# ME 441: FEM Homework Assignment 3


Afnan Mostafa


December 2023

## Problem 1

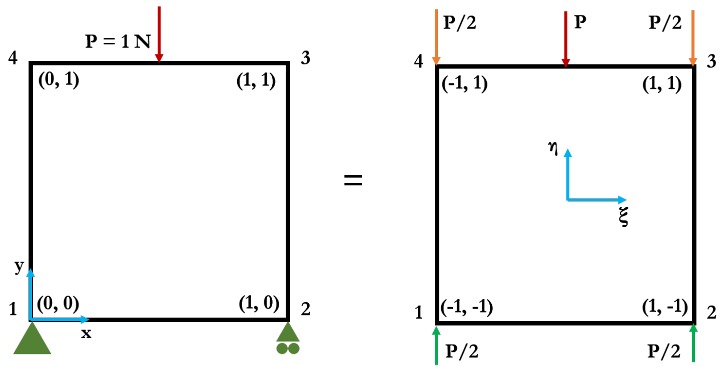


Figure 1: Q4 element

Here,

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

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

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

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

Now,

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

$$[\alpha]_{3\times4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad [\beta]_{4\times4} = \begin{bmatrix} [J]^{-1}_{2\times2} & [0]_{2\times2} \\ [0]_{2\times2} & [J]^{-1}_{2\times2} \end{bmatrix}$$

$$[\gamma]_{4\times8} = \frac{1}{4}\begin{bmatrix} -1+\eta & 0 & 1-\eta & 0 & 1+\eta & 0 & -1-\eta & 0 \\ -1+\xi & 0 & -1-\xi & 0 & 1+\xi & 0 & 1-\xi & 0 \\ 0 & -1+\eta & 0 & 1-\eta & 0 & 1+\eta & 0 & -1-\eta \\ -0 & -1+\xi & 0 & -1-\xi & 0 & 1+\xi & 0 & 1-\xi \end{bmatrix}$$

$$[\mathrm{B}]_{3\times8} = [\alpha]_{3\times4}[\beta]_{4\times4}[\gamma]_{4\times8} \qquad [K]_{8\times8} = \int_{-1}^{1}[B]^T_{8\times3}[E]_{3\times3}[B]_{3\times8}t|J|\,d\xi d\eta$$

$$[\mathrm{B}]_{3\times8} = \begin{bmatrix} \frac{\eta}{2}-\frac{1}{2} & 0 & \frac{1}{2}-\frac{\eta}{2} & 0 & \frac{\eta}{2}+\frac{1}{2} & 0 & -\frac{\eta}{2}-\frac{1}{2} & 0 \\ 0 & \frac{\xi}{2}-\frac{1}{2} & 0 & -\frac{\xi}{2}-\frac{1}{2} & 0 & \frac{\xi}{2}+\frac{1}{2} & 0 & \frac{1}{2}-\frac{\xi}{2} \\ \frac{\xi}{2}-\frac{1}{2} & \frac{\eta}{2}-\frac{1}{2} & -\frac{\xi}{2}-\frac{1}{2} & \frac{1}{2}-\frac{\eta}{2} & \frac{\xi}{2}+\frac{1}{2} & \frac{\eta}{2}+\frac{1}{2} & \frac{1}{2}-\frac{\xi}{2} & -\frac{\eta}{2}-\frac{1}{2} \end{bmatrix}$$

$$[\mathrm{K}]_{8\times8} = \int_{-1}^{1}[B]^T_{8\times3}[E]_{3\times3}[B]_{3\times8}t|J|\,d\xi d\eta = \int_{-1}^{1}[\text{integrand}]_{8\times8}\,d\xi d\eta$$

Using 2nd order Gauss Quadrature to solve the integral:

$$[K]_{8\times8} = \begin{pmatrix} \frac{5000}{9} & 250 & -\frac{3500}{9} & \frac{250}{3} & -\frac{2500}{9} & -250 & \frac{1000}{9} & -\frac{250}{3} \\ 250 & \frac{5000}{9} & -\frac{250}{3} & \frac{1000}{9} & -250 & -\frac{2500}{9} & \frac{250}{3} & -\frac{3500}{9} \\ -\frac{3500}{9} & -\frac{250}{3} & \frac{5000}{9} & -250 & \frac{1000}{9} & \frac{250}{3} & -\frac{2500}{9} & 250 \\ \frac{250}{3} & \frac{1000}{9} & -250 & \frac{5000}{9} & -\frac{250}{3} & -\frac{3500}{9} & 250 & -\frac{2500}{9} \\ -\frac{2500}{9} & -250 & \frac{1000}{9} & -\frac{250}{3} & \frac{5000}{9} & 250 & -\frac{3500}{9} & \frac{250}{3} \\ -250 & -\frac{2500}{9} & \frac{250}{3} & -\frac{3500}{9} & 250 & \frac{5000}{9} & -\frac{250}{3} & \frac{1000}{9} \\ \frac{1000}{9} & \frac{250}{3} & -\frac{2500}{9} & 250 & -\frac{3500}{9} & -\frac{250}{3} & \frac{5000}{9} & -250 \\ -\frac{250}{3} & -\frac{3500}{9} & 250 & -\frac{2500}{9} & \frac{250}{3} & \frac{1000}{9} & -250 & \frac{5000}{9} \end{pmatrix}_{8\times8}$$

Now,

$$[K]_{8\times8}\{d\}_{8\times1} = \{F\}_{8\times1}$$

Applying boundary conditions,

$$\{\mathrm{d}\} = \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ u_2 \\ 0 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix} = \begin{Bmatrix} u_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix}_{5\times1} \qquad \{F\} = \begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{2x} \\ f_{2y} \\ f_{3x} \\ f_{3y} \\ f_{4x} \\ f_{4y} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0.5 \\ 0 \\ 0.5 \\ 0 \\ -0.5 \\ 0 \\ -0.5 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ -0.5 \\ 0 \\ -0.5 \end{Bmatrix}_{5\times1}$$

$$[\mathrm{K}]_{8\times8} \xrightarrow{\text{apply BC}} [K]_{5\times5}$$

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

$$\{\mathrm{d}\}_{5\times1} = [K]^{-1}_{5\times5}\{F\}_{5\times1}$$

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

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

```
%% FEM, HW 3, Problem 1
%% Afnan Mostafa
%% 11/28/2023

%% %%%%%%%%%%%%%%%%%%%% clearing space %%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear
close all
rng('shuffle')

%% %%%%%%%%%%%%%%%%%%%% material properties %%%%%%%%%%%%%%%%%%%%%

E_0 = 1e6;
nu = 0.5;
t = 0.001;

%% %%%%%%%%%%%%%%%%%%%% symbolic math %%%%%%%%%%%%%%%%%%%%%%%%%%%

syms n e x y u2 u3 v3 u4 v4

%%             |
%              V
%  4  ------------------3
%  |                    |
%  |                    |
%  |                    |
%  |                    |
%  |         #1         |
%  |                    |
%  |                    |
%  1--------------------2
%  /\                  /\
%  --------------------oo--

%% %%%%%%%%%%%%%%%%%%%% coordinate: bottom left %%%%%%%%%%%%%%%%%

xyMat = [0 0; 1 0; 1 1; 0 1];

%% %%%%%%%%%%%%%%%%%%%% coordinate: middle %%%%%%%%%%%%%%%%%%%%%%

% xyMat = [-0.5 -0.5; 0.5 -0.5; 0.5 0.5; -0.5 0.5];

%% %%%%%%%%%%%%%%%%%%%% Jacobian Matrix %%%%%%%%%%%%%%%%%%%%%%%%%

J = 1/4*[-(1-n) (1-n) (1+n) -(1+n);
     -(1-e) -(1+e) (1+e) (1-e)]*xyMat;
invJ = inv(J);
Zer = zeros(size(invJ));

%% %%%%%%%%%%%%%%%%%%%% Alpha Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%

alphaMat = [1 0 0 0; 0 0 0 1; 0 1 1 0];

%% %%%%%%%%%%%%%%%%%%%% Beta Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

betaMat = [[invJ] [Zer]; [Zer] [invJ]];

```

```
%% %%%%%%%%%%%%%%%%%%%% Gamma Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%

gammaMat = (1/4)*[-1+n 0 1-n 0 1+n 0 -1-n 0;
    -1+e 0 -1-e 0 1+e 0 1-e 0;
    0 -1+n 0 1-n 0 1+n 0 -1-n;
    0 -1+e 0 -1-e 0 1+e 0 1-e];

%% %%%%%%%%%%%%%%%%%%%% Strain-Displacement Matrix %%%%%%%%%%%%%%

B = alphaMat*betaMat*gammaMat;
B_t = transpose(B);

%% %%%%%%%%%%%%%%%%%%%% Constitutive Matrix: plane stress %%%%%%%

E = (E_0/(1-(nu)^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];

%% %%%%%%%%%%%%%%%%%%%% Stiffness Matrix %%%%%%%%%%%%%%%%%%%%%%%%

integrand = eval(B_t*E*B*t*det(J));

%% %%%%%%%%%%%%%%%%%%%% Gauss Quadrature (2nd order) %%%%%%%%%%%%

gaussPoints = [-1/sqrt(3) -1/sqrt(3);
    1/sqrt(3) -1/sqrt(3)
    1/sqrt(3) 1/sqrt(3)
    -1/sqrt(3) 1/sqrt(3)];

for i=1:length(integrand)
    for j=1:length(integrand)
        func = integrand(i,j);
        c=0;
        for k=1:4
            e = gaussPoints(k,1);
            n = gaussPoints(k,2);
            gq(k) = eval(subs(func));
        end
        integral(i,j) = gq(1)+gq(2)+gq(3)+gq(4);
    end
end

%% %%%%%%%%%%%%%%%%%%%% disp, force matrices %%%%%%%%%%%%%%%%%%%

disp = [0;0;u2;0;u3;v3;u4;v4];
force = [0;0.5;0;0.5;0;-0.5;0;-0.5];

%% %%%%%%%%%%%%%%%%%%%% apply BCs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp_BC = [u2;u3;v3;u4;v4];
force_BC = [0;0;-0.5;0;-0.5];
new_integral = integral;

%% %%%%%%%%%%%% remove K singularity %%%%%%%%%%%%%%%%%%%%%%%%%%%%

new_integral(:,1) = [];
new_integral(:,1) = [];
new_integral(:,2) = [];
new_integral(1,:) = [];
new_integral(1,:) = [];
new_integral(2,:) = [];

%% %%%%%%%%%%%%%%%%%%%% solve Kd = F %%%%%%%%%%%%%%%%%%%%%%%%%%%%

nod_disp = inv(new_integral)*force_BC;

allDisp = [
    0 0; nod_disp(1) 0; nod_disp(2) nod_disp(3); nod_disp(4) nod_disp(5)
        ];

%% %%%%%%%%%%%%%%%%%%%% print out nodal displ %%%%%%%%%%%%%%%%%%%

[id_v] = find(ismember(abs(allDisp(:,2)), max(abs(allDisp(:,2)))));
[id_u] = find(ismember(abs(allDisp(:,1)), max(abs(allDisp(:,1)))));

sprintf('Max Vertical displacement occurs at node %d: %0.4f units', id_v
```

```matlab
    , allDisp(id_v,2))
sprintf('Max Horizontal displacement occurs at node %d: %0.4f units', id_u, allDisp(id_u,1))

%%             |
%              V
%  4 -------------------3
%  |                    |
%  |                    |
%  |                    |
%  |         #1         |
%  |                    |
%  |                    |
%  |                    |
%  1--------------------2
%  /\                  /\
%  ---------------------oo--

%% %%%%%%%%%%% plot original and deformed systems %%%%%%%%%%%

plotDeform=1;
if plotDeform
    hold on
    % plots the original system
    l1 = line([0,1],[0,0],'LineWidth',3,'Color','k');
    line([1,1],[0,1],'LineWidth',3,'Color','k');
    line([0,0],[0,1],'LineWidth',3,'Color','k');
    line([1,0],[1,1],'LineWidth',3,'Color','k');
    x = [0; 1; 1; 0; 0];
    y = [0; 0; 1; 1; 0];
    disp_u = [allDisp(1,1); allDisp(2,1); allDisp(3,1); allDisp(4,1); allDisp(1,1)];

    disp_v = [allDisp(1,2); allDisp(2,2); allDisp(3,2); allDisp(4,2); allDisp(1,2)];

    defX = x + disp_u;
    defY = y + disp_v;
    box on
    % plots the deformed system
    p1 = plot(defX, defY, 'r-', 'LineWidth', 1.5);
    set(gca,'FontName','Garamond','FontSize',18,'FontWeight','bold',...
        'LineWidth',2,'XMinorTick','off',...
        'YMinorTick','off','GridAlpha',0.07,...
        'GridLineStyle','--','LineWidth',2);
    title('Deformation Plot in Real Space (global coordinate system)');
    xlabel('X');
    ylabel('Y');
    legend([l1 p1],{'Original System', 'Deformed System'},'Location','southeast',...
        'Color',[0.941176470588235 0.941176470588235 0.941176470588235]);
    % set(gcf,'units','points','position',[100,100,1024,700])
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 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

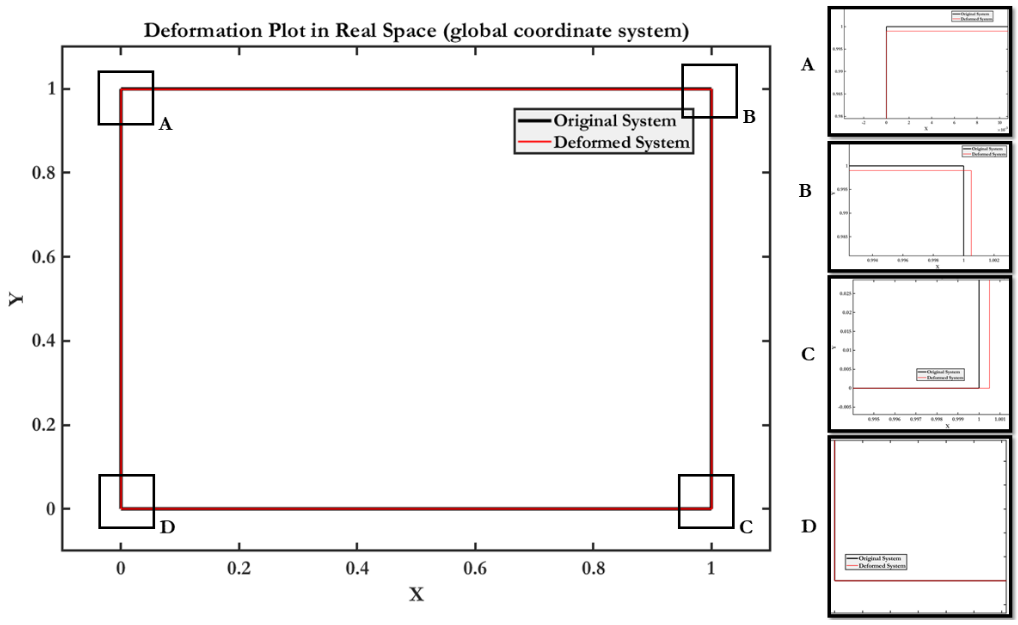


Figure 2: Deformation Plot of Q4 single element

## Problem 2

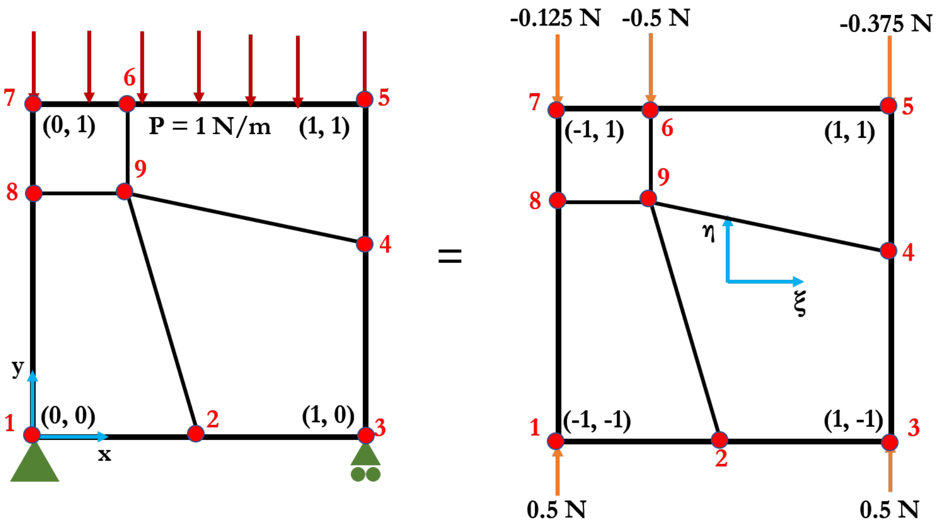


Figure 3: Q4 element with 4 meshes

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

1. Get $[J]_{2\times2}$ for each element using:

$$[J]_{2\times2} = \frac{1}{4}\begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix}_{2\times4} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}_{4\times2},$$

   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),

$$[B]^T_{8\times3}[E]_{3\times3}[B]_{3\times8}|J|t = [\text{integrand}]_{8\times8}$$

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

$$[K]_{8\times8} = \int_{-1}^{1}[B]^T_{8\times3}[E]_{3\times3}[B]_{3\times8}t|J|\,d\xi d\eta = \int_{-1}^{1}[\text{integrand}]_{8\times8}\,d\xi d\eta$$

4. Globalize element stiffness matrices, $[K_i]_{8\times8};\;\; i = 1, 2, 3, 4$ (no of elements) **(globalizeStiffMat.m)** (see Appendix D) and add them to get the global stiffness matrix $[\mathbf{K}]$

$$[K_i]_{8\times8} \xrightarrow{\text{globalize}} [\mathbf{K}_i]_{18\times18}$$

$$[\mathbf{K}_{\text{struc}}]_{18\times18} = \sum_{i=1}^{4} [\mathbf{K}_i]_{18\times18}$$

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

$$[\mathbf{K}_{\text{struc}}]_{18\times18} \xrightarrow{\text{apply BC}} [\mathbf{K}_{\text{struc}}]_{15\times15}$$

6. Solve for nodal displacements using:

$$[\mathbf{K}_{\text{struc}}]_{15\times15}\{d\}_{15\times1} = \{F\}_{15\times1} \qquad \{d\}_{15\times1} = [\mathbf{K}_{\text{struc}}]^{-1}_{15\times15}\{F\}_{15\times1}$$

$$\{d\}_{15\times1} = \begin{Bmatrix} u_2 \\ v_2 \\ u_3 \\ u_4 \\ v_4 \\ u_5 \\ v_5 \\ u_6 \\ v_6 \\ u_7 \\ v_7 \\ u_8 \\ v_8 \\ u_9 \\ v_9 \end{Bmatrix} = \begin{Bmatrix} 0.0005 \\ -0.001 \\ 0.0008 \\ 0.0006 \\ -0.0011 \\ 0.0007 \\ -0.0015 \\ 0.0003 \\ -0.0015 \\ 0.0002 \\ -0.0016 \\ 0.0003 \\ -0.0014 \\ 0.0004 \\ -0.0013 \end{Bmatrix}$$

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

```
% ==================================================================
%% ME 441: FEM, HW 3, Problem 2
%% Afnan Mostafa
%% 11/29/2023
% ==================================================================
%%
% this script needs 3 functions files: IntegrandStiffMatQ4.m,
% GaussQuadQ4.m, and globalizeStiffMat.m files (put them in the same dir
    )
%
%%
%% %%%%%%%%%%%%%%%%%%%%%%% clearing space %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
close all
```

```
rng('shuffle')

%% %%%%%%%%%%%%%%%%%%%% material properties %%%%%%%%%%%%%%%%%%

E_0 = 1e6;
nu = 0.5;
t = 0.001;

%% %%%%%%%%%%%%%%%%%%%% symbolic math %%%%%%%%%%%%%%%%%%%%%%%%%

syms n e x y u2 v2 u3 v3 u4 v4 u5 v5 u6 v6 u7 v7 u8 v8 u9 v9 X Y

%% %%%%%%%%%%%%%%%%%%%% coordinate: bottom left %%%%%%%%%%%%%%%%

elements = 4;
order = 2;
totNodes = 9;

xyMat = [0 0.75; 0.25 0.75; 0.25 1; 0 1];
xyMat2 = [0 0; 0.5 0; 0.25 0.75; 0 0.75];
xyMat3 = [0.5 0; 1 0; 1 0.6; 0.25 0.75];
xyMat4 = [0.25 0.75; 1 0.6; 1 1; 0.25 1];

%% %%%%%%%%%%%%%%%%%%%%%%%%% xy matrix %%%%%%%%%%%%%%%%%%%%%%%%%

xy = cell(1,4);
xy{1,1} = xyMat;
xy{1,2} = xyMat2;
xy{1,3} = xyMat3;
xy{1,4} = xyMat4;

%% %%%%%%%%%%%%%%%%%%%% cell creation for storage %%%%%%%%%%%%%%

kmat_all = cell(1,elements);
intgrd_all = cell(1,elements);

%% %%%%%% function callback: to get elemental stiff mat. %%%%%%%

%auxiliary function files
for p=1:elements
    [integrand,B,B_t] = IntegrandStiffMatQ4(xy{1,p},t,E_0,nu,0,1);
    intgrd_all{1,p} = integrand;
    [stiffMat] = GaussQuadQ4(order,intgrd_all{1,p});
    kmat_all{1,p} = stiffMat;
end

%% %%%%%%%%%%%%%%%%%%%% globalization starts here %%%%%%%%%%%%%%

%%    |   |    |    |    |
%     V   V    V    V    V
%  7  _______6__________5
%  |          |          |
%  |    #1    |    #4    |
%  8________9___         |
%  |         |    __     |
%  |          |      __  4
%  |    #2     |    #3   |
%  |            |        |
%  1__________2________3
%  /\                   /\
%  ____________________oo__


% order of nodes in an element matter
ele1 = [8,9,6,7];
ele2 = [1,2,9,8];
ele3 = [2,3,4,9];
ele4 = [9,4,5,6];

nod_ele = [ele1; ele2; ele3; ele4];

% function callback for globalization
for u=1:elements
    [mat_cp] = globalizeStiffMat(kmat_all{1,u},nod_ele(u,:),4,9,2);
```

```
    stiffMatSet{u,1} = mat_cp;
end

globalMat = zeros(size(mat_cp));

for v=1:elements
    globalMat = globalMat + stiffMatSet{v,1};
end

%% %%%%%%%%%%%%%%%%%%%% disp, force matrices %%%%%%%%%%%%%%%%%%

% disp = [0;0;u2;v2;u3;0;u4;v4;u5;v5;u6;v6;u7;v7;u8;v8;u9;v9];
% force = [0;0.25;0;0.5;0;0.25;0;-0.125;0;-0.75/2;0;-0.5;0;-0.25/2;0;0.1250;0;-0.5];


disp = [0;0;u2;v2;u3;0;u4;v4;u5;v5;u6;v6;u7;v7;u8;v8;u9;v9];
force = [0;0.5;0;0;0;0.5;0;0;0;-0.75/2;0;-0.5;0;-0.25/2;0;0;0;0];

%% %%%%%%%%%%%%%%%%%%%% apply BCs %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% solved by hand
disp_BC = [u2;v2;u3;u4;v4;u5;v5;u6;v6;u7;v7;u8;v8;u9;v9];
force_BC = [0;0;0;0;0;0;-0.75/2;0;-0.5;0;-0.25/2;0;0;0;0];

new_integral = globalMat;

%% %%%%%%%%%%%%%%%%%%%% remove K singularity %%%%%%%%%%%%%%%%%%

% dynamic deletion (matrix index changes with each progressive line)
new_integral(:,1) = []; % delete 1st column
new_integral(:,1) = []; % delete 1st column (2nd col. of original) of mod. matrix
new_integral(:,4) = []; % delete 4th column (6th col. of original) of mod. matrix
new_integral(1,:) = []; % same as above but for rows
new_integral(1,:) = [];
new_integral(4,:) = [];

%% %%%%%%%%%%%%%%%%%%%% solve Kd = F %%%%%%%%%%%%%%%%%%%%%%%%%%%

nod_disp = new_integral\force_BC;

%% %%%%%%%%%%%%%%%%%%%% plot displacement contours %%%%%%%%%%%

allDisp = [0 0; nod_disp(1) nod_disp(2); nod_disp(3) 0; nod_disp(4) nod_disp(5); ...
    nod_disp(6) nod_disp(7); nod_disp(8) nod_disp(9); nod_disp(10) ...
    nod_disp(11); nod_disp(12) nod_disp(13); nod_disp(14) nod_disp(15)];

% side lengths of quadrilateral
xlo = 0; ylo = 0;
xhi = 1; yhi = 1;

% no of nodes from the size of coordinate matrix (for plotting purposes)
reShapingSize = sqrt(totNodes); % totNodes = 9;

% reshape u and v matrices for contouring
uMat = reshape(allDisp(:,1),reShapingSize,reShapingSize)';
vMat = reshape(allDisp(:,2),reShapingSize,reShapingSize)';

xcoords = [0 0.5 1; 0 0.25 1; 0 0.25 1];
ycoords = [0 0 0; 0.75 0.75 0.6; 1 1 1];

hold on
box on
%% u-disp contour plot
% subplot(2,1,1);
contourf(xcoords, ycoords, uMat, 20,'LineWidth',2);
set(gca,'FontName','Garamond','FontSize',18,'FontWeight','bold',...
    'LineWidth',2,'XMinorTick','off',...
    'YMinorTick','off','GridAlpha',0.07,...
    'GridLineStyle','--','LineWidth',2);
title('Contour Plot: u');
```

```
xlabel('X');
ylabel('Y');
colorbar;

showQ4 = 1;
if showQ4 == 1
    hold on
    line([0,0.25],[0.75,0.75],'LineWidth',2,'Color','r');
    line([0.25,0.25],[0.75,1],'LineWidth',2,'Color','r');
    line([0.25,1],[0.75,0.6],'LineWidth',2,'Color','r');
    line([0.25,0.5],[0.75,0],'LineWidth',2,'Color','r');
    hold off
end
[gcf] = plotNodes(gcf,totNodes);

%% v-disp contour plot
figure;
% subplot(2,1,2);
contourf(xcoords, ycoords, vMat, 20,'LineWidth',2);
hold on
% plots the original system
if showQ4 == 1
    line([0,0.25],[0.75,0.75],'LineWidth',3,'Color','r');
    line([0.25,0.25],[0.75,1],'LineWidth',3,'Color','r');
    line([0.25,1],[0.75,0.6],'LineWidth',3,'Color','r');
    line([0.25,0.5],[0.75,0],'LineWidth',3,'Color','r');
end

set(gca,'FontName','Garamond','FontSize',18,'FontWeight','bold',...
    'LineWidth',2,'XMinorTick','off',...
    'YMinorTick','off','GridAlpha',0.07,...
    'GridLineStyle','--','LineWidth',2);
title('Contour Plot: v');
xlabel('X');
ylabel('Y');
colorbar;
[gcf] = plotNodes(gcf,totNodes);

%% %%%%%%%%%%% plot original and deformed systems %%%%%%%%%%%

plotDeform=0;
if plotDeform
    hold off
    figure;
    % plots the original system
    l1 = line([0,0.25],[0.75,0.75],'LineWidth',3,'Color','k');
    line([0.25,0.25],[0.75,1],'LineWidth',3,'Color','k');
    line([0.25,1],[0.75,0.6],'LineWidth',3,'Color','k');
    line([0.25,0.5],[0.75,0],'LineWidth',3,'Color','k');
    hold on
    x = [0; 0.5; 1; 1; 1; 0.5; 0; 0; 0];
    y = [0; 0; 0; 0.6; 1; 1; 1; 0.75; 0];
    disp_u = [allDisp(1,1); allDisp(2,1); allDisp(3,1); allDisp(4,1); ...
        allDisp(5,1); allDisp(6,1); allDisp(7,1); allDisp(8,1); allDisp(9,1)];

    disp_v = [allDisp(1,2); allDisp(2,2); allDisp(3,2); allDisp(4,2); ...
        allDisp(5,2); allDisp(6,2); allDisp(7,2); allDisp(8,2); allDisp(9,2)];

    defX = x + disp_u;
    defY = y + disp_v;
    hold on
    box on
    % plots the deformed system
    p1 = plot(defX, defY, 'c-', 'LineWidth', 1.5);
    set(gca,'FontName','Garamond','FontSize',18,'FontWeight','bold',...
        'LineWidth',2,'XMinorTick','off',...
        'YMinorTick','off','GridAlpha',0.07,...
        'GridLineStyle','--','LineWidth',2);
    title('Deformation Plot in Real Space (global coordinate system)');
    xlabel('X');
```

```
    ylabel('Y');
    legend([l1 p1],{'Original System', 'Deformed System'},'Location','southeast',...
        'Color',[0.941176470588235 0.941176470588235 0.941176470588235]);
    set(gcf,'units','points','position',[100,100,1024,700])
end

[id_v] = find(ismember(abs(allDisp(:,2)), max(abs(allDisp(:,2)))));
[id_u] = find(ismember(abs(allDisp(:,1)), max(abs(allDisp(:,1)))));

sprintf('Max Vertical displacement occurs at node %d: %0.4f units', id_v, allDisp(id_v,2))
sprintf('Max Horizontal displacement occurs at node %d: %0.4f units', id_u, allDisp(id_u,1))

%%    |   |    |    |    |
%     V   V    V    V    V
%  7  _______6__________5
%  |         |          |
%  |    #1   |     #4   |
%  8________9___        |
%  |        |    __     |
%  |         |      __  4
%  |    #2    |    #3   |
%  |           |        |
%  1__________2________3
%  /\                  /\
%  ---------------------oo--

%% %%%%%%%%%% auxiliary func to plot nodes %%%%%%%%%%%
function [gcf] = plotNodes(gcf,totNodes)
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];
str = string(1:totNodes);
plotNodes=1;
if plotNodes
    for b=1:totNodes
        text(pos(b,1),pos(b,2),str(b),'Color','red','FontSize',18,'FontName','Garamond','FontWeight','bold')
    end
end
end
%% %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 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

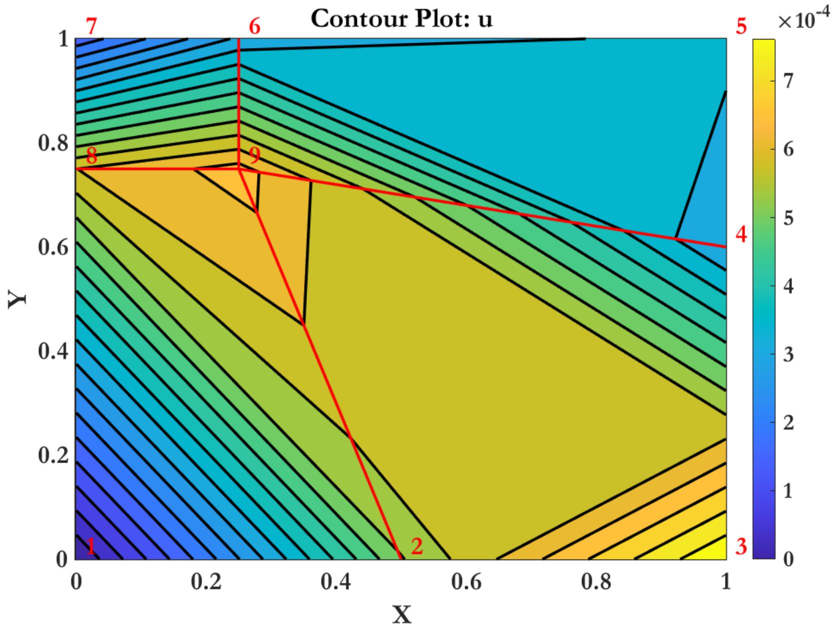


Figure 4: Contour Plot of u-displacement

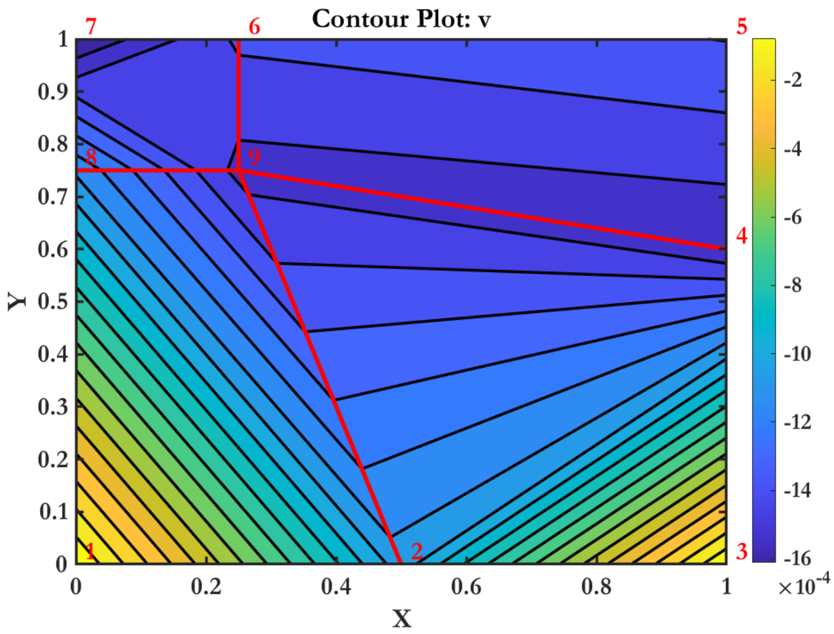


Figure 5: Contour Plot of v-displacement

## Problem 3

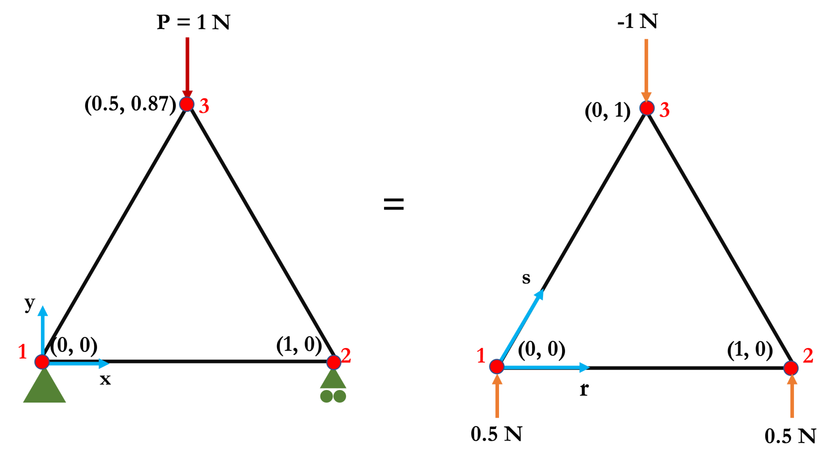


Figure 6: CST

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

1. Get shape functions, $N$, for CST:

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

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

$$x(r,s) = x_1N_1 + x_2N_2 + x_3N_3 = rx_2 - x_1(r+s-1) + sx_3$$

$$y(r,s) = y_1N_1 + y_2N_2 + y_3N_3 = ry_2 - y_1(r+s-1) + sy_3$$

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):

$$u(r,s) = u_1N_1 + u_2N_2 + u_3N_3 = ru_2 - u_1(r+s-1) + su_3$$

$$v(r,s) = v_1N_1 + v_2N_2 + v_3N_3 = rv_2 - v_1(r+s-1) + sv_3$$

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

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

$$\{v_{,real}\}_{2\times1} = \begin{bmatrix} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{bmatrix}_{2\times1} = [J]^{-1}_{2\times2} \begin{bmatrix} \frac{\partial v}{\partial r} \\ \frac{\partial v}{\partial s} \end{bmatrix}_{2\times1}$$

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

$$\{\epsilon\}_{3\times1} = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{bmatrix}_{3\times1}$$

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

$$\{\epsilon\}_{3\times1} = [B]_{3\times6}\{d\}_{6\times1}$$

8. Strain-displacement matrix is:

$$[B]_{3\times6} = \frac{1}{2A}\begin{bmatrix} y_2 - y_3 & 0 & y_3 - y_1 & 0 & y_1 - y_2 & 0 \\ 0 & x_3 - x_2 & 0 & x_1 - x_3 & 0 & x_2 - x_1 \\ 2x_1 - x_2 - x_3 & 2y_1 - y_2 - y_3 & x_1 - x_3 & y_3 - y_1 & x_2 - x_1 & y_1 - y_2 \end{bmatrix}_{3\times6}$$

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

$$[\mathbf{K}]_{6\times6} = \int_{-1}^{1} [B]^T_{6\times3}[E]_{3\times3}[B]_{3\times6}t|J|\,drds = \int_{-1}^{1} [\text{integrand}]_{6\times6}\,drds =$$

$$[\mathbf{K}]_{6\times6} = [\text{integrand}]_{6\times6} \int_{-1}^{1} drds$$

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

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

$$[\mathbf{K}]_{6\times6} \xrightarrow{\text{apply BC}} [\mathbf{K}]_{3\times3}$$

11. Solve for nodal displacements using:

$$[\mathbf{K}]_{3\times3}\{d\}_{3\times1} = \{F\}_{3\times1} \qquad \{d\}_{3\times1} = [\mathbf{K}]^{-1}_{3\times3}\{F\}_{3\times1}$$

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

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 varaibles, $u(x, y)$ 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

```
%% FEM, HW 3, Problem 3
%% Afnan Mostafa
%% 11/28/2023

%% %%%%%%%%%%%%%%%%%%%% clearing space %%%%%%%%%%%%%%%%%%%%%%

clc
clear
close all
rng('shuffle')

%% %%%%%%%%%%%%%%%%%%%% symbolic math %%%%%%%%%%%%%%%%%%%%%%%

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

%% %%%%%%%%%%%%%%%%%%%%%%% required variables %%%%%%%%%%%%%%%

E_0 = 1e6;
nu = 0.5;
t = 0.001;
l = 1;
totNodes = 3;
% xyMat = [x1 y1; x2 y2; x3 y3];
xyMat = [0 0; l 0; l/2 (sqrt(3)*l)/2]; %% xy coords of equilateral
    triangle
x1 = xyMat(1,1); y1 = xyMat(1,2);
x2 = xyMat(2,1); y2 = xyMat(2,2);
x3 = xyMat(3,1); y3 = xyMat(3,2);

%% %%%%%%%%%%%%%%%%%%%% coordinate: bottom left %%%%%%%%%%%%%

xyMat = [x1 y1; x2 y2; x3 y3];
N1 = 1-r-s; N2 = r; N3 = s;

x = N1*x1+N2*x2+N3*x3;
y = N1*y1+N2*y2+N3*y3;

J11 = diff(x,r);
J21 = diff(x,s);
J12 = diff(y,r);
J22 = diff(y,s);

J = [J11 J12;
    J21 J22];

%% %%%%%%%%%%%%%%%%%%%% Jacobian Matrix %%%%%%%%%%%%%%%%%%%%%%

invJ = inv(J);
Zer = zeros(size(invJ));

u = N1*u1+N2*u2+N3*u3;
v = N1*v1+N2*v2+N3*v3;

du_dr = diff(u,r);
du_ds = diff(u,s);
dv_dr = diff(v,r);
dv_ds = diff(v,s);

dU_iso = [du_dr; du_ds];
dV_iso = [dv_dr; dv_ds];

dU_real = invJ*dU_iso;
```

```
dV_real = invJ*dV_iso;

du_dx = dU_real(1);   %% epsilon_xx
du_dy = dU_real(2);   %% epsilon_xy
dv_dx = dV_real(1);   %% epsilon_yx
dv_dy = dV_real(2);   %% epsilon_yy

gamma_xy = du_dy + dv_dx;

%% %%%%%%%%%%%%%%%%%%%%%%%% Strain-Displacement Matrix %%%%%%%%%%%%%%%%
% solving by hand (transforming equations into matrix)
B = [
    (y1-y3)-(y1-y2)             0             -(y1-y3)          0          (y1-y2)         0;
    0            ((x1-x2)-(x1-x3))        0         (x1-x3)            0          (x2-x1);
    ((x1-x2)-(x3-x1))     ((y1-y3)-(y2-y1))     (x1-x3)     (y3-y1)      (x2-x1)      (y1-y2)]*1/(0.866);
B_t = transpose(B);

%% %%%%%%%%%%%%%%%% Constitutive Matrix: plane stress %%%%%%%%%%

E = (E_0/(1-(nu)^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];

%% %%%%%%%%%%%%%%%%%%%%%%%% Stiffness Matrix %%%%%%%%%%%%%%%%%%%%%%%%

integrand = eval(B_t*E*B*t*det(J));

%% %%%%%%%%%%%%%%%%% Gauss Quadrature (3rd order) %%%%%%%%%%%%%%%%

order = 3;
% [stiffMat] = GaussQuadCST(order,integrand);

switch (order)
    case 1
        gaussPt = [1/3 1/3];
        wt = 1;
    case 3
        gaussPt = [
            2/3 1/6;
            1/6 2/3;
            1/6 1/6];
        wt = [1/3; 1/3; 1/3];
    case 4
        gaussPt = [
            1/3 1/3;
            3/5 1/5;
            1/5 1/5;
            1/5 3/5];
        wt = [-27/48; 25/48; 25/48; 25/48];
        % case 5
    otherwise
        error("Can't do more than 4th order GQ")
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%% apply gauss quadrature %%%%%%%%%%%%%%%%%

isIntegrandConst = 1; % for CSTs
if isIntegrandConst == 0
    gq = [];
    integral = [];
    for i=1:length(integrand)
        for j=1:length(integrand)
            func = integrand(i,j);
            for k=1:length(gaussPt)
                r = gaussPt(k,1);
                s = gaussPt(k,2);
                gq(k) = eval(subs(func))*wt(k,1);
            end
            integral(i,j) = sum([gq]);
        end
    end
else
stiffMat = integrand*(1*1*1);
```

```matlab
end

%% %%%%%%%%%%%%%%%%%%%% disp, force matrices %%%%%%%%%%%%

dsplmnt = [u1;v1;u2;v2;u3;v3];
force = [0;0.5; 0;0.5; 0;-1];

%% %%%%%%%%%%%%%%%%%%%% apply BCs %%%%%%%%%%%%%%%%%%%%%%%%

% solved by handw
disp_BC = [u2;u3;v3];
force_BC = [0;0;-1];

new_integral = stiffMat;

%% %%%%%%%%%%%%%%%%%%%% remove K singularity %%%%%%%%%%%%

% dynamic deletion (matrix index changes with each progressive line)
new_integral(:,1) = []; % delete 1st column
new_integral(:,1) = []; % delete 1st column (2nd col. of original) of mod. matrix
new_integral(:,2) = []; % delete 2st column (4th col. of original) of mod. matrix
new_integral(1,:) = []; % same as above but for rows
new_integral(1,:) = [];
new_integral(2,:) = [];

%% %%%%%%%%%%%%%%%%%%%% solve Kd = f %%%%%%%%%%%%%%%%%%%%%

nod_disp = new_integral\force_BC; % (inverse)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% PART B
%% u(x,y), v(x,y)
N_xy = [1 x_xy y_xy]*inv([1 x1 y1; 1 x2 y2; 1 x3 y3]);
u1 = 0; v1 = 0; u2 = nod_disp(1); v2 = 0; u3 = nod_disp(2); v3 = nod_disp(3);
u_xy = u1*N_xy(1,1) + u2*N_xy(1,2) + u3*N_xy(1,3);
v_xy = v1*N_xy(1,1) + v2*N_xy(1,2) + v3*N_xy(1,3);

%% if I plug in x, y values into u and v fields, I get right answers
double(subs(u_xy, [x_xy,y_xy], [1,0])) == nod_disp(1)
double(subs(u_xy, [x_xy,y_xy], [0.5,0.866])) == nod_disp(2)
abs(double(subs(v_xy, [x_xy,y_xy], [0.5,0.866]))-nod_disp(3)) %% very low number

%% %%%%%%%%%%%%%%%%%%%% plot system %%%%%%%%%%%%%%%%%%%%%%

% plots the original system
showCST = 1;
if showCST == 1
    hold on
    line([0,1],[0,0],'LineWidth',3,'Color','b');
    line([0,0.5],[0,sqrt(3)/2],'LineWidth',3,'Color','b');
    line([0.5,1],[sqrt(3)/2,0],'LineWidth',3,'Color','b');
end

% plots the deformed system
plotDeform=1;
if plotDeform
    x = [0; 1; 0.5; 0];
    y = [0; 0; sqrt(3)/2; 0];
    disp_u = [0; subs(nod_disp(1)); subs(nod_disp(2)); 0];
    disp_v = [0; 0; subs(nod_disp(3)); 0];
    pt1 = [0 1]; pt2 = [1 0]; pt3 = [0.5 sqrt(3)/2];
    defX = x + disp_u;
    defY = y + disp_v;
    hold on
    box on
    plot(defX, defY, 'r-', 'LineWidth', 1.5);
    set(gca,'FontName','Garamond','FontSize',18,'FontWeight','bold',...
        'LineWidth',2,'XMinorTick','off',...
        'YMinorTick','off','GridAlpha',0.07,...
```

```
        'GridLineStyle','--','LineWidth',2);

    title('Deformation Plot in Real Space (global coordinate system)');
    xlabel('X');
    ylabel('Y');
end
subs(nod_disp);
allDisp = [0 0; subs(nod_disp(1)) 0; subs(nod_disp(2)) subs(nod_disp(3))
    ;];

%% %%%%%%%%%%%%%% plot displacement contours %%%%%%%%%%%%%%%%%%%

% plots contour
isContour=0;
if isContour

    % side lengths of quadrilateral
    xlo = 0; ylo = 0;
    xhi = 1; yhi = 1;

    % no of nodes from the size of coordinate matrix (for plotting
        purposes)
    reShapingSize = 3;

    % reshape u and v matrices for contouring
    uMat = reshape(allDisp(:,1),reShapingSize,reShapingSize)';
    vMat = reshape(allDisp(:,2),reShapingSize,reShapingSize)';

    % mesh a grid between [xlo, xhi] and [ylo, yhi]
    [coordsX, coordsY] = meshgrid(linspace(xlo, xhi, reShapingSize),
        linspace(ylo, yhi, reShapingSize));

    hold on
    box on

    % subplot(2,1,1);
    contour(coordsX, coordsY, uMat, 20,'LineWidth',2);
    set(gca,'FontName','Garamond','FontSize',18,'FontWeight','bold',...
        'LineWidth',2,'XMinorTick','off',...
        'YMinorTick','off','GridAlpha',0.07,...
        'GridLineStyle','--','LineWidth',2);

    title('Contour Plot: u');
    xlabel('X');
    ylabel('Y');
    colorbar;

    figure;
    % subplot(2,1,2);
    contour(coordsX, coordsY, vMat, 20,'LineWidth',2);
    box on
    title('Contour Plot: v');
    xlabel('X');
    ylabel('Y');
    colorbar;

    set(gca,'FontName','Garamond','FontSize',16,'FontWeight','bold',...
        'LineWidth',2,'XMinorTick','off',...
        'YMinorTick','off','GridAlpha',0.07,...
        'GridLineStyle','--','LineWidth',2);
end

%% %%%%%%%%%%%%%%%%%%%%%%%% print out nodal displ %%%%%%%%%%%%%%%%%%%

disp('Displacement Matrix: ')
eval(subs(allDisp))

[id_v] = find(ismember(abs(allDisp(:,2)), max(abs(allDisp(:,2)))));
[id_u] = find(ismember(abs(allDisp(:,1)), max(abs(allDisp(:,1)))));

sprintf('Max Vertical displacement occurs at node %d: %0.6f units', id_v
    , allDisp(id_v,2))
sprintf('Max Horizontal displacement occurs at node %d: %0.6f units',
    id_u, allDisp(id_u,1))
```

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 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

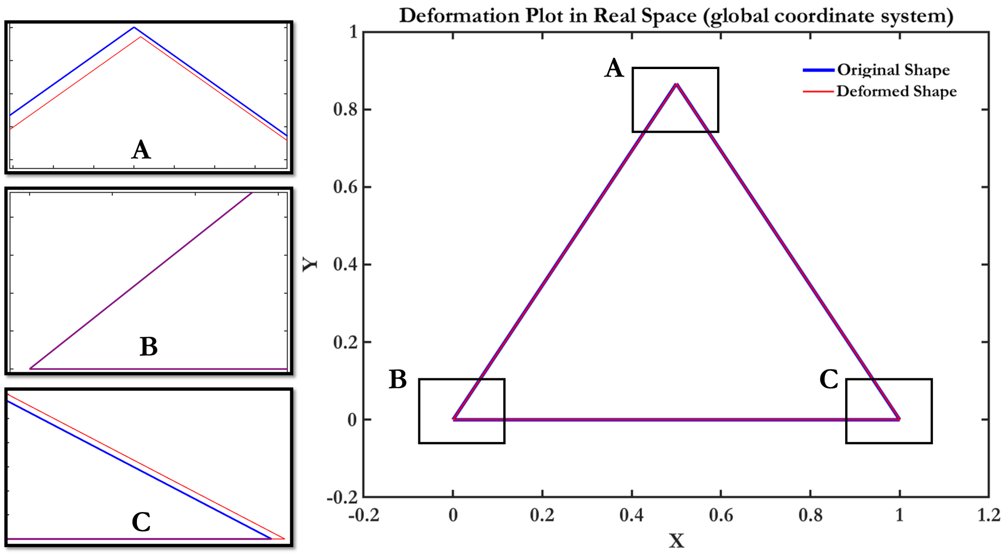


Figure 7: Deformation Plot of a CST element with zoomed-in views of the nodal displacements

## Problem 4

Steps:

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

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

$$\begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix}_{1\times 4} = \begin{bmatrix} 1 & x & y & xy \end{bmatrix}_{1\times 4} \begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \\ 1 & x_4 & y_4 & x_4y_4 \end{bmatrix}_{4\times 4}^{-1}$$

$$\begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix}_{1\times 4} = \begin{bmatrix} 1 & x & y & xy \end{bmatrix}_{1\times 4} \begin{bmatrix} 1 & -0.5 & -0.5 & (-0.5)(-0.5) \\ 1 & 0.5 & -0.25 & 0.5(-0.25) \\ 1 & 0.5 & 0.25 & 0.5(0.25) \\ 1 & -0.5 & 0.5 & (-0.5)0.5 \end{bmatrix}_{4\times 4}^{-1}$$

$$= [\ (x\,y - \tfrac{y}{2} - \tfrac{x}{2} + \tfrac{1}{4}) \quad (\tfrac{x}{2} - y - 2\,x\,y + \tfrac{1}{4}) \quad (\tfrac{x}{2} + y + 2\,x\,y + \tfrac{1}{4}) \quad (\tfrac{y}{2} - \tfrac{x}{2} - x\,y + \tfrac{1}{4})\ ]$$

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, $[\mathbf{K}]_{4\times 4}$

$$[\mathbf{K}]_{4\times 4} = \int_\Omega (N_{,x}^T N_{,x} + N_{,y}^T N_{,y}) d\Omega$$

$$[\mathbf{K}]_{4\times 4} = \begin{bmatrix} 0.5182 & -0.0156 & -0.3594 & -0.1432 \\ -0.0156 & 1.1771 & -0.8021 & -0.3594 \\ -0.3594 & -0.8021 & 1.1771 & -0.0156 \\ -0.1432 & -0.3594 & -0.0156 & 0.5182 \end{bmatrix}$$

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

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

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

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

6. We get $\phi_1$ and $\phi_4$ 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,

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

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

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

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

```
% ==========================================================================
%% ME 441: FEM, HW 3, Problem 4
%% Afnan Mostafa
%% 12/05/2023
% ==========================================================================

%% %%%%%%%%%%%%%%%%%%%% clearing space %%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
close all
rng('shuffle')

%% %%%%%%%%%%%%%%%%%%%% material properties %%%%%%%%%%%%%%%%%%%

E_0 = 1e6;
nu = 0.5;
t = 0.001;
u_0 = 1;

%% %%%%%%%%%%%%%%%%%%%% symbolic math %%%%%%%%%%%%%%%%%%%%%%%%%%%

syms x y x1 y1 x2 y2 x3 y3 x4 y4 u_0 r2 r3 phi_x phi_y

%% %%%%%%%%%%%%%%%%%%%% coordinate: center %%%%%%%%%%%%%%%%

x1 = -0.5; y1 = -0.5;
x2 = 0.5; y2 = -1/4;
x3 = 0.5; y3 = 1/4;
x4 = -0.5; y4 = 0.5;

%% %%%%%%%%%%%%%%%%%%%% Shape functions %%%%%%%%%%%%%%%%

X = [1 x y x*y];
A = [
    1 x1 y1 x1*y1;
    1 x2 y2 x2*y2;
    1 x3 y3 x3*y3;
    1 x4 y4 x4*y4];

N = X*(inv(A));

%% %%%%%%%%%%%%%%%%%%%% Integrand %%%%%%%%%%%%%%%%

N_dx = diff(N,x);
```

```
N_dx_T = transpose(N_dx);

N_dy = diff(N,y);
N_dy_T = transpose(N_dy);

integrand = (N_dx_T*N_dx) + (N_dy_T*N_dy);

%% %%%%%%%%%%% Integral = Stiffness Matrix %%%%%%%%%%%

integral_1 = int(integrand,y,[(-3/8)+(x/4), (3/8)-(x/4)]);
K = double(int(integral_1,x,[-0.5 0.5]));

K_BC = K;

%% %%%%%%%%%%% Apply Boundary Conditions %%%%%%%%%%%

K_BC(:,2) = [];
K_BC(:,2) = [];
K_BC(2,:) = [];
K_BC(2,:) = [];

%% %%%%%%%%%%% Phi matrix from phi = K*f %%%%%%%%%%%

phiMat_BC = K_BC\[u_0/2; u_0/2];

phi_all = [phiMat_BC(1); 0; 0; phiMat_BC(2)];

%% %%%%%%%%%%% Force matrix %%%%%%%%%%%

force = [u_0/2; r2; r3; u_0/2];
force_mat = K*phi_all;

% integration of N w.r.t. dy gives area of the N-y plot
Area = (1*1/2)/2;

%% l = 1, m = 0 at right vertical boundary

u_2 = force_mat(2)/(Area); %% ANSWER
u_3 = force_mat(3)/(Area); %% ANSWER
fprintf('Velocity at node 2 (bottom right) is: %s \n', char(u_2))
fprintf('Velocity at node 2 is: %s units \n', char(subs(u_2,[u_0],[1])))
fprintf('Velocity at node 3 (top right) is: %s \n', char(u_3))
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

```
function [J,invJ,betaMat] = JacobianMatQ4(xyMat)
%% written by Afnan Mostafa as part of ME 441 at UR
%JacobianMat evaluates the Jacobian matrix for Q4 in isoparametric space
%
%   takes the [x y] matrix (4x2) and multiplies it with a prefactor and
%   another matrix (2x4) that consists of the derivatives of shape
%   functions w.r.t. eta and n. Also, it calculates the inverse of Jacobian
%   and then assembles the Beta matrix needed for stiffness matrix of Q4
%   elements in isoparametric space.
%
% input: [x y] matrix
% outputs: Jacobian matrix, inverse Jacobian matrix, and Beta Matrix

%% sanity check for symbolic e,n
% if sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) == 2
%
% elseif sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) < 2
    syms n e
% end

%% main body function
J = (1/4)*[-(1-n) (1-n) (1+n) -(1+n);
    -(1-e) -(1+e) (1+e) (1-e)]*xyMat;

invJ = inv(J);

Zer = zeros(size(invJ));
betaMat = [invJ Zer; Zer invJ];

end
```

# Appendix B

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

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

```matlab

%% hard-coding plane conditions (commented-out)

%isPlaneStrain = 0;
%isPlaneStress = 1;

%% %%%%%%%%%%%%%%%%%% sanity check for plane conds %%%%%%%%%%%%%%%%%%%%%%%

if sum([isPlaneStrain, isPlaneStress]) == 2
    error("Can't use both plane strain and plane stress, use any one")
elseif sum([isPlaneStrain, isPlaneStress]) == 1
    % do nothing
else
   warning('Choosing plane stress condition by default')
end

%% %%%%%%%%%%%%%%%%%% sanity check for for symbolic e,n %%%%%%%%%%%%%%%%%%

% if sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) == 2
%     % do nothing
% elseif sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) < 2
    syms n e
% end

%% %%%%%%%%%%%%%%%%%%%% call JacobianMatQ4 function %%%%%%%%%%%%%%%%%%%%%%

[J,~,betaMat] = JacobianMatQ4(xyMat);

%% %%%%%%%%%%%%%%%%%%%% Alpha Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

alphaMat = [1 0 0 0; 0 0 0 1; 0 1 1 0];

%% %%%%%%%%%%%%%%%%%%%% Beta Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%betaMat = [[invJ] [Zer]; [Zer] [invJ]];  %% no need to redefine

%% %%%%%%%%%%%%%%%%%%%% Gamma Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

gammaMat = (1/4)*[-1+n 0 1-n 0 1+n 0 -1-n 0;
    -1+e 0 -1-e 0 1+e 0 1-e 0;
    0 -1+n 0 1-n 0 1+n 0 -1-n;
    0 -1+e 0 -1-e 0 1+e 0 1-e];

%% %%%%%%%%%%%%%%%%%%%% Strain-Displacement Matrix %%%%%%%%%%%%%%%%%%%%%%%

B = alphaMat*betaMat*gammaMat;
B_t = transpose(B);

%% %%%%%%%%%%%%%%%%%%%% Constitutive Matrix: plane stress %%%%%%%%%%%%%%%%

if isPlaneStress == 1 && isPlaneStrain == 0
    E = (E_0/(1-(nu)^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];
elseif isPlaneStrain == 1 && isPlaneStress == 0
    E = (E_0/((1+nu)*(1-2*nu)))*[1-nu nu 0; nu 1-nu 0; 0 0 0.5-nu];
else
    disp('Choosing plane stress condition by default')
    E = (E_0/(1-(nu)^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];
end

%% %%%%%%%%%%%%%%%%%%%% Stiffness Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

integrand = eval(B_t*E*B*t*det(J));

end
```

# Appendix C

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

```
function [stiffMat] = GaussQuadQ4(order,integrand)
%% written by Afnan Mostafa as part of ME 441 at UR
%GaussQuadQ4 evaluates the gauss integral for Q4 in isoparametric space to
%obtain the stiffness matrix of a Q4 element.
%
%   takes the GQ order (either 1 or 2 or 3) and integrand obtained from the
%   IntegrandStiffMatQ4.m file (function file) and then performs the GQ
%   integration to obtain stiffness matrix (integral)
%
% input: GQ order and integrand matrix in terms of e,n
% outputs: Stiffness Matrix

%% %%%%%%%%%%%%%%%%%% sanity check for for symbolic e,n %%%%%%%%%%%%%%%%%%%%

% if sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) == 2
%
% elseif sum([strcmp(class(n),'sym'), strcmp(class(e),'sym')]) < 2
    syms n e
% end

%% %%%%%%%%%%%% gauss points and weights for 2d integration %%%%%%%%%%%%%%%%%

switch (order)
    case 1
        gaussPt = 0; wt = 4;
    case 2
        gaussPt = [
            -1/sqrt(3) -1/sqrt(3);
            1/sqrt(3) -1/sqrt(3);
            1/sqrt(3) 1/sqrt(3);
            -1/sqrt(3) 1/sqrt(3)];
        wt = [
            1 1;
            1 1;
            1 1;
            1 1];
    case 3
        gaussPt = [
            0 0;
            0 -sqrt(3/5);
            0 sqrt(3/5)
            sqrt(3/5) 0;
            -sqrt(3/5) 0;
            sqrt(3/5) sqrt(3/5);
            sqrt(3/5) -sqrt(3/5);
            -sqrt(3/5) sqrt(3/5);
            -sqrt(3/5) -sqrt(3/5)];
        wt = [
            8/9 8/9;
            8/9 5/9;
            8/9 5/9;
            5/9 8/9;
            5/9 8/9;
            5/9 5/9;
            5/9 5/9;
            5/9 5/9;
            5/9 5/9];
        % case 4
    otherwise
        error("Can't do more than 3rd order GQ")
end

%% %%%%%%%%%%%%%%%%%%%%%% apply gauss quadrature %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
gq = [];
integral = [];
for i=1:length(integrand)
    for j=1:length(integrand)
        func = integrand(i,j);
        for k=1:length(gaussPt)
            e = gaussPt(k,1);
            n = gaussPt(k,2);
            gq(k) = eval(subs(func))*wt(k,1)*wt(k,2);
        end
        integral(i,j) = sum([gq]);
    end
end
stiffMat = integral;
end
```

# Appendix D

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

```
function [mat3] = globalizeStiffMat(mat,posNodes,eleNodes,totNodes,DoF)
%globalizeStiffMat = globalizes element matrix if given nodes (CCW)
% mat = matrix for globalization, posNodes = nodal positions (CCW) from
% bottom left, eleNodes = how many nodes in an element, totNodes = total
% nodes in the entire system, DoF = 2 for u, v (per node)
%
%
%%
mat_cp = mat;
diffDOF = totNodes*DoF - length(mat);
mat_cp(end+diffDOF,:)= 0;
mat_cp(:,end+diffDOF)= 0;
mat2=zeros(size(mat_cp));

%% %%%%%%%%%%%%%%%%%%%% rearrange columns %%%%%%%%%%%%%%%%%%
posDisp = [(posNodes.*2)-1; posNodes.*2];
p=0;
for m=1:eleNodes
    for n=1:DoF
        mat2(:,posDisp(n,m)) = mat_cp(:,n+(2*p));
    end
    p=p+1;
end

%% %%%%%%%%%%%%%%%%%%%% rearrange rows %%%%%%%%%%%%%%%%%%%%%%%
mat3=zeros(size(mat2));
p=0;
for m=1:eleNodes
    for n=1:DoF
        mat3(posDisp(n,m),:) = mat2(n+(2*p),:);
    end
    p=p+1;
end
end
```