MHA021 - FEM Assignment 3 - Group 33

Nils Helgesson
Gabriel Wendel

December 2023

Task 1

Task 1 a) Derive strong form

To derive the strong form, we start by establishing a force equilibrium. Traction forces, $\mathbf{t}$, on the body surface and body forces, $\mathbf{b}$ must be equal to zero. Stresses inside a point on a body is represented by the stress matrix, $\mathbf{S}$, where $\mathbf{S} = \mathbf{S}^T$:

$$\mathbf{S} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix} \tag{1}$$

Stress on the surface, given by the normal vector $\mathbf{n}$, is represented by the traction vector, $\mathbf{t}$:

$$\mathbf{t}(\mathbf{n}) = \begin{bmatrix} \sigma_{xx}n_x + \sigma_{xy}n_y \\ \sigma_{xy}n_x + \sigma_{yy}n_y \end{bmatrix} = \mathbf{Sn} \tag{2}$$

To derive the strong form we need balance law, kinematics and constitutive relationship. Balance law in 2D states that:

$$\begin{cases} \dfrac{\partial \sigma_{xx}}{\partial x} + \dfrac{\partial \sigma_{xy}}{\partial y} + bx = 0 \\ \dfrac{\partial \sigma_{yx}}{\partial x} + \dfrac{\partial \sigma_{yy}}{\partial y} + by = 0 \end{cases} \tag{3}$$

Equation 3 can be rewritten using divergence operator:

$$\begin{aligned} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + bx &= \operatorname{div}(\begin{bmatrix} \sigma_{xx} \\ \sigma_{xy} \end{bmatrix}) + bx = \operatorname{div}(\mathbf{s_x}) + bx \\ \frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + bx &= \operatorname{div}(\begin{bmatrix} \sigma_{yx} \\ \sigma_{yy} \end{bmatrix}) + bx = \operatorname{div}(\mathbf{s_y}) + by \end{aligned}$$

Thus equations of equilibrium becomes:

$$\operatorname{div}(\mathbf{S}) + \mathbf{b} = \mathbf{0} \tag{4}$$

Equation of equilibirum can be written in Voigt form using given $\tilde{\boldsymbol{\nabla}}$ operator:

$$\begin{bmatrix} \dfrac{\partial}{\partial x} & 0 & \dfrac{\partial}{\partial y} \\ 0 & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} = \tilde{\boldsymbol{\nabla}}^T \boldsymbol{\sigma} + \mathbf{b} = \mathbf{0} \tag{5}$$

As previously stated we also need kinematic strains. Normal- and shear strains are defined as:

$$\begin{cases} \epsilon_{xx} = \dfrac{\partial u_x}{\partial x} \\ \epsilon_{yy} = \dfrac{\partial u_y}{\partial y} \\ \epsilon_{zz} = \dfrac{\partial u_z}{\partial z} \end{cases} \tag{6}$$

$$\begin{cases} \gamma_{xy} = \dfrac{\partial u_x}{\partial y} + \dfrac{\partial u_y}{\partial x} \\ \gamma_x z = \dfrac{\partial u_x}{\partial z} + \dfrac{\partial u_z}{\partial x} \\ \gamma_{yz} = \dfrac{\partial u_y}{\partial z} + \dfrac{\partial u_z}{\partial y} \end{cases} \tag{7}$$

From this we can define a strain vector, $\boldsymbol{\epsilon}$:

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} = \tilde{\boldsymbol{\nabla}} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \tilde{\boldsymbol{\nabla}}\mathbf{u} \tag{8}$$

Constitutive relationship can be stated using Hooke's generalised law and constitutive matrix, $\mathbf{D}$:

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\epsilon} \tag{9}$$

Combining equilibrium Equation 5, kinematics-strain Equation 8 and constitutive relationship Equation 9 we arrive at:

$$\tilde{\boldsymbol{\nabla}}^T \mathbf{D} \tilde{\boldsymbol{\nabla}} \mathbf{u} + \mathbf{b} = \mathbf{0} \tag{10}$$

Next step is to define boundary conditions. Right side of the beam is foxed to the wall, which means that displacement is zero along this edge. Upper right side near the wheel experiences Hertzian pressure, which is added to the traction vector in y-direction. On the left side of the half beam there is a symmetry line, indicating no displacement in the x-direction and traction is equal to zero. Additionally, there is a body force acting on the beam. The half beam can be divided into five boundaries, $\Gamma_1 - \Gamma_5$. Figure 1 illustrates defined boundaries for the beam.

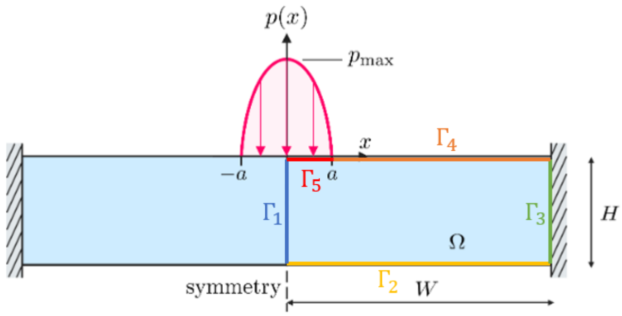


Figure 1: Defined boundaries for the beam.

Thus strong form for the specific problem at hand given previously defined boundaries Γ_1 to Γ_5 can be stated as:

Find $\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$ such that

$$
\begin{cases}
\tilde{\boldsymbol{\nabla}}^T \mathbf{D} \boldsymbol{\nabla} \mathbf{u} = \mathbf{0} & & \\
\mathbf{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} & u_x = 0 & \text{on } \Gamma_1 \\
\mathbf{t} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} & & \text{on } \Gamma_2 \\
\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} & & \text{on } \Gamma_3 \\
\mathbf{t} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} & & \text{on } \Gamma_4 \\
\mathbf{t} = \begin{bmatrix} 0 \\ -p(x) \end{bmatrix} = \begin{bmatrix} 0 \\ -p_{max}\sqrt{1 - \left(\dfrac{x}{a}\right)^2} \end{bmatrix} & & \text{for } 0 \leq x \leq a \text{ on } \Gamma_5 \\
\mathbf{b} = \begin{bmatrix} 0 \\ \rho g \end{bmatrix} & & \text{on } \Omega
\end{cases}
\tag{11}
$$

Task 1 b) Global FE form

Weak form, WE, is given by:

$$
\int_{\Omega} (\boldsymbol{\nabla} \mathbf{v})^T \mathbf{D} t \tilde{\boldsymbol{\nabla}} \mathbf{u} dA = \int_{\Omega} \mathbf{v}^T \mathbf{b} t dA + \oint_{\Gamma} \mathbf{v}^T \mathbf{t} t d\Gamma \tag{12}
$$

Where $\mathbf{v} = \begin{bmatrix} v_x(x, y) \\ v_y(x, y) \end{bmatrix}$ is a vector valued weight function.

Boundary term in Equation 12 can be expanded:

$$\int_{\Gamma} \mathbf{v}^T \mathbf{t} d\Gamma = \int_{\Gamma_g} \mathbf{v}^T \mathbf{t} d\Gamma + \int_{\Gamma_h} \mathbf{v}^T \mathbf{t} d\Gamma$$
$$= \int_{\Gamma_g} \mathbf{v}^T \mathbf{t} d\Gamma + \int_{\Gamma_h} \mathbf{v}^T \mathbf{h} d\Gamma \tag{13}$$

Weak form in Equation 12 can thus be rewritten as:

$$\begin{cases} \int_{\Omega} (\boldsymbol{\nabla}\mathbf{v})^T \mathbf{D} t \tilde{\boldsymbol{\nabla}} \mathbf{u} dA = \int_{\Omega} \mathbf{v}^T \mathbf{b} t dA + \int_{\Gamma_g} \mathbf{v}^T \mathbf{t} d\Gamma + \int_{\Gamma_h} \mathbf{v}^T \mathbf{h} d\Gamma \\ \mathbf{u} = \mathbf{g} \quad \text{on } \Gamma_g \end{cases} \tag{14}$$

Where $\mathbf{h}$ is the prescribed traction on Γ_g and $\mathbf{g}$ is the prescribed displacement on Γ_g. As previously decribed in Equation 11 traction is known for boundaries $\Gamma_1, \Gamma_2, \Gamma_4$ and Γ_5, which means that integration over these edges is given by integration over Γ_h. Deformation is given for boundary Γ_3, thus integration over this edge is given by corresponding integration over Γ_g. Equation 14 can be rewritten:

$$\int_{\Omega} (\boldsymbol{\nabla}\mathbf{v})^T \mathbf{D} t \tilde{\boldsymbol{\nabla}} \mathbf{u} dA = \int_{\Omega} \mathbf{v}^T \mathbf{b} t dA + \int_{\Gamma_3} \mathbf{v}^T \mathbf{t} d\Gamma + \int_{\Gamma_1} \mathbf{v}^T \mathbf{h} d\Gamma + \int_{\Gamma_2} \mathbf{v}^T \mathbf{h} d\Gamma + \int_{\Gamma_4} \mathbf{v}^T \mathbf{h} d\Gamma + \int_{\Gamma_5} \mathbf{v}^T \mathbf{h} d\Gamma \tag{15}$$

As stated in Equation 11, traction vector on edges Γ_1, Γ_2 and Γ_4 are zero. Hence, integration over these edges can be neglected:

$$\int_{\Omega} (\boldsymbol{\nabla}\mathbf{v})^T \mathbf{D} t \tilde{\boldsymbol{\nabla}} \mathbf{u} dA = \int_{\Omega} \mathbf{v}^T \mathbf{b} t dA + \int_{\Gamma_3} \mathbf{v}^T \mathbf{t} d\Gamma + \int_{\Gamma_5} \mathbf{v}^T \mathbf{h} d\Gamma \tag{16}$$

Displacement, $\mathbf{u}$, is primary unknown:

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} u_x(x,y) \\ u_y(x,y) \end{bmatrix} \tag{17}$$

Which can be approximated, $\mathbf{u} \approx \mathbf{u_h}$, using shape functions, $\mathbf{N}$:

$$\begin{cases} \mathbf{u_x} \approx \mathbf{u_{xh}} = \begin{bmatrix} N_1 & \dots & N_n \end{bmatrix} \begin{bmatrix} a_{x1} \\ \vdots \\ a_{xn} \end{bmatrix} = \mathbf{N_x} \mathbf{a_x} \\ \mathbf{u_y} \approx \mathbf{u_{yh}} = \begin{bmatrix} N_1 & \dots & N_n \end{bmatrix} \begin{bmatrix} a_{y1} \\ \vdots \\ a_{yn} \end{bmatrix} = \mathbf{N_y} \mathbf{a_y} \end{cases} \Rightarrow$$

$$\mathbf{u} \approx \mathbf{u_h} = \begin{bmatrix} u_{xh} \\ u_{yh} \end{bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & \dots & N_n & 0 & 0 \\ 0 & N_1 & 0 & \dots & 0 & N_n & 0 \\ 0 & 0 & N_1 & \dots & 0 & 0 & N_n \end{bmatrix} \begin{bmatrix} a_{x1} \\ a_{y1} \\ \vdots \\ a_{xn} \\ a_{yn} \end{bmatrix} = \mathbf{N} \mathbf{a} \tag{18}$$

Approximation for weight function:

$$\mathbf{v} = \mathbf{N}\mathbf{c} = \begin{bmatrix} N_1 & 0 & 0 & ... & N_n & 0 & 0 \\ 0 & N_1 & 0 & ... & 0 & N_n & 0 \\ 0 & 0 & N_1 & ... & 0 & 0 & N_n \end{bmatrix} \begin{bmatrix} c_{x1} \\ c_{y1} \\ \vdots \\ c_{xn} \\ c_{yn} \end{bmatrix} \tag{19}$$

Approximation of the gradients:

$$\tilde{\boldsymbol{\nabla}}\mathbf{u} \approx \tilde{\boldsymbol{\nabla}}\mathbf{u}_h = \tilde{\boldsymbol{\nabla}}(\mathbf{N}\mathbf{a}) = \tilde{\boldsymbol{\nabla}}\mathbf{N}\mathbf{a} + \mathrm{N}\underbrace{\tilde{\boldsymbol{\nabla}}\mathbf{a}}_{=\mathbf{0}} = \tilde{\boldsymbol{\nabla}}\mathbf{N}\mathbf{a} = \mathbf{B}\mathbf{a} \tag{20}$$

Weight functions can be determined using Galerkin's method:

$$\tilde{\boldsymbol{\nabla}}\mathbf{v} = \tilde{\boldsymbol{\nabla}}\mathbf{N}\mathbf{c} = \mathbf{B}\mathbf{c} \tag{21}$$

Insert Equation 18, 19, 20 and 21 into weak form, Equation 16:

$$\int_\Omega \mathbf{c}^T\mathbf{B}^T\mathbf{D}t\mathbf{B}\mathbf{a}d\Omega = \int_\Omega \mathbf{c}^T\mathbf{N}^T\mathbf{b}d\Omega + \int_{\Gamma_3} \mathbf{c}^T\mathbf{N}^T\mathbf{t}d\Gamma + \int_{\Gamma_5} \mathbf{c}^T\mathbf{N}^T\mathbf{h}d\Gamma \tag{22}$$

Since $\mathbf{c}$ is arbitrary we can move it outside the integration. Thus FE-form can be defined as:

$$\begin{cases} \int_\Omega \mathbf{B}^T\mathbf{D}t\mathbf{B}\mathbf{a}d\Omega - \int_\Omega \mathbf{N}^T\mathbf{b}d\Omega - \int_{\Gamma_3} \mathbf{N}^T\mathbf{t}d\Gamma - \int_{\Gamma_5} \mathbf{N}^T\mathbf{h}d\Gamma = 0 & \\ \mathbf{u} = \mathbf{0} & \text{on } \Gamma_3 \\ u_x = 0 & \text{on } \Gamma_1 \end{cases}$$

$$\Leftrightarrow$$

$$\begin{cases} \mathbf{K}\mathbf{a} = \mathbf{f}_l + \mathbf{f}_b^g + \mathbf{f}_b^h & \text{in } \Omega \\ \mathbf{u} = \mathbf{g} & \text{on } \Gamma_g \end{cases}$$

Task 1 c) Plot the deformed and undeformed geometry.

Code can be found in appendix A and B. The deformation is plotted in figure 2 with a coarse mesh such that the figure is readable.

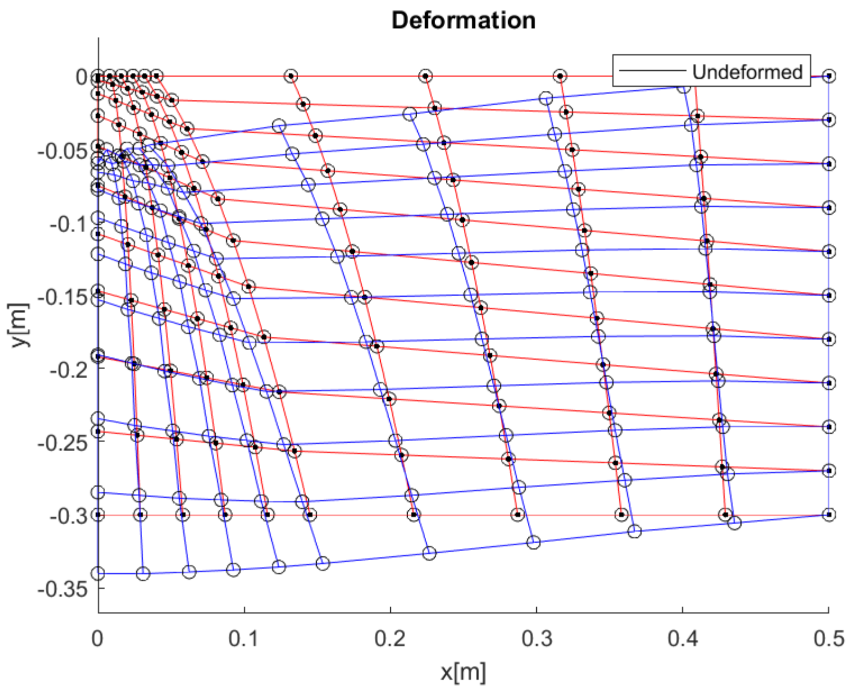


Figure 2: Deformation of the geometry with a coarse mesh. $nelH = nelW = nela = 10$ and scalefactor= 300

Validation of code

In order to validate the code, the external load is set to zero, and the beam dimensions set to:

$$H = 0.3 \text{ m}$$
$$W = 3 \text{ m}$$
$$t = 0.05\text{m}$$

The displacement from its mass is presented in figure 3 and a zoom in on the top left corner in figure 4. It can be noted that the maximum displacement is $\sim 1.7 \times 10^{-4}$ m

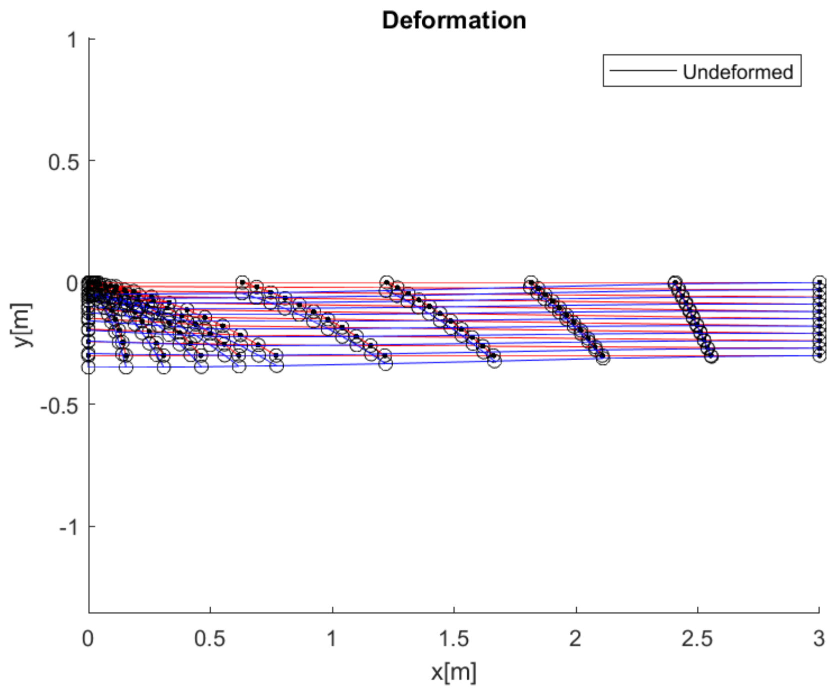


Figure 3: Displacement of beam with no external force. Scalefactor 300.

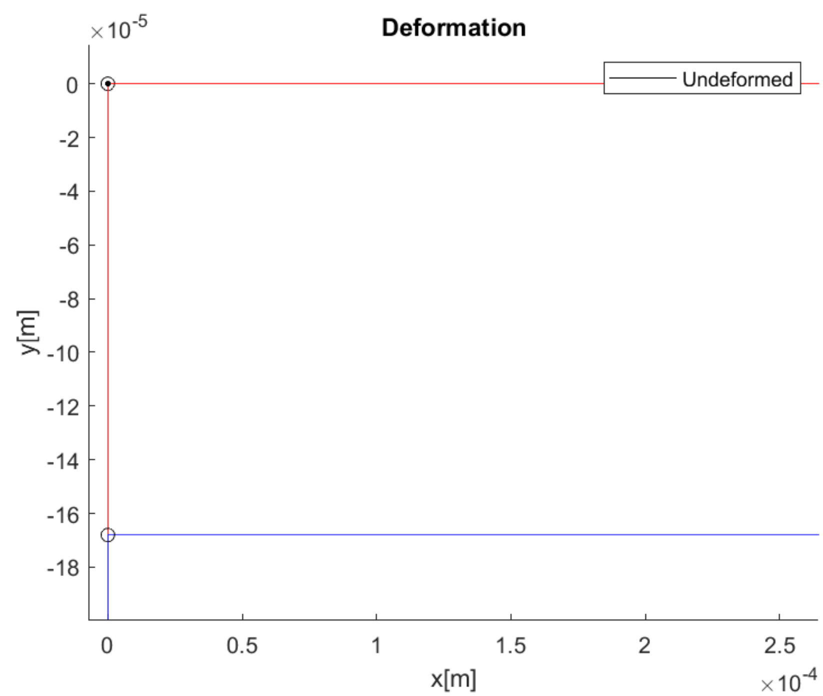


Figure 4: Displacement of top left corner with scalefactor=1

This is to be compared to a analytic solution for the whole beam, including symmetry. Given by figure

5 where

$$
\begin{aligned}
l &= 6 \text{ m} \\
w &= \rho g H t = 1178.4 \text{ N/m} \\
I &= \frac{tH^3}{12} = 1.125 \times 10^{-4} \text{ m}^4 \\
\Rightarrow \Delta_{max}(\text{at center}) &= \frac{wl^4}{384EI} = 1.68 \times 10^{-4} \text{ m}
\end{aligned}
$$

We see that FE code results in the same deflection at the center of the beam. We can also see that the boundary conditions are working, resulting at zero displacement at fixed end and $w'(L/2) = 0$ as the symmetry condition. This concludes that the FE program works as expected and is suitable to use for calculations including external forces.

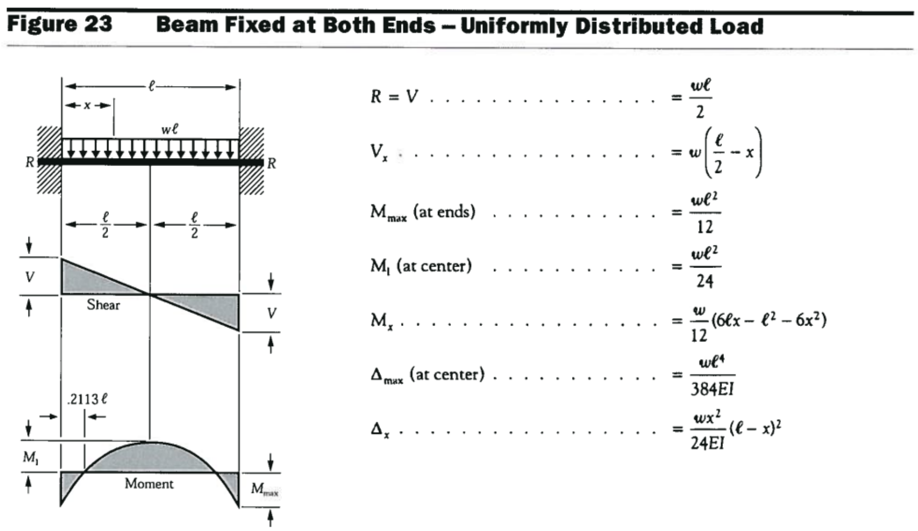


Figure 5: Analytic solution for beam fixed at both ends with distributed load. Taken from compendium *MMA169 - Structural Engineering*

Task 1 d) Derive the expression for stress vector σ

For the case at hand we have plane stress, for which stresses in z-direction are negligible, see Equation 23.

$$
\sigma_{yz} = \sigma_{xz} = \sigma_{zz} = 0 \quad \rightarrow \quad \epsilon_{zz} \neq 0 \tag{23}
$$

We previously defined the constitutive relationship in Equation 9. Thus, for a 2D case the stress strain relationship can be expressed as:

$$
\begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{xy} \end{bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & 1+\nu \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} \tag{24}
$$

By inverting the comliance matrix in Equation 24 we obtain the stiffness matrix for plane stress:

$$
\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & 1-\nu \end{bmatrix} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{xy} \end{bmatrix} \tag{25}
$$

The stress vector, $\boldsymbol{\sigma}$, can be defined for each element, e, using $\mathbf{B}^e$ and $\mathbf{a}^e$ in Equation 9:

$$\boldsymbol{\sigma}^e = \mathbf{D}\boldsymbol{\epsilon}^e = \mathbf{D}\tilde{\boldsymbol{\nabla}}\mathbf{u} = \mathbf{D}\mathbf{B}^e\mathbf{a}^e \tag{26}$$

Task 1 e) Calculate effective stress

The stress vector for each element were calculated according to the prior task. The result is presented in Figure 6. For this the mesh is finer than for the deformation plot.

Von Mises stresses were calculated according to Equation 27 for each element.

$$\sigma_{VonMises} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\sigma_{xy}^2} \tag{27}$$

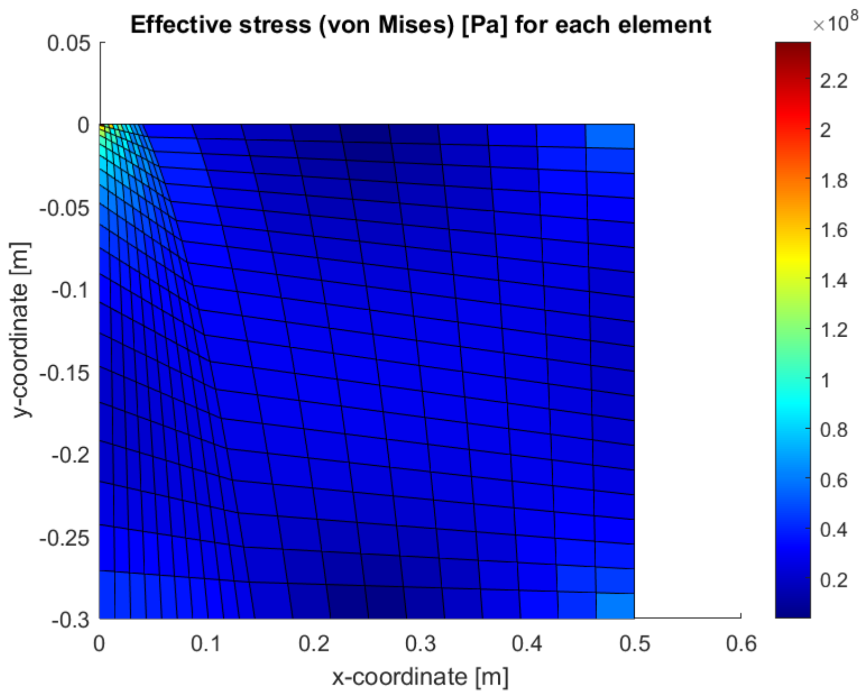
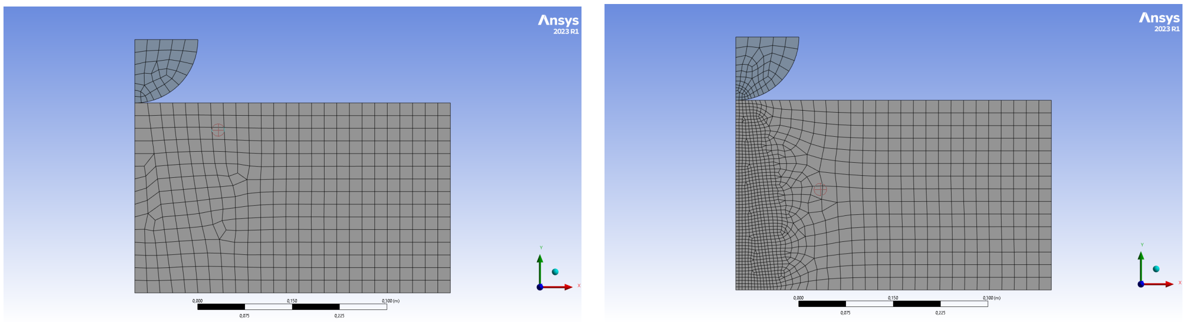


Figure 6: Von Mieses effective stresses for the geometry, here using a finer mesh with $nelH = nelW = nela = 20$

Task 2: FE-analysis of beam subjected to contact load from overhead crane using ANSYS

Two different meshes were used, one coarse, Figure 7a and one fine, Figure 7b. A comparison of results was made between these two meshes.



(a) Coarse mesh, mesh size 2×10^{-2} m for all bodies.

(b) Fine mesh with mesh size 2×10^{-3} m around contact point and deformation zone.

Figure 7: Coarse- and fine mesh for the two bodies.

Boundary conditions was applied in accordance to Equation 11. For the edges where there is no traction, $\mathbf{t} = \mathbf{0}$, no boundary was specified in ANSYS. To ensure that the two bodies do not behave as a single entity, the contact point was switched from bonded to frictionless. Given beam properties can be seen in Table 1. Body force was also applied to the beam.

Table 1: Beam properties.

Young's modulus, E	210 GPa
Poisson's ratio, ν	0.3
Density, ρ	8000 kg/m^3

Task 2 a) Use the commercial FE-software ANSYS to set-up and solve the 2D elasticity problem

Magnitude of the load was chosen so that it corresponds to that of Task 1, $F = \int_{-a}^{a} p(x)tdx$. A wheel radius of 0.1 m and a thickness of 0.05 m were employed. Thus to determine the force, integration was performed over the interval $-a \leq x \leq a$ m:

$$\begin{aligned}\frac{F}{2} &= \int_{-a}^{a} p(x)tdx = \int_{-a}^{a} p_{max}\sqrt{1-\left(\frac{x}{a}\right)^2}tdx \\ &= \int_{-0.04}^{0.04} 200\times 10^6\sqrt{1-\left(\frac{x}{0.04}\right)}0.05dx = \frac{628319}{2}\text{N} = 314159.5\text{N}\end{aligned} \tag{28}$$

Due to symmetry half of the force, F, was applied to the ANSYS model. This force can either be applied as a point force at the center of the wheel or as a distributed load along the radius of the wheel. We decided to apply the force, $-\frac{F}{2}$, on the wheel radius, see Figure 8.

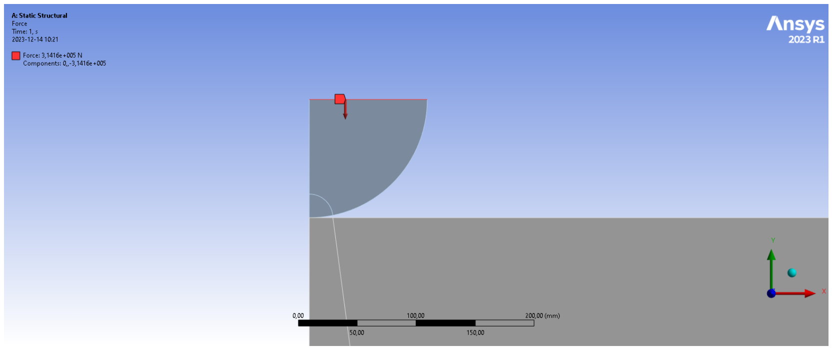


Figure 8: Applied force, $\dfrac{F}{2}$, on wheel radius.

Resulting deformation for the fine- and coarse mesh can be seen in Figure 9 and Figure 10 respectively.

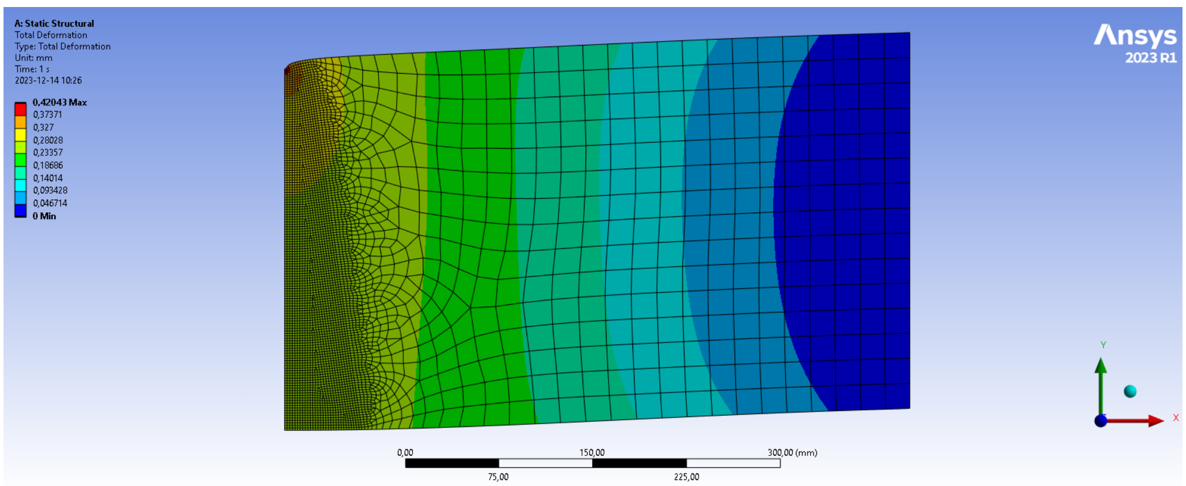


Figure 9: Deformation plot using fine meshing.

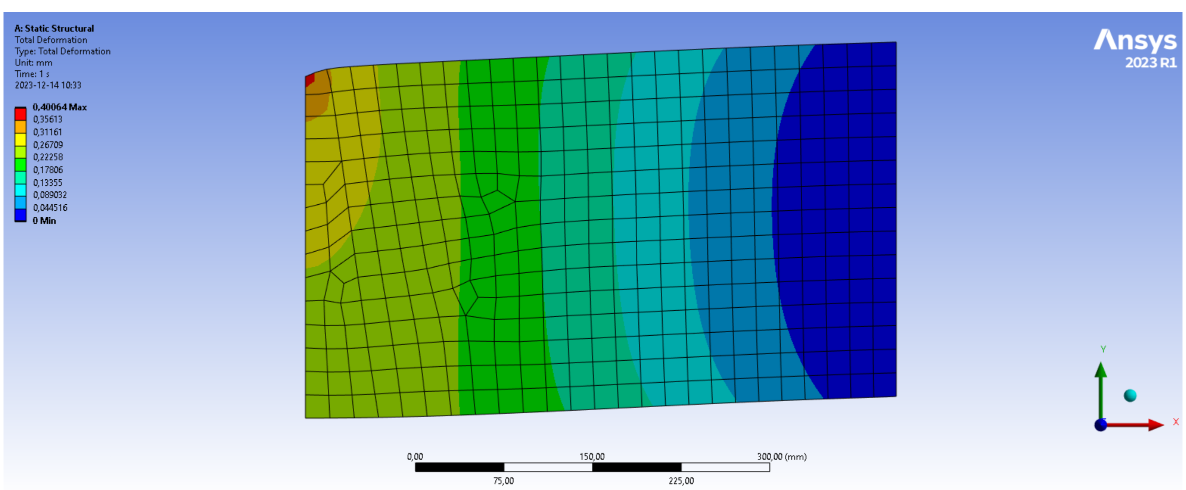


Figure 10: Deformation plot using coarse meshing.

For both cases the maximum deformation can be found near the contact region. The finer mesh results in a larger maximum displacement. Several factors might contribute to this.

Finer meshes provide a more accurate representation of the geometry and loading conditions. As the mesh becomes finer, it captures more details of the structural behavior, resulting in a more accurate solution. Additionally, if the mesh is not fine enough, especially around regions of high stress or strain, the FEA solution may not converge to a stable solution, leading to less accurate results. It's quite possible that the model with coarse mesh has not reached convergence hence the result is not as accurate compared to the fine mesh model. To verify the convergence and reliability of the results obtained from the finer mesh, a simulation with an even finer mesh (mesh size of 1 mm around the contact point) was performed. The maximum deformation observed was 0.42109 mm, a value closely comparable to the fine mesh with a mesh size of 2 mm. Therefore, it can be concluded that the results have most likely converged.

Task 2 b) Determine the maximum contact stress and estimate the width of the contact patch

Contact path between half wheel and beam is illustrated in Figure 11. Length of contact patch is approximately 17 mm.

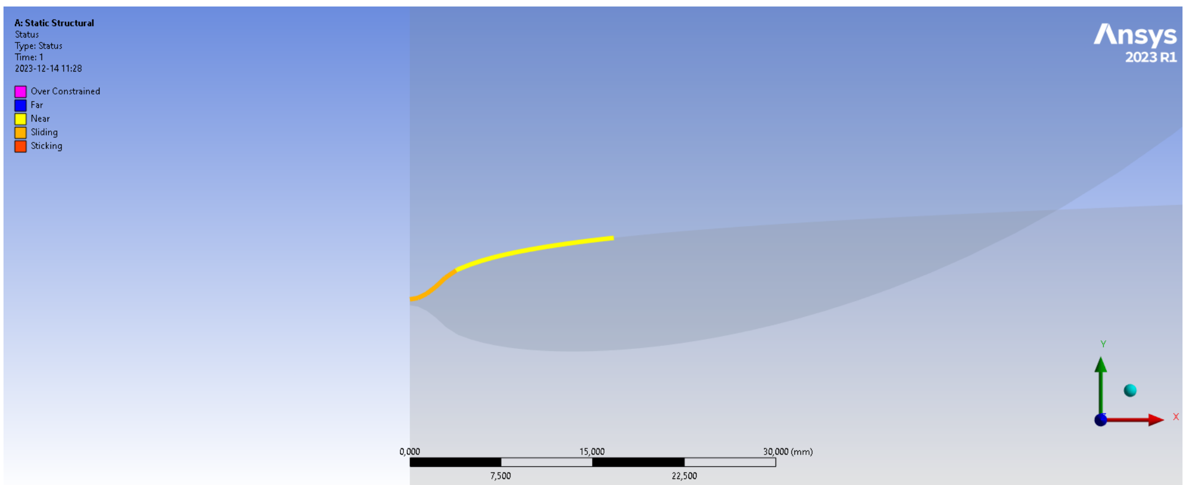


Figure 11: Contact patch.

In Figure 12, the contact pressure distribution between the half wheel and the beam is depicted. The maximum observed contact stress is 1943.3 MPa.

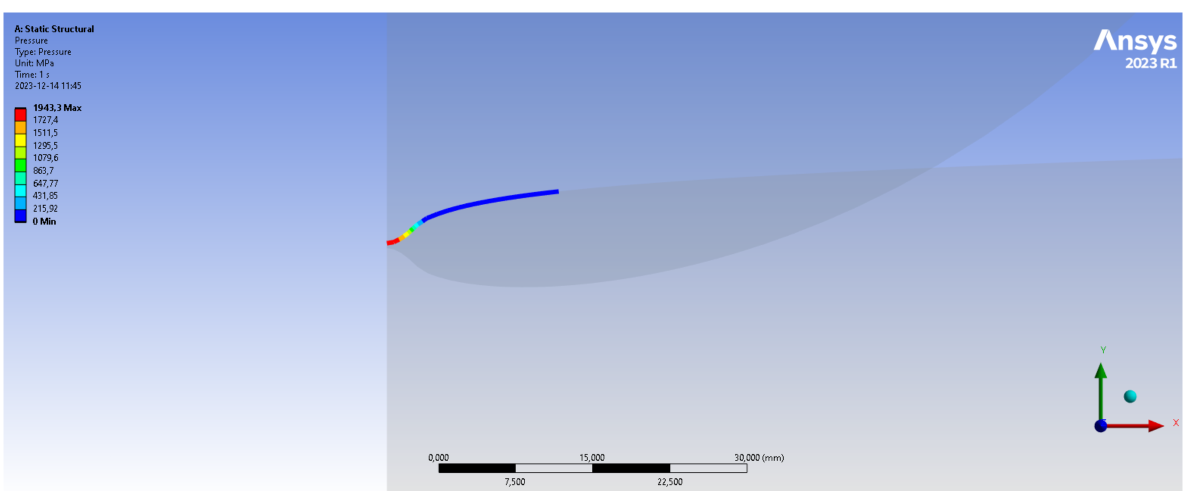


Figure 12: Contact pressure.

Task 2 c) Determine and plot the distribution of von Mises stress in the beam

Equivalent (von-Mises) stress around contact area using the fine mesh can be seen in Figure 13.

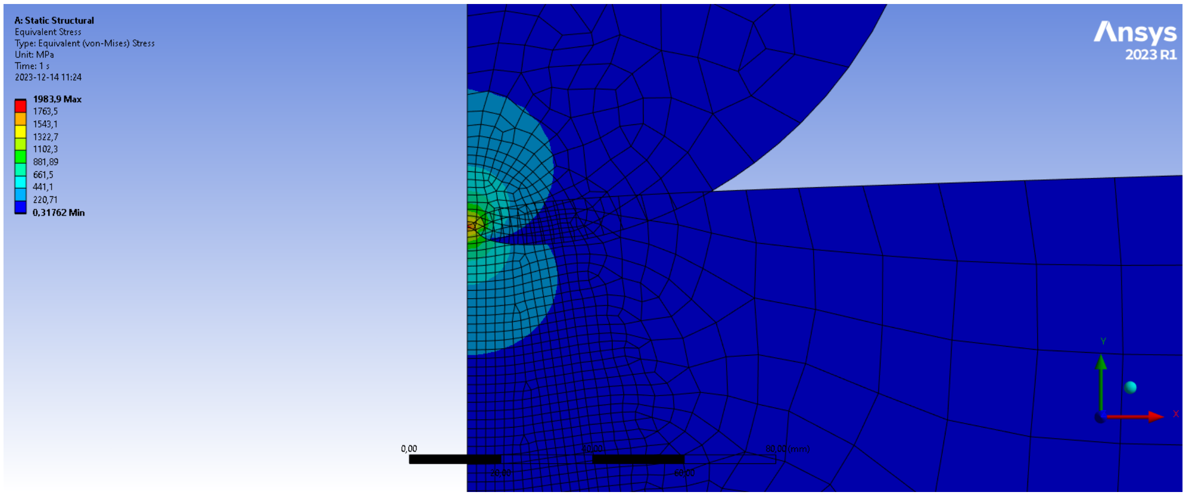


Figure 13: Equivalent stress using fine meshing.

$$\sigma_{vm}^{Ansys} = 1983.9MPa$$
$$\sigma_{vm}^{Matlab} = 234.8MPa$$

We can see that the distributions of the stresses, trough out the geometry looks similar but the maximum stresses differ with a factor of 10. This can be due to a few different reasons, either the matlab code calculating stresses has an error or the Ansys implementation of the force can be wrong. There is also a possibility that the mesh used in Ansys models the force concentrations better than our mesh generated in the matlab code.

Appendix 1

Listing 1: MATLAB code Task 1

```
%% Assignment 3 MHA021
% Group 33
% Nils Helgesson & Gabriel Wendel
close all;
clear all;
clc;

%% Indata
E=210e9; %[Pa]
nu=0.3;
rho=8000; %[kg/m3]
g=9.82; %[m/s^2]
b=[0; -rho*g]; %load, self weight
p_max=200e6; %[Pa]
% p_max=0; %For control calc

%Dimensions of half beam
W=0.5;
H=0.3;
% W=10*H; %For control calc
t=0.05;

% Meshdata
mesh_a=40e-3;
nelH=20;
nelW=20;
nela=20;

%Gauss points
ngp=1;



%% Generate mesh
% INPUT: H      : domian height
%        W      : domain width
%        a      : fraction of the width along the top edge
%        nelH   : number of element edges in vertical direction
%        nelW   : number of element edges in horizontal direction
%        nela   : number of element edges along the distance a (nela<
    nelvW)
%
% OUTPUT:  xy   : matrix with 2 columns; ith row contains the (x,y)-
%                 -coordinate of the ith node (CALFEM name Coord)
%          Dof  : matrix (2 columns) vector with degrees of freedom in
%                 each node; (row i simply has the the two values 2*i
    -1
%                 and 2*i), since there are two dofs in each node, i.e
    .
%                 the degrees of freedom in node i, are 2*i-1 and 2*i
%          Edof : matrix with 9 columns; 1st column is the element
```

```
%                    number; columns 2-9 in row i, contains the degrees
%                    of freedom, in counter clock-wise order, for element
     i.
%  SEE THE CALFEM MANUAL, PAGES 6.2-2 TO 6.2-3, FOR A MORE DETAILED
%  DESCRIPTION OF THE ABOVE OUTPUT VARIABLES.
%           nodL   : list of nodes along left edge
%           nodR   : list of nodes along right edge
%           noda   : list of nodes along part "a" of upper edge



[xy,Dof,Edof,nodL,nodR,noda] = pmesh(H,W,mesh_a,nelH,nelW,nela);

nel=size(Edof,1);
nnodes=size(Dof,1);

%Process mesh data
ex=xy(:,1);
ey=xy(:,2);
Ex=zeros(nel,4);
Ey=zeros(nel,4);
for i=1:nel
    E1=find(ismember(Dof, Edof(i,2:3),'rows'));
    E2=find(ismember(Dof, Edof(i,4:5),'rows'));
    E3=find(ismember(Dof, Edof(i,6:7),'rows'));
    E4=find(ismember(Dof, Edof(i,8:9),'rows'));
    Ex(i,:)=[ex(E1) ex(E2) ex(E3) ex(E4)];
    Ey(i,:)=[ey(E1) ey(E2) ey(E3) ey(E4)];
end

eldraw2(Ex,Ey)
%% Constitutive matrix D
ptype=1; % plane stress
% Determine constitutive matrix using Hooke's generalized law
D=hooke(1,E,nu);

%% Assemble fb

%along r5
GP=0 %middle of interval
iW=2
Ne_xsi=[0.5 0 0.5 0; 0 0.5 0 0.5]; %middle of 1D element
fb=zeros(nnodes*2,1)
for i=1:length(noda)-1
    x1=ex(noda(i));
    x2=ex(noda(i+1));
    L_e=x2-x1;
    x_middle=(x1+x2)/2;
    ty=-p_max*sqrt(1-(x_middle/mesh_a)^2);
    fbe=Ne_xsi'*[0 ty]'*t*L_e;
    fb(Dof(noda(i),1))=fbe(1); %x dof for node 1
    fb(Dof(noda(i),2))=fbe(2); %y dof for node 1
    fb(Dof(noda(i+1),1))=fbe(1); %x dof for node 2
    fb(Dof(noda(i+1),2))=fbe(2); %y dof for node 2
end

```

```
%% Assemble K and fl
K=zeros(nnodes*2,nnodes*2);
fl=zeros(nnodes*2,1);
ep=[t ngp];
for i=1:nel
    [Ke fe]=plan4bilin(Ex(i,:),Ey(i,:), ep, D, b);

    [K, fl]=assem(Edof(i,:),K,Ke,fl,fe);
end

% Boundary conditions

bc=[Dof(nodR,1) zeros(length(nodR),1);
    Dof(nodR,2) zeros(length(nodR),1);
    Dof(nodL,1) zeros(length(nodR),1)];
[a , r]=solveq(K , fl+fb , bc ) ;



% Plots
ed = extract ( Edof , a ) ;
figure (1)
eldraw2( Ex , Ey , [1 4 0])
hold on
eldisp2( Ex , Ey , ed , [1 2 1] , 300) ; % scale factor =300
% eldisp2( Ex , Ey , ed , [1 2 1], 1) ; %scale factor =0
xlabel('x[m]')
ylabel('y[m]')
legend('Undeformed')
title('Deformation')

%% Calculate stresses

nel = size(Edof,1); % number of elements

ep = [ptype,t,ngp];

% Stress vector for element center
sigma = zeros(nel, 3);

for i=1:nel
    % # columns in es follows size of D (3x3) and # rows = n^2, n =
    % integration points
    [es,et,eci] = plani4s(Ex(i,:),Ey(i,:),ep,D,ed(i,:));
    % Store stresses for each element
    sigma(i,:) = es';
end

%% Effective stress (von Mises)

sigma_vm = zeros(1,nel);

for i=1:nel
    % calculate effective stress according to von Mises
    sigma_vm(i) = sqrt(sigma(i,1)^2 + sigma(i,2)^2 - sigma(i,1)*sigma(
       i,2) + 3*sigma(i,3)^2);
end
```

```matlab

figure(2)
xlabel('x-coordinate [m]');
ylabel('y-coordinate [m]');
title('Effective stress (von Mises) [Pa] for each element');

hold on

for i=1:nel
    fill(Ex(i,:),Ey(i,:),sigma_vm(i))
end
colorbar;
colormap('jet')

maxSigma=max(sigma_vm);
```

Listing 2: plan4bilin function

```matlab
  function [ Ke, fe ] = plan4bilin( ex, ey, ep, D, eq)
% [ Ke, fe] = plan4bilin( ex, ey, ep, D, eq)
%-------------------------------------------------------------
% PURPOSE
%  Compute the stiffness matrix and element external force vector
%  for a bilinear plane stress or plane strain element including
%  influence of out-of-plane stress (or strain)
%
% INPUT:  ex = [ x1 x2 x3 x4 ]          element nodal x-coordinates
%         ey = [ y1 y2 y3 y4 ]          element nodal y-coordinates
%
%         ep = [ t ngp]                 t: thickness
%                                       ngp: number of Gauss points in each
%                                            direction ( ksi and eta)
%                                            (ngp = 1 or 2 or 3)
%
%         D(3,3)                        constitutive matrix for 2D
%                                       elasticity (plane stress or plane
%                                       strain)
%
%         eq = [ bx;                    bx: body force x-dir
%                by ]                   by: body force y-dir
%
%--------------------------------------------------------------
% OUTPUT: Ke : element stiffness matrix (8 x 8)
%         fe : equivalent nodal forces (8 x 1)
%--------------------------------------------------------------
% Developed by Peter Moller Dept. Applied Mechanics, Chalmers
%
% MODIFIED for VSM167 FEM BASICS by Dimosthenis Floros 20151201
%
% MODIFIED for VSM167 FEM BASICS by Martin Fagerstrom 20211201
%
%--------------------------------------------------------------
%

t     = ep(1);
```

```
ngp    = ep(2)^2; % Total gauss points = ( NoGaussPoits per direction )
    ^2

%  Initialize Ke and fe with zeros for all of their elements

Ke = zeros(8,8);
fe = zeros(8,1);

%  Tolerance for the Jacobian determinant

minDetJ = 1.e-16;

% Determine constitutive matrix D for plane strain or plane stress

%  Set the appropriate Gauss integration scheme for 1, 2 or 3 Gauss
    points
%  in each direction (ksi, eta). (or else 1, 4 or 9 Gauss points in
    total)

if ngp == 1 % 1x 1 integration
    GP=0;
    iW=2;

 intWeight   = [iW iW];
 GaussPoints = [GP GP];

elseif ngp == 4 % 2 x 2 integration

    GP1=0.5773502691896257;
    GP2=-GP1;
    iW=1;

 intWeight = [iW iW;
     iW iW;
     iW iW;
     iW iW];


 GaussPoints = [GP2 GP2;
     GP1 GP2;
     GP2 GP1;
     GP1 GP1];


elseif ngp == 9 % 3 x 3 integration
    GP1=0.7745966692414834;
    GP2=0

 intWeight = [iW1 iW1;
     iW2 iW1;
     iW1 iW1;
     iW1 iW2;
     iW2 iW2;
     iW1 iW2;
     iW1 iW1;
     iW2 iW1;
```

```
     iW1 iW1];



 GaussPoints =  [-GP1 -GP1 ;
     GP2 -GP1;
     GP1 -GP1;
     -GP1 GP2;
     GP2 GP2;
     GP1 GP2;
     -GP1 GP1;
     GP2 GP1;
     GP1 GP1];


else

 error('Only 1,2 or 3 Gauss Points in each direction apply')

end

%  Loop over all integration points to compute Ke and fe

for gpIndex = 1:ngp

 xsi       = GaussPoints(gpIndex,1);
 weightXsi = intWeight(gpIndex, 1);
 eta       = GaussPoints(gpIndex,2);
 weightEta = intWeight(gpIndex, 2);

% Compute the element shape functions Ne (use xsi and eta from above)
N1e= (xsi-1)*(eta-1)/4;
N2e= -(xsi+1)*(eta-1)/4;
N3e= (xsi+1)*(eta+1)/4;
N4e= -(xsi-1)*(eta+1)/4;

 Ne = [N1e 0 N2e 0 N3e 0 N4e 0;
        0 N1e 0 N2e 0 N3e 0 N4e ];

% Compute derivatives (with respect to xsi and eta) of the
% shape functions at coordinate (xsi,eta). Since the element is
% isoparametric, these are also the derivatives of the basis functions.

dNe_dxsi=[(eta-1)/4, -(eta-1)/4, (eta+1)/4, -(eta+1)/4];
dNe_dxeta=[(xsi-1)/4, -(xsi+1)/4, (xsi+1)/4, -(xsi-1)/4];

dx_dxsi=dNe_dxsi*ex';
dx_deta=dNe_dxeta*ex';
dy_dxsi=dNe_dxsi*ey';
dy_deta=dNe_dxeta*ey';

%  Use shape function derivatives and element vertex coordinates
%  to establish the Jacobian matrix.

 J=[dx_dxsi dx_deta; dy_dxsi dy_deta];

```

```
%  Compute the determinant of the Jacobian and check that it is OK

 detJ = det(J);

 if ( detJ < minDetJ )

  fprintf( 1, 'Bad element geometry in function plan4bilin: detJ =
      %0.5g\n', detJ );
  return;

 end

% Determinant seems OK - invert the transpose of the Jacobian

 J_T_inv=inv(J');

%   Compute derivatives with respect to x and y, of all basis
    functions,

 dNe_dx_dy=J_T_inv*[dNe_dxsi; dNe_dxeta];

% Use the derivatives of the shape functions to compute the element
% B-matrix, Be

 Be = zeros(3,8);
 Be(1, 1:2:end)=dNe_dx_dy(1,:); %Insert every second column
 Be(2, 2:2:end)=dNe_dx_dy(2,:);
 Be(3, 1:2:end)=dNe_dx_dy(2,:); %Insert every column
 Be(3, 2:2:end)=dNe_dx_dy(1,:);



% Compute the contribution to element stiffness matrix and volume load
    vector
% from current Gauss point
% (check for plane strain or plane stress again!)

 Ke = Ke + Be'*D*t*Be*detJ*weightXsi*weightEta;

 fe = fe + Ne'*eq*t*detJ*weightXsi*weightEta;

end

end

%--------------------------end--------------------------------
```