National Taiwan University **Finite Element Method**

Department of Civil Engineering Instructors: C.-S. CHEN

Lab Assignment 9, 05/30/2019, 1800 -- 2000

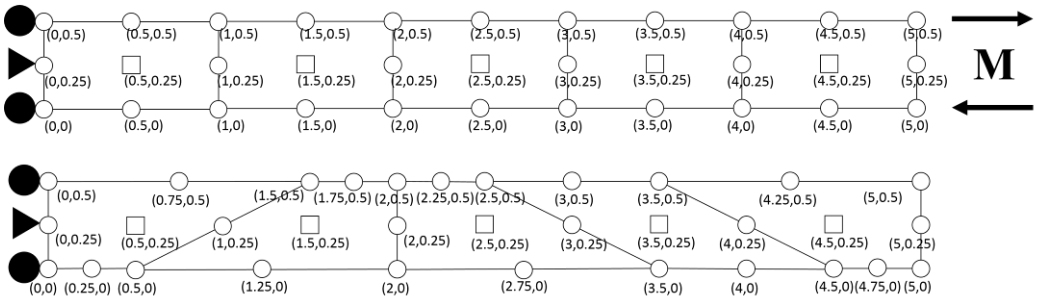
Due 2000

Lab Grading Policy: Attendance 20%, Score 80%, Bonus 40%

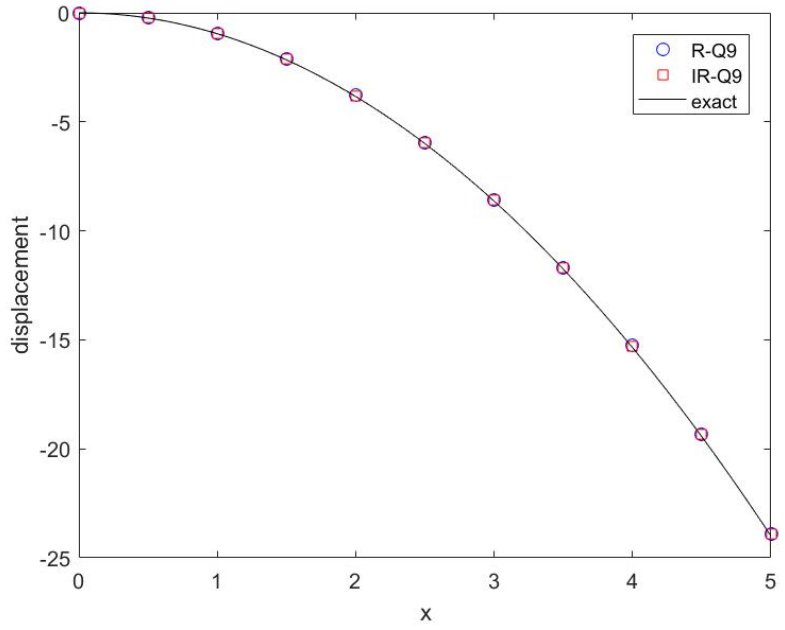
In case you have difficulty in finishing the exercises on time, you should upload them before **2100 on Saturday** and a penalty of 20% discount will be applied on your score. No late submission after 2100 on Saturday is permitted. We will in general post the reference solutions **by Sunday**.

Download `Q9Simple.zip` from the course website and unzip it. You will find a folder containing `Q9Simple.m` script file with six functions, `formStiffnessRec.m` `guass2d.m` `outputDisplacements.m` `shapeFunctionQ9.m` `solution.m`.

1. **(40%)** Consider a pure bending problem with two meshes shown below and model the problem with Q9 elements using 3x3 Gauss integration.



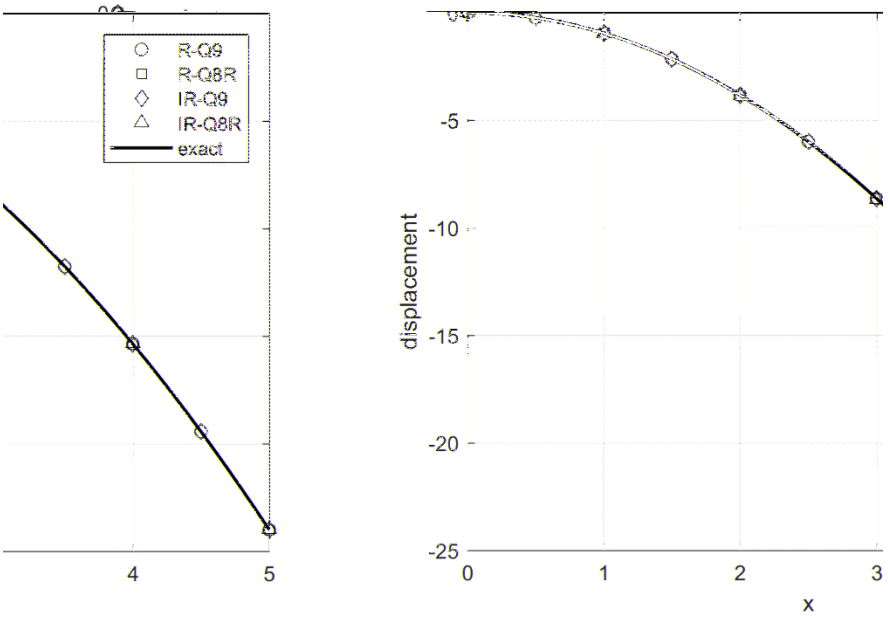
i f!qspqfsuf t! bsf!ubl fo!bt! ! !41-111-! ! !1/4- thickness = 1, and M = 600 which gives a vertical uq!e tqnbdfn fou!pg 24 units. Report the displacement along the centerline and compare your Q9 results with the exact solution from the beam theory $\frac{Mx^2}{2EI}$. Below is a sample plot:




```
% clear memory
clear all; close all; clc;

for ii =1:2
%% initialize
% materials
E  = 30000;     poisson = 0.30;  thickness = 1;

% matrix D
D=E/(1-poisson^2)*[1 poisson 0;poisson 1 0;0 0 (1-poisson)/2];

% mesh
numberNodes = 33;
numberElements = 5;

if ii == 1
    %-------------------------------------------------------------------------
    nodeCoordinates =
[(0:0.5:5)',zeros(11,1);(0:0.5:5)',0.25*ones(11,1);(0:0.5:5)',0.5*ones(11,1)];
    %-------------------------------------------------------------------------
elseif ii == 2
    %-------------------------------------------------------------------------
    nodeCoordinates = [[0,0.25,0.5,1.25,2,2.75,3.5,4,4.5,4.75,5]',zeros(11,1);
                       (0:0.5:5)',0.25*ones(11,1);
                       [0,0.75,1.5,1.75,2,2.25,2.5,3,3.5,4.25,5]',0.5*ones(11,1)];
    %-------------------------------------------------------------------------
end

%-------------------------------------------------------------------------
a=[1 3 25  23 2 14 24 12 13];
elementNodes = [a;a+2;a+4;a+6;a+8];
%-------------------------------------------------------------------------

% GDof: global number of degrees of freedom
GDof=2*numberNodes;

% boundary conditions-----------------------------------------------------
prescribedDof = [1;23;24;45];
```

```
%-------------------------------------------------------------------

% force vector ------------------------------------------------------
force=zeros(GDof,1);
force(21) = -1200;
force(end-1) = 1200;
%-------------------------------------------------------------------

%% Q9
% calculation of the system stiffness matrix
stiffness=formStiffness2D(GDof,numberElements,...
    elementNodes,numberNodes,nodeCoordinates,D,thickness);

% solution
displacements=solution(GDof,prescribedDof,stiffness,force);

%-------------------------------------------------------------------
linecolor = {'bo','rs'};
x = 0:0.5:5;
plot(x,displacements(24:2:44),linecolor{ii}); hold on
%-------------------------------------------------------------------
end

%-Exact-------------------------------------------------------------
I = 1/12*thickness*0.5^3;
u_exact = -600*x.^2/2/E/I;
plot(x,u_exact,'k-')
legend('R-Q9','IR-Q9','exact')
ylabel('displacement');xlabel('x')
%-------------------------------------------------------------------
```

2. **(40%)** Extend your implementation in Problem 1 to include the eight-node quadrilateral element with 2x2 reduced integration and consider the same pure bending problem again. Report the displacement along the centerline and compare your Q8 and Q9 (from Problem 1) results with the exact solution from the beam theory $\frac{Mx^2}{2EI}$. Below is a sample plot:



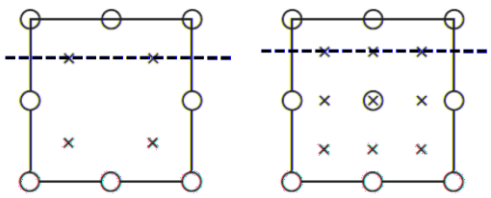

```
% clear memory
clear all; close all; clc;

for ii =1:4 % R-Q9 , R-Q8 , IR-Q9 , IR-Q8
%% initialize
% materials
E  = 30000;     poisson = 0.30;  thickness = 1;

% matrix D
D=E/(1-poisson^2)*[1 poisson 0;poisson 1 0;0 0 (1-poisson)/2];

% mesh
numberElements = 5;

if ii == 1  % R-Q9
    %------------------------------------------------------------------------
    numberNodes = 33;
    nodeCoordinates =
[(0:0.5:5)',zeros(11,1);(0:0.5:5)',0.25*ones(11,1);(0:0.5:5)',0.5*ones(11,1)];

    a=[1 3 25  23 2 14 24 12 13];
    elementNodes = [a;a+2;a+4;a+6;a+8];

    prescribedDof = [1;23;24;45];
    %------------------------------------------------------------------------
elseif ii == 2  % R-Q8
    %------------------------------------------------------------------------
    numberNodes = 28;
    nodeCoordinates =
[(0:0.5:5)',zeros(11,1);(0:1:5)',0.25*ones(6,1);(0:0.5:5)',0.5*ones(11,1)];

    a=[1 3 20  18 2 13 19 12];
    elementNodes = [a;a+2;a+4;a+6;a+8];
    elementNodes(:,[6,8]) = [(13:17)',(12:16)'];

    prescribedDof = [1;23;24;35];
    %------------------------------------------------------------------------
elseif ii == 3  % IR-Q9
    %------------------------------------------------------------------------
```

```
    numberNodes = 33;
    nodeCoordinates = [[0,0.25,0.5,1.25,2,2.75,3.5,4,4.5,4.75,5]',zeros(11,1);
                       (0:0.5:5)',0.25*ones(11,1);
                       [0,0.75,1.5,1.75,2,2.25,2.5,3,3.5,4.25,5]',0.5*ones(11,1)];

    a=[1 3 25  23 2 14 24 12 13];
    elementNodes = [a;a+2;a+4;a+6;a+8];

    prescribedDof = [1;23;24;45];
    %-----------------------------------------------------------------------
elseif ii == 4 % IR-Q8
    %-----------------------------------------------------------------------
    numberNodes = 28;
    nodeCoordinates = [[0,0.25,0.5,1.25,2,2.75,3.5,4,4.5,4.75,5]',zeros(11,1);
                       (0:1:5)',0.25*ones(6,1);
                       [0,0.75,1.5,1.75,2,2.25,2.5,3,3.5,4.25,5]',0.5*ones(11,1)];
    a=[1 3 20  18 2 13 19 12];
    elementNodes = [a;a+2;a+4;a+6;a+8];
    elementNodes(:,[6,8]) = [(13:17)',(12:16)'];

    prescribedDof = [1;23;24;35];
    %-----------------------------------------------------------------------
end
% GDof: global number of degrees of freedom
GDof=2*numberNodes;

% force vector ---------------------------------------------------------
force=zeros(GDof,1);
force(21) = -1200;
force(end-1) = 1200;
%-----------------------------------------------------------------------

%% Q9
if ii ==1 || ii==3
    % calculation of the system stiffness matrix
    stiffness=formStiffness2D(GDof,numberElements,...
        elementNodes,numberNodes,nodeCoordinates,D,thickness);
elseif ii ==2 || ii==4
    % calculation of the system stiffness matrix
    stiffness=formStiffness2DQ8(GDof,numberElements,...
        elementNodes,numberNodes,nodeCoordinates,D,thickness);
end
% solution
displacements=solution(GDof,prescribedDof,stiffness,force);

%-----------------------------------------------------------------------
linecolor = {'ko','ks','kd','k^'};
if ii == 1 || ii ==3;x = 0:0.5:5;
else ; x = 0:1:5; end
plot(x,displacements(24:2:(numberNodes-11)*2),linecolor{ii}); hold on
%-----------------------------------------------------------------------
end

%-Exact---------------------------------------------------------------
x = 0:0.1:5;
I = 1/12*thickness*0.5^3;
u_exact = -600*x.^2/2/E/I;
plot(x,u_exact,'k-','linewidth',1.5)
legend('R-Q9','R-Q8R','IR-Q9','IR-Q8R','exact')
ylabel('displacement');xlabel('x');grid on
%-----------------------------------------------------------------------
```

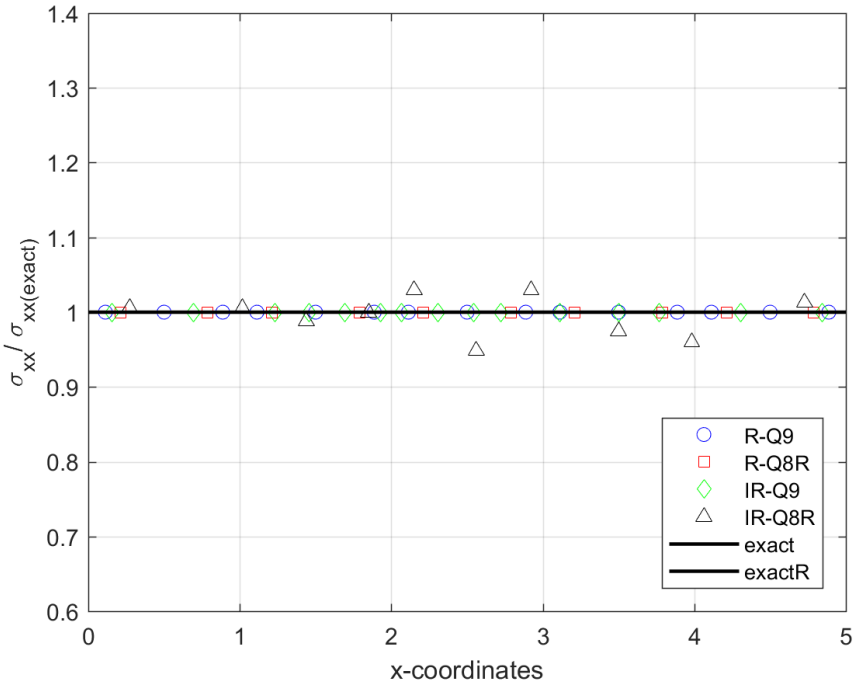
```
% .....................................................................
    function [shape,naturalDerivatives]=shapeFunctionQ8(xi,eta)

    % shape function and derivatives for Q8 elements
    % shape : Shape functions
    % naturalDerivatives: derivatives w.r.t. xi and eta
    % xi, eta: natural coordinates (-1 ... +1)

    shape=[ 1/4*(1-xi)*(1-eta)*(-xi-eta-1),...
           1/4*(1+xi)*(1-eta)*(xi-eta-1),...
           1/4*(1+xi)*(1+eta)*(xi+eta-1),...
           1/4*(1-xi)*(1+eta)*(-xi+eta-1),...
           1/2*(1-eta)*(1-xi^2),...
           1/2*(1+xi)*(1-eta^2),...
           1/2*(1+eta)*(1-xi^2),...
           1/2*(1-xi)*(1-eta^2)];


    naturalDerivatives=...
        [- (xi/4 - 1/4)*(eta - 1) - ((eta - 1)*(eta + xi + 1))/4,...
          ((eta - 1)*(eta - xi + 1))/4 - (xi/4 + 1/4)*(eta - 1),...
          (xi/4 + 1/4)*(eta + 1) + ((eta + 1)*(eta + xi - 1))/4,...
          (xi/4 - 1/4)*(eta + 1) + ((eta + 1)*(xi - eta + 1))/4,...
                                       2*xi*(eta/2 - 1/2),...
                                            1/2 - eta^2/2,...
                                      -2*xi*(eta/2 + 1/2),...
                                            eta^2/2 - 1/2;...
         - (xi/4 - 1/4)*(eta - 1) - (xi/4 - 1/4)*(eta + xi + 1),...
           (xi/4 + 1/4)*(eta - xi + 1) + (xi/4 + 1/4)*(eta - 1),...
           (xi/4 + 1/4)*(eta + 1) + (xi/4 + 1/4)*(eta + xi - 1),...
           (xi/4 - 1/4)*(xi - eta + 1) - (xi/4 - 1/4)*(eta + 1),...
                                            xi^2/2 - 1/2,...
                                      -2*eta*(xi/2 + 1/2),...
                                            1/2 - xi^2/2,...
                                       2*eta*(xi/2 - 1/2)];


    end % end function shapeFunctionQ8



%.....................................................................

function stiffness=formStiffness2DQ8(GDof,numberElements,...
    elementNodes,numberNodes,nodeCoordinates,D,thickness)

% compute stiffness matrix
% for plane stress Q8 elements

stiffness=zeros(GDof);

% 3 by 3 quadrature
[gaussWeights,gaussLocations]=gauss2d('2x2');

for e=1:numberElements
  numNodePerElement = length(elementNodes(e,:));
  numEDOF = 2*numNodePerElement;
  elementDof=zeros(1,numEDOF);
  for i = 1:numNodePerElement
     elementDof(2*i-1)=2*elementNodes(e,i)-1;
```

```
      elementDof(2*i)=2*elementNodes(e,i);
  end

  % cycle for Gauss point
  for q=1:size(gaussWeights,1)
    GaussPoint=gaussLocations(q,:);
    xi=GaussPoint(1);
    eta=GaussPoint(2);

% shape functions and derivatives
    [shapeFunction,naturalDerivatives]=shapeFunctionQ8(xi,eta);

% Jacobian matrix, inverse of Jacobian,
% derivatives w.r.t. x,y
    [Jacob,invJacobian,XYderivatives]=...
       Jacobian(nodeCoordinates(elementNodes(e,:),:),naturalDerivatives);

%  B matrix
    B=zeros(3,numEDOF);
    B(1,1:2:numEDOF)      = XYderivatives(1,:);
    B(2,2:2:numEDOF)  = XYderivatives(2,:);
    B(3,1:2:numEDOF)      = XYderivatives(2,:);
    B(3,2:2:numEDOF)  = XYderivatives(1,:);

% stiffness matrix
    stiffness(elementDof,elementDof)=...
       stiffness(elementDof,elementDof)+...
       B'*D*thickness*B*gaussWeights(q)*det(Jacob);
  end
end
```

3. **(Bonus 40%)** Extend your implementation to compute the stresses at Gauss points and report the axial stress σ_{xx} along the upper line of quadrature points (靠近上邊界的高斯點) for the same pure bending problem.



Compare your Q9 full integration and Q8 reduced integration results with the exact solutions from the elementary beam theory. Notice that the exact solutions differ as you change the Gauss point locations (that is, you will have different exact solutions for 3x3 and 2x2 upper line of quadrature points). Below is the sample plot:




```
% clear memory
clear all; close all; clc;

for ii =1:4 % R-Q9 , R-Q8 , IR-Q9 , IR-Q8
    %% initialize
    % materials
    E  = 30000;     poisson = 0.30;  thickness = 1;

    % matrix D
    D=E/(1-poisson^2)*[1 poisson 0;poisson 1 0;0 0 (1-poisson)/2];

    % mesh
    numberElements = 5;

    if ii == 1  % R-Q9
        %-------------------------------------------------------------------------
        numberNodes = 33;
        nodeCoordinates =
[(0:0.5:5)',zeros(11,1);(0:0.5:5)',0.25*ones(11,1);(0:0.5:5)',0.5*ones(11,1)];

        a=[1 3 25  23 2 14 24 12 13];
        elementNodes = [a;a+2;a+4;a+6;a+8];

        prescribedDof = [1;23;24;45];
        %-------------------------------------------------------------------------
    elseif ii == 2  % R-Q8
        %-------------------------------------------------------------------------
        numberNodes = 28;
        nodeCoordinates =
[(0:0.5:5)',zeros(11,1);(0:1:5)',0.25*ones(6,1);(0:0.5:5)',0.5*ones(11,1)];

        a=[1 3 20  18 2 13 19 12];
        elementNodes = [a;a+2;a+4;a+6;a+8];
        elementNodes(:,[6,8]) = [(13:17)',(12:16)'];
```

```
        prescribedDof = [1;23;24;35];
        %-------------------------------------------------------------------
    elseif ii == 3  % IR-Q9
        %-------------------------------------------------------------------
        numberNodes = 33;
        nodeCoordinates =
[[0,0.25,0.5,1.25,2,2.75,3.5,4,4.5,4.75,5]',zeros(11,1);
                        (0:0.5:5)',0.25*ones(11,1);

[0,0.75,1.5,1.75,2,2.25,2.5,3,3.5,4.25,5]',0.5*ones(11,1)];

        a=[1 3 25  23 2 14 24 12 13];
        elementNodes = [a;a+2;a+4;a+6;a+8];

        prescribedDof = [1;23;24;45];
        %-------------------------------------------------------------------
    elseif ii == 4 % IR-Q8
        %-------------------------------------------------------------------
        numberNodes = 28;
        nodeCoordinates =
[[0,0.25,0.5,1.25,2,2.75,3.5,4,4.5,4.75,5]',zeros(11,1);
                        (0:1:5)',0.25*ones(6,1);

[0,0.75,1.5,1.75,2,2.25,2.5,3,3.5,4.25,5]',0.5*ones(11,1)];
        a=[1 3 20  18 2 13 19 12];
        elementNodes = [a;a+2;a+4;a+6;a+8];
        elementNodes(:,[6,8]) = [(13:17)',(12:16)'];

        prescribedDof = [1;23;24;35];
        %-------------------------------------------------------------------
    end
    % GDof: global number of degrees of freedom
    GDof=2*numberNodes;

    % force vector ------------------------------------------------------
    force=zeros(GDof,1);
    force(21) = -1200;
    force(end-1) = 1200;
    %--------------------------------------------------------------------

    %% Q8 & Q9
    if ii ==1 || ii==3
        % calculation of the system stiffness matrix
        stiffness=formStiffness2D(GDof,numberElements,...
            elementNodes,numberNodes,nodeCoordinates,D,thickness);
    elseif ii ==2 || ii==4
        % calculation of the system stiffness matrix
        stiffness=formStiffness2DQ8(GDof,numberElements,...
            elementNodes,numberNodes,nodeCoordinates,D,thickness);
    end
    % solution
    displacements=solution(GDof,prescribedDof,stiffness,force);

    % stress ------------------------------------------------------------
    if ii == 1 || ii ==3
    [stress_upper_gp,stress_coor]=formstress2D(GDof,numberElements,...
        elementNodes,numberNodes,nodeCoordinates,D,thickness,displacements);
    elseif ii == 2 || ii ==4
    [stress_upper_gp,stress_coor]=formstress2DQ8(GDof,numberElements,...
        elementNodes,numberNodes,nodeCoordinates,D,thickness,displacements);
    end
```

```
    %-Exact stress ------------------------------------------------------------
    I = 1/12*thickness*0.5^3;
    stress_exact = 600*(sqrt(3/5)*0.25)/I;
    stress_exactR = 600*(sqrt(1/3)*0.25)/I;
    %--------------------------------------------------------------------

    % stress -------------------------------------------------------------------
    linecolors = {'bo','rs','gd','k^'};
    if ii == 1 || ii ==3 % Q9 full int
        stress_coor = vec2mat(stress_coor,15);
        stress_upper_gp = vec2mat(stress_upper_gp,15);
        plot(stress_coor,abs(stress_upper_gp/stress_exact),linecolors{ii}); hold
on
    elseif ii == 2 || ii ==4 % reduced int
        stress_coor = vec2mat(stress_coor,10);
        stress_upper_gp = vec2mat(stress_upper_gp,10);
        plot(stress_coor,abs(stress_upper_gp/stress_exactR),linecolors{ii}); hold
on
    end
    %--------------------------------------------------------------------
end

% Exact stress ------------------------------------------------------------
plot([0,5],[stress_exact/stress_exact,stress_exact/stress_exact],'k-
','linewidth',1.5)
plot([0,5],[stress_exactR/stress_exactR,stress_exactR/stress_exactR],'k-
','linewidth',1.5)
legend('R-Q9','R-Q8R','IR-Q9','IR-Q8R','exact','exactR','location','southwest')
ylabel('\sigma_x_x/ \sigma_x_x_(_e_x_a_c_t_)');xlabel('x-coordinates');grid on
title('')
axis([0 5 0.9 1.1])
%--------------------------------------------------------------------


%.............................................................

function [stress_upper_gp,stress_coor]=formstress2D(GDof,numberElements,...
    elementNodes,numberNodes,nodeCoordinates,D,thickness,displacements)

% compute stiffness matrix
% for plane stress Q9 elements-----------------------------------------
pp = [4,7,3];
stress_upper_gp = zeros(numberElements,length(pp));
stress_coor = zeros(numberElements,length(pp));
%--------------------------------------------------------------------------

% 3 by 3 quadrature
[gaussWeights,gaussLocations]=gauss2d('3x3');

for e=1:numberElements
  numNodePerElement = length(elementNodes(e,:));
  numEDOF = 2*numNodePerElement;
  elementDof=zeros(1,numEDOF);
  for i = 1:numNodePerElement
      elementDof(2*i-1)=2*elementNodes(e,i)-1;
      elementDof(2*i)=2*elementNodes(e,i);
  end

  % cycle for Gauss point ------------------------------------------------
  stress_1gp = zeros(length(pp),3);
```

```
  stress_1gp_coor = zeros(1,length(pp));
  k=0;
  for q=pp
    GaussPoint=gaussLocations(q,:);
    xi=GaussPoint(1);
    eta=GaussPoint(2);

% shape functions and derivatives
    [shapeFunction,naturalDerivatives]=shapeFunctionQ9(xi,eta);

% Jacobian matrix, inverse of Jacobian,
% derivatives w.r.t. x,y
    [Jacob,invJacobian,XYderivatives]=...
        Jacobian(nodeCoordinates(elementNodes(e,:),:),naturalDerivatives);

%  B matrix
    B=zeros(3,numEDOF);
    B(1,1:2:numEDOF)      = XYderivatives(1,:);
    B(2,2:2:numEDOF)  = XYderivatives(2,:);
    B(3,1:2:numEDOF)      = XYderivatives(2,:);
    B(3,2:2:numEDOF)  = XYderivatives(1,:);

% stress ---------------------------------------------------------------
    k=k+1;
    stress_1gp(k,:) = (D*B*displacements(elementDof))';
    stress_1gp_coor(k) = shapeFunction*nodeCoordinates(elementNodes(e,:),1); %
only x
  end
  stress_upper_gp(e,:) = stress_1gp(:,1)';
  stress_coor(e,:) = stress_1gp_coor;
end


%.....................................................................

function [stress_upper_gp,stress_coor]=formstress2DQ8(GDof,numberElements,...
    elementNodes,numberNodes,nodeCoordinates,D,thickness,displacements)

% compute stiffness matrix
% for plane stress Q8 elements-------------------------------------------
pp = [4,3];
stress_upper_gp = zeros(numberElements,length(pp));
stress_coor = zeros(numberElements,length(pp));
%-----------------------------------------------------------------------

% 3 by 3 quadrature
[gaussWeights,gaussLocations]=gauss2d('2x2');

for e=1:numberElements
  numNodePerElement = length(elementNodes(e,:));
  numEDOF = 2*numNodePerElement;
  elementDof=zeros(1,numEDOF);
  for i = 1:numNodePerElement
     elementDof(2*i-1)=2*elementNodes(e,i)-1;
     elementDof(2*i)=2*elementNodes(e,i);
  end

  % cycle for Gauss point ------------------------------------------------
  stress_1gp = zeros(length(pp),3);
  stress_1gp_coor = zeros(1,length(pp));
  k=0;
```

```
  for q=pp
    GaussPoint=gaussLocations(q,:);
    xi=GaussPoint(1);
    eta=GaussPoint(2);

% shape functions and derivatives
    [shapeFunction,naturalDerivatives]=shapeFunctionQ8(xi,eta);

% Jacobian matrix, inverse of Jacobian,
% derivatives w.r.t. x,y
    [Jacob,invJacobian,XYderivatives]=...
        Jacobian(nodeCoordinates(elementNodes(e,:),:),naturalDerivatives);

%  B matrix
    B=zeros(3,numEDOF);
    B(1,1:2:numEDOF)      = XYderivatives(1,:);
    B(2,2:2:numEDOF)  = XYderivatives(2,:);
    B(3,1:2:numEDOF)      = XYderivatives(2,:);
    B(3,2:2:numEDOF)  = XYderivatives(1,:);

% stress ---------------------------------------------------------------------
    k=k+1;
    stress_1gp(k,:) = (D*B*displacements(elementDof))';
    stress_1gp_coor(k) = shapeFunction*nodeCoordinates(elementNodes(e,:),1); %
only x
  end
  stress_upper_gp(e,:) = stress_1gp(:,1)';
  stress_coor(e,:) = stress_1gp_coor;
end
```