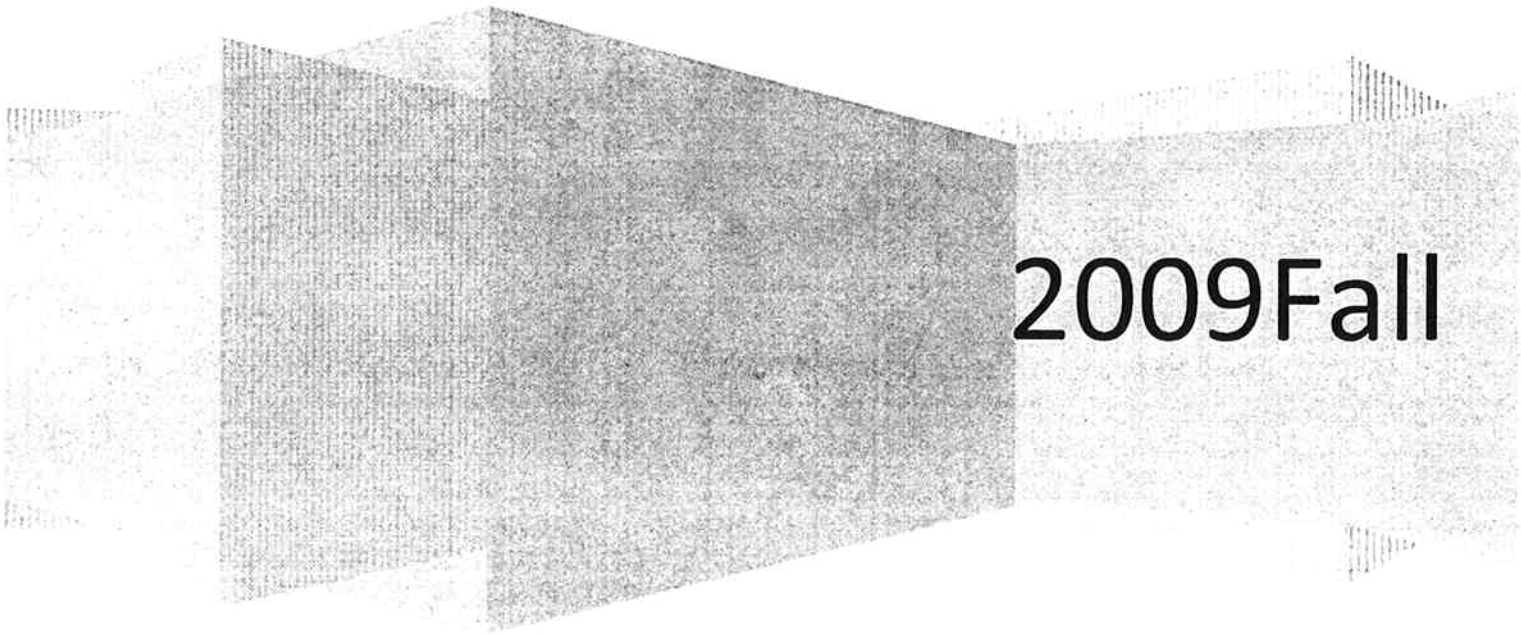
Uakron

Finite Element Analysis 1

Homework 10 Coding

Zhuopei Hu

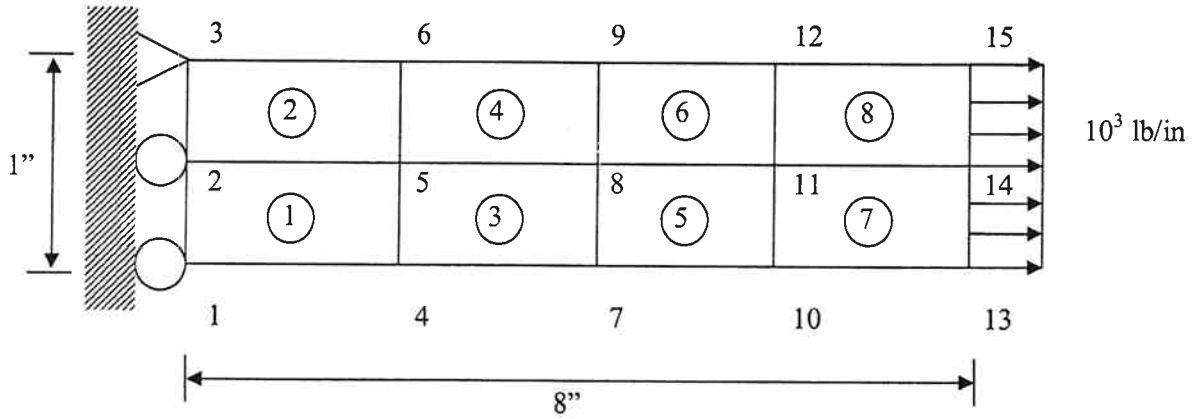


Finite Element Analysis I 4300:609

Homework # 10: Optional (for Bonus Points)

Due is December 11 (Tuesday)

The goal of this computational assignment is to give you a good idea about the complete finite element procedure to solve value problems. To this effect, you will solve the problem below using the educational MATLAB code provided to you. It consists of a simple computer program for two-dimensional stress analysis using the Q4 element. Note that the program is incomplete. You are supposed to complete this MATLAB code. The codes have been posted in springboard.



$E=10^6$ psi $\nu=0.2$

1. Main_FEProgram.m: Main FE program (Incomplete)
2. ScanInput.m: Read Finite Element Model Data
3. GetIDArray.m: Get ID Array
4. SetGaussQ.m: Set Gauss Quadrature Constants
5. GetElemStiff.m: Compute Elemental Stiffness Matrices
6. GetGPV.m: Get Sampling Point Locations and Associated Weight Factor
7. GetSFD.m: Compute Shape Function Derivatives
8. GetCoord.m: Get Coordinate Values of the Elements
9. GetDJacob.m: Get Jacobian Matrix and Determinant of Jacobian Matrix
10. GetBMatrix.m: Get [B] matrix
11. Get_TD_1_full.m: Solve system of linear equilibrium equation using Gauss Elimination Method (Triangular Decomposition)
12. GetDLoad.m: Compute Consistent Load Vector (Incomplete)
13. GetStrsStrn.m: Compute Stresses and Strains at Gauss Points and Print out the Results (Incomplete)

- Study the program provided to you and try to gain a clear understanding of the overall FEM procedure, and the logic behind each of the routines.

- Generate your FE input data file after studying the ScanInput.m program.
- Complete the following function files: Main_FEProgram.m, GetDLoad.m, and GetStrsStrn.m
- Run your Main_FEProgram.m program and print the stresses and the displacements for all the elements.
- You should submit your complete MATLAB code (i.e. all the M files and the output for deflections and stresses).
- Compare your solution with ABAQUS.

Reprot of Hw10

Displacment comparation between Abauqs and CODE

	CODE		ABAQUS	
node	u1	u2	u1	u2
1	0	0.0008	1.00E-33	0.0008
2	0	0.0004	2.00E-33	0.0004
3	0	0	1.00E-33	0
4	0.008	0.0008	0.008	0.0008
5	0.008	0.0004	0.008	0.0004
6	0.008	0	0.008	2.90E-16
7	0.016	0.0008	0.016	0.0008
8	0.016	0.0004	0.016	0.0004
9	0.016	-1E-15	0.016	1.05E-15
10	0.024	0.0008	0.024	0.0008
11	0.024	0.0004	0.024	0.0004
12	0.024	-1E-15	0.024	2.11E-15
13	0.032	0.0008	0.032	0.0008
14	0.032	0.0004	0.032	0.0004
15	0.032	-2E-15	0.032	3.29E-15

Result: The displacment are same. The values in shadow can be treated as 0.

Stress comparation between Abauqs and CODE

	Code			Abaqus		
element	s11	s22	s12	s11	S22	s12
1	4000	2.49E-13	-1.5E-11	4000	0	0
2	4000	9.553E-12	-1.5E-11	4000	0	0
3	4000	9.395E-12	-4E-12	4000	0	0
4	4000	2.029E-11	-1.7E-11	4000	0	0
5	4000	2.274E-11	-1.1E-11	4000	0	0
6	4000	-3.85E-11	-8.4E-12	4000	0	0
7	4000	-5.35E-11	1.83E-12	4000	0	0
8	4000	-3.91E-11	-7.8E-12	4000	0	0

Result: The s22 and S12 can be treated as zeros. From the code, we can get the stress directly.
For the Abaqus, the vaule is calculated. Stress for each node can be deal with output.
Then calcualte the average of the value which shows above.

```
%%  2D Plane Stress Element for HW#6

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

%--- V A R I A B L E   D E F I N I T I O N---------------------------------
%   'title': title of project     --2D Plane stress
%   'npoin': number of nodes      --15 nodes  npoin=15
%   'nelem': number of elements   --8 elements nelem=8
%   'nnode': number of nodes per element  --4 nodes  nnode=4
%   'nmate': number of materials        --nmate=1, 2.05E11
%   'nsect': number of sections         --nsect=1, section=0.01
%   'ndofn': number of DOFs per node    --ndofn=2
%   'nevab': size of elemental stiffness matrix  --nevab=8
%   'nvfix': number of restrained nodes    -- nvfix=4?
%   'lnods': [ node1 node2 node3 node4;
%              node1 node2 node3 node4;
%              ... ];  each row is for each element
%   Note size of 'lnods' is (number of nodes x number of nodes per element)
%   'IDArray': [ ndofn x npoin ] matrix that stores equation number
%   'iffix': vector of size [1 x npoin*ndofn],which store 0 for active DOF
%              and stores 1 for inactive DOF
%   'neq': Total number of equations, that is, total number of active DOFs
%--- V A R I A B L E   D E F I N I T I O N---------------------------------

clear all; clc;

global nelem nnode ndofn neq
global lnods
global IDArray

%----------------------------------------
%  Read FE data from input data file
%----------------------------------------

% open input file that contains FE data (nodal coord. & element
% connectivity & sectional properties & material properties)
ifid = fopen('FEdata.inp','rt');

% scan input file
ScanInput(ifid);

% Get IDArray
IDArray = GetIDArray();

%----------------------------------------
%  Compute Element Stiffness Matrix
%----------------------------------------

% compute size of elemental stiffness matrix
nevab = nnode*ndofn;
```

```
% initialize element stiffness matrix
estif = zeros(nevab,nevab);
xGK = zeros(neq,neq);

% determine gauss quadrature rule to use
ngaus = 2; % two point rule

% set Gauss Quadrature Constants
SetGaussQ(ngaus);

for ielem=1:nelem

    % clear element stiffness matrix
    estif = zeros(nevab,nevab);
    [redacted]


    % calculate linear membrane stiffness matrix
    [estif,[redacted] = GetElemStiff(Jmat,ielem,ngaus,estif);

    % when using IDArray
    for inode=1:nnode
        nod = lnods(ielem,inode);
        for jdofn=1:ndofn
            pos =(inode-1)*ndofn + jdofn;
            KK(pos) = IDArray(jdofn, nod);
        end
    end

    % Assemble element stiffness matrix
    for ievab=1:nevab
        if( KK(ievab) <= 0 )
            continue;
        end
        I=KK(ievab);
        for jevab=1:nevab
            J = KK(jevab);
            if( J < I )
                continue;
            end
            xGK(I,J) = xGK(I,J) + estif(ievab,jevab);
        end
    end

end


% make it symmetric matrix
for ieq=1:neq
    for jeq=ieq:neq
        xGK(jeq,ieq) = xGK(ieq,jeq);
```

```
    end
end
[Load] = GetDload(Jmat,ielem,ngaus)
[stress,Tdisp]=GetStrain(Jmat,Load,xGK,ielem,ngaus)

fclose('all');
```

```
function ScanInput(ifid)
%% Scan input file for Finite element analysis

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

%--- V A R I A B L E   D E F I N I T I O N---------------------------------
%   'title': title of project
%   'npoin': number of nodes
%   'nelem': number of elements
%   'nnode': number of nodes per element
%   'nmate': number of materials
%   'nsect': number of sections
%   'ndofn': number of DOFs per node
%   'nevab': size of elemental stiffness matrix
%   'nvfix': number of restrained nodes
%   'lnods': [ node1 node2 node3 node4;
%              node1 node2 node3 node4;
%              ... ];  each row is for each element
%   Note size of 'lnods' is (number of nodes x number of nodes per element)
%   'IDArray': [ ndofn x npoin ] matrix that stores equation number
%   'iffix': vector of size [1 x npoin*ndofn],which store 0 for active DOF
%            and stores 1 for inactive DOF
%--- V A R I A B L E   D E F I N I T I O N---------------------------------

global title
global npoin nelem nnode nmate nsect ndofn nvfix
global lnods iffix
global COORD
global ELEM_DATA
global SECTION_DATA
global MATERIAL_DATA
global BND_DATA


% Scan control cards
tline = fgetl(ifid);

while ~feof(ifid) % if not the end of file, continue scanning

    % comment line
    if ~isempty(findstr(tline,'**')) ~= 0
        tline = fgetl(ifid);
    elseif ~isempty(findstr(tline,'*tit'))~=0
        tline = fgetl(ifid);
        title = tline;
        tline = fgetl(ifid);
    elseif ~isempty(findstr(tline,'*contr'))~=0
        tline = fgetl(ifid);
        [token,rem] = strtok(tline, ', =');
        while ~isempty(token)   % to the end of this line
```

```
        if strcmp(token,'npoin')
            [token,rem] = strtok(rem,' ,=');
            npoin = str2double(token);
        elseif strcmp(token,'nelem')
            [token,rem] = strtok(rem,' ,=');
            nelem = str2double(token);
        elseif strcmp(token,'nsect')    % get the number of sections
            [token,rem] = strtok(rem,' ,=');
            nsect = str2double(token);
        elseif strcmp(token,'nmate')    % get the number of materials
            [token,rem] = strtok(rem,' ,=');
            nmate = str2double(token);
        elseif strcmp(token,'ndofn')    % get the number of DOFs per node
            [token,rem] = strtok(rem,' ,=');
            ndofn = str2double(token);
        elseif strcmp(token,'nvfix')    % get the number of restrained nodes
            [token,rem] = strtok(rem,' ,=');
            nvfix = str2double(token);
            % set iffix array size and contents(zero)
            iffix = zeros(1,npoin*ndofn);
        end
        [token,rem] = strtok(rem,' ,=');
    end
    tline = fgetl(ifid);
% read nodal coordinate values
elseif ~isempty(findstr(tline,'*node'))~=0
    COORD = fscanf(ifid,'%d %f %f %f\n',[4,npoin]);
    COORD = COORD';
    tline = fgetl(ifid);
% read elemental connectivity data
% #, node1, node2, node3, node4, section #, material #
elseif ~isempty(findstr(tline,'*elem'))~=0
    [token,rem] = strtok(tline, ', =');
    while ~strcmp(token,'nnode')
        [token,rem] = strtok(rem, ', =');
    end
    [token,rem] = strtok(rem, ', =');
    nnode = str2double(token);
    ELEM_DATA = fscanf(ifid,'%d\n',[7,nelem]);
    ELEM_DATA = ELEM_DATA';
    lnods = ELEM_DATA(:,2:(nnode+1));
    tline = fgetl(ifid);
% read material data
% #, Young's modulus, poisson ratio
elseif ~isempty(findstr(tline,'*material')) ~=0
    MATERIAL_DATA =fscanf(ifid,'%i,%f,%f\n',[3,nmate]);
    MATERIAL_DATA = MATERIAL_DATA';
    tline = fgetl(ifid);
% read sectional properties
% #, thickness
elseif ~isempty(findstr(tline,'*section')) ~=0
    SECTION_DATA = fscanf(ifid,'%i,%f\n',[2,nsect]);
```

```
        SECTION_DATA = SECTION_DATA';
        tline = fgetl(ifid);
    % Read boundary conditions
    elseif ~isempty(findstr(tline,'*bound'))~=0
        BND_DATA = fscanf(ifid,'%i,%i,%i\n',[3,nvfix]);
        BND_DATA = BND_DATA';
        nofix = BND_DATA(:,1);
        fix_data = BND_DATA(:,2:3);
        % get infomation on boundary condition
        for ivfix=1:nvfix
            for idofn=1:ndofn
                pos = (nofix(ivfix)-1)*ndofn + idofn;
                iffix(pos) = fix_data(ivfix,idofn);
            end
        end
        tline = fgetl(ifid);
    % Pass unidentified command
    else
        tline = fgetl(ifid);
    end
end

fclose(ifid);

return
```

```
function [IDArray] = GetIDArray()
%% Get ID Array
% Copyright (C) Gunjin Yun(gunjin73@hotmail.com)
% Get the matrix that stores equation number
% Save the Equation number

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

global npoin ndofn
global iffix nofix
global neq

% initialize of IDArray
for ipoin=1:npoin
    iposi = (ipoin-1)*ndofn;
    for idofn=1:ndofn
        if (iffix(iposi+idofn)==0)
            IDArray(idofn,ipoin) = 0;
        elseif (iffix(iposi+idofn)~=0)
            IDArray(idofn,ipoin) = 1;
        end
    end
end

% Generate a table of equation number
neq = 0;
for ipoin=1:npoin
    for idofn=1:ndofn
        % transfer if DOF is fixed. Otherwise, increment neq
        if (IDArray(idofn,ipoin)==0)
            neq = neq+1;
            IDArray(idofn,ipoin) = neq;
        else
            IDArray(idofn,ipoin) = 0;
        end
    end
end

return
```

```
function SetGaussQ(ngaus)
%% set Gauss quadrature constants

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

global posgp weigp

% initialize variables
posgp = zeros(ngaus*ngaus,2);
weigp = zeros(ngaus*ngaus,2);

if ngaus==2
    posgp(1,1) = -0.5773502691896260;   % sqrt(1/3)
    posgp(1,2) = -0.5773502691896260;
    posgp(2,1) = 0.5773502691896260;
    posgp(2,2) = -0.5773502691896260;
    posgp(3,1) = 0.5773502691896260;
    posgp(3,2) = 0.5773502691896260;
    posgp(4,1) = -0.5773502691896260;
    posgp(4,2) = 0.5773502691896260;

    weigp(1,1) = 1.0;
    weigp(1,2) = 1.0;
    weigp(2,1) = 1.0;
    weigp(2,2) = 1.0;
    weigp(3,1) = 1.0;
    weigp(3,2) = 1.0;
    weigp(4,1) = 1.0;
    weigp(4,2) = 1.0;
end

% you can try other gauss quadrature rules


return
```

```
 function [estif,Jmat,Bmat] = GetElemStiff(Jmat,ielem,ngaus,estif,Bmat)
%% compute elemental stiffness matrix

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

global nevab nnode
global ELEM_DATA
global SECTION_DATA
global MATERIAL_DATA

% set variables
xipos=0.0;
etapos=0.0;
xiweight=0.0;
etaweight=0.0;
detJ =0.0;
SFD = zeros(2,nnode);
coord2D = zeros(nnode,2);
Bmat = zeros(3,nevab);
Dmat = zeros(3,3);
Jmat = zeros(2,2);

% get material and section properties for the element
no_sect = ELEM_DATA(ielem,6);
no_mate = ELEM_DATA(ielem,7);

E = MATERIAL_DATA(no_mate,2);
Nu = MATERIAL_DATA(no_mate,3);
thick = SECTION_DATA(no_sect,2);

% get material constitutive matrix
Dmat =[  1, Nu,         0;
        Nu,  1,         0;
         0,  0, (1-Nu)/2 ] ;

Dmat=E*thick/(1-Nu^2)*Dmat;

% loop over integration points in the xi direction
for igaus=1:ngaus

    % loop over integration points in the eta direction
    for jgaus=1:ngaus

        % initialize arrays
        Bmat = zeros(3,nevab);
        SFD = zeros(2,nnode);
        coord2D = zeros(nnode,2);

        % get sampling point location and associated weight factor
        [xipos,etapos,xiweight,etaweight]= GetGPV(ngaus,igaus,jgaus,xipos,etapos,xiweight,↙
```

```
etaweight);
        % compute shape function derivatives
        [SFD] = GetSFD(SFD,xipos,etapos);
        % get coordinate values of the element
        [coord2D] = GetCoord(ielem,coord2D);
        % get Jacobian matrix and determinant of Jmat
        [detJ,Jmat] = GetDJabob(SFD,coord2D,detJ,Jmat);
        % get B natrix
        [Bmat] = GetBMatrix(Jmat,SFD,Bmat);

        dvolu = detJ*xiweight*etaweight;
        estif = estif + Bmat'*Dmat*Bmat*dvolu;

    end
end
```

```
function [xipos,etapos,xiweight,etaweight] = GetGPV(ngaus,igaus,jgaus,xipos,etapos,xiweight,etaweight)
%% get sampling point location and associated weight factor

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

global posgp weigp

%two points rule
if ngaus==2

    if igaus==1
        xipos = posgp(jgaus,1);
        etapos = posgp(jgaus,2);
        xiweight = weigp(jgaus,1);
        etaweight = weigp(jgaus,2);
    else
        xipos = posgp(igaus+jgaus,1);
        etapos = posgp(igaus+jgaus,2);
        xiweight = weigp(igaus+jgaus,1);
        etaweight = weigp(igaus+jgaus,2);
    end

end

% you can try other gauss quadrature rules
```

```
function [SFD] = GetSFD(SFD,xipos,etapos)
%% compute shape function derivatives

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

% xi-derivatives d_N/d_xi
SFD(1,1) = -0.25*(1-etapos);
SFD(1,2) = 0.25*(1-etapos);
SFD(1,3) = 0.25*(1+etapos);
SFD(1,4) = -0.25*(1+etapos);

% eta-derivative d_N/d_eta
SFD(2,1) = -0.25*(1-xipos);
SFD(2,2) = -0.25*(1+xipos);
SFD(2,3) = 0.25*(1+xipos);
SFD(2,4) = 0.25*(1-xipos);

return
```

```
function [coord2D] = GetCoord(ielem,coord2D)
%% get elemental coordinate matrix

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

global lnods nnode
global COORD

for inode=1:nnode
    node = lnods(ielem,inode);
    coord2D(inode,1) = COORD(node,2);
    coord2D(inode,2) = COORD(node,3);
end


return
```

```
function [detJ,Jmat] = GetDJabob(SFD,coord2D,detJ,Jmat)
%% Get Jacobian matrix and its determinant value

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

Jmat = SFD*coord2D;
detJ = det(Jmat);

return
```

```
function [Bmat] = GetBMatrix(Jmat,SFD,Bmat)
%% compute B martix

%  Coded by Prof. Gun Jin Yun (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

%   A = invJ*sfd;           [ N1,x N2,x N3,x N4,x ] -> 4-node element
%                           [ N1,y N2,y N3,y N4,y ]

%   B_m matrix      [N1,x 0      ..... N4,x 0    ]   -> 4node element
%                   [0    N1,y   ..... 0    N4,y ]
%                   [N1,y N1,x   ..... N4,y N4,x ]

% compute inverse of Jacobian matrix
invJ = inv(Jmat);

% compute invJ * SFD
A = invJ*SFD;

% construct B matrix
Bmat(1,1) = A(1,1);
Bmat(2,2) = A(2,1);
Bmat(3,1) = A(2,1);
Bmat(3,2) = A(1,1);

Bmat(1,3) = A(1,2);
Bmat(2,4) = A(2,2);
Bmat(3,3) = A(2,2);
Bmat(3,4) = A(1,2);

Bmat(1,5) = A(1,3);
Bmat(2,6) = A(2,3);
Bmat(3,5) = A(2,3);
Bmat(3,6) = A(1,3);

Bmat(1,7) = A(1,4);
Bmat(2,8) = A(2,4);
Bmat(3,7) = A(2,4);
Bmat(3,8) = A(1,4);

return
```

```
function [Load] = GetDload(Jmat,ielem,ngaus)
%% Get the load
global nnode
global SECTION_DATA
global ELEM_DATA
xipos=1
Eload = zeros(nnode,1)

q=1000;
no_sect = ELEM_DATA(ielem,6);
thick = SECTION_DATA(no_sect,2);
JJmat = zeros(2,1);

JJmat(1,1) = -Jmat(2,2)
JJmat(2,1) = Jmat(2,1)
Nf = zeros(8,2)

    % loop over integration points in the eta direction
    for jgaus=1:ngaus

        % initialize arrays
        SF = zeros(1,nnode);

        if jgaus==1
            etapos=-1/(sqrt(3))
            weta=1
        else
            etapos=1/(sqrt(3))
            weta=1
        end
        % compute shape function derivatives
        [SF] = GetSF(SF,xipos,etapos);
        % get Nmatrix
        % N1 part
          dov=thick*q
        Eload = Eload+SF'*q*weta;



    end
    Load = zeros(26,1);
    Load(23,1)=2*Eload(2,1);
    Load(25,1)=Eload(3,1);
    Load(21,1)=Eload(3,1);
return
```

```
function [SF] = GetSF(SF,xipos,etapos)
%% compute shape function derivatives

%  Coded by Zhuopei Hu (gy3@uakron.edu)
%  October 17 2009
%  4300:609 Finite Element Analysis I Fall 2009

    SF(1,1) = 0.25*(1-xipos)*(1-etapos);
    SF(1,2) = 0.25*(1+xipos)*(1-etapos);
    SF(1,3) = 0.25*(1+xipos)*(1+etapos);
    SF(1,4) = 0.25*(1-xipos)*(1+etapos);

return
```