MECH0059: Advanced Computer Applications in Engineering

Finite Element Analysis - Assignment

Hasha Dar[1]

[1]Department of Mechanical Engineering, University College London, Torrington Place, London WC1E 7JE, UK

Student number: 19090799

January 25, 2023

Word count:

Abstract: This assignment investigates the use of finite element analysis (FEA) to analyse a 2-D plate with a notch-shaped indentation under a plane stress condition. The aim of the assignment is to understand how commercial FEA packages operate from a low-level mathematical perspective. I utilise the finite element method for this analysis to determine the displacement and strain for a variety of Young's Moduli and loading conditions. The assignment stipulates to build a MATLAB programme to compute these variables. To investigate the accuracy of the programme, I replicate the problem in a commercially available FEA package, ANSYS Mechanical APDL, and gather the results for comparison and discussion.

Key words: finite element analysis, finite element method, 2-D plate, plane stress

Introduction: Finite element analysis (FEA) is used to analyse the behaviour of components under certain conditions using the finite element method (FEM). This analysis encompasses a broad scope of physical phenomena, for example, deformation (elastic and plastic), modal analysis and thermal conduction [1]. Many of these problems involve the formulation of partial differential equations that are difficult to solve analytically, if at all possible. The introduction of the finite element method has since been able to solve these problems through discretisation [2]. This is where a domain may be split up through the construction of a mesh into a number of elements, consisting of an assemblage of nodes. Hence, by defining the boundary conditions of a specific problem, a series of algebraic equations are generated, which can be solved numerically. Originally conceived to aid in structural analysis, FEM has widespread use in other areas of computer aided engineering (CAE), such as biomechanics and electromagnetic field problems [3]. We must also note that the use of the FEM to conduct these analyses is not a precise solution, rather a numerical approximation of the underlying partial differential equations that may define a physical system.

Due to the complexity and large size of the calculations typical in FEA, historically FEA packages have been developed specifically for use by commercial server grade computers i.e., commercial mainframes and super-computer clusters. In the last decade, the computing power available on personal computers has risen to a level permitting the use of commercial FEA packages [4]. This assignment aims to create a bare-bones programme within MATLAB to calculate the nodal displacements and strains for a thin plate.

Methods:

Geometry and material properties: A 2-D plate with a notch indentation thin plate ($t = 2\,\text{mm}$) under a plane stress condition was used for the purposes of this investigation. The Young's Modulus of the material was set as $E = 40\,\text{GPa}$ in the base case and Poisson ratio $\nu = 0.3$. The geometry is shown in Figure 1. This domain was discretised into four elements. Four-node quadrilateral elements were used in this analysis and the coordinates of these nodes is shown in Table 1.

Boundary conditions: Nodes 1 and 3 were constrained in x- and y- dimensions to zero displacement (fixed support). A pressure load acting normal to the surface in the positive x direction was applied to the right-hand vertical surface with magnitude $200\,\text{MPa}$. This was discretised into three point loads acting on nodes 5,6,7 with magnitudes $3000\,\text{N}$, $6000\,\text{N}$, $3000\,\text{N}$ respectively.

Part 1: MATLAB was used to write a code to compute the nodal displacement and elemental strains of the geometry. First, nodal geometry was stored in a 3-D matrix, where the 3rd dimension represents element number. Second, the shape functions were found and used to translate the geometry into a ξ-η reference frame. Note, per element, the nodes are being referenced from the bottom left (index = 1) in a clockwise rotation.

$$N = \left[\frac{1}{4}(1-\xi)(1-\eta);\quad \frac{1}{4}(1-\xi)(1+\eta);\quad \frac{1}{4}(1+\xi)(1+\eta);\quad \frac{1}{4}(1+\xi)(1-\eta)\right] \tag{1}$$

Sampling points (weighting factor: 1):

$$\xi = \left[-\frac{1}{\sqrt{3}};\quad -\frac{1}{\sqrt{3}};\quad \frac{1}{\sqrt{3}};\quad \frac{1}{\sqrt{3}}\right] \qquad \eta = \left[-\frac{1}{\sqrt{3}};\quad \frac{1}{\sqrt{3}};\quad \frac{1}{\sqrt{3}};\quad -\frac{1}{\sqrt{3}}\right] \tag{2}$$

From these, we can calculate our Jacobian matrices using (3). This relates the derivatives in the two coordinate frames.

$$J = \begin{bmatrix} \sum \frac{\partial N_i}{\partial \xi} x_i & \sum \frac{\partial N_i}{\partial \xi} y_i \\ \sum \frac{\partial N_i}{\partial \eta} x_i & \sum \frac{\partial N_i}{\partial \eta} y_i \end{bmatrix} \tag{3}$$

Third, the B matrix was formulated and computed.

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \tag{4}$$

$$B_2 = \begin{bmatrix} \frac{J_{22}}{|J|} & -\frac{J_{12}}{|J|} & 0 & 0 \\ -\frac{J_{21}}{|J|} & \frac{J_{11}}{|J|} & 0 & 0 \\ 0 & 0 & \frac{J_{22}}{|J|} & -\frac{J_{12}}{|J|} \\ 0 & 0 & -\frac{J_{21}}{|J|} & \frac{J_{11}}{|J|} \end{bmatrix} \tag{5}$$

$$B_3 = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} & 0 \\ \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} & 0 \\ 0 & \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \frac{\partial N_3}{\partial \xi} & 0 & \frac{\partial N_4}{\partial \xi} \\ 0 & \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & 0 & \frac{\partial N_3}{\partial \eta} & 0 & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \tag{6}$$

$$B = B_1 B_2 B_3 \tag{7}$$

Fourth, the D matrix defines our material properties:

$$D = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{8}$$

Fifth, the stiffness matrix per element can be computed:

$$K_{element} = \int_{-1}^{1}\int_{-1}^{1} B^T D B t |J| \mathrm{d}\xi \mathrm{d}\eta \tag{9}$$

Using Gauss quadrature rule, we can rewrite this equation as:

$$K_{element} = \sum_i \sum_j \left(B^T D B t |J| \right) W_i W_j \tag{10}$$

Where the items in the parentheses are evaluated at each (ξ_i,η_i). These were then assembled into a global stiffness matrix with size 20×20, representing u and v displacements for 10 nodes in x- and y- directions respectively. The elemental strain can be calculated using the nodal displacement and the following formulation:

$$\varepsilon = B \delta_{element} \tag{11}$$

where ε is the elemental strain, B is defined previously and $\delta_{element}$ are the nodal displacements for a given element.

Part 2: The MATLAB code was amended and rerun to gather data for additional cases, namely: $E = 10$ GPa, $E = 100$ GPa, $E = 200$ GPa and $\alpha = 30°$, $\alpha = 60°$, $\alpha = 135°$.

Part 3: The commercial package used to validate the results was ANSYS Mechanical APDL. The base case ($E = 40$ GPa, $\alpha = 90°$) was used to validate the results.

Results:

Part 1: The nodal displacement values in x- and y- directions (u and v respectively) are shown in Figure 2a. Here we see that node 2 has the highest vector sum displacement, peaking at a value of 2.37×10^{-4} m. Nodes 9 and 10 also exhibit high displacement and these correspond to the nodes at the portal of the notch. The elemental strains are shown in Figure 2B and show that elements 2 and 3 undergo relatively high amounts of y-axis strain and shear strain.

Part 2: The nodal displacement in x- and y- directions for various Young's Moduli are shown in Figure 3. Here we see that for small values of E, there was greater displacement in both x- and y- directions. The nodal displacement in x- and y- for various angles of α are shown in Figure 4. Here we see the relationship that the closer the angle to the x- or y- axis, the greater the displacement in that direction. For the x- dimension, we see that $\alpha = 90°$ has the greatest displacement and for the y- dimension, $\alpha = 135°$ has the the highest displacement.

Part 3: Figure 6 shows the contour plot from the simulation conducted in ANSYS. The results generated were for the base case and compared with the results from the MATLAB code. These are tabulated in Table 2.

Discussion: It is evident from the ANSYS results that the code is not functioning properly and is not calculating the displacement of the nodes correctly. There are large percentage value differences and the nodal displacements are an order of magnitude removed from the ANSYS results. However, we do see that some of our results may be considered valid, such as that our displacements do change consistently with what is expected for the case of changing the Young's Modulus. We also see that the changing the pressure angle gives us the intended effect of increasing or decreasing the displacement in the x- or y- dimension.

Figures and tables legend:

Figure 1: geometry of plate.

Figure 2: (a) magnitude of nodal displacement u and v per node. (b) strain components per element

Figure 3: (a) comparison of x-coordinate displacement for a variety of Young's Moduli. (b) comparison of y-coordinate displacement for a variety of Young's Moduli.

Figure 4: (a) comparison of x-coordinate displacement for a variety of pressure angles. (b) comparison of y-coordinate displacement for a variety of pressure angles.

Figure 5: (a) comparison of strain components per element for a variety of Young's Moduli. (b) comparison of strain components per element for a variety of pressure angles.

Figure 6: contour plot of displacement for base case $E = 40\,\text{GPa}$ and $\alpha = 90°$.

Table 1: tabulation of node geometries.

Table 2: tabulation of MATLAB and ANSYS nodal displacements with percentage error.

References:

[1] TWI Ltd. What is finite element analysis (FEA)?, 2023. URL https://www.twi-global.com/technical-knowledge/faqs/finite-element-analysis#:~:text=FEA%20is%20used%20by%20engineers,design%20process%20of%20a%20project.

[2] N.C. Knowles. Finite element analysis. *Computer-Aided Design*, 16(3):134–140, 1984. ISSN 0010-4485. doi: https://doi.org/10.1016/0010-4485(84)90036-8. URL https://www.sciencedirect.com/science/article/pii/0010448584900368.

[3] P Seshu. *Textbook of finite element analysis*. PHI Learning Pvt. Ltd., 2003.

[4] J.Ed Akin. Implementing fea and cad on personal computers and workstations. *Finite Elements in Analysis and Design*, 2(1):19–40, 1986. ISSN 0168-874X. doi: https://doi.org/10.1016/0168-874X(86)90007-7. URL https://www.sciencedirect.com/science/article/pii/0168874X86900077. Special Issue on Unification of Finite Element Software Systems Proceedings of the 8th UFEM Symposium.

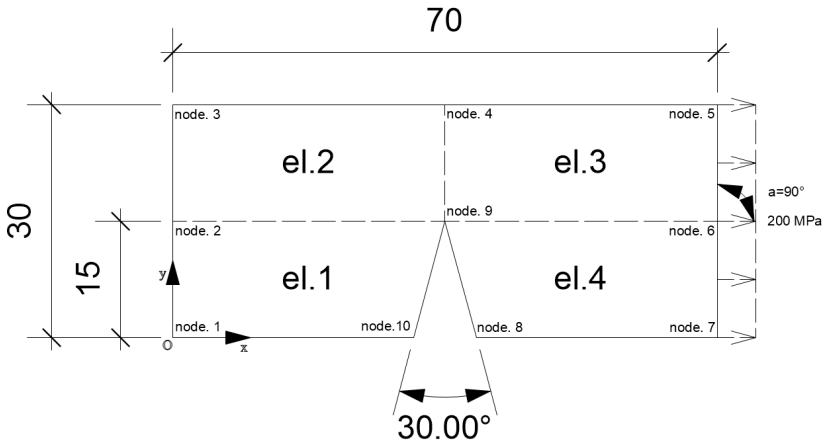
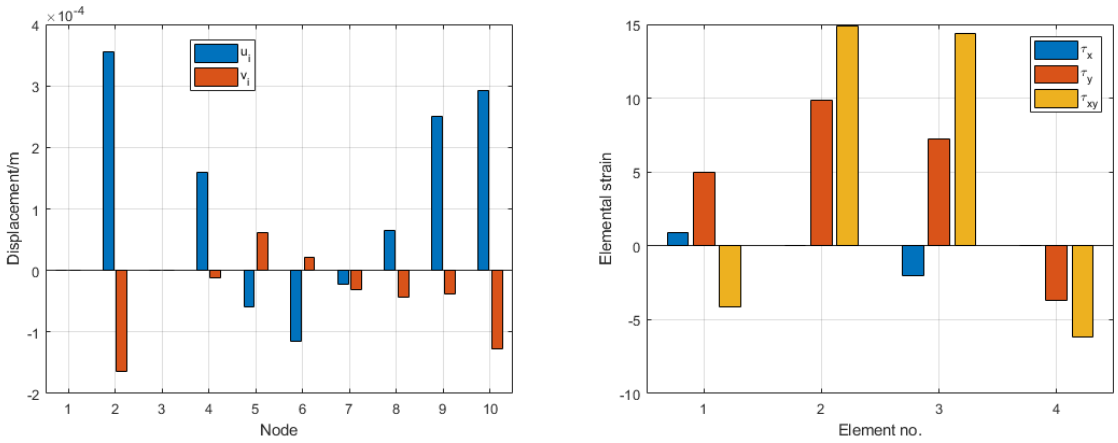
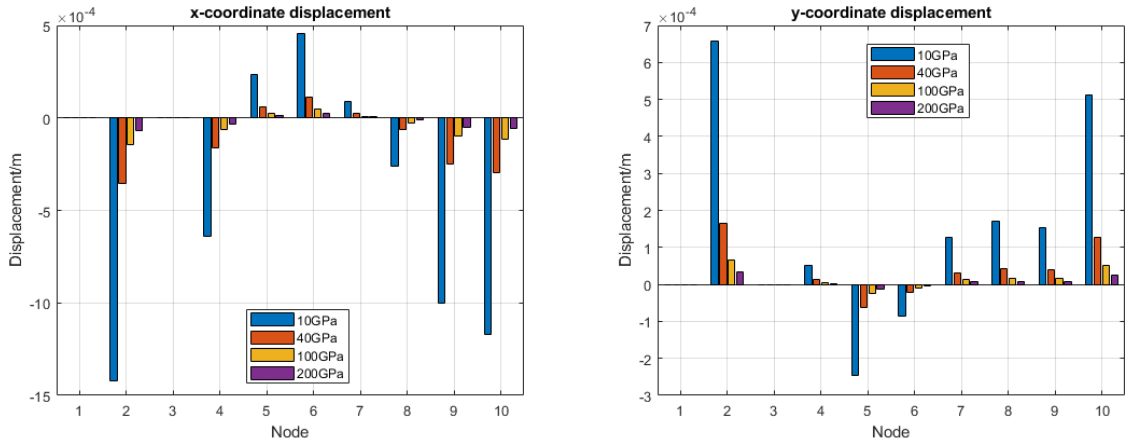


Figure 1



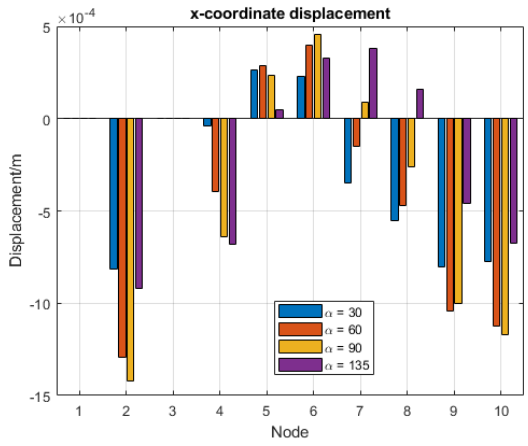
(a) (b)

Figure 2

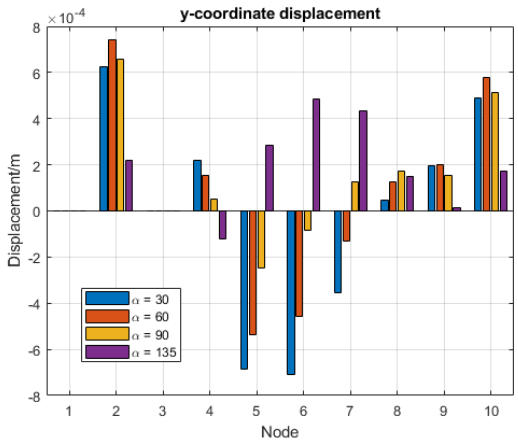


(a) (b)

Figure 3

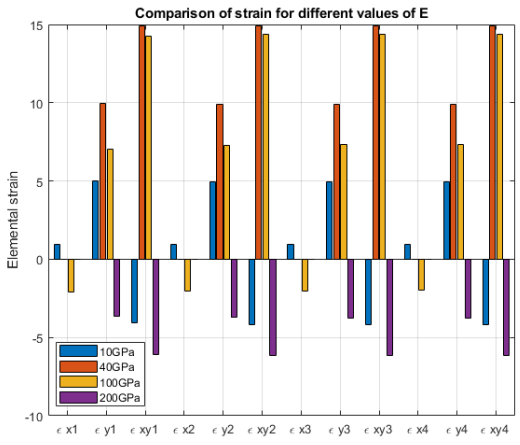


(a)

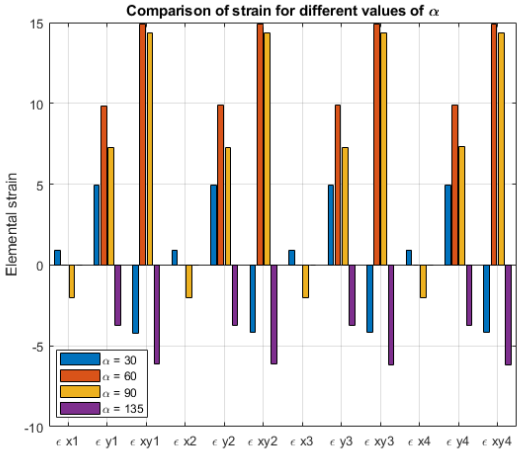


(b)

Figure 4



(a)



(b)

Figure 5

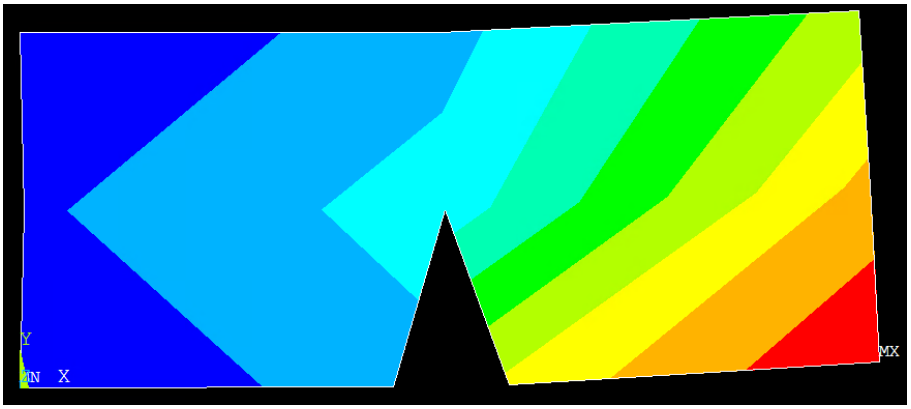


Figure 6

Node	x-coordinate/m	y-coordinate/m
1	0.000	0.000
2	0.000	0.015
3	0.000	0.030
4	0.035	0.030
5	0.070	0.030
6	0.070	0.015
7	0.070	0.000
8	0.039	0.000
9	0.035	0.015
10	0.031	0.000

Table 1

Variable	MATLAB displacement/m	ANSYS displacement/m	Percentage difference
$u1$	0.0	0.0	
$v1$	0.0	0.0	
$u2$	-3.6×10^{-4}	8.4×10^{-5}	-123.57%
$v2$	1.6×10^{-4}	1.2×10^{-6}	-99.24%
$u3$	0.0	0.0	
$v3$	0.0	0.0	
$u4$	-1.6×10^{-4}	1.6×10^{-4}	-201.84%
$v4$	1.3×10^{-5}	2.4×10^{-6}	-81.06%
$u5$	5.8×10^{-5}	2.6×10^{-4}	339.03%
$v5$	-6.2×10^{-5}	4.9×10^{-4}	-893.52%
$u6$	1.1×10^{-4}	5.2×10^{-4}	349.88%
$v6$	-2.1×10^{-5}	5.3×10^{-4}	-2570.43%
$u7$	2.3×10^{-5}	7.1×10^{-4}	3033.13%
$v7$	3.2×10^{-5}	5.7×10^{-4}	1698.52%
$u8$	-6.5×10^{-5}	6.3×10^{-4}	-1061.80%
$v7$	4.3×10^{-5}	7.7×10^{-5}	79.97%
$u9$	-2.5×10^{-4}	2.5×10^{-4}	-200.67%
$v9$	3.8×10^{-5}	1.2×10^{-5}	-69.75%
$u10$	-2.9×10^{-4}	1.5×10^{-4}	-152.45%
$v10$	1.3×10^{-4}	2.3×10^{-5}	-82.11%

Table 2

Appendix:

```
close all
%% definition of elements and nodes
% 4 4-noded elements implemented into 3D matrix
% 3rd dimension represents element no.
% numbered clockwise from bottom left, SI units (m)
A(:,:,1) = [0,0;0,0.015;0.035,0.015;0.031,0;];
A(:,:,2) = [0,0.015;0,0.030;0.035,0.030;0.035,0.015;];
A(:,:,3) = [0.035,0.015;0.035,0.030;0.070,0.030;0.070,0.015;];
A(:,:,4) = [0.039,0;0.035,0.015;0.07,0.015;0.07,0;];
%% shape function + jacobian
% define matrix of xi and nu
syms xi eta
D1 = [xi,eta;];
N = [(1/4)*(1-xi)*(1-eta);(1/4)*(1-xi)*(1+eta);
    (1/4)*(1+xi)*(1+eta);(1/4)*(1+xi)*(1-eta);]; % shape function matrix
jacobianB = sym(zeros(2,4)); % initialise
for i = 1:2 %row
    for j = 1:4 % column
        jacobianB(i,j) = diff(N(j),D1(i));
        % shape function derivative matrix
    end
end
jacobianA = sym(zeros(2,2,4)); %initialise
for i = 1:4 % element
    jacobianA(:,:,i) = jacobianB*A(:,:,i);
    % create jacobian matrix, 3rd dimension represents element
end
% define xi and nu values per node
% bottom left--, top left-+, top right++, bottom right+-
xi2 = [-1/sqrt(3);-1/sqrt(3);1/sqrt(3);1/sqrt(3);];
nu2 = [-1/sqrt(3);1/sqrt(3);1/sqrt(3);-1/sqrt(3);];
jacobian1 = sym(zeros(2,2,4,4)); % initialise
for i = 1:4 % element
    for j = 1:4 % node
        jacobian1(:,:,j,i) = subs(jacobianA(:,:,i),[xi,eta],[xi2(j),nu2(j)]);
        % jacobian with values of xi and nu inputted.
        % 3rd dimension: node, 4th dimension: element
    end
end
%% B matrix
B1 = [1,0,0,0;0,0,0,1;0,1,1,0;];
B2 = zeros(4,4,4,4); %initialise
for i = 1:4
    for j = 1:4
        BTemp1 = inv(jacobian1(:,:,j,i));
        B2(1:2,1:2,j,i) = BTemp1;
        B2(3:4,3:4,j,i) = BTemp1;
    end
end
```

```
B3 = zeros(4,8,4,4); %initialise
jacobianC = zeros(2,4,4,4); %initialise
for i = 1:4 % element
    for j = 1:4 % node
        jacobianC(:,:,j,i) = subs(jacobianB(:,:),[xi,eta],[xi2(j),nu2(j)]);
        % matrix of shape function derviatives with values of xi and nu inputted
    end
end
% index values for placement into matrix:
BTemp2 = [1,3,5,7];
BTemp3 = [2,4,6,8];
for i = 1:4 % element
    for j = 1:4 % node
        for k = 1:4 % column
            for l = 1:2 % row
                B3(l,BTemp2(k),j,i) = jacobianC(l,k,j,i);
                B3(l+2,BTemp3(k),j,i) = jacobianC(l,k,j,i);
            end
        end
    end
end
B = zeros(3,8,4,4); % initialise
for i = 1:4 % element
    for j = 1:4 % node
        B(:,:,j,i) = B1*B2(:,:,j,i)*B3(:,:,j,i);
        % multiplication to form B matrix per node per element
    end
end
B3 = zeros(3,8,4); % initialise
for i = 1:4 % element
    for j = 1:4 % node
        B3(:,:,i) = B3(:,:,i) + B(:,:,j,i);
    end
end
%% D matrix
poisson = 0.3; % poisson ratio
thickness = 0.002; % thickness of plate (m)
E = 40e9; % Young's Modulus (Pa)
D1 = [1,poisson,0;poisson,1,0;0,0,(1-poisson)/2;];
D = (E/1-poisson^2).*D1; % D matrix
%% stiffness matrix
K1 = zeros(8,8,4,4); % initialise
for i = 1:4 % element
    for j = 1:4 % node
        K1(:,:,j,i) =
            (transpose(B(:,:,j,i)))*D*(B(:,:,j,i))*thickness*det(jacobian1(:,:,j,i));
        % stiffness matrix at integration point
        % B^T * D * B * thickness * det(J)
    end
end
```

```
K2 = zeros(8,8,4); % initialise
for i = 1:4 % element
    for j = 1:4 % node
        K2(:,:,i) = K2(:,:,i) + K1(:,:,j,i);
        % element stiffness matrix
    end
end
% assembly of stiffness matrix using global index values
ElementIndex(:,:,1) = [1,1;1,3;1,17;1,19;3,1;3,3;3,17;3,19;17,1;17,3;17,17;17,19;
19,1;19,3;19,17;19,19;];
ElementIndex(:,:,2) =
    [3,3;3,5;3,7;3,17;5,3;5,5;5,7;5,17;7,3;7,5;7,7;7,17;17,3;17,5;17,7;17,17;];
ElementIndex(:,:,3) =
    [17,17;17,7;17,9;17,11;7,17;7,7;7,9;7,11;9,17;9,7;9,9;9,11;11,7;11,7;11,9;11,11;];
ElementIndex(:,:,4) =
    [15,15;15,17;15,11;15,13;17,15;17,17;17,11;17,13;11,15;11,17;11,11;11,13;
13,15;13,17;13,11;13,13;];
% local index values
Indexer = [1,1;1,3;1,5;1,7;3,1;3,3;3,5;3,7;5,1;5,3;5,5;5,7;7,1;7,3;7,5;7,7;];
K = zeros(20,20); % initialise
for j = 1:4
    for i = 1:numel(ElementIndex(:,1))
        ii = ElementIndex(i,1,j);
        jj = ElementIndex(i,2,j);
        kk = Indexer(i,1);
        ll = Indexer(i,2);
        K(ii:ii+1,jj:jj+1) = K2(kk:kk+1,ll:ll+1,j);
        % global stiffness matrix
    end
end
K(:,[1,2,5,6]) = []; % remove fixed displacements
K([1,2,5,6],:) = [];
%% nodal forces
force1 = 3000; % (N)
force2 = 6000; % (N)
alpha = pi/2; % (rad)
Fu3 = force1*sin(alpha); % (N)
Fv3 = force1*cos(alpha); % (N)
Fu5 = Fu3;
Fv5 = Fv3;
Fu4 = force2*sin(alpha); % (N)
Fv4 = force2*cos(alpha); % (N)
F = [zeros(4,1);Fu3;Fv3;Fu4;Fv4;Fu5;Fv5;zeros(6,1)];
displacements = K\F; % find nodal displacements
displacements = [0;0;displacements(1:2);0;0;displacements(3:end)];
D2 = [0,0;
    0+displacements(3),0.015+displacements(4);
    0,0.030;
    0.035+displacements(7),0.030+displacements(8);
    0.070+displacements(9),0.030+displacements(10);
```

```
    0.070+displacements(11),0.015+displacements(12);
    0.070+displacements(13),0+displacements(14);
    0.039+displacements(15),0+displacements(16);
    0.035+displacements(17),0.015+displacements(18);
    0.031+displacements(19),0+displacements(20);]; % global coordinate system with
        displacement
D3(:,:,1) = [D2(1),D2(2),D2(3),D2(4),D2(17),D2(18),D2(19),D2(20)]'; % displacement
    vectors
D3(:,:,2) = [D2(3),D2(4),D2(5),D2(6),D2(7),D2(8),D2(17),D2(18)]';
D3(:,:,3) = [D2(17),D2(18),D2(7),D2(8),D2(9),D2(10),D2(11),D2(12)]';
D3(:,:,4) = [D2(15),D2(16),D2(17),D2(18),D2(11),D2(12),D2(13),D2(14)]';
S4 = zeros(3,4); % initialise
for i = 1:4
    S4(:,i) = B3(:,:,i)*D3(:,:,i);
    % strain
end
displacements(abs(displacements)< 1e-10) = 0; % cleaning
disp4(:,1) = displacements(1:2:end);
disp4(:,2) = displacements(2:2:end);
```