

COMPUTATIONAL METHODS

Coursework 2 of 2



Academic Coursework submitted in partial fulfilment of the requirement of

University of Wales Trinity Saint David for the award of the

BACHELOR OF ENGINEERING

in

MECHANICAL & MANUFACTURING ENGINEERING

Submitted by

JOSHUA J ASHCROFT

(2114692)

Under the guidance of

ARNAUD MAROTIN



PRIFYSGOL CYMRU
Y Drindod Dewi Sant
UNIVERSITY OF WALES
Trinity Saint David
SWANSEA - ABERTAWE

Contents

Contents	2
List of Figures & Tables.....	4
Constitutive Equation.....	5
Discretising The Constitutive Equation	7
Finite Difference Method	11
Time-Dependent Thermal Diffusion Using Finite Difference	12
Explicit Method	13
Stability Criterion	14
Boundary Conditions.....	15
Dirichlet Condition – Northern Boundary.....	15
Eastern and Western Boundaries.....	15
Southern Boundary	15
Code Details and Explanation	15
Section 1.....	16
Section 2.....	17
Section 3.....	18
Section 4.....	19
Validation of the Code	20
Sensitivity Analysis Of Code	22
Influence of Grid Spacing Variation.....	23
Influence of Tolerance Variation	24
Influence of Time Step Variation.....	24
Key Model Limitations.....	25

Coding Limitations	25
Truncation Error	25
Round-Off Error.....	25
Complex Geometry	25
Suggested Improvements and Future Developments	26
Conclusion.....	27
Summary	28
Bibliography	29
Appendix	33

List of Figures & Tables

Figure 1 - Boundary Conditions	
Figure 2 - Section 1 Code	
Figure 3 - Section 2 Code	17
Figure 4 - Section 3 Code	18
Figure 5 - Section 4 Code	19
Figure 6 - Analytical Solution	
Figure 8 - Numerical Heat Map	
Figure 7 - Analytical Heat Map	
Figure 9 - Difference Between Numerical and Analytical Data Table	21
Figure 10 - Grid Spacing Graph	
Figure 11 - Tolerance Variation Graph	24
Figure 12 - Time Step Variation	24
Table 1 - Grid Spacing Variation	22
Table 2 - Tolerance Variation	22
Table 3 - Time Step Variation	23

Constitutive Equation

The behavior of a material is governed by constitutive equations, which work alongside the conservation laws of mass, linear momentum, angular momentum, and energy. Unlike these conservation laws, constitutive equations vary between different materials, as they depend on the specific physical properties of each material.

In this report, we utilize the heat equation. This equation is derived from the fundamental principles of heat conduction by applying the law of conservation of energy to elements within a material. It is relevant to various physical situations, such as the cooling of objects, the thermal behavior of materials, and other heat transfer methods.

The 2D heat equation without a heat source term is mathematically defined as follows:

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

Here, $u(x, y, t)$ represents the temperature distribution at position x, y and time t , and α is the thermal diffusivity, a property of the material.

This equation includes two spatial derivatives—one with respect to x and another with respect to y —which represent the diffusion of heat in two spatial dimensions. The temporal derivative describes how the temperature changes over time. The absence of a heat source term implies that heat conduction, along with the initial and boundary conditions, are the only factors influencing the temperature distribution (Strauss, 2008).

This equation is applicable to problems involving the evolution of temperature in a two-dimensional region where heat is conducted through the material, and there are no additional internal heat sources. In this case, we are focusing on an aluminum plate.

The aim of this report is to determine the temperature distribution within the aluminum plate over time, as illustrated in Figure 1

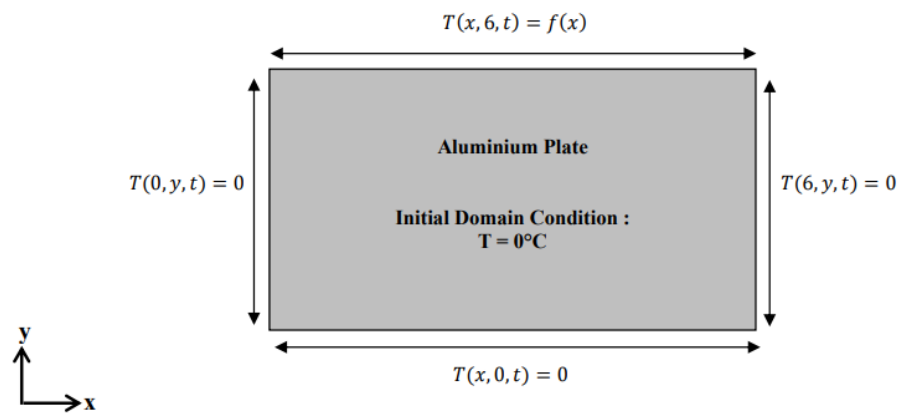


Figure 1 - Boundary Conditions

Discretising The Constitutive Equation

To employ the constitutive equation effectively, it is necessary to discretise it using the Taylor series, as outlined in Equation (0.1). This method helps improve the accuracy of results by replacing differential quantities with suitably small differences.

Taylor series of function $f(x)$ about point $x = x_0$ is

$$f(x) = f(x_0) + f'(x_0) \times \frac{(x-x_0)^1}{1!} + f''(x_0) \times \frac{(x-x_0)^2}{2!} + f'''(x_0) \times \frac{(x-x_0)^3}{3!} + f''''(x_0) \times \frac{(x-x_0)^4}{4!} + \dots \quad (0.1)$$

The Taylor series connecting the function's value at a point x_0 to its value at a point $(x_0 + h)$ is expressed as:

$$f(x_0 + h) = f(x_0) + h \cdot f'(x_0) + \frac{h^2}{2!} f''(x_0) + \frac{h^3}{3!} f^{(3)}(x_0) + \dots \quad (0.2)$$

This equation can then be rearranged to get:

$$\frac{f(x_0+h)-f(x_0)}{h} = f'(x_0) + \frac{h}{2!} f''(x_0) + \frac{h^2}{3!} f^{(3)}(x_0) + \dots \quad (0.3)$$

So, this can be used to make the **Forward Difference Approximation** of $f'(x_0)$ therefore:

$$f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h} \quad (1)$$

This is only an approximate, however the equation (1) gets more accurate as the value of h gets smaller. How the accuracy depends on h can be found out by ϵ_f denoting the error, using this forward difference approximation, so:

$$\epsilon_f \equiv f'(x_0) - \frac{f(x_0+h)-f(x_0)}{h} \quad (1.1)$$

$$= -\frac{h}{2!} f''(x_0) - \frac{h^2}{3!} f^{(3)}(x_0) + \dots \quad (1.2)$$

This approximation is of the order $O(h)$. This means that as the value of h gets smaller the error of the forward difference approximation also gets smaller, linearly.

To give a more accurate approximate the function can be looked at to see how it changes as we decrease the h value. This is creation of the backward difference approximation. Using Taylor series, we can write (0.2) to make:

$$f(x_0 - h) = f(x_0) - h \cdot f'(x_0) - \frac{h^2}{2!} f''(x_0) - \frac{h^3}{3!} f^{(3)}(x_0) - \dots \quad (1.3)$$

Rearranged:

$$\frac{f(x_0 - h) - f(x_0)}{h} = -f'(x_0) + \frac{h}{2!} f''(x_0) - \frac{h^2}{3!} f^{(3)}(x_0) + \dots \quad (1.4)$$

Then it can construct the following **backward difference approximation**:

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

This is only an approximate that is more accurate as h get smaller; this can be given by the the equation:

$$\epsilon_b \equiv f'(x_0) - \frac{f(x_0) - f(x_0 - h)}{h} \quad (2.1)$$

$$= \frac{h}{2!} f''(x_0) - \frac{h^2}{3!} f^{(3)}(x_0) + \dots \quad (2.2)$$

The approximate is again of $O(h)$.

So, in both equations if h decreases the error also decreases, meaning there is no real improvement or benefit from the forward difference approximation.

To enhance the precision of the approximation, we introduce a consistent estimation known as the central difference approximation, which considers the points $(x_0 - h)$ and $(x_0 + h)$ (Faires, 2015) If the function f is three times differential of x :

$$\frac{f(x_0 + h) - f(x_0)}{h} = f'(x_0) + \frac{h}{2!} f''(x_0) + \frac{h^2}{3!} f^{(3)}(x_0) + \dots \quad (0.3)$$

$$f(x_0 - h) = f(x_0) - h \cdot f'(x_0) - \frac{h^2}{2!} f''(x_0) - \frac{h^3}{3!} f^{(3)}(x_0) - \dots \quad (1.3)$$

$$f(x_0 + h) - f(x_0 - h) = 2 \cdot \left[h \cdot f'(x_0) + \frac{h^3}{3!} f^{(3)}(x_0) + \dots \right] \quad (2.3)$$

So,

$$\frac{f(x_0+h)-f(x_0-h)}{2.h} = f'(x_0) + \frac{h^2}{3!} f^{(3)}(x_0) + \dots \quad (2.4)$$

From this the **central difference approximation** can be given as:

$$f'(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2.h} \quad (3)$$

The error of this approximate is given by:

$$\epsilon_c \equiv f'(x_0) - \frac{f(x_0+h)-f(x_0-h)}{2.h} \quad (3.1)$$

$$= -\frac{h^2}{3!} f''(x_0) - \frac{h^4}{5!} f^{(5)}(x_0) + \dots \quad (3.2)$$

The central difference approximate is accurate to order $O(h^2)$ and so, for small values of h the central approximate will be better than both the forward and backward applications.

Second order derivatives

Similar steps are followed for higher order derivatives, if both approximates are added:

$$f(x_0 + h) = \left[f(x_0) + h.f'(x_0) + \frac{h^2}{2!} f''(x_0) + \frac{h^3}{3!} f^{(3)}(x_0) + \dots \right] \quad (3.3)$$

+

$$f(x_0 - h) = \left[f(x_0) - h.f'(x_0) + \frac{h^2}{2!} f''(x_0) - \frac{h^3}{3!} f^{(3)}(x_0) + \dots \right] \quad (3.4)$$

This gives:

$$f(x_0 + h) + f(x_0 - h) = 2. \left[f(x_0) + \frac{h^2}{2!} f''(x_0) + \frac{h^4}{4!} f^{(4)}(x_0) + \dots \right] \quad (3.5)$$

Therefore:

$$\frac{f(x_0+h)+f(x_0-h)-2.f(x_0)}{h^2} = f''(x_0) + \frac{h^2}{12} f^{(4)}(x_0) + \dots \quad (3.6)$$

The **central difference approximation** to the 2nd derivative is:

$$f''(x_0) \approx \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} \quad (4)$$

The error is of the order $O(h^2)$ is given by:

$$\begin{aligned} \epsilon_c &\equiv f''(x_0) - \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} \\ &= -\frac{h^2}{2!} f^{(4)}(x_0) + \dots \end{aligned} \quad (4.1)$$

Finite Difference Method

The Finite Difference Method (FDM) is a numerical technique used to approximate solutions to differential equations by discretizing the domain into a grid of points, known as nodes. Each node represents a discrete point on the material's surface where the temperature will be calculated. These nodes are systematically arranged in rows and columns, creating a mesh that spans the entire domain.

In FDM, the continuous partial differential equations governing heat conduction are replaced with algebraic equations. These algebraic equations establish relationships between the temperature values at adjacent nodes based on finite difference approximations of the derivatives. The boundary conditions, as defined in the brief, play a crucial role in shaping these relationships by providing necessary information about the temperature or heat flux at the edges of the material.

To enhance the accuracy of the numerical solution, one can increase the number of nodes, thus refining the spatial resolution. This refinement allows for a more detailed representation of temperature variations across the material. However, increasing the node density also escalates the computational workload, as more equations must be solved simultaneously. This trade-off between accuracy and computational cost is a key consideration in the application of FDM.

The FDM is one of several numerical methods used to solve partial differential equations, with others including Finite Element Analysis (FEA) and Finite Volume Analysis (FVA). Each method has its strengths and weaknesses. For instance, while FDM is relatively straightforward and easy to implement, FEA offers greater flexibility in handling complex geometries and boundary conditions. FVA, on the other hand, is particularly effective for problems involving fluid flow and heat transfer.

Time-Dependent Thermal Diffusion Using Finite Difference

To achieve the objectives outlined earlier, we will employ the heat conduction equation. In this context, α denotes a constant material property known as thermal diffusivity, which measures the rate at which heat spreads through the material from the edge. We will focus on the thermal diffusion equation in two dimensions, as the problem involves a 2D plate and does not consider the Z direction. The relevant equations (4.3) and (4.4) will be used to model this scenario.

$$\nabla^2 T = \frac{1}{\alpha} \cdot \frac{\partial T}{\partial t} \quad (4.3)$$

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{\alpha} \cdot \frac{\partial T}{\partial t} \quad (4.4)$$

T = Temperature ($^{\circ}\text{C}$) κ = Thermal Conductivity ($\text{W}\cdot\text{m}^{-1}\cdot\text{K}^{-1}$) c = Specific Heat ($\text{J}\cdot\text{Kg}^{-1}\cdot\text{K}^{-1}$)

ρ = Density ($\text{Kg}\cdot\text{m}^{-3}$) $\alpha = \kappa / (\rho \cdot c)$ thermal diffusivity ($\text{m}^2\cdot\text{s}^{-1}$) t = time (s)

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

Explicit Method

The differential thermal diffusion equation is represented in finite-difference form by using Central differences to discretise, $\nabla^2 T$, and Forward differences to discretise $\partial T / \partial t$.

At the point P

$$\frac{\partial^2 T}{\partial x} = \frac{T_E^t - 2.T_P^t + T_W^t}{h^2} \quad (4.5)$$

$$\frac{\partial^2 T}{\partial y} = \frac{T_N^t - 2.T_P^t + T_S^t}{h^2} \quad (4.6)$$

And

$$\frac{\partial T}{\partial t} = \frac{T_P^{t+\Delta t} - T_P^t}{h^2} \quad (4.7)$$

Substituting into the **thermal diffusion equation** gives us the **Explicit Finite difference Equation**

$$T_P^{t+\Delta t} = F_0 \cdot (T_E^t + T_W^t + T_N^t + T_S^t - \left(4 - \frac{1}{F_0}\right) \cdot T_P^t) \quad (5)$$

Where the **Fourier Number**

$$F_0 = \frac{\alpha \cdot \Delta t}{h^2} \quad (6)$$

Stability Criterion

In the explicit finite difference method, the Fourier number, F_0 , is a dimensionless quantity defined as:

$$F_0 = \frac{\alpha \cdot \Delta t}{h^2}$$

This method is termed "explicit" because the temperature $T_P^{t+\Delta t}$ at the next time step can be directly calculated from the temperatures at the current time step.

A significant drawback of the explicit method is the limitation on the time-step size Δt . Once the thermal diffusivity α and the spatial step size h are set, Δt must not exceed a certain maximum value to maintain numerical stability. This constraint is given by the stability criterion:

$$0 < F_0 \leq \frac{1}{4} \quad \text{in } 2D$$

Adhering to this criterion ensures that the numerical solution remains stable over time.

Boundary Conditions

The boundary conditions are shown in figure 3.

Dirichlet Condition – Northern Boundary

For the Dirichlet condition: $T(x, 6, t) = f(x)$

$T(x, 6, t)$: This shows the temperature (T) at the point $(x, 6, t)$.

Here, (x) and (t) are spatial and temporal coordinates, respectively, and 6 refers to the y -coordinate, indicating a point on the top boundary of the region under consideration.

$f(x)$: This is a specified function of (x) .

The Dirichlet condition is showing that the temperature at the top boundary ($y = 6$) at any given time (t) is equal to the value of the function $f(x)$ evaluated at the corresponding x -coordinate. (Farlow, 1993)

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

Eastern and Western Boundaries

Similarly to the Northern boundary, the Eastern and Western boundary conditions are defined in the brief as $T(0, y, t) = 0$ & $T(6, y, t) = 0$. The six in this being the furthest right edge of the system ($x = 6$).

Southern Boundary

As is with the Eastern and Western boundaries, at the Southern boundary the temperature is also maintained at zero, and this condition remains consistent throughout the simulation time (t).

Code Details and Explanation

The MATLAB code is designed to simulate 2D heat transfer in an aluminium plate using the Explicit Finite Difference Method.

Section 1

The user is prompted to input the grid spacing (h) for discretizing the plate. A validation message (msg) is prepared to handle invalid inputs. A while loop is used to check the user input against the required criteria. If the input does not meet these criteria, the validation message is displayed, and the prompt is repeated until a valid input is provided.

This section initializes the variables that define the thermal properties of aluminium. The dimensions of the plate are set according to the specifications, which in this case are 6m x 6m.

```

clc
clear
close all

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

% _____ Define plate dimensions _____
L = 6;             % Length of the plate in meters
W = 6;             % Width of the plate in meters

% _____ User input for grid spacing _____
h = input('Choose a grid spacing between 0-6: ');
msg = ("Grid spacing is not valid. Please enter a number that divides into 6: ");

% _____ Grid spacing _____
while (h>=L || h>=W || rem(L,h)~=0 || rem(W,h)~=0)
    h = input(msg);
end

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

% _____ Time step _____
dt = input("Input time step (seconds): "); %Input time step%
while dt>=h^2/(4*alpha)
    fprintf('Solution is unstable. Enter a smaller value than: %g\n', h^2/(4*alpha))
end

```

Figure 2 - Section 1 Code

In this section of code, the time step criteria are also input into the code, ensuring the user enters a value that is stable for the time step.

Section 2

```

% _____ Initialise temperature grid _____
T = zeros(NR, NC);

% _____ Set Initial Temperature Distribution _____
for c = 1:NC
    x = (c - 1) * h;           % x-coordinate
    % Using the function f(x) for initial temperature distribution
    if 0 <= x && x <= (2 / 3) * L % Apply piecewise function
        T(1, c) = 12.5 * x;      % First part of the piecewise function
    elseif (2 / 3) * L < x && x <= L
        T(1, c) = -25 * x + 150; % Second part of the piecewise function
    end

% _____ Set Boundary Conditions _____
T(:, 1) = 0;           % Left Boundary
T(:, end) = 0;         % Right Boundary
T(end, :) = 0;         % Bottom Boundary (no longer setting the top boundary)

% _____ Time loop for numerical simulation _____
t = 0;
while true
    T_old = T;
    for i = 2:NR - 1
        for j = 2:NC - 1
            % Finite difference equation for heat conduction
            T(i, j) = T_old(i, j) + alpha * dt / h ^ 2 * (T_old(i + 1, j) + T_old(i - 1, j) + T_old(i, j + 1) + T_old(i, j - 1) - 4 * T_old(i, j));
        end
    end

    % Check for Thermal Equilibrium
    if max(max(abs(T - T_old))) < 0.01 % Tolerance
        break;
    end

    t = t + dt;           % Update time
end

```

Figure 3 - Section 2 Code

This section of code sets the initial temperature distribution using specific conditions; it initiates a time loop for numerical simulation using the explicit finite difference method.

$T_{old} = T$ stores the current temperature T as the old temperature matrix for comparison later when the check for thermal equilibrium is carried out. The loop will repeat until the two temperatures are close enough together using a defined tolerance.

Section 3

```
% _____ Display results _____
if t<60
    fprintf('Time taken to reach Thermal Equilibrium is: %g seconds\n',t)
else if t>=60 && t<60^2
    fprintf('Time taken to reach Thermal Equilibrium is: %g minutes\n',t/60)
else if t>=60^2 && t<24*60^2
    fprintf('Time taken to reach Thermal Equilibrium is: %g hours\n',t/60^2)
else
    fprintf('Time taken to reach Thermal Equilibrium is: %g days\n',t/(24*60^2))
end
end
end
disp('Numerical Temperature Distribution at Thermal Equilibrium:');
disp(T);

% _____ Visualisation of the temperature distribution (Figure 1) _____
figure(1)
contourf(flipud(T))           % Using flipud to flip the matrix in the up/down
direction for correct orientation
colormap(jet)                 % Using 'jet' colormap for temperature visualization
colorbar                     % Adding colorbar for scale reference
clim([0 50]);                 % Set colorbar limits (Replace 'min_value' and
'max_value' with your desired range)
xlabel('X Coordinate (m)');
ylabel('Y Coordinate (m)');
title(['Numerical Temperature Distribution at t = ', num2str(t), ' seconds']);
```

Figure 4 - Section 3 Code

This next section displays the results of the time taken to reach thermal equilibrium conducted in the code in graphical form. It also generates the time that it took in numerical form, taking into consideration the best way to display the data based on the time taken, seconds, minutes etc.

Section 4

```

% _____Analytical Solution Calculation_____
% Calculate the analytical solution
num_terms = 100;           % Number of terms in the series to include
T_analytic = zeros(NR, NC);
for i = 1:NR
    for j = 1:NC
        x = (i - 1) * h;
        y = (j - 1) * h;
        T_analytic(i, j) = calculate_analytical_temperature(x, y, num_terms);
    end
end
T_analytic = rot90(T_analytic); % Rotate matrix 90 degrees counterclockwise for ✓
proper orientation

% Display the analytical solution in the command window
disp('Analytical Temperature Distribution at Thermal Equilibrium:');
disp(T_analytic);

% _____Calculate the difference between analytical and numerical solutions_____
difference = T - T_analytic; % Compute the difference between solutions
disp('Difference Between Numerical and Analytical solutions:');
disp(difference);

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

```

Figure 5 - Section 4 Code

This section calculates and displays the analytical temperature distribution at thermal equilibrium. It then displays the difference between numerical and analytical solutions. The visualisation includes contour plots and colour bars for better interpretation. The analytical solution is obtained by evaluating a series at each grid point using the ‘*calculate_analytical_temperature*’ function; this function calculates the analytical temperature distribution at a given point based off the analytical equation for ‘*T*’ seen below.

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

Figure 6 - Analytical Solution

Validation of the Code

The MATLAB-based numerical simulation approximates the heat distribution within the aluminum plate by solving complex mathematical equations that would be challenging to handle manually. This simulation incorporates various input parameters, including material properties, boundary conditions, and initial conditions, to model the heat transfer accurately.

To ensure the reliability of the numerical simulation, its results are often compared with analytical solutions derived from theoretical mathematical models. Analytical methods provide exact solutions by simplifying certain assumptions, as discussed in "The Application of Numerical Methods in Heat Transfer Problems" (n.d.).

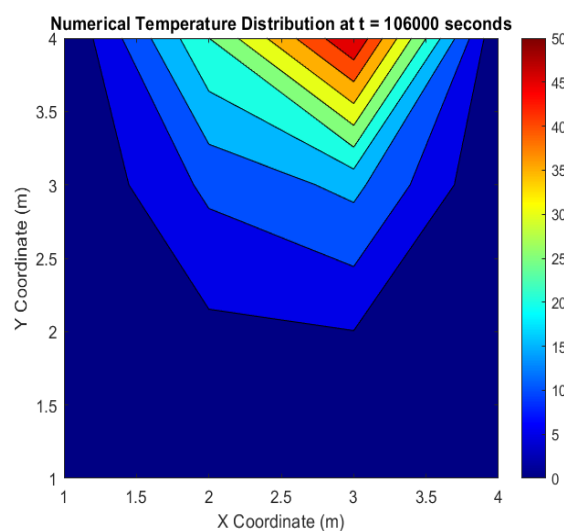


Figure 7 - Numerical Heat Map

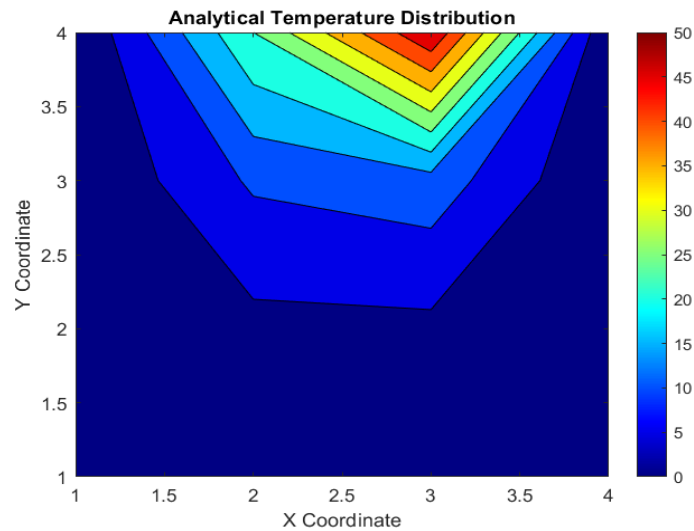


Figure 8 - Analytical Heat Map

In this particular case, a close match between the numerical and analytical heat distribution graphs serves as a strong indicator of the numerical simulation's accuracy. The precision of the numerical method is influenced by factors such as appropriate grid spacing and time step selection, as previously mentioned.

As well as the similarity in the visualisation of the graphical data, the difference can be seen in the matrix below (Figure 9), with a maximum deviation of 1.87°C.

Difference Between Numerical and Analytical solutions:

0	-0.0019	0.0034	-0.0040	0.2268	-0.0021	-0.0000
0	-0.0570	-0.0460	0.2818	1.8734	0.3388	-0.0000
0	-0.0600	-0.0091	0.3019	0.8272	0.3619	-0.0000
0	-0.0527	-0.0185	0.1497	0.3343	0.2024	-0.0000
0	-0.0443	-0.0369	0.0356	0.1090	0.0799	-0.0000
0	-0.0267	-0.0300	-0.0053	0.0226	0.0214	-0.0000
0	0	0	0	0	0	0

Figure 9 - Difference Between Numerical and Analytical Data Table

Sensitivity Analysis Of Code

In order to analyse the code it was possible to change certain parameters to see how the time taken to reach thermal equilibrium would be influenced. In all variations of the analysis the dimensions of the plate were kept the same.

Table 1 - Grid Spacing Variation

h – Grid Spacing	Tol – Minimum difference between two matrixes	dt – Time Step	Time Result
3	0.01	1000	1.31944 days
2	0.01	1000	1.22685 days
1.2	0.01	1000	1.21528 days
1	0.01	1000	1.25 days
0.75	0.01	1000	1.23843 days
0.6	0.01	1000	1.23843 days

Table 1 examines the adjustment of the grid spacing, subject to two validation conditions. The initial condition requires that 'h' must be less than both 'L' and 'W.' The second condition requires that the grid spacing evenly divides both 'L' and 'W.'

Table 2 - Tolerance Variation

h – Grid Spacing	Tol – Minimum Tolerance of Convergence	dt – Time Step	Time Result
2	0.8	1000	16.6667 mins
2	0.65	1000	1.11111 hours
2	0.5	1000	2.5 hours
2	0.2	1000	7.5 hours
2	0.01	1000	1.22685 days
2	0.001	1000	2.02546 days

Table 2 looks at the adjustment of the tolerancing allowed in the code between the first temperature and the next temperature in the sequence; many numerical methods for

solving differential equations use a convergence criteria or tolerance to determine when the solution has reached a sufficiently stable state (equilibrium).

The tolerance represents an acceptable level of change or error in the solution between successive iterations or time steps. (Morton, 2005)

Table 3 - Time Step Variation

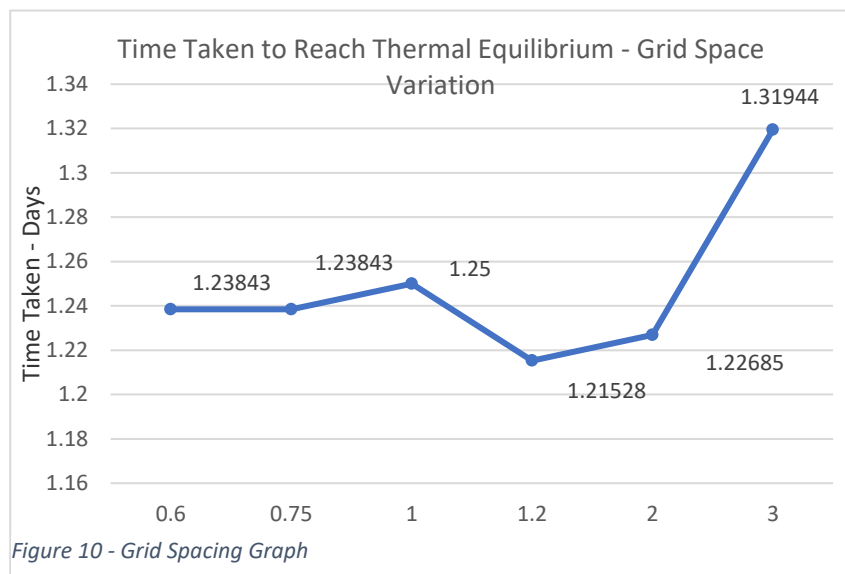
h – Grid Spacing	Tol – Minimum difference between two matrixes	dt – Time Step	Time Result
2	0.01	3600	1.625 days
2	0.01	2500	1.50463 days
2	0.01	1500	1.35417 days
2	0.01	600	1.0625 days
2	0.01	200	17 hours
2	0.01	100	12.111 hours

Table 3 shows the results from a variation in the time step (Δt). Smaller time steps generally lead to more accurate results, but they may require more computational resources.

Conversely, larger time steps can reduce computational costs but may compromise accuracy.

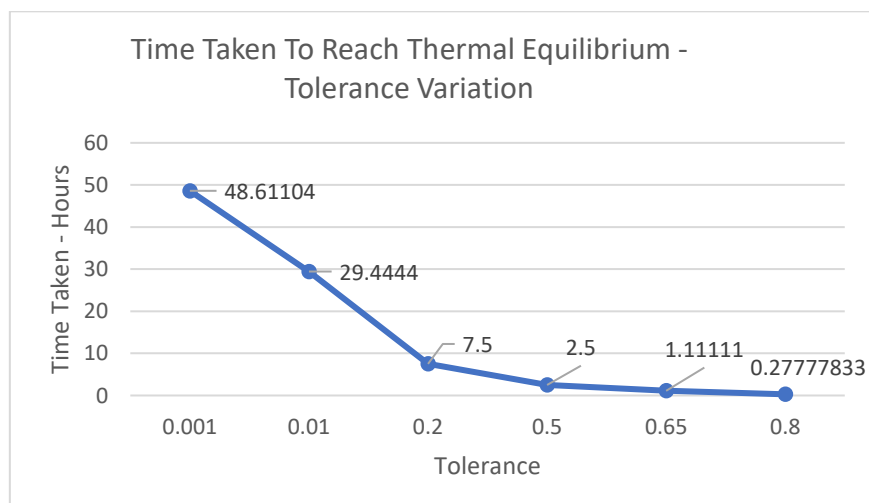
Influence of Grid Spacing Variation

A step size must then be chosen which is as large as possible to minimise the required number of steps, while remaining small enough to maintain the truncation error less than some tolerance. (Lowery, 1977) The round off error that occurs during the changes in step sizing are due to losses of significant figures when calculating. In Figure 8 it is shown that when the jump became larger the time taken to reach equilibrium was a lot longer as there is more error



Influence of Tolerance Variation

The influence of tolerance in terms of finite difference refers to a convergence criterion that the code must meet before producing a result. The convergence is checked by comparing the difference between the change in the solution with a tolerance predefined within the code. A higher tolerance allows more deviation between the two comparatives, typically yielding a much faster but less accurate result. (LeVeque, 2007) Figure 9 shows that as the tolerance increases, the time taken to reach thermal equilibrium is a lot faster as expected.



Influence of Time Step Variation

Key Model Limitations

Coding Limitations

MATLAB's limitations become apparent when it can only produce a figure for thermal equilibrium by comparing two successive temperature matrices. This approach does not capture the true thermal equilibrium state and introduces cumulative errors. Methods that are more physics-dependent could potentially yield more accurate results, as discussed by Anderson (1995).

Truncation Error

Truncation errors occur when infinite processes are approximated using a finite number of terms or grid points. In finite difference methods, these errors arise from discretizing continuous derivatives. By substituting derivatives with finite difference approximations, truncation errors are introduced, as highlighted by LeVeque (2007).

Round-Off Error

Due to the limited capacity of computers to store numbers, round-off errors are inevitable. For example, while π has an infinite number of digits, MATLAB can only store up to 16 digits. Although these errors may seem insignificant, they can accumulate over many calculations, resulting in significant deviations from expected values.

Complex Geometry

Modeling complex geometries with finite difference methods is challenging due to the need for generating an appropriate grid structure. This can become highly intricate and difficult for computational software to manage, often necessitating advanced techniques.

Additionally, achieving suitable accuracy levels may be hindered by computational limitations, as a finer mesh might be required. Finite difference methods also face stability constraints when dealing with irregular geometries, as noted by Ferziger (2002). These methods may lack the adaptability needed to accurately measure complex geometries.

Suggested Improvements and Future Developments

Ongoing collaboration between researchers in numerical methods, computational physics, and software engineering is essential to advance simulation capabilities and overcome the challenges associated with 2D thermal equilibrium modeling. Additionally, advancements in hardware technology and computational resources, such as the utilization of supercomputers, will play a crucial role in improving the accuracy of predictions and achieving the hypothesized outcomes

Conclusion

In conclusion, this report has explored the application of the finite difference method (FDM) for simulating 2D thermal distribution across an aluminium plate, highlighting both its strengths and weaknesses. Despite its limitations compared to other numerical methods, FDM has demonstrated its effectiveness in modelling the dissipation of temperature across a two-dimensional domain.

The accuracy of the simulation heavily relies on the selection of appropriate grid spacing and time step. These parameters are crucial for maintaining the stability and accuracy of the numerical solution. Through careful selection and validation, the FDM was able to produce results that closely matched the analytical solution. This close agreement between numerical and analytical results underscores the reliability of FDM in capturing the temperature distribution given initial conditions and boundary constraints.

One of the significant strengths of FDM is its simplicity and ease of implementation. The method provides a straightforward approach to discretizing and solving the heat equation, making it accessible for various applications. However, this simplicity comes with some drawbacks, particularly concerning the handling of complex geometries and the potential for accumulating numerical errors, such as round-off and truncation errors.

Round-off errors, due to the finite precision of computer arithmetic, and truncation errors, arising from the approximation of continuous derivatives by discrete differences, can accumulate over many iterations, potentially leading to deviations from the true solution. Despite these challenges, the method has shown that, with careful management of these errors, it can produce highly accurate results.

The study also highlighted the limitations of FDM in handling complex geometries and the necessity for advanced techniques to generate appropriate grid structures for such scenarios. These limitations suggest that while FDM is effective for simple, regular geometries, alternative methods like Finite Element Analysis (FEA) might be more suitable for more intricate domains.

Additionally, the report emphasizes the importance of continued research and development in this field. Future advancements in numerical methods, combined with improvements in computational power, will further enhance the accuracy and efficiency of thermal simulations. Collaborative efforts among researchers in numerical methods, computational physics, and software engineering will be vital in overcoming current challenges and pushing the boundaries of what can be achieved with numerical simulations.

Summary

In summary, the finite difference method provides a robust foundation for modelling 2D thermal distribution in materials like aluminium. While it is not without its limitations, its strengths in simplicity and reliability make it a valuable tool. Continued research and technological advancements will undoubtedly enhance its capabilities, paving the way for more accurate and efficient thermal simulations in the future.

Bibliography

Anand, L. (1985). Constitutive equations for hot-working of metals. *International Journal of Plasticity*, 1(3), pp.213-231. Available at:

(<https://www.sciencedirect.com/science/article/abs/pii/074964198590014X>)

[Accessed 20 June 2024].

Ascher, U.M. and Petzold, L.R. (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM. Available at:

<https://epubs.siam.org/doi/book/10.1137/1.9781611971392>) [Accessed 20 June 2024].

Caretto, L. (2014). *Numerical Differentiation*. California State University Northridge.

Available at: https://www.ecs.csun.edu/~lcaretto/comp451/451_NumDiff.pdf)

[Accessed 20 June 2024].

Corliss, G. and Chang, Y.F. (1982). Solving ordinary differential equations using Taylor series. *ACM Transactions on Mathematical Software (TOMS)*, 8(2), pp.114-144.

Available at: <https://dl.acm.org/doi/10.1145/355993.355994>) [Accessed 20 June 2024].

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

https://ramanujan.math.trinity.edu/rdaileda/teach/s12/m3357/lectures/twod_heat.pdf) [Accessed 20 June 2024].

Dehghan, M. (2007). The one-dimensional heat equation subject to a boundary integral specification. *Chaos, Solitons & Fractals*, 32(2), pp.661-675. Available at:

<https://www.sciencedirect.com/science/article/abs/pii/S0960077906005635>)

[Accessed 20 June 2024].

Engelhardt, W. (2016). Ohm's law and Maxwell's equations. *Ann. Fond. L. de Broglie*,

41, p.39. Available at: [<http://aflb.ensmp.fr/AFLB-411/aflb411m806.pdf>] [Accessed 20 June 2024].

Ferziger, J.H. and Perić, M. (2002). Computational Methods for Fluid Dynamics. Springer. Available at: <https://www.springer.com/gp/book/9783540420743>) [Accessed 20 June 2024].

Golub, G.H., Ortega, J.M. and Rheinboldt, W.C. (1996). Scientific Computing: An Introduction with Parallel Computing. Academic Press. Available at: <https://www.elsevier.com/books/scientific-computing/golub/978-0-12-289255-4>) [Accessed 20 June 2024].

Hecht, E. (2008). Energy conservation simplified. The Physics Teacher, 46(2), pp.77-80. Available at: <https://aapt.scitation.org/doi/10.1119/1.2834525>) [Accessed 20 June 2024].

Hsiao, G.C. and Saranen, J. (1993). Boundary integral solution of the two-dimensional heat equation. Mathematical Methods in the Applied Sciences, 16(2), pp.87-114. Available at: <https://onlinelibrary.wiley.com/doi/10.1002/mma.1670160202>) [Accessed 20 June 2024].

Kasner, E. and DeCicco, J. (1946). Geometry of the Fourier heat equation. Transactions of the American Mathematical Society, 60, pp.119-132. Available at: <https://www.ams.org/journals/tran/1946-060-01/S0002-9947-1946-0016680-2/S0002-9947-1946-0016680-2.pdf>) [Accessed 20 June 2024].

Katsikadelis, J.T. (2002). Boundary Elements: Theory and Applications. Elsevier. Available at: <https://www.elsevier.com/books/boundary-elements/katsikadelis/978-0-08-044107-6>) [Accessed 20 June 2024].

Karpatne, A., Atluri, G., Faghmous, J.H., Steinbach, M., Banerjee, A., Ganguly, A., ... and Kumar, V. (2017). Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10), pp.2318-2331. Available at: <https://ieeexplore.ieee.org/document/7933966> [Accessed 20 June 2024].

LeVeque, R.J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM. Available at: <https://epubs.siam.org/doi/book/10.1137/1.9780898717839> [Accessed 20 June 2024].

Nutting, P.G. (1921). A new general law of deformation. *Journal of the Franklin Institute*, 191(5), pp.679-685. Available at: (<https://www.sciencedirect.com/science/article/pii/S0016003221911041>) [Accessed 20 June 2024].

Patankar, S.V. (1980). *Numerical Heat Transfer and Fluid Flow*. CRC Press. Available at: <https://www.routledge.com/Numerical-Heat-Transfer-and-Fluid-Flow/Patankar/p/book/9780891165224> [Accessed 20 June 2024].

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press. Available at: <https://www.cambridge.org/core/books/numerical-recipes/BB13D5100330675ABCAA521F10329D95> [Accessed 20 June 2024].

Roylance, D. (2001). *Stress-strain curves*. Massachusetts Institute of Technology study, Cambridge. Available at: <http://web.mit.edu/course/3/3.11/www/modules/ss.pdf> [Accessed 20 June 2024].

Thompson, J.F., Warsi, Z.U.A. and Mastin, C.W. (2005). Numerical Grid Generation: Foundations and Applications. North. Available at:
<https://www.amazon.com/Numerical-Grid-Generation-Foundations-Applications/dp/0442317999> [Accessed 20 June 2024]

Appendix

```

clc
clear
close all

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

% Define plate dimensions
L = 6; % Length of the plate in meters
W = 6; % Width of the plate in meters

% User input for grid spacing
h = input('Choose a grid spacing between 0-6: ');
msg = ('Grid spacing is not valid. Please enter a number that divides into 6: ');

% Grid spacing
while (h>=L || h>=W || rem(L,h)~=0 || rem(W,h)~=0)
    h = input(msg);
end

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

% Time step
dt = input('Input time step (seconds): '); %Input time step%
while dt>=h^2/(4*alpha)
    fprintf('Solution is unstable. Enter a smaller value than: %g\n', h^2/(4*alpha))
    dt = input('Enter a suitable time step: ');
end

% Initialise temperature grid
T = zeros(NR, NC);

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

```

```

% _____ Set Boundary Conditions _____
T(:, 1) = 0;           % Left Boundary
T(:, end) = 0;         % Right Boundary
T(end, :) = 0;         % Bottom Boundary (no longer setting the top boundary)

% _____ Time loop for numerical simulation _____
t = 0;
while true
    T_old = T;
    for i = 2:NR - 1
        for j = 2:NC - 1
            % Finite difference equation for heat conduction
            T(i, j) = T_old(i, j) + alpha * dt / h ^ 2 * (T_old(i + 1, j) + T_old(i - 1, j) + T_old(i, j + 1) + T_old(i, j - 1) - 4 * T_old(i, j));
        end
    end

    % Check for Thermal Equilibrium
    if max(max(abs(T - T_old))) < 0.01 % Tolerance
        break;
    end

    t = t + dt;           % Update time
end

% _____ Display results _____
if t < 60
    fprintf('Time taken to reach Thermal Equilibrium is: %g seconds\n', t)
else if t >= 60 && t < 60^2
    fprintf('Time taken to reach Thermal Equilibrium is: %g minutes\n', t/60)
else if t >= 60^2 && t < 24*60^2
    fprintf('Time taken to reach Thermal Equilibrium is: %g hours\n', t/60^2)
else
    fprintf('Time taken to reach Thermal Equilibrium is: %g days\n', t/(24*60^2))
end
end
end
disp('Numerical Temperature Distribution at Thermal Equilibrium:');
disp(T);

% _____ Visualisation of the temperature distribution (Figure 1) _____
figure(1)
contourf(flipud(T))      % Using flipud to flip the matrix in the up/down
direction for correct orientation
colormap(jet)           % Using 'jet' colormap for temperature visualization
colorbar                % Adding colorbar for scale reference
clim([0 50]);           % Set colorbar limits (Replace 'min_value' and

```

```

'max_value' with your desired range)
xlabel('X Coordinate (m)');
ylabel('Y Coordinate (m)');
title(['Numerical Temperature Distribution at t = ', num2str(t), ' seconds']);

% _____ Analytical Solution Calculation _____
% Calculate the analytical solution
num_terms = 100;           % Number of terms in the series to include
T_analytic = zeros(NR, NC);
for i = 1:NR
    for j = 1:NC
        x = (i - 1) * h;
        y = (j - 1) * h;
        T_analytic(i, j) = calculate_analytical_temperature(x, y, num_terms);
    end
end
T_analytic = rot90(T_analytic); % Rotate matrix 90 degrees counterclockwise for
proper orientation

% Display the analytical solution in the command window
disp('Analytical Temperature Distribution at Thermal Equilibrium:');
disp(T_analytic);

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

% Visualisation of the analytical solution (Figure 2)
figure(2)
contourf(flip(T_analytic)) % Rotate by 90 degrees (clockwise) and then flip
vertically
colormap(jet)
colorbar % Keep the colorbar on the right side
clim([0 50]); % Set colorbar limits (Replace 'min_value' and
'max_value' with your desired range)
xlabel('X Coordinate'); % Adjust the label accordingly
ylabel('Y Coordinate'); % Adjust the label accordingly
title('Analytical Temperature Distribution');

% Visualisation of the difference (Figure 3)
figure(3)
contourf(flip(difference)) % Flip matrix vertically for correct
orientation
colormap(jet) % Use 'jet' colormap for clarity
colorbar % Add colorbar to denote value ranges
clim([-0.005 2]); % Set colorbar limits (Replace 'min_value'
and 'max value' with your desired range)

ylabel('Y Coordinate Length');
title('Difference Between Numerical and Analytical Solutions');

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

```