**FINITE ELEMENT METHODS IN ENGINEERING**

**(PAPER CODE: ME60407)**

****

**GROUP MEMBERS:**

**ASHISH KUMAR SINGH(23ME63R04)**

**RAJDEEP CHATTERJEE(23ME63R26)**

**MD FAISAL ALAM(23ME63R30)**

**PROBLEM STATEMENT:**

500 N

16

3 (100GPa)

16

2 (50 GPa)

16

1 (25 GPa)

96

(All dimensions are in mm)

**From the given problem we are required to find force, displacement, stress and strain at each node.**

**For this purpose, MATLAB and Ansys softwares can be used.**

**Later we need to compare from both results.**

**METHODOLOGY:**

A cantilever beam as shown above consists of three parallel members of Young’s Modulus say 100 GPa, 50 GPa, 25 GPa respectively.

Each of them has identical dimensions given as: Length L= 96 mm

Width W= 16 mm

Thickness t= 1 mm

In this given problem plane stress condition is assumed (given µ= 0.3).

To solve the problem, we will first discretize the members into smaller elements of sizes as: i) 8\*8 mm2

ii) 4\*4 mm2

iii) 2\*2 mm2

Each node of each element will be assigned node number which will be unique to the particular node.

Each element Stiffness matrix will be assembled to global stiffness matrix.

**USING MATLAB:**

%% Physical Domain

n\_mem=3; % Number of Members

% Youngs Modulus of materials in the Members in parallel in GPa

mu=0.3; %Poisson ratio

E=zeros(n\_mem,1);

for i=1:n\_mem

prompt="Enter the value of Young's Modulus of the Member %d in GPa: ";

E(i)=input(sprintf(prompt,i));

E(i)=E(i)\*10^9;

end

% Width of the Members in parallel in mm

W=zeros(n\_mem,1);

for i=1:n\_mem

prompt="Enter the value of Width of the Member %d in mm : ";

W(i)=input(sprintf(prompt,i));

end

% Thickness of the Members in parallel in mm

t=zeros(n\_mem,1);

for i=1:n\_mem

prompt="Enter the value of Thickness of the Member %d in mm : ";

t(i)=input(sprintf(prompt,i));

end

% Length of the Members in parallel in mm

L=zeros(n\_mem,1);

for i=1:n\_mem

prompt="Enter the value of Length of the element %d in mm : ";

L(i)=input(sprintf(prompt,i));

end

h1=input("Enter the mesh length value : "); % Length of the mesh element

h2=input("Enter the mesh width value : "); % Width of the mesh element

%% Boundary Conditions

bcs=[96 48 -500 0 1]; % x co-ordinate y co-ordinate Force Value X-axis Y-axis

%-------------------------------------------\* End of Input \*--------------------------------------------------

%% Assigning 8 nodes for each element

% Meshing each member into small elements of size l by w

ele=[];

x1=0;

y1=0;

for i=1:n\_mem

ele\_temp=[];

l=h1; % Length of each element

w=h2; % Width of each element

for j=l:l:L(i)

for k=w:w:W(i)

ele\_temp=[ele\_temp [x1 y1] [x1+l y1] [x1+l y1+w] [x1 y1+w] [x1+l/2 y1] [x1+l y1+w/2] [x1+l/2 y1+w] [x1 y1+w/2]];

y1=y1+w;

end

x1=x1+l;y1=y1-W(i);

end

x1=0;y1=y1+W(i);

ele(i,:)=ele\_temp;

end

%% Assigning Unique Node Number to every node

c\_node=[];

node=[1 0 0];p=1;

for i=1:size(ele,1)

for j=1:2:size(ele,2)

f1=0;

for k=1:p

if node(k,2) == ele(i,j) && node(k,3) == ele(i,j+1)

f1=1;

c\_node=[c\_node node(k,1)];

break;

end

end

if f1==0

p=p+1;

node=[node;[p ele(i,j) ele(i,j+1)]];

end

end

end

com\_node=[c\_node(2:length(c\_node))];

com\_node=unique(com\_node);

% unique is a Matlab in-built Function which removes the duplicate elements and sorts the remaining elements from minimum to maximum value

%% Elemental Connectivity

ele\_con=[];ele\_num=1; % For each element the respective element number and

for i=1:size(ele,1) % corresponding node numbers are stored in the matrix

for j=1:2:size(ele,2)

for k=1:size(node,1)

if node(k,2) == ele(i,j) && node(k,3) == ele(i,j+1)

ele\_con=[ele\_con;ele\_num node(k,1)];

break;

end

end

if mod(j+1,16) == 0

ele\_num=ele\_num+1;

end

end

end

%% Plotting the Generated mesh in graph

ne=1; % The plot consists of meshed member with node number and element

for i=1:size(ele,1) % number

for j=0:16:size(ele,2)-1

x\_m=[ele(i,j+1) ele(i,j+3) ele(i,j+5) ele(i,j+7)];

y\_m=[ele(i,j+2) ele(i,j+4) ele(i,j+6) ele(i,j+8)];

figure(1)

plot(x\_m,y\_m,"Color",'k','LineStyle','-')

hold on

x\_m1=[ele(i,j+9) ele(i,j+11) ele(i,j+13) ele(i,j+15)];

y\_m1=[ele(i,j+10) ele(i,j+12) ele(i,j+14) ele(i,j+16)];

plot(x\_m1,y\_m1,'ok')

x\_cg=ele(i,j+1)+h1/2;

y\_cg=ele(i,j+2)+h2/2;

%text(x\_cg,y\_cg,num2str(ne),"FontWeight","bold","FontSize",7);

ne=ne+1;

hold on

end

end

hold on

x\_m=[0 0];y\_m=[sum(W) 0];

plot(x\_m,y\_m,"Color",'k','LineStyle','-','Marker','o')

for i=1:size(node,1)

txt=sprintf('%d',node(i,1));

text(node(i,2),node(i,3),txt,"HorizontalAlignment","left","VerticalAlignment","bottom","FontSize",7);

end

hold off

%% Nodal Displacement Intialization

n\_node=size(node,1); % Number of nodes

n\_disp=2\*n\_node; % Number of nodal Displacements

U=zeros(n\_disp,1); % Assignment of displacement Matrix

kn\_disp=[]; % Known Nodal Displacement Matrix

uk\_disp=[]; % Unknown Nodal Displacement Matrix

for i=1:n\_node

if node(i,2) == 0

kn\_disp=[kn\_disp;2\*(i-1)+1;2\*(i-1)+2];

end % The node numbers for known displacements is stored in this matrix

end

for i=1:n\_node

if node(i,2) ~= 0

uk\_disp=[uk\_disp;2\*(i-1)+1;2\*(i-1)+2];

end % The node numbers for the unknown displacements is stored in this matrix

end

%% Calculation of property matrix for all the beams

D=cell(n\_mem,1);

for i=1:n\_mem

D{i} = (E(i)/(1-mu^2))\*[1 mu 0;mu 1 0;0 0 ((1-mu)/2)];

end

%% Stiffness Matrix

% Member Stiffness matrix

Km=cell(ele\_num-1,1); % The elemental stiffness matrix for every elements for all the three

p=1;x1=[];y1=[]; % members all calculated

for i=1:size(ele,1)

for j=0:16:size(ele,2)-1

x1=[ele(i,j+1) ele(i,j+3) ele(i,j+5) ele(i,j+7) ele(i,j+9) ele(i,j+11) ele(i,j+13) ele(i,j+15)];

y1=[ele(i,j+2) ele(i,j+4) ele(i,j+6) ele(i,j+8) ele(i,j+10) ele(i,j+12) ele(i,j+14) ele(i,j+16)];

Km{p}=stiffness(x1,y1,D{i},t(i));

p=p+1;

end

end

% Assembling Global Stiffness matrix

Kg=zeros(2\*n\_node,2\*n\_node);

me\_con=[];

for i=1:8:size(ele\_con,1)

me\_con1=[ele\_con((i:i+7),2)];

me\_con=me\_con1';

p=ele\_con(i,1);

for j=1:size(me\_con,2)

for k=1:size(me\_con,2)

q1=2\*me\_con(1,j)-1;

q2=2\*me\_con(1,j);

r1=2\*me\_con(1,k)-1;

r2=2\*me\_con(1,k);

Kg(q1,r1)=Kg(q1,r1)+Km{p}((2\*j-1),(2\*k-1));

Kg(q1,r2)=Kg(q1,r2)+Km{p}((2\*j-1),(2\*k));

Kg(q2,r1)=Kg(q2,r1)+Km{p}((2\*j),(2\*k-1));

Kg(q2,r2)=Kg(q2,r2)+Km{p}((2\*j),(2\*k));

end

end

end

%% Calculation of Fu and Fv at every Node

%Initialization of Force matrix

F=zeros(size(Kg,1),1);

Q1=[];

for i=1:size(Kg,1)

Q\_temp1=[];

p=find(kn\_disp==i); % Find is a Matlab in-built function which returns the value of

if p > 0 % linear index number of the respective value

else

for j=1:size(Kg,2)

q=find(kn\_disp==j);

if q > 0

else

Q\_temp1=[Q\_temp1 Kg(i,j)];

end

end

end

Q1=[Q1;Q\_temp1];

end

%Calculation of Force at every node

a=bcs(1,1); % Finding the node where the external force is applied

b=bcs(1,2);

for i=1:size(node,1)

if a==node(i,2) && b==node(i,3)

c=node(i,1);

c1=2\*c-1;

c2=2\*c;

end

end

force\_x=0; % Assigning the value of the external force to force\_x and force\_y variable

force\_y=0;

if bcs(1,4)== 1

force\_x=bcs(1,3);

elseif bcs(1,5)== 1

force\_y=bcs(1,3);

end

kn\_F=[]; % Assigning the values of the external force to known force matrix

for i=1:size(Kg,1)

p=find(kn\_disp==i);

if i==c1

kn\_F=[kn\_F; force\_x];

continue;

end

if i==c2

kn\_F=[kn\_F;force\_y];

continue;

end

if p > 0

else

kn\_F=[kn\_F; 0];

end

end

u\_temp1=inv(Q1)\*kn\_F;

uk\_disp=[uk\_disp u\_temp1]; % Values of the unknown displacements are found

Q2=[];

Q\_temp2=[];

for i=1:(size(Kg,1))

Q\_temp2=[];

p=find(kn\_disp==i);

if p>0

for j=1:(size(Kg,1))

Q\_temp2=[Q\_temp2 Kg(i,j)];

end

end

Q2=[Q2;Q\_temp2];

end

p=1;

for i=1:size(U,1) % Storing the values of displacements of each nodes in U matrix

if i== uk\_disp(p,1)

U(i,1)=uk\_disp(p,2);

p=p+1;

else

U(i,1)=0;

end

end

R\_Force=Q2\*U; % Forces at the support is found

F\_x=0;

F\_y=0;

for i=1:size(R\_Force,1)

if mod(i,2) == 0

F\_y=F\_y+R\_Force(i,1); % Summation of Forces in y-direction at the support

else

F\_x=F\_x+R\_Force(i,1); % Summation of Forces in x-direction at the support

end

end

%% Force Equilbrium

F\_t=zeros(size(F,1),1);

F\_t(c1,1)=force\_x;

F\_t(c2,1)=force\_y;

for i=1:size(kn\_disp,1)

a=kn\_disp(i,1);

F\_t(a,1)=R\_Force(i,1); % Checking whether the forces is in equilibrium

end

Eq=sum(F\_t);

%% Stress Matrix

ele\_n=ele\_num-1; % Total number of meshed elements in the problem

Sm=[];

Sm\_temp=[];

u=[];

p=1;x1=[];y1=[];

for i=1:size(ele,1)

for j=0:16:size(ele,2)-1

u=[];

x1=[ele(i,j+1) ele(i,j+3) ele(i,j+5) ele(i,j+7) ele(i,j+9) ele(i,j+11) ele(i,j+13) ele(i,j+15)];

y1=[ele(i,j+2) ele(i,j+4) ele(i,j+6) ele(i,j+8) ele(i,j+10) ele(i,j+12) ele(i,j+14) ele(i,j+16)];

for k=1:size(x1,2)

for m=1:n\_node

if node(m,2)==x1(1,k) && node(m,3)==y1(1,k)

q=2\*node(m,1)-1;

r=2\*node(m,1);

u=[u;U(q);U(r)];

break;

end

end

end

for n=1:size(x1,2) % Stress at each node for every elements is found

Sm\_temp=[stress(x1,y1,D{i},u,n)];

Sm=[Sm;[p Sm\_temp]];

end

p=p+1;

Sm\_temp=[];

end

end

%% Stress at each node

sm\_n=[ele\_con Sm(:,2:4)];

sm\_node=[];

temp=[];

p=1;

count=groupcounts(sm\_n(:,2));

for i=1:size(count,1)

n1=count(i,1);

for j=1:size(sm\_n,1)

if i==sm\_n(j,2)

temp=[temp;sm\_n(j,3:5)];

end

end % The average stress is each node is found

sx=sum(temp(:,1))/n1;

sy=sum(temp(:,2))/n1;

sxy=sum(temp(:,3))/n1;

sm\_node=[sm\_node;sx sy sxy];

temp=[];

end

%% plot Deformed shape

node\_def=zeros(size(node,1),size(node,2));

for i=1:2:size(uk\_disp,1)

p=(uk\_disp(i,1)+1)/2;

dx=uk\_disp(i,2)\*10^3;

dy=uk\_disp(i+1,2)\*10^3;

node\_def(p,1)=node(p,1);

node\_def(p,2)=node(p,2)+dx;

node\_def(p,3)=node(p,3)+dy;

end

for i=1:2:size(kn\_disp,1)

p=(kn\_disp(i,1)+1)/2;

node\_def(p,1)=node(p,1);

node\_def(p,2)=node(p,2);

node\_def(p,3)=node(p,3);

end

ele\_def=[];

for i=1:size(ele\_con,1)

for j=1:size(node,1)

if ele\_con(i,2) == node(j,1)

ele\_def=[ele\_def;node\_def(j,1:3)];

end

end

end

figure(2)

hold on

x1\_d=[];

y1\_d=[];

for i=0:8:size(ele\_def,1)-1

x1\_d=[ele\_def(i+1,2) ele\_def(i+5,2) ele\_def(i+2,2) ele\_def(i+6,2) ele\_def(i+3,2) ele\_def(i+7,2) ele\_def(i+4,2) ele\_def(i+8,2)];

y1\_d=[ele\_def(i+1,3) ele\_def(i+5,3) ele\_def(i+2,3) ele\_def(i+6,3) ele\_def(i+3,3) ele\_def(i+7,3) ele\_def(i+4,3) ele\_def(i+8,3)];

plot(x1\_d,y1\_d,"Color","b","LineStyle","--","LineWidth",0.5);

hold on

end

hold off

%% Strain matrix

sr\_node=[];

sr\_node\_temp=[];

for i=1:size(sm\_node,1)

if i <= (ele\_n/n\_mem)

t=1;

elseif i > (ele\_n/n\_mem) && i <= 2\*(ele\_n/n\_mem)

t=2;

else

t=3;

end

sm\_temp2=[sm\_node(i,1:3)]; % Strain at each node is calculated

sr\_node\_temp=10^6\*sm\_temp2\*inv(D{t});

sr\_node=[sr\_node;sr\_node\_temp];

end

% stiffness script file

function [K\_ele]=stiffness(X,Y,D,t)

n=zeros(8,1);

ndx=zeros(8,1);

nde=zeros(8,1);

G=zeros(4,16);

K\_ele=zeros(16,16);

xi=[-0.5774 0.5774]; % Number of Sampling points is two

eta=[-0.5774 0.5774]; % Number of Sampling points is two

w=[1 1 1 1]; % Total weight for four sampling points

% Shape functions when eight nodes is considered

for i=1:size(xi,2)

for j=1:size(eta,2)

n(1)=(-1/4)\*(1-xi(i))\*(1-eta(j))\*(1+xi(i)+eta(j));

n(2)=(-1/4)\*(1+xi(i))\*(1-eta(j))\*(1-xi(i)+eta(j));

n(3)=(-1/4)\*(1+xi(i))\*(1+eta(j))\*(1-xi(i)-eta(j));

n(4)=(-1/4)\*(1-xi(i))\*(1+eta(j))\*(1+xi(i)-eta(j));

n(5)=(1/2)\*(1-xi(i))\*(1+xi(i))\*(1-eta(j));

n(6)=(1/2)\*(1+xi(i))\*(1+eta(j))\*(1-eta(j));

n(7)=(1/2)\*(1-xi(i))\*(1+xi(i))\*(1+eta(j));

n(8)=(1/2)\*(1-xi(i))\*(1+eta(j))\*(1-eta(j));

% Derivation of shape functions with respect to xi

ndx(1)=(1/4)\*(1-eta(j))\*(2\*xi(i)+eta(j));

ndx(2)=(1/4)\*(1-eta(j))\*(2\*xi(i)-eta(j));

ndx(3)=(1/4)\*(1+eta(j))\*(2\*xi(i)+eta(j));

ndx(4)=(1/4)\*(1+eta(j))\*(2\*xi(i)-eta(j));

ndx(5)=(-1)\*(1-eta(j))\*xi(i);

ndx(6)=(1/2)\*(1-eta(j))\*(1+eta(j));

ndx(7)=(-1)\*(1+eta(j))\*xi(i);

ndx(8)=(-1/2)\*(1-eta(j))\*(1+eta(j));

% Derivation of shape functions with respect to eta

nde(1)=(1/4)\*(1-xi(i))\*(2\*eta(j)+xi(i));

nde(2)=(1/4)\*(1+xi(i))\*(2\*eta(j)-xi(i));

nde(3)=(1/4)\*(1+xi(i))\*(2\*eta(j)+xi(i));

nde(4)=(1/4)\*(1-xi(i))\*(2\*eta(j)-xi(i));

nde(5)=(-1/2)\*(1+xi(i))\*(1-xi(i));

nde(6)=(-1)\*(1+xi(i))\*eta(j);

nde(7)=(1/2)\*(1+xi(i))\*(1-xi(i));

nde(8)=(-1)\*(1-xi(i))\*eta(j);

G=[ndx(1) 0 ndx(2) 0 ndx(3) 0 ndx(4) 0 ndx(5) 0 ndx(6) 0 ndx(7) 0 ndx(8) 0;

nde(1) 0 nde(2) 0 nde(3) 0 nde(4) 0 nde(5) 0 nde(6) 0 nde(7) 0 nde(8) 0;

0 ndx(1) 0 ndx(2) 0 ndx(3) 0 ndx(4) 0 ndx(5) 0 ndx(6) 0 ndx(7) 0 ndx(8);

0 nde(1) 0 nde(2) 0 nde(3) 0 nde(4) 0 nde(5) 0 nde(6) 0 nde(7) 0 nde(8)];

j11=0;

j12=0;

j21=0;

j22=0;

for k=1:length(n)

j11=j11+(ndx(k)\*X(k)\*10^-3);

j12=j12+(ndx(k)\*Y(k)\*10^-3);

j21=j21+(nde(k)\*X(k)\*10^-3);

j22=j22+(nde(k)\*Y(k)\*10^-3);

end

J=[j11 j12;j21 j22];

jd=det(J);

A=(1/jd)\*[j22 -j12 0 0;0 0 -j21 j11;-j21 j11 j22 -j12];

B=A\*G;

K=B'\*D\*B\*jd\*t;

K\_ele=K\_ele+K;

end

end

end

% The Function file returns the K\_ele matrix to the Main script file at the end

% stress script file

function [st\_node]=stress(X,Y,D,U,i)

n=zeros(8,1);

ndx=zeros(8,1);

nde=zeros(8,1);

G=zeros(4,16);

st\_node=[];

xi=[-1 -1 1 -1 0 1 0 -1]; % Eight points are considered for eight nodes

eta=[-1 1 1 1 -1 0 1 0]; % Eight points are considered for eight nodes

w=[1 1 1 1 1 1 1 1 1]; % Total weight for four sampling points

% Shape functions when eight nodes is considered

n(1)=(-1/4)\*(1-xi(i))\*(1-eta(i))\*(1+xi(i)+eta(i));

n(2)=(-1/4)\*(1+xi(i))\*(1-eta(i))\*(1-xi(i)+eta(i));

n(3)=(-1/4)\*(1+xi(i))\*(1+eta(i))\*(1-xi(i)-eta(i));

n(4)=(-1/4)\*(1-xi(i))\*(1+eta(i))\*(1+xi(i)-eta(i));

n(5)=(1/2)\*(1-xi(i))\*(1+xi(i))\*(1-eta(i));

n(6)=(1/2)\*(1+xi(i))\*(1+eta(i))\*(1-eta(i));

n(7)=(1/2)\*(1-xi(i))\*(1+xi(i))\*(1+eta(i));

n(8)=(1/2)\*(1-xi(i))\*(1+eta(i))\*(1-eta(i));

% Derivation of shape functions with respect to xi

ndx(1)=(1/4)\*(1-eta(i))\*(2\*xi(i)+eta(i));

ndx(2)=(1/4)\*(1-eta(i))\*(2\*xi(i)-eta(i));

ndx(3)=(1/4)\*(1+eta(i))\*(2\*xi(i)+eta(i));

ndx(4)=(1/4)\*(1+eta(i))\*(2\*xi(i)-eta(i));

ndx(5)=(-1)\*(1-eta(i))\*xi(i);

ndx(6)=(1/2)\*(1-eta(i))\*(1+eta(i));

ndx(7)=(-1)\*(1+eta(i))\*xi(i);

ndx(8)=(-1/2)\*(1-eta(i))\*(1+eta(i));

% Derivation of shape functions with respect to eta

nde(1)=(1/4)\*(1-xi(i))\*(2\*eta(i)+xi(i));

nde(2)=(1/4)\*(1+xi(i))\*(2\*eta(i)-xi(i));

nde(3)=(1/4)\*(1+xi(i))\*(2\*eta(i)+xi(i));

nde(4)=(1/4)\*(1-xi(i))\*(2\*eta(i)-xi(i));

nde(5)=(-1/2)\*(1+xi(i))\*(1-xi(i));

nde(6)=(-1)\*(1+xi(i))\*eta(i);

nde(7)=(1/2)\*(1+xi(i))\*(1-xi(i));

nde(8)=(-1)\*(1-xi(i))\*eta(i);

G=[ndx(1) 0 ndx(2) 0 ndx(3) 0 ndx(4) 0 ndx(5) 0 ndx(6) 0 ndx(7) 0 ndx(8) 0;

nde(1) 0 nde(2) 0 nde(3) 0 nde(4) 0 nde(5) 0 nde(6) 0 nde(7) 0 nde(8) 0;

0 ndx(1) 0 ndx(2) 0 ndx(3) 0 ndx(4) 0 ndx(5) 0 ndx(6) 0 ndx(7) 0 ndx(8);

0 nde(1) 0 nde(2) 0 nde(3) 0 nde(4) 0 nde(5) 0 nde(6) 0 nde(7) 0 nde(8)];

j11=0;

j12=0;

j21=0;

j22=0;

for k=1:size(X,2)

j11=j11+(ndx(k)\*X(k)\*10^-3);

j12=j12+(ndx(k)\*Y(k)\*10^-3);

j21=j21+(nde(k)\*X(k)\*10^-3);

j22=j22+(nde(k)\*Y(k)\*10^-3);

end

J=[j11 j12;j21 j22];

jd=det(J);

A=(1/jd)\*[j22 -j12 0 0;0 0 -j21 j11;-j21 j11 j22 -j12];

B=A\*G;

st\_ele=(D\*B\*U)/10^6;

st\_node=st\_ele'; % The Function file returns the st\_node matrix to the Main script file

% strain script file

end

function [B]=strain(X,Y,i)

n=zeros(8,1);

ndx=zeros(8,1);

nde=zeros(8,1);

G=zeros(4,16);

xi=[-1 -1 1 -1 0 1 0 -1]; % Eight points are considered for eight nodes

eta=[-1 1 1 1 -1 0 1 0]; % Eight points are considered for eight nodes

w=[1 1 1 1 1 1 1 1 1]; % Total weight for four sampling points

% Shape functions when eight nodes is considered

n(1)=(-1/4)\*(1-xi(i))\*(1-eta(i))\*(1+xi(i)+eta(i));

n(2)=(-1/4)\*(1+xi(i))\*(1-eta(i))\*(1-xi(i)+eta(i));

n(3)=(-1/4)\*(1+xi(i))\*(1+eta(i))\*(1-xi(i)-eta(i));

n(4)=(-1/4)\*(1-xi(i))\*(1+eta(i))\*(1+xi(i)-eta(i));

n(5)=(1/2)\*(1-xi(i))\*(1+xi(i))\*(1-eta(i));

n(6)=(1/2)\*(1+xi(i))\*(1+eta(i))\*(1-eta(i));

n(7)=(1/2)\*(1-xi(i))\*(1+xi(i))\*(1+eta(i));

n(8)=(1/2)\*(1-xi(i))\*(1+eta(i))\*(1-eta(i));

% Derivation of shape functions with respect to xi

ndx(1)=(1/4)\*(1-eta(i))\*(2\*xi(i)+eta(i));

ndx(2)=(1/4)\*(1-eta(i))\*(2\*xi(i)-eta(i));

ndx(3)=(1/4)\*(1+eta(i))\*(2\*xi(i)+eta(i));

ndx(4)=(1/4)\*(1+eta(i))\*(2\*xi(i)-eta(i));

ndx(5)=(-1)\*(1-eta(i))\*xi(i);

ndx(6)=(1/2)\*(1-eta(i))\*(1+eta(i));

ndx(7)=(-1)\*(1+eta(i))\*xi(i);

ndx(8)=(-1/2)\*(1-eta(i))\*(1+eta(i));

% Derivation of shape functions with respect to eta

nde(1)=(1/4)\*(1-xi(i))\*(2\*eta(i)+xi(i));

nde(2)=(1/4)\*(1+xi(i))\*(2\*eta(i)-xi(i));

nde(3)=(1/4)\*(1+xi(i))\*(2\*eta(i)+xi(i));

nde(4)=(1/4)\*(1-xi(i))\*(2\*eta(i)-xi(i));

nde(5)=(-1/2)\*(1+xi(i))\*(1-xi(i));

nde(6)=(-1)\*(1+xi(i))\*eta(i);

nde(7)=(1/2)\*(1+xi(i))\*(1-xi(i));

nde(8)=(-1)\*(1-xi(i))\*eta(i);

G=[ndx(1) 0 ndx(2) 0 ndx(3) 0 ndx(4) 0 ndx(5) 0 ndx(6) 0 ndx(7) 0 ndx(8) 0;

nde(1) 0 nde(2) 0 nde(3) 0 nde(4) 0 nde(5) 0 nde(6) 0 nde(7) 0 nde(8) 0;

0 ndx(1) 0 ndx(2) 0 ndx(3) 0 ndx(4) 0 ndx(5) 0 ndx(6) 0 ndx(7) 0 ndx(8);

0 nde(1) 0 nde(2) 0 nde(3) 0 nde(4) 0 nde(5) 0 nde(6) 0 nde(7) 0 nde(8)];

j11=0;

j12=0;

j21=0;

j22=0;

for k=1:size(X,2)

j11=j11+(ndx(k)\*X(k)\*10^-3);

j12=j12+(ndx(k)\*Y(k)\*10^-3);

j21=j21+(nde(k)\*X(k)\*10^-3);

j22=j22+(nde(k)\*Y(k)\*10^-3);

end

J=[j11 j12;j21 j22];

jd=det(J);

A=(1/jd)\*[j22 -j12 0 0;0 0 -j21 j11;-j21 j11 j22 -j12];

B=A\*G; % The Function file returns the B matrix to the Main script file

end

**INPUT:**

Enter the value of Young's Modulus of the Member 1 in GPa: 25

Enter the value of Young's Modulus of the Member 2 in GPa: 50

Enter the value of Young's Modulus of the Member 3 in GPa: 100

Enter the value of Width of the Member 1 in mm : 16

Enter the value of Width of the Member 2 in mm : 16

Enter the value of Width of the Member 3 in mm : 16

Enter the value of Thickness of the Member 1 in mm : 1

Enter the value of Thickness of the Member 2 in mm : 1

Enter the value of Thickness of the Member 3 in mm : 1

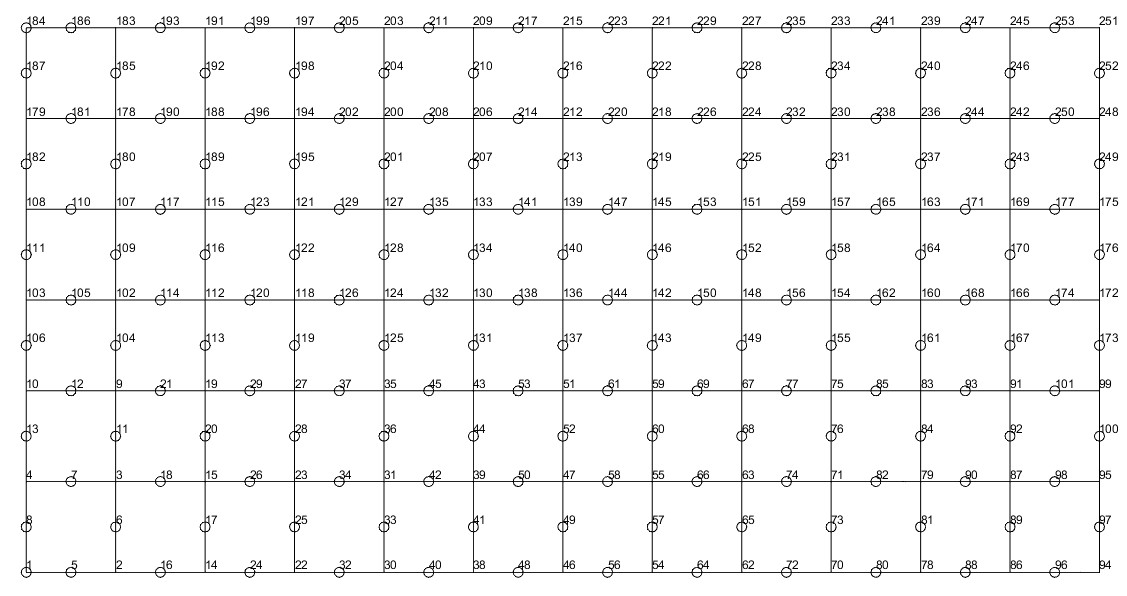
Enter the value of Length of the element 1 in mm : 96

Enter the value of Length of the element 2 in mm : 96

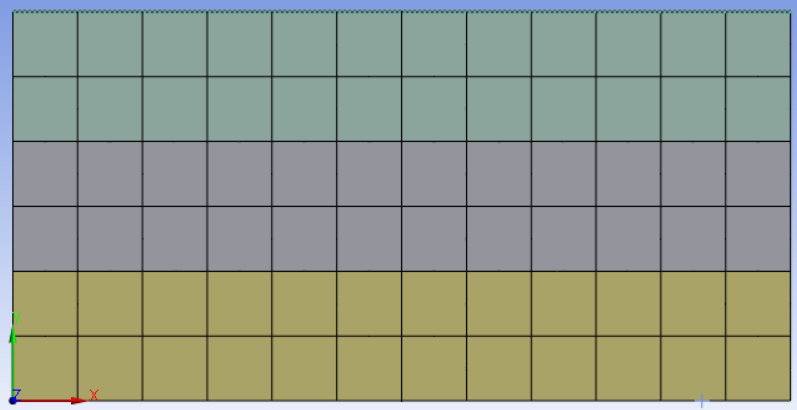
Enter the value of Length of the element 3 in mm : 96

Enter the mesh length value : 8

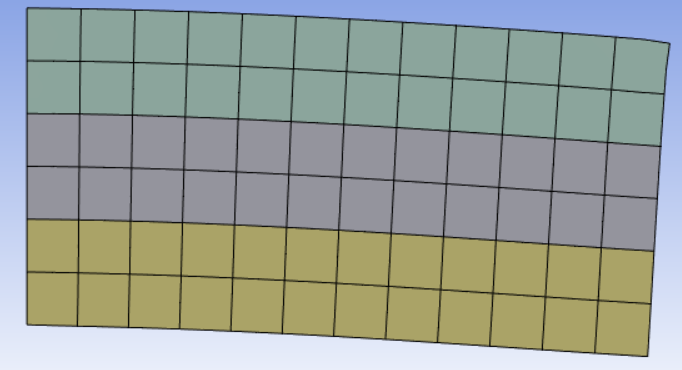
Enter the mesh width value : 8



**USING ANSYS:**

****

**Undeformed 8mm x 8mm mesh**

****

**Deformed 8mm x 8mm mesh**