Lund University
Faculty of Engineering (LTH)

FHLF01, FHLF10

Finite element method

Finite Element Method project

Authors:
Marcel Attar, 941127-2173
Martin Carlberg, 940822-2074

The report was handed in on: May 26, 2017

Lunds universitet
Lunds Tekniska Högskola

Contents

1 Introduction

In this years assignment we were faced with the problem of calculating the temperature as well as the induced stress and displacement of a computer component. The following geometry was presented:

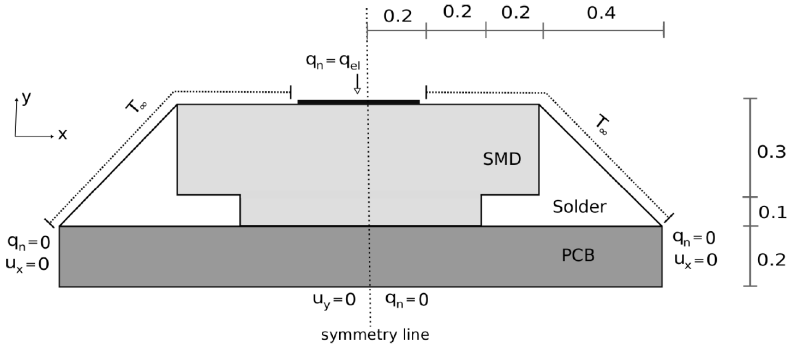


Figure 1: Sketch of the PCB, SMD and Solder that is to be analysed. (Dimensions in mm). The figure is taken from the project description.

We note from figure 1 that the problem is of two dimensions, x and y. The assignment was divided into two main parts:

- **Task 1:** Determine how the temperature distribution changes with respect to time and find the stationary temperature distribution.
- **Task 2:** Determine the effective von Mises stress field and the corresponding displacement field due to thermal expansion from the stationary temperature distribution.

Since there is a symmetry line we could performe our calculations throughout the project on just one side of the component and then mirror our solutions. We defined the boundaries of the component as are being shown in figure 2.

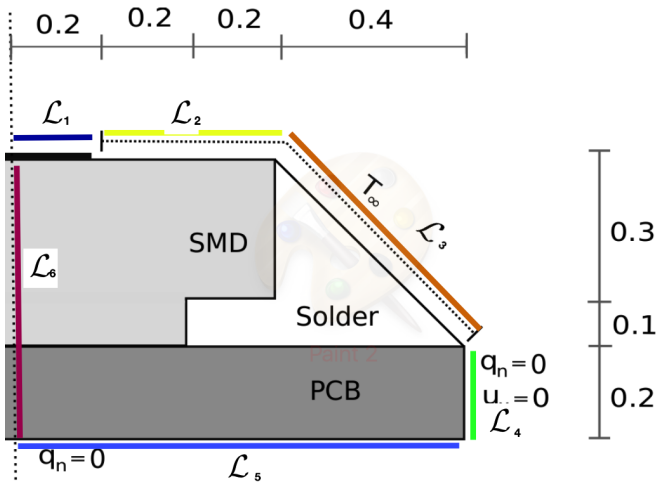


Figure 2: The definitions of the boundaries.

2 Procedure

2.1 Task 1

The first part of calculating the temperature is governed by the following differential equation [3]:

$$c\rho\dot{T} + div(\boldsymbol{q}) - Q = 0 \tag{1}$$

$\boldsymbol{q}$ is the heat flux vector, Q the heat generated inside the device, $\dot{T}$ the time derivative of the temperature, the specified heat c and the density ρ.

Deriving the FE-formulation

We multiply equation 1 with an arbitrary weight function v and integrate over the entire region. This combined with $Q = 0$ for this problem (no internal heat is generated in this problem) produces the equation

$$\int_A v(c\rho\dot{T} + div(\boldsymbol{q}))dA = 0. \tag{2}$$

Applying the Green-Gauss theorem and splitting up the integral, we attain the weak form of equation 1:

$$\int_A vc\rho\dot{T}dA + \int_{\mathcal{L}} v\boldsymbol{q}^T nd\mathcal{L} - \int_A (\nabla v)^T \boldsymbol{q}dA = 0. \tag{3}$$

We use the Galerkin method i.e $v = \boldsymbol{Nc} = \boldsymbol{c}^T\boldsymbol{N}^T$ where $\boldsymbol{c}$ is an arbitrary vector which results in

$$\boldsymbol{c}^T(\int_A \boldsymbol{N}^T c\rho\dot{T}dA + \int_{\mathcal{L}} \boldsymbol{N}^T\boldsymbol{q}^T nd\mathcal{L} - \int_A \boldsymbol{B}^T\boldsymbol{q}dA) = 0 \tag{4}$$

and since $\boldsymbol{c}^T$ is arbitrary

$$\int_A \boldsymbol{N}^T c\rho\dot{T}dA + \int_{\mathcal{L}} \boldsymbol{N}^T\boldsymbol{q}^T nd\mathcal{L} - \int_A \boldsymbol{B}^T\boldsymbol{q}dA = 0. \tag{5}$$

With the interpolation $T = \boldsymbol{Na}$ along with Fourier's law $\boldsymbol{q} = -\boldsymbol{D}\nabla T$ we end up with

$$\int_A \boldsymbol{N}^T c\rho\boldsymbol{N}dA\dot{\boldsymbol{a}} + \int_{\mathcal{L}} \boldsymbol{N}^T q_n d\mathcal{L} + \int_A \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}dA\boldsymbol{a} = 0. \tag{6}$$

which is the FE-formulation of the problem. The boundary term $\int_{\mathcal{L}} \boldsymbol{N}^T q_n d\mathcal{L}$ is zero outside the PMB and smolder sections boundaries to the space surrounding the device. Therefore, we can write

$$\int_{\mathcal{L}} \boldsymbol{N}^T q_n d\mathcal{L} = \int_{\mathcal{L}_1} \boldsymbol{N}^T q_{el} d\mathcal{L} + \int_{\mathcal{L}_2} \boldsymbol{N}^T a_c(T - T_\infty)d\mathcal{L} + \int_{\mathcal{L}_3} \boldsymbol{N}^T a_c(T - T_\infty)d\mathcal{L} \tag{7}$$

where the boundaries $\mathcal{L}_i$ are defined in figure (2). Using the same interpolation $T = \boldsymbol{Na}$, we get

$$\int_{\mathcal{L}} \boldsymbol{N}^T q_n d\mathcal{L} = \int_{\mathcal{L}_1} \boldsymbol{N}^T q_{el} d\mathcal{L} + \int_{\mathcal{L}_2} \boldsymbol{N}^T a_c(\boldsymbol{N}\boldsymbol{a} - T_\infty) d\mathcal{L} + \int_{\mathcal{L}_3} \boldsymbol{N}^T a_c(\boldsymbol{N}\boldsymbol{a} - T_\infty) d\mathcal{L}. \quad (8)$$

By substitution we get

$$\int_A \boldsymbol{N}^T c\rho \boldsymbol{N} dA \dot{\boldsymbol{a}} + \int_A \boldsymbol{B}^T \boldsymbol{D}\boldsymbol{B} dA \boldsymbol{a} = -(\int_{\mathcal{L}_1} \boldsymbol{N}^T(-q_{el}) d\mathcal{L} + \int_{\mathcal{L}_2} \boldsymbol{N}^T a_c(\boldsymbol{N}\boldsymbol{a} - T_\infty) d\mathcal{L} + \int_{\mathcal{L}_3} \boldsymbol{N}^T a_c(\boldsymbol{N}\boldsymbol{a} - T_\infty) d\mathcal{L}), \quad (9)$$

where the minus sign in front of q_{el} is caused by the negative y-direction that the heat is flowing in, which after some rearranging can be written as

$$\int_A \boldsymbol{N}^T c\rho \boldsymbol{N} dA \dot{\boldsymbol{a}} + (\int_A \boldsymbol{B}^T \boldsymbol{D}\boldsymbol{B} dA + \int_{\mathcal{L}_2} a_c \boldsymbol{N}^T \boldsymbol{N} d\mathcal{L} + \int_{\mathcal{L}_3} a_c \boldsymbol{N}^T \boldsymbol{N} d\mathcal{L}) \boldsymbol{a} = \int_{\mathcal{L}_1} \boldsymbol{N}^T q_{el} d\mathcal{L} + \int_{\mathcal{L}_2} \boldsymbol{N}^T a_c T_\infty d\mathcal{L} + \int_{\mathcal{L}_3} \boldsymbol{N}^T a_c T_\infty d\mathcal{L} \quad (10)$$

or in more compact fashion as

$$\boldsymbol{C}\dot{\boldsymbol{a}} + \boldsymbol{K}\boldsymbol{a} = \boldsymbol{f}. \quad (11)$$

The edges of the individual triangle-elements that constitute the boundary are linear. Given a parameterization of the edge with respect to a variable η, the element shape functions can be written $\boldsymbol{N} = [1 - \frac{\eta}{L}, \frac{\eta}{L}, 0]$ where L is the length of the edge. This gives us – with the detailed calculations omitted – the following expressions for single elements

$$\int_0^L \boldsymbol{N}^{\boldsymbol{e}T} q_{el} d\eta = \frac{L q_{el}}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \int_0^L \boldsymbol{N}^{\boldsymbol{e}T} a_c T_\infty d\eta = \frac{L a_c T_\infty}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \int_0^L \boldsymbol{N}^{\boldsymbol{e}T} \boldsymbol{N} d\eta = \frac{6}{L} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

(12)

The mesh

We made a sketch of the prescribed geometry in the matlab application *pdetool* and created a triangular mesh using one of it's built in functions. Since the geometry is symmetrical with respect to the line $x = 0$, we only meshed half the geometry for computational efficiency, as mentioned above. The mesh is exported in the form of three topology matrices p, e and t.

Constructing the global stiffness matrix $\boldsymbol{K}$ and the vector $\boldsymbol{f}$

To construct the global stiffness matrix, we first constructed the element stiffness matrices for all the triangles in the mesh and then added the expanded element matrices to the global one using the calfem function *assem* [2]. The element stiffness matrices were constructed by iterating over the elements, determining in which region the given element was and using the calfem function *flw2te* [2]. The contributions from the boundaries were inserted manually into the K and f matrix, using the formulae in 12.

Time stepping method for the transient solution

To determine how to temperature of the device varies with respect to time we introduced a discrete approximation of the $\dot{\boldsymbol{a}}$ vector

$$\dot{\boldsymbol{a}} = \frac{a(t_{n+1}) - a(t_n)}{t_{n+1} - t_n} \tag{13}$$

where $a(t_n$ denotes the temperature at time $t = n$. As a time stepping scheme we used the implicit θ-method (see [3] for details) which discretizes 11 into

$$C\frac{\boldsymbol{a}(t_{n+1}) - \boldsymbol{a}(t_n)}{t_{n+1} - t_n} + \theta \boldsymbol{K}\boldsymbol{a}(t_{n+1}) = \boldsymbol{f}(t_{n+1}) = \boldsymbol{f} \tag{14}$$

where the second equality is due to the time-independancy of $\boldsymbol{f}$ and $\theta = 1$. We then solved for $\boldsymbol{a(t_{t+1})}$ for every timestep, which was implemented in code as *eulerstep*.

2.2 Task 2

The second part of the assignment is based on the following differential equation:

$$\tilde{\nabla}^T \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0} \tag{15}$$

where $\boldsymbol{\sigma}$ is the stress vector, $\boldsymbol{b}$ the body force and $\tilde{\nabla}^T$ is defined as

$$\tilde{\nabla}^T = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} & 0 \\ 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial z} \\ 0 & 0 & \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix}, \quad \boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

(16)

It is important to note that this is a plane strain problem, which means that the only non zero strains are $\varepsilon_{xx}, \varepsilon_{yy}$ and γ_{xy}. We can therefore redefine the matrices and vectors above to the following

$$\tilde{\nabla}^T = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 & \frac{\partial}{\partial y} \\ 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial x} \end{bmatrix}, \quad \boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \\ 0 \end{bmatrix} \tag{17}$$

The relationship between the stresses, strains $\boldsymbol{\epsilon}$, initial strain $\boldsymbol{\epsilon_0}$ (thermal strain) and displacement $\boldsymbol{u}$ are:

$$\boldsymbol{\sigma} = \boldsymbol{D\epsilon} - \boldsymbol{D\epsilon_0}, \quad \boldsymbol{\epsilon} = \tilde{\boldsymbol{\nabla}}\boldsymbol{u}, \quad \boldsymbol{\epsilon_0} = \alpha\Delta T \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{18}$$

$\boldsymbol{D}$ is the constitutive matrix, α the thermal expansion coefficient [1/°C] and ΔT the temperature difference of the component and the "outside". The $\boldsymbol{D}$ matrix is the following for a plane strain problems, according to the Calfem manual *hooke* command [2] (assuming isotropic material)

$$\boldsymbol{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & 0 \\ 0 & 0 & 0 & \frac{1}{2}(1-2\nu) \end{bmatrix}$$

(19)

E is the Young's modulus and ν Poisson's ratio.

Deriving the FE-formulation

From the strong form in equation 15 we can derive the weak form by multiplying with an arbitrary weight function $\boldsymbol{v}^T$. We also introduce the traction vector $\boldsymbol{t}$ which will be used later.

$$\boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}, \quad \boldsymbol{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{20}$$

Now we integrate the strong form, multiplied with the weight function, over the body A (A is the components surface)

$$\int_A (\boldsymbol{v}^T \tilde{\nabla}^T \boldsymbol{\sigma} + \boldsymbol{v}^T \boldsymbol{b}) dA = \boldsymbol{0} \tag{21}$$

$$\int_A ((\tilde{\nabla}\boldsymbol{v})^T \boldsymbol{\sigma} + \boldsymbol{v}^T \boldsymbol{b}) dA = \boldsymbol{0} \tag{22}$$

From here we use Green-Gauss theorem which results in the weak formulation:

$$\int_A (\tilde{\nabla}\boldsymbol{v})^T \boldsymbol{\sigma} dA = \int_{\mathcal{L}} \boldsymbol{v}^T \boldsymbol{t} d\mathcal{L} + \int_A \boldsymbol{v}^T \boldsymbol{b} dA \tag{23}$$

Now we introduce an approximation for the displacement vector

$$\boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{24}$$

which is approximated by $\boldsymbol{u = Na}$. We also choose the weight function according to the Garlerkin method, $\boldsymbol{v = Nc}$. Here $\boldsymbol{c}$ is an arbitrary vector. It now follows that

$$\tilde{\boldsymbol{\nabla}}\boldsymbol{v} = \boldsymbol{Bc} \quad \text{where} \quad \boldsymbol{B} = \tilde{\boldsymbol{\nabla}}\boldsymbol{N} \tag{25}$$

Inserting these newly introduced variables aswell as the ones in equation 18 into our weak form in equation 23 we concluded that

$$\int_A \boldsymbol{B^T D B} dA \boldsymbol{a} = \int_{\mathcal{L}} \boldsymbol{N^T t} d\mathcal{L} + \int_A \boldsymbol{N^T b} dA + \int_A \boldsymbol{B^T D \epsilon_0} dA \tag{26}$$

which is the FE formulation of two-dimensional elasticity. In this specific task the body force vector $\boldsymbol{b}$ is set to $\boldsymbol{0}$ since there are no external forces acting on the body (e.g. gravity). The traction vector $\boldsymbol{t}$ is also $\boldsymbol{0}$. What we are left with is therefore the following

$$\int_A \boldsymbol{B^T D B} dA \boldsymbol{a} = \int_A \boldsymbol{B^T D \epsilon_0} dA \tag{27}$$

We define the following variables:

$$\boldsymbol{K} = \int_A \boldsymbol{B^T D B} dA, \quad \boldsymbol{f} = \int_A \boldsymbol{B^T D \epsilon_0} dA \tag{28}$$

We are now ready to start with coding

Compute a new Edof matrix

Our old Edof matrix which had the following appearance

$$Edof = \begin{bmatrix} el_1 & dof_1 & dof_2 & dof_3 \\ \vdots & \vdots & \vdots & \vdots \\ el_{nel} & dof_1 & dof_2 & dof_3 \end{bmatrix} \tag{29}$$

needed to be expanded to a matrix with the dimensions $nel \times 7$ since every element now has six degrees of freedom (each node has a displacement in the x and y direction). Our new Edof matrix now looks as following

$$Edof = \begin{bmatrix} el_1 & dof_1 & \dots & dof_6 \\ \vdots & \vdots & \ddots & \vdots \\ el_{nel} & dof_1 & \dots & dof_6 \end{bmatrix} \tag{30}$$

Compute the K matrix and f vector

Looking at equation 28 we see that the $\boldsymbol{K}$ matrix has a material dependence since the $\boldsymbol{D}$ matrix contains E and ν. This means that each area in the geometry (the SMD, Solder and PCB) will have a different $\boldsymbol{D}$ matrix going into our $\boldsymbol{K^e}$ matrix, we therefore needed to be aware of which region each element was in. The $\boldsymbol{D}$ matrix was calculated using the calfem command called *hooke*. The $\boldsymbol{K^e}$ matrix could then

be calculated with the calfem command *plante*. [2]

The same logic followed for the $\boldsymbol{f^e}$ vector, although here we had a slightly more difficult scenario. The temperature difference term in $\boldsymbol{\epsilon_0}$ was calculated using our stationary temperature calculated in task 1, however, we only obtained the temperature for each node and not for each element. This meant that we needed to calculate the temperature of an element using the three nodes in each element and taking the mean temperature of these. $\boldsymbol{f^e}$ could then be calculated using the calfem command *plantf*. The global stiffness matrix $\boldsymbol{K}$ and the global force vector $\boldsymbol{f}$ could be assembled by using the calfem command *assem* with the element stiffness matrix and element force vector. [2]

The displacement

Having both the $\boldsymbol{K}$ matrix and $\boldsymbol{f}$ vector meant that we were almost ready to solve for $\boldsymbol{a}$ in equation 27. To solve the equation we needed the boundary conditions. We therefore went through all our edges in the e matrix and checked which boundary they were in. All the nodes on the vertical PCB edges was assigned the value 0 in the x direction in our boundary conditions matrix that we created. The nodes on the lower horizontal PCB edge was assigned the value 0 in the y-direction.

Once we had a matrix containing the boundary values we used the *solveq* command from calfem, inserting $\boldsymbol{K}$, $\boldsymbol{f}$ and the b.c. matrix. The displacement could now be plotted.

The von Mises stress field

With the displacement of each node we could calculate the stress of each element by using the *plants* command, we got the following components for each element

$$\boldsymbol{\sigma^{e}}^T = \begin{bmatrix} \sigma^e_{xx} & \sigma^e_{yy} & \sigma^e_{zz} & \sigma^e_{xy} \end{bmatrix} \tag{31}$$

With this at our disposal we could calculate the von Mises stress for each element, using the definition below:

$$\sigma_{eff} = \sqrt{\sigma^2_{xx} + \sigma^2_{yy} + \sigma^2_{zz} - \sigma_{xx}\sigma_{yy} - \sigma_{xx}\sigma_{zz} - \sigma_{yy}\sigma_{zz} + 3\sigma^2_{xy}} \tag{32}$$

Calculating the von Mises stress for each node we used the code provided in the project description [1]. We could now plot the von Mises stress field.

3 Results

3.1 Task 1

For the stationary case, we solve the equation (11) subject to $\dot{\boldsymbol{a}} = 0$. That is, we compute $\boldsymbol{a} = \boldsymbol{K}^{-1}\boldsymbol{f}$; the results are found in figure (3). The temperature at a few different times are found in figures 4 and 5

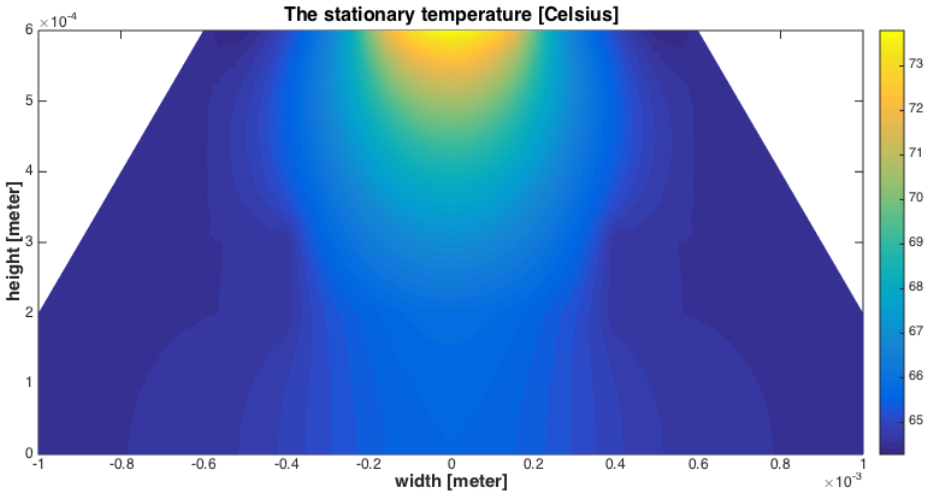
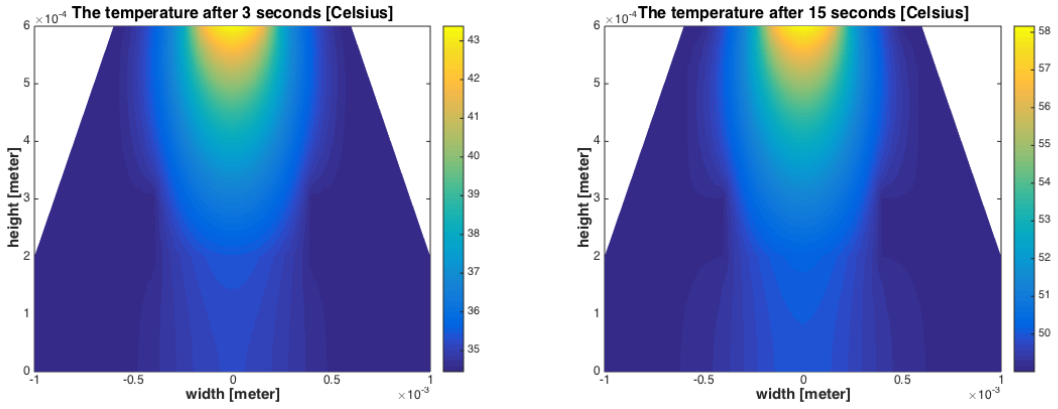
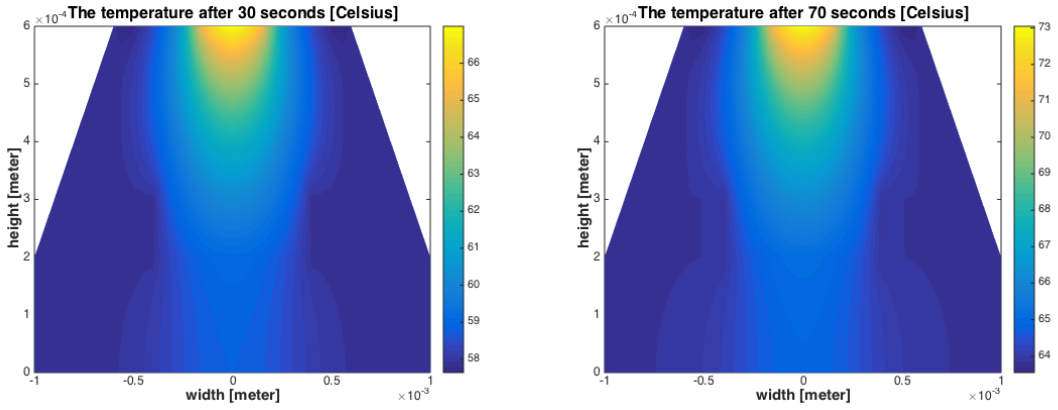


Figure 3: The stationary temperature distribution.



(a) (b)

Figure 4: The temperature after 3 and 15 seconds in the component. The initial temperature was 30 °C



(a) (b)

Figure 5: The temperature after 30 and 70 seconds in the component. As we can see the temperature converges with time towards the stationary solution seen in figure (3)

3.2 Task 2

The displacement field is found in figure (6) and coresponding von Mises stress field in figure (7)

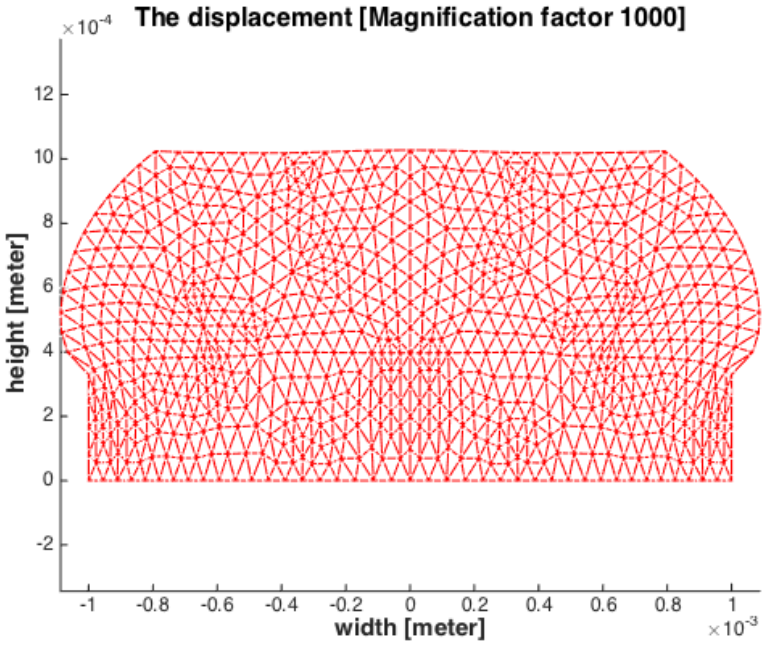


Figure 6: The displacement field caused by the stationary temperature. The height of the device before the heating was 0.6 mm.

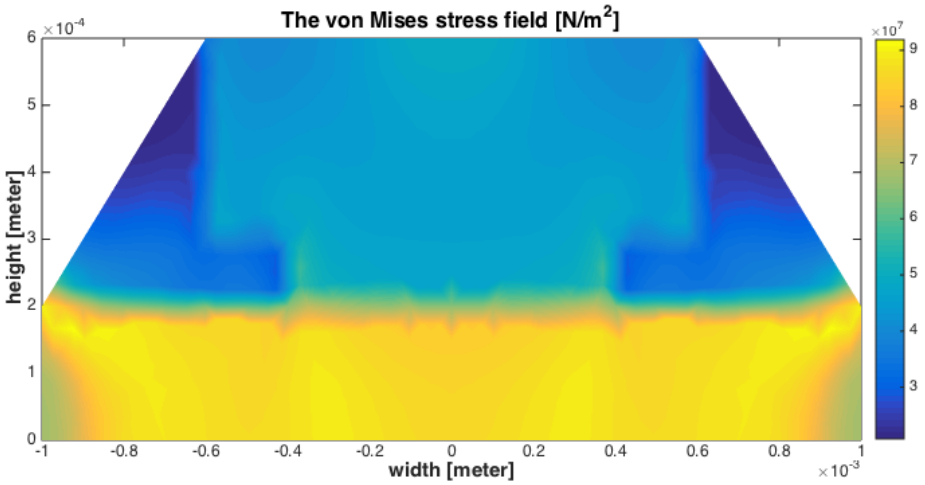


Figure 7: The von Mises stress field caused by the displacement field in figure (6).

4 Discussion

4.1 Task 1

In figure (3) we see that the stationary temperature is the highest at the top center section of the device, i.e the area directly under the boundary $\mathcal{L}_1$. This is reasonable as there is a heat-flux q_{el} at $\mathcal{L}_1$. As the thermal conductivity is higher for the Solder, the temperature will be noticeably lower since the heat will dissipate due to convection along the boundary. This can be observed in the figure.

In figure (4) and (5) we observe the time dependant temperature. There is a clear convergence pattern toward the stationary temperature, this means that the component will reach an equilibrium temperature distribution. This is reasonable as the influx is constant and the outflow due to convection is in a one-to-one correspondence with the devices temperature. Furthermore, we notice that the difference between the highest and lowest temperatures of the device remain at a relatively constant level of around 8 degrees throughout the heating process. This is because of the convections dependancy on the devices temperature.

We conclude that the high thermal conductivity and low (in relation to the other parts) specific heat capacity allows the solder part to transport heat out of the device. Since this is a computer component of some sort it is of interest to dissipate the heat from the component, therefore this is a sought after effect.

4.2 Task 2

In figure (6) we see that the device has grown in the y-direction, aswell as swollen at the solder areas. One of the reasons for the large growth in height is the constraints of growth in the x-direction at the PCB boundaries. This forces the device to grow upwards. The solder parts of the device has grown more noticeably than the other

parts. This is a reflection of the fact that the solder area has a lower elasticity modulus, and is therefore more prone to expand due to the heat.

The Poisson's ratio ν is the ratio of expansion in one axis divided by the compression of the material in the perpendicular axis. From our numerical data we see that the SMD has a quite a small ratio, meaning that when it expands in the y-direction there will be a compression force in the x-direction.

In figure (7) we see that the stress is the highest in the PCB part and the lowest in the solder part. This is to be expected, as the PCB is constrained from growth in the x-direction and negative y-direction and has a low Poisson ratio. The solder part can expand more freely and is thus not subject to the same amount of stress.

4.3 General Remarks

The accuracy of a FEM-analysis is contingent on the fineness of the mesh. However, since the device we have examined is very small, we conclude that the incremental accuracy that a finer mesh would supply would be redundant. All the results are in a reasonable order of magnitude, as far as we can tell.

5 Computer Code

5.1 Task 1

```
%% FEM projekt, loads all the numerical data aswell as the mesh.
clear all;

load t.mat
load e.mat
load p.mat

p=p/1000; % Converts to meter.

nelm=length(t(1,:));
edof(:,1)=1:nelm;
edof(:,2:4)=t(1:3,:)';
coord=p';
ndof=max(max(t(1:3,:)));
[Ex,Ey]=coordxtr(edof,coord,(1:ndof)',3);

%eldraw2(Ex,Ey,[1,4,1])
%grid on

% Numerical data ------------------------------
Esmd = 105e9;           Epcb = 105e9;           Esol = 50e9;
                T0 = 30;
vsmd = 0.118;           vpcb = 0.136;           vsol = 0.36;
                ac = 40;
ksmd = 0.29;            kpcb = 1.059;           ksol = 66.8;
                qel = 9e3;
rhosmd = 1850;          rhopcb = 1850;          rhosol = 7265;
                Tinf = 20;
csmd = 950;             cpcb = 950;             csol = 210;
alphasmd = 1.2e-5;      alphapcb = 2e-5;        alphasol = 1.2e-5;
% ---------------------------------------------
%% PART 1: Calcualtes the K and C matrix

[rowsEx,~] = size(Ex);
K = zeros(ndof);
C = zeros(ndof);

for i=1:rowsEx
    ex = Ex(i,:);
    ey = Ey(i,:);
    ep = 1;                                 % element thickness
    if t(4,i)==1                            % Solder (area 1)
        D = ksol*eye(2);
        x = rhosol*csol;                    % x: material paramater
    for C-matrix
    end
    if t(4,i)==2                            % PCB (area 2)
        D = kpcb*eye(2);
        x = rhopcb*cpcb;
    end
    if t(4,i)== 3                           % SMD (area 3)
```

```
        D = ksmd*eye(2);
        x = rhosmd*csmd;
    end
    edofrow = edof(i,:);
    Ke = flw2te(ex,ey,ep,D);
    Ce=plantml(ex,ey,x);

    K = assem(edofrow,K,Ke);
    C = assem(edofrow,C,Ce);
end
```

We created four functions to test which boundary a specific edge element was in. The boundaries are defined in figure 2

```
function b =isMemberL1(i,e,p)

    b = e(6,i)==3 && e(7,i)==0 && p(2,e(1,i)) > 0.599*10^(-3) && p(2,e
    (2,i)) > 0.599*10^(-3) && p(1,e(1,i)) < 0.21*10^(-3) && p(1,e(2,i))
    < 0.21*10^(-3);
    % multiplies with 10^(-3) to convert to meter.
end
```

```
function b = isMemberL2(i,e,p)

    b = e(6,i)==3 && e(7,i)==0 && p(2,e(1,i)) > 0.599*10^(-3) && p(2,e
    (2,i)) > 0.599*10^(-3) && p(1,e(1,i)) > 0.19*10^(-3) && p(1,e(2,i))
    > 0.19*10^(-3);
    % multiplies with 10^(-3) to convert to meter.
end
```

```
function b = isMemberL3(i,e,p)

 b = e(6,i)==0 && e(7,i)==1;

end
```

```
function b = isMemberL4(i,e,p)
    b = (e(6,i) == 2 && e(7,i) == 0 && p(1,e(1,i)) > 0.99*10^(-3) && p
    (1,e(2,i)) > 0.99*10^(-3));
```

```
function b = isMemberL5(i,e,p)

    b = (e(6,i) == 2 && e(7,i) == 0 && p(2,e(1,i)) < 0.001*10^(-3) &&
    p(2,e(2,i)) < 0.001*10^(-3));

end
```

```
function b = isMemberL6(i,e,p)

    b = p(1,e(1,i)) < 0.001*10^(-3) && p(1,e(2,i)) < 0.001*10^(-3);

end
```

We where now ready to calculate the K matrix and f vector and the solve for a.

```
%% Calcualtes the new K-matrx and f-matrix
[~,colse] = size(e);
f = zeros([ndof 1]);
Kc = zeros(ndof);


for i=1:colse
    l = abs(e(3,colse)-e(4,colse))*10^(-3);                    % length of element (converts to meter)
    if isMemberL1(i,e,p)                                       % Test of which boundary the edge is in
        f(e(1,i)) = f(e(1,i)) + qel*l/2;
        f(e(2,i)) = f(e(2,i)) + qel*l/2;

    elseif isMemberL2(i,e,p) || isMemberL3(i,e,p);             % Test of which boundary the edge is in
        f(e(1,i)) = f(e(1,i)) + ac*Tinf*l/2;
        f(e(2,i)) = f(e(2,i)) + ac*Tinf*l/2;

        Kc(e(1,i),e(1,i)) = Kc(e(1,i),e(1,i)) + ac*2*l/6;      % The diagonal elements.
        Kc(e(2,i),e(2,i)) = Kc(e(2,i),e(2,i)) + ac*2*l/6;

        Kc(e(1,i),e(2,i)) = Kc(e(1,i),e(2,i)) + ac*l/6;        % The non-diagonal elements.
        Kc(e(2,i),e(1,i)) = Kc(e(2,i),e(1,i)) + ac*l/6;

    end
end

Knew = K + Kc;
a=solveq(Knew,f);

%% Plots the stationary solution
ed = extract(edof,a);
ed = [ed;ed];                                                  % This is to mirror the solution
Ex = [Ex;-Ex];                                                 % This is to mirror the solution
Ey = [Ey;Ey];                                                  % This is to mirror the solution

fill(Ex',Ey',ed','EdgeColor','none')                           % Plots the temperature
xlabel('width [meter]', 'fontweight','bold','fontsize',15);
ylabel('height [meter]', 'fontweight','bold','fontsize',15);
title('The stationary temperature [Celsius]','fontsize',16);
set(gca,'fontsize',12);
colorbar;
```

Before we could calculate the transient temperature we needed some form of timestepping method.

```
function anew = eulerstep(aold, f, C, K, N, tf)
    h = tf/N;           % Stepsize
    aOLD = aold;
```

```
    anew = zeros([size(aold) 1]);
    for i=1:N
        anew = (K+C./h)\(f+C*aOLD./h);
        aOLD = anew;
    end
end
```

We could now plot the transient temperature

```
%% Plots the transient solution
aold = T0*ones([ndof 1]);
tf = 70;                                                    % For how
    long you want to go
N = 300;                                                    % Number
    of steps

a = eulerstep(aold, f, C, Knew, N, tf);

ed = extract(edof,a);
ed = [ed;ed];                                               % This is
    to mirror the solution
Ex = [Ex;-Ex];                                              % This is
    to mirror the solution
Ey = [Ey;Ey];                                               % This is
    to mirror the solution

fill(Ex',Ey',ed','EdgeColor','none')                        % Plots
    the temperature
xlabel('width [meter]', 'fontweight','bold','fontsize',15);
ylabel('height [meter]', 'fontweight','bold','fontsize',15);
title('The temperature after 70 seconds [Celsius]','fontsize',16);
set(gca,'fontsize',12);
colorbar;
```

5.2 Task 2

```
%% PART 2: Compute new Edof-matrix

nedof = [edof(:,1), zeros([nelm 1]), 2*edof(:,2), zeros([nelm 1]), 2*
    edof(:,3), zeros([nelm 1]), 2*edof(:,4)];

% This is our new Edof matrix

for i = 1:nelm
    for s = 1:3
        nedof(:,2*s) = nedof(:,2*s+1)-1;
    end
end

ndof = ndof*2;                                              % The new
    degree of freedom is 2x bigger

%% Computes the K and f matrix
ptype = 2;                                                  % Because of
    plain strain problem
K = zeros(ndof);
f = zeros([ndof 1]);
```

```
19
20 for i =1:nelm
21     ep = [ptype 1];
22     ex = Ex(i,:);
23     ey = Ey(i,:);
24
25     DeltaTp1 = a(t(1,i))-T0;                                % The
    temperature difference in each node
26     DeltaTp2 = a(t(2,i))-T0;
27     DeltaTp3 = a(t(3,i))-T0;
28     DeltaT = (DeltaTp1 + DeltaTp2 + DeltaTp3)/3;            % This is the
    mean temperatuer difference for an element
29
30     if t(4,i)==1                                            % Solder
31         D = hooke(ptype,Esol,vsol);
32         Deps0 = Esol*alphasol*DeltaT*[1;1;1;0]./(1-2*vsol);
33     end
34     if t(4,i)==2                                            % PCB
35         D = hooke(ptype,Epcb,vpcb);
36         Deps0 = Epcb*alphapcb*DeltaT*[1;1;1;0]./(1-2*vpcb);
37     end
38     if t(4,i)==3                                            % SMD
39         D = hooke(ptype,Esmd,vsmd);
40         Deps0 = Esmd*alphasmd*DeltaT*[1;1;1;0]./(1-2*vsmd);
41     end
42     Ke=plante(ex,ey,ep,D);
43     fe=plantf(ex,ey,ep,Deps0');
44
45     [K,f]=assem(nedof(i,:),K,Ke,f,fe);
46 end
47
48
49 %% Calculate the boundary Bc.
50 Bc = zeros(30,1);                                           % Our
    boundary matrix which will be used in solveq
51 counter = 1;
52 [~, length] = size(e);
53
54 for i = 1:length;
55     if isMemberL4(i,e,p) || isMemberL6(i,e,p)
56         Bc(counter,:) = [2*e(1,i)- 1];
57         counter = counter + 1;
58         Bc(counter,:) = [2*e(2,i)- 1];
59         counter = counter + 1;
60     end
61     if isMemberL5(i,e,p)
62         Bc(counter,:) = [2*e(1,i)];
63         counter = counter + 1;
64         Bc(counter,:) = [2*e(2,i)];
65         counter = counter + 1;
66     end
67 end
68 Bc = unique(Bc);
69 [length, ~] = size(Bc);
70 Bc = [Bc zeros(length,1)];
```

```
71
72 %% Solve and plots the displacement in all the nodes.
73 u = solveq(K,f,Bc);
74
75 ed = extract(nedof,u);
76
77 %eexpanded = [ed(:,1) ed(:,2) ed(:,3) ed(:,4) ed(:,5) ed(:,6);
78 %   -ed(:,1) ed(:,2) -ed(:,3) ed(:,4) -ed(:,5) ed(:,6)];    % This is
    to mirror the solution
79 %Ex = [Ex;-Ex];                                              % This is
     to mirror the solution
80 %Ey = [Ey;Ey];                                              % This is
     to mirror the solution
81
82 plotpar = [1,4,1];
83 magnfac = 1000;
84
85 %eldisp2(Ex,Ey,eexpanded,plotpar,magnfac);                   % Plots
    the mirrored displacement u
86 eldisp2(Ex,Ey,ed,plotpar,magnfac);                          % Plots
    the displacement u
87 xlabel('width [meter]', 'fontweight','bold','fontsize',15);
88 ylabel('height [meter]', 'fontweight','bold','fontsize',15);
89 title('The displacement [Magnification factor 1000]','fontsize',16);
90 set(gca,'fontsize',12);
91 %grid on
92
93
94 %% Computes the stresses and strains for each element aswell as the
    von Mises stress per element
95
96 [length,~]=size(nedof);
97 stress = zeros(nelm,4);
98 Seff_el = zeros(nelm,1);
99 for i = 1:length
100     ex = Ex(i,:);
101     ey = Ey(i,:);
102     ep = [ptype 1];
103     ed = [u(nedof(i,2)) u(nedof(i,3)) u(nedof(i,4)) u(nedof(i,5)) u(
    nedof(i,6)) u(nedof(i,7))];
104     % ed: All the displacements in one element
105     if t(4,i)==1                                % Solder
106         D = hooke(ptype,Esol,vsol);
107     end
108     if t(4,i)==2                                % PCB
109         D = hooke(ptype,Epcb,vpcb);
110     end
111     if t(4,i)==3                                % SMD
112         D = hooke(ptype,Esmd,vsmd);
113     end
114
115     [es,~]=plants(ex,ey,ep,D,ed);
116     stress(i,:) = stress(i,:) + es;
117
118     Seff_el(i) = sqrt(stress(i,1)^2 + stress(i,2)^2 + stress(i,3)^2 -
```

```
    stress(i,1)*stress(i,2) - stress(i,1)*stress(i,3) - stress(i,2)*
    stress(i,3) + 3*stress(i,4)^2);
    % This is the von Mises stress field for an element
end


%% Find the von Mises stress for each node
[length,~] = size(coord);
Seff_nod = zeros(length,1);

for i=1:size(coord,1)
    [c0,c1]=find(edof(:,2:4)==i);
    Seff_nod(i,1)=sum(Seff_el(c0))/size(c0,1);
end

%%
ed = extract(edof,Seff_nod);
%ed = [ed;ed];                                          % This is
    to mirror the solution
%Ex = [Ex;-Ex];                                         % This is
    to mirror the solution
%Ey = [Ey;Ey];                                          % This is
    to mirror the solution
fill(Ex',Ey',ed','EdgeColor','none')
xlabel('width [meter]', 'fontweight','bold','fontsize',15);
ylabel('height [meter]', 'fontweight','bold','fontsize',15);
title('The von Mises stress field [N/m^2]','fontsize',16);
set(gca,'fontsize',12);
colorbar;
```

References

[1] *Assignement in The Finite Element Method*, 2017.
URL: http://www.solid.lth.se/fileadmin/hallfasthetslara/utbildning/kurser/FHLF01_FEM_F_PI/fem_assignment_2017.pdf

[2] *Calfem, a finite element toolbox*
URL: http://www.solid.lth.se/fileadmin/hallfasthetslara/utbildning/kurser/FHL064_FEM/calfem34.pdf

[3] *Finite element formulation of transient heat transfer*
URL: http://www.solid.lth.se/fileadmin/hallfasthetslara/utbildning/kurser/FHL064_FEM/transheat.pdf

[4] Niels Ottosen & Hans Petersson. *Introduction to the Finite Element Method*, 1992.