AE 675
INTRODUCTION TO FINITE ELEMENT METHODS

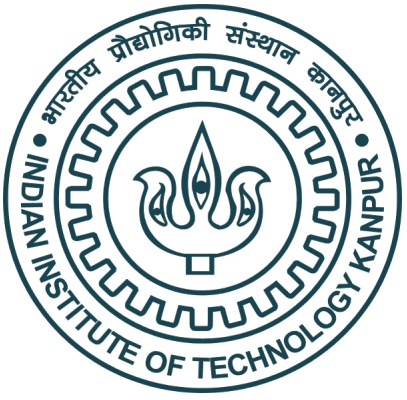
Project Report

by

Rohan Sai Adiserla - 241010051

Karthan Sai Prasad - 241010054

Ganji Revanth Kumar - 241010022



DEPARTMENT OF AEROSPACE ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

April 2025

Problem Statement:

A square steel plate of dimension 1 m × 1 m × 1 cm is heated at a rate of 1 KW at its middle. Assuming two-dimensional steady state heat conduction problem and using linear triangular elements, find the temperature distribution in the plate.

Case 1: When three sides of the plate are insulated, and the 4th side is maintained at 25°C.

a) using 2 elements and $q = 1000\ \delta(x - x_c\ ,\ y - y_c\ )$ W/m^2 , where x_c and y_c are the coordinates of the centre of the plate, and δ is the Dirac-delta function.

Step 1: *Define the geometry, mesh parameters, and material properties.*

```
clc;
clear;


% Define the domain dimensions
W = 1;
H = 1;


% Number of nodes in each direction
Nx = 2;
Ny = 2;


% Element size
dx = W / (Nx-1);
dy = H / (Ny-1);


% Material properties
T_val = 25;
k = 50;


% Number of triangular elements
n = 2*(Nx-1)*(Ny-1);
```

Step 2: *Mesh generation.*

Create a rectangular mesh with 4 nodes positioned at each corner (top-left, top-right, bottom-left, and bottom-right), then connect the nodes with elements along

all four outer edges and add one diagonal element connecting the top-left node to the bottom-right node.

```
% Allocate memory for mesh nodes
p = zeros(2, Nx*Ny);
el = zeros(3, n);

% nodes
index = 0;
for i = 1:Ny
    y = (i-1) * dy;
    for j = 1:Nx
        x = (j-1) * dx;
        index = index + 1;
        p(:, index) = [x; y];
    end
end

% elements
index = 0;
for i = 1:Ny-1
    for j = 1:Nx-1
        index1 = j + (i-1)*Nx;
        index2 = index1 + 1;
        index3 = index2 + Nx;
        index4 = index1 + Nx;


        index = index + 1;
        el(:,index) = [index1; index2; index4];


        index = index + 1;
        el(:,index) = [index2; index3; index4];
    end
end

% Plot mesh
figure;
hold on;
patch('faces', el', 'vertices', p', 'facecolor', 'c', 'edgecolor',
'k');
plot(p(1,:), p(2,:), 'o', 'MarkerFaceColor', 'r');
axis off;
hold off;
```

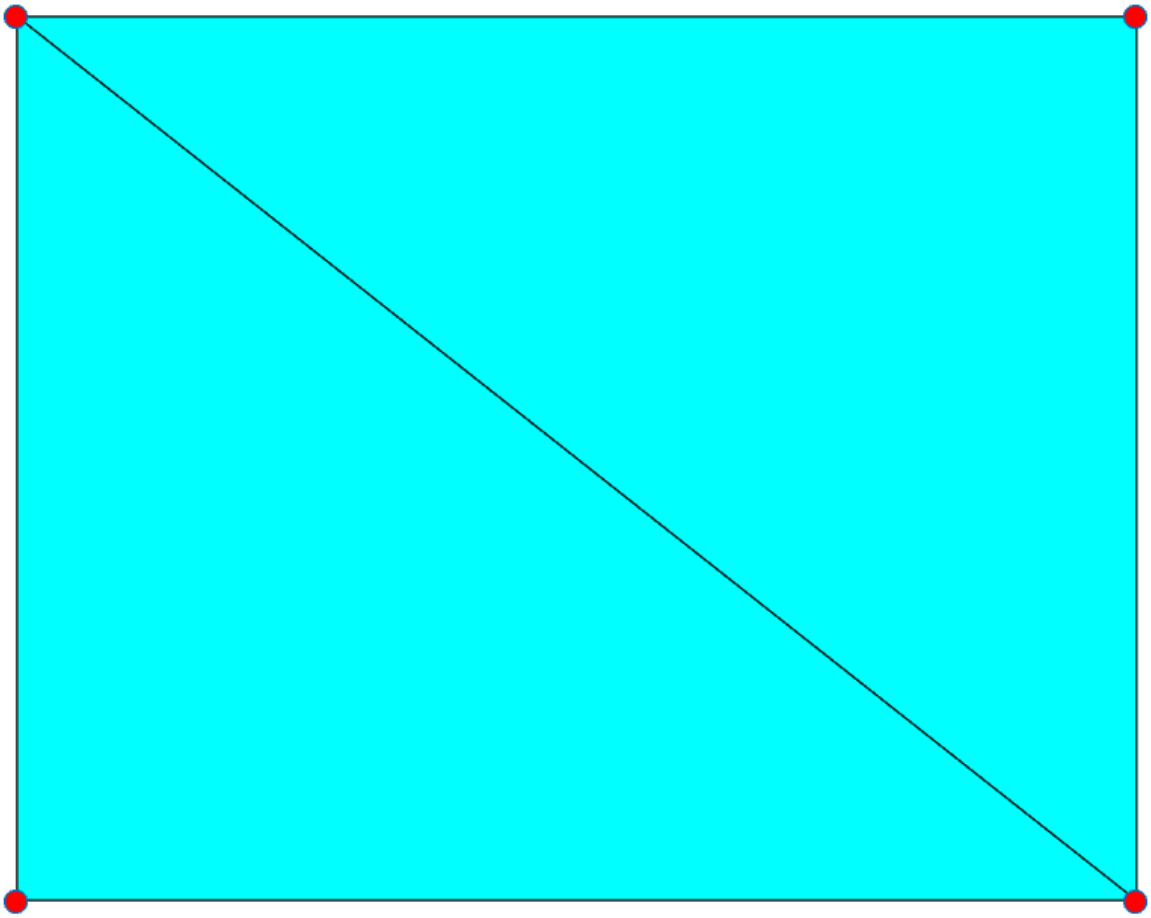


Figure 1: Mesh (Case 1a)

Step 3: *Assembling the Global Stiffness matrix.*

The global stiffness matrix is assembled by looping through each triangular element in the mesh. For each element, the coordinates of its nodes are extracted and used to compute the Jacobian matrix J. The area A of the triangle is calculated using the determinant of the Jacobian. A predefined matrix Q is used to compute the local stiffness matrix K, which incorporates the material's thermal conductivity and the geometry of the element. The local stiffness contributions from each element are then added to the corresponding entries in the global stiffness matrix K_global using nested loops.

```
% Stiffness matrix assembly
K_global = zeros(length(p));
for i = 1:n
    nodes = el(:,i);
```

```
    x = p(1, nodes);
    y = p(2, nodes);

    J = [x(1)-x(3), x(2)-x(3); y(1)-y(3), y(2)-y(3)];
    A = 0.5 * abs(det(J));
    Q = [1, 0, -1; 0, 1, -1];
    K = 50 * A * (Q' / J) * (Q' / J)';


    for j = 1:3
        for k = 1:3
            K_global(nodes(j), nodes(k)) = K_global(nodes(j),
nodes(k)) + K(j,k);
        end
    end
end
```

Step 4: *Assembling the Global Load vector.*

In this step, the global load vector q_global is assembled to represent the effect of a heat source applied at the centre of the domain. The source is defined as a constant value q = 1000 W/m² located at coordinates (xc, yc) = (0.5, 0.5). For each triangular element, the code calculates the area A0 of the element and the sub-areas A1, A2, and A3, which are formed by replacing one vertex of the triangle with the heat source location. These sub-areas are used to compute the shape function values N1, N2, and N3 at the source location using an area-based interpolation method. These values represent the influence of the heat source on each of the triangle's nodes. The total heat is evenly distributed among all elements (c = q/n), and the contributions are added to the corresponding entries in the global load vector q_global. This vector will later be used in conjunction with the stiffness matrix to solve for the temperature distribution.

```
% Load vector (heat source)
xc = 0.5;
yc = 0.5;
q = 1000;
q_global = zeros(size(p,2),1);


% Distribute heat source among elements
for i = 1:n
    nodes = el(:,i);
    x = p(1,nodes);
    y = p(2,nodes);
```

```
    A0 = 0.5 * abs((x(2)-x(1))*(y(3)-y(1)) - (y(2)-y(1))*(x(3)-
x(1)));

    A1 = 0.5 * abs((x(2)-xc)*(y(3)-yc) - (y(2)-yc)*(x(3)-xc));
    A2 = 0.5 * abs((x(1)-xc)*(y(3)-yc) - (y(1)-yc)*(x(3)-xc));
    A3 = 0.5 * abs((x(2)-xc)*(y(1)-yc) - (y(2)-yc)*(x(1)-xc));

    N1 = A1/A0;
    N2 = A2/A0;
    N3 = A3/A0;

    c = q/n;
    q_global(nodes(1)) = q_global(nodes(1)) + c * N1;
    q_global(nodes(2)) = q_global(nodes(2)) + c * N2;
    q_global(nodes(3)) = q_global(nodes(3)) + c * N3;
end
```

Step 5: *Applying Boundary Conditions and Solving the System.*

The nodes along the left edge of the domain (where x = 0) are identified as having known (Dirichlet) boundary temperatures (T_val). The rest of the nodes are considered unknowns. The global stiffness matrix and load vector are partitioned accordingly: K1 corresponds to the unknowns, and K2 captures interactions between known and unknown nodes. The reduced system is solved for the unknown temperatures using linear algebra (T_free = K1 \ f). Finally, the full temperature vector T is assembled by combining known and computed values.

```
% Apply boundary conditions
Known = find(p(1,:) == 0); %assuming left side to be 4th side
Unknown = setdiff(1:length(p), Known);

% Partition stiffness matrix and force vector
K1 = K_global(Unknown, Unknown);
K2 = K_global(Unknown, Known);
T_calc = T_val * ones(length(Known),1);
f = q_global(Unknown) - K2 * T_calc;


% Solve for unknown temperatures
T_free = K1 \ f;
% Assemble final temperature vector
T = zeros(length(p),1);
T(Unknown) = T_free;
T(Known) = T_val;
```

Step 6: *Plotting Results*

This step displays the nodal temperatures and visualizes the temperature distribution using a contour plot. It also plots the temperature variation along the horizontal centreline (y = 0.5).

```
% Display results
disp('Temperature Distribution (C):');
for i = 1:size(p,2)
    disp(['Node ', num2str(i), ' (', num2str(p(1,i)), ',',
num2str(p(2,i)), '): ', num2str(T(i))]);
end


% Plot contour of temperature
grid_x = linspace(0, W, 50);
grid_y = linspace(0, H, 50);
[Xq, Yq] = meshgrid(grid_x, grid_y);
Tq = griddata(p(1,:), p(2,:), T, Xq, Yq, 'cubic');
figure;
contourf(Xq, Yq, Tq, 200, 'LineColor', 'none');
colorbar;
title('Temperature Contour Plot');
xlabel('X');
ylabel('Y');


% Plot temperature along y = 0.5
x_vals = linspace(0, W, 50);
T_vals = griddata(p(1,:), p(2,:), T, x_vals, repmat(0,
size(x_vals)), 'cubic');
figure;
plot(x_vals, T_vals, '-', 'LineWidth', 2);
title('Temperature Variation along y = 0.5');
xlabel('X');
ylabel('Temperature (C)');
grid on;
```

```
Temperature Distribution (C):
Node 1 (0,0): 25
Node 2 (1,0): 38.3333
Node 3 (0,1): 25
Node 4 (1,1): 31.6667
```

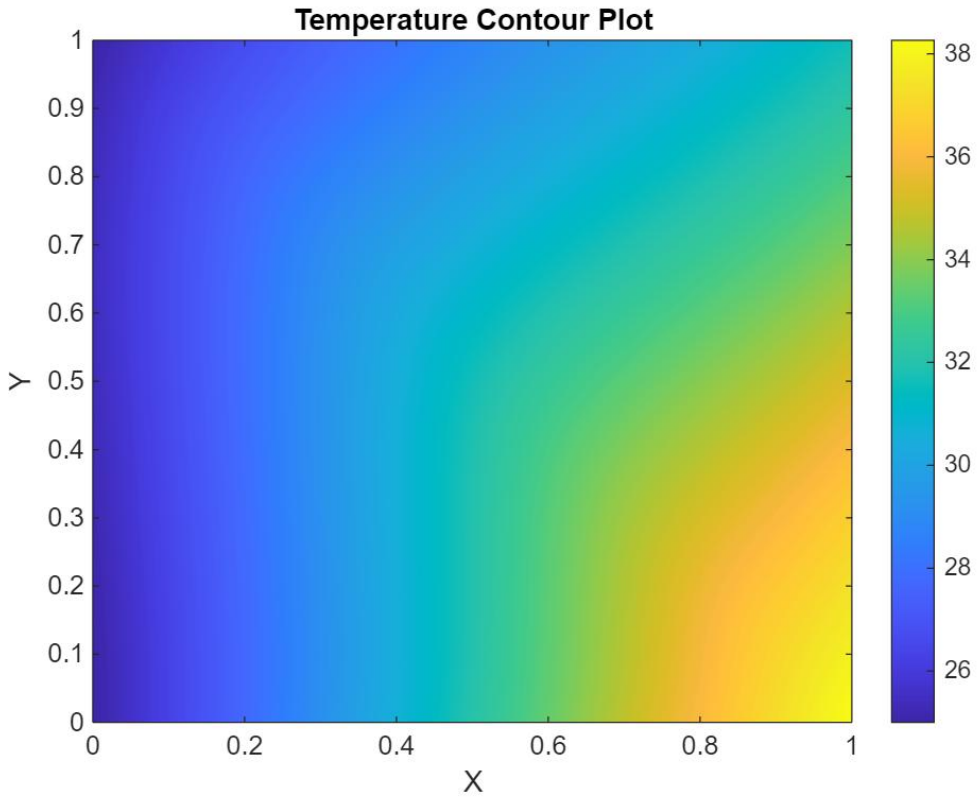


Figure 2: Contour Plot (Case 1a)

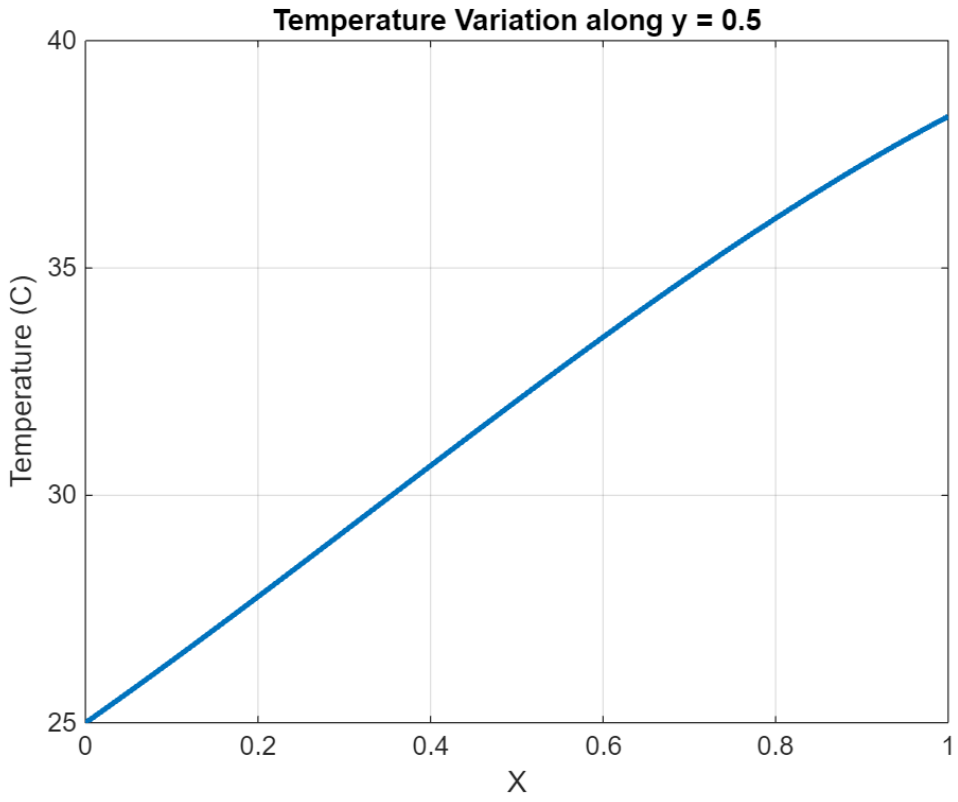


Figure 3: Temperature variation along the horizontal centreline (Case 1a)

b) using 4 elements intersecting at the centre.

Mesh 1: Structured mesh

Step 1: *Define the geometry, mesh parameters, and material properties.*

Step 2: *Mesh generation. (The only change from 1a)*

```
% nodes
p = zeros(2, Nx*Ny);
index = 0;
for i = 1:Ny
    for j = 1:Nx
        index = index + 1;
        p(:, index) = [(j-1)*dx; (i-1)*dy];
    end
end
% Center nodes
x_cells = Nx-1;
y_cells = Ny-1;
centers = zeros(2, x_cells*y_cells);
index = 0;
for i = 1:y_cells
    for j = 1:x_cells
        index = index + 1;
        centers(:, index) = [(j-0.5)*dx; (i-0.5)*dy];
    end
end
p = [p centers]; % Combine original and center nodes
% Generate triangular elements
el = [];
center_offset = Nx * Ny;
for i = 1:y_cells
    for j = 1:x_cells
        % Corner node indices
        index1 = (i-1)*Nx + j;
        index2 = index1 + 1;
        index3 = index2 + Nx;
        index4 = index3 - 1;

        % Center node index
        center_idx = center_offset + (i-1)*x_cells + j;

        % Create four triangular elements per cell ensuring symmetry
        el = [el [index1; index2; center_idx]];
        el = [el [index2; index3; center_idx]];
        el = [el [index3; index4; center_idx]];
        el = [el [index4; index1; center_idx]];
    end
end
```

```
figure;hold on;
patch('faces', el', 'vertices', p', 'facecolor', 'c', 'edgecolor',
'k');
plot(p(1,:), p(2,:), 'o', 'MarkerFaceColor', 'r');
axis off;hold off;
```

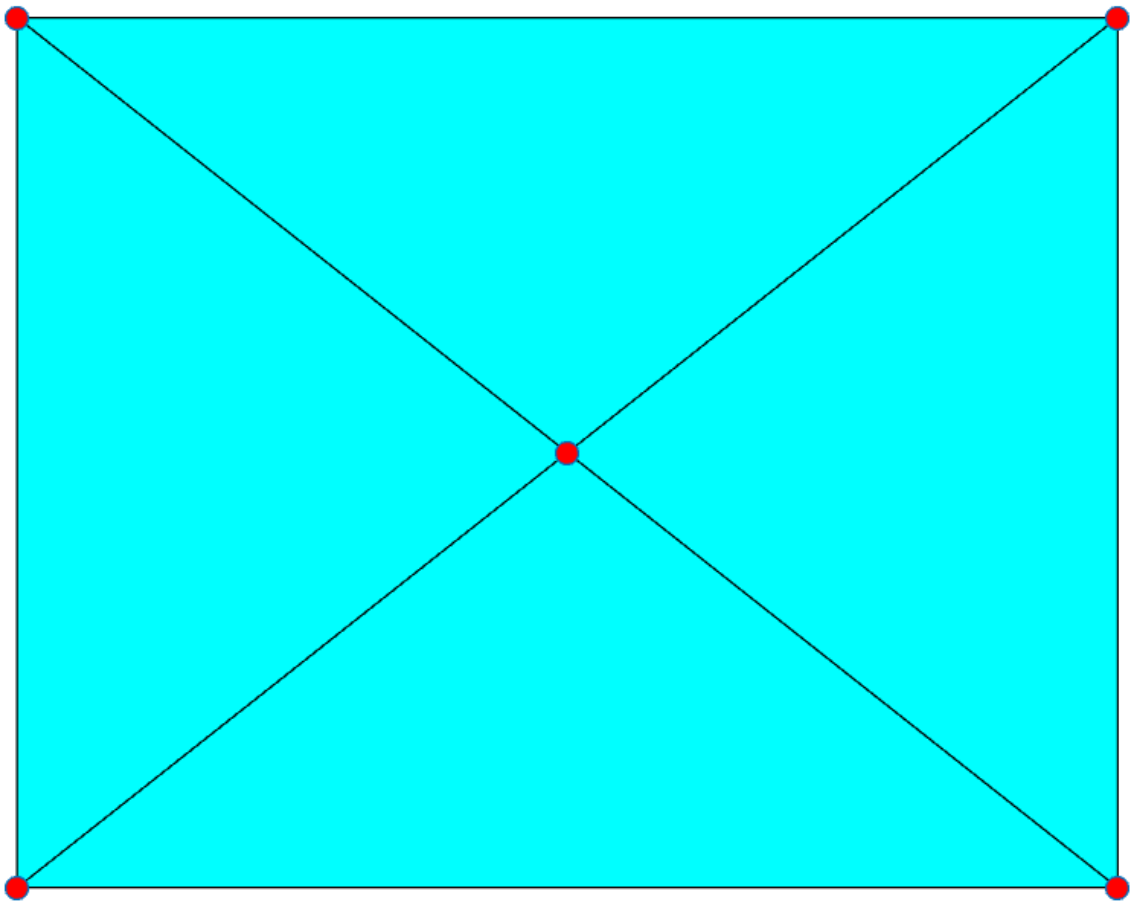


Figure 4: Mesh (Case 1b.1)

Step 3: *Assembling the Global Stiffness matrix.*

Step 4: *Assembling the Global Load vector.*

Step 5: *Applying Boundary Conditions and Solving the System.*

Step 6: *Plotting Results*

```
Temperature Distribution (C):
Node 1 (0,0): 25
Node 2 (1,0): 35
Node 3 (1,1): 35
Node 4 (0,1): 25
Node 5 (0.5,0.5): 35
```

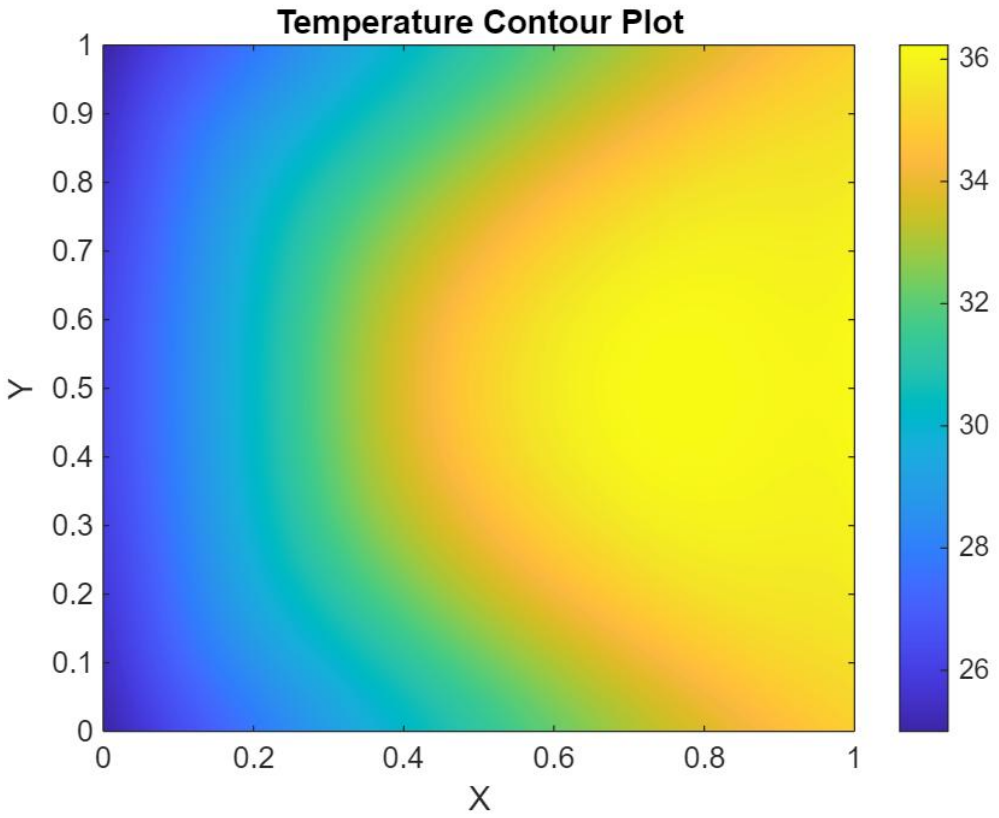


Figure 5: Contour Plot (Case 1b.1)

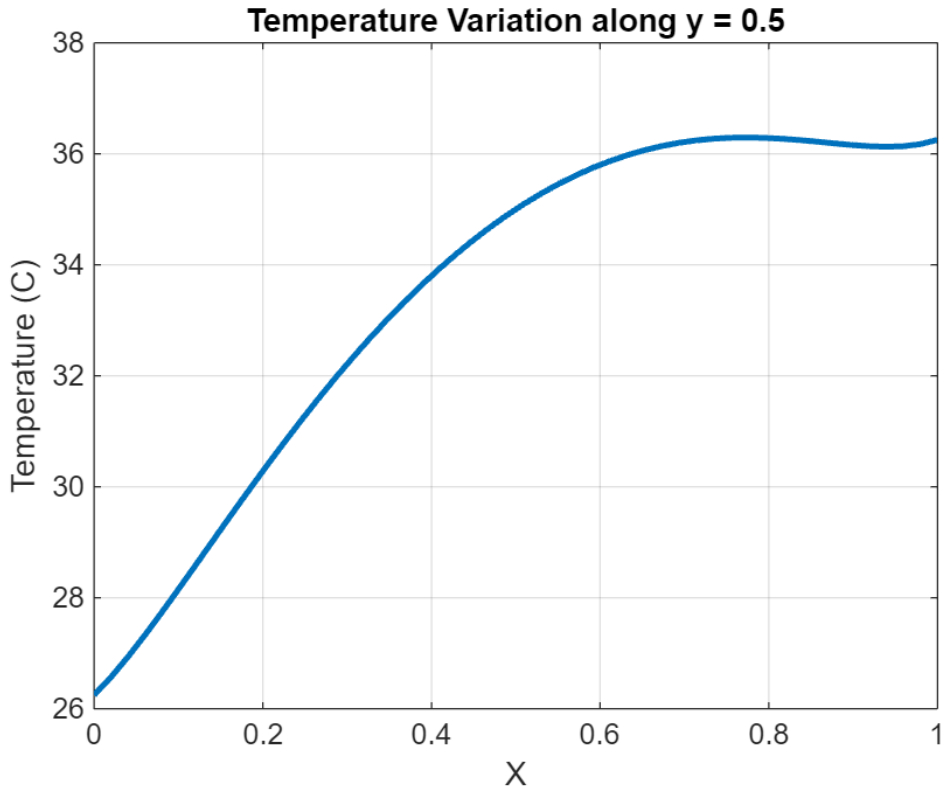


Figure 6: Temperature variation along the horizontal centreline (Case 1b.1)

Mesh 2: Elements intersecting at the centre.

Step 1: *Define the geometry, mesh parameters, and material properties.*

Step 2: *Mesh generation. (The only change from 1a)*

```
% generating nodes
p = [];
% bottom boundary nodes
for j = 1:Nx
    p = [p [W*(j-1)/(Nx-1); 0]];
end
% right boundary nodes
for i = 2:Ny
    p = [p [W; H*(i-1)/(Ny-1)]];
end
% top boundary nodes
for j = Nx-1:-1:1
    p = [p [W*(j-1)/(Nx-1); H]];
end
% left boundary nodes
for i = Ny-1:-1:2
    p = [p [0; H*(i-1)/(Ny-1)]];
end
% center node
center_point = [W/2; H/2];
p = [p center_point];
center_idx = size(p, 2);
% Generate triangular elements
el = [];
num_boundary = center_idx - 1;
% Connect adjacent boundary nodes with the center
for i = 1:num_boundary
    next_i = i + 1;
    if next_i > num_boundary
        next_i = 1;
    end
    el = [el [i; next_i; center_idx]];
end

% Plot the mesh
figure;
hold on;
patch('faces', el', 'vertices', p', 'facecolor', 'c', 'edgecolor',
'k');
plot(p(1,1:end), p(2,1:end), 'o', 'MarkerFaceColor', 'r');
axis off;
hold off;
```

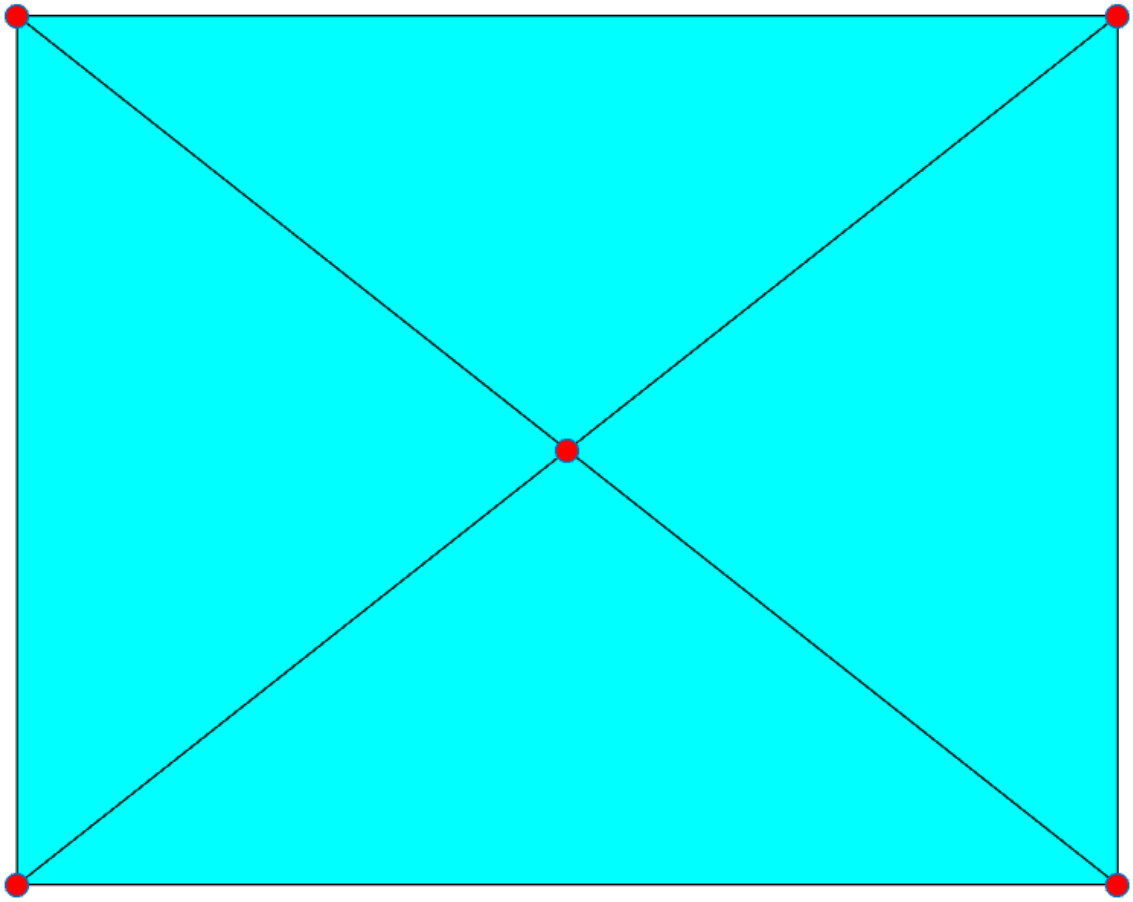


Figure 7: Mesh (Case 1b.2)

Step 3: *Assembling the Global Stiffness matrix.*

Step 4: *Assembling the Global Load vector.*

Step 5: *Applying Boundary Conditions and Solving the System.*

Step 6: *Plotting Results*

```
Temperature Distribution (C):
Node 1 (0,0): 25
Node 2 (1,0): 35
Node 3 (1,1): 35
Node 4 (0,1): 25
Node 5 (0.5,0.5): 35
```

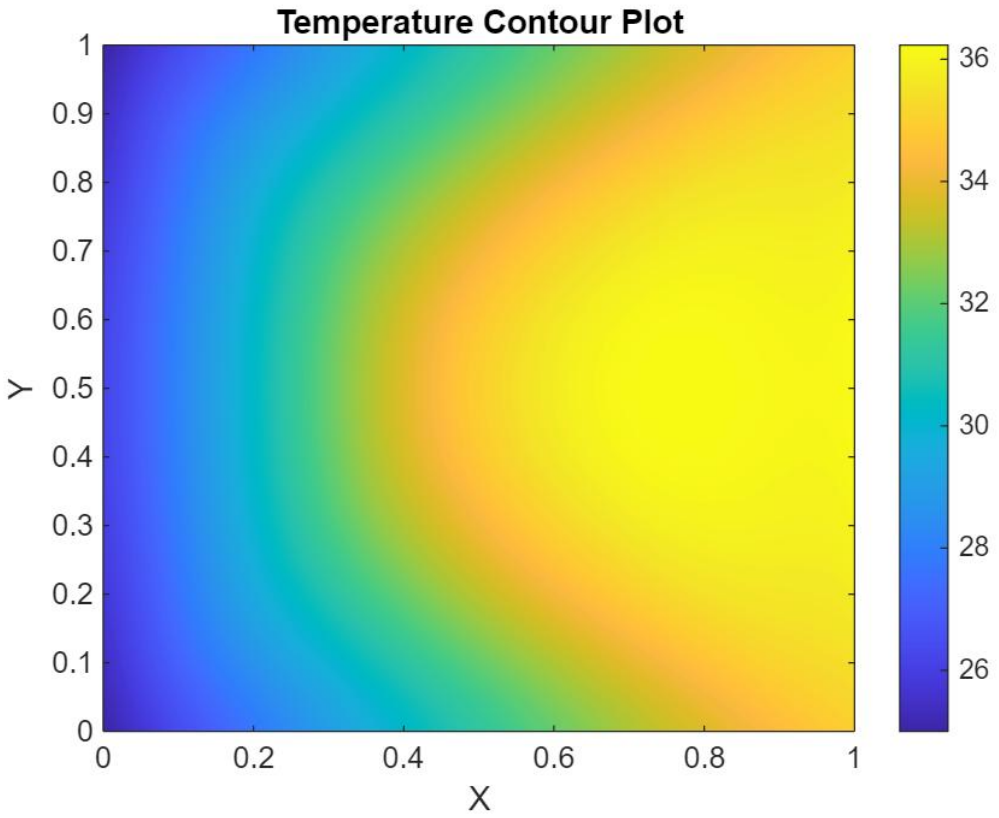


Figure 8: Contour Plot (Case 1b.2)

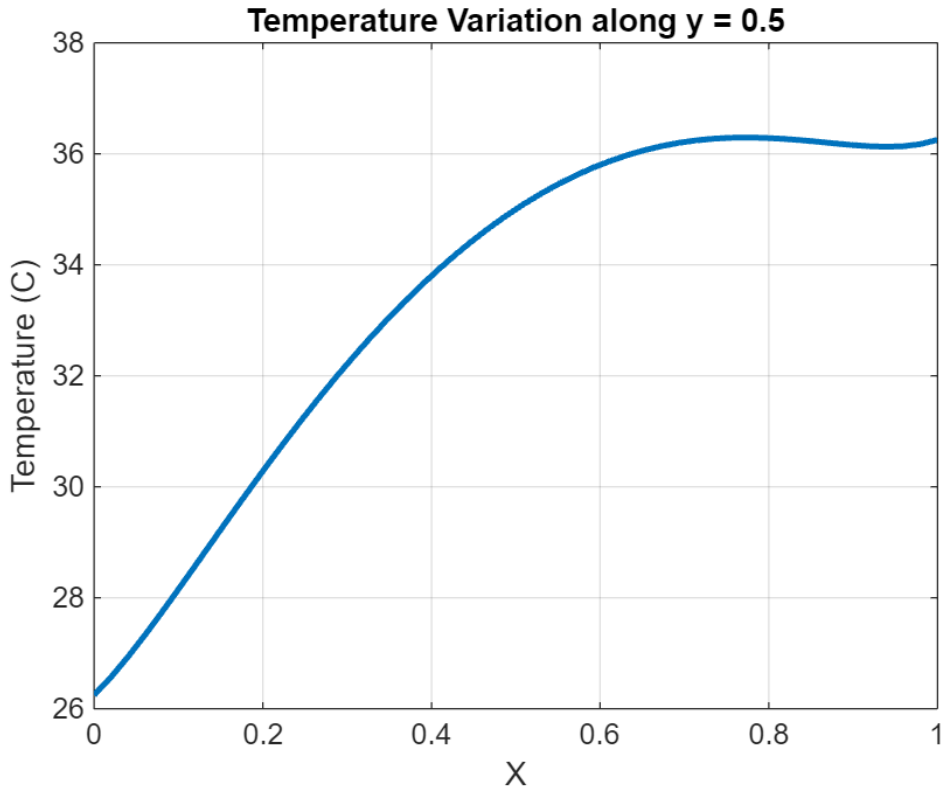


Figure 9: Temperature variation along the horizontal centreline (Case 1b.2)

c) using ~10 elements with heat source at the intersection of elements.

Mesh 1: Structured mesh

Note: 10 triangular elements are not achievable with this meshing approach, so taking 12 elements.

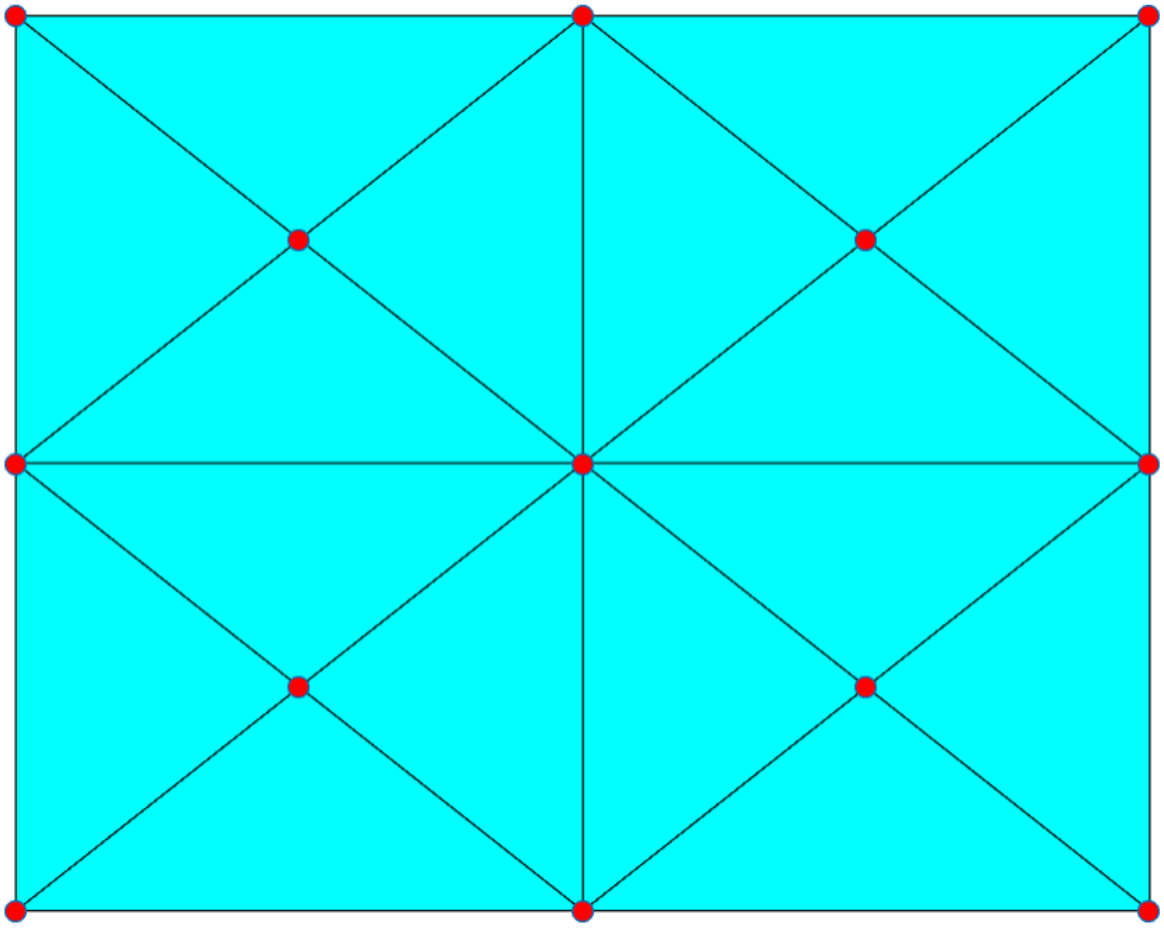


Figure 10: Mesh (Case 1c.1)

```
Temperature Distribution (C):
Node 1 (0,0): 25              Node 2 (0.5,0): 32.5
Node 3 (1,0): 35              Node 4 (0,0.5): 25
Node 5 (0.5,0.5): 37.5        Node 6 (1,0.5): 35
Node 7 (0,1): 25              Node 8 (0.5,1): 32.5
Node 9 (1,1): 35              Node 10 (0.25,0.25): 30
Node 11 (0.75,0.25): 35       Node 12 (0.25,0.75): 30
Node 13 (0.75,0.75): 35
```

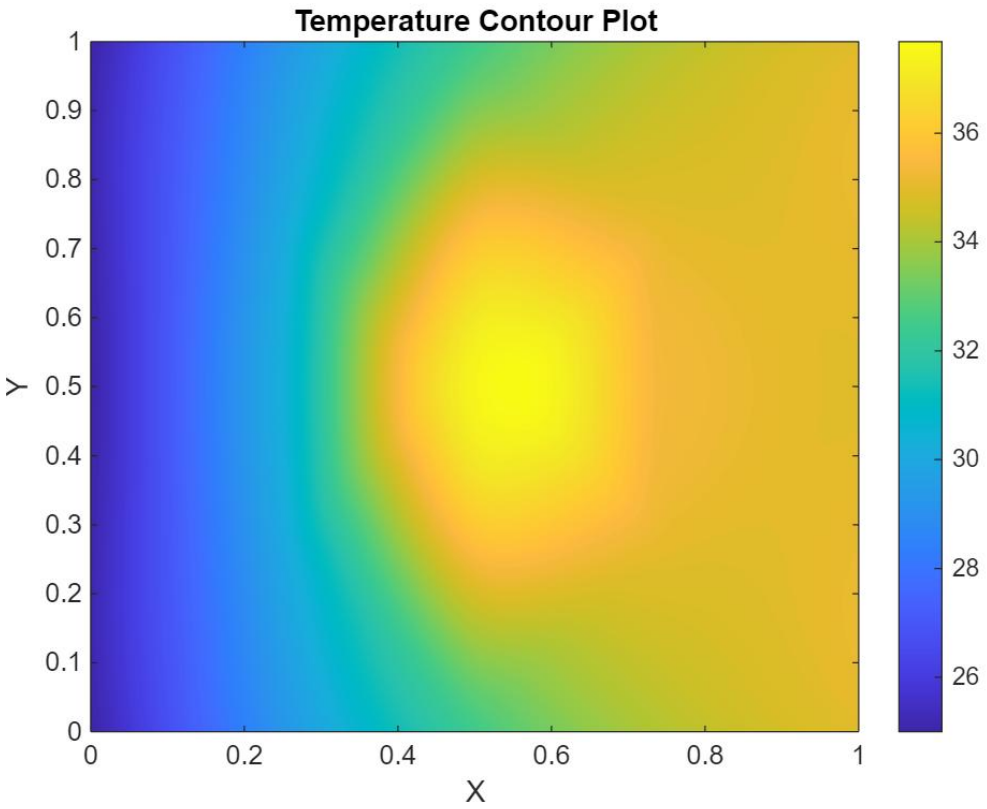


Figure 11: Contour Plot (Case 1c.1)

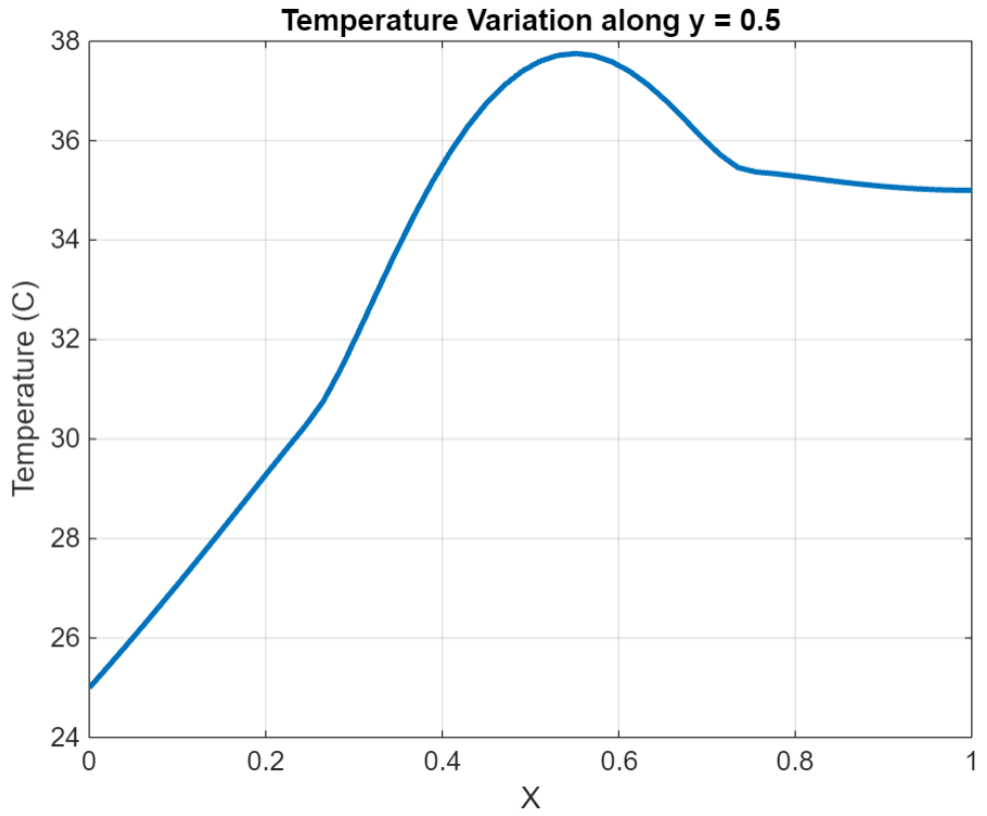


Figure 12: Temperature variation along the horizontal centreline (Case 1c.1)

Mesh 2: Elements intersecting at the centre.

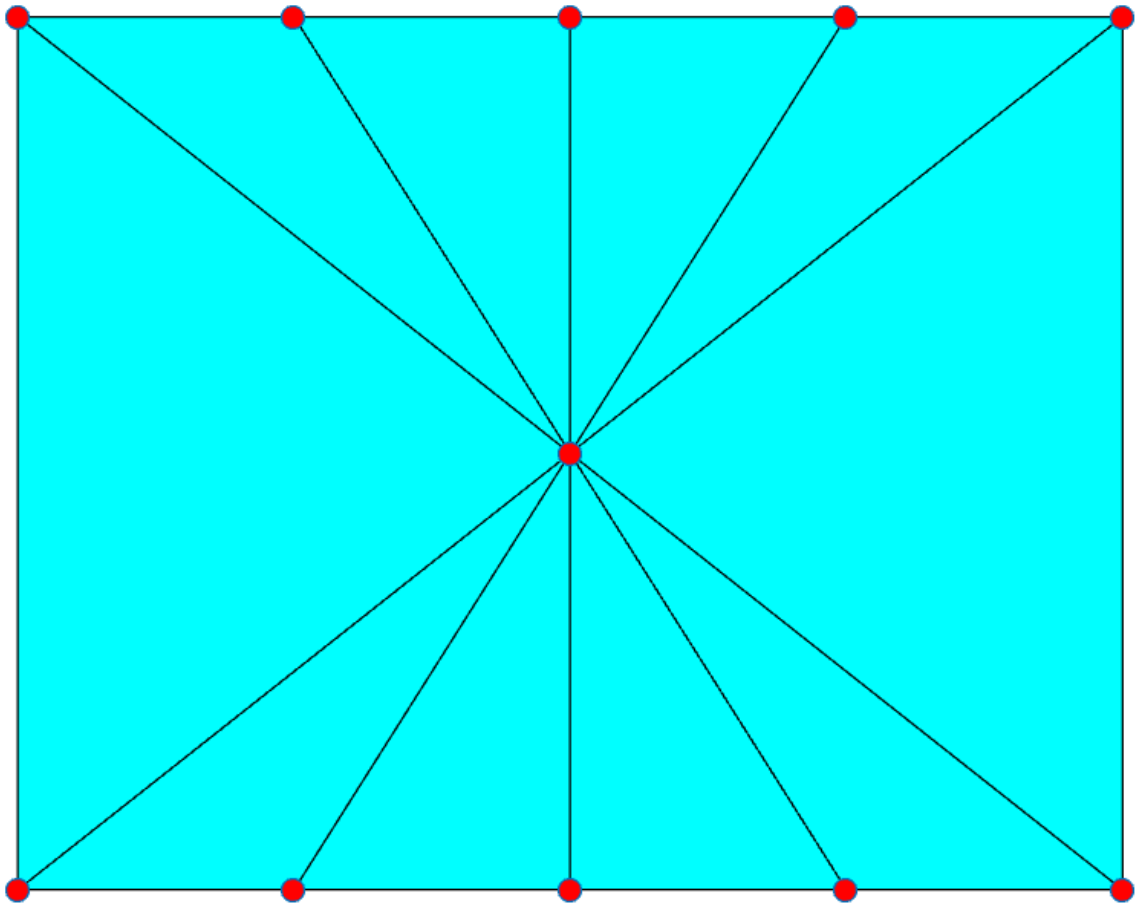


Figure 13: Mesh (Case 1c.2)

```
Temperature Distribution (C):
Node 1 (0,0): 25
Node 2 (0.25,0): 29.6875
Node 3 (0.5,0): 33.125
Node 4 (0.75,0): 34.6875
Node 5 (1,0): 35
Node 6 (1,1): 35
Node 7 (0.75,1): 34.6875
Node 8 (0.5,1): 33.125
Node 9 (0.25,1): 29.6875
Node 10 (0,1): 25
Node 11 (0.5,0.5): 36.875
```

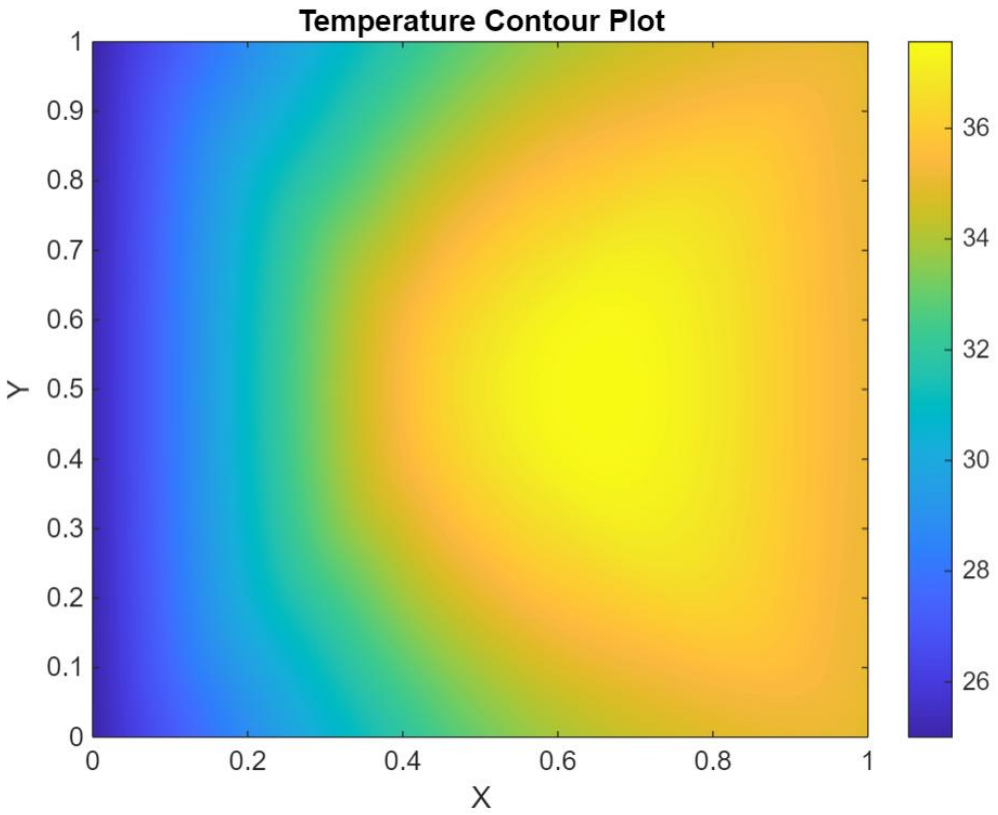


Figure 14: Contour Plot (Case 1c.2)

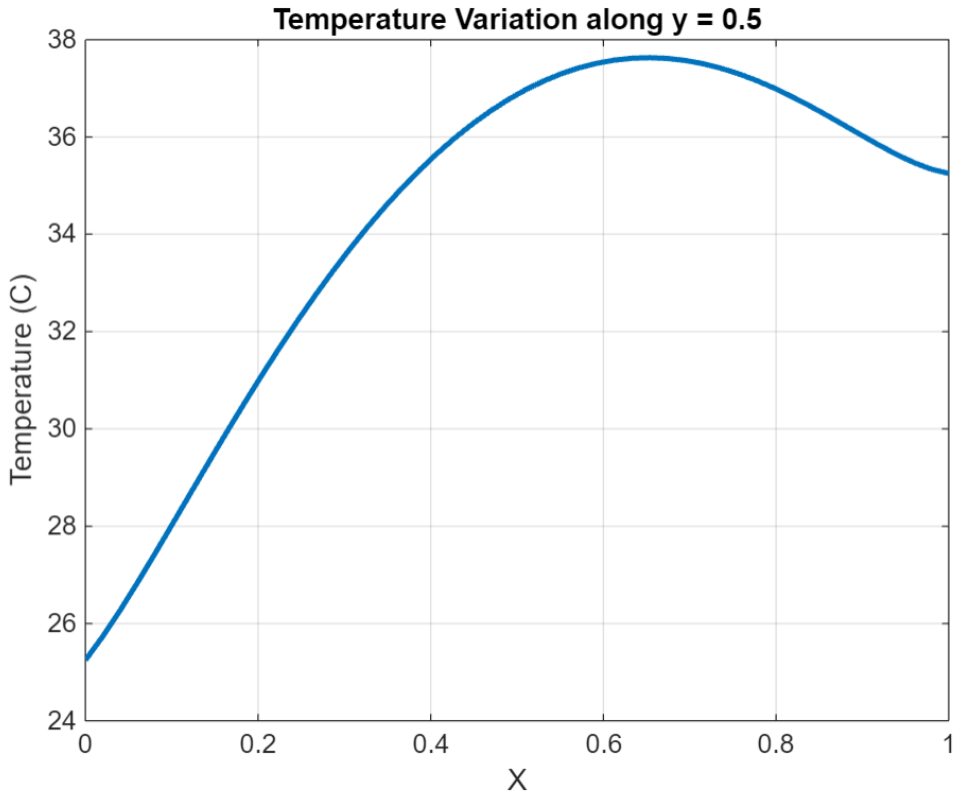


Figure 15: Temperature variation along the horizontal centreline (Case 1c.2)

d) using ~100 elements with heat source at the intersection of elements.

Mesh 1: Structured mesh

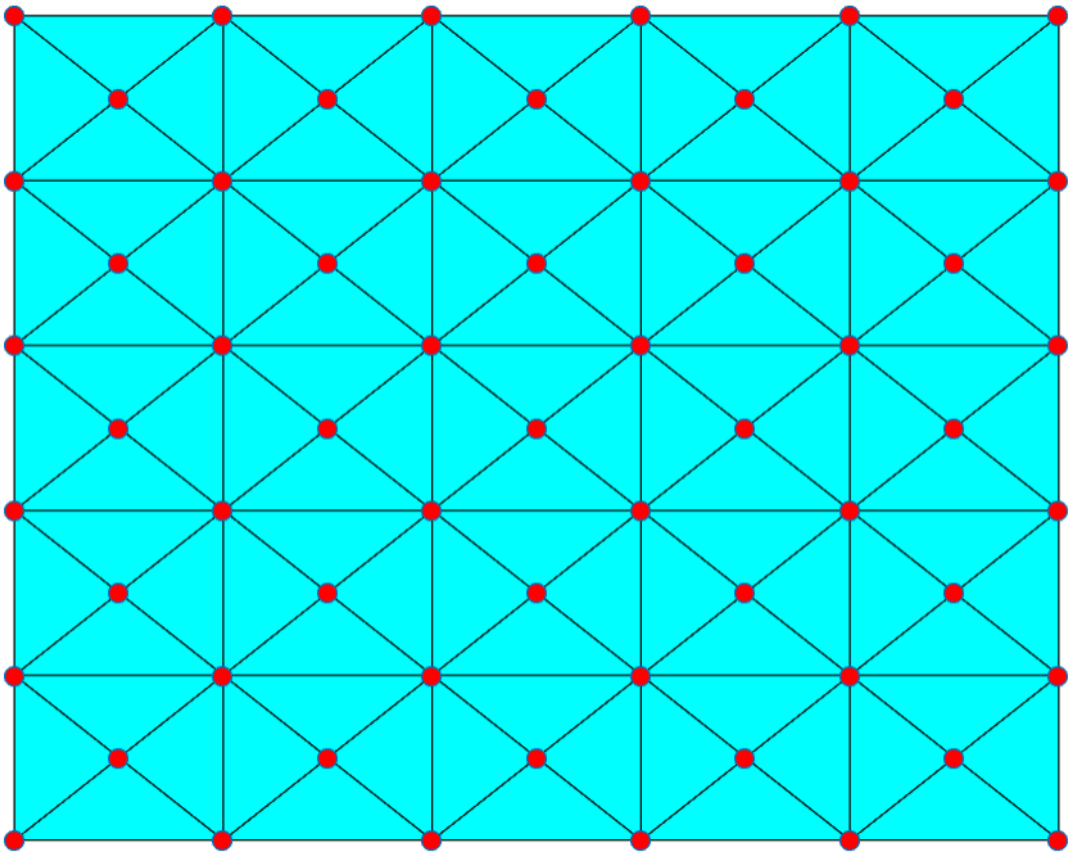


Figure 16: Mesh (Case 1d.1)

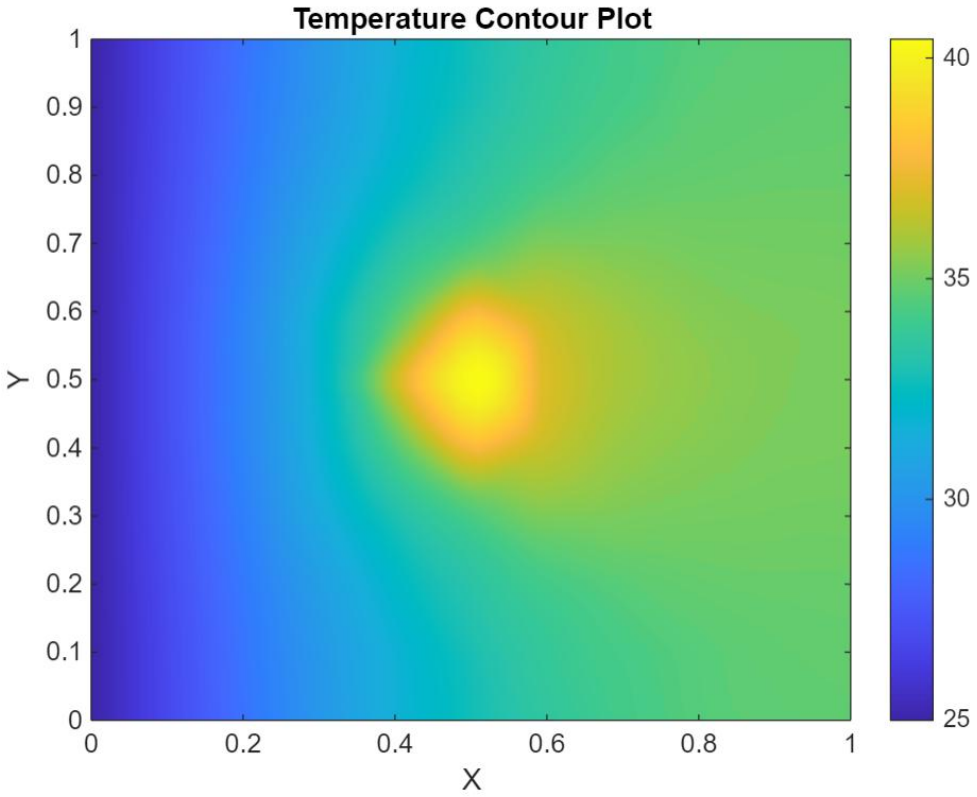


Figure 17: Contour Plot (Case 1d.1)

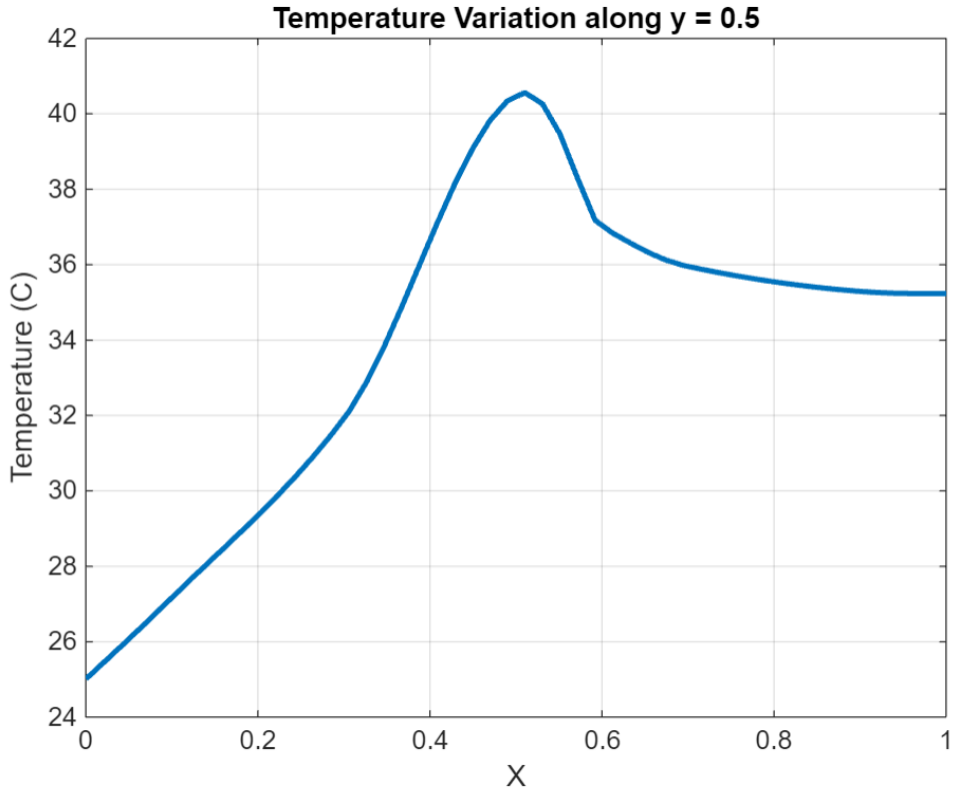


Figure 18: Temperature variation along the horizontal centreline (Case 1d.1)

Mesh 2: Elements intersecting at the centre.

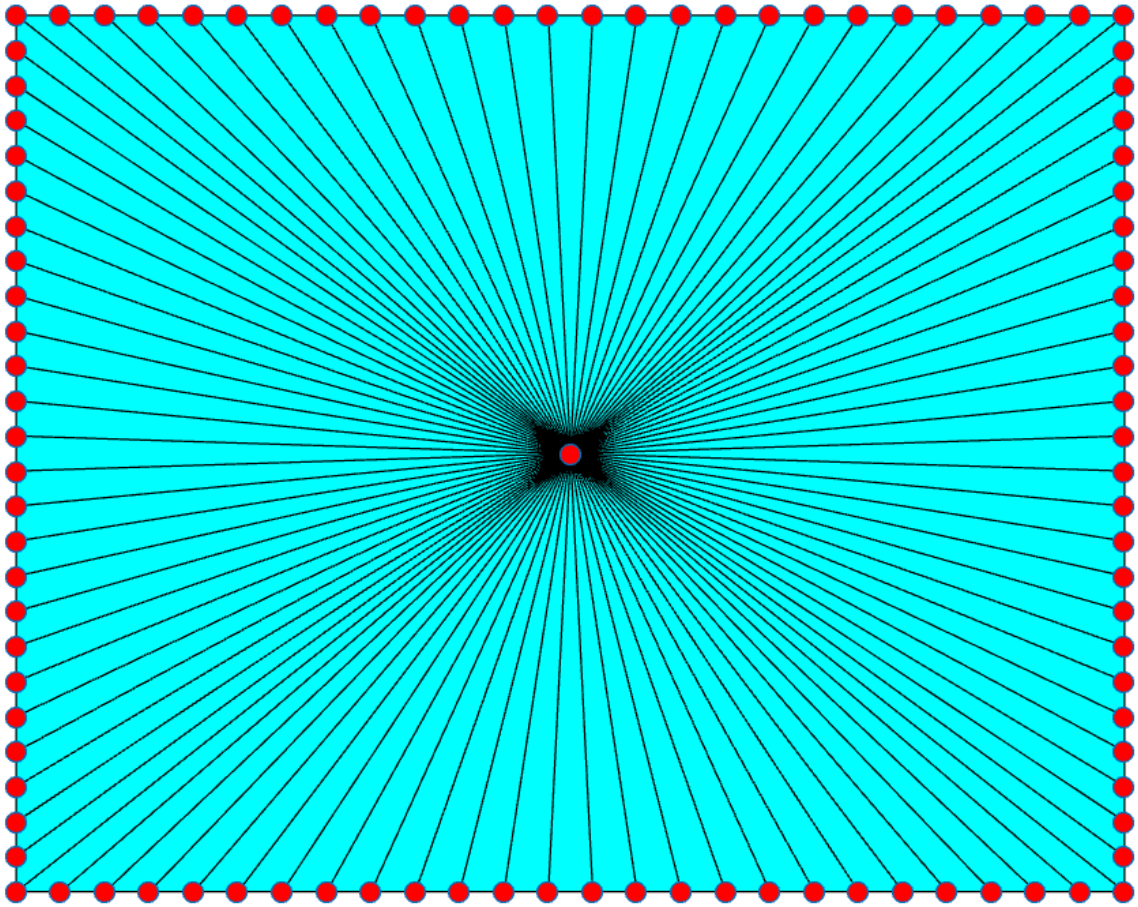


Figure 19: Mesh (Case 1d.2)

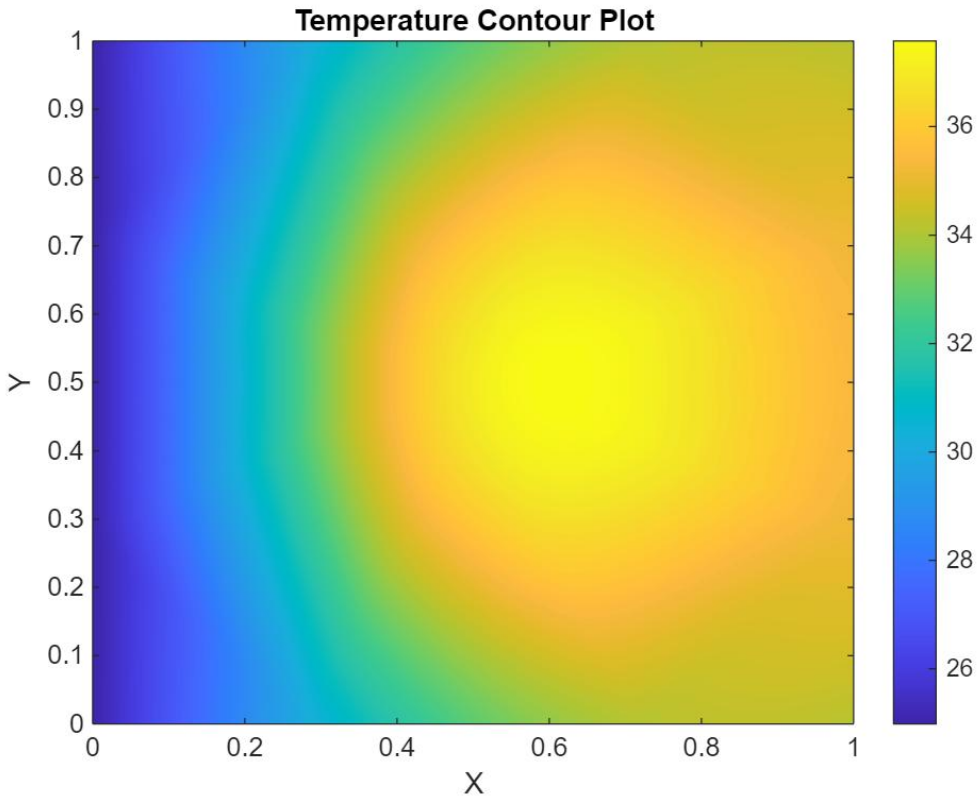


Figure 20: Contour Plot (Case 1d.2)

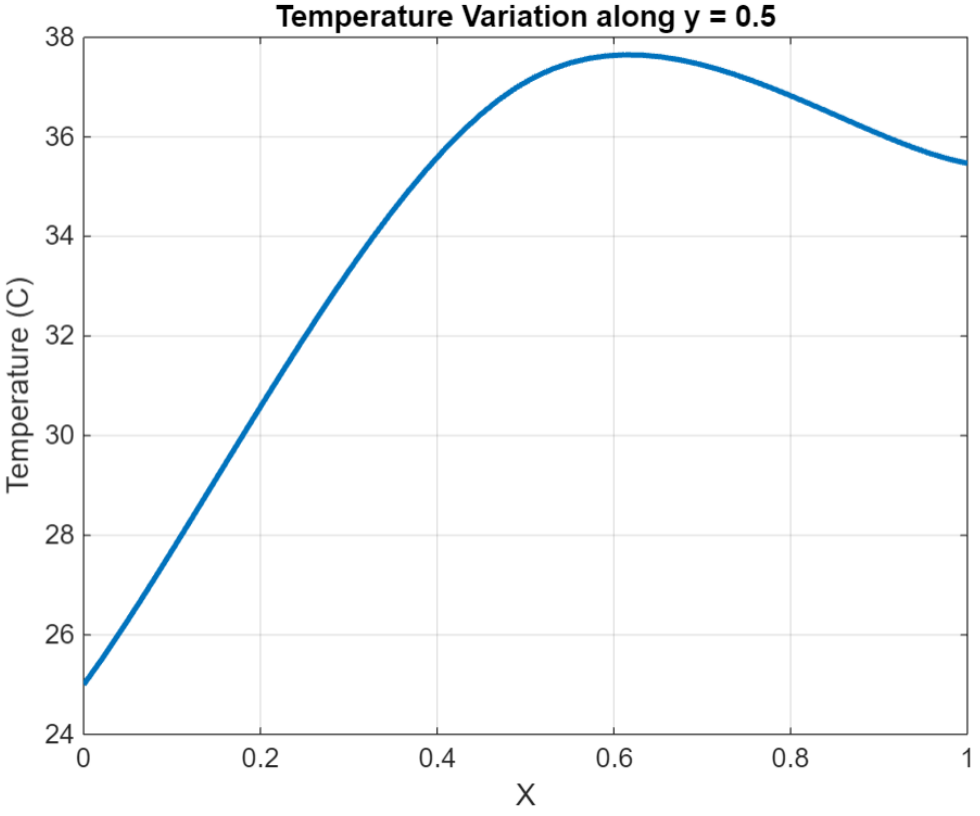


Figure 21: Temperature variation along the horizontal centreline (Case 1d.2)

Comment on accuracy of the different solutions:

a) The mesh with only two triangular elements provides a very coarse discretization, resulting in poor accuracy. The diagonal between nodes 2 and 3 introduces asymmetry, and the heat source applied as a Dirac delta function is not well represented. This leads to an uneven distribution of temperature and a solution that lacks both symmetry and physical accuracy.

b) The use of four triangular elements with a central node significantly improves the accuracy of the temperature distribution. The added node allows direct application of the force term and better captures temperature gradients, resulting in a smoother and more symmetric contour. This finer mesh offers a more realistic and numerically stable solution compared to the coarse two-element mesh.

c) Using 10 elements improves the solution significantly, providing better resolution and a more accurate temperature distribution. However, the solution has not yet fully converged, and minor discrepancies remain.

d) Increasing the mesh to 100 elements results in a much finer discretization, and the solution is observed to be nearly converged. The temperature field becomes smooth and stable, closely approximating the expected physical behaviour.

Case 2: When three sides of the plate are insulated, and the 4th side is subjected to convective heat transfer such that $qn = h(T - T0)$ where T0 = 25°C and h = 10 W/(m^2 -K). Solve using ~100 elements with heat source at the intersection of elements.

Mesh 1: Structured mesh

Step 1: *Define the geometry, mesh parameters, and material properties.*

Step 2: *Mesh generation.*

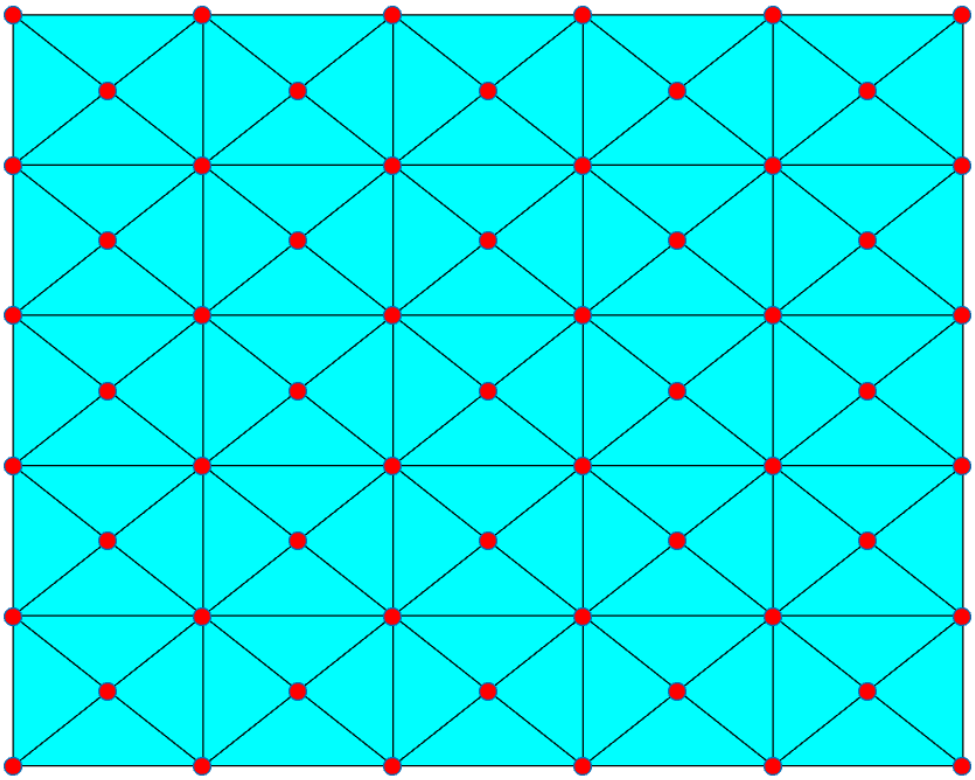


Figure 22: Mesh (Case 2.1)

Step 3: *Assembling the Global Stiffness matrix.*

First part of the assembly follows the same procedure as case 1, after which convection contributions are added to model boundary heat transfer. For each line segment on the left edge, the code calculates segment length, constructs a local convection matrix, and incorporates these scaled contributions into the global stiffness matrix. This addition effectively captures the heat exchange between the mesh boundary and the surrounding environment through convection.

```
% Stiffness matrix assembly
n = length(el);
K_global = zeros(length(p));


for i = 1:n
    nodes = el(:,i);
    x = p(1, nodes);
    y = p(2, nodes);

    J = [x(1)-x(3), x(2)-x(3); y(1)-y(3), y(2)-y(3)];
    A = 0.5 * abs(det(J));
    Q = [1, 0, -1; 0, 1, -1];
    K = k_val * A * (Q' / J) * (Q' / J)';

    for j = 1:3
        for k = 1:3
            K_global(nodes(j), nodes(k)) = K_global(nodes(j),
nodes(k)) + K(j,k);
        end
    end
end


% Extract the left edge of the mesh (nodes with x = 0)
left_edge = find(p(1,:) == 0);


[~, s_o] = sort(p(2, left_edge));
left_edge = left_edge(s_o);


% left edge elements
l_lines = zeros(2, length(left_edge)-1);
for i = 1:length(left_edge)-1
    l_lines(:,i) = [left_edge(i); left_edge(i+1)];
end


% Add convection contribution to the global stiffness matrix
for i = 1:size(l_lines, 2)
    node1 = l_lines(1,i);
    node2 = l_lines(2,i);
    x1 = p(1, node1); y1 = p(2, node1);
    x2 = p(1, node2); y2 = p(2, node2);
    L = sqrt((x2-x1)^2 + (y2-y1)^2);

    S = L/6 * [2 1; 1 2];
```

```
    K_global(node1, node1) = K_global(node1, node1) + h_val *
S(1,1);
    K_global(node1, node2) = K_global(node1, node2) + h_val *
S(1,2);
    K_global(node2, node1) = K_global(node2, node1) + h_val *
S(2,1);
    K_global(node2, node2) = K_global(node2, node2) + h_val *
S(2,2);
end
```

Step 4: *Assembling the Global Load vector.*

The load vector assembly process begins by initializing a zero vector for all mesh nodes, then adds a concentrated heat source of 1000 units at the domain centre (0.5, 0.5). Next, the code calculates convective heat flux contributions along the left boundary by processing each edge segment, determining its length, and creating local flux vectors scaled by the convection coefficient, external temperature, and segment length. These flux contributions are distributed equally between adjacent nodes and added to the global load vector.

```
% Load vector (heat source)
xc = 0.5;
yc = 0.5;
q = 1000;
q_global = zeros(size(p,2),1);


% Heat source at center
for i = 1:length(p)
    x = p(1,i);
    y = p(2,i);
    if x == 0.5 && y == 0.5
        q_global(i) = 1000;
        break;
    end
end


% Flux
F_conv = zeros(size(p,2), 1);
for i = 1:size(l_lines, 2)
    node1 = l_lines(1,i);
    node2 = l_lines(2,i);
    x1 = p(1, node1); y1 = p(2, node1);
    x2 = p(1, node2); y2 = p(2, node2);
    L = sqrt((x2-x1)^2 + (y2-y1)^2);
```

```
    F_vec = h_val * T_val * L/ 2 * [1; 1];

    F_conv(node1) = F_conv(node1) + F_vec(1);
    F_conv(node2) = F_conv(node2) + F_vec(2);
end
```

Step 5: *Solving the System.*

Finally, the complete system of equations is solved by dividing the combined load vector (internal heat source plus boundary convection) by the global stiffness matrix, yielding temperature values at all nodes in the mesh.

```
% Solve
q_global = q_global + F_conv;
T = K_global \ q_global;
```

Step 6: *Plotting Results*

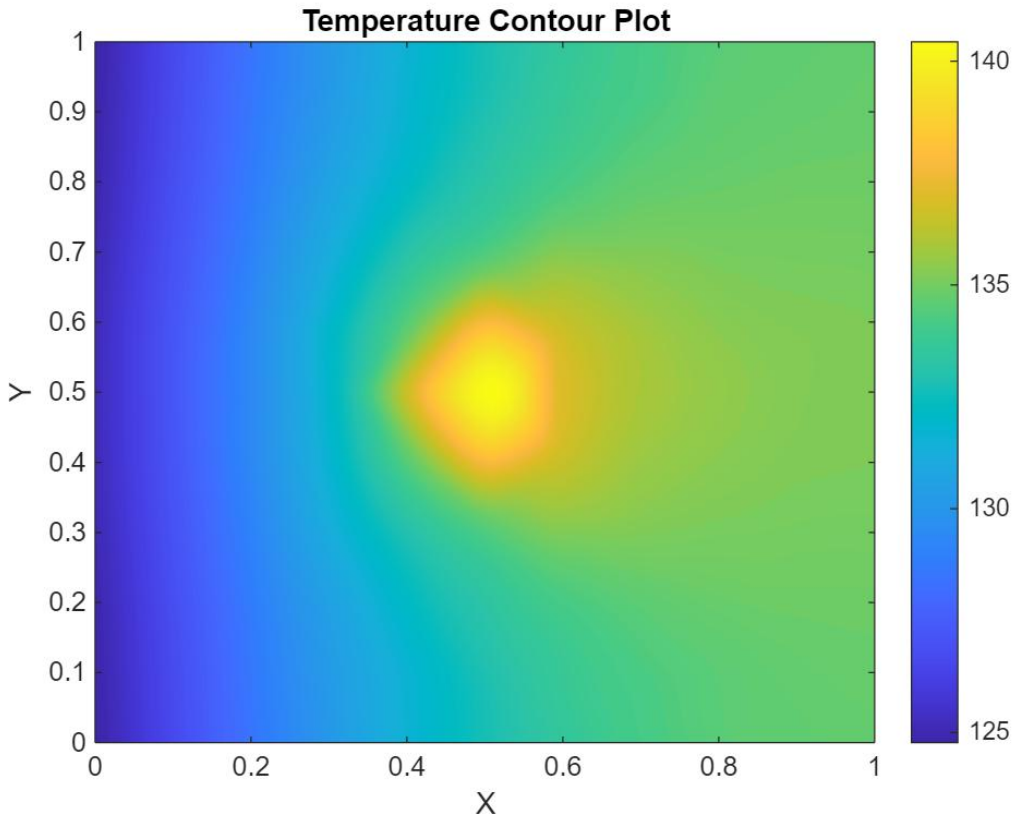


Figure 23: Contour Plot (Case 2.1)

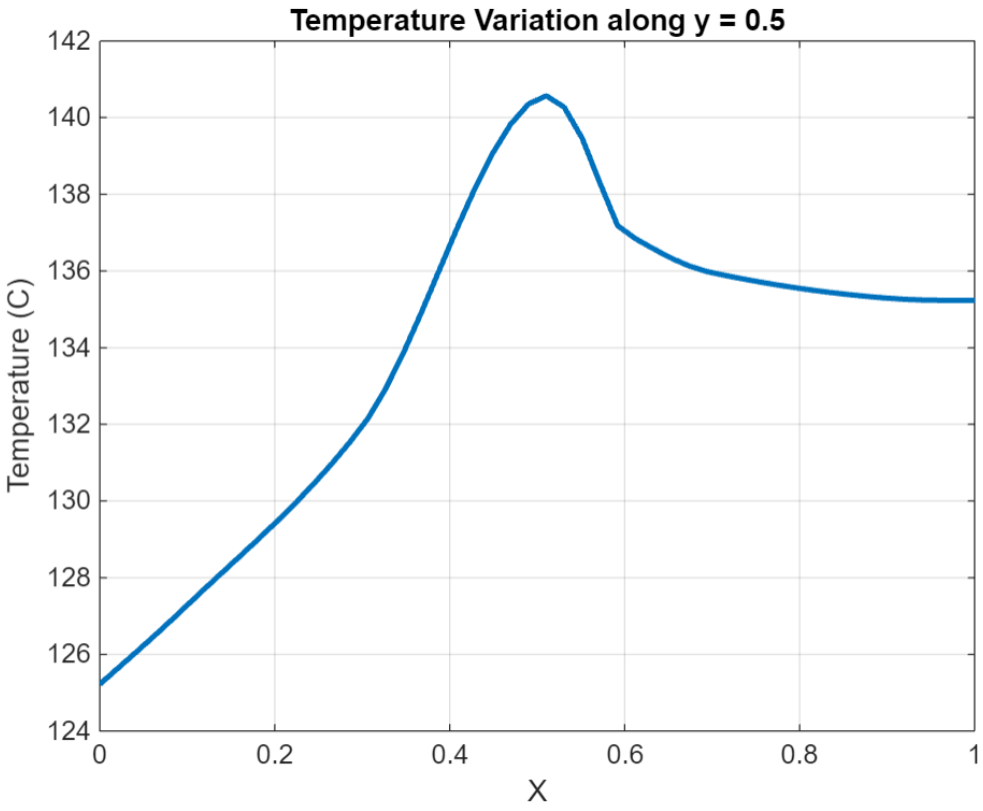


Figure 24: Temperature variation along the horizontal centreline (Case 2.1)

Mesh 2: Elements intersecting at the centre.

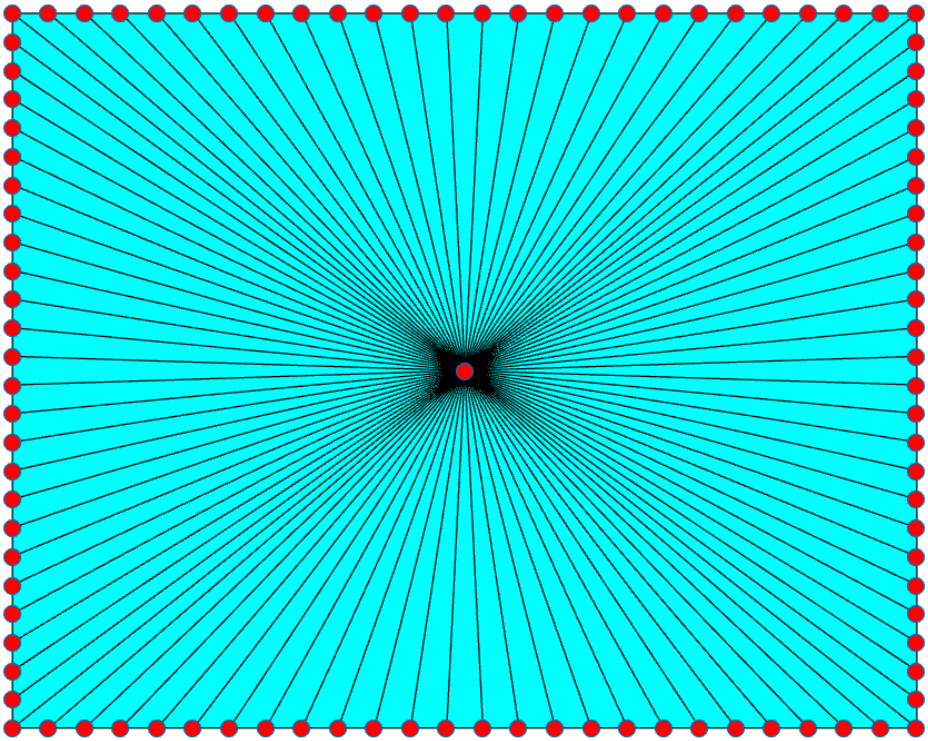


Figure 25: Mesh (Case 2.2)

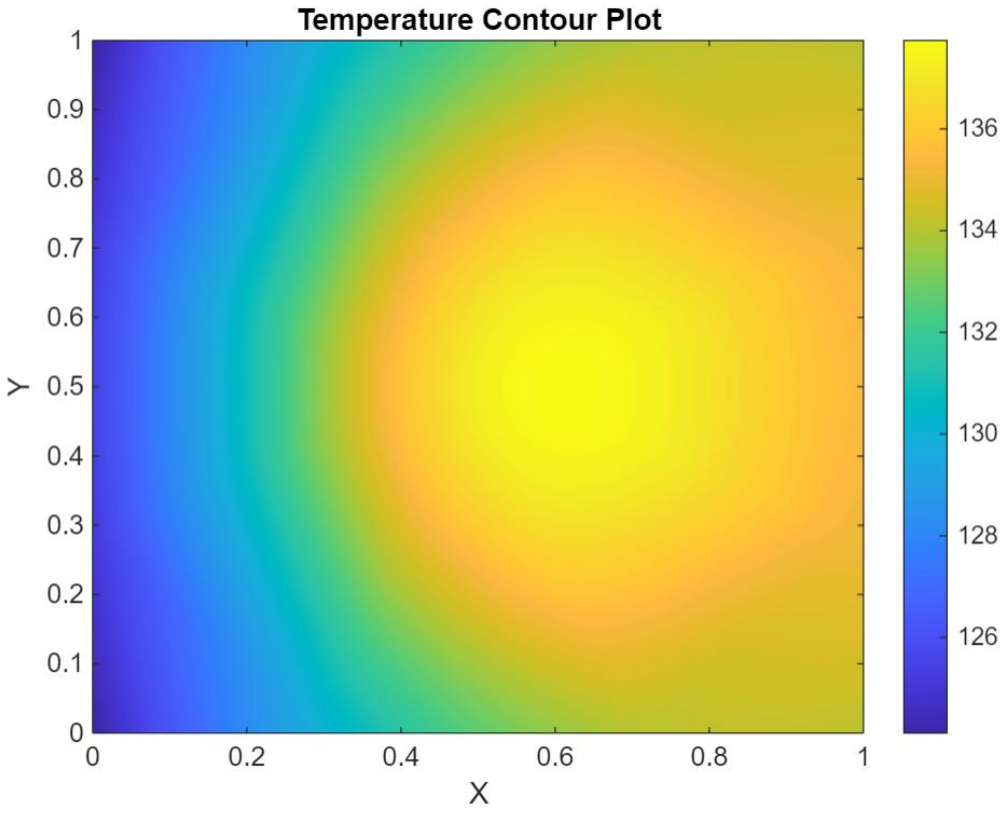


Figure 26: Contour Plot (Case 2.2)

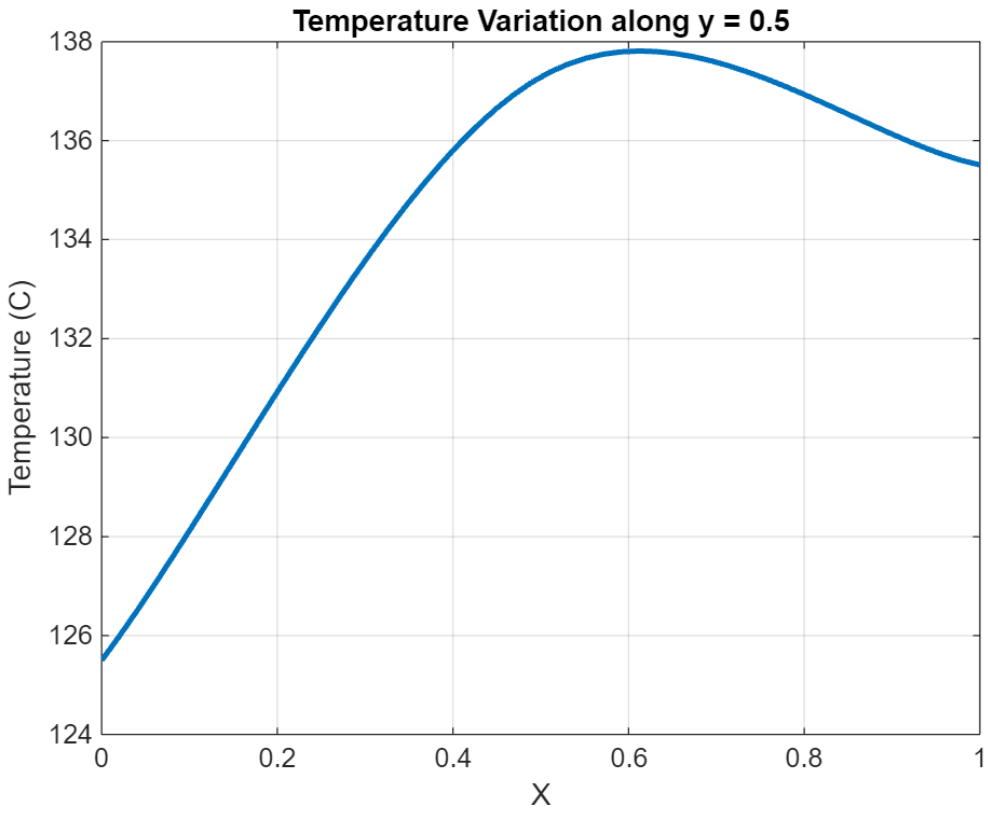
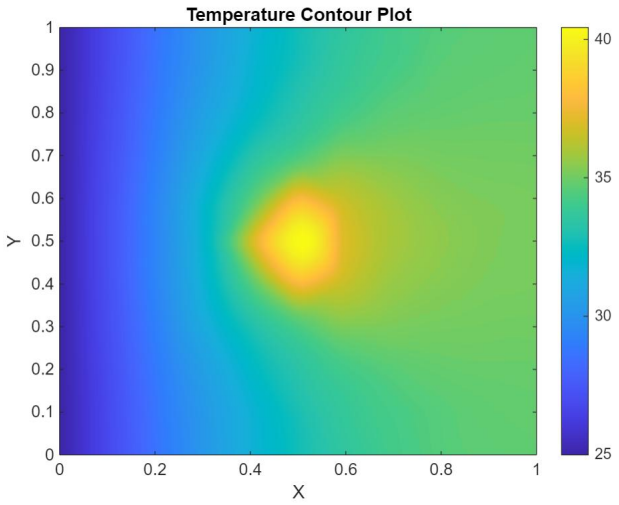


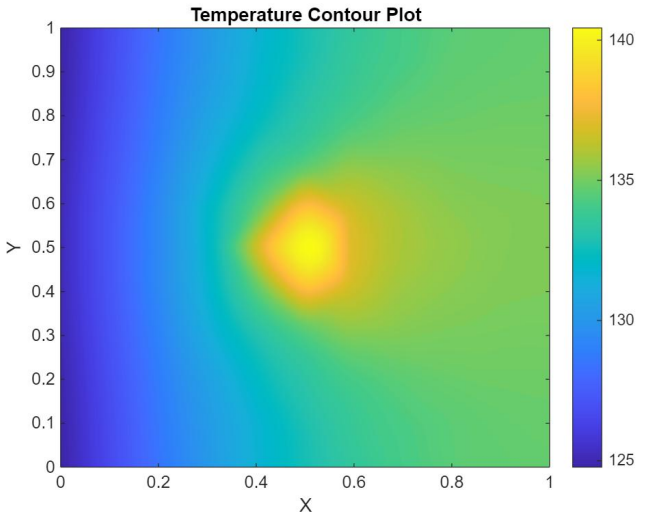
Figure 27: Temperature variation along the horizontal centreline (Case 2.2)

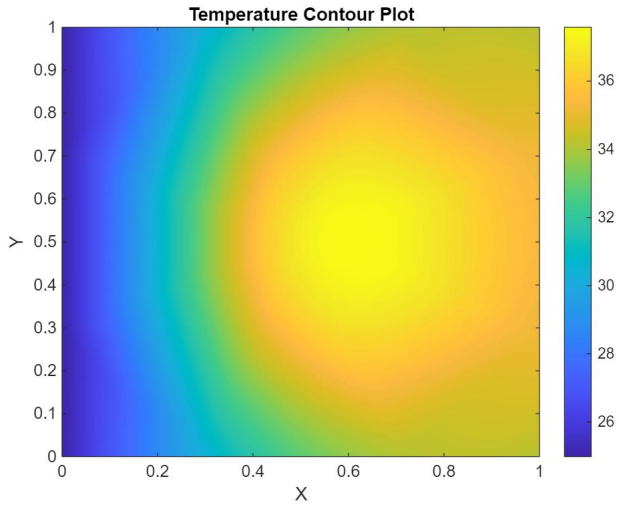
Comparing case 1.d and case 2 (both mesh types)

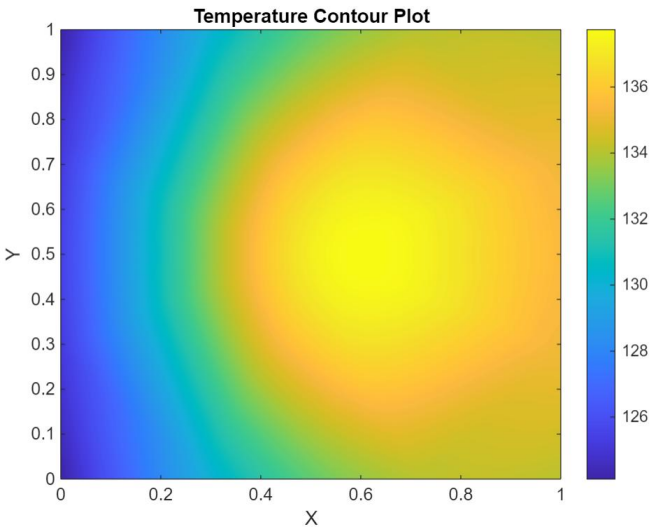
Case 1d



Case 2







Note: Row 1 are the results of Structured Mesh and Row 2 are the results of mesh with Elements intersecting at the centre.

Observation:

The thermal distribution pattern is identical in both cases; however, case 2 consistently exhibits temperatures exactly 100 °C higher than those in case 1d throughout the entire domain. This uniform offset is observed across both the Structured Mesh and Elements Intersecting at the Centre methods, confirming the robustness of the solution approach.