Bauhaus-Universität Weimar **Project**: Finite Difference Method for stationary 2D heat conduction problem

Finite Difference Method for stationary 2D heat conduction problem

Group: 1

By **Ankit Anand** (*126774*), **Ajay Satish Patil** (*126655*), **Syed Wali Ahmad Rizvi** (*126657*)

Published on January *25, 2024*

Contents

1. Introduction

The assignment requires students to develop a numerical solution for a specified 2D heat conduction problem using computational software like MATLAB, Maple, or Octave. Key tasks include creating a finite difference term for the heat conduction differential equation, implementing the solution for variable increments Δx and Δy in a 2.0m x 1.0m rectangular region, analyzing the temperature field under given boundary conditions, and producing color contour plots of the temperature distribution. The project also involves submitting a detailed report and all software code files, emphasizing the practical application of FDM in modeling heat conduction in real-world scenarios.

2. Methodology

2.1 Finite Difference Method

The finite difference method (FDM) is a numerical technique used to approximate solutions to differential equations. By discretizing the domain, the 2D heat conduction equation was transformed into a set of algebraic equations. These equations were then used to estimate the temperature distribution within the domain.

2.2 Understanding the Problem

First, let's understand the physical problem we are dealing with. We have a 2D rectangular region (2.0m x 1.0m) where heat conduction occurs. Our goal is to find the temperature distribution within this region under given boundary conditions and material properties.

2.3 Heat Flux

For the heat flux boundary, use Fourier's law of heat conduction, which relates heat flux Q to the temperature gradient. The discretized form of Fourier's law at a boundary is:

$$Q = -k\frac{\Delta T}{\Delta x}$$

Where k is the thermal conductivity, ΔT is the temperature difference across a small distance Δx, and Q is the heat flux.

Rearranging for ΔT:

$$\Delta T = -\frac{Q\Delta x}{k}$$

2.4 Governing Equation

We are solving a stationary 2D heat conduction problem. The governing equation for 2D heat conduction without internal heat generation in a Cartesian coordinate system is given by:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) = 0$$

where T is the temperature, and k is the thermal conductivity.

For stationary 2D heat conduction, the governing equation is typically a second-order partial differential equation (PDE) given by:

$$k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) = 0$$

2.5 Finite Difference Approximation

Next, we apply the Finite Difference Method to approximate the second derivatives in the governing equation. The central difference approximation is commonly used:

$$k(\frac{\partial^2 T}{\partial x^2})|_{(i,j)} \approx \frac{T_{i+1,j}-2T_{i,j}+T_{i-1,j}}{(\Delta x)^2}$$
$$k(\frac{\partial^2 T}{\partial y^2})|_{(i,j)} \approx \frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{(\Delta y)^2}$$

Substituting these approximations into the governing equation, we obtain a set of algebraic equations for each interior grid point:

$$k(\frac{T_{i+1,j}-2T_{i,j}+T_{i-1,j}}{(\Delta x)^2} + \frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{(\Delta y)^2}) = 0$$

Let us assume equal grid spacing in both directions, Δx = Δy, and a uniform thermal conductivity k.

For K W/(mK) and equal spacing Δx = Δy, the equation becomes:

$$0 = k(\frac{T_{i+1,j}-2T_{i,j}+T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{\Delta x^2})$$

Assemble the System of Equations

Arrange these equations in a matrix form Ax = b, where:

• A is the coefficient matrix.

• x is the vector of unknown temperatures at each node.

• b is the right-hand side vector, which incorporates boundary conditions.

Each row in the matrix corresponds to an equation for a node. For an interior node (i, j), the corresponding row will have:

• Coefficients for $T_{i+1,j}$, $T_{i-1,j}$, $T_{i,j+1}$, and $T_{i,j-1}$ based on the finite difference equation.

• A coefficient for $T_{i,j}$ (typically negative due to the −2 in the finite difference approximation).

• A zero or specified value (from boundary conditions) in the corresponding entry of the vector b.

Solving the System Using Gauss-Seidel Method:

1. Set Initial Guess: Initialize the temperature at each node. For simplicity, we can start with all temperatures set to zero. Let's denote the temperature at node (i, j) by $T^{(h)}_{i,j}$, where h is the iteration number. Initially, $T^{(0)}_{i,j} = 0$ for all i, j.

Updating Temperatures: In each iteration, update the temperature at each node based on the latest available values of neighbouring nodes. The updating formula for node (i, j) is derived from the discretized heat equation:

$$T_{i,j}^{(h+1)} = \frac{1}{2(\frac{h}{\Delta x^2}+\frac{h}{\Delta x^2})}(\frac{h}{\Delta x^2}(T_{i+1,j}^{(h)} + T_{i-1,j}^{(h)}) + \frac{h}{\Delta x^2}(T_{i,j+1}^{(h)} + T_{i,j-1}^{(h+1)}))$$

This equation incorporates the latest values from the east and north neighbours $T^{(h)}_{i+1,j}$ and $T^{(h)}_{i,j+1}$ from the h-th iteration and the west and south neighbours $T^{(h)}_{i-1}$,j and $T^{(h+1)}_{i,j-1}$ from the (h + 1)-th iteration.

After updating all nodes in the domain, checking if the solution has converged. This can be done by checking if the maximum change in temperature across all nodes between two successive iterations is below a specified threshold ϵ:

$$\max_{i,j} |T_{i,j}^{(h+1)} - T_{i,j}^{(h)}| < \epsilon$$

If this condition is not met, repeat the iteration.

Updating the temperature at each grid point using the latest available values. For each iteration h and for each grid point (i, j), updating $T_{i,j}$ as:

$$T_{i,j}^{(h+1)} = \frac{1}{4}(T_{i+1,j}^{(h)} + T_{i-1,j}^{(h)} + T_{i,j+1}^{(h)} + T_{i,j-1}^{(h+1)})$$

In this scheme:

• $T^{(h+1)}_{i-1,j}$ and $T^{(h+1)}_{i,j-1}$ are the temperatures at the west and south neighbours, updated in the current iteration.

• $T^{(h)}_{i+1,j}$ and $T^{(h)}_{i,j+1}$ are the temperatures at the east and north neighbours, from the previous iteration. We can rewrite the above equation as below for $T_{i,j}$:

$$T_{i,j} = \frac{1}{4}(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1})$$

2.6 Implementation in MATLAB

MATLAB was chosen for its robust numerical computation capabilities. The domain, a rectangle of size 2.0m x 1.0m, was discretized into a grid with variable increments Δx and Δy. The code iteratively solved the discretized equations until the temperature distribution reached a steady state

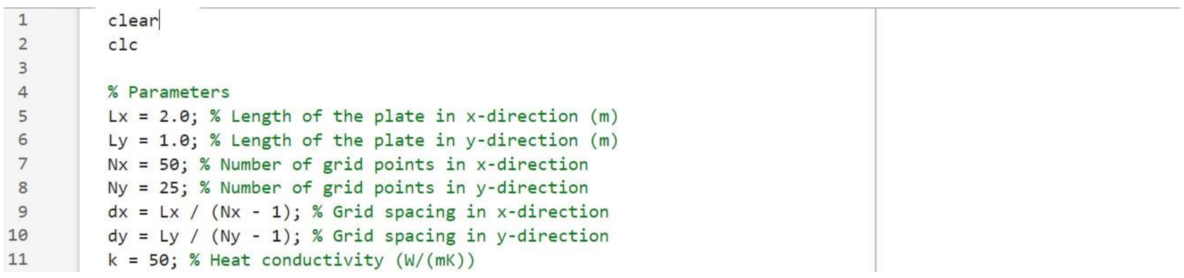
2.6.1 Setting Parameters

• Defining the plate dimensions: Lx and Ly specify the length and width of the plate.

• Setting grid resolution: Nx and Ny determine the number of points for the temperature grid.

• Calculating grid spacing: dx and dy are the distances between grid points.

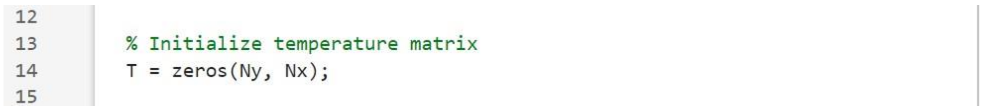
• Heat conductivity: k is the thermal conductivity of the plate material.



```matlab
clear
clc

% Parameters
Lx = 2.0; % Length of the plate in x-direction (m)
Ly = 1.0; % Length of the plate in y-direction (m)
Nx = 50; % Number of grid points in x-direction
Ny = 25; % Number of grid points in y-direction
dx = Lx / (Nx - 1); % Grid spacing in x-direction
dy = Ly / (Ny - 1); % Grid spacing in y-direction
k = 50; % Heat conductivity (W/(mK))
```

2.6.2 Initializing Temperature Matrix

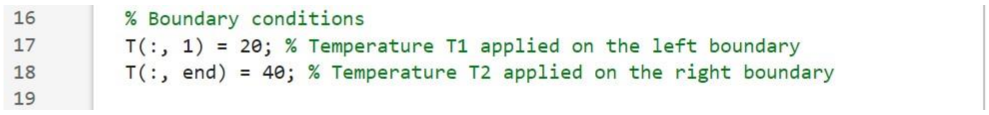
• Creating temperature grid: Initializes a matrix T with zeros, representing the initial temperature at each grid point.



```matlab

% Initialize temperature matrix
T = zeros(Ny, Nx);

```

2.6.3 Applying Boundary Conditions

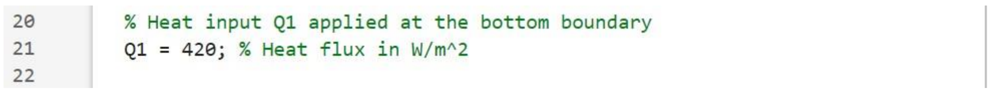
• Setting fixed temperatures: Assigns temperatures to the left (20°C) and right (40°C) boundaries of the plate.



```
% Boundary conditions
T(:, 1) = 20; % Temperature T1 applied on the left boundary
T(:, end) = 40; % Temperature T2 applied on the right boundary

```

2.6.4 Setting Heat Input

• Bottom boundary heat input: Q1 specifies the heat flux applied to the bottom boundary.

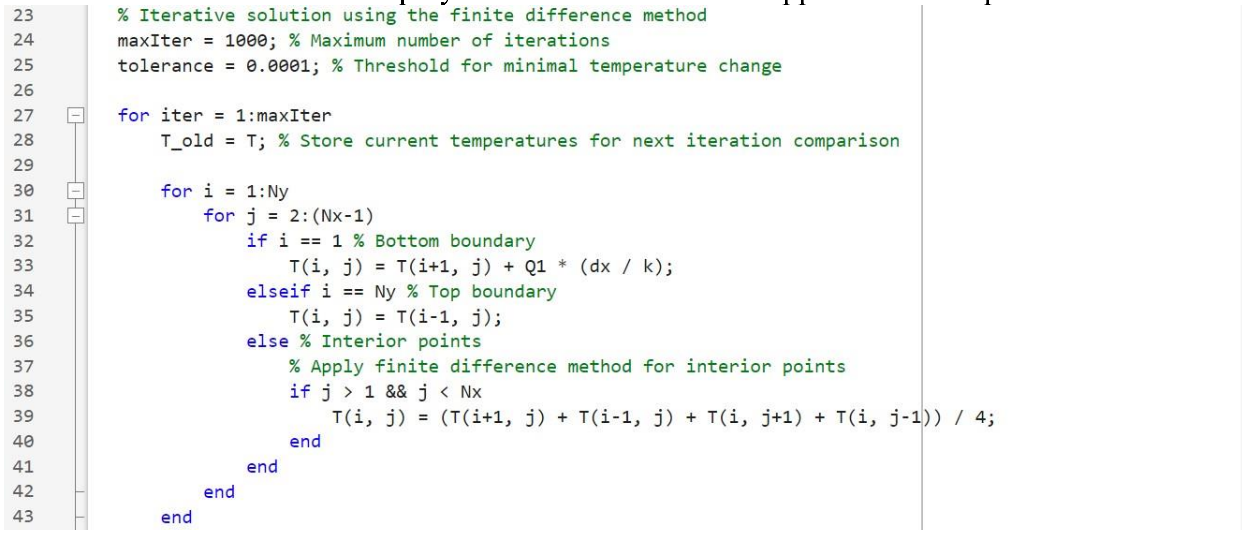


```
% Heat input Q1 applied at the bottom boundary
Q1 = 420; % Heat flux in W/m^2

```

2.6.5 Iterative Solution with Finite Difference Method

• Iterative process: Repeatedly updates the temperature grid until steady state is reached.

• Finite difference method: Employs a numerical method to approximate temperature distribution.

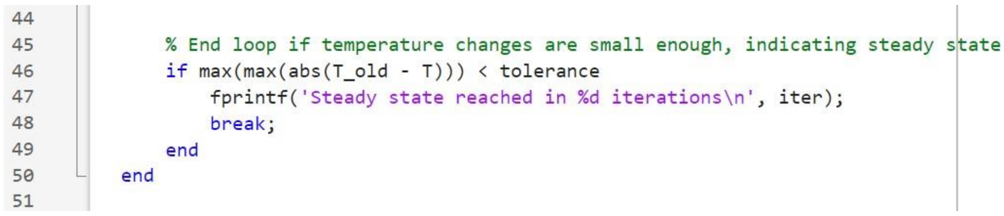


```
% Iterative solution using the finite difference method
maxIter = 1000; % Maximum number of iterations
tolerance = 0.0001; % Threshold for minimal temperature change

for iter = 1:maxIter
    T_old = T; % Store current temperatures for next iteration comparison

    for i = 1:Ny
        for j = 2:(Nx-1)
            if i == 1 % Bottom boundary
                T(i, j) = T(i+1, j) + Q1 * (dx / k);
            elseif i == Ny % Top boundary
                T(i, j) = T(i-1, j);
            else % Interior points
                % Apply finite difference method for interior points
                if j > 1 && j < Nx
                    T(i, j) = (T(i+1, j) + T(i-1, j) + T(i, j+1) + T(i, j-1)) / 4;
                end
            end
        end
    end
```

2.6.6 Checking for Steady State

• Convergence check: Determines if the temperature changes are below a set threshold (tolerance), indicating that the system has reached steady state.

• Iteration report: Outputs the number of iterations taken to reach steady state.



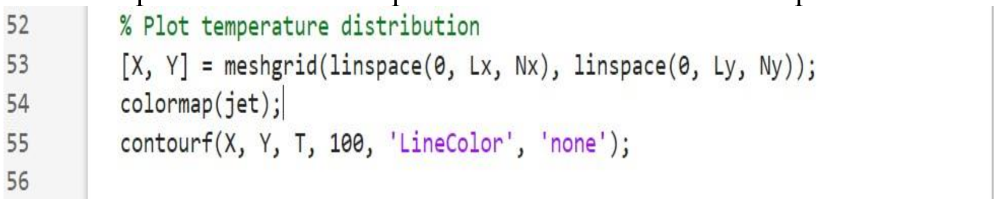
```

    % End loop if temperature changes are small enough, indicating steady state
    if max(max(abs(T_old - T))) < tolerance
        fprintf('Steady state reached in %d iterations\n', iter);
        break;
    end
end

```

2.6.7 Plotting Temperature Distribution

• Generating coordinate grid: Creates a mesh grid for plotting.

• Setting the color map: colormap(jet) applies the ’jet’ color map, which is a range of colors from blue to red, to visualize different temperature levels more clearly.

• Contour plot: Visualizes temperature distribution across the plate.



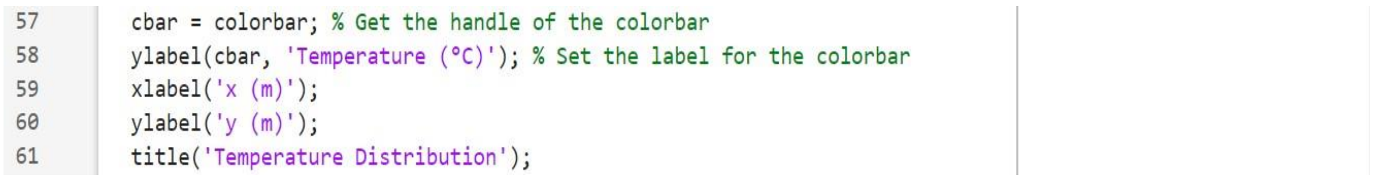
```
% Plot temperature distribution
[X, Y] = meshgrid(linspace(0, Lx, Nx), linspace(0, Ly, Ny));
colormap(jet);
contourf(X, Y, T, 100, 'LineColor', 'none');

```

2.6.8 Enhancing the Plot

• Colorbar addition: Adds a colorbar to the plot to represent temperature scales.

• Axis labeling: Labels the x and y axes with units.

• Title: Adds a title to the plot for context.



```
cbar = colorbar; % Get the handle of the colorbar
ylabel(cbar, 'Temperature (°C)'); % Set the label for the colorbar
xlabel('x (m)');
ylabel('y (m)');
title('Temperature Distribution');
```

3. Results

3.1 Temperature Distribution

The simulation resulted in a color contour plot that visually represented the temperature distribution within the domain. The plot showed a clear gradient from the hotter to the cooler side, aligned with the prescribed boundary conditions. The temperature increased from left to right, influenced by the imposed heat flux at the top boundary.

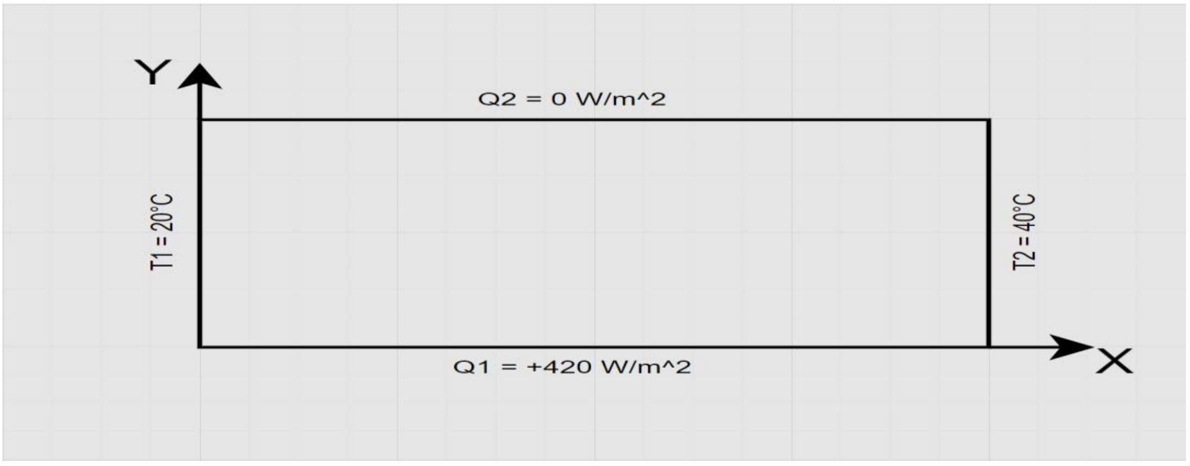


Figure 1: Boundary Conditions for 2D Heat Conduction Problem

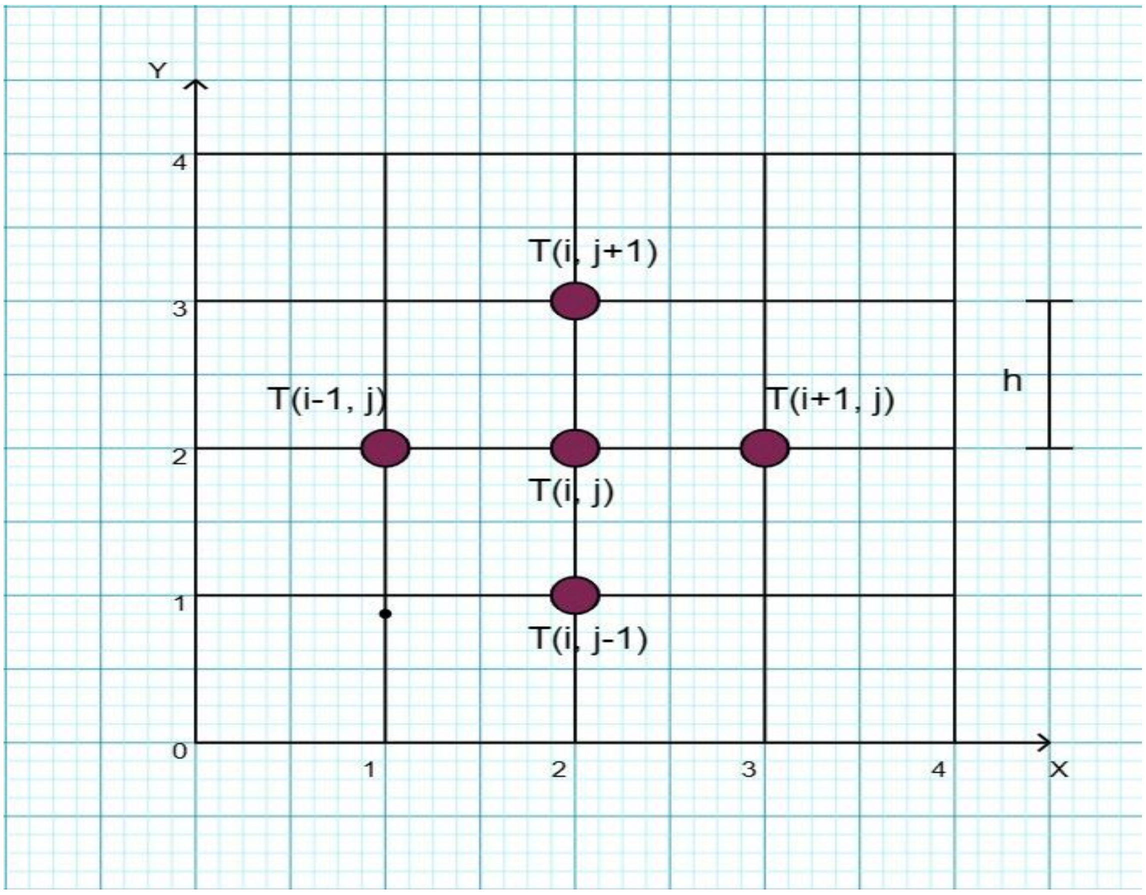
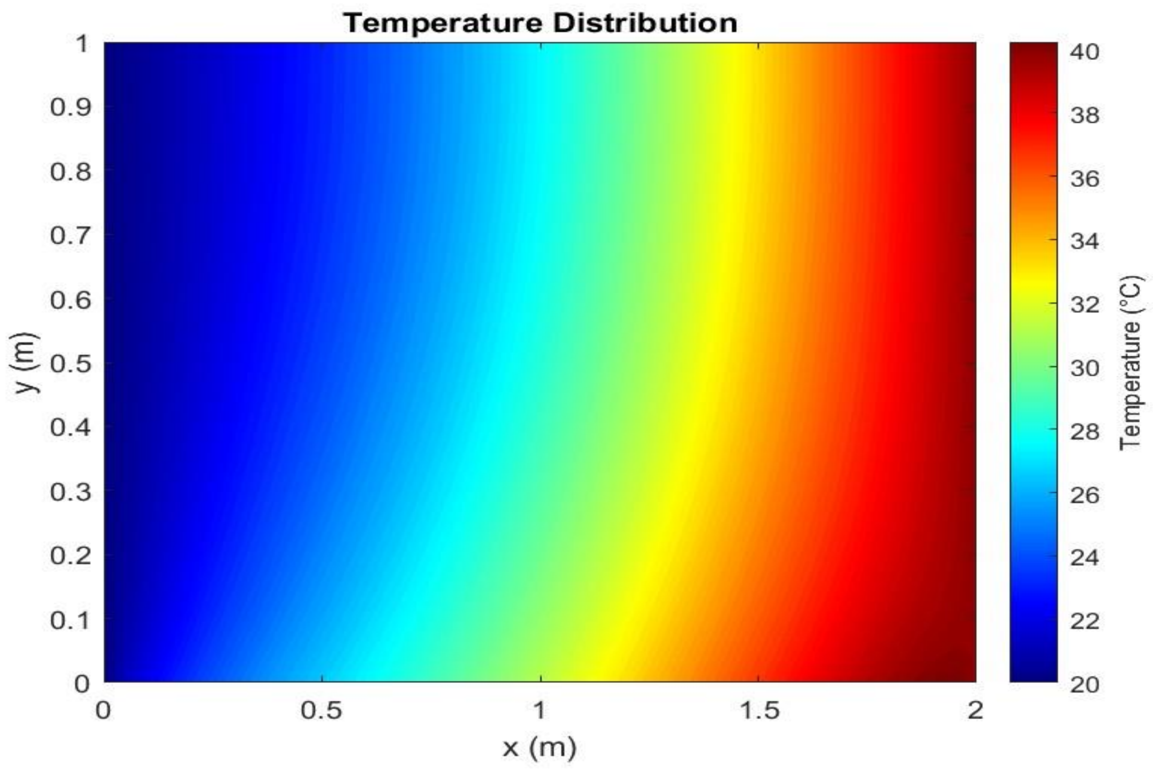


Figure 2: Finite Difference Grid for 2D Heat Equation

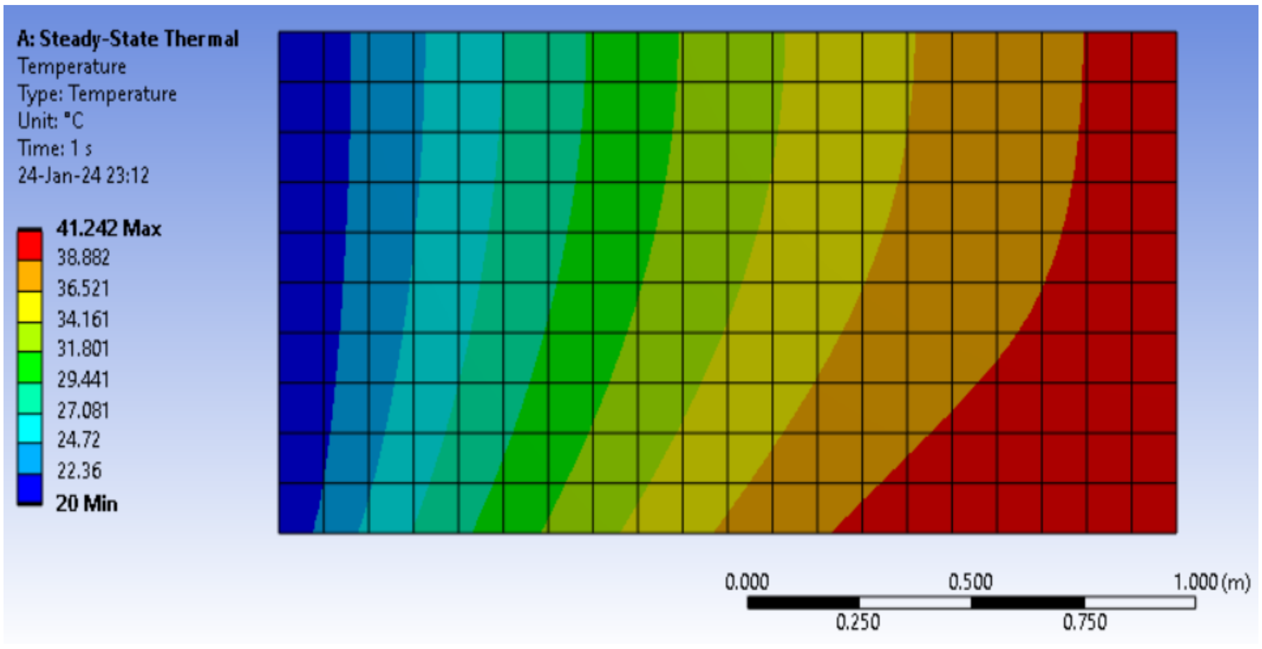


Temp Distribution graph by MATLAB

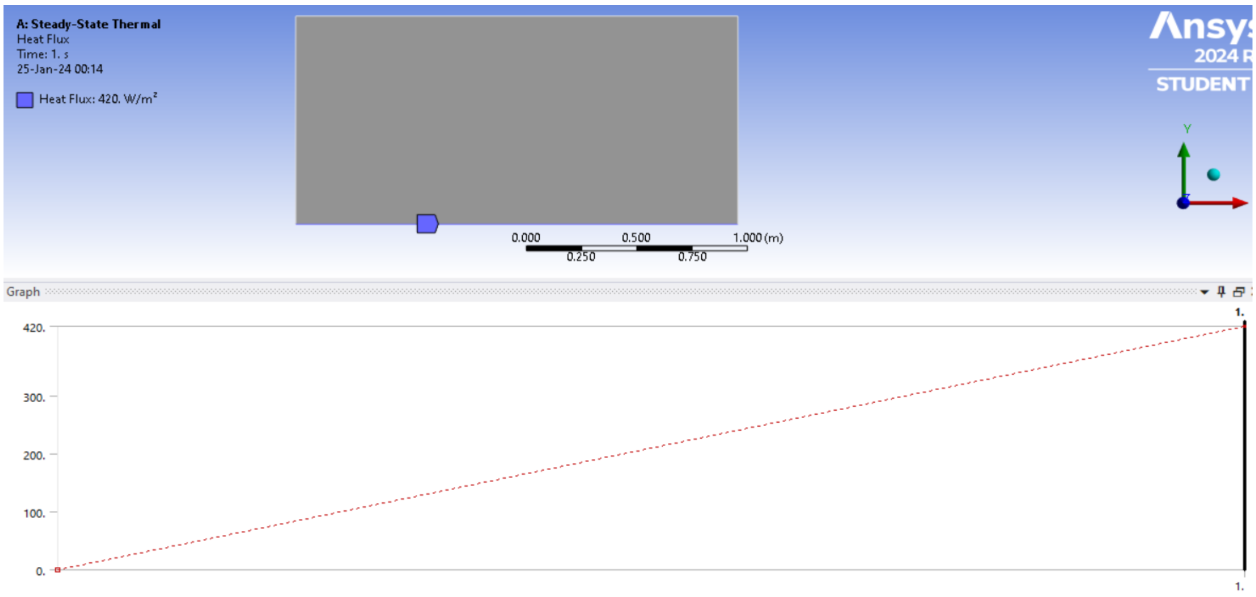
4. ANSYS Results for verification

4.0.1 Temperature Distribution in the plate.

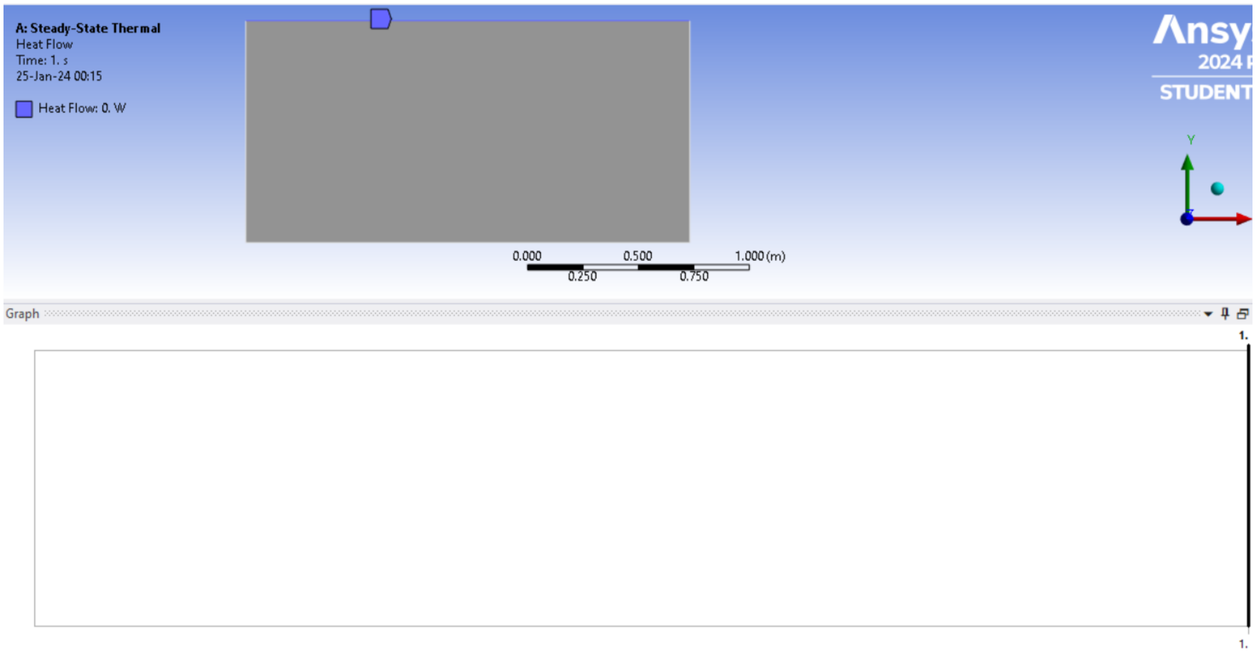
4.0.2 Temperature Graphs for boundary conditions.



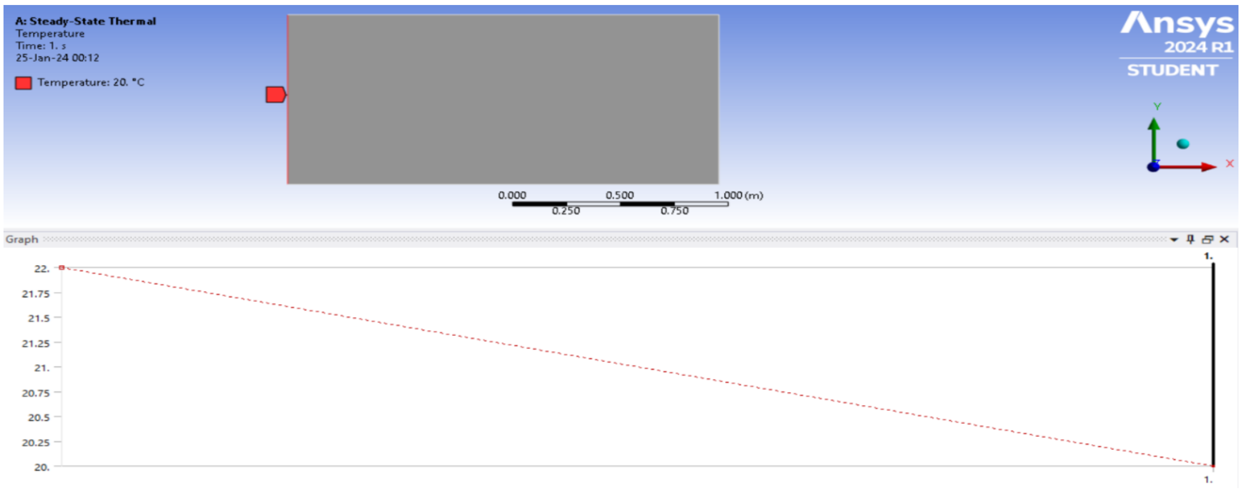
Temperature Distribution Across the Plate



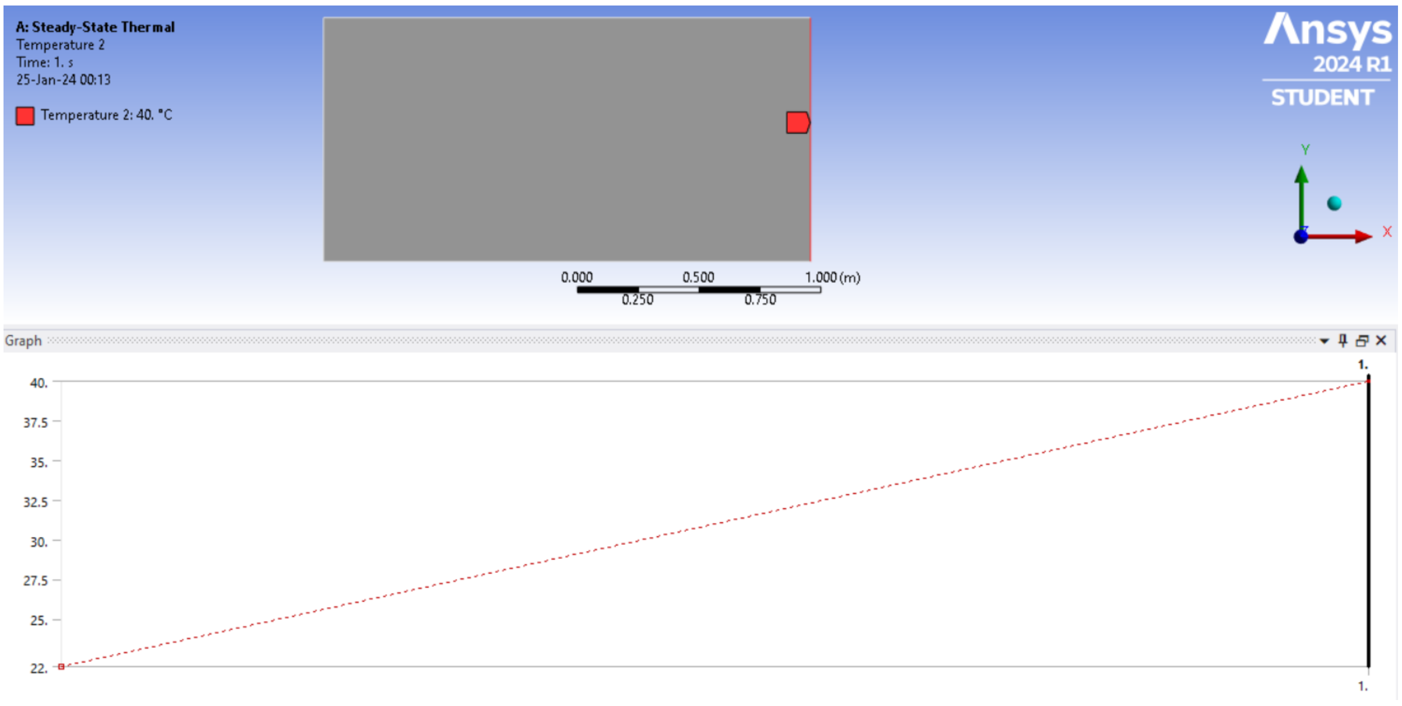
Graph 1 - Heat Flux Input Boundary Condition



Graph 2 - No Heat Flux Boundary Condition



Graph 3 - Constant Temperature Boundary at 20°C



Graph 4 - Constant Temperature Boundary at 40°C

6. Conclusion

• The finite difference method successfully simulated the stationary 2D heat conduction problem, as

evidenced by consistent temperature profiles obtained in both MATLAB and ANSYS.

• The temperature distribution, as modeled, shows a clear gradient from the cooler region (20°C) to

the warmer region (40°C), which matches the imposed boundary conditions.

• The specified heat flux on one side of the plate and the lack of heat input on the other was properly

reflected in the temperature contours, validating the boundary condition implementation.

• The consistent temperature distribution in both MATLAB and ANSYS simulations corroborates

the reliability of the numerical methods and boundary conditions used in the analysis.

• The findings confirm the reliability of using finite difference methods for solving similar heat conduction problems and demonstrate the robustness of the approach across different simulation plat forms.

7. References

• MATLAB documentation and user guides.

• Heat Transfer Textbook by Lienhard Lienhard.