Finite Element Method

STIFFNESS & DISPLACEMENT MATRIX OF A COMPOSITE PLANE STRESS PROBLEM (MATLAB CODE)

Authors:

AYUSH CHAPPARIA (11ME33028)

SAHIL GROVER (11ME33026)

For any further communication:

ayush.9924@gmail.com | +91 - 9874361217

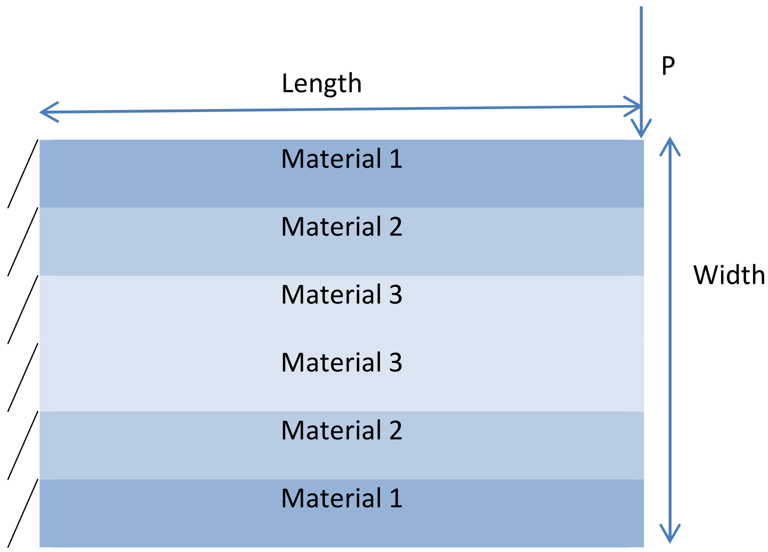
sahilgroversahil@gmail.com | +91 - 9434363574

Contents

Introduction

Problem Statement

A composite plane plate attached to the wall on left side as shown below undergoes a force of magnitude P on the top right corner.



Perform an FEM analysis on the plate assuming the thickness of the plate is very small as compared to the width and length of the plate. Also, assume that the width of each strip is equal for all materials.

Inputs:

- Length, Width and Thickness of the plate
- The Modulus and Poisson's ratio of all three materials
- Force (P)

Solution

Assuming that the inputs are:
Length = 100 mm
Width = 60 mm
Thickness = 1 mm
Elasticity, Poisson's ratio =

- Material 1 : 100 GPA, 0.25
- Material 2: 50 GPA, 0.3
- Material 3: 25 GPA, 0.35

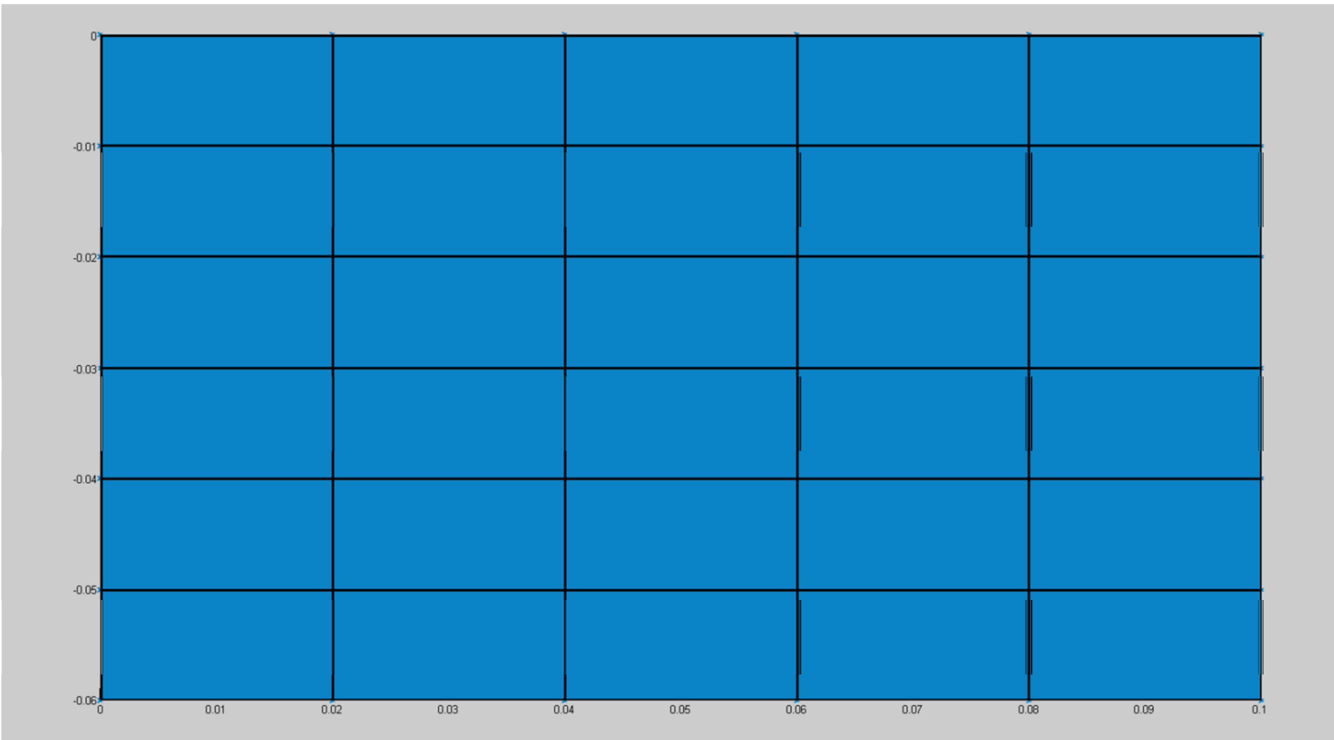
Force = 5000 N

Simulation1:

Assumptions

No of Elements in X direction =5;

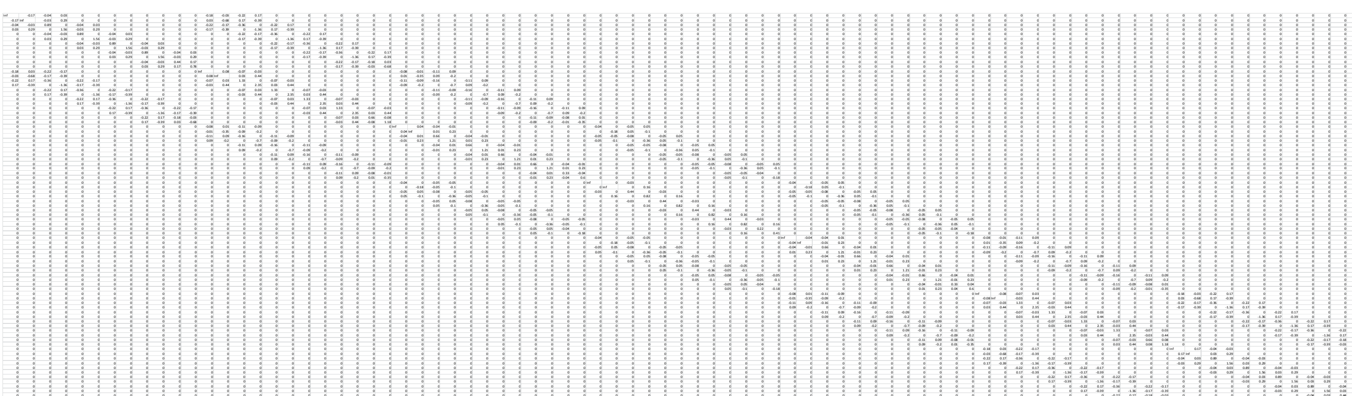
No of elements in Y direction = 6;



The Global Stiffness matrix

After the code runs, the Global Stiffness Matrix Obtained is of order 84 X 84.

The Global Stiffness matrix obtained https://goo.gl/JNZIJO (the values in the file needs multiplication with "1e08" for real values). The file looks like (see below)



Nodal Displacements

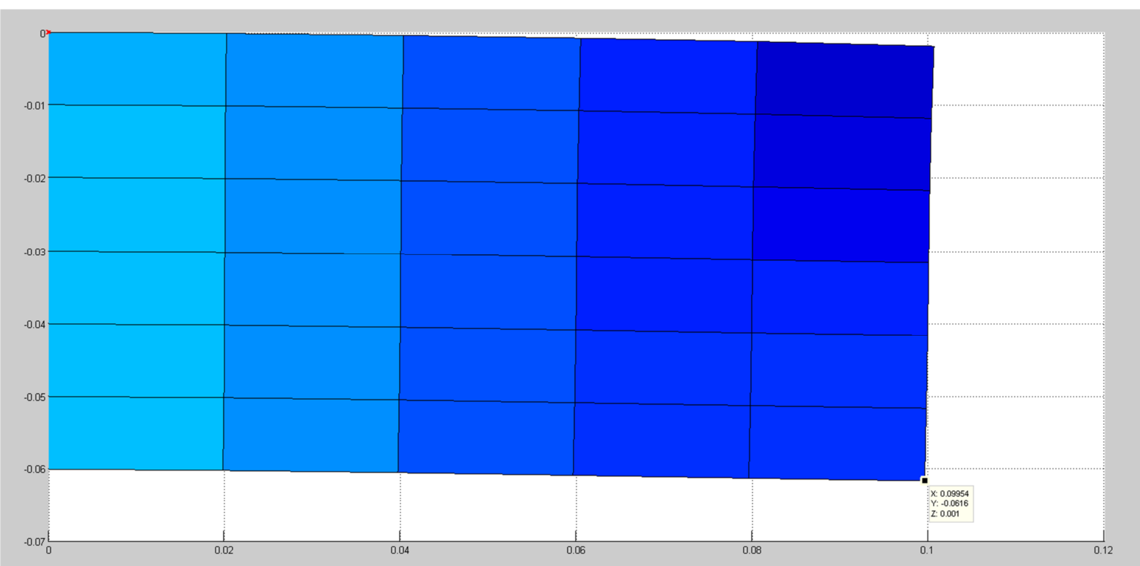
The nodal displacements obtained are:
X displacements =

1.0e-03 *

0	0.1971	0.3340	0.4392	0.5305	0.5929
0	0.0891	0.1783	0.2431	0.2697	0.2903
0	0.0219	0.0551	0.0736	0.0661	0.0889
0	0.0013	-0.0032	-0.0213	-0.0422	-0.0208
0	-0.0197	-0.0572	-0.0995	-0.1294	-0.1316
0	-0.0889	-0.1778	-0.2411	-0.2775	-0.2880
0	-0.2008	-0.3374	-0.4243	-0.4593	-0.4640

Y displacements =

0	-0.0002	-0.0005	-0.0008	-0.0013	-0.0019
0	-0.0002	-0.0004	-0.0008	-0.0013	-0.0019
0	-0.0001	-0.0004	-0.0008	-0.0013	-0.0018
0	-0.0002	-0.0004	-0.0008	-0.0013	-0.0017
0	-0.0002	-0.0004	-0.0008	-0.0012	-0.0016
0	-0.0002	-0.0005	-0.0008	-0.0012	-0.0016
0	-0.0002	-0.0005	-0.0008	-0.0012	-0.0016



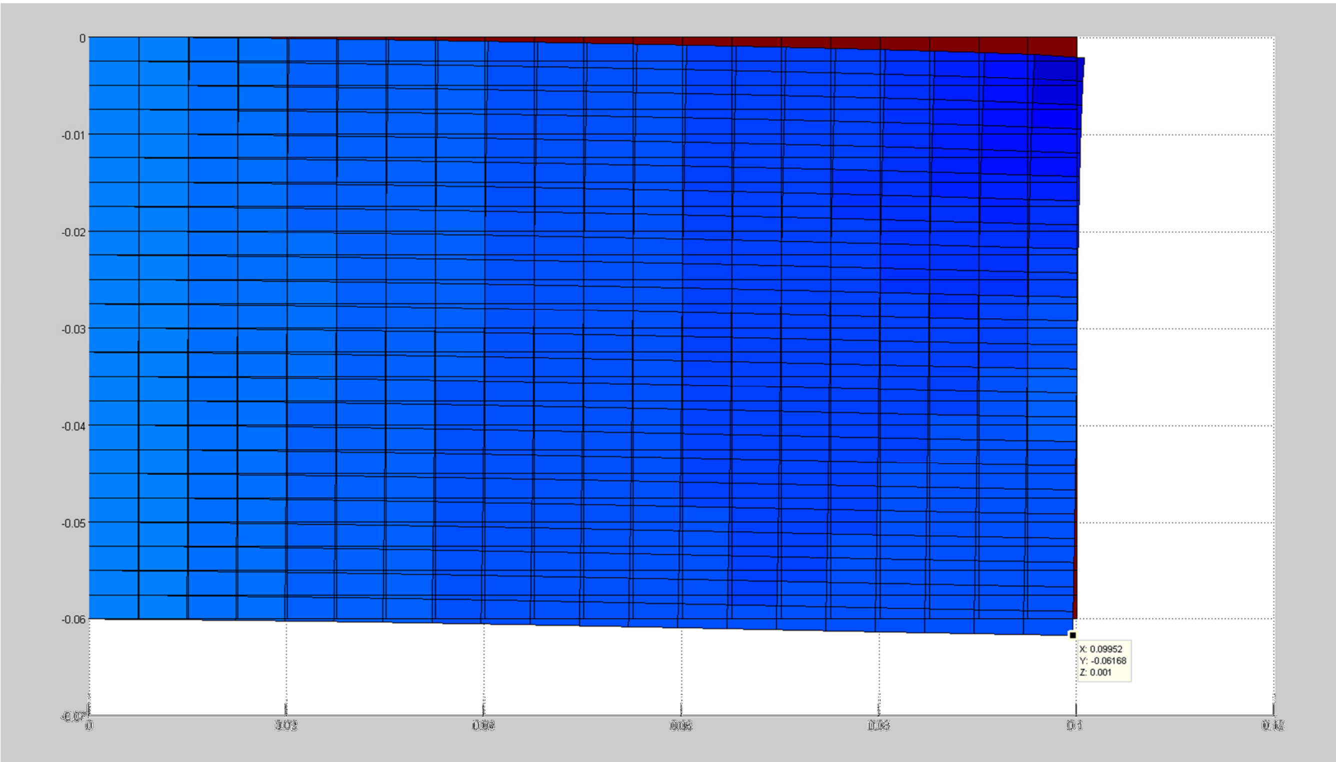
Simulation 2:

Assumptions

No of Elements in X direction =20;

No of elements in Y direction = 24;

Nodal Displacements:



Theory

Approach:

The following flow chart explains the approach used. Each step is coded in MATLAB such that the user would have full control over the inputs.

Inputs

Initially user is asked to define all the input variables. These inputs include:

1. **Beam definition**: Length, Width, Thickness, and modulus of elasticity and poison's ratio of each layer.
2. **Mess Definition**: Number of elements in x and y direction.
3. **Load definition**: Magnitude and location of the applied load.

Rigidity Matrix

Using the material property of each layer (modulus of elasticity and poison's ratio) the rigidity matrix is calculated for each layer.
Since we have three different types of layers we calculate the three rigidity matrices and store them in an array: ***rigidity** [3][3][3]*

Element Stiffeness Matix

The following process is used for calculating the element stiffness matrices:

1. **Shape Functions** are calculated assuming 4-nodal elements.
2. Using the shape function and the rigidity matrix, stiffness matrix at a point is calculated.
3. **Gauss Quadrature** method is used for numerical integration in 2-D.

Global Stiffness Matrix

Once the element stiffness matrices for each type of layer is calculated, they are assembled together to form the Global stiffness Matrix.

Boundary Conditions

1. **External Forces**: The external forces provided by the user are applied on the respective nodes in the respective direction.
2. **Support**: Since the beam is a cantilever type the displacements of the nodes attached to the support will we zero.

Nodal Displacement

Once we have the Global stiffness matrix and the boundary conditions, we would have N systems of equations with N unknowns [displacements]. We apply the **GAUSSIAN ELIMINATION** to solve these equations and calculate the displacements of each node in 2 dimensions.

ELement Stress Matrix

Using the Nodal displacement matrix, the elemental rigidity matrices and the elemental stiffness matrices the longitudinal, transversal and shear stresses at each element are calculated using Plane stress equations.

Appendix

MATLAB CODE

Download the code files from https://goo.gl/rSDbZ2

Main File

Name of the file -> FEM_final.m

```
clear all;
%% Defining variables
len = 100/1000; % in m
width = 60/1000; % in m
nx=5;
ny=6;
ndeg=2;
no_gauss=2;
thick=1/1000; % in m
elasticity=[100*1000000000,50*1000000000,25*1000000000]; % in PA
mu=[0.25,0.3,0.35];
force=-5000;
force_loc=(nx+1)*2;
% Defining variables ends

len_elem=len/nx;
width_elem=width/ny;
total_node=(nx+1)*(ny+1);
tot_free=(nx+1)*(ny+1)*ndeg;
free_elem=4*ndeg;


a=meshgrid(0:len_elem:len,1:ny+1)
b=transpose(meshgrid((0:width_elem:width)*-1,1:nx+1))
c=thick+zeros(ny+1,nx+1);
surf(a,b,c)
view(2);
hold on;
%*****************************************************************************
*******
%*****************************************************************************
*******
%% Initialize Rigidity Matrix
rigidity=zeros(3,3,3);
for i = 1 : 3
    d=elasticity(i)/(1-mu(i)*mu(i));
    rigidity(1,1,i)=d;
    rigidity(1,2,i)=d*mu(i);
    rigidity(1,3,i)=0;
    rigidity(2,1,i)=d*mu(i);
    rigidity(2,2,i)=d;
    rigidity(2,3,i)=0;
    rigidity(3,1,i)=0;
```

```
    rigidity(3,2,i)=0;
    rigidity(3,3,i)=d*(1-mu(i))/2;
end
%**********************************************************************************
%**********************************************************************************
 xg=[0,len_elem,len_elem,0];
 yg=[width_elem,width_elem,0,0];
 zx=[-1,1,1,-1];
 zy=[1,1,-1,-1];

sfd = zeros(4);
sfdz = zeros(4);
sfde = zeros(4);
stiff=zeros(free_elem,free_elem,3);
%% Developing a local element stiffness matrix using shape functions and concept of
zi and eta.
for i = 1:no_gauss
    for j = 1: no_gauss
        [zeta,eta,hi,hj]=gauss_cau(i,j,no_gauss); %Refer gauss_cau function file to
get a insight on how it works
        for l = 1:4
            zetaz=zeta*zx(l);
            etaz=eta*zy(l);
            sfd(l) = 0.25*(1+etaz)*(1+zetaz);
            sfdz(l) = 0.25*(1+etaz)*zx(l);
            sfde(l) = 0.25*(1+zetaz)*zy(l);
        end
        BB=zeros(3,free_elem);
        DXZ=0;
        DYZ=0;
        DXE=0;
        DYE=0;
        for k=1:4
            DXZ=DXZ+sfdz(k)*xg(k);
            DYZ=DYZ+sfdz(k)*yg(k);
            DXE=DXE+sfde(k)*xg(k);
            DYE=DYE+sfde(k)*yg(k);
        end
        ZAC=DXZ*DYE-DYZ*DXE;
        DXZI=DYE/ZAC;
        DYZI=-DYZ/ZAC;
        DXEI=-DXE/ZAC;
        DYEI=DXZ/ZAC;
        for ii=1:4;
            k=2*(ii-1);
            DNX=sfdz(ii)*DXZI+sfde(ii)*DYZI;
            DNY=sfdz(ii)*DXEI+sfde(ii)*DYEI;
            BB(1,k+1)=DNX;
            BB(2,k+2)=DNY;
            BB(3,k+1)=DNY;
            BB(3,k+2)=DNX;
        end
        ASAT=zeros(free_elem,free_elem,3);
        for ii =1:3
            for m=1:free_elem;
                for n=1:free_elem;
                    if n>=m
```

```
                        for l=1:3
                            for ki=1:3
ASAT(m,n,ii)=ASAT(m,n,ii)+BB(l,m)*rigidity(l,ki,ii)*BB(ki,n)*ZAC*thick;
                            end
                        end
                    end
                end
            end
        end
        for ii =1:3
            for m=1:free_elem;
                for n=1:free_elem;
                    if n>=m
                        stiff(m,n,ii)=stiff(m,n,ii)+ASAT(m,n,ii)*hi*hj;
                    end
                end
            end
        end

    end
end

for ii =1:3
    for i=1:free_elem
        for j=1:free_elem
            stiff(j,i,ii)=stiff(i,j,ii);
        end
    end
end
% Stiffness Matrix Building Done

%% NOD = Matrix containing Nodes of each element
NOD=zeros(nx*ny,4);
for i = 1:nx*ny
    nx1=nx+1;
    nxi=floor((i-1)/nx);
    for j= 1:2
        NOD(i,j)=nx1*nxi+(i-nx*nxi)+j-1;
    end
    for j=3:4
        NOD(i,j)=nx1*(nxi+1)+(i-nx*nxi)+4-j;
    end
end

%% Assembling the local stifness Matrix on the global stiffness matrix
stiffo = zeros(tot_free,tot_free);
for i=1:nx*ny
    if i<=nx*ny/6 || i>nx*ny*5/6  %condition1
        for j=1:4
            for k=1:4
                stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2-1)) =
stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2-1)) + stiff(j*2-1,k*2-1,1);
                stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2))   =
stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2)) + stiff(j*2-1,k*2,1);
```

```
                stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2-1)) =
stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2-1)) + stiff(j*2,k*2-1,1);
                stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2))   =
stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2)) + stiff(j*2,k*2,1);
            end
        end
        elseif i<=nx*ny*1/3 || i>nx*ny*2/3 %condition2
                for j=1:4
                    for k=1:4
                        stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2-1)) =
stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2-1)) + stiff(j*2-1,k*2-1,2);
                        stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2))   =
stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2)) + stiff(j*2-1,k*2,2);
                        stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2-1)) =
stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2-1)) + stiff(j*2,k*2-1,2);
                        stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2))   =
stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2)) + stiff(j*2,k*2,2);
                    end
                end
            else
                for j=1:4
                    for k=1:4
                        stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2-1)) =
stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2-1)) + stiff(j*2-1,k*2-1,3);
                        stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2))   =
stiffo(int64(NOD(i,j)*2-1),int64(NOD(i,k)*2)) + stiff(j*2-1,k*2,3);
                        stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2-1)) =
stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2-1)) + stiff(j*2,k*2-1,3);
                        stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2))   =
stiffo(int64(NOD(i,j)*2),int64(NOD(i,k)*2)) + stiff(j*2,k*2,3);
                    end
                end

    end
end

% Assembling done

% Code for forming the halfband stiffness matrix
% NHB = (nx+3)*2;
% halfband_stiffo = zeros(tot_free,NHB);
% for j=1:NHB
%   for i=1:tot_free
%       if(j+i-1<=tot_free)
%           halfband_stiffo(i,j) = stiffo(i,j+i-1);
%       end
%   end
% end
% halfband_stiffo


%% Developing Force matrix
pload=zeros(tot_free,1);
pload(force_loc,1)=force;
% Force matrix developed
%% Boundary conditions
```

```
for i = 1:ny+1
    j=(i-1)*(nx+1)+1;
    stiffo(j*2,j*2)=Inf;
    stiffo(j*2-1,j*2-1)=Inf;
end
%Boundary fitting over

%% Calculating Displacements
x=zeros(tot_free,1);
u=zeros(tot_free,tot_free);
[x,u]=gausselim(stiffo,pload);
%Displacement Calculation done

%%  Printing displacements
X_displacement = x(1:2:(length(x)-1));
Y_displacement = x(2:2:length(x));
X_displacement= transpose(reshape(X_displacement,(nx+1),(ny+1))) % Prints the X
displacement of the nodes
Y_displacement= transpose(reshape(Y_displacement,(nx+1),(ny+1))) % Prints the Y
displacement of the nodes

surf(a+X_displacement,b+Y_displacement,c,gradient(Y_displacement));
view(2);
%% Stress Generation Starts
for n=1:nx*ny
        for j=1:4
            for l = 1:4
                zetaz=zx(j)*zx(l);
                etaz=zy(j)*zy(l);
                sfd(l) = 0.25*(1+etaz)*(1+zetaz);
                sfdz(l) = 0.25*(1+etaz)*zx(l);
                sfde(l) = 0.25*(1+zetaz)*zy(l);
            end
            BB=zeros(3,free_elem);
            DXZ=0;
            DYZ=0;
            DXE=0;
            DYE=0;
            for k=1:4
                DXZ=DXZ+sfdz(k)*xg(k);
                DYZ=DYZ+sfdz(k)*yg(k);
                DXE=DXE+sfde(k)*xg(k);
                DYE=DYE+sfde(k)*yg(k);
            end
            % ZAC=DXZ*DYE-DYZ*DXE;
            DXZI=DYE/ZAC;
            DYZI=-DYZ/ZAC;
            DXEI=-DXE/ZAC;
            DYEI=DXZ/ZAC;
            for ii=1:4;
                k=2*(ii-1);
                DNX=sfdz(ii)*DXZI+sfde(ii)*DYZI;
                DNY=sfdz(ii)*DXEI+sfde(ii)*DYEI;
                BB(1,k+1)=DNX;
                BB(2,k+2)=DNY;
                BB(3,k+1)=DNY;
```

```
                BB(3,k+2)=DNX;
            end
            elemnode_disp=zeros(free_elem,1);
            for ii=1:4
                for l=1:2
                    k=(NOD(n,ii)-1)*ndeg+l;
                    m=(ii-1)*ndeg+l;
                    elemnode_disp(m,1)=x(k,1);
                end
            end
            stress=zeros(3,1);
            for m=1:3
                for l =1:3
                    for k=1:free_elem

stress(m,1)=stress(m,1)+rigidity(m,l)*BB(l,k)*elemnode_disp(k,1)/2;
                    end
                end
            end
            % fprintf('Elem node = %i , Node no = %i\n',n,j);
        end
end
%Stress Generation ends

%% Code: if required to print global stiffness
% fileID = fopen('globalstiffnes.csv','w');
% for i= 1:tot_free
% fprintf(fileID,'%f,',stiffo(i,1:tot_free));
% fprintf(fileID,'\n');
% end
% fclose(fileID);
```

Gauss Integration File

Filename -> gauss_cau.m

```
function [zeta,eta,hi,hj] = gauss_cau(I,J,no_gauss)
        if no_gauss==2
            hi=1;
            hj=1;
                if I==1
                    zeta = -0.577350269189626;
                else
                    zeta = 0.577350269189626;
                end
                if J==1
                    eta = -0.577350269189626;
                else
                    eta = 0.577350269189626;
                end
        else
                if I==1
                    zeta=-0.774596669241483;
```

```
                hi=5/9;
            elseif I==2
                zeta=0;
                hi=8/9;
            else
                zeta=0.774596669241483;
                hi=5/9;
            end
            if J==1
                eta=-0.774596669241483;
                hj=5/9;
            elseif I==2
                eta=0;
                hj=8/9;
            else
                eta=0.774596669241483;
                hj=5/9;
            end
    end
end
```

Gauss Simultaneous Linear Equation Solver

Filename -> gausselim.m

```
function [x,U]=gausselim(A,b)

% function to perform gauss eliminination

%FORWARD ELIMINATION

n=length(b);

m=zeros(n,1);

x=zeros(n,1);

for k =1:n-1;

    %compute the kth column of M
    m(k+1:n) = A(k+1:n,k)/A(k,k);
    %compute
    %An=Mn*An-1;
    %bn=Mn*bn-1;
    for i=k+1:n
        A(i, k+1:n) = A(i,k+1:n)-m(i)*A(k,k+1:n);
    end;
    b(k+1:n)=b(k+1:n)-b(k)*m(k+1:n);
end

U= triu(A);
```

```
%BACKWARD ELIMINATION

x(n)=b(n)/A(n,n);

for k =n-1:-1:1;

    b(1:k)=b(1:k)-x(k+1)* U(1:k,k+1);
    x(k)=b(k)/U(k,k);
end

end
```