Final Project Report

Finite Elements option

EMA 471 Intermediate Problem Solving for Engineers

Spring 2016
Engineering Mechanics Department
University of Wisconsin, Madison

Instructor: Professor Robert J. Witt

By

Nasser M. Abbasi

December 30, 2019

Contents

List of Tables

List of Figures

Chapter 1

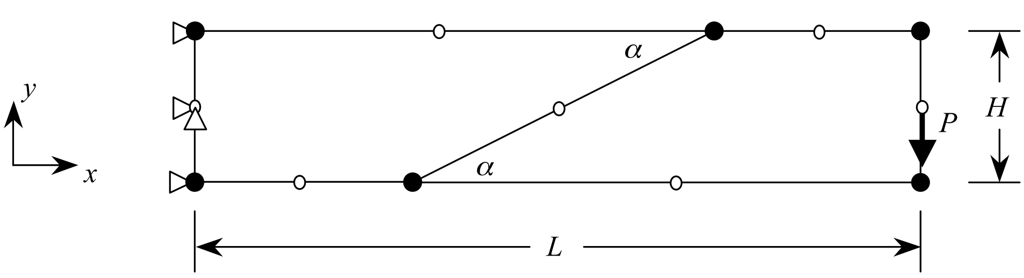
Description of the problem, geometry and element description

1.1 Problem statement

For completion of the report, the problem statement is given below taken from the project handout.

Assembly of Planar FE Problem

In EMA 405, we typically illustrate the element assembly process with 1D elements (bars or beams). The purpose of this project is to give you an idea of what's involved in assembling a 2D FE continuum problem (plane stress) from first principles. One of the problems we typically investigate is a two-Q8-element mesh distortion problem, where the geometry looks like this:



For the specific illustration in EMA 405, the numbers are these: $E = 1 \times 10^4$, $\nu = 0.3$, $L = 10$, $H = 2$, and $P = 20$. The angle α is a measure of the mesh distortion (departures of angles between adjacent edges from 90°) and is a parameter to be varied. The expected vertical displacement at the node where force P is applied, based on beam theory, is 1.031. You are to assemble a global stiffness matrix from the two element stiffness matrices and solve for the deflections at the nodes. You can compare your results to a comparable ANSYS FE model.

The global stiffness matrix is assembled from element stiffness matrices and embodies internal strain energy. Briefly, it starts out from integrating volumetric strain energy over the volume of the element. If we had a simple, linear spring, initially undeformed, and stretched it a distance x, the energy stored in that spring would be

$$\text{Energy} = \frac{1}{2}x \cdot kx = \frac{1}{2}kx^2$$

In a continuum problem in 2D, we have three modes of deformation: normal strain in x, normal strain in y and shear strain (xy). Instead of displacement × force [energy], we use strain × stress [N/m^2 or N·m/m^3], the latter being energy density. The energy in the material is:

Figure 1.1: EMA project problem description

The problem described above was solved for the following values of the angle α (in degrees)

$$\boxed{\{0,15,30,45,50,55\}}$$

Where α is the distortion angle between the first and the second elements as shown in the above diagram. The method of finite elements was implemented in Matlab to solve for the displacements at the nodes. 4 Gaussian points were used for the integration step (this is also called the 2×2 integration rule). After the global stiffness matrix was assembled from the two elements stiffness matrices, the system equations given by $KD = F$ were solved using the direct linear system solver.

1.2 Geometry, nodes and element description

Two elements were used each having 8 nodes. 4 of these nodes are at the corners, and the other 4 are at the mid point of the element edges. The following diagram shows the global coordinates of the elements nodes. The origin of the global coordinate system is at the lower left corner as shown in the diagram below.

L is the overall length of the beam, which is 10 meters, and h is the height of the beam (which is 2 meters).

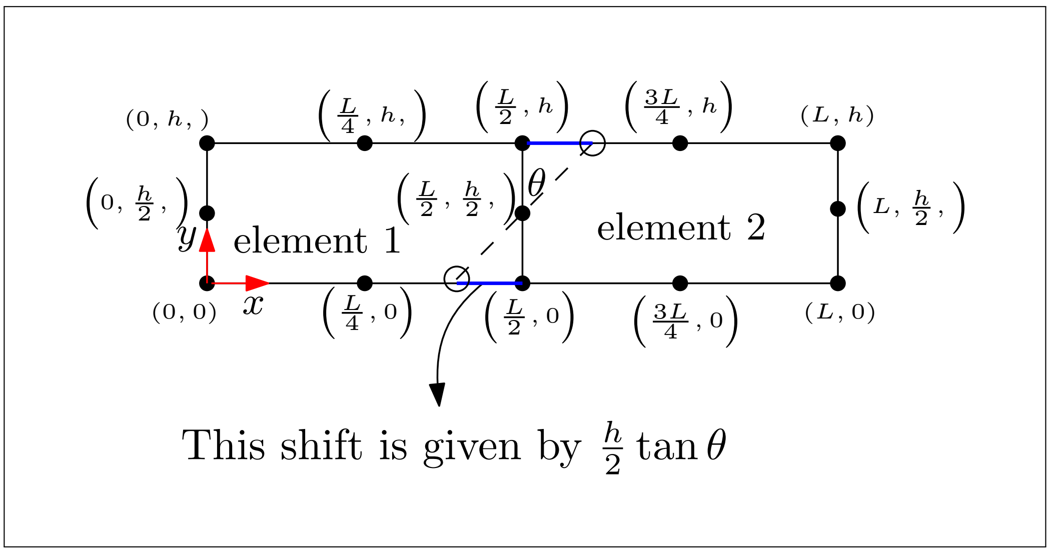


Figure 1.2: Global node coordinates using general coordinates, showing the distortion angle

The vertical and horizontal displacement of each node was solved for using the finite elements method for each of the different values of the angle α and the vertical displacement at the right bottom edge of the beam was compared to the expected theoretical value in order to see the effect of the element distortion on the accuracy of the finite element result using the element selected.

The idea is that a good finite element should produce the same displacement at its nodes regardless of how it was fitted to the physical region in place. This report was to determine if the element selected would still produce good results when deformed. The first step was to map each element local node number to a global node number. Local element numbers go from 1 to 8 since there are only 8 nodes per element, but the global node numbers enumerates over all the nodes in all the elements.

Local element node numbering is made in the standard anti clock wise direction by numbering the corner nodes first from 1 to 4, followed by numbering the middle nodes also in the anti clock wise sense from 5 to 8.

The following diagram shows the mapping between the local element node numbers and the global node numbers. The top diagram shows the global node numbers and the lower diagram shows each element local node numbers.

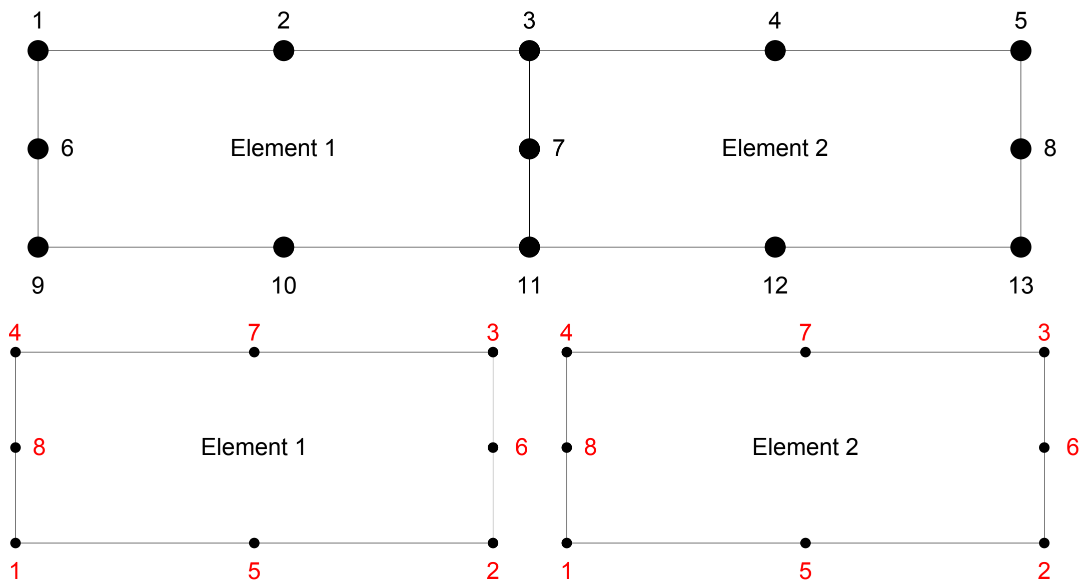


Figure 1.3: Global and element node numbering used for project EMA

Based on the the above, the table `elem_map_nodes` was constructed. This table gives the global node number (the entries inside the table) for an element number (the row of the table) and the node number within that specific element (the column in the table). For example, for element 1 with local node number 1 the table above shows that the global node number is 9.

element # \ element node #	1	2	3	4	5	6	7	8
1	9	11	3	1	10	7	2	6
2	11	13	5	3	12	8	4	7

Table 1.1: elem_map_nodes table. Mapping element node to global nodes

There is also a need to lookup the global coordinates $\{x_i, y_i\}$ given the global node number. This is needed when finding the Jacobian.

The following table called `global_coordinates_tbl` was constructed for this purpose. In this table, $H = 2, L = 10$ and $\Delta = \tan\theta\frac{H}{2}$ is the amount of shift in meters of the global node 3 and 11. These are the only two nodes which shift location when changing the angle α. When α is zero, there will be no distortion of the elements.

global node # \ global node coordinates	x_i	y_i
1	0	H
2	$\frac{L}{4}$	H
3	$\frac{L}{2}+\Delta$	H
4	$\frac{3}{4}L$	H
5	L	H
6	0	$\frac{H}{2}$
7	$\frac{L}{2}$	$\frac{H}{2}$
8	L	$\frac{H}{2}$
9	0	0
10	$\frac{L}{4}$	0
11	$\frac{L}{2}-\Delta$	0
12	$\frac{3}{4}L$	0
13	L	0

Table 1.2: global_coordinates_tbl. Mapping global node number to global coordinates

There are two degrees of freedom at each node. These are u, v, representing the horizontal and vertical displacement of a node. Hence there is a need for a lookup table called `elem_map_dofs` which gives the degree of freedom number of each element's local node. This table is used for assembling the global stiffness matrix.

Using the method in the project handout, the following table was generated.

element # \ element node #	node 1		node 2		node 3		node 4		node 5		node 6		node 7		no
1	17	18	21	22	5	6	1	2	19	20	13	14	3	4	11
2	21	22	25	26	9	10	5	6	23	24	15	16	7	8	13

Table 1.3: elem_map_dof table. Mapping local element DOF to global stiffness matrix locations

The following diagram, generated in the Matlab program, gives the degree of freedom number corresponding to each element node (u, v). These DOF numbers represent the position of the unknowns in the solution of the $KD = F$. This means that there are a total of 26 degrees of freedom initially. However, due to boundary conditions constraints, the total number of degrees of freedom reduces to 22.

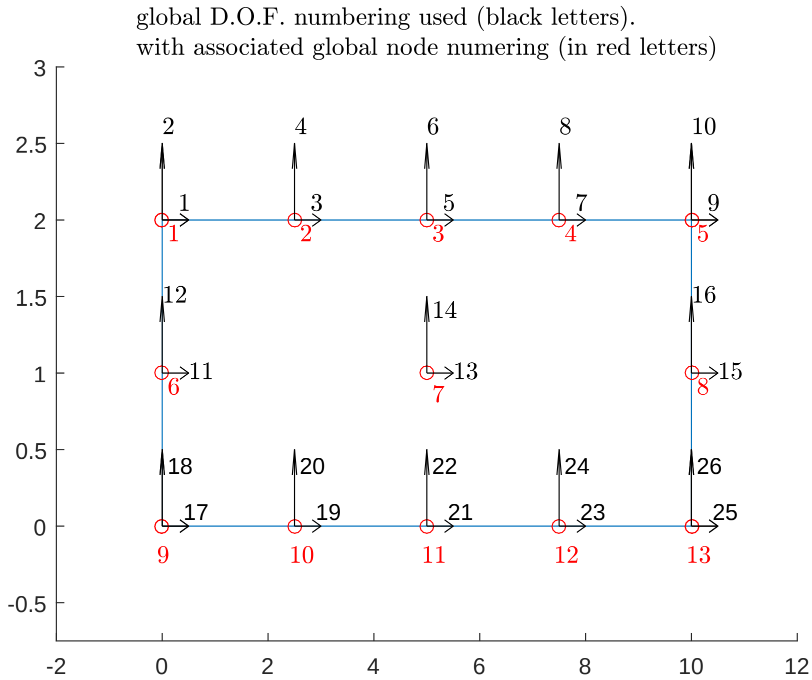


Figure 1.4: global DOF numbering used

The above was a general description of the problem and the geometry and data structures used in the Matlab implementation. Next is a discussion of the analytical derivation and the post processing stage which starts after solving for the displacements, followed by discussion of how stress was calculated at the element nodes from the stress value at the four Gaussian points.

Chapter 2

Theory and analytical derivation

2.1 Shape functions

Since an 8 node element is used, then there will be 8 shape functions. In ANSYS, the element used is called `PLANE183`. This element is a serendipity element, which means it has nodes only on the edges and no node in the middle of the element.

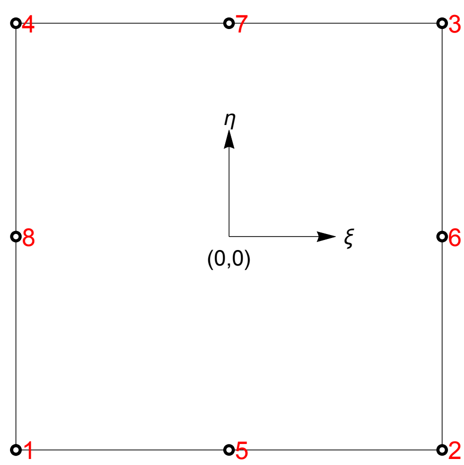


Figure 2.1: 8-node element used for the finite elements

The following are the 8 shape functions used

$$
\begin{aligned}
f_1 &= -\frac{1}{4}\left(1-\eta^2\right)(1-\xi) - \frac{1}{4}(1-\eta)\left(1-\xi^2\right) + \frac{1}{4}(1-\eta)(1-\xi) \\
f_2 &= -\frac{1}{4}\left(1-\eta^2\right)(\xi+1) - \frac{1}{4}(1-\eta)\left(1-\xi^2\right) + \frac{1}{4}(1-\eta)(\xi+1) \\
f_3 &= -\frac{1}{4}\left(1-\eta^2\right)(\xi+1) - \frac{1}{4}(\eta+1)\left(1-\xi^2\right) + \frac{1}{4}(\eta+1)(\xi+1) \\
f_4 &= -\frac{1}{4}\left(1-\eta^2\right)(1-\xi) - \frac{1}{4}(\eta+1)\left(1-\xi^2\right) + \frac{1}{4}(\eta+1)(1-\xi) \\
f_5 &= \frac{1}{2}(1-\eta)\left(1-\xi^2\right) \\
f_6 &= \frac{1}{2}\left(1-\eta^2\right)(\xi+1) \\
f_7 &= \frac{1}{2}(\eta+1)\left(1-\xi^2\right) \\
f_8 &= \frac{1}{2}\left(1-\eta^2\right)(1-\xi)
\end{aligned}
$$

The global coordinates x, y are now expressed as functions of the natural coordinates ξ, η

using

$$x(\xi,\eta)=\sum_{i=1}^{M}x_i f_i(\xi,\eta)$$
$$y(\xi,\eta)=\sum_{i=1}^{M}y_i f_i(\xi,\eta)$$

Where M is the number of nodes of each element (which is 8) and x_i, y_i are the global coordinates of these nodes. Expanding the above gives the result below.

The result below is shown for some element number k. In this expansion, x_i^k means the global x coordinate of the i^{th} node in the k^{th} element.

Similarly, y_i^k means the global y coordinate of the i^{th} node in the k^{th} element. These are read in the Matlab code using the `global_coordinates_tbl` table, which gives the global x, y coordinates of each global node and by using `elem_map_nodes` table to map the element node number to the global node number.

$$x(\xi,\eta)=x_1^k\left(-\frac{1}{4}\left(1-\eta^2\right)(1-\xi)-\frac{1}{4}(1-\eta)\left(1-\xi^2\right)+\frac{1}{4}(1-\eta)(1-\xi)\right)+x_2^k\left(-\frac{1}{4}\left(1-\eta^2\right)(\xi+1)-\frac{1}{4}(1-\eta)\left(1-\xi^2\right)-\right.$$
$$y(\xi,\eta)=y_1^k\left(-\frac{1}{4}\left(1-\eta^2\right)(1-\xi)-\frac{1}{4}(1-\eta)\left(1-\xi^2\right)+\frac{1}{4}(1-\eta)(1-\xi)\right)+y_2^k\left(-\frac{1}{4}\left(1-\eta^2\right)(\xi+1)-\frac{1}{4}(1-\eta)\left(1-\xi^2\right)-\right.$$

2.2 Finding the Jacobian

The Jacobian is evaluated at each Gaussian integration point during the integration step. It has the form

$$J=\begin{pmatrix}\frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi}\\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta}\end{pmatrix}$$

Each of the above derivatives is evaluated and used in the Matlab code.

$$\frac{\partial x}{\partial\xi}=x_1\left(\frac{1}{4}\left(1-\eta^2\right)+\frac{1}{2}(1-\eta)\xi+\frac{\eta-1}{4}\right)+x_2\left(\frac{1}{4}\left(\eta^2-1\right)+\frac{1}{2}(1-\eta)\xi+\frac{1-\eta}{4}\right)+x_3\left(\frac{1}{4}\left(\eta^2-1\right)+\frac{1}{2}(\eta+1)\xi+\frac{\eta+}{4}\right.$$
$$\frac{\partial y}{\partial\xi}=y_1\left(\frac{1}{4}\left(1-\eta^2\right)+\frac{1}{2}(1-\eta)\xi+\frac{\eta-1}{4}\right)+y_2\left(\frac{1}{4}\left(\eta^2-1\right)+\frac{1}{2}(1-\eta)\xi+\frac{1-\eta}{4}\right)+y_3\left(\frac{1}{4}\left(\eta^2-1\right)+\frac{1}{2}(\eta+1)\xi+\frac{\eta+}{4}\right.$$
$$\frac{\partial x}{\partial\eta}=x_1\left(\frac{1}{2}\eta(1-\xi)+\frac{1}{4}\left(1-\xi^2\right)+\frac{\xi-1}{4}\right)+x_2\left(\frac{1}{2}\eta(\xi+1)+\frac{1}{4}\left(1-\xi^2\right)+\frac{1}{4}(-\xi-1)\right)+x_3\left(\frac{1}{2}\eta(\xi+1)+\frac{1}{4}\left(\xi^2-1\right)+\right.$$
$$\frac{\partial y}{\partial\eta}=y_1\left(\frac{1}{2}\eta(1-\xi)+\frac{1}{4}\left(1-\xi^2\right)+\frac{\xi-1}{4}\right)+y_2\left(\frac{1}{2}\eta(\xi+1)+\frac{1}{4}\left(1-\xi^2\right)+\frac{1}{4}(-\xi-1)\right)+y_3\left(\frac{1}{2}\eta(\xi+1)+\frac{1}{4}\left(\xi^2-1\right)+\right.$$

The above is now used to find the Jacobian and its determinant and also find Γ and the matrix B_2. To find the matrix B_3 the derivatives of each shape function is taken w.r.t. ξ and η.

This below gives the result of this computation

$$
\begin{aligned}
\frac{\partial f_1}{\partial \xi} &= \frac{1}{4}\left(1-\eta^2\right) + \frac{1}{2}(1-\eta)\xi + \frac{\eta-1}{4}\\
\frac{\partial f_1}{\partial \eta} &= \frac{1}{2}\eta(1-\xi) + \frac{1}{4}\left(1-\xi^2\right) + \frac{\xi-1}{4}\\
\frac{\partial f_2}{\partial \xi} &= \frac{1}{4}\left(\eta^2-1\right) + \frac{1}{2}(1-\eta)\xi + \frac{1-\eta}{4}\\
\frac{\partial f_2}{\partial \eta} &= \frac{1}{2}\eta(\xi+1) + \frac{1}{4}\left(1-\xi^2\right) + \frac{1}{4}(-\xi-1)\\
\frac{\partial f_3}{\partial \xi} &= \frac{1}{4}\left(\eta^2-1\right) + \frac{1}{2}(\eta+1)\xi + \frac{\eta+1}{4}\\
\frac{\partial f_3}{\partial \eta} &= \frac{1}{2}\eta(\xi+1) + \frac{1}{4}\left(\xi^2-1\right) + \frac{\xi+1}{4}\\
\frac{\partial f_4}{\partial \xi} &= \frac{1}{4}\left(1-\eta^2\right) + \frac{1}{2}(\eta+1)\xi + \frac{1}{4}(-\eta-1)\\
\frac{\partial f_4}{\partial \eta} &= \frac{1}{2}\eta(1-\xi) + \frac{1}{4}\left(\xi^2-1\right) + \frac{1-\xi}{4}\\
\frac{\partial f_5}{\partial \xi} &= -(1-\eta)\xi\\
\frac{\partial f_5}{\partial \eta} &= \frac{1}{2}\left(\xi^2-1\right)\\
\frac{\partial f_6}{\partial \xi} &= \frac{1}{2}\left(1-\eta^2\right)\\
\frac{\partial f_6}{\partial \eta} &= -\eta(\xi+1)\\
\frac{\partial f_7}{\partial \xi} &= -(\eta+1)\xi\\
\frac{\partial f_7}{\partial \eta} &= \frac{1}{2}\left(1-\xi^2\right)\\
\frac{\partial f_8}{\partial \xi} &= \frac{1}{2}\left(\eta^2-1\right)\\
\frac{\partial f_8}{\partial \eta} &= -\eta(1-\xi)
\end{aligned}
$$

With the above, the matrix B_3 matrix was calculated giving

$$B = B_1 B_2 B3$$

And calculate the element stiffness matrix given by

$$
\begin{aligned}
k_{\text{elem}} &= \int B^T E B \, dV\\
&= \int_{-1}^{+1}\int_{-1}^{+1} B^T E B |J| \, dV
\end{aligned}
$$

The elements stiffness matrices k_{elem} are then combined to make the global stiffness matrix K and then the system $KD = F$ was solved for the unknowns D which are the nodal displacements in the x and y direction.

In the above F is the load vector, which in this problem contains only one non-zero entry, which is the vertical load of $-20N$ at the middle of the right edge of the beam.

In the above, the matrix B_1 is

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 1 & 1 & 0 \end{bmatrix}$$

And the matrix B_2 is

$$B_2 = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & 0 & 0 \\ \Gamma_{21} & \Gamma_{22} & 0 & 0 \\ 0 & 0 & \Gamma_{11} & \Gamma_{12} \\ 0 & 0 & \Gamma_{21} & \Gamma_{22} \end{bmatrix}$$

Where Γ is the inverse of the Jacobian matrix $\Gamma = J^{-1}$. And the matrix B_3 is

$$B_3 = \begin{bmatrix} \frac{\partial f_1}{\partial \xi} & 0 & \frac{\partial f_2}{\partial \xi} & 0 & \dots & \frac{\partial f_8}{\partial \xi} & 0 \\ \frac{\partial f_1}{\partial \eta} & 0 & \frac{\partial f_2}{\partial \eta} & 0 & \dots & \frac{\partial f_8}{\partial \eta} & 0 \\ 0 & \frac{\partial f_1}{\partial \xi} & 0 & \frac{\partial f_2}{\partial \xi} & \dots & 0 & \frac{\partial f_8}{\partial \xi} \\ 0 & \frac{\partial f_1}{\partial \eta} & 0 & \frac{\partial f_2}{\partial \eta} & \dots & 0 & \frac{\partial f_8}{\partial \eta} \end{bmatrix}$$

The following diagram shows the internal structure of the global stiffness matrix, found using the `spy()` command in Matlab. It shows the bands along the diagonals and illustrates how sparse the matrix is.

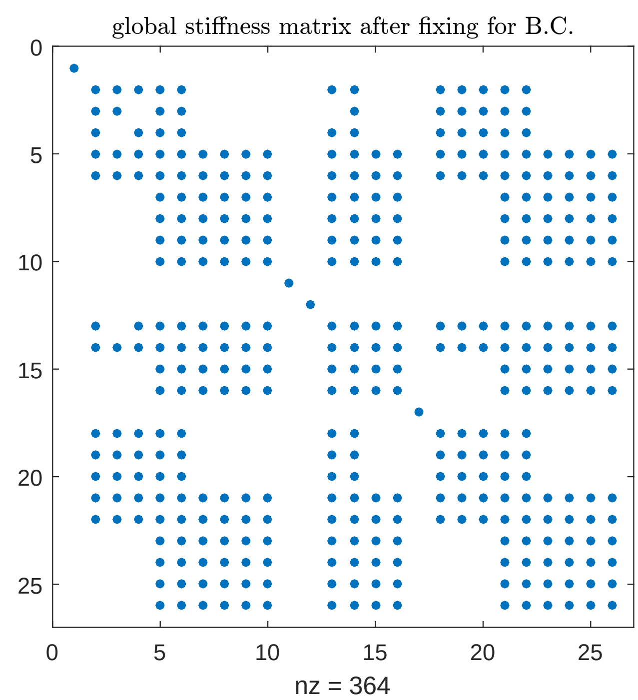


Figure 2.2: Global stiffness matrix spy() output showing the bands

The above concludes the solve stage. The next stage is the post processing, where stress calculations are performed.

2.3 Stress recovery

This is a discussion of how the stress at the elements nodes was found. Initially the direct method was used to find the stress at any point in the element.

This method was found to be accurate as long as there was no distortion. Once the angle was increased, this method did not produce stress results which agreed with ANSYS. Therefore, this method was not used, and instead a new implementation was made based on the extrapolation method.

This method is described in reference [1], pages 230-232. This method is more complicated that the direct method, but it is much more accurate. It uses two coordinates systems. The original natural coordinates system of the element (ξ, η), which extends from $-1 \dots 1$ across the length and height of the element, and a new coordinates systems called (r, s) which extends across what is called the Gaussian element.

Therefore, when $\xi = \frac{1}{\sqrt{3}}$ the value of r is one. And when $\eta = \frac{1}{\sqrt{3}}$ then $s = 1$ also.

The following diagram shows this layout more clearly.

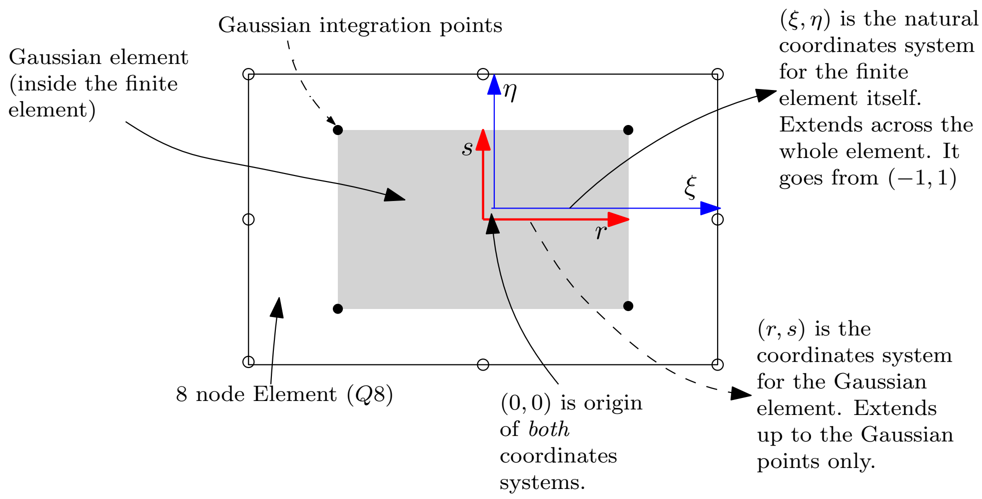


Figure 2.3: Stress recovery using r, s coordinates system inside ξ, η

Therefore, the relation between (r, s) coordinates system and (ξ, η) coordinates system is as follows

$$r = \xi\sqrt{3}$$
$$s = \eta\sqrt{3}$$

The following diagram shows the mapping at two different points for illustration

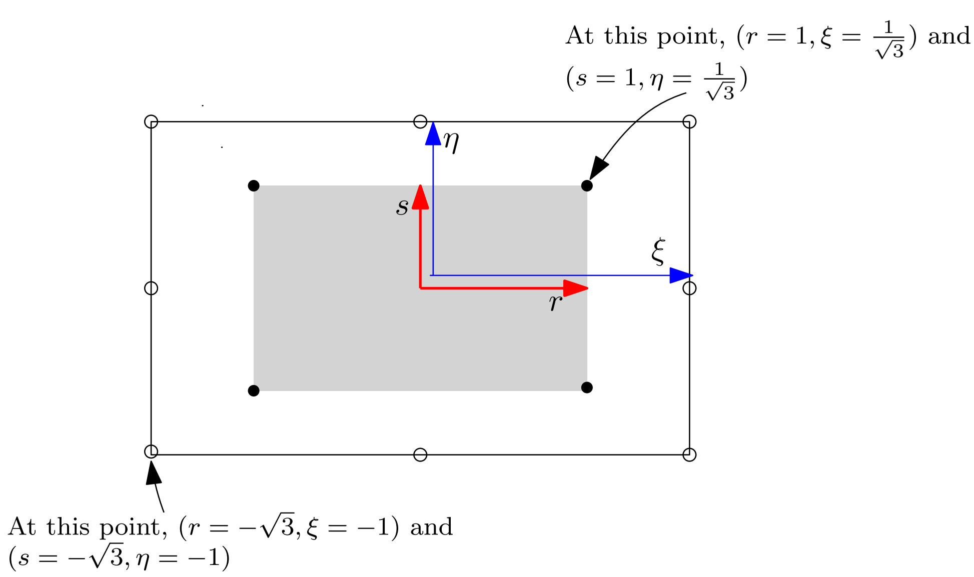


Figure 2.4: Stress recovery using r, s coordinates system inside ξ, η

Now that the mapping between ξ, η and (r, s) is determined, the next step was to find (r, s) at each point where the stress needs to be found at by extrapolating the stress value from the 4 Gaussian points to that point of interest. The reason for doing all of the above, is because (r, s) are used when evaluating the N_i shape functions described below, and not the original (ξ, η) values.

Therefore, for each point where the stress needs to be found, say point p, its coordinates in the (r, s) system are found first, and then the following extrapolation formula is applied

$$\sigma_p = \sum_{i=1}^{4} N_i \sigma_i$$

Where σ_i is the stress at the Gaussian point (which was found using the direct method based on the full 8 shape functions of the main element).

In the above, N_i are the shape functions used for extrapolation. These are not the same shape functions used in the original element. These shape functions are based on 4 node Gaussian element and are given by

$$
\begin{aligned}
N_1 &= \frac{1}{4}(1-r)(1-s) \\
N_2 &= \frac{1}{4}(1+r)(1-s) \\
N_3 &= \frac{1}{4}(1+r)(1+s) \\
N_4 &= \frac{1}{4}(1-r)(1+s)
\end{aligned}
$$

For example, to find the stress at point $(\xi, \eta) = (0, -1)$, the first step is to determine this point's (r_p, s_p) coordinates. Since $r_p = \sqrt{3}\xi$ then $r_p = 0$ and since $s_p = \sqrt{3}\eta$ then $s_p = -\sqrt{3} =$. Applying the extrapolation formula above gives

$$
\begin{aligned}
\sigma_p &= \left(\frac{1}{4}(1-r_p)(1-s_p)\right)\sigma_1 + \left(\frac{1}{4}(1+r_p)(1-s_p)\right)\sigma_2 + \left(\frac{1}{4}(1+r_p)(1+s_p)\right)\sigma_3 + \left(\frac{1}{4}(1-r_p)(1+s_p)\right)\sigma_4 \\
&= \left(\frac{1}{4}(1+\sqrt{3})\right)\sigma_1 + \left(\frac{1}{4}(1+\sqrt{3})\right)\sigma_2 + \left(\frac{1}{4}(1-\sqrt{3})\right)\sigma_3 + \left(\frac{1}{4}(1-\sqrt{3})\right)\sigma_4 \\
&= 0.6830127\sigma_1 + 0.6830127\sigma_2 - 0.1830127\sigma_3 - 0.1830127\sigma_4
\end{aligned}
$$

Since the stresses at four Gaussian points σ_i are known, the stress at the point p is now found from the above. The above method was found to produce more accurate result for σ_p than using the direct method to find σ_p and the result found for the stress at the nodes agreed with those found by ANSYS.

The following diagram shows the (r, s) coordinates of all the element points used to calculate the stresses at using this method. A total of 13 points was used per element. These are the 8 nodes of the element, and also the center of the element and the Gaussian points themselves giving a total of 13 points. These are used to generate the stress contour. This was done for both elements. The generated stress contour agreed with ANSYS results.

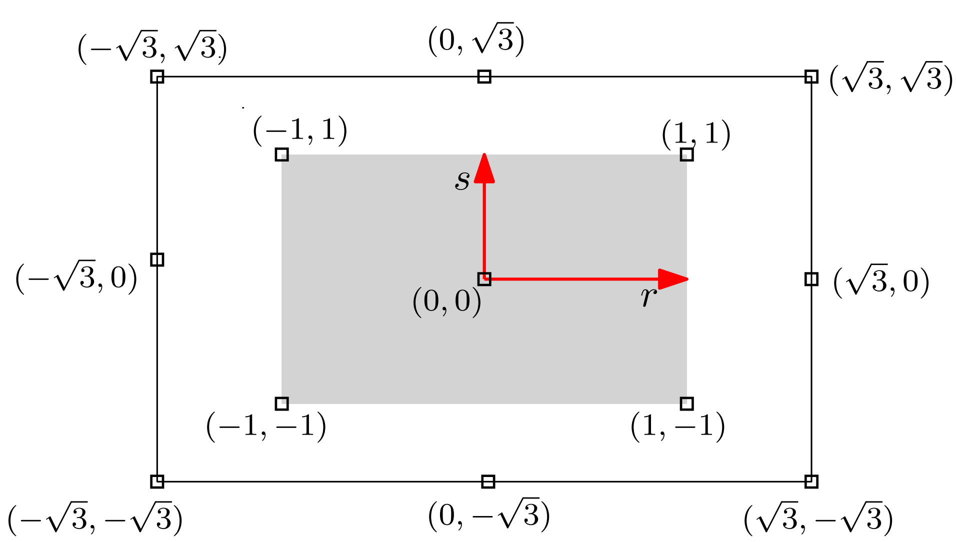


Figure 2.5: Location of points used for stress recovery in the r, s coordinate system

Chapter 3

Results

The results are listed by the angle θ used to distort the elements with. For each angle, the deflection and σ_x stress contour and tables are generated using Matlab and also using ANYS to compare with side by side.

The following angles are used (in degrees) $\{0, 15, 30, 45, 50, 55\}$. When trying to use 60 degrees distortion, ANSYS complained and gave number of computational error messages relating to the element shape. It is not clear why ANSYS did not accept such large angle, since the Matlab implementation worked. But since ANSYS did not produce result for this case, the angle 55 degrees was the maximum distortion used for both Matlab and ANSYS.

For each angle, a short summary of the result in the form of a table is first given that compares Matlab and ANSYS result. This short summary contains only the deflection at the bottom right corner, which is node 13. After this, the full result and stress plots are given.

3.1 No element distortion. zero angle

3.1.1 summary of result

	x (meter)	y (meter)
Matlab	−0.15	−1.02895
ANSYS	−0.15	−1.0289

Table 3.1: Short summary of test case zero degree distortion

3.1.2 Matlab result

global node #	x (meter)	y (meter)
1	0.000000	−0.004650
2	0.065794	−0.096522
3	0.112725	−0.329300
4	0.140794	−0.655828
5	0.150000	−1.028950
6	0.000000	0.000000
7	−0.000000	−0.327050
8	−0.000000	−1.029100
9	0.000000	−0.004650
10	−0.065794	−0.096522
11	−0.112725	−0.329300
12	−0.140794	−0.655828
13	−0.150000	−1.028950

Table 3.2: Matlab result. nodal solutions, angle [0] degree

The following figure shows the deformation found

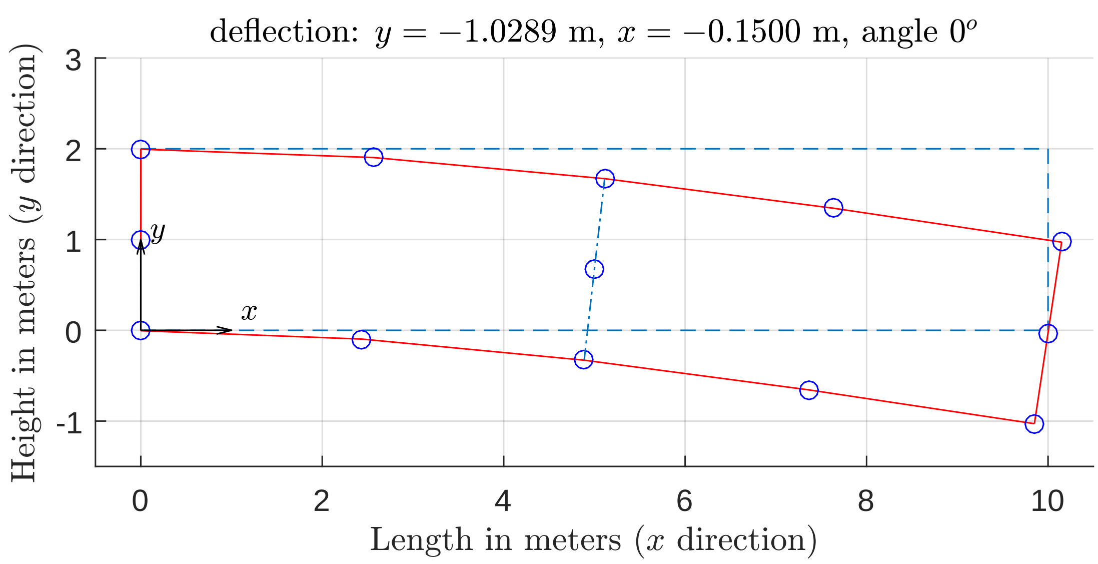


Figure 3.1: deflection found using 2 elements using Matlab, zero degree

The following table shows Matlab result for the direct stress σ_x found at each node for each element.

global node #	x	y	σ_x N/m$\hat{2}$
9	0.0000	0.0000	−300.000
11	5.0000	0.0000	−150.000
3	5.0000	2.0000	150.000
1	0.0000	2.0000	300.000
10	2.5000	0.0000	−225.000
7	5.0000	1.0000	−0.000
2	2.5000	2.0000	225.000
6	0.0000	1.0000	−0.000
center	2.5000	1.0000	−0.000
Gauss point 1	1.0566	0.4226	−154.904
Gauss point 2	3.9434	0.4226	−104.904
Gauss point 3	3.9434	1.5774	104.904
Gauss point 4	1.0566	1.5774	154.904

Table 3.3: Matlab result. direct stress σ_x at each node, First element, angle [0] degree

global node #	x	y	σ_x N/m$\hat{2}$
11	5.0000	0.0000	−150.000
13	10.0000	0.0000	−0.000
5	10.0000	2.0000	0.000
3	5.0000	2.0000	150.000
12	7.5000	0.0000	−75.000
8	10.0000	1.0000	0.000
4	7.5000	2.0000	75.000
7	5.0000	1.0000	0.000
center	7.5000	1.0000	0.000
Gauss point 1	6.0566	0.4226	−68.301
Gauss point 2	8.9434	0.4226	−18.301
Gauss point 3	8.9434	1.5774	18.301
Gauss point 4	6.0566	1.5774	68.301

Table 3.4: Matlab result. direct stress at each node, Second element, angle [0] degree

The following shows the direct stress contour generated in Matlab

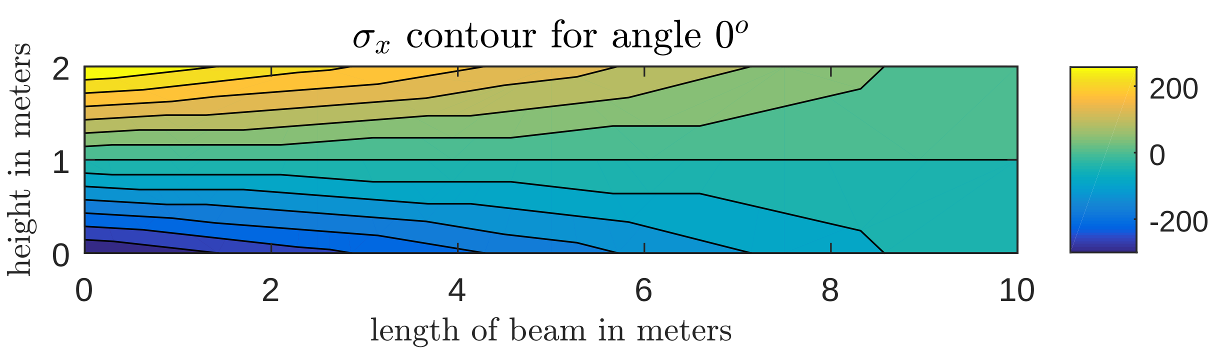


Figure 3.2: Contour of direct stress found using 2 elements using Matlab, zero degree

3.1.3 ANSYS result

```

PRINT U    NODAL SOLUTION PER NODE

 ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM

   NODE       UX           UY          UZ          USUM
      1   0.0000      -0.46500E-002  0.0000      0.46500E-002
      2  0.65794E-001-0.96522E-001  0.0000      0.11681
      3  0.11272      -0.32930       0.0000      0.34806
      4  0.14079      -0.65583       0.0000      0.67077
      5  0.15000       -1.0289       0.0000       1.0398
      6   0.0000        0.0000       0.0000       0.0000
      7  0.12540E-013-0.32705       0.0000      0.32705
      8  0.14395E-013 -1.0291       0.0000       1.0291
      9   0.0000      -0.46500E-002  0.0000      0.46500E-002
     10 -0.65794E-001-0.96522E-001  0.0000      0.11681
```

```
      11 -0.11272     -0.32930       0.0000       0.34806
      12 -0.14079     -0.65583       0.0000       0.67077
      13 -0.15000      -1.0289       0.0000        1.0398

MAXIMUM ABSOLUTE VALUES
NODE          5            8            0           13
VALUE    0.15000      -1.0291       0.0000       1.0398

/OUTPUT FILE= ansys_stress_solution_0.txt
```

The following figure shows the deformation found

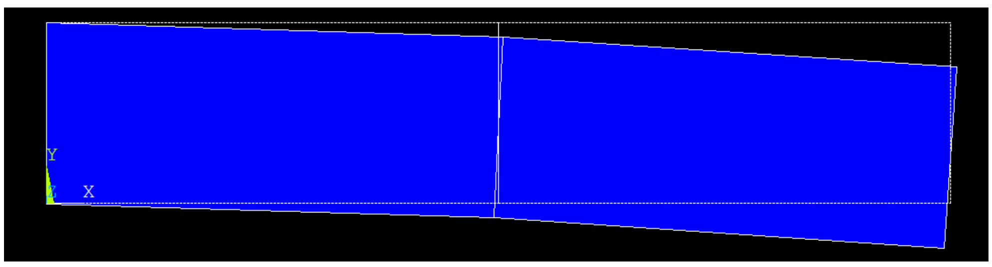


Figure 3.3: deflection found using 2 elements using ANSYS, zero degree

The following table shows ANSYS result for the direct stress σ_x found at each node for each element.

```

PRINT S    ELEMENT SOLUTION PER ELEMENT

***** POST1 ELEMENT NODAL STRESS LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING X,Y,Z VALUES ARE IN GLOBAL COORDINATES


 ELEMENT=       1        PLANE183
   NODE     SX           SY           SZ           SXY          SYZ
  SXZ
      9  -300.00       3.0000       0.0000      -10.000       0.0000
  0.0000
     11  -150.00     -0.18598E-010  0.0000      -10.000       0.0000
  0.0000
      3   150.00      0.21138E-010  0.0000      -10.000       0.0000
  0.0000
      1   300.00      -3.0000       0.0000      -10.000       0.0000
  0.0000

 ELEMENT=       2        PLANE183
   NODE     SX           SY           SZ           SXY          SYZ
  SXZ
     11  -150.00      0.36168E-010  0.0000      -10.000       0.0000
  0.0000
     13 -0.45564E-010 -3.0000       0.0000      -10.000       0.0000
  0.0000
      5  0.45869E-010  3.0000       0.0000      -10.000       0.0000
  0.0000
      3   150.00     -0.51808E-010  0.0000      -10.000       0.0000
  0.0000
```

The following shows the direct stress contour generated in ANSYS

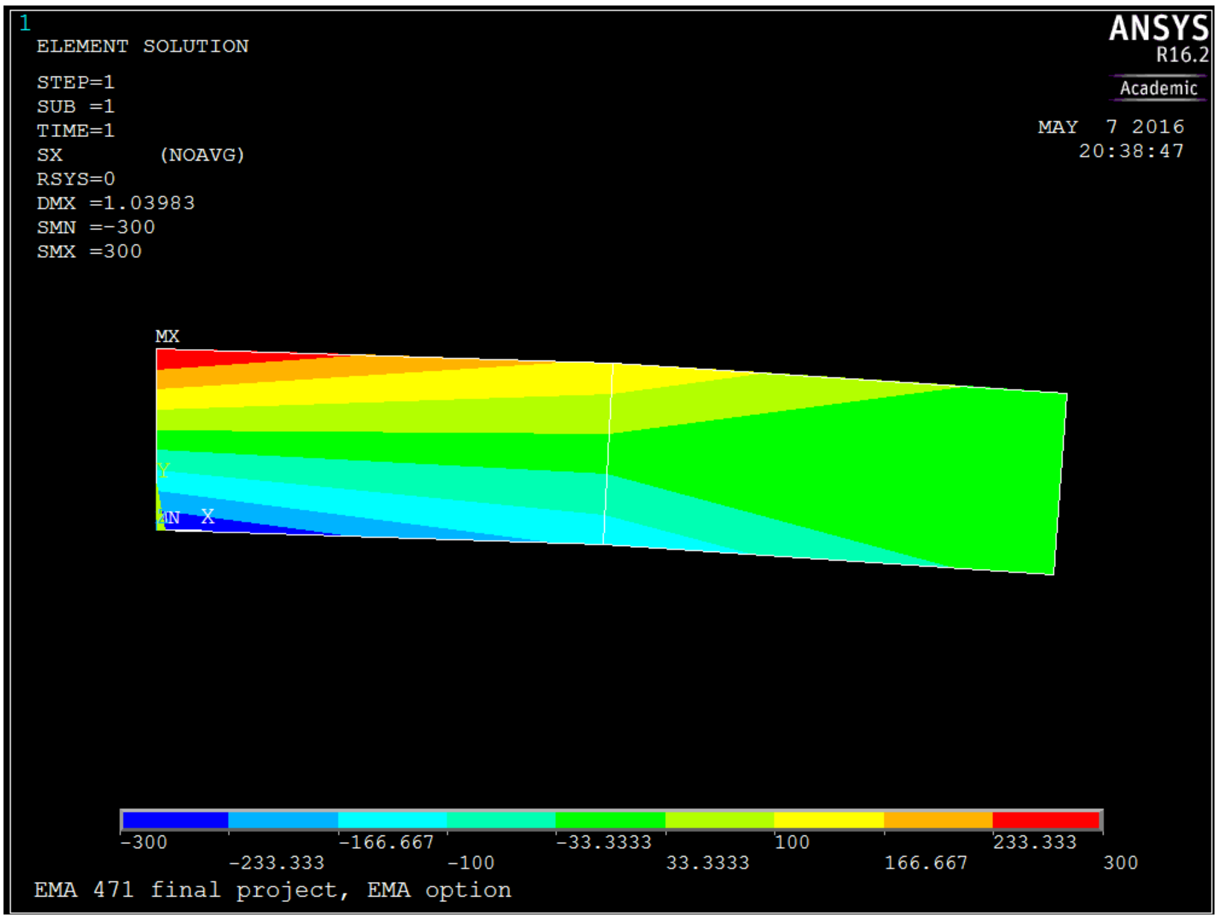


Figure 3.4: Contour of direct stress found using 2 elements using ANSYS, zero degree

3.2 15 degrees distortion

3.2.1 summary of result

	x (meter)	y (meter)
Matlab	−0.150262	−1.02555
ANSYS	−0.15026	−1.0256

Table 3.5: Short summary of test case 15 degrees distortion

3.2.2 Matlab result

global node #	x (meter)	y (meter)
1	0.000000	−0.008808
2	0.066221	−0.096603
3	0.115141	−0.356575
4	0.138750	−0.653161
5	0.146026	−1.012233
6	0.000000	0.000000
7	0.000596	−0.326056
8	0.001059	−1.018939
9	0.000000	−0.000457
10	−0.064844	−0.096198
11	−0.108540	−0.300195
12	−0.139066	−0.648279
13	−0.150262	−1.025550

Table 3.6: Matlab result. nodal solutions, angle [15] degree

The following figure shows the deformation found

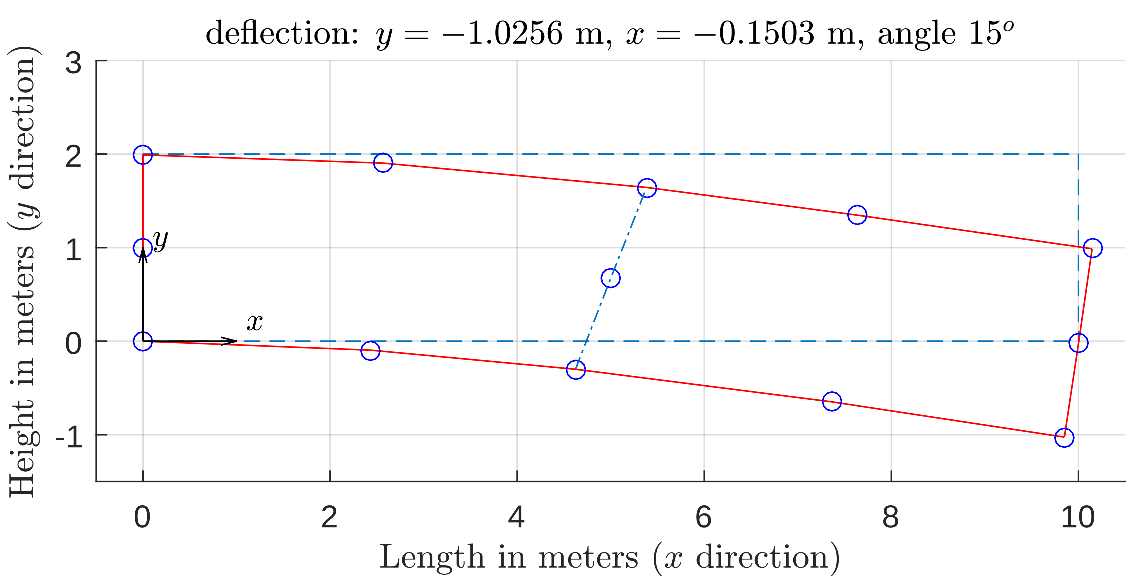


Figure 3.5: deflection found using 2 elements using Matlab, 15 degrees

The following table shows Matlab result for the direct stress σ_x found at each node for each element.

global node #	x	y	σ_x N/m$\hat{2}$
9	0.0000	0.0000	−299.049
11	4.7321	0.0000	−148.195
3	5.2679	2.0000	144.369
1	0.0000	2.0000	300.912
10	2.5000	0.0000	−223.622
7	5.0000	1.0000	−1.913
2	2.5000	2.0000	222.641
6	0.0000	1.0000	0.932
center	2.5000	1.0000	−0.491
Gauss point 1	1.0566	0.4226	−154.111
Gauss point 2	3.9434	0.4226	−104.520
Gauss point 3	3.9434	1.5774	101.897
Gauss point 4	1.0566	1.5774	154.772

Table 3.7: Matlab result. direct stress σ_x at each node, First element, angle [15] degree

global node #	x	y	σ_x N/m$\hat{2}$
11	4.7321	0.0000	−146.183
13	10.0000	0.0000	−0.994
5	10.0000	2.0000	2.746
3	5.2679	2.0000	142.734
12	7.5000	0.0000	−73.588
8	10.0000	1.0000	0.876
4	7.5000	2.0000	72.740
7	5.0000	1.0000	−1.724
center	7.5000	1.0000	−0.424
Gauss point 1	6.0566	0.4226	−67.181
Gauss point 2	8.9434	0.4226	−18.150
Gauss point 3	8.9434	1.5774	18.803
Gauss point 4	6.0566	1.5774	64.831

Table 3.8: Matlab result. direct stress at each node, Second element, angle [15] degree

The following shows the direct stress contour generated in Matlab

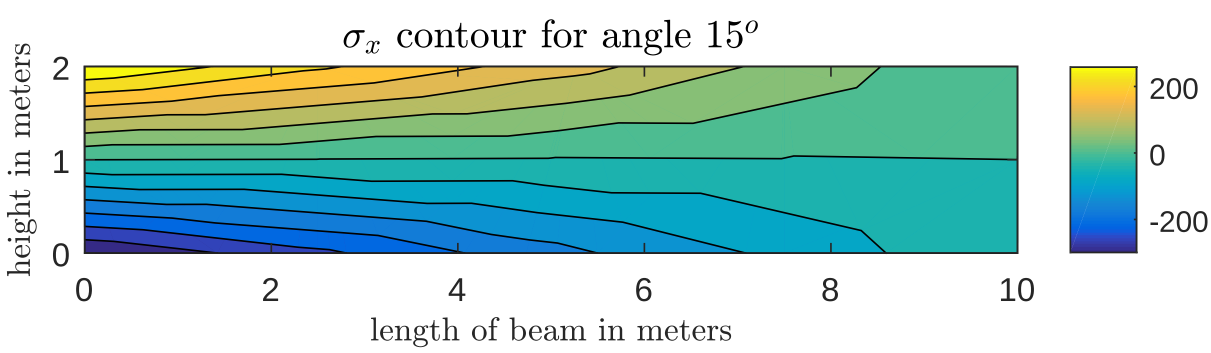


Figure 3.6: Contour of direct stress found using 2 elements using Matlab, 15 degrees

3.2.3 ANSYS result

```

PRINT U    NODAL SOLUTION PER NODE

 ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=      1  SUBSTEP=      1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM

   NODE      UX          UY          UZ          USUM
      1   0.0000     -0.88076E-02  0.0000     0.88076E-02
      2  0.66221E-01-0.96603E-01  0.0000     0.11712
      3  0.11514     -0.35658     0.0000     0.37470
      4  0.13875     -0.65316     0.0000     0.66774
      5  0.14603      -1.0122     0.0000      1.0227
      6   0.0000       0.0000     0.0000      0.0000
      7  0.59649E-03-0.32606     0.0000     0.32606
      8  0.10590E-02 -1.0189     0.0000      1.0189
      9   0.0000     -0.45680E-03  0.0000     0.45680E-03
     10 -0.64844E-01-0.96198E-01  0.0000     0.11601
```

```
      11 -0.10854    -0.30019      0.0000      0.31921
      12 -0.13907    -0.64828      0.0000      0.66303
      13 -0.15026     -1.0256      0.0000       1.0365

MAXIMUM ABSOLUTE VALUES
NODE          13          13           0          13
VALUE  -0.15026     -1.0256      0.0000      1.0365

/OUTPUT FILE= ansys_stress_solution_15.txt
```

The following figure shows the deformation found

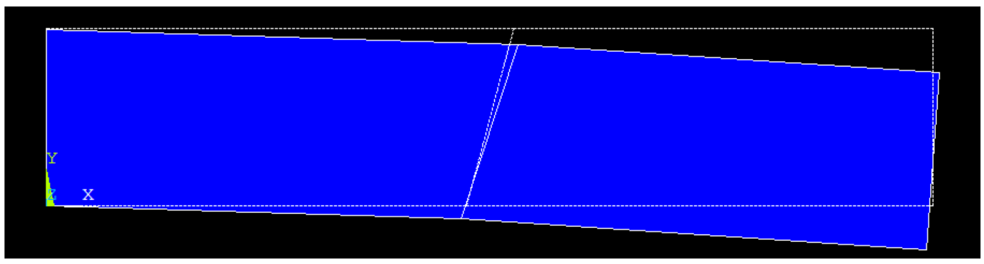


Figure 3.7: deflection found using 2 elements using ANSYS, 15 degrees

The following table shows ANSYS result for the direct stress σ_x found at each node for each element.

```

PRINT S    ELEMENT SOLUTION PER ELEMENT

***** POST1 ELEMENT NODAL STRESS LISTING *****

 LOAD STEP=      1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING X,Y,Z VALUES ARE IN GLOBAL COORDINATES


 ELEMENT=       1         PLANE183
   NODE    SX          SY          SZ          SXY         SYZ         SXZ
      9  -299.05      9.0115      0.0000     -31.003      0.0000      0.0000
     11  -148.20     -8.2859      0.0000     -10.894      0.0000      0.0000
      3   144.37     -11.096      0.0000     -11.079      0.0000      0.0000
      1   300.91      4.9358      0.0000      11.245      0.0000      0.0000

 ELEMENT=       2         PLANE183
   NODE    SX          SY          SZ          SXY         SYZ         SXZ
     11  -146.18      1.3738      0.0000      4.0905      0.0000      0.0000
     13 -0.99365     -3.0444      0.0000     -16.666      0.0000      0.0000
      5   2.7463     0.33483      0.0000     -3.5896      0.0000      0.0000
      3   142.73      7.0677      0.0000     -23.081      0.0000      0.0000
```

The following shows the direct stress contour generated in ANSYS

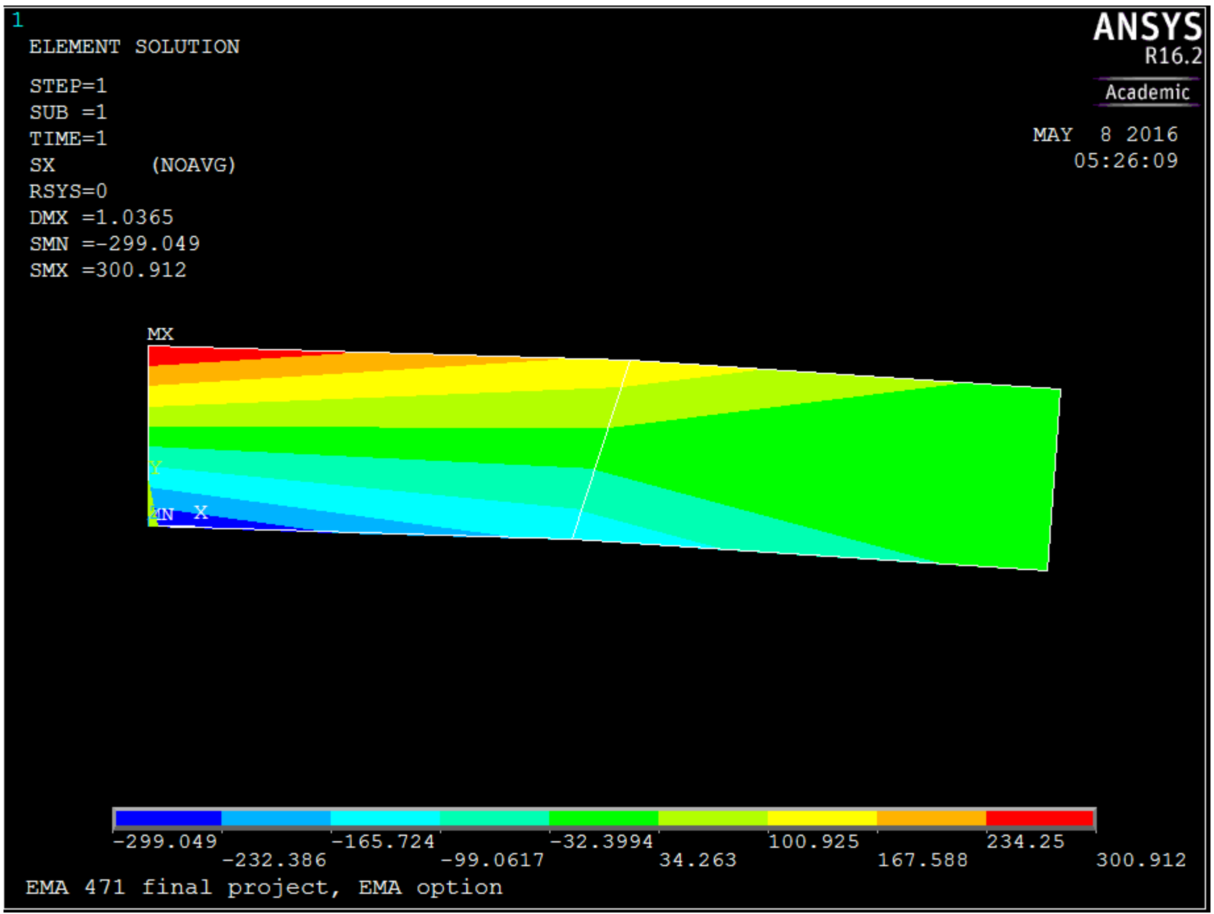


Figure 3.8: Contour of direct stress found using 2 elements using ANSYS, 15 degrees

3.3 30 degrees distortion

3.3.1 summary of result

	x (meter)	y (meter)
Matlab	−0.146118	−0.998214
ANSYS	−0.14612	−0.99821

Table 3.9: Short summary of test case 30 degrees distortion

3.3.2 Matlab result

global node #	x (meter)	y (meter)
1	0.000000	−0.013054
2	0.065955	−0.096602
3	0.115155	−0.384450
4	0.132363	−0.638148
5	0.137945	−0.972694
6	0.000000	0.000000
7	0.001094	−0.322637
8	0.002043	−0.985167
9	0.000000	0.003878
10	−0.063350	−0.095375
11	−0.102487	−0.265899
12	−0.132998	−0.628986
13	−0.146118	−0.998214

Table 3.10: Matlab result. nodal solutions, angle [30] degree

The following figure shows the deformation found

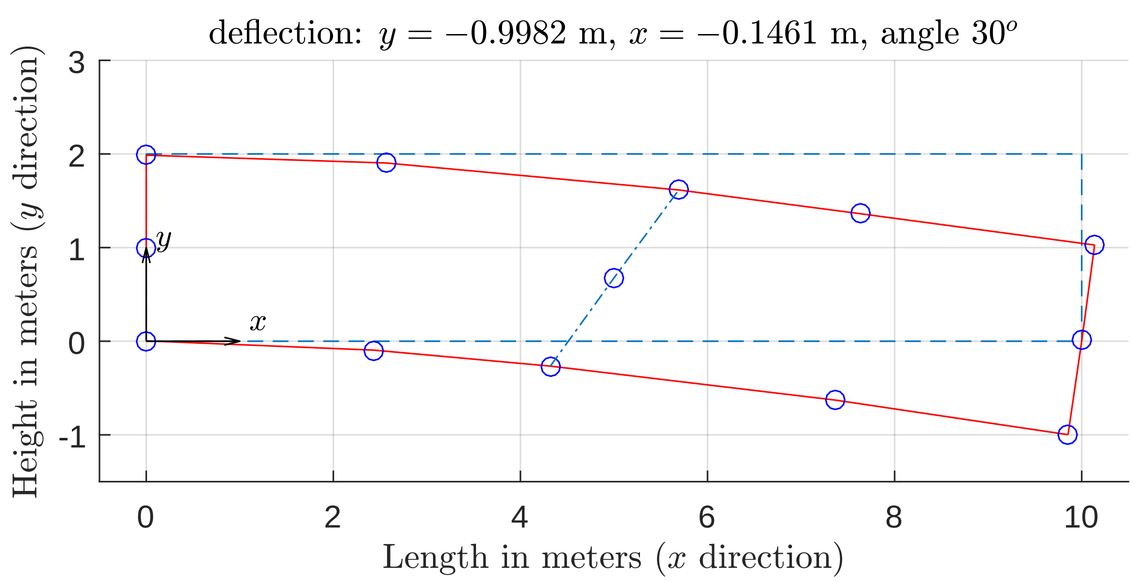


Figure 3.9: deflection found using 2 elements using Matlab, 30 degrees

The following table shows Matlab result for the direct stress σ_x found at each node for each element.

global node #	x	y	σ_x N/m$\hat{2}$
9	0.0000	0.0000	−297.609
11	4.4226	0.0000	−138.871
3	5.5774	2.0000	128.858
1	0.0000	2.0000	302.166
10	2.5000	0.0000	−218.240
7	5.0000	1.0000	−5.007
2	2.5000	2.0000	215.512
6	0.0000	1.0000	2.279
center	2.5000	1.0000	−1.364
Gauss point 1	1.0566	0.4226	−152.145
Gauss point 2	3.9434	0.4226	−101.010
Gauss point 3	3.9434	1.5774	94.076
Gauss point 4	1.0566	1.5774	153.623

Table 3.11: Matlab result. direct stress σ_x at each node, First element, angle [30] degree

global node #	x	y	σ_x N/m$\hat{2}$
11	4.4226	0.0000	−129.964
13	10.0000	0.0000	−6.206
5	10.0000	2.0000	9.722
3	5.5774	2.0000	123.499
12	7.5000	0.0000	−68.085
8	10.0000	1.0000	1.758
4	7.5000	2.0000	66.611
7	5.0000	1.0000	−3.232
center	7.5000	1.0000	−0.737
Gauss point 1	6.0566	0.4226	−60.856
Gauss point 2	8.9434	0.4226	−18.386
Gauss point 3	8.9434	1.5774	19.792
Gauss point 4	6.0566	1.5774	56.500

Table 3.12: Matlab result. direct stress at each node, Second element, angle [30] degree

The following shows the direct stress contour generated in Matlab

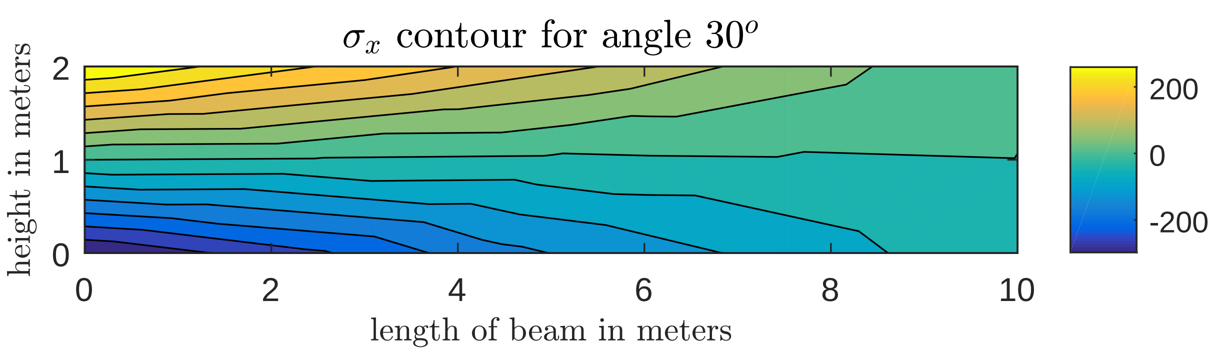


Figure 3.10: Contour of direct stress found using 2 elements using Matlab, 30 degrees

3.3.3 ANSYS result

```

PRINT U    NODAL SOLUTION PER NODE

 ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM

   NODE      UX          UY         UZ          USUM
      1   0.0000     -0.13054E-01  0.0000     0.13054E-01
      2  0.65955E-01-0.96602E-01  0.0000     0.11697
      3  0.11515     -0.38445     0.0000     0.40133
      4  0.13236     -0.63815     0.0000     0.65173
      5  0.13795     -0.97269     0.0000     0.98243
      6   0.0000       0.0000     0.0000      0.0000
      7  0.10941E-02-0.32264     0.0000     0.32264
      8  0.20432E-02-0.98517     0.0000     0.98517
      9   0.0000      0.38778E-02  0.0000     0.38778E-02
     10 -0.63350E-01-0.95375E-01  0.0000     0.11450
```

```
      11 -0.10249    -0.26590      0.0000      0.28497
      12 -0.13300    -0.62899      0.0000      0.64289
      13 -0.14612    -0.99821      0.0000       1.0089

MAXIMUM ABSOLUTE VALUES
NODE          13          13            0          13
VALUE   -0.14612    -0.99821      0.0000      1.0089

/OUTPUT FILE= ansys_stress_solution_30.txt
```

The following figure shows the deformation found

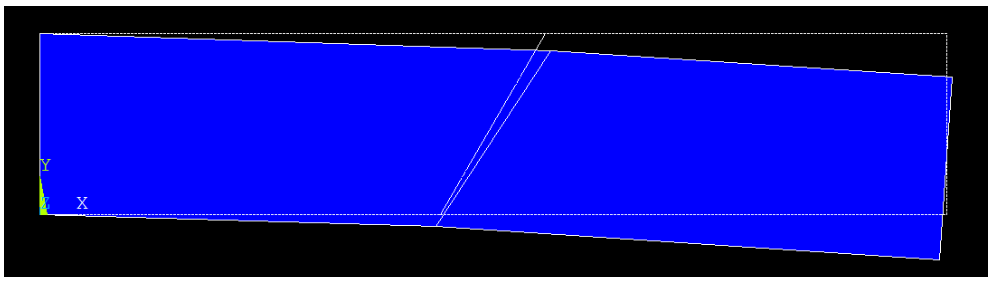


Figure 3.11: deflection found using 2 elements using ANSYS, 30 degrees

The following table shows ANSYS result for the direct stress σ_x found at each node for each element.

```

PRINT S    ELEMENT SOLUTION PER ELEMENT

***** POST1 ELEMENT NODAL STRESS LISTING *****

 LOAD STEP=      1  SUBSTEP=      1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING X,Y,Z VALUES ARE IN GLOBAL COORDINATES


 ELEMENT=       1         PLANE183
   NODE    SX          SY          SZ          SXY         SYZ         SXZ
      9   -297.61      12.886      0.0000     -52.150      0.0000      0.0000
     11   -138.87     -13.160      0.0000     -13.331      0.0000      0.0000
      3    128.86     -25.087      0.0000     -15.021      0.0000      0.0000
      1    302.17      15.057      0.0000      33.142      0.0000      0.0000

 ELEMENT=       2         PLANE183
   NODE    SX          SY          SZ          SXY         SYZ         SXZ
     11   -129.96     -3.2502      0.0000      17.425      0.0000      0.0000
     13   -6.2065    -0.12448      0.0000     -22.705      0.0000      0.0000
      5    9.7219     -5.3409      0.0000      1.4979      0.0000      0.0000
      3    123.50      21.262      0.0000     -32.751      0.0000      0.0000
```

The following shows the direct stress contour generated in ANSYS

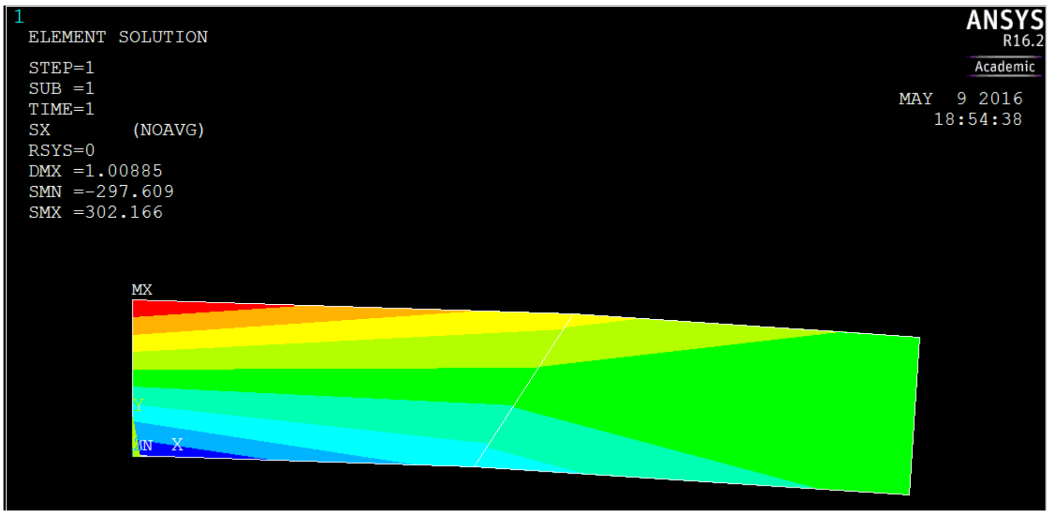


Figure 3.12: Contour of direct stress found using 2 elements using ANSYS, 30 degrees

3.4 45 degrees distortion

3.4.1 summary of result

	x (meter)	y (meter)
Matlab	−0.135417	−0.934503
ANSYS	−0.13542	−0.93450

Table 3.13: Short summary of test case 45 degrees distortion

3.4.2 Matlab result

global node #	x (meter)	y (meter)
1	0.000000	−0.017363
2	0.064453	−0.096794
3	0.110558	−0.415091
4	0.119874	−0.603430
5	0.124609	−0.901330
6	0.000000	0.000000
7	0.001251	−0.315104
8	0.002702	−0.916990
9	0.000000	0.008169
10	−0.061186	−0.093436
11	−0.093955	−0.220369
12	−0.120788	−0.592443
13	−0.135417	−0.934503

Table 3.14: Matlab result. nodal solutions, angle [45] degree

The following figure shows the deformation found

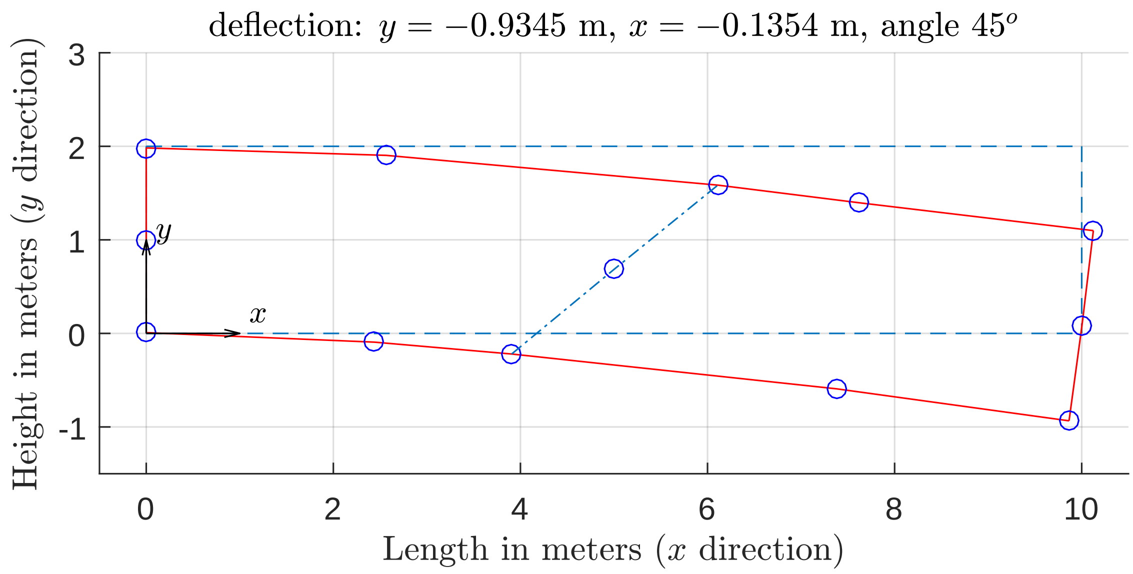


Figure 3.13: deflection found using 2 elements using Matlab, 45 degrees

The following table shows Matlab result for the direct stress σ_x found at each node for each element.

global node #	x	y	σ_x N/m$\hat{2}$
9	0.0000	0.0000	−294.696
11	4.0000	0.0000	−120.055
3	6.0000	2.0000	96.964
1	0.0000	2.0000	304.372
10	2.5000	0.0000	−207.376
7	5.0000	1.0000	−11.546
2	2.5000	2.0000	200.668
6	0.0000	1.0000	4.838
center	2.5000	1.0000	−3.354
Gauss point 1	1.0566	0.4226	−148.254
Gauss point 2	3.9434	0.4226	−94.038
Gauss point 3	3.9434	1.5774	77.871
Gauss point 4	1.0566	1.5774	151.005

Table 3.15: Matlab result. direct stress σ_x at each node, First element, angle [45] degree

global node #	x	y	σ_x N/m$\hat{2}$
11	4.0000	0.0000	−98.498
13	10.0000	0.0000	−17.052
5	10.0000	2.0000	22.022
3	6.0000	2.0000	91.497
12	7.5000	0.0000	−57.775
8	10.0000	1.0000	2.485
4	7.5000	2.0000	56.759
7	5.0000	1.0000	−3.501
center	7.5000	1.0000	−0.508
Gauss point 1	6.0566	0.4226	−47.876
Gauss point 2	8.9434	0.4226	−19.267
Gauss point 3	8.9434	1.5774	21.706
Gauss point 4	6.0566	1.5774	43.404

Table 3.16: Matlab result. direct stress at each node, Second element, angle [45] degree

The following shows the direct stress contour generated in Matlab

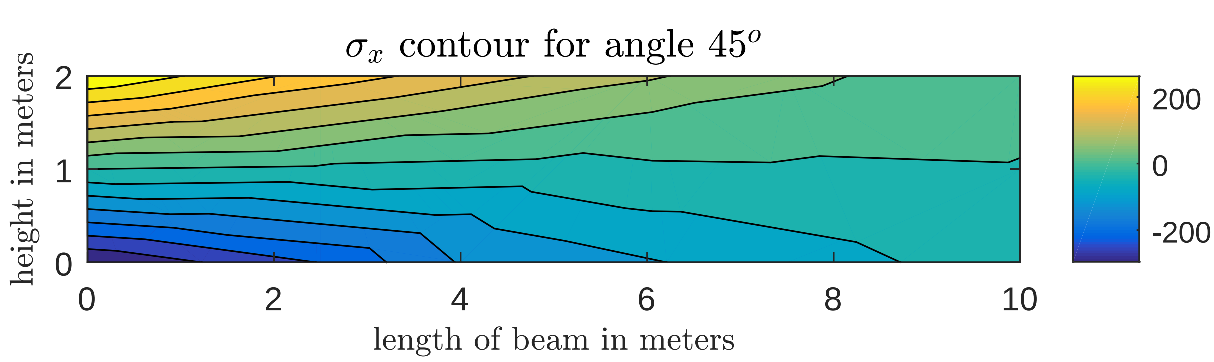


Figure 3.14: Contour of direct stress found using 2 elements using Matlab, 45 degrees

3.4.3 ANSYS result

```

PRINT U    NODAL SOLUTION PER NODE

 ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM

   NODE      UX          UY          UZ          USUM
      1   0.0000    -0.17363E-01  0.0000     0.17363E-01
      2  0.64453E-01-0.96794E-01  0.0000     0.11629
      3  0.11056    -0.41509      0.0000     0.42956
      4  0.11987    -0.60343      0.0000     0.61522
      5  0.12461    -0.90133      0.0000     0.90990
      6   0.0000      0.0000      0.0000      0.0000
      7  0.12507E-02-0.31510      0.0000     0.31511
      8  0.27021E-02-0.91699      0.0000     0.91699
      9   0.0000     0.81687E-02  0.0000     0.81687E-02
     10 -0.61186E-01-0.93436E-01  0.0000     0.11169
     11 -0.93955E-01-0.22037      0.0000     0.23956
     12 -0.12079    -0.59244      0.0000     0.60463
     13 -0.13542    -0.93450      0.0000     0.94426

MAXIMUM ABSOLUTE VALUES
NODE          13          13           0          13
VALUE  -0.13542    -0.93450      0.0000     0.94426

/OUTPUT FILE= ansys_stress_solution_45.txt
```

The following figure shows the deformation found

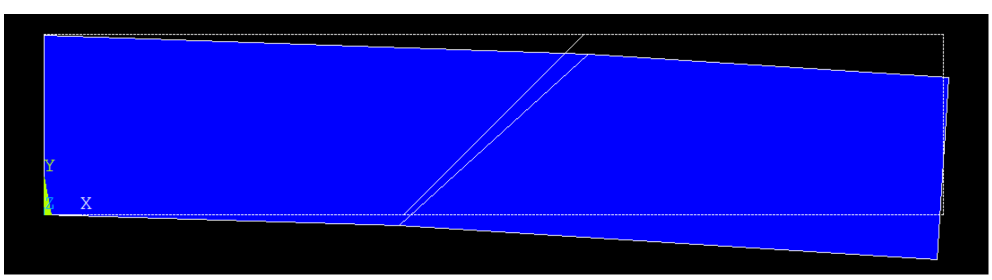


Figure 3.15: deflection found using 2 elements using ANSYS, 45 degrees

The following table shows ANSYS result for the direct stress σ_x found at each node for each element.

```

PRINT S    ELEMENT SOLUTION PER ELEMENT

***** POST1 ELEMENT NODAL STRESS LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING X,Y,Z VALUES ARE IN GLOBAL COORDINATES


 ELEMENT=       1         PLANE183
   NODE    SX          SY          SZ          SXY         SYZ         SXZ
      9  -294.70      13.722      0.0000     -72.599      0.0000      0.0000
     11  -120.06     -12.923      0.0000     -16.680      0.0000      0.0000
      3   96.964     -41.242      0.0000     -23.566      0.0000      0.0000
```

```
         1    304.37       27.298       0.0000       54.789       0.0000       0.0000

  ELEMENT=        2         PLANE183
    NODE     SX           SY           SZ           SXY          SYZ          SXZ
      11   -98.498      -15.367       0.0000       25.908       0.0000       0.0000
      13   -17.052       6.6768       0.0000      -26.033       0.0000       0.0000
       5    22.022      -14.911       0.0000       2.3133       0.0000       0.0000
       3    91.497       45.921       0.0000      -32.004       0.0000       0.0000
```

The following shows the direct stress contour generated in ANSYS

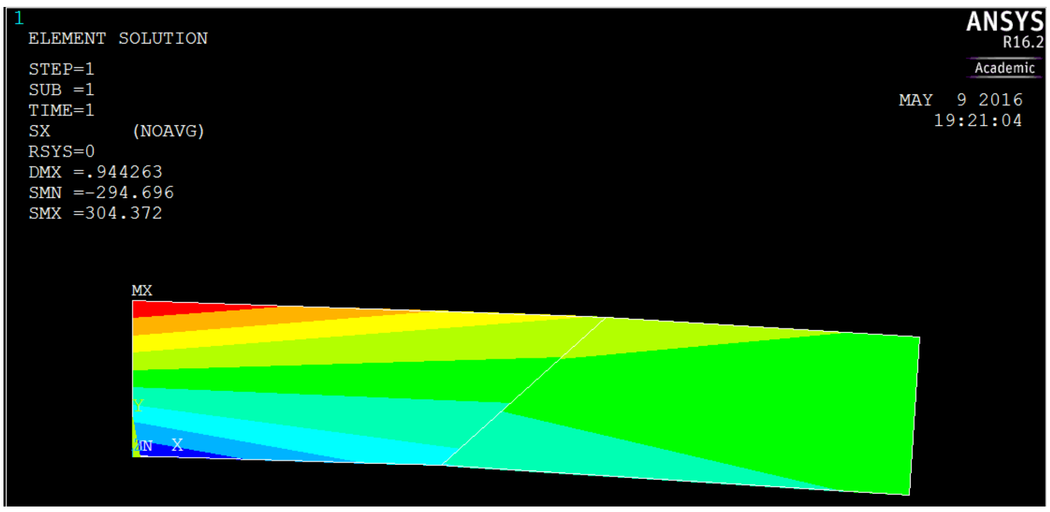


Figure 3.16: Contour of direct stress found using 2 elements using ANSYS, 45 degrees

3.5 50 degrees distortion

3.5.1 summary of result

	x (meter)	y (meter)
Matlab	−0.129803	−0.901557
ANSYS	−0.12980	−0.90156

Table 3.17: Short summary of test case 50 degrees distortion

3.5.2 Matlab result

global node #	x (meter)	y (meter)
1	0.000000	−0.018726
2	0.063489	−0.097037
3	0.107149	−0.426310
4	0.113960	−0.585035
5	0.118879	−0.868383
6	0.000000	0.000000
7	0.001133	−0.311076
8	0.002731	−0.883753
9	0.000000	0.009386
10	−0.060301	−0.092294
11	−0.090400	−0.200709
12	−0.114927	−0.574883
13	−0.129803	−0.901557

Table 3.18: Matlab result. nodal solutions, angle [50] degree

The following figure shows the deformation found

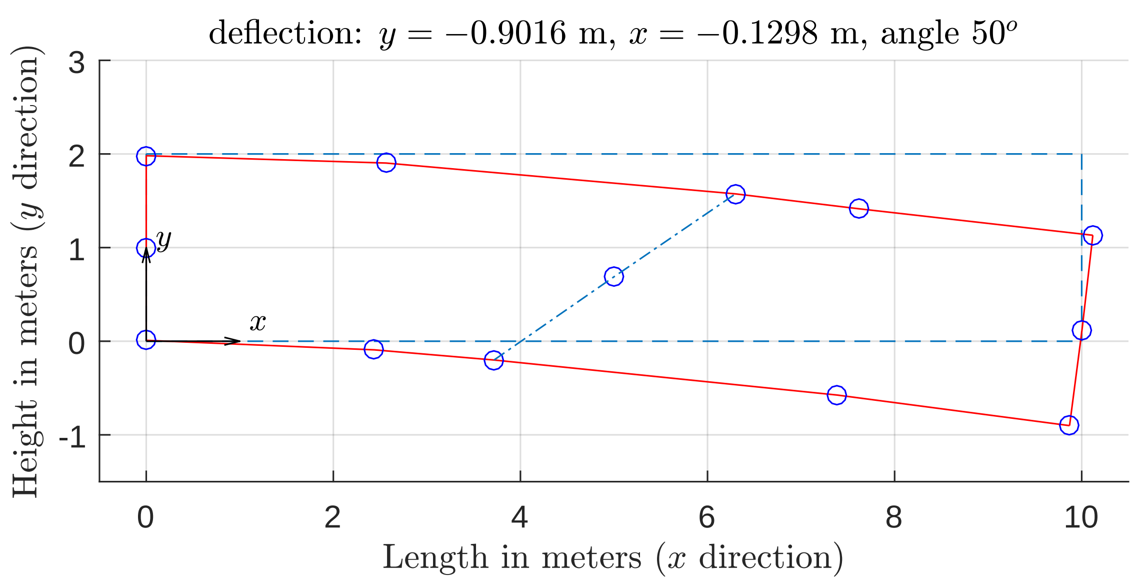


Figure 3.17: deflection found using 2 elements using Matlab, 50 degrees

The following table shows Matlab result for the direct stress σ_x found at each node for each element.

global node #	x	y	σ_x N/m$\hat{2}$
9	0.0000	0.0000	−292.998
11	3.8082	0.0000	−111.352
3	6.1918	2.0000	80.771
1	0.0000	2.0000	305.494
10	2.5000	0.0000	−202.175
7	5.0000	1.0000	−15.290
2	2.5000	2.0000	193.133
6	0.0000	1.0000	6.248
center	2.5000	1.0000	−4.521
Gauss point 1	1.0566	0.4226	−146.283
Gauss point 2	3.9434	0.4226	−90.990
Gauss point 3	3.9434	1.5774	69.513
Gauss point 4	1.0566	1.5774	149.676

Table 3.19: Matlab result. direct stress σ_x at each node, First element, angle [50] degree

global node #	x	y	σ_x N/m$\hat{2}$
11	3.8082	0.0000	−84.728
13	10.0000	0.0000	−22.141
5	10.0000	2.0000	27.249
3	6.1918	2.0000	79.463
12	7.5000	0.0000	−53.435
8	10.0000	1.0000	2.554
4	7.5000	2.0000	53.356
7	5.0000	1.0000	−2.633
center	7.5000	1.0000	−0.039
Gauss point 1	6.0566	0.4226	−41.931
Gauss point 2	8.9434	0.4226	−19.803
Gauss point 3	8.9434	1.5774	22.719
Gauss point 4	6.0566	1.5774	38.858

Table 3.20: Matlab result. direct stress at each node, Second element, angle [50] degree

The following shows the direct stress contour generated in Matlab

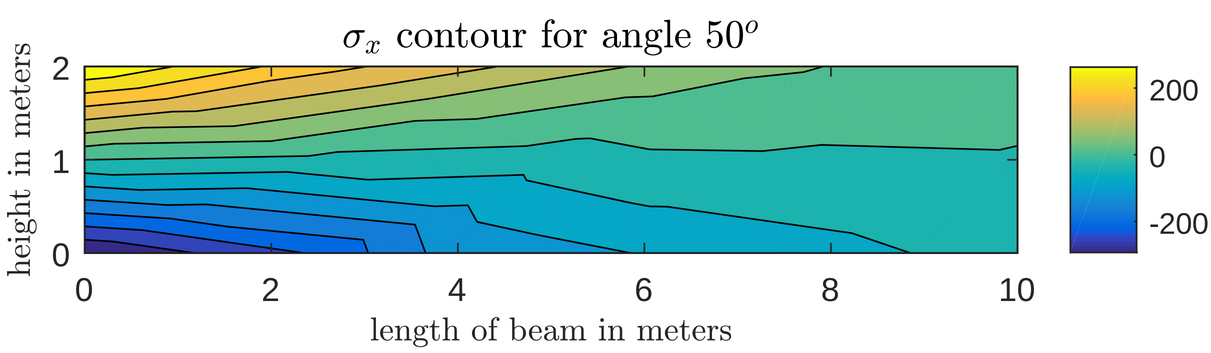


Figure 3.18: Contour of direct stress found using 2 elements using Matlab, 50 degrees

3.5.3 ANSYS result

```

PRINT U    NODAL SOLUTION PER NODE

 ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=      1  SUBSTEP=     1
  TIME=    1.0000       LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM

   NODE       UX          UY          UZ          USUM
      1   0.0000    -0.18726E-01  0.0000     0.18726E-01
      2  0.63489E-01-0.97037E-01  0.0000     0.11596
      3  0.10715    -0.42631      0.0000     0.43957
      4  0.11396    -0.58503      0.0000     0.59603
      5  0.11888    -0.86838      0.0000     0.87648
      6   0.0000      0.0000      0.0000      0.0000
      7  0.11328E-02-0.31108      0.0000     0.31108
      8  0.27311E-02-0.88375      0.0000     0.88376
      9   0.0000     0.93858E-02  0.0000     0.93858E-02
     10 -0.60301E-01-0.92294E-01  0.0000     0.11025
```

```
     11 -0.90400E-01-0.20071      0.0000      0.22013
     12 -0.11493    -0.57488      0.0000      0.58626
     13 -0.12980    -0.90156      0.0000      0.91085

MAXIMUM ABSOLUTE VALUES
NODE         13          13           0          13
VALUE  -0.12980    -0.90156      0.0000     0.91085

/OUTPUT FILE= ansys_stress_solution_50.txt
```

The following figure shows the deformation found

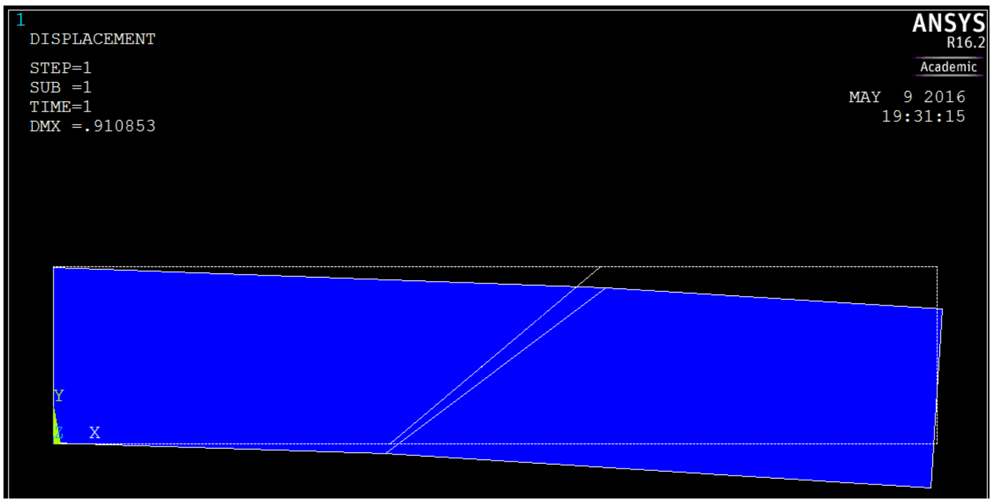


Figure 3.19: deflection found using 2 elements using ANSYS, 50 degrees

The following table shows ANSYS result for the direct stress σ_x found at each node for each element.

```

PRINT S    ELEMENT SOLUTION PER ELEMENT

***** POST1 ELEMENT NODAL STRESS LISTING *****

 LOAD STEP=      1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING X,Y,Z VALUES ARE IN GLOBAL COORDINATES


 ELEMENT=       1        PLANE183
   NODE    SX          SY          SZ          SXY         SYZ         SXZ
      9  -293.00      13.200      0.0000     -78.471      0.0000      0.0000
     11  -111.35     -11.561      0.0000     -17.821      0.0000      0.0000
      3   80.771     -46.122      0.0000     -27.702      0.0000      0.0000
      1   305.49      31.354      0.0000      61.078      0.0000      0.0000

 ELEMENT=       2        PLANE183
   NODE    SX          SY          SZ          SXY         SYZ         SXZ
     11  -84.728     -21.076      0.0000      26.004      0.0000      0.0000
     13  -22.141      9.8112      0.0000     -25.739      0.0000      0.0000
      5   27.249     -18.956      0.0000     0.41536      0.0000      0.0000
      3   79.463      57.025      0.0000     -26.399      0.0000      0.0000
```

The following shows the direct stress contour generated in ANSYS

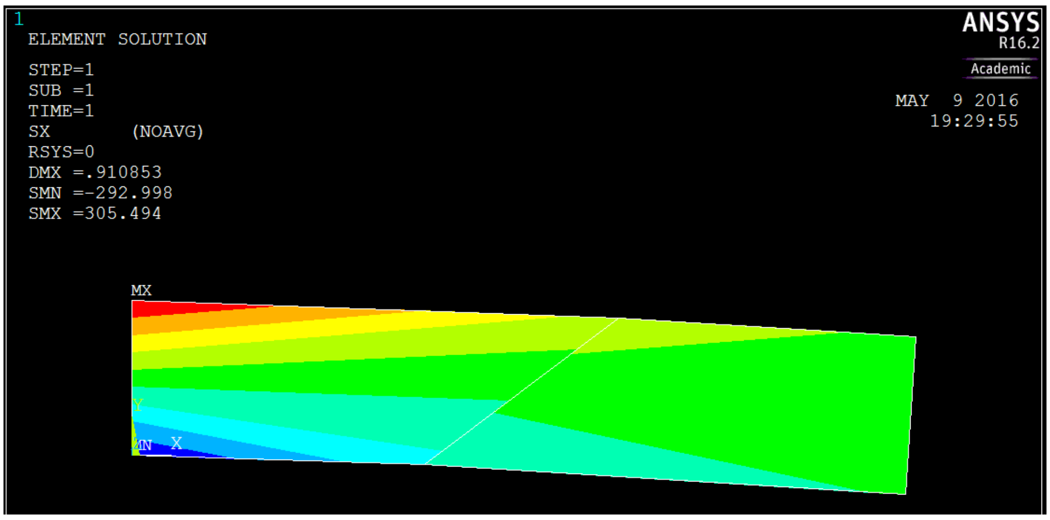


Figure 3.20: Contour of direct stress found using 2 elements using ANSYS, 50 degrees

3.6 55 degrees distortion

3.6.1 summary of result

	x (meter)	y (meter)
Matlab	−0.123001	−0.86157
ANSYS	−0.123	−0.86157

Table 3.21: Short summary of test case 55 degrees distortion

3.6.2 Matlab result

global node #	x (meter)	y (meter)
1	0.000000	−0.019943
2	0.062204	−0.097514
3	0.102304	−0.438312
4	0.107168	−0.562073
5	0.112704	−0.830774
6	0.000000	0.000000
7	0.000874	−0.305999
8	0.002574	−0.844631
9	0.000000	0.010255
10	−0.059355	−0.090697
11	−0.086450	−0.177473
12	−0.108133	−0.554224
13	−0.123001	−0.861570

Table 3.22: Matlab result. nodal solutions, angle [55] degree

The following figure shows the deformation found

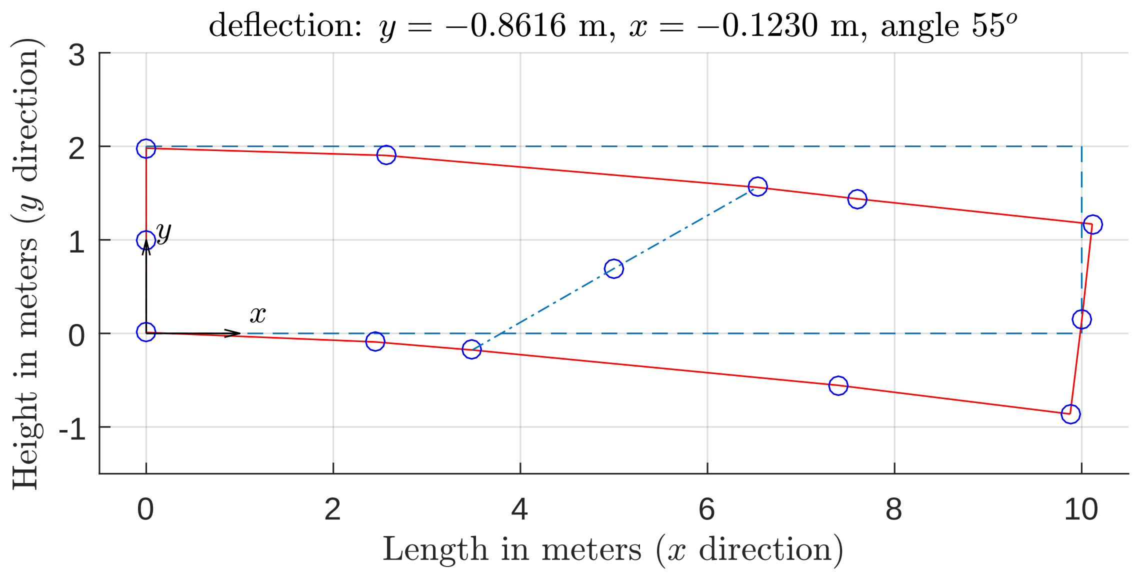


Figure 3.21: deflection found using 2 elements using Matlab, 55 degrees

The following table shows Matlab result for the direct stress σ_x found at each node for each element.

global node #	x	y	σ_x N/m$\hat{2}$
9	0.0000	0.0000	−290.606
11	3.5719	0.0000	−101.874
3	6.4281	2.0000	61.164
1	0.0000	2.0000	306.879
10	2.5000	0.0000	−196.240
7	5.0000	1.0000	−20.355
2	2.5000	2.0000	184.022
6	0.0000	1.0000	8.137
center	2.5000	1.0000	−6.109
Gauss point 1	1.0566	0.4226	−143.860
Gauss point 2	3.9434	0.4226	−87.902
Gauss point 3	3.9434	1.5774	59.234
Gauss point 4	1.0566	1.5774	148.092

Table 3.23: Matlab result. direct stress σ_x at each node, First element, angle [55] degree

global node #	x	y	σ_x N/m$\hat{2}$
11	3.5719	0.0000	−70.386
13	10.0000	0.0000	−27.809
5	10.0000	2.0000	32.546
3	6.4281	2.0000	69.339
12	7.5000	0.0000	−49.097
8	10.0000	1.0000	2.369
4	7.5000	2.0000	50.943
7	5.0000	1.0000	−0.523
center	7.5000	1.0000	0.923
Gauss point 1	6.0566	0.4226	−35.405
Gauss point 2	8.9434	0.4226	−20.508
Gauss point 3	8.9434	1.5774	24.022
Gauss point 4	6.0566	1.5774	35.581

Table 3.24: Matlab result. direct stress at each node, Second element, angle [55] degree

The following shows the direct stress contour generated in Matlab

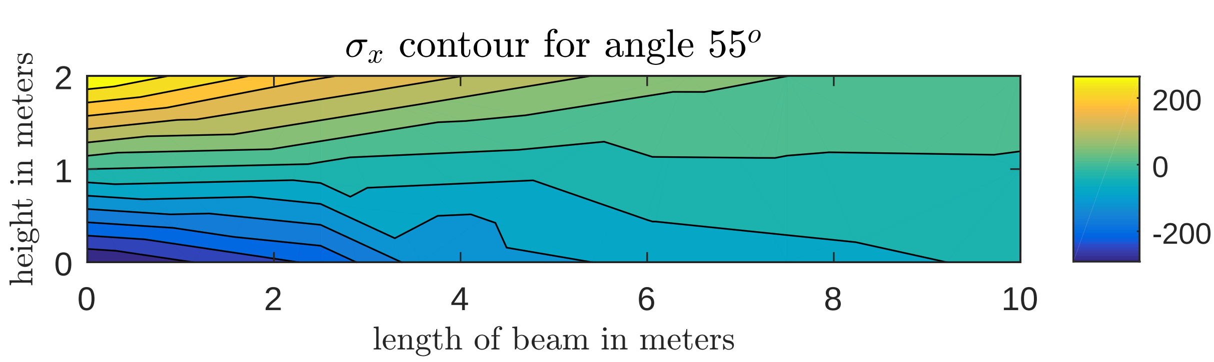


Figure 3.22: Contour of direct stress found using 2 elements using Matlab, 55 degrees

3.6.3 ANSYS result

```

PRINT U    NODAL SOLUTION PER NODE

 ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM

   NODE      UX          UY          UZ          USUM
      1   0.0000    -0.19943E-01  0.0000     0.19943E-01
      2  0.62204E-01-0.97514E-01  0.0000     0.11566
      3  0.10230    -0.43831      0.0000     0.45009
      4  0.10717    -0.56207      0.0000     0.57220
      5  0.11270    -0.83077      0.0000     0.83838
      6   0.0000      0.0000      0.0000      0.0000
      7  0.87390E-03-0.30600      0.0000     0.30600
      8  0.25744E-02-0.84463      0.0000     0.84463
      9   0.0000     0.10255E-01  0.0000     0.10255E-01
     10 -0.59355E-01-0.90697E-01  0.0000     0.10839
     11 -0.86450E-01-0.17747      0.0000     0.19741
     12 -0.10813    -0.55422      0.0000     0.56467
     13 -0.12300    -0.86157      0.0000     0.87031

MAXIMUM ABSOLUTE VALUES
NODE          13          13            0          13
VALUE  -0.12300    -0.86157      0.0000     0.87031

/OUTPUT FILE= ansys_stress_solution_55.txt
```

The following figures show the deformation found

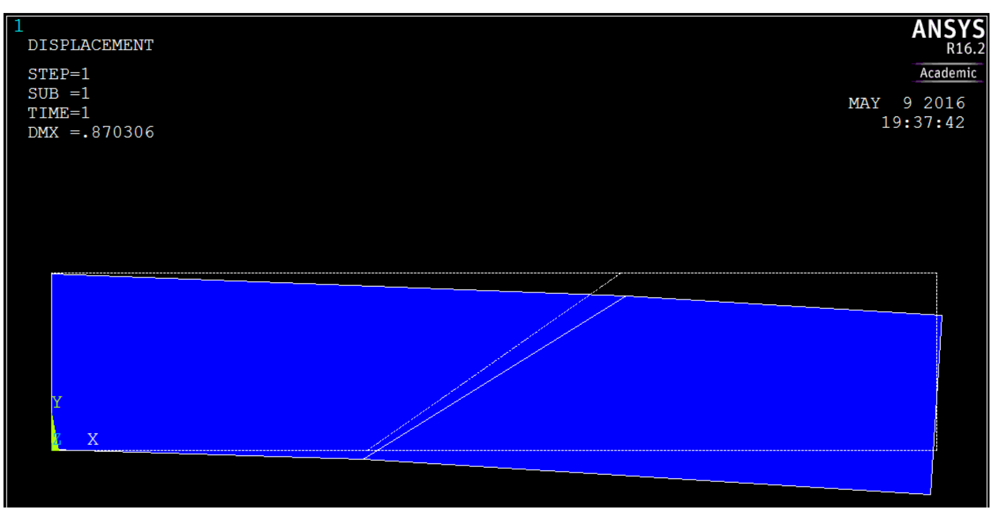


Figure 3.23: deflection found using 2 elements using ANSYS, 55 degrees

The following table shows ANSYS result for the direct stress σ_x found at each node for each element.

```

PRINT S    ELEMENT SOLUTION PER ELEMENT

***** POST1 ELEMENT NODAL STRESS LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  TIME=    1.0000      LOAD CASE=   0

 THE FOLLOWING X,Y,Z VALUES ARE IN GLOBAL COORDINATES

```

```

 ELEMENT=       1         PLANE183
   NODE     SX          SY          SZ          SXY         SYZ         SXZ
       9  -290.61      12.453      0.0000     -82.985      0.0000      0.0000
      11  -101.87     -10.078      0.0000     -19.122      0.0000      0.0000
       3   61.164     -49.512      0.0000     -32.211      0.0000      0.0000
       1   306.88      34.681      0.0000      65.950      0.0000      0.0000

 ELEMENT=       2         PLANE183
   NODE     SX          SY          SZ          SXY         SYZ         SXZ
      11  -70.386     -27.088      0.0000      23.673      0.0000      0.0000
      13  -27.809      13.053      0.0000     -24.279      0.0000      0.0000
       5   32.546     -23.115      0.0000     -3.3748      0.0000      0.0000
       3   69.339      69.307      0.0000     -15.921      0.0000      0.0000
```

The following shows the direct stress contour generated in ANSYS

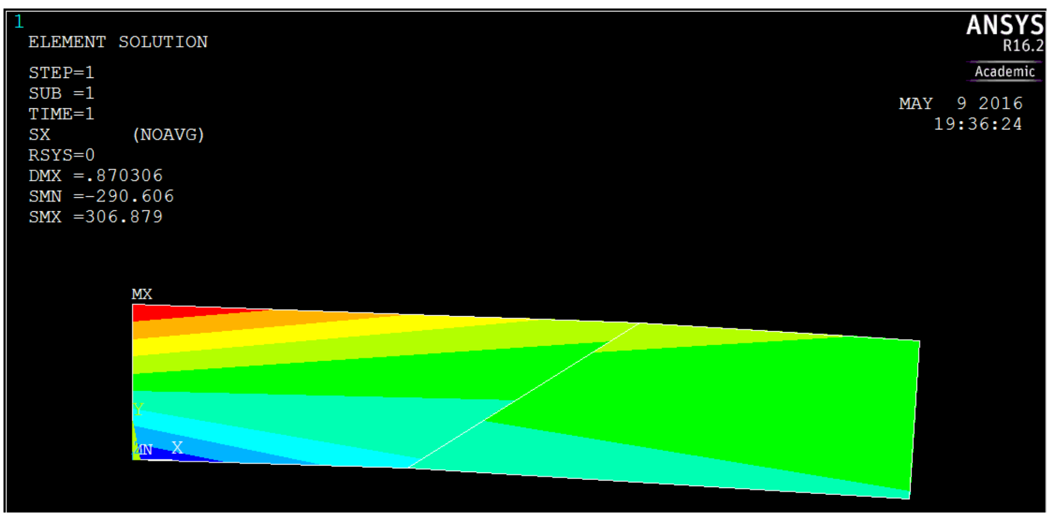


Figure 3.24: Contour of direct stress found using 2 elements using ANSYS, 55 degrees

Chapter 4

Observations, discussion and conclusions

It is clear from the above result that the 8 node element used did not behave well at all when it became distorted.

The element used is a serendipity element[1]. These elements are known not to give good results when they distorted[2]. There are a number of distortions that an element could have, such as an edge distortion or angular distortion and others. In this report we only looked at angular distortion.

As the angular distortion increased, the result became less accurate. The inaccuracy also accelerated when the angle was above 35^0 based on the diagram generated below. At angle 55^0, the vertical deflection reported by the finite element Matlab program (and ANSYS as well) was -0.86157 meters where the theoretical result should be close to -1.03 meters. This is over 16% error. A signification error in accuracy. The following graph shows how the error in the vertical deflection changed as a function of the distortion angle.

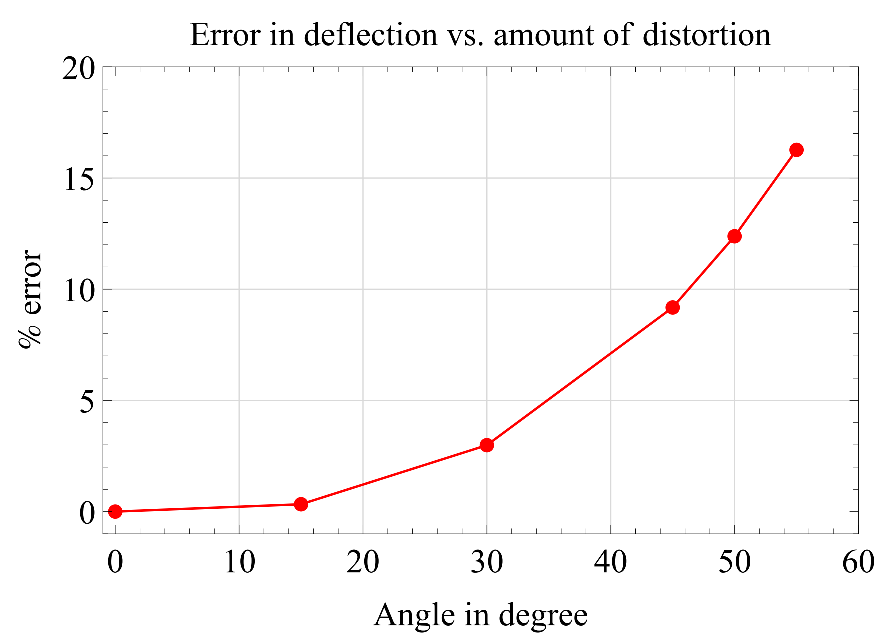


Figure 4.1: Error as function of amount of of angular element distortion

From the above one can see that to keep the error below 5%, the angular distortion should not be more than about 35^0 as the element behaves very badly after that. ANSYS will not even accept the analysis when trying angle of 60^0 and gave number of errors relating to element shape. This means this element is not suitable for modeling physical regions which are highly irregular. This element is suitable for fitting in physical region which has straight edges and mostly rectangular regions. Curved regions and other such regions would need to be modeled using other elements.

[1]Element has no node in the middle

[2]Reference [2]

When it comes to post processing and stress recovery, which the term used for calculating stresses in the post processing stage, there are three methods that can be used. These are

1. direct method. In this method the stress is found at arbitrary points by directly evaluating $\sigma = EBu$ at the point. Where in this, the u is the nodal solution. This requires finding the Jacobian at point in question. This method is simple and works very well as long as there is no distortion.

   Once distortion is added, it produced bad result in stress values when compared to ANSYS results.

2. Method of extrapolation. This method is described in reference [1], pages 230-232. In this method, the stress at the nodes of the element (or at any other arbitrary point) is found by extrapolating the stress values found at the four Gaussian points. This works well because the stress at the Gaussian points is the most accurate since these are also the integration points used. This was the method used in this project and worked very well. Results from the Matlab implementation all agreed with ANSYS result for the direct stress at the nodes.

3. Patch Recovery. This is based on using a polynomial of same order as the shape functions and then using such polynomial on a small patch around the point on interest to find the stress. It would be interesting to compare this method in the future with the extrapolation method to see which is more accurate or easier to implement.

When making the contour plots for σ_x, one can see that the stress along the same line between the two elements is no longer smooth as the angle increases. ANSYS shows this to be smooth transition in the stress contour, but this must be because ANSYS did averaging across element boundaries.

In the Matlab implementation, No stress averaging was made between nodes across elements, hence the distortion (contour lines) is more clear in the stress contour along the line between the two elements as the following plot shows when the angle is 55^0. This shows clearly that stress across elements is not smooth and changes abruptly now.

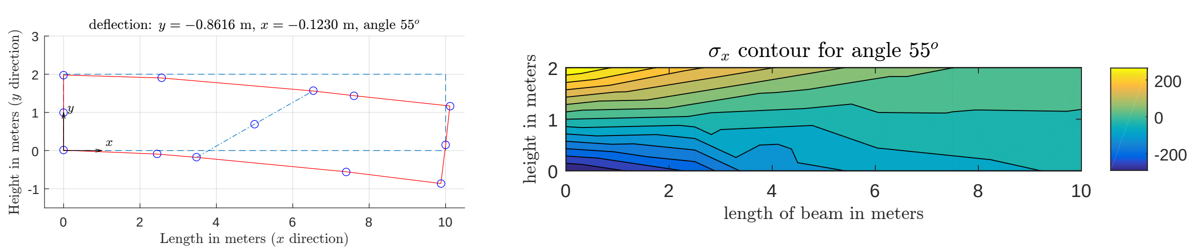


Figure 4.2: Stress contour, at 55 degrees

The more angular distortion there is, the sharper this distortion in the stress contour between the two element became. This is due to the element becoming less accurate as it distorts. Comparing the above plot to the one when the angle was zero (no distortion) one can see that in the no distortion case the stress across the elements is smooth and has same values at the nodes connecting the two elements.

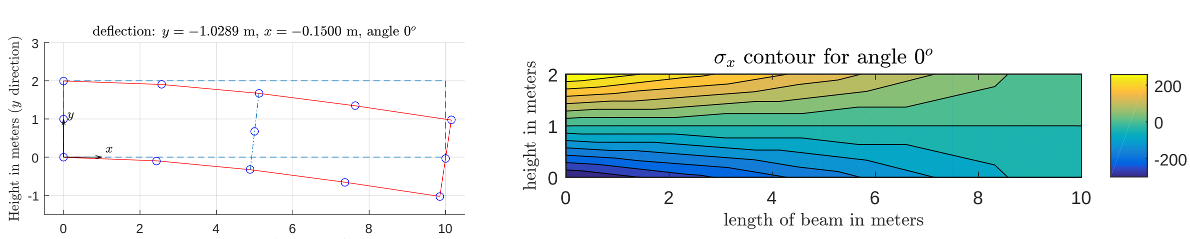


Figure 4.3: Stress contour, at zero degrees

In conclusion, it is recommended that this element be used only when there is no distortion in the geometry.

4.1 References

1 Concepts And Applications Of Finite Element Analysis. 4th edition. Robert D. Cook, David S. Malkus, Michael E. Plesha, Robert J. Witt. John Wiley & Sons. Inc.

2 Effect Of Element Distortions On The Performance Of Isoparametric Elements. Nam-Sua Lee, Klaus-Jurgen Bathe. Dept of Mechanical Engineering. MIT. International Journal For Numerical Methods In Engineering. Vol 36, 3553-3676 (1993)

3 ANSYS help manuals for APDL and general ANSYS use.

Chapter 5

Appendix

5.1 APDL used for ANSYS

The following is the APDL script used for ANSYS analysis. ANSYS version 16.02, student version was used.

```
!APDL script to generate solution for EMA 471 final project
!to use to compare result with my Matlab Finite element program
!Nasser M. Abbasi
!ANSYS 16.02

!To read this from the APDL mechanical, simply use the
!  FILE->read input from ...
!that is all. This will process eveything and will generate
!table of stresses and deformation into text files in the default
!directory and will plot the deformed shape on the screen

/CWD,'X:\data\public_html\my_courses\univ_wisconsin_madison\spring_2016\EMA_471
    \project\my_project\ansys'

/FILNAM,EMA_471_ansys_APDL
/title, EMA 471 final project, EMA option
/prep7

!KEYOPT(1)=0 (8 node), KEYOPT(3)=0 (plane stress), KEYOPT(6)=0 (pure
    displacement)
ET,1,PLANE183,0,,0,,,0        ! QUAD 8, plain stress plain strain element
MP, ex,   1, 1.0E4   !Elastic moduli
MP, prxy, 1, 0.3     !Major Poisson's ratios
MP, nuxy, 1, 0.3     !minor Poisson's ratios, same for isotropic

!define distortion angle. Change as needed. See report.
PI=ACOS(-1)
*SET,L,10.0  ! length of beam
*SET,H,2.0   ! depth of beam
*SET,angle,0.0*PI/180.0 ! angle, set it to any value needed
shift = 0.5*H*TAN(angle)!


! DEFINE ALL NODES, 13 of them.
N, 1, 0.0             , H   ,0      ! node 1, top left corner
N, 2, 0.25*L          , H   ,0      ! node 2, etc...
N, 3, 0.5*L+shift     , H   ,0      !
N, 4, 0.75*L          , H   ,0      !
N, 5, L               , H   ,0      ! last node in top of beam, node 5
N, 6, 0.0             , 0.5*H ,0    ! node 6, middle of left edge of beam.
N, 7, 0.5*L           , 0.5*H ,0
N, 8, L               , 0.5*H ,0
N, 9, 0.0             , 0.0   ,0     ! node 9, at origin, bottom left corner
```

```
N, 10, 0.25*L         , 0.0   ,0
N, 11, 0.5*L-shift    , 0.0   ,0
N, 12, 0.75*L         , 0.0   ,0
N, 13, L              , 0.0   ,0


MAT, 1
!real, 1,2,1
EN, 1, 9,11,3,1,10,7,2,6

MAT, 1
!real, 1,2,1
EN, 2, 11,13,5,3,12,8,4,7

!set degree of freedom.
D, 9, UX, 0.0
D, 6, UX, 0.0
D, 6, UY, 0.0
D, 1, UX, 0.0

F, 8, fy, -20.0

!ERESX,NO !do this to see GAUSSIAN points stress
finish

/solu
antyp, static
solve
SAVE EMA_471_ansys_APDL
finish

/post1

/OUTPUT,ansys_nodel_solution_0,txt
PRNSOL,U,COMP


/OUTPUT,ansys_stress_solution_0,txt
PRESOL,S,COMP
/OUTPUT


!I need to find out how to send image to a file to include it
!in report, plot deformation
/REPLOT
GPLOT
PLDISP,1

!Plot stress contour
PLESOL, S,X, 0,1.0

/output

!makes  SAVE EMA_471_ansys_APDL.dbb
SAVE EMA_471_ansys_APDL
```

5.2 Matlab source code

The following is the listing of the m file for Matlab implementation. For making the contour plot, an external m file was used from Mathworks file exchange called `tricontf` and is included in the zip file. This function is not listed here.

To run the Matlab program, the command is nma_project_EMA_471

```
function nma_project_EMA_471()
%Final project Finite Elements option, EMA option.
%By Nasser M. Abbasi
%EMA 471, spring 2016, Univ. Of Wisconsin, Madison
%This files uses a Mathworks file exchange function
%which is included in the zip file and is in the same folder
%as this m file. Needed for making the contour plot.

close all; clc;

data      = PRE_PROCESSOR();    %set up data structure
solution  = SOLVE(data);        %assemble and solve KD=F

%stress calculations and print results
POST_PROCESSOR(solution,data);

end
%============================================
function data = PRE_PROCESSOR()
%In this function we allocate the mapping tables and
%set all the problem parameters. All is saved in a struct data

%get folder name we are running from. Needed to save results
if(~isdeployed)
    data.baseFolder = fileparts(which(mfilename));
    cd(data.baseFolder);
end

%we are using 2 by 2 Gaussian rule for integration.
wt(1) = 1;           wt(2) = 1;
gs(1) = -0.57735027; gs(2)  = 0.57735027;

%allocate data parameters
data.wt = wt;
data.gs = gs;
data.num_elements = 2;
data.angle_degree = 0;  %change as needed
data.angle = data.angle_degree*pi/180;

%  global node numbering:
%
%  1     2      3       4         5
%  o-----o------o-------o---------o
%  |            |                 |
%  o 6          o7                o8
%  |            |                 |
%  o-----o------o-------o---------o
%  9     10     11     12         13
%
%
data.elem_map_nodes = [9,11,3,1,10,7,2,6;   %element (1)
                       11,13,5,3,12,8,4,7];  %element (2)

L          = 10;
H          = 2;
data.L     = L;  %meter
data.H     = H;   %meter
data.shift = (H/2)*tan(data.angle); %meter

data.global_coord_tbl = [ ...  %global coordinates of the 13 nodes
    0,          H;         %node 1, top left corner
    L/4,        H;         %node 2, etc...
```

```
    L/2+data.shift,  H;    %if angle is 90 degrees, then shift=0
    (3/4)*L,    H;
    L,          H;         %last node in top of beam, node 5
    0,          H/2;       %node 6, middle of left edge of beam.
    L/2,        H/2;
    L,          H/2;
    0,          0;         %node 9, at origin, bottom left corner
    L/4,        0;
    L/2-data.shift,  0;
    (3/4)*L,    0;
    L,          0];

data.young_module  = 10^4;
data.mu            = 0.3;
data.E0            = data.young_module/(1-data.mu^2)*...
                       [1,data.mu,0;data.mu,1,0;0,0,(1-data.mu)/2];

data.elem_map_dofs = zeros(data.num_elements ,16);
display_diagram_of_dof(data.L,data.H);

%elem_map_dofs is used to merge the element k to the global K
for i=1:data.num_elements
    for j=1:8
        data.elem_map_dofs(i,2*j-1)= 2*data.elem_map_nodes(i,j)-1;
        data.elem_map_dofs(i,2*j)  = 2*data.elem_map_nodes(i,j);
    end
end

end
%=====================================================
function solution = SOLVE(data)
%all the problem data description is now in struct data. This
%function assembles the stiffness matrix and solves KD=F
%and returns the solution

N        = size(data.global_coord_tbl,1);
k_global = zeros(2*N,2*N);

%do initial verification that shape functions adds to one.

%throws error if not verified
verify_shape_functions_sum_to_one(data.gs);
%fprintf('verified shape functions ok....\n');

for k = 1:data.num_elements
    %obtain global coordinates for the node of this element
    [x_coord,y_coord] = find_XY_coordinates(k,...
                      data.elem_map_nodes,data.global_coord_tbl);
    k_elem = zeros(16,16); %allocate local element k

    for i = 1:2  %we are using 2 by 2 Gaussian integration rule
        xi = data.gs(i);
        for j = 1:2
            eta   = data.gs(j);
            J     = get_J(xi,eta,x_coord,y_coord);
            detJ  = det(J);
            if detJ<0
                error('Internal error. negative |J| detected.');
            end
            B = get_B(xi,eta,J);  %find the strain rate matrix
            v = B'*data.E0*B;

            for ii = 1:16  %numerical integration
```

```
                for jj = 1:16
                    if(ii<=jj) %only do upper diagonal, symmetry
                        k_elem(ii,jj) = k_elem(ii,jj)+...
                            data.wt(i)*data.wt(j)*v(ii,jj)*detJ;
                    end
                end
            end

        end
    end

    %First copy the upper part of k to the lower part,  symmetric
    k_elem  = k_elem + triu(k_elem,1)';

    %Merge it to global stiffness matrix
    k_global = merge_element_to_global(k, k_elem, ...
                                   k_global,data.elem_map_dofs);
end

%we need now to zero out rows/cols {1,11,12,17} from the
%above, since these correspond to the fixed boundary conditions.
%Make sure to keep the diagonal element. Since the boundary
%conditions are zero at these, we do not have to patch the F
%vector on the RHS as normally we would. put a 1 in the diagonal

fix   = [1,11,12,17];  %these are the fixed DOF to zero out

%----- COMMENTED OUT ------------------------
%below is one method to fix. It gives same as the next method.
%But the next method below this is simpler since it keeps the
%same sizes of data kept for reference

%k_global(fix,:)=[];
%k_global(:,fix)=[];
%r_global       = zeros(22,1);
%r_global(end)  = -20;  % Newton
%d              = k_global\r_global;  %solve for displacement

%----- END COMMENTED --------------------

%second method to fix K

for i = 1:length(fix)
    tmp                      =  k_global(fix(i),fix(i));
    k_global(fix(i),:)       = 0;
    k_global(:,fix(i))       = 0;
    k_global(fix(i),fix(i)) = tmp; %same effect as setting to 1.
end

r_global      = zeros(26,1);
r_global(16)  = -20;  % Newton
solution      = k_global\r_global;  %solve for displacement

%finished. Now write the solution to file to include it in report
write_nodal_solution_to_file(solution,data.baseFolder,...
                                               data.angle_degree);

figure(); %show the global stiffness matrix structure using spy
spy(k_global);
title('global stiffness matrix after fixing for B.C.',...
    'interpreter','Latex','Fontsize',11);
%print(gcf, '-dpdf', '-r600','../images/spy.pdf');
```

```
end
%=======================================================
function POST_PROCESSOR(solution,data)
%This is final stage. We find stress and make contour plots
%and draw the deflection shape of the beam

draw_deflection(solution,data);
generate_stress_diagram(solution,data);
end
%========================================
function write_nodal_solution_to_file(d,baseFolder,angle)
%d is the nodal solution vector. 26 by 1.
%write the X,Y solution to text file to use in report

d               = reshape(d,2,13)';
fileName        = [baseFolder sprintf(...
                         '/data/deformation_matlab_%d.tex',angle)];
fileName        = strrep(fileName, '/', filesep);
[fileID0,errMsg] = fopen(fileName,'w');

if fileID0<0
    fprintf('Error opening %s\n, the message is [%s\n]',...
                                                fileName,errMsg);
    error(errMsg);
end

fprintf(fileID0,'\\begin{table}[!htbp]\n');
fprintf(fileID0,'\\centering\n');
fprintf(fileID0,'\\captionsetup{width=.8\\textwidth}\n');
fprintf(fileID0,'\\begin{tabular}{|l|l|l|}\\hline\n');
fprintf(fileID0,'global node \\#& $x$ (meter)& $y$ (meter)\\\\\\hline\n');
for i=1:size(d,1)
    fprintf(fileID0,'$%d$ & $%7.6f$& $%7.6f$\\\\ \n',i,d(i,1),d(i,2));
end
fprintf(fileID0,'\\hline\n\\end{tabular}\n');
fprintf(fileID0,...
  '\\caption{Matlab result. nodal solutions, angle [$%d$] degree}\n',angle);
fprintf(fileID0,'\\end{table}\n');
fprintf(fileID0,'\\FloatBarrier\n');

fclose(fileID0);

end
%===========================================
function generate_stress_diagram(solution,data)

%find the stress at node of each element
element_stress_1 = stress_calculation(solution,data,1);
element_stress_2 = stress_calculation(solution,data,2);

%make one diagram of the overall stress contour across the beam
stress_diagram(data,element_stress_1,element_stress_2);
end
%==============================================
function element_stress = stress_calculation(solution,data,...
                                                element_number)

%first calculate stress at the 4 Gaussian points for
%element in order to use for extrapolation. These are ordered
%anticlock wise.

%used to store stress at the 4 Gaussian points
gauss_stress = zeros(4,1);
```

```

%these are the r,s coordinates of the Gaussian element inside
%the element itself used for extrapolation. See report for
%more details. These are ordered anticlock wise, with one in
%the center. So there are 9 of them. Also, the Gaussian stress
%is added as well. So we end up with 9+4=13 total stress points
%for each element. This should be enough to make nice contuor with

z  = sqrt(3);
r  = [-z,z,z,-z,0,z,0,-z];
s  = [-z,-z,z,z,-z,0,z,0];

%these are the natural coordinates in xi,eta space used
%to calculate the stress at Gaussian points, using the full 8
%shape functions
z   =  1/sqrt(3);
xi  = [-z,z,z,-z];
eta = [-z,-z,z,z];

L  = data.L;
H  = data.H;

%this is used to find global coordinates of all stress points, for
%contour plot this is center of element in global space
if element_number == 1
    X0=L/4;
    Y0=H/2;
else
    X0=(3/4)*L;
    Y0=H/2;
end

%find element nodal coordinates in global space
%this only has nodes. We will add the center and the gaussian
%points also later to make contour plot
[x_coord,y_coord] = find_XY_coordinates(element_number,...
                     data.elem_map_nodes,data.global_coord_tbl);

%Now make matrix to store all stress result in for element 1.
%We are finding stress at 13 points. 8 for nodes, one for center,
%and the 4 Gaussian points. we need 4 columns. First two are
%the x,y in global space of the point, and the stress. The
%first column is just the point ID, for tracking. Not used for
%plotting.

element_stress = zeros(13,4);

global_node_numbers = data.elem_map_nodes(element_number,:);
U   = solution(data.elem_map_dofs(element_number,:));
for i = 1:length(xi)
    J    = get_J( xi(i), eta(i), x_coord, y_coord); %jacobian
    B    = get_B( xi(i), eta(i), J); %strain rate matrix
    %find actual strain from displacements at nodes
    strain  = B  * U;
    stress  = data.E0 * strain;
    gauss_stress(i) = stress(1); %use direct stress only
end

%Now do the extrapolation. See report
for i=1:length(r)
    element_stress(i,4)=0;
    element_stress(i,1)=global_node_numbers(i);
    element_stress(i,2)=x_coord(i);
```

```
    element_stress(i,3)=y_coord(i);
    for j=1:4 %extrapolation
        switch j
            case 1,
                f = (1/4)*(1-r(i))*(1-s(i));
            case 2,
                f = (1/4)*(1+r(i))*(1-s(i));
            case 3,
                f = (1/4)*(1+r(i))*(1+s(i));
            case 4,
                f = (1/4)*(1-r(i))*(1+s(i));
        end
        element_stress(i,4) = element_stress(i,4)+ f * ...
                                                     gauss_stress(j);
    end
end

%now add the center and the 4 gaussian points we found before.
%These come after the nodes. This is in order to improve the
%contour plot by having more points.
element_stress(9,4) = 0;
element_stress(9,1) = -1;  %we do not have a global node number
                           %for this. this is just a place holder
element_stress(9,2)=X0;
element_stress(9,3)=Y0;
for j=1:4 %extrapolation
  element_stress(9,4)= element_stress(9,4)+ 1/4 * gauss_stress(j);
end

%now add the acutal Gaussian stress found. This make it up to
%13 points first find the global coordinates of the element
%gaussian points
z = 1/sqrt(3);
gauss_global_coordinates=[X0-z*(1/4)*L,Y0-z*(1/2)*H;...
    X0+z*(1/4)*L,Y0-z*(1/2)*H;...
    X0+z*(1/4)*L,Y0+z*(1/2)*H;...
    X0-z*(1/4)*L,Y0+z*(1/2)*H];

for i=1:length(gauss_global_coordinates)
    element_stress(9+i,4)=gauss_stress(i);
    element_stress(9+i,1)=-1; %place holder
    element_stress(9+i,2)=gauss_global_coordinates(i,1);
    element_stress(9+i,3)=gauss_global_coordinates(i,2);
end

end
%===========================================
function stress_diagram(data,element_stress_1,element_stress_2)

%we are done!  Now we can make contour of first element stress
%This uses tricontf, which is a mathworks file exchange file
%since Matlab does not have such a function build in
figure;

% I commented out the stress average last minute. I think it is
%better NOT to do stress averging across elements, in order
%to more clearly see the difference. I left the code here for
%reference in case need to use it later

%----------- COMMENTED OUT ---------------------
%now do stress avergaing on the nodes that are between
%element one and two. These nodes have global node
%numbers of 2,7,11 which correspond to local nodes
```

```
% 2,6,3 for first element and nodes 1,8,4 for second element.

%element_stress_1(2,4) = (element_stress_1(2,4)+element_stress_2(1,4))/2;
%element_stress_1(6,4) = (element_stress_1(6,4)+element_stress_2(8,4))/2;
%element_stress_1(3,4) = (element_stress_1(3,4)+element_stress_2(4,4))/2;

%now that we averaged the stress, remove these entry from the second
%element before merging, since it is duplicate
%element_stress_2 = element_stress_2([2:3,5:7,9:end],:);

%----------- END COMMENTED OUT ----------------

x = [element_stress_1(:,2);element_stress_2(:,2)];
y = [element_stress_1(:,3);element_stress_2(:,3)];
z = [element_stress_1(:,4);element_stress_2(:,4)];
M = delaunay(x,y);
max_stress = max(z);
min_stress = min(z);
range_of_stress = linspace(min_stress,max_stress,15);

%this below uses mathworks file exchange function.
%It is in the same folder
[~,h]=tricontf(x,y,M,z,range_of_stress,'-k');
%set(h,'edgecolor','none');
axis equal tight;
%hold on;
%[~,h]=tricont(x,y,M,z,range_of_stress,'-k');
colorbar;

title(sprintf('$\\sigma_x$ contour for angle $%d^o$',...
                                           data.angle_degree),...
                            'Fontsize',12,'interpreter','Latex');

xlabel('length of beam in meters','Fontsize',10,'interpreter','Latex');
ylabel('height in meters','interpreter','Latex','Fontsize',10);

%uncomment to write the plot
%print(gcf, '-dpdf', '-r600',...
%    sprintf('../images/stress_matlab_%d.pdf',data.angle_degree));

write_the_stress_table(data,element_stress_1,element_stress_2);

end
%===========================================
function write_the_stress_table(data,element_stress_1,...
                                                element_stress_2)

fileName         = [data.baseFolder sprintf(...
                                 '/data/stress_matlab_%d.tex',...
                    data.angle_degree)];
fileName         = strrep(fileName, '/', filesep);
[fileID0,errMsg] = fopen(fileName,'w');

if fileID0<0
    fprintf('Error opening %s\n, the message is [%s\n]',...
                                                fileName,errMsg);
    error(errMsg);
end

fprintf(fileID0,'\\begin{table}[!htbp]\n');
fprintf(fileID0,'\\centering\n');
fprintf(fileID0,'\\begin{minipage}{0.49\\textwidth}\n');
fprintf(fileID0,'\\centering\n');
```

```
fprintf(fileID0,'\\captionsetup{width=.95\\textwidth}\n');
fprintf(fileID0,'\\begin{tabular}{|l|l|l|l|}\\hline\n');
fprintf(fileID0,['global node \\# & $x$ & $y$ & $\\sigma_x$',...
                 '{\\footnotesize N/m^2} \\\\\\hline\n']);
for i=1:size(element_stress_1,1)
    if i==9
        fprintf(fileID0,...
               'center & $%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n',...
               element_stress_1(i,2),element_stress_1(i,3),...
               element_stress_1(i,4));
    elseif i==10
        fprintf(fileID0,['{\\footnotesize Gauss point 1}&',...
                       '$%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n'],...
            element_stress_1(i,2),element_stress_1(i,3),...
                                           element_stress_1(i,4));
    elseif i==11
        fprintf(fileID0,['{\\footnotesize Gauss point 2}&',...
                      '$%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n'],...
            element_stress_1(i,2),element_stress_1(i,3),...
                                           element_stress_1(i,4));
    elseif i==12
        fprintf(fileID0,['{\\footnotesize Gauss point 3}&',...
                        '$%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n'],...
            element_stress_1(i,2),element_stress_1(i,3),...
            element_stress_1(i,4));
    elseif i==13
        fprintf(fileID0,['{\\footnotesize Gauss point 4}&',...
                       '$%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n'],...
            element_stress_1(i,2),element_stress_1(i,3),...
            element_stress_1(i,4));
    else
        fprintf(fileID0,'$%d$ & $%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n',...
            element_stress_1(i,1),element_stress_1(i,2),...
            element_stress_1(i,3),...
            element_stress_1(i,4));
    end
end
fprintf(fileID0,'\\hline\n\\end{tabular}\n');
fprintf(fileID0,...
    ['\\caption{Matlab result. direct stress $\\sigma_x$ at',...
     ' each node, First element, angle [$%d$] degree}\n'],...
    data.angle_degree);
fprintf(fileID0,'\\end{minipage}\n');
fprintf(fileID0,'\\hfill\n');

fprintf(fileID0,'\\begin{minipage}{0.49\\textwidth}\n');
fprintf(fileID0,'\\centering\n');
fprintf(fileID0,'\\captionsetup{width=.95\\textwidth}\n');

fprintf(fileID0,'\\begin{tabular}{|l|l|l|l|}\\hline\n');
fprintf(fileID0,['global node \\# & $x$ & $y$ & $\\sigma_x$',...
                       ' {\\footnotesize N/m^2} \\\\\\hline\n']);
for i=1:size(element_stress_2,1)
    if i==9
        fprintf(fileID0,'center & $%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n',...
           element_stress_2(i,2),element_stress_2(i,3),...
           element_stress_2(i,4));
    elseif i==10
        fprintf(fileID0,['{\\footnotesize Gauss point 1}&',...
                      ' $%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n'],...
            element_stress_2(i,2),element_stress_2(i,3),...
            element_stress_2(i,4));
    elseif i==11
```

```
        fprintf(fileID0,['{\\footnotesize Gauss point 2}&',...
                       ' $%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n'],...
            element_stress_2(i,2),element_stress_2(i,3),...
            element_stress_2(i,4));
    elseif i==12
        fprintf(fileID0,['{\\footnotesize Gauss point 3}&',...
                        '$%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n'],...
            element_stress_2(i,2),element_stress_2(i,3),...
            element_stress_2(i,4));
    elseif i==13
        fprintf(fileID0,['{\\footnotesize Gauss point 4}&',...
                       ' $%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n'],...
            element_stress_2(i,2),element_stress_2(i,3),...
            element_stress_2(i,4));
    else
        fprintf(fileID0,'$%d$ & $%4.4f$ & $%4.4f$ & $%5.3f$ \\\\ \n',...
            element_stress_2(i,1),element_stress_2(i,2),...
            element_stress_2(i,3),...
            element_stress_2(i,4));
    end
end
fprintf(fileID0,'\\hline\n\\end{tabular}\n');

fprintf(fileID0,...
    ['\\caption{Matlab result. direct stress at each node,',...
     ' Second element, angle [$%d$] degree}\n'],...
      data.angle_degree);
fprintf(fileID0,'\\end{minipage}\n');
fprintf(fileID0,'\\end{table}\n');
fprintf(fileID0,'\\FloatBarrier\n');

fclose(fileID0);

end
%========================================
function B = get_B(xi,eta,J)
%calculate the B matrix

B1 = [1,0,0,0;
      0,0,0,1;
      0,1,1,0];

gamma = 1/det(J) * [J(2,2) ,-J(1,2);
                    -J(2,1) ,J(1,1)];

B2    = [gamma, zeros(2,2);
         zeros(2,2),gamma];

Z  = zeros(2,1);

N1 = [dfdxi(1,xi,eta);
      dfdeta(1,xi,eta)];

N2 = [dfdxi(2,xi,eta);
      dfdeta(2,xi,eta)];

N3 = [dfdxi(3,xi,eta);
      dfdeta(3,xi,eta)];

N4 = [dfdxi(4,xi,eta);
      dfdeta(4,xi,eta)];

N5 = [dfdxi(5,xi,eta);
```

```
      dfdeta(5,xi,eta)];

N6 = [dfdxi(6,xi,eta);
      dfdeta(6,xi,eta)];

N7 = [dfdxi(7,xi,eta);
      dfdeta(7,xi,eta)];

N8 = [dfdxi(8,xi,eta);
      dfdeta(8,xi,eta)];

B3 = [N1,Z,  N2,Z,  N3,Z, N4,Z, N5,Z, N6,Z, N7,Z, N8,Z;
      Z,N1,  Z,N2,  Z,N3, Z,N4, Z,N5, Z,N6, Z,N7, Z,N8];

B  = B1*B2*B3;
end
%===========================================
function [x_coord,y_coord] = find_XY_coordinates(k,...
                                               elem_map_node,...
                                               global_coord_tbl)
%This function returns the x,y global coordinates of
%specific element nodes

N       = size(elem_map_node,2);  %number of nodes in element
x_coord = zeros(N,1);  %x for this element
y_coord = zeros(N,1);  %y for this element

%collect this element node coordinates, go over each node
%of this element and find its global x,y coordinates
for i = 1:N
  global_node_of_this_element_node = elem_map_node(k,i);
  x_coord(i) = global_coord_tbl(global_node_of_this_element_node,1);
  y_coord(i) = global_coord_tbl(global_node_of_this_element_node,2);
end
end
%=======================================
function J = get_J(xi,eta,x_coord,y_coord)

J = [ ddxi(xi,eta,x_coord),  ddxi(xi,eta,y_coord);
      ddeta(xi,eta,x_coord), ddeta(xi,eta,y_coord)];

end
%===================================
function v = ddxi(xi,eta,c)
%find dx/d(xi) or dy/d(xi)
v = c(1)*(1/4*(1-eta^2)+(1/2)*(1-eta)*xi+(eta-1)/4)...
    +c(2)*(1/4*(eta^2-1)+(1/2)*(1-eta)*xi+(1-eta)/4)...
    +c(3)*(1/4*(eta^2-1)+(1/2)*(eta+1)*xi+(eta+1)/4)...
    +c(4)*(1/4*(1-eta^2)+(1/2)*(eta+1)*xi+(-eta-1)/4)...
    -c(5)*(1-eta)*xi...
    +c(6)*(1/2)*(1-eta^2)...
    -c(7)*xi*(eta+1)...
    -c(8)*(1/2)*(1-eta^2);
end
%===================================
function v = ddeta(xi,eta,c)
%find dx/d(eta) or dy/d(eta)
v = c(1)*(1/2*eta*(1-xi)+(1/4)*(1-xi^2)+(xi-1)/4)...
    +c(2)*((1/2)*(1+xi)*eta+1/4*(1-xi^2)+(-xi-1)/4)...
    +c(3)*((1/2)*eta*(1+xi)+1/4*(xi^2-1)+(xi+1)/4)...
    +c(4)*((1/2)*(1-xi)*eta+1/4*(xi^2-1)+(1-xi)/4)...
    -c(5)*(1/2)*(1-xi^2)...
    -c(6)*eta*(xi+1)...
```

```
    +c(7)*(1/2)*(1-xi^2)...
    -c(8)*eta*(1-xi);
end
%==================================
function v = dfdxi(shape_function_number,xi,eta)
%evaluate shape function at some x,y point
switch shape_function_number
    case 1
        v=(1/4)*(1-eta^2)+(1/2)*(1-eta)*xi+(eta-1)/4;
    case 2
        v=(1/4)*(eta^2-1)+(1/2)*(1-eta)*xi+(1-eta)/4;
    case 3
        v=(1/4)*(eta^2-1)+(1/2)*(eta+1)*xi+(eta+1)/4;
    case 4
        v=(1/4)*(1-eta^2)+(1/2)*(eta+1)*xi+(-eta-1)/4;
    case 5
        v=-(1-eta)*xi;
    case 6
        v=(1/2)*(1-eta^2);
    case 7
        v=-(eta+1)*xi;
    case 8
        v=(1/2)*(eta^2-1);
end

end
%==================================
function v = dfdeta(shape_function_number,xi,eta)
switch shape_function_number
    case 1
        v=(1/2)*eta*(1-xi)+(1/4)*(1-xi^2)+(xi-1)/4;
    case 2
        v=(1/2)*eta*(xi+1)+(1/4)*(1-xi^2)+(-xi-1)/4;
    case 3
        v=(1/2)*eta*(xi+1)+(1/4)*(xi^2-1)+(xi+1)/4;
    case 4
        v=(1/2)*eta*(1-xi)+(1/4)*(xi^2-1)+(-xi+1)/4;
    case 5
        v=(1/2)*(xi^2-1);
    case 6
        v=-eta*(xi+1);
    case 7
        v=(1/2)*(1-xi^2);
    case 8
        v=-eta*(1-xi);
end

end
%=============================
function k_global = merge_element_to_global(k,k_elem,...
                                         k_global,elem_map_dofs)

%assemble local element k to global K

for i=1:16
    for j =1:16
       global_i =  elem_map_dofs(k,i);
       global_j =  elem_map_dofs(k,j);
       k_global(global_i,global_j) = k_global(global_i,global_j)...
                                                    + k_elem(i,j);
    end
end

```

```
end
%======================
function draw_deflection(solution,data)
%This function is called at the end, after we have solved
%the problem using finite elements and have nodal (x,y)
%deformations. It plots the deformed shape against the undeformed
%original shape.

u = reshape(solution,2,13)';
L = data.L;
H = data.H;

figure();
%This is before demformation shape

top_line    = [0,L/4,L/2+data.shift,(3/4)*L,L;H,H,H,H,H];
left_line   = [0,0,0;H,H/2,0];
right_line  = [L,L,L;H,H/2,0];
bottom_line = [0,L/4,L/2-data.shift,(3/4)*L,L;0,0,0,0,0];

line(top_line(1,:),top_line(2,:),'LineStyle','--'); hold on;
line(left_line(1,:),left_line(2,:),'LineStyle','--');
line(right_line(1,:),right_line(2,:),'LineStyle','--');
line(bottom_line(1,:),bottom_line(2,:),'LineStyle','--');

%this is after adding deformation
line(top_line(1,:)+u(1:5,1)',top_line(2,:)+u(1:5,2)',...
                                                'Color','red');
line(left_line(1,:)+u([1 6 9],1)',left_line(2,:)+u([1 6 9],2)',...
                                                    'Color','red');
line(right_line(1,:)+u([5 8 13],1)',right_line(2,:)+...
                                   u([5 8 13],2)','Color','red');
line(bottom_line(1,:)+u(9:13,1)',bottom_line(2,:)+...
                                       u(9:13,2)','Color','red');

%There are the nodes. Draw nodes on top line
for i=1:size(top_line,2)
    plot(top_line(1,i)+u(i,1),top_line(2,i)+u(i,2),'bo');
end

%draw nodes on left
idx=5;
plot(left_line(1,2)+u(1+idx,1),left_line(2,2)+u(1+idx,2),'bo');

%draw nodes on right
idx=7;
plot(right_line(1,2)+u(1+idx,1),right_line(2,2)+u(1+idx,2),'bo');

%draw nodes in middle
idx=6;
plot(L/2+u(1+idx,1),H/2+u(1+idx,2),'bo');

%Draw nodes on bottom line
idx=8;
for i=1:size(bottom_line,2)
    plot(bottom_line(1,i)+u(i+idx,1),bottom_line(2,i)+...
                                                u(i+idx,2),'bo');
end

%draw dashed line between elements
line([bottom_line(1,3)+u(11,1),...
      L/2+u(7,1),...
      L/2+data.shift+u(3,1)],...
```

```
      [bottom_line(2,3)+u(11,2),...
      H/2+u(7,2),...
      H+u(3,2)],...
      'LineStyle','-.');

%put title, x,y arrows at (0,0) and save the image to include in
%document/report at end

title(sprintf('deflection: $y=%3.4f$ m, $x=%3.4f$ m, angle $%d^o$',...
    u(end,2),u(end,1),data.angle_degree),...
    'interpreter','Latex','Fontsize',11);

xlabel('Length in meters ($x$ direction)','interpreter',...
                                          'Latex','Fontsize',11);
ylabel('Height in meters ($y$ direction)','interpreter',...
                                          'Latex','Fontsize',11);

quiver(0,0,1,0,1,'MaxHeadSize',0.5,'Color','black');
text(1.1,.2,'$x$','interpreter','Latex');
quiver(0,0,0,1,1,'MaxHeadSize',0.5,'Color','black');
text(0.1,1.1,'$y$','interpreter','Latex');
axis equal;
xlim([-0.5,L+0.5]);
ylim([-1.5,H+1]);
grid;

%print(gcf, '-dpdf', '-r600',...
%sprintf('../images/deflection_matlab_%d.pdf',data.angle_degree));

end

%===============================
function v = get_shape_function(shape_function_number,xi,eta)

switch shape_function_number
    case 1
        v=-(1/4)*(1-eta^2)*(1-xi)-(1/4)*(1-eta)*(1-xi^2)+...
                                            (1/4)*(1-eta)*(1-xi);
    case 2
        v=-(1/4)*(1-eta^2)*(1+xi)-(1/4)*(1-eta)*...
                                   (1-xi^2)+(1/4)*(1-eta)*(1+xi);
    case 3
        v=-(1/4)*(1-eta^2)*(1+xi)-(1/4)*(1+eta)*...
                                   (1-xi^2)+(1/4)*(1+eta)*(1+xi);
    case 4
        v=-(1/4)*(1-eta^2)*(1-xi)-(1/4)*(1+eta)*...
                                   (1-xi^2)+(1/4)*(1+eta)*(1-xi);
    case 5
        v=(1/2)*(1-eta)*(1-xi^2);
    case 6
        v=(1/2)*(1-eta^2)*(1+xi);
    case 7
        v=(1/2)*(1+eta)*(1-xi^2);
    case 8
        v=(1/2)*(1-eta^2)*(1-xi);
end

end
%=======================================
function verify_shape_functions_sum_to_one(gs)
for i=1:2
    xi = gs(i);
    for j=1:2
```

```
        eta   = gs(j);
        chk_1 = 0;
        for k=1:8 %sum all shape functions at this Gaussian point
            chk_1 = chk_1 + get_shape_function(k,xi,eta);
        end
        if chk_1 ~= 1
            error(['Internal error. sum of shape functions',...
                   ' not 1 at $\xi=%3.3f,\eta=%3.3f'],...
                xi,eta);
        end
    end
end
end
%================================
function display_diagram_of_dof(L,H)

figure();
top_line    = [0,L/4,L/2,(3/4)*L,L;
    H,H,H,H,H];
left_line   = [0,0,0;
    H,H/2,0];
right_line  = [L,L,L;
    H,H/2,0];
bottom_line = [0,L/4,L/2,(3/4)*L,L;
    0,0,0,0,0];
middle_line = [L/2;H/2];

line(top_line(1,:),top_line(2,:)); hold on;
%axis equal;
xlim([-2,L+2]);
ylim([-.75,H+1]);
line(left_line(1,:),left_line(2,:));
line(right_line(1,:),right_line(2,:));
line(bottom_line(1,:),bottom_line(2,:));

k=0;
node_number=0;
for i=1:size(top_line,2)
    x=top_line(1,i); y=top_line(2,i);
    plot(x,y,'ro');
    quiver(x,y,0.5,0,1,'MaxHeadSize',2,'Color','black');
    k=k+1;
    node_number=node_number+1;
    text(x+.1,y-.1,sprintf('$%d$',node_number),...
                                       'interpreter','Latex',...
        'Fontsize',11,'Color','red');
    text(x+.3,y+.1,sprintf('$%d$',k),'interpreter',...
                                         'Latex','Fontsize',11);
    quiver(x,y,0,0.5,1,'MaxHeadSize',2,'Color','black');
    k=k+1;
    text(x,y+.6,sprintf('$%d$',k),'interpreter',...
                                          'Latex','Fontsize',11);
end
for i=1:size(left_line,2)
    x=left_line(1,i); y=left_line(2,i);
    plot(x,y,'ro');
    quiver(x,y,0.5,0,1,'MaxHeadSize',2,'Color','black');
    quiver(x,y,0,0.5,1,'MaxHeadSize',2,'Color','black');
end
k=k+1;
node_number=node_number+1;
text(.1,H/2-.1,sprintf('$%d$',node_number),'interpreter','Latex',...
    'Fontsize',11,'Color','red');
```

```
text(.5,H/2,sprintf('%d',k),'interpreter','Latex','Fontsize',11);
k=k+1;
text(x,H/2+.5,sprintf('%d',k),'interpreter','Latex','Fontsize',11);

for i=1:size(middle_line,2)
    x=middle_line(1,i); y=middle_line(2,i);
    plot(x,y,'ro');
    quiver(x,y,0.5,0,1,'MaxHeadSize',2,'Color','black');
    k=k+1;
    node_number=node_number+1;
    text(x+.1,H/2-.15,sprintf('$%d$',node_number),...
                                        'interpreter','Latex',...
        'Fontsize',11,'Color','red');
    text(x+.5,H/2,sprintf('%d',k),'interpreter',...
                                          'Latex','Fontsize',11);
    quiver(x,y,0,0.5,1,'MaxHeadSize',2,'Color','black');
    k=k+1;
    text(x+.1,H/2+.4,sprintf('%d',k),'interpreter',...
                                          'Latex','Fontsize',11);
end

for i=1:size(right_line,2)
    x=right_line(1,i); y=right_line(2,i);
    plot(x,y,'ro');
    quiver(x,y,0.5,0,1,'MaxHeadSize',2,'Color','black');
    quiver(x,y,0,0.5,1,'MaxHeadSize',2,'Color','black');
end
k=k+1;
node_number=node_number+1;
text(L+.1,H/2-.1,sprintf('$%d$',node_number),...
          'interpreter','Latex','Fontsize',11,'Color','red');
text(L+.5,H/2,sprintf('%d',k),'interpreter','Latex','Fontsize',11);
k=k+1;
text(L,H/2+.5,sprintf('%d',k),'interpreter','Latex','Fontsize',11);


for i=1:size(bottom_line,2)
    x=bottom_line(1,i); y=bottom_line(2,i);
    plot(x,y,'ro');
    quiver(x,y,0.5,0,1,'MaxHeadSize',2,'Color','black');
    k=k+1;
    node_number=node_number+1;
    text(x-.1,-.2,sprintf('$%d$',node_number),...
                                       'interpreter','Latex',...
        'Fontsize',11,'Color','red');
    text(x+.4,.1,sprintf('%d',k));
    quiver(x,y,0,0.5,1,'MaxHeadSize',2,'Color','black');
    k=k+1;
    text(x+.1,.4,sprintf('%d',k));
end

title({'global D.O.F. numbering used (black letters).',...
    'with associated global node numering (in red letters)'},...
    'interpreter','Latex','Fontsize',11);

%
%print(gcf, '-dpdf', '-r600', sprintf('../images/dof.pdf'));
%
end
```