MATRIX STRUCTURAL ANALYSIS

GROUP 11

PRESENTED BY:

AVINASH PANDEY
22CE10013

SUMEDH DONGRAJKAR
22CE10023

SUCHISMITA SAHOO
22CE10079

YEDLA GOWTHAMI
22CE10088

DHRUV RAJ SINGH RATHORE
22CE30009

CONTENT

MOTIVATION

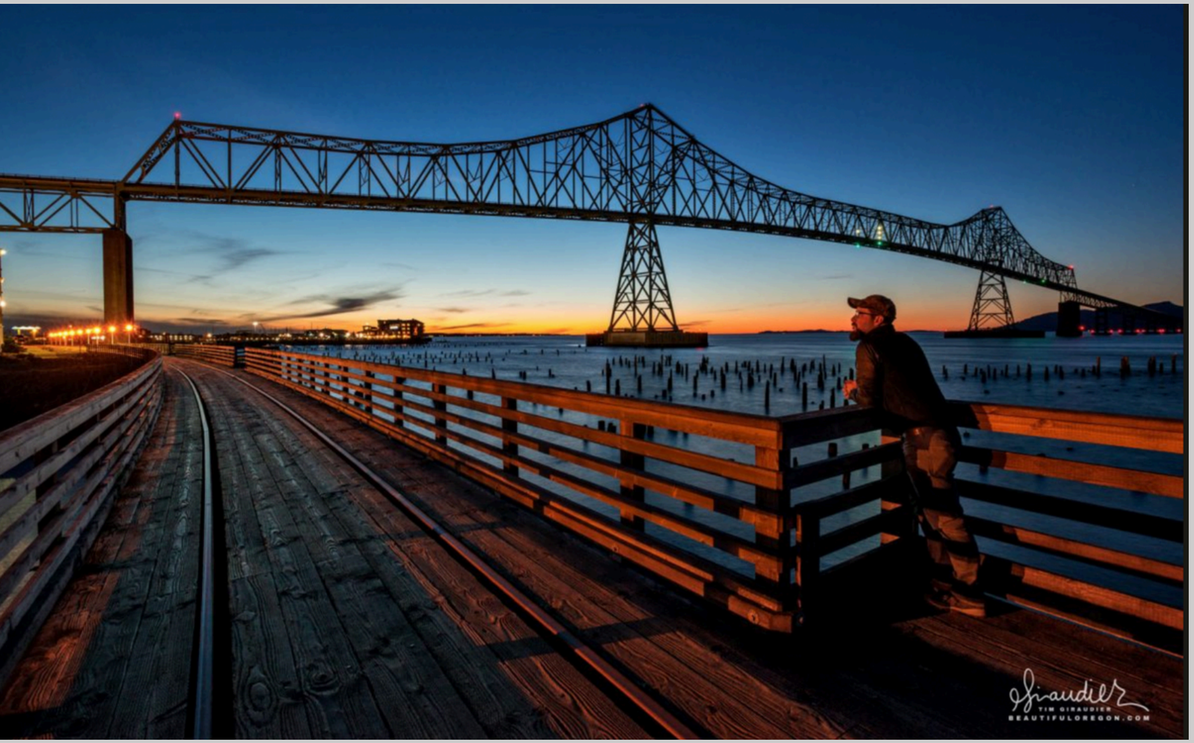
- This structure is a 2-D Truss model of Astoria Megler. We are curious to find the deflected shape, unknown reactions, bending moments, shear force, and axial force in each member. We have considered wind load, dead load (road), and live load (vehicles). We tried to optimize deflection by selecting the supports before taking the truss.
- Unlike cantilever bridges, the continuous truss has fewer joints and is supported by a series of trusses that share load across a continuous span. This structure minimizes bending moments and allows for efficient load transfer over long distances.
- The bridge’s location near the Columbia River exposes it to high winds, which the truss design helps mitigate by allowing wind to pass through, reducing overall pressure on the structure.

PROBLEM STATEMENT

We have replicated a 2D truss structure by analysing the side view geometry of Astoria - Megler Bridge constructed in Astoria, USA. There are a total of 141 nodes and 266 members. The Loads and supports have been mentioned with view of realistic conditions.

We have to plot the deflected shape, find the unknown reactions, and report the forces (bending moment, shear force and axial force) in different members.

ABOUT THE TRUSS



The Astoria-Megler Bridge spans the Columbia River at its mouth, connecting Astoria and Megler located near the Pacific Ocean.

It’s one of the longest continuous truss bridges in North America at 4.1 miles, spanning a large body of water with no piers for extensive stretches.

The truss structure provides flexibility and can absorb and dissipate seismic forces, making the bridge more resilient in the event of an earthquake.

The Astoria-Megler Bridge is a celebrated landmark, drawing visitors for its stunning views of the Columbia River,Astoria, and surrounding landscapes. It’s frequently featured in films, media, and regional promotions.

METHODOLOGY OVERVIEW

Application of Loads

Gather a comprehensive dataset including the nodes and the members of the 2D truss.

Matrix Formation

Transforming local stiffness matrix to global stiffness matrix, and applying boundary conditions.

Determination

Solving for displacements at free DOFs and support reactions using global stiffness matrix.

Plotting

Finally we plotted the nodal displacement points to get deformed shape of the structure.

MATRIX STRUCTURAL ANALYSIS

We have taken a 2D Truss with the number of nodes as shown earlier and prepared the joint coordinate and member connectivity matrices.
Under the application of Live Loads, Dead Loads, and Wind Load, we have proceeded to the structural analysis.

INTRODUCTION

This solution for the truss allows us to calculate the deformed shape of any structure, which will help us optimize deflection in various ways and improve safety and structural integrity.

APPLICATION

We got the deflected shape, bending moment, axial force, shear force and unknown reactions. The bridge is more deflected in the middle, one more support can be added to reduce deformation

RESULT

05

CODES

```
% Initialize workspace
clear; clc;

% Define scale factor for plotting
applied_scale_factor = 2000;

% Specify restrained degrees of freedom (DOFs)
restrained_dofs = [205, 206, 260, 301, 302, 311, 323, 338, 344, 346, 347, 386, 388, 389];


% Read joint coordinates and member connectivity from Excel
Node_Coordinates = xlsread("DRSR.xlsx", "Joint Coordinates1");
Member_Connectivity = xlsread("DRSR.xlsx", "Member Connectivity1");

% Get the number of joints and members
[num_joints, ~] = size(Node_Coordinates);
[num_members, ~] = size(Member_Connectivity);
```

```
% Initialize global stiffness matrix and force vectors
K_global = zeros(3 * num_joints);
fixed_end_forces = zeros(3 * num_joints, 1);
nodal_forces = zeros(3 * num_joints, 1);
settlements = zeros(3 * num_joints, 1);

% Assemble global stiffness matrix and load vectors
for member = 1:num_members
    % Extract member properties
    area = Member_Connectivity(member, 4);
    elasticity = Member_Connectivity(member, 5);
    inertia = Member_Connectivity(member, 6);
    load_w = Member_Connectivity(member, 7);
    joint_case = Member_Connectivity(member, 8);
    node_i = Member_Connectivity(member, 2);
    node_j = Member_Connectivity(member, 3);

    % Extract settlements
    settlement_x_i = Node_Coordinates(node_i, 4);
    settlement_y_i = Node_Coordinates(node_i, 5);
    settlement_x_j = Node_Coordinates(node_j, 4);
    settlement_y_j = Node_Coordinates(node_j, 5);

    % Extract concentrated forces and moments
    conc_force_x_i = Node_Coordinates(node_i, 6);
    conc_force_y_i = Node_Coordinates(node_i, 7);
    conc_moment_z_i = Node_Coordinates(node_i, 8);
    conc_force_x_j = Node_Coordinates(node_j, 6);
    conc_force_y_j = Node_Coordinates(node_j, 7);
    conc_moment_z_j = Node_Coordinates(node_j, 8);

    % Coordinates of nodes
    x_i = Node_Coordinates(node_i, 2);
    y_i = Node_Coordinates(node_i, 3);
    x_j = Node_Coordinates(node_j, 2);
    y_j = Node_Coordinates(node_j, 3);

    % Calculate member length and orientation
    member_length = sqrt((x_j - x_i)^2 + (y_j - y_i)^2);
    cos_theta = (x_j - x_i) / member_length;
    sin_theta = (y_j - y_i) / member_length;

```

```
67      % Determine local stiffness matrix and fixed-end forces based on joint case
68      switch joint_case
69          case 1
70              kloc = ((inertia * elasticity) / member_length^3) * ...
71                  [(area * member_length^2) / inertia, 0, 0, -(area * member_length^2) / inertia, 0, 0;
72                   0, 12, 6 * member_length, 0, -12, 6 * member_length;
73                   0, 6 * member_length, 4 * member_length^2, 0, -6 * member_length, 2 * member_length^2
74                   -(area * member_length^2) / inertia, 0, 0, (area * member_length^2) / inertia, 0, 0;
75                   0, -12, -6 * member_length, 0, 12, -6 * member_length;
76                   0, 6 * member_length, 2 * member_length^2, 0, -6 * member_length, 4 * member_length^2
77              Q = [0; (load_w * member_length) / 2; (load_w * member_length^2) / 12; 0; (load_w * member_
78          case 2
79              kloc = ((inertia * elasticity) / member_length^3) * ...
80                  [(area * member_length^2) / inertia, 0, 0, -(area * member_length^2) / inertia, 0, 0;
81                   0, 3, 0, 0, -3, 3 * member_length;
82                   0, 0, 0, 0, 0, 0;
83                   -(area * member_length^2) / inertia, 0, 0, (area * member_length^2) / inertia, 0, 0;
84                   0, -3, 0, 0, 3, -3 * member_length;
85                   0, 3 * member_length, 0, 0, -3 * member_length, 3 * member_length^2];
86              Q = [0; (load_w * member_length) / 2 - (1.5 / member_length) * (load_w * member_length^2) /
87          case 3
88              kloc = ((inertia * elasticity) / member_length^3) * ...
89                  [(area * member_length^2) / inertia, 0, 0, -(area * member_length^2) / inertia, 0, 0;
90                   0, 3, 3 * member_length, 0, -3, 0;
91                   0, 3 * member_length, 3 * member_length^2, 0, -3 * member_length, 0;
92                   -(area * member_length^2) / inertia, 0, 0, (area * member_length^2) / inertia, 0, 0;
93                   0, -3, -3 * member_length, 0, 3, 0;
94                   0, 0, 0, 0, 0, 0];
95              Q = [0; (load_w * member_length) / 2 - (1.5 / member_length) * (-load_w * member_length^2) /
96          case 4
97              kloc = (elasticity * area) / member_length * ...
98                  [1, 0, 0, -1, 0, 0;
99                   0, 0, 0, 0, 0, 0;
100                  0, 0, 0, 0, 0, 0;
101                  -1, 0, 0, 1, 0, 0;
102                  0, 0, 0, 0, 0, 0;
103                  0, 0, 0, 0, 0, 0];
104             Q = [0; (load_w * member_length) / 2; 0; 0; (load_w * member_length) / 2; 0];
```

```
105         otherwise
106             error('Invalid joint case specified.');
107     end
108
109     % Transformation matrix
110     T = [cos_theta, sin_theta, 0, 0, 0, 0;
111         -sin_theta, cos_theta, 0, 0, 0, 0;
112          0, 0, 1, 0, 0, 0;
113          0, 0, 0, cos_theta, sin_theta, 0;
114          0, 0, 0, -sin_theta, cos_theta, 0;
115          0, 0, 0, 0, 0, 1];
116
117     % Global stiffness matrix for the member
118     k_global_member = T' * kloc * T;
119
120     % Global DOF indices
121     global_dof = [3 * (node_i - 1) + 1, 3 * (node_i - 1) + 2, 3 * (node_i - 1) + 3, 3 * (node_j
122
123     % Assemble into global stiffness matrix
124     K_global(global_dof, global_dof) = K_global(global_dof, global_dof) + k_global_member;
```

```
126         % Account for settlements
127         delta = zeros(6, 1);
128         delta(1) = settlement_x_i;
129         delta(2) = settlement_y_i;
130         delta(4) = settlement_x_j;
131         delta(5) = settlement_y_j;
132         settlements(global_dof([1, 2, 4, 5])) = [settlement_x_i; settlement_y_i; settlement_x_j; settlem
133         z = k_global_member * delta;
134         fixed_end = T' * Q + z;
135
136         % Assemble fixed-end forces
137         fixed_end_forces(global_dof) = fixed_end_forces(global_dof) + fixed_end;
138
139         % Assemble nodal forces
140         nodal_forces(global_dof) = nodal_forces(global_dof) + [conc_force_x_i; conc_force_y_i; conc_mome
141     end
142
143     % Apply boundary conditions by removing restrained DOFs
144     K_reduced = K_global;
145     nodal_forces_reduced = nodal_forces;
```

09

```
146     fixed_end_forces_reduced = fixed_end_forces;
147
148     for idx = 1:length(restrained_dofs)
149         dof = restrained_dofs(idx) - (idx - 1);
150         K_reduced(dof, :) = [];
151         K_reduced(:, dof) = [];
152         nodal_forces_reduced(dof) = [];
153         fixed_end_forces_reduced(dof) = [];
154     end
155
156     % Solve for displacements
157     displacements_reduced = K_reduced \ (nodal_forces_reduced - fixed_end_forces_reduced);
158
159     % Reconstruct full displacement vector including restrained DOFs
160     displacements_full = zeros(3 * num_joints, 1);
161     free_dofs = setdiff(1:3 * num_joints, restrained_dofs);
162     displacements_full(free_dofs) = displacements_reduced;
163     displacements_full = displacements_full + settlements;
164
```

```
165     % Write member forces to Excel
166     headers = ["MEMBER_NUMBER", "F_x_NODEi", "F_y_NODEi", "M_z_NODEi", "
167     writematrix(headers, 'Output DRSR.xlsx', 'Range', 'A1');
168
169     for member = 1:num_members
170         % Extract member properties
171         area = Member_Connectivity(member, 4);
172         elasticity = Member_Connectivity(member, 5);
173         inertia = Member_Connectivity(member, 6);
174         load_w = Member_Connectivity(member, 7);
175         joint_case = Member_Connectivity(member, 8);
176         node_i = Member_Connectivity(member, 2);
177         node_j = Member_Connectivity(member, 3);
178
179         % Coordinates of nodes
180         x_i = Node_Coordinates(node_i, 2);
181         y_i = Node_Coordinates(node_i, 3);
182         x_j = Node_Coordinates(node_j, 2);
183         y_j = Node_Coordinates(node_j, 3);
```

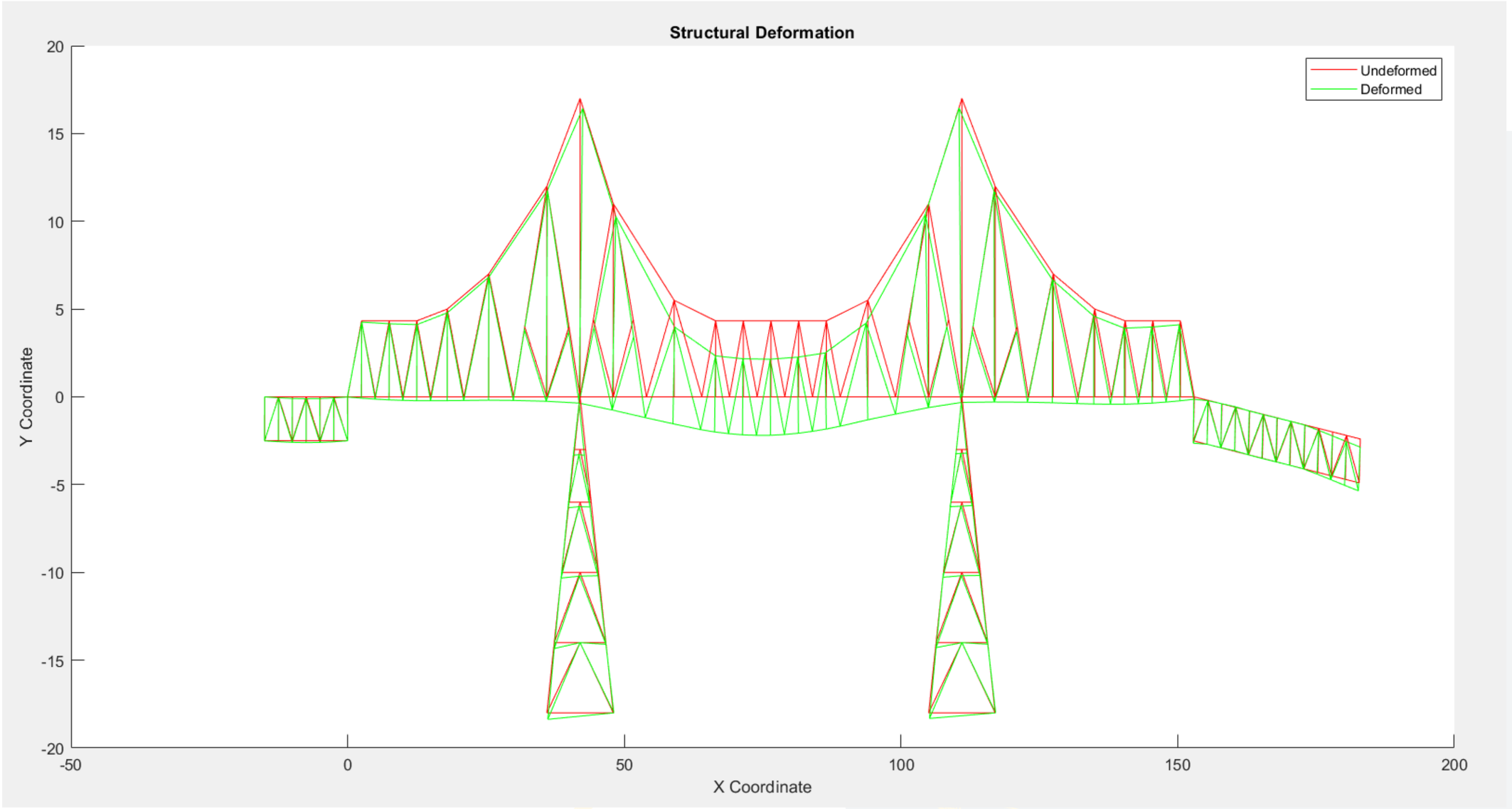
```
% Calculate member length and orientation
member_length = sqrt((x_j - x_i)^2 + (y_j - y_i)^2);
cos_theta = (x_j - x_i) / member_length;
sin_theta = (y_j - y_i) / member_length;

% Determine local stiffness matrix and fixed-end forces based on joint case
switch joint_case
    case 1
        kloc = ((inertia * elasticity) / member_length^3) * ...
            [(area * member_length^2) / inertia, 0, 0, -(area * member_length^2) / inertia, 0, 0;
             0, 12, 6 * member_length, 0, -12, 6 * member_length;
             0, 6 * member_length, 4 * member_length^2, 0, -6 * member_length, 2 * member_length^2;
             -(area * member_length^2) / inertia, 0, 0, (area * member_length^2) / inertia, 0, 0;
             0, -12, -6 * member_length, 0, 12, -6 * member_length;
             0, 6 * member_length, 2 * member_length^2, 0, -6 * member_length, 4 * member_length^2];
        Q = [0; (load_w * member_length) / 2; (load_w * member_length^2) / 12; 0; (load_w * member_length) / 2;
    case 2
        kloc = ((inertia * elasticity) / member_length^3) * ...
            [(area * member_length^2) / inertia, 0, 0, -(area * member_length^2) / inertia, 0, 0;
             0, 3, 0, 0, -3, 3 * member_length;
             0, 0, 0, 0, 0, 0;
             -(area * member_length^2) / inertia, 0, 0, (area * member_length^2) / inertia, 0, 0;
             0, -3, 0, 0, 3, -3 * member_length;
             0, 3 * member_length, 0, 0, -3 * member_length, 3 * member_length^2];
        Q = [0; (load_w * member_length) / 2 - (1.5 / member_length) * (load_w * member_length^2) / 12; 0; 0; (l
    case 3
        kloc = ((inertia * elasticity) / member_length^3) * ...
            [(area * member_length^2) / inertia, 0, 0, -(area * member_length^2) / inertia, 0, 0;
             0, 3, 3 * member_length, 0, -3, 0;
             0, 3 * member_length, 3 * member_length^2, 0, -3 * member_length, 0;
             -(area * member_length^2) / inertia, 0, 0, (area * member_length^2) / inertia, 0, 0;
             0, -3, -3 * member_length, 0, 3, 0;
             0, 0, 0, 0, 0, 0];
        Q = [0; (load_w * member_length) / 2 - (1.5 / member_length) * (-load_w * member_length^2) / 12; (load_w
    case 4
        kloc = (elasticity * area) / member_length * ...
            [1, 0, 0, -1, 0, 0;
             0, 0, 0, 0, 0, 0;
             0, 0, 0, 0, 0, 0;
             -1, 0, 0, 1, 0, 0;
```

```
            Q = [0; (load_w * member_length) / 2; 0; 0; (load_w * member_length) / 2; 0];
        otherwise
            error('Invalid joint case specified.');
    end

    % Transformation matrix
    T = [cos_theta, sin_theta, 0, 0, 0, 0;
        -sin_theta, cos_theta, 0, 0, 0, 0;
         0, 0, 1, 0, 0, 0;
         0, 0, 0, cos_theta, sin_theta, 0;
         0, 0, 0, -sin_theta, cos_theta, 0;
         0, 0, 0, 0, 0, 1];

    % Global DOF indices
    global_dof = [3 * (node_i - 1) + 1, 3 * (node_i - 1) + 2, 3 *

    % Extract displacements for member DOFs
    displacements_member = displacements_full(global_dof);
```

```
    % Compute member end forces
    member_end_forces = kloc * (T * displacements_member) + Q;

    % Prepare data for writing
    member_forces_output = round(0.001 * member_end_forces, 2);
    data_row = {member, member_forces_output(1), member_forces_output(2), member_force

    % Write data to Excel
    writecell(data_row, 'Output DRSR.xlsx', 'Range', strcat('A', num2str(member + 1)))
end

% Plot undeformed and deformed structures
x_coords_undeformed = Node_Coordinates(:, 2);
y_coords_undeformed = Node_Coordinates(:, 3);
x_coords_deformed = x_coords_undeformed + displacements_full(1:3:end) * applied_scale_
y_coords_deformed = y_coords_undeformed + displacements_full(2:3:end) * applied_scale_

figure;
hold on;
for member = 1:num_members
```

```
figure;
hold on;
for member = 1:num_members
    node_i = Member_Connectivity(member, 2);
    node_j = Member_Connectivity(member, 3);

    % Plot undeformed member
    plot([x_coords_undeformed(node_i), x_coords_undeformed(node_j)], [y_coords_undeformed(node_i), y_coords_unde

    % Plot deformed member
    plot([x_coords_deformed(node_i), x_coords_deformed(node_j)], [y_coords_deformed(node_i), y_coords_deformed(n
end
hold off;
legend('Undeformed', 'Deformed');
title('Structural Deformation');
xlabel('X Coordinate');
ylabel('Y Coordinate');
disp('The deformed and undeformed structures have been plotted. The deformation is scaled for visibility.');
fprintf('The deformed shape has been exaggerated %d times for visualization purposes.\n', applied_scale_factor);
```

RESULT


Structural Deformation
Undeformed
Deformed
Y Coordinate
X Coordinate
20
15
10
5
0
-5
-10
-15
-20
-50
0
50
100
150
200

CONCLUSION

1. The large deformations in the middle span is due to absence of any ground support in the middle, point to lower stiffness.
2. Load Distribution and Stability : The triangular design of trusses provides excellent stability by efficiently transferring loads from the deck to the bridge supports.
3. Bridge demonstrates a highly efficient and stable structure, capable of withstanding environmental forces while providing safe and reliable transportation.

THANK YOU