LUNDS UNIVERSITET

Lunds Tekniska Högskola

FHLF25

Project: The Finite Element Method

By: Ali Bakly, Davy Than

2023 May

Contents

1 Introduction

For our project in Finite Element Method and Introduction to Strength of Materials (FHLF25) we are tasked to investigate figure 1. The figure shows a thermo-mechanical gripper that generates a tip motion that grips small objects. There all also thermal boundary conditions $q_n = h$ and $q_n = 0$ at the left side of the gripper. The rest of the boundaries has Newton convection $q_n = \alpha_c(T - T_\infty)$ where $\alpha_c = 40\, W/m^2K$. Note also that the left boundary is clamped and we assume the gripper can't move out-of-plane meaning that plane strain is assumed.

To investigate the gripper's performance we will implement a finite element analysis. Specifically we want to examine these three problem formulations:

1. Finding the stationary temperature distribution of the gripper if it is in an environment of $T_\infty = 18\,°C$ and heat flux $h = -1 \cdot 10^5\; W/m^2$
2. Investigate at what time the temperature reaches 90% of the max temperature in problem 1). Assuming once again $T_{\text{inf}} = 18\,°C$, $h = -1 \cdot 10^5\; W/m^2$ and also initial temperature $T_0 = 18\,°C$
3. Plot and determine the effective von Mises stress field and displacement fields.

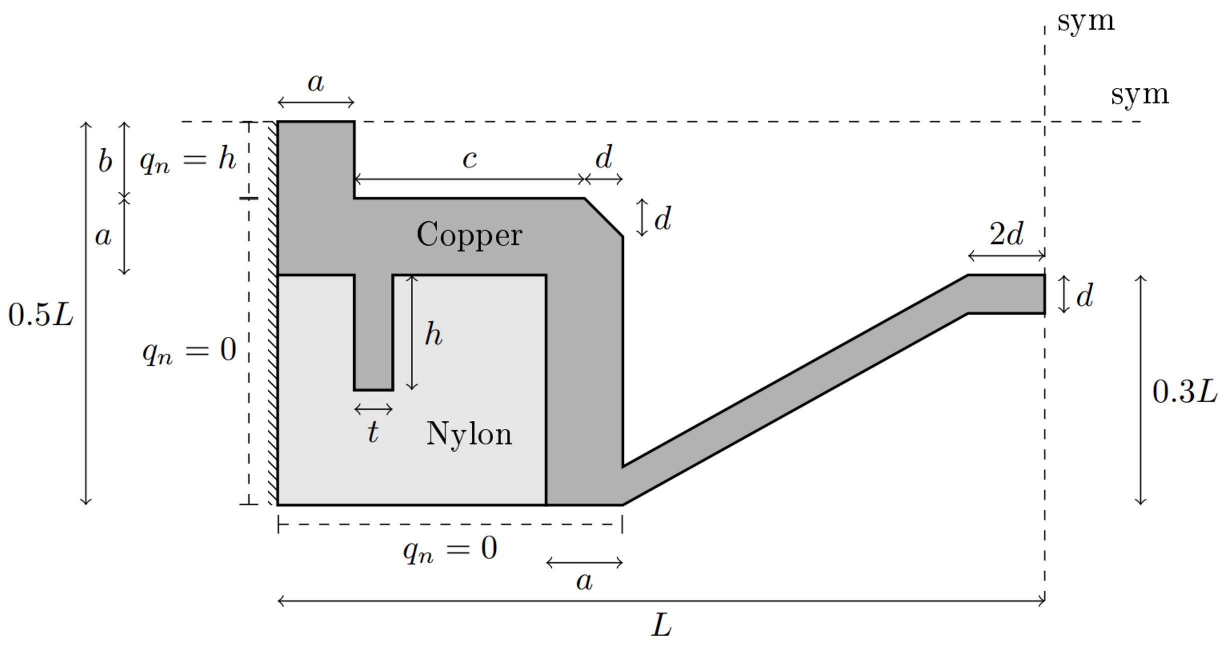


Figure 1: A quarter of a thermo-mechanical gripper

In the two tables below we find the relevamt dimensions and material parameters.

L	a	b	c	d	h	t
5 mm	$0.1L$	$0.1L$	$0.3L$	$0.05L$	$0.15L$	$0.05L$

Table 1: Dimensions for figure 1.

	Copper	Nylon
Young's modulus, E [GPa]	128	3.00
Poission's ratiom ν [-]	0.36	0.39
Expansion coefficient, α	$17.6 \cdot 10^{-6}$	$80 \cdot 10^{-6}$
Density, ρ [1/K]	8930	1100
Specific heat, c_p [J/kg-K]	386	1500
Thermal conductivity, k [W/m-K]	385	0.26

Table 2: Material data

2 Procedure

Regarding the theory parts of this section we would like to mention that the source of literature was [3], and more detailed explanations can be found there.

2.1 PDEtool Mesh and Boundaries

Our first step to implement the finite element method and solve the given problems is to draw our figure 1 and mesh it as seen in figure 2 using Matlab's PDEtool. The figure in PDEtool is conviniently divided in to different subdomain and edges. As a result, after exporting the mes, we get three matrices p, e and t describing or geometry. They contain x, y cordinates for every node, all triangular element and which subdomain they lie within etc.

Before delving deeper into the theory we should also keep in mind what kind of conditions we have at all our boundaries. All information on the boundaries is found in the plot in figure (3) below.

2.2 Transient Heat Transfer

2.2.1 Finite Element Formulation

The procedure of finding the FE formulation of the problem consists of first setting up a balance equation then deriving the strong form and the weak form of the equation and finally choosing the weight function. In a body with the specific heat capacity c and density ρ where generated internal heat supply per unit of area is Q and the heat flux is $\boldsymbol{q}$, the balance equation is given by

$$\int_V Q \, dV - \int_S q_n \, dS = \int_V \rho c \dot{T} \, dV. \tag{1}$$

Here V denotes the volume of the body, S denotes the surface, $T = T(x, y, z, t)$ denotes the temperature and q_n is the heat flux component normal to the surface, that is $q_n = \boldsymbol{q}^T \boldsymbol{n}$. Equation (1) together with Gauss' divergence theorem, which states

$$\int_S q_n \, dS = \int_V \operatorname{div} \boldsymbol{q} \, dV,$$

yields

$$\int_V \Big(\rho c \dot{T} + \operatorname{div} \boldsymbol{q} - Q \Big) \, dV = 0. \tag{2}$$

Since equation (2) holds for an arbitrary volume V, the integrand must be equal to zero. Thus, the strong form reads

$$\rho c \dot{T} + \operatorname{div} \boldsymbol{q} - Q = 0. \tag{3}$$

Thanks to the symmetry of our problem we can view it as a 2D problem and thus make the changes $V \longrightarrow S$, $S \longrightarrow \mathcal{L}$ and introduce a thickness t of the material. Since we assume the thickness is constant

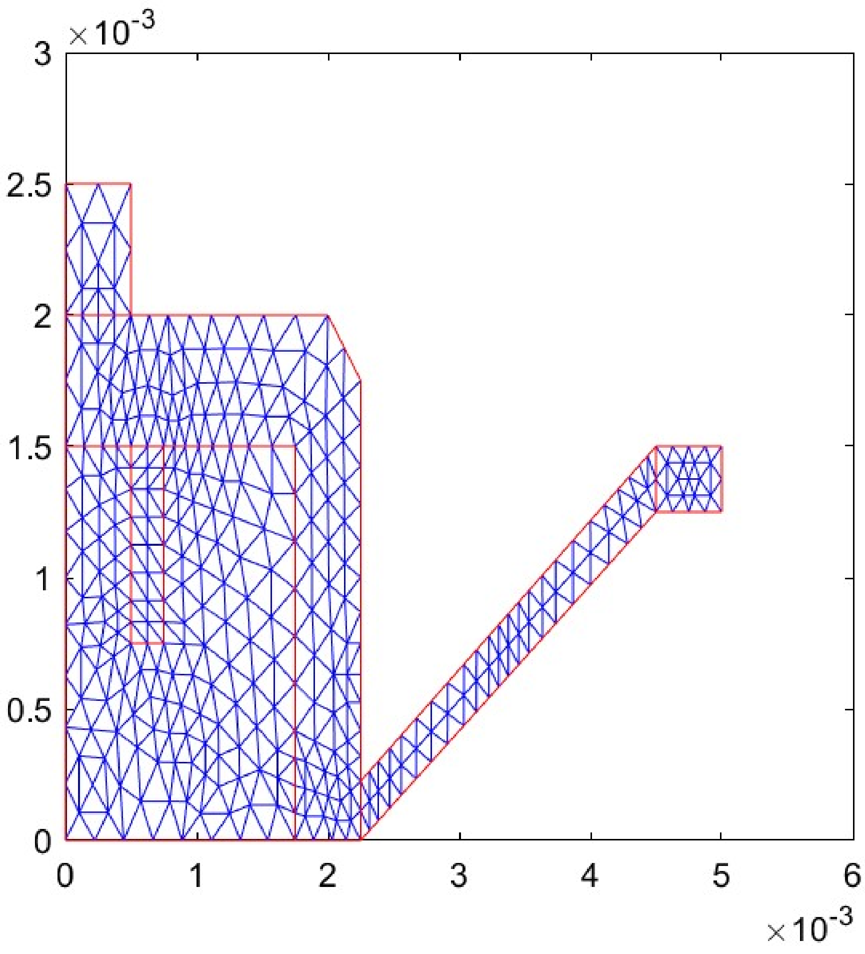


Figure 2: Mesh of figure 1 (axes in meters).

in all of the material, the strong form in equation (3) is unaffected. Altering t can be used as a sanity check when solving this heat transfer problem since it should not affect the solutions.

When deriving the weak form of corresponding to equation (3) we first multiply by a weight function v and integrate over the body (using S instead of V):

$$\int_S v\rho c\dot{T}\, dV + \int_S v \operatorname{div} \boldsymbol{q}\, dV - \int_S vQ\, dV = 0.$$

Using the fact that no internal heat is generated, $Q = 0$, and the Green-Gauss theorem on the integral that has a divergence operator in its integrand we find

$$\int_S v\rho c\dot{T}\, dS + \int_{\mathcal{L}} vq_n\, d\mathcal{L} - \int_S (\boldsymbol{\nabla} v)^T \boldsymbol{q}\, dS = 0. \tag{4}$$

Dividing up the line integral in equation (4) with the help of figure **??** yields

$$\int_{\mathcal{L}} vq_n\, d\mathcal{L} = \int_{\mathcal{L}_c} v\alpha_c(T - T_\infty)\, d\mathcal{L}_c + \int_{\mathcal{L}_h} vh\, d\mathcal{L}_h \tag{5}$$

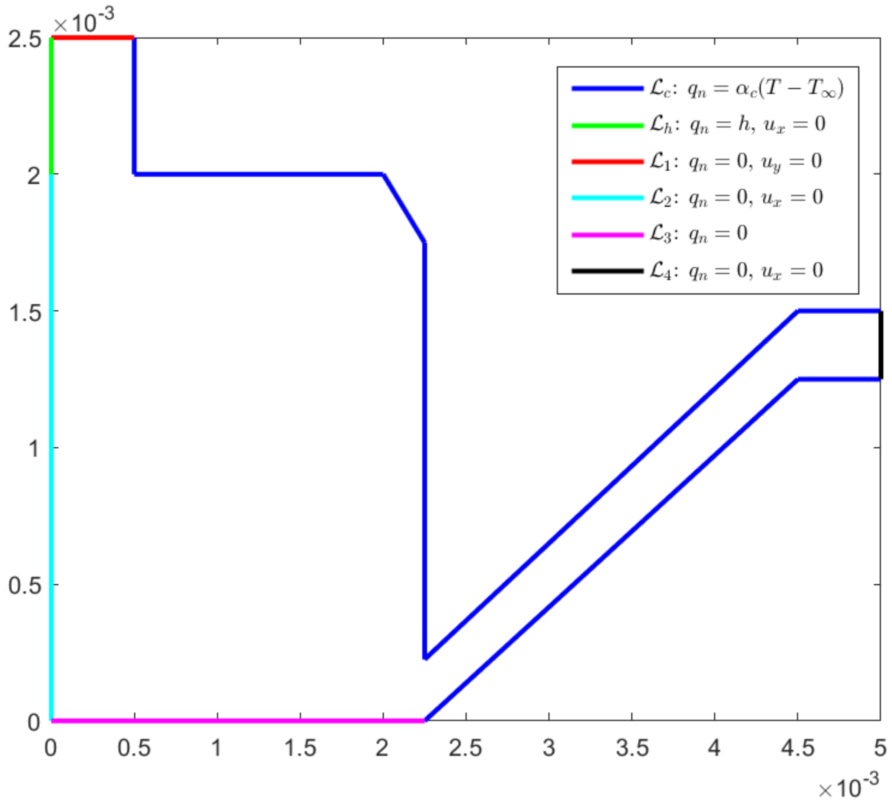


Figure 3: Labeling of each boundary and corresponding conditions.

Equation (5) together with equation (4) (and some convenient rearranging) gives us the weak form

$$\int_S v\rho c\dot{T}\,dS - \int_S (\boldsymbol{\nabla} v)^T \boldsymbol{q}\,dS + \int_{\mathcal{L}_c} v\alpha_c T\,d\mathcal{L}_c = -\int_{\mathcal{L}_h} vh\,d\mathcal{L}_h + \int_{\mathcal{L}_c} v\alpha_c T_\infty\,d\mathcal{L}_c. \tag{6}$$

Furthermore, Fourier's law tells us that

$$\boldsymbol{q} = -\boldsymbol{D}\boldsymbol{\nabla} T. \tag{7}$$

To find the FE formulation we make use the Galerkin method to choose the weight functions from the interpolation

$$T = \boldsymbol{N}\boldsymbol{a} \implies v = \boldsymbol{N}\boldsymbol{c} = \boldsymbol{c}^T\boldsymbol{N}^T, \tag{8}$$

where $\boldsymbol{N}$ are the element shape functions an $\boldsymbol{c}$ is an arbitrary vector. Letting $\boldsymbol{\nabla}\boldsymbol{N} = \boldsymbol{B}$ also gives

$$\boldsymbol{\nabla} v = \boldsymbol{B}\boldsymbol{c}, \quad \boldsymbol{\nabla} T = \boldsymbol{B}\boldsymbol{a} \tag{9}$$

We also note that

$$\dot{T} = \dot{\boldsymbol{N}\boldsymbol{a}} = \dot{\boldsymbol{N}}\boldsymbol{a} + \boldsymbol{N}\dot{\boldsymbol{a}} = \boldsymbol{N}\dot{\boldsymbol{a}}. \tag{10}$$

Using equations (7), (8), (9) and (10) together with weak form in equation (6) results in

$$\boldsymbol{c}^T\left(\int_S \boldsymbol{N}^T\rho c\boldsymbol{N}\,dS\,\dot{\boldsymbol{a}} + \int_S \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}\,dS\,\boldsymbol{a} + \int_{\mathcal{L}_c} \boldsymbol{N}^T\alpha_c\boldsymbol{N}\,d\mathcal{L}_c\,\boldsymbol{a}\right) =$$
$$\boldsymbol{c}^T\left(-\int_{\mathcal{L}_h} \boldsymbol{N}^T h\,d\mathcal{L}_h + \int_{\mathcal{L}_c} \boldsymbol{N}^T\alpha_c T_\infty\,d\mathcal{L}_c\right).$$

Since $\boldsymbol{c}^T$ is arbitrary the terms inside the parentheses must be equal to each other ans the FE formulation is abbreviated

$$\boldsymbol{C}\dot{\boldsymbol{a}} + (\boldsymbol{K}_n + \boldsymbol{K}_c)\boldsymbol{a} = \boldsymbol{f}_b + \boldsymbol{f}_c, \tag{11}$$

where

$$\begin{aligned} \boldsymbol{C} &= \int_S \boldsymbol{N}^T \rho c \boldsymbol{N}\, dS & \boldsymbol{K}_n &= \int_S \boldsymbol{B}^T \boldsymbol{D}\boldsymbol{B}\, dS & \boldsymbol{K}_c &= \int_{\mathcal{L}_c} \boldsymbol{N}^T \alpha_c \boldsymbol{N}\, d\mathcal{L}_c \\ \boldsymbol{f}_b &= -\int_{\mathcal{L}_h} \boldsymbol{N}^T h\, d\mathcal{L}_h & \boldsymbol{f}_c &= \int_{\mathcal{L}_c} \boldsymbol{N}^T \alpha_c T_\infty\, d\mathcal{L}_c. \end{aligned} \tag{12}$$

We mention yet again that we did not account for varying thickness and that such a variable could be introduced to these equations. In the code it is tested whether thickness affects the solutions (it should not).

2.2.2 Finding the Stiffness Matrix

The Finite Element toolbox CALFEM [1] is used to implement and solve the problem. Finding the full stiffnes matrix $\boldsymbol{K} = \boldsymbol{K}_n + \boldsymbol{K}_c$ given in equation (12) is broken down into finding the 3×3 element matrices of these and adding them, $\boldsymbol{K}_n^e + \boldsymbol{K}_c^e$. It is important to note that $\boldsymbol{K}_c^e$ is only added if the current element has an edge that is on the convection boundary. Other than doing this for each element in a loop, the element stiffens matrix is also assembled into the full $nnod \times nnod$ stiffnes matrix. Here $nnod$ is the number of nodes which also in this case is the same number as the degrees of freedom, hence why we do not introduce a variable $ndof$. To be able to assemble the element stiffness matrices into the full stiffnes matrix an element topology matrix `edof` is defined by representing which nodes/degrees of freedom that each element is connected to. Finding $\boldsymbol{K}_n^e$ is done by using the CALFEM function `flw2te` where among other things the $\boldsymbol{D}$ matrix is specified according to the material the current element resides in. Finding $\boldsymbol{K}_c^e$ is done analytically. It can be shown for 3-node elements that

$$\boldsymbol{K}_c^e = \int_{\mathcal{L}_c} \boldsymbol{N}^{e^T} \alpha_c \boldsymbol{N}^e\, d\mathcal{L}_c = \frac{\alpha_c L}{6} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix},$$

where L is the length of the particular convection edge.

2.2.3 Boundary Conditions and Finding the Force Vector

All the boundary conditions are in the form of natural boundary conditions and thus they are all caught in the vectors $\boldsymbol{f}_b$ and $\boldsymbol{f}_c$ as specified in equation (12). Just like before this done in a loop in an elementwise fashion. If our element has en edge in the convection boundary we account for this by adding $\boldsymbol{f}_c^e$ and correspondingly , for an element with a flux boundary we add $\boldsymbol{f}_b^e$ to the element force vector. After the loop and the assembling we have a full force vector $\boldsymbol{f} = \boldsymbol{f}_b + \boldsymbol{f}_c$ of size $nnod \times nnod$.

Finding $\boldsymbol{f}_b$ and $\boldsymbol{f}_c$ is done analytically. It can be shown that

$$\boldsymbol{f}_c^e = \frac{\alpha_c T_\infty L}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{f}_b^e = -\frac{hL}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

2.2.4 Time Integration for the Transient

With both $\boldsymbol{K} = \boldsymbol{K}_n + \boldsymbol{K}_c$ and $\boldsymbol{f} = \boldsymbol{f}_b + \boldsymbol{f}_c$ calculated, the stationary problem can be solved with the CALFEM routine `solveq`. To simulate transient heat trasnfe and include time dependency we have to account for the $\boldsymbol{C}\dot{\boldsymbol{a}}$ term in equation (11) with some type of time stepping scheme. We make an approximation of the time derivative with

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

We also use the implicit theta method which for our case states

$$\begin{aligned} \boldsymbol{a} &= \theta \boldsymbol{a}_{n+1} + (1-\theta)\boldsymbol{a}_n \\ \boldsymbol{f} &= \theta \boldsymbol{f}_{n+1} + (1-\theta)\boldsymbol{f}_n. \end{aligned} \tag{14}$$

With $\theta = 1$, and combining equations (11, 13, 14) and noting that $\boldsymbol{f}$ is time-independet we can find

$$\boldsymbol{a}_{n+1} = (\boldsymbol{C} + \Delta t \boldsymbol{K})^{-1}(\Delta t \boldsymbol{f} + \boldsymbol{C}\boldsymbol{a}_n). \tag{15}$$

We remind that $\boldsymbol{K} = \boldsymbol{K}_n + \boldsymbol{K}_c$ and $\boldsymbol{f} = \boldsymbol{f}_b + \boldsymbol{f}_c$. Now it remains to calculate the $\boldsymbol{C}$ matrix given in equation (12). This is done with the provided function `plantml`, yet again in a elementwise fashion where we check in which material the element resides ρ and c, calculate $\boldsymbol{C}^e$ and assemble. All the nodal temperatures are stored in a matrix where each column represents a time point, depending on the size of our time step and how long to run it for. In our implementation of the problem we use the time step $\Delta t = 0.05\,\mathrm{s}$ and we let the simulation run for 100 s.

2.3 Thermoelasticity

2.3.1 Finite Element Formulation

We derive the equilibrium equations (the strong form in this problem). Assuming a body with surface S and volume V has a traction vector/stress tensor $\boldsymbol{t}$ acting on S and a body force vector $\boldsymbol{b}$ on V then we have the balance

$$\int_S \boldsymbol{t}\, dS + \int_V \boldsymbol{b}\, dV = \boldsymbol{0}. \tag{16}$$

Cauchys formula and Gauss' divergence theorem yields

$$\begin{cases} t_x^T + b_x = 0 \\ t_y^T + b_y = 0 \\ t_z^T + b_z = 0 \end{cases}$$

or equivalently

$$\tilde{\boldsymbol{\nabla}}^T \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0}, \tag{17}$$

where

$$\tilde{\boldsymbol{\nabla}}^T = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} & 0 \\ 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial z} \\ 0 & 0 & \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix}, \quad \boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{yz} \end{bmatrix} \tag{18}$$

For the plain strain problem this is reduced to

$$\tilde{\boldsymbol{\nabla}}^T = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \end{bmatrix} \boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{bmatrix},$$

where we need to calculate σ_{zz} separately with the relation

$$\sigma_{zz} = \nu(\sigma_{xx} + \sigma_{yy}) - \alpha E \Delta T. \tag{19}$$

After the usual procedure of finding the weak form of equation (16), that is multiplying by weight functions $\boldsymbol{v}$ integrating over the body and applying Green-Gauss theorem we retrieve

$$\int_{\mathcal{L}} \boldsymbol{v}^T \boldsymbol{t} d\mathcal{L} - \int_S (\tilde{\boldsymbol{\nabla}} \boldsymbol{v})^T \boldsymbol{\sigma} dS + \int_S \boldsymbol{v}^T \boldsymbol{b}\, dS = \boldsymbol{0}. \tag{20}$$

Here we made the changes $V \longrightarrow S$, $S \longrightarrow \mathcal{L}$ since as before, thanks to the symmetry of our problem we can view it as a 2D problem. We can connect the displacement $\boldsymbol{u}$ to the stress via the two relations

$$\boldsymbol{\sigma} = \boldsymbol{D}(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{\Delta T}), \;\; \boldsymbol{\varepsilon} = \tilde{\boldsymbol{\nabla}} \boldsymbol{u} \implies \boldsymbol{\sigma} = \boldsymbol{D} \tilde{\boldsymbol{\nabla}} \boldsymbol{u} - \boldsymbol{D} \boldsymbol{\varepsilon}^{\Delta T}. \tag{21}$$

Note that we accounted for the thermoelastic strain by subtracting from the the total strain, since we are only interested on the elastic strain. The constitutive matrix is given by

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

where E denotes Young's modulus and ν denotes Poisson's ratio.

We make the usual approximations with the shape function matrix $\boldsymbol{N}$ and the nodal displacements $\boldsymbol{a}$.

$$\boldsymbol{u} = \boldsymbol{N}\boldsymbol{a} \implies \boldsymbol{\varepsilon} = \tilde{\boldsymbol{\nabla}} \boldsymbol{u} = \tilde{\boldsymbol{\nabla}} \boldsymbol{N} \boldsymbol{a} = \boldsymbol{B}\boldsymbol{a}, \tag{22}$$

Here we have to account for having two degrees of freedom per node,

$$\boldsymbol{N} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & \ldots & N_{nnod} & 0 \\ 0 & N_1 & 0 & N_2 & \ldots & 0 & N_{nnod} \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} u_{x_1} \\ u_{y_1} \\ u_{x_2} \\ u_{y_2} \\ \ldots \\ u_{x_{nnod}} \\ u_{y_{nnod}} \end{bmatrix}.$$

Using Galerkins method we have

$$\boldsymbol{v} = \boldsymbol{N}\boldsymbol{c} \implies \tilde{\boldsymbol{\nabla}} \boldsymbol{v} = \boldsymbol{B}\boldsymbol{c}. \tag{23}$$

Equations (21, 22, 23) together with the weak form in equation (20) yields

$$\boldsymbol{c}^T \left(\int_{\mathcal{L}} \boldsymbol{N}^T \boldsymbol{t}\, d\mathcal{L} - \int_S \boldsymbol{B}^T \boldsymbol{D} (\boldsymbol{B}\boldsymbol{a} - \boldsymbol{\varepsilon}^{\Delta T})\, dS + \int_S \boldsymbol{N}^T \boldsymbol{b}\, dS \right) = \boldsymbol{0}.$$

Rearranging and using the fact that $\boldsymbol{c}$ is an arabitrary vector gives

$$\int_S \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B}\, dS\, \boldsymbol{a} = \int_{\mathcal{L}} \boldsymbol{N}^T \boldsymbol{t} + \int_S \boldsymbol{N}^T \boldsymbol{b}\, dS + \int_S \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{\varepsilon}^{\Delta T}\, dS$$

No external forces act on our surface in this specific problem, therefore we can set $\boldsymbol{b} = \boldsymbol{0}$. With this in mind we have the full FE formulation in its abbreviated form

$$\boldsymbol{K}\boldsymbol{a} = \boldsymbol{f}_b + \boldsymbol{f}_{\Delta T}, \tag{24}$$

where

$$\boldsymbol{K} = \int_S \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B}\, dS, \qquad \boldsymbol{f}_b = \int_{\mathcal{L}} \boldsymbol{N}^T \boldsymbol{t}, \qquad \boldsymbol{f}_{\Delta T} = \int_S \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{\varepsilon}^{\Delta T}\, dS. \tag{25}$$

Note that the thickness of the material was not included in the derivation, but such a parameter could be introduced.

2.3.2 Finding the Stiffness Matrix

The stiffness matrix $\boldsymbol{K}$ specified in equation (25) has the dimensions $ndof \times ndof$ where $ndof = 2 \cdot nnod$, since there are two displacements per node. Again, $\boldsymbol{K}$ is calculated by assembling $\boldsymbol{K}^e$ for all of the elements. Finding $\boldsymbol{K}^e$ is done by using the CALFEM function `plante`, where among other things, $\boldsymbol{D}$ and the type of the problem (plane strain) is specified. As before, the material parameters to construct the constitutive matrix $\boldsymbol{D}$ are found by checking in which material the current element resides in. The assebmling is done with the help of the topology matrix `edof_S`. Each element in `edof_S` will be connected to six degrees of freedom (displacements). If the node index is i then the x-displcement will have the index i but the y-displacement will have the index $i + nnod$.

2.3.3 Finding the Strain Vectors and the Displacement Field

To find the $ndof \times 1$ vector $\boldsymbol{f}_{\Delta T}$ we first need to specify $\boldsymbol{\varepsilon}^{\Delta T}$. The thermal strains for each element can be found with

$$\boldsymbol{\varepsilon}^{\overline{\Delta T}} = \alpha \overline{\Delta T} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix},$$

where α is the materials Expansion coefficient and $\overline{\Delta T}$ is the mean of the nodal temprature changes in the element between the start and the stationary state. The element vector $\boldsymbol{f}^e_{\Delta T}$ is then found with CALFEM function `plantf` where $\boldsymbol{D}\boldsymbol{\varepsilon}^{\Delta T}$ is specified according to the material in which the current element resides in. The full vector $\boldsymbol{f}_{\Delta T}$ is then found after assembling, again with the help of `edof_S`.

To find the displacements, technically we would also need to calculate $\boldsymbol{f}_b$. This is unnecessary since we know the displacements at some of the boundaries ($= 0$), and we also know that the traction vector $\boldsymbol{t} = \boldsymbol{0}$ at all the other boundaries since no forces act on these. The CALFEM solver `solveq` let's us do this by specifying a boundary condition matrix, where the known displacements values are associated with respective, as input parameter. The known displacements can be found in figure 3. After supplying the stiffness matrix $\boldsymbol{K}$, the vector $\boldsymbol{f}_{\Delta T}$ and boundary conditions to the solver `solveq` the displacements are found.

2.3.4 Finding the Von Mises Stress Field

The von Mises stress is defined as

$$\sigma_{\text{Mises}} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 - \sigma_{xx}\sigma_{yy} - \sigma_{xx}\sigma_{zz} - \sigma_{yy}\sigma_{zz} + 3\tau_{xy}^2 + 3\tau_{xz}^2 + 3\tau_{yz}^2}. \tag{26}$$

A $1 \times nelm$ vector `von_mises` is defined where the von Mises stress for each element will be inserted. The total strain for each element is calculated with the CALFEM function `plants` where $\boldsymbol{D}$, the previously calculated displacements, among other parameters, are given as input. We remind that this is the stress related to the total strain, and that we are searching for the the stress related only to the elastic strains. Thus according to equation 21 we need to subtract the element stresses calculated with `plants` by $\boldsymbol{D}\boldsymbol{\varepsilon}^{\overline{\Delta T}}$, that is

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{bmatrix} = \boldsymbol{\sigma}_{plants} - \boldsymbol{D}\boldsymbol{\varepsilon}^{\overline{\Delta T}}.$$

The stress σ_{zz} is calculated seperately according to equation (19). Once all the stresses are found the von Misses stress is calculated according to equation (26) for the current element and inserted into the vector `von_mises`.

To find the von Mises stress at each node we take the average of all connected elements von Mises stresses.

3 Results

3.1 The Heat Transfer Problem

Below, in figure 4, the solution to the stationary heat transfer problem is found.

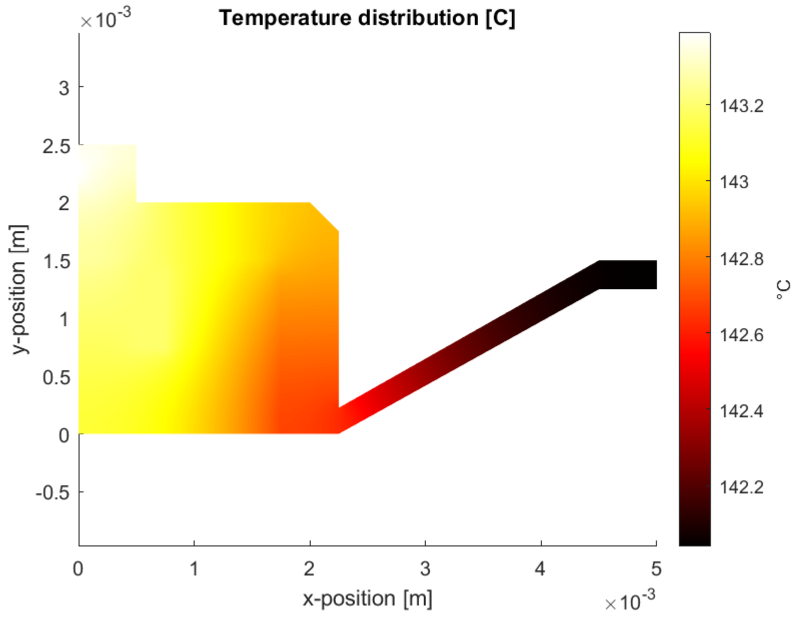
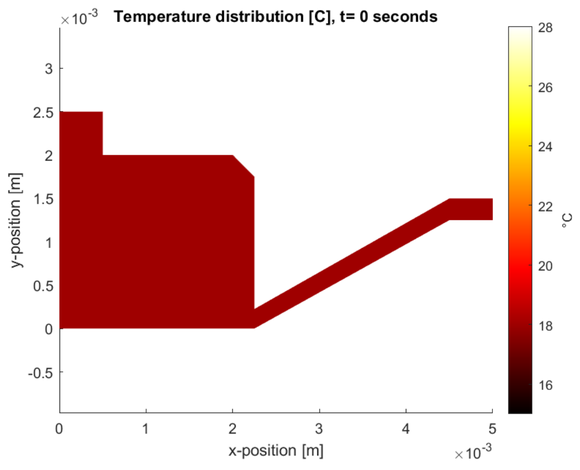
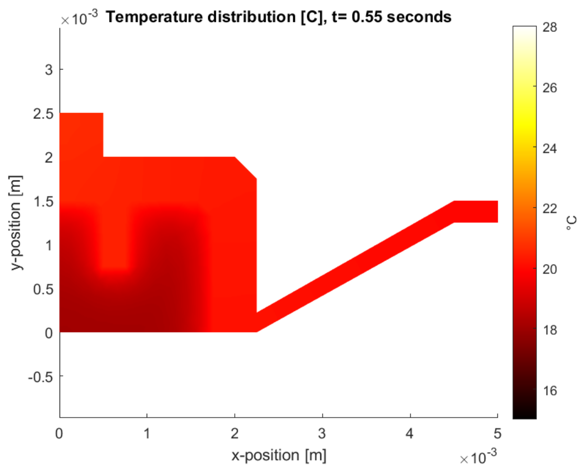


Figure 4: The solution to the stationary heat transfer problem.

The minimum temperature is 142.04 $^{\circ}C$ and the maximum temperature is 143.39 $^{\circ}C$.

For the the transient heat transfer we found that 90% of the max temperature of the stationary solution is reached after 76.95 s. We produce 5 plots at evenly spaced time points of the first 3% of 76.95 s in figure 5.





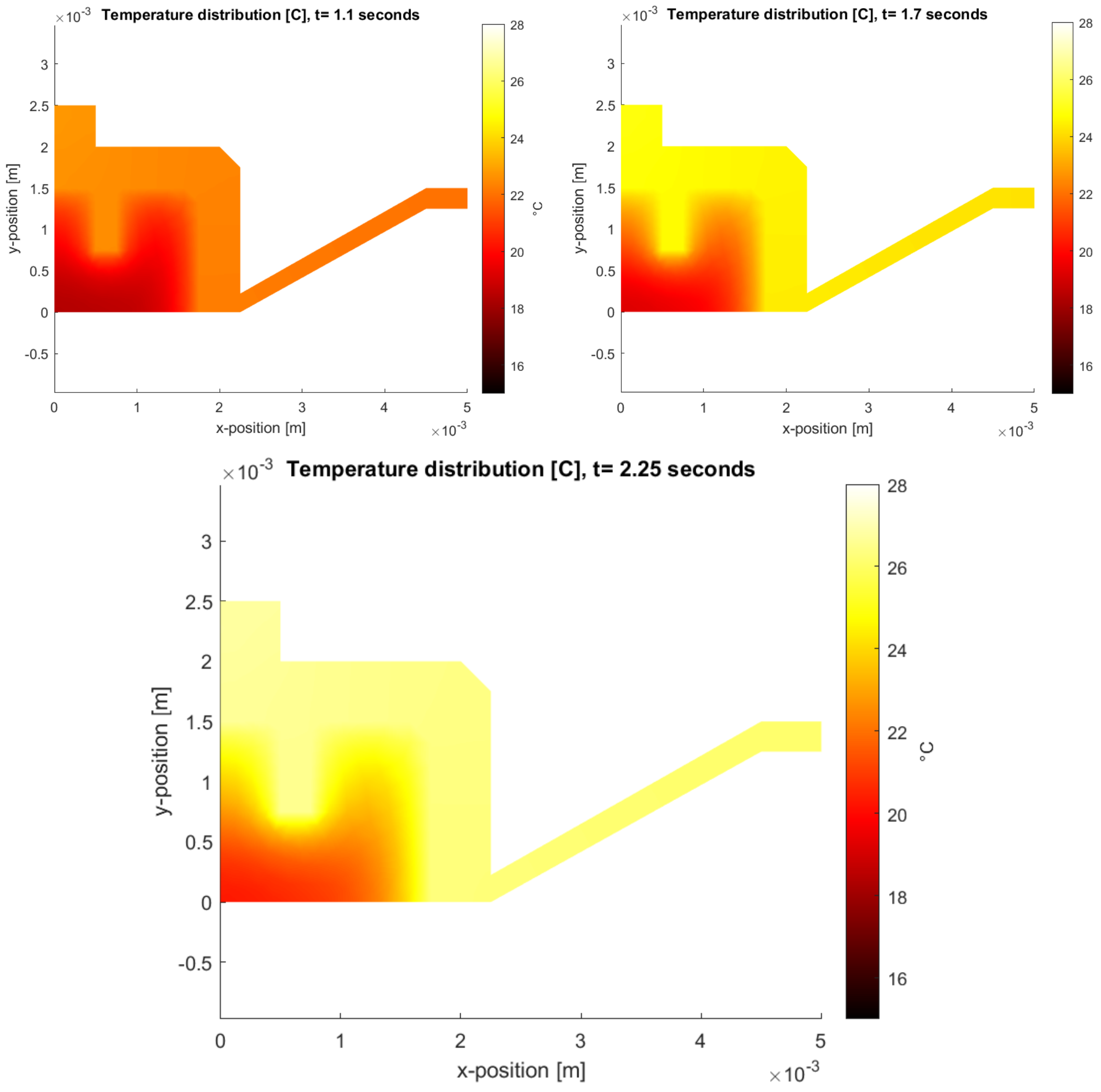


Figure 5: Computed transient temperatures at 5 snapshots.

3.2 Displacements and the Von Mises Stress Field

The displacement field, due to thermal expansion from the stationary temperature distribution is plotted with enhanced magnitude in figure 6. See short clip of the displacement and the transient heat transfer evolving through time.

The effective von Mises displacement field is also plotted in figure 7. The blue marker marks the point of highest stress on the gripper, at approximately $4.11 \cdot 10^8\ N/m^2$.

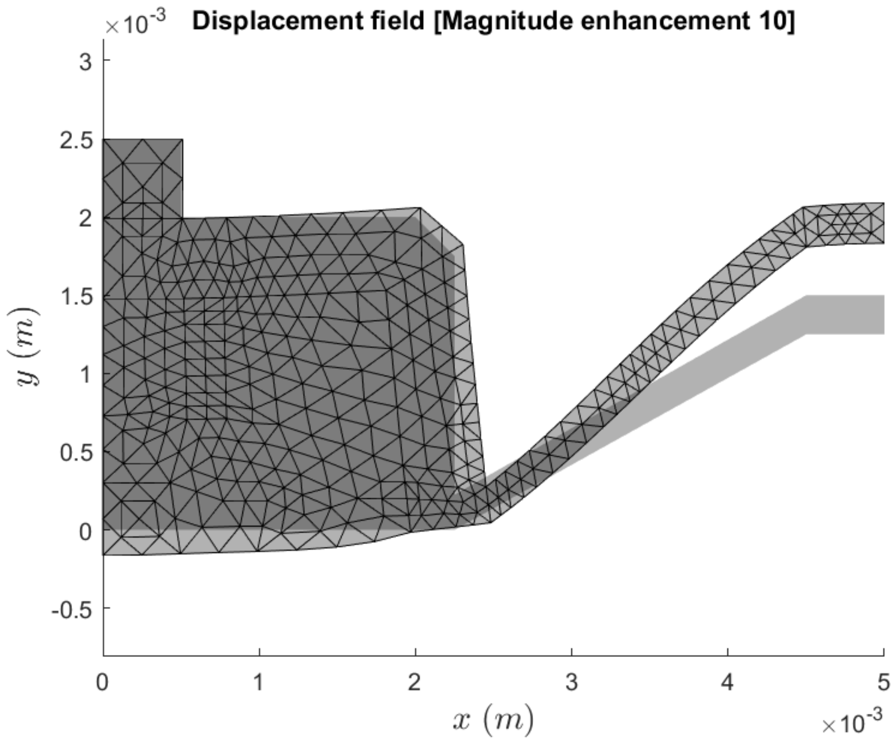


Figure 6: The displacement field, due to thermal expansion

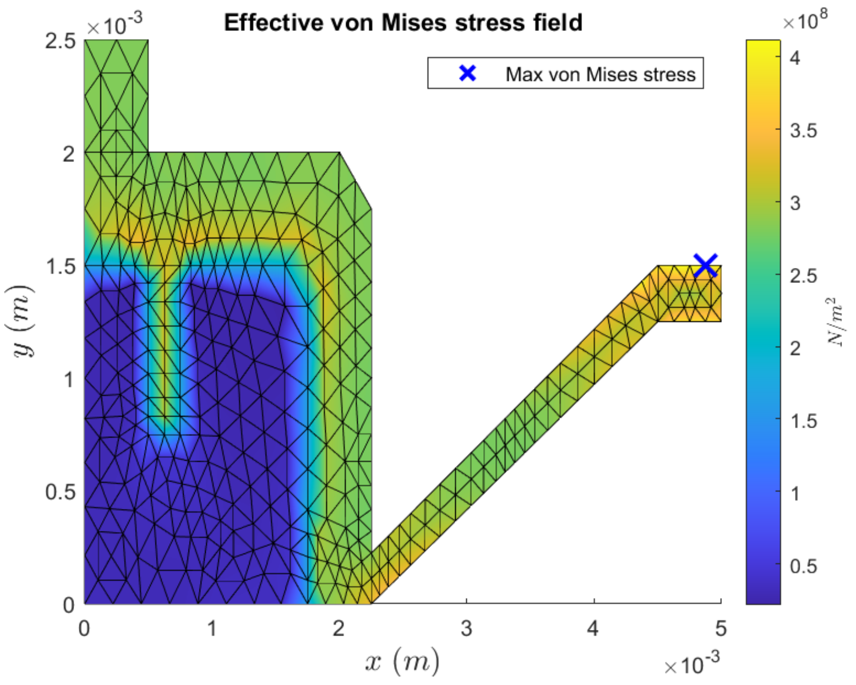


Figure 7: The effective von Mises stress field

4 Discussion

In the heat transfer problem for both the stationary and the transient case we see that the temperature is higher in the area where the gripper is made of copper compared to the nylon area. This seems highly probable since copper has a specific heat, c_p (the amount of heat needed to uniformly raise the temperature of the material divided by the mass) of 386 J/kg-K compared to nylon that has a a specific heat of 1500 J/kg-K. This means that it takes more energy to heat up nylon compared to copper and thus it makes sense that copper has a higher temperature in our plots. We also reach our maximum temperatures at the boundary $\mathcal{L}_h$ which seems obvious since there is where we have a heat flux $q_n = h = -1 \cdot 10^5\, W/m^2$, meaning this is where heat comes to the material. The lower temperatures are at end of the gripper (to the right) at boundary $\mathcal{L}_4$, we believe the reason for that is because it's far from $\mathcal{L}_h$ and surrounding $\mathcal{L}_4$ is $\mathcal{L}_c$ which leaks heat due to convection. Important to note though is that the temperature differences between the lowest and highest temperature is not that high. In the stationary heat transfer problem the difference was $1.35\,°\mathrm{C}$. A reason for that might be that the gripper is very small in size. In the transient heat transfer we retrieve plots on how the heat flows through the body and through time. The provided link of the animation shows that the temperatures converge to the temperatures of the stationary problem which reassures that it has been implemented correctly.

In figure 6 we find that the displacements have the gripping property that we try to achieve, observing the tipping motion. We also note that the boundaries where the displacements are zero, have no displacements in the plot which reassures us that we handled the boundaries correctly
In figure 7 we see that the copper areas have a lot higher effective von Mises stress. This is because while calculating σ in equation 21, or rather in the constitutive matrix $\boldsymbol{D}$, we use Young's modulus E which is significantly higher in copper than in nylon. We also see that we get high stress on the "gripper" which might be due to the assumption of plane strain, where the out-of-plane strain are set to zero, as if the gripper is constrained in the z direction between two rigid walls. According to [2] the yield strength of copper is around $62 - 69\,\mathrm{MPa}$ and tensile strength at around $172 - 220\,\mathrm{MPa}$. We can conclude that the computed stresses surpass these values, which indicates that we will either have permanent deformation and possibly fracture.

The number of triangular elements we used was 652. Using a higher number of elements will give a more precise answer, and a low number of elements might be a source of error. Therefore, in figure 8 we compute the stationary temperature distribution for 2608 elements to see if the solution differs. As

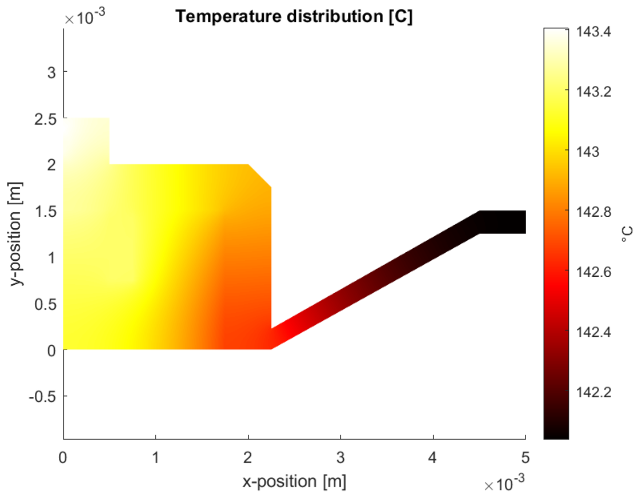


Figure 8: The solution to the stationary heat transfer problem with a finer mesh.

we clearly see comparing figures 4 and 8, there are no visible differences.

We do the same for the effective von Mises stresses in figure 9. We find when comparing figures 7 and

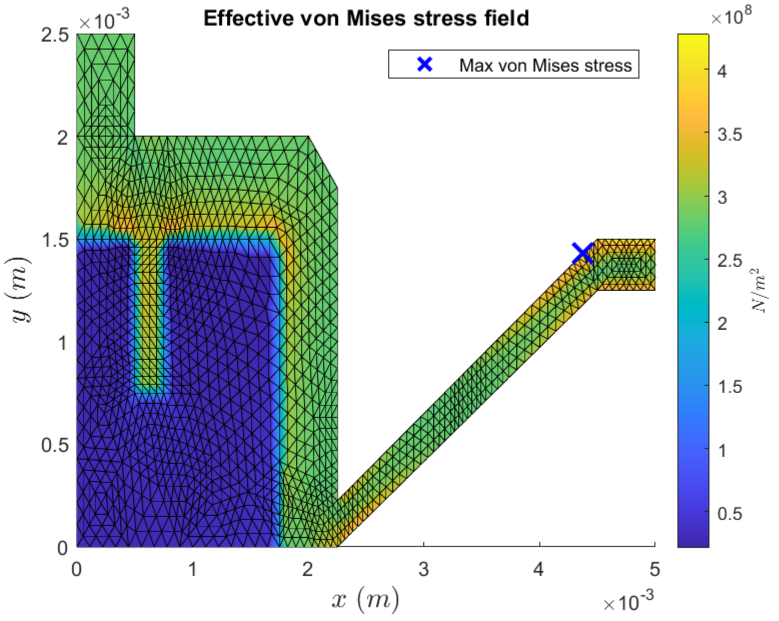


Figure 9: The effective von Mises stress when using a finer mesh.

9 that there are not large differences. We see that the position of highest stress has moved somewhat, but the general picture stays the same. We conclude that our original mesh was sufficient.

A Computer Code

A.1 Preprocessing

Here we do some prepossessing, declaring and creating convenient variables.

```
      %% Preprocessing
% material data (table 1)
ECopper = 128*10^9; ENylon = 3*10^9; % E [Pa]
vCopper = 0.36; vNylon = 0.39; % Poisson's ratio
alphaCopper = 17.6 * 10^(-6); alphaNylon = 80 * 10^(-6); % Expansion
    coefficient, alpha [1/K]
acopper = 40; %Convection
rhoCopper = 8930; rhoNylon = 1100; %Density [kg/m^3]
cpCopper = 386; cpNylon = 1500; %Specific heat [J/kg-K]
kCopper = 385; kNylon = 0.26; %Thermal conductivity [W/m-K]

%Subdomain 4 is Nylon, 1,2,3,4,5 are copper
rho_subdomain = [rhoCopper, rhoCopper, rhoCopper, rhoNylon, rhoCopper,
    rhoCopper];
cp_subdomain = [cpCopper, cpCopper, cpCopper, cpNylon, cpCopper,
    cpCopper];
k_subdomain = [kCopper, kCopper, kCopper, kNylon, kCopper, kCopper];
alpha_subdomain = [alphaCopper, alphaCopper, alphaCopper, alphaNylon,
    alphaCopper, alphaCopper];
v_subdomain = [vCopper, vCopper, vCopper, vNylon, vCopper, vCopper];
E_subdomain = [ECopper, ECopper, ECopper, ENylon, ECopper, ECopper];
T_inf = 18; % Environment temperature
h = -10^5;  % Heat flux
thickness = 100; % Should not matter

mesh = load('4nmesh.mat');
p = mesh.p;  % points/nodes (x:y for each column)
e = mesh.e;  % edges (rows 1,2: node # of el. seg., 5: edge label bd.
    seg.)
t = mesh.t;  % triangle info

coord = p';
enod=t(1:3,:)'; % nodes of elements
nelm=size(enod,1); % number of elements
nnod=size(coord,1); % number of nodes

dof=(1:nnod)'; % dof number is node number
dof_S=[(1:nnod)',(nnod+1:2*nnod)']; % give each dof a number

% Create topology/connectivity matrices
for ie=1:nelm
edof_S(ie,:)=[ie dof_S(enod(ie,1),:), dof_S(enod(ie,2),:),dof_S(enod(ie
    ,3),:)];
edof(ie,:)=[ie,enod(ie,:)];
end
```

```
[Ex, Ey] = coordxtr(edof, coord, dof, 3); %Extract nodal coordinate
   data for each element
                                           %from the global coordinate
                                              matrix.

er = e([1 2 5],:); % Reduced e
conv_segments = [1 2 3 5 13 15 20 25 ];     % Edges from pdetool with
   convection
flux_segments = [18];                       % Edges from pdetool with
   heat flux
xdisplacement_segments = [4 16 17 18 ];     % Edges from pdetool with
   known x displacements
ydisplacement_segments = [21];              % Edges from pdetool with
   known x displacements

%Where the actual edges (node pairs) will be stored
edges_conv = [];
edges_flux = [];
edges_xdisplacement = [];
edges_ydisplacement = [];

% Translating segments to edges
for i = 1:size(er,2)
    if ismember(er(3,i),conv_segments)
        edges_conv = [edges_conv er(1:2,i)];
    end

    if ismember(er(3,i), flux_segments)
        edges_flux = [edges_flux er(1:2,i)];
    end

    if ismember(er(3,i), xdisplacement_segments)
        edges_xdisplacement = [edges_xdisplacement er(1:2,i)];
    end

    if ismember(er(3,i), ydisplacement_segments)
        edges_ydisplacement = [edges_ydisplacement er(1:2,i)];
    end
end
```

A.2 The Stationary Heat Transfer Problem.

Below we solve the stationary heat transfer problem.

```
    %% Task a) Heat convection
D = eye (2);         % Constitutive matrix
K = zeros(nnod);     % Stiffnes matrix
f = zeros(nnod, 1); % Force vector

for elementnbr=1:nelm                     % Lopping through the elements
    subdomain = t(4, elementnbr);         % The current subdomain
    current_k = k_subdomain(subdomain); % Conductivity for the current
       subdomain
```

```
current_D = current_k*D;            % D matrix for current
    subdomain (copper/nylon)
element_x = Ex(elementnbr,:);       % x-coordinates of nodes for
    the current element
element_y = Ey(elementnbr,:);       % y-coordinates of nodes for
    the current element
[Ke, fe] = flw2te(element_x, element_y, thickness, current_D, 0);%
    solve (no convection/flux)

%fe will actually be a 3x1 vector with only zeros since Q=0
for i=1:length(edges_conv(1,:)) % Looping though edges with
    convection
     % If we find a convection edge in current element
     if ismember([edges_conv(1, i), edges_conv(2, i)], edof(
        elementnbr,2:4))

         % coordinates for the nodes of the edge
         x1 = coord(edges_conv(1, i), 1);
         y1 = coord(edges_conv(1, i), 2);
         x2 = coord(edges_conv(2, i), 1);
         y2 = coord(edges_conv(2, i), 2);

         L = sqrt((x1-x2)^2+(y1-y2)^2); %Length of edge

         Kc = thickness*acopper*(L/6)*[0 0 0; 0 2 1; 0 1 2]; % See
            report!
         fc = thickness*acopper*T_inf*(L/2)*[0 1 1]'; % See report!
         Ke = Ke+Kc;  % Modify element stiffnes with convection
         fe = fe + fc;% Modify element force vector with convection
     end
end

% And we do similar as above and modify for heat flux in force
    vector.
for i=1:length(edges_flux(1,:))
     if ismember([edges_flux(1, i), edges_flux(2, i)], edof(
        elementnbr,2:4))
         x1 = coord(edges_flux(1, i), 1);
         y1 = coord(edges_flux(1, i), 2);
         x2 = coord(edges_flux(2, i), 1);
         y2 = coord(edges_flux(2, i), 2);

         L = sqrt((x1-x2)^2+(y1-y2)^2);
         fe = fe - thickness*h*(L/2)*[0 1 1]'; % See report!
     end
end

% Assembling
indx = edof(elementnbr,2:end);
K(indx,indx) = K(indx,indx)+Ke;

indx = edof(elementnbr,2:end);
```

```
    f(indx) = f(indx) + fe;

end
%K = (K+K')./2; We could ensure symmetric matrix like this if we want(
    removing rounding errors)
T = solveq(K, f); % Solve the stationary problem with convection and
    flux

% Plotting
eT=extract(edof,T); % Extract element tempratures
figure()
patch(Ex',Ey',eT')
figure()
patch(Ex',Ey',eT','EdgeColor','none')
title('Temperature distribution [C]')
colormap(hot);
cb = colorbar;
ylabel(cb,' C ','FontSize', 14);
xlabel('x-position [m]')
ylabel('y-position [m]')
axis equal
```

A.3 The Transient Heat Transfer Problem

Below we solve the transient heat transfer problem.

```
    %% Task b) Transient Heat Convection

C = zeros(nnod); %C-matrix for transient, see report!
T_max = 0.9*max(T); % 90% of the max temp in the stationary solution
for elementnbr=1:nelm   %looping through elements
    subdomain = t(4, elementnbr);             % The current subdomain
    current_rho = rho_subdomain(subdomain); % Conductivity for the
       current subdomain
    current_cp =  cp_subdomain(subdomain);  % D matrix for current
       subdomain (copper/nylon)
    element_x = Ex(elementnbr,:);             % x-coordinates of nodes
       for the current element
    element_y = Ey(elementnbr,:);             % y-coordinates of nodes
       for the current element

    Ce = plantml(element_x, element_y, current_rho*current_cp*thickness
       ); % element C-matrix

    % Assemble
    indx = edof(elementnbr,2:end);
    C(indx,indx) = C(indx,indx)+Ce;
end

time_step=0.05;
end_time = 100; % How long we want to run the simulation
nbr_steps = end_time/time_step +1;
```

```
T_0 = 18; %initial temperature
T_transient=zeros(nnod, nbr_steps); % Each column stores nodal
    temperatures at time point
T_transient(:,1) = T_0.*ones(nnod,1);

for i=2:end_time/time_step+1 % Step through time
    T_previous = T_transient(:,i-1); % Previous nodal temperatures
    T_transient(:,i) = (C+time_step.*K)\(time_step.*f+C*T_previous); %
        See report!
end

T_transient_max = max(T_transient); % Max temprature at each time point

% Calculate at what time we reach 90% of max stationary temperature
time90 = 0; % Here we will store the time it took
index90 = 0;
for i=1: nbr_steps
    if(T_transient_max(i)>= T_max) % We have found the amount of
        time_steps
         time90 = i*time_step;
         index90 = i;
         break
    end
end

%% Plot Snapshots for transient
index3 = 0.03*index90; % index for 3 % of index90

% Evenly spaced 5 indices for Transient tempratures
snapshots = [1 round((1+index3/2)/2) round(index3/2) round((index3+
    index3/2)/2) floor(index3)];

for i=1:5
    time = (snapshots(i)-1)*time_step;
    eT=extract(edof,T_transient(:, snapshots(i))); % Extract element
        tempratures
    figure()
    patch(Ex',Ey',eT','EdgeColor','none')
    title(['Temperature distribution [C], t= ', num2str(time), '
        seconds'])
    colormap(hot);
    cb = colorbar;
    caxis([15, 28])
    ylabel(cb,' C ','FontSize', 14);
    xlabel('x-position [m]')
    ylabel('y-position [m]')
    axis equal
end
```

A.4 The Thermoelasticity Problem

Below we find the displacements and von Mises stresses due to thermal expansion

```
    %% Task c) ThermoElasticity
ndof = 2*nnod; % two displacements per node
deltaT = T-T_0; % T and T_0 are from the stationary problem;
K = zeros(ndof,ndof); % Stiffness matrix
f = zeros(ndof,1);    % Force vector

for elementnbr=1:nelm % Loop through elements
    subdomain = t(4, elementnbr);       % Current subdomain
    current_E = E_subdomain(subdomain); % E-module for the current
       subdomain
    current_v = v_subdomain(subdomain); % Poissions ratio for current
       subdomain
    current_alpha = alpha_subdomain(subdomain); % Expansion Coefficient
        for current subdomain

    D_constant = current_E/((1+current_v)*(1-2*current_v));
    current_D = D_constant.*[1-current_v, current_v, 0;
                             current_v, 1-current_v, 0;
                             0, 0, 0.5*(1-2*current_v)]; % D matrix for
                                 current subdomain (copper/nylon)

    mean_deltaT = deltaT(t(1,elementnbr)) + deltaT(t(2,elementnbr)) +
       deltaT(t(3,elementnbr));
    mean_deltaT = mean_deltaT./3; %mean of the nodal temprature changes
    D_epsilon_deltaT = current_alpha.*mean_deltaT.*current_D*[1;1;0]; %
        Stress from thermal strain

    element_x = Ex(elementnbr,:); % x-coordinates of nodes for the
       current element
    element_y = Ey(elementnbr,:); % y-coordinates of nodes for the
       current element

    Ke = plante(element_x, element_y, [2 thickness], current_D); %
       Element stiffness
    f_deltaT = plantf(element_x, element_y, [2, thickness],
       D_epsilon_deltaT'); % See report!

    % Assemble
    indx = edof_S(elementnbr,2:end);
    K(indx,indx) = K(indx,indx)+Ke;

    indx = edof_S(elementnbr,2:end);
    f(indx) = f(indx) + f_deltaT;
end

% The nodes where the displacements are known(=0)
bc= [edges_xdisplacement(1,:)'; % x-displacements dof have same index
    as node
      edges_xdisplacement(2,:)';
      nnod+edges_ydisplacement(1,:)'; %Note that we have to add nnod,
        since the dof of
```

```
     nnod+edges_ydisplacement(2,:)'  %y-displacement for node i is
        defined to be i+nnod
     ];

 bc = [unique(bc), zeros(length(unique(bc)),1)]; % Boundary conditions,
     all knownn
                                                  % displacements are 0.

[u,Q] = solveq(K,f,bc); %Solve and find diplacements

ed = extract(edof_S,u); % Extract element displacements

% Calculate displaced coordinates
mag = 10; % Magnification (due to small deformations)
exd = Ex + mag*ed(:,1:2:end);
eyd = Ey + mag*ed(:,2:2:end);
% Plot
figure()
patch(Ex',Ey',[0 0 0],'EdgeColor','none','FaceAlpha',0.3)
hold on
patch(exd',eyd',[0 0 0],'FaceAlpha',0.3)
axis equal
title('Displacement field [Magnitude enhancement 10]')
xlabel('$x \;(m)$', 'Interpreter','latex','FontSize', 14)
ylabel('$y \;(m)$', 'Interpreter','latex','FontSize', 14)

% VON MISES STRESS
von_mises = zeros(1, nelm); % Here we will store element von mises
   stresses

for elementnbr=1:nelm  % Loop through elements
    subdomain = t(4, elementnbr);       % Current subdomain
    current_E = E_subdomain(subdomain); % Conductivity for the current
       subdomain
    current_v = v_subdomain(subdomain); % Poissions ratio for current
       subdomain
    current_alpha = alpha_subdomain(subdomain); % Expansion Coefficient
        for current subdomain

    D_constant = current_E/((1+current_v)*(1-2*current_v));
    current_D = D_constant.*[1-current_v, current_v, 0;
                             current_v, 1-current_v, 0;
                             0, 0, 0.5*(1-2*current_v)]; % D matrix for
                                current subdomain (copper/nylon)

    mean_deltaT = deltaT(t(1,elementnbr)) + deltaT(t(2,elementnbr)) +
       deltaT(t(3,elementnbr));
    mean_deltaT = mean_deltaT./3;  %mean of the nodal temprature
       changes

    element_x = Ex(elementnbr,:); % x-coordinates of nodes for the
       current element
    element_y = Ey(elementnbr,:); % y-coordinates of nodes for the
       current element
    element_displacement = ed(elementnbr,:); % The element
       displacements we just calculated

    % Stresses from the TOTAL strain
    [es, et] = plants(element_x, element_y, [2,thickness], current_D,
       element_displacement);

    % Stress from thermal strain
    D_epsilon_deltaT = current_alpha.*mean_deltaT.*current_D*[1;1;0];

    es = es - D_epsilon_deltaT';  % correction for thermal stress

    sigma_xx = es(1);
    sigma_yy = es(2);
    tau_xy = es(3);

    % Calculate sigma_zz seperately
    sigma_zz = current_v*(sigma_xx + sigma_yy) - current_alpha*
       mean_deltaT*current_E;

    % Calculate von mises stress
    von_mises(elementnbr) = sqrt(sigma_xx^2 + sigma_yy^2 + sigma_zz^2 -
        sigma_xx*sigma_yy - sigma_xx*sigma_zz - sigma_yy*sigma_zz + 3*
       tau_xy^2);
end


von_mises_nodes = zeros(nnod, 1); % Here we will store the nodal von
   mises stresses

for i=1:size(coord,1)
[c0,c1]=find(edof(:,2:4)==i); % Find which elements connect to the node
von_mises_nodes(i,1)=sum(von_mises(c0))/size(c0,1); % Take the average
   von mises stresses
                                                     % of connected
                                                        elements.
end

% Plot stress field
stress = extract(edof, von_mises_nodes);
figure()
patch(Ex',Ey',stress')
cb = colorbar;
ylabel(cb,'$N/m^2$', 'Interpreter','latex','FontSize', 14);
xlabel('$x \;(m)$', 'Interpreter','latex','FontSize', 14)
ylabel('$y \;(m)$', 'Interpreter','latex','FontSize', 14)
title('Effective von Mises stress field')
hold on
% Find max stress and plot with cross
node_index = find(von_mises_nodes == max(von_mises_nodes));
```

```
pl = plot(coord(node_index,1), coord(node_index,2), 'x','MarkerSize',
   14, 'LineWidth', 2, 'color', 'b');
legend(pl,'Max von Mises stress')
```

References

[1] P-E Austrell et al. *CALFEM a finite element toolbox version 3.4*. URL: `https://www.solid.lth.se/fileadmin/hallfasthetslara/utbildning/kurser/FHL064_FEM/calfem34.pdf` (visited on 05/24/2023).

[2] eFunda. *Typical Physical/Mechanical Properties*. URL: `https://en.wikipedia.org/wiki/Syllable` (visited on 05/24/2023).

[3] Niels Ottosen and Glen Peters. *Introduction Finite Element Method*. en. Philadelphia, PA: Prentice Hall, June 1992.