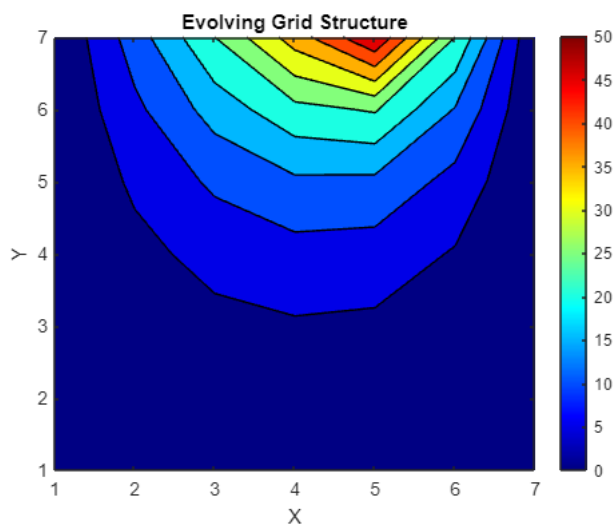


MODULE: COMPUTATIONAL METHODS

Code: ACMA6012

Title: 2D Diffusion Equation Modelled using the Explicit Finite Difference Method



Name: Ryan Harris
Student Number: 2115244

Contents

Introduction	3
Report Structure	3
Aims and Objectives	3
Constitutive Equation of the Problem	4
Introducing the Heat Transfer Equation	4
Interpreting the Evolution of Temperature	4
The Explicit Finite Difference Form	5
Taylor Series (polynomial approximation)	5
First & Second Derivatives of the Heat Transfer Equation	5
The Method for Discretisation	6
Approximating The First Derivative from The Equation	7
Forward Difference Approximation Derivation for The First Derivative (FDA)	7
Backward Difference Approximation for The First Derivative (BDA)	7
Central Difference Approximation for The First Derivative (CDA)	8
Second Derivative of The Heat Transfer Equation	8
Replacing the Temporal First Derivative and the Spatial Second Derivative	9
Discretised form of the Heat Equation	9
Central Difference Approximation for the Second Derivative	9
Forward Difference Approximation for the First Derivative	10
Fourier Number	11
The Initial and Boundary Conditions	12
Initial Domain Conditions	12
Constant Boundary Condition	12
North Boundary Condition	12
Analytical Solution	12
MATLAB Code	13
Initialise, Thermal Properties & User Input	13
Timestep, Stability Criteria & Grid Dimension	14
Initialising Matrix with Boundary Conditions	14
Iterative Process to Achieve Equilibrium	15
Temperature Matrix and Figure Display	16
Analytical Solution for Comparison	17
Validating the Code	18
Resulting temperature matrix from EFDM:	18
Resulting analytical matrix:	18

Sensitivity Analysis	19
Grid Spacing V Time to Reach Thermal Equilibrium.....	19
Time Step V Time to Reach Thermal Equilibrium.....	20
Thermal Equilibrium Tolerance V Time to Reach Thermal Equilibrium	21
Limitations & Future Recommendations	22
Model Limitations	22
Round-off Error	22
Truncation Error	22
Complex Geometry	22
Coding Limitations.....	22
Recommendations for Improvement	23
Implicit Finite Difference Method	23
Numerical Alternatives.....	23
Conclusion	24
List of References	25
Appendix.....	26

Introduction

Understanding and modelling heat diffusion is critical in various engineering and scientific applications (Britannica, 2023). Some of the notable applications include material science and thermal management processes. The way heat propagates within a material influence both its inherent properties and the decisions made by manufacturers in selecting and utilizing the material (US Department of Energy, 2004). The report explores a heat diffusion model, detailing both theoretical foundations and numerical implementation. The goal is to provide a detailed account of the numerical method, it's application of the heat equation, and analysis of the model under several conditions.

Report Structure

The following report is organised into several key sections, each contributing to the understanding of the heat diffusion model. The first stage introduces the governing heat equation that lays the models foundation. The report delves into the governing equations' mathematical definition in the context of 2D geometry. The following section explores the numerical method for discretizing the governing equation, encompassing the process of derivation and consideration of stability. The initial boundary conditions are presented along with the analytical solution for validation, establishing the foundation for subsequent analysis.

Aims and Objectives

1. Use Taylor Series expansion and apply to the Heat Equation.
2. Use Explicit Finite Difference Method to Approximate replacements for the 1st and 2nd Derivatives.
3. Re-formulate the equation to align with the coding process for predicting temperature at each point.
4. Develop a code using MATLAB to display temperature propagation over time in an aluminium plate.
5. Prompt the user for a grid spacing and time step.
6. Ensure these meet the stability criteria.
7. Compute and display temperature at every node.
8. Utilise the analytical equation to verify results.

Constitutive Equation of the Problem

Introducing the Heat Transfer Equation

The equation that governs heat transfer is a crucial tool for comprehending and calculating the temporal rate of temperature change in diverse physical systems. Temporal relates to the way temperature as a property evolves over time, which is an essential part of the heat transfer equation. The heat transfer equation originates from the second law of thermodynamics, the equation has applications across many fields that include but are not limited to engineering, environmental science, and material science.

The heat equation itself is used to calculate heat distribution in a specific region as a function of both space and time. The importance of the heat transfer equation is to provide insight into temperature profile behaviours in specific materials. This provides the analysis of systems in which heat propagation plays a critical role (toppr, 2024).

The following equation is for a two-dimensional geometry without heat source:

$$\frac{1}{\alpha} \times \frac{\partial T}{\partial t} = \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

Equation 1 – Heat Transfer Equation for 2D geometry (Croft D.R, Lilley D.G, 1977)

In equation 1, α represents the thermal diffusivity of the material, T is the temperature distribution, t is time, and x and y represent the spatial co-ordinates.

Interpreting the Evolution of Temperature

The equation encompasses the evolution of temperature over time, this indicates that the rate of temperature change is proportional to the Laplacian of temperature distribution. The Laplacian considers the variation of temperature in both the x and y axis in space. The Laplacian provides information about how scalar quantities such as temperature change across different directions of space, it is a fundamental concept in various fields including physics and mathematics (LibreTextsMathematics, 2024).

To further interpret equation 1, $\frac{\partial T}{\partial t}$ represents the temporal rate of change in the system, $\frac{\partial^2 T}{\partial x^2}$ and $\frac{\partial^2 T}{\partial y^2}$ represent the spatial second derivatives in the system. The spatial second derivatives refer to the second order partial derivatives of a function with respect to time (TECS, 2020). Alpha represents the heat conductivity of the material. The equation itself serves as a fundamental for modelling heat conduction in a varying range of scenarios.

The Explicit Finite Difference Form

Discretization is a numerical method that allows the solving of partial differential equations, this method forms a fundamental step in converting what is known as a continuous problem into a discrete one. This section of the report explains the discretization procedure, incorporating Taylor series expansions and explicit finite difference approximations, with a particular application of solving a heat transfer equation.

The numerical procedure starts with the application of Taylor series expansions to generate difference approximations for the first and second derivatives. Taylor series is a mathematical equation that expresses a function as an infinite sum of terms. By utilizing Taylor series, the difference approximations can be leveraged which serve as the foundation for discretizing the differential equations required.

Taylor Series (polynomial approximation)

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!} (x - x_0)^1 + \frac{f''(x_0)}{2!} (x - x_0)^2 + \dots \sum_{n=0}^{\infty} \frac{f^n(x_0)}{n!} (x - x_0)^n$$

Equation 2 - Taylor Series Expansion

First & Second Derivatives of the Heat Transfer Equation

$$\frac{1}{\alpha} \times \frac{\partial T}{\partial t} = \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

In the equation above, red represents the first derivative of the heat equation. Green represents the second derivative of the equation. This equation would be far too difficult to solve and necessitates the adoption of the Explicit Finite Difference Method. This is to streamline the computational process; it becomes essential to approximate and replace the intricate first and second derivatives present in the equation.

The Method for Discretisation

This report looks at the development of a computational model in MATLAB that calculates the temperature distribution within a 6m x 6m Aluminium plate over time, utilizing the Explicit Finite Difference Method (EFDM) as stated previous. An important part of utilising Taylor Series is understanding the problem. Firstly, the code is required to superimpose a grid structure on the aluminium plate as shown in Figure 1. Within the mesh (domain), temperature is to be calculated at each node until thermal equilibrium is reached.

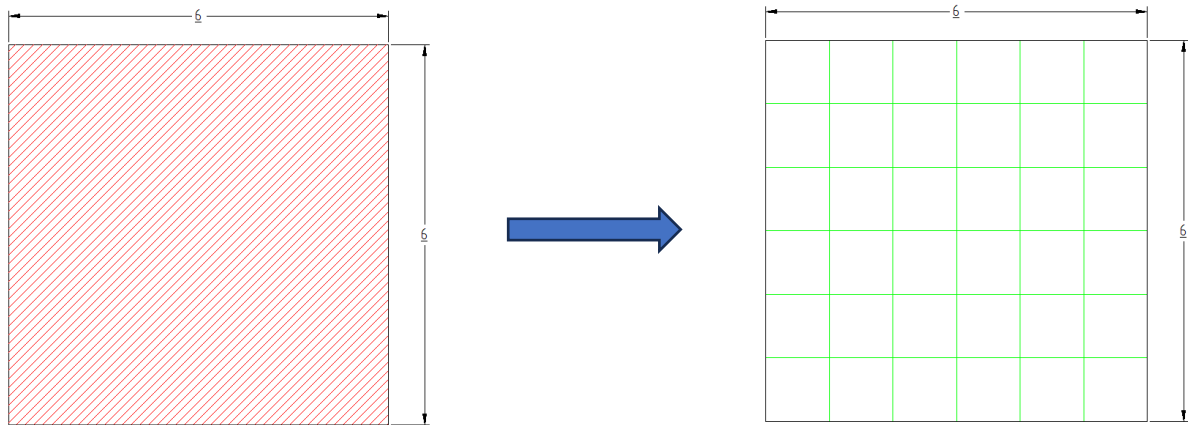
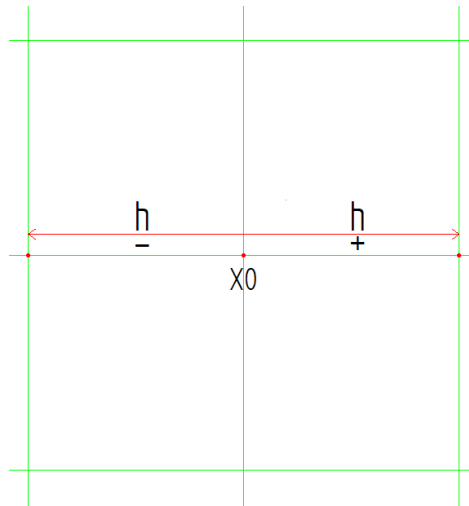


Figure 1 - Super Impose Mesh

heat transfer equation, it is imperative that Taylor series is adapted to align with the specific characteristics of the problem.



When examining a point on the mesh, consider x_0 as the current position. Moving h units from x_0 in a forward direction ($x_0 + h$), while moving h units in the opposite direction ($x_0 - h$) leads to the previous or backward point. The smaller the h value (grid spacing), the finer the mesh and the more accurate the calculation.

Taylor Series re-written for problem (Forward):

$$f(x_0 + h) = f(x_0) + f'(x_0) \frac{h}{1!} + f''(x_0) \frac{h^2}{2!} + \dots \sum_{n=0}^{\infty} f^n(x_0) \frac{h^n}{n!}$$

Approximating The First Derivative from The Equation

Utilising a numerical technique that uses the finite difference method based off Taylor series to form a linear equation.

Forward Difference Approximation Derivation for The First Derivative (FDA)

Make f' the subject:

$$f(x_0 + h) - f(x_0) = f'(x_0)h + f''(x_0)\frac{h^2}{2!} + \dots \sum_{n=0}^{\infty} f^{(n)}(x_0)\frac{h^n}{n!}$$

Divide both sides by h :

$$\frac{f(x_0 + h) - f(x_0)}{h} = f'(x_0) + (f''(x_0)\frac{h}{2!} + f'''(x_0)\frac{h^2}{3!} + \dots)$$

The part of the above equation that is in brackets represents the error value (Oh), if this is negligible then the approximation for the 1st derivative known as the **forward difference approximation (FDA)** is below:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h} + (Oh)$$

FDA is estimated using the current point and the point ahead hence $f(x_0 + h)$, it is a “forward-looking” approach. This method relies on increasing values of x .

Backward Difference Approximation for The First Derivative (BDA)

BDA looks at the current point and the point before, it is “backward looking” and relies on decreasing values of x , it is derived similarly to FDA but is represented $f(x_0 - h)$.

Make f' the subject:

$$f(x_0 - h) = f(x_0) - f'(x_0)h + f''(x_0)\frac{h^2}{2!} - f'''(x_0)\frac{h^3}{3!}$$

Similarly, if the error (Oh) is negligible then the approximation is as follows:

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - h)}{h} + (Oh)$$

Replacing the first derivative of the heat equation by either the forward or backward difference approximation makes a linear equation that is much easier to solve.

The final approximation that can be used to replace the 1st derivative is the Central Difference Approximation (CDA), this approximation is considered more accurate than the forward and backward difference approximation. This is due to increased accuracy that arises from its symmetric nature, the approximation involves information from both sides of the chosen point.

CDA is derived from combining the FDA and BDA, subtracting the forward from the backward difference approximation.

Central Difference Approximation for The First Derivative (CDA)

$$f(x_0 + h) - f(x_0 - h) = 2f'(x_0)h + 2f'''(x_0)\frac{h^3}{3!} + 2f^{(5)}(x_0)\frac{h^5}{5!}$$

Now divide both sides by $2h$:

$$\frac{f(x_0+h)-f(x_0-h)}{2h} = f'(x_0) + (f'''(x_0)\frac{h^2}{3!} + f^{(5)}(x_0)\frac{h^4}{5!} \dots)$$

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h} + (Oh)^2$$

Part of the equation that is brackets represents the error value which is (Oh^2) . This error value is far smaller, which is another reason why CDA is a more accurate approximation for the 1st derivative. There are now three approximations to replace the 1st derivative by, Central, Forward and Backward difference approximation.

Replacing the first derivative by FDA is known as the explicit finite difference method, BDA is known as the implicit difference method and the central difference is known as the finite difference method. The backward difference approximation has multiple unknowns to solve and becomes very difficult in the coding. With Central Difference Approximations, more points are required, and this again becomes a difficult equation to solve. Due to using the Explicit Finite Difference method, the forward difference approximation simplifies the computational process and facilitates the explicit calculation of future values based on current information.

Second Derivative of The Heat Transfer Equation

The Central Difference Approximation is utilized to replace the second derivative and discretises the spatial second derivative term of the heat equation. CDA will consider temperature points on both sides of a given point, leading to a far more accurate representation of how temperature is changing spatially.

Add the FDA to the BDA

Looking to the right of the point (FDA):

$$f(x_0 + h) = f(x_0) + f'(x_0)h + f''(x_0)\frac{h^2}{2!} + \dots \sum_{n=0}^{\infty} f^{(n)}(x_0)\frac{h^n}{n!}$$

Looking to the left of the point (BDA):

$$f(x_0 - h) = f(x_0) - f'(x_0)h + f''(x_0)\frac{h^2}{2!} - \dots \sum_{n=0}^{\infty} f^{(n)}(x_0)\frac{h^n}{n!}$$

$$f(x_0 + h) + f(x_0 - h) = 2f'(x_0) + 2f''(x_0)\frac{h^2}{2!} + 2f^{(4)}(x_0)\frac{h^4}{4!} \dots$$

Take $2f'(x_0)$ to the other side of the equation:

$$f(x_0 + h) - 2f'(x_0) + f(x_0 - h) = f''(x_0)h^2 + 2f^{(4)}(x_0)\frac{h^4}{4!} \dots$$

Divide by h^2 :

$$\frac{f(x_0 + h) - 2f'(x_0) + f(x_0 - h)}{h^2} = f''(x_0) + oh^2$$

Central Difference Approximation for the second derivative:

$$f''(x_0) = \frac{f(x_0 + h) - 2f'(x_0) + f(x_0 - h)}{h^2} + oh^2$$

Replacing the Temporal First Derivative and the Spatial Second Derivative

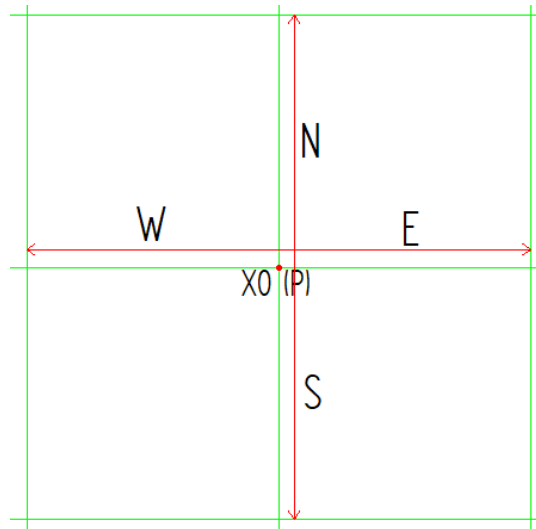


Figure 2 - Point P & Cardinal Notation

looking at point p as the current temperature T_p . In the replacement, considering movement in x and y following the cardinal notation as shown in figure 2.

Discretised form of the Heat Equation

$$\frac{d^2 T_p}{Dx^2} + \frac{d^2 T_p}{Dy^2} = \frac{1}{\alpha} \cdot \frac{dT_p}{dt}$$

Central Difference Approximation for the Second Derivative

$$f''(x_0) = \frac{f(x_0 + h) - 2f'(x_0) + f(x_0 - h)}{h^2}$$

Moving spatially in the X axis:

$$\frac{d^2Tp}{Dx^2} = \frac{TE - 2Tp + TW}{h^2}$$

Moving spatially in the Y axis:

$$\frac{d^2Tp}{Dy^2} = \frac{TN - 2Tp + TS}{h^2}$$

Forward Difference Approximation for the First Derivative

$$f'(x0) \approx \frac{f(x0 + h) - f(x0)}{h}$$

With the first derivative, this is now with respect to time and not space.

$$\frac{dT}{dt} = \frac{Tp^{t+\Delta t} - Tp^t}{\Delta t}$$

This means that the first derivative is equal to the future state of temperature at point P minus the temperature at point P, divided by the time step. Now replace the second and first derivative in the heat equation:

$$\frac{d^2Tp}{Dx^2} + \frac{d^2Tp}{Dy^2} = \frac{1}{\alpha} \cdot \frac{dT}{dt}$$

Substitute values into the equation:

$$\frac{TE - 2Tp + TW}{h^2} + \frac{TN - 2Tp + TS}{h^2} = \frac{1}{\alpha} \cdot \frac{Tp^{t+\Delta t} - Tp^t}{\Delta t}$$

Factorise $\frac{1}{h^2}$:

$$\frac{1}{h^2} (TE^t + TW^t + TN^t + TS^t - 4Tp^t) = \frac{Tp^{t+\Delta t} - Tp^t}{\alpha \Delta t}$$

Fourier Number:

$$FO = \frac{\alpha \Delta t}{h^2}$$

Bring $\alpha \Delta t$ and Tp^t to the other side of the equation:

$$\frac{\alpha \Delta t}{h^2} (TE^t + TW^t + TN^t + TS^t - 4Tp^t) + Tp^t = Tp^{t+\Delta t}$$

Multiply and divide Tp^t by FO to make FO common:

$$FO (TE^t + TW^t + TN^t + TS^t - 4Tp^t + \frac{Tp^t}{FO}) = Tp^{t+\Delta t}$$

Factorise by Tp^t and re-arrange, this is the heat equation using the explicit finite difference method and will be used in the code to predict the heat propagation until thermal equilibrium:

$$Tp^{t+\Delta t} = FO (TE^t + TW^t + TN^t + TS^t - Tp^t(4 - \frac{1}{FO}))$$

Fourier Number

$$0 < FO \leq \frac{1}{4}$$

Fourier number is crucial in the numerical simulation. Fourier (Fo) represents the ratio of diffusion time by the total analysis time. This indicates how quickly heat has propagated.

The code must ensure that the max time step to meet the stability criterion above is, otherwise the model will become unstable due to oscillation and divergence:

$$0 < \Delta t \leq \frac{h^2}{4\alpha}$$

The Initial and Boundary Conditions

There are initial and boundary conditions to consider, this is to define the starting state of the heat transfer model. These conditions are critical to define to accurately model process. The plate is a 6m x 6m Aluminium plate.

Initial Domain Conditions

Condition: $T(x, y, 0) = 0$ for $0 < x < 6$ and $0 < y < 6$

This condition states that the initial temperature distribution within the domain of 6m x 6m plate at time $t=0$ is 0.

Constant Boundary Condition

Condition 1:

$$T(0, y, t) = 0 \text{ and } T(6, y, t) = 0 \text{ for } 0 \leq y \leq 6 \text{ and } 0 \leq t \leq \text{thermal equilibrium}$$

Condition 2:

$$T(x, 0, t) = 0 \text{ and } T(x, 6, t) = 0 \text{ for } 0 \leq x \leq 6 \text{ and } 0 \leq t \leq \text{thermal equilibrium}$$

The above conditions represent the constant temperature boundaries on both sides and bottom of the plate. These boundaries are kept at 0 during the model.

North Boundary Condition

$$T(x, 6, t) = f(x) \text{ for } 0 \leq x \leq 6 \text{ and } 0 \leq t \leq \text{thermal equilibrium}$$

Where $f(x)$ is defined as:

$$f(x) = 12.5(x) \text{ if } 0 \leq x \leq 4 \text{ and } f(x) = -25(x) + 150 \text{ if } 4 \leq x \leq 6$$

The above condition applies only to the northern boundary of the plate. The temperature distribution is defined as $f(x)$ which changes linearly in the specified ranges. Applying $12.5(x)$ to values between 0m and 4m, and $= -25(x) + 150$ to values from 4m to 6m.

To summarise the initial and boundary conditions, they are essential for constructing a meaningful and accurate model for the heat transfer process in the aluminium plate. This ensures that the simulation reflects the physical reality of the system being analysed.

Analytical Solution

Equation 3 - Analytical Solution (Daileda R.C., 2012)

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

Equation 3 is a mathematical solution to the heat equation. The solution represents the temperature distribution at thermal equilibrium for the aluminium plate, which is subjected to the above conditions. The expression involves an infinite series and is based on the mathematical analysis used in solving partial differential equations.

The analytical solution is incorporated to the model code to compare the explicit finite difference method. Comparing the numerical results against the numerical solution helps to identify any discrepancies or limitations to the code/model.

MATLAB Code

This section of the report provides a breakdown of the MATLAB code written to model the heat propagation in the aluminium plate.

Initialise, Thermal Properties & User Input

```
clc;
clear;

% Thermal properties of Aluminium
c = 795;    % Specific heat units J/kg.C
k = 143;    % Thermal Conductivity units W/m.C
rho = 2.8e3; % Density of material with units kg/m^3

% Thermal Diffusivity Equation
alpha = k / (rho*c);

% Displaying alpha
fprintf('Thermal Diffusivity alpha is: %.2e m^2/s^-1\n', alpha);

% Define the square plate dimensions
x = 6; % X direction is 6m
y = 6; % Y direction is 6m

% User input for grid spacing (h)
h = input('Enter value for grid spacing h: ');

% Check if h is acceptable
while ~(h < 6 && mod(x, h) == 0 && mod(y, h) == 0)
    disp('The value of h does not meet the condition. ');
    h = input('Enter another value for h: '); % Ask user for acceptable h value
end
```

Figure 3

Figure 3 shows a snip of the MATLAB code that first clears the command workspace using 'clc' and 'clear' ready to run the code. The thermal properties of the aluminium plate are then defined: specific heat capacity 'c', thermal conductivity 'k', and density 'rho'. These are used to calculate Thermal diffusivity 'alpha' using the formula ' $\alpha = k / (\rho \cdot c)$ '. In this section the code also defines the square plate dimensions as well as requesting the user input for the grid spacing 'h'. The code also ensures that the 'h' value is an acceptable number (integer). If not, there is a while loop that continues until this condition is met.

Timestep, Stability Criteria & Grid Dimension

```
disp(['Value of h = ', num2str(h), ' (selected for the grid spacing)']);

% Ask user to input value for the timestep (delta t)
dt = input('Enter a value for the timestep in seconds: ');

% Calculate max allowable timestep
max_dt = h^2 / (4 * alpha);

% Check if input dt meets criteria
while ~(dt > 0 && dt <= max_dt)
    disp('The input timestep is not acceptable. ');
    disp(['Max timestep for stability is: ', num2str(max_dt), ' seconds']);
    dt = input('Enter another value for the timestep in seconds: ');
end

% Calculate the Fourier number
Fourier = alpha * dt / (h^2);

% Display the valid Fourier number
fprintf('Valid Fourier number is: %.2e\n', Fourier);

% Calculate number of rows & columns using user input h
nrows = y / h + 1;
ncols = x / h + 1;

fprintf('Number of rows: %d\n', nrows);
fprintf('Number of columns: %d\n', ncols);
```

Figure 4

In the next section of the code, the user is asked to input a value for the timestep. The code then calculates the maximum allowable timestep '**dt**' and checks to ensure that the user request is acceptable. If the timestep does not meet the stability criteria, then the user is prompted to choose another result displaying the maximum timestep '**max_dt**' to ensure the user does not waste time. The Fourier number is also calculated at this point once an acceptable time step is chosen. In this snip, the grid dimension is also calculated based on the h value, calculating the number of column and rows of the matrix.

Initialising Matrix with Boundary Conditions

```
% Initialize the temperature matrix t with boundary conditions
t = zeros(nrows, ncols);

for col = 1:ncols
    d = (col - 1) * h;

    if d <= 4
        t(1, col) = 12.5 * d; % Apply boundary condition to row 1 of the matrix
    else
        t(1, col) = -25 * d + 150; % Apply boundary condition to row 1 of the matrix
    end
end
```

Figure 5

Having initialised the matrix of zeros with respect to the number of columns and number rows calculated above, '**nrows = y / h + 1**' and '**ncols = x / h + 1**'. The next step is to apply the boundary conditions to the northern boundary of the steel plate. This is to begin the model at the temperature state required. Any number that is equal to or less than 4 has the first condition applied, and above 4 the second boundary condition is applied.

Iterative Process to Achieve Equilibrium

```
% Iterative process using a tolerance to check for equilibrium
equilibrium = false; % Boolean flag initially false
iterations = 0; % Counting the number of iterations
tolerance = 0.01; % Tolerance used to determine equilibrium
t_new = t;

while ~equilibrium % Continue iterating until equilibrium is true
    iterations = iterations + 1; % Iteration counter
    max_diff = 0; % Initialize maximum difference

    for i = 2:nrows-1
        for j = 2:ncols-1
            % Neighboring points
            TE = t(i, j + 1);
            TW = t(i, j - 1);
            TN = t(i - 1, j);
            TS = t(i + 1, j);

            Tp = t(i, j); % Temperature at the current point

            % Prediction using the equation
            t_new(i, j) = Fourier * (TE + TW + TN + TS - Tp * (4 - 1/Fourier));

            % Calculate the difference and update max_diff
            diff = abs(t_new(i, j) - t(i, j));
            max_diff = max(max_diff, diff);
        end
    end
end
```

Figure 6

A while loop begins that iterates until the temperature matrix has reached equilibrium, indicating that the heat propagation has concluded. The model computes the updated temperature using the explicit finite difference method. The code checks for equilibrium by using a tolerance threshold. Once the difference between the final matrix and the matrix before is 0.01 the coding finishes.

Temperature Matrix and Figure Display

```
% Check for equilibrium
if max_diff <= tolerance
    equilibrium = true;
    fprintf('Equilibrium reached after %d iterations.\n', iterations);
    disp('Final Temperature Matrix: ');
    disp(t);
else
    t = t_new; % Update the temperature matrix for the next iteration
end
end

% Calculate the time taken
time_taken = (iterations * dt) / 3600; % Time elapsed in hours

% Display time taken to reach thermal equilibrium
fprintf('Time elapsed reaching thermal equilibrium: %.2f hours\n', time_taken);

% Displaying data graphically
figure;
contourf(flip(t_new));
colormap(jet);
clim([0, 50]);
title('Evolving Grid Structure');
xlabel('X');
ylabel('Y');
zlabel('Temperature');
colorbar;
```

Figure 7

Once equilibrium is achieved, the code displays the temperature matrix result. It also calculates the amount of time elapsed reaching equilibrium, illustrating the heat propagation graphically.

Analytical Solution for Comparison

```
% Analytical Solution Comparison
t_analytical = zeros(nrows, ncols);

for col = 1:ncols
    d = (col - 1) * h;
    if d <= 4
        t_analytical(1, col) = 12.5 * d; % Apply the same boundary condition to row 1 of the matrix t_analytical
    else
        t_analytical(1, col) = -25 * d + 150; % Apply the same boundary condition to row 1 of the matrix t_analytical
    end
end

for r = 2:nrows-1
    for c = 2:ncols-1
        x_pos = (c - 1) * h;
        y_pos = (r - 1) * h;

        for n = 1:200
            t_analytical(r, c) = t_analytical(r, c) + (sin((2 * n * pi) / 3) / (n^2 * sinh(n * pi))) * (sin((n * pi * x_pos) / 6)) * (sinh((n * pi * y_pos) / 6));
        end
        t_analytical(r, c) = t_analytical(r, c) * (450 / pi^2);
    end
end

% Adjust t_analytical matrix
t_analytical = flipud(t_analytical);
temp_row = t_analytical(1, :);
t_analytical(1, :) = t_analytical(end, :);
t_analytical(end, :) = temp_row;

% Display t_analytical matrix
disp('Analytical Solution Matrix: ');
disp(t_analytical);

% Display t_analytical matrix graphically
figure;
contourf(flip(t_analytical));
colormap(jet);
clim([0, 50]);
title('Analytical Solution');
xlabel('X');
ylabel('Y');
zlabel('Temperature');
colorbar;

%Code ends
```

Figure 8

This part of the code computes the analytical solution for the temperature distribution. It initialises a new matrix of zeros known as `t_analytical` applying the same number of rows and columns as well as applying the same northern boundary condition. During the temperature computation, the nested loop iterates over the internal domain nodes. Utilising a Fourier series expansion, each of the x and y positions are evaluated 200 times and the temperature at each point is calculated. The final part of the script displays the t-analytical matrix and uses the 'contourf' plot for visualisation. This is to enable a comparison to be drawn to the numerical solution.

Validating the Code

The analytical solution has been used alongside the numerical simulation to draw analysis of the thermal behaviour of the aluminium plate. The analytical solution serves as a reference, which allows for the validation of the results gathered from the explicit finite difference method. The simulation was run using a grid spacing (h) of 1, a timestep (dt) of 200 seconds and a tolerance criterion of 0.01.

Resulting temperature matrix from EFDM:

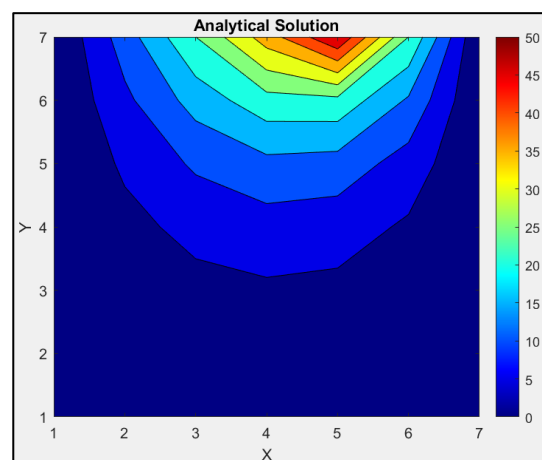
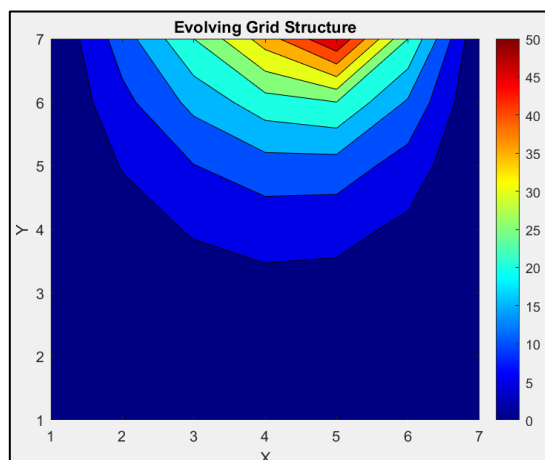
Final Temperature Matrix:						
0	12.5000	25.0000	37.5000	50.0000	25.0000	0
0	8.4885	16.4439	22.8261	24.9019	14.3376	0
0	5.2286	9.8434	12.9062	12.8369	7.6776	0
0	2.9390	5.4183	6.8485	6.5024	3.9095	0
0	1.4867	2.7028	3.3404	3.0934	1.8537	0
0	0.6036	1.0883	1.3284	1.2144	0.7249	0
0	0	0	0	0	0	0

Resulting analytical matrix:

Analytical Solution Matrix:						
0	12.5000	25.0000	37.5000	50.0000	25.0000	0
0	8.8572	17.0326	23.1754	23.5790	14.3182	0
0	5.8115	10.7628	13.6629	12.9330	7.8515	0
0	3.5691	6.4420	7.8677	7.1874	4.2986	0
0	2.0086	3.5713	4.2716	3.8276	2.2630	0
0	0.8967	1.5822	1.8731	1.6622	0.9764	0
0	0	0	0	0	0	0

The final temperature matrices from both the EFDM and analytical approaches, highlight several similarities. Both exhibit the same temperature profiles diminishing as the heat propagates through the plate. The consistency in comparing both numerical Solution and analytical solution indicates that the EFDM approach effectively captures the thermal dynamics of the system. The EFDM approach produces results that are closely aligned with the analytical method.

Utilising the analytical solution in combination with the EFDM approach provides a valuable insight into the thermal behaviour. While the EFDM method does provide practicality in solving the thermal problem, The analytical solution helps lay the benchmark to allow the validation of the numerical results. This further stimulates greater confidence in the credibility of the simulation. This gives confidence in the precision of the model. With reference to the resulting temperature maps, the closeness in results is depicted where the graphs are almost symmetrical in propagation.

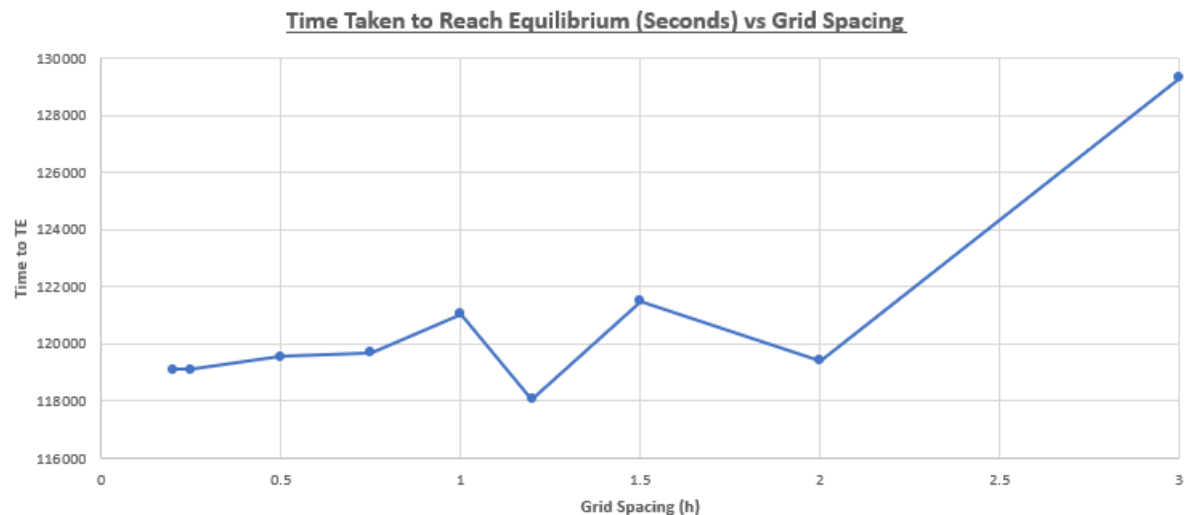


Sensitivity Analysis

A sensitivity analysis was carried out on the code, this was to optimize the computational parameters, understand the model behaviour and identify limitations. Parameters such as the grid spacing, time step and tolerance threshold were varied to look at the resulting output (Time to reach Thermal Equilibrium). The following part of the report displays the results from such analysis.

Grid Spacing V Time to Reach Thermal Equilibrium

Grid Spacing (h)	Fixed Timestep (Seconds)	Time Taken to Reach Equilibrium (Seconds)	Tolerance
3	150	129300	0.001
2	150	119400	0.001
1.5	150	121500	0.001
1.2	150	118050	0.001
1	150	121050	0.001
0.75	150	119700	0.001
0.5	150	119550	0.001
0.25	150	119100	0.001
0.2	150	119100	0.001

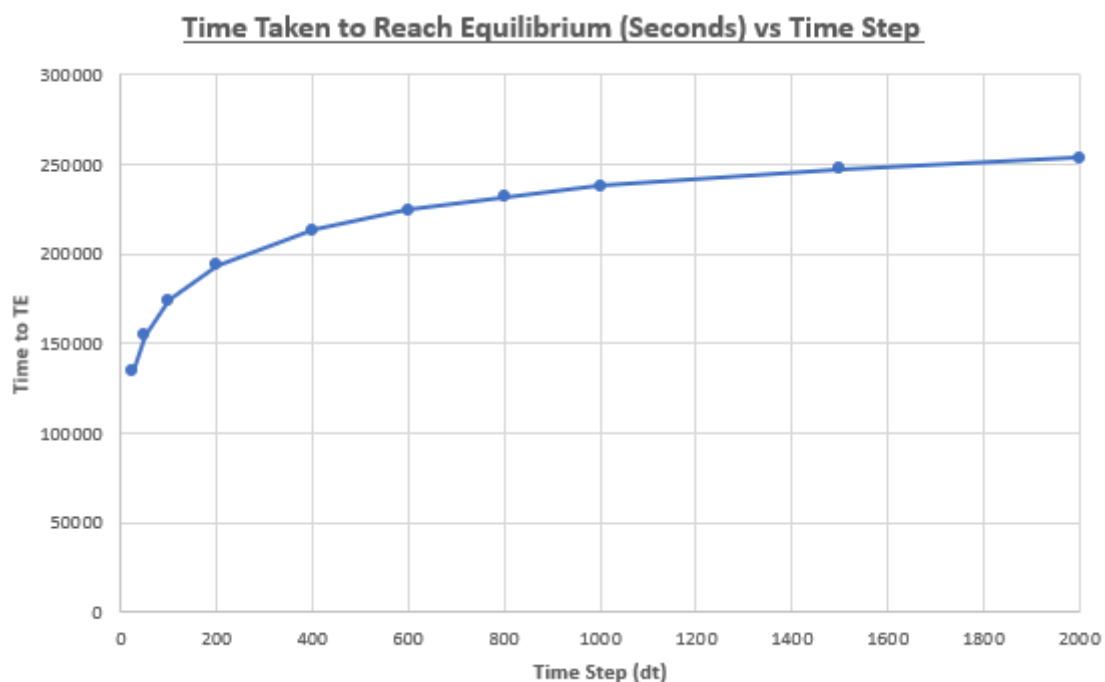


As the grid spacing reduces, the time taken to reach thermal equilibrium also reduces. The Grid Spacing (h) represents the size of the mesh in the simulation. A smaller spacing lead to finer resolution, allowing for more precise modelling. The timestep is fixed at 150 seconds which is the rate at which temperature change is calculated. Observing the results of the analysis, at $h = 0.5$, the further reduction of h does not significantly decrease the time taken to thermal equilibrium. This suggests that a grid spacing from 0.5 to 1 provides a good balance between computational efficiency and accuracy in capturing temperature distribution.

The grid spacing is an important feature of the model, finer resolution leads to faster convergence. This can be observed in the graph. Once the return in reducing the grid spacing diminishes, it is important to strike a balance between computational resource and simulation accuracy.

Time Step V Time to Reach Thermal Equilibrium

Time step	Fixed Grid Spacing (h)	Time Taken to Reach Equilibrium (Seconds)	Tolerance
2000	0.75	254000	0.0001
1500	0.75	247500	0.0001
1000	0.75	238000	0.0001
800	0.75	232000	0.0001
600	0.75	224400	0.0001
400	0.75	213200	0.0001
200	0.75	194000	0.0001
100	0.75	174200	0.0001
50	0.75	154350	0.0001
25	0.75	134475	0.0001

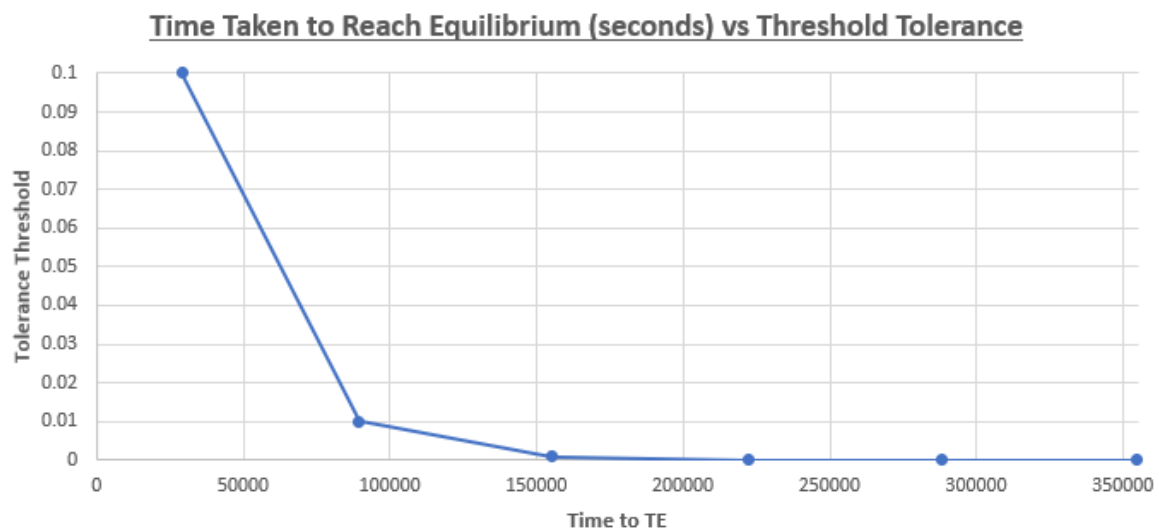


The timestep is a crucial parameter in the simulation, determining the temporal resolution and overall granularity of time where the simulation computes the temperature change. Above shows the sensitivity analysis that was conducted changing the timestep with a fixed tolerance and grid spacing. As the timestep decreases from 2000 to 25 seconds, the time to reach equilibrium decreases. The following observation aligns with expectation as a smaller timestep allows the simulation to capture more detailed temperature change in shorter intervals, this leads to a faster convergence.

Around a timestep of 200s, further reduction does not substantially decrease the time to equilibrium. In terms of computational efficiency reducing the timestep beyond this threshold sacrifices optimal modeling of the heat propagation in the aluminium plate. A range of 200 to 1000 seconds provides a good balance.

Thermal Equilibrium Tolerance V Time to Reach Thermal Equilibrium

Fixed Time step	Fixed Grid Spacing (h)	Time Taken to Reach Equilibrium (Seconds)	Tolerance
500	1	29000	0.1
500	1	89500	0.01
500	1	155500	0.001
500	1	222000	0.0001
500	1	288000	0.00001
500	1	354500	0.000001



The tolerance threshold was adjusted whilst keeping the grid spacing and timestep fixed. Varying the tolerance impacts the time taken to equilibrium. As the tolerance decreases in size, the time to equilibrium increases. This is evident from the results shown above. Looking at specific results, tolerance decreases from 0.1 to 0.000001 and the time taken to equilibrium increases from 29000 to 354500 seconds.

The tolerance control serves as criterion for acceptable convergence. A higher tolerance allows for a larger difference between iterations, this means that less iterations are required to take place before the model is considered converged. A lower tolerance threshold required much finer accuracy hence more iterations and computational effort.

Decreasing the tolerance improves accuracy of the model. However, balance is critical between the computational resource required versus time. A tolerance of 0.001 and 0.0001 seems acceptable. After 0.0001, there is not a great return in terms of time taken to equilibrium. The timestep and tolerance have an inverse relationship, A timestep that is too large will cause divergence in the simulation and cause the code to miss important transient information.

Limitations & Future Recommendations

Model Limitations

The model may encounter limitations which come from the numerical method used to discretise as well as the coding implementation.

Round-off Error

With any numerical computation infinite precision is not possible. Round off errors accumulate during the simulation; this has the potential to lead to inaccurate results. Computing the thermal distribution is possible using the code written however there are issues with the rounding of the temperatures. This is due to computational limitation. (Kenton, W. 2021)

Truncation Error

Truncation Error arises in numerical methods such as the explicit finite difference method with the use of approximations. In the context of the simulation, truncation error emerges when the continuous system used (Taylor Series) is discretised to finite elements or grid structure.

By decreasing the grid spacing (h) these errors are minimised, a more accurate model is also achieved but at the cost of efficiency. Decreasing h decreases interval size capturing the function behaviour more accurately, in turn reducing error caused by truncation. (University of Utah, 2024).

Complex Geometry

The numerical method chosen may struggle to precisely capture the thermal affects due to the complex geometries or uneven boundary conditions. It's impossible to simulate the truest form of the thermal behaviour due to the impossibility of infinite calculation.

Coding Limitations

Modeling thermal equilibrium comes with the limitation of relying solely on the difference between temperature matrices. This is over-simplifying the physics and not giving a true indication of the behaviour. Thermal equilibrium in a real-world application involves complex interactions which may not be captured truly using the code.

The numerical technique relies on assumptions, simplification, and approximations. This can deviate from the physical behaviour observed. Some examples include the boundary conditions, the starting temperatures and even the material properties.

Recommendations for Improvement

Implicit Finite Difference Method

The implicit finite difference method could be utilised to improve the current approach using the explicit finite difference method. In the implicit method, temperature at the next time step is calculated using a system of equations that include values from both the current and next time step. This would make the code more stable and handle the system effectively.

Numerical Alternatives

Finite element method (FEM) would be a suitable numerical alternative to the explicit finite difference method, FEM is a method used for solving differential equations in mathematical modeling. FEM can solve differential equations in two or three space variables. According to a mesh construction is similarly applied to the problem formulating a system of algebraic equations, again approximating unknown functions over a domain. (Cebula, A. 2014).

Finite Volume Method (FVM) could also be used in the problem, FVM discretises the domain into control volumes and computes fluxes between said volumes. According to Machine Design (2016) FVM is a more physically orientated approach in comparison to the finite difference method. Like FEM, FVM is first divided into small finite-sized elements of geometrical shapes. The elements in FVM are identified as cells, using structured or unstructured meshes FVM still leads to robust schemes. FVM may be a more suitable alternative for handling the complex geometry of the problem.

Conclusion

To conclude, understanding heat diffusion models and the numerical implementation has offered an important insight into thermal behaviour in an engineering environment. The report has taken a comprehensive approach in understanding foundations of theory, as well as numerical methods used in allowing the modeling of heat propagation.

Exploring the detail of the heat transfer equation has highlighted the pivotal role in understanding temperature variation over time. A major part of the equation was the temporal evolution of temperature, connected to the spatial position described by the Laplacian operator. By further delving into the heat equation, the report underpinned the significance of accurately modeling heat conduction in a simulated environment.

The report used the Explicit Finite Difference Method to make approximations to replace the first and second derivatives of the heat equation. Transforming a continuous problem into a discrete one. EFDM allowed for the prediction of temperature distribution of an aluminium plate with an unconventional temperature profile applied to the boundary.

Using this methodology did highlight some limitations, both numerically and computationally. Round off and truncation errors are inherited by the very nature of computational modeling, which can lead to inaccuracy if not understood and reduced. Relying on temperature matrix differences also highlights the need for more accurate representations of the true temperature propagation to equilibrium.

These limitations can be reduced further by the exploration of alternative methods and coding strategies. Implicit finite difference method would further increase both accuracy and overall stability. In addition, considering both finite element and finite volume method could provide a more robust approach.

List of References

Britannica (2023) Heat Transfer. Available at: [Heat transfer | Definition & Facts | Britannica](#) (Accessed: 1st January 2024).

Cebula A. (2014) Finite Volume Method in Heat Conduction. Available at: [Finite Volume Method in Heat Conduction | SpringerLink](#) (Accessed: 21st January 2024).

Croft D.R, Lilley D.G. (1977) Heat Transfer Calculations using the Finite Difference Equations. London: Applied Science Publishers Ltd.

Daileda R.C. (2012) The Two-Dimensional Heat Equation. Trinity University, San Antonio, Texas, USA.

Kenton W. (2021) Rounding Error. Available at: [Rounding Error: What it is, How it Works, Examples \(investopedia.com\)](#) (Accessed: 15th January 2024).

LibreTextsMathematics (2024) Steady State Temperature and the Laplacian. Available at: [4.9: Steady State Temperature and the Laplacian - Mathematics LibreTexts](#) (Accessed: 4th January 2024).

Machine Design (2016) What's the Difference Between FEM, FDM and FVM? Available at: [What's The Difference Between FEM, FDM and FVM? | Machine Design](#) (Accessed: 23rd January 2024).

TECS (2020) Derivation of Heat Equation. Available at: [Derivation of heat equation \(diffusion equation\) - tec-science](#) (Accessed: 5th January 2024).

Toppr (2024) Heat Transfer Formula. Available at: [Heat Transfer Formula: Definition, Concepts and Examples \(toppr.com\)](#) (Accessed: 3rd January 2024).

University of Utah. (2024) Truncation Errors & Taylor Series. Available at: [Microsoft PowerPoint - Lect3 RoundOffErrors.ppt \(utah.edu\)](#) (Accessed: 18th January 2024).

US Department of Energy (2004) Material Selection Considerations for Thermal Process Equipment. Available at: [Materials Selection Considerations for Thermal Process Equipment: A BestPractices Process Heating Technical Brief. Industrial Technologies Program \(ITP\) \(Brochure\). \(energy.gov\)](#) (Accessed: 1st January 2024).

Appendix

```
close all
clc;
clear;

% Thermal properties of Aluminium
c = 795;    % Specific heat units J/kg.C
k = 143;    % Thermal Conductivity units W/m.C
rho = 2.8e3; % Density of material with units kg/m^3

% Thermal Diffusivity Equation
alpha = k / (rho*c);

% Displaying alpha
fprintf('Thermal Diffusivity alpha is: %.2e m^2/s^-1\n', alpha);

% Define the square plate dimensions
x = 6; % X direction is 6m
y = 6; % Y direction is 6m

% User input for grid spacing (h)
h = input('Enter value for grid spacing h: ');

% Check if h is acceptable
while ~(h < 6 && mod(x, h) == 0 && mod(y, h) == 0)
    disp('The value of h does not meet the condition. ');
    h = input('Enter another value for h: '); % Ask user for acceptable h value
end

disp(['Value of h = ', num2str(h), ' (selected for the grid spacing)']);

% Ask user to input value for the timestep (delta t)
dt = input('Enter a value for the timestep in seconds: ');

% Calculate max allowable timestep
max_dt = h^2 / (4 * alpha);

% Check if input dt meets criteria
while ~(dt > 0 && dt <= max_dt)
    disp('The input timestep is not acceptable. ');
    disp(['Max timestep for stability is: ', num2str(max_dt), ' seconds']);
    dt = input('Enter another value for the timestep in seconds: ');
end

% Calculate the Fourier number
Fourier = alpha * dt / (h^2);

% Display the valid Fourier number
fprintf('Valid Fourier number is: %.2e\n', Fourier);

% Calculate number of rows & columns using user input h
nrows = y / h + 1;
ncols = x / h + 1;

fprintf('Number of rows: %d\n', nrows);
fprintf('Number of columns: %d\n', ncols);

% Initialize the temperature matrix t with boundary conditions
```

```

t = zeros(nrows, ncols);

for col = 1:ncols
    d = (col - 1) * h;

    if d <= 4
        t(1, col) = 12.5 * d; % Apply boundary condition to row 1 of the matrix
    else
        t(1, col) = -25 * d + 150; % Apply boundary condition to row 1 of the
matrix
    end
end

% Iterative process using a tolerance to check for equilibrium
equilibrium = false; % Boolean flag initially false
iterations = 0; % Counting the number of iterations
tolerance = 0.01; % Tolerance used to determine equilibrium
t_new = t;

while ~equilibrium % Continue iterating until equilibrium is true
    iterations = iterations + 1; % Iteration counter
    max_diff = 0; % Initialize maximum difference

    for i = 2:nrows-1
        for j = 2:ncols-1
            % Neighboring points
            TE = t(i, j + 1);
            TW = t(i, j - 1);
            TN = t(i - 1, j);
            TS = t(i + 1, j);

            Tp = t(i, j); % Temperature at the current point

            % Prediction using the equation
            t_new(i, j) = Fourier * (TE + TW + TN + TS - Tp * (4 - 1/Fourier));

            % Calculate the difference and update max_diff
            diff = abs(t_new(i, j) - t(i, j));
            max_diff = max(max_diff, diff);
        end
    end

    % Check for equilibrium
    if max_diff <= tolerance
        equilibrium = true;
        fprintf('Equilibrium reached after %d iterations.\n', iterations);
        disp('Final Temperature Matrix: ');
        disp(t);
    else
        t = t_new; % Update the temperature matrix for the next iteration
    end
end

% Calculate the time taken
time_taken = (iterations * dt); % Time elapsed in seconds

% Display time taken to reach thermal equilibrium
fprintf('Time elapsed reaching thermal equilibrium: %.2f seconds\n', time_taken);

```

```

% Displaying data graphically
figure;
contourf(flip(t_new));
colormap(jet);
clim([0, 50]);
title('Evolving Grid Structure');
xlabel('X');
ylabel('Y');
zlabel('Temperature');
colorbar;

% Analytical Solution Comparison
t_analytical = zeros(nrows, ncols);

for col = 1:ncols
    d = (col - 1) * h;

    if d <= 4
        t_analytical(1, col) = 12.5 * d; % Apply the same boundary condition to
row 1 of the matrix t_analytical
    else
        t_analytical(1, col) = -25 * d + 150; % Apply the same boundary condition
to row 1 of the matrix t_analytical
    end
end

for r = 2:nrows-1
    for c = 2:ncols-1
        x_pos = (c - 1) * h;
        y_pos = (r - 1) * h;

        for n = 1:200
            t_analytical(r, c) = t_analytical(r, c) + (sin((2 * n * pi) / 3) /
(n^2 * sinh(n * pi))) * (sin((n * pi * x_pos) / 6)) * (sinh((n * pi * y_pos) /
6));
        end
        t_analytical(r, c) = t_analytical(r, c) * (450 / pi^2);
    end
end

%Adjust t_analytical matrix
t_analytical = flipud(t_analytical);
temp_row = t_analytical(1, :);
t_analytical(1, :) = t_analytical(end, :);
t_analytical(end, :) = temp_row;

%Display t_analytical matrix
disp('Analytical Solution Matrix: ');
disp(t_analytical);

%Display t_analytical matrix graphically
figure;
contourf(flip(t_analytical));
colormap(jet);
clim([0, 50]);
title('Analytical Solution');
xlabel('X');
ylabel('Y');
zlabel('Temperature');

```

```
colorbar;

% Calculate the difference matrix
difference_matrix = t_analytical - t_new;

% Display the difference matrix
disp('Difference Matrix:');
disp(difference_matrix);

%Code ends
```