Computation of heat and stress distribution in lens system using the Finite Element Method

Matti Ågren, Nils Hallerfelt

May 2022

1 Introduction

In this report the aim is to model the heat development and subsequent material stress in an Iphone lens system that is exposed to sunlight. The lens system is illustrated in figure 1 and is made up of glass lenses and a plastic casing of polymethyl methacrylate (PMMA). All material parameters are shown in table 1. The lens system is modeled in two dimensions with the thickness 5 mm. Only the lenses are approximated to heat up in the sun and these are therefore modeled to have an internal heat source $Q = 3 \cdot 10^6$ Wm^{-3}. Additionally, the entire lens system has surface convection to the surrounding air which is assumed to be at constant 20°C. This convection is modeled as Newton convection $q_s = a_c(T - T_\infty)$, $a_c = 100$ W / (m^2 K). The initial temperature of the lens system is also 20°C and the lens system is assumed to be isolated at the border $\mathcal{L}$. For the mechanical part of the problem plane stress conditions are assumed to hold. Also, two forms of boundary conditions are used. Firstly, where the nodes are fixated along $\mathcal{L}$ and secondly where the fixture is approximated with artificial springs modeled as $\bar{t} = -k_{spring}\overline{u}$, where $\overline{u}$ is the deformation. Since the lens system is symmetric along the x-axis in figure 1, the problem is only solved for the top half of the lens system and the solution is later mirrored to include the bottom half.

Table 1: Material paramters for PMMA and glass taken from the project manual.

Material parameter	PMMA	Glass
Young's modulus, E [GPa]	2.8	67
Poisson's ratio, ν [-]	0.35	0.2
Expansion coefficient, α [1 / K]	$70 \cdot 10^{-6}$	$7 \cdot 10^{-6}$
Density, ρ[kg / m^3]	1185	3860
Specific heat, c_p [J / (kg K)]	1466	670
Thermal conductivity, k [W / (m K)]	2.8	0.8

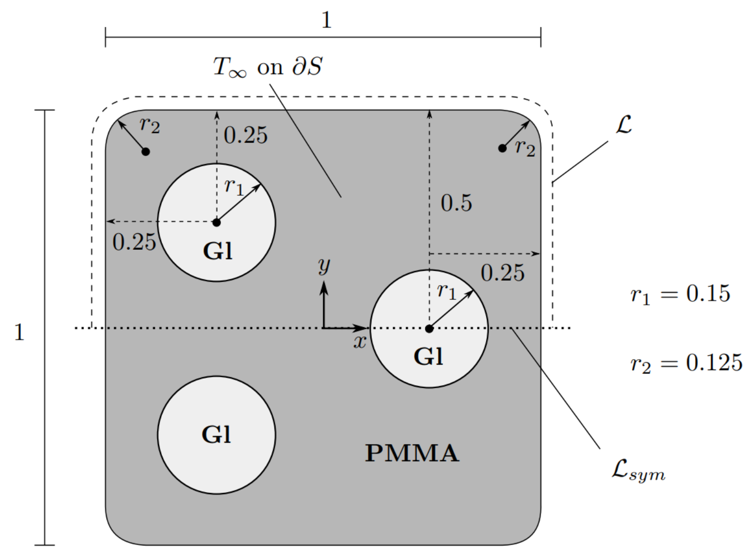


Figure 1: Sketch of the lens system taken from the project manual.

2 Procedure

2.1 Stationary heat distribution

In this section the aim is to solve the stationary heat distribution. The strong and weak form of the heat equation are the following[1].

$$div(t \cdot \overline{q}) = Q \cdot t - q_s \tag{1}$$
$$\int_S (\nabla v)^T \overline{q} dA = \oint_{\mathcal{L}} vt\overline{q}^T \overline{n} d\mathcal{L} - \int_S v \cdot Q \cdot tdA + \int_S v \cdot q_s dA \tag{2}$$

Here q is the heat flux, Q is the internal production, t the thickness, q_s is the surface convection, v is an arbitrary weight function and S is the area where the heat equation applies. Applying the Fourier equation $\overline{q} = -k\nabla T$ and Newton convection for the surface convection term $q_s = a_c(T - T_\infty)$ and moving all terms with T to one side of the equation results in the following.

$$\int_S ((\nabla v)^T k \nabla T + v \cdot a_c T) dA = \int_S v \cdot Q \cdot tdA + \int_S v \cdot a_c T_\infty \cdot dA - \oint_{\mathcal{L}} vt\overline{q}^T \overline{n} d\mathcal{L}$$

Now it is possible to apply the finite element formulation of the problem. The temperature T is approximated by $N\overline{a}$ where N are the global form functions and $\overline{a}$ the node temperatures. Since the node temperatures in this case are geometry independent $\nabla T = \nabla N \cdot \overline{a} = B\overline{a}$. Similarly, using Galerkins choice of weight function, v is chosen as $N\overline{c}$ and $\nabla v = \nabla N \cdot \overline{c} = B\overline{c}$. Applying this to the weak form results in the following.

$$\int_S (B\overline{c})^T k B\overline{a} + N\overline{c} \cdot a_c N\overline{a}) dA = \int_S N\overline{c} \cdot Q \cdot tdA + \int_S N\overline{c} \cdot a_c T_\infty dA - \oint_{\mathcal{L}} N\overline{c} t\overline{q}^T \overline{n} d\mathcal{L}$$

$$\Longleftrightarrow \overline{c}^T(\int_S (B^T kB\overline{a} + N^T \cdot a_c N\overline{a})dA - \int_S N^T \cdot Q \cdot tdA - \int_S N^T \cdot a_c T_\infty dA + \oint_\mathcal{L} N^T t\overline{q}^T \overline{n} d\mathcal{L}) = 0$$

Where it has been used that $N\overline{c} = \overline{c}^T N^T$. Since $\overline{c}$ is arbitrary this is equivalent to the following.

$$\int_S (B^T kB\overline{a} + N^T a_c N\overline{a})dA - \int_S N^T \cdot Q \cdot tdA - \int_S N^T a_c T_\infty dA + \oint_\mathcal{L} N^T t\overline{q}^T \overline{n} d\mathcal{L} = \overline{0}$$

After rearranging the terms the finite element formulation has been achieved.

$$\int_S (B^T kB + N^T \cdot a_c N)dA \cdot \overline{a} = \int_S N^T \cdot Q \cdot tdA + \int_S N^T \cdot a_c T_\infty dA - \oint_\mathcal{L} N^T t\overline{q}^T \overline{n} d\mathcal{L} \tag{3}$$

$$\Longleftrightarrow K\overline{a} = f_l + f_b$$

Here K is called the global stiffnes matrix and $f_l + f_b$ the global force vector. In this model it is approximated that the lens system is isolated at $\mathcal{L}$. This means that $\overline{q} \cdot \overline{n} = 0 \Longrightarrow f_b = 0$.

2.2 Transient heat distribution

For the transient heat distribution there appears another term in the strong and weak form of the heat equation. They now appear as follows.

$$t\rho c\dot{T} + div(t \cdot \overline{q}) = Q \cdot t - q_s \tag{4}$$

$$\int_S vt\rho c\dot{T} dA + \int_S (\nabla v)^T \overline{q} dA = \oint_\mathcal{L} vt\overline{q}^T \overline{n} d\mathcal{L} - \int_S v \cdot Q \cdot tdA + \int_S v \cdot q_s dA \tag{5}$$

For the finite element formulation the same approach is used as for the stationary heat distribution but with the addition of the time derivative $\dot{T}$. Since the form functions are constant in time $\dot{T} = (\dot{N\overline{a}}) = N \cdot \dot{\overline{a}}$.

$$\int_S N^T t\rho cN dA \cdot \dot{\overline{a}} + \int_S (B^T kB + N^T a_c N)dA \cdot \overline{a} = \int_S N^T \cdot Q \cdot tdA + \int_S N^T \cdot a_c T_\infty dA - \oint_\mathcal{L} N^T t\overline{q}^T \overline{n} d\mathcal{L} \tag{6}$$

$$\Longleftrightarrow C\dot{\overline{a}} + K\overline{a} = f_l + f_b = f_l$$

To compute how the heat distributions evolves over time the implicit euler method for time stepping is applied. This approximates the time derivative as follows where n is a certain point in time and Δ t is the length of one step in time.

$$C\frac{\overline{a}_{n+1}-\overline{a}_n}{\Delta t}+K\overline{a}_{n+1}=f_l \iff \overline{a}_{n+1}=(C+K)^{-1}(C\overline{a}_n+\Delta t f_l) \tag{7}$$

Utilising this formula the time evolution from the starting point $T_0 = 20°C$ can be calculated.

2.3 Mechanical problem - Fixated boundary

The aim is now to solve the mechanical problem. The problem has been modeled such that plane stress condition holds, which means that the only non-zero stresses are σ_{xx}, σ_{yy} and σ_{xy}. In this section the boundary condition is set so that all nodes are fixated along the outer border $\mathcal{L}$. The strong form of the differential equations of equilibrium is the following[2]:

$$\tilde{\nabla}^T\overline{\sigma}+\overline{b}=\overline{0}$$

where the operator $\tilde{\nabla}$, stresses $\overline{\sigma}$, and the internal body forces $\overline{b}$ are defined as follows (in two dimensions):

$$\tilde{\nabla}^T=\begin{bmatrix}\frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial x}\\ 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial y}\end{bmatrix} \quad \overline{\sigma}=\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{xy}\end{bmatrix} \quad \overline{b}=\begin{bmatrix}b_x\\ b_y\end{bmatrix}.$$

It is important to notice that this is now a vector equation, unlike the strong form of the heat problems. The weak form of the problem can now be formulated as: [3]

$$\int_A(\tilde{\nabla}\overline{v})^T\overline{\sigma}dA=\oint_{\mathcal{L}}\overline{v}^T\overline{t}d\mathcal{L}+\int_A\overline{v}^T\overline{b}dA \tag{8}$$

where the thickness t has been omitted as it is constant throughout the body. As one can see from 8, the traction vector $\overline{t}$ has appeared. The traction vector is the stress at the boundary in the direction of the normal to the boundary.

From the weak form it is possible to make the FE formulation of the problem. The displacement vector $\overline{u} = [u_x u_y]^T$ is to be approximated by $\overline{u} = N\overline{a}$. By using the Galerkin method for the weight function v:

$$\overline{v}=N\overline{c} \implies \overline{v}^T=\overline{c}^TB^T \tag{9}$$
$$\tilde{\nabla}\overline{v}=\tilde{\nabla}N\overline{c}=B\overline{c} \implies (\tilde{\nabla}\overline{v})^T=\overline{c}^TB^T \tag{10}$$

By inserting 9 and 10 into 8 the following is obtained.

$$\overline{c}^T\left(\int_A B^T\overline{\sigma}tdA-\oint_{\mathcal{L}}N^T\overline{t}td\mathcal{L}-\int_A N^T\overline{b}tdA\right)=0 \tag{11}$$

As $\overline{v}$ is arbitrary, $\overline{c}$ must also be arbitrary. The only vector which is orthogonal to any arbitrary vector is the nullvector $\overline{0}$, implying that the parenthesis in 11 must be identical to the nullvector for the equality to hold, which leads to the following result:

$$\int_A B^T \overline{\sigma} t dA - \oint_{\mathcal{L}} N^T \overline{t} t d\mathcal{L} - \int_A N^T \overline{b} t dA = \overline{0} \tag{12}$$

The constitutive relation for the material behaviour, assuming linear elasticity and small deformations, can now be introduced using Hooke's generalized law to obtain an expression for $\overline{\sigma}$:

$$\overline{\sigma} = D\overline{\epsilon} - D\overline{\epsilon}_0 \quad \overline{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} \quad \overline{\epsilon} = \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix} \tag{13}$$

where $\overline{\epsilon}_0$ is the initial strain due to temperature changes (thus this is assumed to be known as the change in temperature is already known). The displacements $\overline{u}$ can be derived by the kinematic relation between the strains $\overline{\epsilon}$ and the displacement $\overline{u}$:

$$\overline{\epsilon} = \tilde{\nabla} \overline{u}$$

which together with the displacement approximation $\overline{u} = N\overline{a}$ yields:

$$\overline{\epsilon} = B\overline{a} \tag{14}$$

By inserting 14 into the constitutive relation given by 13

$$\overline{\sigma} = DB\overline{a} - D\overline{\epsilon}_0 \tag{15}$$

and furthermore insert 15 into 12, we finally arrive at the FE formulation of the problem:

$$\left(\int_A B^T DB dA \right) \overline{a} = \oint_{\mathcal{L}} N^T \overline{t} d\mathcal{L} + \int_A N^T \overline{b} dA + \int_A B^T D \overline{\epsilon}_0 dA \tag{16}$$

$$K\overline{a} = \overline{f}_b + \overline{f}_l + \overline{f}_0 \tag{17}$$

where K is the global stiffness matrix, $\overline{a}$ the node strains, $\overline{f}_b$ the boundary vector, $\overline{f}_l$ the load vector, and $\overline{f}_0$ the initial strain vector. As the FE formulation is obtained, an interpretation of the boundary vector $\overline{f}_b$, the load vector $\overline{f}_l$, and the initial strain vector $\overline{f}_0$ for the given problem will be given.

2.3.1 Boundary Vector

As mentioned in the beginning of this section the boundary condition is that there are no deformations on the outer boundary, i.e. the nodes along $\mathcal{L}$ in figure 1 are completely fixated. Because of symmetry its also possible to set a boundary condition on the $\mathcal{L}_{sym}$ boundary as well. As a result of the symmetry in the y-direction, the nodes on $\mathcal{L}_{sym}$ will be fixated in the y-direction. However, in the x-direction the nodes are not fixated and allowed to move. One gets the following, where the boundary has been separated into the different segments $\mathcal{L}$ and $\mathcal{L}_sym$

$$\overline{f}_b = \oint_{\mathcal{L}} N^T \overline{t} d\mathcal{L} = \int_{\mathcal{L}} N^T \overline{h} d\mathcal{L} + \int_{\mathcal{L}_{sym}} N^T \overline{t} d\mathcal{L}$$
$$\overline{u} = \overline{0} \quad \text{on } \mathcal{L}$$
$$\overline{u}_x = \overline{0} \quad \text{on } \mathcal{L}_{sym}.$$

2.3.2 Load Vector

The load vector constitutes the sum of all infinitely small body forces with regards to the form functions N^T. An example of such a force would be gravitational force. As the gravitational force is in the z-direction in the stated problem, and since plane stress is assumed to hold, we can set the gravitational force to zero. No other body forces are present, and thus the load vector can be set to zero.

2.3.3 Initial Strain Vector

The initial strain vector is a force caused by the thermal expansion of the material. Assuming plane stress and isotropic materials as in the stated problem, one can quantify the $D\overline{\epsilon}_0$ vector in the following way[4]:

$$D\overline{\epsilon}_0 = \frac{\alpha E \Delta T}{1-\nu} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

where α is the materials thermal expansion coefficient, E is Young's modulus, ΔT is the change in temperature, ν is the material Poisson's ratio

2.4 Mechanical problem - Artificial spring boundary

In this section the mechanical problem is solved once again, but this time under different boundary conditions. Now the problem is modeled using "artificial springs" along the boundary $\mathcal{L}$. Thus the traction force is now defined by $\overline{t} = -k\overline{u}$ along $\mathcal{L}$, while the boundary conditions along $\mathcal{L}_{sym}$ are the same as in the previous section (i.e. the nodes are fixated in the y-direction on $\mathcal{L}_{sym}$).

Since only the boundary conditions have changed, the strong and weak form are the same as in the previous section. The FE formulation is very similar, and thus equation 16 can be used as a starting point. Using that $\overline{t} = -k\overline{u}$ along $\mathcal{L}$, the following results

$$\left(\int_A B^T DBdA\right)\overline{a} = \int_{\mathcal{L}} N^T\overline{t}d\mathcal{L} + \int_{\mathcal{L}_{sym}} N^T\overline{h}d\mathcal{L} + \int_A N^T\overline{b}dA + \int_A B^T D\overline{\epsilon}_0 dA$$

$$\overline{t} = -k\overline{u} \quad \text{on } \mathcal{L} \implies \overline{t} = -kN\overline{a} \quad \text{on } \mathcal{L}$$

where on the last line it was used that $\overline{u} = N\overline{a}$. Inserting this and doing some simple rewriting leads to the following FE formulation:

$$\left(\int_A B^T DBdA + \int_{\mathcal{L}} N^T kNd\mathcal{L}\right)\overline{a} = \int_{\mathcal{L}_{sym}} N^T\overline{h}d\mathcal{L} + \int_A N^T\overline{b}dA + \int_A B^T D\overline{\epsilon}_0 dA$$

$$(K + K_l)\overline{a} = \tilde{K}\overline{a} = \overline{f}_b + \overline{f}_l + \overline{f}_0$$

$$\overline{u}_x = \overline{0} \quad \text{on } \mathcal{L}_{sym}.$$

As one can see the stiffness matrix $\tilde{K}$ now consists of a line integral over the boundary as well as the "regular" stiffness matrix.

2.5 Implementing the finite element formulation

To use our finite element formulation a mesh of the geometry is needed. PDEtool in matlab was used to create the mesh visualised in figure 2. This mesh is exported to matlab with the node coordinates and which nodes are connected in triangles and edges.

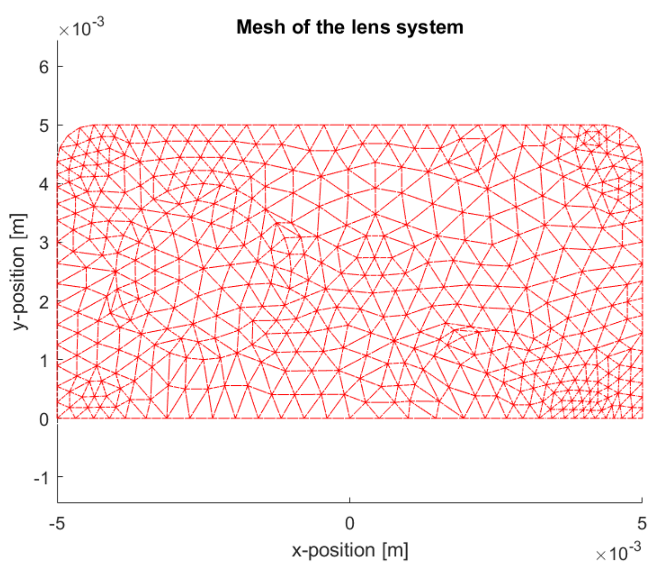


Figure 2: Mesh of the lens system created with PDEtool in matlab.

In the resulting temperature distribution each node has a specific temperature. In between the nodes the temperature is interpolated from the surrounding nodes to create a smooth transition. Each triangle in the mesh represents one element and the elements are numbered. In the temperature problem, each node has one degree of freedom, the temperature of the node, and in the mechanical problem each node has two degrees of freedom, a displacement in the x- and y-directions. Each element is connected two certain degrees of freedoms and this can be represented by the Element Degree of Freedom or edof matrix. Each row in the edof matrix specifies the element number and the connected degrees of freedom. Since the goal of the finite element method is to divide a certain geometry into smaller elements and solve a problem element by element, the edof matrix is essential. When constructing the global stiffness matrix and the global force vector this allows for constructing the element stiffness matrix and element force vector by themselves and assembling them into a global matrix and vector that is connected to the specific degrees of freedom.

To compute the calculations the CALFEM package for matlab and a matlab function plantml, which was given at the start of the assignment, was used. This contains functions for computing the element stiffnes matrices and load vectors and for assembling them into global vectors and matrices. For the heat distribution flw2te and plantml was used and for the mechanical problem the functions plante, plantf and plants was used. To assemble the element matrices and load vectors the function assem was used. With the artificial springs boundary a line integral has to be calculated. This is done with the self-implemented function THE-NILS-LINE-FUNCTION.

3 Results

The stationary temperature distribution is displayed in figure 3. When equilibrium has been reached the lens system are in a temperature range between 50 and 54 °C. The maximum temperatures are in the center of the lenses on the left hand side and the minimum temperatures are in the right hand top and bottom corners. The temperature evolution over time is displayed in the figures 4, 5 and 6. The heat can be seen spreading out from the lenses and after 400 seconds the temperature distribution is almost identical to the stationary one.

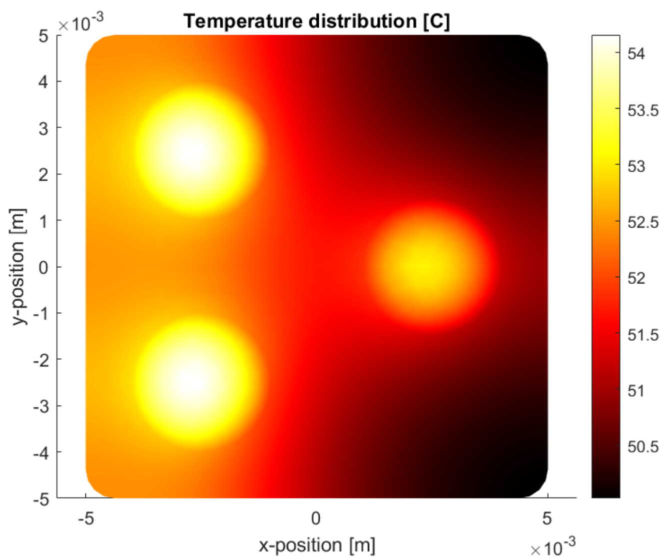
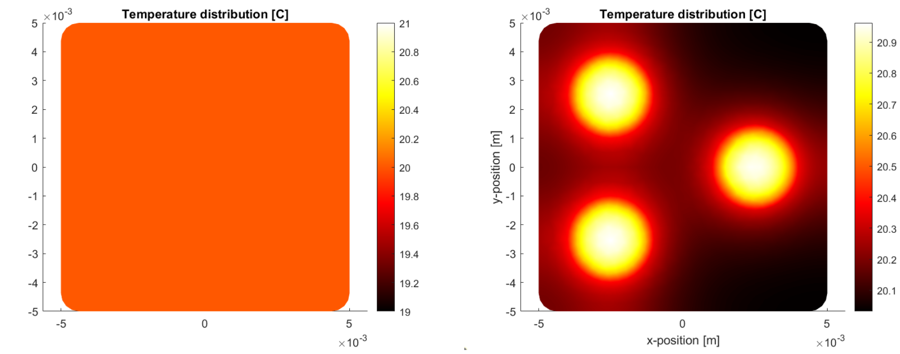
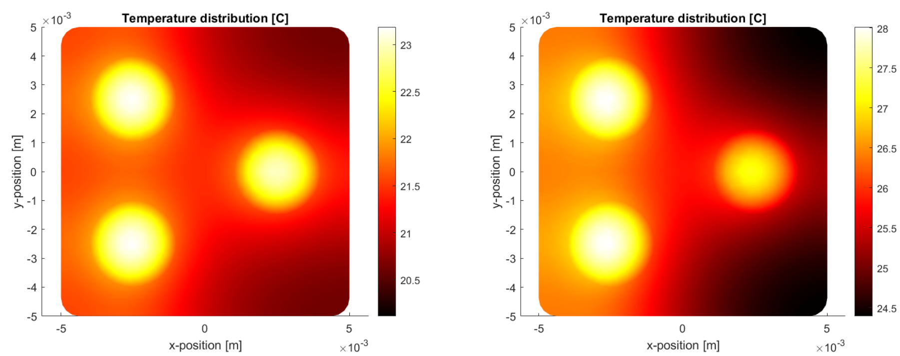


Figure 3: Plot of the stationary temperature distribution.



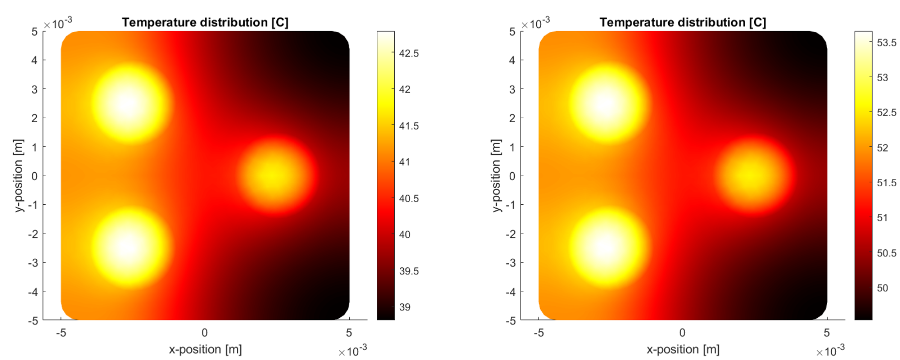
(a) (b)

Figure 4: Plot of the transient heat distribution after 0 (a) and 1 (b) seconds.



(a) (b)

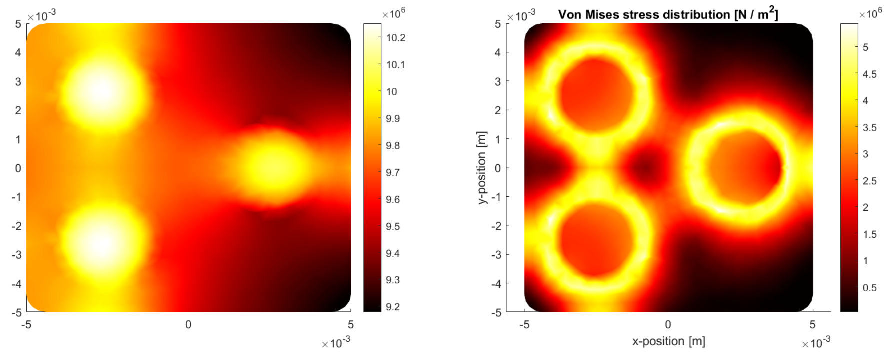
Figure 5: Plot of the transient heat distribution after 2 (a) and 10 (b) seconds.



(a) (b)

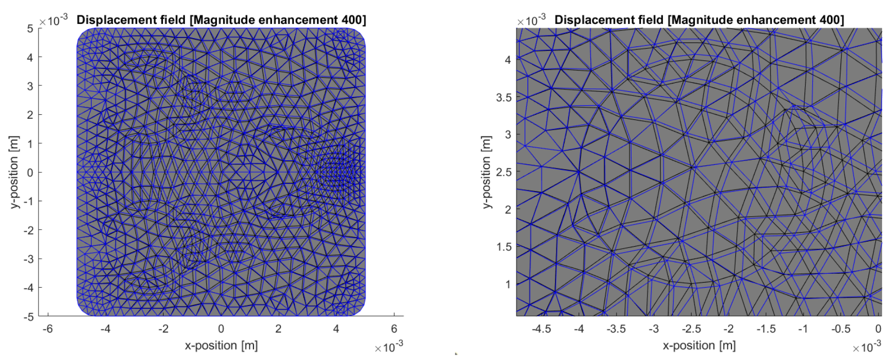
Figure 6: Plot of the transient heat distribution after 100 (a) and 400 (b) seconds.

The Von Mises stress distribution for the two different types of boundary conditions are shown in figure 7. The different cases give significantly different results. With the fixated nodes the stress is highest in the center of the glass lenses while with the artificial springs it is highest in the area where the glass and the plastic meet. Also with the artificial springs, the stress is smaller in magnitude and more concentrated to certain areas whereas with the fixated nodes the stress is quite evenly distributed throughout the lens system. The subsequent node displacements because of the thermal strains are shown in figure 8 and 9. Obviously, the artificial springs leads to the lens system expanding beyond its original edges where as the fixated nodes keeps the lens system in place. The expansion is greatest furthest away from the lenses. Note that the magnitude enhancement is 8 times bigger in the fixated nodes case than in the artificial springs case.



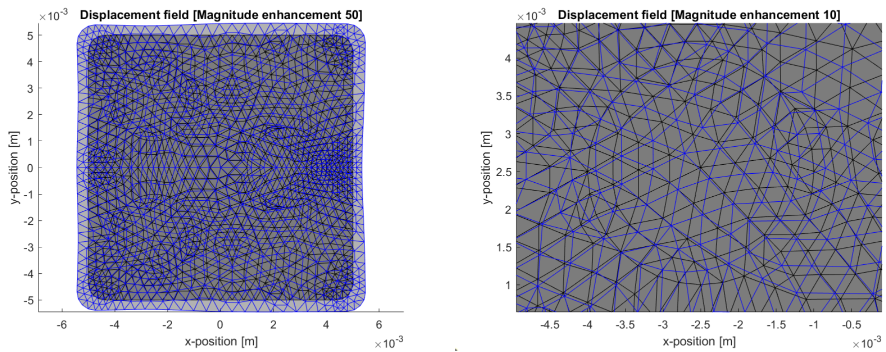
(a) (b)

Figure 7: Plot of the Von Mises stress distribution with fixated nodes (a) and artificial springs (b) as boundary conditions.



(a) (b)

Figure 8: Plot of the node displacements with fixated nodes. Subfigure a) shows the entire lens system and b) is zoomed in on the top left lens.



(a) (b)

Figure 9: Plot of the node displacements with artificial springs as boundary conditions. Subfigure a) shows the entire lens system and b) is zoomed in on the top left lens.

4 Discussion

The resulting stationary temperature distribution is a reasonable solution to the problem formulation as the temperature is highest in the lenses where the heat is emanating from and coldest in the corners the furthest away from the heat source. It is difficult to tell how well the calculated temperatures approximate reality as it all depends on how well the Q and q_s values approximate actual conditions, but the results are in a similar temperature range as to what one might experience on a warm day. Much of the same goes for the transient heat distribution. That it takes 400 seconds for the heat production and convection in the lens system to reach equilibrium is neither unreasonably long nor short. A further validation of the solution is that the heat seems to spread out evenly when the heat source is ”turned on”.

Changing the boundary conditions produces a great change in the Von Mises stress distribution. With the fixated boundary conditions the stress is about twice as great and more spread out in the entire lens system compared to the artificial springs case. It makes sense that the stress is greater when the material is prohibited from expanding beyond its edges and therefore ends up pushing in on itself. The parameter that determines how much the material expands at a certain increase in temperature is the expansion coefficient α. This is 10 times greater for plastic than for glass which provides an explanation as to why the stress becomes concentrated to the glass lenses as they are inhibited from expanding by the pressure from the surrounding plastic. The same line of reasoning explains why in the artificial springs case the plastic has expanded less beyond the edges close to the lenses. This would also explain why the artificial springs lead to the stress maximising where the glass meets the plastic as this is where glass and plastic is inhibited from expanding.

This model is of course not a perfect replica of a real life lens system. Errors include assuming that the boundary of the lens system is fully isolated, and that only the lenses generate heat when exposed to sunlight. These are however, not errors in the model itself but only of its application. Since it is a finite element model, the approximation improves by using a finer mesh. Additionally, this report uses a two dimensional model. This means assuming that the the only variations in the material when it comes to for example heat or stress is in the plane. In extension, this assumes that the heat and stress transfers instantaneously through the depth of the material. This is a bad approximation as the thickness of the material is 0.5 cm which is only half of the width and height and therefore not negligible in comparison.

5 References

1. Ottosen N, Petersson H. *Introduction to the Finite Element Method*. Pearson Education Limited. Essex, England. 1992. p. 206

2. Ottosen N, Petersson H. *Introduction to the Finite Element Method*. Pearson Education Limited. Essex, England. 1992. p. 292

3. Ottosen N, Petersson H. *Introduction to the Finite Element Method*. Pearson Education Limited. Essex, England. 1992. p. 302

4. Ottosen N, Petersson H. *Introduction to the Finite Element Method*. Pearson Education Limited. Essex, England. 1992. p. 254

A Appendix

A.1

Listing 1: Main Source Code

```
%% Mesh -> Calfem
load('mesh.mat');

% Node coordinates
coord=p';
% number of freedom-degrees
ndof=max(max(t(1:3,:)));
% nodes of elements
enod=t(1:3,:)';
% number of elements
nelm=size(enod,1);
% number of nodes
nnod=size(coord,1);
% one degree of freedom for each node (temperature)
dof=(1:nnod)';
% two degrees of freedom for each node (mechanical)
dof_S=[(1:nnod)',(nnod+1:2*nnod)'];
for elnbr=1:nelm
    % edof for two dof per node
    edof_S(elnbr,:)=[elnbr dof_S(enod(elnbr,1),:),...
        dof_S(enod(elnbr,2),:),dof_S(enod(elnbr,3),:)];
    % edof for one dof per node
    edof(elnbr,:)=[elnbr,enod(elnbr,:)];
end
% element coordinates
[Ex,Ey]=coordxtr(edof,coord,(1:ndof)',3);

% find the elements in the lenses (domain 2 and 3 in geometry)
domains = t(4,:);
lens_domains = [2 3];
lens_elements = [];
for i = 1:length(domains)
    if ismember(domains(i), lens_domains)
        lens_elements = [lens_elements i];
    end
end

%% konstanter

% GENERAL CONSTANTS

% Lenssystem thickness
thick = 0.005;
% Air temperature
T_0 = 293;

% HEAT AND CONVECTION CONSTANTS

% Heat source term
Q = 3*10^6;
% Convection parameter
a_c = 100;
% Thermal conductivity in glass
k_G = 0.8;
% Thermal conductivity in plastic
k_P = 2.8;
% Desity glass
rho_G = 3860;
% Density plastic
rho_P = 1185;
% Specific heat glass
c_G = 670;
% Specific heat plastic
c_P = 1466;
% Material heat parameter for lens
Dheat_G = k_G*eye(2);
% Material heat parameter for lens
Dheat_P = k_P*eye(2);
```

```
% MECHANICAL CONSTANTS AND PARAMETERS

% Expansion coef glass
alpha_G = 7*10^(-6);
% Expansion coef plastic
alpha_P = 70*10^(-6);
% Poisson ratio glass
v_G = 0.2;
% Poisson ratio platic
v_P = 0.35;
% Youngs module Glass
E_G = 67*10^9;
% Youngs Plastic
E_P = 2.8*10^9;
% Material mechanical parameter Glass
Dmek_G = hooke(1, E_G, v_G);
% Material mechanical parameter Plastic
Dmek_P = hooke(1, E_P, v_P);
% Material expansion parameter due to heatdiff
Deps_G = alpha_G*E_G/(1-v_G) .* [1;1;0];
% Material expansion parameter due to heatdiff
Deps_P = alpha_P*E_P/(1-v_P) .* [1;1;0];
% Spring coef
k_s = 1000;

% CALFEM INPUT PARAMETERS

ep = [1 thick];
eq = [0 0];

%% Stationary temp distribution

% Global Stiffness Matrix
K_heat_stat = zeros(ndof);
% Global Force Matrix
F_heat_stat = zeros(ndof,1);

for elnr=1:nelm
    %Calculates the Global Stiffness Matrix
   if ismember(elnr, lens_elements)
   [Ke_B,fe] = flw2te(Ex(elnr,:), Ey(elnr,:),thick, Dheat_G,...
           Q + (a_c*T_0)/thick);
   else
       [Ke_B,fe] = flw2te(Ex(elnr,:), Ey(elnr,:),thick, Dheat_P,...
           a_c*T_0/thick);
   end
   % Calculates the conv part of Ke
   Ke_N = plantml(Ex(elnr, :), Ey(elnr, :), a_c);
   % Stiffness matrix for element
   Ke = Ke_B + Ke_N;
   % Assembles element into complete Stiffness Matrix
   [K_heat_stat,F_heat_stat] = assem(edof(elnr, :), K_heat_stat, Ke,...
       F_heat_stat, fe);
end

% solves the nodes temperatures for the stationary heat problem
a_heat_stat = solve(K_heat_stat,F_heat_stat);
% Extract solution according to the topology matrix defined by edof
ed_heat_stat = extract(edof, a_heat_stat);

%plot
patch([Ex; Ex]', [Ey; -Ey]', [ed_heat_stat; ed_heat_stat]');
colorbar

%% Transient v rmeledning

 % Global Stiffness Matrix
K_heat_trans = zeros(ndof);
% Global Force Matrix
F_heat_trans = zeros(ndof,1);
% Transient part of the stiffnes matrix
C = zeros(ndof);

% Time stepping parameters
tend = 30*60;
N = 30*60;
```

```
dt = tend/N;
% Initial temperature and solution matrix
a_heat_trans = zeros(ndof, N);
a_heat_trans(:, 1) = 293;

for elnr=1:nelm
   % Calculates the global stiffnes matrix and load vector
    if ismember(elnr, lens_elements)
    [Ke_B,fe] = flw2te(Ex(elnr,:), Ey(elnr,:),thick, Dheat_G,...
             Q + (a_c*T_0)/thick);
        Ce = plantml(Ex(elnr, :), Ey(elnr, :), thick*rho_G*c_G);
    else
        [Ke_B,fe] = flw2te(Ex(elnr,:), Ey(elnr,:),thick, Dheat_P,...
             a_c*T_0/thick);
        Ce = plantml(Ex(elnr, :), Ey(elnr, :), thick*rho_P*c_P);
   end
   % Calculates the conv part of Ke
   Ke_N = plantml(Ex(elnr, :), Ey(elnr, :), a_c);
   Ke = Ke_B + Ke_N;
   % Stiffness matrix for element
    [K_heat_trans,F_heat_trans] = assem(edof(elnr, :), K_heat_trans, Ke,...
    F_heat_trans, fe);
   % Assembles element into global Stiffness Matrix
   C = assem(edof(elnr, :), C, Ce);
end

% Computes the time stepping
for i = 2:N
     a_heat_trans(:, i) = (C + dt*K_heat_trans)\(C*a_heat_trans(:, i - 1)...
         + dt*F_heat_trans);
end

% Plots solution
ed_heat_trans = extract(edof, a_heat_trans(:, 1));
figure;
patch([Ex; Ex]', [Ey; -Ey]', [ed_heat_trans; ed_heat_trans]');
colorbar

ed_heat_trans = extract(edof, a_heat_trans(:, 2));
figure;
patch([Ex; Ex]', [Ey; -Ey]', [ed_heat_trans; ed_heat_trans]');
colorbar

ed_heat_trans = extract(edof, a_heat_trans(:, 5));
figure;
patch([Ex; Ex]', [Ey; -Ey]', [ed_heat_trans; ed_heat_trans]');
colorbar

ed_heat_trans = extract(edof, a_heat_trans(:, 10));
figure;
patch([Ex; Ex]', [Ey; -Ey]', [ed_heat_trans; ed_heat_trans]');
colorbar

ed_heat_trans = extract(edof, a_heat_trans(:, 20));
figure;
patch([Ex; Ex]', [Ey; -Ey]', [ed_heat_trans; ed_heat_trans]');
colorbar

ed_heat_trans = extract(edof, a_heat_trans(:, 100));
figure;
patch([Ex; Ex]', [Ey; -Ey]', [ed_heat_trans; ed_heat_trans]');
colorbar

%% Mechanics - no node displacement on boundary

% Global stiffnes matrix and load vector
K_mek_1 = zeros(ndof*2);
f0_mek_1 = zeros(ndof*2,1);

% find boundary nodes
er = e([1 2 5],:);
bound_seg = [1 2 9 14 15];
bound_seg_sym = [3 4 5 6 7 8];
nodes_bound = [];
nodes_bound_sym = [];

for i = 1:length(er)
```

```
    if ismember(er(3,i),bound_seg)
        for j = 1:2
            nodes_bound = [nodes_bound er(j,i)];
        end
    elseif ismember(er(3,i),bound_seg_sym)
        for j = 1:2
            nodes_bound_sym = [nodes_bound_sym er(j,i)];
        end
    end
end

% sets boundary condition
bc1 = [];
for i = 1:length(nodes_bound)
%0 in u_x and u_y for boundary
   bc1 = [bc1; nodes_bound(i) 0; nodes_bound(i)+nnod 0];
end
for i = 1:length(nodes_bound_sym)
%0 in u_y for boundary_sym
    bc1 = [bc1; nodes_bound_sym(i)+nnod 0];
end
bc1_unique = unique(bc1(:,1));
bc1_unique = [bc1_unique zeros(length(bc1_unique), 1)];
bc1 = bc1_unique;
clearvars bc1_unique;

for elnr=1:nelm
   % Calculate global stiffnes matrix and load vector
   nodes = enod(elnr, :);
   mean_T = (a_heat_stat(nodes(1))+a_heat_stat(nodes(2))+...
       a_heat_stat(nodes(3))) / 3 - 293;
   if ismember(elnr, lens_elements)
        Ke = plante(Ex(elnr, :), Ey(elnr, :), ep, Dmek_G);
        f0e = plantf(Ex(elnr, :), Ey(elnr, :), ep, (Deps_G*mean_T)');
   else
        Ke = plante(Ex(elnr, :), Ey(elnr, :), ep, Dmek_P);
        f0e = plantf(Ex(elnr, :), Ey(elnr, :), ep, (Deps_P*mean_T)');
   end
   % Assembles the global stiffnes matrix and load vector
   K_mek_1 = assem(edof_S(elnr, :), K_mek_1, Ke);
   f0_mek_1 = insert(edof_S(elnr, :), f0_mek_1, f0e);
end

% Solution vector for the fixed boundary conditions
a_mek_1 = solve(K_mek_1,f0_mek_1, bc1);

% Calculate von mises stress for each element
Seff_el1 = zeros(elnr, 1);
for elnr=1:nelm
   nodes = enod(elnr, :);
   mean_T = (a_heat_stat(nodes(1))+a_heat_stat(nodes(2))+a_heat_stat(nodes(3))) / 3 - 293;
   if ismember(elnr, lens_elements)
        [es,et] = plants(Ex(elnr, :), Ey(elnr, :), ep, Dmek_G, ...
            extract(edof_S(elnr, :), a_mek_1));
        es = es - Deps_G'*mean_T;
   else
        [es,et] = plants(Ex(elnr, :), Ey(elnr, :), ep, Dmek_P, ...
            extract(edof_S(elnr, :), a_mek_1));
        es = es - Deps_P'*mean_T;
   end
   Seff_el1(elnr) = sqrt(es(1)^2 + es(2)^2 + es(3)^2 - es(1)*es(2));
end

% Calculate the von mises stress for each node
Seff_nod1 = zeros(nnod, 1);
for i = 1:nnod
    [c0,c1] = find(edof(:,2:4)==i);
    Seff_nod1(i) = sum(Seff_el1(c0))/size(c0,1);
end

% Plot the von mises stress field
ed_mek_1 = extract(edof, Seff_nod1);
figure;
patch([Ex; Ex]', [Ey; -Ey]', [ed_mek_1; ed_mek_1]')
colorbar
colormap(hot);

```

```
%% plot node displacement

mag = 100; % Magnification (due to small deformations)

edx = extract(edof, a_mek_1(1:nnod));
edy = extract(edof, a_mek_1(nnod+1:end));

% Deformed element coordinates
Exd = Ex + mag*edx;
Eyd = Ey + mag*edy;

% Plot the deformation
figure()
patch(Ex',Ey',[0 0 0],'FaceAlpha',0.3)
hold on
patch(Exd',Eyd',[0 0 0],'EdgeColor','blue','FaceAlpha',0.3)
axis equal
title('Displacement field [Magnitude enhancement 100]')

%% Mechanics - artificial springs on boundary
% Global Stiffness Matrix
K_mek_2 = zeros(ndof*2);
% Global Spring Matrix
M = zeros(ndof*2);
f0_mek_2 = zeros(ndof*2,1);




% find nodes on the symmetry boundary
er = e([1 2 5],:);
bound_seg_sym = [3 4 5 6 7 8];
nodes_bound_sym = [];
for i = 1:length(er)
    if ismember(er(3,i),bound_seg_sym)
        for j = 1:2
            nodes_bound_sym = [nodes_bound_sym er(j,i)];
        end
    end
end

% place boundary condition for symmetry in bc (0 in y direction)
bc2 = [];
for i = 1:length(nodes_bound_sym)
%0 in u_y for boundary_sym
    bc2 = [bc2; nodes_bound_sym(i)+nnod 0];
end
bc2_unique = unique(bc2(:,1));
bc2_unique = [bc2_unique zeros(length(bc2_unique), 1)];
bc2 = bc2_unique;
clearvars bc_unique;

% find the edges on the outer boundary
boundary_segments = [1 2 9 14 15];
edges_bound = [];
for i = 1:size(er,2)
    if ismember(er(3,i), boundary_segments)
        edges_bound = [edges_bound er(1:2,i)];
    end
end

% find what element the edge is in
nodepair_el = zeros(1, length(edges_bound));
for i = 1:length(edges_bound)
    for j = 1:length(enod)
        if ismember(edges_bound(1, i), enod(j,:))
            if ismember(edges_bound(2, i), enod(j,:))
                nodepair_el(i) = j;
            end
        end
    end
end

% calculate egdge lengths
edges_bound_L = zeros(1, length(edges_bound));
for i = 1:length(edges_bound)
```

```
    L = sqrt((coord(edges_bound(1,i),1)-coord(edges_bound(2,i),1))^2 + ...
        (coord(edges_bound(1,i),2)-coord(edges_bound(2,i),2))^2);
    edges_bound_L(i) = L;
end
% concatenate
n_n_L_el = [edges_bound; edges_bound_L; nodepair_el];
clearvars edges_bound edges_bound_L nodepair_el L

% Calculate and assemble global spring matrix
for edgnr = 1:length(n_n_L_el)
    L = n_n_L_el(3, edgnr);
    elnr = n_n_L_el(4, edgnr);
    LIA = ismember(edof(elnr, 2:end), n_n_L_el(1:2, edgnr)');
    M_e = THE_NILS_LINE_FUNCTION(LIA, L, -k_s);
    M = assem(edof_S(elnr, :), M, M_e);
end
clearvars L elnr b_nodes


for elnr=1:nelm
   % Calculate global stiffnes matrix and load vector
   nodes = enod(elnr, :);
   mean_T = (a_heat_stat(nodes(1))+a_heat_stat(nodes(2)) ...
       + a_heat_stat(nodes(3))) / 3 - 293;
   if ismember(elnr, lens_elements)
        Ke = plante(Ex(elnr, :), Ey(elnr, :), ep, Dmek_G);
        f0e = plantf(Ex(elnr, :), Ey(elnr, :), ep, (Deps_G*mean_T)');
   else
        Ke = plante(Ex(elnr, :), Ey(elnr, :), ep, Dmek_P);
        f0e = plantf(Ex(elnr, :), Ey(elnr, :), ep, (Deps_P*mean_T)');
   end
   % Assembles element into complete Stiffness Matrix
   K_mek_2 = assem(edof_S(elnr, :), K_mek_2, Ke);
   f0_mek_2 = insert(edof_S(elnr, :), f0_mek_2, f0e);
end
K_M = K_mek_2-M;

% Solve the displacements
a_mek_2 = solve(K_M, f0_mek_2, bc2);

% Calculate the von mises stress in each element
Seff_el2 = zeros(elnr, 1);
for elnr=1:nelm
   nodes = enod(elnr, :);
   mean_T = (a_heat_stat(nodes(1))+a_heat_stat(nodes(2))+...
        a_heat_stat(nodes(3))) / 3 - 293;
   if ismember(elnr, lens_elements)
        [es,et] = plants(Ex(elnr, :), Ey(elnr, :), ep, Dmek_G, ...
            extract(edof_S(elnr, :), a_mek_2));
        es = es - Deps_G'*mean_T;
   else
        [es,et] = plants(Ex(elnr,:), Ey(elnr, :), ep, Dmek_P, ...
            extract(edof_S(elnr,:), a_mek_2));
        es = es - Deps_P'*mean_T;
   end
   Seff_el2(elnr) = sqrt(es(1)^2 + es(2)^2 + 3*(es(3))^2 - es(1)*es(2));
end

% Calculate the von mises stress in each node
Seff_nod2 = zeros(nnod, 1);
for i = 1:nnod
    [c0,c1] = find(edof(:,2:4)==i);
    Seff_nod2(i) = sum(Seff_el2(c0))/size(c0,1);
end

% Plot the von mises stress field
ed_mek_2 = extract(edof, Seff_nod2);
figure;
patch([Ex; Ex]', [Ey; -Ey]', [ed_mek_2; ed_mek_2]')
colorbar;
colormap(hot);

%% plot node displacement with artificial spring

mag = 50; % Magnification (due to small deformations)

edx = extract(edof, a_mek_2(1:nnod));
```

```
edy = extract(edof, a_mek_2(nnod+1:end));

% Deformed element coordinates
Exd = Ex + mag*edx;
Eyd = Ey + mag*edy;

% Plot the deformation
figure()
patch([Ex; Ex]',[Ey; -Ey]',[0 0 0],'FaceAlpha',0.3)
hold on
patch([Exd; Exd]',[Eyd; -Eyd]',[0 0 0],'EdgeColor','blue','FaceAlpha',0.3)
axis equal
title('Displacement field [Magnitude enhancement 5]')
```

A.2

Listing 2: Element Spring Matrix Function

```
function Me = THE_NILS_LINE_FUNCTION(LIA, L, k)
% Me = plant_line(LIA,L, k)
%-------------------------------------------------------------
% PURPOSE
%  Compute the quantity: Me=k*int(N^T*N)dL
%
% INPUT:  LIA;        Node numbering of element nodes
%
%         k;          Constant
%
% OUTPUT: Me - spring element matrix :      Matrix 6 x 6
%-------------------------------------------------------------

% if node 1 in the element is not on boundary
if LIA(1) == 0
    n1n1 = 0;
    n1n2 = 0;
    n1n3 = 0;
    n2n2 = L/3;
    n2n3 = L/6;
    n3n3 = L/3;
% if node 2 in the element is not on boundary
elseif LIA(2) == 0
    n1n1 = L/3;
    n1n2 = 0;
    n1n3 = L/6;
    n2n2 = 0;
    n2n3 = 0;
    n3n3 = L/3;
% if node 3 in the element is not on boundary
elseif LIA(3) == 0
    n1n1 = L/3;
    n1n2 = L/6;
    n1n3 = 0;
    n2n2 = L/3;
    n2n3 = 0;
    n3n3 = 0;
end

Me = k * [n1n1 0 n1n2 0 n1n3 0
           0 n1n1 0 n1n2 0 n1n3
           n1n2 0 n2n2 0 n2n3 0
           0 n1n2 0 n2n2 0 n2n3
           n1n3 0 n2n3 0 n3n3 0
           0 n1n3 0 n2n3 0 n3n3];
end
```