Project II

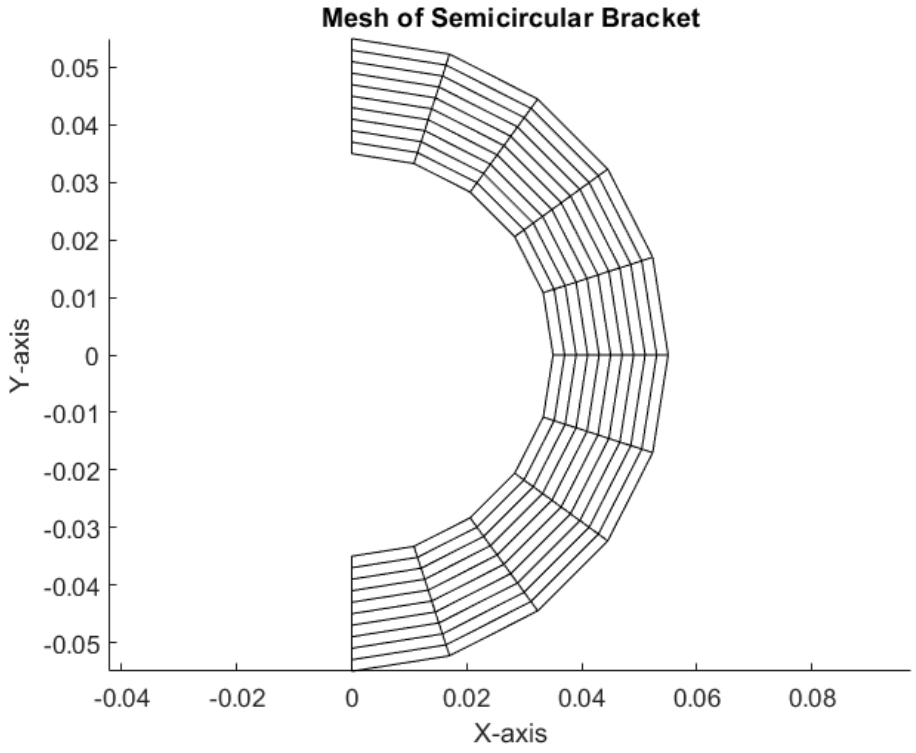
Muthu Ram Kumar Avichi
Student Number : 1010188967

Theory

Initially, I plotted the boundaries of the semi-circular bracket on matlab.

After that the approach was to divide this bracket into multiple divisions radially and angularly which would result in a structure that essentially divides the entire structure into iso-parametric quadrilateral elements.

The nodes are numbered from the bottom left most element in the outer radius to be 1, and moving forward to nodes 2, 3, and so on in the angular direction.



Then, after defining various parameters of the material, including its geometry and the material properties like E, v which are the following:

inner_radius = 0.035m
outer_radius = 0.055m

thickness = 0.002m

E = 200 GPa
v = 0.3

It is mentioned that the bracket is in plane stress, so we calculate the D matrix using the formula corresponding to plane stress:

D = (E/(1-v^2))*[1 v 0; v 1 0; 0 0 (1-v)/2];

To create the mesh for the structure I have defined elements from the nodal numbers based on logic loops. Each element array will consist of all the nodes pertaining to that element, and the plot function is written to depict the same.

After numbering is done, we assign degrees of freedom to each node as well. Each node has two degrees of freedom. Ex: Node 1 is assigned dof – 1,2; Node 2 is assigned dof – 3,4 and so on.

Sparse memory is allocated for the Global stiffness matrix as [2*DOF, 2*DOF], and the respective Ke values of DOF pairs summed into the global matrix.

To calculate the displacements of the isoparametric elements, the formula

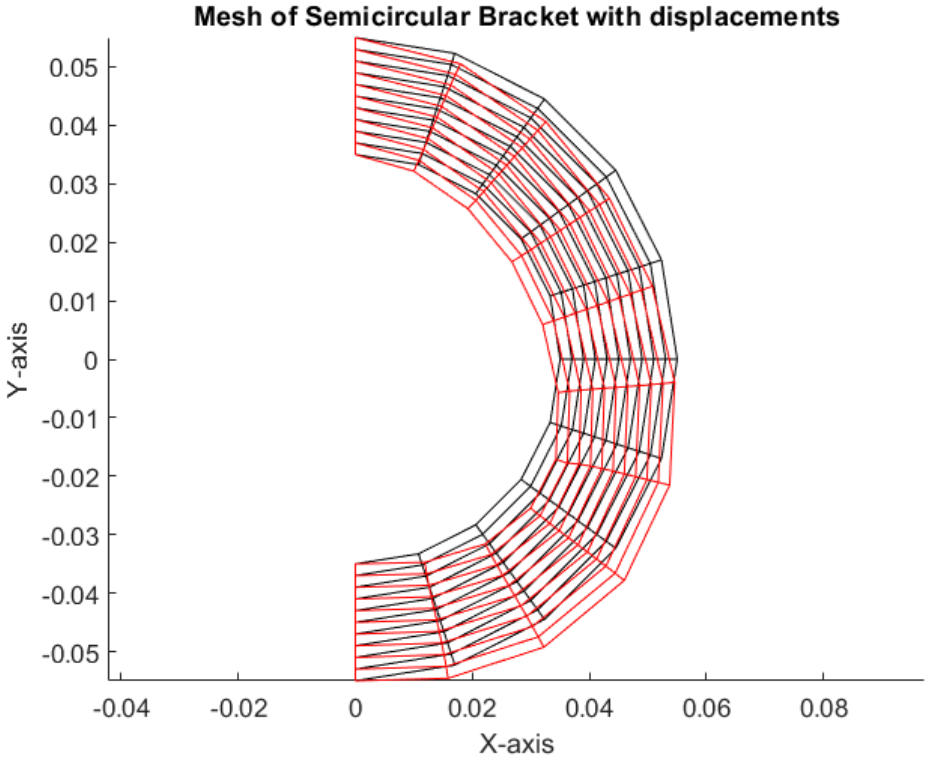
$$F = Kd$$

is used, where f is the force vector, K is the global stiffness matrix and d is the displacements. The force vector acting on the centre point of the struct was calculated from the logic underscoring the numbering of the nodes.

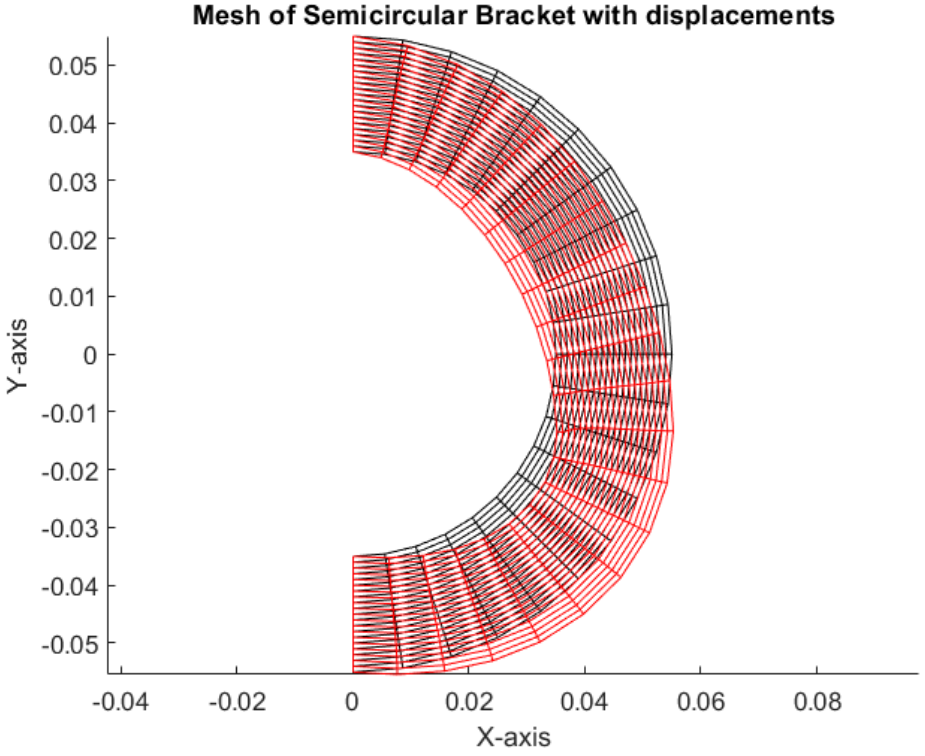
Performing an inverse operation on the computed K_Global in combination with this force vector yields the displacement vector, which is in a magnitude of 10^-4.

Displacements are plotted in a similar fashion atop the original element plot to yield the deformed shape plot.

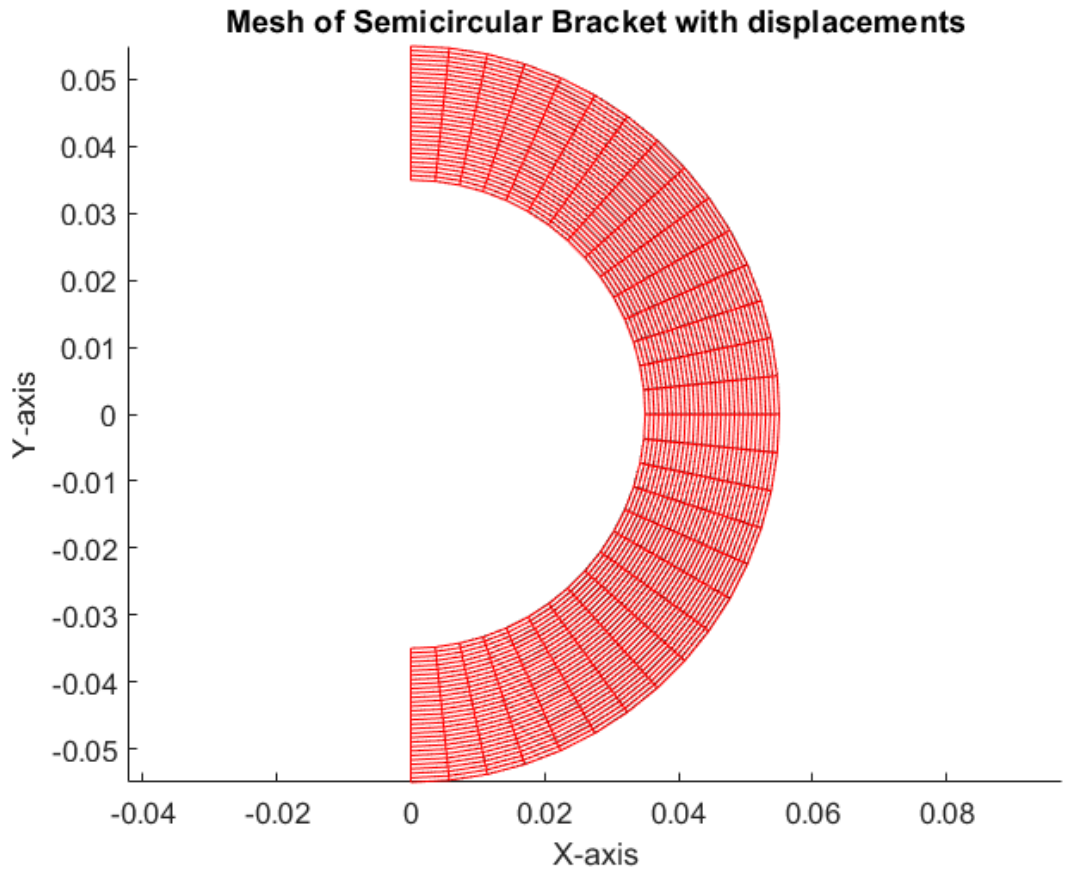
Convergence Analysis



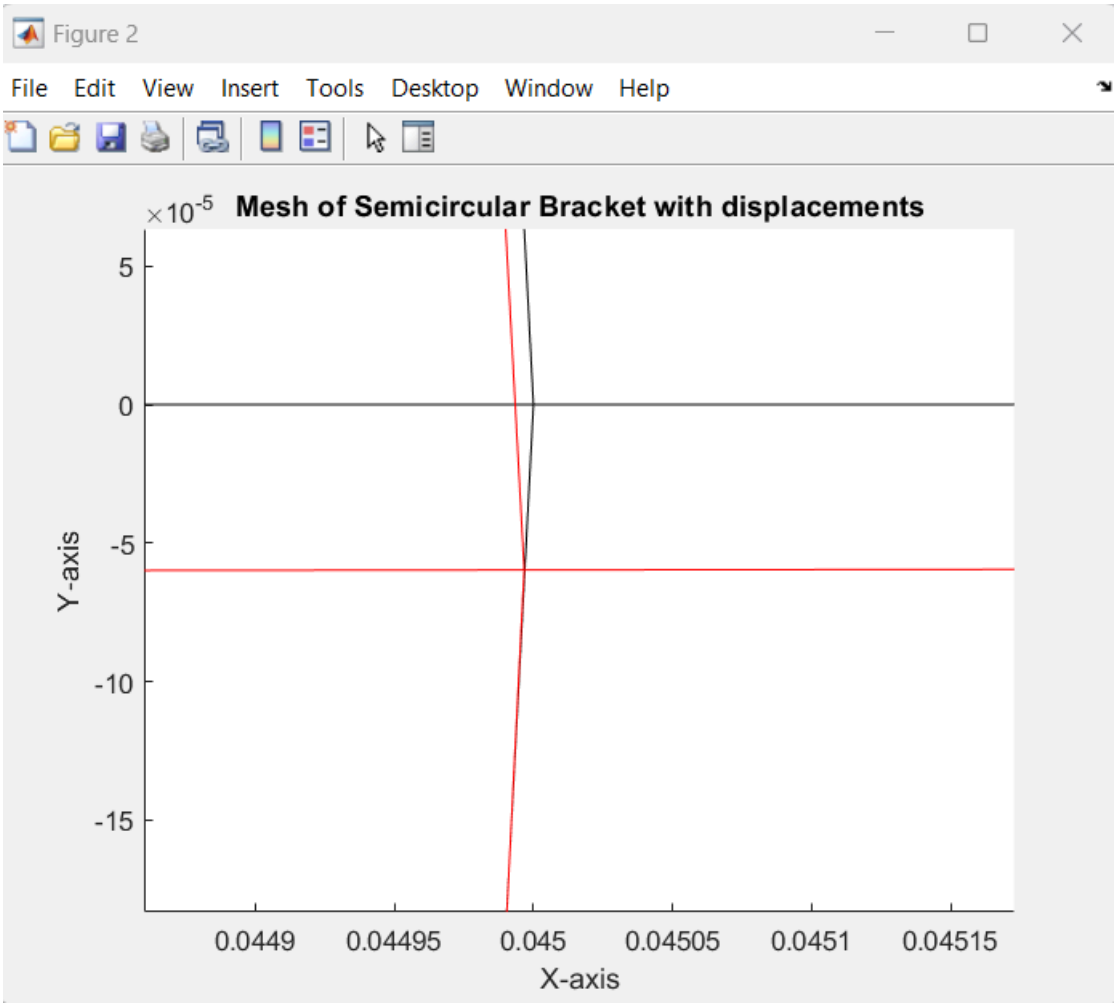
The first iteration performed with 100 elements and 10 divisions, both in the radial and angular directions. The deformation is magnified by a factor of 100 to show the deflection of the structure.
Increasing the iteration by a factor of 4 to 400 elements and 20 divisions in both directions.



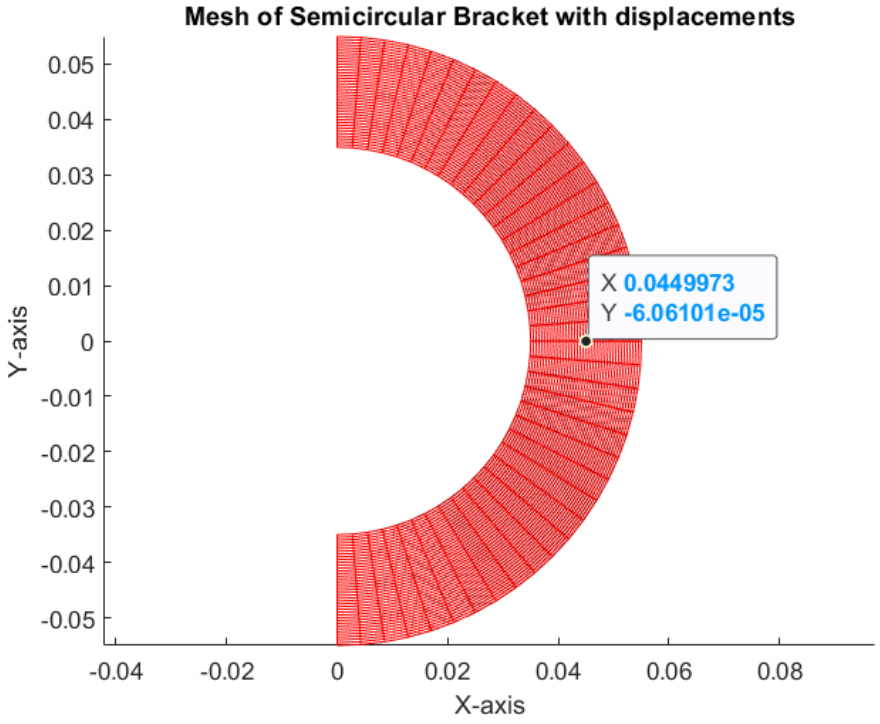
As depicted the convergence is clearer in this image, which is also magnified to a factor of 100. The deformation is closer to 0.05 which implies a 5*10^-4 at the horizontal line in which the force is acting.



In this iteration the factor is unapplied for authenticity of data collected. The deformation at the centre line is converging to 5*10^-5 m for this 900 element analysis as depicted below.



The convergence analysis is completed with 1600 elements, and a 5% error limit, with the deformation limited to 6.6 * 10^-5 at the centre line where the force of 4000 N is acting, as depicted.


Mesh of Semicircular Bracket with displacements
X 0.0449973
Y -6.06101e-05
Y-axis
X-axis
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.04
-0.02
0
0.02
0.04
0.06
0.08

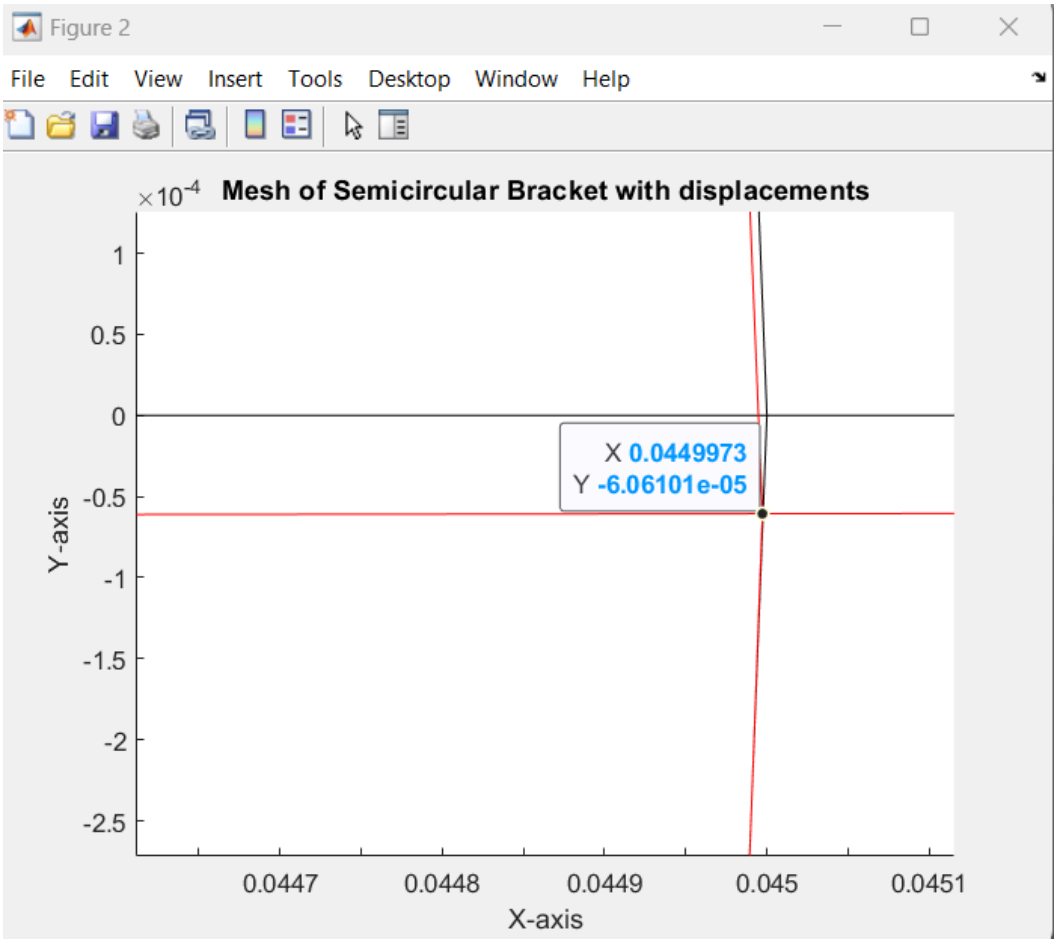


Figure 2
File Edit View Insert Tools Desktop Window Help
×10^-4
Mesh of Semicircular Bracket with displacements
X 0.0449973
Y -6.06101e-05
Y-axis
X-axis
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0.0447
0.0448
0.0449
0.045
0.0451

Stress-Strain Plots

To compute the three components of strain, the equation

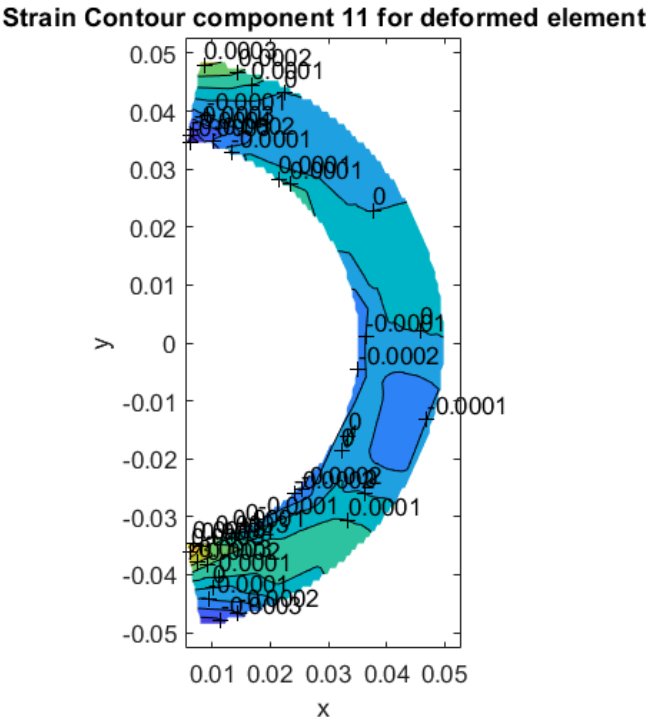
Strain = H*d

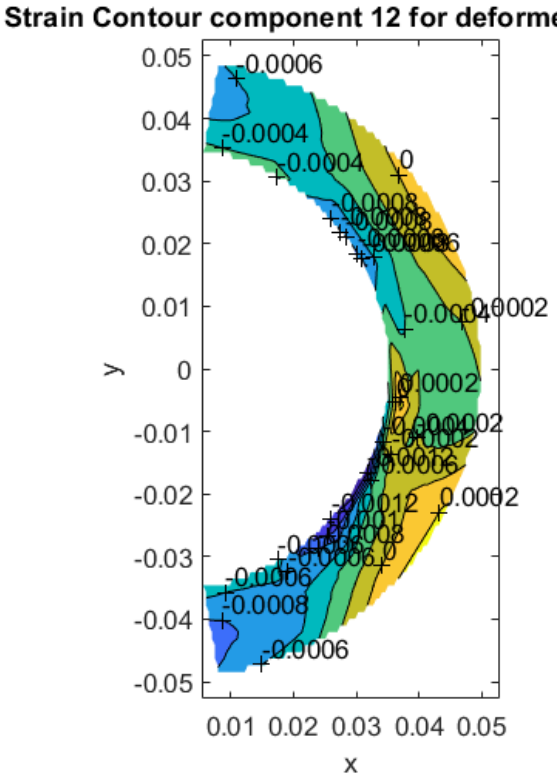
is used, where H is computed for the centre of each element (xsi=0, eta=0). This simplified the calculations and yielded significantly faster results.

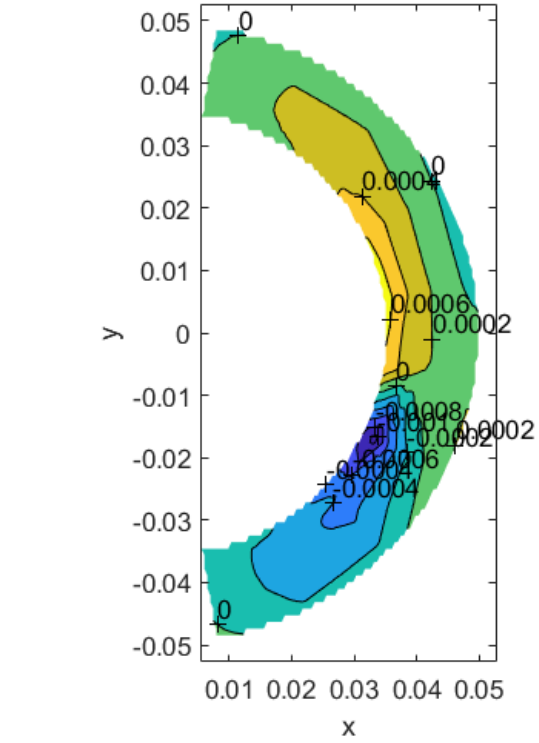
The three components of strain are defined as variables, and stored separately. However, the contour plotting is challenging as the corresponding (x,y) coordinates of the respective strain components are not linearly increasing or decreasing.

Hence, the corresponding x and y values are mapped to new X and Y coordinates that are linearly increasing based on the control factor of strains.

This yields the contour plots in the x-y directions for three strain components. Plots are masked with the semi-circular structure to increase clarity of depiction.



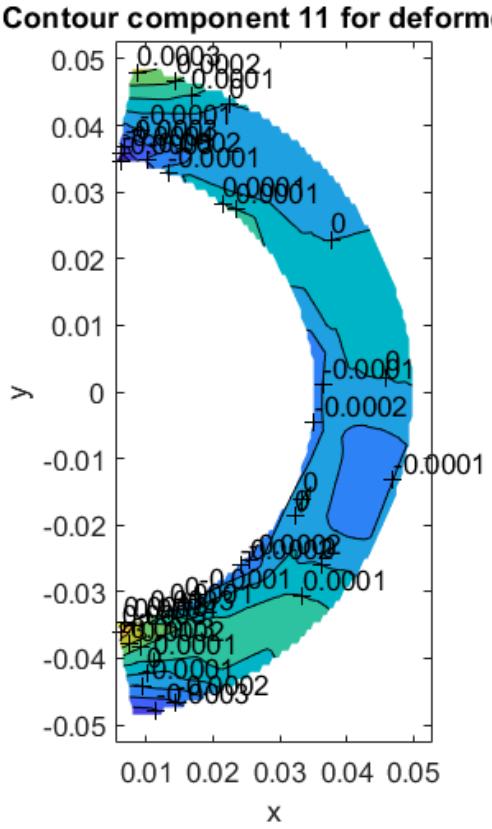




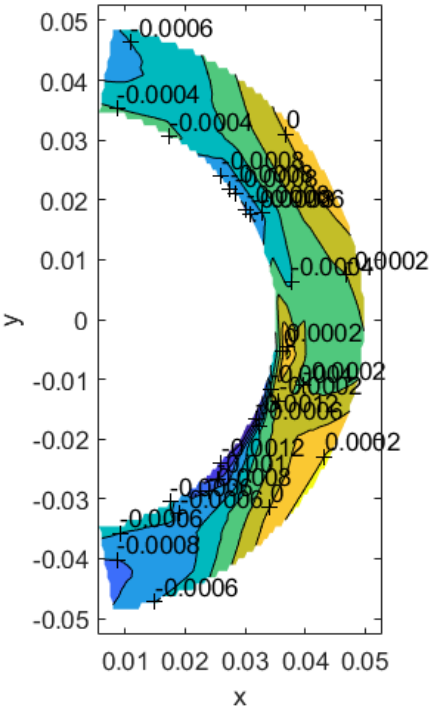
All x and y terms are in m.
To calculate stresses,

Stress = D * strain

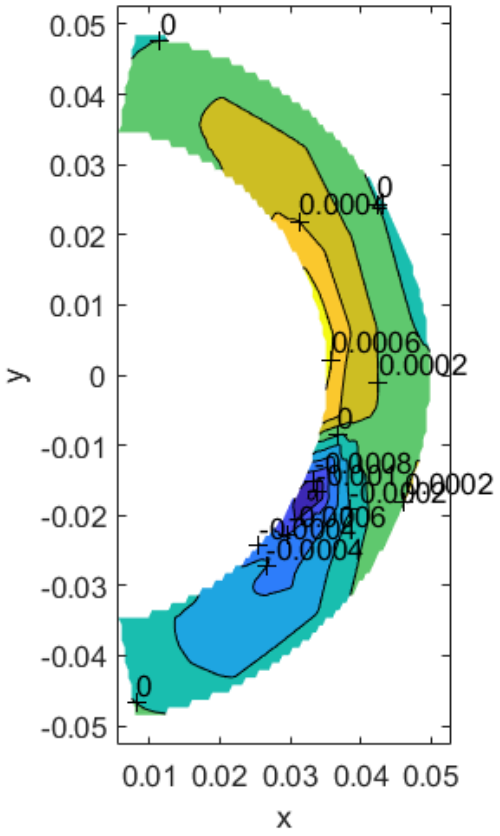
Where D is defined as mentioned above. This is plotted as contours in a similar fashion.



Stress Contour component 12 for deformed element



Stress Contour component 22 for deformed element



Errors

- The plots depict the respective contours for the directional strains and stresses. As is visible from the plots, the retrieved data is not continuous near the boundary, which signifies that the overall data depicted will have interpolation errors.

- The data is also extrapolated beyond the specified range, which can increase complexity in analysis and usefulness of the data.

Conclusion

The finite element method truly unlocks the capabilities that complement structural analysis. The computational efficiency and power that a few lines of code has on a real live aircraft is exciting to consider. Small changes in the code, the force vectors, displacements and the number of elements has a massive impact on the generated results, and in turn the overall effectiveness of the analysis. Hand calculations and derivations have depicted the similarity that the FEM has to strong form analysis, when the solution converges.

Contour plots, graphical representations and introduced mathematical complexities to reduce computation provide comprehensive tools that uplift the designing process. The manufactured structures will be better equipped to handle failure due to stresses and strains in a more comprehensive and detail-oriented manner.

However, as the number of elements increases, the time taken to run the analysis increases drastically, which hinders the analysis process. Hence, there is scope to explore the mathematical functions and tools that can reduce computation time and consequently increase analysis efficiency.

Matlab Code

```
% ------------------------------------------------------------------------
% Matlab code for Project 2 - Meshing
% ------------------------------------------------------------------------
% Parameter of the given Semi-Circular Bracket.
radius_inner = 35E-3;
radius_outer = 55E-3;
beam_width = 20E-3;
n_theta = 40;                    % Circumferential Number of Elements
n_radial = 40;                   % Radial Number of Elements
% Generate coordinates for nodes
theta = linspace(-pi/2, pi/2, n_theta + 1);                 % Angle range
from -pi/2 to pi/2
r = flip(linspace(radius_inner, radius_outer, n_radial + 1));     % Radial
values from inner to outer radius
% Create nodes
nodes = zeros((n_theta + 1) * (n_radial + 1), 2);
n = 1;
for i = 1:(n_radial + 1)
   for j = 1:(n_theta + 1)
       nodes(n, 1) = r(i) * cos(theta(j));        % Convert polar to
cartesian co-ordinates
       nodes(n, 2) = r(i) * sin(theta(j));
       n = n + 1;
   end
end
% The position of each node in the mesh to draw isoparametric elements
elements = zeros(n_theta * n_radial, 5);
element_index = 1;
for i = 1:n_radial
   for j = 1:n_theta
       node1 = (i - 1) * (n_theta+1) + j;
       node2 = node1 + 1;                              % Value 1 added to
elements since splitting an element
       node3 = node1 + n_theta + 2;                    % by n_radial will
create n_radial + 1 nodes.
       node4 = node1 + n_theta + 1;
       elements(element_index, :) = [node1, node2, node3, node4, node1];
% Store contour numbers element by element column wise.
       element_index = element_index + 1;
   end
end
% ------------------------------------------------------------------------
% Matlab code for Plotting the Mesh
% ------------------------------------------------------------------------
figure(1);
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;
% -------------------------------------------------------------------------
% Matlab code for Calculating the Stiffness Matrices.
% -------------------------------------------------------------------------
% Matlab Code for defining the global stiffness matrices
n_fixed = 2*(n_radial+1);                                   % Number of fixed
nodes calculation
n_global = 2*( (n_radial+1)*(n_theta+1));                   % Total Number of
degrees of freedom
Kglobal = spalloc(n_global, n_global, n_global);            % Global matrix
memory allocation
% For loop to calculate each stiffness matrix value
syms xsi eta;
for i = 1:size(elements, 1)
    n = elements(i, :);
    x = nodes(n, 1);
    y = nodes(n, 2);

    % Initialising the xsi and eta matrices

    xsip = [-1/sqrt(3) , -1/sqrt(3) , 1/sqrt(3) , 1/sqrt(3)];
    etap = [-1/sqrt(3) , 1/sqrt(3) , -1/sqrt(3) , 1/sqrt(3)];

    % Material Properties of the bracket.

    E = 200e9;
    nu = 0.3;

    % D matrix from equation for plane stress condition

    D = [1 nu 0; nu 1 0; 0 0 ((1-nu)/2)];
    D = (E/(1-nu^2))*D;

    % Nf matrix initialised from lecture equations

    Nf = cell(4);

    Nf{1} = (1/4)*(1 - xsi)*(1 - eta);
    Nf{2} = (1/4)*(1 + xsi)*(1 - eta);
    Nf{3} = (1/4)*(1 + xsi)*(1 + eta);
    Nf{4} = (1/4)*(1 - xsi)*(1 + eta);

    % Iteration to initialise Nf4Q matrix -
    % differentiated wrt xsi, eta &
    % substituted for each co-ordinate in the graph
```

```
    Ke = zeros(8, 8);

    for i = 1:4
        Nfx = zeros(4);
        Nfy = zeros(4);

        for ck = 1:4
            Nfx(ck) = subs(diff(Nf{ck}, xsi), eta, etap(i));
            Nfy(ck) = subs(diff(Nf{ck}, eta), xsi, xsip(i));
        end

        Nfq = double([Nfx(1) Nfx(2) Nfx(3) Nfx(4); Nfy(1) Nfy(2) Nfy(3)
Nfy(4)]);

        xy = [x(1) y(1); x(2) y(2); x(3) y(3); x(4) y(4)];

        J = Nfq*xy;

        Jinv = inv(J);

        % H star matrix from the lecture slides.
        Hs = Jinv*Nfq;

        H = [Hs(1, 1) 0 Hs(1, 2) 0 Hs(1, 3) 0 Hs(1, 4) 0; 0 Hs(2, 1) 0 Hs(2,
2) 0 Hs(2, 3) 0 Hs(2, 4); Hs(2, 1) Hs(1, 1) Hs(2, 2) Hs(1, 2) Hs(2, 3) Hs(1,
3) Hs(2, 4) Hs(1, 4);];

        detJ = det(J);

        Ke = Ke + detJ*H'*D*H;
    end

    % Matrix to store the Degree of Freedom values for corresponding nodal
positions that the stiffness matrix is calculated for.
    edof = [(2*n(1)-1), 2*n(1), (2*n(2)-1), 2*n(2), (2*n(3)-1), 2*n(3),
(2*n(4)-1), 2*n(4)];

    % Global Matrix calculation
    Kglobal(edof, edof) = Kglobal(edof, edof) + Ke;
end
% Code block to display the stiffness matrix
% Declare all the degree of freedom
dof_all = 1:1:n_global;
dof_fixed = zeros(1, 2*n_fixed);
% Fixed nodes and corresponding degrees of freedom
in = 1;
jn = 1;
while in <= 2*n_fixed
    dof_fixed(in) = (jn - 1) * 2 * (n_theta+1) + 1;
    dof_fixed(in+1) = dof_fixed(in) + 1;
    dof_fixed(in+2) = jn * 2 * (n_theta+1) - 1;
    dof_fixed(in+3) = dof_fixed(in+2) + 1;
    jn = jn + 1;
```

```
   in = in+4;
end
% Calculate nodes with degrees of freedom
free_dof = setdiff(dof_all, dof_fixed);
disp('Stiffness matrix for the given linear semi-circular beam is: \n');
disp(full(Kglobal(free_dof, free_dof)));
% ------------------------------------------------------------------------
% Matlab code for Calculating the Stiffness Matrices.
% ------------------------------------------------------------------------
% Given thickness of the semi-circular structure.
thickness = 2e-3;
% Code to find the node that the given force is acting on
f_node = (n_radial - 1)*(n_theta + 1) + (n_theta/2);
ld = length(dof_fixed);
% Force vector initialisation
le = length(free_dof);
force = zeros(1, le+ld);
% Given Applied load at center of semi-circular structure.
applied_load = -4e3; % in Newtons
force(1, f_node*2) = applied_load;
% Code block to calculate the displacement vector.
d = spalloc(1, le, 1);
d(free_dof) = (1/thickness).*((Kglobal(free_dof,
free_dof)\(force(free_dof)')));
% ------------------------------------------------------------------------
% Matlab code for Post-Processing the displacement vector points.
% ------------------------------------------------------------------------
% Resolving the displacements into x and y components
% Create nodes
newnodes = zeros((n_theta + 1) * (n_radial + 1), 2);
n = 1;                                  % Counter for incrementing the nodal
position variable
for i = 1:(n_radial + 1)
   for j = 1:(n_theta + 1)
       newnodes(n, 1) = r(i) * cos(theta(j));        % Convert polar to
cartesian co-ordinates
       newnodes(n, 2) = r(i) * sin(theta(j));
       n = n + 1;
   end
end
mult = 1;                               % Magnifying displacements to make sure
they are visbible in the plot.
[row_indices, col_indices] = find(d);       % Row and Column Indices of
displacement vector
ncf = length(col_indices);
for i = 1:ncf
   c = col_indices(i);
   if rem(c, 2) == 1                        % If c is odd
       cx =(c+1)/2;
       newnodes(cx, 1) = newnodes(cx, 1) + d(1, c)*mult;
   elseif rem(c, 2)== 0                     % If c is even
       cy = c/2;
       newnodes(cy, 2) = newnodes(cy, 2) + d(1, c)*mult;
```

```
   end
end
% The position of each node in the mesh to draw isoparametric elements
newelements = zeros(n_theta * n_radial, 5);
newelement_index = 1;
for i = 1:n_radial
   for j = 1:n_theta
       node1 = (i - 1) * (n_theta+1) + j;
       node2 = node1 + 1;                              % Value 1 added to
elements since splitting an element
       node3 = node1 + n_theta + 2;                    % by n_radial will
create n_radial + 1 nodes.
       node4 = node1 + n_theta + 1;
       newelements(newelement_index, :) = [node1, node2, node3, node4,
node1];       % Store contour numbers element by element column wise.
       newelement_index = newelement_index + 1;
   end
end
% ---------------------------------------------------------------------
% Matlab code for Plotting the Mesh with deformed shape
% ---------------------------------------------------------------------
figure(2);
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
for i = 1:size(newelements, 1)
   n = newelements(i, :);
   x = newnodes(n, 1);
   y = newnodes(n, 2);
   plot(x, y, 'r'); % Plotting edges of each element
end
xlabel('X-axis');
ylabel('Y-axis');
title('Mesh of Semicircular Bracket with displacements');
axis equal;
hold off;
% ---------------------------------------------------------------------
% Matlab code for finding H - Matrix at the center of each node, and
% subsequently get the Strain matrix = H*d
% ---------------------------------------------------------------------
dnew = zeros(1, n_global);
dnew(row_indices, col_indices) = d(row_indices, col_indices);
strain11g = zeros(1, size(elements, 1));
strain22g = zeros(1, size(elements, 1));
strain12g = zeros(1, size(elements, 1));
stress11g = zeros(1, size(elements, 1));
stress22g = zeros(1, size(elements, 1));
stress12g = zeros(1, size(elements, 1));
xnew = zeros(1, size(elements, 1));
```

```
ynew = zeros(1, size(elements, 1));
for i = 1:size(elements, 1)

   % Material Properties of the bracket.

   E = 200e9;
   nu = 0.3;

   % D matrix from equation for plane stress condition

   D = [1 nu 0; nu 1 0; 0 0 ((1-nu)/2)];
   D = (E/(1-nu^2))*D;
   n = elements(i, :);
   x = nodes(n, 1);
   y = nodes(n, 2);

   % Initialising the xsi and eta values to zero, to find the
   % corresponding values at center of isoparametric element.

   xsip = 0;
   etap = 0;

    % Iteration to initialise Nf4Q matrix -
   % differentiated wrt xsi, eta &
   % substituted for each co-ordinate in the graph

   Nfq = 0.25.*[-1 1 1 -1; -1 -1 1 1];
   xy = [x(1) y(1); x(2) y(2); x(3) y(3); x(4) y(4)];
   J = Nfq*xy;

   Jinv = inv(J);
   % H star matrix from the lecture slides.
   Hs = Jinv*Nfq;

   H = [Hs(1, 1) 0 Hs(1, 2) 0 Hs(1, 3) 0 Hs(1, 4) 0; 0 Hs(2, 1) 0 Hs(2, 2) 0
Hs(2, 3) 0 Hs(2, 4); Hs(2, 1) Hs(1, 1) Hs(2, 2) Hs(1, 2) Hs(2, 3) Hs(1, 3)
Hs(2, 4) Hs(1, 4);];

   dindex = zeros(1, 8);
   j = 1;
   for k = 1:4
       dindex(j) = n(k)*2 - 1;
       dindex(j+1) = n(k)*2;
       j = j+2;
   end

   dvalues = dnew(1, dindex);
   strain = H*dvalues';
   stress = D*strain;

   strain11g(i) = strain(1);
   strain22g(i) = strain(2);
   strain12g(i) = strain(3);
```

```
    stress11g(i) = strain(1);
    stress22g(i) = strain(2);
    stress12g(i) = strain(3);
    xnew(i) = (x(1) + x(2) + x(3) + x(4)) / 4;
    ynew(i) = (y(1) + y(2) + y(3) + y(4)) / 4;
end
% ------------------------------------------------------------------------
% Plot - Contours for strains with respect to position.
% ------------------------------------------------------------------------
% Define angles for the circles
theta_outer = linspace(-pi/2, pi/2, 100);
theta_inner = linspace(-pi/2, pi/2, 100);
% Outer circle parameters
outer_radius = 0.055;
x_outer = outer_radius * cos(theta_outer);
y_outer = outer_radius * sin(theta_outer);
% Inner circle parameters
inner_radius = 0.035;
x_inner = inner_radius * cos(theta_inner);
y_inner = inner_radius * sin(theta_inner);
x_overall = [x_outer, flip(x_inner), x_outer(1)];
y_overall = [y_outer, flip(y_inner), y_outer(1)];
% Interpolating onto a regular grid
x_regular = linspace(min(xnew), max(xnew), 100); % Regular X grid
y_regular = linspace(min(ynew), max(ynew), 100); % Regular Y grid
[X_regular, Y_regular] = meshgrid(x_regular, y_regular);
% Create a mask for contour within the polygon
in_polygon = inpolygon(X_regular, Y_regular, x_overall, y_overall);
% ------------------------------------------------------------------------
% Plot contours for strain component 11
% ------------------------------------------------------------------------
% Interpolating Z values onto the regular grid
Z_regular = griddata(xnew, ynew, strain11g, X_regular, Y_regular);
% Plot - Contours for strains with respect to position.
figure(3);
% Plot the outer circle
plot(x_overall, y_overall, 'k', 'LineWidth', 2);
% Create a mask for contour outside the polygon
outside_polygon = ~inpolygon(X_regular, Y_regular, x_overall, y_overall);
% Set values outside the polygon to NaN in the Z matrix
Z_regular(outside_polygon) = NaN;
% Plot the masked contourf
clabel(contourf(X_regular, Y_regular, Z_regular));
% Set axis equal to maintain circular proportions
axis equal;
hold on;
xlabel('x');
ylabel('y');
title('Strain Contour component 11 for deformed element');
% ------------------------------------------------------------------------
% Plot contours for strain component 22
% ------------------------------------------------------------------------
% Interpolating Z values onto the regular grid
```

```
Z_regular = griddata(xnew, ynew, strain22g, X_regular, Y_regular);
% Plot - Contours for strains with respect to position.
figure(4);
% Plot the outer circle
plot(x_overall, y_overall, 'k', 'LineWidth', 2);
% Create a mask for contour outside the polygon
outside_polygon = ~inpolygon(X_regular, Y_regular, x_overall, y_overall);
% Set values outside the polygon to NaN in the Z matrix
Z_regular(outside_polygon) = NaN;
% Plot the masked contourf
clabel(contourf(X_regular, Y_regular, Z_regular));
% Set axis equal to maintain circular proportions
axis equal;
hold on;
xlabel('x');
ylabel('y');
title('Strain Contour component 22 for deformed element');
% ------------------------------------------------------------------------
% Plot contours for strain component 12
% ------------------------------------------------------------------------
% Interpolating Z values onto the regular grid
Z_regular = griddata(xnew, ynew, strain12g, X_regular, Y_regular);
% Plot - Contours for strains with respect to position.
figure(5);
% Plot the outer circle
plot(x_overall, y_overall, 'k', 'LineWidth', 2);
% Create a mask for contour outside the polygon
outside_polygon = ~inpolygon(X_regular, Y_regular, x_overall, y_overall);
% Set values outside the polygon to NaN in the Z matrix
Z_regular(outside_polygon) = NaN;
% Plot the masked contourf
clabel(contourf(X_regular, Y_regular, Z_regular));
% Set axis equal to maintain circular proportions
axis equal;
hold on;
xlabel('x');
ylabel('y');
title('Strain Contour component 12 for deformed element');
% ------------------------------------------------------------------------
% Plot for stresses in a similar fashion
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% Plot contours for stress component 11
% ------------------------------------------------------------------------
% Interpolating Z values onto the regular grid
Z_regular = griddata(xnew, ynew, stress11g, X_regular, Y_regular);
% Plot - Contours for strains with respect to position.
figure(6);
% Plot the outer circle
plot(x_overall, y_overall, 'k', 'LineWidth', 2);
% Create a mask for contour outside the polygon
outside_polygon = ~inpolygon(X_regular, Y_regular, x_overall, y_overall);
% Set values outside the polygon to NaN in the Z matrix
```

```
Z_regular(outside_polygon) = NaN;
% Plot the masked contourf
clabel(contourf(X_regular, Y_regular, Z_regular));
% Set axis equal to maintain circular proportions
axis equal;
hold on;
xlabel('x');
ylabel('y');
title('Stress Contour component 11 for deformed element');
% ------------------------------------------------------------------------
% Plot contours for stress component 22
% ------------------------------------------------------------------------
% Interpolating Z values onto the regular grid
Z_regular = griddata(xnew, ynew, stress22g, X_regular, Y_regular);
% Plot - Contours for strains with respect to position.
figure(7);
% Plot the outer circle
plot(x_overall, y_overall, 'k', 'LineWidth', 2);
% Create a mask for contour outside the polygon
outside_polygon = ~inpolygon(X_regular, Y_regular, x_overall, y_overall);
% Set values outside the polygon to NaN in the Z matrix
Z_regular(outside_polygon) = NaN;
% Plot the masked contourf
clabel(contourf(X_regular, Y_regular, Z_regular));
% Set axis equal to maintain circular proportions
axis equal;
hold on;
xlabel('x');
ylabel('y');
title('Stress Contour component 22 for deformed element');
% ------------------------------------------------------------------------
% Plot contours for strain component 12
% ------------------------------------------------------------------------
% Interpolating Z values onto the regular grid
Z_regular = griddata(xnew, ynew, stress12g, X_regular, Y_regular);
% Plot - Contours for strains with respect to position.
figure(8);
% Plot the outer circle
plot(x_overall, y_overall, 'k', 'LineWidth', 2);
% Create a mask for contour outside the polygon
outside_polygon = ~inpolygon(X_regular, Y_regular, x_overall, y_overall);
% Set values outside the polygon to NaN in the Z matrix
Z_regular(outside_polygon) = NaN;
% Plot the masked contourf
clabel(contourf(X_regular, Y_regular, Z_regular));
% Set axis equal to maintain circular proportions
axis equal;
hold on;
xlabel('x');
ylabel('y');
title('Stress Contour component 12 for deformed element');
```