Perseverance Under Stress

- A finite element analysis of optical instruments

Project in Introduction to the finite element method

Simon Danielsson, Nils Romanus Myrberg

FHLF01
Lunds Tekniska Högskola, Lunds Universitet
Maj 2021

1 Introduction

1.1 Purpose

The Nasa rover *Perseverance* is currently on a mission in search for signs of ancient life on Mars. To enable this search the rover has been equiped with 23 cameras. The Martian climate is harsh, with large swings in temperature, and it is of utmost importance that these cameras can withstand the unforgiving climate. The project aims to analyze the thermal and mechanical properties of Perserverance's optical instruments. This is done by using a finite element(FE) analysis to solve the problems presented in section 1.2

1.2 Problem Description

The project aims to answer a specific set of questions by solving the problems presented in section 1.2.2. These problems are related to heat flow or thermoelastic solid mechanics. An overview of the model used and the conditions for which the problems are solved is presented in section 1.2.1.

1.2.1 Description

A simplified model of the optical instrument of Perserverance consists a two dimensional cross section of the camera that is constituted of glass lenses enclosed in a titanium alloy shell. An image depicting the cross section of the camera is presented in figure 1.

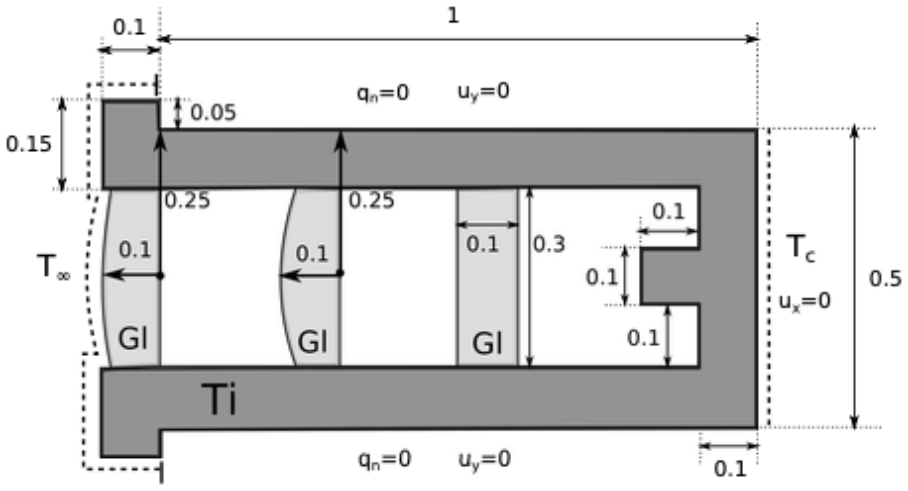


Figure 1: Cross section of Perserverance's camera. All dimensions are in [cm]

Figure 1 shows the domain on which the problems in section 1.2.2 are solved. The sketch of the cross section also displays the characteristics of the boundaries

of the domain. When heat flow is examined the right- and leftmost boundary are subject to Newton convection whilst the top, bottom and inside of the camera have insulated walls. For solid mechanics applications top and bottom boundaries are constrained in the vertical direction and the leftmost boundary in the horizontal direction. Boundaries with no displacement constraints are traction free. Also the model prescribes no external heat or mechanical load on the body. In figure 1 Gl and Ti denotes the respective material of each subdomain. The parameters for these materials are listed in table 1.

Constant	Titanium Alloy	Glass
Young's modulus E [GPa]	110	67
Poisson's ratio ν [-]	0.34	0.2
Expansion coefficient α [1/K]	$9.4 \cdot 10^{-6}$	$7 \cdot 10^{-6}$
Density ρ [kg/m^3]	4620	3860
Specific heat c_p [J/(kg K)]	523	670
Thermal conductivity, k [W/(mK)]	17	0.8
Thickness t [cm]	1	1
Convection constant α_c [W/(mK)]	100	100

Table 1: Material parameters for titanium and glass

Figure 1 also contains prescribed temperatures on the exterior side of the domain. T_∞ and T_c denotes the temperature exterior to the right- and leftmost boundaries respectively. Furthermore, T_0 denotes the temperature for which the structure is stress free. Values for these temperatures, under day and night conditions, are found in table 2

Quantity	Value during day [$^\circ C$]	Value during night [$^\circ C$]
T_∞	40	-96
T_c	20	20
T_0	20	20

Table 2: Prescribed temperatures for day- and night conditions.

With paramaters and conditions specified, the analysis of the thermal and mechanical properties can begin. The analysis is based on the problems presented in section 1.2.2.

1.2.2 Problems

The report aims to answer and discuss the following set of problems. All problems are solved for both day and night.

1. Stationary heat flow
    - Determine the stationary temperature distribution within the body

- Compute the maximum temperature

2. Transient heat flow
   - Determine the transient temperature evolution
3. Mechanical problem
   - Determine the effective von Mises stress field.
   - Determine which subdomains are subject to the most stress.
4. Displacement field with stationary temperature distribution
   - Determine the deformation pattern in the leftmost lens
   - Quantify the total displacement in the leftmost lens

2 Procedure

The problems presented in section 1.2.2 are solved using a finite element analysis. The set-up of the finite element formulation is similar for both problem types. The procedure is based on the following routine.

- Establish the strong formulation of the problem.
- Derive the weak form of the problem.
- Choose a suitable approximation
- Choose the weight function

The procedure for establishing the FE formulation the stationary heat flow problem is explained in section 2.1. Section 2.2 explains the procedure of solving the transient heat flow problem. Finally, section 2.3 derives the FE formulation of the solid mechacis problem.

2.1 FE formulation of 2-D stationary heat flow

2.1.1 Strong form

By using the fundemental balance equations of classical physics, and deploying the constitutive relation of Fourier's law, the strong form of a stationary heat flow problem can be derived. Let T denote the temperature in a 2-D body Ω, Q the amount of heat supplied to the body per unit area. Using this notation the strong form of the 2-D heat flow problem is presented in equation 1

$$\nabla \cdot (t\mathbf{D}\nabla T) + tQ = 0 \quad (x, y) \in \Omega \tag{1}$$

$$q_n = \mathbf{q}^{\mathbf{T}}\mathbf{n} = h \quad (x, y) \in \mathcal{L}_h$$

$$T = g \quad (x, y) \in \mathcal{L}_g$$

where (x, y) denotes the spatial position, t the thickness of the body, $\mathbf{D}$ the constitutive matrix from Fourier's law and q_n the normal heat flux on the sub-boundary $\mathcal{L}_h$. The last two rows of equation 1 establish the boundary conditions for the problem where h is a the prescribed heat flux value on $\mathcal{L}_h$ and g the prescribed temperature on $\mathcal{L}_g$. Note that the union of the non-intersecting boundaries $\mathcal{L}_h$ and $\mathcal{L}_g$ form the closed boundary $\partial\Omega$ of Ω.

2.1.2 Weak form

The weak form of the problem is derived by multiplying equation 1 with an arbitrary weight function v and integrating over Ω. This operation coupled with the use of the Green-Gauss theorem yields the weak form of the stationary heat flow problem found in equation 2.

$$\int_{\Omega} (\nabla v)^T \mathbf{D} \nabla T t dA = -\int_{\mathcal{L}_h} v h t d\mathcal{L} - \int_{\mathcal{L}_g} v q_n t d\mathcal{L} + \int_{\Omega} v Q t dS \tag{2}$$

Note that the boundary conditions specified in equation 1 still hold for equation 2.

2.1.3 Choice of approximation

To solve the problem a two dimensional mesh consisting of linear triangular elements is introduced. In practice this is done by using Matlabs *pdetoolbox*. A graphical representation of the mesh is found in figure 2. To keep track of the topology of the mesh, we define the matrix **Edof** which contains the global enumeration of elements and its associated degrees of freedom. Using linear element shape functions for both spatial dimensions, completeness is obviously satisfied since such a function can represent an arbitrary constant gradient and value. It is not difficult to see that along every linear curve between two nodal points in such an element, the element shape function will only have two paramaters. Therefore the value of the element shape function on the element boundary is well defined. Hence, the compatibility requirement is also satisfied. The fulfillment of both criterias guarantees convergance and a continuous solution can therefore be acquired on the domain. The choice of approximation, coupled with its gradient, are presented in equation 3.

$$T = \mathbf{N}\mathbf{a} \tag{3}$$

$$\nabla T = \nabla \mathbf{N}\mathbf{a} = \mathbf{B}\mathbf{a},$$

where $\mathbf{N}$ and $\mathbf{B}$ denotes the global element shape functions and their respective gradients. The nodal displacement vector $\mathbf{a}$ contains the temperature at the nodal points. Since the camera is symmetric the domain can be split in to two equivalent subdomains. The problem can then be solved on one of these subdomains and mirrored to obtain a solution on all of the camera. Using

the symmetry condition is obviously advantageous since it decreases computational load. Thus we choose Ω to be the top half of the camera. This domain is illustrated in figure 2. Note that homogeneous Neumann conditions must hold on the boundary where the split has been performed, this follows from the symmetry property.

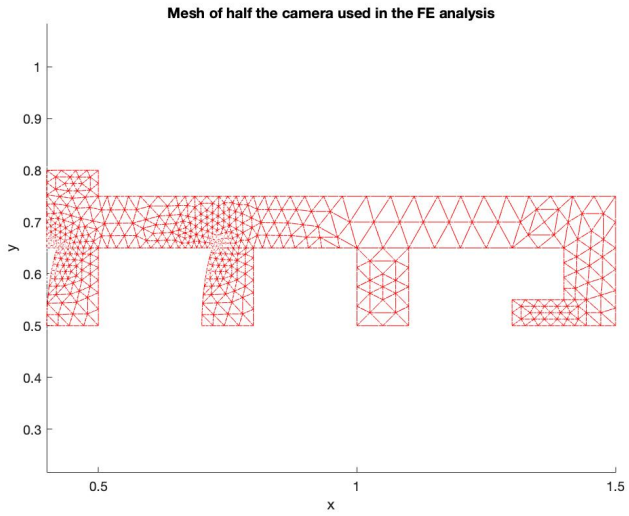


Figure 2: Mesh of linear triangular elements

2.1.4 Choice of weight function

Using Galerkin's method v in equation 2 is chosen as a linear combination of the element shape functions with arbitrary coefficients $\mathbf{c}$, i.e $v = \mathbf{Nc}$. Since v is a scalar function it is obviously symmetric, i.e $v = v^T = (\mathbf{Nc})^T = \mathbf{c}^T\mathbf{N}^T$. This choice of v can be deployed in equation 2 yielding equation 4.

$$\mathbf{c}^T\left[\int_\Omega \mathbf{B}^T\mathbf{DB}dS\mathbf{a} + \int_{\mathcal{L}_h} \mathbf{N}^T h t d\mathcal{L} + \int_{\mathcal{L}_g} \mathbf{N}^T q_n t d\mathcal{L} - \int_\Omega \mathbf{N}^T Q t dS\right] = 0 \quad (4)$$

Since the choice of $\mathbf{c}^T$ is arbitrary the expression in the brackets in equation 4 must be equal to the zero-vector since it is the only one that is orthogonal to all vectors. This fact can be reformulated as equation 6 using the notation supplied by equation 5.

$$\mathbf{K} = \int_\Omega \mathbf{B}^T\mathbf{DB}dS \quad (5)$$

$$\mathbf{f}_b = -\int_{\mathcal{L}_h} \mathbf{N}^T h t d\mathcal{L} - \int_{\mathcal{L}_g} \mathbf{N}^T q_n t d\mathcal{L}$$

$$\mathbf{f}_l = \int_\Omega \mathbf{N}^T Q t dS$$

Where $\mathbf{K}$ is called the stiffnes matrix, $\mathbf{f}_b$ the boundary load vector and $\mathbf{f}_l$ the load vector. Using this notation the FE formulation of a 2-D heat flow problem can be written as equation 6.

$$\mathbf{Ka} = \mathbf{f}_b + \mathbf{f}_l \tag{6}$$

As specified in the problem formulation in section 1.2 no amount of external heat is supplied to the body, i.e $Q = 0$. The load vector $\mathbf{f}_l$, computed by the integral in expression 5, is therefore zero. The FE formulation is therefore reduced to expression 7.

$$\mathbf{Ka} = \mathbf{f}_b \tag{7}$$

2.1.5 Boundary Conditions

An integral part of solving differential equations using finite element analysis is dealing with the boundary conditions of the problem. As specified in section 1.2.1 and shown in figure 1 the problem has homogeneous Neuman conditions on all boundaries except for on the right- and leftmost parts of the body. The part of the integral that is computed on these parts of the boundary is therefore zero since the heat flux q_n is zero as prescribed by the boundary condition. The right- and leftmost boundary however has a Robin boundary condition. This stems from Newton convection, defined in expression 8, being present on these boundaries.

$$q_n = \alpha(T - T_\infty) \tag{8}$$

Where T_∞ denotes the temperature on the exterior side of the boundary, i.e the "room temperatue". The value of this temperature is found in table 2. The Robin condition and the homogeneous Neuman condition yield expression 9 for computing the boundry load vector.

$$\mathbf{f}_b = -\int_{\mathcal{L}_h} \mathbf{N}^T \alpha(T - T\infty) t d\mathcal{L} \iff \tag{9}$$

$$\mathbf{f}_b = -\int_{\mathcal{L}_h} \mathbf{N}^T \alpha \mathbf{N} \mathbf{a} t d\mathcal{L} + \int_{\mathcal{L}_h} \mathbf{N}^T \alpha T_\infty t d\mathcal{L}$$

This can be symbolically rewritten, using a matrix-vector notation, as the following.

$$\mathbf{f}_b = -\mathbf{Ma} + \mathbf{f}_0$$

Using expression 9 in the original FE formulation, defined in equation 7 one receives the modified expression

$$(\mathbf{K} + \mathbf{M})\mathbf{a} = \mathbf{f}_0 \iff \tag{10}$$

$$\tilde{\mathbf{K}}\mathbf{a} = \mathbf{f}_0$$

2.1.6 Solving the problem at hand

The differential equation has thus been reduced to a simple linear algebra problem which can be solved using Gaussian elimination. The nodal temperatures are computed using this procedure and can then be plotted yielding the results in section 3.1. The following sections explains how the matrices in the linear algebra problem was determined.

2.1.7 Determining the stiffness matrix

The stiffnes matrix $\mathbf{K}$ is computed using expression 5 element-wise. The element stiffnes matrix $\mathbf{K}^e$ is computed for all the elements in the mesh using the CALFEM command `flw2te`. The integral contains the matrix $\mathbf{D}$ that captures the constitutive relation from Fourier's law. $\mathbf{D}$ is dependant on the material parameters for each subdomain, i.e if the element is in the glass subdomain or the titanium subdomain. Therefore an element constitutive matrix $\mathbf{D}^e$ is defined for each element as $\mathbf{D}^e = k^e\mathbf{I}$ due to isotropy. Here, k^e denotes the thermal conductivity for the element and $\mathbf{I}$ the identity matrix. The values of k^e for the different materials are found in table 1. The global stiffnes matrix was subsequently assembled using the topology matrix **Edof** that was extracted from `pdetoolbox`.

2.1.8 Determing the boundary load vector

The boundary load vector is defined in equation 5. As explained in section 2.1.5 values of q_n are prescribed on all of $\partial\Omega$, i.e $\mathcal{L}_g = \emptyset$. Therefore the integral computed on $\mathcal{L}_g$ vanishes and we are left with the integral on $\mathcal{L}_h$ where q_n is prescribed. The boundary load vector $\mathbf{f}_b$, was computed by firstly determining which nodes lie on the convection boundary. Since the global shape functions corresponding to these nodes are the only ones that are not identically zero on the convection boundary, these are the only ones contributing to the boundary load vector. Thus we can construct the element matrices $\mathbf{M}^e$ and element vectors $\mathbf{f}_0^e$, from equation (9), by iterating over the elements containing associated nodes and computing the corresponding integrals for the elements. The value of the integrals are easy to compute given the linearity of the element shape functions. Computing the integral essentially boils down to solving a purely geometrical problem using the lengths of the element boundary which has a non-empty intersection with the convection boundary. Since the nodes contributing to $\mathbf{M}^e$ and $\mathbf{f}_0^e$ can be deduced from the mesh, these are easily assembled into $\mathbf{M}$ and $\mathbf{f}_0$.

2.2 FE formulation of 2-D transient heat flow

The problem of heat flow is now expanded to contain a time dimension. The FE formulation with transiet heat flow follows the same routine as the one presented in the introduction of section 2.

2.2.1 Strong form

The strong form of a transient heat flow problem is derived by using the same fundemental physical principle of balance, coupled with the constitutive relation of Fourier's law. However, this type of problem allows for a net flux of heat to or from the body over time. Thus the original strong form of the problem in equation 1 is modified to contain time dependance as shown in equation 11

$$\rho c t\dot{T} + \nabla \cdot (t\mathbf{D}\nabla T) + tQ = 0 \quad (x, y) \in \Omega \tag{11}$$

$$q_n = \mathbf{q}^{\mathbf{T}}\mathbf{n} = h \quad (x, y) \in \mathcal{L}_h$$

$$T = g \quad (x, y) \in \mathcal{L}_g$$

$$T(0) = T_0$$

Where ρ denotes the density- and c the specific heat capacity of the materials the body Ω is constituted of. The values of these material parameters are found in table 1. Note that the original boundary conditions now has been supplemented by an initial condition in the last row.

2.2.2 Final formulation

By starting from the modified strong form described in equation 11 and following the procedure described in detail in section 2.1 one arrives at the FE formulation for 2-D transient heat flow. The formulation bears a striking resemblance of expression 6 modified with the matrix $\mathbf{C}$ defined in equation 12.

$$\mathbf{C} = \int_\Omega \mathbf{N}^T \rho c \mathbf{N} t dA \tag{12}$$

With $\mathbf{K}$, $\mathbf{f}_l$ and $\mathbf{f}_b$ defined in expression 5 and $\mathbf{C}$ defined in expression 12, one arrives at the FE formulation of 2-D transient heat flow in expression 13

$$\mathbf{C}\dot{\mathbf{a}} + \mathbf{K}\mathbf{a} = \mathbf{f}_b + \mathbf{f}_l \tag{13}$$

As noted in section 2.1.4 no external load is present which results in the load vector $\mathbf{f}_l$ being zero. The boundary conditions for the transient heat flow problems are identical to the stationary ones which results in a modified stiffness matrix from the convection part of the boundary load vector as explained in section 2.1.5. Because of the boundary conditions, expression 13 therefore takes the form of equation 14.

$$\mathbf{C}\dot{\mathbf{a}} + \tilde{\mathbf{K}}\mathbf{a} = \mathbf{f}_0 \tag{14}$$

2.2.3 Time stepping

Using a finite element approach the partial differential equation 11 is reduced to a system of first order ordinary differential equations which can be solved using a simple time stepping method. In order to not have to bother with choosing a suitable time stepping size Δt to maintain stability, an implicit time steppig

method is used. The A-stable backwards Euler method is defined in equation 15.

$$y_{k+1} = y_k + \Delta t f(t_{k+1}, y_{k+1}) \tag{15}$$

Deploying this method on the FE formulation of the transient heat flow problem in 14 yields the following recursive scheme for computing a.

$$(\mathbf{C} + \Delta t \mathbf{K})\mathbf{a}_{k+1} = \mathbf{C}\mathbf{a}_k + \Delta t \mathbf{f}_0 \tag{16}$$

Once again we arrive at a linear algebra problem that is easily solved using Gaussian elimination. The nodal temperatures $\mathbf{a}$ for each time is computed using this procedure and can then be plotted yielding the results in section 3.2.

2.2.4 Computing the matrix C

The computation required for determining the matrix $\mathbf{C}$ in equation 16 is defined by equation 12. This expression can be evaluated for all elements using the element shape function, density, and heat capacity for each respective element. By iterating over the elements and using the CALFEM function *plantml* the element matrix $\mathbf{C}^e$ can be computed. These element matrices $\mathbf{C}^{\ e}$ are subsequently assembled using the topology matrix **Edof**.

2.3 FE formulation of mechanical problem, assuming plane strain

To establish a finite element formulation of the mechanical problem, we follow the procedure presented in the introduction of section 2, just as for the heat flow problem. We then, we introduce a useful quantity, the von Mises stress, for visualising stresses within a body. To conclude, we incorporate the specific boundary conditions of our problem, and describe how one can actually calculate the integrals present in the weak formulation of our particular problem.

2.3.1 Strong and weak form

As with the heat flow problem, the strong form of the mechanical problem is based on a fundamental balance equation. It is essentially stating that, pointwise, the sum of forces must equal zero. More formally it says that

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

where $\boldsymbol{\sigma} = [\sigma_{xx}\ \sigma_{yy}\ \sigma_{zz}\ \sigma_{xy}\ \sigma_{xz}\ \sigma_{yz}]^{\mathrm{T}}$ is the stress vector, $\tilde{\nabla}$ the 3×6 matrix differential operator, and $\mathbf{b}$ is a given body force. Of course, for the solution of (17) to be unique, the equation has to be supplemented by suitable boundary conditions on $\partial\Omega$. In this case, the boundary conditions will consist of a prescribed traction vector $\boldsymbol{t}$ on the boundary of the body. First when we introduce our constitutive relation, we can get essential boundary conditions.

Comparing to equation (1), a great difference to equation (17) is that the dependent quantity $\boldsymbol{\sigma}$ is vector valued. This fact will modify the usual procedure

of obtaining the full FE formulation slightly, in comparison to the case with scalar quantities.

By multiplying each of the three equations in the system of equations (17) by arbitrary weight functions, and integrating over the domain, one obtains three weak form equations. If these weight functions are then collected in a vector $\mathbf{v}$, one can write these three equations on vector form as

$$\int_\Omega (\tilde{\nabla}\boldsymbol{v})^{\mathrm{T}}\boldsymbol{\sigma} t dA = \int_{\partial\Omega} \boldsymbol{v}^{\mathrm{T}}\boldsymbol{t} d\mathcal{L} + \int_\Omega \boldsymbol{v}^{\mathrm{T}}\boldsymbol{b} t dA \tag{18}$$

We somehow want to relate the stress to the displacement vector $\boldsymbol{u} = [u_{1x}\ u_{1y}\ u_{2x}\ u_{2x}/...\ u_{nx}\ u_{ny}]^{\mathrm{T}}$. This requires us to introduce material properties through a constitutive relation.

2.3.2 Constitutive relation

In our specific problem, elasticity is assumed to hold. This means that the stress is linearly dependent on the strain $\boldsymbol{\epsilon}$. If we further assume thermoelasticity, this elasticity property is modified such that the stress vanishes at some temperature $T = T_0$ (which in an isotropic material yields a corresponding initial strain $\boldsymbol{\epsilon_0} = \alpha(T - T_0)[1\ 1\ 1\ 0\ 0\ 0]^{\mathrm{T}}$ for some constant α), i.e.

$$\boldsymbol{\sigma} = \boldsymbol{D\epsilon} - \boldsymbol{D\epsilon_0}, \tag{19}$$

where $\boldsymbol{D}$ is the constitutive matrix. $\boldsymbol{D}$ is dependent on the Young's modulus E and Poisson's ratio ν of the medium, and if isotropy is assumed, $\boldsymbol{D}$ is symmetric. One can show that the strain is connected to the displacement through the kinematic relation $\boldsymbol{\epsilon} = \tilde{\nabla}\boldsymbol{u}$. Substituting the thermoelastic description of the stress in (19), and using the kinematic relation, the weak formulation in (18) is now transformed into a description of the displacement of the body.

2.3.3 Choice of approximation and weight function

In order to reach the compact FE formulation we are familiar with, one must introduce a suitable approximation of $\boldsymbol{u}$ and a weight function. As for heat flow, we choose $\boldsymbol{u} = \boldsymbol{Na}$, where $\boldsymbol{a}$ now is the vector containing the displacement of all degrees of freedom within the body (i.e. two per node). Taking arbitrary Galerkin weights $\boldsymbol{v} = \boldsymbol{Nc}$, and using $\boldsymbol{B} \equiv \tilde{\nabla}\boldsymbol{N}$, it is not difficult to end up with the following FE formulation

$$\left(\int_\Omega \boldsymbol{B}^{\mathrm{T}}\boldsymbol{DB} t dA\right)\boldsymbol{a} = \int_{\mathcal{L}_h} \boldsymbol{N}^{\mathrm{T}}\boldsymbol{h} d\mathcal{L} + \int_{\mathcal{L}_g} \boldsymbol{N}^{\mathrm{T}}\boldsymbol{t} d\mathcal{L} + \int_\Omega \boldsymbol{N}^{\mathrm{T}}\boldsymbol{b} t dA + \int_\Omega \boldsymbol{B}^{\mathrm{T}}\boldsymbol{D\epsilon_0} t dA \tag{20}$$

where we have a prescribed value $\boldsymbol{h}$ of the traction vector along $\mathcal{L}_h \subseteq \partial\Omega$, and prescribed displacement values $\boldsymbol{g}$ along $\mathcal{L}_g = \partial\Omega \setminus \mathcal{L}_h$. The essential boundary condition $\boldsymbol{g}$ is incorporated into the formulation by prescribing values in $\boldsymbol{a}$ at

the corresponding indeces (correponding to a dof). Finally, (20) can be written more compactly as

$$\boldsymbol{K}\boldsymbol{a} = \boldsymbol{f_b} + \boldsymbol{f_l} + \boldsymbol{f_0}, \tag{21}$$

a system of equations easily solveable by for instance Gaussian elimination.

2.3.4 Plane strain

The procedure of computing the integrals in (20) can be eased by assuming plain strain, i.e. all strains are confined in the xy-plane. Thus, we can reduce the dimensions of the vector quantity $\boldsymbol{D\epsilon_0}$ to 4×1. Note that this assumption does not imply vanishing σ_{zz} stress, as this quantity is dependent on the (generally non-zero) normal strains in the x and y directions.

2.3.5 Effective von Mises stress

In order to easily visualize the stresses in a body, it is benificial to transform the collection of components of $\boldsymbol{\sigma}$ into some representative scalar. One way to do this is by defining the *effective von Mises stress*

$$\sigma_{\text{eff}} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 - \sigma_{xx}\sigma_{yy} - \sigma_{xx}\sigma_{zz} - \sigma_{yy}\sigma_{zz} + 3\sigma_{xy}^2 + 3\sigma_{xz}^2 + 3\sigma_{yz}^2}. \tag{22}$$

2.3.6 Problem-specific remarks

The procedure of evaluating the integrals in (20) might need some clarification in this specific setting, before being able to invoke CALFEM's `solveq` method. The double integrals taken over the whole domain Ω are, as for the heat flow problem, computed element-wise and then assembled into the global matrices using the topology matrix. This is necessary in order to account for the different material properties through the usage of different constitutive matrices depending on if the element is in a glass or titanium part of the body. This also allows for less computational load in comparison to using the expanded element formulation, at least when using non-sparse matrices.

As $\boldsymbol{K^e}$ is essentially only dependent on the geometry of the element and the simple linear shape functions, its value is can be computed using for instance `plante` from the CALFEM library. The convection "load" vector $\boldsymbol{f_0^e}$ is calculated using a trick: if one interprets the quantity $\boldsymbol{D\epsilon_0}$ as a stress vector, one can use CALFEM's method `plantf` to compute the corresponding internal force vector $\boldsymbol{f_0^e}$. Further, as every sub-boundary without prescribed displacement values are traction free, i.e. $\boldsymbol{h} = \boldsymbol{0}$, the integral over $\mathcal{L}_h$ vanishes. The integral over $\mathcal{L}_g$ is performed over elements with dofs that has vanishing displacements. This means that as long as we set the components of $\boldsymbol{a}$ corresponding to these specific dofs to zero, we do not have to bother with computing the value of the integral over $\mathcal{L}_g$ for these elements. This is because we are not here interested

in these traction forces, and the method `solveq` for solving (21) works as long as we provide the essential boundary conditions in $\boldsymbol{a}$. We also note that for any dof not lying on $\mathcal{L}_g$, its associated global shape function is identically zero on $\mathcal{L}_g$. Thus we can set $\boldsymbol{f_b} = \mathbf{0}$. Neither do we have any distributed body force, why $\boldsymbol{f_b} = \mathbf{0}$ aswell. Thus, only $\boldsymbol{f_0}$ is non-vanishing.

By invoking `plants` element-wise, the stress in every element is computed using the calculated displacements. However, before calculating σ_{eff}, we have to subtract the initial strain as we in this context defined the strain to be thermoelastic. Before plotting σ_{eff} using nodal stress values, the mean stress of all circumjacent elements of a node is prescribed to that specific node.

2.3.7 Topology data

One aspect worth emphasizing in the context of the mechanical problem is the fact that the topological data in **Edof** has to be adjusted when the number of dofs are doubled (and not equal to the number of nodes). It is necessary to transform the heat flow topology matrix into a topology matrix which retains the structure of $\boldsymbol{u}$. This is not difficult, and code describing the procedure can be found in section 6.4.

2.4 Quantifying the displacement of the lens

To quantify the total displacement of the lens the square sum of the displacement in the lens is computed. This acts as a good measure for estimating the total amount of displacement in the body, thus enabling us to establish how close the lens remains to it's original design during temperature shifts. The scalar quantity $\mathbf{a}^T\mathbf{T}\mathbf{a}$, aptly named *the displacement magnitude squared*, is defined in equation 23.

$$\int_{\Omega_{lens}} \mathbf{u}^T\mathbf{u}t dA = \mathbf{a}^T \int_{\Omega_{lens}} \mathbf{N}^T\mathbf{N}t dA\mathbf{a} = \mathbf{a}^T\mathbf{T}\mathbf{a} \tag{23}$$

where Ω_{lens} denotes the subdomain of Ω that constitutes the leftmost lens. Since the quantity $\mathbf{u}^T\mathbf{u}t$ is integrated over a surface, the unit of the displacement magnitude squared becomes [cm^5]. The actual procedure to compute this measure starts with determining which elements lie in the subdomain Ω_{lens}. By iterating over these elements and using a modified version of the function `plantml`, the element matrix $\mathbf{T}^e$ can be computed for each element. The modification of the original function is done to account for that there are two degrees of freedom for each node since they can be displaced both horizontally and vertically. The technical implementation of the slightly modified function `plantml2d` is found in the appended code. The matrices $\boldsymbol{T}^e$ can subsequently be assembled by using the topology matrix **Edof** since the nodes that lie in the subdomain Ω_{lens} are known. Finally the matrix vector multiplication, defined in equation 23, is performed, yielding the result presented in section 3.4. The actual computation of the displacement magnitude squared is only done on the top part of the lens and subsequently doubled to obtain the total displacement magnitude

squared of the lens. This more computationally effective method can be used by virtue of the symmetry of Ω_{lens}.

3 Results

The results from the solutions to the problems presented in section 1.2 are given in the following sections. Firstly, we pose the results of the stationary heat flow problem, followed by the results of its time dependent counterpart. Pursuing, the solution to the mechanical problem is displayed. Lastly, the displacement field is visualized and quantified.

3.1 Stationary heat flow

The temperature distribution of the lens during day and night are found in figures 3 and 4, respectively. During the day, the temperature can be read to range from being approximately 34 °C in the lens closest to the ambiance, to being about 31 °C in the back of the camera. The temperature is continuously decreasing between these two subsets of the body. In contrast, the situation is essentially reversed during the night. Figure 4 displays that the lens is now about -82 °C, and roughly -46 °C in the rear of the camera. Similarly, the temperature changes in a continuous manner.

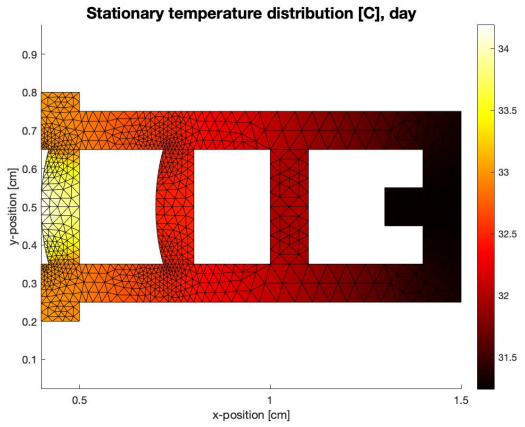


Figure 3: Stationary temperature field within the lens during daytime. The ambient temperature is $T_\infty = 40$ °C, and the temperature at the rightmost boundary is $T_c = 20$ °C.

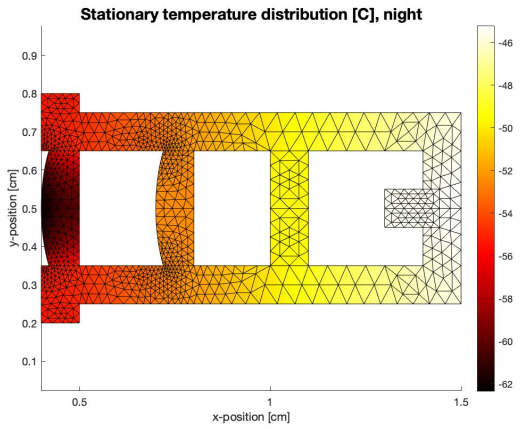


Figure 4: Stationary temperature field within the lens during nighttime. The ambient temperature is $T_\infty = -96$ °C, and the temperature at the rightmost boundary is $T_c = 20$ °C.

The maximal temperature during the day is 34.0 °C, and -45 °C during the night.

3.2 Transient heat flow

The transient temperature field is shown at equidistant points in time between time $t_0 = 0$ s and $t_f = 360$ s during day and night in figures 5, 6, respectively. The temperature color-scale is the same in both figures, and range from about -70 °C to 40 °C. As is observed, after roughly the time t_f, the temperature has reached a stationary state. For instance, on the day, the temperature evolves from it's initial 'night' stationary temperature to the stationary temperature during the day calculated in section 3.1. The analogous case is observed in figure 6 for temperature evolution during the night.

To more clearly display the local temperature variations within the camera body at different points in time, the temperature field is also plotted at different times using a non-constant colormap scale. This is found in figures 7, 8 for day and night, respectively. In both figures, the local temperature distribution at time $t = 0$ resembles that of the initial condition, see figure 7a, 8a. Analogously, after time t_f, the local temperature distribution resembles that of the stationary temperature distribution for the corresponding part of the 24-hour period, see 7d, 8d.

According to figure 7b, at a time of the day when the temperature everywhere is less than both the ambient temperature to the left and to the right of the camera, the coolest part of the camera is found within the two middles lenses (those without direct contact to the surroundings). Also, at the same position in the x-direction as the two middle lenses, the temperature in the surrounding titanium alloy is greater than in the middle lenses. When the temperature

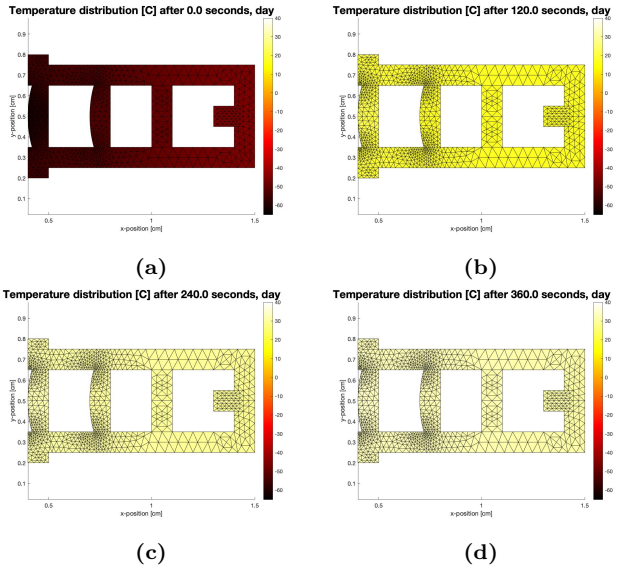


Figure 5: Temperature fields at different times after day has started, using stationary temperature distribution at night as initial condition.

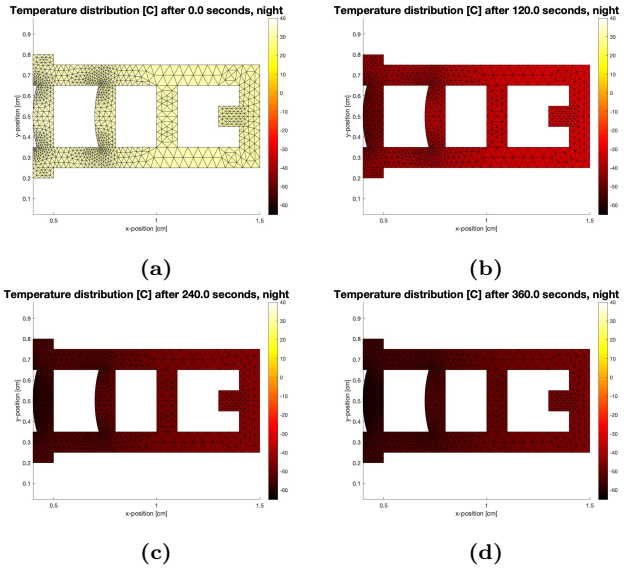


Figure 6: Temperature fields at different times after night has started, using stationary temperature distribution at day as initial condition.

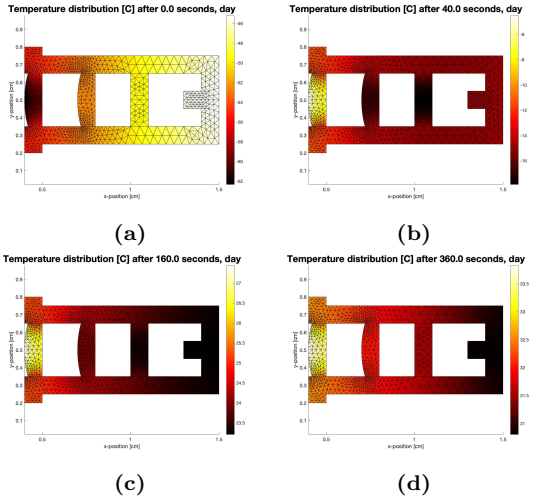


Figure 7: More refined local temperature variations at four times of the day. A variable color map scale is used.

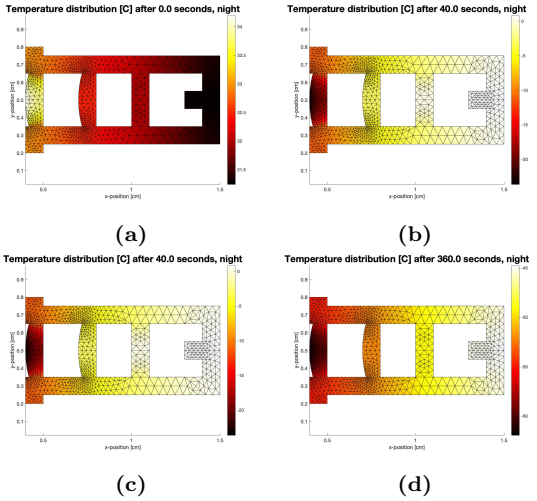
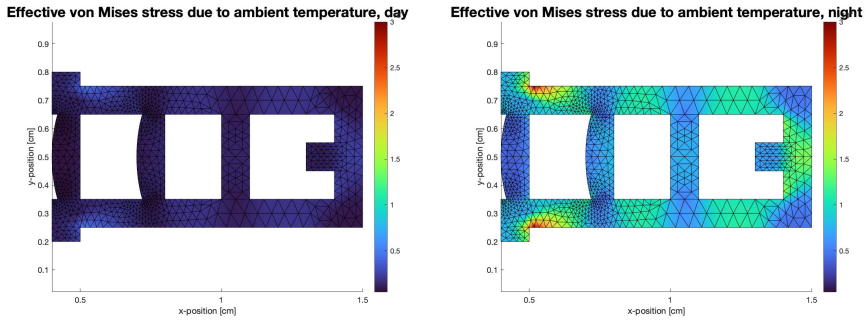


Figure 8: More refined local temperature variations at four times of the night. A variable color map scale is used.



(a) Day **(b)** Night

Figure 9: Effective von Mises stress distribution due to stationary temperature distribution in the body during the (a) day, and (b) night. Thermoelasticity is assumed. The color scale represents stresses ranging from 0 N cm^{-2} to $3 \cdot 10^4$ N cm^{-2}.

everywhere starts to exceed that of the rightmost boundary temperature, as in figure 7c, the coolest part of the camera is now translating to the right.

An analogous event is occurring during the night. After only a short period of time, e.g. 19 seconds as in figure 8b, the temperature field is mirrored (in a relative sense). The coolest part of the camera is now on the opposite side. It is also observed that subset of the body with the highest temperature at this time is confined in the middle part of the rightmost lens.

3.3 Plane strain mechanical problem

The effective von Mises stress in the camera during stationary temperature conditions during day and night are plotted in figure 9. For comparability, the stresses are plotted using the same stress color-scale. As observed, the von Mises stress is greater in the body during the night than during the day. This holds pointwise everywhere in the camera's body. Also, during the night, the local stress variations is of greater magnitude than during the day: ranging from about 3.5 kNcm^{-2} to 29 kNcm^{-2} (relationship about 8.3:1) over different regions. During the day, the stress varies between 0.6 kNcm^{-2} and 5.1 kNcm^{-2} (about 8.5:1).

The regions of great stress coincide during day and night, as observed when comparing figures 9b and 10. Generally, the greatest stresses are found within the titanium alloy. Particularly, the region in the titanium to the right of the leftmost lens, right next to the boundary, is subject to very large stresses. The stress there is about eight times greater than in the leftmost lens, and about twice as great as in the other high-stress areas of the titanium alloy. Also, the middle region in the rightmost part of the camera is subject to greater-than-average stresses. Other, weaker, stresses are also found in the regions of the titanium alloy between the lenses.

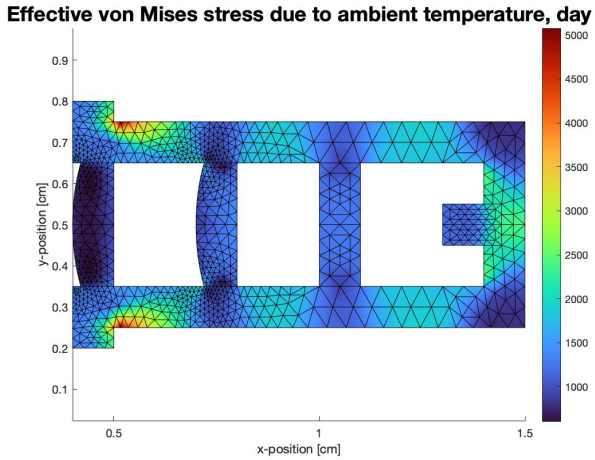


Figure 10: Effective von Mises stress distribution due to stationary temperature distribution in the body during the day, assuming thermoelasticity. The color scale represents stresses ranging from 0 N cm^{-2} to $5 \cdot 10^3$ N cm^{-2}.

3.4 Visualizing and quantifying the displacement

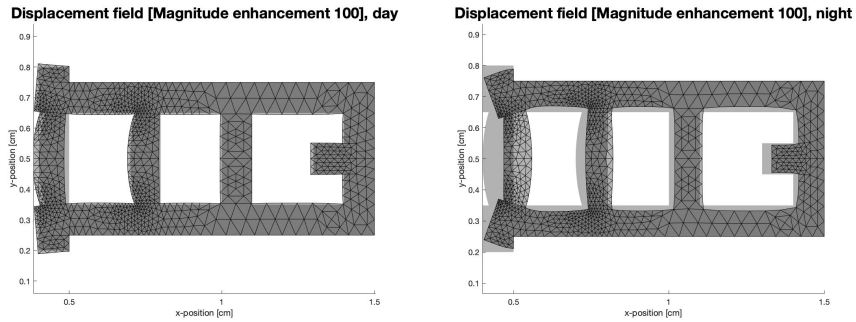
The displacement fields of the camera during day and night are displayed in figure 11. All displacements are magnified by a factor of 100 for the deformation be to visually observable. For both day and night, the displacement is the most severe around the position of the leftmost lens. In general the lens seem to be more deformed than the titanium alloy. Also, the deformation of the leftmost lens seem to be much greater during the night than during the day.

The displacement magnitude squared, defined in equation 23, in the lens for day and night conditions are presented in table 3.

	Night	Day
$\mathbf{a}^{\mathrm{T}}\mathbf{T}\mathbf{a}$ [cm^5]	$5.4 \cdot 10^{-9}$	$1.6 \cdot 10^{-10}$

Table 3: Displacement magnitude squared under day and night conditions.

To compare the structural integrity of the body under day and night conditions we define the relative displacement magnitude squared as $(\mathbf{a}^{\mathrm{T}}\mathbf{T}\mathbf{a})_{\mathrm{night}}/(\mathbf{a}^{\mathbf{T}}\mathbf{T}\mathbf{a})_{\mathrm{day}}$. The relative displacement magnitude squared is approximately 33.6. Judging from this and figure 11 one can conclude that the displacement is larger during the night compared to the day.



(a) Day **(b)** Night

Figure 11: Displacement fields during the (a) day and (b) night. All displacements are magnified by a factor of 100. The light grey region represents the camera without displacements. The dark grey region represents the deformed camera.

4 Discussion

By examining the results from solving the problems presented in section 1.2.2 we can draw conclusions regarding the thermal and thermoelastic mecanical properties of Perserverance's optical instruments. The following sections examines and discusses the results from section 3, in the same order as they were presented.

4.1 Stationary heat flow

Figures 3 and 4 show a result that is congruent with the intuition on heat flow. One notices a continuous temperature distribution where the temperature lies between the ambient temperatures at the boundaries. The temperature in the body gradually shifts from being cold on the colder side to warm on the warmer side. This is also mathematically reasonable as when the difference in temperature with the ambiance is greater, the flux q_n should also be greater. When a steady state is reached the heat influx on the colder side should be balanced by the outflow of heat on the other side, as prescribed by the balance principle from which the strong form of the problem is derived. Thus, a temperature distribution in between the two ambient temperatures should be expected. Figures 3 and 4 display that this result holds for both day and night.

The continuity of the temperature field in these figures also display that the use of linear triangular element, with linear element shape functions, fulfill the convergence criteria (and particularly compatibility) as was postulated in section 2.1.3.

4.2 Transient heat flow

The results in figures 7 and 8 also show a result that is congruent with the intuiton for how one expects the heat distribution to change over time. The temperature starts in one stationary state and gradually changes to the other one when the exterior temperatures are changed. This is to be expected since the word stationary implies a solution that is reached when sufficient time has elapsed. It is therefore reasonable that the solution to the transient problem, with the initial conditions of the stationary night time distribution shown in figure 4, gradually approaches the stationary day time temperature field shown in figure 3, and vice versa. This transition occurs over a time span of approximately six minutes. One might come to the conclusion that this is an unreasonable result since a Martian day lasts approximately 24 hours and 37 minutes. This observation is worth noting since it illustrates a fatal flaw in the model used for solving the transient heat flow problem. The flaw in our model is the non-physicality of the ambient temperature T_c instantly flipping from -96 °C to 40 °C. The night time stationary temperature on the leftmost convection boundary is approximately -62 °C in the initial state. The large difference between the ambient temperature of 40 °C and the boundary temperature of -62 °C creates a large Newton convection flux as prescribed by equation (8). This huge influx of heat rapidly changes the temperature and explains why the daytime stationary temperature field is reached so swiftly. One can conclude that the model needs improvement to account for this fact if it is to be used in practice. To account for the day time cycle of Mars the ambient temperature also needs to be non-constant. However, this would result in a more complicated FE formulation since the model would contain a time dependent convection boundary condition. The boundary load vector $\mathbf{f}_b$ would in this case be time dependent and thus complicate the scheme used for time stepping.

By examining, for example, figure 8b), where the initial condition is the stationary night time field and the ambient temperature is flipped to the day time temperature, one can make another interesting observation. Looking at the plot after an elapsed time of 40 seconds one may note that the temperature of the glass lenses in the middle of the camera are colder, compared to the titanium alloy in the same horizontal location. The large influx of heat generated by the Newton convection on the left boundary seem to reach the middle part of the titanium faster than it reaches the middle part of the glass. This can be explained by the thermal conductivity of titanium alloy, found in table 1, being substantially larger than the one of glass. This observation illustrates how the model is able to capture the physical properties of the materials in the camera. Finally, the result found in figure 7c also seems physically reasonable. When the temperature everywhere in the body starts to exceed that of the right hand side ambient temperature (of 20 °C), the right hand side of the titanium alloy starts to get relatively cooler in comparison to the rest of the body. That is, the right hand side is getting relatively darker than before. This is because at this point in time, the high temperature of the body makes heat starting to flow out of the body's right hand side boundary, while heat is still flowing in from the

left hand side.

4.3 Mechanical problem

The result that the stress within the body is everywhere pointwise greater during the night than during the day seems reasonable. As we have assumed thermoelastic stress to exist within the body, the far greater temperature difference during the night than during the day, in relation to its stress free condition, should also yield greater internal stress. Also, the fact that the quotient between the maximum and minimum stresses is almost the same during day and night might follow from the stress distribution being dependent on structural and geometrical properties of the body rather than its absolute temperature. This seems reasonable, at least for small enough temperature differences such that linear elasticity still holds.

The plausibility of the observation that the regions of greatest stress are confined in the titanium alloy can be reinforced in two ways. Firstly, the material properties of titanium should yield a greater stress per unit of strain than in glass (cf. the values of E and ν used in the constitutive matrix $\boldsymbol{D}$). Secondly, most of the titanium alloy's boundary is assumed to be fixed in place (through the prescription of homogeneous Dirichlet conditions along almost all of the outer boundary), which in turn requires a reactive stress to counteract any displacement from other parts of the body that are free to expand. This explains why the area of greatest stress lie so close to the boundary.

Furthermore, it is reasonable that the lump in the rear of the camera is is close to stress free. This stems from the lack of displacement restrictions around the lump making it free to expand. Hence, there should be no counteracting internal stress in the body around that area.

4.4 Visualizing and quantifying the displacement

The displacement field in figure shown in figure 11 show an expected deformation pattern for the camera. During daytime the temperature is larger compared to the stress free temperature T_0 and one would therefore expect thermal expansion of both the lenses and the enclosing titanium alloy (cf. the definition of thermoelastic stress). As prescribed by the model, no displacement occurs on the top-, bottom- and rightmost boundary. In the right side of the body, the lens and circumjacent alloy starts to protrude outwards due to the thermal expansion. This creates a bend at the location where the traction free boundary meets the boundary that is fixed in place. The locations of these bends coincides with the points of highest stress in the body. Judging from the figures displaying stress coupled with the figures displaying displacement one would intuitively assume that these points have the highest risk of a tear.

Figure 11b, which shows the displacement during nighttime conditions, illustrate an inverted scenario. The temperature is now lower than the stress free temperature T_0 and the body is therefore subject to thermal contraction. The contraction of lens and titanium makes the leftmost lens and alloy cave in,

changing the shape of the lens from convex to concave. Similarly to the daytime result, the contraction of the left hand part of the body creates a bend where the traction free boundary meets the one that is fixed in place. Note that the location of this bend also coincides with the highest stress concentration in the lens for nighttime conditions found in 9b. Furthermore, the large stresses in the titanium alloy between the parts of the body where the lenses are attached, as seen in figure 9, can be explained by the displacement pattern in figure 11b. The thermal contraction is severe in these areas, which because of the fixed outer boundary of the titanium alloy has to come with a reactive internal stress. Else, the boundary condition would not be satisfied.

The displacement in figure 11 seems much larger for nighttime conditions (b) when compared to daytime conditions (a). This observation is strengthened by the quantitative results in table 3 that contains the displacement magnitude squared for day- and nighttime conditions. As explained in section 3.4 the displacement magnitude squared is 33 times larger during the night compared to during the day. This is to be expected since the driving force behind the displacement is the thermal expansion or contraction. During nighttime the temperature difference between the current temperature and the stress free temperature is larger compared to this difference during the daytime which results in a greater nightly displacement.

It is important to note that the deformation pattern shown in figure 11 is magnified by a factor of 100. The displacement of the lens is in reality quite small, however even small deformations might interfere with the optical properties of the lens, by for instance causing optical abberations. It is also worth noting that the amount of deformation varies greatly from day to night. This might be undesirable from a performance perspective and a possible improvement could therefore be to construct the camera in a way such that the stress free temperature T_0 is evenly centered between the average nighttime and daytime temperature of the body.

5 Final remarks

This project has enriched our understanding of how the finite element method is applied in practice. By using the theory acquired from the lectures and the book and applying it in practice we have received a glimpse of how it is to work as a real engineer.

In retrospect there are a few improvements we would like to make to our model. Most notably the computer code, found in the appendix, does not deploy a sparse matrix implementation. This is unfortunate since most of the matrices used in the finite element analysis are sparse and therefore not optimally implemented. A use of sparse would make our program more effective since it would conserve memory. It would furthermore be interesting to expand our model to include non-linear concepts such as fracture mechanics or non-elastic materials. Our somewhat educated guess is that one has to deal with non-linear materials when constructing more refined models of interplanetary optical instrument. We

look forward to learn more about these advanced topics in the future when we will take part in second cycle courses *Computational inelasticity* and *Fracture mechanics* provided by the division of solid mechanics.

6 Computer Code

6.1 Main method

```
%% Stationary heat flow
clear all
clc

% Create mesh
load("petIMPORTANT.mat"); % pdetool % -> p e t

% Params
Tleft_day = 40;
Tleft_night = -96;

% Solve problem for day and night: calculate nodal temperatures
[a_day, K_day, f_day, ex, ey, eT_day, edof] = stat_heatflow(Tleft_day, p, e, t);
[a_night, K_night, f_night, ex, ey, eT_night, edof] = stat_heatflow(Tleft_night,
                                                                    p, e, t);

% Plot solution
plot_patch(ex, ey, eT_day, 'Stationary temperature distribution [C], day',
                    hot, 15);
plot_patch(ex, ey, eT_night, 'Stationary temperature distribution [C], night',
                    hot, 15);

% Calculate maximal temperatures
maxTempDay = max(a_day)
maxTempNight = max(a_night)

% Save variables
save("a_variables.mat", "a_day", "a_night", "K_day", "K_night",
        "f_day", "f_night", "ex", "ey", "eT_day", "eT_night", "edof");

%% b) Transient heat flow
clear all
clc

% Import data from a)
load("petIMPORTANT.mat");
load("a_variables.mat", "a_day", "a_night", "K_day", "K_night",
```

```
        "f_day", "f_night", "ex", "ey", "edof");

% Params
total_time = 6*60;
number_of_timesteps = 1000;
initial_value_day = a_night;
initial_value_night = a_day;

% Calculate nodal temperatures at all times
[aTot_day] = transient_heatflow(K_day, f_day, total_time, number_of_timesteps,
                                        initial_value_day, p, t);
[aTot_night] = transient_heatflow(K_night, f_night, total_time,
                            number_of_timesteps, initial_value_night, p, t);

% Plot nodal temperatures at different times
number_of_figs = 20;
plot_transient_temps(aTot_day, edof, ex, ey, number_of_figs, total_time,
                                 number_of_timesteps, "day");
plot_transient_temps(aTot_night, edof, ex, ey, number_of_figs, total_time,
                                 number_of_timesteps, "night");

%% c) Mechanical problem: thermoelasticity (plane strain)
clear all
clc

% Import data from a)
load("petIMPORTANT.mat");
load("a_variables.mat", "eT_day", "eT_night", "ex", "ey");

[vm_day, u_day, ue_day, Edof] = mechanical_2d(eT_day, ex, ey, p, e, t);
[vm_night, u_night, ue_night, Edof] = mechanical_2d(eT_night, ex, ey, p, e, t);

% Plot effective stress
plot_patch(ex, ey, vm_day, "Effective von Mises stress due to ambient
                    temperature, day", turbo, 18)
plot_patch(ex, ey, vm_day, "Effective von Mises stress due to ambient
                    temperature, day", turbo, 18)
caxis([500 4e4]);
plot_patch(ex, ey, vm_night, "Effective von Mises stress due to ambient
                            temperature, night", turbo, 18)
caxis([500 4e4]);

save("c_variables.mat", "u_day", "ue_day", "u_night", "ue_night", "Edof")
%% d) Displacement fields
clear all
clc
```

```
% Load needed data
load("petIMPORTANT.mat")
load("a_variables.mat", "ex", "ey");
load("c_variables.mat", "u_day", "u_night", "ue_day", "ue_night", "Edof");

[square_sum_day] = square_sum_displacements(u_day, Edof, p, t);
[square_sum_night] = square_sum_displacements(u_night, Edof, p, t);

% Plot displacement field
plot_displacement_field(ue_day, ex, ey, 'Displacement field
            [Magnitude enhancement 100], day', 19);
plot_displacement_field(ue_night, ex, ey, 'Displacement field
            [Magnitude enhancement 100], night', 19);
```

6.2 Stationary heat flow

```
function [a, K, f, ex, ey, eT, edof] = stat_heatflow(Tleft, p, e, t)
%Solve problem a)

% Init params
coord = p';
numberOfNodes = length(coord);
k_gl = 0.8e-2; % W/cm K
k_ti = 17e-2;
thickness = 1;
load = 0;
alpha_c = 100e-4; % W/cm2 K

% Convert mesh into CALFEM quantities: e.g. edof
nelm=length(t(1,:));
edof(:,1)=1:nelm;
edof(:,2:4)=t(1:3,:)';
coord=p' ;
ndof=max(max(t(1:3,:)));
[Ex,Ey]=coordxtr(edof,coord,(1:ndof)',3);
eldraw2(Ex,Ey,[1,4,1])
title("Mesh of half the camera used in the FE analysis")

% a) Stationary heat flow: Compute and assemble element stiffness matrices and e
% Parameters
Kt = zeros(numberOfNodes);
fl = zeros(numberOfNodes, 1);

for i=1:length(t)
    % Get node coordinates
```

```
    node1 = t(1, i);
    node2 = t(2, i);
    node3 = t(3, i);

    subdomain = t(4, i);

    x1 = coord(node1, 1);
    y1 = coord(node1, 2);
    x2 = coord(node2, 1);
    y2 = coord(node2, 2);
    x3 = coord(node3, 1);
    y3 = coord(node3, 2);

    ex = [x1 x2 x3];
    ey = [y1 y2 y3];

    % Get constitutive matrix
    if (subdomain == 1) % subdomain 1 is titanium domain
        k = k_ti;
    else
        k = k_gl;
    end

    D = k*eye(2);

    % Calculate element stiffnes matrix for triangular element
    ep = thickness;
    eq = load;
    [Kte, fe] = flw2te(ex, ey, ep, D, eq);

    % Assemble element matrices
    el = i; % element number (== edof(1))
    indx = edof(el,2:end);
    Kt(indx,indx) = Kt(indx,indx)+Kte;
    fl(indx) = fl(indx) + fe;
end

% Handling the convection term
% Check which segments that should have convections
% Parameters
Tright = 20;

er = e([1 2 5],:); % Reduced e

conv_segments_left = [1 2 20 31]; % Choosen boundary segments (left side)
conv_segments_right = [8 9]; % Choosen boundary segments
```

```
edges_conv_left = []; % Matrix with node pair numbers that lie
            along (the left) convection boundary
edges_conv_right = [];
for i = 1:size(er,2)
    if ismember(er(3,i),conv_segments_left)
        edges_conv_left = [edges_conv_left er(1:2,i)];
    elseif ismember(er(3,i), conv_segments_right)
        edges_conv_right = [edges_conv_right er(1:2,i)];
    end
end

fb = zeros(numberOfNodes, 1);

% Iterate over all element boundaries along convection sub-boundaries
% Start with left convection boundary
for i=1:length(edges_conv_left)
    node1 = edges_conv_left(1, i);
    node2 = edges_conv_left(2, i);

    x1 = coord(node1, 1)
    y1 = coord(node1, 2);
    x2 = coord(node2, 1)
    y2 = coord(node2, 2);
    Li = sqrt((x1-x2)^2 + (y1-y2)^2) % length of the i:th convection sub-boundar

    integral = alpha_c*1/2*Li*Tleft*thickness; % boundary vector integral
    fb(node1) = fb(node1) + integral;
    fb(node2) = fb(node2) + integral;
end
edges_conv_left
edges_conv_right
% Right hand side convection boundary
for i=1:length(edges_conv_right)
    node1 = edges_conv_right(1, i);
    node2 = edges_conv_right(2, i);

    x1 = coord(node1, 1);
    y1 = coord(node1, 2);
    x2 = coord(node2, 1);
    y2 = coord(node2, 2);
    Li = sqrt((x1-x2)^2 + (y1-y2)^2); % length of the i:th
        convection sub-boundary

    integral = alpha_c*1/2*Li*Tright*thickness; % boundary vector integral
    fb(node1) = fb(node1) + integral;
```

```
    fb(node2) = fb(node2) + integral;
end

% Calculating the M-matrix (from the convection boundary term)
M = zeros(numberOfNodes);

% Iterate over all element boundaries along
convection sub-boundaries
% Start with left convection boundary
for i=1:length(edges_conv_left)
    node1 = edges_conv_left(1, i);
    node2 = edges_conv_left(2, i);

    x1 = coord(node1, 1);
    y1 = coord(node1, 2);
    x2 = coord(node2, 1);
    y2 = coord(node2, 2);
    Li = sqrt((x1-x2)^2 + (y1-y2)^2); % length of the i:th
                        convection sub-boundary

    integral = alpha_c*thickness*1/6*Li*[2, 1; 1, 2]; % Easy integral of
        element shape function

    % Assemble
    M(node1, node1) = M(node1, node1) + integral(1, 1);
    M(node1, node2) = M(node1, node2) + integral(1, 2);
    M(node2, node1) = M(node2, node1) + integral(2, 1);
    M(node2, node2) = M(node2, node2) + integral(2, 2);
end

% Now right hand side convection boundary
for i=1:length(edges_conv_right)
    node1 = edges_conv_right(1, i);
    node2 = edges_conv_right(2, i);

    x1 = coord(node1, 1);
    y1 = coord(node1, 2);
    x2 = coord(node2, 1);
    y2 = coord(node2, 2);
    Li = sqrt((x1-x2)^2 + (y1-y2)^2); % length of the i:th
                    convection sub-boundary

    integral = alpha_c*thickness*1/6*Li*[2, 1; 1, 2]; % Easy integral
        of element shape function

    % Assemble
```

```
    M(node1, node1) = M(node1, node1) + integral(1, 1);
    M(node1, node2) = M(node1, node2) + integral(1, 2);
    M(node2, node1) = M(node2, node1) + integral(2, 1);
    M(node2, node2) = M(node2, node2) + integral(2, 2);
end

% Formulate full problem
K = Kt + M;
f = fb + fl;

a = solveq(K, f);

% Plot
% Extract element temperatures
eT = extract(edof, a);
ex = zeros(length(eT), 3); % numberOfElements long, number of nodes
per element wide
ey = zeros(length(eT), 3);

for i=1:length(edof)
    x1 = coord(edof(i,2), 1);
    y1 = coord(edof(i,2), 2);
    x2 = coord(edof(i,3), 1);
    y2 = coord(edof(i,3), 2);
    x3 = coord(edof(i,4), 1);
    y3 = coord(edof(i,4), 2);

    ex(i,:) = [x1 x2 x3];
    ey(i,:) = [y1 y2 y3];
end

end
```

6.3 Transient heat flow

```
function [aTot] = transient_heatflow(K, f, total_time,
            number_of_timesteps, initial_value, p, t)
% PURPOSE: Calculates nodal temperatures of the body, between
                times 0 and total_time.
%
% K: global stiffness matrix
% f: global force vector
% initial_value: field values at time 0

%
% Convert mesh into CALFEM quantities: e.g. edof
```

```
nelm=length(t(1,:));
edof(:,1)=1:nelm;
edof(:,2:4)=t(1:3,:)';
coord=p';
ndof=max(max(t(1:3,:)));

% Parameters
thickness = 1;
rho_c_gl = 3860e-6*670; % J/K cm3
rho_c_ti = 4620e-6*523;
numberOfNodes = length(coord);

% Compute C-matrix elementwise and assemble
C = zeros(numberOfNodes);
rho_c = 0;
for i=1:length(t)
    % Get node coordinates
    node1 = t(1, i);
    node2 = t(2, i);
    node3 = t(3, i);

    subdomain = t(4, i);

    x1 = coord(node1, 1);
    y1 = coord(node1, 2);
    x2 = coord(node2, 1);
    y2 = coord(node2, 2);
    x3 = coord(node3, 1);
    y3 = coord(node3, 2);

    posx = [x1 x2 x3];
    posy = [y1 y2 y3];

    % Get correct constant rho*c for corresponding subdomain
    if (subdomain == 1) % subdomain 1 is titanium domain
        rho_c = rho_c_ti;
    else
        rho_c = rho_c_gl;
    end

    Ce = plantml(posx, posy, rho_c);

    indx = edof(i,2:end);
    C(indx,indx) = C(indx,indx)+Ce;
end
```

```
% Time stepping parameters
dt = total_time/number_of_timesteps;

% Implicit Euler, append to large vector aTot
aTot = zeros(numberOfNodes, number_of_timesteps);
aTot(:,1) = initial_value;
for i=2:number_of_timesteps
    aTot(:, i) = (C+dt*K)\(C*aTot(:,i-1) + dt*f);
end

end
```

6.4 Mechanical problem

```
function [vonMises_nodal_perelement, u, ue, Edof] =

mechanical_2d(eT, ex, ey, p, e, t)

% Convert mesh into CALFEM quantities: e.g. edof
nelm=length(t(1,:));
edof(:,1)=1:nelm;
edof(:,2:4)=t(1:3,:)';
coord=p';
ndof=max(max(t(1:3,:)));

% Params
E_ti = 110e5; % N/cm2
E_gl = 67e5;
nu_ti = 0.34;
nu_gl = 0.2;
expansion_ti = 9.4e-6;
expansion_gl= 7e-6;
numberOfNodes = length(coord);
numberOfDofs = 2*numberOfNodes;

% Construct new Edof
Edof = zeros(length(edof), 7);
for i=1:length(edof)
    Edof(i, 1) = i;
    Edof(i, 2) = 2*edof(i, 2) - 1;
    Edof(i, 3) = 2*edof(i, 2);
    Edof(i, 4) = 2*edof(i, 3) - 1;
    Edof(i, 5) = 2*edof(i, 3);
    Edof(i, 6) = 2*edof(i, 4) - 1;
    Edof(i, 7) = 2*edof(i, 4);
end
```

```
% Determine K-matrix and f0 (thermoelasticity force vector)

% Params
nu = 0;
E = 0;
T0 = 20;
expansion = 0;
ep = [2 1];

% Initialize matrices/vectors
K = zeros(numberOfDofs);
f0 = zeros(numberOfDofs, 1);
epsilon_0 = zeros(numberOfDofs, 1);

% Iterate over all elements
for i=1:length(t)
    % Determine constitutive matrix depending on region
    subdomain = t(4, i);
    if subdomain == 1
        E = E_ti;
        nu = nu_ti;
        expansion = expansion_ti;
    else
        E = E_gl;
        nu = nu_gl;
        expansion = expansion_gl;
    end
    %D = E/(1+nu)/(1-2*nu)*[1-nu, nu, 0; nu, 1-nu, 0; 0, 0, 1/2*(1-2*nu)];
    D = hooke(2, E, nu);

    % Element stiffness matrix
    Ke = plante(ex(i,:), ey(i,:), ep, D);

    % Element thermoelastic
    epsilon_0 = expansion*(mean(eT(i,:)) - T0) * [1; 1; 1; 0];
    D_epsilon_0 = D*epsilon_0; % using average
    nodal temperature in element for deltaT.
    f0e = plantf(ex(i,:), ey(i,:), ep, D_epsilon_0');

    % Assemble f0e:s, D_epsilon_0
    f0 = insert(Edof(i,:), f0, f0e);

    % Assemble element matrices
    K = assem(Edof(i,:), K, Ke);
end
```

```
% fb: hom neumann boundary gives zero integral. The other integral can also
% be set to zero, as either the corresponding dof has a zero shape function
% at the boundary (e.g. if the element does not share element boundary with
% global boundary) or the element traction vecotr is unknown, why we can
% set it to whatever as we know we have prescribed values (0) in a (vector).

% Find dofs that lie on the homogenous Dirichlet (in x or y direction)
%boundary to create bc-vector
dir_segments_y = [14 15 10 11 28 29 30]; %last 5 are from symmetry
dir_segments_x = [4 8 9];
edges_dir_x = [];
edges_dir_y = [];
er = e([1 2 5],:); % Reduced e

for i = 1:size(er,2)
    if ismember(er(3,i), dir_segments_x)
        edges_dir_x = [edges_dir_x er(1:2,i)];
% matrix with all nodes lying on a boundary with hom. dir. bcs.
    elseif ismember(er(3,i), dir_segments_y)
        edges_dir_y = [edges_dir_y er(1:2,i)];
% matrix with all nodes lying on a boundary with hom. dir. bcs.
    end
end

% Remove duplicate nodes
unique_edges_dir_x = unique(edges_dir_x);
unique_edges_dir_y = unique(edges_dir_y);

dof_x = [];
dof_y = [];

% Convert the node labels into corresponding dofs
for i=1:length(unique_edges_dir_x)
    dof_x = [dof_x 2*unique_edges_dir_x(i)-1];
    % rule to convert to dof for x: dof = 2*node-1
end
for i=1:length(unique_edges_dir_y)
    dof_y = [dof_y 2*unique_edges_dir_y(i)];
    % rule to convert to dof for y: dof = 2*node
end

dirichlet_dofs = [dof_x dof_y]';

% Insert correct dofs into bc-vector
```

```
bc = [dirichlet_dofs zeros(length(dirichlet_dofs), 1)];

% Calculate nodal displacements
u = solveq(K, f0, bc);

% Get stresses and compute von Mises stress
ue = extract(Edof, u); % Element dof displacements
vonMises_element = zeros(length(t), 1);

for i=1:length(ue)
    % Check subdomain to determine constitutive matrix D
    subdomain = t(4, i);
    if subdomain == 1
        E = E_ti;
        nu = nu_ti;
        expansion = expansion_gl;
    else
        E = E_gl;
        nu = nu_gl;
        expansion = expansion_gl;
    end
    D = hooke(2, E, nu);
    epsilon_0 = expansion*(mean(eT(i,:)) - T0) * [1; 1; 1; 0];

    [es, et] = plants(ex(i,:), ey(i,:), ep, D, ue(i,:)); % es element stress
    es_temperature = es' - D*epsilon_0;
    vonMises_element(i) = sqrt(es_temperature(1)^2 +
    es_temperature(2)^2 + es_temperature(3)^2
    - es_temperature(1)*es_temperature(2)
    - es_temperature(1)*es_temperature(3)
    - es_temperature(2)*es_temperature(3)
    + 3*es_temperature(4)^2); %
    + 3*es(5)^2 + 3*es(6)^2); % per element
end

% Convert to nodal vonMises
vonMises_nodal = zeros(numberOfNodes, 1);

for i=1:size(coord,1) % for each node
    [c0, c1] = find(edof(:,2:4)==i);
    % get elements that contains node i (in a vector)
    vonMises_nodal(i,1)=sum(vonMises_element(c0))/size(c0,1);
% calculate mean stress on node
end

% Put correct nodal von Mises stresses in vector, a row per element
```

```
vonMises_nodal_perelement = extract(edof, vonMises_nodal);
end
```

6.5 Displacement visualisation

```
function [mag_squared] = square_sum_displacements(u, Edof, p, t)

coord = p';

% -- Compute square of the sum of the displacements --
% Get elements and dofs in lefternmost lens
lens_elements = [];
lens_dofs = [];
for i = 1:length(t)
    subdomain = t(4,i);
    if(subdomain == 3)
        lens_elements = [lens_elements i];
        lens_dofs = [lens_dofs; Edof(i, 2:end)];
    end
end
unique_lens_dofs = unique(lens_dofs);

% All other dofs displacement values are zero, except those
    who lie in lens.
u_lens = zeros(length(u),1);
for i = 1:length(unique_lens_dofs)
    u_lens(unique_lens_dofs(i)) = u(unique_lens_dofs(i));
end

% Get coords of nodes on every element, and compute The elementwise.
% Then assemble
T = zeros(length(u));
for i=1:length(lens_elements)
    element = lens_elements(i);

    x1 = coord(t(1, element), 1);
    y1 = coord(t(1, element), 2);
    x2 = coord(t(2, element), 1);
    y2 = coord(t(2, element), 2);
    x3 = coord(t(3, element), 1);
    y3 = coord(t(3, element), 2);

    Te = plantml2d([x1 x2 x3], [y1 y2 y3], 1);

    % Assemble
    T = assem(Edof(element,:), T, Te);
```

```
end

mag_squared = u_lens'*T*u_lens
end
```

6.6 Plotting

```
function [] = plot_patch(ex, ey, eT, title_str, colormap_obj, fontsize)
%UNTITLED6 Summary of this function goes here
%   Detailed explanation goes here

figure()
patch(ex',ey',eT')
title(title_str, 'FontSize', fontsize)
colormap(colormap_obj);
colorbar;
xlabel('x-position [cm]')
ylabel('y-position [cm]')
axis equal

hold on
patch(ex',-ey'+1,eT')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = plot_transient_temps(aTot, edof, ex, ey, number_of_figs,
    total_time, number_of_timesteps, day_or_night)

plot_times_indeces = round(linspace(1, number_of_timesteps, number_of_figs));
times = linspace(0, total_time, number_of_timesteps);
for i=1:length(plot_times_indeces)
    eT = extract(edof, aTot(:,plot_times_indeces(i)));
    if day_or_night == "day"
        str_title = sprintf("Temperature distribution [C] after %0.1f
        seconds, day",
        times(plot_times_indeces(i)));
    else
        str_title = sprintf("Temperature distribution [C] after %0.1f
        seconds, night", times(plot_times_indeces(i)));
    end
     plot_patch(ex, ey, eT, str_title, hot, 19);
    %caxis([-90 50])
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [] = plot_displacement_field(ue, ex, ey, title_str, fontsize)
mag = 100;
exd = ex + mag*ue(:,1:2:end);
eyd = ey + mag*ue(:,2:2:end);

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(title_str, 'FontSize', fontsize)

hold on
patch(ex',-ey'+1,[0 0 0],'EdgeColor','none','FaceAlpha', 0.3)
hold on
patch(exd',-eyd'+1,[0 0 0],'FaceAlpha',0.3)
xlabel('x-position [cm]')
ylabel('y-position [cm]')
end
```

6.7 Other methods

```
function Me=plantml(ex,ey,x)
% Ce=plantml(ex,ey,x)
%-------------------------------------------------------------------
% PURPOSE
%  Compute the quantity: Ce=x*int(N^T*N)dA
%
% INPUT:  ex,ey;        Element coordinates
%
%         x
%
% OUTPUT: Theta :       Matix 3 x 3
%-------------------------------------------------------------------


Area=1/2*det([ones(3,1) ex' ey']);



L1=[0.5 0 0.5];
L2=[0.5 0.5 0];
L3=[0 0.5 0.5];

NtN=zeros(6);
```

```
for i=1:3
        NtN=NtN+1/3*[L1(i)^2 0 L1(i)*L2(i) 0 L1(i)*L2(i) 0
                        0 L1(i)^2 0 L1(i)*L2(i) 0 L1(i)*L2(i)
                        L2(i)*L1(i) 0 L2(i)^2 0 L2(i)*L3(i) 0
                        0 L2(i)*L1(i) 0 L2(i)^2 0 L2(i)*L3(i)
                        L3(i)*L1(i) 0 L3(i)*L2(i) 0 L3(i)^2 0
                        0 L3(i)*L1(i) 0 L3(i)*L2(i) 0 L3(i)^2];
end

Me1=NtN*Area*x;


Me=Me1([1 3 5],[1 3 5]);

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Me=plantml2d(ex,ey,x)
% Ce=plantml(ex,ey,x)
%----------------------------------------------------------------
% PURPOSE
%  Compute the quantity: Ce=x*int(N^T*N)dA
%
% INPUT:   ex,ey;        Element coordinates
%
%          x
%
% OUTPUT:  Theta :       Matix 3 x 3
%----------------------------------------------------------------


Area=1/2*det([ones(3,1) ex' ey']);



L1=[0.5 0 0.5];
L2=[0.5 0.5 0];
L3=[0 0.5 0.5];

NtN=zeros(6);


for i=1:3
        NtN=NtN+1/3*[L1(i)^2 0 L1(i)*L2(i) 0 L1(i)*L2(i) 0
                        0 L1(i)^2 0 L1(i)*L2(i) 0 L1(i)*L2(i)
                        L2(i)*L1(i) 0 L2(i)^2 0 L2(i)*L3(i) 0
                        0 L2(i)*L1(i) 0 L2(i)^2 0 L2(i)*L3(i)
```

```
                                    L3(i)*L1(i)  0  L3(i)*L2(i)  0  L3(i)^2  0
                                    0  L3(i)*L1(i)  0  L3(i)*L2(i)  0  L3(i)^2];
end

Me=NtN*Area*x;
```