Finite Element Project

Erik Sandström 940220-8350

Matilda Lundin 941102-4004

May 2017

Faculty of Engineering, Lund University

Division of Solid Mechanics

FHLF01, FHLF10 Finite Element Method

Spring 2017

Contents

1 Introduction

The given problem is two dimensional and concerns a surface mounted resistor on a printed circuit board together with connecting solder. An illustration of the geometry is given in Figure 1.

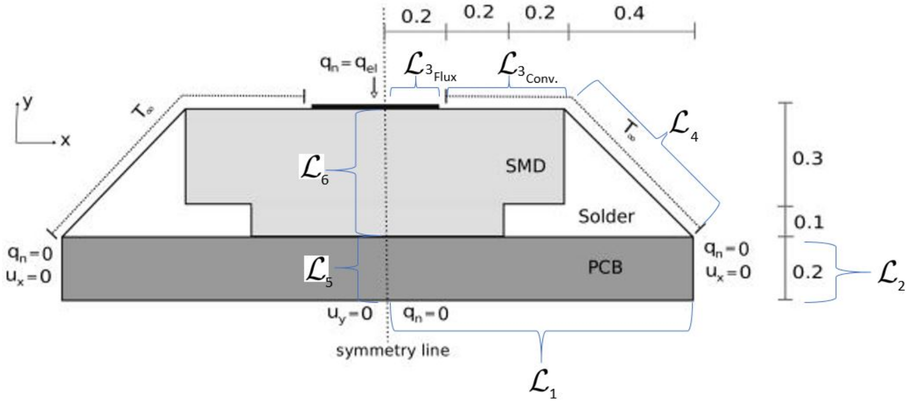

Figure 1: Geometry of the resistor, circuit board and solder together with labelled boundaries and boundary conditions. Dimensions are in mm [2].

Material parameters can be found in Table 1.

Table 1: Numerical values of material parameters and boundary condition values. PCB - Printed Circuit Board, SMD - Surface Mounted Device [2].

	SMD	PCB	Solder
E	105 GPa	105 GPa	50 GPa
ν	0.118	0.136	0.36
k	0.29 W/(mK)	1.059 W/(mK)	66.8 W/(mK)
ρ	1850 kg/m^3	1850 kg/m^3	7265 kg/m^3
c	950 J/(kgK)	950 J/(kgK)	210 J/(kgK)
α	$1.2 \cdot 10^{-5} K^{-1}$	$2 \cdot 10^{-5} K^{-1}$	$1.2 \cdot 10^{-5} K^{-1}$

$$T_0 = 30°C \quad \alpha_c = 40W/(m^2K) \quad q_{el} = 9 \cdot 10^3 W/m^2 \quad T_\infty = 20°C$$

The initial temperature of the body is T_0 and the boundary conditions are given by Figure 1. Note that the structure is symmetric, hence it is sufficient to consider only the right half. Along boundaries $\mathcal{L}_5$ and $\mathcal{L}_6$ in Figure 1, i.e. at the symmetry line, the structure is assumed to be thermally insulated and restricted from moving in the x-direction for symmetry reasons. Furthermore, along $\mathcal{L}_{3Conv.}$ and $\mathcal{L}_4$ Newton convection is present which yields $q_n = \alpha_c(T_0 - T_\infty)$ [2].

The task at hand can be divided into two parts where the first part is to determine the temperature distribution with respect to time and also to find the stationary temperature distribution. The second part concerns finding the von Mises stress field and corresponding displacement field due to thermal expansion. Both parts are solved using the finite element method [2].

2 Procedure

2.1 Mesh and boundary nodes

Initially the structure is meshed using Matlab `PDEtool` to obtain 397 nodes and 712 triangular three-node elements. The elements are separated into three subdomains as illustrated by Figure 2. Note that it is still sufficient to consider only the right half of the structure, as both temperature and stress distribution will be symmetric on both halves.

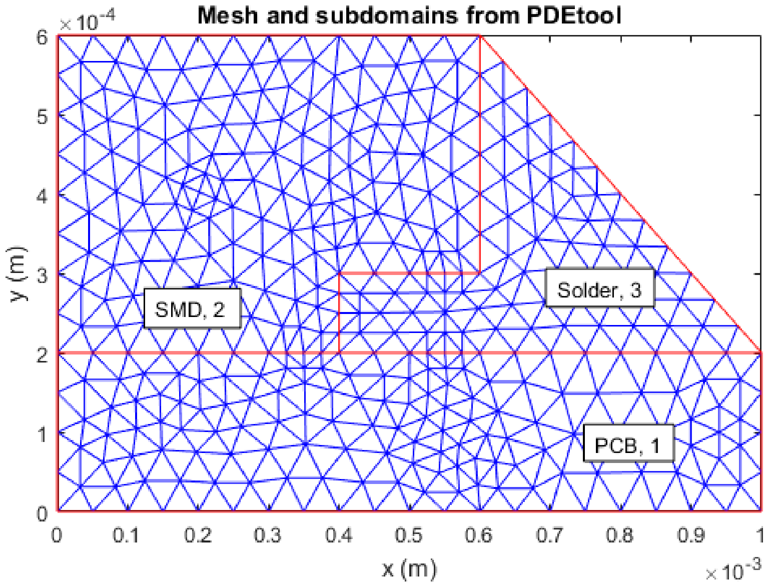

Figure 2: Mesh and subdomains of the structure as created by Matlab `PDEtool`. Only the right half of the structure is displayed due to symmetry. Dimensions are in m.

The boundary nodes and conditions are represented in the following illustration.

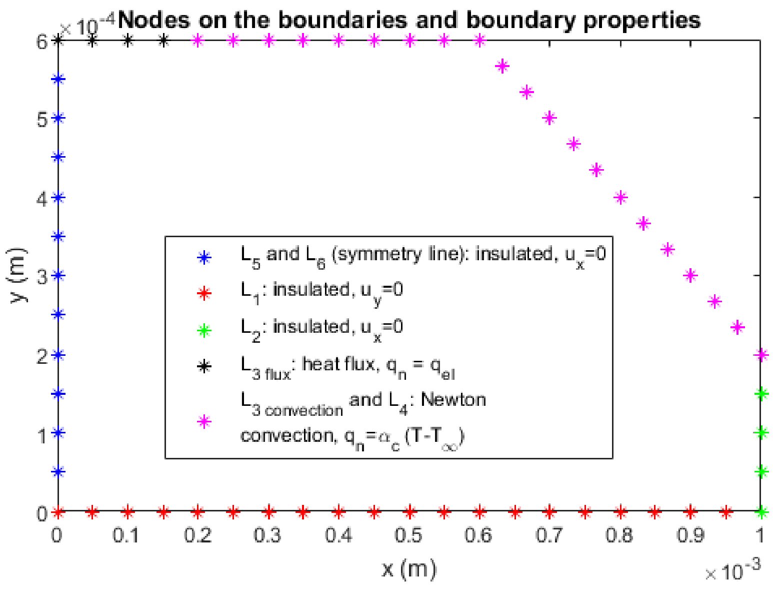

Figure 3: Boundary nodes of the structure and corresponding boundary conditions.

Note that every intersection between two boundaries in Figure 3 contains two nodes representing each boundary condition, so for example the bottom boundary of the PCB extends to $x = 1$ and the red node is printed under the green node.

2.2 Thermal distribution

The balance equation of the problem reads

$$\int_{\mathcal{S}} tQ \, \mathrm{d}\mathcal{S} - \int_{\mathcal{L}} tq_n \, \mathrm{d}\mathcal{L} = \int_{\mathcal{S}} t\rho c\dot{T} \, \mathrm{d}\mathcal{S}, \tag{1}$$

where t is the thickness of the structure, Q the global heat supply per area and time, q_n the normal heat flux to the boundary, ρ the density of the material, c the specific heat capacity and $\dot{T}$ the time derivative of the temperature distribution. Applying Gauss' divergence theorem to (1) and using that the equality holds for an arbitrary surface yields the corresponding strong form

$$Q - div\boldsymbol{q} = \rho c\dot{T}, \tag{2}$$

given by [3]. Here it is assumed that the thickness, t, is constant over the structure and can be set to $t = 1$ without affecting the result.

To reach the weak form, equation (2) is multiplied with an arbitrary weight function $v = v(x, y)$ and integrated over the entire body.

$$\int_{\mathcal{S}} vQ \, \mathrm{d}\mathcal{S} - \int_{\mathcal{S}} v div \boldsymbol{q} \, \mathrm{d}\mathcal{S} = \int_{\mathcal{S}} v\rho c\dot{T} \, \mathrm{d}\mathcal{S}, \tag{3}$$

given by [3]. Note that the reference to (3) is formulated in 3D, while (3) is the 2D representation of the same equation. Applying Green-Gauss theorem,

$$div(v\boldsymbol{q}) = (\nabla v)^T \boldsymbol{q} + v div(\boldsymbol{q}),$$

on (3) gives the final weak form

$$\int_{\mathcal{S}} v\rho c\dot{T} \, \mathrm{d}\mathcal{S} - \int_{\mathcal{S}} (\nabla v)^T \boldsymbol{q} \, \mathrm{d}\mathcal{S} + \int_{\mathcal{L}} v q_n \, \mathrm{d}\mathcal{L} - \int_{\mathcal{S}} vQ \, \mathrm{d}\mathcal{S} = 0, \tag{4}$$

given by [3]. To obtain the FE-formulation of the thermal distribution problem, the mesh from the previous section is introduced to divide the structure into discrete triangular elements. Time independent element shape functions are given for each element and are used to approximate the temperature field over the structure as described below.

The weight functions are chosen in accordance with the Galerkin method to

$$v = \boldsymbol{N}\boldsymbol{c}, \tag{5}$$

where $\boldsymbol{N}$ is the element shape function vector and $\boldsymbol{c}$ is an arbitrary coefficient vector. Furthermore, the global temperature function is approximated by a linear combination of the element shape functions as

$$T = \boldsymbol{N}\boldsymbol{a} \tag{6}$$

$$\dot{T} = \boldsymbol{N}\dot{\boldsymbol{a}}, \tag{7}$$

where $\boldsymbol{a}$ is a vector containing the nodal temperatures [5, p. 207]. Fourier's law, together with the approximation in (6), reads

$$\boldsymbol{q} = -\boldsymbol{D}\nabla T = \boldsymbol{D}\nabla \boldsymbol{N}\boldsymbol{a} = -\boldsymbol{D}\boldsymbol{B}\boldsymbol{a}, \tag{8}$$

where $\boldsymbol{B} = \nabla \boldsymbol{N}$ [5, p. 207]. Here $\boldsymbol{D}$ is the thermal conducitvity matrix, which is simply $\boldsymbol{D} = k\boldsymbol{I}$ for this isotropic two dimensional problem ($\boldsymbol{I}$ is the two by two identity matrix and k the thermal conductivity of the material). Applying the choice of weight functions from (5) and the approximations found in (7) and (8) to the weak formulation in (4) results in the FE formulation

$$\int_{\mathcal{S}} \boldsymbol{N}^T \rho c \boldsymbol{N} \, \mathrm{d}\mathcal{S}\dot{\boldsymbol{a}} + \int_{\mathcal{S}} \boldsymbol{B}^T \boldsymbol{D}\boldsymbol{B} \, \mathrm{d}\mathcal{S}\boldsymbol{a} + \int_{\mathcal{L}} \boldsymbol{N}^T q_n \, \mathrm{d}\mathcal{L} - \int_{\mathcal{S}} \boldsymbol{N}^T Q \, \mathrm{d}\mathcal{S} = 0, \tag{9}$$

given by [3]. Here we have exploited the fact that the coefficient vector $\boldsymbol{c}$ is arbitrary and that $\boldsymbol{N}\boldsymbol{c} = v = v^T = \boldsymbol{c}^T \boldsymbol{N}^T$, since v is a scalar. Usually, the FE formulation is written in a shorter form. While also putting Q = 0 in equation (9), since there is no internal heat supply, the result can be written as

$$\boldsymbol{C}\dot{\boldsymbol{a}} + \boldsymbol{K}\boldsymbol{a} = -\int_{\mathcal{L}} \boldsymbol{N}^T q_n \,\mathrm{d}\mathcal{L}, \tag{10}$$

where the matrices

$$\boldsymbol{C} = \int_{\mathcal{S}} \boldsymbol{N}^T \rho c \boldsymbol{N} \,\mathrm{d}\mathcal{S} \quad \text{and}$$
$$\boldsymbol{K} = \int_{\mathcal{S}} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \,\mathrm{d}\mathcal{S}$$

have been introduced.

The curve integral on the right hand side of equation (10) can be further investigated. Separating the boundaries along which the structure is insulated from those where convection occurs or the flux is unknown gives the expanded expression

$$\oint_{\mathcal{L}} \boldsymbol{N}^T q_n \,\mathrm{d}\mathcal{L} = \int_{\mathcal{L}_{1+2+5+6}} \boldsymbol{N}^T \cdot 0 \,\mathrm{d}\mathcal{L} + \int_{\mathcal{L}_{3_{Conv.}+4}} \boldsymbol{N}^T \alpha_c (T - T_0) \,\mathrm{d}\mathcal{L} + \int_{\mathcal{L}_{3_{Flux}}} \boldsymbol{N}^T q_{el} \,\mathrm{d}\mathcal{L}. \tag{11}$$

The temperature T included in the term related to convection is approximated using equation (6) and moved to the left side of the equation. What is left is regarded as the complete FE formulation.

$$\boldsymbol{C}\dot{\boldsymbol{a}} + \boldsymbol{K}\boldsymbol{a} + \int_{\mathcal{L}_{3_{Conv.}+4}} \boldsymbol{N}^T \alpha_c \boldsymbol{N} \,\mathrm{d}\mathcal{L} \boldsymbol{a} = \int_{\mathcal{L}_{3_{Conv.}+4}} \boldsymbol{N}^T \alpha_c T_\infty \,\mathrm{d}\mathcal{L} - \int_{\mathcal{L}_{3_{Flux}}} \boldsymbol{N}^T q_{el} \,\mathrm{d}\mathcal{L}$$

Which simplifies to

$$\boldsymbol{C}\dot{\boldsymbol{a}} + \boldsymbol{K}\boldsymbol{a} + \boldsymbol{K}_c \boldsymbol{a} = \boldsymbol{f}_c - \boldsymbol{f}_b \tag{12}$$

where

$$\boldsymbol{K}_c = \int_{\mathcal{L}_{3_{Conv.}+4}} \boldsymbol{N}^T \alpha_c \boldsymbol{N} \,\mathrm{d}\mathcal{L},$$
$$\boldsymbol{f}_c = \int_{\mathcal{L}_{3_{Conv.}+4}} \boldsymbol{N}^T \alpha_c T_\infty \,\mathrm{d}\mathcal{L} \quad \text{and}$$
$$\boldsymbol{f}_b = \int_{\mathcal{L}_{3_{Flux}}} \boldsymbol{N}^T q_{el} \,\mathrm{d}\mathcal{L}.$$

2.2.1 Calculating $\boldsymbol{K}$

First the $\boldsymbol{K}$-matrix is declared as an $[ndof \times ndof]$ matrix filled with zeros, where ndof is the number of degrees of freedom. For the thermal distribution problem, ndof is the number of nodes introduced in the meshing of the structure, as each node corresponds to one unknown temperature value. For every element, a three by three large $\boldsymbol{K}^e$ matrix is calculated using the CALFEM function `flw2te`. The appropriate material parameters included in $\boldsymbol{D}$ are also chosen depending on what subdomain the element is located in. Every element matrix is then assembled into the global $\boldsymbol{K}$-matrix and placed in accordance with the numbering of the nodes contained in the element.

2.2.2 Calculating $\boldsymbol{K}_c$

The contribution of $\boldsymbol{K}_c$ to the complete FE formulation comes from the boundary convection term over $\mathcal{L}_{3_{Conv.}}$ and $\mathcal{L}_4$. When a local coordinate system is used over the boundary line for each element it can be shown that the contribution from each element to $\boldsymbol{K}_c$ only depends on the distance L between adjacent boundary nodes. Figure 4 shows the element shape functions for each element along the boundary. Note that for every new element, a separate coordinate system η is introduced.

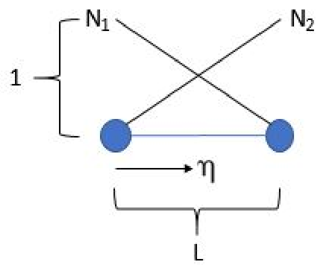

Figure 4: Illustration of the element shape functions projected along the boundary for an arbitrary element of length L. The local coordinate system η is also shown.

The element shape functions according to Figure 4 are

$$N_1 = 1 - \frac{\eta}{L} \tag{13}$$

$$N_2 = \frac{\eta}{L} \tag{14}$$

The elementwise contribution $\boldsymbol{K}_c^e$ is described by

$$
\begin{aligned}
\boldsymbol{K}_c^e &= \int_0^L \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} \alpha_c \begin{bmatrix} N_1 & N_2 \end{bmatrix} \mathrm{d}\boldsymbol{\eta} \\
&= \alpha_c \int_0^L \begin{bmatrix} (1-\frac{\eta}{L})^2 & \frac{\eta}{L}(1-\frac{\eta}{L}) \\ \frac{\eta}{L}(1-\frac{\eta}{L}) & \frac{\eta^2}{L^2} \end{bmatrix} \mathrm{d}\boldsymbol{\eta} \\
&= \alpha_c \frac{L}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.
\end{aligned}
$$

A loop is created over the elements related to the boundary term $\mathcal{L}_{3_{Conv.}+4}$. Each elementwise $\boldsymbol{K}_c^e$ matrix is assembled into $\boldsymbol{K}_c$ and placed in accordance with the numbering of the nodes contained in the element.

2.2.3 Calculating $\boldsymbol{f}_c$ and $\boldsymbol{f}_b$

The contribution of $\boldsymbol{f}_c$ and $\boldsymbol{f}_b$ to the complete FE formulation comes from the boundary terms over $\mathcal{L}_{3_{Conv.}+4}$ and $\mathcal{L}_{3_{Flux}}$. The procedure is closely related to calculating $\boldsymbol{K}_c$ and every elementwise contribution is only dependent on the distance L between adjacent nodes. Figure 4 shows the element shape functions for each element along the boundary. Note that for every new element, a separate coordinate system η is introduced. The element shape functions are in accordance with (13) and (14) for both $\boldsymbol{f}_c$ and $\boldsymbol{f}_b$.

The elementwise contribution $\boldsymbol{f}_c$ is described by

$$
\begin{aligned}
\boldsymbol{f}_c^e &= \int_0^L \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} \alpha_c T_\infty \, \mathrm{d}\boldsymbol{\eta} \\
&= \alpha_c T_\infty \int_0^L \begin{bmatrix} 1-\frac{\eta}{L} \\ \frac{\eta}{L} \end{bmatrix} \mathrm{d}\boldsymbol{\eta} \\
&= \alpha_c T_\infty \frac{L}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.
\end{aligned}
$$

The elementwise contribution $\boldsymbol{f}_b$ is described by

$$\begin{aligned}\boldsymbol{f}_b^e &= \int_0^L \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} q_{el}\, \mathrm{d}\boldsymbol{\eta} \\ &= q_{el} \int_0^L \begin{bmatrix} 1 - \frac{\eta}{L} \\ \frac{\eta}{L} \end{bmatrix} \mathrm{d}\boldsymbol{\eta} \\ &= q_{el} \frac{L}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.\end{aligned}$$

A loop is created over the elements related to the boundary terms $\mathcal{L}_{3_{Conv.}+4}$ and $\mathcal{L}_{3Flux}$. Each elementwise $\boldsymbol{f}_c^e$ and $\boldsymbol{f}_b^e$ matrix is assembled into $\boldsymbol{f}_c$ and $\boldsymbol{f}_b$ respectively and placed in accordance with the numbering of the nodes contained in the element.

2.2.4 Calculating $\boldsymbol{C}$

First the $\boldsymbol{C}$-matrix is declared as an $[ndof \times ndof]$ matrix filled with zeros, where ndof is the number of degrees of freedom. The procedure is very similar to calculating $\boldsymbol{K}$, but in this case the CALFEM function `plantml` is used to calculate the three by three large $\boldsymbol{C}^e$ elementwise matrix. The appropriate material parameters ρ and c are also chosen depending on what subdomain the element is located in. Every element matrix is then assembled into the global $\boldsymbol{C}$-matrix and placed in accordance with the numbering of the nodes contained in the element.

2.2.5 Time integration

The approximation of the time derivative is chosen to

$$\dot{a} = \frac{a_{n+1} - a_n}{\Delta t}, \tag{15}$$

where a_n is the nodal temperature vector evaluated at time t_n and Δt is the difference between two consecutive time steps [3]. The implicit Euler method is then used to solve (12) stepwise. This yields,

$$\boldsymbol{C}\frac{a_{n+1} - a_n}{\Delta t} + (\boldsymbol{K} + \boldsymbol{K_c})a_{n+1} = f_{n+1}, \tag{16}$$

where $f = f_b + f_c$. Since f, however, is constant in time the resulting expression of a_{n+1} is

$$a_{n+1} = (\boldsymbol{C} + \boldsymbol{K}\Delta t)^{-1}(\boldsymbol{f}\Delta t + \boldsymbol{C}a_n), \tag{17}$$

which is determined by the Matlab function `eulerFEMstep`, see appendix. By deciding a set number of time steps N and a time interval over which to perform the numerical integration, the result can be calculated with a for-loop in Matlab. This is performed by the Matlab function `eulerFEMint`, see appendix. This gives a $[ndof \times N]$ large matrix where every column represents the nodal temperature at a specific point in time.

The number of time steps chosen in this case are $N = 1000$ over 100 seconds meaning the time step $\Delta t = 10^{-1}$ s.

2.2.6 Calculating the stationary solution

The CALFEM function `solveq` is used to calculate the static solution. The imputs are $\boldsymbol{K}$ and $\boldsymbol{f}_b + \boldsymbol{f}_c$. No bc matrix exists since there are no Dirichlet boundary conditions for the thermal distribution problem. The static solution is furthermore defined by no transients, which is why the $\boldsymbol{C}$ matrix is not taken into account.

2.3 Stress field

The force balance equation of the static two dimensional structure reads

$$\int_{\mathcal{L}} \boldsymbol{t}\,\mathrm{d}\mathcal{L} - \int_{\mathcal{S}} \boldsymbol{b}\,\mathrm{d}\mathcal{S} = \boldsymbol{0}, \tag{18}$$

where $\boldsymbol{t}$ is the two-dimensional traction vector and $\boldsymbol{b}$ is the body force over the surface $\mathcal{S}$. Note that (18) is a vector equation, so the right hand side is a vector zero, $\boldsymbol{0}$. Here, as in the previous part, the thickness of the structure is constant and set to $t = 1$ and thus omitted.

Using Cauchy's formula and applying Gauss' divergence theorem in accordance with [5, p. 240-241] enables the equlibrium equation to be rewritten into

$$\tilde{\nabla}^T \boldsymbol{\sigma} + \boldsymbol{b} = 0, \tag{19}$$

where $\tilde{\nabla}^T = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}$, $\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}$ and $\boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix}$. Multiplying equation (19) with an arbitrary weight function $\boldsymbol{V} = \begin{bmatrix} V_x \\ V_y \end{bmatrix}$, integrating over the entire body and finally applying Green-Gauss theorem results in the weak formulation

$$\int_{\mathcal{S}} \tilde{\nabla \boldsymbol{V}}^T \boldsymbol{\sigma}\,\mathrm{d}\mathcal{S} = \int_{\mathcal{L}} \boldsymbol{V}^T \boldsymbol{t}\,\mathrm{d}\mathcal{L} + \int_{\mathcal{S}} \boldsymbol{V}^T \boldsymbol{b}\,\mathrm{d}\mathcal{S}. \tag{20}$$

In order to determine the displacement field of the structure, a connection between the displacement vector $\boldsymbol{u}$ and the stress $\boldsymbol{\sigma}$ is needed. The relationship is provided by Hooke's law

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon}^{\boldsymbol{e}}, \tag{21}$$

relating $\boldsymbol{\sigma}$ to the strain $\boldsymbol{\varepsilon}$ and then by the definition of strain

$$\boldsymbol{\varepsilon} = \tilde{\nabla}\boldsymbol{u}. \tag{22}$$

Here $\boldsymbol{\varepsilon}^{\boldsymbol{e}}$ is the elastic strain vector, $\boldsymbol{\varepsilon} = [\varepsilon_{xx} \quad \varepsilon_{yy} \quad \gamma_{xy}]^T$ is the complete strain vector and

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

is the two dimensional constitutive matrix, with E representing Young's modulus and ν Poisson's ratio. For a thermoelastic material $\boldsymbol{\varepsilon}^{\boldsymbol{e}}$ can be expressed as

$$\boldsymbol{\varepsilon}^{\boldsymbol{e}} = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{\boldsymbol{\Delta T}}, \tag{23}$$

where $\boldsymbol{\varepsilon}^{\boldsymbol{\Delta T}}$ is the thermal contribution to the complete strain vector [5, p. 249-257].

To finalize the FE formulation, an approximation of $\boldsymbol{u}$ is needed. The displacement vector $\boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$ is a function of (x,y) and and is modelled after the nodal displacements in the x- and y-direction together with the element shape functions. The approximation used is

$$\boldsymbol{u} = \boldsymbol{N}\boldsymbol{a}, \tag{24}$$

where $\boldsymbol{N}$ are the global shape functions and $\boldsymbol{a}$ is the nodal displacements [5]. Note that $\boldsymbol{N}$ and $\boldsymbol{a}$ are not the same as in the thermal distribution problem. Twice as many rows are used in $\boldsymbol{a}$ compared to the equivalent in the thermal distribution problem. This is due to the fact that the stress problem consists of two degrees of freedom per node (x- and y-direction) compared to one degree of freedom per node for the thermal distribution problem. Thus each node in the mesh corresponds to two rows in $\boldsymbol{a}$, i.e.

$$\boldsymbol{a} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ \vdots \\ u_{xn} \\ u_{yn} \end{bmatrix}.$$

The shape function matrix $\boldsymbol{N}$ consists of two rows and twice the amount of columns compared to the shape function vector used in the thermal distribution problem, i.e.

$$\boldsymbol{N} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & \dots & N_n & 0 \\ 0 & N_1 & 0 & N_2 & \dots & 0 & N_n \end{bmatrix}$$

where n is the total number of nodes introduced in the mesh [5]. This is basically how it has to be designed to meet the dimension critera of $\boldsymbol{u}$.

Applying the approximation in (24) to the strain given by (22) yields the matrix equation

$$\boldsymbol{\varepsilon} = \tilde{\nabla}^T \boldsymbol{u} = \tilde{\nabla}^T \boldsymbol{N}\boldsymbol{a} = \boldsymbol{B}\boldsymbol{a}, \tag{25}$$

where $\boldsymbol{B} = \tilde{\nabla}^T \boldsymbol{N}$.

The Galerkin method is once more used to give the weight functions $\boldsymbol{V}$ as

$$\boldsymbol{V} = \boldsymbol{N}\boldsymbol{c}, \tag{26}$$

where $\boldsymbol{c}$ is the arbitrary coefficient vector, see [5, p. 152-153] for details about weight function choices. Equation (20) thus becomes

$$\int_{\mathcal{S}} \boldsymbol{B}^T \boldsymbol{\sigma} \, \mathrm{d}\mathcal{S} = \int_{\mathcal{L}} \boldsymbol{N}^T \boldsymbol{t} \, \mathrm{d}\mathcal{L} + \int_{\mathcal{S}} \boldsymbol{N}^T \boldsymbol{b} \, \mathrm{d}\mathcal{S} \tag{27}$$

where $\boldsymbol{c}$ has been removed (as it is arbitrary).

Putting the approximations in to equation (27) and using equation (21) and (23) to replace $\boldsymbol{\sigma}$ with $\boldsymbol{\varepsilon}$ brings us one step further. When also taking into account that no body forces act in the region ($\boldsymbol{b} = \boldsymbol{0}$), this results in the complete FE formulation.

$$\int_{\mathcal{S}} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \, \mathrm{d}\mathcal{S} \boldsymbol{a} = \int_{\mathcal{L}} \boldsymbol{N}^T \boldsymbol{t} \, \mathrm{d}\mathcal{L} + \int_{\mathcal{S}} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{\varepsilon}^{\Delta T} \, \mathrm{d}\mathcal{S} \tag{28}$$

which can be written more compactly as

$$\boldsymbol{K}\boldsymbol{a} = \boldsymbol{f}_b + \boldsymbol{f}^{\Delta T}, \tag{29}$$

where

$$\boldsymbol{K} = \int_{\mathcal{S}} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \, \mathrm{d}\mathcal{S}, \quad \boldsymbol{f}_b = \int_{\mathcal{L}} \boldsymbol{N}^T \boldsymbol{t} \, \mathrm{d}\mathcal{L} \quad \text{and } \boldsymbol{f}^{\Delta T} = \int_{\mathcal{S}} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{\varepsilon}^{\Delta T} \, \mathrm{d}\mathcal{S}.$$

2.3.1 Calculating $\boldsymbol{K}$

As in the thermal problem, $\boldsymbol{K}$ is declared as an $[ndof \times ndof]$ matrix filled with zeros, where ndof is the number of degrees of freedom. Here $ndof$ is twice the number of nodes in the mesh, corresponding to a displacement in the x- and y-direction for each node. For a node with index i, the correspinding degrees of freedom are thus $2i - 1$ and $2i$. For every element, a six by six large $\boldsymbol{K}^e$ matrix is calculated using the CALFEM function `plante`. The appropriate material parameters included in the constitutive matrix $\boldsymbol{D}$ are chosen depending on whether the element is part of the solder, SMD och PCB, see Table 1. Every element matrix is then assembled into the global $\boldsymbol{K}$-matrix and placed in accordance with the specific degrees of freedom related to the nodes contained in the element.

2.3.2 Calculating $\boldsymbol{f}^{\Delta T}$

To compute the thermal strain vector $\boldsymbol{f}^{\Delta T}$ the thermal strains in each element are needed. Since the studied material is isotropic and the problem is two-dimensional, the thermal strains are given by

$$\boldsymbol{\varepsilon}_e^{\Delta T} = \alpha \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \Delta T_e, \tag{30}$$

where α is the thermal expansion coefficient of the current material, see Table 1. The temperature difference ΔT_e of an element is taken as the average of the temperature differences of its nodes, the latter being defined as $\Delta T = T_{static} - T_0$, where T_{static} is the static nodal temperature distribution, obtained from the CALFEM function `solveq` using the stiffness matrix and force vector for the thermal distribution problem. T_0 is the initial temperature of the structure (see Introduction). Thus the temperature difference in an arbitrary element with nodes i, j and k is given by $\Delta T_e = (\Delta T_i + \Delta T_j + \Delta T_k)/3$ where $\Delta T_i = T^i_{static} - T_0$ is the temperature difference in node i.

Next, the elementwise thermal strain vector $\boldsymbol{f}_e^{\Delta T}$ is computed using the CALFEM function `plantf` for the case of plane strain, and then assembled into a global strain vector using the specific degrees of freedom related to the current element.

2.3.3 Calculating the displacement field

To compute the nodal displacement vector, $\boldsymbol{a}$, the CALFEM function `solveq` is applied. Note that the boundary force vector $\boldsymbol{f}_b$ is not explicitly needed for this calculation. Instead, a matrix bc with the boundary conditions of the structure is supplied to `solveq`. The matrix is created by placing all degrees of freedom for which the displacement is known (i.e. along $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_5$ and $\mathcal{L}_6$, see Figure 1) in the first column, and the condition, in this case that the displacement is zero, in the second column. For the remaining boundaries, we note that there are no external forces acting on the structure, so the traction vector is $\boldsymbol{t} = \boldsymbol{0}$. We then have everything necessary to compute $\boldsymbol{a}$. To obtain the displacement of each element in the mesh, the CALFEM function `extract` is used. This gives a matrix ed where each row contains the nodal displacements of the element with the corresponding index, i.e. for element number m with nodes indexed i, j and k, the m_{th} row of ed is

$$ed_{m,:} = \begin{bmatrix} u_{xi} & u_{yi} & u_{xj} & u_{yj} & u_{xk} & u_{yk} \end{bmatrix}.$$

2.3.4 Calculating the von Mises stress field

From the ed-matrix with the elementwise displacements, the stresses in the structure is calculated from Hooke's law and equation (22) using the CALFEM function `plants`. However, the function does not take the thermal strains into account, so the contribution from these have to be substracted from the result manually. The out-of-plane stress (in the z-direction) also has to be calculated separately. This is done for each element in accordance with [5, p. 255-256],

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{bmatrix} = \sigma_{plants} - \frac{\alpha E \Delta T_e}{1 - 2\nu} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\sigma_{zz} = \nu(\sigma_{xx} + \sigma_{yy}) - \alpha E \Delta T_e.$$

Here σ_{plants} denotes the result of the Matlab call to `plants`. The material parameters are as before chosen depending on what subdomain the element belongs to.

The effective von Mises stress for each element is then calculated according to the definition in [4, p. 91]

$$\sigma_{Mises} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 - \sigma_{xx}\sigma_{yy} - \sigma_{yy}\sigma_{zz} - \sigma_{xx}\sigma_{zz} + 3\tau_{xy}^2 + 3\tau_{yz}^2 + 3\tau_{zx}^2}.$$

3 Results

3.1 Temperature distribution fields

For the given problem, three specific points in time were chosen to illustrate how the temperature is distributed in the body and how the distribution varies. Firstly, a field showing the temperature after 0.1 s was chosen, see Figure 5, in order to see the immediate impact of the heat flow through the structure (recall that the initial temperature is $T_0 = 30°C$ in the entire body).

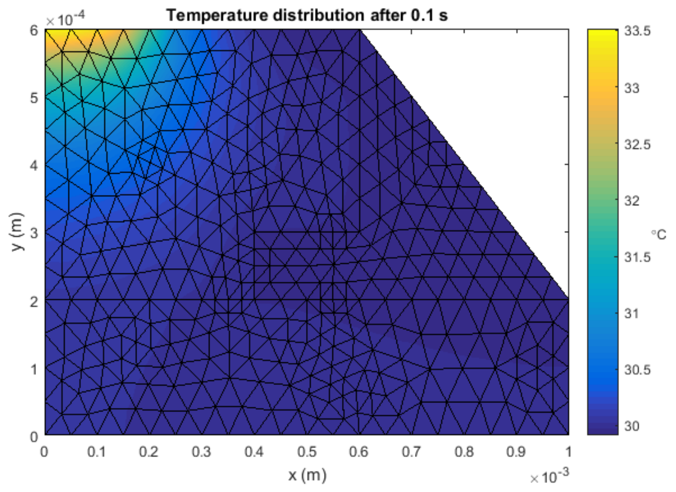

Figure 5: Temperature distribution after 0.1 s. The temperature is given in degrees Celsius and the range is approximately $T \in (30, 33.5)$.

Secondly, a field showing the temperature after 25 s was chosen, see Figure 6, in order to illustrate how the temperature is distributed when the transient heat flow is still ongoing.

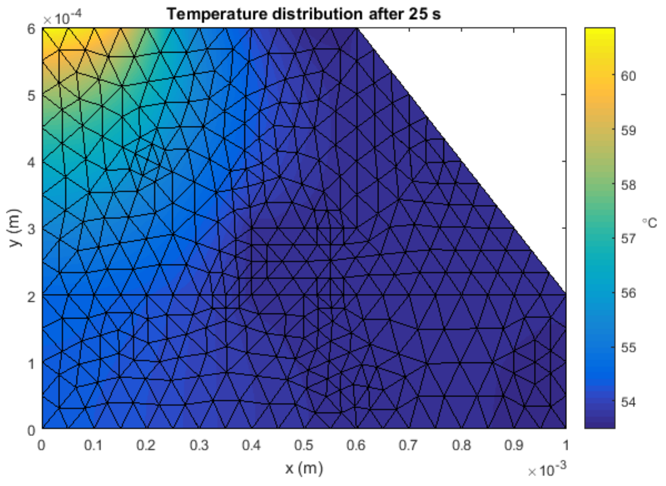

Figure 6: Temperature distribution after 25 s. Temperature in degrees Celsius, range approximately $T \in (53, 61)$.

The third and final field was chosen to visualize an approximation of the solution after a long period of time, when the transients are negligible. It was discovered that the field has approximately reached the stationary distribution after $100s$, see Figure 7. The close resemblance to the stationary solution is shown by comparison to Figure 8 (computed directly without respect to transients).

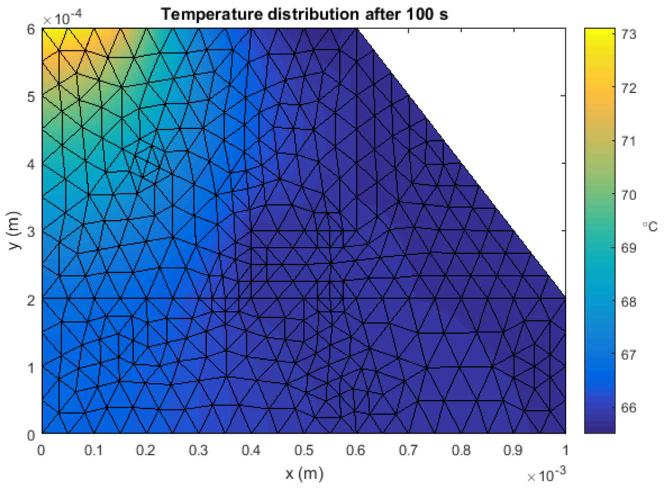

Figure 7: Temperature distribution after 100 s. The temperature is given in degrees Celsius and the range is approximately $T \in (65.5, 73)$.

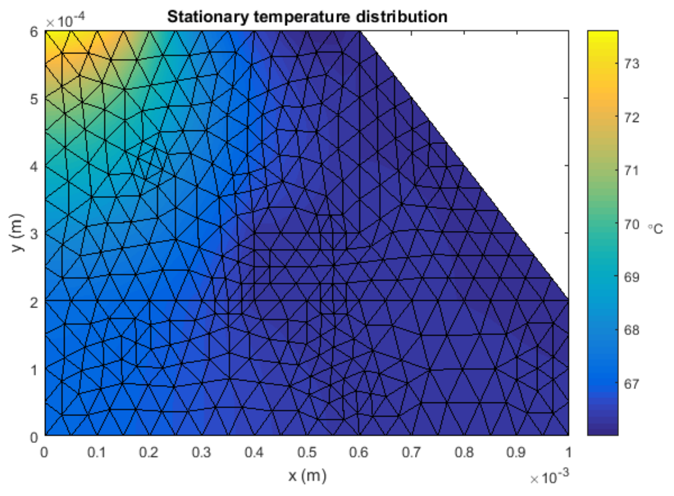

Figure 8: Stationary temperature distribution. Temperature in degrees Celsius, range approximately $T \in (65.5, 73)$.

3.2 Stress and displacement fields

Figure 9 shows the effective von Mises stress field of the structure in the stationary temperature case given by Figure 8. The corresponding displacement field, illustrated as a deformed mesh on top of the original mesh, is shown in Figure 10. Note that the displacements have been multiplied by a factor of 100 for visual purposes (the deformations are small).

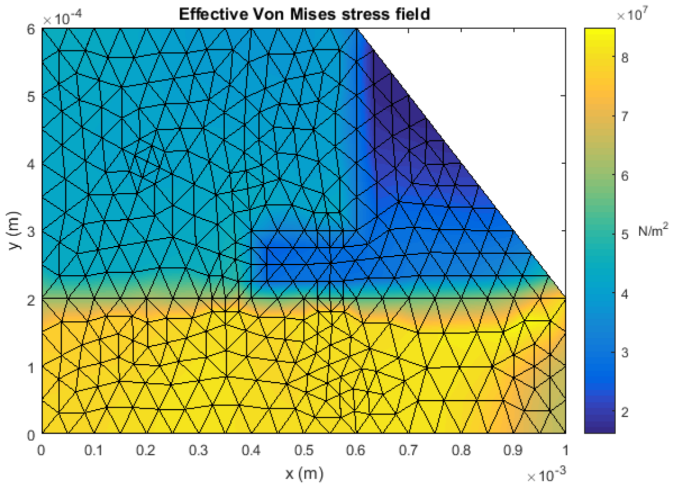

Figure 9: Effective von Mises stress field given. Stress given in N/m^2*, range approximately* $\sigma_{Mises} \in (2 \cdot 10^7, 8 \cdot 10^7)$.

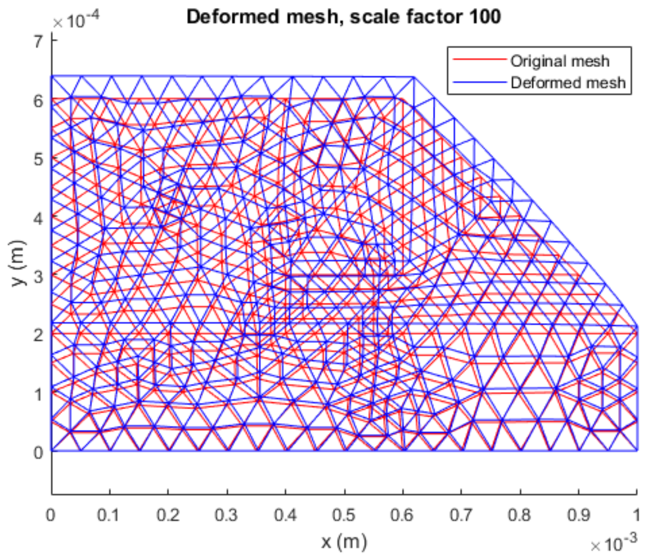

Figure 10: Deformed mesh magnified a factor 100, displayed over the original mesh.

4 Discussion

Overall the results in both part one and two are reasonable for the given problem. The temperature field is expected to have a maximum where the structure is exposed to the electric current, i.e. along $\mathcal{L}_{3flux}$, consistent with the result displayed in Figures 5, 6 and 7. As can be seen in the later figures, the heat spreads in the structure and heats the entire body. Most heat is, however, retained in parts close to the source and along insulated boundaries, whereas the border where convection occurs is cooled off by the surroundings. Here too the results are in accordance with the expected.

In the second part, the stress field in Figure 9 shows a clear difference between the different subdomains. This is reasonable as the subdomains consist of different materials with corresponding material parameters that affect the properties and thus the strains and stresses induced. As can be seen in Figure 9 the stresses are higher in the PCB than in the SMD and solder. This is justified by the assumption that the PCB is restricted from moving [2]. Thus the structure cannot expand and high stress is induced. Along $\mathcal{L}_4$ in the solder and $\mathcal{L}_3$ in the SMD the structure is free to move and this is also where the largest displacements occur, as shown by Figure 10.

In order to explore the effects of the degree of refinement of the mesh, the stationary temperature distribution and von Mises stress field were calculated for two additional meshes. One with lower resolution (178 elements) and one with higher resolution (2848 elements) than the mesh used for the above results. The temperature distribution was not affected much by the number of elements used, in fact the result is very similar in all three cases, see Figures 8, 11 and 12.

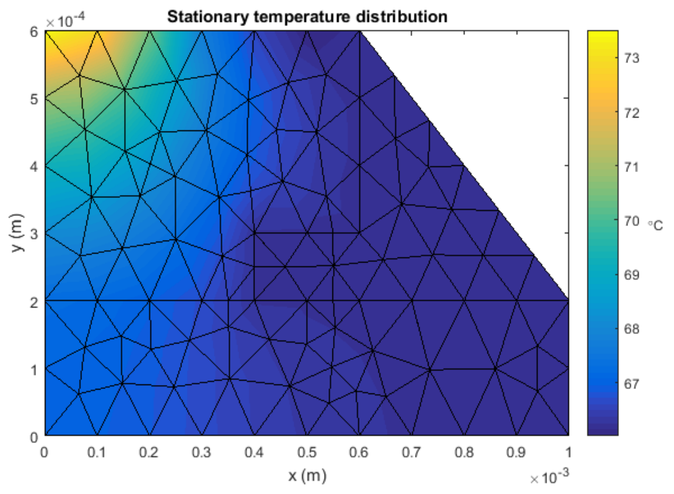

Figure 11: Stationary temperature distribution using 107 nodes and 178 elements. Temperature displayed in degrees Celsius.

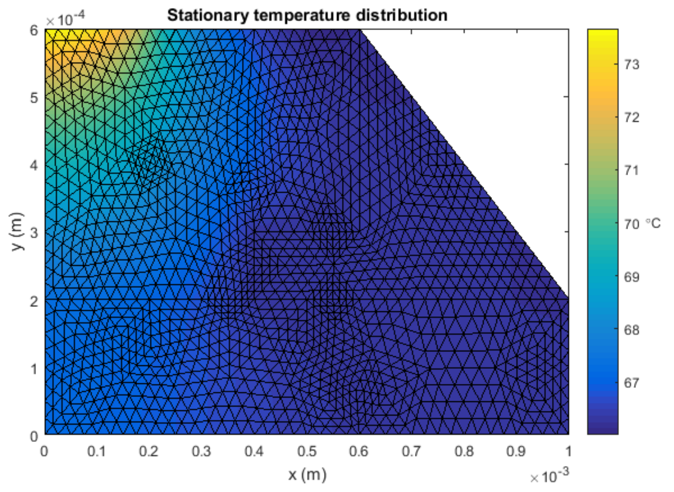

Figure 12: Stationary temperature distribution using 1485 nodes and 2848 elements. Temperature displayed in degrees Celsius.

The stress field was not affected much by the lower resolution either. Clear differences between the subdomains are still visible in Figure 13 and the magnitude of the stress lies in the same range as in Figure 9.

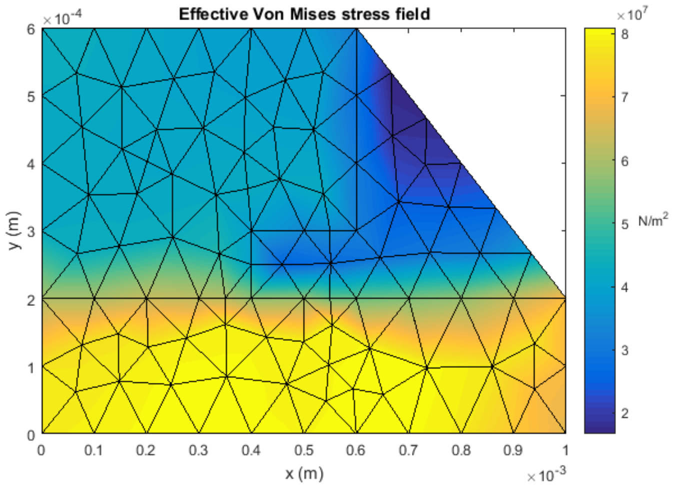

Figure 13: Effective von Mises stress field using 107 nodes and 178 elements. Stress given in N/m^2.

Unfortunately, the von Mises effective stress could not be computed for the high resolution mesh with 2848 elements. The computer crashed with error message 'out of memory' when it was attempted.

Another potential source of error is the approximations used, such as the choice of element type. In this project three node triangular elements have been used, motivated by the fact that they fulfill the convergence criteria while still being quite simple. As a consequence of this, the solution is guaranteed to converge to the exact solution when the elements get arbitrarily small [5, p. 124-125]. However, from the discussion above it is clear that the approximated solution does not vary much with the element size, which leads to the conclusion that the computed solution is close to the exact one.

Another approximation is made in the time integration in part 1, where the implicit Euler method is used to give a numerical solution to the transient thermal distribution problem. The error of the result depends on the size of the time steps used in the numerical integration, here chosen to 1/10 s. However, courtesy to the method being *implicit* we do not have to worry about numerical instability, as method stable for all time steps $\Delta t > 0$ [1].

References

[1] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations.* Cambridge University Press, Cambridge, United Kingdom, 2009.

[2] Division of Solid Mechanics. Assigment in the finite element method, 2017.

http://www.solid.lth.se/fileadmin/hallfasthetslara/utbildning/kurser/FHLF01_FEM_F_PI/fem_assignment_2017.pdf.

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

[4] Niels Ottosen and Christer Ljung et al. *Hållfasthetslära för Pi.* Avdelningen för Hållfasthetslära, Lunds Universitet, Lund.

[5] Niels Ottosen and Hans Petersson. *Introduction to the Finite Element Method.* Pearson Prentice Hall, Edinburgh Gate, Harlow, England, 1992.

A Computer Code

This is the Matlab script used to calculate the above results:

```
%% Material Coefficients
Esmd= 105*10^9;  Epcb= 105*10^9; Esol= 50*10^9;
nysmd= 0.118;    nypcb= 0.136;   nysol= 0.36;
ksmd= 0.290;       kpcb= 1.059;    ksol= 66.80;
rosmd= 1850;     ropcb= 1850;    rosol= 7265;
csmd= 950;       cpcb= 950;      csol= 210;
alphasmd= 1.2*10^-5; alphapcb= 2*10^-5; alphasol= 1.2*10^-5;

T0= 30+273.15; %Converted from Celsius to degrees Kelvin
Tinf = 20+273.15; %Degrees Kelvin
ac= 40;
qel= 9*10^3;

%% Export mesh from PDE-tool
%Imports the mesh created by PDE-tool
load('e.mat'); load('p.mat'); load('t.mat');
figure %Displays the mesh
pdemesh(p,e,t)

%% Create edof matrix and plot nodes and edges
%The number of columns in t corresponds to the number of triangles in our
%mesh:
nelm=length(t(1,:));
edof(:,1)=1:nelm; %The first column in edof contains the element index
%Columns 2, 3 and 4 are the nodes in anti-clockwise order for the
%corresponding element (triangle):
edof(:,2:4)=t(1:3,:)';
coord=p'; %Column vector with all nodes (x- and y-coordinate)

%max(t(1:3,:) returns a row vector with the largest number in each column.
ndof=max(max(t(1:3,:)));
%Gives the element coordinate matrices. Each row in Ex contains
%x-coordinates of one element:
[Ex,Ey]=coordxtr(edof,coord,(1:ndof)',3);
eldraw2(Ex,Ey,[1,2,0]);
```

```
%% Boundaries
%Index of the nodes which x-coordinate has less distance to 0 than 1e-3.
%This gives nodes on the symmetry line, L_5 & L_6:
bcSym=find(abs(coord(:,1)-0)<1e-6);
%Same as above, but for y, gives lower boundary L_1:
bcBottom=find(abs(coord(:,2)-0)<1e-6);
%Index for nodes with x~1, i.e. right boundary L_2:
bcRight=find(abs(coord(:,1)-0.001)<1e-6);
%Index for nodes with y~0.6, i.e. upper boundary L_3:
bcTop=find(abs(coord(:,2)-0.0006)<1e-6);
%All nodes on the line y=-x+1.2:
bcConv=find(abs(coord(:,1)+coord(:,2)-0.0012)<1e-6);
bcFlux =[];

%Loop to divide L_3 into L_3 flux and L_3 convection
for i=1:length(bcTop)
    if (coord(bcTop(i),1) < 0.0002-1e-6)
        bcFlux = [bcFlux; bcTop(i)]; %Add node number (bcTop(i) to bcFlux
    elseif (coord(bcTop(i),1) > 0.0002+1e-6)
        bcConv = [bcConv; bcTop(i)]; %Nodes that don't belong to bcFlux belong
        %to L_3 convection, except th node at x=0.2 which belongs to both
    elseif (abs(coord(bcTop(i),1)-0.0002)<1e-6)
        bcFlux = [bcFlux; bcTop(i)];
        bcConv = [bcConv; bcTop(i)];
    end
end

%Displays nodes belonging to the different boundaries
%Note that all corner nodes are double, but only shown in one color
figure
plot(coord(bcSym,1),coord(bcSym,2),'b*')
hold on
plot(coord(bcBottom,1),coord(bcBottom,2),'r*')
plot(coord(bcRight,1),coord(bcRight,2),'g*')
plot(coord(bcFlux,1),coord(bcFlux,2),'k*')
plot(coord(bcConv,1),coord(bcConv,2),'m*')
legend('L_5 and L_6 (symmetry line): insulated, u_x=0', ...
    'L_1: insulated, u_y=0', 'L_2: insulated, u_x=0', 'L_{3 flux}', ...
    'L_{3 convection} and L_4')

%% Compute global stiffness matrix (without convection terms)
b=1; %The thickness of the structure bears no relevance, set to 1 everywhere
ndep=edof(:,2:4); %Takes out only nodal dependancies from edof
K = zeros(ndof); %Create empty global stiffness matrix
for i=1:nelm %Loop over all 712 triangular elements
    if t(4,i) == 1
        %If the subdomain nr (found in 4th column of t-matrix) is 1
        k = kpcb; %Then the current subdomain is the PCB
    elseif t(4,i) == 2
        k = ksmd; %Subdomain number 2 is the SMD
    else
        k = ksol; %Subdomain number 3 is the solder
    end
    D = k*eye(2); %Isotropic material
    Ke = flw2te(Ex(i,:), Ey(i,:), b, D); %Q=0 in all subdomains
    %Assemble the stiffness matrix:
```

```
    K(ndep(i,:),ndep(i,:)) = K(ndep(i,:),ndep(i,:)) + Ke;
end

%% Convection Boundary (using edges)
ConvEdges = [];

for i=1:length(e)
    if (e(5,i) == 4)
        ConvEdges = [ConvEdges; e(1,i) e(2,i)];

    elseif ( (e(5,i) == 3) && (coord(e(1,i),1) >= 0.0002-1e-6) ...
            && (coord(e(2,i),1) >= 0.0002-1e-6) )
        ConvEdges = [ConvEdges; e(1,i) e(2,i)];
    end
end


%Stiffness matrix for convection terms
startNodes = ConvEdges(:,1); %Nodes in the left column of ConvEdges
endNodes = ConvEdges(:,2); %Nodes in the right column of ConvEdges

fc = zeros(length(K),1); %Creates empty column vector for storage
%of boundary convection terms

for i= 1:length(startNodes) %length(startNodes)=length(endNodes) ...
%= # edges in ConvEdges
    %Current edge length = sqrt((startx-endx)^2 + (starty-endy)^2)
    ELength = sqrt( (coord(startNodes(i),1)-coord(endNodes(i),1))^2 ...
    + (coord(startNodes(i),2)-coord(endNodes(i),2))^2);
    Kce = ELength*(ac/6)*[2 1; 1 2];
    %Assemble the stiffness matrix:
    K(ConvEdges(i,:), ConvEdges(i,:)) = K(ConvEdges(i,:), ConvEdges(i,:)) + Kce;

    fce = ac*Tinf*(ELength/2)*[1; 1]; %Convection terms along current edge
    fc(ConvEdges(i,:)) = fc(ConvEdges(i,:)) + fce; %Assemble convection vector
end

%% Boundary 'force' vector, temperature
FluxEdges = [];

for i=1:length(e)
    if ( (e(5,i) == 3) && (coord(e(1,i),1) <= 0.0002+1e-6)...
            && (coord(e(2,i),1) <= 0.0002+1e-6) )
        FluxEdges = [FluxEdges; e(1,i) e(2,i)];
    end
end

%Creates empty column vector for storage of boundary 'forces':
fb = zeros(length(K),1);
startNodes = FluxEdges(:,1); %Nodes in the left column of FluxEdges
endNodes = FluxEdges(:,2); %Nodes in the right column of FluxEdges

for i= 1:length(startNodes) %length(startNodes)=length(endNodes) ...
%= # edges in FluxEdges
    %Current edge length = sqrt((startx-endx)^2 + (starty-endy)^2)
    ELength = sqrt( (coord(startNodes(i),1)-coord(endNodes(i),1))^2...
    + (coord(startNodes(i),2)-coord(endNodes(i),2))^2);
    fbe = qel*(ELength/2)*[1; 1]; %Flux terms along current edge
    %Assemble flux vector:
    fb(FluxEdges(i,:)) = fb(FluxEdges(i,:)) + fbe;
end

%% Transients
C = zeros(ndof); %Create empty global C-matrix
for i=1:nelm %Loop over all 712 triangular elements
    if t(4,i) == 1
        %If the subdomain nr (found in 4th column of t-matrix) is 1:
        roc = ropcb*cpcb; %Then the current subdomain is the PCB
    elseif t(4,i) == 2
        roc = rosmd*csmd; %Subdomain number 2 is the SMD
    else
        roc = rosol*csol; %Subdomain number 3 is the solder
    end
    Ce = plantml(Ex(i,:), Ey(i,:), roc);
    C(ndep(i,:),ndep(i,:)) = C(ndep(i,:),ndep(i,:)) + Ce; %Assemble C-matrix
end

%% Time integration
f = fc + fb; %Global force vector, temperature
%Numerical integration using implicit Euler method:
a = eulerFEMint(C, K, f, T0, 0, 100, 1000);

%% Thermal distribution plots
ed1 = extract(edof,a(:,1));
figure
fill(Ex',Ey',ed1');
c= colorbar
label = ylabel(c,'\circC');
set(label,'Rotation',0);
title('Temperature distribution after 0.1 s');
xlabel('x (m)');
ylabel('y (m)');

ed2 = extract(edof,a(:,250));
figure
fill(Ex',Ey',ed2');
c= colorbar
label = ylabel(c,'\circC');
set(label,'Rotation',0);
title('Temperature distribution after 25 s');
xlabel('x (m)');
ylabel('y (m)');

ed3 = extract(edof,a(:,1000));
figure
fill(Ex',Ey',ed3');
c= colorbar
label = ylabel(c,'\circC');
set(label,'Rotation',0);
title('Temperature distribution after 100 s');
xlabel('x (m)');
ylabel('y (m)');

%Stationary temperature distribution
```

```
statT = solveq(K,f);
deltaT = statT-T0; %Temperature deviation from T_0 to stationary ...
%disitribution in each node
figure
fill(Ex',Ey',edstat');
c= colorbar
label = ylabel(c,'\circC');
set(label,'Rotation',0);
title('Stationary temperature distribution');
xlabel('x (m)');
ylabel('y (m)');


%% Part 2: Plane elasticity
%Stationary temperature distribution
statT = solveq(K,f);
deltaT = statT-T0; %Temperature deviation from T_0 to stationary ...
%disitribution in each node

ndofs = 2*ndof; %Degrees of freedom in solid mechanics is two per node

%Each node, i. corresponds to the degrees of freedom 2i-1 och 2i
%The new edof-matrix has dimensions [nelm,7] and is called edofs
edofs(:,1)=1:nelm;  %First column in edofs contains the elemtent indices
edofs(:,[2 4 6])=(2*t(1:3,:)-1)'; %Columns 2, 4 & 6 are 2*node index-1 ...
%for the current element
edofs(:,[3 5 7])=2*t(1:3,:)'; %Columns 3, 5 & 7 are 2*node index

%% Compute global temperature force vector:
b=1; %The thickness of the structure bears no relevance, set to 1
ep = [2, b]; %2: plane strain, b: element thickness
ndeps=edofs(:,2:7); %Takes out only nodal dependencies from edofs
fdeltaT = zeros(ndofs, 1); %Create empty global thermal strain vector
elemDeltaT = zeros(nelm,1); %Create empty vector for storing the average
%temperature deviation in each element

%Define elasticity module for the three subdomains:
Dpcb = Epcb/((nypcb+1)*(1-2*nypcb))*[1-nypcb nypcb 0;nypcb 1-nypcb 0;...
    0 0 1/2*(1-2*nypcb)]; %Isotropic material
Dsmd = Esmd/((nysmd+1)*(1-2*nysmd))*[1-nysmd nysmd 0;nysmd 1-nysmd 0;...
    0 0 1/2*(1-2*nysmd)]; %Isotropic material
Dsol = Esol/((nysol+1)*(1-2*nysol))*[1-nysol nysol 0;nysol 1-nysol 0;...
    0 0 1/2*(1-2*nysol)]; %Isotropic material

for i=1:nelm %Loop over all 712 triangular elements
    if t(4,i) == 1
        %If the subdomain nr (found in 4th column of t-matrix) is 1
        D = Dpcb; %Then the current subdomain is the PCB
        alpha = alphapcb;
    elseif t(4,i) == 2
        D = Dsmd; %Subdomain number 2 is the SMD
        alpha = alphasmd;
    else
        D = Dsol; %Subdomain number 3 is the solder
        alpha = alphasol;
    end
    %Compute the average temperature deviation in the current element:
```

```
    elemDeltaT(i) = (deltaT(t(1,i)) + deltaT(t(2,i)) + deltaT(t(3,i)))/3;
    %Compute the contribution of the element to fdeltaT
    fdeltaTe = elemDeltaT(i)*plantf(Ex(i,:), Ey(i,:), ep, (alpha*D*[1; 1; 0])');
    %Assemble the global temperature strain vector:
    fdeltaT(ndeps(i,:)) = fdeltaT(ndeps(i,:)) + fdeltaTe;
end

%% Compute stiffness matrix for plane strain
%we call it Ks to differentiate it from K for temp.

Ks = zeros(ndofs);
for i=1:nelm %Loop over all 712 triangular elements
    if t(4,i) == 1
        %If the subdomain nr (found in 4th column of t-matrix) is 1
        D = Dpcb; %Then the current subdomain is the PCB
    elseif t(4,i) == 2
        D = Dsmd; %Subdomain number 2 is the SMD
    else
        D = Dsol; %Subdomain number 3 is the solder
    end
    Ke = plante(Ex(i,:), Ey(i,:), ep, D); %Elementwise contribution
    %Assemble the stiffness matrix:
    Ks(ndeps(i,:),ndeps(i,:)) = Ks(ndeps(i,:),ndeps(i,:)) + Ke;
end

%% Boundary conditions
%Nodes where u_x=0 are in bcRight and bcSym
%On bcBottom u_y=0
bc = [2*bcRight-1; 2*bcSym-1; 2*bcBottom];
bc(:,2) = 0;
%On remaining boundaries the traction vector is 0

%%Calculate solution with solveq
[u,Q] = solveq(Ks,fdeltaT,bc);

ed = extract(edofs,u);
figure
eldraw2(Ex,Ey,[1,4,3]);
hold on
%Displays the element distortions with a factor 100 magnification:
eldisp2(Ex,Ey,ed,[1 5 3],100);
title('Deformed mesh, scale factor 100')
legend('Original mesh','Deformed mesh')


%%
es = zeros(nelm,3);
et = zeros(nelm,3);
sigmazz = zeros(nelm,1);
for i=1:nelm %Loop over all 712 triangular elements
    if t(4,i) == 1
        %If the subdomain nr (found in 4th column of t-matrix)=1
        D = Dpcb; %Then the current subdomain is the PCB
        E = Epcb;
        ny = nypcb;
        alpha = alphapcb;
    elseif t(4,i) == 2
```

```
        D = Dsmd; %Subdomain number 2 is the SMD
        E = Esmd;
        ny = nysmd;
        alpha = alphasmd;
    else
        D = Dsol; %Subdomain number 3 is the solder
        E = Esol;
        ny = nysol;
        alpha = alphasol;
    end
    %Compute the stresses and strains of the element:
    [es(i,:), et(i,:)] = plants(Ex(i,:), Ey(i,:), ep, D, ed(i,:));
    %Take thermoelasticity into account (not done automatically by plants):
    es(i,:) = es(i,:)-(alpha*E*elemDeltaT(i))/(1-2*ny)*[1 1 0];
    %Sigmazz = ny(sigmaxx+sigmayy)-alpha*E*deltaT according to (13.42)
    sigmazz(i) = ny*(es(i,1)+es(i,2))-alpha*E*elemDeltaT(i);
end

%% Compute von Mises effective stress for each element
Seff_el = zeros(nelm, 1);
Seff_nod = zeros(length(coord),1);

for i=1:nelm %Loop over all 712 triangular elements
    %Find the effective von Mises stress for each element:
    Seff_el(i) = sqrt(es(i,1)^2+es(i,2)^2+sigmazz(i)^2-es(i,1)*es(i,2)...
        -es(i,1)*sigmazz(i)-es(i,2)*sigmazz(i)+3*es(i,3)^2);
end

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

%% Plot stresses
figure
eds = extract(edof,Seff_nod);
fill(Ex',Ey',eds');
colorbar
```

The functions **eulerFEMint** and **eulerFEMstep** are given below.

```
function a = eulerFEMint(C, K, f, T0, tstart, tend, N)
%
%   N is the number of steps
%   tstart is the initial time
%   tend is finish time of the recursion
%
%   a is the computed nodal temperatures for t=tstart+deltat:tend

%Matrix containing the nodal temperatures at time t in column t:
a = zeros(length(K),N);

%Sets the initial nodal temperatures to T_0:
anew = T0*ones(length(K),1);
%Determines step size from time interval and number of steps:
deltat = (tend-tstart)/N;
```

```
for i=1:N %Loop over steps
    %Recursion for nodal temperatures:
    anew = eulerFEMstep(C, K, f, deltat, anew);
    a(:,i) = anew;
end

end


function anew = eulerFEMstep(C, K, f, deltat, aold)
%eulerFEM Computes the new nodal temperatures using implicit Euler
%   C is the transient matrix
%   K is the stiffness matrix (including convection)
%   deltat is the time between aold and anew
%   f is the RHS vector (from boundary conditions, flux and convection)

 anew = (C+deltat*K)\(C*aold+deltat*f);
end
```