Project 2
Arjun Posarajah (1004881737)
December 15, 2023

UNIVERSITY OF TORONTO

Question i)

```
%Project 2: Finite Element Solution for a Curved Bracket
%Name: Arjun Posarajah Student#:1004881737

% Parameters/Properties
inner_radius = 35E-3;
outer_radius = 55E-3;
width = 20E-3;
thickness = 2E-3;
numtheta = 4;                    % Circumferential Number of Elements
numradial = 4;                   % Radial Number of Elements
elasticmodulus = 200e9;
poissonratio = 0.3;
appliedforce = 4e3; % in Newtons

% Generate coordinates for nodes
thetavals = linspace(-pi/2, pi/2, numtheta + 1);                % Angle range from -
pi/2 to pi/2
radialvals = flip(linspace(inner_radius, outer_radius, numradial + 1));     % Radial
values from inner to outer radius

% Create nodes
nodes = zeros((numtheta + 1) * (numradial + 1), 2);
nodeindex = 1;
for i = 1:(numradial + 1)
    for j = 1:(numtheta + 1)
        nodes(nodeindex, 1) = radialvals(i) * cos(thetavals(j));        % Convert
polar to Cartesian coordinates
        nodes(nodeindex, 2) = radialvals(i) * sin(thetavals(j));
        nodeindex = nodeindex + 1;
    end
end

% The position of each node in the mesh to draw isoparametric elements
elements = zeros(numtheta * numradial, 5);
element_index = 1;
for i = 1:numradial
    for j = 1:numtheta
        node1 = (i - 1) * (numtheta+1) + j;      %Example: When dividing by 10 radial
elements, 11 nodes will happen which is why we add by 1
        node2 = node1 + 1;
        node3 = node1 + numtheta + 2;
        node4 = node1 + numtheta + 1;
        elements(element_index, :) = [node1, node2, node3, node4, node1];       %
Store contour numbers element by element column wise.
        element_index = element_index + 1;
    end
end

% Plotting the mesh
```

```
figure;
hold on;
for i = 1:size(elements, 1)
    n = elements(i, :);
    x = nodes(n, 1);
    y = nodes(n, 2);
    plot(x, y, 'k'); % Plotting edges of each element
end
xlabel('X-axis');
ylabel('Y-axis');
title('Mesh of Semicircular Bracket');
axis equal;
hold off;

% Given geometry is in plane stress:

D = (elasticmodulus/1-((poissonratio)^2)) * [1, poissonratio, 0;poissonratio, 1, 0;0,
0, ((1-poissonratio)/2)];       % Pa

%Calculating global stiffness matrix
%Using 2-point Gauss quadrature to calculate each GN4Q value:

xi = [(1/sqrt(3)) (1/sqrt(3)) -(1/sqrt(3)) -(1/sqrt(3))];
n = [(1/sqrt(3)) -(1/sqrt(3)) (1/sqrt(3)) -(1/sqrt(3))];

GN4Q1 = (1/4) * [n(1)-1 1-n(1) 1+n(1) -n(1)-1; xi(1)-1 -xi(1)-1 1+xi(1) 1-xi(1)];
GN4Q2 = (1/4) * [n(2)-1 1-n(2) 1+n(2) -n(2)-1; xi(2)-1 -xi(2)-1 1+xi(2) 1-xi(2)];
GN4Q3 = (1/4) * [n(3)-1 1-n(3) 1+n(3) -n(3)-1; xi(3)-1 -xi(3)-1 1+xi(3) 1-xi(3)];
GN4Q4 = (1/4) * [n(4)-1 1-n(4) 1+n(4) -n(4)-1; xi(4)-1 -xi(4)-1 1+xi(4) 1-xi(4)];

%Allocating memory for the global stiffness matrix
TotalDOFs= (numtheta+1)*(numradial+1)*2;
Totalelements=numtheta*numradial;
k = sparse(TotalDOFs, TotalDOFs);
DOF=zeros(Totalelements,8);

XDOFs = (2 .* (elements(:,1:4))) - 1;
YDOFs = (2 .* (elements(:,1:4)));

for i = 1:Totalelements
        DOF(i,:) = [XDOFs(i,1); YDOFs(i,1); XDOFs(i,2); YDOFs(i,2); XDOFs(i,3);
YDOFs(i,3); XDOFs(i,4); YDOFs(i,4);];
end

%Single Loop iteration calculates individual element stiffness matrix and
%Pulls to the global stiffness matrix
xcoord= nodes(:,1);
ycoord= nodes(:,2);

for i = 1: Totalelements
    coord_ind = elements(i,:);
    Xval = xcoord(coord_ind);
    Yval = ycoord(coord_ind);
    coordmat = [Xval(1) Yval(1); Xval(2) Yval(2); Xval(3) Yval(3); Xval(4) Yval(4)];
    J1 = GN4Q1 * coordmat;
```

```
    J2 = GN4Q2 * coordmat;
    J3 = GN4Q3 * coordmat;
    J4 = GN4Q4 * coordmat;
    detJ1 = det(J1);
    invJ1 = inv(J1);
    detJ2 = det(J2);
    invJ2 = inv(J2);
    detJ3 = det(J3);
    invJ3 = inv(J3);
    detJ4 = det(J4);
    invJ4 = inv(J4);
    Hstar1 = invJ1\GN4Q1;
    Hstar2 = invJ2\GN4Q2;
    Hstar3 = invJ3\GN4Q3;
    Hstar4 = invJ4\GN4Q4;
    H1 = [Hstar1(1,1) 0 Hstar1(1,2) 0 Hstar1(1,3) 0 Hstar1(1,4) 0; 0 Hstar1(2,1) 0
Hstar1(2,2) 0 Hstar1(2,3) 0 Hstar1(2,4); Hstar1(2,1) Hstar1(1,1) Hstar1(2,2)
Hstar1(1,2) Hstar1(2,3) Hstar1(1,3) Hstar1(2,4) Hstar1(1,4)];
    H2 = [Hstar2(1,1) 0 Hstar2(1,2) 0 Hstar2(1,3) 0 Hstar2(1,4) 0; 0 Hstar2(2,1) 0
Hstar2(2,2) 0 Hstar2(2,3) 0 Hstar2(2,4); Hstar2(2,1) Hstar2(1,1) Hstar2(2,2)
Hstar2(1,2) Hstar2(2,3) Hstar2(1,3) Hstar2(2,4) Hstar2(1,4)];
    H3 = [Hstar3(1,1) 0 Hstar3(1,2) 0 Hstar3(1,3) 0 Hstar3(1,4) 0; 0 Hstar3(2,1) 0
Hstar3(2,2) 0 Hstar3(2,3) 0 Hstar3(2,4); Hstar3(2,1) Hstar3(1,1) Hstar3(2,2)
Hstar3(1,2) Hstar3(2,3) Hstar3(1,3) Hstar3(2,4) Hstar3(1,4)];
    H4 = [Hstar4(1,1) 0 Hstar4(1,2) 0 Hstar4(1,3) 0 Hstar4(1,4) 0; 0 Hstar4(2,1) 0
Hstar4(2,2) 0 Hstar4(2,3) 0 Hstar4(2,4); Hstar4(2,1) Hstar4(1,1) Hstar4(2,2)
Hstar4(1,2) Hstar4(2,3) Hstar4(1,3) Hstar4(2,4) Hstar4(1,4)];
    kstar = (detJ1.*(H1.')*D*H1) + (detJ2.*(H2.')*D*H2) + (detJ3.*(H3.')*D*H3) +
(detJ4.*(H4.')*D*H4);    % Pa
    elementDOF = DOF(i,:);
    k(elementDOF,elementDOF) = k(elementDOF,elementDOF) + kstar;      % Pa
end

%Fixed Points
dof_fixed= zeros(1,(numradial+1)*2)
n_fixed = 2*(numtheta+1);
i=1;
j=1;
while i <= n_fixed
    dof_fixed(i) = (j - 1) * 2 * (numtheta+1) + 1;
    dof_fixed(i+1) = dof_fixed(i) + 1;
    dof_fixed(i+2) = j * 2 * (numtheta+1) - 1;
    dof_fixed(i+3) = dof_fixed(i+2) + 1;
    j = j + 1;
    i = i+4;
end

dof_all=(1:(nodeindex-1)*2);
free_dof = setdiff(dof_all, dof_fixed);

%disp('Stiffness matrix for the given linear semi-circular beam is: \n');
%disp(full(k(free_dof, free_dof)));
K=(full(k(free_dof, free_dof)));
```

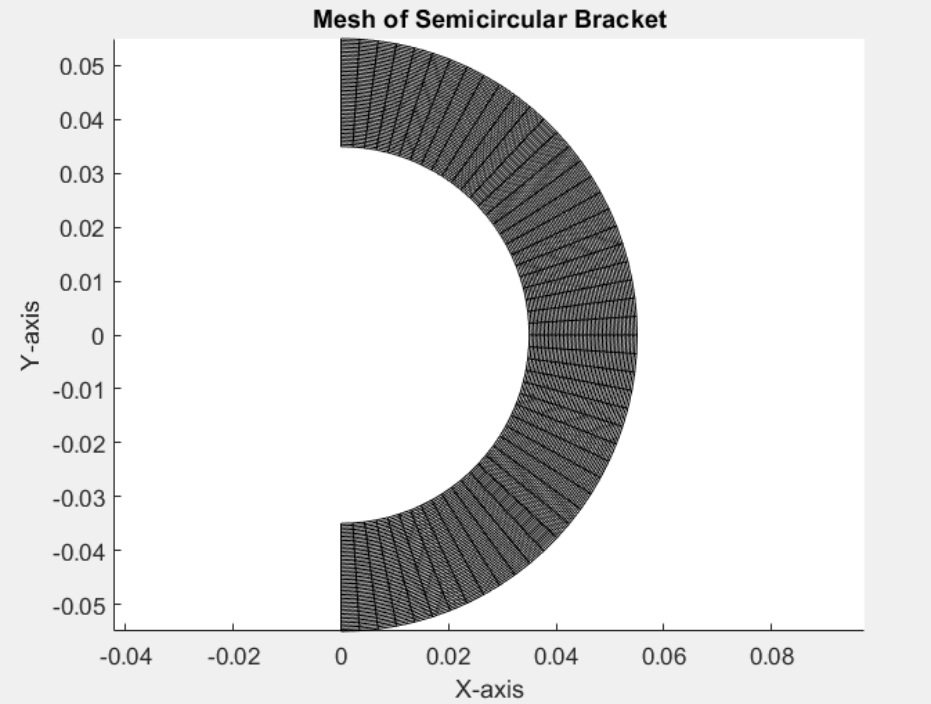
```
% force position finder function:
% Find the middle node of the middle thickness of the middle shape
fnode = (numradial - 1)*(numtheta + 1) + (numtheta/2);

forcedof=fnode*2

% Apply the force at the closest node
force_magnitude = -4000; % 4 kN
applied_force_vector = zeros(TotalDOFs, 1);
applied_force_vector(forcedof,1)=force_magnitude;

displacement=(1/2).* (K\applied_force_vector(free_dof));

[row_indices, col_indices] = find(displacement);
newnodes = zeros(size(nodes));
% Create nodes
nodeindex = 1;
for i = 1:(numradial + 1)
    for j = 1:(numtheta + 1)
        newnodes(nodeindex, 1) = radialvals(i) * cos(thetavals(j));        % Convert
polar to Cartesian coordinates
        newnodes(nodeindex, 2) = radialvals(i) * sin(thetavals(j));
        nodeindex = nodeindex + 1;
    end
end
for i = 1:length(col_indices)
    c = col_indices(i);
    r = row_indices(i);

    if rem(c, 2) == 1  % If c is odd, update x-coordinate
        cx = (c + 1) / 2;
        newnodes(cx, 1) = newnodes(cx, 1) + displacement(i);
    elseif rem(c, 2) == 0  % If c is even, update y-coordinate
        cy = c / 2;
        newnodes(cy, 2) = newnodes(cy, 2) + displacement(i);
    end
end

% The position of each node in the mesh to draw isoparametric elements
newelements = zeros(numtheta * numradial, 5);
element_index = 1;
for i = 1:numradial
    for j = 1:numtheta
        node1 = (i - 1) * (numtheta+1) + j;
        node2 = node1 + 1;
        node3 = node1 + numtheta + 2;
        node4 = node1 + numtheta + 1;
        newelements(element_index, :) = [node1, node2, node3, node4, node1];       %
Store contour numbers element by element column wise.
        element_index = element_index + 1;
    end
end
% Plotting the original mesh
figure;
hold on;
```

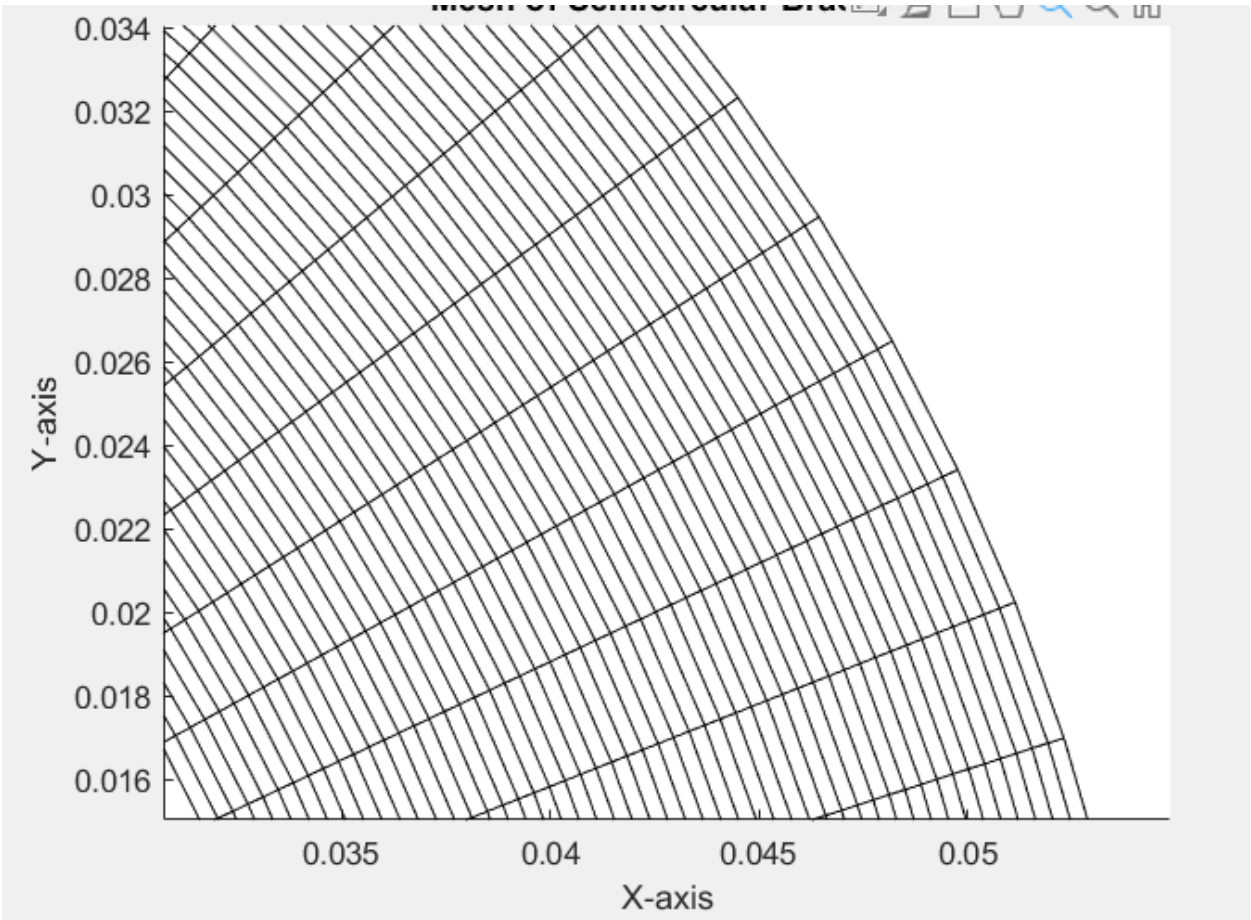
```
for i = 1:size(elements, 1)
    n = elements(i, :);
    x = nodes(n, 1);
    y = nodes(n, 2);
    plot(x, y, 'k'); % Plotting edges of each element
end

% Plotting the deformed shape
for i = 1:size(newelements, 1)
    n = newelements(i, :);
    x = newnodes(n, 1);
    y = newnodes(n, 2);
    plot(x, y, 'r'); % Plotting deformed shape
end

xlabel('X-axis');
ylabel('Y-axis');
title('Deformed Shape of Semicircular Bracket');
axis equal;
hold off;
```

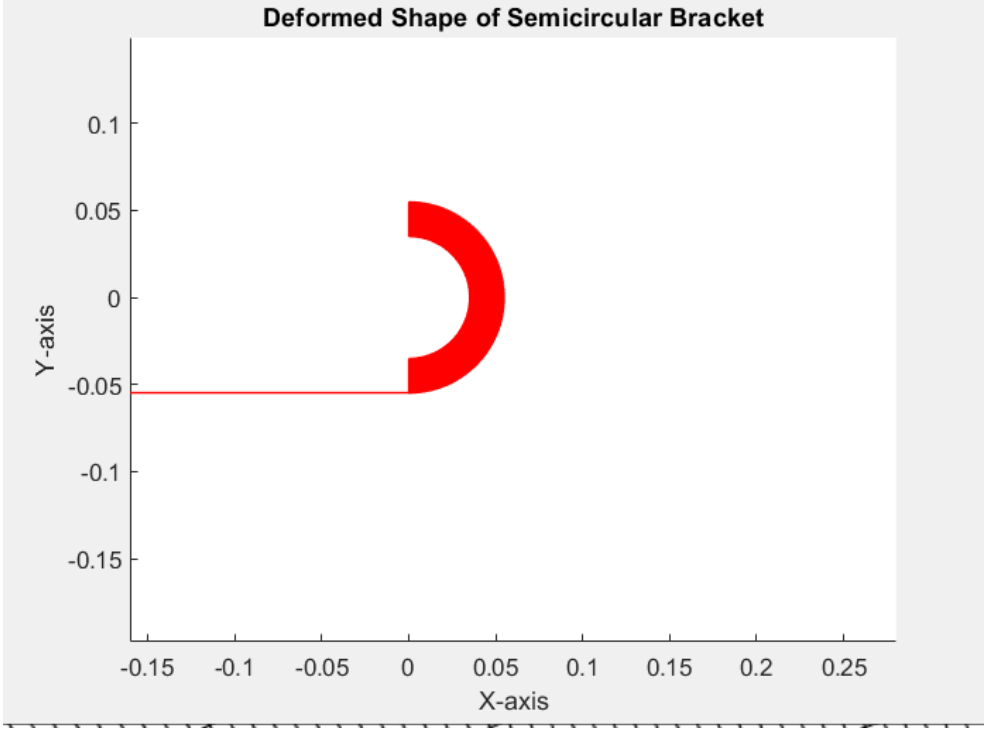
Question ii)

I have used more than enough elements and have not completed a convergence analysis. I have decided to use 50 radial and 50 theta values to create this finite element package analysis.

Question iii)




0.034
0.032
0.03
0.028
0.026
0.024
0.022
0.02
0.018
0.016
Y-axis
0.035
0.04
0.045
0.05
X-axis


Deformed Shape of Semicircular Bracket
0.1
0.05
0
-0.05
-0.1
-0.15
Y-axis
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2
0.25
X-axis

Question iv)

Question v)

Code was not working for this portion of the analysis.

The way it would be calculated though is by first finding H* and H for all the elements and once this is complete, I would take the transpose of H and multiply by the displacements found in the code provided above. Next would create a meshgrid around the specific shape of the bracket to see how the contour plot would occur.

Question vi)
I would then take the strain value and multiply by E to then get the stress components and ultimately do the same meshgrid like the previous section to plot the contours.

Question vii)

Firstly, I set up all the parameters. Next, I created the mesh of the shape, by first creating theta and radial values. Then created nodes through for loops that used the trig functions to get the curvature needed. After was to find the nodes and this was done through node2 being +1 and node 3 and node4 being the next row of values +2 and +1 respectively. Then I plotted these values. Next calculated the plane stress, started the 2 point gauss quadrature. I remembered to allocate space the entire time to speed the computing time. Just like in assignment 8 question 3, I found the J, detJ Hstar, and H for all the nodes and Kstar to start assembling the element. Then used the function provided in class (`k(elementDOF,elementDOF) = k(elementDOF,elementDOF) + kstar;`) to compile the k without first considering the fixed nodes. Next, I found an algorithm that finds the all the fixed nodes and the i+4 is there because it sets the while loop up for the next set of nodes. After I used another function provided in class to get the correct dofs considering the fixed points aswell using: `free_dof = setdiff(dof_all, dof_fixed);.` Next I found the node in which the force will be acting on, and considering each node has 2 dof, I multiplied it by 2 to find the dof in which the force is acting on. Next creating a force vector, with the compliant force and having that \ function to solve for the displacements. Next part of the code then goes the column and rows to to if the c which is the x coordinate is odd, and if it is you add it by 1 and divide by 2, then the input that into the new node plus the displacement and this process repeats for the y. The next portion of the code is similar to the original mesh graphing but with the deformed nodes overlapped above to get an understanding of how much displacement occurred.

Question viii)

I have provided the explanation in the question, due to my code not successfully running.

Question ix)
Due to not receiving a result it is hard to analyze, however from the general study it will be a suitable bracket for the considered loading conditions as I have attempted this in a different FEA package in CATIA and the von mises and strains.