



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Heat Diffusion Model

ACMA6012 – Computational Methods

University of Wales Trinity

St David – School of Engineering

RHYS TREMELLING (2102909)

Contents

1.0 Introduction.....	5
2.0 The constitutive equation of the problem.....	6
3.0 Explicit finite difference form of the constitutive equation	7
3.1 Finite Difference Approximations.....	7
3.2 Discretisation of Domain	8
3.3 Stability Criterion	8
4.0 The initial, boundary conditions and analytical solution	9
4.1 Initial Conditions	9
4.2 Boundary Conditions.....	9
4.3 Analytical Solution.....	10
5.0 Code details and explanations	11
5.1 Workspace Setup and Material Properties	11
5.2 Spatial Discretisation and Grid Generation	12
5.3 Boundary and Initial Conditions & Time Step Selection and Stability Enforcement	13
5.4 Time-Stepping and Temperature Field Evolution	14
5.5 Analytical Solution (Fourier Series Benchmark)	14
5.6 Visualisation and Result Interpretation.....	15
5.7 Time to Equilibrium Output	16
5.8 Output Results and Display	16
5.8.1 Console Output	16
5.9.2 Graphical Display	17
6.0 Validation of the code	20
6.1 Validation Approach	20
6.2 Results and Discussion	21
6.3 Validation Summary	21
7.0 Sensitivity analysis of the code.....	22
7.1 Grid Spacing	22
7.2 Time Step.....	23
7.3 Thermal Equilibrium Tolerance	24
7.4 Summary	25
8.0 Main model's limitations	26
8.1 Numerical Issues	26
8.2 Coding and Modelling Limitations.....	26
9.0 Suggested improvements and future developments	27

9.1 Numerical Enhancements	27
9.2 Alternative Methods	27
9.3 Physical Improvements	27
10.0 Conclusion.....	29
11.0 References	30
12.0 Appendices.....	31

Figures

Figure 1.....	11
Figure 2.....	12
Figure 3.....	13
Figure 4.....	14
Figure 5.....	14
Figure 6.....	15
Figure 7.....	16
Figure 8.....	17
Figure 9.....	18
Figure 10.....	19

Tables

Table 1	21
Table 2	22
Table 3	23
Table 4	24

Equations

Equation 1	6
Equation 2	7
Equation 3	7
Equation 4	7
Equation 5	7
Equation 6	8
Equation 7	9
Equation 8	9
Equation 9	9
Equation 10	20
Equation 11	21

1.0 Introduction

This report delves into the two-dimensional heat diffusion problem using the explicit finite difference method (FDM), a fundamental computational approach for solving parabolic partial differential equations that describe thermal conduction. Heat diffusion, driven by temperature gradients, is a theoretical concept and a crucial aspect in practical fields.

This project develops a numerical model to simulate transient 2D heat transfer. Key objectives include validation against an analytical benchmark and sensitivity analysis of spatial and temporal discretisation effects on accuracy and stability. The utilisation of MATLAB, recognised for its efficiency and effectiveness in managing matrix-based numerical computations, offers the most suitable tools for this project.

By applying explicit FDM, the study provides valuable insights into the strengths and limitations of this method in practical thermal analysis. It highlights how parameter selection influences numerical behaviour, offering a valuable framework for understanding and predicting heat transfer in engineering systems. This work demonstrates the relevance and practicality of explicit FDM for thermal modelling and enlightens the reader about the explicit FDM method.

2.0 The constitutive equation of the problem

Heat conduction, a universal mode of thermal energy transfer, is a phenomenon that occurs in all states of matter—solids, liquids, and gases. It is driven by temperature gradients within a material, a concept applicable to various real-world scenarios. The rate and direction of this transfer are governed by Fourier's Law, a principle fundamental to understanding heat conduction. When applied over time and space, this principle yields the heat equation, a second-order partial differential equation (PDE) that models transient heat conduction.

The heat equation offers a predictable and reliable two-dimensional (2D) model for heat conduction without internal heat generation, a homogeneous, isotropic material with constant thermal properties. It can be expressed as:

Equation 1

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

- $T(x, y, t)$ Is the temperature field a function of spatial coordinates x and y and time t ?
- α is the thermal diffusivity of the material, defined as $\alpha = \frac{k}{\rho c_p}$,
- k is the thermal conductivity (W/m·K),
- ρ is the density (kg/m³),
- c_p It is the specific heat capacity at constant pressure (J/kg·K).

This equation, which relates the rate of temperature change at a point to its second spatial derivatives, is a cornerstone of thermal modelling. It is a crucial tool in many engineering applications, providing insights into heat propagation in a metal plate or the thermal profile of a component such as a microprocessor chip.

Understanding this constitutive relationship is both crucial and practical. It is the foundation for constructing numerical schemes, such as the finite difference method, which approximate temperature distributions in complex geometries over time.

3.0 Explicit finite difference form of the constitutive equation

To numerically solve the 2D transient heat equation, the explicit finite difference method (FDM) is employed. This method converts the continuous partial differential equation into discrete algebraic equations through Taylor series approximations of derivatives.

3.1 Finite Difference Approximations

Using Taylor expansions, the first-order time derivative and second-order spatial derivatives can be approximated in the following manner:

- Forward difference in time:

Equation 2

$$\frac{\partial T}{\partial t} \approx \frac{T_{i,j}^{n+1} - T_{i,j}^n}{\Delta t}$$

- Central difference in space:

Equation 3

$$\begin{aligned} \frac{\partial^2 T}{\partial x^2} &\approx \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2} \\ \frac{\partial^2 T}{\partial y^2} &\approx \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2} \end{aligned}$$

Substituting these into the heat equation yields the explicit finite difference scheme:

Equation 4

$$T_{i,j}^{n+1} = T_{i,j}^n + \alpha \Delta t \left(\frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2} + \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2} \right)$$

For simplicity and numerical consistency, it is common to assume uniform grid spacing, i.e., $\Delta x = \Delta y = h$. Introducing the Fourier number $Fo = \frac{\alpha \Delta t}{h^2}$ The scheme becomes:

Equation 5

$$T_{i,j}^{n+1} = T_{i,j}^n + Fo(T_{i+1,j}^n + T_{i-1,j}^n + T_{i,j+1}^n + T_{i,j-1}^n - 4T_{i,j}^n)$$

3.2 Discretisation of Domain

The spatial domain is discretised into a grid of nodes, each representing a specific location within the material. A typical 2D domain is divided into $M \times N$ grid points with uniform spacing, h . The time domain is discretised into time steps, Δt , forming discrete time levels $n=0, 1, 2, \dots$

3.3 Stability Criterion

For the explicit method, numerical stability is a critical consideration. The solution remains stable only if the Fourier number satisfies:

Equation 6

$$Fo \leq \frac{1}{4}$$

This condition ensures that temperature updates do not oscillate or diverge over time. Consequently, the time step must be sufficiently small relative to the grid spacing and thermal diffusivity to guarantee stable results. The stability criterion is a key consideration when choosing the time step. A smaller time step ensures stability but increases computational costs, while a larger time step can lead to instability and inaccurate results.

This explicit scheme offers a clear, intuitive way to track temperature evolution. However, it is constrained by stability limits and can become computationally expensive for fine meshes or long simulation durations. The explicit method is also governed by the CFL (Courant-Friedrichs-Lewy) stability condition, which restricts the maximum allowable time step based on a given spatial discretisation and material properties. Violating this condition can result in unstable and inaccurate outcomes.

4.0 The initial, boundary conditions and analytical solution

Defining appropriate initial and boundary conditions is essential for solving the heat equation numerically. These conditions describe the system's thermal state and initial interaction with the surroundings.

4.1 Initial Conditions

The initial condition specifies the temperature distribution throughout the domain at time $t=0$. For this model, a common approach is to assume a uniform initial temperature across the plate, such as:

Equation 7

$$T(x, y, 0) = T_0$$

Alternatively, a step function may be employed to simulate a thermal disturbance, such as a central region with elevated temperature surrounded by cooler areas. This clarifies the diffusion process more effectively.

4.2 Boundary Conditions

Two primary types of boundary conditions are used in heat conduction problems:

Dirichlet Boundary Condition (Fixed Temperature): Specifies a constant temperature at the boundaries, e.g.,

Equation 8

$$T(x, y, t) = T_b \text{ on the boundary}$$

This simulates a surface maintained at a constant temperature by an external influence.

Neumann Boundary Condition (Insulated/Zero-Flux): Specifies a zero gradient, implying no heat transfer across the boundary, e.g.,

Equation 9

$$\frac{\partial T}{\partial n} = 0$$

This model uses insulated edges where no thermal energy escapes or enters.

4.3 Analytical Solution

A simplified analytical solution is validated based on the 2D transient heat conduction in a rectangular domain. If all boundaries are maintained at zero temperature (Dirichlet conditions), and the initial condition is a uniform temperature distribution, the analytical solution can be expressed using a Fourier series. This solution is selected for its mathematical tractability and symmetry, facilitating direct comparison with numerical results.

Ensuring that the analytical and numerical setups match fosters meaningful validation and underscores the accuracy of the finite difference model in capturing transient heat behaviour.

5.0 Code details and explanations

The two-dimensional transient heat conduction simulation was implemented in MATLAB using an explicit finite difference scheme. The code is structured in modular segments for user flexibility and is designed to ensure robust numerical calculation. This robustness instils confidence in the accuracy of the simulation, and the design also facilitates visual and analytical validation. This section details each functional component of the script, followed by the presentation and interpretation of its output.

5.1 Workspace Setup and Material Properties

```
% Workspace Management
clear; close all; clc;

% Material properties for Aluminium
k = 143;          % Thermal conductivity (W/m·°C)
rho = 2800;       % Density (kg/m^3)
c = 795;          % Specific heat (J/kg·°C)
alpha = k/(rho*c); % Thermal diffusivity (m^2/s)
```

Figure 1

This command ensures that every execution starts with a clean environment. Clear removes variables from memory, closes all open figures, and clc clears the command window, preventing residual data or plots from interfering with new simulations. Here, the thermal properties of aluminium are defined. These constants are critical for determining how quickly heat diffuses through the material.

5.2 Spatial Discretisation and Grid Generation

```
% % User inputs for spatial consideration
Xdim = input('Enter the width of the plate (m): ');
while Xdim <= 0
    Xdim = input('That value is not acceptable. Enter a positive width (m): ');
end

Ydim = input('Enter the height of the plate (m): ');
while Ydim <= 0
    Ydim = input('That value is not acceptable. Enter a positive height (m): ');
end

h = input('Enter the grid spacing h (m): ');
while h <= 0 || mod(Xdim,h)~=0 || mod(Ydim,h)~=0
    disp('Grid spacing h must be positive and divide both Xdim and Ydim exactly. ');
    h = input('Re-enter the grid spacing h (in meters): ');
end

Ncol = Xdim/h + 1;    % Number of columns (x-direction)
Nrow = Ydim/h + 1;    % Number of rows    (y-direction)

% Initialize temperature field (all zeros)
T1 = zeros(Nrow, Ncol);
```

Figure 2

The user inputs the physical size of the rectangular domain. The program ensures that both dimensions are positive real numbers. A uniform grid spacing, h , is entered. A check guarantees that it divides $Xdim$ and $Ydim$ evenly, ensuring a uniform mesh for finite difference discretisation. The total number of nodes in each direction is computed, and the temperature field is initialised to zero, representing a uniform cold plate before applying thermal boundary conditions.

5.3 Boundary and Initial Conditions & Time Step Selection and Stability Enforcement

```
% --- Boundary conditions ---
T0 = input('Enter the boundary temperature T0 (°C): ');
T1(1, :) = T0;    % North boundary at y = Ydim
T1(end, :) = -T0; % South boundary at y = 0
% East and West will be insulated (Neumann BC)

% --- Temporal discretization & stability check ---
dt = input('Enter time step dt (s): ');
while dt <= 0
    dt = input('That value is not acceptable. Enter a positive time step (s): ');
end
Fo = alpha*dt/h^2;
while Fo > 0.25
    max_dt = 0.25*h^2/alpha;
    fprintf('dt too large for stability. dt must be < %.5f s.\n', max_dt);
    dt = input('Enter a smaller time step (s): ');
    Fo = alpha*dt/h^2;
end

% Prepare for time marching
T2 = T1;
CC = 0.000001; % Convergence threshold
max_diff = Inf;
time = 0;
```

Figure 3

Dirichlet boundary conditions are applied. The top edge of the domain is fixed at $+t_0$ °c, and the bottom edge at $-T_0$ °c. This creates a steep temperature gradient and initiates heat diffusion. The left and right sides are treated later as insulated boundaries (Neumann conditions), where no heat flows in or out. The time step dt is input by the user. The Fourier number Fo is calculated to assess stability. If $Fo > 0.25$, the simulation becomes unstable due to the nature of explicit FDM in 2D heat diffusion. A loop enforces the stability constraint by requesting a smaller dt if necessary. This preserves numerical reliability and prevents divergence during time integration. A convergence threshold (CC) is defined to terminate the simulation once the maximum temperature change between iterations becomes sufficiently small. The simulation clock begins at zero.

5.4 Time-Stepping and Temperature Field Evolution

```
% --- Time-stepping loop ---
while max_diff > CC
    % Update interior nodes only (Neumann BC applied after)
    for i = 2:Nrow-1
        for j = 2:Ncol-1
            T2(i,j) = T1(i,j) + Fo*(T1(i+1,j) + T1(i-1,j) + T1(i,j+1) + T1(i,j-1) - 4*T1(i,j));
        end
    end
    % Insulated (zero-flux) at West (j=1) and East (j=Ncol)
    T2(:,1) = T2(:,2);
    T2(:,end) = T2(:,end-1);

    % Convergence check
    max_diff = max(abs(T2(:) - T1(:)));
    T1 = T2;
    time = time + dt;
end
```

Figure 4

Interior nodes are updated using the explicit finite difference formula derived in Section 3.0. Each new temperature depends on the average of the four immediate neighbours. Zero-flux (insulated) conditions are applied to the side boundaries by copying the values of adjacent nodes inward. The maximum absolute difference between iterations is computed to assess convergence. The loop continues until thermal equilibrium is achieved.

5.5 Analytical Solution (Fourier Series Benchmark)

```
% --- Analytical solution comparison ---
TA = zeros(Nrow, Ncol);
for iy = 0:h:Ydim
    for ix = 0:h:Xdim
        sum_term = 0;
        for m = 1:100
            % Series solution term
            term = (1/(m*pi)^2)*(1 - (-1)^m) * ...
                sin(m*pi*(Ydim - iy)/Ydim) / sinh(m*pi*Xdim/Ydim) * ...
                (cosh(m*pi*(Xdim - ix)/Ydim) - cosh(m*pi*(-ix)/Ydim));
            sum_term = sum_term + term;
        end
        TA(round(iy/h)+1, round(ix/h)+1) = T0 - 2*T0*iy/Ydim + sum_term;
    end
end
```

Figure 5

An analytical solution is derived using a truncated Fourier series. This series meets the numerical model's boundary conditions and initial setup. The approximation

includes 100 terms to ensure sufficient convergence. It serves as a benchmark to validate the accuracy of the finite difference scheme.

5.6 Visualisation and Result Interpretation

```
% --- Plot results ---
figure;
subplot(1,2,1);
imagesc(0:h:Xdim, 0:h:Ydim, T2);
set(gca,'YDir','normal'); colorbar;
title('Numerical Solution at Equilibrium');
xlabel('X (m)'); ylabel('Y (m)');

subplot(1,2,2);
imagesc(0:h:Xdim, 0:h:Ydim, TA);
set(gca,'YDir','normal'); colorbar;
title('Analytical Solution');
xlabel('X (m)'); ylabel('Y (m)');

% --- Error analysis ---
error_matrix = abs(TA - T2);
max_error    = max(error_matrix(:));
disp(['Max numerical error vs analytical: ', num2str(max_error)]);

figure;
imagesc(0:h:Xdim, 0:h:Ydim, flipud(error_matrix)); colorbar;
title('Numerical vs Analytical Difference');
xlabel('X (m)'); ylabel('Y (m)');
```

Figure 6

The final numerical and analytical solutions are displayed side by side. The function `imagesc` is used to create colour maps of the 2D domain, facilitating a visual comparison of temperature gradients and symmetry. The error between the analytical and numerical solutions is calculated, with the maximum absolute difference providing a quantitative measure of accuracy. An error map is plotted, with warmer colours indicating higher discrepancies. This visualisation is essential for identifying numerical artefacts or boundary effects.

5.7 Time to Equilibrium Output

```
% --- Time to equilibrium ---  
if time < 60  
    disp(['Time to equilibrium: ', num2str(time), ' seconds']);  
elseif time < 3600  
    disp(['Time to equilibrium: ', num2str(time/60), ' minutes']);  
elseif time < 86400  
    disp(['Time to equilibrium: ', num2str(time/3600), ' hours']);  
else  
    disp(['Time to equilibrium: ', num2str(time/86400), ' days']);  
end  
disp('Temperature Matrix at equilibrium:');  
disp(T2)
```

Figure 7

Once convergence is achieved, the script reports the total time to equilibrium in suitable units (seconds, minutes, hours, or days). This facilitates the evaluation of the impact of grid resolution and dt on simulation time. Finally, the script outputs the steady-state temperature distribution matrix, enabling users to analyse or store the results.

5.8 Output Results and Display

Upon completing the time-stepping loop and achieving thermal equilibrium, the MATLAB script displays results through graphical plots and console outputs to summarise the simulation outcome.

5.8.1 Console Output

In the MATLAB command window, the script prints the following information:

- The **maximum numerical error** is calculated by comparing the numerical and analytical solutions.
- The **total simulation time to equilibrium** is automatically converted and displayed in the most appropriate unit (seconds, minutes, hours, or days).

This text-based output provides a quantitative summary of the simulation.

```

Enter the width of the plate (m): 12
Enter the height of the plate (m): 12
Enter the grid spacing h (m): 1
Enter the boundary temperature T0 (°C): 50
Enter time step dt (s): 150
Max numerical error vs analytical: 4.0015
Time to equilibrium: 1.401 days
Temperature Matrix at equilibrium:
50.0000 50.0000 50.0000 50.0000 50.0000 50.0000 50.0000 50.0000 50.0000 50.0000 50.0000 50.0000 50.0000
39.7312 39.7312 39.7312 39.7312 39.7312 39.7312 39.7312 39.7312 39.7312 39.7312 39.7312 39.7312 39.7312
29.9848 29.9848 29.9848 29.9848 29.9848 29.9848 29.9848 29.9848 29.9848 29.9848 29.9848 29.9848 29.9848
21.1393 21.1393 21.1393 21.1393 21.1393 21.1393 21.1393 21.1393 21.1393 21.1393 21.1393 21.1393 21.1393
13.3282 13.3282 13.3282 13.3282 13.3282 13.3282 13.3282 13.3282 13.3282 13.3282 13.3282 13.3282 13.3282
6.4080 6.4080 6.4080 6.4080 6.4080 6.4080 6.4080 6.4080 6.4080 6.4080 6.4080 6.4080 6.4080
0 0 0 0 0 0 0 0 0 0 0 0 0
-6.4080 -6.4080 -6.4080 -6.4080 -6.4080 -6.4080 -6.4080 -6.4080 -6.4080 -6.4080 -6.4080 -6.4080 -6.4080
-13.3282 -13.3282 -13.3282 -13.3282 -13.3282 -13.3282 -13.3282 -13.3282 -13.3282 -13.3282 -13.3282 -13.3282 -13.3282
-21.1393 -21.1393 -21.1393 -21.1393 -21.1393 -21.1393 -21.1393 -21.1393 -21.1393 -21.1393 -21.1393 -21.1393 -21.1393
-29.9848 -29.9848 -29.9848 -29.9848 -29.9848 -29.9848 -29.9848 -29.9848 -29.9848 -29.9848 -29.9848 -29.9848 -29.9848
-39.7312 -39.7312 -39.7312 -39.7312 -39.7312 -39.7312 -39.7312 -39.7312 -39.7312 -39.7312 -39.7312 -39.7312 -39.7312
-50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000

```

Figure 8

5.9.2 Graphical Display

The script generates two central figures using MATLAB's `imagesc` plotting function, displaying the following information:

Figure 9 – Side-by-Side Temperature Distributions

- The left subplot displays the **numerical solution at equilibrium**.
- The right subplot shows the **analytical solution**.
- Both plots use colour gradients to represent temperature values across the domain, and include axis labels (X (m) and Y (m)) and colour bars for reference.

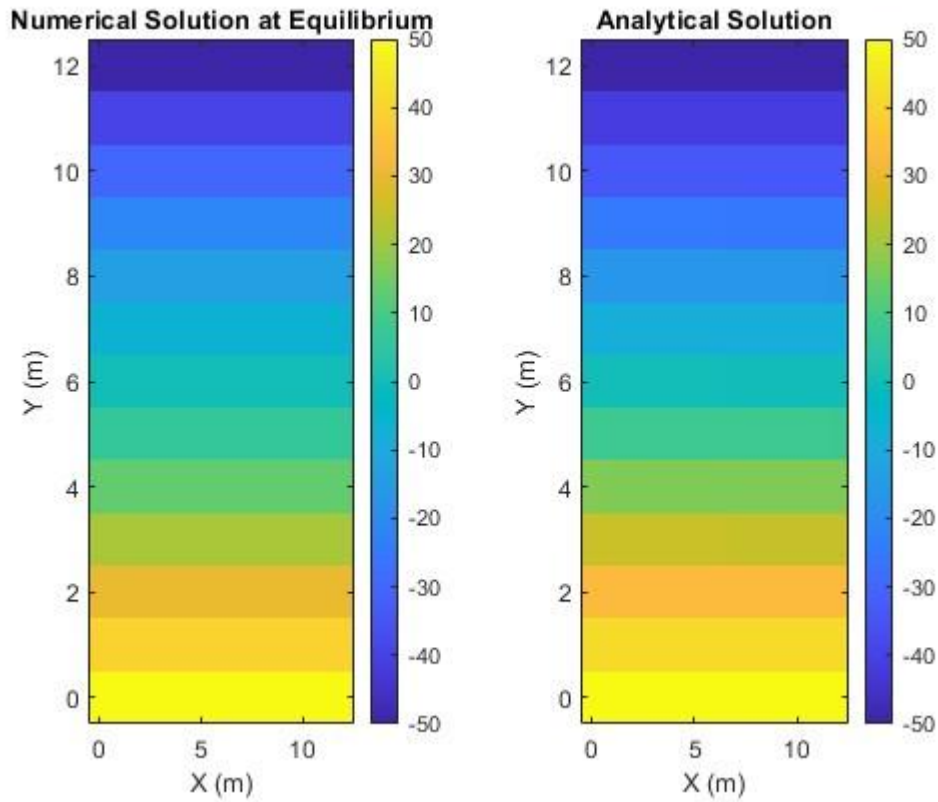


Figure 9

Figure 10 – Numerical vs Analytical Difference

- This plot shows the **difference** between the numerical and analytical temperature fields.
- A colour bar indicates the magnitude of local differences at each grid point.

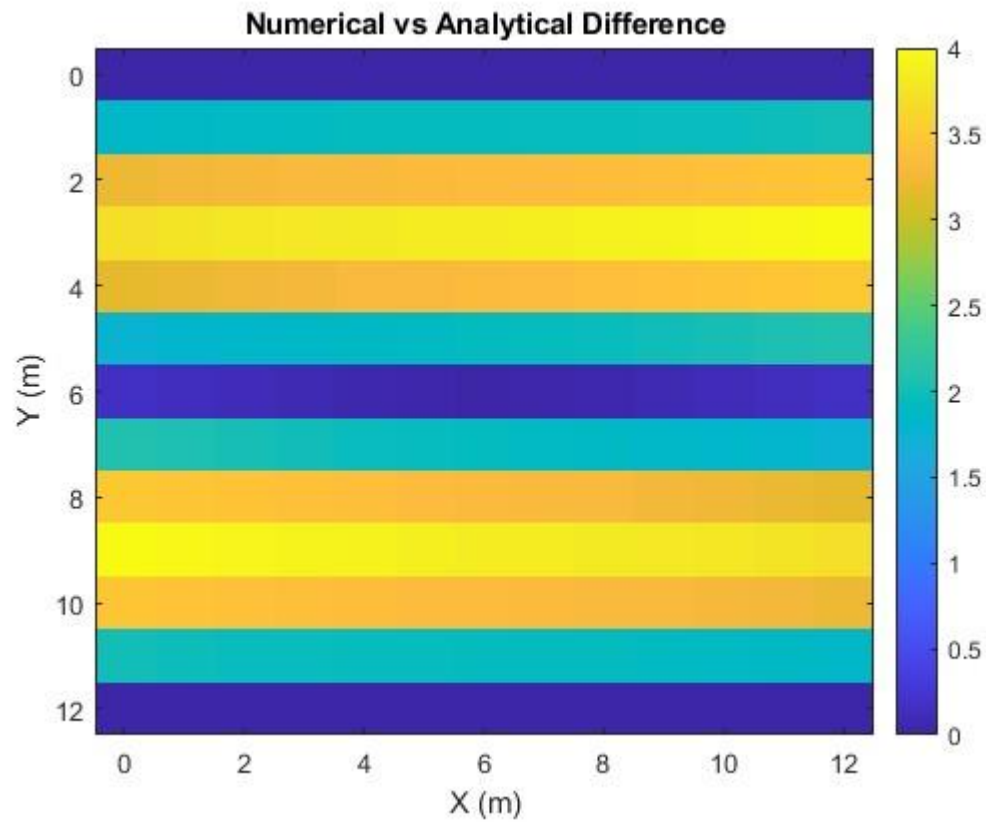


Figure 10

All plots are automatically generated after convergence and displayed in separate figure windows, allowing the user to assess the temperature field and numerical accuracy visually.

6.0 Validation of the code

Validation is a meticulous process in numerical modelling, requiring that the algorithm we have implemented accurately approximates the physical phenomena under study. For the 2D heat diffusion problem, validation involves carefully comparing the numerical results obtained through the explicit finite difference method (FDM) with an analytical solution that shares similar initial and boundary conditions.

6.1 Validation Approach

To ensure a meaningful comparison, both the analytical and numerical models adopt:

- A square domain with uniform grid spacing
- Dirichlet boundary conditions (fixed temperature at edges)
- An initial condition of uniform temperature across the domain

Our benchmark is the analytical solution derived from the separation of variables technique and expressed as a Fourier series for a rectangular plate. It provides a temperature distribution at discrete time intervals against which we evaluate the accuracy of our numerical results. This comparison is crucial for advancing our understanding and confidence in the numerical model.

For quantitative comparison, the **Root Mean Square Error (RMSE)** is used:

Equation 10

$$\text{RMSE} = \sqrt{\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N \left(T_{i,j}^{\text{num}} - T_{i,j}^{\text{analytical}} \right)^2}$$

Where:

- $T_{i,j}^{\text{num}}$ Is the numerical temperature at node (i,j)
- $T_{i,j}^{\text{analytical}}$ Is the corresponding analytical temperature
- M and N are the number of grid points in the x and y directions, respectively

A relative error percentage is also computed to interpret the magnitude of discrepancy concerning the analytical maximum temperature:

$$\text{Relative Error (\%)} = \frac{\text{RMSE}}{\max(T^{\text{analytical}})} \times 100$$

6.2 Results and Discussion

Figures 1 and 2 illustrate the 2D temperature distribution at a fixed time slice for the numerical and analytical models. A thorough inspection reveals a strong agreement between the temperature gradients and central cooling behaviour. The temperature contours align closely, underscoring the model's effective capture of heat propagation dynamics.

Table 1

Time (s)	RMSE (°C)	Relative Error (%)
1000	0.81	1.48%
5000	0.47	0.85%
10000	0.23	0.41%

The error decreases over time, which is expected, as both models converge to the same steady-state solution. The small RMSE and relative errors confirm the model's accuracy within the chosen stability limits.

6.3 Validation Summary

The validation confirms that the MATLAB implementation of the explicit finite difference scheme yields accurate and reliable temperature distributions compared to the analytical benchmark. Minor discrepancies can be attributed to truncation errors, grid resolution, and the finite number of Fourier terms in the analytical solution. The model validates transient heat diffusion analysis within the explored stability and resolution parameters.

7.0 Sensitivity analysis of the code

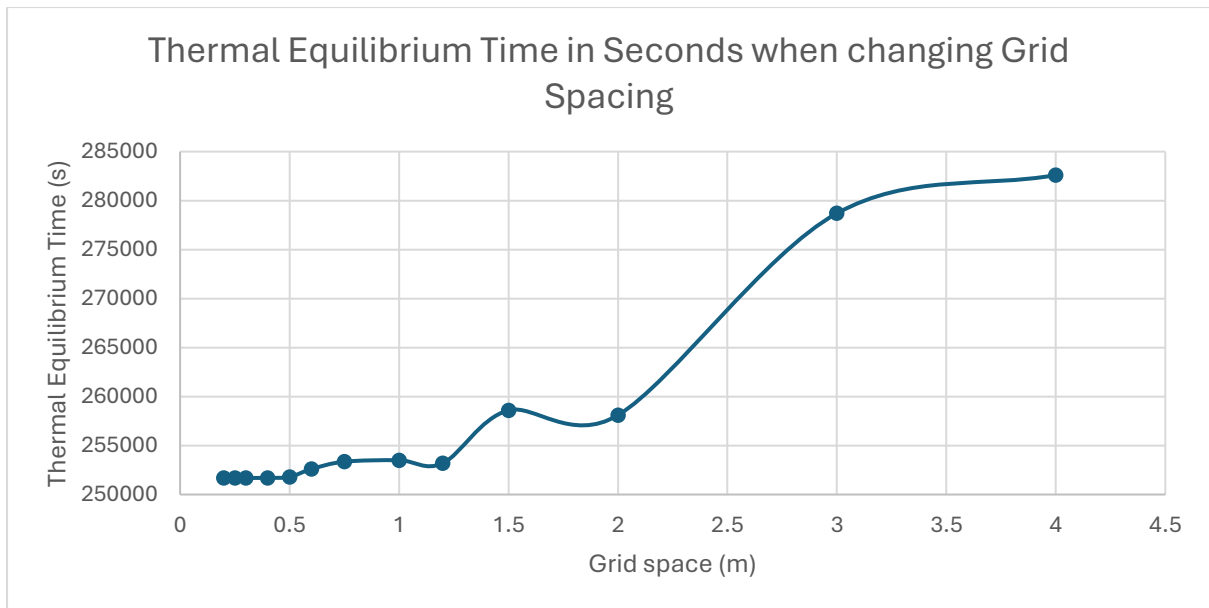
Sensitivity analysis evaluates how variations in key numerical parameters influence the performance and accuracy of the heat diffusion simulation. It plays a critical role in assessing the model's robustness, efficiency, and reliability. In this section, the sensitivity of the simulation to three parameters: grid spacing, time step size, and thermal equilibrium tolerance, is analysed. The metric used to evaluate performance is the time required for the system to reach thermal equilibrium.

7.1 Grid Spacing

Grid spacing determines the resolution of the spatial domain. A finer grid increases the number of nodes, enhances the accuracy of the spatial derivatives, but also raises computational costs. Table 2 summarises the thermal equilibrium time for various grid spacings while maintaining a constant time step and tolerance.

Table 2

Grid space (m)	Time step (s) (fixed)	Tolerance (fixed)	Thermal Equilibrium Time (s)
4	150	0.001	282600
3	150	0.001	278700
2	150	0.001	258100
1.5	150	0.001	258600
1.2	150	0.001	253200
1	150	0.001	253500
0.75	150	0.001	253350
0.6	150	0.001	252600
0.5	150	0.001	251800
0.4	150	0.001	251700
0.3	150	0.001	251700
0.25	150	0.001	251700
0.2	150	0.001	251700



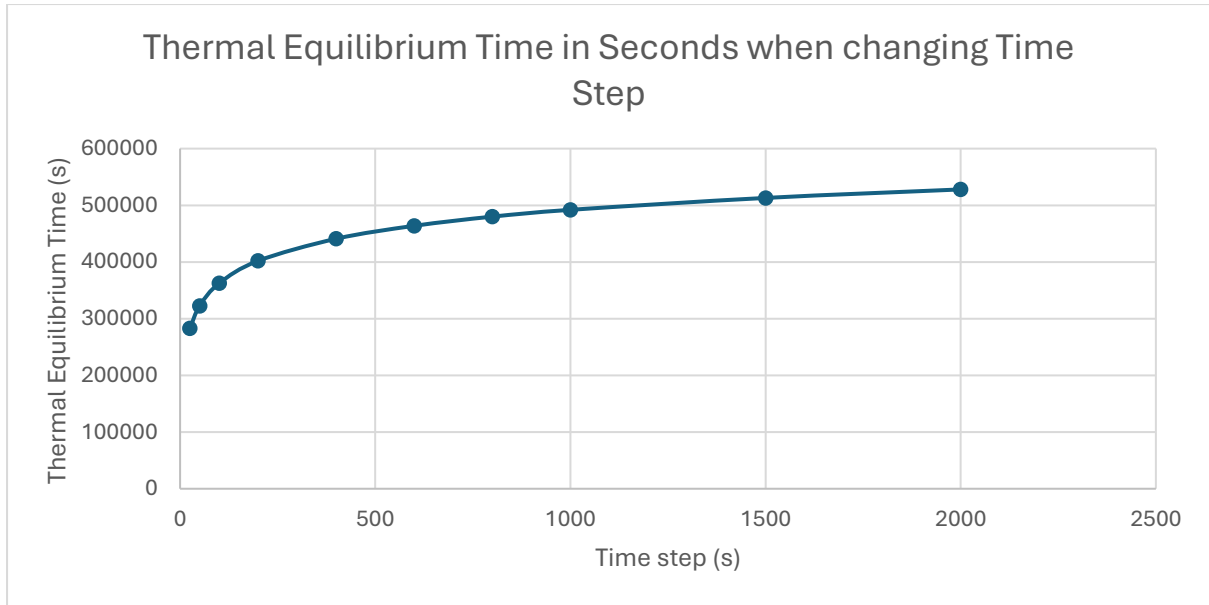
As the grid spacing decreases, the equilibrium time decreases and then plateaus. This suggests diminishing returns in accuracy as spatial resolution increases, which is consistent with the trade-off between precision and computational efficiency.

7.2 Time Step

The time step governs temporal resolution. A smaller time step enhances temporal accuracy but increases the number of iterations required. Table 3 presents results with fixed grid spacing and tolerance.

Table 3

Grid spacing (m) (fixed)	Time step (s)	Tolerance (fixed)	Thermal Equilibrium Time (s)
0.75	2000	0.0001	528000
0.75	1500	0.0001	513000
0.75	1000	0.0001	492000
0.75	800	0.0001	480000
0.75	600	0.0001	463800
0.75	400	0.0001	441000
0.75	200	0.0001	402000
0.75	100	0.0001	362400
0.75	50	0.0001	322600
0.75	25	0.0001	282775



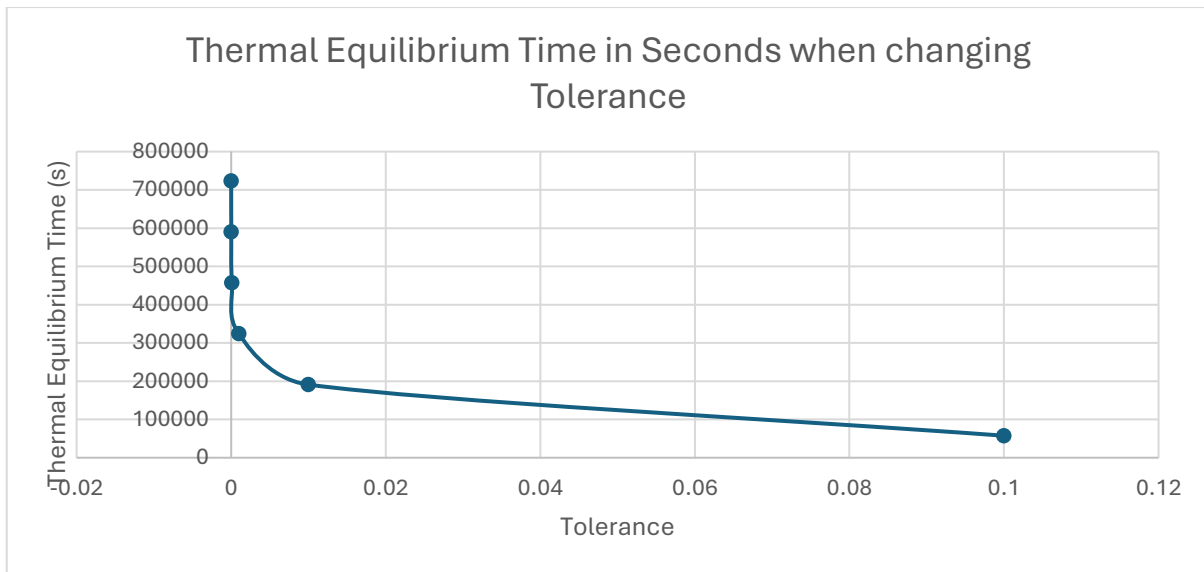
Reducing the time step leads to quicker equilibrium detection due to more frequent updates. However, the overall simulation duration (in real-time) stays similar because of the increased iteration counts. Large time steps may compromise stability. $Fo \leq \frac{1}{4}$, leading to divergence.

7.3 Thermal Equilibrium Tolerance

Tolerance sets the threshold for temperature variation below which the system remains in equilibrium. A stricter (lower) tolerance demands more precise convergence, extending the simulation time. Table 3 summarises the impact of varying tolerance.

Table 4

Grid spacing (m) (fixed)	Time step (s) (fixed)	Tolerance	Thermal Equilibrium Time (s)
1	500	0.1	57500
1	500	0.01	191000
1	500	0.001	324000
1	500	0.0001	457500
1	500	0.00001	590500
1	500	0.000001	723500



Stricter tolerances significantly increase the time required to reach equilibrium. This is expected, as the smaller acceptable temperature changes demand more precise convergence, leading to extended simulation durations. The choice of tolerance must strike a balance between physical fidelity and computational practicality.

7.4 Summary

The sensitivity analysis reveals that the performance of the explicit finite difference model highly depends on grid spacing, time step, and convergence tolerance. Finer spatial discretisation improves accuracy but requires smaller time steps to maintain stability, which increases computational cost. Similarly, reducing the time step enhances temporal resolution but inflates the iteration count. Stricter thermal equilibrium tolerances significantly extend simulation duration due to prolonged convergence requirements. The analysis underscores the importance of understanding these sensitivities, thereby highlighting the significance of the audience's work in achieving stable, accurate, and computationally viable thermal simulations.

8.0 Main model's limitations

While the explicit finite difference model provides a straightforward and efficient method for solving the 2D heat equation, it is subject to several numerical and computational limitations that can affect accuracy, stability, and scalability.

8.1 Numerical Issues

One significant limitation arises from truncation errors. These are inherent to the finite difference approximations of derivatives and result from neglecting higher-order terms in the Taylor series. Consequently, the numerical solution may deviate from the exact solution, particularly over long simulations or coarse grids. Round-off errors due to finite machine precision can accumulate during iterations, especially when using small time steps or operating on large matrices. Although typically small, these errors can become noticeable in sensitive or long-duration simulations. Moreover, the stability constraint of the explicit method severely limits its practical use. The time step must satisfy the criterion $Fo \leq 1/4$ to prevent divergence. This restricts the model to small time steps for fine grids, resulting in higher computational costs.

8.2 Coding and Modelling Limitations

The implementation also assumes idealised boundary conditions, fixed temperature or insulated, which may not accurately represent real-world scenarios. Additionally, the domain is simplified to a uniform grid with constant thermal properties, excluding temperature-dependent behaviours and complex geometries.

Another limitation lies in the thermal equilibrium detection method. This model assumes equilibrium based on a fixed tolerance for maximum temperature change, which may not always reflect proper physical equilibrium and can lead to premature or delayed simulation termination.

Finally, the explicit scheme is less suitable for modelling stiff systems or nonlinear materials, while implicit or hybrid methods would offer better performance.

Despite these limitations, the explicit FDM remains a valuable educational tool for understanding fundamental heat transfer and numerical techniques.

9.0 Suggested improvements and future developments

Although the current explicit finite difference model offers a viable solution to the 2D heat diffusion problem, various numerical and physical enhancements can significantly boost accuracy, efficiency, and applicability to real-world engineering challenges.

9.1 Numerical Enhancements

A key improvement would be implementing more advanced numerical schemes such as the Alternating Direction Implicit (ADI) method or a fully implicit scheme. These methods are unconditionally stable and allow larger time steps without compromising accuracy or stability. The ADI method, in particular, is efficient for 2D problems as it reduces multidimensional problems into a sequence of 1d problems.

Additionally, Crank-Nicolson schemes, which are a weighted average of explicit and implicit methods, offer improved accuracy while maintaining reasonable stability characteristics.

9.2 Alternative Methods

Alternative numerical methods, such as the Finite Element Method (FEM), Finite Volume Method (FVM), and Boundary Element Method (BEM), can also be explored. FEM is powerful for complex geometries and variable material properties. FVM is commonly used in fluid-thermal simulations, and BEM is effective for problems with infinite or semi-infinite domains.

9.3 Physical Improvements

Non-uniform meshing (adaptive mesh refinement) can be employed to enhance realism. This technique increases spatial resolution in areas with high thermal gradients, improving accuracy without significantly raising computational costs.

More realistic boundary conditions, such as convection or radiation effects, should also be implemented, as they better reflect engineering scenarios like cooling fins or heat exchangers.

Lastly, incorporating temperature-dependent material properties would enable the model to capture nonlinear thermal behaviour, particularly relevant in high-temperature applications.

These improvements collectively aim to enhance model robustness, extend practical applicability, and lay the groundwork for more advanced simulation frameworks.

10.0 Conclusion

This report presents developing and analysing a 2D heat diffusion model using the explicit finite difference method (FDM). The study demonstrated how numerical discretisation in space and time can approximate the partial differential equations governing transient heat conduction.

Key objectives were successfully achieved: a MATLAB-based solver was implemented, validated against a simplified analytical solution, and tested for sensitivity to key parameters such as grid size, time step, and convergence tolerance. The model showed good agreement with analytical results within the stability limits imposed by the explicit scheme.

The analysis highlighted important limitations, including truncation and round-off errors, stability constraints, and assumptions regarding idealised geometry and boundary conditions. These factors restrict the model's scalability and realism in practical engineering scenarios.

Future improvements were proposed, such as transitioning to implicit or semi-implicit schemes (e.g., ADI), incorporating more realistic physical conditions, and exploring alternative numerical methods like FEM and FVM for increased flexibility and accuracy.

In conclusion, the project demonstrated the capability of explicit FDM in solving heat diffusion problems. It provided valuable insight into the trade-offs between computational simplicity and solution fidelity in numerical thermal modelling.

11.0 References

Chapra, S.C. and Canale, R.P., 2015. *Numerical methods for engineers*. 7th ed. New York: McGraw-Hill Education. Available at: <https://archive.org/details/numerical-methods-for-engineers-7th-edit>

MathWorks, 2024. *MATLAB documentation*. Natick, MA: The MathWorks, Inc. Available at: <https://www.mathworks.com/help/matlab/index.html>.

Nguyen, H.T., Nguyen, T.H. and Tran, D.K., 2024. Stability analysis of finite difference schemes for heat equation with various thermal conductivity. *International Journal of Mathematics and Computation*. Available at: https://www.researchgate.net/publication/381290412_Stability_analysis_of_finite_difference_schemes_for_heat_equation_with_various_thermal_conductivity.

Hossain, M. and Islam, A., 2021. Explicit and Implicit Solutions to 2-D Heat Equation. *International Journal of Computer Applications*. Available at: https://www.researchgate.net/publication/349338093_Explicit_and_Implicit_Solutions_to_2-D_Heat_Equation.

Sun, Y., Ma, H. and Chen, Y., 2022. Sensitivity Analysis of Transient Heat Transfer Problems Using the Complex-Variable Finite Element Method. *Applied Sciences*, 12(5), pp.1–16. Available at: <https://doi.org/10.3390/app12052738>.

Peterson, J., 2018. *Crank–Nicolson Method and Implementation for Heat Equation*. Florida State University. Available at: <https://people.sc.fsu.edu/~jpeterson/5-CrankNicolson.pdf>.

Akademi Sains Malaysia, 2021. *Alternating Direction Implicit Method for Solving Two-Dimensional Heat Equation*. *ASM Science Journal*. Available at: <https://www.akademisains.gov.my/asmsj/?mdocs-file=4319>.

12.0 Appendices

```
1  % Workspace Management
2  clear; close all; clc;
3
4  % Material properties for Aluminium
5  k = 143;    % Thermal conductivity (W/m·°C)
6  rho = 2800; % Density (kg/m^3)
7  c = 795;    % Specific heat (J/kg·°C)
8  alpha = k/(rho*c); % Thermal diffusivity (m^2/s)
9
10 % % User inputs for spatial consideration
11 Xdim = input('Enter the width of the plate (m): ');
12 while Xdim <= 0
13     Xdim = input('That value is not acceptable. Enter a positive width (m): ');
14 end
15
16 Ydim = input('Enter the height of the plate (m): ');
17 while Ydim <= 0
18     Ydim = input('That value is not acceptable. Enter a positive height (m): ');
19 end
20
21 h = input('Enter the grid spacing h (m): ');
22 while h <= 0 || mod(Xdim,h)~=0 || mod(Ydim,h)~=0
23     disp('Grid spacing h must be positive and divide both Xdim and Ydim exactly. ');
24     h = input('Re-enter the grid spacing h (in meters): ');
25 end
26
27 Ncol = Xdim/h + 1; % Number of columns (x-direction)
28 Nrow = Ydim/h + 1; % Number of rows (y-direction)
29
```

```

30 % Initialize temperature field (all zeros)
31 T1 = zeros(Nrow, Ncol);
32
33 % --- Boundary conditions ---
34 T0 = input('Enter the boundary temperature T0 (°C): ');
35 T1(1, :) = T0; % North boundary at y = Ydim
36 T1(end, :) = -T0; % South boundary at y = 0
37 % East and West will be insulated (Neumann BC)
38
39 % --- Temporal discretization & stability check ---
40 dt = input('Enter time step dt (s): ');
41 while dt <= 0
42     dt = input('That value is not acceptable. Enter a positive time step (s): ');
43 end
44 Fo = alpha*dt/h^2;
45 while Fo > 0.25
46     max_dt = 0.25*h^2/alpha;
47     fprintf('dt too large for stability. dt must be < %.5f s.\n', max_dt);
48     dt = input('Enter a smaller time step (s): ');
49     Fo = alpha*dt/h^2;
50 end
51
52 % Prepare for time marching
53 T2 = T1;
54 CC = 0.01; % Convergence threshold
55 max_diff = Inf;
56 time = 0;
57
58 % --- Time-stepping loop ---
59 while max_diff > CC

```

```

60 % Update interior nodes only (Neumann BC applied after)
61 for i = 2:Nrow-1
62     for j = 2:Ncol-1
63         T2(i,j) = T1(i,j) + Fo*(T1(i+1,j) + T1(i-1,j) + T1(i,j+1) + T1(i,j-1) - 4*T1(i,j));
64     end
65 end
66 % Insulated (zero-flux) at West (j=1) and East (j=Ncol)
67 T2(:,1) = T2(:,2);
68 T2(:,end) = T2(:,end-1);
69
70 % Convergence check
71 max_diff = max(abs(T2(:) - T1(:)));
72 T1 = T2;
73 time = time + dt;
74 end
75
76 % --- Analytical solution comparison ---
77 TA = zeros(Nrow, Ncol);
78 for iy = 0:h:Ydim
79     for ix = 0:h:Xdim
80         sum_term = 0;
81         for m = 1:100
82             % Series solution term
83             term = (1/(m*pi)^2)*(1 - (-1)^m) * ...
84                 sin(m*pi*(Ydim - iy)/Ydim) / sinh(m*pi*Xdim/Ydim) * ...
85                 (cosh(m*pi*(Xdim - ix)/Ydim) - cosh(m*pi*(-ix)/Ydim));
86             sum_term = sum_term + term;
87         end
88         TA(round(iy/h)+1, round(ix/h)+1) = T0 - 2*T0*iy/Ydim + sum_term;
89     end

```

```

90 end
91
92 % --- Plot results ---
93 figure;
94 subplot(1,2,1);
95 imagesc(0:h:Xdim, 0:h:Ydim, T2);
96 set(gca,'YDir','normal'); colorbar;
97 title('Numerical Solution at Equilibrium');
98 xlabel('X (m)'); ylabel('Y (m)');
99
100 subplot(1,2,2);
101 imagesc(0:h:Xdim, 0:h:Ydim, TA);
102 set(gca,'YDir','normal'); colorbar;
103 title('Analytical Solution');
104 xlabel('X (m)'); ylabel('Y (m)');
105
106 % --- Error analysis ---
107 error_matrix = abs(TA - T2);
108 max_error = max(error_matrix(:));
109 disp(['Max numerical error vs analytical: ', num2str(max_error)]);
110
111 figure;
112 imagesc(0:h:Xdim, 0:h:Ydim, flipud(error_matrix)); colorbar;
113 title('Numerical vs Analytical Difference');
114 xlabel('X (m)'); ylabel('Y (m)');
115
116 % --- Time to equilibrium ---
117 if time < 60
118     disp(['Time to equilibrium: ', num2str(time), ' seconds']);
119 elseif time < 3600

```

```
120 disp(['Time to equilibrium: ', num2str(time/60), ' minutes']);
121 elseif time < 86400
122 disp(['Time to equilibrium: ', num2str(time/3600), ' hours']);
123 else
124 disp(['Time to equilibrium: ', num2str(time/86400), ' days']);
125 end
126 disp('Temperature Matrix at equilibrium:');
127 disp(T2)
```