

University of Wales Trinity Saint David

School of Engineering

2D Diffusion Equation Model using the Explicit Finite Difference Method

Computational Methods (ACMA 6012)

Nathan Skyrme

2111290

Energy and Environmental Engineering (BEng)

13/02/2024

Table of Contents

1 Introduction	3
2 Heat Equation	4
3 Numerical Method	5
3.1 Taylor Series	5
3.2 Forward, Backward, and Central Difference Approximation	5
3.3 Spatial Domain	8
3.4 Temporal Domain	8
3.5 Time Step	8
3.6 Stability	8
3.7 Explicit Finite Difference	9
4 Initial, and Constant Boundary Conditions	11
5 Program Code	13
6 Code Validation	18
7 Sensitivity Analysis	21
7.1 Grid Spacing	21
7.2 Time Step	22
7.3 Tolerance Threshold	23
8 Model Limitations	25
9 Further Work	27
10 Conclusion	28
References	29
Appendix	33
Figure 1 - Numerical Solution - Temperature Map	18
Figure 2 - Analytical Solution - Temperature Map	18
Figure 3 - Difference between numerical and analytical solutions - Temperature Map	19
Figure 4 - Relationship between thermal equilibrium and grid space	22
Figure 5 - Relationship between thermal equilibrium and time step	23
Figure 6 - Relationship between thermal equilibrium and tolerance threshold	24
Table 1 - Difference between numerical and analytical solutions	19
Table 2 - Grid Spacing Variation	21
Table 3 - Time Step Variation	23
Table 4 - Tolerance Variation	24

1 Introduction

The following report details the creation and analysis of a MATLAB model to calculate and represent the diffusion of heat across a 2D aluminium plate using the explicit finite difference method. The report aims to give a brief background into the methods deployed and describe in detail the numerical methods. The programming process will be described along with appropriate justification for its contents, including boundary conditions and tolerances. Additionally, the results retrieved from the model will be analysed, culminating in an assessment of the limitations of the model, identifying differences in results when in comparison to the analytical solution. Finally suggested further work and model development will be discussed and conclusions will be drawn.

The following were the aims and objectives of the model created, the model was to:

- Create a domain representative of a 2D aluminium model based on defined characteristics and user input
- Receive input from a user which will determine the models time step and grid size parameters and check whether they conform to the stability criteria
- Discretise the spatial and temporal domain, based on the user input and defined characteristics
- Calculate and display the current and evolution of temperature at each time step
- Cease calculations once the thermal equilibrium threshold has been reached
- Calculate the analytical solution and compare the numerical solution against it

The code for the model referenced throughout this report can be found in the appendix.

2 Heat Equation

The heat equation was first formulated in 1822 by French physicist Jean Fourier (Narasimhan, 1999). According to (Gorguis et al., 2008) the heat equation can be defined as, ‘a partial differential equation that describes the variation of temperature in a given region over a period of time.’ The heat equation for the model is so fundamental as it is the entire guiding principle behind what the model is aiming to achieve. The heat equation is known to be a reliable way to calculate accurate results (Fatima, 2021).

The heat equation can come in various forms, the equation for conduction across a solid is defined as (Nimona et al., 2022),

$$\therefore \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{\alpha} \cdot \frac{\partial T}{\partial t}$$

Eq. 2-1

Where,

- T is the temperature [$^{\circ}\text{C}$]
- t is time [S]
- $\alpha = \frac{k}{(\rho \cdot c)}$ is the thermal diffusivity [$m^2 \cdot s^{-1}$]
- c is the specific heat [$J \cdot kg^{-1} \cdot K^{-1}$]
- k is the thermal diffusivity [$W \cdot m^{-1} \cdot K^{-1}$]
- ρ is the density [$kg \cdot m^{-3}$]

This form of the heat equation is used for steady-state calculations, where time is not a variable.

The heat equation given as above, given by (Croft D.R, Lilley D.G, 1977) was the subject of the model.

This particular form is a two-dimensional parabolic heat equation. Using a parabolic heat equation allows the calculation to account for the diffusion of heat over time (Britannica, 2024).

3 Numerical Method

To be able to represent temperatures across a 2D plane using a numerical technique it is essential to discretise the spatial and temporal domains, in order to approximate the Heat equation.

Discretisation in essence is the representation of continuous data sets into discrete spatial and temporal intervals (Lustgarten, 2008). The explicit finite difference method used in the model, is derived from the Taylor series equation.

3.1 Taylor Series

The Taylor series is a polynomial mathematical instrument that is used to approximate function values for mathematical analysis. Taylors theorem states that a differentiable function can be approximated by some form of polynomial for a given range, in certain instances for an infinite number of sums (Bi, 2018).

The written form of the Taylor series equation is,

$$f(x) = f(x_0) + f'(x_0) \cdot \frac{(x - x_0)^1}{1!} + f''(x_0) \cdot \frac{(x - x_0)^2}{2!} + \dots$$

Eq. 3-1

$f(x)$ about $x = x_0$

While the Taylor series is theoretically infinite, for the model only values up to the second derivative were needed.

The Taylor series equation can be written more compactly as,

$$f(x) = \sum_{n=0}^{\infty} f^{(n)}(x_0) \cdot \frac{(x - x_0)^n}{n!}$$

Eq. 3-2

From this the forward, backward, and central differences can be derived.

3.2 Forward, Backward, and Central Difference Approximation

Forward Difference Approximation is employed in order to relate the value of a function at point x_0 to the value of a function at a nearby point, this is represented by $x = (x_0 + h)$, the forward approximation, approximates the function value as it increases.

The forward difference approximation equation is derived from the Taylor series function $f(x)$ as follows,

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

Eq. 3-3

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

Eq. 3-4

This gives the final forward difference approximation equation as,

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

Eq. 3-5

Backward Difference Approximation relates the function value at x_0 to a function value at a nearby point as the function value decreases, represented by, $x = (x_0 - h)$. It is essentially the inverse of forward difference approximation.

The backward difference approximation equation is derived from the Taylor series function $f(x)$ as follows,

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

Eq. 3-6

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

Eq. 3-7

This gives the final backward difference approximation equation as,

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

Eq. 3-8

Central Difference Approximation is a combination of the forward and backward approximations, that can yield a higher accuracy than the individual approximation methods.

The central difference approximation equation is derived from combining the forward and backward difference approximations.

$$f(x_0 + h) - f(x_0 - h) = 2 \cdot \left[h \cdot f'(x_0) + \frac{h^3}{3!} f^{(3)}(x_0) + \dots \right]$$

Eq. 3-9

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

Eq. 3-10

This gives the final central difference approximation equation as,

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

Eq. 3-11

Using the central difference approximation a more accurate discretisation is able to be derived. The difference approximations are used to replace the derivative functions within the heat equation. To do this a difference approximation for the second order derivative function in the equation is required. Therefore, it was decided to construct the second order derivative of the central difference approximation.

By taking the forward and backward differences to the second order derivative and combining them, we can construct the second order derivative of the central difference approximation.

$$f(x_0 + h) + f(x_0 - h) = 2 \cdot \left[f(x_0) + \frac{h^2}{2!} f''(x_0) + \frac{h^4}{4!} f^{(4)}(x_0) + \dots \right]$$

Eq. 3-12

$$\frac{f(x_0 + h) - 2 \cdot f(x_0) + f(x_0 - h))}{h^2} = f''(x_0) + \frac{h^2}{12} f^{(4)}(x_0) + \dots$$

Eq. 3-13

This gives the final equation for the second order derivative of the central difference approximation as,

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

Eq. 3-14

The difference approximations discussed above include an error term known as truncation error. This error occurs in difference approximations when higher-order terms are truncated during approximation (Kalinowska et al., 2007). This truncation error therefore creates some amount of difference (error) between the true and approximated value of the function.

3.3 Spatial Domain

Accurately capturing the solution will require the creation of a grid to discretise the information across the domain. The domain created within the model is 6x6m. The grid sizing will therefore be uniform across the domain, dividing the domain into smaller square grids with nodes at each location representing a point in the matrix. In its mathematical context the grid spacing is defined as $h = \Delta x = \Delta y$. The minimum acceptable grid size will be 3m and any grid size will need to be divisible by 6 with no remainder in order to create an evenly distributed domain discretisation.

3.4 Temporal Domain

As the model is aiming to calculate a time dependant problem, the temporal domain must be discretised, in order to facilitate the capture of the diffusion of temperature over time (Huang et al., 1999). Therefore, to create a discretised temporal domain a temporal derivative term, t , is required within the numerical solution. This then converts the solution equation from steady-state to time-dependant (Huang et al., 1994). This also divides the temporal domain into equally spaced time steps.

3.5 Time Step

The time step is the specified period between which each iteration of the calculation is performed and will continually repeat the calculation for until thermal equilibrium is reached. The time step can be used to determine the amount of time to reach thermal equilibrium, as well as the number of iterations needed. Defining a time step is essential to capture the diffusion of temperature over time.

3.6 Stability

The explicit finite difference method incorporates a stability criterion that is known as the Fourier number, *eq.3-15*. The Fourier number characterises the rate at which heat is transferred across and

stored in the material surface. A high Fourier number therefore indicates a faster rate of heat transfer, and a lower number indicates a slower rate of heat transfer. If the Fourier number is greater than the stability criterion then it can be known that an unstable grid space and time step are being employed, either the grid spacing is too fine for the time step desired, or the time step is too large for the grid spacing specified.

For a 2D body the Fourier number must adhere to the following condition,

$$0 < F_0 \leq \frac{1}{4}$$

Eq. 3-15

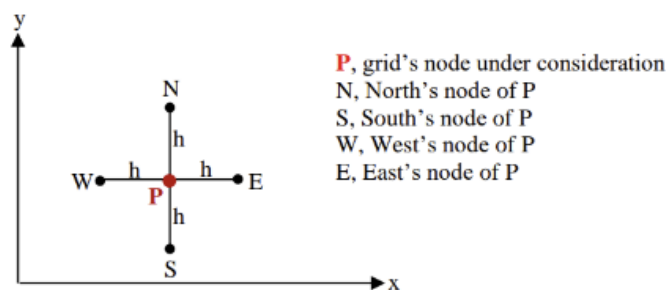
3.7 Explicit Finite Difference

The finite difference method is a numerical technique that transforms the governing differential equation by replacing the derivatives using the difference approximations derived from Taylor series. The resulting algebraic sets can be solved according the initial and boundary conditions.

For the diffusion model the difference method used was the explicit finite difference method. This was used to represent two discretisations in the model. To replace the first derivative forward difference is substituted to discretise the rate of temperature change over time $\partial T / \partial t$ and to replace the second derivative the central difference is substituted to discretise the heat equation as a representative of the amount of change of heat across the plate.

To calculate a variety of temperatures at individual nodes across the plate, a uniform grid is created, where the grid spacing is defined as, $h = \Delta x = \Delta y$.

For the explicit finite difference method to be implemented the equation is related to a point P which has within the grid domain a northern, eastern, western and southern coordinate position relative to it.



(Marotin, 2023)

When transforming the approximations to relate to a square domain at a point P, the equation becomes,

At point P

$$\begin{cases} \frac{\partial^2 T}{\partial x^2} = \frac{T_E^t - 2.T_p^t + T_W^t}{h^2} \\ \frac{\partial^2 T}{\partial y^2} = \frac{T_N^t - 2.T_p^t + T_S^t}{h^2} \end{cases}$$

Eq. 3-16

And,

$$\frac{\partial T}{\partial t} = \frac{T_P^{t+\Delta t} - T_P^t}{\Delta t}$$

Eq. 3 17

Where,

$t = \text{time (s)}$

$T = \text{temperature (}^\circ\text{C)}$

The transformed approximations are then substituted into the diffusion equation,

$$T_P^{t+\Delta t} = F_0(T_E^t + T_W^t + T_N^t + T_S^t - (4 - \frac{1}{F_0}) \cdot T_P^t)$$

Eq. 3-18

Where the Fourier Number is, $F_0 = \frac{\alpha \cdot \Delta t}{h^2}$

4 Initial, and Constant Boundary Conditions

The initial boundary conditions used in the model are those which pertain to the analytical solution given, as the primary aim of the model is to test the accuracy of the numerical model, the boundary conditions must remain the same in order to validate and compare the results against the analytical solution that is given.

Initial boundary conditions as discussed in section 3.3, will need to be able to adhere to the creation of an evenly distributed grid structure across the 6x6m plate, so that a uniform node structure can be used to create accurate results and a realistically representative thermal diffusion model. To ensure this the grid can only be divided into a mesh where the size specified is divisible by 6, e.g., 3, 1, etc. If the grid is not divided into a size where there is no remainder then an even grid mesh will not be achieved, and this will cause the diffusion calculation to be non-uniform.

Initial conditions to define the domain region will be,

$$T(x, y, 0) = 0 \quad 0 < x < 6 \quad 0 < y < 6,$$

Where, T is temperature of the plate at the position (x, y) and time t .

The above condition defines the initial domain as a 6x6m square, where x and y are both at a point between 0 and 6 and are related to temperature and time.

Once the initial domain has been specified, the constant boundary conditions then define where the heat source is targeted and how greatly it affects the boundary regions. Further it is specified within the conditions that the model should run until thermal equilibrium is reached.

The equation for the constant boundary conditions are,

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

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

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

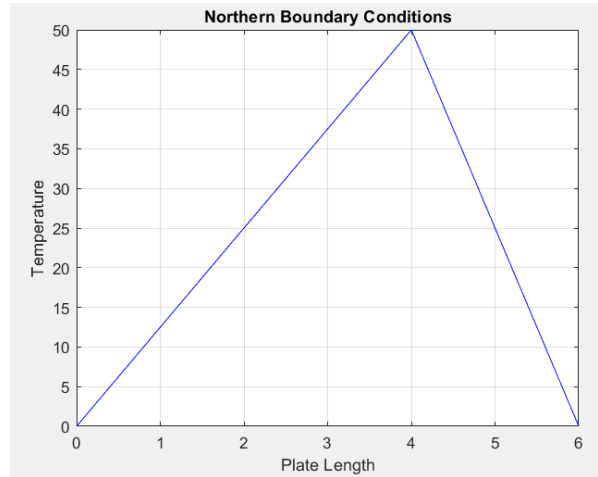
$$T(x, 0, t) = 0 \quad \text{for } 0 \leq x \leq 6 \quad \text{and } 0 \leq t \leq \text{thermal equilibrium}$$

Where,

$$f(x) = \begin{cases} 12.5x & \text{if } 0 \leq x \leq \left(\frac{2}{3} \times 6\right) \\ -25x + 150 & \text{if } \left(\frac{2}{3} \times 6\right) < x \leq 6 \end{cases}$$

Eq. 4-1

The constant boundary conditions derived from this equation show that the heat source impacts the northern boundary of the domain, while there is no heat source targeting the other boundaries. A heat source of 50°C impacts the plate at 4m across its length. The heat source diffuses across the plate, mostly in the left region of the plate (0-3m), and quickly reduces its heat across the right region of plate (4-6m), reaching 0°C at its 6m end. The northern boundary condition can be visualised as below,



As a means to test the accuracy of and compare the numerical model, the analytical solution will be calculated, in order to garner insight into the accuracy of the numerical method used and thereby justify its inclusion within the model. The analytical solution does not account for the time step, but rather is solved for a specified range. To attain a final result this range should be large enough to allow the analytical solution to achieve convergence.

The equation for the analytical solution is,

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

Eq. 4-2

(Daileda, 2012)

The boundary conditions discussed above are chosen for the model, as the analytical equation has been derived from them, and therefore they are integral to the equation. As discussed previously, using these conditions allows for the comparison of the numerical and analytical solution and therefore they must adhere to the same boundary conditions.

5 Program Code

Throughout this section, all details and components of the program will be provided and explained to give an overview of the models coding, delineating how each section is intended to function and its purpose within the overall program.

```
clc
clear
close all

%***** PLATE PROPERTIES *****
k = 143; %W/m.C (Conductivity)
p = 2.8*10^3; %kg/m^3 (Density)
c = 795; %J/kg.C (Specific Heat)
alpha = k/(p*c); %m^2.s^-1 (Thermal Diffusivity)
xdim = 6; %metres (x dimension)
ydim = 6; %metres (y dimension)
```

The code is firstly initialised with clear and close commands, to remove all results and windows from any previous runs, refreshing the user interface each time. Additionally, the properties of the plate are defined here.

```
%***** GRID SPACING USER INPUT & VALIDATION *****
GS = input('Please enter the grid spacing (m): '); %grid spacing input

while GS <= 0 || GS >= xdim || rem(xdim,GS)~=0 || rem(ydim,GS)~=0
    fprintf('\n*Note* A "valid" grid space, must be less than 6 and uniform, e.g. divisible by 6.\n')
    GS = input('\nPlease enter a valid grid spacing value (m): ');
end
```

Subsequently, the program requests the desired grid size as an input from the user. As the grid size needs to be within a specified range and be uniform, the user is asked to enter another grid spacing and given a warning if the grid size entered does not adhere to the specified criteria.

To check that the input is valid the code performs a remainder check to ensure that the size creates a uniform grid (e.g. is divisible by the dimension 6m), alongside validating that the input is greater than 0, but less than the x dimension, e.g. 6.

```
%***** TEMPERATURE MATRIX *****
nR = (ydim/GS) + 1; %number of rows
nC = (xdim/GS) + 1; %number of columns
TMatrix = zeros(nR,nC); %initialise temperature matrix
```

Next, the temperature matrix is initialised as a matrix to the plate dimensions and each space within it populated with 0.

```

%***** TIME STEP INPUT AND VALIDATION *****
maxTS = (GS^2)/(4*alpha); %maximum time step
TS = input('Please enter the time step (seconds): '); %time step input

while TS > maxTS
    fprintf('The maximum time step is %g seconds',maxTS);
    TS = input('\nPlease enter a valid time step (seconds): ');
end

```

The user is then asked to enter a time step, which must pass validation criteria and not exceed the maximum permissible time step, if this happens, the user is given a brief warning and asked to input a valid time step lower than the calculated maximum time step. The maximum time step is dependent on the grid spacing and therefore changes with each program iteration.

```

%***** TEMPERATURE MATRIX POPULATION *****
for c=1:nC
    x=(c-1)*GS; %number of columns to distance conversion
    if x <=((2/3)*xdim)
        TMatrix(1,c) = 12.5*x;
    else
        TMatrix(1,c) = (-25*x)+150;
    end
end
disp(TMatrix)

```

Once the matrix has been initialised the boundary conditions according to $f(x)$ are implemented into the program, and the zeroes which populate the matrix are overwritten by the constant boundary conditions. The northern boundary conditions are implemented as given, as shown in section 4.

The variable 'c' is used to allocate the distance along the northern boundary of each node point. Based on each node points distance, the temperature value (initialised as 0), is overwritten with the boundary condition initial temperature. As a results the matrix has now been populated with its correct initial temperature conditions.

```

%***** THERMAL DIFFUSIVITY INITIALISATIONS *****
F0 = (alpha*TS)/(GS^2); %fourier number
T = TMatrix; %Temperature Matrix after time step
tE = 0.01; % thermal equilibrium threshold
MDT=tE+1; %maximum difference in the temperature
TSCount = 0; %Time Step Counter
figure(1) %initialise temperature visualisation

```

For the diffusivity to be calculated, the values needed are initialised and defined. The thermal equilibrium threshold is defined here as 0.01, this will be the maximum amount of difference between the subsequent runs before the calculation is considered 'converged'.

```

***** THERMAL DIFFUSIVITY CALCULATION & VISUALISATION *****
while MDT > tE %repeat until thermal equilibrium threshold reached
    for r = 2:nR-1
        for c = 2:nC-1
            TMatrix(r,c) = F0*(T(r,c+1) + T(r,c-1) + T(r-1,c) + T(r+1,c) - (4-1/F0)*T(r,c));
        end
    end
    D = abs(T - TMatrix); %temperature difference between iterations
    MDT = max(D,[],'all'); %maximum temperature difference
    T = TMatrix
    TSCount = TSCount + 1;
    TLim = (max(T,[],'all')); %maximum value for plot limit

```

The thermal diffusivity is calculated in a while loop, repeating until the maximum difference between the previous and current iteration of the loop is less than the thermal equilibrium threshold. For each node in the matrix the temperature diffusivity is calculated and displayed on the console window, until the threshold is reached.

Calculating this requires that at the beginning of every iteration, a check is performed to determine whether or not the given tolerance threshold has been reached. Then to populate the temperature matrix at each iteration, for each node within the matrix the numerical calculation is performed, and the temperature result stored within the matrix location. Once the matrix is populated, the difference between the current and previous iteration of the loop is calculated, the maximum difference between the matrices is extracted and this is what is used to determine whether or not the thermal equilibrium threshold has been reached or not. If the maximum temperature difference between the matrices is less than the tolerance value, then the loop is closed, and thermal equilibrium is considered to have been reached. At the end of each iteration the previous iteration of the temperature matrix is allocated to the variable 'TMatrix', which allows the numerical solution to be calculated for an updated set of values, giving the ability to model the diffusivity over time.

```

%Graph plot
contourf(flip(T))
colormap("jet")
colorbar
clim([0 round(TLim)])
xlabel('Column Number')
ylabel('Row Number')
title('Temperature Diffusion across a 2D Aluminium Plate - Numerical')
pause(0.2)

end

disp(TMatrix)
TSCount = TSCount*TS;

```

Additionally, for each iteration in the while loop, a contoured heat map of the thermal diffusivity is displayed, pausing for 0.2 seconds between iterations, to create an evolving graph which displays the diffusivity over time as the iterations are processed.

```

if TSCount<60
    fprintf('The time to reach thermal equilibrium is: %g seconds.\n',TSCount)
elseif TSCount>=60 && TSCount<3600
    TSCount = TSCount/60;
    fprintf('The time to reach thermal equilibrium is: %g minutes.\n',TSCount)
elseif TSCount>=3600 && TSCount<86400
    TSCount = TSCount/3600;
    fprintf('The time to reach thermal equilibrium is: %g hours.\n',TSCount)
elseif TSCount>=86400
    TSCount = TSCount/86400;
    fprintf('The time to reach thermal equilibrium is: %g days.\n',TSCount)
end

```

The time to converge is then displayed to the user in an appropriate unit, dependant on its size in seconds it is converted to an alternative unit and displayed to the user.

```

%***** ANALYTICAL COMPARISON *****
TA = zeros(nR,nC); %Analytical solution matrix initialisation

for r = 1:nR
    for c = 1:nC
        y = (r-1)*GS;
        x = (c-1)*GS;
        for n = 1:225
            TA(r,c) = TA(r,c) + (sin(2*n*pi/3)/(n^2*sinh(n*pi)))*(sin(n*pi*x/6))*(sinh(n*pi*y/6));
        end
        TA(r,c) = TA(r,c) * 450/pi^2;
    end
end

TA = flip(TA)

```

To understand the accuracy of the numerical solution the analytical solution is calculated and displayed. This follows the same process as for calculating the numerical solution. The temperature within each point in the matrix is calculated and allocated to its node point.

However, unlike the numerical solution, the analytical solution is not run for multiple iterations until a point is reached and is calculated once for the solution.

```

%***** ANALYTICAL SOLUTION VISUALISATION *****
TALim = (max(TA,[],'all')); %maximum value for colourbar boundary condition

figure(2)
contourf(flip(TA))
colormap("jet")
colorbar
clim([0 round(TALim)])
xlabel('Column Number')
ylabel('Row Number')
title('Temperature Diffusion across a 2D Aluminium Plate - Analytical')

%***** DIFFERENCE BETWEEN NUMERICAL & ANALYTICAL SOLUTION *****
TD = T - TA; %Difference calculation between matrices
TD = abs(TD)
MNAD = (max(TD,[],'all')); %maximum value of difference between numerical and analytical solution

%Visualisation
figure(3)
contourf(flip(TD))
colormap("jet")
colorbar
clim([0 round(MNAD)])
xlabel('Column Number')
ylabel('Row Number')
title('Difference Between Numerical and Analytical Solutions')

```

Finally, the program creates a contour heatmap to display the analytical solution, and the difference between the numerical and analytical solutions are calculated and plotted as a contour heatmap.

6 Code Validation

In order to ensure that the code can appropriately perform its intended function, the difference between the numerical and analytical solution was calculated and a temperature map created in order to understand the quantity and location of variations within the results.

An initial run was performed with a grid size of 1 and a time step of 3600 seconds.

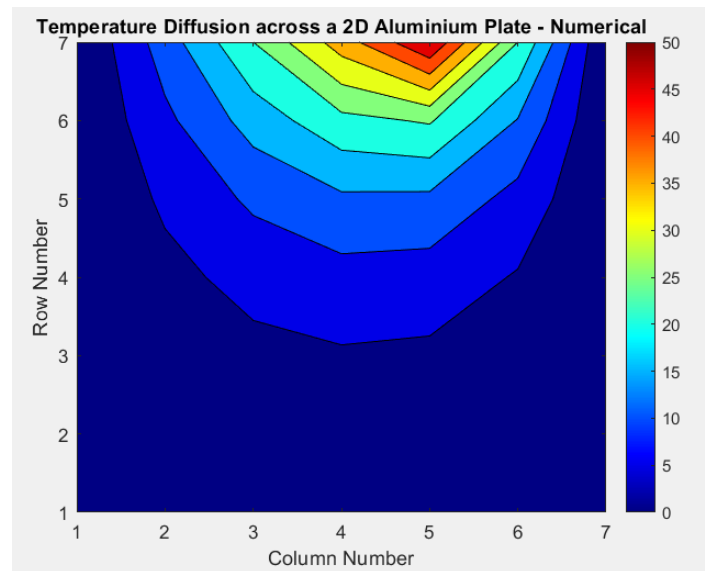


Figure 1 - Numerical Solution - Temperature Map

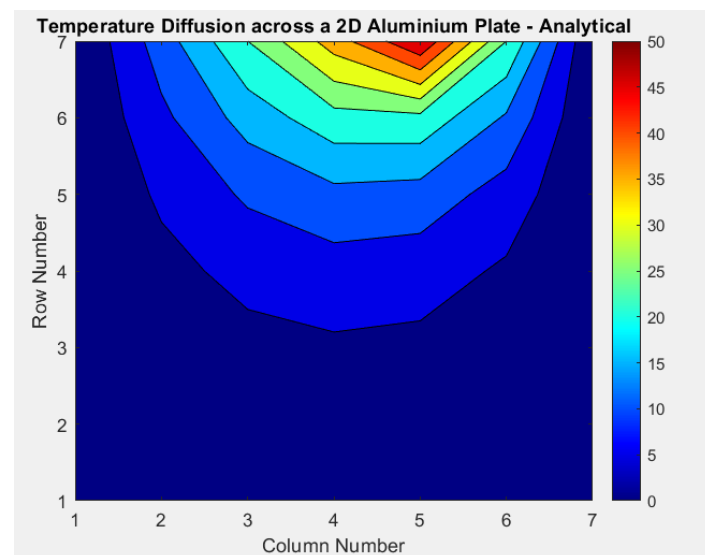


Figure 2 - Analytical Solution - Temperature Map

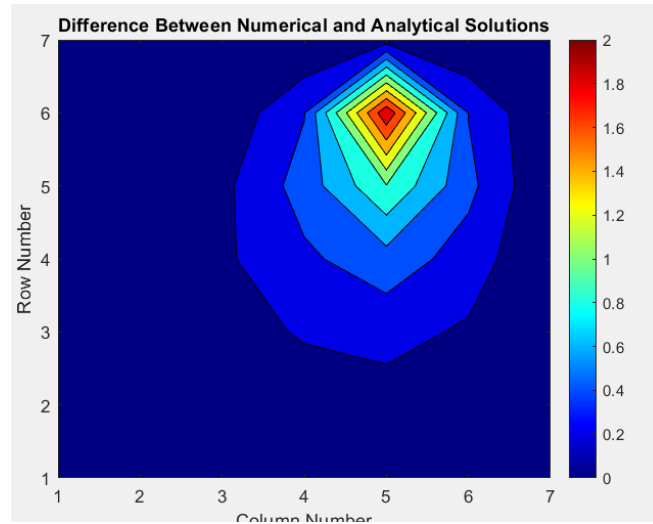


Figure 3 - Difference between numerical and analytical solutions - Temperature Map

TD =

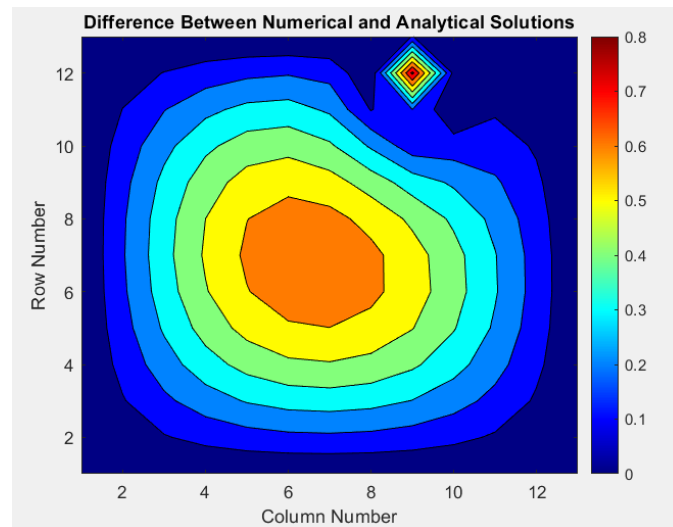
0	0.0000	0.0000	0.0000	0.1027	0.0000	0.0000
0	0.0035	0.0467	0.3889	1.9662	0.3924	0.0000
0	0.0324	0.1510	0.4869	0.9875	0.4545	0.0000
0	0.0535	0.1656	0.3624	0.5186	0.3089	0.0000
0	0.0473	0.1219	0.2191	0.2680	0.1718	0.0000
0	0.0261	0.0614	0.1003	0.1141	0.0742	0.0000
0	0	0	0	0	0	0

Table 1 - Difference between numerical and analytical solutions

The largest differences between values occur within the regions where the heat source diffuses across at its highest temperature. The maximum difference in temperature is 1.9662, seen in table 1, which occurs at the point below where the impacting heat source reaches its maximum temperature of 50°C. Beside these most other differences in temperature are minimal, mostly differing by around 0.2 – 0.5°C. The highest difference in values between the numerical and analytical solutions are concentrated around the point at which the maximum difference of 1.9662 occurs and become more accurate the further the matrix location is from this result.

There is expected to be an amount of error between the result of the numerical and analytical solutions, therefore it is reasonable given the small difference between the results, to conclude that the model is appropriate for numerical calculation and can with a reasonable accuracy calculate the diffusion of heat across the domain.

To further verify the behaviour of the model a second run was performed for a grid space of 0.5 and a time step of 360 seconds.



From this second run it can be seen how overall, the error is far lower than that of the previous run, with the previous run having a maximum error of almost 2, whilst this successive run peaks at a maximum difference of just below 0.8. The main concern in validating the model in this instance is verifying that the amount of error reduces the finer the grid space and therefore mesh becomes. It can be seen between the two runs that this is the case, and it is therefore fair to conclude the model validated.

7 Sensitivity Analysis

Further validating the code will require an in-depth understanding of the values that govern the accuracy of the models results. The tolerance and user input values can greatly impact the results that are computed, and it is vital to understand how each of these influences the results in order to justify the model. The sensitivity analysis performed, will explore the effect of the grid spacing, time step, and thermal equilibrium tolerance. From this it is expected that a thorough understanding will be gained, of how the model behaves and computes results based on these values.

Each of the specified variables will have their effect on the model output measured against their effect on the time to reach thermal equilibrium.

7.1 Grid Spacing

To analyse the influence of the grid space value, the time step was fixed at 150 seconds, and the thermal equilibrium tolerance fixed at 0.001.

Grid Spacing (h)	Time Step (fixed)	Tolerance (fixed)	Thermal Equilibrium Time (seconds)
3	150	0.001	129300
2	150	0.001	119400
1.5	150	0.001	121500
1.2	150	0.001	118050
1	150	0.001	121050
0.75	150	0.001	119700
0.5	150	0.001	119550
0.25	150	0.001	119100
0.2	150	0.001	119100

Table 2 - Grid Spacing Variation

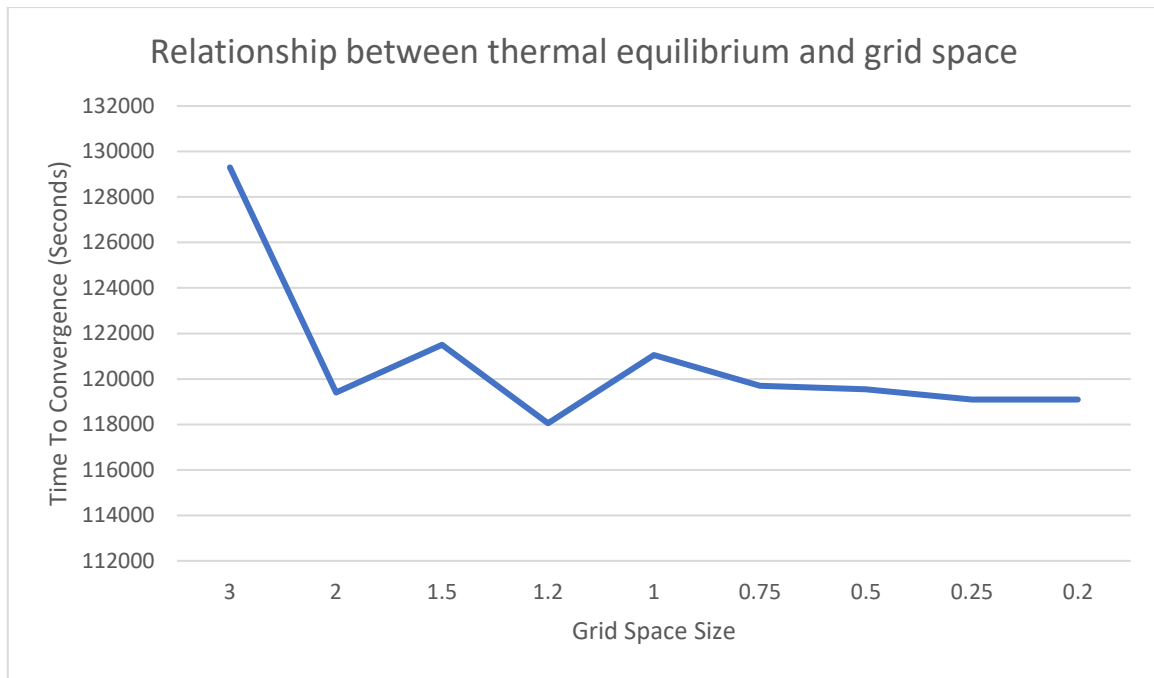


Figure 4 - Relationship between thermal equilibrium and grid space.

As shown in fig.4, decreasing the grid spacing value converges the solution towards higher stability, with less fluctuation between iterations (Liu et al., 2019). As the grid space value decreases, the mesh becomes finer. As the mesh becomes finer in difference approximation, the error term decreases, leading to greater accuracy in calculations. The explicit finite difference approach achieves greater accuracy as the grid space value decreases, resulting in a computation closer to convergence (Slater, 2021). This vast increase in accuracy and similarity between iterations occurs because as the grid space size and therefore mesh becomes finer, the truncation error reduces (Frayse et al., 2011). Additionally, there will exist some level of round-off error within the result, due to the quantity of decimal places a value is calculated at being reduced within the numerical model (Qin and Liao, 2018).

7.2 Time Step

To analyse the influence of the time step value, the grid space was fixed at 0.75, and the thermal equilibrium tolerance fixed at 0.0001.

Grid spacing h (fixed)	Time Step	Tolerance (fixed)	Thermal Equilibrium Time (seconds)
0.75	2000	0.0001	254000
0.75	1500	0.0001	247500
0.75	1000	0.0001	238000
0.75	800	0.0001	232000
0.75	600	0.0001	224400
0.75	400	0.0001	213200
0.75	200	0.0001	194000

0.75	100	0.0001	174200
0.75	50	0.0001	154350
0.75	25	0.0001	134475

Table 3 - Time Step Variation

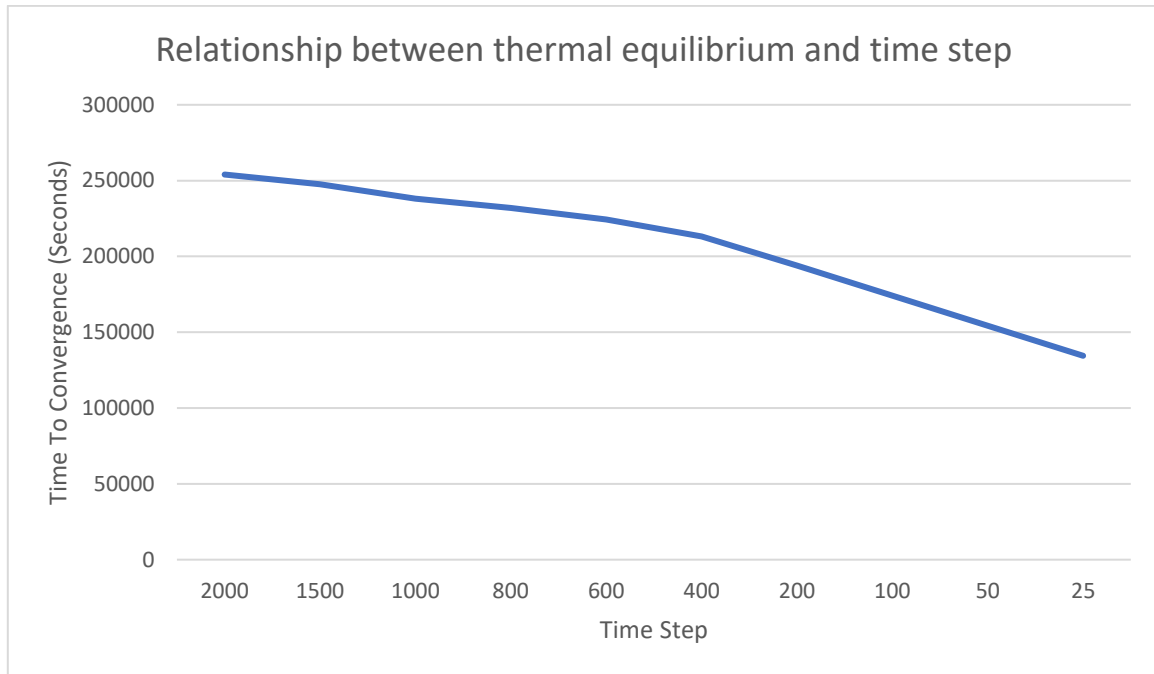


Figure 5 - Relationship between thermal equilibrium and time step.

The time step size as displayed in the graph, *fig.5.*, greatly impacts the accuracy of the results. If too large a time step is used, then the code may stop too soon before a reasonable level of accuracy has been achieved (Li et al., 2017). A smaller time step will capture more data as the heat source diffuses and will therefore be able to output a more accurate calculation (Li et al., 2022), however despite computing far more accurate results, smaller time steps require a much higher level of computational power, and will take a longer time to converge, there is therefore some level of balance that will need to be considered, between the accuracy of results and the amount of time they take to compute (Ghasemi, 2014). A smaller time step achieves a greater accuracy as it directly correlates to a reduction of the truncation error (Abdul et al., 2018).

7.3 Tolerance Threshold

To analyse the influence of the tolerance value, the grid space was fixed at 1, and the time step fixed at 500.

Grid spacing h (fixed)	Time step (fixed)	Tolerance	Thermal Equilibrium Time (seconds)
1	500	0.1	29000
1	500	0.01	89500

1	500	0.001	155500
1	500	0.0001	222000
1	500	0.00001	288000
1	500	0.000001	354500

Table 4 - Tolerance Variation

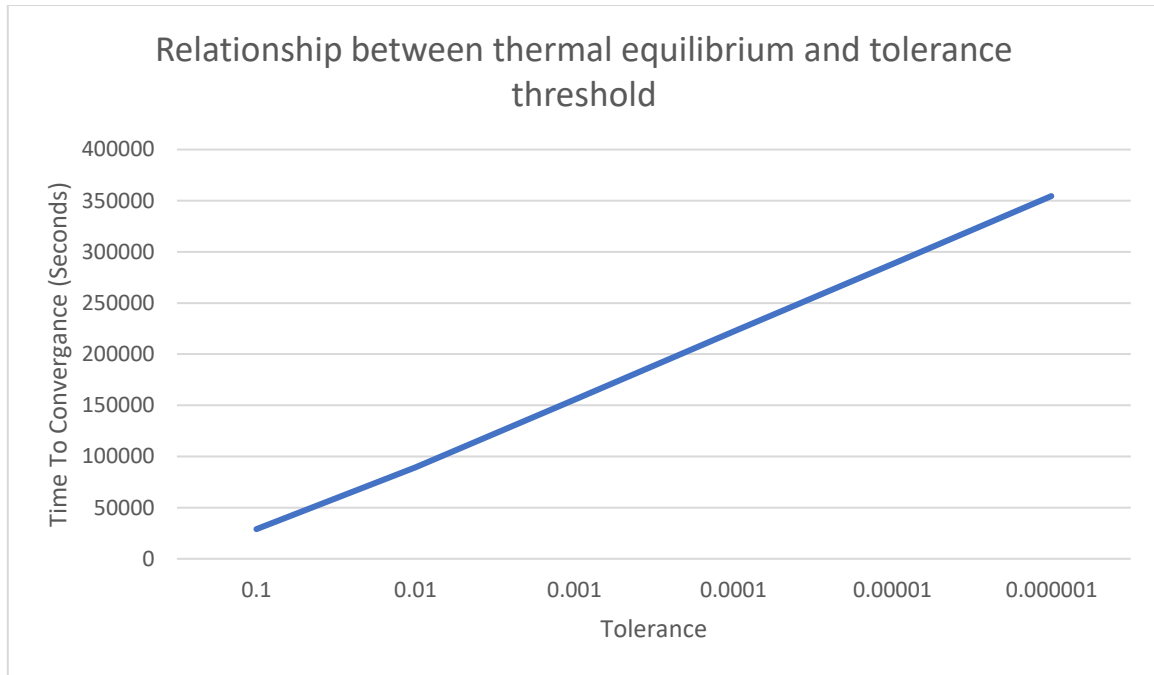


Figure 6 - Relationship between thermal equilibrium and tolerance threshold

The tolerance value is the point at which the model concludes that a high enough level of accuracy has been achieved and it considers this point 'thermal equilibrium', however it is unlikely that true equilibrium has been reached, and more so assumes that any further precision between values can be considered negligible. The tolerance as discovered increases the convergence time significantly, the finer it becomes (Yasuda, 2020). While a small tolerance value makes the convergence time greater (Kim et al., 2012), it does not necessarily follow with a higher accuracy. A tolerance value which is too small also risks the programme running indefinitely and not reaching convergence, however if too large a tolerance value is used then the code may stop too soon and output inaccurate results (Harris et al., 2023). The trade-off between the level of accuracy achieved and the time to compute is the determining factor for selecting the tolerance value. For the model a lower level of accuracy may be considered acceptable, as the aim is simply to validate a model, meaning that a slightly lower level of accuracy can be permissible, whilst still being able to validate the model.

8 Model Limitations

Limitations to the model stem from the numerical method used and the models coding.

Due to the numerical method used there is an error term that is carried across the calculations, which is caused by one or more of the following errors.

Round-off error: the round-off error is caused by the rounding of values to fewer decimal points. This rounding across large data sets can quickly snowball into a large amount of difference between the numerical and analytical solutions (Rosinger, 1985).

Truncation error: the truncation error is the difference of the function's analytical derivative and the numerically approximated derivative (Neill et al., 2018). In any approximation this is inevitable, as by its very nature the approximation is a way of redefining the function with a similar but not exact polynomial. The effect of this error can be minimised by calculating for an increased number of iterations; however, this cannot guarantee an identical numerical solution to analytical (Nayak et al., 2018).

These two error types can be seen in the sensitivity analysis performed in section 7 above. As the grid spacing and time step are reduced the truncation and round off errors are reduced.

Additionally, there are limitations to the uses of the model, these being,

Complex Geometry: the explicit finite difference method used in the code is unable to calculate a complex geometry and is generally used for simple shapes (Santana et al., 2022). This is as it requires the shape to be conducive to a uniform grid, to be able to accurately calculate a result, there is further the issue of generating a quality grid to begin with (Seeber et al., 1981). Additionally, creating boundary conditions would be significantly more complicated, as the explicit finite difference method generally requires conditions to be applied explicitly at each time step (Vigmond et al., 2019).

Stability: the explicit finite difference method is constrained by the stability criteria. Due to the inclusion of the Fourier number, the time step used cannot be any desired value, but must be within the stability criteria.

There are numerous limitations in the model due to the coding, the main limitation here is the method of modelling thermal equilibrium. Alternative methods such as analytical approximation or alternative numerical methods could yield a higher level of accuracy (Mazumder, 2015). A more practical approach that could be implemented into the program without changing the numerical scheme could be to monitor the maximum difference between the matrix values for more than two successive runs, for example, if it the code were able to retain the values for 5-10 iterations, and compare the maximum differences, concluding thermal equilibrium when they have all achieved maximum values below the tolerance threshold, a more definite conclusion that thermal equilibrium has been reached could be made.

Further, the tolerance value creates a significant limitation as, the program defines one tolerance value and assumes that it will be the most appropriate for all input values. It is therefore difficult to assign a value that can output accurate results in all scenarios. In line with this, the model defines thermal equilibrium as the point at which the maximum value of difference between two successive

temperature maps is lower than the tolerance threshold. In the aim of creating a realistic model, this can create inaccuracies as it assumes that if values have lowered beneath the threshold for two successive iterations, that they will not exceed this again, and further iterations are not necessary, therefore assuming thermal equilibrium. However, there is no guarantee that in a real scenario or, even in the model that just because values have reached below the threshold, that they will not exceed this again with further iterations and it therefore has not necessarily reached convergence.

9 Further Work

Improving the model will require the exploration of alternative methods and further and more complex model developments.

There exist several alternative numerical methods, which could be used in place of the explicit finite difference method employed in the model, in order to overcome the geometry compatibility and allow complex geometries to be modelled. These being,

Finite Element Method: this method approximates continuous functions into discrete models, using this method allows for an easier discretisation and higher accuracy across complex geometries, additionally making calculation across a 3D geometry possible (Tekkaya et al., 2014).

Finite Volume Method: this method discretises the spatial domain into a series of control volumes, over which the equations can be integrated (Cai, 1990). Again, this numerical method allows for calculation across complex geometries with a non-uniform grid structure not being required and is also able to remain within its stability criteria for a broader range of values (Kolditz, 2002).

Boundary Element Method: this method only operates within the spatial domain boundary, discretising only the values within it (Hsiao, 2006). Compared to the explicit finite difference method, the boundary element method has a much greater efficiency when calculating results for problems with bigger spatial domains, due to its discretisation of only the boundaries. Additionally, because it discretises only the boundaries, it does not require the geometry to have a grid across it, and therefore is optimal at solving problems with complex geometries (Yin et al., 2022).

The stability criteria included in the model, severely restricts the time step values which can be used, in an attempt to overcome this problem, the Alternating-direction Implicit method and Implicit Finite Difference method.

Alternating-direction Implicit Method: this method increases the accuracy of output by dissecting the problem into parts by alternate direction and then calculating these parts individually. This allows for larger time steps, reducing the amount of computational inaccuracy and can output accurate results for more complex problems (Jiang et al., 1991).

Implicit Finite Difference Method: this method solves differential equations by discretising the spatial domain, but also considering the unknown values of further time steps. The implicit method creates higher stability by calculating based on the effect of future values as well as present, rather than merely present values like the explicit method (Wais, 2014).

To require the implementation of any of these methods, it will depend on the criteria specified in the problem being solved. For example, the use of an alternative numerical method may not be needed unless conditions were such that it is required, i.e. complex spatial domain geometry. The specific criteria will depend on the conditions of the analytical solution being used as a basis for validation and comparison.

10 Conclusion

Throughout the report, it has been shown how using the explicit finite difference method a model is able to be created within MATLAB which can within a reasonable level of accuracy calculate and visually represent the diffusion of heat across a 2D aluminium plate. The numerical methods used and justification for them have been shown and have proven to be able to allow the model to calculate the result within an acceptable level of accuracy. The model structure and code has been explained along with the boundary conditions applied and can be found unabridged within the appendix of this report. Once created the code was able to be validated and compared to the analytical solution, both visually and numerically, confirming definitively its ability to compute accurate results. Further, a sensitivity analysis was performed in order to gain an understanding of the influence of the key parameters which control the output accuracy of the model. From this a justification of the values used within the programme was able to be established. Limitations were discussed with regard to both the numerical method and code, with suggestions for further work and alternative methods.

References

- Croft D.R, Lilley D.G. (1977), *Heat Transfer Calculations using the Finite Difference Equations*. London: Applied Science Publishers Ltd.
- Wheeler, N. (1997) *Applied Theta Functions*. Portland: Reed College.
- Gorguis, A. and Benny Chan, W.K. (2008). Heat equation and its comparative solutions. *Computers & Mathematics with Applications*, [online] 55(12), pp.2973–2980.
doi:<https://doi.org/10.1016/j.camwa.2007.11.028>.
- Narasimhan, T.N. (1999). Fourier's heat conduction equation: History, influence, and connections. *Journal of Earth System Science*, 108(3), pp.117–148.
doi:<https://doi.org/10.1007/bf02842327>.
- Fatima, N. (2021). The Study of Heat Conduction Equation by Homotopy Perturbation Method. *SN Computer Science*, 3(1). doi:<https://doi.org/10.1007/s42979-021-00947-4>.
- Nimona Ketema Kebeba and Gizaw Debito Haifo (2022). Analysis of Two-Dimensional Heat Transfer Problem Using the Boundary Integral Equation. *Advances in Mathematical Physics*, 2022, pp.1–7.
doi:<https://doi.org/10.1155/2022/1889774>.
- Encyclopedia Britannica. (n.d.). *Parabolic equation*. [online] Available at:
<https://www.britannica.com/science/parabolic-equation>.
- Lustgarten, J.L., Gopalakrishnan, V., Grover, H. and Shyam Visweswaran (2008). Improving classification performance with discretization on biomedical datasets. *American Medical Informatics Association Annual Symposium*, pp.445–9.
- Chang, Y.F. and Corliss, G. (1994). ATOMFT: solving ODEs and DAEs using Taylor series. *Computers & Mathematics with Applications*, 28(10-12), pp.209–233. doi:[https://doi.org/10.1016/0898-1221\(94\)00193-6](https://doi.org/10.1016/0898-1221(94)00193-6).
- Koffi Messan Agbavon, Appanah Rao Appadu, Bilge Inan and Herve Michel Tenkam (2022). Convergence Analysis and Approximate Optimal Temporal Step Sizes for Some Finite Difference Methods Discretising Fisher's Equation. *Frontiers in Applied Mathematics and Statistics*, 8.
doi:<https://doi.org/10.3389/fams.2022.921170>.
- Rosinger, E.E. (1985). Propagation of round-off errors and the role of stability in numerical methods for linear and nonlinear PDEs. *Applied Mathematical Modelling*, 9(5), pp.331–336.
doi:[https://doi.org/10.1016/0307-904x\(85\)90019-8](https://doi.org/10.1016/0307-904x(85)90019-8).
- Neill, S.P. and Hashemi, M.R. (2018). Ocean Modelling for Resource Characterization. *Fundamentals of Ocean Renewable Energy*, pp.193–235. doi:<https://doi.org/10.1016/b978-0-12-810448-4.00008-2>.

Nayak, S. and Chakraverty, S. (2018). Interval Arithmetic. *Interval Finite Element Method with MATLAB*, pp.1–6. doi:<https://doi.org/10.1016/b978-0-12-812973-9.00001-1>.

Tekkaya, A.E. and Soyarslan, C. (2014). Finite Element Method. *CIRP Encyclopedia of Production Engineering*, [online] pp.508–514. doi:https://doi.org/10.1007/978-3-642-20617-7_16699.

Cai, Z. (1990). On the finite volume element method. *Numerische Mathematik*, 58(1), pp.713–735. doi:<https://doi.org/10.1007/bf01385651>.

Kolditz, O. (2002). Finite Volume Method. *Computational Methods in Environmental Fluid Mechanics*, pp.173–190. doi:https://doi.org/10.1007/978-3-662-04761-3_8.

Hsiao, G.C. (2006). Boundary element methods—An overview. *Applied Numerical Mathematics*, 56(10-11), pp.1356–1369. doi:<https://doi.org/10.1016/j.apnum.2006.03.030>.

Jiang, H. and Yau Shu Wong (1991). A parallel alternating direction implicit preconditioning method. *Journal of Computational and Applied Mathematics*, 36(2), pp.209–226. doi:[https://doi.org/10.1016/0377-0427\(91\)90027-h](https://doi.org/10.1016/0377-0427(91)90027-h).

Wais, P. (2014). Implicit Finite-Difference Method for Solving Transient Heat Conduction Problems. *Springer eBooks*, pp.2368–2387. doi:https://doi.org/10.1007/978-94-007-2739-7_898.

Bi, Z. (2018). Description of Elements. *Elsevier eBooks*, pp.61–111. doi:<https://doi.org/10.1016/b978-0-12-809952-0.00003-0>.

Liu, Y., Zhang, W. and Kou, J. (2019). Mode multigrid - A novel convergence acceleration method. *Aerospace Science and Technology*, 92, pp.605–619. doi:<https://doi.org/10.1016/j.ast.2019.06.001>.

Slater, J.W. (2021). *Examining Spatial (Grid) Convergence*. [online] www.grc.nasa.gov. Available at: <https://www.grc.nasa.gov/www/wind/valid/tutorial/spatconv.html>.

Ghasemi, A., Mousavi, S. and Banisi, S. (2014). Effect of time step on accuracy of DEM calculations. XXVII International Mineral Processing Congress.

Li, Y., Choi, Y. and Kim, J. (2017). Computationally efficient adaptive time step method for the Cahn–Hilliard equation. *Computers & Mathematics with Applications*, 73(8), pp.1855–1864. doi:<https://doi.org/10.1016/j.camwa.2017.02.021>.

Li, X., Li, X., Wu, Y., Wu, L. and Yue, Z. (2022). Selection criteria of mesh size and time step in FEM analysis of highly nonlinear unsaturated seepage process. *Computers and Geotechnics*, 146, p.104712. doi:<https://doi.org/10.1016/j.compgeo.2022.104712>.

Yasuda, Y., Saito, K. and Sekine, H. (2020). Effects of the convergence tolerance of iterative methods used in the boundary element method on the calculation results of sound fields in rooms. *Applied Acoustics*, 157, p.106997. doi:<https://doi.org/10.1016/j.apacoust.2019.08.003>.

Kim, S.Y., Kim, I.Y. and Mechefske, C.K. (2012). A new efficient convergence criterion for reducing computational expense in topology optimization: reducible design variable method. *International Journal for Numerical Methods in Engineering*, 90(6), pp.752–783.

doi:<https://doi.org/10.1002/nme.3343>.

Seeber, Renato. and Stefani, Stefano. (1981). Explicit finite difference method in simulating electrode processes. *Analytical Chemistry*, 53(7), pp.1011–1016.

doi:<https://doi.org/10.1021/ac00230a020>.

Santana, H.S., R.P, M. and Júnior, S. (2022). Intensification of biodiesel production through computational fluid dynamics. *Elsevier eBooks*, pp.231–271. doi:<https://doi.org/10.1016/b978-0-12-824117-2.00011-9>.

Vigmond, E. and Plank, G. (2019). Cardiac Modeling. *Encyclopedia of Biomedical Engineering*, pp.1–20. doi:<https://doi.org/10.1016/b978-0-12-801238-3.99985-9>.

Mazumder, S. (2015). *Numerical Methods for Partial Differential Equations*. Academic Press.

Yin, H., Song, G., Zhang, L. and Wu, C. (2022). Introduction. *Elsevier eBooks*, pp.1–28.

doi:<https://doi.org/10.1016/b978-0-12-819384-6.00009-7>.

Kalinowska, M.B. and Rowiński, P.M. (2007). Truncation errors of selected finite difference methods for two-dimensional advection-diffusion equation with mixed derivatives. *Acta Geophysica*, 55(1), pp.104–118. doi:<https://doi.org/10.2478/s11600-006-0046-3>.

Marotin, A. (2023). *ACMA6012 Sem2 CW1 2023-24*. Microsoft Teams.

Huang, H.-C., Li, Z.-H. and Usmani, A.S. (1999). Temporal Discretisation. *Springer eBooks*, pp.75–94.

doi:https://doi.org/10.1007/978-1-4471-0799-6_5.

Huang, H.-C. and Usmani, A.S. (1994). Temporal Discretisation for Heat Conduction. *Springer eBooks*, pp.49–61. doi:https://doi.org/10.1007/978-1-4471-2091-9_4.

Daileida R.C. (2012, March 6). The Two-Dimensional Heat Equation. Trinity University, San Antonio, Texas, USA.

Frayse, F., Rubio, G. and Valero, E. (2011). Mesh adaptation driven by truncation error estimates. In: *4th European Conference for Aerospace Sciences*. Spain: EUCASS.

Qin, S. and Liao, S. (2018). Influence of round-off errors on the reliability of numerical simulations of chaotic dynamic systems. *Journal of Applied Nonlinear Dynamics*, 7(2), pp.197–204.

doi:<https://doi.org/10.5890/jand.2018.06.008>.

Abdul Hamid, S.A., Adam, A., Jackson, M.D. and Muggeridge, A.H. (2018). Impact of truncation error and numerical scheme on the simulation of the early time growth of viscous fingering. *International Journal for Numerical Methods in Fluids*, 89(1-2), pp.1–15. doi:<https://doi.org/10.1002/flid.4680>.

Harris, R., Alkhraijah, M. and Molzahn, D.K. (2023). Optimally Managing the Impacts of Convergence Tolerance for Distributed Optimal Power Flow. *arXiv (Cornell University)*.
doi:<https://doi.org/10.48550/arxiv.2311.08305>.

Appendix

```
clc
clear
close all

%***** PLATE PROPERTIES *****
k = 143; %W/m.C (Conductivity)
p = 2.8*10^3; %kg/m^3 (Density)
c = 795; %J/kg.C (Specific Heat)
alpha = k/(p*c); %m^2.s^-1 (Thermal Diffusivity)
xdim = 6; %metres (x dimension)
ydim = 6; %metres (y dimension)

%***** GRID SPACING USER INPUT & VALIDATION *****
GS = input('Please enter the grid spacing (m): '); %grid spacing input

while GS <= 0 || GS >= xdim || rem(xdim,GS)~=0 || rem(ydim,GS)~=0
    fprintf('\n*Note* A "valid" grid space, must be less than 6 and uniform, e.g.
    divisible by 6.\n')
    GS = input('\nPlease enter a valid grid spacing value (m): ');
end

%***** TEMPERATURE MATRIX *****
nR = (ydim/GS) + 1; %number of rows
nC = (xdim/GS) + 1; %number of columns
TMatrix = zeros(nR,nC); %initialise temperature matrix

%***** TIME STEP INPUT AND VALIDATION *****
maxTS = (GS^2)/(4*alpha); %maximum time step
TS = input('Please enter the time step (seconds): '); %time step input

while TS > maxTS
    fprintf('The maximum time step is %g seconds',maxTS);
    TS = input('\nPlease enter a valid time step (seconds): ');
end

%***** TEMPERATURE MATRIX POPULATION *****
for c=1:nC
    x=(c-1)*GS; %number of columns to distance conversion
    if x <=((2/3)*xdim)
        TMatrix(1,c) = 12.5*x;
    else
        TMatrix(1,c) = (-25*x)+150;
    end
end
disp(TMatrix)

%***** THERMAL DIFFUSIVITY INITIALISATIONS *****
F0 = (alpha*TS)/(GS^2); %fourier number
T = TMatrix; %Temperature Matrix after time step
tE = 0.000001; % thermal equilibrium threshold
MDT=tE+1; %maximum difference in the temperature
TSCount = 0; %Time Step Counter
figure(1) %initialise temperature visualisation

%***** THERMAL DIFFUSIVITY CALCULATION & VISUALISATION *****
while MDT > tE %repeat until thermal equilibrium threshold reached
```

```

        for r = 2:nR-1
            for c = 2:nC-1
                TMatrix(r,c) = F0*(T(r,c+1) + T(r,c-1) + T(r-1,c) + T(r+1,c) - (4-
1/F0)*T(r,c));
            end
        end
        D = abs(T - TMatrix); %temperature difference between iterations
        MDT = max(D,[],'all'); %maximum temperature difference
        T = TMatrix
        TSCount = TSCount + 1;
        TLim = (max(T,[],'all')); %maximum value for plot limit

        %Graph plot
        contourf(flip(T))
        colormap("jet")
        colorbar
        clim([0 round(TLim)])
        xlabel('Column Number')
        ylabel('Row Number')
        title('Temperature Diffusion across a 2D Aluminium Plate - Numerical')
        pause(0.2)

    end

    disp(TMatrix)
    TSCount = TSCount*TS;
    fprintf('The time to reach thermal equilibrium is: %g seconds.\n',TSCount)

    if TSCount<60
        fprintf('The time to reach thermal equilibrium is: %g seconds.\n',TSCount)
    elseif TSCount>=60 && TSCount<3600
        TSCount = TSCount/60;
        fprintf('The time to reach thermal equilibrium is: %g minutes.\n',TSCount)
    elseif TSCount>=3600 && TSCount<86400
        TSCount = TSCount/3600;
        fprintf('The time to reach thermal equilibrium is: %g hours.\n',TSCount)
    elseif TSCount>=86400
        TSCount = TSCount/86400;
        fprintf('The time to reach thermal equilibrium is: %g days.\n',TSCount)
    end

    %***** ANALYTICAL COMPARISON *****
    TA = zeros(nR,nC); %Analytical solution matrix initialisation

    for r = 1:nR
        for c = 1:nC
            y = (r-1)*GS;
            x = (c-1)*GS;
            for n = 1:225
                TA(r,c) = TA(r,c) +
(sin(2*n*pi/3)/(n^2*sinh(n*pi)))*(sin(n*pi*x/6))*(sinh(n*pi*y/6));
            end
            TA(r,c) = TA(r,c) * 450/pi^2;
        end
    end

    TA = flip(TA)

    %***** ANALYTICAL SOLUTION VISUALISATION *****

```

```

TALim = (max(TA,[],'all')); %maximum value for colourbar boundary condition

figure(2)
contourf(flip(TA))
colormap("jet")
colorbar
clim([0 round(TALim)])
xlabel('Column Number')
ylabel('Row Number')
title('Temperature Diffusion across a 2D Aluminium Plate - Analytical')

%***** DIFFERENCE BETWEEN NUMERICAL & ANALYTICAL SOLUTION *****
TD = T - TA; %Difference calculation between matrices
TD = abs(TD)

%Visualisation
figure(3)
contourf(flip(TD))
colormap("jet")
colorbar
xlabel('Column Number')
ylabel('Row Number')
title('Difference Between Numerical and Analytical Solutions')

```