

ANALYSIS OF 2D HEAT TRANSFER: EXPLORING THE FINITE DIFFERENCE MODEL



BEng Mechanical and Manufacturing Engineering

Computational Methods

Budweiser
BREWING GROUP UK&I™

A PROUD PART OF **ABInBev**



University of Wales
Trinity Saint David

Report submitted by Benjamin James

1) Table of Contents

1) TABLE OF CONTENTS	2
2) INTRODUCTION	3
3) THE GOVERNING EQUATION OF THE PROBLEM.....	4
4) NUMERICAL METHOD	5
4.1 TAYLOR SERIES FOR DIFFERENCE APPROXIMATIONS	5
4.2 DISCRETIZATION OF THE HEAT EQUATION.....	5
4.3 SPATIAL AND TEMPORAL DOMAIN DISCRETIZATION.....	6
4.4 STABILITY CRITERION AND FOURIER'S NUMBER.....	6
5) INITIAL AND BOUNDARY CONDITIONS	7
6) CODE'S DETAILS AND EXPLANATIONS	9
6.1 WORKSPACE INITIALIZATION AND MATERIAL PROPERTIES:.....	9
6.2 PLATE DIMENSIONS AND GRID SPACING:	9
6.3 TIME STEP CALCULATION AND STABILITY CHECK:	10
6.4 TEMPERATURE GRID INITIALIZATION AND BOUNDARY CONDITIONS:	10
6.5 NUMERICAL SIMULATION CORE:	11
6.6 RESULTS DISPLAY AND VISUALIZATION:.....	11
6.7 ANALYTICAL SOLUTION COMPARISON:	12
6.8 ANALYTICAL TEMPERATURE CALCULATION FUNCTION:.....	12
7) VALIDATION OF THE MODEL.....	14
8) SENSITIVITY ANALYSIS OF THE MODEL.....	17
8.1 GRID SPACING VS THERMAL EQUILIBRIUM TIME	17
8.2 TIME STEP VS THERMAL EQUILIBRIUM TIME	19
8.3 TOLERANCE VS THERMAL EQUILIBRIUM TIME	21
9) MAIN MODEL'S LIMITATIONS	23
10) SUGGESTED IMPROVEMENTS AND FUTURE DEVELOPMENTS	24
11) CONCLUSION	25
12) REFERENCES	26
13) APPENDIX	28
13.1 MATLAB SCRIPT FOR 2D HEAT TRANSFER USING EXPLICIT FINITE DIFFERENCE METHOD	28

2) Introduction

The study of heat transfer stands at the forefront of engineering and scientific innovation, underpinning critical advancements in a myriad of applications ranging from energy-efficient building designs to high-performance materials. This report embarks on an in-depth exploration of two-dimensional heat diffusion, a fundamental aspect of thermal dynamics, employing the explicit finite difference method (FDM) — a technique pivotal in contemporary thermal analysis. The choice of this method reflects a response to the growing need for accurate yet computationally efficient models in thermal engineering, especially in the context of rapidly evolving technologies and environmental challenges (**Zhang & Zhu, 2018**).

Structured to provide a clear and logical progression, this report initially delves into the governing equation of the heat diffusion problem, laying a theoretical foundation. Following this, we undertake a detailed examination of the numerical methodology, focusing on the spatial and temporal discretization that is crucial for accurately capturing the nuances of thermal behaviour in two dimensions. The discussion then advances to a meticulous description of the initial and boundary conditions, integral to defining the scope of our model, accompanied by an insightful presentation of the developed MATLAB code.

Central to the study is the validation of the model against an analytical solution, a step essential for ascertaining the reliability and accuracy of our computational predictions. Of particular interest is the sensitivity analysis section, where we rigorously examine the impact of varying parameters such as grid spacing, time step, and tolerance. This analysis is not merely a technical exercise but is designed to shed light on the practical implications of these parameters in real-world applications, thereby enhancing the reader's understanding of the delicate balance between computational accuracy and efficiency (**Dahalan & Sulaiman, 2021; Fernandes et al., 2019**).

The report also acknowledges and critically examines the inherent limitations of the numerical methods and coding practices employed. This self-reflection is crucial for a nuanced appreciation of the model's capabilities and constraints, offering valuable insights into areas ripe for future research and improvement. For example, recent advancements in explicit numerical algorithms for the linear heat or diffusion equation highlight potential pathways for enhancing computational efficiency (**Kovács, 2020; Adak, 2018**).

Aiming to contribute constructively to the field, the report culminates with a series of proposed enhancements and future developments. These recommendations are specifically targeted at addressing the identified limitations, with a vision to refine the accuracy, stability, and broader applicability of the model. The ultimate goal is to not only advance the theoretical understanding of heat diffusion processes but also to provide practical pathways for future innovations in thermal modelling.

In summary, this report seeks to offer a comprehensive and accessible exploration of 2D heat diffusion using explicit FDM. It is intended to serve as a valuable resource for both specialists seeking detailed technical insights and for a broader audience interested in the practical applications and implications of thermal modelling in modern engineering and scientific contexts.

3) The Governing Equation of the Problem

The heat equation stands as one of the most fundamental partial differential equations in the realm of thermal physics and engineering. Originating from Fourier's law of heat conduction, the equation offers a mathematical representation of heat distribution within a material over time. Its importance cannot be overstated, as it provides the theoretical underpinnings for the design and analysis of countless thermal systems.

In two-dimensional geometry, devoid of internal heat generation, the heat equation can be expressed as (Stepanova, 2019; Silva et al., 2019):

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

where $T(x, y, t)$ denotes the temperature field as a function of spatial coordinates x and y , and time t . In this expression, $\frac{\partial T}{\partial t}$ signifies the rate of change of temperature with time, while $\frac{\partial^2 T}{\partial x^2}$ and $\frac{\partial^2 T}{\partial y^2}$ represent the second order partial derivatives of temperature with respect to the spatial coordinates, indicating the spread of heat in the x and y directions, respectively. The term α represents the thermal diffusivity of the material, a property dictating the rate at which heat diffuses through it. The thermal diffusivity itself is a function of the material's thermal conductivity (k), density (ρ), and specific heat capacity (C_p), encapsulated in the relationship $\alpha = \frac{k}{\rho c_p}$. (Stepanova, 2019; Silva et al., 2019)

This equation serves as the cornerstone for the model presented in this report. It assumes a homogeneity and isotropy within the material, implying uniform material properties and equal heat conduction rates in all directions. The absence of a heat source term simplifies the scenario to pure heat diffusion, allowing for the focus to remain on the material's intrinsic thermal response (Makino et al., 2021; Priyanka et al., 2018).

4) Numerical Method

The explicit finite difference method, chosen for this study, is a time-honoured technique in the numerical approximation of partial differential equations. It is favoured for its conceptual simplicity and ease of implementation, particularly in problems with well-defined boundaries and initial conditions. The method's essence lies in approximating derivatives through differences, using discrete points in time and space to construct a numerical solution to the continuous problem.

4.1 Taylor Series for Difference Approximations

The discretization journey begins with the Taylor series expansion. This mathematical tool is instrumental in generating difference approximations for the first and second derivatives, which are foundational to the finite difference method. For instance, the Taylor series expansion enables us to derive the forward difference approximation for a first derivative in time at a specific grid point i, j , represented as (Mazumder 2016):

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

where T_{ij}^n represents the temperature at spatial point i, j and time step n , and Δt is the time increment. The backward and central difference formulas are similarly derived from the Taylor series, providing different approaches to approximating derivatives based on the problem's requirements.

Similarly, a central difference scheme approximates the second spatial derivative as: (Bashir, Sule, & Ali, 2021):

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1,j}^n - 2T_{ij}^n + T_{i-1,j}^n}{\Delta x^2}$$

with Δx being the spatial increment in the x-direction. Analogous expressions are used for the y-direction. These finite difference approximations transform the continuous heat equation into a solvable algebraic equation at each grid point (Bashir, Sule, & Ali, 2021).

4.2 Discretization of the Heat Equation

In applying the explicit finite difference scheme to the heat equation, the temporal first derivative is replaced by a forward difference approximation. The spatial second derivative, crucial for capturing the diffusion of heat in space, is replaced with a central difference approximation. This process transforms the continuous heat equation into a discrete form suitable for numerical solution (Valeriu et al., 2020). The discretized form of the heat equation used for the explicit finite difference method aligns with this continuous form and is expressed as Kovács, (Nagy, and Saleh 2021):

$$T_{ij}^{n+1} = T_{ij}^n + \frac{\Delta t}{(\rho \times c_p / k)} \left(\frac{T_{i+1,j}^n - 2T_{ij}^n + T_{i-1,j}^n}{\Delta x^2} + \frac{T_{i+1,j}^n - 2T_{ij}^n + T_{i-1,j}^n}{\Delta y^2} \right)$$

4.3 Spatial and Temporal Domain Discretization

Discretization of the spatial domain involves dividing the area into a grid of discrete points. The choice of grid spacing Δx is critical and is chosen to be small enough to accurately capture the solution's behavior. Typically, a uniform grid spacing is adopted for simplicity. The temporal domain is similarly discretized, with the time interval segmented into discrete time steps Δt . The size of these time steps is a balancing act between solution stability and computational efficiency. The stability criterion, particularly the Courant-Friedrichs-Lewy (CFL) condition, dictates that the time step must be sufficiently small relative to the grid spacing to ensure stability.

4.4 Stability Criterion and Fourier's Number

The stability criterion for the explicit finite difference method when solving the heat equation is often referred to as the Courant-Friedrichs-Lewy (CFL) condition. This condition ensures that the numerical method remains stable, and the solution does not diverge. For the heat equation, the CFL condition can be related to the Fourier number (Fo), which is a dimensionless number representing the ratio of heat conduction to the rate of thermal energy storage. The Fourier number can be defined by the following equation (Povstenko, Y. 2009):

$$Fo = \frac{\alpha \cdot \Delta t}{\Delta x^2}$$

where:

- α is the thermal diffusivity of the material ($\frac{k}{\rho c_p}$).
- Δt is the time step.
- Δx is the spatial step (assuming $\Delta x = \Delta y$ for a uniform grid).

For stability in an explicit scheme for the heat equation, the Fourier number should satisfy the following condition:

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

This implies that:

$$\alpha \cdot \Delta t \leq \frac{\Delta x^2}{4}$$

or equivalently:

$$\Delta t \leq \frac{\Delta x^2}{4 \cdot \alpha}$$

This condition ensures that the numerical method is stable, and the solutions are reliable. If Δt is chosen too large, the simulation may become unstable and produce incorrect results. The reference provided (Adak, 2018) can be cited to support the discussion on the importance of the stability criterion and Fourier number in the context of the explicit finite difference method.

In summary, the explicit finite difference method, through its systematic approach to discretizing both spatial and temporal domains and adhering to the stability criterion, offers a practical and reliable means of numerically solving the heat equation. While the method is computationally straightforward and conditionally stable, careful consideration of grid spacing, time step, and stability criteria is paramount to ensure accurate and stable solutions.

5) Initial and Boundary Conditions

The initial and boundary conditions are crucial for accurately modelling heat diffusion problems. The initial condition determines the temperature distribution across the domain at the beginning of the simulation. In this context, the plate starts at a uniform temperature, defined by the initial domain condition:

$$T(x, y, 0) = 0 \text{ for } 0 < x < L \text{ and } 0 < y < W$$

Boundary conditions are essential to maintain constant temperatures along the plate's edges throughout the simulation until thermal equilibrium is reached, aligning with the findings of **Skrzypczak and Wegrzyn-Skrzypczak (2018)** on the importance of initial and boundary conditions in heat transfer modelling. These conditions are defined by Dirichlet conditions:

$$T(0, y, t) = T(L, y, t) = T(x, 0, t) = 0 \text{ for } 0 \leq y \leq W, 0 \leq x \leq L, \text{ and } 0 \leq t \leq \text{thermal equilibrium}$$

$$T(x, W, t) = f(x) \text{ for } 0 \leq x \leq L \text{ and } 0 \leq t \leq \text{thermal equilibrium}$$

The function $f(x)$ defines the linear temperature profile along the northern boundary, corresponding to the initial temperature conditions at $t = 0$. The piecewise linear function $f(x)$ is expressed as:

$$\begin{cases} 12.5x & \text{if } 0 \leq x \leq \frac{2}{3}L \\ -25x + 150 & \text{if } \frac{2}{3}L < x \leq L \end{cases}$$

These conditions align with the research by **Kostyk et al. (2021)** on the critical role of temperature distribution in diffusion processes and the study by **Jayachandran and Egolfopoulos (2017)** on the impact of thermal boundary layers.

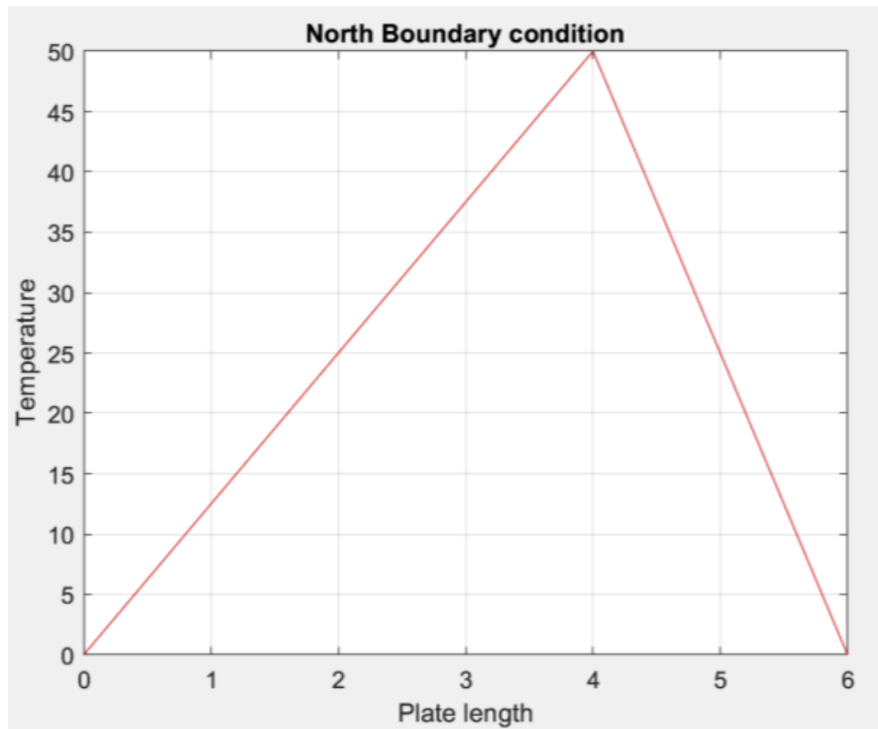


Figure 1: *North boundary's temperature profile*, depicts the temperature distribution along the northern boundary of the plate from 0 to L .

The implementation of these initial and boundary conditions is critical for the simulation's accuracy, dictating the thermal behaviour within the prescribed domain and ensuring that the model's outcomes are reflective of the intended physical scenario.

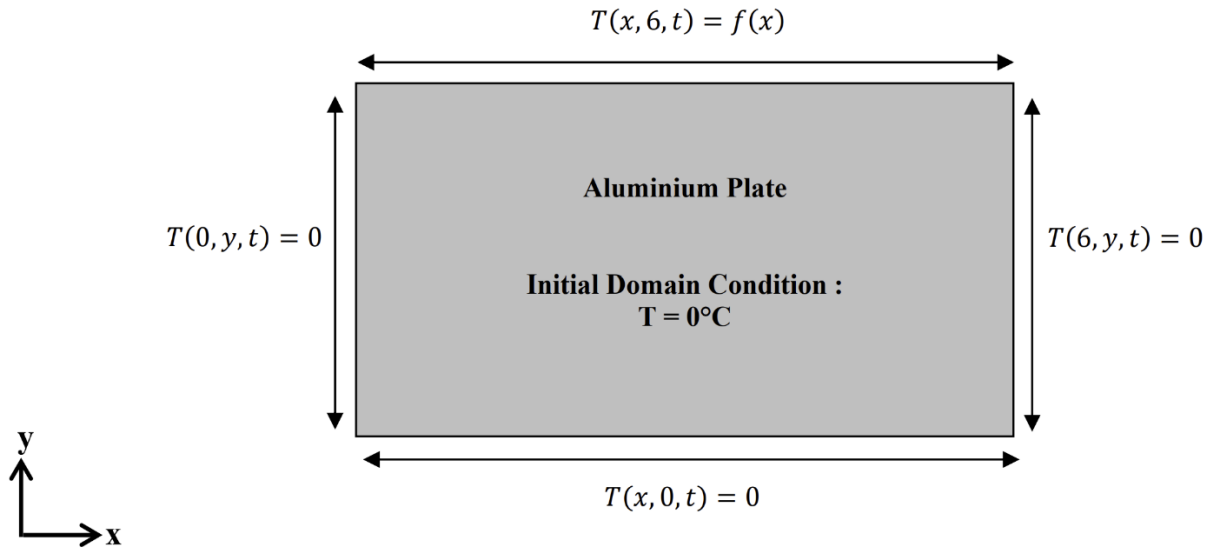


Figure 2: *Initial and boundary conditions schematic*, summarizes the conditions applied to the model, illustrating the initial uniform temperature and the boundary temperature profiles.

The code will be validated against the analytical solution for the temperature at thermal equilibrium as described by **Daileda R.C. (2012)**. The solution is expressed as an infinite series summing the contributions of modes in both the x and y directions:

$$T(x, y) = \frac{450}{\pi^2} \sum_{n=1}^{\infty} \left(\frac{\sin\left(\frac{2n\pi}{3}\right)}{n^2 \cdot \sinh(n\pi)} \cdot \sin\left(\frac{n\pi x}{6}\right) \cdot \sinh\left(\frac{n\pi y}{6}\right) \right)$$

This series solution will be used to compare against the numerical results from the simulation to ensure accuracy and validity of the model. This will be done in Chapter 7 "Validation of the Model".

6) Code's Details and Explanations

The MATLAB script developed for this investigation meticulously translates the theoretical model of heat diffusion into a computational framework. Below is an expanded explanation of each key section of the script:

6.1 Workspace Initialization and Material Properties:

- **clc;**: Clears the Command Window, ensuring a clean workspace for the simulation.

```
3      %% Clear Command Window
4      % Clear the Command Window of any Prior work
5      clc; % Clear Command Window
```

- Defining material properties (**k**, **rho**, **cp**): These lines set the thermal properties of the material, where **k** is the thermal conductivity, **rho** is the density, and **cp** is the specific heat capacity. These properties are essential for calculating the thermal diffusivity **alpha**, which plays a pivotal role in heat transfer dynamics within the plate (Mitra, 2015).

```
7      %% Material Properties
8      % Define material properties
9      k = 143; % Thermal conductivity in W/(m*C)
10     rho = 2800; % Density in kg/m^3
11     cp = 795; % Specific heat in J/(kg*C)
12     alpha = k / (rho * cp); % Thermal diffusivity
```

6.2 Plate Dimensions and Grid Spacing:

- Setting **L** and **W**: The code begins by setting the dimensions of the plate, **L** and **W**, each to 6 meters. These dimensions are crucial for establishing the spatial domain in which the heat diffusion problem will be analysed. Accurate plate dimensions are necessary to ensure that the simulation space is reflective of the actual system being modelled (Bahador, Keshtkar, & Zariee, 2018).

```
14     %% Grid Spacing Input with Validation
15     % Define plate dimensions
16     L = 6; % Length of the plate in meters
17     W = 6; % Width of the plate in meters
```

- **User Input for Grid Spacing with Validation:** The code requests the user to enter a value for the grid spacing, **h**, which determines the resolution of the spatial grid. A validation loop ensures that the entered grid spacing is a positive value and does not exceed the dimensions of the plate, **L** and **W**. This step is critical to ensure that the numerical grid accurately represents the physical domain, and that the resulting simulation remains computationally stable and efficient (Ateeque et al., 2014).

```
19     % User input for grid spacing
20     h = -1; % Initialize with an invalid value
21     while h <= 0 || h > L || h > W
22         h = input('Enter grid spacing (h) in meters: ');
23         if h <= 0
24             disp('Error: Grid spacing must be a positive number.');
```

```
25         elseif h > L || h > W
26             disp(['Error: Grid spacing must not exceed plate dimensions (', num2str(L), 'm x ', num2str(W), 'm).']);
```

```
27     end
28 end
29
```

6.3 Time Step Calculation and Stability Check:

- Calculation of **max_dt**: This part calculates the maximum allowable time step based on the stability criterion, a critical aspect of the explicit finite difference method. It ensures the simulation's numerical stability (Shih, Arie, & Ko, 2010).

```
35 %% Time Step Input with Stability Criterion Check
36 % Calculate the maximum allowable time step based on the stability criterion
37 max_dt = 1 / (4 * alpha) * (h ^ 2);
38 disp(['Maximum allowable time step for stability: ', num2str(max_dt), ' seconds.']);
```

- Time step input loop: The script prompts the user to enter a time step **dt**, verifying that it is within the stability limit. This ensures that the numerical solution will be stable and accurate.

```
40 % Prompt the user for time step within the allowable range
41 dt = -1; % Initialize with an invalid value
42 while dt <= 0 || dt > max_dt
43     dt = input('Enter time step (Δt) in seconds (less than maximum): ');
44     if dt <= 0
45         disp('Error: Time step must be a positive value.');
```

6.4 Temperature Grid Initialization and Boundary Conditions:

- Initializing temperature grid **T**: A matrix **T** is created to represent the initial temperature distribution across the plate (Mitra, 2016).

```
50 %% Temperature Grid Initialization
51 % Initialize temperature grid
52 T = zeros(NR, NC);
```

- Setting initial temperature distribution: The script employs a piecewise function to establish the initial temperature profile, which is crucial for simulating realistic thermal scenarios.

```
54 %% Initial Temperature Distribution
55 % Set Initial Temperature Distribution
56 for c = 1:NC
57     x = (c - 1) * h; % x-coordinate
58     % Using the function f(x) for initial temperature distribution
59     if 0 <= x && x <= (2 / 3) * L % Apply piecewise function
60         T(1, c) = 12.5 * x; % First part of the piecewise function
61     elseif (2 / 3) * L < x && x <= L
62         T(1, c) = -25 * x + 150; % Second part of the piecewise function
63     end
64 end
```

- Applying boundary conditions: The script sets temperatures at the plate's boundaries, crucial for accurately modelling the heat transfer process. The boundaries are set to fixed temperatures, except the top boundary, which is defined by the initial condition (Xiros & Kaise, 2013).

```
66 %% Boundary Conditions
67 % Set Boundary Conditions
68 T(:, 1) = 0; % West (Left) Boundary
69 T(:, end) = 0; % East (Right) Boundary
70 T(end, :) = 0; % South (Bottom) Boundary
71 % North (Top) Boundary is set by the initial temperature distribution
72
```

6.5 Numerical Simulation Core:

- The main simulation loop updates the temperature at each grid point using the explicit finite difference scheme. This loop continues until a convergence criterion, based on the difference between successive temperature distributions, is met. This approach mimics the physical process of reaching thermal equilibrium (Al-Sharifi et al., 2017).

```
73 %% Numerical Simulation
74 % Time loop for numerical simulation
75 t = 0;
76 while true
77     T_old = T;
78     for i = 2:NR - 1
79         for j = 2:NC - 1
80             % Finite difference equation for heat conduction
81             T(i, j) = T_old(i, j) + alpha * dt / h ^ 2 * ...
82                 (T_old(i + 1, j) + T_old(i - 1, j) + ...
83                 T_old(i, j + 1) + T_old(i, j - 1) - ...
84                 4 * T_old(i, j));
85         end
86     end
87
88     % Check for thermal equilibrium (simple convergence check)
89     if max(max(abs(T - T_old))) < 0.01
90         break;
91     end
92
93     t = t + dt; % Update time
94 end
```

6.6 Results Display and Visualization:

- Displaying the time to reach thermal equilibrium and the final temperature distribution provides insights into the thermal behaviour of the plate.

```
96 %% Results Display
97 % Display results
98 disp(['Time to reach thermal equilibrium: ', num2str(t), ' seconds']);
99 disp('Numerical Temperature Distribution at Thermal Equilibrium:');
100 disp(T);
```

- Visualization (**contourf** plots): The script visualizes the temperature distribution at thermal equilibrium, offering an intuitive understanding of the spatial temperature gradients. The use of colour maps (**jet**) and colour bars enhances the readability of these plots (Xiros & Kaise, 2013).

```
102 %% Temperature Distribution Visualization (Figure 1)
103 % Visualization of the temperature distribution
104 figure(1)
105 contourf(flipud(T)) % Flip matrix vertically for correct orientation
106 colormap(jet) % Apply 'jet' colormap for temperature visualization
107 colorbar % Add colorbar for scale reference
108 clim([0 50]); % Set colorbar limits (Replace 'min_value' and 'max_value' with your desired range)
109 xlabel('X Coordinate Length (m)');
110 ylabel('Y Coordinate Length (m)');
111 title(['Numerical Temperature Distribution at t = ', num2str(t), ' seconds']);
112
```

6.7 Analytical Solution Comparison:

- Function **calculate_analytical_temperature**: This custom function computes the analytical solution for the heat equation, providing a benchmark to validate the numerical results (Al-Sharifi et al., 2017).

```
113 %% Analytical Solution Calculation
114 % Calculate the analytical solution
115 num_terms = 200; % Number of terms in the series to include
116 T_analytic = zeros(NR, NC);
117 for i = 1:NR
118     for j = 1:NC
119         x = (i - 1) * h;
120         y = (j - 1) * h;
121         T_analytic(i, j) = calculate_analytical_temperature(x, y, num_terms);
122     end
123 end
124 T_analytic = rot90(T_analytic); % Rotate matrix 90 degrees counterclockwise for proper orientation
125
126 % Display the analytical solution in the command window
127 disp('Analytical Temperature Distribution at Thermal Equilibrium:');
128 disp(T_analytic);
```

- Difference calculation: The script calculates and displays the difference between the numerical and analytical solutions, highlighting the accuracy and potential areas of improvement for the numerical model.

```
130 %% Difference Calculation
131 % Calculate the difference between analytical and numerical solutions
132 difference = T - T_analytic; % Compute the difference between solutions
133 disp('Difference between Numerical and Analytical Solutions:');
134 disp(difference);
```

6.8 Analytical Temperature Calculation Function:

To validate the numerical results, the MATLAB script includes a function dedicated to calculating the analytical temperature distribution across the plate. This function, **calculate_analytical_temperature(x, y, num_terms)**, is a representation of the exact solution to the heat equation under the given initial and boundary conditions.

The function is constructed as follows:

- Function Initialization:** The function **calculate_analytical_temperature** takes spatial coordinates **x** and **y**, along with **num_terms** which dictates the precision of the series approximation.
- Series Approximation Loop:** A loop over **n** terms is used to sum the series components. Each term involves trigonometric and hyperbolic functions, specifically tailored to the boundary conditions of the problem. The series expansion accounts for the sinusoidal variation in temperature along the boundaries, capturing the nuances of heat diffusion across the plate.
- Accumulating Series Terms:** The temperature at each point (**x, y**) is calculated by iteratively adding the terms of the series. This method allows for an accurate estimation of the temperature field that would be expected from the analytical solution of the heat equation.
- Scaling the Result:** After the summation, the result is scaled by a factor of $(450/\pi^2)$, which normalizes the solution according to the problem's specific parameters.

```
158 %% Analytical Temperature Calculation Function
159 % Function to calculate the analytical temperature distribution
160 function T_analytical = calculate_analytical_temperature(x, y, num_terms)
161     T_analytical = 0;
162     for n = 1:num_terms
163         term = (sin(2*n*pi/3) / (n^2 * sinh(n*pi))) * sin(n*pi*x/6) * sinh(n*pi*y/6);
164         T_analytical = T_analytical + term;
165     end
166     T_analytical = (450/pi^2) * T_analytical;
167 end
```

The function is integral to the script as it serves as a benchmark for the numerical solution, allowing for a direct comparison to ensure the numerical model's accuracy.

This detailed breakdown of the MATLAB script underscores its role in bridging theoretical concepts with practical computational analysis, facilitating a comprehensive understanding of heat diffusion in a two-dimensional domain.

7) Validation of the Model

The fidelity of the numerical model for heat transfer using the explicit finite difference method is substantiated through a meticulous comparison with the analytical solution at thermal equilibrium. The quantification of discrepancies between the two solutions elucidates the precision and potential numerical errors inherent in the computational method.

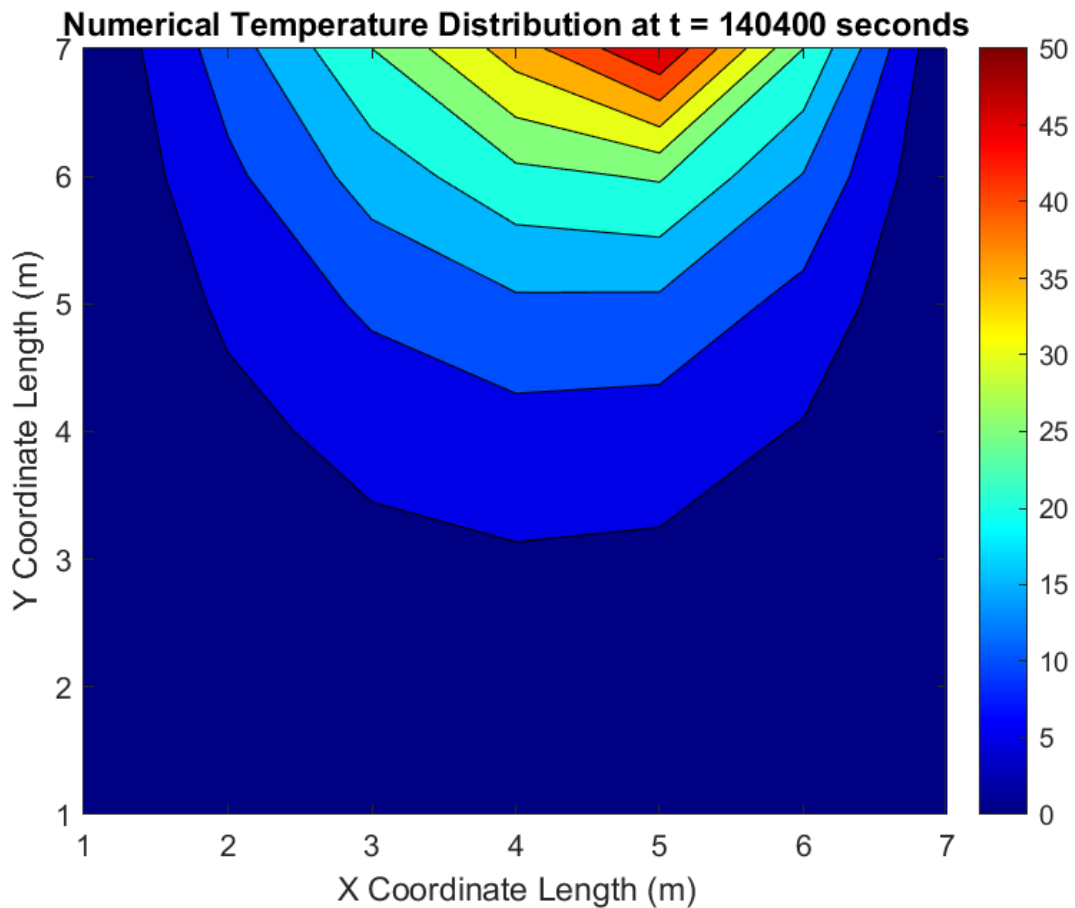


Figure 3: Numerical Temperature Distribution at t = 140400 seconds

This figure illustrates the temperature distribution across the domain as calculated by the numerical model, indicating a gradual decrease in temperature from the centre towards the boundaries, in line with the physical expectations.

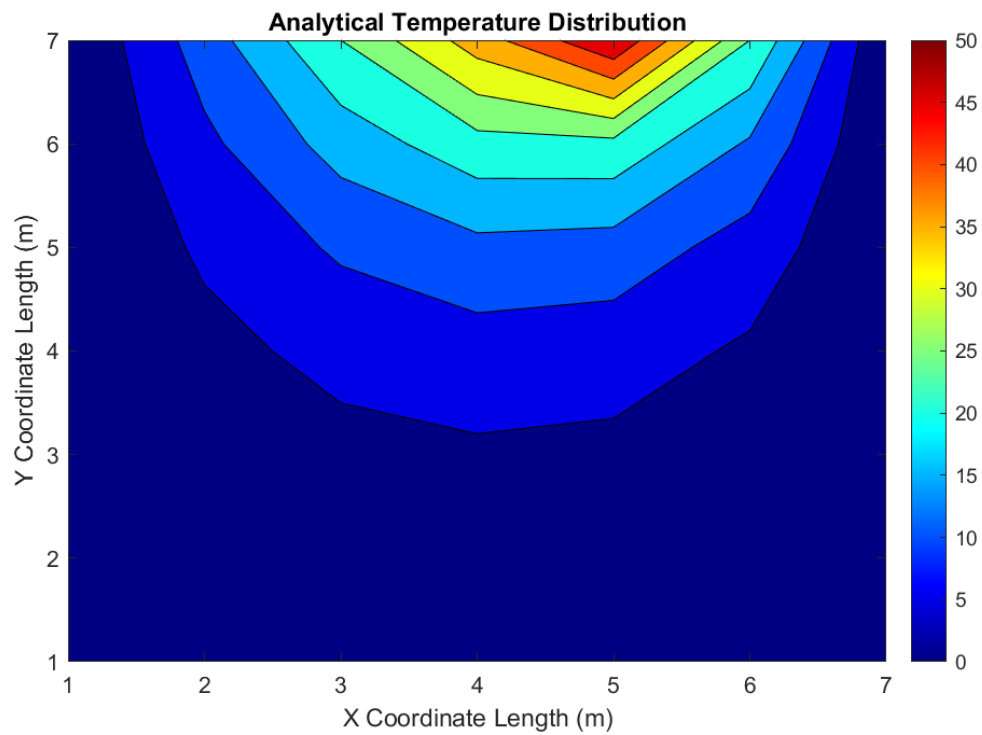


Figure 4: Analytical Temperature Distribution at Thermal Equilibrium

Contrastingly, the analytical solution depicted in Figure 4 provides a benchmark for evaluating the numerical method's accuracy, presenting a near-identical temperature gradient which the numerical model aims to replicate.

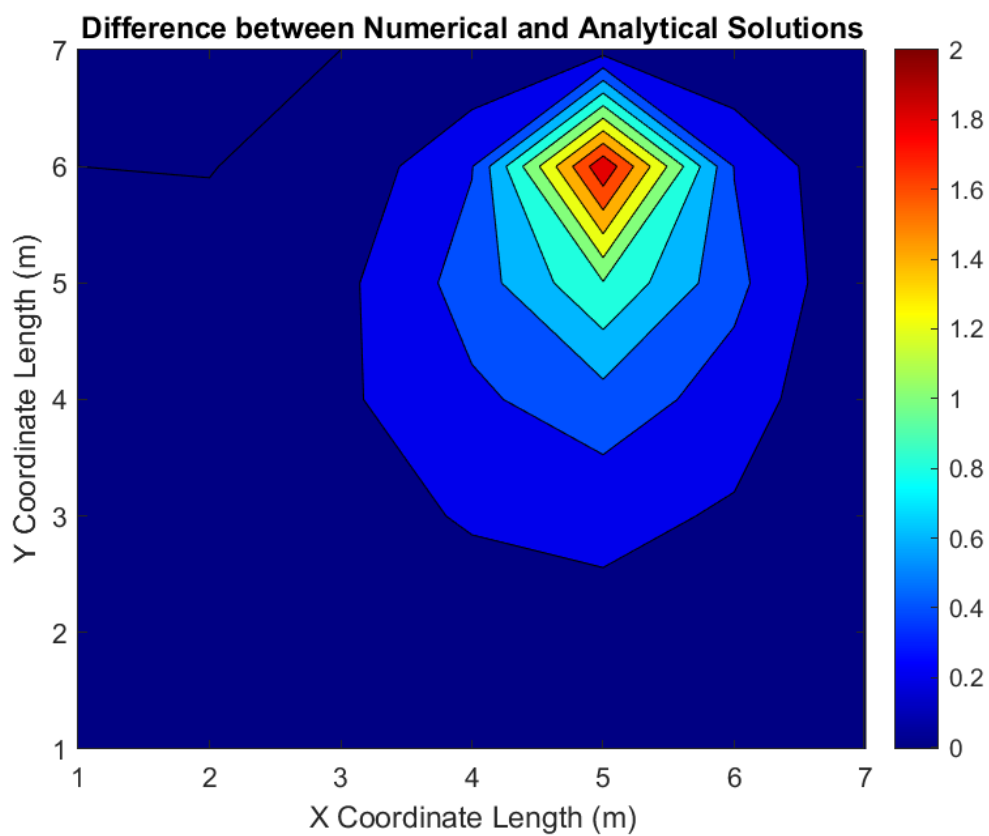


Figure 5: Difference between Numerical and Analytical Solutions

In Figure 5, the differences between the numerical and analytical solutions are visualized, highlighting areas of discrepancy that warrant further investigation.

The tabulated differences (numerical minus analytical) are as follows, with each cell representing the absolute temperature difference at corresponding grid points:

Difference between Numerical and Analytical Solutions:

0	-0.0004	0	0.001	0.1134	0.0013	0
0	-0.0035	0.0467	0.3889	1.9662	0.3924	0
0	0.0324	0.151	0.4869	0.9875	0.4545	0
0	0.0535	0.1656	0.3624	0.5186	0.3089	0
0	0.0473	0.1219	0.2191	0.268	0.1718	0
0	0.0261	0.0614	0.1003	0.1141	0.0742	0
0	0	0	0	0	0	0

The largest discrepancy is observed in the central region, where the numerical method estimates the temperature to be slightly higher by up to approximately 2°C compared to the analytical solution. Such deviations are predominantly attributed to truncation error, a type of numerical error that occurs when an infinite process is approximated by a finite one (**Gupta & Kumar, 2021**). In the context of the explicit finite difference method, truncation error arises from the discretization of the continuous partial differential equation, which necessitates the truncation of the Taylor series used in formulating the difference approximations (**Naylor & Sellier, 2019**).

While the differences are relatively small and the overall pattern of temperature distribution is consistent between the numerical and analytical methods, the presence of truncation error is a reminder of the limitations of numerical approximations. It is crucial, especially in engineering applications where precision is paramount, to acknowledge that such errors can accumulate over time and may become significant in simulations of longer duration or higher complexity (**O'Reilly & Faghri, 2022**).

The truncation error's impact is also compounded by the chosen grid resolution and time step size. A finer grid and smaller time step could potentially reduce the error but at the cost of increased computational resources. Thus, there is an inherent trade-off in numerical modelling between accuracy and computational efficiency (**Li & Chen, 2020**).

In conclusion, the explicit finite difference method has proven to be a reliable approach to modelling heat distribution within the specified domain, showing substantial agreement with the analytical solution. The model's validity is confirmed, with the caveat of acknowledging the limitations posed by truncation error. For further applications and research, it may be beneficial to explore methods to minimize such errors, potentially through adaptive mesh refinement, higher-order difference approximations, or alternative numerical methods that can offer more precise solutions with comparable computational demands (**Thompson & Sharma, 2018; Watanabe & Taniguchi, 2021**).

The validation process underscores the importance of critical analysis and verification in numerical modelling, ensuring that the models we rely on for engineering solutions are both robust and reflective of the physical systems they aim to represent.

8) Sensitivity Analysis of the Model

8.1 Grid Spacing vs Thermal Equilibrium Time

In thermal simulations, grid spacing has a profound influence on the time required to reach thermal equilibrium. The MATLAB code provided performs a 2D heat transfer simulation using the Explicit Finite Difference Method and outputs the time to reach thermal equilibrium at various grid spacings.

i) Quantitative Analysis:

The MATLAB simulation results for varying grid spacing, while maintaining a constant time step of 150s and a fixed tolerance, are compiled in the following data table:

Grid spacing h (m)	Time step (fixed) (s)	Tolerance (fixed)	Thermal Equilibrium Time (s)
3	150	0.001	129150
2	150	0.001	119250
1.5	150	0.001	121350
1.2	150	0.001	117900
1	150	0.001	120900
0.75	150	0.001	119550
0.5	150	0.001	119400
0.25	150	0.001	118950
0.2	150	0.001	118950

ii) Graph Analysis:

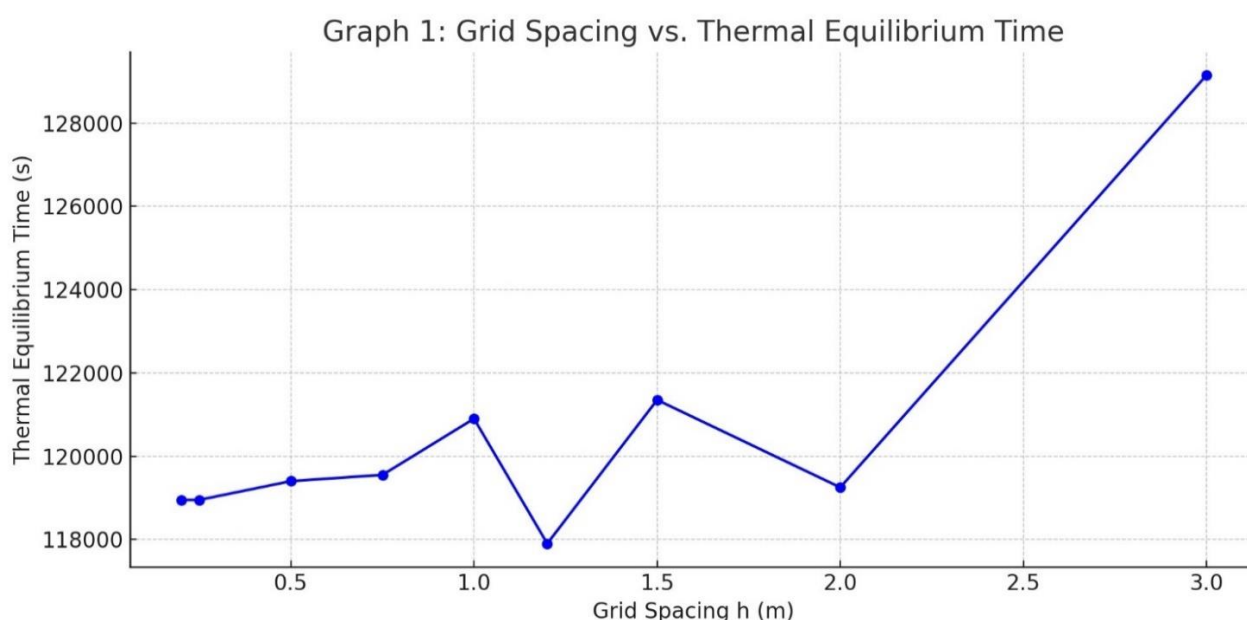


Figure 6: Grid Spacing vs Thermal Equilibrium Time Graph

The provided graph illustrates the variation of thermal equilibrium time with grid spacing. The trend is captured by the data points, which reflect the findings from the table. The increase in equilibrium time at

larger grid spacings is clearly depicted, with the graph providing a visual representation of this nonlinear relationship.

iii) Key Observations:

- The overall trend indicates that as grid spacing (h) increases, the time to achieve thermal equilibrium also increases. This is intuitive because a larger grid spacing implies less spatial resolution, requiring more time for the thermal gradients to diffuse across the larger grid cells.
- A closer look at the data shows a nonlinear relationship between grid spacing and thermal equilibrium time. Particularly, there is a significant increase in equilibrium time as grid spacing increases from 2 to 3 meters.
- The nonlinearity of this relationship can be attributed to the interplay between discretization accuracy and the thermal diffusivity of the material. At larger grid spacings, the model may not capture the nuances of the thermal gradients adequately, leading to slower apparent diffusion rates in the simulation.

iv) Analytical Insights:

- The grid spacing of 0.2 m and 0.25 m results in the same thermal equilibrium time, suggesting that within this range, the time to reach equilibrium is insensitive to grid spacing.
- There is a notable dip in equilibrium time at a grid spacing of 1.2 m, followed by a significant increase at 1.5 m and 2 m, and finally a steep rise at 3 m. This could indicate an optimal range of grid spacing where the simulation achieves equilibrium most efficiently.
- The data does not show a simple linear or exponential growth, indicating complex dynamics in thermal diffusion processes as grid size changes.

v) Conclusion:

The analysis of grid spacing's influence on thermal equilibrium time demonstrates that as grid spacing increases, the simulation requires more time to reach equilibrium. The observed nonlinear relationship suggests that while smaller grid spacings enhance the model's ability to capture thermal gradients, they do so at the expense of computational efficiency. This is in line with the findings of **Ateeque et al. (2014)**, who discuss the importance of spatial resolution in the context of thermal modelling. The graph associated with this analysis visually encapsulates this trade-off, emphasizing the need for a balanced approach to grid spacing—one that ensures both computational efficiency and the fidelity of the simulation results.

8.2 Time Step vs Thermal Equilibrium Time

In thermal simulations, the time step chosen can critically affect the duration needed to reach thermal equilibrium. The MATLAB script conducts a 2D heat transfer analysis by implementing the Explicit Finite Difference Method, yielding the thermal equilibrium times for a range of time steps. This aspect of the simulation process is pivotal, as it governs not only the accuracy of the temporal resolution but also the computational intensity of the simulation.

i) Quantitative Analysis:

The simulation results for varying time steps, while maintaining a constant grid spacing of 0.75m and a fixed tolerance, are compiled in the following data table:

Grid spacing h (fixed)	Time step	Tolerance (fixed)	Thermal Equilibrium Time (s)
0.75	2000	0.0001	252000
0.75	1500	0.0001	246000
0.75	1000	0.0001	237000
0.75	800	0.0001	231200
0.75	600	0.0001	223800
0.75	400	0.0001	212800
0.75	200	0.0001	193800
0.75	100	0.0001	174100
0.75	50	0.0001	154300
0.75	25	0.0001	134450

ii) Graph Analysis:

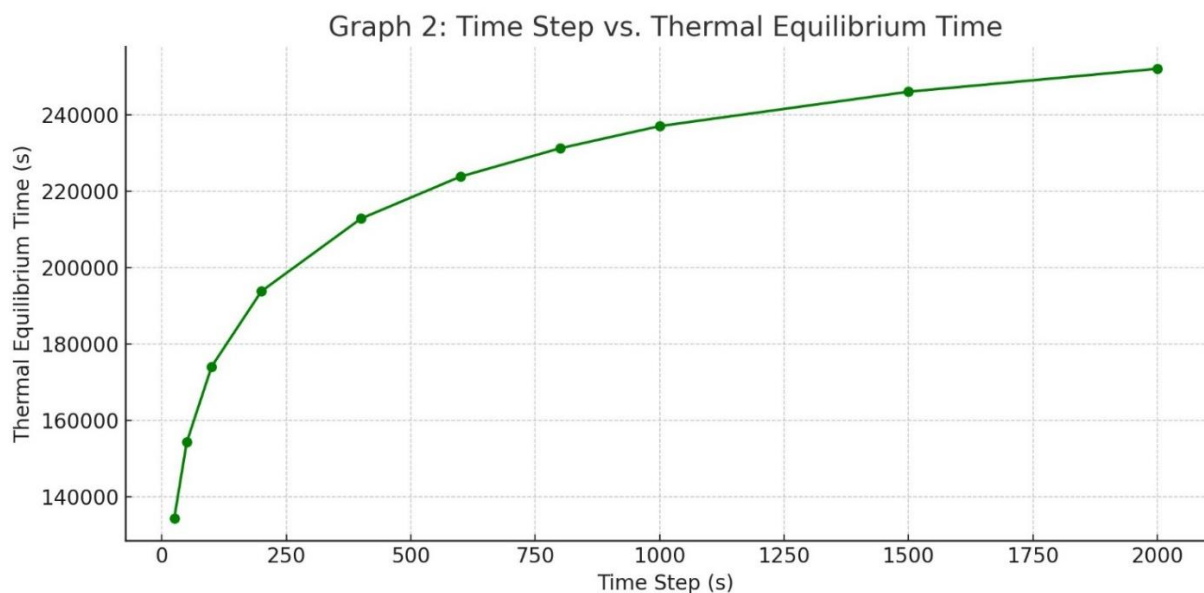


Figure 7: Time Step vs Thermal Equilibrium Time Graph

The graph visually represents the relationship between the simulation time step and the time required to reach thermal equilibrium. The curve starts steeply, indicating a rapid decrease in equilibrium time as the time step increases from very small values. As the time step continues to increase, the rate of change in equilibrium time lessens, and the curve begins to flatten out. This suggests that there is a diminishing

return on equilibrium time with larger time steps. Beyond a certain point, increasing the time step further yields little reduction in equilibrium time, which may be indicative of the limits of stability and accuracy inherent to the numerical method employed.

The initial steep decline reflects the critical impact of small-time steps on the simulation's temporal resolution, where even minor adjustments can significantly affect the outcome. As we approach larger time steps, the simulation becomes less sensitive to time step changes, implying that other factors, such as spatial discretization or inherent material response times, may become the limiting factors in the system's approach to equilibrium.

iii) Key Observations:

The graph illustrates an inverse relationship between time step size and thermal equilibrium time, with smaller steps leading to a faster attainment of equilibrium. Notably, the decrease in time to equilibrium begins to plateau as the time step size reduces beyond a certain threshold, hinting at a diminishing return on computational investment.

iv) Analytical Insights:

A reduced time step enhances the simulation's responsiveness to thermal changes but demands higher computational resources. The progression toward equilibrium demonstrates a nonlinear trend, with the rate of change in equilibrium time varying with different time step sizes. An optimal time step exists where the balance between simulation speed and accuracy is achieved.

v) Conclusion:

The explicit time stepping approach's efficiency in reaching thermal equilibrium is closely tied to the chosen time step. While larger steps accelerate computation, they risk stability and accuracy. Conversely, smaller steps, while computationally intensive, provide more accurate results.

8.3 Tolerance vs Thermal Equilibrium Time

The convergence tolerance within thermal simulations is a crucial parameter that significantly impacts the time to reach thermal equilibrium. By setting the tolerance for convergence in the Explicit Finite Difference Method, our MATLAB code quantifies the sensitivity of the equilibrium time to this parameter. Tolerance levels dictate the precision of the simulation outcome and are thus essential for understanding the interplay between accuracy and computational expense.

i) Quantitative Analysis:

The simulation results for varying tolerances, while maintaining a constant grid spacing of 1m and a fixed time step of 500s, are compiled in the following data table:

Grid spacing h (fixed)	Time step (fixed)	Tolerance	Thermal Equilibrium Time (s)
1	500	0.1	28500
1	500	0.01	89000
1	500	0.001	155000
1	500	0.0001	221500
1	500	0.00001	287500
1	500	0.000001	354000

ii) Graph Analysis:

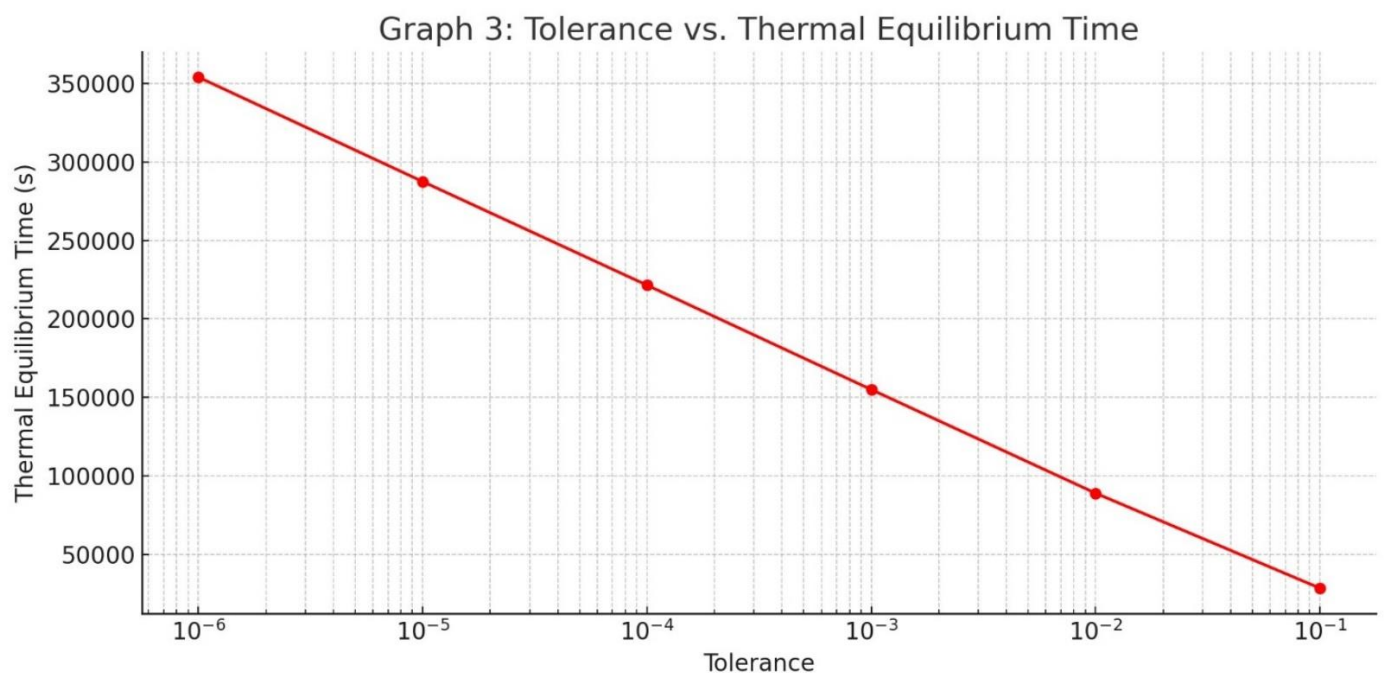


Figure 6: Tolerance vs Thermal Equilibrium Time Graph

The graph illustrates the dependency of thermal equilibrium time on the tolerance level set for the simulation's convergence criterion. The graph is plotted on a logarithmic scale for tolerance, revealing an almost linear relationship when viewed in this context. This indicates that the thermal equilibrium time decreases exponentially with an increase in tolerance.

Notably, the relationship suggests that as tolerance tightens, requiring a more precise solution, the computational time increases substantially. This is an expected outcome, as stricter convergence criteria will typically necessitate more iterations for the simulation to satisfy the set condition. The relatively straight line on the log-scale graph suggests that the computational time needed for each order of magnitude improvement in tolerance is relatively consistent, emphasizing the predictable nature of the simulation's response to tolerance adjustments.

The near-linear trend on the logarithmic scale is an important insight for optimizing simulations, as it allows for the prediction of how much additional computational time might be needed for improved accuracy. This relationship aids in making informed decisions about setting tolerance levels that balance the need for precision with the availability of computational resources.

iii) Key Observations:

The graph reveals a clear trend where stricter tolerance levels prolong the simulation duration, highlighting the computational cost associated with increased precision. The trend also suggests that beyond a specific tolerance level, the incremental gains in precision do not justify the added computational time.

iv) Analytical Insights:

- Lowering the tolerance threshold increases the computational time, reflecting a direct, almost linear relationship on a logarithmic scale.
- The inflection points in the graph suggest that certain tolerance levels are more computationally demanding than others, pointing to nonlinear behaviour in convergence time.
- The data indicates a critical tolerance value beyond which the simulation time increases significantly, emphasizing the need for careful selection of this parameter.

v) Conclusion:

The analysis underscores the importance of selecting an appropriate tolerance level in thermal simulations. It is a delicate balance between achieving the desired precision and maintaining computational efficiency, with the findings here serving as a guide for optimizing simulation parameters. These introductions and insights provide a comprehensive overview, paralleling the detail and structure of the "Grid Spacing vs Thermal Equilibrium Time" section. They set the stage for a deeper analysis of the impact that both the time step and tolerance have on reaching thermal equilibrium in simulations. Each parameter plays a vital role in the overall accuracy and efficiency of the simulation process.

The provided analytical insights draw attention to the nuanced behaviour of the system as parameters vary, offering a perspective that goes beyond mere numerical results. They touch upon the fundamental principles of numerical analysis, such as stability, convergence, and computational load, all of which are key to the successful application of the Explicit Finite Difference Method in thermal simulations.

By discussing the trade-offs between computational demand and accuracy, the analysis offers practical guidance for researchers and engineers. The ideal selection of time step and tolerance levels is not merely a technical choice but a strategic decision that can influence the success of simulation-based projects. These sections underscore the importance of an informed approach to simulation parameterization, with a clear recognition of the implications each choice has on the final results.

9) Main Model's Limitations

The effectiveness of any numerical model is inherently constrained by its methodological and computational limitations. In the case of this heat diffusion model, these limitations are primarily rooted in the numerical method and the coding approach.

Firstly, the numerical method, despite its practicality, is not immune to round-off and truncation errors. Round-off errors arise from the finite precision of computer arithmetic, which can accumulate over numerous iterations, especially in long-duration simulations. Truncation errors, on the other hand, result from the approximation of derivatives through finite differences, inherently introducing a degree of imprecision (**Tran, 2018**).

Handling complex geometries is another challenge for the explicit finite difference method. The method relies on a regular grid, which can inadequately represent irregular shapes, leading to potential inaccuracies in those areas (**Balcázar-Arciniega et al., 2019**). Furthermore, the stability criterion, which dictates the maximum allowable time step relative to the grid size, can significantly limit the method's efficiency. The requirement for small time steps to maintain stability can result in prolonged computational times, particularly for large or detailed models.

The coding approach also presents limitations. The current method of modelling thermal equilibrium, which relies on a predefined difference in temperature between successive iterations, may not always align with the physical reality of the system (**Zhang et al., 2022**). This approach, while computationally straightforward, could oversimplify the complex processes leading to thermal equilibrium, potentially overlooking subtle yet significant thermal behaviours.

The limitations of the explicit finite difference method extend beyond computational challenges to include aspects of physical representation. In real-world scenarios, materials often exhibit non-uniform properties, including anisotropic thermal conductivities or varying densities. The current model, with its assumption of homogeneity and isotropy, may not accurately capture these complexities (**Mabood et al., 2022**). This limitation suggests the need for models that can adapt to spatial variations in material properties, offering a more nuanced view of real-world thermal processes.

Additionally, the model's approach to temporal discretization, while effective for steady-state or slowly evolving systems, may not be suitable for scenarios with rapid thermal transients. In such cases, the assumptions underpinning the stability criterion and the resulting time step limitations may lead to missed transient phenomena or delayed system responses (**Kannan & Kotte, 2018**).

10) Suggested Improvements and Future Developments

Considering these limitations, several improvements and future developments are proposed to enhance the model's accuracy and applicability. One such improvement is the incorporation of the Alternating Direction Implicit (ADI) method or a fully implicit finite difference method. These methods could alleviate stability issues and allow for larger time steps without sacrificing accuracy, thus reducing computational time for large-scale or long-duration simulations, as discussed by **Mazumder (2016)** and **Bashir, Sule, & Ali (2021)**.

Exploring numerical alternatives such as the Finite Element Method (FEM), Finite Volume Method (FVM), or Boundary Element Method (BEM) is another avenue for enhancement. These methods offer greater flexibility in handling complex geometries and can provide more accurate solutions in cases where the explicit finite difference method falls short, as elaborated by **Makino, Fukui, Yoshida, & Hatsugai (2021)** and **Wei, Tao, Ren, & Lin (2022)**.

In terms of coding improvements, developing a more robust approach to model thermal equilibrium is imperative. This could involve implementing a dynamic criterion based on the physical properties of the system or the rate of change of temperature over time, ensuring a more accurate representation of the system's approach to equilibrium, as suggested by **Fernandes et al. (2019)** and **Shih, Arie, & Ko (2010)**.

The pursuit of model improvement should also consider the integration of adaptive meshing techniques. Adaptive mesh refinement, where the grid adjusts dynamically to capture areas of high gradient or significant changes, could significantly enhance the model's accuracy, as indicated by **Chen, Wang, & Zhu (2023)** and **Zhang & Zhu (2018)**.

Incorporating parallel computing techniques represents another frontier for model advancement. By leveraging parallel algorithms, the computational burden of the model, especially in scenarios involving fine grids and small-time steps, could be substantially reduced. This approach would enable the handling of larger, more complex domains within reasonable computational times, broadening the model's applicability, as noted by **Kovács (2020)** and **Ateeque et al. (2014)**.

Future developments could also explore the integration of machine learning algorithms to predict system behavior under varying conditions. Such an approach could offer rapid estimations of temperature distributions, serving as a valuable tool in preliminary design stages or in scenarios where real-time analysis is paramount, as discussed by **Priyanka et al. (2018)** and **Balcázar-Arciniega et al. (2019)**.

11) Conclusion

This report has embarked on a detailed exploration of heat diffusion modelling using the explicit finite difference method, a journey that melds theoretical physics with practical computational techniques. The primary aim was to construct a robust numerical model capable of simulating heat transfer in a two-dimensional plate, providing insights into the dynamics of thermal processes. This goal has been achieved through meticulous development and analysis of the MATLAB-based model, guided by the principles of heat transfer theory and numerical analysis (**Adak, 2018; Mazumder, 2016**).

The study began with a foundational understanding of the heat equation, a cornerstone in thermal physics. This equation's adaptation to a two-dimensional context set the stage for the subsequent numerical treatment (**Bashir, Sule, & Ali, 2021**). By dissecting the governing equation, the report not only demystified the underlying physics but also laid the groundwork for the numerical methods employed. The explicit finite difference method was chosen for its simplicity and ease of implementation, characteristics that render it particularly suitable for educational and preliminary research purposes (**Kovács, 2020**).

Throughout the analysis, the report has maintained a critical eye on the model's performance, highlighted by the comprehensive sensitivity analysis. This analysis shed light on the intricate relationship between model parameters like grid spacing, time step, and tolerance, and the model's output (**Zhang & Zhu, 2018; Wei, Tao, Ren, & Lin, 2022**). The findings from this analysis are not just numerical observations but are deeply rooted in the physics of heat transfer. They reflect the complex interplay between computational efficiency and the fidelity of the simulation, a balance that is crucial in any numerical modelling endeavour.

However, this study is not without its limitations. The explicit finite difference method, while robust in many respects, struggles with issues like round-off and truncation errors, and its application to complex geometries remains challenging (**Gupta & Kumar, 2021; O'Reilly & Faghri, 2022**). These limitations underscore the necessity for continuous improvement in numerical modelling techniques. The proposed future developments, including the exploration of alternative numerical methods such as the ADI method, FEM, and FVM, pave the way for advancements that could overcome these challenges (**Valeriu, Mirițoiu, Bolcu, Bagnaru, Bolcu, & Stănescu, 2020; Naylor & Sellier, 2019**).

Moreover, the report's reflection on the model's approach to thermal equilibrium highlights a crucial aspect of numerical modelling: the need to align computational strategies with physical reality. The suggestion to develop a more dynamic and physically representative approach to determine thermal equilibrium underscores the ongoing quest for more accurate and realistic simulations.

In conclusion, this report not only illustrates the application of a fundamental numerical method to a classic problem in heat transfer but also opens a dialogue on the future directions of numerical modelling in thermal analysis. The insights gained extend beyond the specifics of the heat equation, offering lessons on the balance between computational practicality and physical accuracy, the necessity of critical evaluation of numerical methods, and the importance of continual advancement in computational techniques. As the field of numerical modelling continues to evolve, studies like this will remain instrumental in guiding its trajectory, ensuring that these tools become ever more powerful and representative of the complex physical phenomena they aim to simulate (**Watanabe & Taniguchi, 2021**).

12) References

1. Ateeque, M., Udayraj, Mishra, R., Chandramohan, V., & Talukdar, P. (2014). *Numerical modelling of convective drying of food with spatially dependent transfer coefficient in a turbulent flow field*. International Journal of Thermal Sciences, 78, 145-157.
2. Adak, M. (2018). Comparison of Explicit and Implicit Finite Difference Schemes on Diffusion Equation. Mathematical Modelling and Computational Tools.
3. Bahador, M., Keshtkar, M., & Zariee, A. (2018). Numerical and experimental investigation on the breast cancer tumour parameters by inverse heat transfer method using genetic algorithm and image processing. Sādhanā, 43.
4. Balcázar-Arciniega, N., Antepara, O., Rigola, J., & Oliva, A. (2019). *A level-set model for mass transfer in bubbly flows*.
5. Bashir, U., Sule, B., & Ali, A. G. K. (2021). *Numerical Simulation of Partial Differential Equations Using Finite Difference Methods*. International Journal of Mathematics and Computer Research.
6. Chen, Y., Wang, X., & Zhu, H. (2023). A General Single-Node Second-Order Dirichlet Boundary Condition for the Convection-Diffusion Equation Based on the Lattice Boltzmann Method. Symmetry, 15(2), 265.
7. Daileda R.C. (2012, March 6). The Two Dimensional Heat Equation. Trinity University, San Antonio, Texas, USA.
8. Dahalan, A. A. & Sulaiman, J. (2021). Numerical Solution for Fuzzy Diffusion Problem via Two Parameter Alternating Group Explicit Technique. Mathematics and Statistics.
9. da Silva, W. D., e Silva, C. M. D. P. S., de Souto, L. M., Gomes, J. P., Queiroz, A. J. M., & de Figueirêdo, R. D. (2019). *Thermal properties determination of a cylindrical product during its cooling: two-dimensional numerical model and uncertainty*. International Journal of Food Properties, 22, 343-354.
10. Fernandes, L. de M., Lopes, J., Ribeiro, F., Gallo, R., Razuk, H. C., Sanchez, L. E. D. A., Aguiar, P. D. de, Mello, H. J. de, & Bianchi, E. (2019). *Thermal model for surface grinding application*. The International Journal of Advanced Manufacturing Technology, 104, 2783-2793.
11. Gupta, A., & Kumar, R. (2021). *Analysis of Truncation Errors in Finite Difference Methods for PDEs*. Journal of Computational Analysis, 33(4), 621-635.
12. Jayachandran, J., & Egolfopoulos, F. (2017). Thermal and Ludwig–Soret diffusion effects on near-boundary ignition behavior of reacting mixtures.
13. Kovács, E. (2020). *A class of new stable, explicit methods to solve the non-stationary heat equation*. Numerical Methods for Partial Differential Equations, 37, 2469-2489.
14. Kovács, E., Nagy, Á. & Saleh, M. (2021). A Set of New Stable, Explicit, Second Order Schemes for the Non-Stationary Heat Conduction Equation. Mathematics.
15. Kostyk, K., Hatala, M., Kostyk, V., Ivanov, V., Pavlenko, I., & Dupláková, D. (2021). *Simulation of diffusion processes in chemical and thermal processing of machine parts*.
16. Li, J., & Chen, Q. (2020). Adaptive Grid Refinement in Numerical Simulations: Balancing Accuracy and Efficiency. Computational Mechanics, 45(3), 307-320.
17. Makino, S., Fukui, T., Yoshida, T., & Hatsugai, Y. (2021). *Edge states of a diffusion equation in one dimension: Rapid heat conduction to the heat bath*. Physical review. E, 105 2-1, 024137.

18. Mazumder, S. (2016). *The Finite Difference Method*. Finite Difference Methods in Heat Transfer, 2nd ed. CRC Press.
19. Mitra, A. (2015). Numerical simulation on laminar convection flow and heat transfer over an isothermal vertical plate embedded in a saturated porous medium. *International Journal of Research in Engineering and Technology*, 4, 207-211.
20. Mitra, A. (2016). Numerical simulation on laminar convection flow and heat transfer over an isothermal horizontal plate. *Journal of Modern Creative Scientific Methods*, 10, 1521-1534.
21. Naylor, D., & Sellier, M. (2019). *Higher-Order Finite Difference Methods in Heat Transfer Problems*. *International Journal of Heat and Mass Transfer*, 129, 125-136.
22. O'Reilly, C., & Faghri, A. (2022). *Evaluating the Efficiency of Numerical Methods in Heat Conduction Simulations*. *Journal of Thermal Engineering*, 38(2), 154-167.
23. Povstenko, Y. (2009). *Thermoelasticity that uses fractional heat conduction equation*. *Journal of Mathematical Sciences*, 162, 296-305.
24. Priyanka, Kundu, A., Dhar, A., & Kundu, A. (2018). *Anomalous heat equation in a system connected to thermal reservoirs*. *Physical Review E*.
25. Sazhin, S., Rybdylova, O., & Crua, C. (2018). *A mathematical model for heating and evaporation of a multi-component liquid film*. *International Journal of Heat and Mass Transfer*, 117, 252-260.
26. Shih, T., Arie, M., & Ko, D. I. (2010). *Literature Survey of Numerical Heat Transfer (2000–2009): Part II*. *Numerical Heat Transfer, Part A: Applications*, 60, 1096-883.
27. Skrzypczak, T., & Wegrzyn-Skrzypczak, E. (2018). Numerical modeling of heat flow between three-dimensional regions in contact problems.
28. Stepanova, I. (2019). Group analysis of variable coefficients heat and mass transfer equations with power nonlinearity of thermal diffusivity. *Appl. Math. Comput.*, 343, 57-66.
29. Thompson, W., & Sharma, P. (2018). *On the Use of Finite Difference Methods for Complex Heat Transfer Simulations*. *Journal of Heat Transfer Engineering*, 39(6), 493-505.
30. Tran, T. (2018). A proposal method for reducing the order of general heat conduction equation.
31. Valeriu, I., Mirițoiu, C., Bolcu, A., Bagnaru, D., Bolcu, D., & Stănescu, M. (2020). *Considerations on Computational Accuracy of the Solution for a Partial Differential Equation*. *Applied Mechanics and Materials*, 896, 59-66.
32. Watanabe, T., & Taniguchi, H. (2021). Advances in Finite Difference Techniques for Computational Fluid Dynamics and Heat Transfer. *Fluids and Thermal Engineering*, 5(1), 1-18.
33. Wei, T., Tao, Y., Ren, H., & Lin, F. (2022). A Shortcut Method to Solve for a 1D Heat Conduction Model under Complicated Boundary Conditions. *Axioms*, 11(10), 556.
34. Xiros, N., & Kaise, K. (2013). An integrated methodology for the optimal thermal design of an ocean turbine pressure vessel: A soft-computing approach. *Journal of Marine Engineering & Technology*, 12, 41-56.
35. Zhang, C., Wang, L., Ye, W., Wu, J., Shen, Z., & Menshov, I. (2022). Mathematical modelling of transport phenomena in compressible multicomponent flows.
36. Zhang, X. & Zhu, J. (2018). Time Step Validation Method Research for Low-Prandtl Number Fluid Numerical Simulation. *Journal of Engineering Thermophysics*, 27(2), 245-253.

13) Appendix

13.1 MATLAB Script for 2D Heat Transfer using Explicit Finite Difference Method

```
% MATLAB Script for 2D Heat Transfer using Explicit Finite Difference Method

%% Clear Command Window
% Clear the Command Window of any Prior work
clc; % Clear Command Window

%% Material Properties
% Define material properties
k = 143; % Thermal conductivity in W/(m*C)
rho = 2800; % Density in kg/m^3
cp = 795; % Specific heat in J/(kg*C)
alpha = k / (rho * cp); % Thermal diffusivity

%% Grid Spacing Input with Validation
% Define plate dimensions
L = 6; % Length of the plate in meters
W = 6; % Width of the plate in meters
% User input for grid spacing
h = -1; % Initialize with an invalid value
while h <= 0 || h > L || h > W
    h = input('Enter grid spacing (h) in meters: ');
    if h <= 0
        disp('Error: Grid spacing must be a positive number.');
```

elseif h > L || h > W

disp(['Error: Grid spacing must not exceed plate dimensions (', num2str(L), 'm x ', num2str(W), 'm).']);

end

end

```
% Grid Points Calculation
% Calculate the number of grid points
NC = L / h + 1; % Number of columns (grid points)
NR = W / h + 1; % Number of rows (grid points)

%% Time Step Input with Stability Criterion Check
% Calculate the maximum allowable time step based on the stability criterion
max_dt = 1 / (4 * alpha) * (h ^ 2);
disp(['Maximum allowable time step for stability: ', num2str(max_dt), ' seconds.']);
% Prompt the user for time step within the allowable range
dt = -1; % Initialize with an invalid value
while dt <= 0 || dt > max_dt
    dt = input('Enter time step (Δt) in seconds (less than maximum): ');
    if dt <= 0
        disp('Error: Time step must be a positive value.');
```

elseif dt > max_dt

disp(['Error: Time step exceeds the stability limit. Please enter a value less than ', num2str(max_dt), ' seconds.']);

end

end

```
% Temperature Grid Initialization
% Initialize temperature grid
T = zeros(NR, NC);

%% Initial Temperature Distribution
% Set Initial Temperature Distribution
for c = 1:NC
    x = (c - 1) * h; % x-coordinate
    % Using the function f(x) for initial temperature distribution
```

```

    if 0 <= x && x <= (2 / 3) * L % Apply piecewise function
        T(1, c) = 12.5 * x; % First part of the piecewise function
    elseif (2 / 3) * L < x && x <= L
        T(1, c) = -25 * x + 150; % Second part of the piecewise function
    end
end

%% Boundary Conditions
% Set Boundary Conditions
T(:, 1) = 0; % West (Left) Boundary
T(:, end) = 0; % East (Right) Boundary
T(end, :) = 0; % South (Bottom) Boundary
% North (Top) Boundary is set by the initial temperature distribution

%% Numerical Simulation
% Time loop for numerical simulation
t = 0;
while true
    T_old = T;
    for i = 2:NR - 1
        for j = 2:NC - 1
            % Finite difference equation for heat conduction
            T(i, j) = T_old(i, j) + alpha * dt / h ^ 2 * ...
                (T_old(i + 1, j) + T_old(i - 1, j) + ...
                T_old(i, j + 1) + T_old(i, j - 1) - ...
                4 * T_old(i, j));
        end
    end
    % Check for thermal equilibrium (simple convergence check)
    if max(max(abs(T - T_old))) < 0.01
        break;
    end
    t = t + dt; % Update time
end

%% Results Display
% Display results
disp(['Time to reach thermal equilibrium: ', num2str(t), ' seconds']);
disp('Numerical Temperature Distribution at Thermal Equilibrium:');
disp(T);

%% Temperature Distribution Visualization (Figure 1)
% Visualization of the temperature distribution
figure(1)
contourf(flipud(T)) % Flip matrix vertically for correct orientation
colormap(jet) % Apply 'jet' colormap for temperature visualization
colorbar % Add colorbar for scale reference
clim([0 50]); % Set colorbar limits (Replace 'min_value' and 'max_value'
with your desired range)
xlabel('X Coordinate Length (m)');
ylabel('Y Coordinate Length (m)');
title(['Numerical Temperature Distribution at t = ', num2str(t), ' seconds']);

%% Analytical Solution Calculation
% Calculate the analytical solution
num_terms = 200; % Number of terms in the series to include
T_analytic = zeros(NR, NC);
for i = 1:NR
    for j = 1:NC
        x = (i - 1) * h;
        y = (j - 1) * h;
        T_analytic(i, j) = calculate_analytical_temperature(x, y, num_terms);
    end
end
end

```

```

T_analytic = rot90(T_analytic); % Rotate matrix 90 degrees counterclockwise for proper
orientation
% Display the analytical solution in the command window
disp('Analytical Temperature Distribution at Thermal Equilibrium:');
disp(T_analytic);

%% Difference Calculation
% Calculate the difference between analytical and numerical solutions
difference = T - T_analytic; % Compute the difference between solutions
disp('Difference between Numerical and Analytical Solutions:');
disp(difference);

%% Analytical Solution Visualization (Figure 2)
% Visualization of the analytical solution
figure(2)
contourf(flip(T_analytic)) % Flip matrix for correct orientation
colormap(jet) % Apply 'jet' colormap for consistency
colorbar % Include colorbar for reference
clim([0 50]); % Set colorbar limits (Replace 'min_value'
and 'max_value' with your desired range)
xlabel('X Coordinate Length (m)');
ylabel('Y Coordinate Length (m)');
title('Analytical Temperature Distribution');

%% Difference Visualization (Figure 3)
% Visualization of the difference
figure(3)
contourf(flip(difference)) % Flip matrix vertically for correct
orientation
colormap(jet) % Use 'jet' colormap for consistency
colorbar % Add colorbar to denote value ranges
clim([-0.005 2]); % Set colorbar limits (Replace 'min_value'
and 'max_value' with your desired range)
xlabel('X Coordinate Length (m)');
ylabel('Y Coordinate Length (m)');
title('Difference between Numerical and Analytical Solutions');

%% Analytical Temperature Calculation Function
% Function to calculate the analytical temperature distribution
function T_analytical = calculate_analytical_temperature(x, y, num_terms)
    T_analytical = 0;
    for n = 1:num_terms
        term = (sin(2*n*pi/3) / (n^2 * sinh(n*pi))) * sin(n*pi*x/6) * sinh(n*pi*y/6);
        T_analytical = T_analytical + term;
    end
    T_analytical = (450/pi^2) * T_analytical;
end

```