Computational Materials Modeling FHLN05
Computer lab

Motivation

In the basic Finite Element (FE) course, the analysis is restricted to materials where the relationship between stress and strain is linear. For most many applications this is not the case and nonlinear and inelastic effects enters the formulation. In this computer lab you will write a FE program that is capable of dealing with such effects. The structure of a nonlinear finite element program will be examined. The understanding of the structure of a nonlinear finite element program is absolutely vital since it is virtually the same irrespectively of the cause of the nonlinear behaviour. One of the most extensively used algorithms for solving the global equilibrium equations in FE-analysis is the Newton-Raphson algorithm, which possesses a quadratic convergence rate. The main purpose of this computer lab is to gain understanding of the different aspects involved when establishing a nonlinear FE program. It is emphasized that the same Newton-Raphson algorithm can be used in the written assignment.

General instructions

Two assignments will be carried out during the course: one nonlinear elastic problem; this computer lab, and one elasto-plastic problem; the written assignment. Since a similar program structure will be used in the elasto-plastic problem as in the elastic problem it is of great importance that the computer lab is properly done before you start solving the assignment.
This computer lab should be carried out during a computer session and needs to be approved. You are not required to hand in a formal report but **in order to pass the lab it is required that you complete tasks a-h, given in the section 'Instructions' herein, and get an approval from your teaching assistant no later than October 5 at 10.00**. It is recommended that you work in groups of two but it is allowed to work alone. Please note that preparations (found on page 4) should be done before the lab.

Problem description

A toy for dogs is constructed from a nonlinear elastic, rubber like material and has the geometry shown in Fig. 1. The geometry is given in meters and the toy has a thickness of 3 cm. The dogs pull at different ends of the toy, causing it to deform. The material used to construct the toy is isotropic nonlinear elastic. The strain energy is assumed to be independent of the third generic strain invariant, i.e. $W = W(\tilde{I}_1, \tilde{J}_2)$. The stress strain

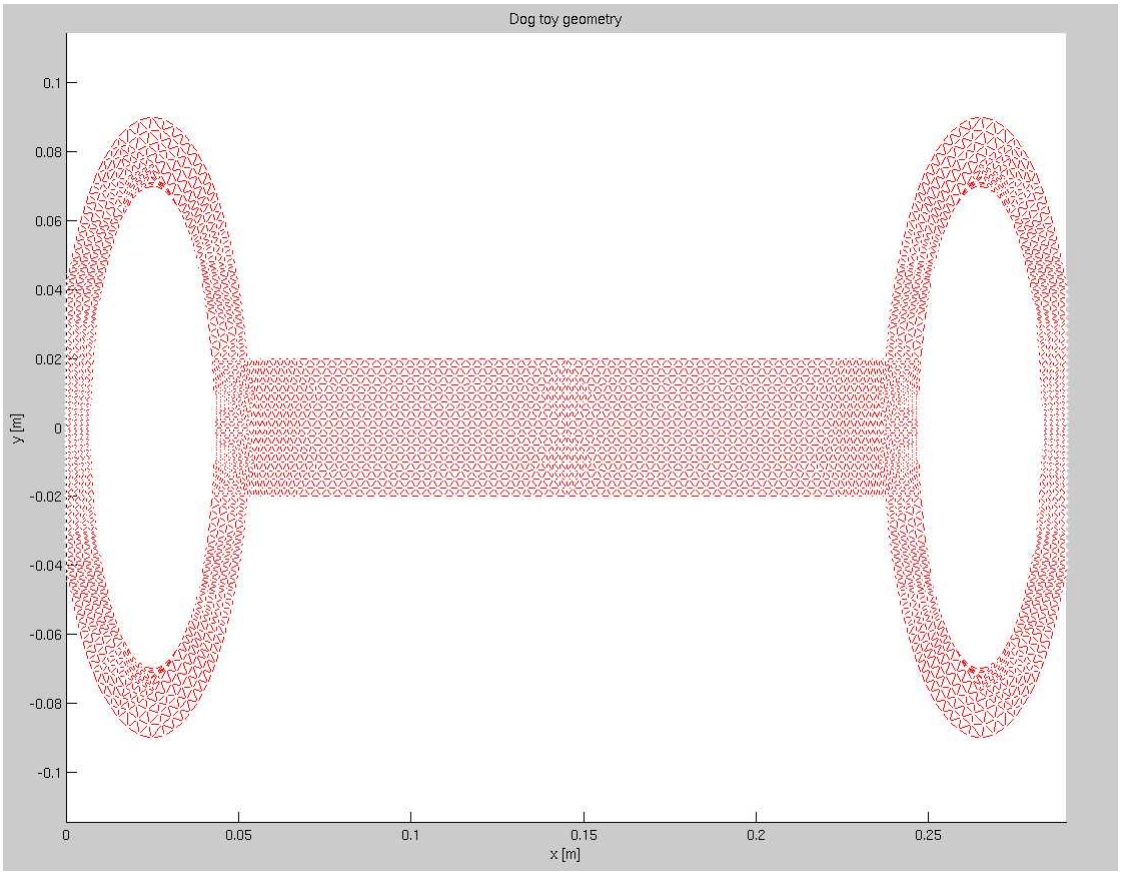


Figure 1: Dog toy geometry.

relations are derived in chapter 4.9 in the course book and are given by

$$\sigma_{kk} = 3K\epsilon_{kk} \qquad s_{ij} = 2Ge_{ij}$$

Here σ_{ij} and ϵ_{ij} denotes the stress and strain tensors, respectively. Moreover, $s_{ij} = \sigma_{ij} - 1/3\delta_{ij}\sigma_{kk}$ and $e_{ij} = \epsilon_{ij} - 1/3\delta_{ij}\epsilon_{kk}$ are the deviatoric stress and strain tensors, respectively. From experimental tests it is concluded that in the small-strain regime, the shear modulus G of the material varies with the strains according to the relationship

$$G = A\exp\left(B\tilde{J}_2\right),$$

where the invariant $\tilde{J}_2$ is given by $\tilde{J}_2 = 1/2e_{ij}\,e_{ji}$. It can be assumed that the bulk modulus $K = \frac{E}{3(1-2\nu)}$ remains constant. The initial elastic modulus of the material is $E = 10$ MPa and the Poisson's ratio is set to $\nu = 0.45$ as the material shows high resistance to volume changes. The parameter B is taken as $B = 1 \cdot 10^5$ while the value of A needs to be determined.

Instructions

The task is to solve the linear momentum equation for the toy,

$$\sigma_{ij,j} + b_i = m\ddot{u}_i$$

when 1) a force is applied at the external vertical boundaries and 2) the external vertical boundaries are given a known displacement. For simplicity static conditions are assumed and body forces are neglected. In order to solve the problem the CALFEM-toolbox should be used. In CALFEM, certain general FE-routines are already established however you need to establish extra routines in order to solve the nonlinear elastic boundary value problem.

The following steps need to be complete in order to pass this computer lab

a) Generate a mesh with `pdetool`. Instructions for this are found in appendix B.

b) Derive expressions for the stress σ_{ij} and the tangent operator D^t_{ijkl}. Some hints are found in appendix A.

c) Compare your expression for the tangent with the linear expression in eq. (4.89) in the course book. The tangents should coincide for zero strain. What is the value of A? Use this value for the parameter.

d) Write element routines `my_stress.m` and `my_tangent.m` that calculate the stress and tangent. See appended manuals in appendix A for the structure of the functions.

e) Construct a FE-program using a force controlled Newton-Raphson algorithm to solve the global equilibrium equation.

f) Plot the force-displacement curve, of `plot_dof`, for force controlled loading. Load the structure so that the final applied load is approximately 30kPa.

g) Plot the force-displacement curve, of `plot_dof`, during a load cycle (loading and unloading) for displacement controlled loading. Load the structure up to 3 mm and back.

h) Plot the von Mises effective stress field for force controlled and displacement controlled loading. The following code can be made to extract the element effective stress to nodes to be able to plot them using MATLAB plot command `fill`:

```
for node = 1:nnod
    [c0,c1] = find(enod==node);
    eff_node(node,n) = sum(vMises(c0)/size(c0,1));
end
```

where `vMises` is the effective element stresses.

Three node triangular elements should be used in the finite element calculations. The calculations should be done for plane strain condition (for simplicity).
For the global equilibrium loop the Newton-Raphson method should be adopted. An appropriate amount of Newton-Raphson iterations is around 2-6 in each load step. The stress and tangent stiffness routines should be contained in separate Matlab functions so that they can easily be reused later on.
After step **c)** is completed it is recommended that you consult a teacher assistant before you proceed to program the element routines.

Preparations

Before the lab you should;

- Read through the lab instructions.
- Read pages 423-445 in the course book.
- Generate a mesh i.e., step **a)**. Instructions for this are found in appendix B. In order to check that your geometry is correct it should be plotted using the CALFEM routine `eldraw2`;

  ```
  eldraw2(ex,ey,[1 2 0],1:nbr_elem)
  ```

- *On only a few pages* present the derivations of the stress and tangential stiffness tensors as well as the value of the parameter A i.e., **b)** and **c)**. Present each component of the tangent stiffness tensor in the following format;

$$\mathbf{D^t} = \begin{bmatrix} D_{1111} & D_{1122} & D_{1112} \\ D_{2211} & D_{2222} & D_{2212} \\ D_{1211} & D_{1222} & D_{1212} \end{bmatrix}$$

The derivations should be approved by the lab tutor *before* the computer session.

Implementation

After starting MATLAB, begin with adding CALFEM into your MATLAB library using the `editpath` command. Write the two functions `my_stress.m` and `my_tangent.m`. All MATLAB functions begin with the following structure; `function [output] = function name (input)`. See the manuals for the specific inputs and outputs to use for the the two routines. Check your element routines by loading the file `checkroutines.mat`. This file contain an example of a strain tensor `eps` and the corresponding stress `sigma` and tangential stiffness matrix `Dt`. Use the strain tensor together with the correct material parameters

as input to your routines and check that they agree with the stress and stiffness matrices supplied by the file.

Main program

When writing the main program use the structure for the Newton-Raphson algorithm sketched in Tab. 1 and follow the instructions given for the program segments `a)-l)`.

Description of program structure

`a)` Define your material parameters and an element property vector `ep` that is needed for some of the CALFEM routines.

`ep = [ptype t]` where;

`ptype =` analyse type (ptype=2 for plane strain) and

`t =` thickness of the specimen.

Additional quantities, like the size of the tolerance required for the Newton-Rapshon loop and number of load steps should also be set here.

`b)` Initiate the following quantities;

```
a = zeros(nbr_dof,1);          %Displacement vector
K = zeros(nbr_dof);            %Stiffness matrix
f = zeros(nbr_dof,1);          %External force vector
f_int = zeros(nbr_dof,1);      %Internal force vector
eps = zeros(nbr_elem,3);       %Element strain matrix
```

`c)` Calculate an initial elastic stiffness matrix and tangential stiffness matrix using the CALFEM routines `hooke` and `plante` respectively. See hint for assembling the stiffness matrix in step `j)`.

`d)` For each load step, apply the incremental load. In the case of force-controlled loading, add a incremental external force `df` to the total external force `f` in each load step. In case of displacement-controlled loading the load is controlled by the boundary conditions given in `bc`.

`e)` Calculate the residual used for checking convergence of the iterations.

`f)` Calculate the incremental displacements from the linearised residual using the CALFEM routine `solveq`.

`g)` Update the strains using the CALFEM routine `plants`.

`h)` Calculate stresses and tangential stiffness matrix using your own routines `my_stress` and `my_tangent`.

i) Calculate internal forces and stiffness matrix for each element using the CALFEM routines `plantf` (note the orientation of `fe` retrieved from `plantf`) and `plante` respectively. **Don't forget to reset the global matrices in each iteration before assembling them again**, otherwise you will just continue adding numbers in an old matrix/vector.

j) Assemble element matrices to global matrices using CALFEMs `assem` routine or add the following lines inside your element-loop;

```
indx = edof(j,2:end);
K(indx,indx)=K(indx,indx)+Ke; f_int(indx)=f_int(indx)+fe;
```

k) Update residual, **set residual to zero at nodes constrained by Dirichlet boundary conditions**.

l) Reset the second column of `bc`, you only want to move one displacement increment in each load step.

Finally for plotting the deformed shapes the CALFEM routine `eldisp2` is useful.

```
% a) define material parameters and
% b) initiate quantities
% c) calc. initial tangential stiffness matrix and stiffness matrix

% load-loop
for n=1:nbr_step

    % d) apply load
    % e) calc. the linearised residual

    % iteration-loop
    while res>TOL

        % f) calc. the incremental displacement

        for j=1:nbr_elem

            % g) update the strains
            % h) calc. stresses and tangential stiffness matrix
            % i) calc. element internal forces and stiffness matrix
            % j) assembly element matrices to global matrices

        end

        % k) update the residual and set residual to zero at appropriate places
        % l) set boundary conditions to zero at appropriate places

    end
end
```

Table 1: Newton-Raphson algorithm.

Appendix A

A.1 Hints

The tangential stiffness tensor is found by differentiating the stress tensor with respect to the strain tensor according to

$$D^t_{ijkl} = \frac{\partial \sigma_{ij}}{\partial \varepsilon_{kl}}$$

In performing the calculations the derivative of the second strain invariant might be useful

$$\frac{\partial \tilde{J}_2}{\partial \varepsilon_{ij}} = e_{ij}$$

Finally, when differentiating symmetric tensors – as σ_{ij} and ε_{ij} – with respect to themselves, symmetry is preserved by differentiating according to

$$\frac{\partial \varepsilon_{ij}}{\partial \varepsilon_{kl}} = \frac{1}{2}\left(\delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\right) \quad \text{and} \quad \frac{\partial \sigma_{ij}}{\partial \sigma_{kl}} = \frac{1}{2}\left(\delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\right)$$

where δ_{ij} is the Kronecker delta.

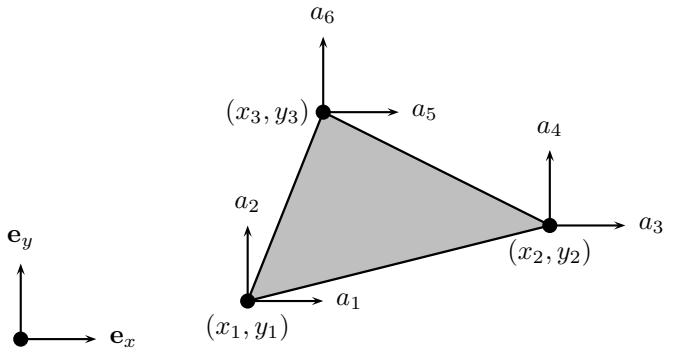
A.2 Manuals

Function:

my_stress

Purpose:

Compute the stress in a triangular 3 node element in plane strain.



Syntax:

sigma=my_stress(eps,mp)

Description:

my_stress provides the stresses σ_{11}, σ_{22}, σ_{33} and σ_{12} for a triangular 3 node element. The element strains ε_{11}, ε_{22} and $\gamma_{12} = 2\varepsilon_{12}$ are supplied by eps;

$$\text{eps} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \end{bmatrix}$$

The material parameters are defined in a vector mp. Calculate and send out all four stress components in the following format;

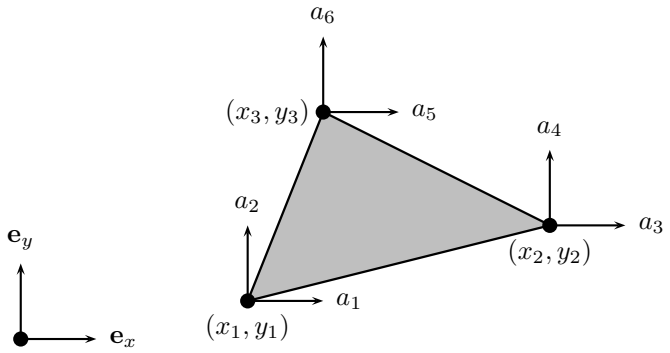
$$\text{sigma} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \end{bmatrix}$$

Function:

my_tangent

Purpose:

Compute the material tangent matrix D for a triangular 3 node element.



Syntax:

[D]=my_tangent(eps,mp)

Description:

my_tangent computes the material tangent $\mathbf{D}$ for a triangular 3 node element. Inputs to the function are the strains eps and material parameters as defined in `my_stress`.

Store the following components of the tangential stiffness matrix;

$$\mathsf{D} = \mathbf{D}_t = \begin{bmatrix} D^t_{1111} & D^t_{1122} & D^t_{1112} \\ D^t_{2211} & D^t_{2222} & D^t_{2212} \\ D^t_{1211} & D^t_{1222} & D^t_{1212} \end{bmatrix}$$

Appendix B

B.1 Generating mesh

In this assignment you should create your own mesh using the built-in PDEtool in MATLAB. 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. Due to symmetry of the toy one fourth of the bone is meshed. The segment of the dog bone geometry that should be meshed can be seen in figure B.1.

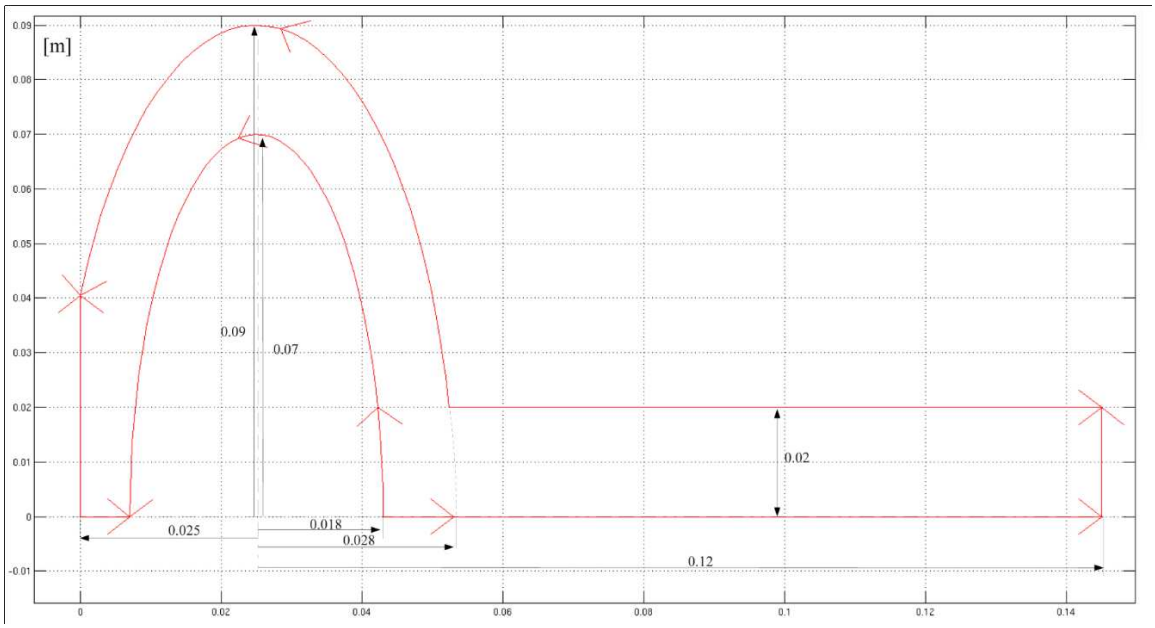


Figure B.1: Dog toy geometry.

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 B.2, note polygons are not necessary to create the geometry for the lab).

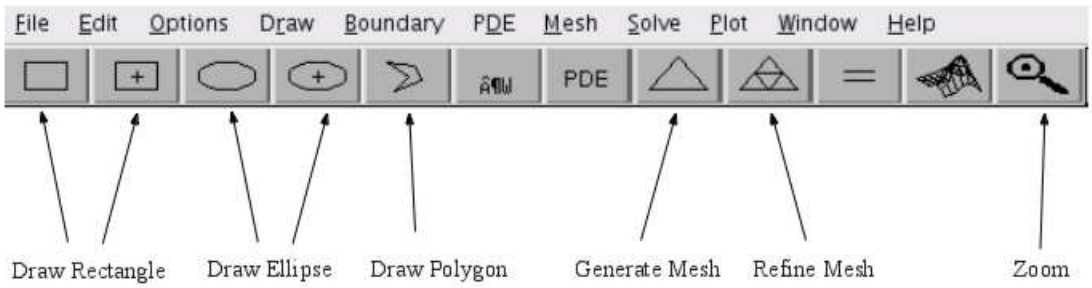


Figure B.2: Basic tools for drawing in PDEtool.

Draw a rectangle: Create a rectangle by using the rectangle tool (see figure B.2) 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 B.3.

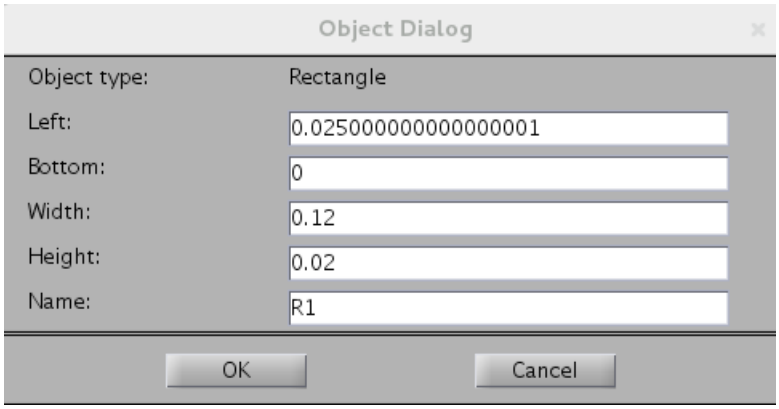


Figure B.3: Rectangle object dialog.

Draw an ellipse: Create a ellipse by using the draw ellipse tool (see figure B.2) 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 B.4, B.6 and B.5). Note that to see the resulting geometry we need to go to either meshing or boundary view (see toolbar).

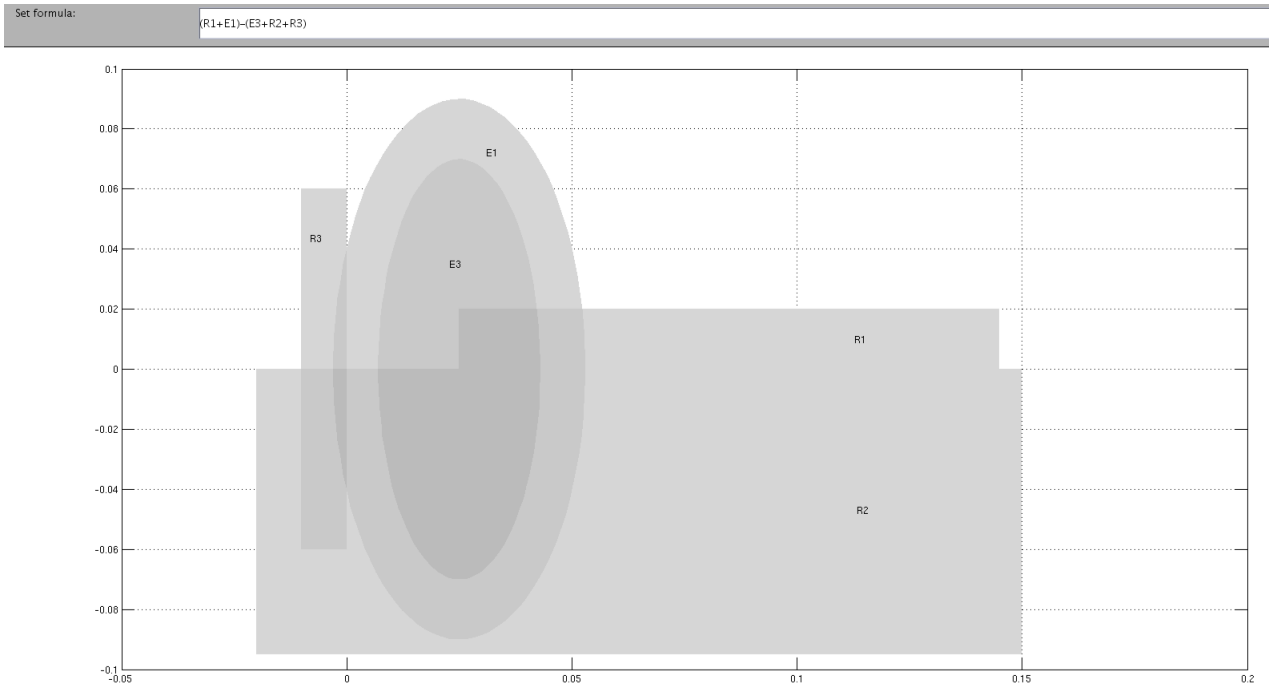


Figure B.4: Set of combined shapes.

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>.

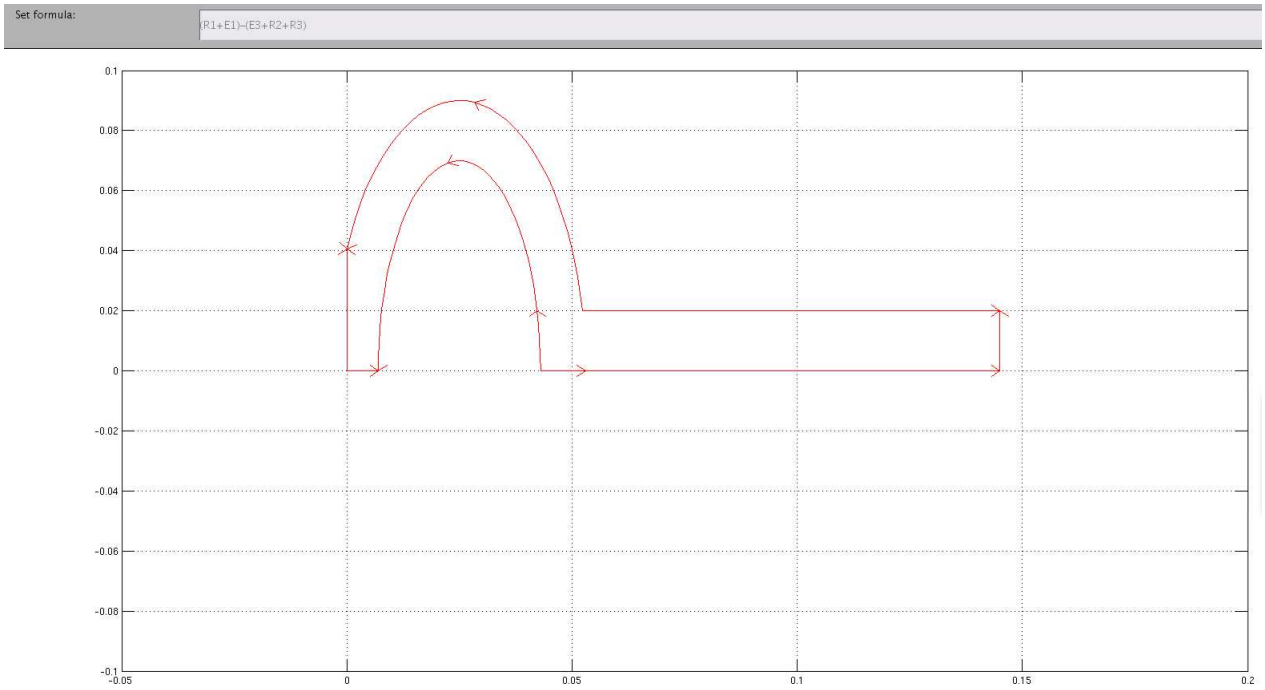


Figure B.5: Resulting geometry seen in boundary mode (ctrl+B) after all subdomains have been removed.

To obtain the geometry provided in figure B.5 the formula used is given in figure B.6 and the object names is seen in figure B.4.

Figure B.6: Formula determining active areas.

To generate a mesh simply press mesh tool when the geometry seen in boundary mode is satisfactory (see figure B.2). 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). To save time in the lab do not use too many elements.

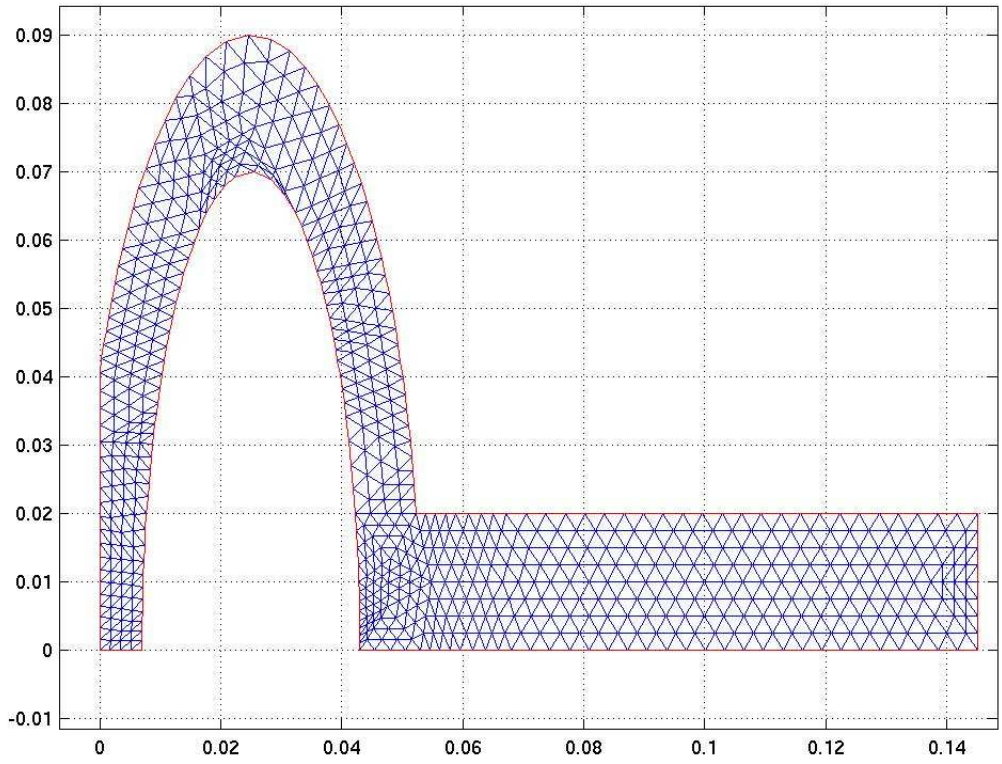


Figure B.7: Mesh after two refinements.

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 during the lab. From `p,e,t` the traditional CALFEM quantities `edof`, `dof`, `coord` etc can be extracted. On the course web page for the finite element course, instructions of how this is done may be found in the FAQs, however to save time a function called `TopConstMod.m` has been created to extract the relevant quantities for the lab and assignment. The syntax of `TopConstMod.m` is given by

```
[bc,df,edof,dof,coord,enod,plot_dof] = TopConstMod(p,t,dtau_x,du,th,control)
```

Variable	Description	Size
`bc`	Dirichlet boundary conditions	
`coord`	Coordinates of nodes	[`nbr_node`×2]
`dof`	Degrees of freedom	[`nbr_node`×2]
`edof`	Element topology matrix	[`nbr_elem`×7]
`ex`	Element x-coordinates	[`nbr_elem`×3]
`ey`	Element y-coordinates	[`nbr_elem`×3]
`df`	External force increment vector	[`nbr_dof`×1]
`plot_dof`	Degree of freedom used for plotting	
`dtau_x`	Incremental traction stress in x-dir	
`du`	Incremental displacement in x-dir	
`th`	Thickness	
`control`	0 = force controlled, 1 = if displacement controlled	

Table B.1: Nomenclature.