Assignment in
The Finite Element Method, 2024

Division of Solid Mechanics

The task is to write a finite element program to analyze temperature and stress distributions in a battery prototype. The problem should be solved with MATLAB or Python with the aid of CALFEM.

Problem description

Due to an explosion in electrification the demand for cheap, effective and durable batteries has grown significantly in the past decade. A well-known problem engineers are faced with in battery design is fatigue due to repeated charging and discharging. One possible way of alleviating this is by actively cooling the battery. Computational engineers (you) have been assigned with analyzing a new battery prototype, see the diagram below.

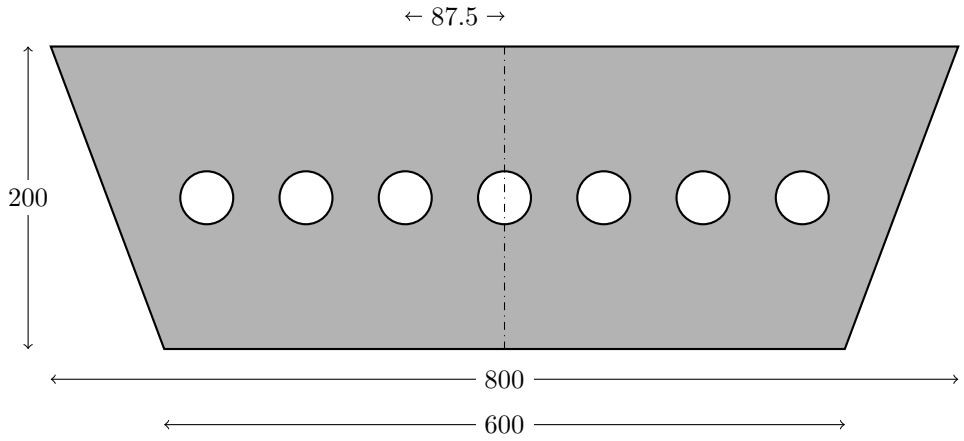
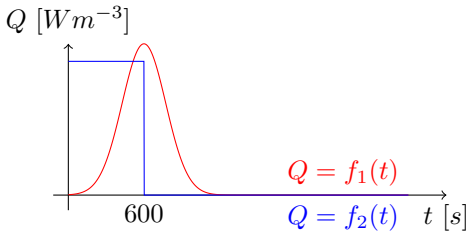


Figure 1: Diagram of a cross section of the battery cell. Measurements are given in mm, and the radii of the circles are $25mm$, the out-of-plane thickness is $1600mm$. The battery is symmetrical, which is indicated by the dashed line.

Heat is generated in the battery when it is charged. Due to natural convection at the top boundary a small portion of this heat is irradiated to the surrounding environment. Additionally, seven cooling channels have been placed along a horizontal line in the middle of the battery, to reduce thermal strains. Every other channel carries cool liquid from a cooling system and every other carries heated liquid back. Since the battery is very thick, you can safely use plane strain to approximate the out-of-plane behavior.

Two different charging strategies are considered, a 'smart' charging strategy and a 'normal' charging strategy. While the total amount of heat supplied to the battery is the same, the effect varies over time when using the former and is constant using the latter. The effect generated at time t seconds can be modeled by two functions, $f_1(t)$ and $f_2(t)$,



$$f_1(t) = 100.0 \exp\left(-144\left(\frac{600-t}{3600}\right)^2\right) kWm^{-3}$$

$$f_2(t) = 88.42\ \theta\left(\frac{600-t}{3600}\right) kWm^{-3}$$

where $\theta(\tau) = 1$ if $\tau > 0$, else 0, exp denotes the exponential function with base e, and the time t is in seconds.

As previously mentioned, your task is to analyze the thermo-mechanical behavior of the battery. You know that having an accurate boundary description is essential, so you ask your experienced colleague for some advice. They give you the sketch shown in the figure below. You realize that due to symmetry you only need to model half the battery, which saves some computational time.

The left half contains the boundary conditions of the mechanical problem. On the top boundary and on the circular holes you prescribe zero traction. On the remaining boundary you prescribe zero displacement.

The right half contains the boundary conditions for the thermal problem. You model the top boundary with newton convection, with $\alpha_n = 40W/m^2K$ and $T_\infty = 293°K$. The cooling channels can also be modeled as newton convection, with $\alpha_c = 120W/m^2K$, and $T_{in} = 277°K$, $T_{out} = 285°K$. The initial temperature of the battery is $T_0 = 293°K$.

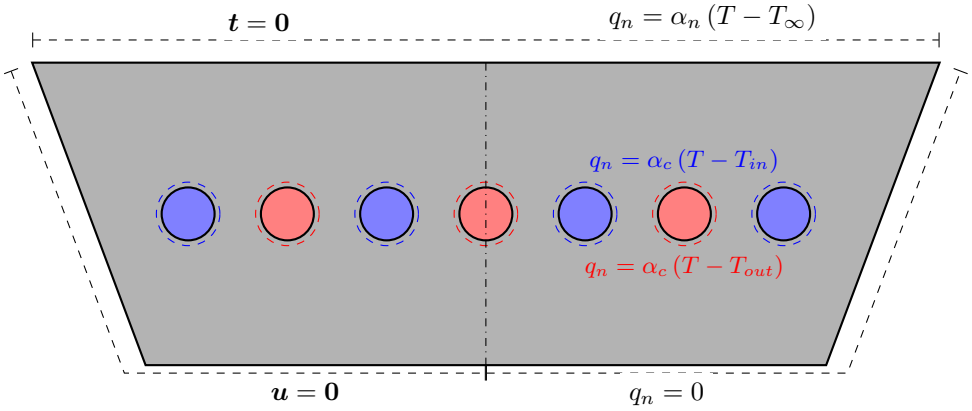


Figure 2: Sketch of boundary conditions of battery cell.

For the sake of simplicity you decide to model the battery as a homogeneous material, and find the following material parameters in a trusted material-parameter encyclopedia.

	Battery
Young's modulus, E [GPa]	5
Poisson's ratio, ν [-]	0.36
Expansion coefficient, α [$1/K$]	$60 \cdot 10^{-6}$
Density, ρ [kg/m^3]	540
Specific heat, c_p [J/kg-K]	3600
Thermal conductivity, k [W/m-K]	80

Table 1: Material data.

Tasks

A summary of the tasks to be considered are given below.

a) Compute the stationary temperature distribution in the battery. What are the maximum and minimum temperatures in the battery, and which parts of the battery reach these temperatures? What is the largest deviation from the initial temperature?

b) Compute the transient temperature distribution for the first hour of the charging sequence, considering both charging strategies. Plot the maximum and minimum temperatures in the battery over time. Also plot the largest deviation from the ambient temperature over time, that is $\max(|T - T_0|)$. Lastly, plot the temperature distribution in the battery at 6 interesting time-steps.

c) Based on the previous results, when do you expect the largest von Mises stresses to appear? Compute the displacement field due to the thermal strains when the maximum deviation from the initial temperature is found. Plot the von Mises stress distribution in the battery at this time step. If the battery material starts to yield at around $\sigma_{eff} = 30MPa$, do you expect the battery to sustain at least one charging sequence? What problems can you identify with the current design? What problems can you identify with the physical model?

Hint: The von Mises stress is defined as:

$$\sigma_{eff} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 - \sigma_{xx}\sigma_{yy} - \sigma_{xx}\sigma_{zz} - \sigma_{yy}\sigma_{zz} + 3\tau_{xy}^2 + 3\tau_{xz}^2 + 3\tau_{yz}^2}$$

where it is important to not forget the out-of-plane stress component due to the assumption of plane strain.

Details

Triangular, 3-node elements are suitable for this task. A fully implicit time integration scheme should be used. Note that the element function for forming, $\boldsymbol{C}^e$, is available on the course homepage. As a starting point, use the strong formulations of the heat and the mechanical problems. Also note that you must derive the contributions for the convection boundary condition yourselves. This can easily be done by hand for the 3-node triangular element.

Report

A fundamental ingredient in all research is that it should be possible to regenerate the results obtained based on the report. In the present situation this implies that the appended source code should only be considered as supporting material. Moreover, note that one variable for grading the report is the structure of the computer code, i.e. you should choose suitable names for variables. A suitable structure for the report is:

- **Introduction**: Description of the problem, geometry and boundary conditions.
- **Procedure**: How the problems are solved (weak formulation, application of boundary conditions, treatment of thermal strains and so on). Derivation of the FE formulation and other important theoretical aspects should be included here. Note that you are encouraged to make references to textbooks. Though, it is important to carefully present all calculations that are not available in the literature.
- **Results**: Present the results in illustrative figures and/or tables. Note that the results should be commented such that the reader can not misunderstand the results (correct labels, units, figure texts etc.)

- **Discussion**: A discussion of the results, their meaning and significance. You might want to discuss sources of errors and accuracy in this section.
- **Computer Code**: Note that the code should be easy to follow and all declared variables should have intuitive names and be well commented.

A well structured report in **English** should be handed to the Division of Solid Mechanics no later than **2024-05-25**. Source files should be well commented and submitted alongside the report. The reader is assumed to have the same knowledge level as the author. If the report contains major programming or theoretical errors, the report is returned in order to be corrected. It is possible to obtain up to 5 points which are added to the points obtained at the exam in May 2024. The assignment should be approved no later than **2024-05-25**. You should submit your report in **PDF** format in Canvas. Note that the bonus points obtained is only valid for the examination in May 2024.

Collaboration

The task should be solved in groups of *two* or *individually*.

Generating mesh in python

The mesh should be generated using GMSH, which is wrapped in calfem. We recommend you follow the examples found in the documentation.

Generating mesh in MATLAB

In this assignment you should create your own mesh using the built-in PDEtool in MATLAB. This tutorial considers the example geometry seen in figure 3, which of course must be changed for your specific application.

PDEtool can be used directly to create simple geometries with little effort. To start just type `pdetool` into the MATLAB terminal to open PDEtool user interface. Although the tool is originally designed for solving partial differential equations only the meshing aspect are of interest in this assignment.

A step by step tutorial of how to use some of the basic features of PDEtool are provided here. Before starting to draw it is often convenient to activate grid and change the axis scales. This is done using <Options> menu. To build our geometry simple geometries are added one by one. The basic shapes used in PDEtool are rectangles, ellipses and polygons. Basic drawing tools can be found on the toolbar (see figure 4, note polygons are not necessary to create the geometry for the lab).

Draw a rectangle: Create a rectangle by using the rectangle tool (see figure 4) and simply hold left mouse button and move the mouse cursor to obtain the desired size. To adjust the size and position double click on the rectangle and a dialog will appear where coordinates and dimensions can be provided, see figure 5.

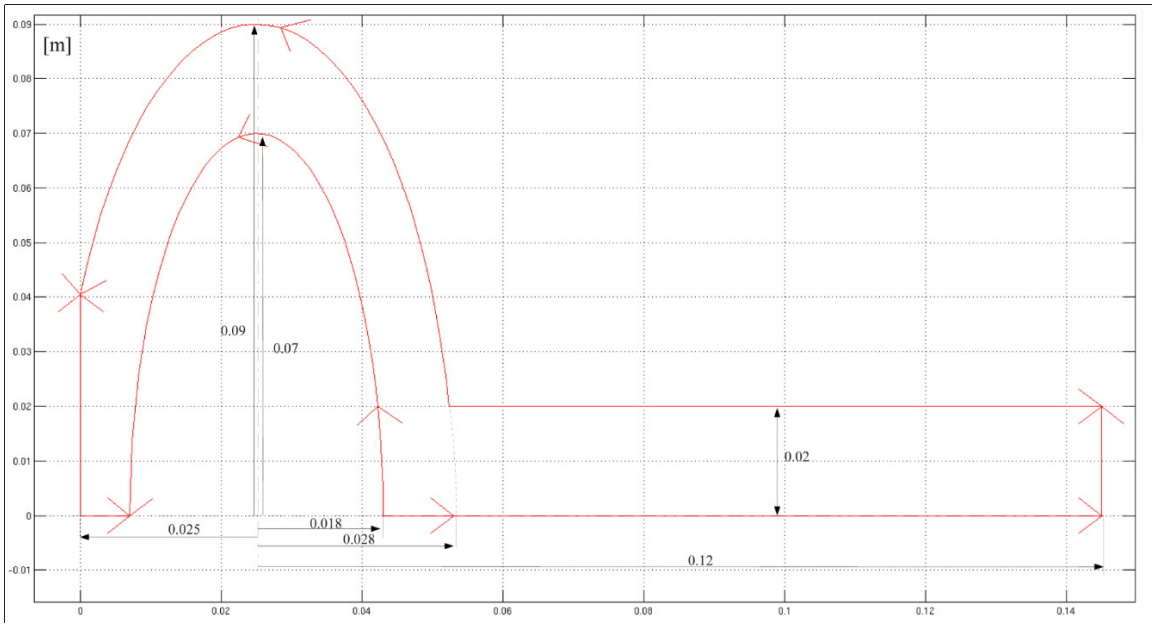


Figure 3: Dog toy geometry.

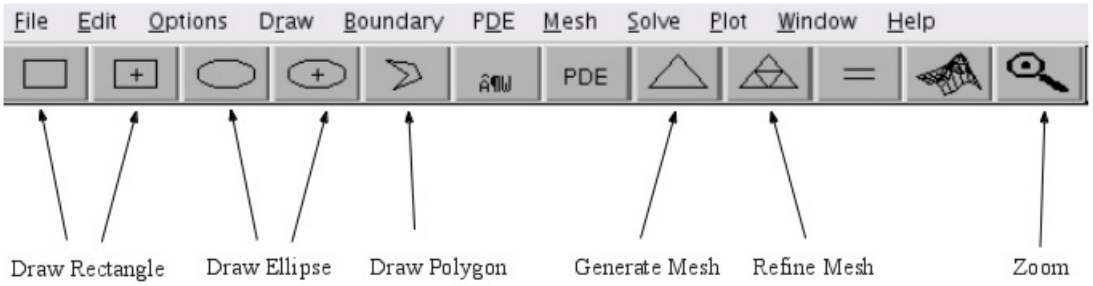


Figure 4: Basic tools for drawing in PDEtool.

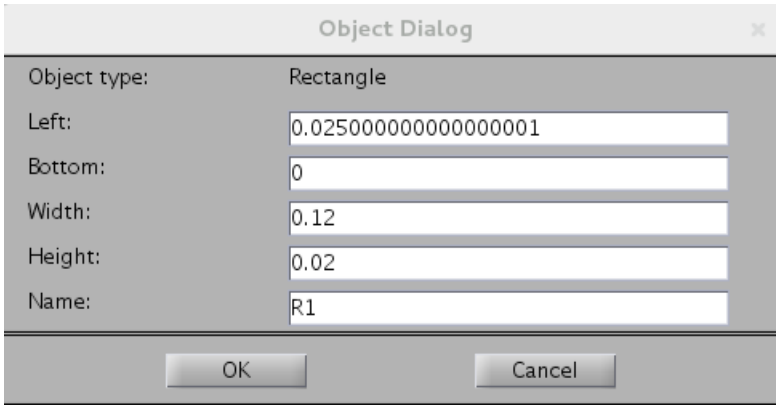


Figure 5: Rectangle object dialog.

Draw an ellipse: Create a ellipse by using the draw ellipse tool (see figure 4) and simply hold left mouse button and move the mouse cursor to obtain the desired size. Double click to adjust position and size in similar fashion as for rectangles.

Determining combined geometry: In the box below shape-buttons the present features are presented by name (which may be edited using object dialog). The plus sign indicates that geometries are added and minus signs indicate that they are cut out. Because of this the order is important i.e **R1+E1-E2 ≠ R1-E2+E1 in general**. Below figues showing drawn features and the resulting body after combining by grouping and changing signs in the set formula row (see figure 6, 8 and 7). Note that to see the resulting geometry we need to go to either meshing or boundary view (see toolbar).

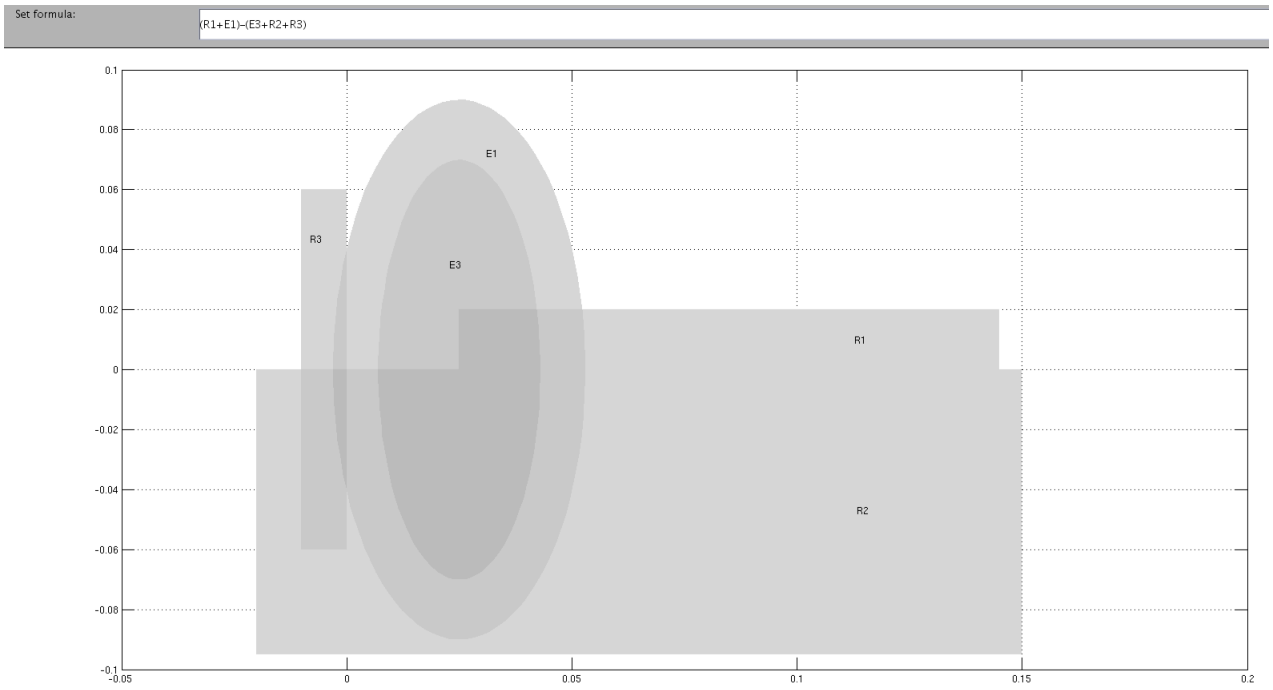


Figure 6: Set of combined shapes.

When the necessary features are added go to <Boundary> / <Boundary mode> to see which are the boundaries of your geometry. The red arrows define the main borders and the grey contours represent the so called subdomains. For instance, subdomains appear where features overlap. To get a uniform mesh it is convenient to remove the subdomains. To do so use <Boundary> / <Remove all Subdomain Borders>. If the mesh consist of different natural domains such as different materials it is favourable to keep the subdomains.

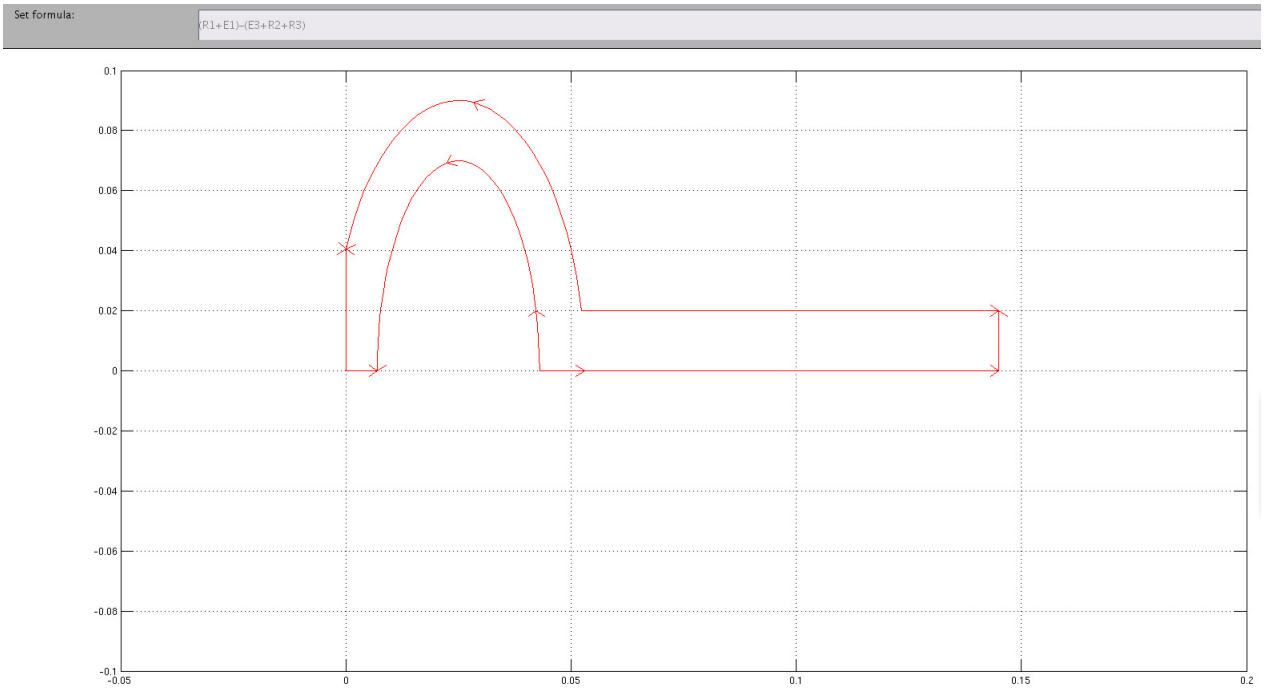


Figure 7: Resulting geometry seen in boundary mode (ctrl+B) after all subdomains have been removed.

To obtain the geometry provided in figure 7 the formula used is given in figure 8 and the object names is seen in figure 6.

Figure 8: Formula determining active areas.

To generate a mesh simply press mesh tool when the geometry seen in boundary mode is satisfactory (see figure 4). To refine the mesh, i.e. create more elements, press refine mesh until enough elements are generated (remember that finer mesh means higher computational cost but better accuracy). Note that a high number of elements increases the computational time.

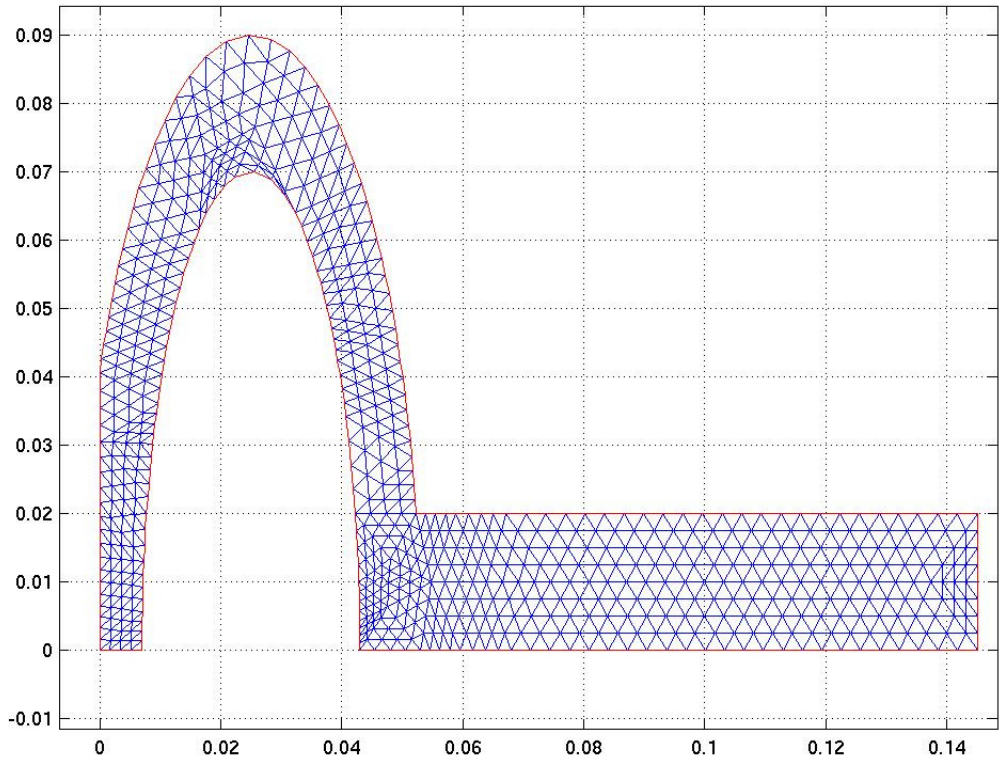


Figure 9: Mesh after two refinements.

When you are content with your mesh use <Mesh> / <Export mesh> to save the topology matrices associated with the current mesh (note you have to save the work separately to keep geometries etc. since <Export mesh> will only save matrices). The topology matrices are `p,e,t` (points, edges, triangles). The exported matrices will directly appear in the active MATLAB workspace. It is strongly advised to save them directly in a `.mat` file (mark variables in workspace, right click and save) so it can be loaded in the scrips you write. From `p,e,t` the CALFEM quantities `edof`, `dof`, `coord` etc can be extracted. On the course web page for FEM FAQ are some instructions of how this is done.

General tips for python

For general tips regarding calfem, see the documentation.

You may need to manually specify the backend for calfem's visualization engine, to do so add the following code at the top of your python script.

```
# IMPORTS
import calfem.vis_mpl as cfv
import matplotlib as mpl
mpl.use('TkAgg')
# YOUR PROGRAM
```

General tips for MATLAB

The output data from PDEtool is in the form `p,e,t` where `p` is *points*, it is a 2 row matrix where the first row is the x-coordinates and the second row is the y-coordinates. There are one column for each node i.e. in CALFEM notation `coord = p'`.

The output `t` is *triangles* and it has dimensions $4 \times$ nelm. The three first rows are the node numbers associated with each element (three nodes for linear triangular elements) while the fourth row is its subdomain. The subdomain is great to use when you want to associate specific material properties to each element. Rows 1,2 and 3 can be used to calculate our `edof` matrices, as an example

```
enod=t(1:3,:)';                         % nodes of elements
nelm=size(enod,1);                      % number of elements
nnod=size(coord,1);                     % number of nodes
dof=(1:nnod)';                          % dof number is node number
dof_S=[(1:nnod)',(nnod+1:2*nnod)'];     % give each dof a number
for ie=1:nelm
edof_S(ie,:)=[ie dof_S(enod(ie,1),:), dof_S(enod(ie,2),:),dof_S(enod(ie,3),:)];
edof(ie,:)=[ie,enod(ie,:)];
end
```

The output `e` is *edges* and contains among other things node-pairs belonging to a certain boundary segment. Thus if you activate the option in PDEtool, <Show Edge Labels> and <Show Subdomain Labels> in boundary mode you can see how the different segments are numbered in `e` and subdomains in `t`. For our intents and purposes the rows 1,2 and 5 are the relevant ones. Row 1 and 2 includes the node numbers of the element-segment and row 5 is the edge label, i.e. which boundary segment it belongs to. For instance a list of all the convective boundaries for the heat problems can be found from

```
% Check which segments that should have convections
er = e([1 2 5],:);   % Reduced e

%conv_segments = [10 11 12]; % Choosen boundary segments
edges_conv = [];
for i = 1:size(er,2)
if ismember(er(3,i),conv_segments)
edges_conv = [edges_conv er(1:2,i)];
end
end
```

`edges_conv` now contains the node numbers of the node pairs that belongs to the convective part of the global boundary. From this the length of each segment L can be determined when

calculating the contributions to boundary force vector and the stiffness matrix. Figure 10 shows an example of how the segment numbering and subdomains can look like.

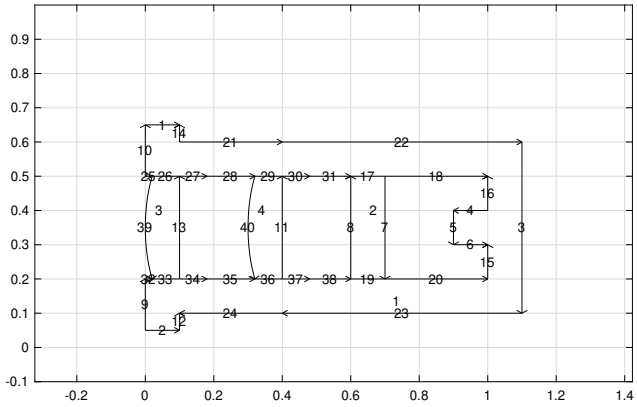


Figure 10: Example geometry of a lens with subdomains (red numbers) and edge labels (black numbers). Note that the meshing will be constrained by the edges (grey lines) so that an element is wholly within a single subdomain. Boundary segment 1 and 14 in this case corresponds to where the boundary conditions are changing at the outer boundary, e.g. 14 is isolated whereas 1 has convection.

Having made a mesh in PDE it is trivial to refine the mesh so it is advantageous to use a small mesh to test your code since it is much faster. When you think it is working use a finer mesh to get better resolved results.

From experience the CALFEM function `assem` and `insert` are very slow. The following code rows does the same thing:

```
%  Kt = assem(edof(el,:),Kt,Kte);
indx = edof(el,2:end);
Kt(indx,indx) = Kt(indx,indx)+Kte;

% f = insert(edof(el,:),f,fe);
indx = edof(el,2:end);
f(indx) = f(indx) + fe;
```

It is always nice to present nice plots, here are some short tips for plotting.

For instance the Matlab function `patch` can be used to generate field plots, e.g. `patch(ex',ey',eT')` where `eT` is the element temperatures obtained from `eT=extract(edof,T)` with `T` as the nodal temperatures. To plot the whole component (both sides of the symmetry cut) you can therefore write:

```
patch(ex',ey',eT')
hold on
patch(-ex',ey',eT')
```

If you use a fine mesh, the mesh lines can sometimes obscure the actual result. To remove the mesh lines from the plotting the option `'EdgeColor'` can be turned off (note that somewhere in the report the mesh used for the results should be shown), i.e. `patch(ex',ey',eT','EdgeColor','none')`.

Don't forget to turn on the colorbar and choose an illustrative color scale, e.g. `colormap(hot)` and set the axis scales to the correct proportions. Example:

```
figure()
patch(ex',ey',eT','EdgeColor','none')
title('Temperature distribution [C]')
```

```
colormap(hot);
colorbar;
xlabel('x-position [m]')
ylabel('y-position [m]')
axis equal
```

In order to compare several plots with each other it is recommended to use the same scale. By setting option `caxis([Tmin Tmax])` where `Tmin` and `Tmax` can be calculated or chosen.

To compare the deformation patterns it is preferred they are plotted on top of each other. This can be done in several ways but if `patch` is used the opacity can be set by the option `'FaceAlpha'` where 1 is completely opaque and 0 completely transparent. Example:

```
% Calculate displaced coordinates
mag = 100; % Magnification (due to small deformations)
exd = ex + mag*ed(:,1:2:end);
eyd = ey + mag*ed(:,2:2:end);

figure()
patch(ex',ey',[0 0 0],'EdgeColor','none','FaceAlpha',0.3)
hold on
patch(exd',eyd',[0 0 0],'FaceAlpha',0.3)
axis equal
title('Displacement field [Magnitude enhancement 100]')
```

where the third argument is a color triplet, i.e. `[0 0 0]` is black, so with `'FaceAlpha'=0.3` the result will be a grey-scale plot. You might have to change the magnitude to get nice plots.