel casting.

dation of the convergence is noticed. The algebraic contact approach does not work for this example. The problem lies in the sudden change of the constraint equations at each iteration, so that the global residual equations are not continuous. The Newton iteration gets stuck in a dead-lock cycle of opening and closing constraints and the convergence rate stales. The simple algorithm described above for enabling and disabling the constraints, which generally works well for a simple half-space rigid-body, needs to be amended. However, it is not clear if a robust method valid for any geometric form can be found.

References

[1] T. A. LAURSEN, *Computational Contact and impact Mechanics*, Springer, 2002.

[2] T. A. WRIGGERS, *Computational Contact Mechanics*, Springer, 2007.

2.24 Continuous casting of steel

The discover and the first production of steel started long time ago in Anatolia around the 15th century BC and todays we are approaching a worldwide production of a billion of tons p.a. Around 1930, Siegfired Junghans developed a continuous casting process for melted brass. Its application in the steel industry was advocated and pioneered by Irving Rossi who in 1954 founded in Zürich the *Concast* company, now part of the *SMS* group and a major supplier of steel casting machines. Since then, the continuous casting of steel in forms of billets, blooms and slabs has become the standard production process. From a hot furnace, the melt is transferred into a *tundish* reservoir allowing to continuously feed the casting machine. The melt is then drained into a water-cooled copper mold, where the hot metal directly in contact with the mold solidifies into a thin shell. Here, the mold may oscillates or may be lubricated in order to prevent sticking. The thin solidified shell acts as a containment for the melt inside the

now called *strand*. After exiting the mold, the strand is further cooled by water sprays and its movement is redirected and supported by rolls. The cooling and solidification process continues until the whole strand section solidifies and the strand can be cut into fixed lengths. The steel is still very ductile and can be further formed into its final shape by milling. Not only the chemical composition of the melt but the whole casting process has a strong impact on the quality of the steel. As first one has to avoid the breakout of the melt due to a broken shell at the exit of the mold, then a key point is the minimization of the residual stresses in the strand due to thermal strains and mechanical contacts by the rolls which may lead to cracks and so to a poor steel quality. Direct quality measurements in the production line are inherently difficult due to the high temperatures and the moving strand. Therefore there is a major interest for a numerical simulation of the casting process which should give qualitative answers on temperatures and stresses of the strand.

We present here a numerical model for the continuous casting process from the exit of the mold up to the cut of the strand possessing the major ingredients of a full featured simulation. In particular, we use simplified material laws, simple support for mechanical contacts and a 2D modeling domain, but otherwise the model is pretty complete. As first we note that within or after the mold the melt in the strand can be a turbulent fluid due to the presence of magnetic stirrers. However, if we abstain ourself from such complex situations and if the strand does not get in mechanical resonance, the continuous casting of steel is a quasi-stationary or steady process with the shape, temperature and stresses of the strand at a given spatial point being constant in time and only during the start-up phase we have unsteady conditions. This initial phase may be modeled as well, but it is not done here. As second we note that we have to consider a coupled thermo-mechanical problem, however, due to the high temperatures and the large amount of convected thermal energy, the heat production due to mechanical inelastic deformations of the strand has little impact on the temperature and can be neglected. Therefore the coupling is one-directional, one has first to determine the temperature by solving the governing equation of energy transport and then with known temperature profiles, one solves the governing equation of elasticity to compute displacements and stresses. A correct computation of the temperature is important for the subsequent mechanical problem. The most challenging part here are correct values for the thermal conductivity and capacity, but otherwise the numerical solution of the convected thermal transport is robust and standard. In the following, we will not perform this step and instead we assume an unrealistic constant temperature of the strand.

Let us start with a Eulerian or spatial view of the casting problem as customary when dealing with steady flow conditions. Momemtum conservation in spatial coordinates $\mathbf{x} \in \Omega$ with Ω the domain representing the strand yields the governing equations

$$\rho D_t \mathbf{v} = \nabla.\mathbf{s} + \mathbf{f}\,, \tag{2.122}$$

with ρ the mass density, $\mathbf{v}$ the velocity, $\mathbf{s}$ the stress, $\mathbf{f}$ the body force and $D_t(\bullet) = \partial_t(\bullet) + \nabla(\bullet)\cdot\mathbf{v}$ the material derivative. Steel at high temperatures is a visco-plastic material with an elastic and an inelastic response determined by a yield function $f(\mathbf{s})$, where the elastic region is characterized by values $f(\mathbf{s}) < 0$. Differently from plastic materials where we must have $f(\mathbf{s}) \leq 0$ and values $f(\mathbf{s}) > 0$ are forbidden, for visco-plastic material values $f(\mathbf{s}) > 0$ are permitted and the material will relax to the state

$f(\mathbf{s}) = 0$ in a characteristic time. A common visco-plastic law is the following J_2-flow model

$$\begin{aligned} \mathbf{s} &= k\mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{Id} + 2\mu(\mathrm{dev}(\boldsymbol{\varepsilon}) - \boldsymbol{\varepsilon}_p)\,, \\ f &= |\mathrm{dev}(\mathbf{s})| - \sqrt{2/3}\sigma_y\,, \\ \mathbf{n} &= \mathrm{dev}(\mathbf{s})/|\mathrm{dev}(\mathbf{s})|\,, \\ D_t\boldsymbol{\varepsilon}_p + (\nabla.\mathbf{v})\boldsymbol{\varepsilon}_p &= \Pi_p = \begin{cases} 0 & \text{if } f < 0\,, \\ f\mathbf{n}/\tau & \text{if } f \geq 0\,, \end{cases} \end{aligned} \tag{2.123}$$

with $\boldsymbol{\varepsilon}_p$ the plastic strain, k the bulk modulus, μ the shear modulus, τ a relaxation time, $\mathrm{tr}(\boldsymbol{\varepsilon}_p) = 0$, $\mathrm{dev}(\boldsymbol{\varepsilon}_p) = \boldsymbol{\varepsilon}_p$, $\mathrm{tr}(\boldsymbol{\varepsilon}) = \varepsilon_{ii}$ the trace of a second order tensor and $\mathrm{dev}(\boldsymbol{\varepsilon}) = \boldsymbol{\varepsilon} - \mathbf{Id}\mathrm{tr}(\boldsymbol{\varepsilon})/3$ the deviator operator. In the following we assume a constant yield stress σ_y and so we do not consider any kinds of hardening. Evaluating the material derivative D_t together with the assumption $\nabla.\mathbf{v} = 0$ and the steady condition $\partial(\bullet)/\partial t = 0$, yields the following system of PDEs

$$\begin{cases} \rho\nabla\mathbf{v}\cdot\mathbf{v} = \nabla\cdot\mathbf{s} + \mathbf{f}\,, \\ \nabla\boldsymbol{\varepsilon}_p\cdot\mathbf{v} = \Pi_p\,. \end{cases} \tag{2.124}$$

The first system determines the stress $\mathbf{s}$ and since the casting velocity is generally small, the term $\rho\nabla\mathbf{v}\cdot\mathbf{v}$ can be neglected. The second system determines the convected plastic strain $\boldsymbol{\varepsilon}_p$, it is a system of first order hyperbolic PDEs coupled just by the right-hand side Π_p. The steady state condition implies that for $\Pi_p \neq 0$ we must have $\mathbf{v} \neq 0$, otherwise there is a contradiction.

Our model formulation for continuous casting is almost complete, except that we still have a spatial formulation and we do not yet know where to solve the equations (2.124). In other words, we do not know the exact shape of the spatial domain Ω and therefore an Eulerian or spatial solution approach typical of fluid dynamics cannot be used. The matter is also a little bit more involved than for classical elasticity, where with the help of the Piola identity the spatial formulation is pushed-back to obtain a Lagrangian or material one with respect to a referential domain Ω_0. Due to the continuous flow, a steady solution of a material formulation with respect to an initial, stress-free referential system with particles at rest does not exist, since it would imply infinite deformations. Therefore, to be able computing steady solutions of (2.124), we need an Arbitrary Lagrange-Euler or ALE approach. There is nothing special about an ALE approach and everything boil down in the definition of a referential stress-free system with particles not necessarily at rest. This is the key point and the matter depends on the problem at hand. Therefore, we assume a stress-free strand at the exit of the mold and its free flow in space, with all mechanical loads turned off, will determine our initial reference system Ω_0. Due to the continuous flow, the shape of this initial domain Ω_0 is generally straight but there are molds specially devised to give the strand an initial constant curvature.

Knowing the stress-free referential system Ω_0, we can push-back the spatial formulation and start computing solutions by applying all mechanical loads present in the casting process to obtain the deformation $\phi : \Omega_0 \to \Omega = \phi(\Omega_0)$ or equivalently the shape of the strand. Depending on the layout of the casting machine and especially for strands with initial curvature, the referential domain Ω_0 and the spatial one Ω may not be close and large deformations ϕ will result. This fact complicates the numerical analysis a lot, since a geometric non-linear pushed-back formulation

of the governing equation (2.124) needs to be used. However, with the assumption of small strains, the formulation can be simplified by considering that the several rolls of the casting machine set sharp bounds on the possible shape of the strand. In other words, from the geometrical layout of the rolls, we can make a good guess Ω_1 on the final shape of the strand and we can assume $\Omega_1 \approx \Omega$. The deformation $\phi = \phi_0 + \mathbf{u}$ now consist of a known contribution ϕ_0 from the stress-free referential system to this initial-guess system $\phi_0 : \Omega_0 \to \Omega_1 = \phi_0(\Omega_0)$ and an unknown but small displacement $\mathbf{u} : \Omega_1 \to \Omega = (\mathbf{x}_1 + \mathbf{u})(\Omega_1)$. Therefore, we are left to solve the same equations of (2.124) now in the initial-guess system Ω_1 and with an additional initial strain $\boldsymbol{\varepsilon}_0 = (\nabla\phi_0 + (\nabla\phi_0)^T)/2 - \mathbf{Id}$ to be added to $\boldsymbol{\varepsilon}$ and representing the deformation $\phi_0 : \Omega_0 \to \Omega_1 = \phi_0(\Omega_0)$.

A possible solution approach for (2.124) consists in keeping the time derivative $(\partial/\partial t)\boldsymbol{\varepsilon}_p$ and by computing a series of mechanical solutions with the plastic strain $\boldsymbol{\varepsilon}_p$ directly evaluated by the method of characteristics until a steady solution emerges. The drawbacks are that convergence to the steady solution is slow and evaluating the characteristics works well for analytical functions, not however for discretized ones, expecially discontinuous ones. Here one needs some sort of artificial diffusion in upwind-downwind direction and a careful analysis for a stable evaluation since one has also to integrate the plastic strain rates. Another solution approach advocated here consists in defining the plastic strain $\boldsymbol{\varepsilon}_p$ as dof-fields and solve the steady transport equations $\nabla\boldsymbol{\varepsilon}_p \cdot \mathbf{v} = \Pi_p$ with stabilized finite elements. To find a solution of the non-linear system of equations (2.124), we can have a fully coupled algorithm solving all equations together or a uncoupled algorithm solving iteratively for the displacement $\mathbf{u}$ and the plastic strain $\boldsymbol{\varepsilon}_p$. A priori, one cannot say which method will be faster, so we give all partial derivatives associated with material laws required for a coupled solution with optimal convergence rate. In our example, the body force $\mathbf{f}$ is a dead-load and so the following partial derivatives $\partial_{(\boldsymbol{\varepsilon},\boldsymbol{\varepsilon}_p)}\Pi_p$ are required. *SESES* will then use internally these derivatives to solve the residual equations $(\vec{R}^{\mathbf{u}}, \vec{R}^{\boldsymbol{\varepsilon}_p}) = 0$ associated with the degrees-of-freedoms $(\vec{u}, \vec{\boldsymbol{\varepsilon}}_p)$ using a Newton-Raphson algorithm with exact derivatives $\partial(\vec{R}^{\mathbf{u}}, \vec{R}^{\boldsymbol{\varepsilon}_p})/\partial(\vec{u}, \vec{\boldsymbol{\varepsilon}}_p)$.

In order to keep the same notation, we give here directional derivatives $D(\bullet)[\mathbf{a}]$ with respect to a vector $\mathbf{a}$. As first we note that since $\mathrm{dev}(k\mathrm{tr}((\boldsymbol{\varepsilon})\mathbf{Id}) = 0$, we have $\partial_{\boldsymbol{\varepsilon}} f\mathbf{n} = -\partial_{\boldsymbol{\varepsilon}_p} f\mathbf{n}$ and, then by considering the property $\mathbf{n} \cdot \mathrm{dev}(\mathbf{a}) = \mathbf{n} \cdot \mathbf{a}$, after some algebra we obtain

$$\begin{aligned}
D_{\boldsymbol{\varepsilon}_p}\mathbf{s}[\mathbf{a}] &= -2\mu\,\mathrm{dev}(\mathbf{a})\,,\\
D_{\boldsymbol{\varepsilon}}\mathbf{s}[\mathbf{a}] &= k\,\mathrm{tr}(\mathbf{a})\mathbf{Id} + 2\mu\,\mathrm{dev}(\mathbf{a})\,,\\
-D_{\boldsymbol{\varepsilon}_p} f[\mathbf{a}] = D_{\boldsymbol{\varepsilon}} f[\mathbf{a}] &= 2\mu\,\mathbf{n}\cdot\mathrm{dev}(\mathbf{a}) = 2\mu\,\mathbf{n}\cdot\mathbf{a}\,,\\
-D_{\boldsymbol{\varepsilon}_p}\mathbf{n}[\mathbf{a}] = D_{\boldsymbol{\varepsilon}}\mathbf{n}[\mathbf{a}] &= 2\mu\tfrac{\mathrm{dev}(\mathbf{a}) - \mathbf{n}\cdot(\mathbf{n}\cdot\mathbf{a})}{|\mathrm{dev}(\mathbf{s})|}\,,\\
-D_{\boldsymbol{\varepsilon}_p} f\mathbf{n}[\mathbf{a}] = D_{\boldsymbol{\varepsilon}} f\mathbf{n}[\mathbf{a}] &= \mathbf{n}D_{\boldsymbol{\varepsilon}} f[\mathbf{a}] + f D_{\boldsymbol{\varepsilon}}\mathbf{n}[\mathbf{a}]\,.
\end{aligned}$$

Numerical model

The numerical example of a 2D the steel casting machine can be found at `example/Casting.s2d`. The mold gives the strand an initial curvature and rolls are placed so that the strand makes a turn of 90 degrees then straighten and proceeds horizontally, see Fig. 2.106. This shape defines the computational domain Ω_1. Without me-

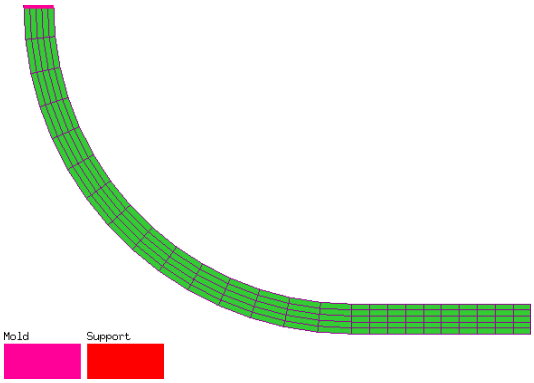

Figure 2.106: The computational domain with mechanical contacts.

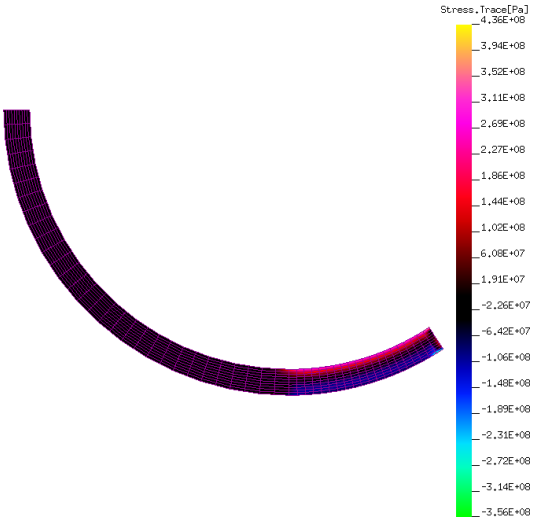

Figure 2.107: The stress free referential domain obtained by turning-off all mechanical loads. The solution shows some residual stress.

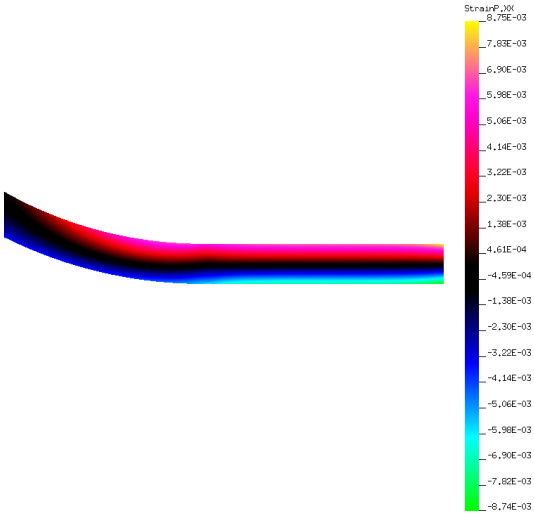

Figure 2.108: Plastic strain component $\varepsilon_{p,xx}$.

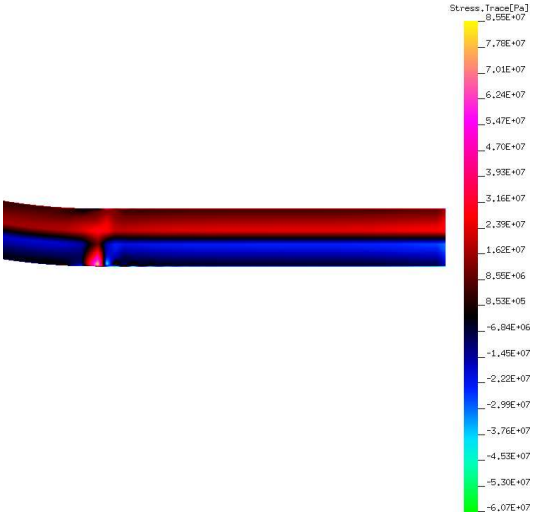

Figure 2.109: Trace of the stress $\mathrm{tr}(\mathbf{s})$

chanical loads, the strand shape will be circular and to consider the bending forces straightening the strand, on the horizontal part of the domain, we define the strain $\varepsilon_{xx} = (y - y_0)/R$ with R the radius of the strand and y_0 the middle vertical position. By just fixing the strand at the exit of the mold, this initial strain with a non-linear geometric formulation will result in a circular strand of constant curvature R, see Fig. 2.107. There are no predefined dof-fields for pure convected transport by a velocity $\mathbf{v}$, therefore the dof-fields `StrainP` representing the plastic strain tensor ε_p must be declared with the statement

```
ConvectDef        StrainP[T2Z]
```

Afterwards we define the material laws as discussed previously and BCs with a clamped strand at the exit of the mould and as crude approximation also in the presence of the rolls, which for simplicity are defined all along the straight zone. Here a slight improvement of the J_2-flow model $\Pi_p = f(|\mathrm{dev}(\mathbf{s})|)\mathbf{n}$ is used by considering a rate expressed in the form $\Pi_p = f(|\mathrm{dev}(\mathbf{s})|, |\varepsilon_p|)\mathbf{n}$. This new form allows e.g. for isotropic hardening and the required derivatives $D_{\mathbf{s}}f$, $D_{\varepsilon_p}f$ readily follow from the two scalar ones $D_{|\mathrm{dev}(\mathbf{s})|}f$ and $D_{|\varepsilon_p|}f$ which are returned together with the rate $f(|\mathrm{dev}(\mathbf{s})|)$ by the routine `FlowRule`.

For the solution algorithm and the dof-fields $\mathbf{u}$ and ε_p, we can choose between a fully coupled or uncoupled algorithm, however, a coupled algorithm is generally required due to the *stiffness* of the system. This is mainly due to the fact that for a velocity $\mathbf{v} \to 0$ and if $\Pi_p \neq 0$, then $\nabla\varepsilon_p \to \infty$ and the problem does not have a solution. In practice, smaller velocities will result in larger condition numbers of the linear system to be solved and for a fixed length floating point format, there is a velocity lower bound below the one no solutions can be computed. To improve the condition of the system, one should check that the linear mechanical solution without plastic rates should be smooth without any singularities in the stress. If the stress is singular, the plastic rates Π_p are unbounded with strong gradients making the problem numerically ill-conditioned and very hard or even unsolvable.

Fig. 2.106 shows for some arbitrary material parameters, the convected plastic strain component $\varepsilon_{p,xx}$. This solution is obtained by first computing the linear mechanical solution without plastic rates and then with a coupled algorithm by suddenly turning-on the plastic rates. The plastic strain is always a continuous field due to the artificial diffusion introduced when solving the convected transport equations with $H^1(\Omega)$ conformal finite elements. Fig. 2.109 shows the trace values of the stress. At the beginning of the straight zone, we have singular stress values which are then convected and smoothed down-wind. For a more realistic model, we have implemented the plastic rate as presented by [1]. This model is solved by setting the variable `MODEL_SIMPLE` to zero and it is apparent that the solution is now much more hard to compute. The plastic rates are slowly turned-on by the homotopic parameter `HOMOT` and the solution is found at `HOMOT` $\to \infty$. The step-length of `HOMOT` is chosen adaptively but due to the large and increasing system condition, the homotopic process stops long before. This is an example showing that singular stress values generate very stiff and ill-conditioned problems.

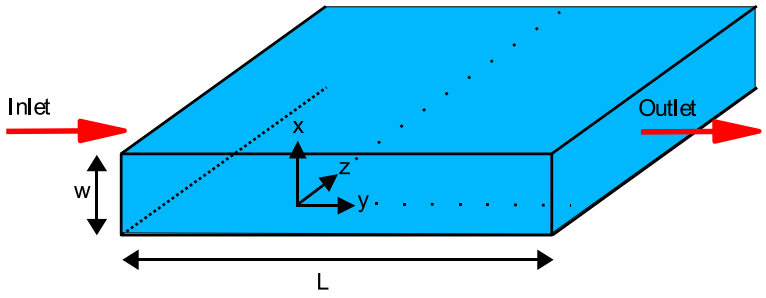

Figure 2.110: A rectangular flow channel.

References

[1] S. KORIC, B. G. THOMAS, *Efficient thermo-mechanical model for solidification processes*, Int. J. Numer. Meth. Engng., Vol. 66, pp. 1955-1989, 2006.

2.25 Hagen-Poiseuille Model of Viscous Laminar Flow

To introduce the basics of fluid flow, we present the Hagen-Poiseuille flow in a straight channel as displayed in Fig. 2.110. This laminar viscous flow problem is one of the few known analytical solutions of the incompressible Navier-Stokes equations. The flow is characterized by a constant mass density ρ, a constant viscosity μ, an inlet/outlet at constant pressure $P_{\text{in}}/P_{\text{out}}$ with zero tangential flow $v_y = 0$ and no slip conditions $\mathbf{v} = 0$ at the other boundaries. The 2D solution for an infinitely deep channel is given by the flow velocity

$$v = \frac{(P_{\text{in}} - P_{\text{out}})}{8\mu L}\left(W^2 - 4x^2\right), \quad x \in [-\tfrac{W}{2}, \tfrac{W}{2}], \tag{2.125}$$

with L the channel length and W the channel width. The total mass flow at the inlet/outlet correspond to the mass flow integral over the channel area

$$M = \int_{-\frac{W}{2}}^{\frac{W}{2}} \rho v \mathrm{d}x = \frac{\rho(P_{\text{in}} - P_{\text{out}})W^3}{12\mu L}. \tag{2.126}$$

The *SESES* input file for this Hagen-Poiseuille problem can be found at `example/PoiseuilleFlow.s2d`. The initial section starts with the definition of the channel dimensions, viscosity, density and pressure drop as user variables in order to parameterize the computation. These values correspond to an air flow channel of dimension $0.25 \times 0.05\,\mathrm{m}^2$

```
Define
  length  = 0.25    (* m *)
  width   = 0.05    (* m *)
  visco   = 0.0008  (* Pa*s *)
  density = 1000    (* kg/m**3 *)
  dp      = 0.01    (* Pa *)
  massflow= density*dp*width*width*width/(12*visco*length)
```

The simple rectangular channel geometry is defined with the next statements defining a macro element mesh of 25×10 elements

```
QMEI Start -width/2 nx=5 width/nx
QMEJ ny=4 length/ny
```

The physical properties as well as the equations to be solved are defined with the next statements. Here, we have to solve for the incompressible Navier-Stokes equations and to define the viscosity as well as the mass density. The mass density ρ does not really enter the incompressible Navier-Stokes but it is used together with BCs and therefore must be defined. The parameter `PressStab` is the penalty parameter used to solve the mass conservation law. Since this equation is not an elliptic, it has been stabilized with a second order dissipative term in the pressure proportional to `PressStab` and its value should not be chosen too small or too large

```
MaterialSpec Air
  Equation  CompressibleFlow Enable
  Parameter Density.Val  density kg/m**3
  Parameter Viscosity    visco  Pa*s
  Parameter FlowStab     zero
  Parameter PressPenalty 1e-6 s
```

The defined material `Air` is then mapped on the whole domain with the statement

```
Material Air 1
```

The next statements define the BCs as stated at the beginning. Since the solution only depends on the pressure drop, we define a value of $0\,\mathrm{Pa}$ on the outlet and the constant pressure drop `dp` at the inlet

```
BC Inlet  0 0  IType nx
  Dirichlet Pressure    dp Pa
  Dirichlet Velocity.X  0 m/s

BC Outlet  0 ny IType nx
  Dirichlet Pressure    0 Pa
  Dirichlet Velocity.X  0 m/s

BC NoSlip  0 0 JType ny   nx 0 JType ny
  Dirichlet Velocity    0,0 m/s
```

The next statements are part of the command section. Here the solution of the Navier-Stokes equations is defined with the dof-fields for the pressure p and velocity $\mathbf{v}$ computed all together in a single block. The Navier-Stokes equations are non-linear, however, the Hagen-Poiseuille we are solving is a linear problem and we just need to perform a single block iteration

```
BlockStruct Block Pressure Velocity Convergence 1
Solve Stationary
```

Since for the Hagen-Poiseuille problem the analytical solution is known, it is of interest to check the numerical error affecting the solution with the statement

```
Write "Error for Velocity %e %e\n"
  maxvalue(Nodal;Velocity-(0,dp*(width*width-4*x*x)/(8*visco*length)))
```

By starting the computation, the textual output will look similar to

```
> STATIONARY SOLUTION
-> SOLVING AT time= 0.0000e+00 STEP 0 (0.000000e+00)
        FIELD: Pressure AbsResidNorm: 1.13e-10
        FIELD: VelocityX AbsResidNorm: 0.00e+00
        FIELD: VelocityY AbsResidNorm: 4.71e-06
<BC_data>
  <BC Name=Inlet Type=Basic>
    <Field= Pressure Flux= MassFlux TypeBC= Dirichlet > 0.500000002 kg/s
    <Field= VelocityX Flux= ForceX TypeBC= Dirichlet > -2.81167505e-16 N
  <BC Name=Outlet Type=Basic>
    <Field= Pressure Flux= MassFlux TypeBC= Dirichlet > -0.500000002 kg/s
    <Field= VelocityX Flux= ForceX TypeBC= Dirichlet > 4.96040282e-15 N
  <BC Name=NoSlip Type=Basic>
    <Field= VelocityX Flux= ForceX TypeBC= Dirichlet > -5.22497695e-15 N
    <Field= VelocityY Flux= ForceY TypeBC= Dirichlet > -0.0005 N
</BC_data>
File Data written
The maximal error for VelocityX is 4.726968e-14
The maximal error for VelocityY is 1.950801e-13
```

From this output, we can see that the mass flow of $0.5\,\mathrm{kg/s}$ at the inlet/outlet differs from (2.126) by ca. 4%, the viscous flow exerts a total force of $0.0005\,\mathrm{N}$ on the channel and the accuracy of the velocity is close to machine accuracy.

The precision of the velocity field seems surprising, if one considers the very limited number of used elements and the error affecting the total mass flow. This type of superconvergence is occasional and a little modification of the problem will remove it. For the Hagen-Poiseuille problem and from the relation (2.126), we see that the total mass flow and the pressure drop in the channel are equivalent for the problem specification. In practice, however, the pressure at the inlet/outlet is unknown, so that one prefer to specify the total mass flow. We can do this by a little amendment at the inlet BC, where instead of setting the pressure, we define the total mass flow

```
BC Inlet IType 0  nx 0
  Pressure  Floating  massflow kg/s
  VelocityX Dirichlet 0 m/s
```

Differently from the previous case, the maximal error in the velocity is now about 4% and by changing the number of elements, you will see a linear convergence behavior. The dramatical drop in accuracy is due to the fact that finite elements used to discretize the Navier-Stokes equations cannot reproduce the quadratic solution (2.125), the relation (2.126) does not hold exactly and the superconvergence behavior is lost.

2.26 Blasius Plate Flow At Zero Incidence

Due to the non-linearity of the stationary incompressible Navier-Stokes equations,

$$\nabla.\mathbf{v} = 0\,, \quad \rho(\mathbf{v}\cdot\nabla)\mathbf{v} - \mu\nabla^2\mathbf{v} + \nabla p = \mathbf{f}\,, \tag{2.127}$$

with $\mathbf{v}$ the velocity, p the pressure, $\mathbf{f}$ the body forces, ρ, μ the constant density and viscosity, very few analytical solutions are known, see [1]. For small viscosities or large Reynolds numbers, it is well known that solutions are characterized by thin layers around boundaries. These layers can be studied by solving the simplified Prandtl boundary layer equations obtained by removing small terms after a dimensional analysis at large Reynolds numbers and solutions to these equations can be found e.g. in

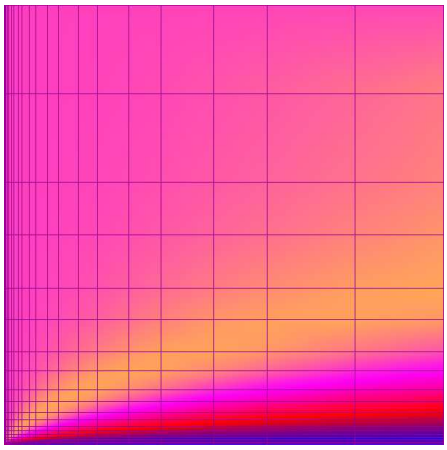

Figure 2.111: Velocity distribution for $\nu = 0.01$ and computational mesh locally adapted to resolve the boundary layer.

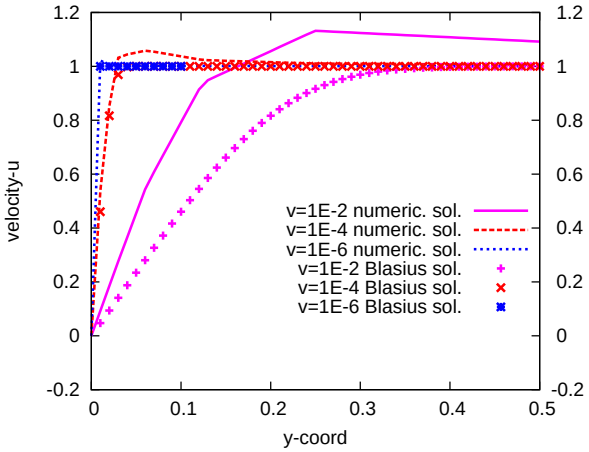

Figure 2.112: Comparison of numerical and Blasius solutions at $x = 0.5$ for the kinematic viscosities $\nu = 0.01, 10^{-4}, 10^{-6}$.

[1]. One of them is the 2D Blasius boundary layer flow over a flat plate at zero angle of incidence with respect to a uniform stream of velocity U_∞ at infinity. Although just an approximation of the Navier-Stokes equations (2.127), this solution can be used to test and calibrate Navier-Stokes solvers, which is what we are going to present in this example.

Without going into the details of the Prandtl boundary layer theory, the Blasius solution for a plate located along the x-axis can be obtained as follow. For the Blasius problem we have $\mathbf{f} = 0$ and by assuming a priori $\partial_x p = 0$, just the incompressibility condition $\partial_x u + \partial_y v = 0$ and balance of momentum in x-direction $u\partial_x u + v\partial_y u = \nu(\partial_x^2 u + \partial_y^2 u)$ need to be solved together uin order to the obtain velocity $\mathbf{v} = (u, v)$ with $\nu = \mu/\rho$ the kinematic viscosity. Within the thin boundary layer, we have $\partial_x^2 u \ll \partial_y^2 u$ and therefore the boundary layer approximation consists in neglecting the term $\partial_x^2 u$. The continuity equation is automatically satisfied by working with a stream function $\psi(x, y)$ where we have $u = \partial_y \psi$ and $v = -\partial_x \psi$ and hence we are left to solve the single equation

$$\partial_y \psi \partial_{xy} \psi - \partial_x \psi \partial_y^2 \psi = \nu \partial_y^3 \psi \,.$$

Blasius found a separation of variables by introducing the similarity variable $\eta = y\sqrt{U_\infty/(\nu x)}$ and working with the Ansatz $\psi(x, \eta) = \sqrt{U_\infty \nu x} f(\eta)$. By considering the new independent variables (x, η) substitution yields $u = U_\infty f'$, $v = \sqrt{U_\infty \nu/(4x)}(\eta f' - f)$ and

$$\frac{U_\infty^2}{2x}(f'' f + 2f''') = 0 \,,$$

so that we are left to solve the Blasius equation $f'' f + 2f''' = 0$ with the boundary conditions $f(0) = f'(0) = 0$ and $f'(\infty) = 1$. The first two conditions represent the no-slip condition $\mathbf{u} = 0$ on the plate and the last one the stream velocity $\mathbf{u} = (U_\infty, 0)$ at infinity. Due to its non-linearity, this boundary value problem needs to be solved numerically, as done for example with the program Maple and the input below by considering $\eta = 150$ as infinity.

```
INFITY:=150: ODE:=2*diff(f(e),e$3)+f(e)*diff(f(e),e$2)=0:
```

```
SOL:=dsolve({ODE,f(0)=0,D(f)(0)=0,D(f)(INFITY)=1},type=numeric,maxmesh=1024,
             approxsoln=[f(e)=e]):
plots[odeplot](SOL,[e,diff(f(e),e)],e=0..5);
```

Computations of the Blasius flow are done on a unit square domain for various kinematic viscosities and the input file can be found at `example/BlasiusFlow.s2d`. The boundary conditions for this incompressible solution are $\mathbf{v} = (1, 0)$ at $x = 0$ and $y = 1$ and $\mathbf{v} = (0, 0)$ at $y = 0$, on the point $(1, 0)$ we fix the pressure $p = 0$ to avoid floating values. The boundary layer at $x = 0$ must be resolved by the grid in order to obtain the correct solutions therefore as shown in Fig. 2.111, we use a geometrically spaced mesh in the y-direction. By the setting of the boundary conditions, the velocity is discontinuous at $(0, 0)$ and the pressure has a singularity there. Therefore to resolve this singularity we also use a geometrically spaced mesh along the x-direction. The numerical and the Blasius solutions for the x-velocity component on the line at $x = 0.5$ are compared in Fig. 2.112 for the kinematic viscosities $\nu = 0.01, 10^{-4}, 10^{-6}$. For $\nu = 0.01$, the numerical solution has a clear overshoot caused by the mass conservation and the setting of the BCs, however, the thickness of the boundary layer is for both solutions the same. The mesh is chosen so that for $\nu = 10^{-4}$ it can still resolve the boundary layer, not so for $\nu = 10^{-6}$ which is then given by the thickness of the first element. The drag on the plate of length l is given by integrating the shear stress component $\boldsymbol{\tau}_{xy}$

$$\text{Drag} = b \int_0^l \mu \partial_y u|_{y=0} \mathrm{d}x = b\mu U_\infty f''(0) \int_0^l \partial_y \eta \mathrm{d}x = 2b\rho U_\infty f''(0) \sqrt{U_\infty \mu l}\,,$$

with b the thickness of the plate and $f''(0) \approx 0.33205733$. The numerical values agree well with the analytical ones for thin boundary layers resolved by the computational mesh.

The incompressible Navier-Stokes defined by the statement `Equation IncompressibleFlow Enable` uses residual-based stabilization to circumvent the various instabilities associated with a Galerkin discretization of the equations (2.127). In practice, to the residuals equations obtained by the classical unstable Galerkin discretization, one adds the integrals of the strong residuals weighted by special test functions. Since the strong residuals tends to zero, this stabilization method is consistent, however, the strong residuals can be freely scaled and therefore we always have free parameters to fit. A priori error analysis of the stabilized discretization schemes give us the order of magnitude of these parameters, however, a fine tuning is generally required for optimal numerical results. Here, the singularity at $(0, 0)$ of the Blasius flow may be used to calibrate the incompressible Navier-Stokes solver. By refining the mesh around this singularity, the calibration is performed by defining the stabilization parameters through the material parameter `FlowStab` in order for a coarse mesh to have approximately the same solution and by minimizing over- or under-shooting for both the pressure and the velocity. The dependency of the stabilization parameters from the velocity $\mathbf{v}$ and viscosity μ is according to the theory [2], however, one needs to calibrate the constant scaling factors. The proposed values are also used by the default setting of `FlowStab`.

References

[1] H.SCHLICHTING, K. GERSTEN, *Boundary Layer Theory*, Springer-Verlag, 8^{th} Ed., 2000.

[2] M. BRAACK, E. BURMAN, V. JOHN, G. LUBE, *Stabilized finite element methods for the generalized Oseen problem*, Comput. Methods Appl. Mech. Engrg, No.196, pp. 853-866, 2007.

2.27 Microfluidic Mixing in a Straight Channel

Microfluidics – a part of the microsystems domain – is about flows of liquids and gases, single or multiphase, through microdevices fabricated by MEMS technology. In microfluidics, the ability to mix two or more fluids thoroughly and in a reasonable amount of time is fundamental to the creation of fully integrated on-chip microelectromechanical fluid processing systems. Effective mixing requires that the fluids be manipulated or directed so that the contact area between the fluids is increased and the distance over which diffusion must act is decreased to the point that complete mixing is achieved in an acceptable amount of time. In macroscopic devices this is generally done by using turbulence, three-dimensional flow structures or mechanical actuators. As MEMS devices are fabricated in a planar lithographic environment, design constraints mitigate against mechanical actuators or three-dimensional flow structures. Instead, innovative static mixing concepts are pursued to increase the mixing efficiency. The mixing efficiency of the classical T-junction mixer [2, 3], for example, can be increased using the multiple splitting of streams [1] or through the use of slanted wells [4].

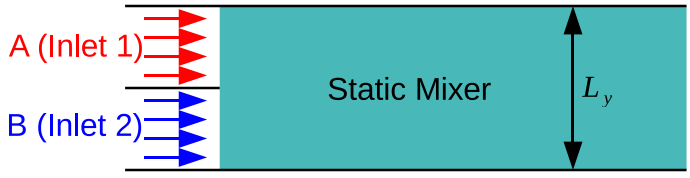

Figure 2.113: Mixing of a fluid of species A with a fluid of species B in a straight channel.

This tutorial discusses the ability of *SESES* to accurately predict mixing phenomena in microfluidic devices by considering the simplest possible design of a static mixer, i. e. a straight channel with two inlets as shown in Fig. 2.113 where a fluid of species A is mixed with a fluid of species B. Note that the transport of matter in the flow direction is mainly by convection, whereas diffusion is the transport mechanism in the cross-direction. We first present the stationary diffusion-convection equations for a binary mixture and solve them analytically for our geometry. Thereafter, a *SESES* model solving these equations by the Finite-Element (FE) method will be set up and the numerical results will be compared with the analytical solution.

Diffusion-Convection Equation

Consider first the mass-balance for some species α

$$\frac{\partial \rho_\alpha}{\partial t} + \nabla.(\rho_\alpha \, \mathbf{W}_\alpha) = \Pi_\alpha \,, \tag{2.128}$$

with $\rho_\alpha = m_\alpha / V$ the partial mass-density, W_i^α the local velocity and Π_α the production (or consumption) rate of α, respectively [5]. Since in general, the transport of α happens by convection as well as by diffusion, we split W_i^α into its diffusive and convective contributions. To specify the latter, consider the local mass-average velocity

$$\mathbf{W} = \sum_{\alpha=1}^{\nu} \frac{\rho_\alpha}{\rho} \mathbf{W}_\alpha \,, \tag{2.129}$$

with the total mixture density given by $\rho = \sum_{\alpha=1}^{\nu} \rho_\alpha$ and ν the number of species. $\mathbf{W}_\alpha$ can now be expressed relative to $\mathbf{W}$ as $\mathbf{W}_\alpha = \mathbf{W} + \mathbf{J}_\alpha / \rho_\alpha$ with $\mathbf{J}_\alpha$ the diffusion-flux of α and its insertion into (2.128) leads to

$$\frac{\partial \rho_\alpha}{\partial t} + \nabla.(\rho_\alpha \, \mathbf{W}) + \nabla.\mathbf{J}_\alpha = \Pi_\alpha \,. \tag{2.130}$$

From irreversible thermodynamics, we know that $\mathbf{J}_\alpha$ can be written as

$$\mathbf{J}_\alpha = \sum_{\beta=1}^{\nu-1} \frac{B_{\alpha\beta}}{T} \nabla.(\mu_\beta - \mu_\alpha) \,, \tag{2.131}$$

with μ_α and μ_β the chemical potentials of species α and β and $B_{\alpha\beta}$ a phenomenological coefficient [5]. Consider now the particular case of an isothermal, incompressible fluid mixture for which $(T, \rho) = \text{const}$. Then, the chemical potentials of all species are only functions of concentrations $c_\alpha = \rho_\alpha / \rho$ and $\mathbf{J}_\alpha$ simplifies to

$$\mathbf{J}_\alpha = -\sum_{\beta=1}^{\nu-1} \rho \, D_{\alpha\beta} \nabla.c_\beta \,, \tag{2.132}$$

with $D_{\alpha\beta}$ the diffusion coefficient of α in β [5]. For a binary mixture of species A and B, we have $\nu = 2$ and the diffusive flux $\mathbf{J}_A$ follows from (2.132) as

$$\mathbf{J}_A = -\rho \, D_{AB} \nabla.c_A = -D_{AB} \nabla.\rho_A \,. \tag{2.133}$$

Since $\sum_{\alpha=1}^{\nu} \mathbf{J}_\alpha = 0$, the diffusive flux of A is balanced by a reverse flux of B, i. e. $\mathbf{J}_B = -\mathbf{J}_A$. Taking into account (2.133), the mass-balance of A follows from (2.130) as

$$\frac{\partial \rho_A}{\partial t} + \nabla.\rho_A \, \mathbf{W} - D_{AB} \nabla^2 \rho_A = \Pi_\alpha \,. \tag{2.134}$$

For mass transport under steady-state conditions and in the absence of any chemical reactions, eq. (2.134) further simplifies to

$$\nabla.\rho_A \, \mathbf{W} = D_{AB} \nabla^2 \rho_A \,. \tag{2.135}$$

By noting that for $\rho = \text{const}$ the overall mass-balance reduces to $\nabla.\mathbf{W} = 0$, eq. (2.135) takes its final form as $\mathbf{W} \cdot \nabla \rho_A = D_{AB} \nabla^2 \rho_A$ and division by ρ leads to the diffusion-convection equation in terms of concentrations

$$\mathbf{W} \cdot \nabla c_A = D_{AB} \nabla^2 c_A \,. \tag{2.136}$$

Diffusion-Convection Equation for a Plug Flow

Consider a two-dimensional system in the Cartesian coordinates (x, y) and assume a straight fluid flow in x-direction with a uniform velocity $V_x = \text{const}$, i. e. a plug flow. In this case, eq. (2.136) simplifies to

$$V_x \frac{\partial c_A}{\partial x} = D_{AB} \left(\frac{\partial^2 c_A}{\partial x^2} + \frac{\partial^2 c_A}{\partial y^2} \right) . \tag{2.137}$$

Now suppose the transport of A in x-direction is mainly by convection, the term $\partial^2 c_A / \partial x^2$ can then be neglected with the result that

$$V_x \frac{\partial c_A}{\partial x} = D_{AB} \frac{\partial^2 c_A}{\partial y^2} . \tag{2.138}$$

For convenience, we introduce the dimensionless length-variable $\hat{y} \equiv y / L_y$ in the y-direction together with the abbreviation

$$K \equiv \frac{D_{AB}}{V_x \, L_y^2} , \tag{2.139}$$

so that (2.138) can be written as

$$\frac{\partial c_A}{\partial x} = K \frac{\partial^2 c_A}{\partial \hat{y}^2} . \tag{2.140}$$

This equation subjected to the following BCs

$$c_A(x = 0, \hat{y}) = c_A^0(\hat{y}), \quad \left. \frac{\partial c_A}{\partial \hat{y}} \right|_{\hat{y}=0} = 0 , \quad \left. \frac{\partial c_A}{\partial \hat{y}} \right|_{\hat{y}=1} = 0 , \tag{2.141}$$

represents a homogeneous boundary-value problem and can be solved by the method of separation of variables [6]. Inserting the Ansatz $c_A(x, \hat{y}) = \Theta(x) \, \Psi(\hat{y})$ into (2.140) leads to

$$\frac{1}{K} \frac{\Theta_x}{\Theta} = \frac{\Psi_{\hat{y}\hat{y}}}{\Psi} = -m^2 , \tag{2.142}$$

with m a constant and a particular solution for $c_A(x, \hat{y})$ is given by

$$c_A(x, \hat{y}) = e^{-K m^2 x} \left[C_1 \, \sin(m \, \hat{y}) + C_2 \, \cos(m \, \hat{y}) \right] . \tag{2.143}$$

It follows from the BC (2.141b) that $C_1 \, e^{-K m^2 x} = 0$, which leads to $C_1 = 0$. Evaluation of (2.141c) gives $e^{-K m^2 x} \, C_2 \, \sin(m) = 0$, which implies that $m = 0, \pi, 2\pi, ... = i \, \pi$ with $i = 0 ... \infty$. The general solution for $c_A(x, \hat{y})$ then follows as the superposition of the particular solutions,

$$c_A(x, \hat{y}) = C_{2,0} + \sum_{i=1}^{\infty} e^{-K (i \, \pi)^2 x} C_{2,i} \, \cos(i \, \pi \, \hat{y}) . \tag{2.144}$$

The coefficients $C_{2,i}$ with $i = 0 ... \infty$ yet have to be adjusted to the BC (2.141a), i. e. they have to be determined from

$$c_A^0(\hat{y}) = C_{2,0} + \sum_{i=1}^{\infty} C_{2,i} \, \cos(i \, \pi \, \hat{y}) . \tag{2.145}$$

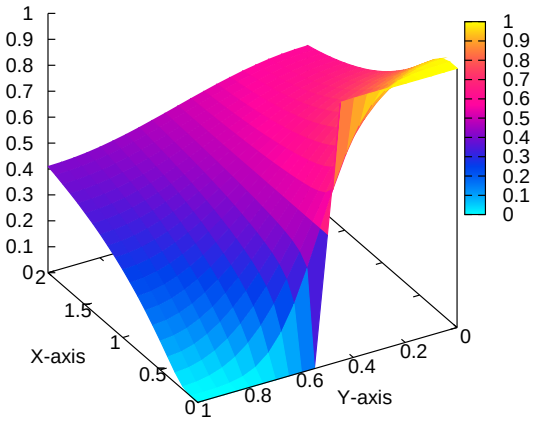

Figure 2.114: Predictions of (2.148) for the concentration field along the straight channel shown in Fig. 2.113 for $K = 0.1\,\mathrm{m}^{-1}$.

Suppose $c_A^0(\hat{y})$ is given by the square pulse

$$c_A^0(\hat{y}) = \begin{cases} 1, & 0 \le \hat{y} \le 1/2\,, \\ 0, & 1/2 < \hat{y} \le 1\,. \end{cases} \tag{2.146}$$

A Fourier-series analysis of $c_A^0(\hat{y})$ leads to

$$C_{2,0} = \frac{1}{2} \text{ and } C_{2,i} = \frac{2}{i\,\pi} \sin\left(\frac{i\,\pi}{2}\right)\,, \tag{2.147}$$

and insertion of (2.147a) and (2.147b) into (2.144) gives the final result

$$c_A(x,\hat{y}) = \frac{1}{2} + \sum_{i=1}^{\infty} e^{-K\,(i\,\pi)^2\,x}\,\frac{2}{i\,\pi}\sin\left(\frac{i\,\pi}{2}\right)\,\cos(i\,\pi\,\hat{y})\,. \tag{2.148}$$

Fig. 2.114 shows predictions of (2.148) for the concentrations along our straight channel. For the parameter value $K = 0.1\,\mathrm{m}^{-1}$ chosen here, the initial square pulse is transformed into the uniform concentration of $c_A = 0.5$ during a distance of about $x = 2.0\,\mathrm{m}$. Since the local cross-flow diffusion-flux is proportional to the local concentration-gradient, the homogenization process is much stronger at distances closer to the inlets.

Model Specification

The input file `example/DiffusionDuct.s2d` starts with the statement

```
Species  A B
```

defining the names for the considered species. Next, the geometry is defined

```
QMEI  nx=40 (lx=2.0)/nx
QMEJ  ny=21 (ly=1.0)/ny
```

followed by the statements

```
  K         = 0.1                  (* K = D/(Vx*ly^2) in 1/m *)
  density   = 1e3                  (* kg/m**3 *)
  diffCoeff = 1.0e-9               (* m**2/s *)
  velocity  = diffCoeff/(K*ly**2)  (* m/s *)
  mmolA     = 1E-3                 (* kg/m**3 *)
  mmolB     = 1E-3                 (* kg/m**3 *)
MaterialSpec  m_Fluid
  Equation   CompressibleFlow TransportA TransportB Enable
  Parameter  Density.Val density kg/m**3
  Parameter  Mmol.A      mmolA kg/mol
  Parameter  Mmol.B      mmolB kg/mol
  Parameter  Diff      { @DiagDiff2(diffCoeff*mmolA*mmolB,A,B) } SIunit
  Model      TransConvDefined  Enable
  Parameter  Rate D_A D_B D_AGrad D_BGrad @CONVECTION2(Rate,A,B,Velocity*density)  SIunit
Material m_Fluid   1
```

to define and map the material `m_Fluid`. On its domain, the Navier-Stokes equations are set up together with the species mass-balances. However, as will be discussed later, when the velocity profile and the average pressure are given, only the species mass-balances need to be solved. As last step, the inlet concentrations are defined as `Dirichlet` boundary conditions

```
BC b_inlet1  0 0    JType ny/2-1 Dirichlet A 1.0 Dirichlet B 0.0
BC b_inlet2  0 ny/2 JType ny/2   Dirichlet A 0.0 Dirichlet B 1.0
```

Now that the initial section is complete, parameters that control the actual computation need to be given within the command section. First, we specify initial values for the relevant fields `A`, `B`, `Pressure`, `Velocity.Y`, and `Velocity.X`

```
Solve Init A=0.5 B=0.5 Velocity=1?(velocity):(6*velocity*(y/ly-(y/ly)**2)),0
```

Note that for `Velocity.X` we allow for two different velocity distributions, either the homogeneous plug flow or the parabolic Hagen-Poiseuille flow. Next, we define the fields to be solved for using the `BlockStruct` statement, set convergence criteria and execute the actual calculation with `Solve Stationary`

```
ConvergGlobal  AbsIncr.A<1.E-12 && AbsIncr.B<1.E-12
BlockStruct Block A B

Solve Stationary
```

Finally, post-processing statements are added to write the concentrations at different cross-sections to external files.

Numerical Results

Fig. 2.115 shows the concentration fields as computed by *SESES* for the cases $K = 0.1$ and $K = 0.01$, respectively. One sees that the mixing efficiency decreases for smaller K-values. In Fig. 2.116, the same numerical results are compared with the analytical solution. In general, there is an excellent agreement between the two methods and the agreement becomes better for smaller K-values. The reason could be that in the derivation of (2.148), the diffusion-transport has been neglected in the x-direction, which is only valid when $K \ll 1$. Fig. 2.117 shows again the numerical results for the plug flow but now compared with the results obtained for the parabolic Hagen-Poiseuille flow. Note that for a slit, the Hagen-Poiseuille profile is given by

$$V_x(y) = 6\, V_{\text{aver}} \left[\frac{y}{L_y} - \left(\frac{y}{L_y} \right)^2 \right], \tag{2.149}$$

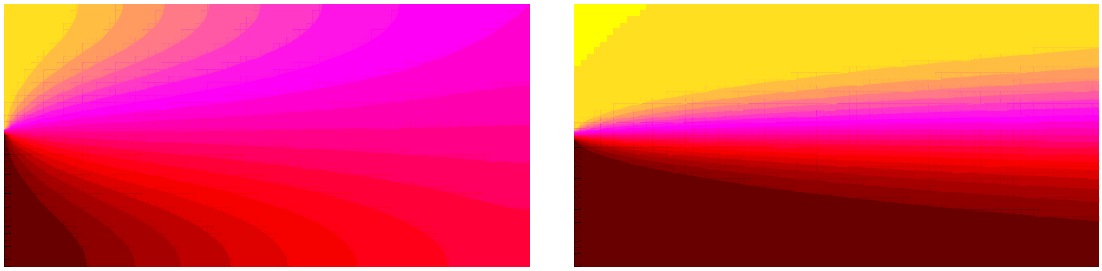

Figure 2.115: Concentration fields in the straight channel for (left) $K = 0.1$, and (right) $K = 0.01$, as predicted from the *SESES* model.

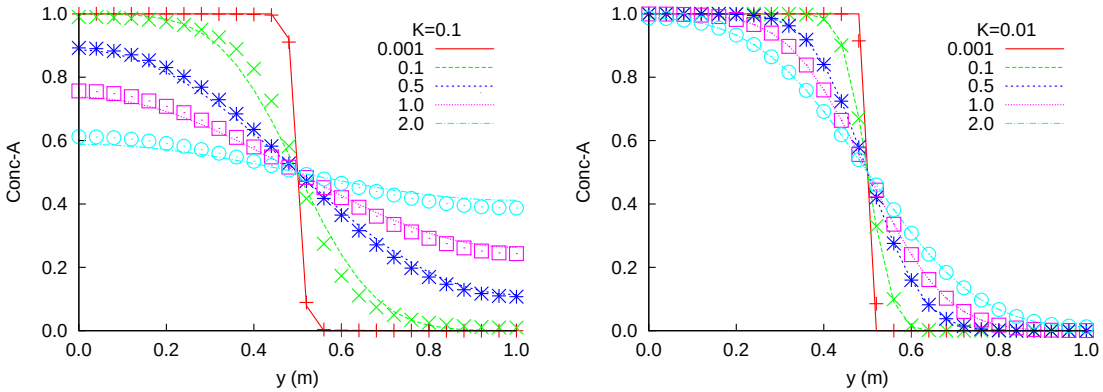

Figure 2.116: Concentration profiles over different cross-sections along the straight channel shown in Fig. 2.113. The parameter values are (left) $K = 0.1$ and (right) $K = 0.01$, respectively. Comparison of the analytical results based on (2.148) (shown as the lines) with the *SESES* predictions (shown as the symbols).

where V_{aver} represents the average velocity [7]. As expected, there is better mixing in the Hagen-Poiseuille compared to the plug flow case. However, for $K = 0.1$, the differences are rather small but become larger as K decreases.

References

[1] B. He, B. J. Burke, X. Zhang, R. Zhang, F. E. Regnier, *A picoliter-volume mixer for microfluidic analytical systems*, Analytical Chemistry, Vol. 73, No. 9, pp. 1942-1947, 2001.

[2] A. E. Kamholz, B. H. Weigl, B. A. Finlayson, P. Yager, *Quantitative Analysis of Molecular Interaction in a Microfluidic Channel: The T-Sensor*, Analytical Chemistry, Vol. 71, No. 23, pp. 5340-5347, 1999.

[3] F. Ismagilov, D. Rosmarin, P. J. A. Kenis, D. T. Chiu, W. Zhang, H. A. Stone, G. M. Whitesides, *Pressure-Driven Laminar Flow in Tangential Microchannels: an Elastomeric Microfluidic Switch Rustem*, Analytical Chemistry, Vol. 73, No. 19, pp. 4682-4687, 2001.

[4] T. J. Johnson, D. Ross, L. E. Locascio, *Rapid Microfluidic Mixing*, Analytical Chemistry, Vol. 74, No. 1, pp. 45-51, 2002.

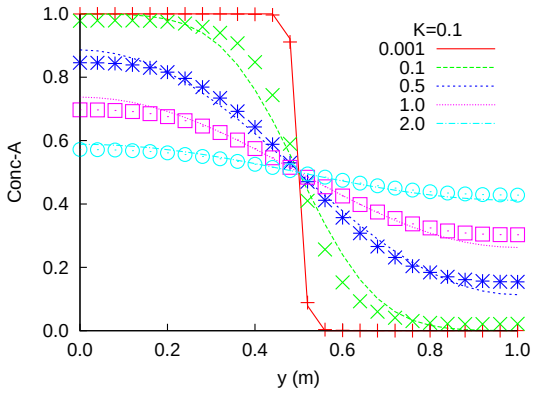

Figure 2.117: Concentration profiles at different cross-sections for $K = 0.1$. Comparison of the already shown *SESES* results based on a plug flow (shown as the lines) with those based on a Hagen-Poiseuille flow (shown as the symbols).

[5] I. MÜLLER, *Grundzüge der Thermodynamik*, 3rd Ed., Springer, 2001.

[6] M. N. ÖZIŞIK, *Boundary Value Problems of Heat Conduction*, Dover-Reprint, 1989.

[7] R. B. BIRD, W. E. STEWART, E. N. LIGHTFOOT, *Transport Phenomena*, Wiley, 1960.

2.28 Heat Transfer and Natural Convection in a Closed Cavity

Natural convection can be found nearly everywhere in nature. For example, the circulation of the earth atmosphere and the global wind systems are mainly driven by natural convection. In the presence of a gravity field, natural convection is initiated by density differences in liquids or gases. These differences may be due to the presence of different chemical species or due to temperature differences within the field of a pure fluid. Natural convection also is important in numerous technical applications. In engineering, natural convection plays e.g. an important role in the heating, ventilation, and air conditioning of buildings as well as in the field of solar energy. A well-known phenomena of natural convection can be found in living rooms where the air cools down at cold surfaces, mainly at the windows. Due to the buoyancy forces the cold air which is heavier than the warmer air in the room falls off to the floor. On the other hand, the cold air is heated up again at the warm inner walls and at the hot surfaces of the heating radiators. The interaction of heating (in the center) and cooling (at the windows) produces an air circulation in the room. Human beings perceive this air movement as a draught. Such a circulation can be found in any simple cavity where one side is heated and the other side is cooled. The flow pattern which is established can be seen from Fig. 2.118. The aim of this tutorial example is to develop a simple FE-model which reproduces natural convection flow patterns similar to the one shown in Fig. 2.118. To set up the model consider the two-dimensional cavity shown in Fig. 2.119. The cavity is heated from the left side with a temperature of $T_{\mathrm{hot}} = 310\,\mathrm{K}$ and cooled from the right side with $T_{\mathrm{cold}} = 300\,\mathrm{K}$. The width and the height of the cavity are chosen to be the same, i.e. $h = s = 50{\times}10^{-3}\,\mathrm{m}$. To analyze convective heat

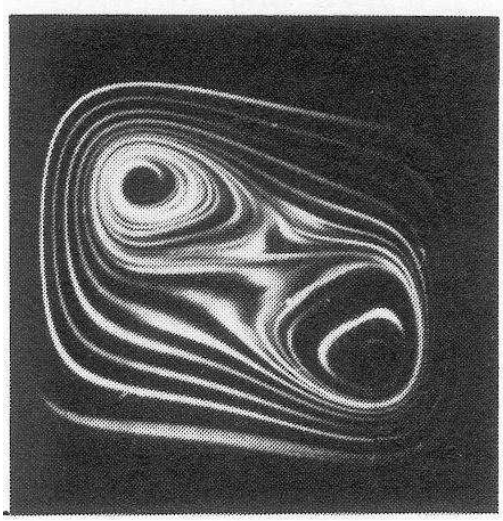

Figure 2.118: Experimental investigation of the flow pattern in a cavity where an external temperature gradient is maintained [1].

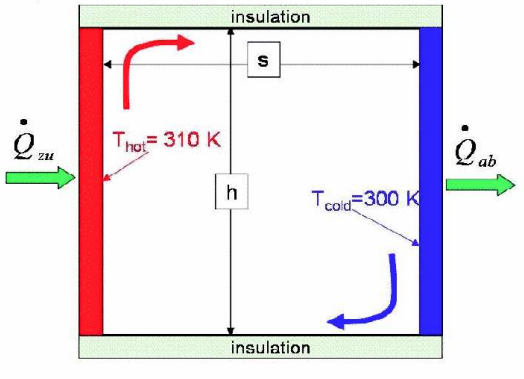

Figure 2.119: Geometry and boundary conditions for the air filled cavity on which the presented *SESES* model is based; width $s = 50{\times}10^{-3}\,\mathrm{m}$, height $h = 50{\times}10^{-3}\,\mathrm{m}$ and $\mathrm{Ra} = 1.08{\times}10^{5}$.

transfer phenomena, dimensionless numbers can be derived from similarity considerations. These numbers characterize the type of flow field as well as the observed heat transfer. In the case of natural convection, three numbers, the Nusselt number, the Grashof number and the Prandtl number are sufficient to describe any situation [3]. The Nusselt number Nu is a dimensionless form of the heat transfer coefficient α defined as

$$\mathrm{Nu} = \frac{\alpha L}{\kappa}. \tag{2.150}$$

It relates α to its limiting value in the case of pure heat conduction κ/L where κ denotes the thermal conductivity and L is a characteristic length. The Grashof number Gr represents the relation between buoyancy forces and viscous forces. It can be interpreted as the Reynolds number in natural convection problems and is defined as

$$\mathrm{Gr} = \frac{g\,h^3}{\nu^2}\frac{\Delta T}{T}\,. \tag{2.151}$$

The parameter h is the characteristic length of the considered flow problem e.g. the height of a heated vertical wall. Furthermore, g is the gravity constant, ν is the kinematic fluid viscosity, $T = T_{\mathrm{fluid}}$ and $\Delta T = T_{\mathrm{wall}} - T_{\mathrm{fluid}}$ is the external temperature gradient applied. Note that in the case of a cavity, the applied temperature difference is $\Delta T = T_{\mathrm{hot}} - T_{\mathrm{cold}}$. The third dimensionless number considered is the Prandtl number, Pr, which represents the ratio of the thickness of the hydrodynamic to the thermal boundary layer,

$$\mathrm{Pr} = \frac{\nu}{a} = \frac{\mu\,c_p}{\kappa}\,. \tag{2.152}$$

Here μ, c_p, and a represent the dynamic viscosity, the heat capacity, and the thermal diffusivity of the considered fluid, respectively. Note that for air, we have $\mathrm{Pr} \approx 0.7$. For natural convection problems, the heat transfer coefficient then can be expressed as

$$\mathrm{Nu} = \mathrm{Nu}(\mathrm{Gr}, \mathrm{Pr})\,. \tag{2.153}$$

It turns out that the type of flow field observed depends on another dimensionless number, the so-called Rayleigh number Ra defined as the Grashof times the Prandtl number

$$\mathrm{Ra} = \mathrm{Gr}\,\mathrm{Pr} = \frac{g\,h^3}{\nu\,a}\frac{\Delta T}{T}\,. \tag{2.154}$$

For $\mathrm{Ra} < 10^8$, the fluid flow is always laminar. Note that at the present time, flow simulations using *SESES* are restricted to laminar flow only. For setting up the problem, we therefore have to ensure that the applied temperature difference ΔT as well as the characteristic length h are chosen such that the condition $\mathrm{Ra} < 10^8$ is fulfilled. For the situation of free convection at vertical walls, results from a large number of experiments were combined to derive the following empirical relationship (see VDI-Wärmeatlas [2])

$$\mathrm{Nu} = \left[0.825 + 0.387\,(\mathrm{Ra}\,f_1)^{1/6}\right]^2\,, \tag{2.155}$$

$$f_1 = \frac{1}{\left[1 + \left(\frac{0.492}{\mathrm{Pr}}\right)^{9/16}\right]^{16/9}}\,. \tag{2.156}$$

To evaluate the accuracy of the obtained *SESES* simulation results, they will be compared with the above correlation. For this, to apply (2.155) to the cavity shown in Fig. 2.119, ΔT as it appears in (2.152) is defined as $\Delta T = (T_{\mathrm{hot}} - T_{\mathrm{cold}})/2$.

Setting up the problem

To boot up the calculations in two different modes, the input file `example/FreeConv.s2d` starts by defining the control parameter `TVAR`.

```
Define
TVAR    = 1                 (* 1: variable temperature, 0: variable mesh *)
lsys0   = 50*1e-3           (* minimum system dimension in m *)
nMax    = 3                 (* number of solutions *)
dT      = 1                 (* temperature increments *)
dlsys   = 15*1e-3           (* length increments in m *)
lsys    = lsys0             (* system dimension in m *)
Thot    = TVAR?301:305      (* hot plate temperature in K *)
```

Using the mode `TVAR = 1`, the temperature `Thot` of the hot plate is raised in increments of `dT`. Similarly, for `lVAR=0` , the system dimension `lsys` is raised in increments of `dlsys`. This allows one to increase the Ra number – which depends on both, the applied external temperature gradient, as well as on the system dimension – by small increments (see Section 2.28 for details). The temperature on the hot side starts with the value of $301\,\mathrm{K}$ in case of `TVAR=1`. The temperature is increased in 10 steps (`nMax=10`) by temperature increments of $1\,\mathrm{K}$ to its final value of $311\,\mathrm{K}$. The temperature of the cold side remains unchanged and is specified through the parameter `T0` which is defined in the section `physical parameters`. The procedure is similar in the case when the system dimensions are enlarged. There the initial system length of `lsys=50*1e-3m` is increased in increments of `dlsys=15*1e-3m` to its final value of `200*1e-3m`.

The next paragraph deals with the system geometry. We consider a square domain with a minimum size of `lsys0` and (`nx`,`ny`) macro element subdivisions in x- and y-directions. To be able to change the system size within a single *SESES* run to the size `lsys`, we apply a scaling transformation.

```
QMEI nx=10  lsys0/nx QMEJ ny=10  lsys0/ny

CoordNonConst

Coord  coord*lsys/lsys0  1 (* scale by a factor of lsys/lsys0 *)
```

We next specify the physical parameters of air. First, *SESES* routines are defined for the following properties of air: the heat capacity c_p, the density ρ, the molar mass M, the shear viscosity μ and the thermal conductivity κ as well as its derivative with respect to temperature. As the reference state, air at a pressure of $p_0 = 10^5\,\mathrm{Pa}$ and a temperature of $T_0 = 300\,\mathrm{K}$ is considered. Since we only allow κ to vary with temperature, all other material functions are evaluated at T_0. The force vector in the momentum balance is identified with the gravity force $\mathbf{f} = \rho(0, g, 0)^{\mathrm{T}}$ acting in the y-direction.

```
GlobalSpec (* global parameters *)
  Parameter  AmbientTemp  T0 K

MaterialSpec fluid (* fluid parameters *)
  Equation   CompressibleFlow ThermalEnergy Enable
  Parameter  FlowStab        zero
  Parameter  Density         IdealGas()            SIunit
  Parameter  Mmol            M_AIR()               kg/mol
  Parameter  Viscosity       Visco_AIR(T0)         Pa*s
  Parameter  KappaIso        Kappa_AIR(T0)         W/(K*m)
  Parameter  Force D_Temp D_Pressure (* buoyance y-force *)
  {
     Force.X          =0;
     Force.X_DTemp    =0;
     Force.X_DPressure=0;
     Force.Y          =Density.Val       *gAccel;
     Force.Y_DTemp    =Density.DTemp     *gAccel;
     Force.Y_DPressure=Density.DPressure *gAccel;

  }  SIunit
  Parameter ThermStab  zero
  Parameter ThermConv  Cp_AIR(T0)*Density.Val*Velocity W/(m**2*K)
  Parameter ThermConvDVelocity  Cp_AIR(T0)*Density.Val  J/(m**3*K)
```

Before we can start our simulation, a complete set of boundary conditions must be defined. Since our simulation domain is the 2D cross-section through a closed box, no-slip boundary conditions are applied to all four boundaries

```
BC  zeroVelocity OnChange 1
    Dirichlet Velocity 0,0 m/s (* zero velocities at boundaries *)
```

In addition, the boundaries on the left/right are kept at the constant temperature of `Thot/T0`

```
BC  hotPlate    0 0 JType ny
    Dirichlet Temp  Thot K (* temperature left plate = Thot *)

BC  coldPlate   nx 0 JType ny
    Dirichlet Temp  T0  K (* temperature right plate = T0 *)
```

Note that *SESES* uses `Natural` boundary conditions when nothing is specified otherwise. This means that the normal-component of the flux associated with a certain field variable is set to zero. Since we do not specify explicit thermal conditions for the upper and lower boundaries, this implies zero heat flux over these boundaries, which is consistent with the boundary conditions shown in Fig. 2.119. With the specification of the boundary conditions the initial section of the input file is now complete.

Computation and postprocessing

The following section deals with the actual computation. The fields to be solved for are defined within a single `Block`, and convergence criteria are set.

```
BlockStruct Block Pressure Velocity Temp
  Convergence (AbsIncr.Temp<1.E-8 && AbsIncr.Pressure<1.E-5 &&
               AbsIncr.Velocity<1.E-10)?1:
               (nIter>=15)?failure("too many iterations"):0
```

As mentioned earlier, the calculations can be performed in two different modes, one where the temperature of the hot plate is raised in steps of `dT` (with `TVAR = 1`) and the other one where the system size is raised in steps of `dlsys` (with `lVAR = 0`). For each mode, separate instructions to control the calculations are defined to compute a stationary solution.

```
Solve Init Temp=T0
Solve Stationary
  If TVAR { ForSimPar Thot Step dT     Until T0+dT*nMax    }
  Else    { ForSimPar lsys Step dlsys  Until  lsys0+nMax*dlsys }
```

For the analysis of the simulation results, we add a number of post-processing statements to write the desired information to external files. We first calculate the heat-fluxes (per area) over the left and right boundaries, the resulting heat-transfer coefficient and Nusselt number, as well as the Prandlt and Grashof numbers.

```
  lm     = lsys                              (* system dimension in m *)
  A      = lm*1e-3                           (* cross-section in m^2 *)

  qfluxC = coldPlate.Temp.Flux/A
  qfluxH = hotPlate.Temp.Flux/A

  alpha  = qfluxH/(Thot-T0)
  Nu     = alpha*lm/Kappa_AIR(T0)

  Pr     = Visco_AIR(T0)*Cp_AIR(T0)/Kappa_AIR(T0) (* Prandtl number *)
  Gr     = gAccel*pow(lm,3)*(Thot-T0)/(T0*pow(kinVisco,2))
```

Together with the material properties of air (already defined in the initial section), for each run, these results are then written to external properties files.

Solution strategies

As explained in the introduction, the dimensionless Rayleigh number, Ra, characterizes the flow regime that follows from exposing a fluid in a gravity field to an external temperature gradient. In our example where $h = s = 50{\times}10^{-3}\,\mathrm{m}$ and $\Delta T = T_{\mathrm{hot}} - T_{\mathrm{cold}} = 10\,\mathrm{K}$, this yields a Rayleigh number of $\mathrm{Ra} = 1.08{\times}10^{5} < 10^{8}$, i.e. a laminar flow field is achieved. If the height and width of the square cavity is enlarged to a value of $h = s = 200{\times}10^{-3}\,\mathrm{m}$, the Rayleigh number increase to $\mathrm{Ra} = 6.9{\times}10^{6}$, i.e., it is now much closer to the Ra number of $\mathrm{Ra} = 10^{8}$, where the transition between laminar and turbulent flow occurs. In an experiment, this would mean that any disturbances which might occur in the flow need some time to be dampened by viscous forces; eventually, these disturbances completely disappear. The numerical simulation shows a similar behavior. However, a simulation under these conditions is likely to fail. Convergence might not be achieved due to numerical errors which

will not be dampened out during the iteration process. If the simulation is initiated with the initial condition that the air is at rest ($u_{\text{ini}} = v_{\text{ini}} = 0$) and with a uniform temperature field of $T_{\text{ini}} = (T_{\text{hot}} - T_{\text{cold}})/2$ the solution process will only be successful if the Ra number is significantly lower. In fact, the limit is around $\text{Ra} = 10^5$. The convergence problems for higher Ra numbers can be overcome by starting the simulation from a previous, already converged solution. In this previous solution , the temperature and velocity distributions were obtained on the same computational domain, but for a smaller Ra number. There are two strategies to reduce the Ra number. The first is to start with a similar geometry which is scaled down so that the factor h^3 in the expression for Ra is reduced. Alternatively, one starts with a smaller temperature difference ΔT between the hot and cold plates which also reduces the value for Ra. As mentioned, convergence may be achieved starting the simulation from air at rest for a Ra number of $\text{Ra} < 10^5$. After finishing the first simulation with a low value, the Ra number has to be increased to its final value by enlarging the geometry or the temperature difference step by step.

Comparison of *SESES* with CFX-TASCflow simulations and experimental verification

Several results obtained from running the above described *SESES* file are shown in following graphs. In Fig. 2.120, the velocity distribution as obtained from the *SESES* simulation is compared with CFX-TASCflow [4] simulation results that were obtained under identical conditions. There flow field predictions of the two simulation-tools agree very well with each other. Note also that there is some similarity between the simulated flow field and the experimental one shown in Fig. 2.118. The same good agreement can be seen by comparing the temperature distributions obtained from *SESES* and CFX-TASCflow simulations, shown in Fig. 2.121. Finally Table 2.3 compares the values for the heat flux in horizontal direction obtained from the simulations with the calculated value as obtained from (2.155). Again, there is satisfying agreement between the different solutions.

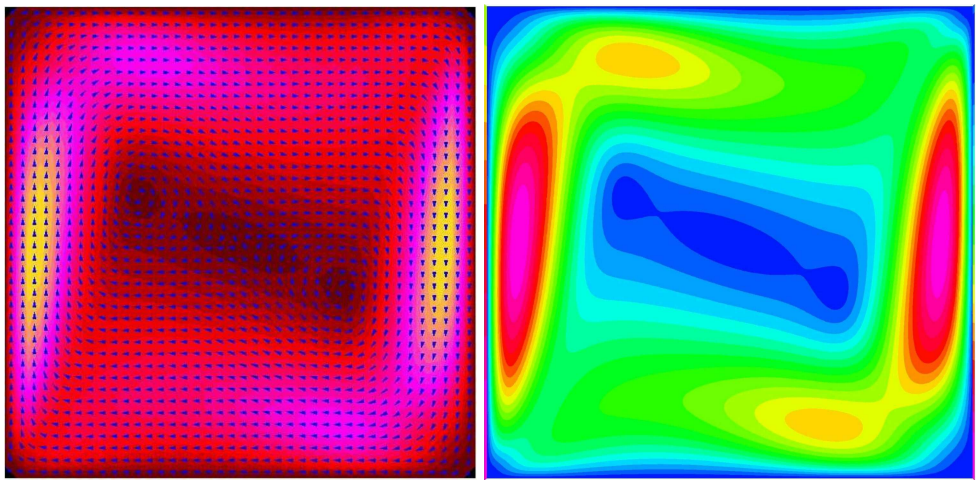

Figure 2.120: Calculated velocity field in cavity with *SESES*(left) and with CFX-TASCflow [4] (right) for $\text{Ra} = 10^5$.

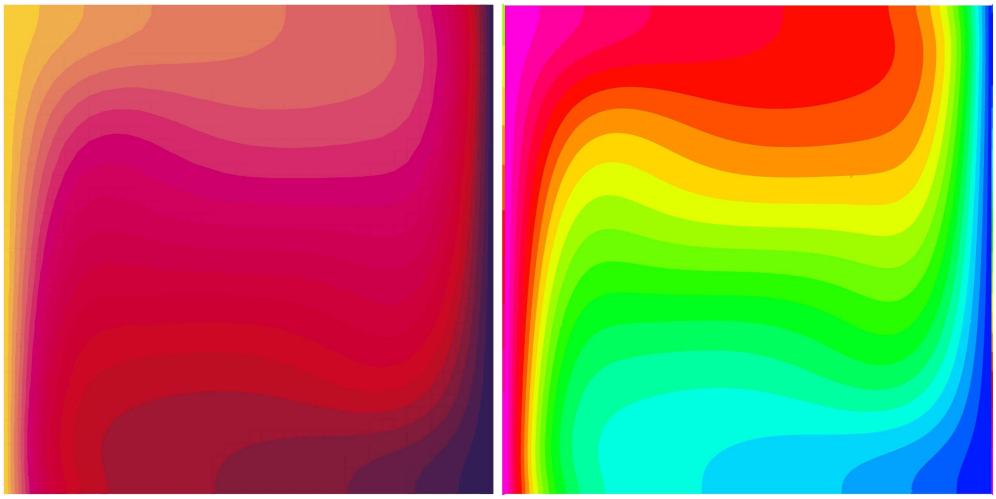

Figure 2.121: Calculated temperature field in cavity with *SESES*(left) and with CFX-TASCflow [4] (right) for $\mathrm{Ra} = 10^5$.

	SESES	CFX-TASCflow [4]	VDI-Wärmeatlas [2]	
$\mathrm{Ra} = 10^5$	23.8	23.7	21.0	$\mathrm{W/m^2}$

Table 2.3: Comparison of the horizontal heat flux q_{wall} in $\mathrm{W/m^2}$ for $\mathrm{Ra} = 10^5$.

References

[1] S.J.M. LINTHORST, W.M.M. SCHINKEL, C.J. HOOGENDOORN, *Flow Structure with Natural-Convection in Inclined Air-Filled Enclosures*, Journal of Heat Transfer–Transactions of the ASME, Vol. 103, No. 3, pp. 535-539, 1981.

[2] VDI-WÄRMEATLAS, *Berechnungsblätter für den Wärmeübergang*, Hrsg.: Verein Deutscher Ingenieure, Springer-Verlag, 2002.

[3] YUNUS A. ÇENGEL, *Heat Transfer–A Practical Approach*, Mc Graw Hill, 2003.

[4] CFX-TASCFLOW, CFD-Tool, ANSYS Inc., Canonsburg PA (USA), http://www-waterloo.ansys.com/cfx/.

2.29 Calorimetric Flow Sensor

In this tutorial example the operating principle of a calorimetric flow sensor and its implementation in *SESES* is discussed. Rather than focusing on the geometrical design of realistic flow sensors, we wish to illustrate a modeling method for this specific microfluidic device that relies on the solution of the heat conduction equation with forced convection.

Many different physical principles for measuring flow are employed in practice [1]. For conductive fluids, an electromagnetic measuring principle based on the induced volt-

[1] see, for instance, www.flowmeterdirectory.com

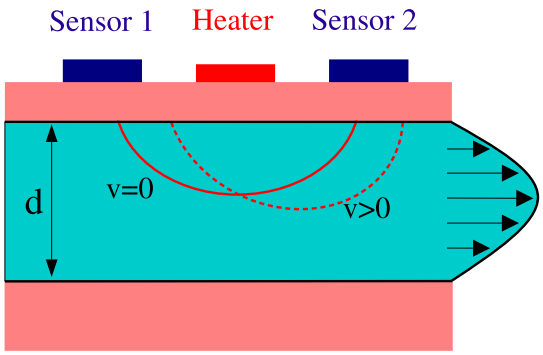

Figure 2.122: Geometry of a typical calorimetric flow sensor.

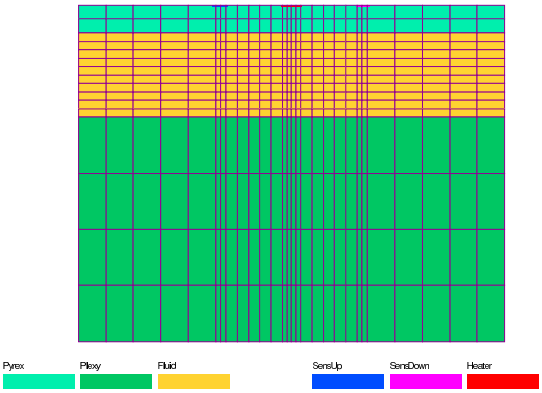

Figure 2.123: Simulation domain with the boundary conditions highlighted.

Operation Mode	Characteristics	Application	Comments
Constant Power P	$\Delta T_{2-1} = f(v_0)$	small fluid flows, gaseous media	no temperature regulation necessary
	$T_H = f(v_0)$	fluid flows	
Constant Temperature T_H	$\Delta T_{2-1} = f(v_0)$	small fluid flows	temperature regulation
	$P = f(v_0)$	medium fluid flows	fast response

Table 2.4: Operating principles of the calorimetric flow sensor.

age in a magnetic field is employed. For microfluidic systems, the calorimetric flow sensor principle is widely used. Such sensors contain a heater and two temperature sensors that are in good thermal contact with the flow channel volume, see Fig. 2.122. Typically, the sensor and heater pads are separated from the flow channel by a thin membrane. If the medium is at rest, a symmetric temperature distribution around the heater is expected and the two sensors, one upstream and one downstream from the heater location, measure identical temperatures. For non-zero flow rates, however, the temperature distribution is not symmetric any more. The difference of the two sensor temperatures is then proportional to, and therefore a measure of, the flow rate. The design of a specific sensor geometry will depend on the application of interest and the sensor electronics at hand. The simulations documented here allow for efficient sensor characterization prior to fabrication. In particular, a qualitative and quantitative understanding of sensor operation is enabled by the simulation. We attempt to illustrate what kind of questions might arise when developing a calorimetric flow sensor [1]. Table 2.4 gives an overview of the operating principles commonly used with calorimetric flow sensors.

Each operation mode has its benefits and drawbacks. For instance, in constant power mode, a regulation of the temperature of the heater is not necessary but the sensor signal will react slowly to an abrupt change in the flow rate. The simulation results for different operating principles will be discussed below.

Model Specification

Let us now build a *SESES* model of the flow sensor. Since we focus on the solution algorithm and the operating principle, we choose a simple 2D model of the sensor geometry and solve the heat transport equation only. I.e. rather than solving a fully coupled thermo-fluidic model, we shall solve the heat transport equation only and prescribe the fluid flow independently as a parabolic velocity profile

$$v = v_0\,(1 - \frac{4y^2}{d^2}), \tag{2.157}$$

where v_0 is the maximum speed at the center ($y = 0$) of the channel and d the height of the channel. Equation (2.157) is the *Hagen-Poiseuille* law for viscous laminar flow, see also the example on page 163. This simplification is justified whenever the temperature rise is relatively small and thus not changing the fluid properties themselves.

Let us discuss now the key parts of the input file `example/FlowSens.s2d` provided for this example. The geometry of our sensor is comprised of a horizontal fluid channel, a medium above (pyrex) and a medium below (plexy glass), see Fig. 2.123. The temperature sensors and the signal analysis electronics are both located on the upper side of the thin pyrex plate. For the fluid flow, we define the material `Fluid` solving for the convective and diffusive heat transport.

```
MaterialSpec Fluid Equation   ThermalEnergy
ElmtFieldDef veloc(DofField Temp; ForMaterial Fluid)[2]

MaterialSpec Fluid
  Parameter  KappaIso      0.585  W/(K*m)
  Parameter  Density.Val   rhoFl  kg/m**3
  Parameter  Enthalpy      cpFl*Temp,cpFl,0 J/kg-J/(K*kg)-J/(K**2*kg)
  Parameter  ThermConv     Enthalpy.DTemp*Density.Val*veloc W/(m**2*K)
  Parameter  ThermStab     zero
```

The parameters stated above correspond to water. In order to consider the convective heat transport, we have to define the term $\rho\partial_T h\mathbf{v}$ with the material parameter `ThermConv` with h the enthalpy, ρ the density and $\mathbf{v}$ the flow velocity. The user element field `veloc` is used here to store the velocity profile. Since for this flow sensor, the convective flow is small with respect to the diffusive one, there is no need to use streamline diffusion stabilized finite elements and therefore we set to zero the stabilization term.

For the boundary of the simulation domain we formulate a heat transfer boundary condition (not shown in the screenshot of Fig. 2.123)

```
BC Transfer  OnChange 1  OnBC Heater  Disable
   Neumann Temp D_Temp alpha*(Temp-Tamb) alpha (W/cm**2)-W/(cm**2*K)
```

with α the heat transfer coefficient. The heater is specified either with a `Dirichlet` or a `Neumann` type boundary condition, depending on whether we wish to operate the sensor in constant heater temperature or power mode, also compare with Table 2.4. To implement the constant power mode we write

```
BC Heater    nnx3 nytot IType nnx4-nnx3
   Neumann Temp -Power/(lc*hcb) W/um**2
```

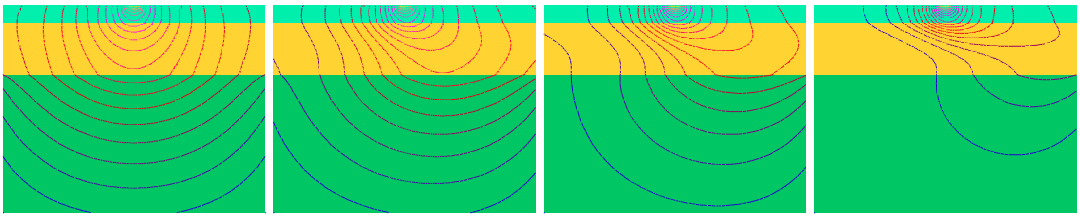

Figure 2.124: Steady-state temperature fields for different flow rates at identical power of the heater. The upper left shows the temperature field when the fluid is at rest, followed by examples of increasing flow rates.

where the negative sign prescribes the inward direction of the heat flux. The temperature sensors have been defined as boundary conditions of `Floating` type

```
BC SensUp    nnx1+1 nytot IType nnx2-nnx1-2 Floating Temp 0 W
BC SensDown  nnx5+1 nytot IType nnx6-nnx5-2 Floating Temp 0 W
```

and are visible in the graphics window. The temperature and the thermal flux (power) at these boundaries will be exported for postprocessing.

In the command section we first initialize the velocity and we specify the temperature field as the only `dof` field to be calculated, since the velocity field is prescribed and it is stored in the user element field `veloc`

```
Store veloc=v0*(1-4*y*y/(h*h)),0
```

Constant Heating Power Mode

For designing a realistic flow sensor, the geometry is optimized such that the sensor characteristics are as desired for the application of interest. In particular, the relation between the flow rate and the sensor temperatures for a given heater power range needs to be analyzed. This relation will then be used for calibration in practice. First, let us consider the situation when the fluid is not flowing for reference. For a non-zero heating power and a zero flow, one expects a symmetric temperature field, as shown in the left corner of Fig. 2.124. Steady-state temperature fields for increasing flow rate are shown in the other examples of Fig. 2.124. The difference in the thermal conductivity is apparent in the contour lines traversing material boundaries. As expected, the temperature field is moving downstream with increasing flow rate.

The sensor calibration curve is the curve relating the measurement signal and the quantity of interest, i.e. the flow rate. In our example, we calculate the dependence of the sensor temperature difference $\Delta T_{2-1} = T_2 - T_1$ on the mid-channel velocity v_0. Fig. 2.125 shows the dependence of ΔT_{2-1} and the heater temperature rise above ambient temperature ΔT_H at a given power density applied to the heater. We note that ΔT_{2-1} exhibits a maximum value while ΔT_H decreases monotonically with increasing velocity in the channel. The slope of the ΔT_{2-1} curve decreases and thus the sensitivity of the sensor is reduced at higher velocities. The heating power influences the flow rate range in which the sensor can be operated. As an example, Fig. 2.126 shows such a dependence for the same geometry as above.

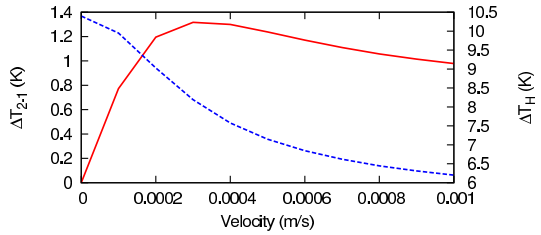

Figure 2.125: Sensor calibration curve relating the sensor temperature difference ΔT_{2-1} (red) and the heater temperature rise ΔT_H (blue) to the velocity v_0 in the center of the flow channel in constant power mode (heating power $P = 10^{-3}$ W).

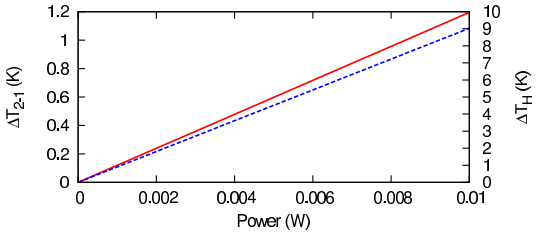

Figure 2.126: Plot of the sensor temperature difference $\Delta T_{2-1} = T_2 - T_1$ (red) and ΔT_H (blue) versus the power applied to the heater (velocity $v_0 = 2 * 10^{-4}$ m/s).

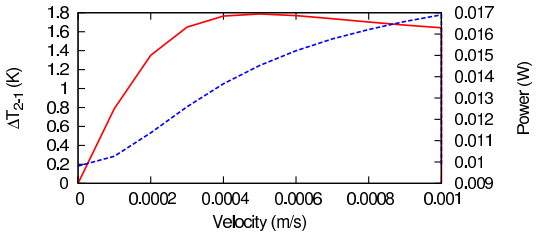

Figure 2.127: Sensor calibration curve relating the sensor temperature difference ΔT_{2-1} (red) and the heater power to the velocity v_0 in constant temperature mode (heater temperature rise above ambient: $\Delta T_H = 10\,K$).

Constant Sensor Temperature Mode

As indicated in Table 2.4, an alternative measurement mode adjusts the heater power density at constant heater temperature. For this mode one needs to measure the local heater temperature and then regulate the heating power to maintain a constant heater temperature. The sensor temperature difference ΔT_{2-1} and the heater power is plotted in Fig. 2.127 versus the flow velocity v_0. The temperature difference ΔT_{2-1} assumes a local maximum and decays at large velocities as in Fig. 2.125. By contrast, the heater power density increases monotonically with increasing flow velocity.

Dynamic Response

For sensor applications, the dynamic response behaviour to a stepwise increase of the flow rate is often critical. In order to simulate the time-dependent evolution of the temperature field in the flow sensor, the thermal problem is solved dynamically while the velocity field is kept constant. This method is justified, by arguing that heat transport happens on a time scale much longer than the fluidic mass flow. Therefore, we simply increase the magnitude of the velocity field at certain instants in time and calculate how the temperature field evolves and approaches a new steady-state.

For instance, in constant power mode, we perform a calculation of the transient temperature field for a series of flow rate steps. The flow rate staircase and the corresponding unstationary solution of the temperature field is carried out with the statements

```
For i From 0 To nFlowSteps
```

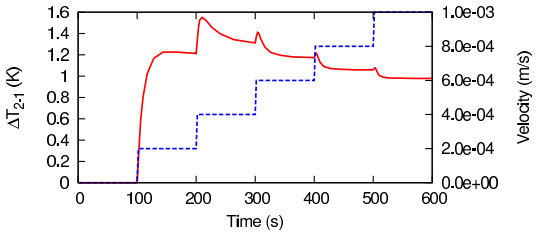

Figure 2.128: Dynamic response of the sensor temperature difference ΔT_{2-1} (red) to a series of velocity v_0 steps (blue) in constant power mode ($p = 10\,\mathrm{mW}$).

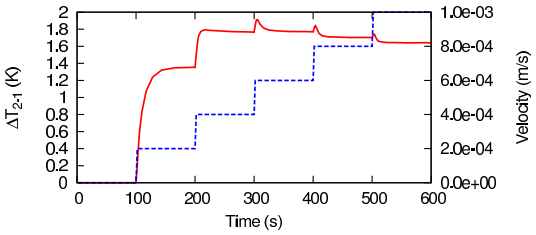

Figure 2.129: Dynamic response of the sensor temperature difference ΔT_{2-1} (red) to a series of velocity v_0 steps (blue) in constant temperature mode ($\Delta T_H = 10K$).

```
{
   Define v0=@i/nFlowSteps*v0max
   Store veloc=v0*(1-4*y*y/(h*h)),0
   Write AtStep 1 File "pdyn.txt" Append
     "%12.10f " time+@i*tStop
     "%12.10f " SensDown.Temp.Shift-Tamb
     "%12.10f " SensUp.Temp.Shift-Tamb
     "%12.10f " integrate(Bound Heater; Temp)/(lc*micron)-Tamb
     "%.3e\n" v0

   Solve Unstationary At 0
     Step tStop/50,automatic_step Until tStop Failure step/2
}
```

where we have used a loop for the different flow rate levels and the unstationary temperature solutions. The average temperature at the heater boundary is calculated using the integral of the temperature at that boundary condition. The transient sensor temperatures are written to the file `pdyn.txt`. The resulting transient temperature difference ΔT_{2-1} is depicted in Fig. 2.128 and resembles the curve in Fig. 2.125, as expected. At the first flow rate step the temperature field changes from a symmetric to an unsymmetrical field as a function of time in a similar manner as depicted in Fig. 2.124 for the different flow rates. For the constant heater temperature mode, the sensor temperature transient ΔT_{2-1} is shown in Fig. 2.129. In this plot the approach to steady state upon each flow rate step is faster than for the constant power mode.

As discussed above in the stationary simulation shown in Fig. 2.125, the heater temperature can be used as the sensor signal in constant power mode. The transient response of this signal is depicted in Fig. 2.130 with an monotonic decrease of the steady-state values that is consistent with Fig. 2.125. Finally, we may also use the heater power as the sensor signal if the sensor is operated in constant heater temperature mode. In this case, the power increases to the steady state values as can be seen in the plot of Fig. 2.131. The plots of the dynamic responses show, that the constant heater temperature mode allows for faster response at the cost of power regulation.

In summary, we reported a modeling method for the time-dependent thermal analysis of flow-rate dependent heat transport in a calorimetric flow sensor with a straight microfluidic channel. Some general measurement principles have been discussed with the help of a simple 2D *SESES* model.

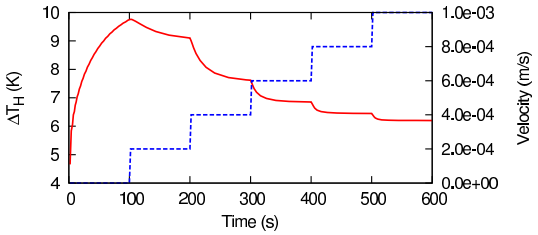

Figure 2.130: Dynamic response of the heater temperature rise ΔT_H (blue) to a series of velocity v_0 steps (blue) in constant power mode ($p = 10\,\mathrm{mW}$).

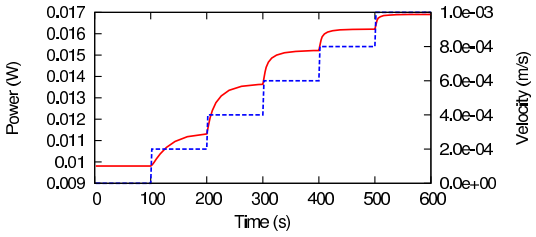

Figure 2.131: Dynamic response of the heater power (blue) to a series of velocity v_0 steps (blue) in constant temperature mode ($\Delta T_H = 10\,\mathrm{K}$).

References

[1] G. GERLACH, W. DÖTZEL, *Grundlagen der Mikrosystemtechnik*, Hanser Verlag, München, 1997.

2.30 Flow Around a Circular Cylinder

The flow around a circular cylinder has been the subject of intense research in the past, mostly by experiments but also by numerical simulations and important findings have been made. The flow situation is of relevance for many industrial applications, e.g. tubes in heat exchangers, chimneys, towers, antennae, masts, cables and offshore risers. The main interest is focused on the wake behind the cylinder and the forces on the cylinder, as they may cause the cylinder to oscillate with a certain frequency. If this frequency equals the eigen-frequency of the cylinder, the movement of the cylinder will increase and a material failure due large tensions may occur.

Depending on the Reynolds number, the flow around the cylinder can roughly be divided into five regimes. For very small Reynolds numbers up to 5, the flow is dominated by viscous forces and the flow field is the same in front and behind the cylinder. In the Reynolds number range from 5 to 48, the flow starts to separate behind the cylinder forming two separation regions with recirculating flow. The length of the bubbles increases with increasing Reynolds number. From Reynolds number 40 to 300, the vortices detach periodically from the cylinder and travel down stream. This phenomenon is called the Karman vortex street. For higher Reynolds numbers, the shedding of vortices becomes irregular and the flow becomes turbulent. At very high Reynolds numbers, the vortices are shed periodically again. A detailed description of the phenomenon of the flow around a circular cylinder can be found in [1].

In this tutorial, we will concentrate our interest to flow situations with Reynolds numbers ranging from 5 to 48. We will compare the calculated force, the length of the bubbles and the pressure distribution on the cylinder surface to data obtained by experiments and taken from [2]. The equations to be solved are the continuity and the Navier Stokes equations. For a steady incompressible flow, the continuity equation can be simplified as

$$\nabla \cdot \mathbf{v} = 0\,, \tag{2.158}$$

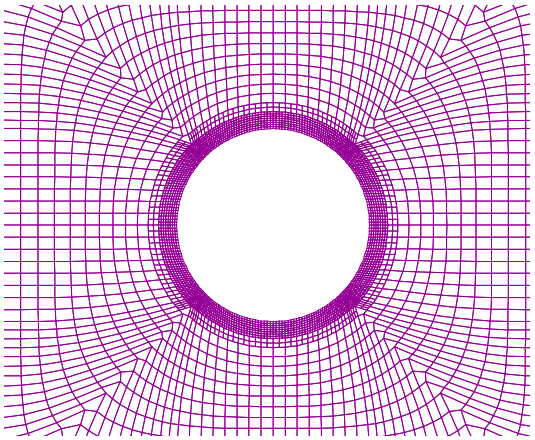

Figure 2.132: Generated grid around the cylinder.

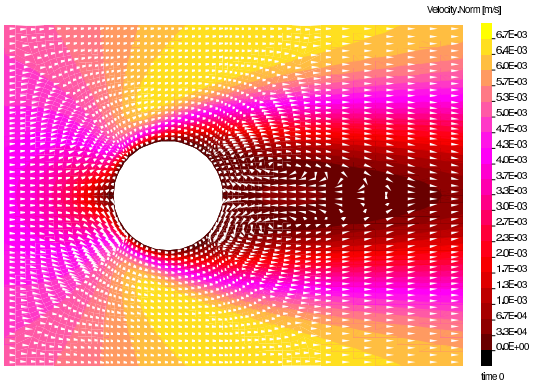

Figure 2.133: The computed velocity field behind the cylinder for $\mathrm{Re} = 46$, with the cones showing the direction of flow thus indicating the region of recirculating flow.

and the Navier Stokes equation as

$$\rho(\mathbf{v} \cdot \nabla)\mathbf{v} = -\nabla p + \mu \nabla^2 \mathbf{v}\,, \tag{2.159}$$

with ρ the constant density, $\mathbf{v}$ the velocity, p the pressure and μ the viscosity. These equations can not be solved analytically for the flow field at hand. For our computations, we will consider a cylinder of radius $0.05\,\mathrm{m}$ and infinite length and can therefore assume a 2D flow situation. We then define a domain of rectangular shape and dimensions $0.7 \times 1.4\,\mathrm{m}^2$ with the center of the cylinder placed at $0.55\,\mathrm{m}$ behind the flow entrance and in the center of the domain width. The fluid is assumed to be air of density $\rho = 1.156684\,\mathrm{kg/m}^3$ and viscosity $\mu = 1.849830 \times 10^{-05}\,\mathrm{kg/(ms)}$. We compute two flow fields with Reynolds numbers 23 and 45, where the Reynolds number is defined as

$$\mathrm{Re} = \frac{d\,u_\infty}{\nu}\,, \tag{2.160}$$

with d the diameter of the cylinder, u_∞ the velocity of the undisturbed flow and $\nu = \mu/\rho$ the kinematic viscosity.

Model Specification

We start the problem specification of the input file `example/FlowAroundCyl.s2d` by defining some user parameters to be used within algebraic expressions for pre- and post-processing purposes.

```
DensityAIR = 1.156684E+00                          (* air density in  kg/m3 *)
ViscosAIR  = 1.849830E-05                       (* air viscosity in  kg/(m*s) *)
rCyl       = 0.050                                 (* cylinder radius in m *)
CharLength = 2*rCyl                        (* characteristic length scale in m *)
NueAIR     = ViscosAIR/DensityAIR             (* kinematic viscosity in m^2/s *)
EpsP       = 1e-2                          (* regularization parameter in s *)
TauHydro   = CharLength^2/NueAIR  (* hydr. time scale in s to est regul par *)
Reynolds   = 23                                          (* Reynolds number *)
Veloc      = Reynolds*NueAIR/CharLength          (* mean flow velocity in m/s *)
```

The parameter `epsP` is a regularization parameter used to attenuate *numerical noise*, see the *SESES* manual for further details. As second step, we define the problem's

geometry and an initial rectangular mesh is reshaped into a domain modeling the circle and its exterior with the help of homotopic functions loaded with the `Include` statement. Four domain's boundaries `Inlet`, `Outlet`, `SymmetryWalls`, `Cylinder` are defined representing the flow inlet, the flow outlet, the surface of the cylinder and the upper-lower sides of the rectangular domain. This initial macro element is refined manually around the cylinder and the result is shown in Fig. 2.132. With the next statement, the material model for solving the stationary incompressible Navier-Stokes equations is defined.

```
MaterialSpec m_AIR
  Equation  CompressibleFlow Enable
  Parameter PressPenalty EpsP         s
  Parameter FlowStab     zero
  Parameter Density.Val  DensityAIR   kg/m**3
  Parameter Viscosity    ViscosAIR Pa*s
```

Before we start our simulation, the BC values must defined. The BCs for the side walls of our channel are so called symmetry conditions, no fluid can cross the boundary and the fluid can move along the boundary without experiencing friction (slip conditions). The BCs for the cylinder surface are no-slip conditions with a zero flow velocity. At the inlet, Dirichlet conditions are used to specify the flow velocity. Furthermore the total mass flux must be specified through a pressure floating condition. The pressure relative to the reference state is specified as a Dirichlet condition at the outlet.

```
BC SymmetryWalls Dirichlet Velocity.Y  0                    m/s
BC Cylinder      Dirichlet Velocity    0,0                  m/s
BC Inlet         Dirichlet Velocity    Veloc,0              m/s
                 Floating  Pressure   -Veloc*DensityAIR*Area kg/s
BC Outlet        Dirichlet Pressure    0                    Pa
                 Dirichlet Velocity.Y  0                    m/s
```

In the command section of the input file, we define the fields to be solved, set a convergence criteria on the velocity and call `Solve Stationary` to compute a steady state solution

```
BlockStruct
  Block       Pressure Velocity
  Convergence (MaxIncr.Velocity<1.E-12)?1:(nIter>100)?failure(""):0
  Increment   Standard #3 ReuseFactoriz #30

Solve Stationary

Dump Pressure Velocity MassFlow
```

The `Dump Pressure Velocity MassFlux` statement defines the field variables to be displayed in the *SESES* GUI. For the analysis of the simulation results, we need to add post-processing commands to write the desired information to external files. This will be explained in the next section.

Numerical Results

In order to check the accuracy of the results, we compare them to experimental data available from [2]. In particular, we are interested in the force exerted on the cylinder by the fluid flow. For a Reynolds number between 4 and 48, we expect two separation bubbles behind the cylinder with small velocities. In order to get an impression of

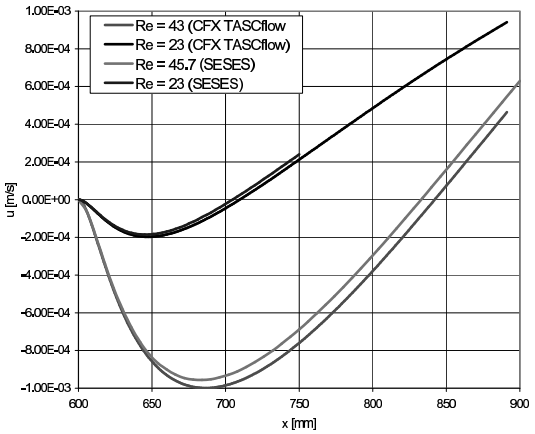

Figure 2.134: x-velocity at the center line behind the cylinder.

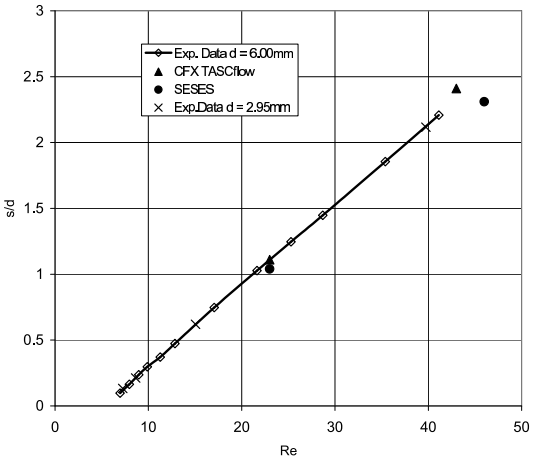

Figure 2.135: Bubble length as a function of Reynolds number.

the size of these recirculation regions, we plot arrows indicating the direction of the velocity. After zooming in, we notice two separation regions, as shown in Fig. 2.133. To determine their size, we use the velocity distribution on the straight line running through the center of the cylinder and pointing in the down stream direction. Behind the cylinder, we expect a zero velocity on the cylinder's surface and a negative velocity along the line to the point where the recirculation area ends. Further down stream, the velocity will steadily increase towards the value at the inlet. We therefore need the values of the velocity behind the cylinder on a straight line running through the center of the cylinder. To this end, we enter the statement

```
(* along channel center behind cylinder *)
Lattice lat1 Index i=0..dataPoints (0.600 0.350)+(0.150 0.0)*i/dataPoints
```

to define the location where the data is sampled. The straight line starts at $(600, 350)$ mm i.e. on the surface of the cylinder and stretches $150\,\mathrm{mm}$ in the x-direction. The parameter `dataPoints` specified earlier, defines the number of equally spaced data points on the straight line. To obtain the numerical data on the lattice points for plotting, we use a single `Write` statement

```
Write For i From 1 To 6
{
   File "data"@@{i}".txt" Lattice lat@i "%.9E %.9E %.9E %.9E %.9E %.9E\n"
    x y 2*Pressure/(Velocˆ2*Density.Val) Velocity norm(Velocity)
}
```

writing the x-, y-coordinates of the sampling point followed by the normalized pressure, density or a velocity component. In Fig. 2.134, the results of the *SESES* calculations for $\mathrm{Re} = 23, 45.7$ are compared with the results obtained from *CFX TASCflow* simulations. Notice the close agreement between these predictions. The length of the separation region is the distance between the end of the cylinder where the x-velocity is zero and the point where the graph of the x-velocity intersects the line of zero x-velocity. We compared our results with data from [2], where the length of the separation bubbles were determined in an experiment by observation. Fig. 2.135 shows the computed and the observed lengths of the bubbles, the agreement is very good for all computations.

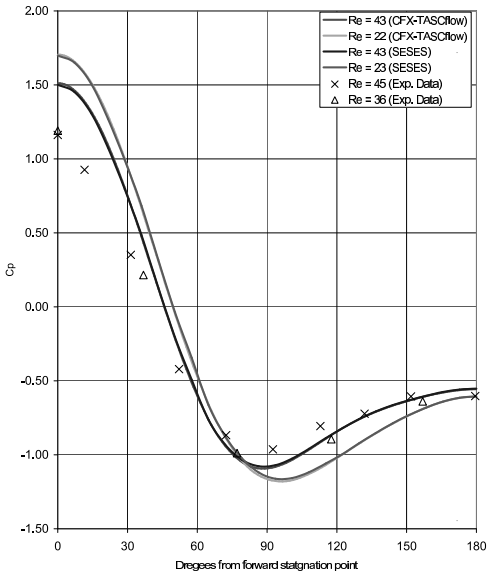

Figure 2.136: Pressure distribution on the surface of the cylinder.

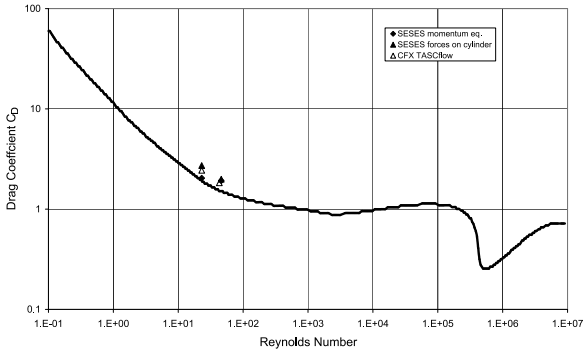

Figure 2.137: Drag coefficient as a function of Reynolds number.

As next, we compare the pressure distribution on the surface of the cylinder. As before, we first define a lattice of sampling points around the cylinder. The data points are located on a circle with the distance dr away from the cylinder surface. The first point is placed at the forward stagnation point of the cylinder and the last at the rear stagnation point of the cylinder. Plotted is the dimensionless pressure coefficient c_p defined as

$$c_p = \frac{2\,p}{\rho\,u^2}\,. \tag{2.161}$$

The computed pressure distributions based on *SESES* and *CFX TASCflow* are shown in Fig. 2.136. They are nearly identical and in fair agreement with the experimental data. The discrepancy is due the size of our computational domain in the lateral direction. In the experiment, the cylinder diameter is very small compared the channel width, whereas in the simulation the cylinder blocks a considerable part of the channel cross section. This leads to increased velocities around the cylinder and in turn to a higher stagnation pressure at 0° and to a lower pressure at 90° away from the forward stagnation point.

The force exerted on the cylinder is given by evaluating on the cylinder's surface, the integral

$$F_x = \int_S (\boldsymbol{\tau}_{\text{wall}} \cdot \mathbf{n} - p\mathbf{n}) \cdot \mathbf{e}_x \,\mathrm{d}A\,, \tag{2.162}$$

with p the pressure, $\boldsymbol{\tau}_{\text{wall}} = \mu(\nabla\mathbf{v} + (\nabla\mathbf{v})^{\mathrm{T}})$ the wall shear stress, $\mathbf{n}$ the unit vector normal to the cylinder and $\mathbf{e}_x$ the unit vector in $\mathbf{x}$-direction which coincides with the main stream direction. In order to compare the results from the computations and the experiments, the drag coefficient per unit length of the cylinder

$$c_d = \frac{2F_x}{\rho\,u^2\,d}\,, \tag{2.163}$$

is calculated and the results for both Reynolds numbers $\mathrm{Re} = 23$ and $\mathrm{Re} = 45$ are plotted in Fig. 2.137, together with results from *CFX TASCflow*. The coefficients of both simulation are somewhat bigger than the experimental data from literature. This is again due to the channeling effect of the lateral BCs, as explained above.

References

[1] M. M. ZDRAVKOVICH, *Flow around circular Cylinders Vol. 1: Fundamentals*, Oxford University Press, 1997.

[2] S. TANEDA, *Experimental Investigation of the Wakes behind Cylinders and Plates at Low Reynolds Numbers*, Journal of the Physical Society of Japan, Vol. 11, No. 3, 1956.

2.31 Developing pipe flow with heat transfer

In this example, we consider the heat transfer in a pipe as it occurs in a heat exchanger or a boiler. In the case of a household boiler hot flue gas from a gas or fuel oil burner passes through a number of vertical pipes. The water to be heated moves freely around the pipes. The entry temperature of the flue gas is around $1000\,^{\circ}\mathrm{C}$. The pressure of the gas is assumed to be equal to the ambient pressure. We assume the water temperature to be $30\,^{\circ}\mathrm{C}$. The goal of this tutorial is to calculate the hydrodynamic and the thermodynamic developing flow of the gas in the pipe.

As shown in Fig. 2.138, we consider a pipe with a circular cross section and a diameter of $d = 0.0513\,\mathrm{m}$ and a length of $L = 0.70\,\mathrm{m}$. The mass flow rate for a single pipe is $\dot{m} = 0.73{\times}10^{-3}\,\mathrm{kg/s}$. The properties of the flue gas are assumed to be those of air. In this case, with an absolute pressure of $1\,\mathrm{bar}$ at the pipe inlet, the density of the air is $\rho = 0.2737\,\mathrm{kg/m^3}$ which leads to an inlet velocity of $v = 1.29\,\mathrm{m/s}$. The fluid is entering the pipe with a top hat velocity profile. The velocity at the pipe surface is zero and a boundary layer is developing as the fluid is moving through the pipe. As the fluid passes through the pipe the boundary layer increases with increasing travel length. After a certain length, the boundary layer fills the whole pipe cross section. The entering gas has a uniform temperature of $T_E = 1000\,^{\circ}\mathrm{C}$. The wall temperature is constant $T_W = 30\,^{\circ}\mathrm{C}$. A thermal boundary layer starts to develop and convective heat transfer occurs. As the fluid flows through the pipe the physical properties of the fluid are changing according to the local temperature. Thus the density increases downstream of the entry cross section of the pipe, and as a consequence the mean velocity decreases. With the gas properties at the inlet, we calculate a Reynolds $\mathrm{Re} = d\,v/\nu = 92.4$ indicating a laminar flow regime. The hydrodynamic entry length may be obtained from an expression of the form (see [1])

$$(\frac{l_{e,h}}{d})_{\mathrm{lam}} \approx 0.05\,\mathrm{Re}\,, \tag{2.164}$$

This leads in our case to a hydrodynamic entry length of $x_{e,h} = 0.237\,\mathrm{m}$. The thermal

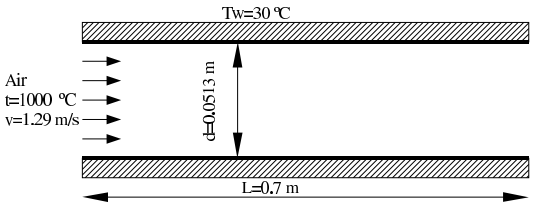

Figure 2.138: Sketch of the pipe.

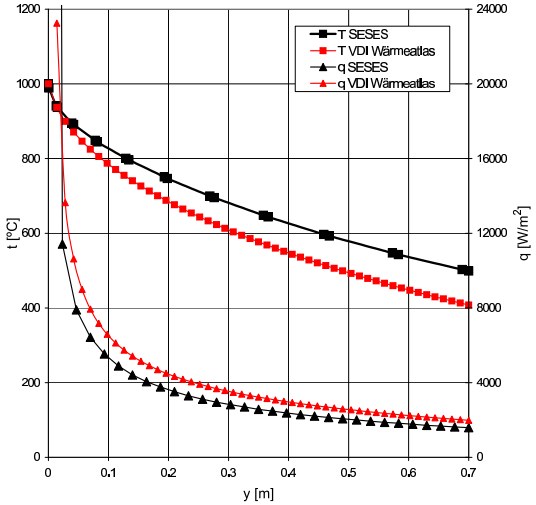

Figure 2.139: Temperature distribution and heat flux along pipe.

entry length may be expressed as (see [2])

$$(\frac{l_{e,t}}{d})_{\text{lam}} = 0.05\,\text{Re}\,\text{Pr}\,, \tag{2.165}$$

where Pr denotes the Prandtl number. Here the entry length is $l_{e,t} = 0.171\,\text{m}$. The Prandtl number is defined as

$$\text{Pr} = \frac{\nu\, c_p}{\lambda}\,, \tag{2.166}$$

here ν, c_p and λ denote the dynamic viscosity, the heat capacity and the thermal conductivity, respectively. The decrease of the mean temperature due convective heat transfer to the wall can be calculate using the equations given in the VDI Wärmeatlas [3]. The convection heat transfer coefficient can be computed using the Nusselt number

$$\text{Nu} = \frac{\alpha\, d}{\lambda}\,, \tag{2.167}$$

where α, λ and d denote the convection heat transfer coefficient, the thermal conductivity and the pipe diameter, respectively. According to [3], the Nusselt number is computed as

$$\text{Nu} = \sqrt[3]{\text{Nu}_1^3 + 0.7^3 + [\text{Nu}_2 - 0.7^3]^3 + \text{Nu}_3^3}\,, \tag{2.168}$$

with

$$\text{Nu}_1 = 3.66,\;\; \text{Nu}_2 = 1.615\,\sqrt[3]{Re\,\text{Pr}\frac{d}{x}},\;\; \text{Nu}_3 = (\frac{2}{1+22\text{Pr}})^{1/6}\sqrt{(\text{Re}\,\text{Pr}\frac{d}{x})}\,. \tag{2.169}$$

Based on the Nusselt number calculation and the computation of the convection heat transfer coefficient, the mean fluid temperature distribution from the inlet to a certain point downstream can be determined. The temperature distribution along the pipe axis is shown for our case in Fig. 2.139.

Setting up the problem

The *SESES* file for this example can be found at `example/HeatPipe.s2d`. In the first part of the input file, the geometry is defined where the mesh is denser close to the wall and at the inlet, as the velocity and the temperature gradients are large due to the developing boundary layer.

```
QMEI  nx  radius/nx
QMEJ  ny  length/ny
Routine double shift(double val,double v[2],double fac)
{
   double f=(val-v[0])/(v[1]-v[0]); return val+fac*(v[1]-v[0])*(-f*f+f);
}
Coord shift(x,0,radius,0.9) y     1
Coord x    shift(y,0,length,-0.9) 1
```

The fluid considered is air with the temperature dependent physical properties. We implement the temperature dependency by defining *SESES* routines which are taken from a library database

```
Routine double Viscosity_AIR[1](double T) {
(* Viscosity of air in  (pa s) - temperature range: 250k - 2273k *)
     return
     3.4746851634130747e-7 + T*(7.254574139642276e-8
     + T*(-4.474301400558843e-11 + (1.5696157920120257e-14
     - 1.2112463749591405e-18*T)*T));
}
Routine double Kappa_AIR[1](double T) {
(* Heat conductivity of air in (w/(m k)) - temperature range: 300k - 1500k *)
     return
     0.006292410832933179 + T*(0.00006858567183567118 +
     (-3.7488608369754205e-9 - 1.2382966512126441e-12*T)*T);
}
```

As we already noted, the problem at hand with reference to geometry and fluid flow is symmetric with respect to the pipe axis. We can make use of this property by including the statement

```
GlobalSpec
  Model      AxiSymmetric      Enable
```

together with the specification of the physical properties.

Numerical Results

We like to see how the hydrodynamic and the temperature boundary layer develop. For this purpose, we write the velocity and temperature profiles at various locations along the length of the pipe. The statement

```
Write File "section.txt"
  Lattice lat "%8.4f %8.4f %13.6f %13.6f\n" x y Velocity.Y Temp
```

writes the x- and y- coordinate values, the y-component of the velocity and the temperature to the file `section.txt` and the previous statement `Lattice` defines the points where the data is sampled. The profiles consist of eleven equally distributed points on a straight radial line. These data can be plotted with a suitable program. The velocity profiles at different locations along the pipe length are shown in Fig. 2.140.

The profiles show a constant velocity at the entrance of the pipe, and a boundary layer which has extended to about half the radius at $x = 0.1\,\mathrm{m}$. At $x = 0.2\,\mathrm{m}$ the boundary layer has almost penetrated the whole pipe cross section. This is in accordance with the estimation of the hydrodynamic entry length. The maximum velocity at the pipe center line is decreasing with travel length of the fluid. This is due to the temperature reduction which in turn leads to a density increase and in the end to smaller velocities. In Fig. 2.141, temperature profiles at various locations along the pipe length are shown. The development of the thermal boundary layer is also clearly shown. The development of the temperature profiles proof that the thermal entry length is correctly estimated.

If we like to compare our result of the numerical simulation with the calculation according to VDI Wärmeatlas, we must compute average temperatures. Here, we calculate a mass weighted average of the temperature as

$$< T >= \frac{\int \rho\, v\, T \mathrm{d}A}{\int \rho\, v \mathrm{d}A}\,, \tag{2.170}$$

which is formulated in the following way

```
T_a = integrate(Bound b_pos0a;Temp*Velocity.Y*x)/
      integrate(Bound b_pos0a;Velocity.Y*x)
```

We consider a slice and calculate the energy balance for this control volume. The diffusive heat transfer through the two cross sections and to the wall is computed. To do this at different locations along the pipe, we define a loop as shown below

```
For i From 1 To noData-1
{
   Define
   T_a     = integrate(Bound b_pos@{i}a;Temp*Velocity.Y*x)/
             integrate(Bound b_pos@{i}a;Velocity.Y*x)
   T_b     = integrate(Bound b_pos@{i}b;Temp*Velocity.Y*x)/
             integrate(Bound b_pos@{i}b;Velocity.Y*x)
   F_a = integrate(Bound b_pos@{i}a;(TempDiff.Y)*  PI*x)
   F_b = integrate(Bound b_pos@{i}b;(TempDiff.Y)*  PI*x)
   F_c = integrate(Bound b_pos@{i}c;(TempDiff.X)*2*PI*x)

   Lattice pos@{i}a node(nx,@{i}*dy)
   Lattice pos@{i}b node(nx,@{i}*dy+1)
   Write
     Lattice pos@{i}a "y=%6.3f Temp=%5.2f DFlux=%6.2f\n" y T_a F_a
     Lattice pos@{i}b "y=%6.3f Temp=%5.2f DFlux=%6.2f"   y T_b F_b
     Text " %6.2f\n" F_c
}
```

Fig. 2.139 compares the computed and estimated temperature distribution and one can see that these errors are very small. The graph also shows that the computed average temperature along the pipe is smaller than the estimated one. This means that the heat transfer to the wall is larger in the simulation. Several effects may cause these discrepancies. The Nusselt number calculation of the VDI Wärmeatlas is based on a large number of experiments. However, sensitivities studies has shown that the inlet conditions influence the temperature distribution as well as the fluid properties. In our estimation, we used fluid properties at the mean temperature of the inlet and outlet of the pipe. We suspect that these effects then lead to the different temperature distributions. In Fig. 2.139, the convection heat transfer coefficient is also shown. As expected the coefficient is very high close to the inlet and drops to a value of $\alpha \simeq 3\,\mathrm{W/m^2}$ downstream.

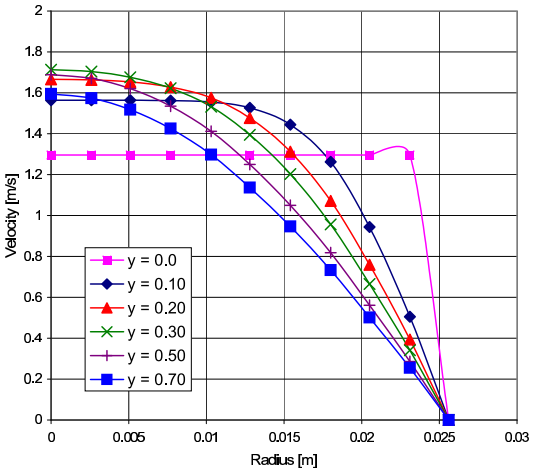

Figure 2.140: Velocity profiles across pipe cross sections at various locations.

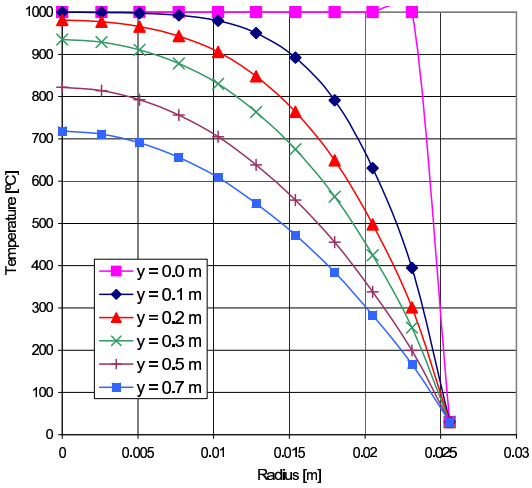

Figure 2.141: Temperature profiles across pipe cross sections at various locations.

References

[1] H.L. LANGHAAR, *J. Appl. Mech., Vol. 64, A-55*, 1942.

[2] W.M. KAYS, M.E. CRAWFORD, *Convective Heat an Mass Transfer*, McGraw-Hill, 1980.

[3] WÄRMEATLAS, *VDI Düsseldorf*, 1980.

2.32 Hot Spots in a Tubular Reactor

In this example, we show how *SESES* can be coupled with other simulation software. As illustrated in Fig. 2.142, we consider a developing flow in a chemical reactor. The open channel of the reactor has a circular cross section. Air at ambient temperature enters the channel with a homogeneous velocity of $0.04\,\mathrm{m/s}$. A porous shell is placed around the open channel with an annulus cross-section. A mixture of hydrogen and water vapor enters the shell from the face surface and diffuses through the porous media. Oxygen from the air penetrates the porous material and reacts with the hydrogen according to

$$2\,H_2 + O_2 \rightarrow 2\,H_2O. \tag{2.171}$$

The air flow is simulated based on the CFD-tool CFX-5 [1]. The resulting velocities from this computation are written on a regular grid to files which are then imported into *SESES* and used in combination with the diffusion, reaction and heat-transfer calculations performed in *SESES*. A one-way coupling is considered here, i. e. no information of the *SESES*-computations are fed back to CFX-5. Use is made of the axial symmetry to set up the geometry of the reactor. The Reynolds number of the free flow in the open channel is small hence the flow regime is laminar. Due to the no-slip condition at the interface of open channel and porous region, the velocity at the interface

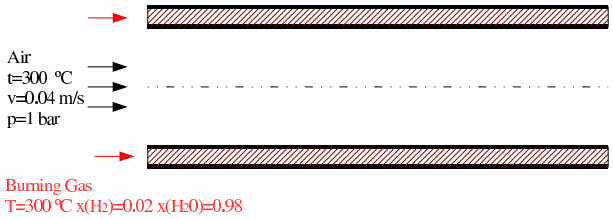

Figure 2.142: Sketch of the single channel of a catalyst converter.

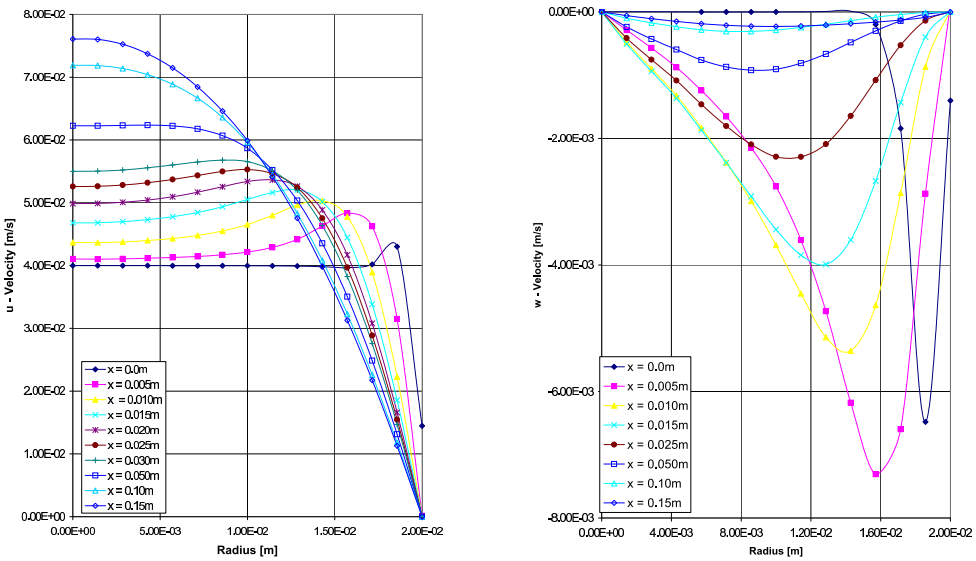

Figure 2.143: Velocity-profiles in the (left) main and (right) radial flow-directions at locations along the open channel.

is zero. A boundary layer starts to develop. The development of the boundary layer can be observed in Fig. 2.143a, where the velocity-component in the main flow direction is shown. The cross sectional area where the velocity is constant becomes smaller at each position in the downstream direction. The transition from a plug flow to a parabolic velocity profile causes the fluid to move also towards the channel axis. This secondary flow is shown in Fig. 2.143b.

Model Specification

Within the initial section of the input file `example/Reactor.s2d`, we first specify the the four different species associated with the burning of hydrogen in air (2.171). The species are defined with the help a macro list, which is useful later on for automatically generating input statements.

```
Macro SPEC_LIST(M,A,B) { @M(H2,@A,@B)@M(H2O,@A,@B)@M(O2,@A,@B)@M(N2,@A,@B) }
Macro DEF_SPEC(X,A,B)  { @X  }
Species @SPEC_LIST(@DEF_SPEC,,)
```

Next, all geometry parameters are defined and an initial mesh is specified with the `QMEI` and `QMEJ` statements. To ensure a fine mesh between the regions of free and porous flow, the homotopic function `shift(...)` is used to decrease the mesh-size at this boundary. Within the next section, the *SESES*-routines

```
Routine double u_veloc(double x, double y) FromData "u_veloc.txt"
Routine double w_veloc(double x, double y) FromData "w_veloc.txt"
```

are defined to specify the velocities within the free flow-region. Instead of giving their functional forms, the velocity data is read from external files using the `FromData` statement. In this example, the data has been generated from CFX-5 simulations of a developing laminar flow. The so-defined routines `u_veloc` and `w_veloc` linearly interpolate between the given discrete data points. In the next section, various operational parameters are defined: reference temperature and pressure as well as the mole-fractions of all species at the air and burning gas inlets, respectively. The next section defines global and material parameters. As first we define the production rates for the burning-reaction of hydrogen $H_2 + O_2 \rightarrow 2H_2O$ with the help of several macros following a common scheme used to define chemical reactions. This model relies on a phenomenological kinetics formula for the species reaction rates Π_α

$$\Pi_\alpha = kM_\alpha\nu_\alpha \frac{P + P_{\text{amb}}}{P_{\text{ref}}} \left[x_{\text{H}_2}^2 x_{\text{O}_2} - x_{\text{H}_2\text{O}}^2 \exp(\frac{\Delta h - T\,\Delta S}{RT}) \right] ,$$

with k the reaction rate, ν_α the stoichiometric coefficient for α, M_α the species molar mass, Δh the reaction enthalpy, ΔS the reaction entropy, P_{ref} a reference pressure parameter and R the universal gas constant. Next, the global definitions

```
GlobalSpec
  Model      AxiSymmetric Enable
  Parameter  AmbientTemp T0 K
```

are used to employ the rotational-symmetric model-description as well as the ambient temperature. As next, we define the material parameters and models specifying the physical-chemical model of the free flow-region `m_void`.

```
MaterialSpec m_void
  Equation   CompressibleFlow ThermalEnergy
             TransportH2 TransportH2O TransportO2 TransportN2  Enable
  Parameter  Density    IdealGas(AmbientPress p0 Pa) SIunit
Macro Renew  BURNH2_RATE { (1.0e+3)  (* mol/(s*m**3) *) }
Macro Renew  BURNH2_DH   { (-2.48e5) (* J/mol *)        }
Macro Renew  BURNH2_DS   { (-55.0)   (* J/(K*mol) *)    }
  Parameter  RREA  @BURNH2(RREA)
```

Here, the `Equation` statement activates the balance equation `NavierStokes` for the overall mass and momentum, the mass-balances `H2 H2O O2 N2` for each species as well as the energy balance `Temp`. Within the `Parameter` section, we define the ideal gas density law and the producion rates for the hydrogen burning reaction by calling the macro `@BURNH2`. By newly defining macros before the macro call of `@BURNH2`, we can redefine default parameters used by the macro ifself. The large negative enthalpy-change indicates that the burning of hydrogen is strongly exothermic, i. e. it produces a large amount of heat. The other `Parameter` statements define the following properties: the overall thermal conductivity `KappaIso`, the viscosity `Viscosity` of the gas mixture, the molecular weights `Mmol<A>`, the heat capacities `Cp<A>` and the pair-wise diffusion coefficients `Diff<A>_<B>` of all species. Note that the properties of the porous flow-region `m_reac` are very similar to those of `m_void`. To transfer the properties of `m_void` to `m_reac`, we use the statement `m_reac From m_void`. In addition, the material `m_reac` contains the parameter-setting

```
MaterialSpec m_reac From m_void
  Parameter  KappaIso       0.10  W/(m*K)
```

to adjust the thermal conductivity to its value within the porous material. Finally, the defined models are assigned to their computational domains using the `Material` statement and boundary conditions are set. Here, we give the compositions and temperatures at the air and burning gas inlets. With this, the *SESES*-model of the reactor is complete.

What remains is to control and start the actual simulation and to specify the kind of output generated. Both is done within the command section of the input file. We start with the initialization of the various field-variables using the `Init` statement. As already mentioned, the velocity-fields within the free flow-region are imported from CFX-5 simulations. We initialize those fields with the above defined functions `w_veloc(x,y)` and `u_veloc(x,y)`. The temperature and pressure are set to their reference values and the compositions on the regions of free and porous flow are set to their values at the inlets. In *SESES*, this is accomplished using the function `material(<mat>)` which is 1 on `<mat>` and 0 elsewhere. We then use the `Remesh` statement to locally refine the mesh at the boundary between the regions of free and porous flow. Once all initialization-parameters are set, `Solve Init` performs the actual initialization. The next statements

```
BlockStruct Block H2 H2O O2 N2 Temp
  Convergence MaxIncr.H2<1e-7 && MaxIncr.O2<1e-7 && MaxIncr.Temp<1e-4
  Increment   Standard #2, ReuseFactoriz #10, Standard Control -1

Solve Stationary
```

defines the fields to be solved for, sets convergence criteria, an optimized solution strategy and evokes the actual calculation.

Numerical Results and Discussion

Fig. 2.144 shows the axial and radial velocity-ditributions within the reactor as predicted from the CFX-5 simulations. At the inlet of the free flow-region, we have a

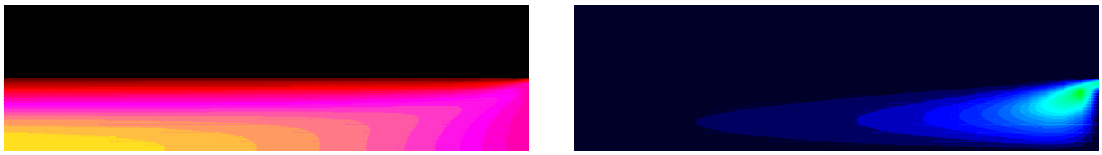

Figure 2.144: Velocity-distributions in axial (a) and radial (b) directions along the reactor; the lighter the color, the higher the velocity.

uniform axial velocity of $0.040\,\mathrm{m/s}$. Due to the no-flow condition at the boundary between the free and porous regions, the fluid slows down near that boundary. Consequently, the continuity of mass leads to an increase of the velocity near the center, with a maximum velocity of $0.075\,\mathrm{m/s}$ at the outlet.

Since the considered burning of hydrogen is strongly exothermic, a substantial generation of heat occurs. This leads to a temperature increase within the reactor. Within

the prorous region, the heat transfer is rather slow since it only happens by conduction and diffusion. Consquently, the thermal conductivity value of the porous material has a large impact on the observed temperature distribution. This is illustrated in Fig. 2.145, where the reactor-temperatures are shown for for three different thermal conductivities of the porous material. The lowest temperature of $300\,\mathrm{K}$ is at the inlet (shown in dark blue). As expected, the small thermal conductivity of $\kappa = 0.1\,\mathrm{W/(m\,K)}$ leads to a *hot spot* within the porous region, with a peak temperature of about $490\,\mathrm{K}$ (shown as the light yellow region). As the thermal conductivity increases, the temperature-distribution within the reactor becomes more uniform. The concentration-distributions of O_2, N_2, H_2, and H_2O along the reactor are shown

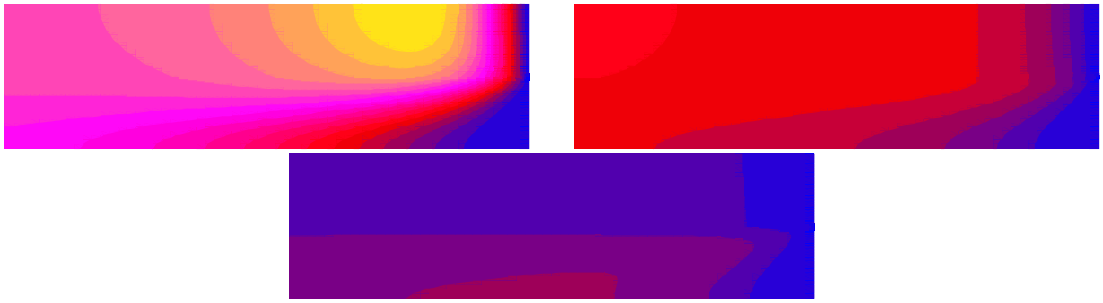

Figure 2.145: Temperature-distribution along the reactor for three different thermal conductivities of the porous material: (a) $\kappa = 0.1$, (b) $\kappa = 1.0$ and (c) $\kappa = 5.0\,\mathrm{W/(m\,K)}$.

in Fig. 2.146. Note that the distributions of oxygen and nitrogen look very similar. This can be explained by the very small hydrogen-concentration, i. e. for hydrogen burning, only small amount of oxygen are required. Due to this excess in oxygen, the distributions of O_2 and N_2 are primarily influenced by the flow conditions and the gas-diffusivities. However, the distribution of H_2 is different in that the H_2-conversion – which happens mainly at the boundary between the free and porous regions – creates a strong H_2-flux in the radial direction.

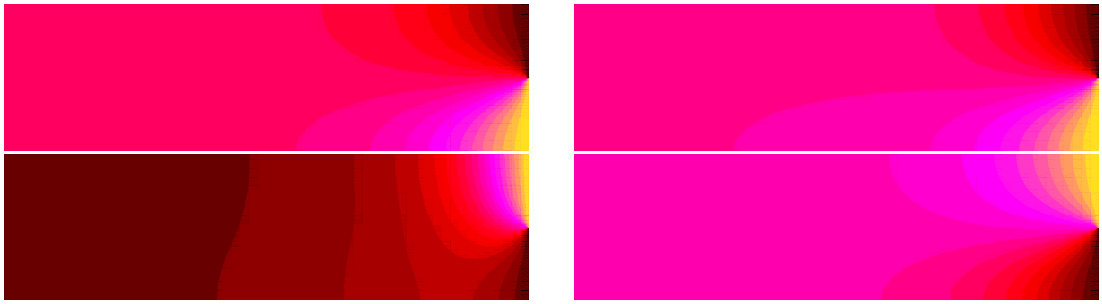

Figure 2.146: Concentration-distributions along the reactor for: (a) O_2, (b) N_2, (c) H_2, and (d) H_2O.

References

[1] CFX-5, CFD-Tool, ANSYS Inc., Canonsburg PA (USA), http://www-waterloo.ansys.com/cfx/.

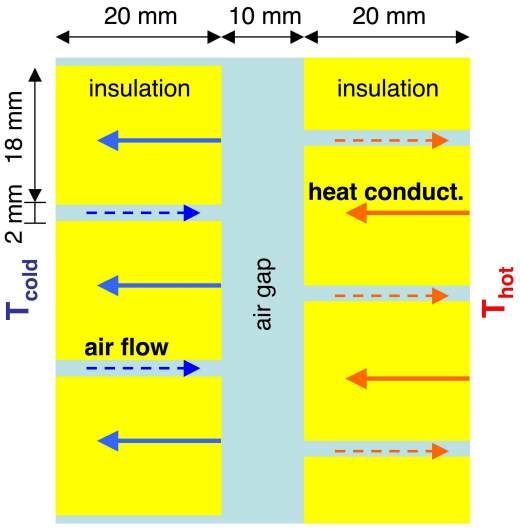

Figure 2.147: Section of the thermal insulation structure consisting of two insulation walls with air channels and an air gap between the walls. Not shown is an offset of 18 mm in the depth location of the air channels between the two walls.

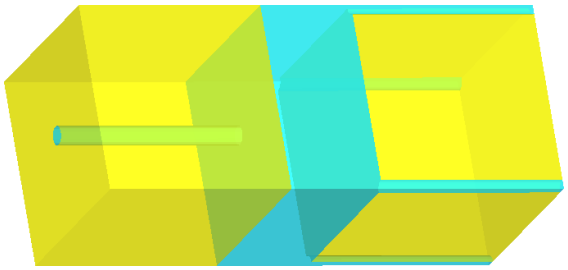

Figure 2.148: Computational domain with air channels `AIR` (light blue) and insulator `ISO` (yellow) representing the smallest repetitive structure of our thermal insulation structure.

2.33 Effective Transport Parameters from Volume Averaging

In many complex technical devices such as fuel cells, a multitude of coupled physical and chemical processes take place within the assembly: fluid flow, diffusion, charge and heat transport, as well as electro-chemical reactions. For design and optimization purposes, direct numerical simulation of the full 3D structure using CFD tools is often not feasible due to the large range of length scales that are associated with the various physical and chemical phenomena. However, since many fuel cell components such as gas ducts, current collectors or thermal insulation assemblies are made of repetitive structures, volume averaging techniques can be employed to replace details of the original structure by their averaged counterparts [1, 2, 3]. An efficient approach is based on the following two-step procedure: first, for all repetitive structures, detailed 3D FE-simulations are performed to obtain effective parameters for the transport equations. The complex structural information is thereby cast into effective material properties. In a second step, these averaged quantities are used to simulate the original 3D structure but now without all fine details, thereby decreasing the computational cost.

In this tutorial, the volume averaging method will be illustrated for the thermal insulation structure shown in Fig. 2.147. It consists of two insulation walls with air channels and an air gap between the walls. An external pressure gradient causes an air-flow from the left to the right. In addition, there is an external temperature gradient causing a heat conduction flow in the opposite direction. In the most simple case, i. e. when the inertia-terms in the momentum balance are negligible, the internal resistance of a

solid structure with respect to fluid flow is described by Darcy's law [4],

$$\nabla p = -\frac{\mu}{k}\langle \mathbf{w} \rangle \,. \tag{2.172}$$

Darcy's law states that the average fluid velocity in a given direction is proportional to the external pressure gradient that causes the fluid motion. The proportionality constant is given by the ratio of the shear viscosity μ to the permeability k. The shear viscosity is a property of the considered fluid, whereas the permeability takes into account geometrical details of the solid structure through which fluid flow happens. To characterize the internal flow resistance of the thermal insulation assembly shown in Fig. 2.147, we solve (2.172) for the permeability

$$k_z = -\frac{\mu\, L_z}{\Delta p}\langle w_z \rangle \,. \tag{2.173}$$

Here we have assumed that the main flow goes along the z-direction. Consequently, k_z can be obtained from a simple simulation, where for a given external pressure drop Δp, the resulting average fluid velocity $\langle w_z \rangle$ is determined. Using the integral version of the continuity equation for steady-state flow $\dot{m}_z = A\,\rho\,\langle w_z \rangle$ with $\dot{m}_z$ the total mass flux, eq. (2.173) can be re-written as

$$k_z = -\frac{\mu\, L_z}{\Delta p}\frac{\dot{m}_z}{A\,\rho} \,. \tag{2.174}$$

In this example, we will then show how to compute k_z for the considered thermal insulation assembly. With this information we are then able to approximate the flow characteristics of the original structure by performing a simple simulation of an unstructured *porous* material.

Similarly to the porous flow, the overall characteristics of an assembly with respect to heat conduction can also be described by effective thermal conductivities. They are obtained from Fourier's law [4], $\nabla T = -(1/\kappa)\mathbf{F}$ stating that the heat flux $\mathbf{F}$ is proportional to the external temperature gradient ∇T. The proportionality constant is the inverse of the thermal conductivity κ and solving for κ gives

$$\kappa_z = -\frac{q_z\, L_z}{\Delta T} \,. \tag{2.175}$$

As for the permeability, we will then show how to compute κ_z.

Model Specification

This numerical example can be found at `example/VolAverage.s3d`. To run the calculations in different modes, some control parameters are introduced first. The the flag `permYN` is set, the effective permeability of the considered structure is calculated applying an external pressure drop of `dpress=12.0`. Similarly, when the flag `condYN` is set, the effective thermal conductivity of the considered insulation structure is calculated applying the external temperature gradient that results from the difference between `T_Hot=600` and `T_Cold=350`.

```
Define
permYN = 1                              (* permeability for isothermal system *)
condYN = 0                          (* thermal conductivity for stagnant system *)
dpress = 12.0                                (* given pressure drop in Pa *)
T_Cold = 350                                     (* cold temperature in K *)
T_Hot  = 600                                      (* hot temperature in K *)
T_ref  = T_Cold+(T_Hot-T_Cold)/2              (* reference temperature in K *)
```

As next, we construct a three-dimensional cube as the computational domain representing the smallest repetitive part of the structure shown in Fig. 2.147. To generate the tubular air channels, we apply the homotopic function `cylzS` which transforms a cube into a cylinder. Two materials representing air `AIR` and the insulation material `ISO` are then defined and assigned to the computational domain, see Fig. 2.148.

We next specify the physical parameters. First, a number of *SESES* routines are defined which provide the thermal conductivity of the insulation walls as well as the required fluid properties of air. Note that for material `AIR`, the mass-, momentum- and energy-balance need to be solved, whereas for `ISO`, only the energy-balance is required.

```
(* average heat capacity of air for conduction *)
Define Cp_AIR_ref = (CpITemp_AIR(T_Hot)-CpITemp_AIR(T_Cold))/(T_Hot-T_Cold)
(* global specifications *)
GlobalSpec
  Parameter  AmbientTemp  T_0 = 293.15 K
(* air channels *)
MaterialSpec AIR
  Equation   CompressibleFlow ThermalEnergy Enable
  Parameter  FlowStab        zero
  Parameter  Density         IdealGas(AmbientPress 1e5 Pa)  SIunit
  Parameter  Mmol            M_AIR()                  kg/mol
  Parameter  Viscosity       Viscosity_AIR(Temp)      Pa*s
  Parameter  PressPenalty    1e-5                     s
  Parameter  KappaIso D_Temp Kappa_AIR(Temp)
                             KappaDTemp_AIR(Temp)     W/(K*m)-W/(m*K**2)
  Parameter  ThermStab       zero
  Parameter  ThermConv       Velocity*Cp_AIR_ref*Density.Val W/(K*m**2)
(* insulation material *)
MaterialSpec ISO
  Equation   ThermalEnergy Enable
  Parameter  Density.Val     1                            SIunit
  Parameter  KappaIso D_Temp Kappa_CER_DURATEC(Temp)
                             KappaDTemp_CER_DURATEC(Temp) W/(K*m)-W/(m*K**2)
```

Before we can start our simulation, a complete set of boundary conditions must be defined. Depending on the mode in which we run the simulation – as specified through the flags `permYN` and `condYN` – different sets of boundary conditions are used. The boundary conditions `BC_fluid1` and `BC_fluid2` specify the inlets and outlet of the air-flow, respectively. In addition, when thermal simulations are performed (`condYN=1`), the temperature boundary conditions `BC_wall1` and `BC_wall2` are set at the outer surfaces of the insulation material. No-slip boundary conditions have to be applied to all fluid-solid interfaces. These surfaces are elegantly determined using the `OnChange material(...)` statement, which specifies all surfaces of a given material (in this case `ISO`).

```
  BC    BC_noFlow   OnChange material(ISO)     (* no flow at channel walls *)
        Dirichlet Velocity   0,0,0 m/s
```

Finally, we give symmetry conditions with respect to the flow field in those air-channels, where only $1/4$ of total the cross-section is considered. With the specification of the boundary conditions the initial section of the input file is now complete.

In the command section of the input file, we first perform a manual refinement of the air channels, initialize the temperature field and run the solution procedure depending on the setting of the flags `condYN` and `permYN`. At the end, we add post-processing commands to write the desired information allowing us to compute averaged values of permeability and conductivity based on (2.174) and (2.175).

Numerical Results

The content shown below is an excerpt of the output when running the example in mode `permYN=1`.

```
 (* input parameters *)
 T_0                 =   293.15 (K)
 p_0                 =     1.000e+05 (Pa)
 dpress              =     1.200e+01 (Pa)

(* properties of air *)
 viscosity(293.15 K) =     0.0185 (cp)

 (* calculated densities *)
 densiFluid1         =     0.7306 (kg/m^3)

 (* calculated mass fluxes *)
 mfluxFluid1         =    -2.997e-06 (kg/s)

 (* calculated effective permeability *)
 permEff             =     1.111e-09 (m^2)
```

Based on (2.174), the permeability is calculated as

```
  permEff     = -1e3*Viscosity_AIR(T_ref)*mfluxFluid1
                   *lengthZ/(areaXY*densiFluid1*dpress)
```

with `mfluxFluid1` the computed total mass flux. The small value $k_z = 1.111e{\times}10^{-09}$ m^2 means that the insulation assembly exhibits a small flow-resistance. A similar informative output is generated when running in mode `condYN=1` to compute the average thermal conductivity κ_z. Based on (2.175), the conductivity follows as

```
  kappaEff    = 1e3*lengthZ/(T_Hot-T_Cold)*qFluxWall1/areaXY
```

with `qFluxWall1` the computed total heat flux. As expected, the value of $\kappa_z = 0.1448\,\mathrm{W/(K\,m)}$ lies in between those for air and the insulation material.

References

[1] M. ROOS, E. BATAWI, U. HARNISCH, T. HOCKER, *Efficient simulation of fuel cell stacks with the volume averaging method*, Journal of Power Sources, Vol. 118, pp. 86-96, 2003.

[2] ST. WHITAKER, *The Method of Volume Averaging*, Kluver, 1999.

[3] H. BRENNER, D. A. EDWARD, *Macro Transport Processes*, Butterworth-Heinemann, 1993.

[4] R. B. BIRD, W. E. STEWART, E. N. LIGHTFOOT, *Transport Phenomena*, Wiley, 1960.

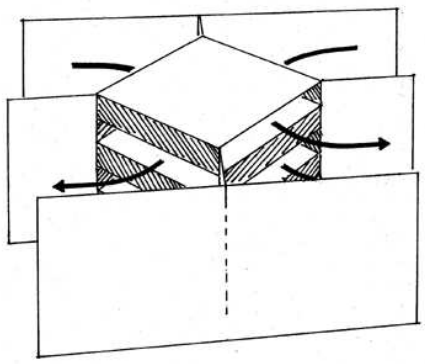

Figure 2.149: View of the heat exchanger. Visible are two crossed channels which carry a hot and a cold air flow (arrows).

2.34 Heat Exchange between Air Flows

Air temperature and humidity are key parameters for comfort as well as for many fabrication processes. The heat exchange between cold and hot air flow plays an important role in temperature control. Fig. 2.149 displays a drawing of a heat exchanger, which is commercially available. These heat exchangers consist of hundreds of crossed channels for the hot and cold air flows, separated by thin metal sheets. The channels lay horizontally and have heights between 3 and $15\,\mathrm{mm}$. The efficiency of the exchanger is expressed with the averaged ratio of hot and cold flows at the in- and outlet

$$\phi = \frac{T_{\mathrm{cold,out}} - T_{\mathrm{cold,in}}}{T_{\mathrm{hot,in}} - T_{\mathrm{cold,in}}}\,,$$

where values of $\phi = 0.5$ are considered as good. For ISO certification, the efficiency of heat exchangers needs to be measured. Efficiency predictions by model calculations are difficult due to the complex and irregular channel shapes and turbulent air flows. Air conditioning engineers, however, suggest, that the presence of turbulence plays a minor role for the value of ϕ. It seem to be the channel structure and dimensions, that govern the ϕ value. When modeling just two channels, it is possible to calculate ϕ as a function of channel thickness and compare the values to measurements.

Implementation of the Model

In order to simply the matter, we have modeled two air channels only with a given rectangular flow profile. Implementing a more realistic parabolic flow profile involves either coupled velocity calculations of many air slabs or a complicated parameterization of the flow profile. The numerical example can be found at `example/HeatExchanger.s3d`, it defines a cold channel in the center and half a channel of hot air above and below the cold channel, see the left of Fig. 2.150. In the `Geometry definition` section, an initial rectangular geometry is transformed to a rotated rhombus with the following statements

```
Include "Homotopic.sfc"
```

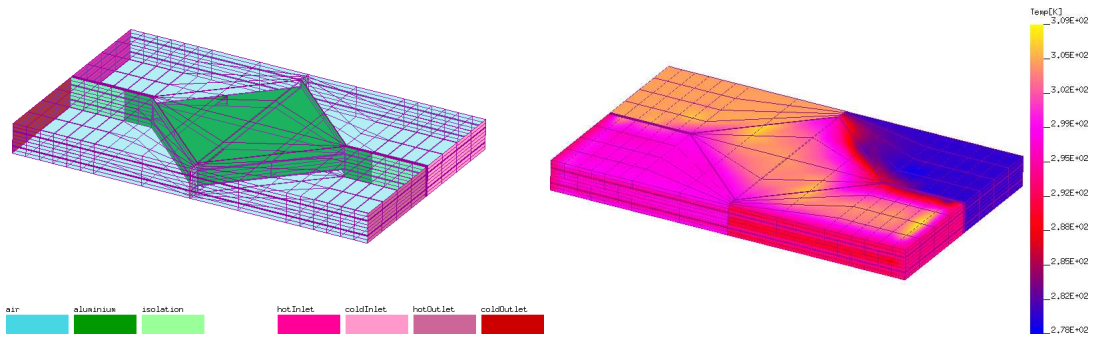

Figure 2.150: Left: Computational domain with two air channels. Right: temperature profile

```
Coord coord+(0,-x3,0)*bump(x,ax2,ax3,ax4,ax5,1)
                     *bump(y,ay0,ay1,ay2,ay3,1)
      block(nx0 nx7 ny0 ny4 nz0 nz5)

Coord coord+(0,x3,0)*bump(x,ax2,ax3,ax4,ax5,1)
                    *bump(y,ay3,ay4,ay5,ay6,1)
      block(nx0 nx7 ny3 ny6 nz0 nz5)
```

In the input file, it is further important to define several blocks and domains in order to define the velocity profile later on.

In the `Material definition` section, we define a user field `velocity` to store the velocity profile in the flow channels. The velocity is just initialized later on and not directly computed. Therefore, only the temperature equation for conduction and convection of heat need to be solved. For this heat exchanger model, the Péclet mesh number is large and of the order $\approx 10^3$ so that one needs stabilized finite elements in order to compute numerical solutions. Here streamline diffusion is automatically applied when defining the thermal convection with the material parameter `ThermConv`. Without stabilization, in order to obtain smooth solutions a factor 10^3 larger artificial diffusion would be otherwise required.

```
ElmtFieldDef velocity(DofField Temp)[3]

MaterialSpec air
  Equation   ThermalEnergy
  Parameter  Viscosity  0.0000186   Pa*s
  Parameter  KappaIso   0.0262 W/(K*m)
  (* cp=29/0.029 J/(K*kg) density=1.16 kg/m**3 *)
  Parameter ThermConv  velocity*29/0.029*1.16 W/(m**2*K)
```

In the `BC definition` section, we define boundary conditions. The hot air flow is set to a constant temperature of $305\,\mathrm{K}$ at the inlet and to a constant but yet unknown temperature at the outlet. A constant temperature profile at the boundary is attached to the cold air flow as well. The temperature at the inlet is set to $280\,\mathrm{K}$.

Results and Discussion

The right of Fig. 2.150 shows the calculated temperature distribution for a constant air velocity of $2.5\,\mathrm{m/s}$, and for incoming air temperatures of $305\,\mathrm{K}$ and $280\,\mathrm{K}$, respectively. The temperature change of the air flow at the end of each channel yields the following

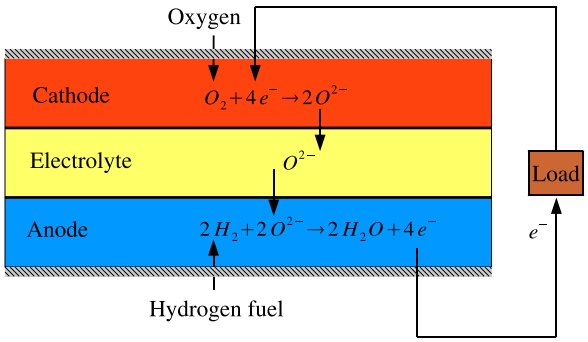

Figure 2.151: A schematic view of a SOFC and involved chemical reactions.

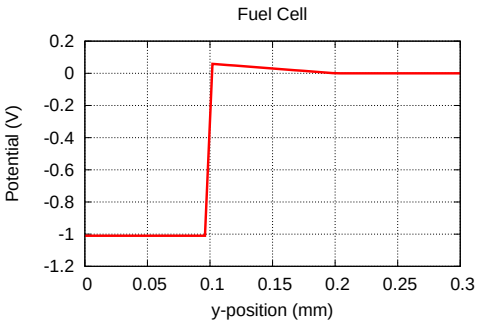

Figure 2.152: Potential distribution along the vertical.

efficiency

$$\phi = \frac{T_{\text{cold,out}} - T_{\text{cold,in}}}{T_{\text{hot,in}} - T_{\text{cold,in}}} = \frac{292K - 280K}{305K - 280K} = 0.48\,.$$

According to our definition at the beginning of this example, this value corresponds to a good heat exchange efficiency.

2.35 A first model of a SOFC fuel cell

In this example, we are going to model a simple solid oxide fuel cell. SOFCs are solid state devices working at high temperature of $800 - 1100\,^{\circ}\text{C}$, not requiring complex catalytic reactions at the electrodes to run the electrochemical fuel oxidation which may be hydrogen or carbon monoxide. In their simple form, they consist of a cathode and an anode made of porous materials and in between a solid porous electrolyte, see [1, 2]. Fuel cell modeling is in general a complex task involving chemical reactions, current flow, mass flow and energy conservation. In this example, we consider the combustion of hydrogen and focus on the chemistry of a 1D system at constant temperature leaving out, for other examples, topics like convective flow, fuel delivery, fuel recombination, temperature distribution and fuel cell optimization.

Our fuel cell system is depicted in Fig. 2.151, the hydrogen fuel is delivered at the anode and oxygen is delivered at the cathode by an external pipe system not show in the figure. Both gases diffuse towards the electrolyte and at its boundary, we have two heterogeneous reactions taking place. At the interface between cathode and electrolyte we have the reduction of oxygen, where electrons from the external circuitry combine with the oxygen. The oxygen ions diffuse then towards the anode-electrolyte interface where the oxidation of hydrogen takes place delivering electrons to the external circuitry. The stoichiometric coefficients of both reactions are given by

$$\begin{array}{lrcl} \text{Oxidation of } H_2: & 2H_2 + 2O^{2-} & \rightarrow & 2H_2O + 4e^-\,, \\ \text{Reduction of } O_2: & O_2 + 4e^- & \rightarrow & 2O^{2-}\,. \end{array} \tag{2.176}$$

By closing the external circuitry a current starts flowing, because at the anode-electrolyte interface, the oxidation of hydrogen pumps carriers uphill resulting in a discontinuous potential of ca. $1 - 1.2\,\text{V}$. This discontinuity, called the Nernst potential, is an

expression of the difference in the free Gibbs energy between educts and products of the heterogeneous reaction. It is an optimization task to minimize the electrical lost within the fuel cell and have most of the Nernst potential available to drive the external load. The rates of the reduction and oxidation reactions (2.176) are the rates of production and consumption of electrons at both interfaces and therefore proportional to the normal current flow at the interface and given for the stoichiometric coefficients of (2.176) and a unit of moles per surface and time by

$$\text{rate} = \frac{\mathbf{j} \cdot \mathbf{n}}{4\, q N_{\text{avo}}}\,, \tag{2.177}$$

with $\mathbf{j} \cdot \mathbf{n}$ the electrical current normal to the interface anode-electrolyte, q the elementary charge, N_{avo} the Avogadro's constant and the factor 4 stemming from the stoichiometric coefficients of e^- in (2.176). The Nernst potential is actually the open circuit voltage since as soon as a current flows, irreversible processes reduce this voltage barrier. In general this voltage drop also called *overpotential* is a complex function of the normal current, but for SOFC under normal working conditions, a linear dependency is a good approximation and for the cell voltage we have

$$\Delta\Psi = \Delta\Psi_{\text{Nernst}} - \frac{\mathbf{j} \cdot \mathbf{n}}{G}\,, \tag{2.178}$$

with G a proportionality constant called the effective surface conductance.

Setting up the problem

In the previous section we have given all necessary details for the working of a SOFC and we are now ready to set up a *SESES* example `example/FuelCell.s2d` corresponding to our description. Although we are going to make a 2D simulation, our model will be laterally invariant so that we are actually performing a 1D simulation embedded in a 2D model. For this first example, we are going to model just a small neighborhood of the electrolyte. Because of the small dimensions, we can consider the temperature to be constant and the mass flow of the species to be diffusive since there is no external pressure gradient driving a convective flow. As a further simplification, we are not going to model the diffusion of oxygen ions O^{2-} in the electrolyte since here the electrical drift current is dominant.

In summary, we have the following scenario. Our fuel cell consists of three materials, the cathode, the electrolyte and the anode stacked above of each other. Each layer has a thickness of $0.1\,\text{mm}$ and the width and depth of the cell is $1\,\text{mm}$. Within the cathode, we have the diffusion of oxygen so that we need to solve for the oxygen mass flow there. At the top of the cathode, we supply oxygen and since we suppose there is no shortage, we can assume a constant oxygen concentration corresponding to a Dirichlet BCs. The oxygen diffuses towards the interface with the electrolyte, where the oxygen's reduction acts as a sink with a rate given by (2.177) and modeled by a Neumann BCs depending on the normal electrical current. At the anode we have a similar situation for the hydrogen, at the bottom of the anode we prescribe a fixed hydrogen concentration with a Dirichlet BC and at the interface with the electrolyte we have a sink of hydrogen modeled by a Neumann BCs with a rate of (2.177). At the

anode-electrolyte interface, we have additionally the production of water steam, but we assume the steam can freely escape through the anode and therefore does not need to be considered. Over the whole device, we solve for the electric potential and set the top of the cathode at ground. By the generated electrical field, electrons drift towards the electrolyte. Here the potential undergoes a first jump due to the Nernst potential of the oxygen reduction, however this value is small ca. $0.1\,\mathrm{V}$ and it is not considered here at this point. At the interface electrons are replaced by negative oxygen ions drifting and diffusing towards the anode. As noted before, in the electrolyte the ions diffusion can be neglected so that we just solve for the drift current like within the cathode and anode, just with another electric conductivity since now we have ions instead of electrons. When the ions arrive at the interface with the anode, the potential undergoes another negative jumps due to the hydrogen oxidation which is modeled following (2.178) by a jump BC depending on the normal electrical current. Although the Nernst potential $\Delta\Psi_{\mathrm{Nernst}}$ is a function of the educt and product concentrations, we neglect this small non-linear dependency and use a constant value of $\Delta\Psi_{\mathrm{Nernst}} = 1.1\,\mathrm{V}$ so that we can also include here the neglected jump at the cathode-electrolyte interface. At this interface, the ions are replaced again by electrons drifting towards the anode contact and here we have two choices, either defining the contact's voltage with a Dirichlet BC for the electric potential, or the total current through the device with a floating BC. We prefer the latter and set a value of $3\,\mathrm{mA/mm^2}$, since a zero current corresponds to a turned-off system, whereas for a full fledged non-linear model it is not always clear which voltage corresponds to this state. It may also be useful to connect both contacts with a resistive load, by defining the total current to be a linear function of the floating potential. In summary, the materials are defined with the statements

```
MaterialSpec Electrolyte
  Equation  OhmicCurrent Enable
  Parameter SigmaIso 5 S/m

MaterialSpec Cathode From Electrolyte
  Equation  TransportO2  Enable
  Parameter SigmaIso   1.2E4 S/m
  Parameter Diff.O2_O2 1.0e-5  kg/(s*m)

MaterialSpec Anode From Electrolyte
  Equation  TransportH2  Enable
  Parameter SigmaIso   3.0E4 S/m
  Parameter Diff.H2_H2 1.0e-5  kg/(s*m)
```

and the boundary conditions with the statements

```
BC  Cathode 0 ny IType nx
  Dirichlet Psi   0 V
  Dirichlet O2    xO2

BC  Anode   0 0 IType nx
  Floating  Psi  -current mA
  Dirichlet H2    xH2

Define
 NAVO    = 6.02202E23
 Ginv    = 1/1.0E5
 O2flux  =-1/4*mmolO2/NAVO/Q0
 H2flux  = 2/4*mmolH2/NAVO/Q0

Macro CUR { (Current.X*Normal.X+Current.Y*Normal.Y) }

BC OxyRedox  Restrict material(Cathode)  0 ny0+ny1 IType nx
```

```
  Neumann O2 D_Current O2flux*@CUR,(Normal,0)*O2flux  kg/s/m**2-kg/(A*s)#3

BC HydOxid   Restrict material(Anode)    0 ny0 IType nx
  Jump    Psi D_Current 1.10-@CUR*Ginv,(Normal,0)*(-Ginv) V-V/(A/m**2)#3
  Neumann H2  D_Current H2flux*@CUR,(Normal,0)*H2flux     kg/s/m**2-kg/(A*s)#3
```

Since for this simple model, the potential jump is just a function of the current, the potential is independent from the chemistry and should be solved in a first step and the mass flow equations for O_2, H_2 in a second step. These two equations are also uncoupled to each other, they just depend on the normal current flow at the anode-electrolyte interface and so we can solve each equation in turn to get a solution of this SOFC system. However, this 1D model is also very small that we may decide to solve all equations together and since the full system is linear, just a single step is required.

```
Convergence 1
BlockStruct Block Psi H2 O2
Solve Stationary
```

Numerical Results

For this example, Fig. 2.152 shows the potential distribution along the vertical line. The potential jump (2.178) is given by $\approx 1.01\,\mathrm{V}$ and we have quite a large potential drop within the electrolyte due to the small electric conductivity. Because of the small vertical dimensions, the concentrations of O_2, H_2 are almost constant and equal to the ones set with the Dirichlet BCs. The current is overall constant and given by the prescribed value at the floating BC for the potential. For this example, the only figure of merit is the knowledge of the potential at the anode which gives us the electric power supplied by the cell. This value is given by the computed potential shift at the floating BC. A summary of these working conditions is given with the following statement.

```
Write
  Text "Potential=%.3f V Current=%.3f mA Power=%.3f mW\n"
       -Anode.Psi.Shift, current, -Anode.Psi.Shift*current
```

As stated before, for this simple example the potential is completely independent from the chemistry so that the figure of merit may be obtained by just solving for the potential. However, we have computed the O_2, H_2 concentrations so that if one defines the Nernst potential as a function of these concentrations, one can right away study a full coupled system. As a next step in refining the modeling of a SOFC, one should consider the temperature dependency of the heterogeneous reactions and the fuel supply which may decrease the reaction rates whenever a shortage occurs.

References

[1] J. LARMINIE, A. DICKS, *Fuel Cell Systems Explained*, Wiley, 2000.

[2] A. A. KULIKOVSKY, *Analytical Modelling of Fuel Cells*, Elsevier, 2010.

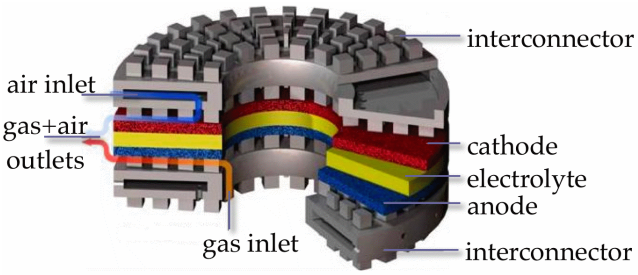

Figure 2.153: View of a single layer in a SOFC stack system.

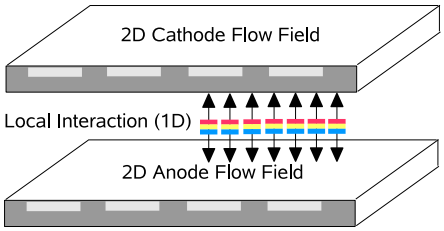

Figure 2.154: The 2D+1D modeling approach with 2D cathode and anode domains and a 1D point-to-point coupling.

2.36 A planar 2D+1D SOFC model

In this example, we further develop the previous tutorial example and go a step further in the modeling of a solid oxide fuel cell (SOFC) system like the one shown in Fig. 2.153. In particular, we are going to use a 2D+1D modeling approach which allows us to avoid computationally intensive 3D calculations, without impairing too much on the model's quality, see Fig. 2.154. The anode and cathode are modeled separately by 2D transport models accounting for multicomponent fluid flow and possibly heat transport. The layer in between consisting of the gas diffusion layer (GDL) for the cathode and the anode together with the electrolyte is modeled locally by a 1D model connecting both the 2D cathode and anode domains point-to-point, i.e. each point within the cathode is connected with a point at the anode through a point-to-point transversal cross-coupling possibly accounting for porous flow, multicomponent diffusion, heat transport, charge transport and electrochemical reactions. The definition of the 1D model is completely left to the user and it is part of the input. Simple analytical or semi-analytical models are generally directly specified within the *SESES* input syntax, however, there is also the possibility to solve sophisticated 1D problems by finite difference or finite element methods with the help of numerical routines embedded in dynamic libraries to be loaded at run time. The question is of course if such 2D+1D models make any sense. It depends, but for SOFC and proton exchange membrane (PEM) fuel cells where the lateral dimensions are order of magnitude larger than the electrolyte and GDL layers together, they surely do. Because of the large ratio between thickness and lateral dimensions, you may assume that lateral transport within the thin vertical layer may be neglected, with obvious advantages. The modeling of the 1D vertical transport is completely decoupled from the 2D transport and so numerical difficulties related to the large geometric ratios are avoided. Further no matter how complex the 1D model may be, the overall cost is just given by its evaluation growing linearly with the number of elements used by the 2D transport model. So the limiting factor is always represented by the solution of the 2D transport problem which is mainly determined by the number of dof-fields and the type of cross-coupling between the cathode and anode domains. For this 2D+1D approach, there is a special built-in model `Coupling1D` allowing to couple point-to-point two non-overlapping domains through the right-hand-side of the modeling equation associated with some dof-fields. The list of dof-fields which can be coupled is free and the implementation is optimal in the sense that if the whole system to be modeled is linear, then the solution is obtained with a single linear solution step. However, the computational work grows

with the number of dof-fields involved in the cross-coupling, so that if the coupling is weak, one may be faster in computing solutions by using a Gauss-Seidel solution strategy and setting to zero some or all cross-coupling derivatives.

Mathematical model

Even for the simplified 2D+1D approach, there are many possible choices available for the model's complexity as e.g. the number of species involved, thermal dependency, dynamic behavior so that even here, the computational expenditure may be of concern. Therefore, we are not going to present in full details the modeling of a SOFC system ready for production, but limit ourself to the main features. In particular, we make the isothermal assumption and assume a constant working temperature of $T = 1223K$. Extensions to include the temperature dependency are quite straightforward, once the temperature dependency of the the material laws is known. One just has to provide these dependencies and for fast convergence their derivatives with respect to the temperature. We also are not going to consider to many species, since they just increase the computational cost without necessarily improving the insight understanding. We thus consider the species O_2, N_2 at the cathode and H_2, H_2O, N_2, CO, CO_2 at the anode. The 2D cathode and anode domains are actually two distinct flow channels, the cathode is an in-air flow suppling oxygen and at the cathode we have a gas flow suppling hydrogen for a proper operation of the the electrochemical reaction in the fuel cell. We assume both flows to be laminar and irrotational, i.e. a potential flow, which allow us to use a simple linear model to compute the velocity $\mathbf{v}$ and pressure p with the model equation `PorousFlow`. Within both channels, additionally to the overall mass transport there is mixing of species by diffusion, convection and possibly chemical reactions. Therefore for the cathode we solve additionally for the equations `TransportO2`, `TransportN2` and at the anode for `TransportN2`, `TransportH2O`, `TransportH2`, `TransportCO`, `TransportCO2`. Species convection is automatically considered by solving for the velocity $\mathbf{v}$ and for species diffusion we use the Stefan-Maxwell diffusion law available in *SESES* with the built-in model `StefanMaxwellDiff`. In order to work with the Stefan-Maxwell model, we need to work with species mole-fractions x_α for the dof-fields and the constraint $\sum_\alpha x_\alpha = 1$. Numerical solutions will satisfy this latter condition by a proper setting of the BCs and if the sum of the species production rates Π_α is everywhere zero $\sum_\alpha \Pi_\alpha = 0$, a fact proven in *SESES* manual. Actually, these 2D transport equations should not only consider the transport in the air and gas channels, but also the lateral transport in the underlying GDL of cathode and anode. However, being these GDL layers made of solid matter, their contributions to transport may be neglected.

Due to the high temperature, the hydrogen at the anode may burn out if there are some oxygen's leftovers, but since we are excluding oxygen as species at the anode this case does not apply. However, as an example, we will consider the so called CO shift reaction $CO + H_2O \rightarrow CO_2 + H_2$ with the following kinetic model relying on phenomenological formulas. For species compositions far from thermodynamic equilibrium, the forward rate expression is given by

$$r_{\text{fwd}} = k_0 \exp(-\frac{E_a}{R_{\text{gas}} T}) x_{CO} \,, \tag{2.179}$$

with R_{gas} the gas constant, k_0 a pre-exponential factor and E_a a typical activation energy of the process. For the overall reaction rate we use ad hoc the expression

$$r = r_{\text{fwd}} \cdot \left[1 - \frac{x_{H_2} x_{CO_2}}{x_{CO} x_{H_2O}} \exp\left(\frac{\Delta h - T\,\Delta S}{R_{\text{gas}} T}\right)\right], \tag{2.180}$$

with Δh the reaction enthalpy and ΔS the reaction entropy. While the expression (2.179) is correct far from equilibrium in the forward direction, it is no longer correct near thermodynamic equilibrium and (2.180) represents an amendment in order to attain equilibrium. Even for compositions favoring moderately the reverse reaction, the expression seems to be acceptable since it assumes small negative values forcing the gas composition back to equilibrium. Far form equilibrium with reverse reaction conditions, the applicability is questionable since the reaction kinetics will almost certainly fail to be correct. However, this condition will not be attained during sensible SOFC-simulations. From the reaction rate (2.180) in unit of mole/s, we obtain the species production rates as $\Pi_\alpha = M_\alpha \nu_\alpha r$ with M_α the species molar mass and ν_α the stoichiometric reaction coefficients and since mass is not created nor destroyed, we satisfy the necessary condition $\sum_\alpha \Pi_\alpha = 0$.

Both gas and air channels are interleaved with nopples used to electrically contact the anode and cathode surfaces. No convective transport is possible within the nopples, so that here we just need to model the 2D conductive transport in the solid GDL. We use here the same Stefan-Maxwell diffusion laws as for free convection but we scale down all coefficients by a constant factor. However, as we will see later, underneath the nopples and because of the 1D cross-coupling, the total mass production rate is not zero. The governing equations for the species transport cannot take this into account and so in order to collect together and not to loose this mass and to have a correct account on the mass flow, we still solve for the convective mass transport but by forcing a slow velocity. Electrial current transport is also considered at the anode and cathode by defining the model equation `OhmicCurrent`. Since it is not possible to set BCs over an open domain just the nopple's boundary is defined as an electrical contact and within the nopples we use a large fictive electric conductivity to numerical enforce an equipotential condition over the nopple's area. A factor of $10 - 100$ larger than usual will do fine without impairing too much the numerical stability.

With the description of the 2D transport models for anode and cathode we are done and we are left with the 1D point-to-point model specification. Several choices are possible here, but it is also true that these 1D models are easily changed, adapted to any particular situation and plugged-in without the need to change the structure of the 2D transport model, so that for our purposes any 1D model will actually do fine. Essentially we are going to use the same model of the previous tutorial example with some adaption towards non-linearity. The 1D species transport within the GDLs as well as the oxygen ions O^{2-} transport in the electrolyte are left unchanged. For the potential jump at the anode-electrolyte interface, we now use a non-linear model

$$\Delta\Psi = \Delta\Psi_{\text{Nernst}}(x_\alpha) - \frac{2R_{\text{gas}}T}{qN_{\text{avo}}}\text{asinh}(jF(x_\alpha))\,, \tag{2.181}$$

with j the positive current flow, q the elementary charge, N_{avo} the Avogadro's constant and where the Nernst potential $\Psi_{\text{Nernst}}(x_\alpha)$ and the function $F(x_\alpha)$ are now dependent from the species concentrations x_α at the interface. The asinh dependency

of the overpotential from the current j stems from the Butler-Volmer electrochemical reaction model with an equal symmetry factor for anode and cathode and it is a generally accepted model. We also use the same model at the cathode-electrolyte interface to model the potential jump for the oxygen's reduction. For this simplified 1D system, where we use constant diffusion coefficients without cross-coupling diffusion of the species, it is not hard to write down the scalar governing equation $r(j, \vec{p}) = 0$ for the current j as function of the input parameters $\vec{p} = (x_\alpha^a, x_\alpha^c, \psi^a, \psi^c)$ representing the species concentration and potential values at the borders of the anode and cathode GDLs. The solution to this equation $j = j(\vec{p})$ as function of $\vec{p}$ is generally not available in analytical form and so we may use a bisection or Newton's algorithm to find the solutions. What is needed as input to *SESES* are the current values and species production rates at the borders of both GDLs as function of the parameters $\vec{p}$ together with all non-zero partial derivatives. But since all production rates are directly proportional to the current j, we see that actually all what we need is the current j and the derivatives $\partial j/\partial \vec{p}$. The equation $r(j, \vec{p})$ together with the partial derivatives $\partial r/\partial(j, \vec{p})$ can be conveniently assembled symbolically and some *Maple* code is provided within the input. If one uses a Newton's algorithm to find solutions, then one just performs the iteration $j := j - r(j)/(\partial r/\partial j)$ up to convergence of j and at the end we have $\partial j/\partial \vec{p} = -(\partial r/\partial \vec{p})/(\partial r/\partial j)$. In our case, this Newton's iteration is initialized with the analytical solution available when species diffusion is neglected and by taking $\mathrm{asinh(x)} \approx x$ in (2.181).

What is left to do is the definition of the production rates and partial derivatives required by *SESES* as function of the computed values j, $\partial j/\partial \vec{p}$. This step only depends on the stoichiometry of the electro-chemical reaction and it is therefore independent from the particular 1D model used. However, we now come to an important fact, which absolutely cannot be neglected and it is often a source of troubles for beginners. The problem lies in the fact that oxygen in the cathode air channel disappears in order to reappear at the anode side in form of O^{2-} ions before the electrochemical reaction takes place. Nothing special but now the sums of the single production rates at the cathode $\Pi_{\mathrm{cathode}} = \Pi_{O_2} + \Pi_{N_2}$ and at the anode $\Pi_{\mathrm{anode}} = \Pi_{H_2} + \Pi_{H_2O} + \Pi_{N_2} + \Pi_{CO} + \Pi_{CO_2}$ is not anymore zero, in other words mass disappears at the cathode $\Pi_{\mathrm{cathode}} < 0$ in order to reappear at the anode $\Pi_{\mathrm{anode}} > 0$, however we still have $\Pi_{\mathrm{cathode}} + \Pi_{\mathrm{anode}} = 0$. This is an inconsistency for the single 2D transport problems with the result that the constraint for the sum of the mole fractions at the cathode $x_{O_2} + x_{N_2} = 1$ and at the anode $x_{H_2} + x_{H_2O} + x_{N_2} + x_{CO} + x_{CO_2} = 1$ cannot be anymore obtained by the numerical solutions which would require $\Pi_{\mathrm{cathode}} = \Pi_{\mathrm{anode}} = 0$. In order to proceed, we need to amend the transport equations and in order to see how this is done, we state the governing balance equations for the stationary mass conservation and species transport

$$\begin{aligned} \nabla \cdot (\rho \mathbf{v}) &= \Pi_0 \,, \\ \nabla \cdot (\rho_\alpha \, \mathbf{v}) &= -\nabla \cdot \mathbf{j}_\alpha + \Pi_\alpha \,, \end{aligned}$$

with ρ_α the species mass density, $\rho = \sum_\alpha \rho_\alpha$ the total mass density, $\mathbf{j}_\alpha$ the species diffusion currents, Π_α the species production rates and $\Pi_0 = \sum_\alpha \Pi_\alpha$ the total mass production rate. In *SESES*, the above species transport equations are not solved in this form, but they are transformed with the help of the first equation and by considering the usual case $\Pi_0 = 0$. Without this assumption, we can repeat this algebraic step to

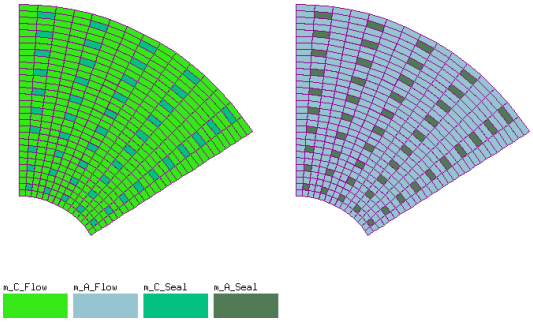

Figure 2.155: Computational domains for cathode and anode with nopples.

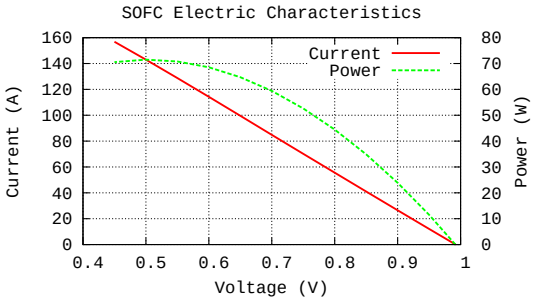

Figure 2.156: Electrical characteristics of the SOFC system.

obtain

$$\nabla \cdot (\rho \mathbf{v}) = \Pi_0 \,,$$
$$\nabla(\tfrac{\rho_\alpha}{\rho}) \cdot \rho \mathbf{v} = -\nabla \cdot \mathbf{j}_\alpha + \Pi_\alpha - \Pi_0 \tfrac{\rho_\alpha}{\rho} \,.$$

The left-hand side of the species transport equation are actually the equations solved by *SESES*, so that if $\Pi_0 \neq 0$ we need an amendment for the right-hand side. With this amendment, we see that the sum of the new productions rates is always zero thus assuring the validity of the constraint $\sum_\alpha x_\alpha = 1$ during the solution. It is to be noted that it is well possible to compute mass and species transport problems without requiring the mole fraction's sum to be one. In fact, it is up to the user to enforce this constraint by properly choosing correct BCs and production rates, however, if this is not the case we cannot use anymore the `StefanMaxwellDiff` built-in model. Clearly, if the mole fraction's sum must be one and this constraint remains valid during the solution, one mole fraction is completely redundant and in fact for speed-up purposes we are not going to solve the transport equation for one judiciously chosen species taken as N_2. However, some numerical amendments are then required which however run much faster then the solution of one transport equation. If the amendments of the right-hand side of the species transport equations are strictly required, the one on the right-hand side of the total mass transport is not. In fact, one can argue that in our case Π_0 does not have a strong impact on the velocity $\mathbf{v}$ profile and therefore a value of $\Pi_0 = 0$ may be considered in order to completely decouple the total mass flow from the other transport equations resulting in a computational speed-up. However, since the coupling through a non-zero Π_0 is weak, there is no need to specify cross-coupling derivatives and therefore there is almost no slow down in considering this type of coupling. For a correct account of the mass flow, we need to solve for the mass transport everywhere where $\Pi_0 \neq 0$ and this is include the area underneath the nopples where virtually there is no convection at all.

Model specification

The *SESES* input file for this model can be found at `example/SOFC2p1.s2d`. For the cathode and anode, we define two disjoint domains with the same geometry and a total of four materials, see Fig. 2.155. Two materials are for the cathode and anode flow channels and other two materials model the nopple's area where convective flow is almost turned-off. Some details about the material's properties have already been

discussed and are not repeated here. The thickness of the third dimension is set with the statement

```
GlobalSpec Parameter Depth        0.75E-3 m
```

to be the same of the channel thickness of $0.75\,\mathrm{mm}$ so that the computed mass flows within the channels as displayed by the BC characteristics are effectively the real ones. This however will yield wrong values for the electric current through the nopples since this current do not need to be multiplied by the virtual thickness of the device.

In order to use the point-to-point domain cross-coupling, the cathode is defined as the master domain and therefore for the cathode material we enable the built-in model

```
Model Coupling1D(BlockOffset 1; NoRow H2 H2O O2 CO N2 CO2 Pressure Psi;
                                Psi H2 H2O O2 CO N2 CO2 Pressure
```

The `BlockOffset` model's parameter is used to locate the slave domain i.e. the anode domain within the ME mesh. In this example, the cathode is the whole ME-block 0 and the anode is the ME-block 1 so that we have a ME-block offset of one. The parameter `NoRow` will be discussed shortly. As last model's parameter, we list the dof-fields being cross-coupled together between the cathode and anode domains which in our example are all defined dof-fields. This list is followed by the numerical values of the right-hand-sides associated with the governing equations solved both on the cathode and anode domains together with all non-zero cross-coupling derivatives. As discussed previously, these numerical values are obtained by solving locally for a 1D transport model using Newton's algorithm to first obtain the electric current and then by setting all required input values according to stoichiometry of the electro-chemical reaction. For convenience, we have used *Maple* to generate once the rather lengthly input text which is not shown here.

At the bottom of the cathode, we prescribe an air flux of $\approx 0.0001\,\mathrm{kg}/(\mathrm{sm}^2)$ with a floating BC for the pressure and fix the mole fraction's value with Dirichlet BCs by respecting the constraint $\sum_\alpha x_\alpha = 1$. At the top of the cathode, we do not prescribe any BCs, so that convective flow will be free to leave the cathode, however, conductive flow will be blocked. In a similar fashion, we set the BCs on the anode with a gas flux of $\approx 0.001\,\mathrm{kg}/(\mathrm{sm}^2)$. Larger values of air or gas fluxes will impair or prevent the convergence of the solution algorithm due to the coarse element mesh around the nopples where the velocity field is forced to change direction. The dof-fields are initialized in order to satisfying from the beginning the constraint $\sum_\alpha x_\alpha = 1$ and they are set to the values of the Dirichlet BCs.

As noted previously, since the mole fractions must sum to one, a species is completely redundant and for speed-up it would be nice to remove this redundancy. As first we have to choose the species to become the dependent species and as will be clear in moment, one should take the most inert one, in our case N_2. One then proceeds exactly as when computing solutions with N_2 and the problem specification is virtually unchanged up to computing the numerical solutions. In general by transport problems with more than one species, one solves for all species together in a single dof-field block and here as first step one just drops the equation for the N_2 dof-field. However, without any other contrivance, the constraint $\sum_\alpha x_\alpha = 1$ valid after initialization will be immediately violated after the first Newton's update. Therefore, at the end of each

Newton's step, we set up an off-block dof-field correction for N_2 to restore this condition. This is done with the `Increment` directive and the built-in function `setdof` as follow

```
Increment Standard
If !WithN2
{ {
   setdof(N2; 1-(Dof.O2-Incr.O2)-(Dof.H2-Incr.H2)-(Dof.H2O-Incr.H2O)-
                (Dof.CO-Incr.CO)-(Dof.CO2-Incr.CO2));
} } # 3
```

The preprocessor switch `If !WithN2` is used to enable/disable the direct computation of the N_2 dof-field within this example and the variable `WithN2` is defined at the beginning of the input. It is to be noted, that in order to compute this new value, we need both the values of the other dof-fields and their block increments, since dof-field values within `setdof` are values before the block dof-increments are subtracted to the dof-fields. Also such an amendment makes only sense if without it, a solution is computed satisfying the constraint $\sum_\alpha x_\alpha = 1$. Otherwise nothing will converge, since the correction is actually inconsistent with the solution. Assuming that Newton's algorithm for the solution of the residual equations $\vec{R}_\beta(\vec{x}_\alpha) = 0$ for all species converges quadratically, dropping the computation of one species and solving just for $\vec{R}_{\beta'}(\vec{x}_{\alpha'}, \vec{x}_{N_2}) = 0$ with $\alpha', \beta' \neq N_2$ and by applying the off-block dof-field correction for $\vec{x}_{N_2}$ generally leads to slow and/or non-convergence of Newton's iteration. The reason is that if we do not solve for $\vec{x}_{N_2}$, the native derivatives $\partial\vec{R}/\partial\vec{x}_{\alpha'}$ are not exact derivatives anymore since we are missing terms stemming from the dependency $\vec{x}_{N_2} = \vec{1} - \sum_{\alpha'} \vec{x}_{\alpha'}$ in $\vec{R}(\vec{x}_{\alpha'}, \vec{x}_{N_2})$. For user defined derivatives within material laws, the user may amend these values to obtain exact derivatives, however, this is not possible for the internal derivatives computed by *SESES*. Although an ideal quadratic convergence of Newton's algorithm will be unavailable, the situation is not so dramatic if we can still get good convergence rates. In order for this to happen, we have to reduce to a minimum the dependency of the residual equations $\vec{R}(\vec{x}_{\alpha'}, \vec{x}_{N_2})$ from the $\vec{x}_{N_2}$ mole fractions. So the first step consist in the choice of the most inert species N_2 as redundant species which has the minimal impact on the reaction rates. The second contrivance is a little more technical and has to do with the Stefan-Maxwell diffusion law used for the transport process. Within this law, there is cross-diffusion among all species and the diffusion matrix $D_{\alpha\beta}$ is in general a full matrix. Therefore the transport equation for each species is strongly coupled to the N_2 species and neglecting this dependency will in general prevent convergence of Newton's iteration. However, we know that solutions are invariant by addition of any constant row to the diffusion matrix $D_{\alpha\beta}$. So the trick is to add those constant rows resulting in a zero column $D_{\alpha N_2} = 0$ which further reduce the N_2 dependency of the residual equations. This trick is enabled with the option `ZeroEntry N2` in the `StefanMaxwellDiff` model and can only by used if we do not directly solve for N_2, otherwise numerical instabilities arise. In fact, the flexibility available in the choice of the diffusion matrix is used to obtain stable numerical computations when all species are solved together. For this particular SOFC problem, the application of these two ideas lead to very acceptable convergence rates even without amending the user defined derivatives, but we remind that in a very general case it may be necessary to compute all species together.

For this problem, a possible solution strategy is to solve for all dof-fields together ex-

cept for the N_2 mole faction with a fully coupled Newtow's algorithm. The solutions are quite smooth and non-linearities are weak so that we can speed-up the Newton's iteration, by using a direct solver and reusing the factorized stiffness matrix for subsequent iterations. This behavior is obtained with the option `ReuseFactoriz` in the `Increment` statement and in our example is activated after three iterations with full factorization. The implementation of the model `Coupling1D` is optimal in the sense that for linear systems, just a single linear step is required for the numerical solution. The price to pay is that the dof-fields on both the cathode and anode domains become matrix connected and the cost for a linear solution step increases accordingly. This may be a necessity for strongly coupled point-to-point domains to assure acceptable convergence rates, but in our case the coupling is weak and we may neglect some matrix connections between the dof-fields on both domains with the options `NoRow`, `NoCol` of the `Coupling1D` model. Note that the coupling function is left unchanged, we are just discarding cross-coupling derivatives so that the convergence rate of Newton's algorithm will further slow down, but each linear step will be cheaper. A numerical investigation has shown that we can freely neglect all cross-coupling derivatives except, in some circumstances, for the electric potential. This is indeed the case, if we use a floating BC for the potential on a domain in order to set the total current flow, since by neglecting the potential cross-coupling derivatives, the stiffness matrix becomes singular. However, we also see that since the nopples are quite close together and the electric conductivity is large enough, the potential value at the cathode and anode GDLs is almost constant and its computation is not really necessary. There is a switch `NoPsi` to turn-off solving for the electric potential and in order to compute the electric current, we define a user element field

```
ElmtFieldDef LocCur(DofField O2)
```

on the cathode where just the species O_2 is defined. Each time the model `Coupling1D` is called, we then store in the element field `LocCur` the current value and at the end of a computation, a user integral over the cathode's domain

```
Write AtStep 1 "Voltage=%f V, Current=%e A, Power=%e W\n"
{
  double totcur= If WithPsi { Apply.Psi.Flux/Depth; } Else
                            { integrate(Domain domain(dm_Cathode); LocCur); }
  return Voltage,totcur,totcur*Voltage;
}
```

will return the total current together with the fuel cell electrical power, as shown in Fig. 2.156 and displaying a linear I-V characteristic. If the computation of the potential field is enabled, the total current is available as the BC characteristic stored in the variable `Apply.Psi.Flux` which, however, has been multiplied internally by the virtual thickness of the device and need to be amended.