

ME 441: FEM Homework Assignment 3

Afnan Mostafa

December 2023

Problem 1

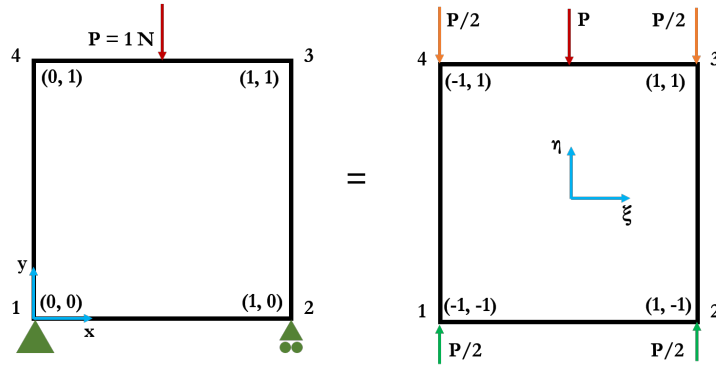


Figure 1: Q4 element

Here,

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}_{4 \times 2} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}_{4 \times 2}$$

$$\text{Jacobian Matrix, } [J]_{2 \times 2} = \frac{1}{4} \begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix}_{2 \times 4} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}_{4 \times 2}$$

$$= \frac{1}{4} \begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$[J]_{2 \times 2} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

Now,

$$[J]_{2 \times 2}^{-1} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{and} \quad |J| = \frac{1}{4}$$

$$[\alpha]_{3 \times 4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad [\beta]_{4 \times 4} = \begin{bmatrix} [J]_{2 \times 2}^{-1} & [0]_{2 \times 2} \\ [0]_{2 \times 2} & [J]_{2 \times 2}^{-1} \end{bmatrix}$$

$$[\gamma]_{4 \times 8} = \frac{1}{4} \begin{bmatrix} -1+\eta & 0 & 1-\eta & 0 & 1+\eta & 0 & -1-\eta & 0 \\ -1+\xi & 0 & -1-\xi & 0 & 1+\xi & 0 & 1-\xi & 0 \\ 0 & -1+\eta & 0 & 1-\eta & 0 & 1+\eta & 0 & -1-\eta \\ -0 & -1+\xi & 0 & -1-\xi & 0 & 1+\xi & 0 & 1-\xi \end{bmatrix}$$

$$[B]_{3 \times 8} = [\alpha]_{3 \times 4} [\beta]_{4 \times 4} [\gamma]_{4 \times 8} \quad [K]_{8 \times 8} = \int_{-1}^1 [B]_{8 \times 3}^T [E]_{3 \times 3} [B]_{3 \times 8} t |J| d\xi d\eta$$

$$[B]_{3 \times 8} = \begin{bmatrix} \frac{\eta}{2} - \frac{1}{2} & 0 & \frac{1}{2} - \frac{\eta}{2} & 0 & \frac{\eta}{2} + \frac{1}{2} & 0 & -\frac{\eta}{2} - \frac{1}{2} & 0 \\ 0 & \frac{\xi}{2} - \frac{1}{2} & 0 & -\frac{\xi}{2} - \frac{1}{2} & 0 & \frac{\xi}{2} + \frac{1}{2} & 0 & \frac{1}{2} - \frac{\xi}{2} \\ \frac{\xi}{2} - \frac{1}{2} & \frac{\eta}{2} - \frac{1}{2} & -\frac{\xi}{2} - \frac{1}{2} & \frac{1}{2} - \frac{\eta}{2} & \frac{\xi}{2} + \frac{1}{2} & \frac{\eta}{2} + \frac{1}{2} & \frac{1}{2} - \frac{\xi}{2} & -\frac{\eta}{2} - \frac{1}{2} \end{bmatrix}$$

$$[K]_{8 \times 8} = \int_{-1}^1 [B]_{8 \times 3}^T [E]_{3 \times 3} [B]_{3 \times 8} t |J| d\xi d\eta = \int_{-1}^1 [\text{integrand}]_{8 \times 8} d\xi d\eta$$

Using 2nd order Gauss Quadrature to solve the integral:

$$[K]_{8 \times 8} = \begin{pmatrix} \frac{5000}{9} & 250 & -\frac{3500}{9} & \frac{250}{3} & -\frac{2500}{9} & -250 & \frac{1000}{9} & -\frac{250}{3} \\ 250 & \frac{5000}{9} & -\frac{250}{3} & \frac{1000}{9} & -250 & -\frac{2500}{9} & \frac{250}{3} & -\frac{3500}{9} \\ -\frac{3500}{9} & -\frac{250}{3} & \frac{5000}{9} & -250 & \frac{1000}{9} & \frac{250}{3} & -\frac{2500}{9} & 250 \\ \frac{250}{3} & \frac{1000}{9} & -250 & \frac{5000}{9} & -\frac{250}{3} & -\frac{3500}{9} & 250 & -\frac{2500}{9} \\ -\frac{2500}{9} & -250 & \frac{1000}{9} & -\frac{250}{3} & \frac{5000}{9} & 250 & -\frac{3500}{9} & \frac{250}{3} \\ -250 & -\frac{2500}{9} & \frac{250}{3} & -\frac{3500}{9} & 250 & \frac{5000}{9} & -\frac{250}{3} & \frac{1000}{9} \\ \frac{1000}{9} & \frac{250}{3} & -\frac{2500}{9} & 250 & -\frac{3500}{9} & -\frac{250}{3} & \frac{5000}{9} & -250 \\ -\frac{250}{3} & -\frac{3500}{9} & 250 & -\frac{2500}{9} & \frac{250}{3} & \frac{1000}{9} & -250 & \frac{5000}{9} \end{pmatrix}_{8 \times 8}$$

Now,

$$[K]_{8 \times 8} \{d\}_{8 \times 1} = \{F\}_{8 \times 1}$$

Applying boundary conditions,

$$\{d\} = \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ u_2 \\ 0 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} = \begin{Bmatrix} u_2 \\ u_3 \\ v_3 \\ v_4 \end{Bmatrix}_{5 \times 1} \quad \{F\} = \begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{2x} \\ f_{2y} \\ f_{3x} \\ f_{3y} \\ f_{4x} \\ f_{4y} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0.5 \\ 0 \\ 0.5 \\ 0 \\ -0.5 \\ 0 \\ -0.5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ -0.5 \\ 0 \\ -0.5 \end{Bmatrix}_{5 \times 1}$$

$$[K]_{8 \times 8} \xrightarrow{\text{apply BC}} [K]_{5 \times 5}$$

(eliminating rows and columns to match with $\{d\}$ and $\{F\}$)

$$\{d\}_{5 \times 1} = [K]_{5 \times 5}^{-1} \{F\}_{5 \times 1}$$

$$\begin{Bmatrix} u_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} = \begin{Bmatrix} 0.0005 \\ 0.0005 \\ -0.001 \\ 0 \\ -0.001 \end{Bmatrix}$$

Listing 1: Nodal displacements of a Q4 element with one mesh

```

1 %% FEM, HW 3, Problem 1
2 %% Afnan Mostafa
3 %% 11/28/2023
4
5 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% clearing space %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 clc
8 clear
9 close all
10 rng('shuffle')
11
12 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% material properties %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13
14 E_0 = 1e6;
15 nu = 0.5;
16 t = 0.001;
17
18 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% symbolic math %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19
20 syms n e x y u2 u3 v3 u4 v4
21
22 %%
23 %%      |
24 %%      V
25 %% 4 ----- 3
26 %% |         |
27 %% |         |
28 %% |         |
29 %% |         |
30 %% |         |
31 %% |         |
32 %% 1 ----- 2
33 %% /\         /\
34 %% -----oo--
35
36 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% coordinate: bottom left %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37
38 xyMat = [0 0; 1 0; 1 1; 0 1];
39
40 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% coordinate: middle %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41
42 % xyMat = [-0.5 -0.5; 0.5 -0.5; 0.5 0.5; -0.5 0.5];
43
44 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Jacobian Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
45
46 J = 1/4*[-(1-n) (1-n) (1+n) -(1+n);
47          -(1-e) -(1+e) (1+e) (1-e)]*xyMat;
48 invJ = inv(J);
49 Zer = zeros(size(invJ));
50
51 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Alpha Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
52
53 alphaMat = [1 0 0 0; 0 0 0 1; 0 1 1 0];
54
55 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Beta Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56
57 betaMat = [[invJ] [Zer]; [Zer] [invJ]];
58

```

```

59 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Gamma Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
60
61 gammaMat = (1/4)*[-1+n 0 1-n 0 1+n 0 -1-n 0;
62 -1+e 0 -1-e 0 1+e 0 1-e 0;
63 0 -1+n 0 1-n 0 1+n 0 -1-n;
64 0 -1+e 0 -1-e 0 1+e 0 1-e];
65
66 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Strain-Displacement Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
67
68 B = alphaMat*betaMat*gammaMat;
69 B_t = transpose(B);
70
71 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Constitutive Matrix: plane stress %%%%%%%%%%%
72
73 E = (E_0/(1-(nu)^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];
74
75 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Stiffness Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76
77 integrand = eval(B_t*E*B*t*det(J));
78
79 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Gauss Quadrature (2nd order) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80
81 gaussPoints = [-1/sqrt(3) -1/sqrt(3);
82 1/sqrt(3) -1/sqrt(3);
83 1/sqrt(3) 1/sqrt(3);
84 -1/sqrt(3) 1/sqrt(3)];
85
86 for i=1:length(integrand)
87     for j=1:length(integrand)
88         func = integrand(i,j);
89         c=0;
90         for k=1:4
91             e = gaussPoints(k,1);
92             n = gaussPoints(k,2);
93             gq(k) = eval(subs(func));
94         end
95         integral(i,j) = gq(1)+gq(2)+gq(3)+gq(4);
96     end
97 end
98
99 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% disp, force matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
100
101 disp = [0;0;u2;0;u3;v3;u4;v4];
102 force = [0;0.5;0;0.5;0;-0.5;0;-0.5];
103
104 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% apply BCs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
105
106 disp_BC = [u2;u3;v3;u4;v4];
107 force_BC = [0;0;-0.5;0;-0.5];
108 new_integral = integral;
109
110 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% remove K singularity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111
112 new_integral(:,1) = [];
113 new_integral(:,1) = [];
114 new_integral(:,2) = [];
115 new_integral(1,:) = [];
116 new_integral(1,:) = [];
117 new_integral(2,:) = [];
118
119 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% solve Kd = F %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
120
121 nod_disp = inv(new_integral)*force_BC;
122
123 allDisp = [
124     0 0; nod_disp(1) 0; nod_disp(2) nod_disp(3); nod_disp(4) nod_disp(5)
125 ];
126
127 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% print out nodal displ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
128
129 [id_v] = find(ismember(abs(allDisp(:,2)), max(abs(allDisp(:,2)))));
130 [id_u] = find(ismember(abs(allDisp(:,1)), max(abs(allDisp(:,1)))));
131
132 sprintf('Max Vertical displacement occurs at node %d: %0.4f units', id_v

```

```

132     , allDisp(id_v,2))
133     sprintf('Max Horizontal displacement occurs at node %d: %0.4f units',
134           id_u , allDisp(id_u,1))
135
136     %%
137     %
138     %
139     %
140     %
141     %
142     %
143     %
144     %
145     %
146     %
147
148     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot original and deformed systems %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
149
150     plotDeform=1;
151     if plotDeform
152         hold on
153         % plots the original system
154         l1 = line([0,1],[0,0], 'LineWidth',3, 'Color', 'k');
155         line([1,1],[0,1], 'LineWidth',3, 'Color', 'k');
156         line([0,0],[0,1], 'LineWidth',3, 'Color', 'k');
157         line([1,0],[1,1], 'LineWidth',3, 'Color', 'k');
158         x = [0; 1; 1; 0; 0];
159         y = [0; 0; 1; 1; 0];
160         disp_u = [allDisp(1,1); allDisp(2,1); allDisp(3,1); allDisp(4,1);
161                  allDisp(1,1)];
162
163         disp_v = [allDisp(1,2); allDisp(2,2); allDisp(3,2); allDisp(4,2);
164                  allDisp(1,2)];
165
166         defX = x + disp_u;
167         defY = y + disp_v;
168         box on
169         % plots the deformed system
170         p1 = plot(defX, defY, 'r-', 'LineWidth', 1.5);
171         set(gca, 'FontName', 'Garamond', 'FontSize', 18, 'FontWeight', 'bold', ...
172               'LineWidth', 2, 'XMinorTick', 'off', ...
173               'YMinorTick', 'off', 'GridAlpha', 0.07, ...
174               'GridLineStyle', '-', 'LineWidth', 2);
175         title('Deformation Plot in Real Space (global coordinate system)');
176         xlabel('X');
177         ylabel('Y');
178         legend([l1 p1], {'Original System', 'Deformed System'}, 'Location', '
179                  southeast', ...
180                  'Color', [0.941176470588235 0.941176470588235 0.941176470588235])
181
182         % set(gcf, 'units', 'points', 'position', [100,100,1024,700])
183     end
184
185     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Output of the code:

Max Vertical displacement occurs at nodes 3 and 4: -0.001 units.

Max Horizontal displacement occurs at nodes 2 and 3: 0.0005 units

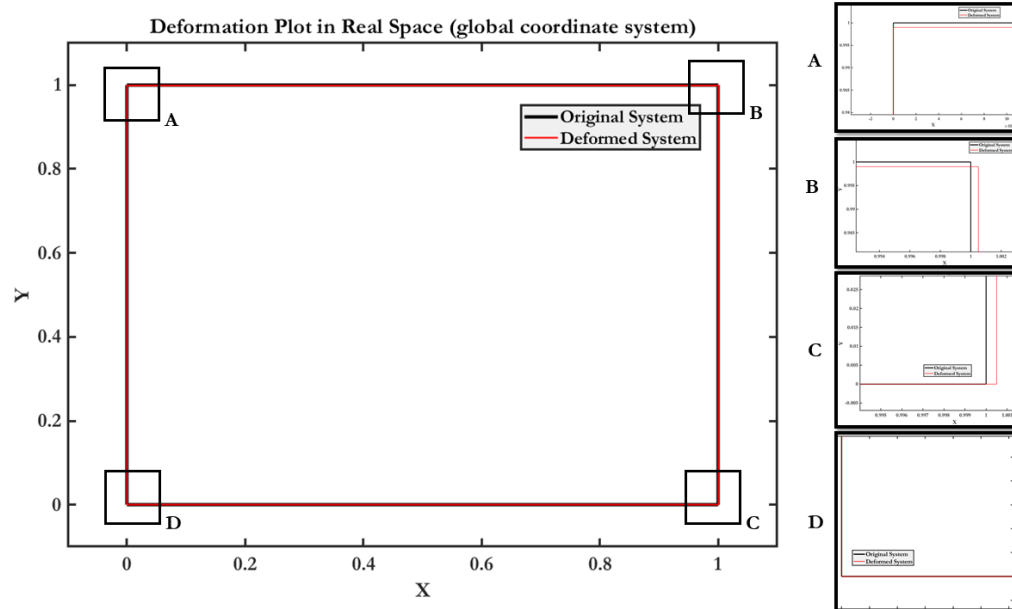


Figure 2: Deformation Plot of Q4 single element

Problem 2

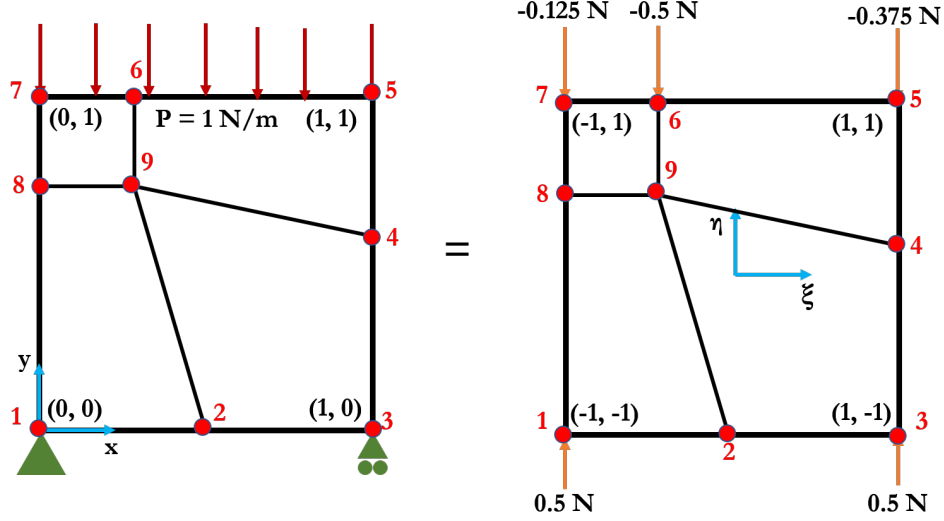


Figure 3: Q4 element with 4 meshes

Algorithm for solving for displacements when 4 Q4 elements are considered:

1. Get $[J]_{2 \times 2}$ for each element using:

$$[J]_{2 \times 2} = \frac{1}{4} \begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix}_{2 \times 4} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}_{4 \times 2},$$

where $\begin{bmatrix} x & y \end{bmatrix}$ is different for each element.

2. Solve for integrand using **IntegrandStiffMatQ4.m** (see Appendix B) function (for each element),

$$[B]_{8 \times 3}^T [E]_{3 \times 3} [B]_{3 \times 8} |J| t = [\text{integrand}]_{8 \times 8}$$

3. Use Gauss Quadrature (**GaussQuadQ4.m**) (see Appendix C) to get the integral (or elemental stiffness matrix) for each element:

$$[K]_{8 \times 8} = \int_{-1}^1 [B]_{8 \times 3}^T [E]_{3 \times 3} [B]_{3 \times 8} |J| d\xi d\eta = \int_{-1}^1 [\text{integrand}]_{8 \times 8} d\xi d\eta$$

4. Globalize element stiffness matrices, $[K_i]_{8 \times 8}$; $i = 1, 2, 3, 4$ (no of elements) (**globalizeStiffMat.m**) (see Appendix D) and add them to get the global stiffness matrix $[K]$

$$[K_i]_{8 \times 8} \xrightarrow{\text{globalize}} [K_i]_{18 \times 18}$$

$$[\mathbf{K}_{\text{struc}}]_{18 \times 18} = \sum_{i=1}^4 [\mathbf{K}_i]_{18 \times 18}$$

5. Apply boundary conditions (3 BCs, hence 3 rows and columns are dropped)

$$[\mathbf{K}_{\text{struc}}]_{18 \times 18} \xrightarrow{\text{apply BC}} [\mathbf{K}_{\text{struc}}]_{15 \times 15}$$

6. Solve for nodal displacements using:

$$[\mathbf{K}_{\text{struc}}]_{15 \times 15} \{d\}_{15 \times 1} = \{F\}_{15 \times 1} \quad \{d\}_{15 \times 1} = [\mathbf{K}_{\text{struc}}]_{15 \times 15}^{-1} \{F\}_{15 \times 1}$$

$$\{d\}_{15 \times 1} = \begin{pmatrix} u_2 \\ v_2 \\ u_3 \\ u_4 \\ v_4 \\ u_5 \\ v_5 \\ u_6 \\ v_6 \\ u_7 \\ v_7 \\ u_8 \\ v_8 \\ u_9 \\ v_9 \end{pmatrix} = \begin{pmatrix} 0.0005 \\ -0.001 \\ 0.0008 \\ 0.0006 \\ -0.0011 \\ 0.0007 \\ -0.0015 \\ 0.0003 \\ -0.0015 \\ 0.0002 \\ -0.0016 \\ 0.0003 \\ -0.0014 \\ 0.0004 \\ -0.0013 \end{pmatrix}$$

It is evident from the two problems described above that using more elements will change the results a bit but they should be close to each other (as can be seen here as well)

Que	Max. Horizontal Displacement, u_{max}	Max. Vertical Displacement, v_{max}
1	+0.0005 (nodes 2, 3)	-0.0010 (nodes 3, 4)
2	+0.0008 (node 3 = node 2 in que 1)	-0.0016 (node 7 = node 3 in que 1)

Listing 2: Nodal displacements of a Q4 element with 4 element meshes

```

1 % =====
2 %% ME 441: FEM, HW 3, Problem 2
3 %% Afnan Mostafa
4 %% 11/29/2023
5 % =====
6 %%
7 % this script needs 3 functions files: IntegrandStiffMatQ4.m,
8 % GaussQuadQ4.m, and globalizeStiffMat.m files (put them in the same dir
9 % )
10 %%
11 %% %%%%%%%%%%%%%clearing space%%%%%%%%%%%%%
12
13 clear
14 clc
15 close all

```



```

16 rng('shuffle')
17
18 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% material properties %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19
20 E_0 = 1e6;
21 nu = 0.5;
22 t = 0.001;
23
24 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% symbolic math %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25
26 syms n e x y u2 v2 u3 v3 u4 v4 u5 v5 u6 v6 u7 v7 u8 v8 u9 v9 X Y
27
28 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% coordinate: bottom left %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29
30 elements = 4;
31 order = 2;
32 totNodes = 9;
33
34 xyMat = [0 0.75; 0.25 0.75; 0.25 1; 0 1];
35 xyMat2 = [0 0; 0.5 0; 0.25 0.75; 0 0.75];
36 xyMat3 = [0.5 0; 1 0; 1 0.6; 0.25 0.75];
37 xyMat4 = [0.25 0.75; 1 0.6; 1 1; 0.25 1];
38
39 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% xy matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40
41 xy = cell(1,4);
42 xy{1,1} = xyMat;
43 xy{1,2} = xyMat2;
44 xy{1,3} = xyMat3;
45 xy{1,4} = xyMat4;
46
47 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% cell creation for storage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
48
49 kmat_all = cell(1,elements);
50 intgrd_all = cell(1,elements);
51
52 %% %%%%%%%%% function callback: to get elemental stiff mat. %%%%%%%%%
53
54 %auxiliary function files
55 for p=1:elements
56     [integrand,B,B_t] = IntegrandStiffMatQ4(xy{1,p},t,E_0,nu,0,1);
57     intgrd_all{1,p} = integrand;
58     [stiffMat] = GaussQuadQ4(order,intgrd_all{1,p});
59     kmat_all{1,p} = stiffMat;
60 end
61
62 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% globalization starts here %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63
64 %%
65 %%      |      |      |      |      |
66 %%      V      V      V      V      V
67 %%  7  -----6-----5
68 %%  |          |          |
69 %%  |      #1      |      #4      |
70 %%  |  8-----9-----|
71 %%  |          |          |
72 %%  |      #2      |      #3      |
73 %%  |          |          |
74 %%  |  1-----2-----3
75 %%  /\              /\
76 %%  -----oo-----
77
78
79 % order of nodes in an element matter
80 ele1 = [8,9,6,7];
81 ele2 = [1,2,9,8];
82 ele3 = [2,3,4,9];
83 ele4 = [9,4,5,6];
84
85 nod_ele = [ele1; ele2; ele3; ele4];
86
87 % function callback for globalization
88 for u=1:elements
89     [mat_cp] = globalizeStiffMat(kmat_all{1,u},nod_ele(u,:),4,9,2);

```

```

90     stiffMatSet{u,1} = mat_cp;
91 end
92
93 globalMat = zeros(size(mat_cp));
94
95 for v=1:elements
96     globalMat = globalMat + stiffMatSet{v,1};
97 end
98
99 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% disp, force matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
100
101 % disp = [0;0;u2;v2;u3;0;u4;v4;u5;v5;u6;v6;u7;v7;u8;v8;u9;v9];
102 % force =
103     [0;0.25;0;0.5;0;0.25;0;-0.125;0;-0.75/2;0;-0.5;0;-0.25/2;0;0.125;0;-0.5];
104
105 disp = [0;0;u2;v2;u3;0;u4;v4;u5;v5;u6;v6;u7;v7;u8;v8;u9;v9];
106 force = [0;0.5;0;0;0.5;0;0;0;-0.75/2;0;-0.5;0;-0.25/2;0;0;0;0];
107
108 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% apply BCs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
109
110 % solved by hand
111 disp_BC = [u2;v2;u3;u4;v4;u5;v5;u6;v6;u7;v7;u8;v8;u9;v9];
112 force_BC = [0;0;0;0;0;0;-0.75/2;0;-0.5;0;-0.25/2;0;0;0;0];
113
114 new_integral = globalMat;
115
116 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% remove K singularity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
117
118 % dynamic deletion (matrix index changes with each progressive line)
119 new_integral(:,1) = []; % delete 1st column
120 new_integral(:,1) = []; % delete 1st column (2nd col. of original) of
121     mod. matrix
122 new_integral(:,4) = []; % delete 4th column (6th col. of original) of
123     mod. matrix
124 new_integral(1,:) = []; % same as above but for rows
125 new_integral(1,:) = [];
126 new_integral(4,:) = [];
127
128 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% solve Kd = F %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
129
130 nod_disp = new_integral\force_BC;
131
132 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot displacement contours %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
133
134 allDisp = [0 0; nod_disp(1) nod_disp(2); nod_disp(3) 0; nod_disp(4)
135     nod_disp(5); ...
136     nod_disp(6) nod_disp(7); nod_disp(8) nod_disp(9); nod_disp(10) ...
137     nod_disp(11); nod_disp(12) nod_disp(13); nod_disp(14) nod_disp(15)];
138
139 % side lengths of quadrilateral
140 xlo = 0; ylo = 0;
141 xhi = 1; yhi = 1;
142
143 % no of nodes from the size of coordinate matrix (for plotting purposes)
144 reShapingSize = sqrt(totNodes); % totNodes = 9;
145
146 % reshape u and v matrices for contouring
147 uMat = reshape(allDisp(:,1),reShapingSize,reShapingSize)';
148 vMat = reshape(allDisp(:,2),reShapingSize,reShapingSize)';
149
150 xcoords = [0 0.5 1; 0 0.25 1; 0 0.25 1];
151 ycoords = [0 0 0; 0.75 0.75 0.6; 1 1 1];
152
153 hold on
154 box on
155 %% u-disp contour plot
156 % subplot(2,1,1);
157 contourf(xcoords, ycoords, uMat, 20,'LineWidth',2);
158 set(gca,'FontName','Garamond','FontSize',18,'FontWeight','bold',...
159     'LineWidth',2,'XMinorTick','off',...
160     'YMinorTick','off','GridAlpha',0.07,...
161     'GridLineStyle','—','LineWidth',2);
162 title('Contour Plot: u');

```

```

159 xlabel('X');
160 ylabel('Y');
161 colorbar;
162
163 showQ4 = 1;
164 if showQ4 == 1
165     hold on
166     line([0,0.25],[0.75,0.75], 'LineWidth',2, 'Color', 'r');
167     line([0.25,0.25],[0.75,1], 'LineWidth',2, 'Color', 'r');
168     line([0.25,1],[0.75,0.6], 'LineWidth',2, 'Color', 'r');
169     line([0.25,0.5],[0.75,0], 'LineWidth',2, 'Color', 'r');
170     hold off
171 end
172 [gcf] = plotNodes(gcf,totNodes);
173
174 %% v-disp contour plot
175 figure;
176 % subplot(2,1,2);
177 contourf(xcoords, ycoords, vMat, 20, 'LineWidth',2);
178 hold on
179 % plots the original system
180 if showQ4 == 1
181     line([0,0.25],[0.75,0.75], 'LineWidth',3, 'Color', 'r');
182     line([0.25,0.25],[0.75,1], 'LineWidth',3, 'Color', 'r');
183     line([0.25,1],[0.75,0.6], 'LineWidth',3, 'Color', 'r');
184     line([0.25,0.5],[0.75,0], 'LineWidth',3, 'Color', 'r');
185 end
186
187 set(gca, 'FontName', 'Garamond', 'FontSize', 18, 'FontWeight', 'bold', ...
188     'LineWidth', 2, 'XMinorTick', 'off', ...
189     'YMinorTick', 'off', 'GridAlpha', 0.07, ...
190     'GridLineStyle', '—', 'LineWidth', 2);
191 title('Contour Plot: v');
192 xlabel('X');
193 ylabel('Y');
194 colorbar;
195 [gcf] = plotNodes(gcf,totNodes);
196
197 %% %%%%%%%%%%% plot original and deformed systems %%%%%%%%%%%
198
199 plotDeform=0;
200 if plotDeform
201     hold off
202     figure;
203     % plots the original system
204     l1 = line([0,0.25],[0.75,0.75], 'LineWidth',3, 'Color', 'k');
205     line([0.25,0.25],[0.75,1], 'LineWidth',3, 'Color', 'k');
206     line([0.25,1],[0.75,0.6], 'LineWidth',3, 'Color', 'k');
207     line([0.25,0.5],[0.75,0], 'LineWidth',3, 'Color', 'k');
208     hold on
209     x = [0; 0.5; 1; 1; 1; 0.5; 0; 0; 0];
210     y = [0; 0; 0; 0.6; 1; 1; 1; 0.75; 0];
211     disp_u = [allDisp(1,1); allDisp(2,1); allDisp(3,1); allDisp(4,1);
212         ...
213         allDisp(5,1); allDisp(6,1); allDisp(7,1); allDisp(8,1); allDisp
214             (9,1)];
215
216     disp_v = [allDisp(1,2); allDisp(2,2); allDisp(3,2); allDisp(4,2);
217         ...
218         allDisp(5,2); allDisp(6,2); allDisp(7,2); allDisp(8,2); allDisp
219             (9,2)];
220
221     defX = x + disp_u;
222     defY = y + disp_v;
223     hold on
224     box on
225     % plots the deformed system
226     p1 = plot(defX, defY, 'c-', 'LineWidth', 1.5);
227     set(gca, 'FontName', 'Garamond', 'FontSize', 18, 'FontWeight', 'bold', ...
228         'LineWidth', 2, 'XMinorTick', 'off', ...
229         'YMinorTick', 'off', 'GridAlpha', 0.07, ...
230         'GridLineStyle', '—', 'LineWidth', 2);
231     title('Deformation Plot in Real Space (global coordinate system)');
232     xlabel('X');

```

```

229 ylabel('Y');
230 legend([l1 p1],{'Original System', 'Deformed System'},'Location','
southeast',...
231 'Color',[0.941176470588235 0.941176470588235 0.941176470588235])
232 set(gcf,'units','points','position',[100,100,1024,700])
233 end
234
235 [id_v] = find(ismember(abs(allDisp(:,2)), max(abs(allDisp(:,2)))));
236 [id_u] = find(ismember(abs(allDisp(:,1)), max(abs(allDisp(:,1)))));
237
238 sprintf('Max Vertical displacement occurs at node %d: %0.4f units', id_v
, allDisp(id_v,2))
239 sprintf('Max Horizontal displacement occurs at node %d: %0.4f units',
id_u, allDisp(id_u,1))
240
241 %%
242 % V V V V V
243 % 7 -----6-----5
244 % |
245 % | #1 | #4 |
246 % 8 -----9-----
247 % | | |
248 % | #2 | #3 | 4
249 % | | |
250 % | 1 -----2-----3
251 % /\ /\
252 % -----o o--
253 %
254
255 %% %%%%%%%%%%% auxiliary func to plot nodes %%%%%%%%%%%
256 function [gcf] = plotNodes(gcf,totNodes)
257 pos = [0 0; 0.5 0; 1 0; 1 0.6; 1 1; 0.25 1; 0 1; 0 0.75; 0.25
0.75]+[0.015 0.025];
258 str = string(1:totNodes);
259 plotNodes=1;
260 if plotNodes
261 for b=1:totNodes
262 text(pos(b,1),pos(b,2),str(b),'Color','red','FontSize',18,'
FontName','Garamond','FontWeight','bold')
263 end
264 end
265 end
266 %% %%%%%%%%%%% END %%%%%%%%%%%

```

Output of the code:

Max Vertical displacement occurs at node 7: -0.0016 units.

Max Horizontal displacement occurs at node 3: 0.0008 units

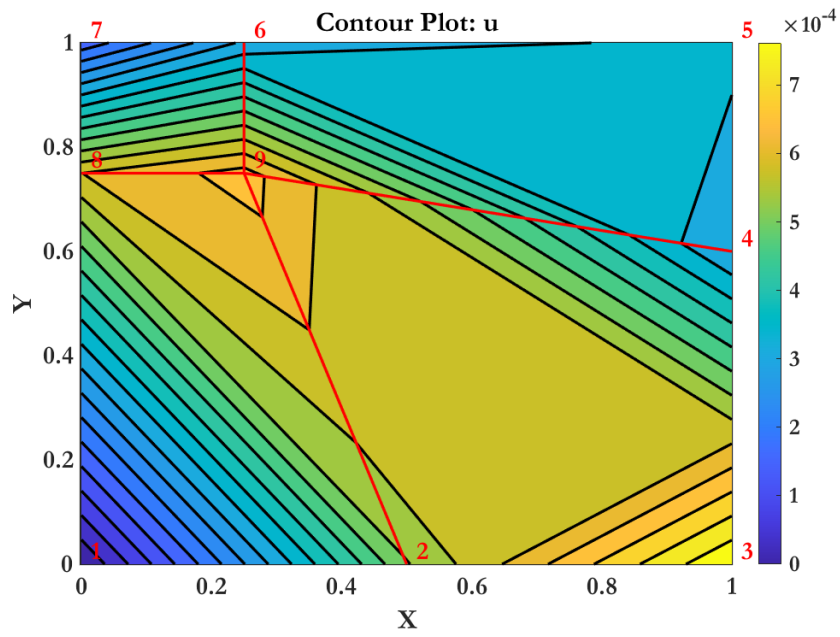


Figure 4: Contour Plot of u-displacement

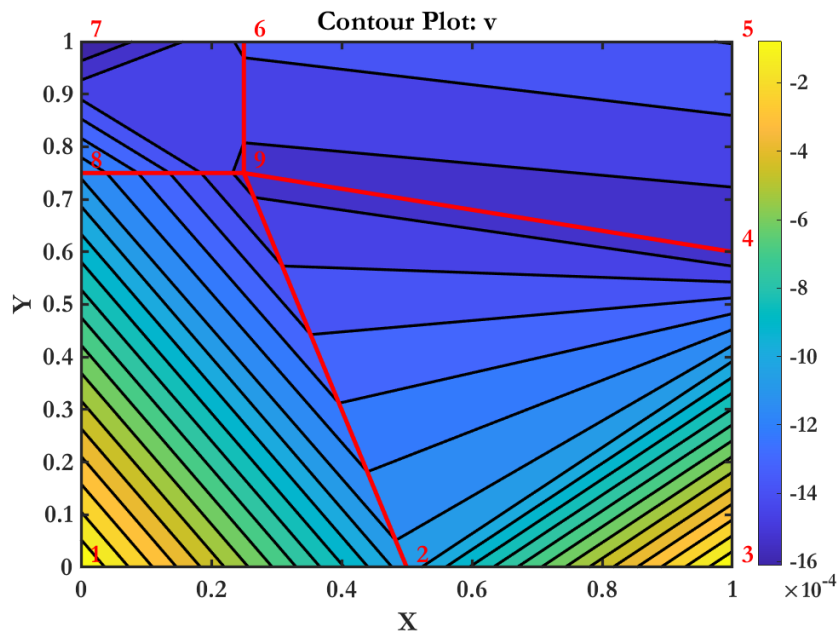


Figure 5: Contour Plot of v-displacement

Problem 3

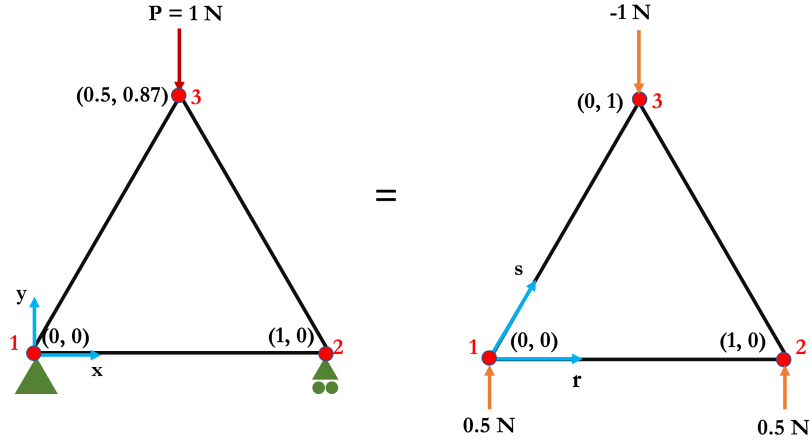


Figure 6: CST

Algorithm for solving for displacements when 1 CST element is considered:

1. Get shape functions, N , for CST:

$$N_1 = 1 - r - s, \quad N_2 = r, \text{ and } N_3 = s$$

2. Get $x(r, s)$ and $y(r, s)$ from

$$x(r, s) = x_1 N_1 + x_2 N_2 + x_3 N_3 = r x_2 - x_1 (r + s - 1) + s x_3$$

$$y(r, s) = y_1 N_1 + y_2 N_2 + y_3 N_3 = r y_2 - y_1 (r + s - 1) + s y_3$$

3. Get Jacobian matrix, $[J]$, from derivatives of x and y w.r.t. r and s

$$[J]_{2 \times 2} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{bmatrix}$$

4. Get $u(r, s)$ and $v(r, s)$:

$$u(r, s) = u_1 N_1 + u_2 N_2 + u_3 N_3 = r u_2 - u_1 (r + s - 1) + s u_3$$

$$v(r, s) = v_1 N_1 + v_2 N_2 + v_3 N_3 = r v_2 - v_1 (r + s - 1) + s v_3$$

5. Get $\{u_{,real}\}$ and $\{v_{,real}\}$:

$$\{u_{,real}\}_{2 \times 1} = \left\{ \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{bmatrix} \right\}_{2 \times 1} = [J]_{2 \times 2}^{-1} \left\{ \begin{bmatrix} \frac{\partial u}{\partial r} \\ \frac{\partial u}{\partial s} \end{bmatrix} \right\}_{2 \times 1}$$

$$\{v_{,real}\}_{2 \times 1} = \begin{bmatrix} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{bmatrix}_{2 \times 1} = [J]_{2 \times 2}^{-1} \begin{bmatrix} \frac{\partial v}{\partial r} \\ \frac{\partial v}{\partial s} \end{bmatrix}_{2 \times 1}$$

6. Get $\epsilon_x, \epsilon_y, \epsilon_{xy}$ from $\{u_{,real}\}$ and $\{v_{,real}\}$

$$\{\epsilon\}_{3 \times 1} = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{bmatrix}_{3 \times 1}$$

7. Matricize these 3 equations: $\epsilon_x, \epsilon_y, \epsilon_{xy}$ and compare $\{\epsilon\}$ with the following equation:

$$\{\epsilon\}_{3 \times 1} = [B]_{3 \times 6} \{d\}_{6 \times 1}$$

8. Strain-displacement matrix is:

$$[B]_{3 \times 6} = \frac{1}{2A} \begin{bmatrix} y_2 - y_3 & 0 & y_3 - y_1 & 0 & y_1 - y_2 & 0 \\ 0 & x_3 - x_2 & 0 & x_1 - x_3 & 0 & x_2 - x_1 \\ 2x_1 - x_2 - x_3 & 2y_1 - y_2 - y_3 & x_1 - x_3 & y_3 - y_1 & x_2 - x_1 & y_1 - y_2 \end{bmatrix}_{3 \times 6}$$

9. Use Gauss Quadrature to get the integral or stiffness matrix:

$$[\mathbf{K}]_{6 \times 6} = \int_{-1}^1 [B]_{6 \times 3}^T [E]_{3 \times 3} [B]_{3 \times 6} t |J| dr ds = \int_{-1}^1 [\text{integrand}]_{6 \times 6} dr ds =$$

$$[\mathbf{K}]_{6 \times 6} = [\text{integrand}]_{6 \times 6} \int_{-1}^1 dr ds$$

here, $\text{integrand} \neq f(r, s) = \text{constant}$

10. Apply boundary conditions (3 BCs, hence 3 rows and columns are dropped)

$$[\mathbf{K}]_{6 \times 6} \xrightarrow{\text{apply BC}} [\mathbf{K}]_{3 \times 3}$$

11. Solve for nodal displacements using:

$$[\mathbf{K}]_{3 \times 3} \{d\}_{3 \times 1} = \{F\}_{3 \times 1} \quad \{d\}_{3 \times 1} = [\mathbf{K}]_{3 \times 3}^{-1} \{F\}_{3 \times 1}$$

$$\begin{Bmatrix} u_2 \\ u_3 \\ v_3 \end{Bmatrix} = \begin{Bmatrix} 0.0005 \\ 0.00025 \\ -0.000866 \end{Bmatrix}$$

It can be seen that CST with single mesh has similar and comparable horizontal and vertical displacements than those in the case of Q4 with single element. However, displacements for 4 Q4 elements differ slightly from those obtained in CST and single Q4 element. Also, the field variables, $u(x, y)$ and $v(x, y)$ have been expressed in terms of x and y (PART B, line 164 in the attached code below).

Que	Max. Horizontal Displacement, u_{max}	Max. Vertical Displacement, v_{max}
1	+0.0005 (nodes 2, 3)	-0.0010 (nodes 3, 4)
2	+0.0008 (node 3 = node 2 in que 1)	-0.0016 (node 7 = node 3 in que 1)
3	+0.0005 (node 2)	-0.000866 (node 3)

Listing 3: Nodal displacements of a CST element with one mesh

```

1 %% FEM, HW 3, Problem 3
2 %% Afnan Mostafa
3 %% 11/28/2023
4
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% clearing space %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 clc
8 clear
9 close all
10 rng('shuffle')
11
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% symbolic math %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13
14 syms r s r1 s1 r2 s2 r3 s3 u1 u2 u3 v1 v2 v3 E_0 nu t l x1 y1 x2 y2 x3
    y3 x_xy y_xy
15
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% required variables %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17
18 E_0 = 1e6;
19 nu = 0.5;
20 t = 0.001;
21 l = 1;
22 totNodes = 3;
23 % xyMat = [x1 y1; x2 y2; x3 y3];
24 xyMat = [0 0; 1 0; 1/2 (sqrt(3)*1)/2]; %% xy coords of equilateral
    triangle
25 x1 = xyMat(1,1); y1 = xyMat(1,2);
26 x2 = xyMat(2,1); y2 = xyMat(2,2);
27 x3 = xyMat(3,1); y3 = xyMat(3,2);
28
29 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% coordinate: bottom left %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30
31 xyMat = [x1 y1; x2 y2; x3 y3];
32 N1 = 1-r-s; N2 = r; N3 = s;
33
34 x = N1*x1+N2*x2+N3*x3;
35 y = N1*y1+N2*y2+N3*y3;
36
37 J11 = diff(x,r);
38 J21 = diff(x,s);
39 J12 = diff(y,r);
40 J22 = diff(y,s);
41
42 J = [J11 J12;
43      J21 J22];
44
45 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Jacobian Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
46
47 invJ = inv(J);
48 Zer = zeros(size(invJ));
49
50 u = N1*u1+N2*u2+N3*u3;
51 v = N1*v1+N2*v2+N3*v3;
52
53 du_dr = diff(u,r);
54 du_ds = diff(u,s);
55 dv_dr = diff(v,r);
56 dv_ds = diff(v,s);
57
58 dU_iso = [du_dr; du_ds];
59 dV_iso = [dv_dr; dv_ds];
60
61 dU_real = invJ*dU_iso;

```



```

62 dV_real = invJ*dV_iso;
63
64 du_dx = dU_real(1); %% epsilon_xx
65 du_dy = dU_real(2); %% epsilon_xy
66 dv_dx = dV_real(1); %% epsilon_yx
67 dv_dy = dV_real(2); %% epsilon_yy
68
69 gamma_xy = du_dy + dv_dx;
70
71 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Strain-Displacement Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
72 % solving by hand (transforming equations into matrix)
73 B = [
74     (y1-y3)-(y1-y2)      0      -(y1-y3)      0      (y1-
75         y2)      0;
76     0      ((x1-x2)-(x1-x3))      0      (x1-x3)      0
77         (x2-x1);
78     ((x1-x2)-(x3-x1))      ((y1-y3)-(y2-y1))      (x1-x3)      (y3-y1)      (x2
79         -x1)      (y1-y2)]*1/(0.866);
80 B.t = transpose(B);
81
82 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Constitutive Matrix: plane stress %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
83 E = (E_0/(1-(nu)^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];
84
85 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Stiffness Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
86 integrand = eval(B.t*E*B*t*det(J));
87
88 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Gauss Quadrature (3rd order) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
89 order = 3;
90 % [stiffMat] = GaussQuadCST(order,integrand);
91
92 switch (order)
93     case 1
94         gaussPt = [1/3 1/3];
95         wt = 1;
96     case 3
97         gaussPt = [
98             2/3 1/6;
99             1/6 2/3;
100            1/6 1/6];
101         wt = [1/3; 1/3; 1/3];
102     case 4
103         gaussPt = [
104             1/3 1/3;
105             3/5 1/5;
106             1/5 1/5;
107             1/5 3/5];
108         wt = [-27/48; 25/48; 25/48; 25/48];
109         % case 5
110     otherwise
111         error("Can't do more than 4th order GQ")
112 end
113
114 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% apply gauss quadrature %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
115
116 isIntegrandConst = 1; % for CSTs
117 if isIntegrandConst == 0
118     gq = [];
119     integral = [];
120     for i=1:length(integrand)
121         for j=1:length(integrand)
122             func = integrand(i,j);
123             for k=1:length(gaussPt)
124                 r = gaussPt(k,1);
125                 s = gaussPt(k,2);
126                 gq(k) = eval(subs(func))*wt(k,1);
127             end
128             integral(i,j) = sum([gq]);
129         end
130     end
131 else
132     stiffMat = integrand*(1*1*1);

```

```

133 end
134
135 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% disp , force matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
136
137 dsplmnt = [u1;v1;u2;v2;u3;v3];
138 force = [0;0.5; 0;0.5; 0;-1];
139
140 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% apply BCs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
141
142 % solved by handw
143 disp_BC = [u2;u3;v3];
144 force_BC = [0;0;-1];
145
146 new_integral = stiffMat;
147
148 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% remove K singularity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
149
150 % dynamic deletion (matrix index changes with each progressive line)
151 new_integral(:,1) = []; % delete 1st column
152 new_integral(:,1) = []; % delete 1st column (2nd col. of original) of
    mod. matrix
153 new_integral(:,2) = []; % delete 2st column (4th col. of original) of
    mod. matrix
154 new_integral(1,:) = []; % same as above but for rows
155 new_integral(1,:) = [];
156 new_integral(2,:) = [];
157
158 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% solve Kd = f %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
159
160 nod_disp = new_integral\force_BC; % (inverse)
161
162 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
163
164 %% PART B
165 %% u(x,y) , v(x,y)
166 N_xy = [1 x_xy y_xy]*inv([1 x1 y1; 1 x2 y2; 1 x3 y3]);
167 u1 = 0; v1 = 0; u2 = nod_disp(1); v2 = 0; u3 = nod_disp(2); v3 =
    nod_disp(3);
168 u_xy = u1*N_xy(1,1) + u2*N_xy(1,2) + u3*N_xy(1,3);
169 v_xy = v1*N_xy(1,1) + v2*N_xy(1,2) + v3*N_xy(1,3);
170
171 %% if I plug in x, y values into u and v fields , I get right answers
172 double(subs(u_xy, [x_xy,y_xy], [1,0])) == nod_disp(1)
173 double(subs(u_xy, [x_xy,y_xy], [0.5,0.866])) == nod_disp(2)
174 abs(double(subs(v_xy, [x_xy,y_xy], [0.5,0.866]))-nod_disp(3)) %% very
    low number
175
176 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot system %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
177
178 % plots the original system
179 showCST = 1;
180 if showCST == 1
181     hold on
182     line([0,1],[0,0], 'LineWidth',3, 'Color','b');
183     line([0,0.5],[0,sqrt(3)/2], 'LineWidth',3, 'Color','b');
184     line([0.5,1],[sqrt(3)/2,0], 'LineWidth',3, 'Color','b');
185 end
186
187 % plots the deformed system
188 plotDeform=1;
189 if plotDeform
190     x = [0; 1; 0.5; 0];
191     y = [0; 0; sqrt(3)/2; 0];
192     disp_u = [0; subs(nod_disp(1)); subs(nod_disp(2)); 0];
193     disp_v = [0; 0; subs(nod_disp(3)); 0];
194     pt1 = [0 1]; pt2 = [1 0]; pt3 = [0.5 sqrt(3)/2];
195     defX = x + disp_u;
196     defY = y + disp_v;
197     hold on
198     box on
199     plot(defX, defY, 'r-', 'LineWidth', 1.5);
200     set(gca, 'FontName', 'Garamond', 'FontSize', 18, 'FontWeight', 'bold', ...
201         'LineWidth', 2, 'XMinorTick', 'off', ...
202         'YMinorTick', 'off', 'GridAlpha', 0.07, ...

```

```

203         'GridLineStyle','—','LineWidth',2);
204
205     title('Deformation Plot in Real Space (global coordinate system)');
206     xlabel('X');
207     ylabel('Y');
208 end
209 subs(nod_disp);
210 allDisp = [0 0; subs(nod_disp(1)) 0; subs(nod_disp(2)) subs(nod_disp(3))
            ];];
211
212 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot displacement contours %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
213
214 % plots contour
215 isContour=0;
216 if isContour
217
218     % side lengths of quadrilateral
219     xlo = 0; ylo = 0;
220     xhi = 1; yhi = 1;
221
222     % no of nodes from the size of coordinate matrix (for plotting
223     % purposes)
224     reShapingSize = 3;
225
226     % reshape u and v matrices for contouring
227     uMat = reshape(allDisp(:,1),reShapingSize,reShapingSize)';
228     vMat = reshape(allDisp(:,2),reShapingSize,reShapingSize)';
229
230     % mesh a grid between [xlo, xhi] and [ylo, yhi]
231     [coordsX, coordsY] = meshgrid(linspace(xlo, xhi, reShapingSize),
232                                   linspace(ylo, yhi, reShapingSize));
233
234     hold on
235     box on
236
237     % subplot(2,1,1);
238     contour(coordsX, coordsY, uMat, 20,'LineWidth',2);
239     set(gca,'FontName','Garamond','FontSize',18,'FontWeight','bold',...
240         'LineWidth',2,'XMinorTick','off',...
241         'YMinorTick','off','GridAlpha',0.07,...
242         'GridLineStyle','—','LineWidth',2);
243
244     title('Contour Plot: u');
245     xlabel('X');
246     ylabel('Y');
247     colorbar;
248
249     figure;
250     % subplot(2,1,2);
251     contour(coordsX, coordsY, vMat, 20,'LineWidth',2);
252     box on
253     title('Contour Plot: v');
254     xlabel('X');
255     ylabel('Y');
256     colorbar;
257
258     set(gca,'FontName','Garamond','FontSize',16,'FontWeight','bold',...
259         'LineWidth',2,'XMinorTick','off',...
260         'YMinorTick','off','GridAlpha',0.07,...
261         'GridLineStyle','—','LineWidth',2);
262 end
263
264 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% print out nodal displ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
265
266 disp('Displacement Matrix: ')
267 eval(subs(allDisp))
268
269 [id_v] = find(ismember(abs(allDisp(:,2)), max(abs(allDisp(:,2)))));
270 [id_u] = find(ismember(abs(allDisp(:,1)), max(abs(allDisp(:,1)))));
271
272 sprintf('Max Vertical displacement occurs at node %d: %0.6f units', id_v,
273         allDisp(id_v,2))
274 sprintf('Max Horizontal displacement occurs at node %d: %0.6f units',
275         id_u, allDisp(id_u,1))

```

```

272 |
273 | %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Output of the code:

Max Vertical displacement occurs at node 3: -0.000866 units.

Max Horizontal displacement occurs at node 2: 0.0005 units

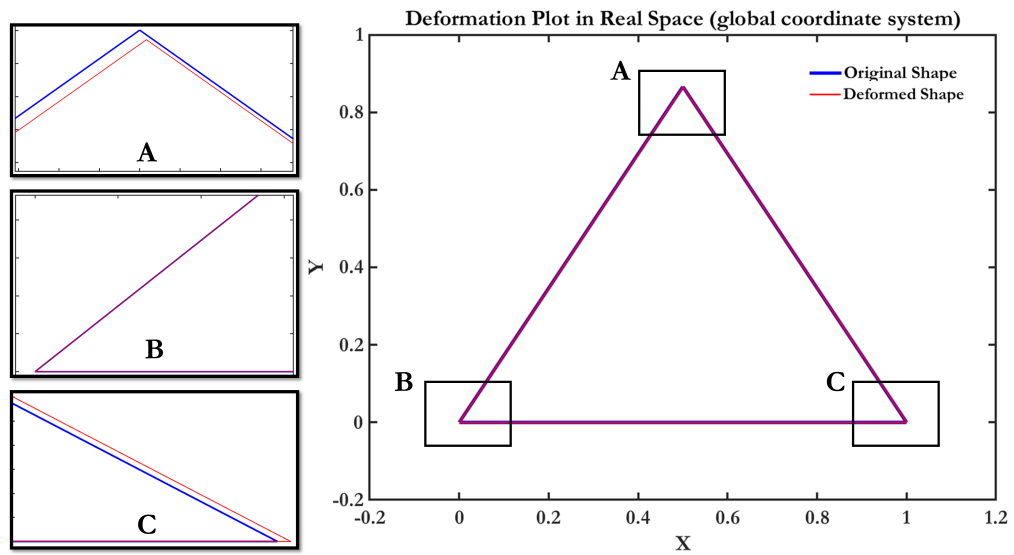


Figure 7: Deformation Plot of a CST element with zoomed-in views of the nodal displacements

Problem 4

Steps:

1. Get shape functions, $[N]$, for Q4:

$$[N] = [x][A]^{-1}$$

$$\begin{aligned}
 [N_1 \quad N_2 \quad N_3 \quad N_4]_{1 \times 4} &= [1 \quad x \quad y \quad xy]_{1 \times 4} \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \\ 1 & x_3 & y_3 & x_3 y_3 \\ 1 & x_4 & y_4 & x_4 y_4 \end{bmatrix}_{4 \times 4}^{-1} \\
 [N_1 \quad N_2 \quad N_3 \quad N_4]_{1 \times 4} &= [1 \quad x \quad y \quad xy]_{1 \times 4} \begin{bmatrix} 1 & -0.5 & -0.5 & (-0.5)(-0.5) \\ 1 & 0.5 & -0.25 & 0.5(-0.25) \\ 1 & 0.5 & 0.25 & 0.5(0.25) \\ 1 & -0.5 & 0.5 & (-0.5)0.5 \end{bmatrix}_{4 \times 4}^{-1} \\
 &= [(xy - \frac{y}{2} - \frac{x}{2} + \frac{1}{4}) \quad (\frac{x}{2} - y - 2xy + \frac{1}{4}) \quad (\frac{x}{2} + y + 2xy + \frac{1}{4}) \quad (\frac{y}{2} - \frac{x}{2} - xy + \frac{1}{4})]
 \end{aligned}$$

2. Get $[N_{,x}]_{1 \times 4}$ and $[N_{,y}]_{1 \times 4}$ through derivative of shape functions w.r.t. x and y .
3. Integrate the integrand over $[y_1, y_2] = [-\frac{3}{8} + \frac{x}{4}, \frac{3}{8} - \frac{x}{4}]$ and $[x_1, x_2] = [-0.5, 0.5]$ domains to get stiffness matrix, $[\mathbf{K}]_{4 \times 4}$

$$\begin{aligned}
 [\mathbf{K}]_{4 \times 4} &= \int_{\Omega} (N_{,x}^T N_{,x} + N_{,y}^T N_{,y}) d\Omega \\
 [\mathbf{K}]_{4 \times 4} &= \begin{bmatrix} 0.5182 & -0.0156 & -0.3594 & -0.1432 \\ -0.0156 & 1.1771 & -0.8021 & -0.3594 \\ -0.3594 & -0.8021 & 1.1771 & -0.0156 \\ -0.1432 & -0.3594 & -0.0156 & 0.5182 \end{bmatrix}
 \end{aligned}$$

4. Apply BC (removing 2nd and 3rd rows and columns) to get $[\mathbf{K}]_{2 \times 2}$ and then use:

$$[\mathbf{K}]_{2 \times 2} = \begin{bmatrix} 0.5182 & -0.1432 \\ -0.1432 & 0.5182 \end{bmatrix}$$

5. Solve for $\{\phi\}_{2 \times 1}$:

$$\{\phi\}_{2 \times 1} = [\mathbf{K}]_{2 \times 2}^{-1} \{f\}_{2 \times 1}$$

6. We get ϕ_1 and ϕ_4 from solving the above equation and use them to find r_2 and r_3 :

$$\{f\}_{4 \times 1} = [\mathbf{K}]_{4 \times 4} \{\phi\}_{4 \times 1}$$

where,

$$\{f\}_{4 \times 1} = \begin{Bmatrix} u_0/2 \\ r_2 \\ r_3 \\ u_0/2 \end{Bmatrix}, \{\phi\}_{4 \times 1} = \begin{Bmatrix} 4u_0/3 \\ 0 \\ 0 \\ 4u_0/3 \end{Bmatrix}$$

7. Then using r_2 , we find velocity $\phi_{,x}$ at right vertical boundary (here, $l=1$, $m=0$, $\phi_{,x} \neq 0$ for right vertical boundary and $l=0$, $m=-1$, but $\phi_{,y} = 0$ for bottom boundary):

$$r_2 = \int_{-1}^1 N_2(\phi_{,x}l + \phi_{,y}m)ds$$

$$\phi_{,x} = r_2/\text{Area}$$

$$\phi_{,x} = (-0.5u_0)/0.25 = -2u_0$$

$$|\phi_{,x}| = 2u_0$$

Similarly,

$$r_3 = \int_1^{-1} N_3(\phi_{,x}l + \phi_{,y}m)ds$$

$$\phi_{,x} = r_3/\text{Area}$$

$$\phi_{,x} = (-0.5u_0)/0.25 = -2u_0$$

$$|\phi_{,x}| = 2u_0$$

```

1  %% =====
2  %% ME 441: FEM, HW 3, Problem 4
3  %% Afnan Mostafa
4  %% 12/05/2023
5  %% =====
6
7  %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% clearing space %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8
9  clear
10 clc
11 close all
12 rng('shuffle')
13
14 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% material properties %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 E_0 = 1e6;
17 nu = 0.5;
18 t = 0.001;
19 u_0 = 1;
20
21 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% symbolic math %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22
23 syms x y x1 y1 x2 y2 x3 y3 x4 y4 u_0 r2 r3 phi_x phi_y
24
25 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% coordinate: center %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26
27 x1 = -0.5; y1 = -0.5;
28 x2 = 0.5; y2 = -1/4;
29 x3 = 0.5; y3 = 1/4;
30 x4 = -0.5; y4 = 0.5;
31
32 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Shape functions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
33
34 X = [1 x y x*y];
35 A = [
36     1 x1 y1 x1*y1;
37     1 x2 y2 x2*y2;
38     1 x3 y3 x3*y3;
39     1 x4 y4 x4*y4];
40
41 N = X*(inv(A));
42
43 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Integrand %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
44
45 N_dx = diff(N,x);

```

```

46 N_dx_T = transpose(N_dx);
47
48 N_dy = diff(N,y);
49 N_dy_T = transpose(N_dy);
50
51 integrand = (N_dx_T*N_dx) + (N_dy_T*N_dy);
52
53 %% %%%%%%%%% Integral = Stiffness Matrix %%%%%%%%%
54
55 integral_1 = int(integrand,y,[-3/8)+(x/4), (3/8)-(x/4]);
56 K = double(int(integral_1,x,[-0.5 0.5]));
57
58 K_BC = K;
59
60 %% %%%%%%%%% Apply Boundary Conditions %%%%%%%%%
61
62 K_BC(:,2) = [];
63 K_BC(:,2) = [];
64 K_BC(2,:) = [];
65 K_BC(2,:) = [];
66
67 %% %%%%%%%%% Phi matrix from phi = K*f %%%%%%%%%
68
69 phiMat_BC = K_BC\[u_0/2; u_0/2];
70
71 phi_all = [phiMat_BC(1); 0; 0; phiMat_BC(2)];
72
73 %% %%%%%%%%% Force matrix %%%%%%%%%
74
75 force = [u_0/2; r2; r3; u_0/2];
76 force_mat = K*phi_all;
77
78 % integration of N w.r.t. dy gives area of the N-y plot
79 Area = (1*1/2)/2;
80
81 %% l = 1, m = 0 at right vertical boundary
82
83 u_2 = force_mat(2)/(Area); %% ANSWER
84 u_3 = force_mat(3)/(Area); %% ANSWER
85 fprintf('Velocity at node 2 (bottom right) is: %s \n', char(u_2))
86 fprintf('Velocity at node 2 is: %s units \n', char(subs(u_2,[u_0],[1])))
87 fprintf('Velocity at node 3 (top right) is: %s \n', char(u_3))
88 fprintf('Velocity at node 3 is: %s units \n', char(subs(u_3,[u_0],[1])))

```

Output:

Velocity at right vertical boundary is: $-2u_0$

Velocity at right vertical boundary is: -2 units (if $u_0 = 1$ unit)

Appendix A

Listing 4: MATLAB auxiliary function to get Jacobian matrix of a Q4 element

```

1 function [J,invJ,betaMat] = JacobianMatQ4(xyMat)
2 %% written by Afnan Mostafa as part of ME 441 at UR
3 %%JacobianMat evaluates the Jacobian matrix for Q4 in isoparametric space
4 %
5 % takes the [x y] matrix (4x2) and multiplies it with a prefactor and
6 % another matrix (2x4) that consists of the derivatives of shape
7 % functions w.r.t. eta and n. Also, it calculates the inverse of
8 % Jacobian
9 % and then assembles the Beta matrix needed for stiffness matrix of Q4
10 % elements in isoparametric space.
11 %
12 % input: [x y] matrix
13 % outputs: Jacobian matrix, inverse Jacobian matrix, and Beta Matrix
14 %% sanity check for symbolic e,n
15 % if sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) == 2
16 %
17 % elseif sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) < 2
18 %     syms n e
19 % end
20
21 %% main body function
22 J = (1/4)*[-(1-n) (1-n) (1+n) -(1+n);
23           -(1-e) -(1+e) (1+e) (1-e)]*xyMat;
24
25 invJ = inv(J);
26
27 Zer = zeros(size(invJ));
28 betaMat = [invJ Zer; Zer invJ];
29
30 end

```

Appendix B

Listing 5: MATLAB auxiliary function to get integrand (matrix function inside the integral of the stiffness matrix of a Q4 element

```

1 function [integrand,B,B_t] = IntegrandStiffMatQ4(xyMat,t,E_0,nu,
2           isPlaneStrain,isPlaneStress)
3 %% written by Afnan Mostafa as part of ME 441 at UR
4 %%IntegrandStiffMatQ4 evaluates the "integrand" inside the integral of
5 % stiffness matrix of Q4 elements in isoparametric space
6 %
7 % takes the coords matrix, thickness, Poisson's ratio, Young's Mod, and
8 % either plane strain or stress condition and then calls JacobianMatQ4
9 % matrix to compute strain-displacement matrix (B) and integrand
10 %
11 % JacobianMatQ4 takes the beta matrix (4x4) and Jacobian (2x2) and then
12 % does: transpose(B)*E*B*thickness*determinant(J)
13 % Please Note: B ~= betaMat
14 % B = strain-displacement matrix, betaMat = matrix of derivatives of N
15 %
16 % inputs: [x y] matrix, thickness (t), Poisson's ratio (nu), Young's
17 % Mod
18 % (E_0), either plane strain or stress condition (just use 1 or
19 % 0)
20 % ex1: IntegrandStiffMatQ4(xyMat,0.001,1e6,0.5,0,1)
21 % ex2: IntegrandStiffMatQ4(xyMat,0.001,1e6,0.5,1,0)
22 % ex3: IntegrandStiffMatQ4(xyMat,0.001,1e6,0.5,0,0) (plane
23 % stress
24 % by default)
25 % ex4: IntegrandStiffMatQ4(xyMat,0.001,1e6,0.5,1,1) (ERROR)
26 %
27 % outputs: integrand, B, transpose of B,

```



```

24
25 %% hard-coding plane conditions (commented-out)
26
27 %isPlaneStrain = 0;
28 %isPlaneStress = 1;
29
30 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% sanity check for plane conds
31 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32 if sum([isPlaneStrain, isPlaneStress]) == 2
33     error("Can't use both plane strain and plane stress, use any one")
34 elseif sum([isPlaneStrain, isPlaneStress]) == 1
35     % do nothing
36 else
37     warning('Choosing plane stress condition by default')
38 end
39
40 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% sanity check for symbolic e,n
41 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42 % if sum([strcmp(class(n), 'sym'), strcmp(class(e), 'sym')]) == 2
43 %     % do nothing
44 % elseif sum([strcmp(class(n), 'sym'), strcmp(class(e), 'sym')]) < 2
45 %     syms n e
46 % end
47
48 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% call JacobianMatQ4 function
49 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50 [J,~, betaMat] = JacobianMatQ4(xyMat);
51
52 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Alpha Matrix
53 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54 alphaMat = [1 0 0 0; 0 0 0 1; 0 1 1 0];
55
56 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Beta Matrix
57 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
58 %betaMat = [[invJ] [Zer]; [Zer] [invJ]]; %% no need to redefine
59
60 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Gamma Matrix
61 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
62 gammaMat = (1/4)*[-1+n 0 1-n 0 1+n 0 -1-n 0;
63 -1+e 0 -1-e 0 1+e 0 1-e 0;
64 0 -1+n 0 1-n 0 1+n 0 -1-n;
65 0 -1+e 0 -1-e 0 1+e 0 1-e];
66
67 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Strain-Displacement Matrix
68 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69 B = alphaMat*betaMat*gammaMat;
70 B_t = transpose(B);
71
72 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Constitutive Matrix: plane stress
73 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74 if isPlaneStress == 1 && isPlaneStrain == 0
75     E = (E_0/(1-(nu)^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];
76 elseif isPlaneStrain == 1 && isPlaneStress == 0
77     E = (E_0/((1+nu)*(1-2*nu)))*[1-nu nu 0; nu 1-nu 0; 0 0 0.5-nu];
78 else
79     disp('Choosing plane stress condition by default')
80     E = (E_0/(1-(nu)^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];
81 end
82
83 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Stiffness Matrix
84 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
85 integrand = eval(B_t*E*B*t*det(J));
86
87 end

```

Appendix C

Listing 6: MATLAB auxiliary function to get the Gauss Quadrature integration of a Q4 element

```

1 function [stiffMat] = GaussQuadQ4(order, integrand)
2 %% written by Afnan Mostafa as part of ME 441 at UR
3 %%GaussQuadQ4 evaluates the gauss integral for Q4 in isoparametric space
4 to
5 %obtain the stiffness matrix of a Q4 element.
6 %
7 % takes the GQ order (either 1 or 2 or 3) and integrand obtained from
8 % the
9 % IntegrandStiffMatQ4.m file (function file) and then performs the GQ
10 % integration to obtain stiffness matrix (integral)
11 %
12 % input: GQ order and integrand matrix in terms of e,n
13 % outputs: Stiffness Matrix
14
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% sanity check for symbolic e,n
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17
18 % if sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) == 2
19 %
20 % elseif sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) < 2
21 %     syms n e
22 % end
23
24 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% gauss points and weights for 2d integration
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26
27 switch (order)
28 case 1
29     gaussPt = 0; wt = 4;
30 case 2
31     gaussPt = [
32         -1/sqrt(3) -1/sqrt(3);
33         1/sqrt(3) -1/sqrt(3);
34         1/sqrt(3) 1/sqrt(3);
35         -1/sqrt(3) 1/sqrt(3)];
36     wt = [
37         1 1;
38         1 1;
39         1 1;
40         1 1];
41 case 3
42     gaussPt = [
43         0 0;
44         0 -sqrt(3/5);
45         0 sqrt(3/5);
46         sqrt(3/5) 0;
47         -sqrt(3/5) 0;
48         sqrt(3/5) sqrt(3/5);
49         sqrt(3/5) -sqrt(3/5);
50         -sqrt(3/5) sqrt(3/5);
51         -sqrt(3/5) -sqrt(3/5)];
52     wt = [
53         8/9 8/9;
54         8/9 5/9;
55         8/9 5/9;
56         5/9 8/9;
57         5/9 8/9;
58         5/9 5/9;
59         5/9 5/9;
60         5/9 5/9;
61         5/9 5/9];
62 % case 4
63 otherwise
64     error("Can't do more than 3rd order GQ")
65 end
66
67 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% apply gauss quadrature
68 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

64
65 gq = [];
66 integral = [];
67 for i=1:length(integrand)
68     for j=1:length(integrand)
69         func = integrand(i,j);
70         for k=1:length(gaussPt)
71             e = gaussPt(k,1);
72             n = gaussPt(k,2);
73             gq(k) = eval(subs(func))*wt(k,1)*wt(k,2);
74         end
75         integral(i,j) = sum([gq]);
76     end
77 end
78 stiffMat = integral;
79 end

```

Appendix D

Listing 7: MATLAB auxiliary function to globalize any local stiffness matrix

```

1 function [mat3] = globalizeStiffMat(mat,posNodes,eleNodes,totNodes,DoF)
2 %globalizeStiffMat = globalizes element matrix if given nodes (CCW)
3 % mat = matrix for globalization, posNodes = nodal positions (CCW) from
4 % bottom left, eleNodes = how many nodes in an element, totNodes = total
5 % nodes in the entire system, DoF = 2 for u, v (per node)
6 %
7 %
8 %%
9 mat_cp = mat;
10 diffDOF = totNodes*DoF - length(mat);
11 mat_cp(end+diffDOF,:) = 0;
12 mat_cp(:,end+diffDOF) = 0;
13 mat2=zeros(size(mat_cp));
14
15 %% %%%%%%%%%%%%%%%%%%%%%%%%%% rearrange columns %%%%%%%%%%%%%%%%%%%%%%%%%%
16 posDisp = [(posNodes.*2)-1; posNodes.*2];
17 p=0;
18 for m=1:eleNodes
19     for n=1:DoF
20         mat2(:,posDisp(n,m)) = mat_cp(:,n+(2*p));
21     end
22     p=p+1;
23 end
24
25 %% %%%%%%%%%%%%%%%%%%%%%%%%%% rearrange rows %%%%%%%%%%%%%%%%%%%%%%%%%%
26 mat3=zeros(size(mat2));
27 p=0;
28 for m=1:eleNodes
29     for n=1:DoF
30         mat3(posDisp(n,m),:) = mat2(n+(2*p),:);
31     end
32     p=p+1;
33 end
34 end

```