MSc in Mechanical Engineering, 3rd Semester

Assignment 2

Finite Element Method

Instructors:
Lili Zhang
Mikkel Melters Pedersen

Teaching Assistants:
Lasse Buus Jacobsen
Mathias Sätherström Boye

Student:
Kristofer Bjarmi Schram
202102204

14. October 2022

Rectangular isoparametric element

A rectangular isoparametric with 6 nodes is shown in physical and reference space in Figure 1. It has quadratic top and bottom sides, but the vertical sides are linear.

The element is fully constrained in nodes 5 and 6, and loaded by force two force couples in the remaining nodes. The loading corresponds to pure bending. The load magnitude is

$$|F| = 100\,\mathrm{N}$$

The element is in a state of plane stress and exhibits linearly elastic and isotropic material behavior with properties

$$t = 1\,\mathrm{mm}, \quad E = 1500\,\mathrm{MPa}, \quad \nu = 0.3$$

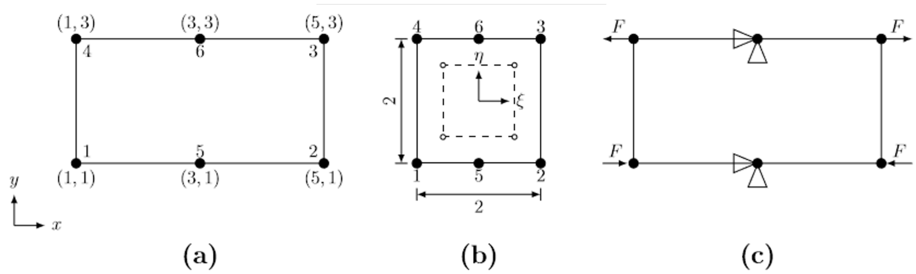

Figure 1: (a) Physical element, (b) reference element & (c) boundary conditions.

1. (Code, lines 20-40) Using $[\boldsymbol{X}] = \begin{bmatrix} 1 & \xi & \eta & \xi\eta & \xi^2 & \xi^2\eta \end{bmatrix}$ as a recipe for the shape function values N_{unit}, i.e., the matrix that multiplied with the generalized coordinates a_i gives the nodal coordinates x_i. The first column is a vector filled with 'ones'. The second and third columns are the reference coordinates ξ and η of each node. See figure 1a above. The last the three columns are calculated according to $[\boldsymbol{X}]$. As an example, the first row would be $\begin{bmatrix} 1 & \xi_1 & \eta_1 & \xi_1\eta_1 & \xi_1^2 & \xi_1^2\eta_1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}$. The full matrix is now:

$$N_{unit} = \begin{bmatrix} 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

The shape function matrix is now, $[N] = [\boldsymbol{X}]N_{unit}^{-1}$, (Cook, 6.1-2).

$$N = \begin{bmatrix} 1 & \xi & \eta & \xi\eta & \xi^2 & \xi^2\eta \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \end{bmatrix},$$

Where,

$$N_1 = -\frac{\eta\xi^2}{4} + \frac{\eta\xi}{4} + \frac{\xi^2}{4} - \frac{\xi}{4} \qquad N_2 = -\frac{\eta\xi^2}{4} - \frac{\eta\xi}{4} + \frac{\xi^2}{4} + \frac{\xi}{4} \qquad N_3 = \frac{\eta\xi^2}{4} + \frac{\eta\xi}{4} + \frac{\xi^2}{4} + \frac{\xi}{4}$$
$$N_4 = \frac{\eta\xi^2}{4} - \frac{\eta\xi}{4} + \frac{\xi^2}{4} - \frac{\xi}{4} \qquad N_5 = \frac{\eta\xi^2}{2} - \frac{\eta}{2} - \frac{\xi^2}{2} + \frac{1}{2} \qquad N_6 = -\frac{\eta\xi^2}{2} + \frac{\eta}{2} - \frac{\xi^2}{2} + \frac{1}{2}$$

Plotting the 1st and 5th shape function yields:

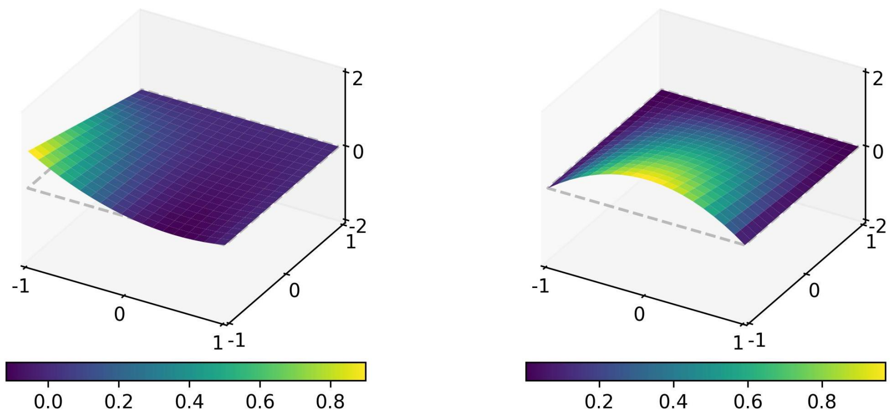

Figure 2: The 1st (left) and 5th (right) shape functions.

This $[X]$ vector simply reflects the fact that two sides of the element are linear and the other quadratic. These are sometimes called transition elements and are used to connect cubic elements to quadratic. For a more common quadratic element, the vector would be $[X] = \begin{bmatrix} 1 & \xi & \eta & \xi\eta & \xi^2 & \xi^2\eta & \xi^2\eta^2 & \xi\eta^2 & \eta^2 \end{bmatrix}$.

When choosing an x-vextor, the values must be balanced in both direction, which this clear ins't as there is no η^2 present.

2. (Code, lines 99-103) The Jacobian matrix is calculated as (see *Cook, 6.2-5*),

$$[\mathbf{J}] = \begin{bmatrix} \sum N_{i,\xi} x_i & \sum N_{i,\xi} y_i \\ \sum N_{i,\eta} x_i & \sum N_{i,\eta} y_i \end{bmatrix},$$

and can be expanded, in mm, to,

$$[\mathbf{J}] = \begin{bmatrix} N_{1,\xi} & \cdots & N_{6,\xi} \\ N_{1,\eta} & \cdots & N_{6,\eta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \\ x_5 & y_5 \\ x_6 & y_6 \end{bmatrix} = \begin{bmatrix} N_{1,\xi} & \cdots & N_{6,\xi} \\ N_{1,\eta} & \cdots & N_{6,\eta} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 5 & 1 \\ 5 & 3 \\ 1 & 3 \\ 3 & 1 \\ 3 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}. \tag{1}$$

The partial derivative in (1) of the shape functions can be obtain by utilizing the `jacobian` function in most CAS-software. This creates a new matrix where the first row is the partial derivative of $[N]$ w.r.t. ξ, and the second row $[N]$ w.r.t. η.

The determinant of the Jacobian matrix is now,

$$J = \det[\mathbf{J}] = \mathbf{J}_{11}\mathbf{J}_{22} - \mathbf{J}_{21}\mathbf{J}_{12} = 2.$$

The Jacobian matrix describes the change in size of the element, due to mapping from physical system, (x, y) to the refrence system (ξ, η). The individual values are the relative changes from a general coordinate in the physical system to a coordinate in the reference system, see eq. (2) below.

$$[\mathbf{J}] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{2}$$

With this, the changing area can now be described with the determinant, i.e.,

$$dxdy = \det[\mathbf{J}]d\xi d\eta = Jd\xi d\eta \tag{3}$$

3. (Code, lines 109-126) By utilizing the equations 6.2-9, 6.2-10 and 6.2-11 in *Cook*, the strain displacement matrix $[B]$ in the reference space, for the 6 node element, can be derived as,

$$[B] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & 0 & 0 \\ \Gamma_{21} & \Gamma_{22} & 0 & 0 \\ 0 & 0 & \Gamma_{11} & \Gamma_{12} \\ 0 & 0 & \Gamma_{21} & \Gamma_{22} \end{bmatrix} \begin{bmatrix} N_{1,\xi} & 0 & \cdots & N_{6,\xi} & 0 \\ N_{1,\xi} & 0 & \cdots & N_{6,\xi} & 0 \\ 0 & N_{1,\xi} & \cdots & 0 & N_{6,\xi} \\ 0 & N_{1,\xi} & \cdots & 0 & N_{6,\eta} \end{bmatrix} \tag{4}$$

$$= \begin{bmatrix} N_{1,\xi} & 0 & N_{2,\xi} & 0 & \cdots & N_{6,\xi} & 0 \\ 0 & N_{1,\eta} & 0 & N_{2,\eta} & \cdots & 0 & N_{6,\eta} \\ N_{1,\eta} & N_{1,\xi} & N_{2,\eta} & N_{2,\xi} & \cdots & N_{6,\eta} & N_{6,\xi} \end{bmatrix} \tag{5}$$

Where the Γ-matrix is the inverse of the Jacobian and describes the mapping from reference space to physical space, instead of the other way around. For (5) the values are,

$$N_{1,\xi} = -\frac{\eta\xi}{4} + \frac{\eta}{8} + \frac{\xi}{4} - \frac{1}{8} \qquad N_{1,\eta} = -\frac{\xi^2}{4} + \frac{\xi}{4}$$

$$N_{2,\xi} = -\frac{\eta\xi}{4} - \frac{\eta}{8} + \frac{\xi}{4} + \frac{1}{8} \qquad N_{2,\eta} = -\frac{\xi^2}{4} - \frac{\xi}{4}$$

$$N_{3,\xi} = \frac{\eta\xi}{4} + \frac{\eta}{8} + \frac{\xi}{4} + \frac{1}{8} \qquad N_{3,\eta} = \frac{\xi^2}{4} + \frac{\xi}{4}$$

$$N_{4,\xi} = \frac{\eta\xi}{4} - \frac{\eta}{8} + \frac{\xi}{4} - \frac{1}{8} \qquad N_{4,\eta} = \frac{\xi^2}{4} - \frac{\xi}{4}$$

$$N_{5,\xi} = \frac{\eta\xi}{2} - \frac{\xi}{2} \qquad N_{5,\eta} = \frac{\xi^2}{2} - \frac{1}{2}$$

$$N_{6,\xi} = -\frac{\eta\xi}{2} - \frac{\xi}{2} \qquad N_{6,\eta} = \frac{1}{2} - \frac{\xi^2}{2}$$

4. (Code, lines 130-147) Stiffness matrix will is calculated with numerical integration in the form of Gaussian quadrature. Here the double integral, one over each physical coordinate (x, y) is calculated as a set of sums. Thus the integral over the element area in the reference system, with a constant thickness t is,

$$[k] = \int_{-1}^{1}\int_{-1}^{1} [B]^T[E][B]tJd\xi d\eta \tag{6}$$

and becomes through summation,

$$[k] = \sum_{i=1}^{3}\sum_{j=1}^{2}[B(\xi_i,\eta_j)]^T[E][B(\xi_i,\eta_j)]tJ(\xi_i,\eta_j)W_iW_j \tag{7}$$

This is done in accordance to eq. 43-44 in the week 5 lecture notes.

Here there are 6 integration points, one for each node. The first summation takes care of the integration points in the ξ-direction, which are 3, and the second takes care of the 2 in the η-direction. Furthermore, W_i and W_j are Gaussian quadrature weight values from their corresponding weight vectors. The weight vectors and evaluation points for ξ_i and η_j are found in Table 6.3-1 in *Cook*. Now from the table,

$$[W_i] = [5/9, 8/9, 5/9] \qquad [\xi_i] = [\sqrt(0.6), 0, -\sqrt(0.6)] \tag{8}$$

$$[W_j] = [1, 1] \qquad [\eta_j] = [1/\sqrt(3), -1/\sqrt(3)] \tag{9}$$

The material behavior matrix, $[E]$, for plane stress is

$$[E] = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{10}$$

The numerical integration resulting in the stiffness matrix, is computed using the dummy code displayed in Table 6.3-2 in *Cook*. Here the final output is rounded and converted to N/mm to fit on the page,

$$[k] = \begin{bmatrix}
795 & 268 & 53 & 7 & 84 & 89 & 167 & 21 & -656 & -27 & -443 & -357 \\
268 & 664 & -7 & -78 & 89 & 126 & -21 & -327 & 27 & -37 & -357 & -348 \\
53 & -7 & 795 & -268 & 167 & -21 & 84 & -89 & -656 & 27 & -443 & 357 \\
7 & -78 & -268 & 664 & 21 & -327 & -89 & 126 & -27 & -37 & 357 & -348 \\
84 & 89 & 167 & 21 & 795 & 268 & 53 & 7 & -443 & -357 & -656 & -27 \\
89 & 126 & -21 & -327 & 268 & 664 & -7 & -78 & -357 & -348 & 27 & -37 \\
167 & -21 & 84 & -89 & 53 & -7 & 795 & -268 & -443 & 357 & -656 & 27 \\
21 & -327 & -89 & 126 & 7 & -78 & -268 & 664 & 357 & -348 & -27 & -37 \\
-656 & 27 & -656 & -27 & -443 & -357 & -443 & 357 & 2081 & 0 & 117 & 0 \\
-27 & -37 & 27 & -37 & -357 & -348 & 357 & -348 & 0 & 2271 & 0 & -1502 \\
-443 & -357 & -443 & 357 & -656 & 27 & -656 & -27 & 117 & 0 & 2081 & 0 \\
-357 & -348 & 357 & -348 & -27 & -37 & 27 & -37 & 0 & -1502 & 0 & 2271
\end{bmatrix} \tag{11}$$

The solution should be exact with two integration point over the linear range and 3 over the quadratic.

5. (Code, lines 151-162) The formula for calculating the force is still

$$\{r\} = [k]\{d\} \tag{12}$$

and can be used to calculate the unknown displacements.

First the boundary condition are such that u_5, v_5, u_6, $v_6 = 0$, giving the displacement vector

$$\{d\} = \begin{bmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & u_4 & v_4 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{13}$$

Furthermore, the load vector is such that the known values correlate to the unknown displacements, resulting in

$$\{r\} = \begin{bmatrix} F & 0 & -F & 0 & F & 0 & -F & 0 & p_{x_5} & p_{y_5} & p_{x_6} & p_{y_6} \end{bmatrix}^T \tag{14}$$

where p are the unknown loads at nodes 5 and 6.

Now, by splitting the load and displacement vectors, and stiffness matrix into groups of known (subscript c) and unknown values (subscript x), the eq in (12) can be solved as a set of equations. First,

$$\{d\} = \begin{bmatrix} d_x \\ d_c \end{bmatrix} \Rightarrow \{d_x\} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix}, \quad \{d_c\} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{15}$$

$$\{r\} = \begin{bmatrix} r_c \\ r_x \end{bmatrix} \Rightarrow \{r_c\} = \begin{bmatrix} F \\ 0 \\ -F \\ 0 \\ F \\ 0 \\ -F \\ 0 \end{bmatrix} = \begin{bmatrix} 100 \\ 0 \\ -100 \\ 0 \\ 100 \\ 0 \\ -100 \\ 0 \end{bmatrix}, \quad \{r_x\} = \begin{bmatrix} p_{x_5} \\ p_{y_5} \\ p_{x_6} \\ p_{y_6} \end{bmatrix} \tag{16}$$

and the stiffness matrix becomes,

$$[k] = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \tag{17}$$

where,

k_{11}:	Corresponds to unknown displ. and known loads,	i.e., k(1:8, 1:8)
k_{12}:	———||——— known displ. and known loads	i.e., k(9:12, 1:8)
k_{21}:	———||——— unknown displ. and unknown loads	i.e., k(1:8, 9:12)
k_{22}:	———||——— known displ. and unknonwn loads	i.e., k(9:12, 9:12)

Finally the set of equations are

$$[k_{11}]\{d_x\} + [k_{12}]\{d_c\} = \{r_c\} \tag{18}$$
$$[k_{21}]\{d_x\} + [k_{22}]\{d_c\} = \{r_x\} \tag{19}$$

solving the first equation for $\{d_x\}$ and substituting $\{d_c\} = \mathbf{0}$ gives the unknown displacements,

$$[k_{11}]\{d_x\} = \{r_c\} \quad \Rightarrow \quad \{d_x\} = [k_{11}]^{-1}\{r_c\} = \begin{bmatrix} 0.364 \\ -0.364 \\ -0.364 \\ -0.364 \\ 0.364 \\ -0.364 \\ -0.364 \\ -0.364 \end{bmatrix} \text{[mm]} \tag{20}$$

6. (Code, lines 165-175) For the 2D case in reference space, the strain displacement vector $\{\varepsilon\}$ consists of three elements, strain in x-direction, strain in y-direction, and the combined shear strain. In pure bending, due to equal and opposing forces in the x direction, the second and third terms are zero.

   The strain displacement is calculated as $\{\varepsilon\} = [B]\{d\}$, and in this case gives,

$$\varepsilon = [B(\xi,\eta)]\{d\} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} 0.182\eta \\ 0 \\ 0 \end{bmatrix} \tag{21}$$

   and shows that the strain along x is a function of the height η. This is to be expected as engineering strain is calculated as $\Delta l/l$, the change in length divided by the initial length. As an example, looking at the top edge where $\eta = 1$, and the initial length is $l = 4$mm,

$$\varepsilon_x = \frac{\Delta l}{l} = \frac{u_3 - u_4}{4} = \frac{0.364 - (-0.364)}{4} = 0.182 \tag{22}$$

   which is the same as with eq. (21). Furthermore, the second two terms in (21) are equal to zero, supporting the initial statements about strain in pure bending.

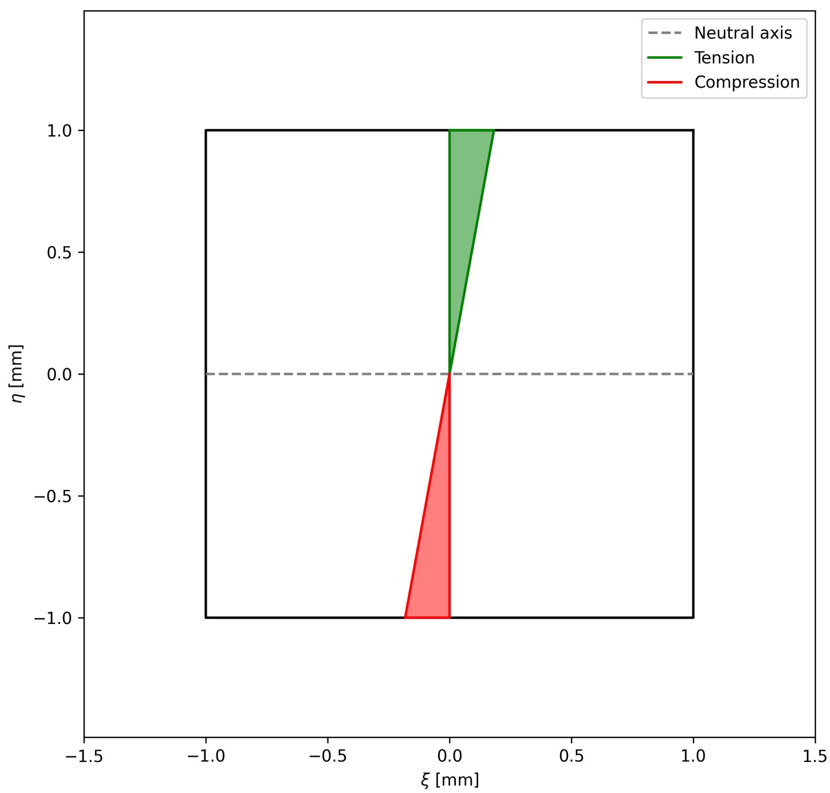

Figure 3

7. (Code, lines 198-211) To plot the deformed position of the element, eq. 6.2-1 in *Cook* is used. It state that,

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = [N]\{c\} \quad \text{and} \quad \begin{Bmatrix} u \\ v \end{Bmatrix} = [N]\{d\} \tag{23}$$

where $\{c\}$ contains the undeformed nodal coordinates in the physical space. Combining the two equations in (23) and applying the disruptive law for matrix manipulation, results in a equation that calculates the deformed coordinates in physical space as a function of ξ and η.

$$\begin{Bmatrix} x+u \\ y+v \end{Bmatrix} = [N]\{c\} + [N]\{d\} = [N]\{c+d\} \tag{24}$$

The undeformed and deformed states can be seen in figure (4) below.

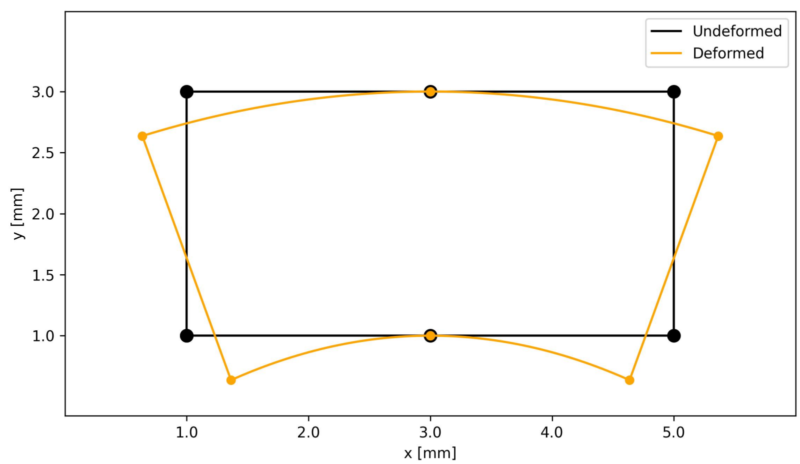

Figure 4

(Code, lines 231-239) The magnitude is calculated as

$$|u| = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2} \tag{25}$$

where i is undeformed and f is deformed. Figure (5) of the magnitude can be seen on the next page.

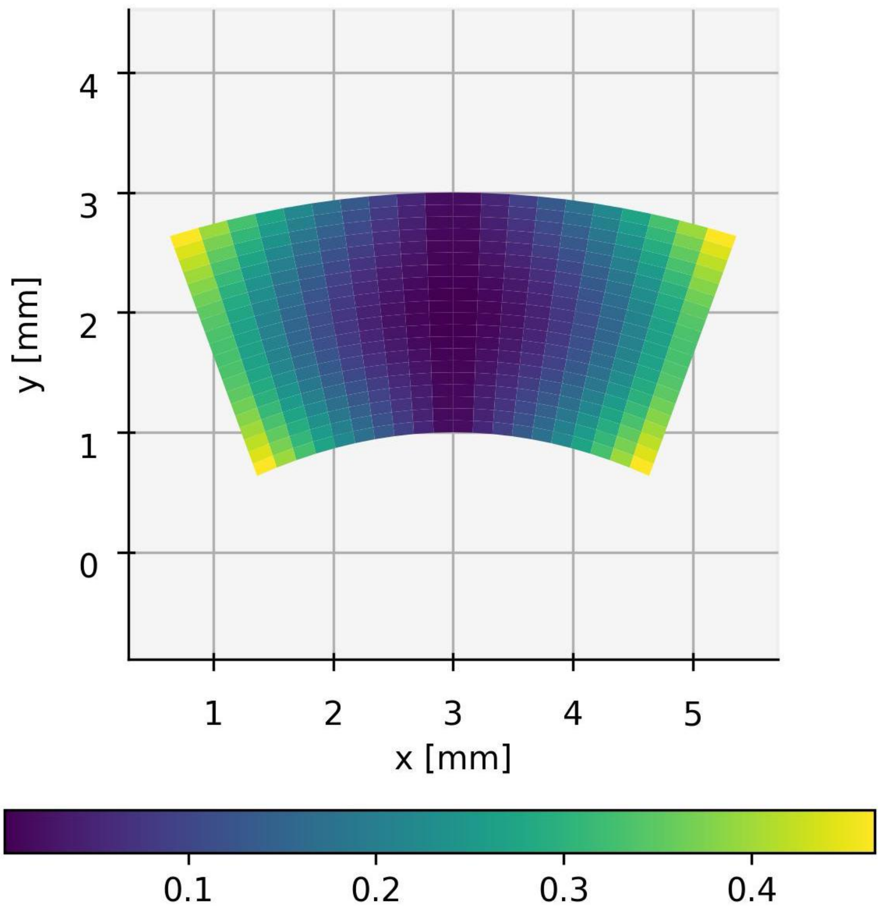

Figure 5

Appendix

Python Code

```
import math
import copy
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator
from mpl_toolkits.mplot3d import Axes3D
import sympy as sym

# Defining known variables
F = 100 #N
t = 1 #mm
E = 1500 #MPa - N/mm2
nu = 0.3
xi, eta = sym.symbols('xi eta')
X = sym.Matrix(1,6, [1, xi, eta, xi*eta, xi**2, xi**2*eta])

######## PROBLEM 1 ########

#xi and eta derived from refrence figure
#              Node =  1,  2, 3,  4,  5, 6
xi_vec = sym.Matrix([ -1,  1, 1, -1,  0, 0])#*1e-3
eta_vec = sym.Matrix([-1, -1, 1,  1, -1, 1])#*1e-3

Ngrid = sym.zeros(6,6)
for i in range(6):
    Ngrid[i, :] = sym.Matrix(1, 6,
        [1, xi_vec[i], eta_vec[i], xi_vec[i]*eta_vec[i],
        xi_vec[i]**2, xi_vec[i]**2*eta_vec[i]])

N = X*Ngrid.inv()  # a simplified 1x6 Shape function vector


xi_m = np.linspace(-1.0, 1.0, num=20)
eta_m = np.linspace(-1.0, 1.0, num=20)
xi_m, eta_m = np.meshgrid(xi_m, eta_m)

# coverting N1 and N5 to a numpy functions of xi and eta
N1 = sym.lambdify((xi, eta), N[0], 'numpy')
N5 = sym.lambdify((xi, eta), N[4], 'numpy')

#PLOTTING#
xh = [1, 1, -1, -1, 1]
yh = [1, -1, -1, 1, 1]
zh = [0, 0, 0, 0, 0]
```

```
fig = plt.figure(figsize=(9,5))
#First plot
ax = fig.add_subplot(1, 2, 1, projection= '3d')
surf = ax.plot_surface(xi_m, eta_m, N1(xi_m,eta_m), cmap=cm.viridis,
                       linewidth=0)
ax.plot(xh, yh, zh, '--', color='grey', alpha=0.5)
ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])
ax.set_zlim([-2, 2])
ax.set_xlabel(r'$\xi$')
ax.set_ylabel(r'$\eta$')
#ax.grid(False)
ax.zaxis.set_major_locator(LinearLocator(5))
ax.set_xticks([-1,0,1], ['-1', '0', '1'])
ax.set_yticks([-1,0,1], ['-1', '0', '1'])
ax.set_zticks([-2,0,2], ['-2', '0', '2'])
ax.xaxis.set_tick_params(pad=-4, labelsize=10)
ax.yaxis.set_tick_params(pad=-4, labelsize=10)
ax.zaxis.set_tick_params(pad=-4, labelsize=10)
ax.set_proj_type('ortho')
ax.grid(False)
fig.colorbar(surf, orientation='horizontal', anchor =(0.6, 2.2), shrink=0.83)
ax.view_init(30,-60)

#Second plot
ax = fig.add_subplot(1, 2, 2, projection= '3d')
surf = ax.plot_surface(xi_m, eta_m, N5(xi_m,eta_m), cmap=cm.viridis,
                       linewidth=0)
ax.plot(xh, yh, zh, '--', color='grey', alpha=0.5)
ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])
ax.set_zlim([-2, 2])
ax.set_xlabel(r'$\xi$')
ax.set_ylabel(r'$\eta$')
#ax.grid(False)
ax.zaxis.set_major_locator(LinearLocator(5))
ax.set_xticks([-1,0,1], ['-1', '0', '1'])
ax.set_yticks([-1,0,1], ['-1', '0', '1'])
ax.set_zticks([-2,0,2], ['-2', '0', '2'])
ax.xaxis.set_tick_params(pad=-4, labelsize=10)
ax.yaxis.set_tick_params(pad=-4, labelsize=10)
ax.zaxis.set_tick_params(pad=-4, labelsize=10)
ax.set_proj_type('ortho')
ax.grid(False)
fig.colorbar(surf, orientation='horizontal', anchor =(0.6, 2.2), shrink=0.83)
ax.view_init(30, -60)

plt.savefig('N1N5.jpg', dpi=300)
plt.show()
```

```python

######## PROBLEM 2 ########

xy = np.array([[1, 5, 5, 1, 3, 3],[1, 1, 3, 3, 1, 3]]).T

N_diff = N.jacobian([xi, eta]).T
Jac = N_diff * xy
J = sym.det(Jac)

######## PROBLEM 3 ########

# form 6.2-9
# Transformation matrix
T = sym.Matrix([[1,0,0,0],[0,0,0,1],[0,1,1,0]])

# from 6.2-10
Gamma = Jac.inv()
# Expanded Gamma
Gamma_exp = sym.zeros(4,4)
Gamma_exp[0:2, 0:2] = Gamma
Gamma_exp[2:4, 2:4] = Gamma

# Expanded derivative of shape-function-matrix
Nshape = sym.Matrix([
    [N[0], 0,  N[1], 0,  N[2], 0,  N[3], 0,  N[4], 0,  N[5], 0],
    [ 0,  N[0], 0,  N[1], 0,  N[2], 0,  N[3], 0,  N[4], 0,  N[5]]])
# from 6.2-11
dN_exp = sym.Matrix([[Nshape[0,:].jacobian([xi, eta]).T],
                     [Nshape[1,:].jacobian([xi,eta]).T]])

B = T * Gamma_exp * dN_exp

######## PROBLEM 4 ########

# Constitutive matrix E for plane stress
Emat = E/(1-nu**2)*np.array([[1, nu, 0],[nu, 1, 0],[0, 0, (1-nu)/2]])
intfunc = B.T*Emat*B*t*J

# Three points in the i-direction and 2 in j-direction
# Total of 6 integration points

xiset = [np.sqrt(0.6), 0, -np.sqrt(0.6)]
wi = [5/9, 8/9, 5/9]
etaset = [1/np.sqrt(3), -1/np.sqrt(3)]
wj = [1, 1]

KE = np.zeros([12,12])

for i, vali in enumerate(xiset):
    for j, valj in enumerate(etaset):
```

```python
        Kpart = intfunc.subs([(xi, vali), (eta, valj)]) * wi[i] * wj[j]
        KE = KE + Kpart

######## PROBLEM 5 ########

# Given that the boundary condition are u5,v5,u6,v6 = 0
u1, v1, u2, v2, u3, v3, u4, v4 = sym.symbols('u1 v1 u2 v2 u3 v3 u4 v4')
Rc = sym.Matrix([F, 0, -F, 0, F, 0, -F, 0])
Dx = (u1, v1, u2, v2, u3, v3, u4, v4)
KE11 = KE[0:8,0:8]
sol1, = sym.linsolve((KE11, Rc), Dx)
print('u1 = %.7f, v1 = %.7f, u2 = %.7f, v2 = %.7f' % sol1.args[0:4])
print('u3 = %.7f, v3 = %.7f, u4 = %.7f, v4 = %.7f' % sol1.args[4:8])

#np.set_printoptions(precision=8)
d = np.zeros([12,1])
d[0:8] = d[0:8] + np.asarray([sol1.args[0:8]], dtype=object).T

######## PROBLEM 6 ########

epsilon = B*d
epsilon
epsx = sym.lambdify((xi, eta), epsilon[0], 'numpy')
xh = [1, 1, -1, -1, 1]
yh = [1, -1, -1, 1, 1]

xstrain_top = [0, epsx(0,1), 0, 0]
ystrain_top = [1,1,0,1]
xstrain_bot = [0, 0, epsx(0,-1), 0]
ystrain_bot = [0,-1,-1,0]

fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(1, 1, 1)
ax.plot(xh, yh, color='black')
ax.plot([-1,1], [0,0], '--', color='grey', label='Neutral axis')
ax.plot(xstrain_top, ystrain_top, color='green', label='Tension')
ax.fill(xstrain_top, ystrain_top, 'green', alpha=0.5)
ax.plot(xstrain_bot, ystrain_bot, color='red', label='Compression')
ax.fill(xstrain_bot, ystrain_bot, 'red', alpha=0.5)
ax.axis('equal')
ax.set(xlim=[-1.5,1.5], ylim=[-1.5,1.5])
ax.set_xlabel(r'$\xi$ [mm]')
ax.set_ylabel(r'$\eta$ [mm]')
ax.legend()


plt.savefig('p6.jpg', dpi=300)
plt.show()

######## PROBLEM 7 ########
```

```python

# Creating undeformed points
undefx = np.array([1, 3, 5, 5, 3, 1, 1])
undefy = np.array([1, 1, 1, 3, 3, 3, 1])

# Computing points after deformation
defx = xy[:,0] + np.array([d[0], d[2], d[4], d[6], d[8], d[10]]).T
defy = xy[:,1] + np.array([d[1], d[3], d[5], d[7], d[9], d[11]]).T

# Creating functions that give the x y at any given xi ant eta.
xf = N*sym.Matrix(defx.T)
yf = N*sym.Matrix(defy.T)
xf = sym.lambdify((xi, eta), xf, 'numpy')
yf = sym.lambdify((xi, eta), yf, 'numpy')

span = np.linspace(-1.0, 1.0, 1000)

# frist plot
fig = plt.figure(figsize=(9,5))
ax = fig.add_subplot(1, 1, 1)
ax.plot(undefx, undefy, color='black', label='Undeformed')
ax.scatter(undefx, undefy, marker='o', color='black', s=70)
ax.plot(xf(span,1)[0][0], yf(span, 1)[0][0], color='orange', label='Deformed')
ax.plot(xf(span,-1)[0][0], yf(span, -1)[0][0], color='orange')
ax.plot(xf(-1,span)[0][0], yf(-1, span)[0][0], color='orange')
ax.plot(xf(1,span)[0][0], yf(1, span)[0][0], color='orange')
ax.scatter(defx, defy, marker='o' , color='orange', s=30, zorder=2)
ax.axis('equal')
ax.set(xlabel='x [mm]', ylabel='y [mm]', xlim=[0,6], ylim=[0,4])
ax.legend()
#ax.set()

plt.savefig('p7.jpg', dpi=300)
plt.show()

xi_m2 = np.linspace(-1.0, 1.0, num=21)
eta_m2 = np.linspace(-1.0, 1.0, num=21)
xi_m2, eta_m2 = np.meshgrid(xi_m2, eta_m2)
disx = xf(xi_m2, eta_m2)[0][0]
disy = yf(xi_m2, eta_m2)[0][0]
x_m = np.linspace(1, 5, num=21)
y_m = np.linspace(1, 3, num=21)
x_m, y_m = np.meshgrid(x_m, y_m)
mag = np.sqrt((disx-x_m)**2+(disy-y_m)**2)

# second plot
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(1, 1, 1, projection='3d')
surf = ax.plot_surface(disx,disy, mag, cmap=cm.viridis,
                       linewidth=0)
```

```


ax.axis('equal')
ax.set_xlabel('x [mm]')
ax.set_ylabel('y [mm]')
ax.set_zticks([], [])
ax.xaxis.set_tick_params(pad=2, labelsize=10)
ax.yaxis.set_tick_params(pad=2, labelsize=10)
ax.zaxis.set_tick_params(pad=2, labelsize=10)
ax.set_proj_type('ortho')
#ax.grid(False)
cbar = fig.colorbar(surf, orientation='horizontal',
            anchor =(0.5, 2.5), shrink=0.6, )
ax.view_init(90, -90)
plt.savefig('p72.jpg',dpi=300)
plt.show()
```