```
function [d_max] = project_3(ey)

v = 0.34;                               % Poisson's ratio %
L = 3.0;                                % Length of beam defined %
r = 0.2;                                % Hole Radius %
E = 69*1e9;                             % Modulus of Elasticity %

m = beam_mesh(ey, L, r);                % Creates a mesh with these parameters %
n = length(m.x);

qpts = [[-1 1 -1 1;-1 -1 1 1]/sqrt(3); 1 1 1 1];   % quadrature points %
K = zeros(2*n);
B = zeros(3,8);                                        % B matrix %
f = zeros(2*n,1);                                     % force matrix %

% ---- Either of the following conditions used during analysis ---- %
D = E / (1-v^2) .*[1 v 0
                   v 1 0
                   0 0 0.5*(1-v)];             % plane stress condition
D = E / ((1+v)*(1-2*v)) .*[1-v v   0
                          v   1-v 0
                          0   0   0.5*(1-2*v)];% plane strain condition

for c = m.conn
    xe = m.x(:,c);                          % Nodal displacements in element
    for q = qpts                            % Using two point Gauss Quadrature
        [N,dNdp] = shape_beam(q);           % Shape function
        J = xe * dNdp';                     % Jacobian
        dNdx = dNdp' / J;
        B(1,1:2:end) = dNdx(:,1);
        B(2,2:2:end) = dNdx(:,2);
        B(3,1:2:end) = dNdx(:,2);
        B(3,2:2:end) = dNdx(:,1);
        cc = [c(1)*2-1;c(1)*2;c(2)*2-1;c(2)*2;c(3)*2-1;c(3)*2;c(4)*2-1;c(4)*2];
        K(cc,cc) = K(cc,cc) + B' * D * B * q(3) * det(J);
    end
end

% ------  Natural Boudary Conditions  ------- %
nodes_R = find(m.x(1,:) == 1.5);              % find nodes at right %
conn_R = [nodes_R(1):ey:nodes_R(end-1);nodes_R(2):ey:nodes_R(end)];
for cc_R = conn_R
    yy_R = m.x(2,cc_R);
    le = abs(yy_R(2)-yy_R(1));
    for q = [-1 1]/sqrt(3)
        qq = 0.5 * [1-q;1+q];
        yq = yy_R * qq;
        f(2*cc_R-1) = f(2*cc_R-1) + qq * 200 * 1e6 * yq * le/2;
    end
end

% ------  Essential Boundary Conditions  -------  %
nodes_L = find(m.x(1,:) == -1.5);                 % find nodes at right %
cc = [nodes_L(:)*2-1, nodes_L(:)*2];
K(:,cc) = 0;
```

```
K(cc,:) = 0;

for i = 1:2*length(cc)
    K(cc(i),cc(i)) = 1;
end

f(cc) = 0;
d = K\f;                                     % Solving the equation %

d_x = d(1:2:end)';                           % X displacement %
d_y = d(2:2:end)';                           % Y displacement %
d_res = sqrt(d_x.^2 + d_y.^2);               % Maximum displacement %
d_max = max(d_res);                          % Maximum Displacement %

% Contour plot of approximated resultant displacement field
figure(1),patch('vertices',m.x','faces',m.conn',...
    'facecolor','interp','facevertexcdata',d_res');
title('Resultant displacement field')
colorbar;
axis equal;
hold on;
p = colorbar;
set(get(p,'title'),'string','Displacement');
xlabel('X (meters)');ylabel('Y (meters)');
figure(1)
quiver(m.x(1,:),m.x(2,:),d_x,d_y,'k');

% Contour plot of approximated displacement field along x direction
figure(2),patch('vertices',m.x','faces',m.conn','facecolor','interp',...
    'facevertexcdata', d_x');
title('Displacement Field along X axis')
colorbar;
axis equal;
hold on;
p = colorbar;
set(get(p,'title'),'string','Displacement');
xlabel('X (meters)');ylabel('Y (meters)');
figure(2)
quiver(m.x(1,:),m.x(2,:),d_x,zeros(1,length(d_x)), 'k');

% Contour plot of approximated displacement field along y direction
figure(3),patch('vertices', m.x','faces',m.conn','facecolor','interp',...
    'facevertexcdata', d_y');
title('Displacement Field along Y axis')
colorbar;
axis equal;
hold on;
p = colorbar;
set(get(p,'title'),'string','Displacement');
xlabel('X (meters)');ylabel('Y (meters)');
figure(3)
quiver(m.x(1,:),m.x(2,:),zeros(1,length(d_x)),d_y, 'k');
u_y = max(abs(d_y))

% Stress field calculation
```

```
strain = zeros(3,length(m.x));            % setting strain field to zero
stress = zeros(3,length(m.x));            % setting stress field to zero
for c = m.conn
    xe = m.x(:,c);                        % Nodal displacements in element
    cc = [c(1)*2-1;c(1)*2;c(2)*2-1;c(2)*2;c(3)*2-1;c(3)*2;c(4)*2-1;c(4)*2];
    de = d(cc);
    i = 1;
    for q = qpts                         % Using two point Gauss Quadrature
        [N,dNdp] = shape_beam(q);        % Shape function
        J = xe * dNdp';                  % Jacobian
        dNdx = dNdp' / J;
        B(1,1:2:end) = dNdx(:,1);
        B(2,2:2:end) = dNdx(:,2);
        B(3,1:2:end) = dNdx(:,2);
        B(3,2:2:end) = dNdx(:,1);
        strain(:,c(i)) = B * de;
        stress(:,c(i)) = D * strain(:,c(i));
        i = i + 1;
     end
end

%  --------  Plotting Sigma XX along AA' line -------  %
down_nodes = find((abs(m.x(1,:))<=1e-2 & m.x(2,:)<0));
top_nodes = find((abs(m.x(1,:))<=1e-2 & m.x(2,:)>0));

figure(5);
plot(m.x(2,down_nodes),stress(1,down_nodes),'ko',m.x(2,top_nodes),...
    stress(1,top_nodes),'ko',m.x(2,down_nodes),stress(1,down_nodes),'k',...
    m.x(2,top_nodes),stress(1,top_nodes),'k')
xlabel('Y (meters)');
ylabel('Sigma-XX (Pascals)');
title('Sigma XX along AA` line');

%  ----------  Contour Plot of Sigma XX  ------------  %
figure(4),patch('vertices',m.x','faces',m.conn','facecolor','interp',...
    'facevertexcdata', stress(1,:)');
title('Sigma-xx Field (Stress along X)')
colorbar;
axis equal;
hold on;
p = colorbar;
set(get(p,'title'),'string','Sigma XX');
xlabel('X (meters)');ylabel('Y (meters)');
max_stress = max(stress(1,:));
end

function [N,dNdp] = shape_beam(q)
N = 0.25*[(1-q(1))*(1-q(2)) (1+q(1))*(1-q(2)) (1+q(1))*(1+q(2)) .../
            (1-q(1))*(1+q(2))];
dNdp = 0.25*[(q(2)-1) (1-q(2)) (q(2)+1) -(q(2)+1);
         (q(1)-1) -(q(1)+1) (q(1)+1) (1-q(1))];
end
```