

```

%%GEOMETRY:-
% Total number of Node, nNode = n ('n' is an integer
number.)
% Total number of Element, nElem = n ('n' is an integer
number.)
% Degree of Freedom Per Node, DofNode = 1 or 2;
% Nodes Connect to Each Element, ElemNode = [E1N1
E1N2;E2N3 E2N4;...]
% Coordiante of Nodes, NodeCoord = [x1 y1; x2 y2;...]
% Area of Each Element, Area = [A1 A2 ...]
%MATERIAL PROPERTY:-
% Young modulus of each element . Emod = [Y1 Y2 ...]
%BOUNDARY CONDITION:-
% Applied Force at every node, Load= [F1x F1y; F2x
F2y...]
% Displacement Constraint at each node, Bc = [u1, v1; u2
v2;...] . Write 'nan' if 'NO Constraint'
% Scaling Deformation for plot, DefScale = n

% % % % % % % % % % % % % % % % % % % % % %
% Zarif Ahmad Razin Bhuiyan %
% Email: zarifahmad369@gmail.com %
% Date Created: 17 July, 2020 %
% Last Modified: 17 July, 2020 %
% % % % % % % % % % % % % % % % % % % % % %

%% Input
clear all;
clc;

%Geometry

nNode = 12; % Total number of Node. nNode = n
nElem = 22; % Total number of Element. nElem = n
DofNode = 2; %Degree of Freedom Per Node. DofNode = 1 or
2;
ElemNode = [1 2;2 3;2 4;1 4;3 4;3 5;3 6;4 6;5 6;5 7;6
8;5 8;6 7;7 8;7 9;8 9;8 10;9 10;9 11;11 12;10 11;10 12];
% Nodes Connect to Each Element. ElemNode = [E1N1
E1N2;E2N3 E2N4;...]
NodeCoord = [0 0;0 4;3 4;3 0;6 4;6 0;9 4;9 0;12 4;12
0;15 4;15 0]; % Coordiante of Nodes. NodeCoord = [x1 y1;
x2 y2;...]
Area = ones(1,nElem)*10^-2; % Area of Each Element
Area = [A1 A2 ...]

% Material Property

```

```

Emod=ones(1,nElem)*1.5*10^11 ; %Young modulus of each
element . Emod = [Y1 Y2 ...]

% Boundary Condition

Load=zeros(nNode,2); % Applied Force at every node.
Load= [F1x F1y; F2x F2y...]
Load(6,2)= -7*10^3;
Bc = ones(nNode,2)*nan; % Displacement Constraint at
each node. Bc = [u1, v1; u2 v2;...] . Write 'nan' if 'NO
Constraint'
Bc(1,:)= [0 0];
Bc(12,:)= [0 0];

%Deformation Scaled in plot

DefScale = 1000;

% Display Input Information

fprintf('Geometry :- \n\n');
fprintf('Total Number of Node : %d\n',nNode);
fprintf('Total Number of Element : %d\n',nElem);
fprintf('Degree of freedom per Node : %d\n', DofNode);
fprintf('Nodes Connect to each Element: \n');
disp(ElemNode);
fprintf('Co-ordinate of node : \n');
disp(NodeCoord);
fprintf('Area of Each Element : \n');
disp(Area);

fprintf('\n Material Property:-\n\n');
fprintf('Young's Modulus of each element : \n');
disp(Emod);

fprintf('\nBoundary Conditions:-\n\n');
fprintf('Applied Force at each node : \n');
disp(Load);
fprintf('Displacement Constraint at each node : \n');
disp(Bc);

%% Generation of System Stiffnes Matrix

nDof = nNode*DofNode; % Degree of freedom of the
system

```

```

    KSys = zeros(nDof); % Generating nDof by nDof
System Stiffness Matrix
for i = 1: nElem

    %Generating Local Stiffness Matrix of Element

    A = Area(i); % Area of i th Element
    E = Emod(i); % Young's Modulus of i th Element
    Node1 = ElemNode(i,1); % First node of i th Element
connect
    Node2 = ElemNode(i,2); % Second node of i th
Element Connect
    dx = NodeCoord(Node1,1)-NodeCoord(Node2,1); % x co-
ordinate of node1- x co-ordinate of node2
    dy = NodeCoord(Node1,2)- NodeCoord(Node2,2); % y
co-ordinate of node1 - y co-ordinate of node2
    L = sqrt(dx^2+dy^2); % Length of i th Element
    kLocalElem = (A*E)/L*[1 0 -1 0;0 0 0 0;-1 0 1 0;0 0
0 0]; %Local Stiffness Matrix of i th Element

    % Generating Global to Local Co-ordinate
Tranformation Matrix of i th Element

    C = dx/L; % cosine of angle
    S = dy/L; % sine of anle
    T = [C S 0 0;-S C 0 0;0 0 C S;0 0 -S C]; %
Tranformation Matrix

    % Global Stiffness Matrix for i th Element

    kGlobalElem = T'*kLocalElem*T;

    % Assemble of Global Stiffness Matrix of i th
Element in System
    % Siffness Marix

    u1 = 1 + (Node1-1)*2; v1 = u1+1; % Degree of freedom
of the node 1 related to i th element
    u2 = 1 + (Node2-1)*2; v2 = u2+1; % Degree of freedom
of the node 2 related to i th element

    KSys([u1 v1 u2 v2],[u1 v1 u2 v2]) = KSys([u1 v1 u2
v2],[u1 v1 u2 v2]) + kGlobalElem;
end
% Display System Stiffness Matrix

% fprintf('System Stiffness Matrix : \n');

```

```

% disp(KSys);

%% Applying Boundary Condition

Force = Load'; %Generating Column Vector
Force = Force(:);
Bc = Bc'; % Generating Column Vector
Bc =Bc(:);
for i = 1:nDof
    if ~isnan(Bc(i))
        Force = Force - KSys(:,i)*Bc(i);
        KSys(:,i)=0;
        KSys(i,:)=0;
        KSys(i,i) = 1;
        Force(i)=Bc(i);
    end
end
% fprintf('System Stiffness Matrix after applying BC''s
: \n');
% disp(KSys);
% fprintf('System Force Vector : \n');
% disp(Force);

%% Solution

uGlobal = KSys\Force;

fprintf('Displacement of Each Node :- \n\n');
disp(uGlobal);

%% Post Processing

Stress = zeros(nElem,1);
Reaction = zeros(nDof,1);
for i = 1: nElem

    %Generating Local Displacement Matrix of Element i

    A = Area(i); % Area of i th Element
    E = Emod(i); % Young's Modulus of i th Element
    Node1 = ElemNode(i,1); % First node of i th Element
connect
    Node2 = ElemNode(i,2); % Second node of i th
Element Connect

    % Generating Global to Local Co-ordinate
Tranformation Matrix of i th

```

```

% element

dx = NodeCoord(Node1,1)-NodeCoord(Node2,1); % x co-
ordinate of node1- x co-ordinate of node2
dy = NodeCoord(Node1,2)- NodeCoord(Node2,2); % y
co-ordinate of node1 - y co-ordinate of node2
L = sqrt(dx^2+dy^2); % Length of i th Element
C = dx/L; % cosine of angle
S = dy/L; % sine of anle
T = [C S 0 0;-S C 0 0;0 0 C S;0 0 -S C]; %
Tranformation Matrix

% Generating Global displacement Matrix for i th
Element

u1 = 1 + (Node1-1)*2; v1 = u1+1; % Degree of freedom
of the node 1 related to i th element
u2 = 1 + (Node2-1)*2; v2 = u2+1; % Degree of freedom
of the node 2 related to i th element
uGlobalElem = uGlobal([u1 v1 u2 v2]);

% Generating Local displacement Matrix for i th
Element

uLocalElem = T*uGlobalElem;
% Stress at i th Element
Stress(i) = (A*E)/L*[-1 0 1 0]*uLocalElem;

%Local force at i th Element

LocalForceElem = [-Stress(i);0;Stress(i);0];

% Transfer Local Force of the Element to Global
Force

GlobalForceElem = T'*LocalForceElem;

% Assemble Global force of i th Element to Reaction
(System Forces)

Reaction([u1 v1 u2 v2])=Reaction([u1 v1 u2 v2])+
GlobalForceElem;
end

%Display Results
fprintf('Stress of the element :- \n\n');
disp(Stress);

```

```

fprintf('Reaction at node points :- \n\n');
disp(Reaction);

%Scaling

uGlobalSc= uGlobal*DefScale;

% Deformed Node Co-ordinate
DefNodeCoord = zeros(nNode,2);
for i = 1:nNode
    u1 = 1+(i-1)*2;
    DefNodeCoord(i,:) =
NodeCoord(i,1:2)+(uGlobalSc(u1:u1+1));
end

% maximum stress
maxStress = max(abs(Stress));
I = maxStress == abs(Stress);
I = (1:nElem)'.*I;
% Ploting Structure
for i = 1:nElem
    Node1 = ElemNode(i,1);
    Node2 = ElemNode(i,2);
    %axis([-50 400 -50 300]);
    if sum((i==I)*1)
        plot([NodeCoord(Node1,1)
NodeCoord(Node2,1)], [NodeCoord(Node1,2)
NodeCoord(Node2,2)], 'r-', [DefNodeCoord(Node1,1)
DefNodeCoord(Node2,1)], [DefNodeCoord(Node1,2)
DefNodeCoord(Node2,2)], 'r--', 'LineWidth', 1);
    else
        plot([NodeCoord(Node1,1)
NodeCoord(Node2,1)], [NodeCoord(Node1,2)
NodeCoord(Node2,2)], 'k-', [DefNodeCoord(Node1,1)
DefNodeCoord(Node2,1)], [DefNodeCoord(Node1,2)
DefNodeCoord(Node2,2)], 'k--', 'LineWidth', 1);
    end
    hold on;
end
title('Structure');
xlabel('x (inch)');
ylabel('y (inch)');
legend('Undeformed',sprintf('Deformed (Scale =
%d)',DefScale));
hold off;

```