

2D Transient Heat Equation Solved with the Explicit Finite Difference Method

Computational Methods – ACMA6012

Tyler Powell – 2318809

12/04/2026

Approx. 3100 words

Contents

Table of Figures	4
1. Introduction	5
1.1. Background and Context.....	5
1.2. Aim	5
1.3. Objectives	5
1.4. Methodology Overview	5
1.5. Structure of the Report	6
2. Governing Equation	7
2.1. Heat Conduction Equation.....	7
2.2. Material Properties	7
2.3. Assumptions	8
3. Finite Difference Method	9
3.1. Finite Difference Approximation	9
3.2. Discretisation of the Domain.....	9
3.3. Explicit Finite Difference Formulation	9
3.4. Stability Criterion.....	10
4. Boundary Conditions.....	11
4.1. Initial Condition	11
4.2. Boundary Conditions	11
4.3. Justification.....	11
5. Numerical Implementation.....	12
5.1. Overview of Implementation	12
5.2. Input Parameters and Material Properties	12
5.3. Grid Generation and Validation	12
5.4. Time Step Selection and Stability Enforcement	13
5.5. Initial and Boundary Condition Implementation	14
5.6. Iterative Solution Procedure	14
5.7. Convergence Criterion	15
5.8. Post-Processing and Visualisation.....	15
6. Validation of Numerical Model.....	17
6.1. Validation Approach	17
6.2. Validation Case Definition.....	17
6.3. Comparison of Temperature Fields.....	17

2D Transient Heat Equation Solved with the Explicit Difference Method

6.4.	Error Distribution	18
6.5.	Quantitative Error Analysis	19
6.6.	Convergence Behaviour	19
6.7.	Validation Conclusion	20
7.	Sensitivity Analysis of Numerical Parameters	21
7.1.	Purpose of Sensitivity Analysis	21
7.2.	Effect of Grid Spacing	21
7.3.	Quantitative Effect on Accuracy	23
7.4.	Interpretation of Error Behaviour	23
7.5.	Summary of Findings	23
8.	Limitations of the Numerical Model	24
8.1.	Numerical Accuracy Limitations	24
8.2.	Stability and Efficiency Constraints	24
8.3.	Convergence Approximation	24
8.4.	Modelling Assumptions	24
8.5.	Summary	24
9.	Suggested Improvements and Future Developments	25
9.1.	Numerical Method Efficiency	25
9.2.	Spatial Resolution Strategy	25
9.3.	Boundary Region Accuracy	25
9.4.	Convergence Definition	25
9.5.	Model Applicability	25
9.6.	Summary	25
10.	Conclusion	26
	References	27
	Appendix	28

Table of Figures

Figure 1: Initial and boundary conditions schematic	7
Figure 2: Finite difference node layout.....	10
Figure 3: Material properties and user inputs MATLAB code	12
Figure 4: Grid spacing input and validation MATLAB code.....	13
Figure 5: Time step input and stability check MATLAB code	13
Figure 6: Grid creation and boundary condition implementation MATLAB code	14
Figure 7: EDFM MATLAB code	15
Figure 8: Numerical temperature distribution	16
Figure 9: Analytical and numerical temperature distribution	18
Figure 10: Error difference between analytical and numerical temperature distribution.....	19
Figure 11: Numerical solution at 0.1 grid spacing.....	21
Figure 12: Numerical solution at 0.25 grid spacing.....	22
Figure 13: Numerical solution at 0.5 grid spacing.....	22
Figure 14: Mean and max error at variable grid spacing	23

1. Introduction

1.1. Background and Context

Heat conduction describes the transfer of thermal energy within a solid due to temperature gradients (Incropera, F.P.; DeWitt, D.P.; Bergman, T.L.; Lavine, A.S., 2007). Accurate prediction of temperature distributions is essential in engineering applications such as thermal management of electronic systems, structural components exposed to heat, and energy systems.

Analytical solutions to heat conduction problems are limited to simple geometries and boundary conditions (Carslaw, H.S.; Jaeger, J.C., 1959). For more general cases, numerical methods are required. This study considers transient heat conduction in a two-dimensional aluminium plate, providing a controlled case where a numerical solution can be validated against an analytical solution.

1.2. Aim

To develop and validate a numerical model that predicts the transient temperature distribution within a two-dimensional aluminium plate.

1.3. Objectives

- Formulate the governing heat conduction equation
- Discretise the spatial and temporal domains
- Implement the Explicit Finite Difference Method (EFDM) in MATLAB
- Apply the specified initial and boundary conditions
- Enforce the numerical stability criterion
- Compare numerical results with the analytical solution

1.4. Methodology Overview

The governing heat equation is solved numerically using the Explicit Finite Difference Method (EFDM). The spatial domain is discretised into a uniform grid, and time is advanced in discrete steps. At each time step, the temperature at a node is updated using values from its neighbouring nodes.

The EFDM is selected due to its simplicity and direct physical interpretation, where heat transfer between adjacent nodes is explicitly represented. This makes the method well suited to structured grids and allows clear analysis of how discretisation parameters, such as grid spacing and time step size, influence accuracy and stability.

Although implicit methods offer improved stability, the EFDM is sufficient for this problem and provides a transparent framework for demonstrating the numerical behaviour of the heat equation. The numerical solution is validated by comparison with an analytical solution corresponding to the same boundary conditions.

1.5. Structure of the Report

Chapter 2 presents the governing heat conduction equation and assumptions. Chapter 3 describes the finite difference method and numerical formulation. Chapter 4 defines the initial and boundary conditions. Chapter 5 outlines the MATLAB implementation. Chapter 6 presents validation against the analytical solution. Chapter 7 provides sensitivity analysis, followed by limitations, improvements, and conclusions in Chapters 8–10.

2. Governing Equation

The problem domain and boundary conditions are illustrated in Figure 1.

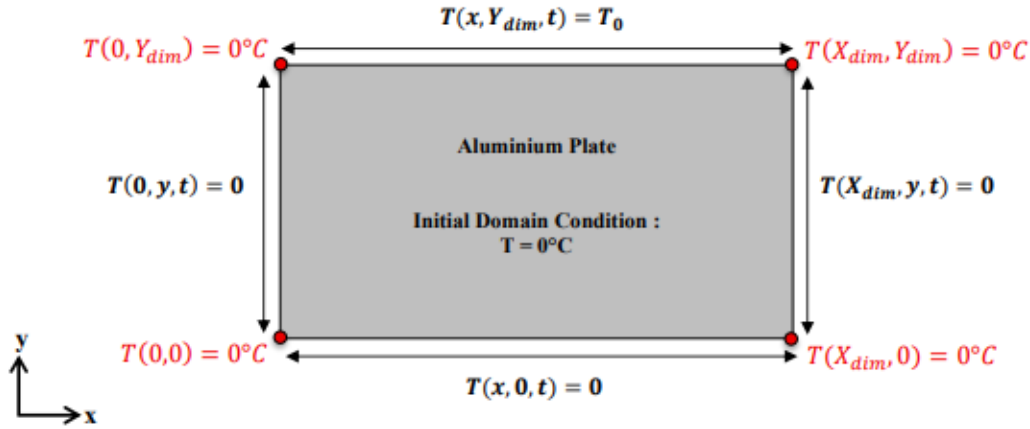


Figure 1: Initial and boundary conditions schematic

2.1. Heat Conduction Equation

The temperature distribution within the aluminium plate is governed by the two-dimensional transient heat conduction equation (Ozisik, M.N., 1993):

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

where T is temperature and α is the thermal diffusivity.

This equation describes the diffusion of heat within the material, where the rate of temperature change is driven by spatial temperature gradients (Carslaw, H.S.; Jaeger, J.C., 1959).

2.2. Material Properties

The thermal diffusivity is defined as:

$$\alpha = \frac{k}{\rho c_p}$$

where:

- k is thermal conductivity (W/m·K)
- ρ is density (kg/m³)
- c_p is specific heat capacity (J/kg·K)

For aluminium, the values used in this study are:

- $k = 143$ W/m·K
- $\rho = 2.8 \times 10^3$ kg/m³
- $c_p = 795$ J/kg·K

2D Transient Heat Equation Solved with the Explicit Difference Method

These properties define the thermal diffusivity, which governs the rate of heat diffusion.

2.3. Assumptions

The model is based on the following assumptions:

- Material properties are uniform and constant
- Heat transfer occurs only by conduction
- No internal heat generation
- Boundary conditions are applied only at the plate edges

These assumptions simplify the problem while retaining the dominant physics of heat diffusion.

3. Finite Difference Method

3.1. Finite Difference Approximation

The governing equation contains continuous derivatives, which are approximated numerically using finite difference expressions derived from Taylor series expansions (LeVeque, R.J., 2007). This converts the partial differential equation into a set of algebraic equations that can be solved computationally. This approximation introduces truncation error, which depends on the grid spacing and time step.

3.2. Discretisation of the Domain

The spatial domain is discretised into a uniform grid with spacing h in both directions. Time is discretised into steps of size Δt . Each node is represented by indices (i, j) , with time levels denoted by n . A uniform grid is used to simplify implementation and ensure consistent numerical behaviour across the domain.

3.3. Explicit Finite Difference Formulation

Applying finite difference approximations to the governing equation results in the explicit update equation (Croft, D.R.; Lilley, D.G., 1977):

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

where the Fourier number r is given by:

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

This formulation updates the temperature at each node using values from its four neighbouring nodes at the previous time step. The method is explicit, as the new temperature is computed directly without solving a system of equations. The finite difference stencil used in the numerical formulation is shown in Figure 2.

2D Transient Heat Equation Solved with the Explicit Difference Method

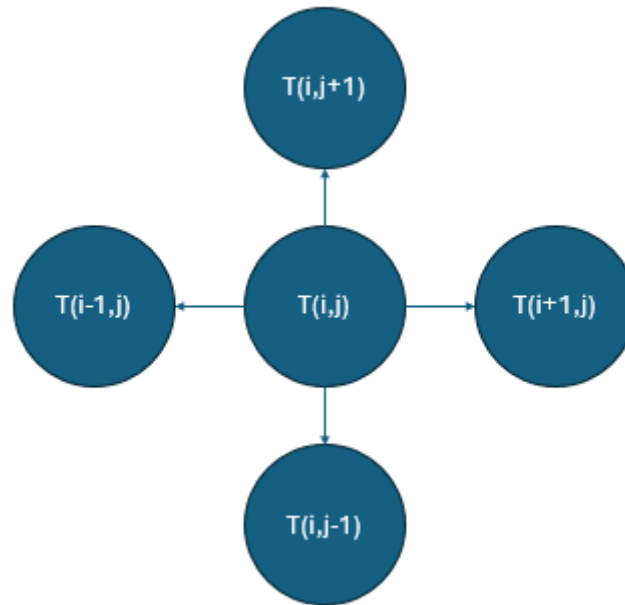


Figure 2: Finite difference node layout

3.4. Stability Criterion

The stability of the explicit finite difference method is governed by the Fourier number. For a two-dimensional problem, stability requires (Crank, J.; Nicolson, P., 1947) (Mojumder, M.S.H.; Rahman, M.M.; Islam, M.S., 2023):

$$r \leq 0.25$$

This condition ensures that numerical errors do not grow during time integration. If violated, the solution becomes unstable and non-physical.

4. Boundary Conditions

4.1. Initial Condition

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

The plate is initially at a uniform temperature of 0°C.

4.2. Boundary Conditions

The following Dirichlet boundary conditions are applied:

- West boundary: $T(0, y, t) = 0$
- East boundary: $T(X_{dim}, y, t) = 0$
- South boundary: $T(x, 0, t) = 0$
- North boundary (excluding corners): $T(x, Y_{dim}, t) = T_0$
- Corner nodes: $T = 0$

This configuration introduces a temperature gradient driving heat into the plate from the top boundary.

4.3. Justification

These boundary conditions are chosen to match the analytical solution provided in the assignment brief. Analytical solutions to the heat equation exist only for specific boundary configurations; therefore, aligning the numerical model with these conditions allows direct validation (Carslaw, H.S.; Jaeger, J.C., 1959). Setting the corner nodes to zero ensures consistency between boundary definitions and avoids discontinuities at the edges of the domain.

5. Numerical Implementation

5.1. Overview of Implementation

The numerical model is implemented in MATLAB using the Explicit Finite Difference Method (EFDM). The solution procedure follows a structured sequence consisting of input definition, grid generation, stability verification, iterative solution of the governing equation, and post-processing.

Key sections of the MATLAB code are presented throughout this chapter to demonstrate how the theoretical formulation is translated into a working computational model.

5.2. Input Parameters and Material Properties

The implementation begins by defining the thermal properties of aluminium, which are used to calculate the thermal diffusivity. The plate dimensions and boundary temperature are then specified by the user to allow flexibility in defining the problem domain. As shown in Figure 3, these inputs are combined with the material properties to fully define the physical system prior to discretisation.

```
%%      Material Properties
k = 143;
rho = 2.8e3;
cp = 795;
alpha = k/(rho*cp);

fprintf('Thermal diffusivity alpha = %.6e m^2/s\n\n', alpha);

%%      User Inputs
Xdim = input('Enter plate width Xdim (m): ');
Ydim = input('Enter plate height Ydim (m): ');
T0 = input('Enter top boundary temperature T0 (C): ');
```

Figure 3: Material properties and user inputs MATLAB code

The grid spacing is also provided as a user input. A validation step ensures that the selected spacing divides the domain dimensions exactly, preventing inconsistencies in node placement and ensuring a uniform computational grid. This is shown in Figure 4.

5.3. Grid Generation and Validation

Once the grid spacing has been defined, the spatial domain is discretised into a structured grid. The number of nodes in each direction is determined from the ratio of the domain dimensions to the grid spacing.

To ensure a consistent discretisation, a validation procedure is implemented using an iterative loop, as shown in Figure 4, which checks that the grid spacing produces integer numbers of intervals in both directions. This step is necessary to avoid partial cells and maintain alignment of the computational nodes.

2D Transient Heat Equation Solved with the Explicit Difference Method

```
%%      Grid Setup
while true
    h = input('Enter grid spacing h (m): ');
    Nx_test = Xdim / h;
    Ny_test = Ydim / h;

    if h > 0 && abs(Nx_test-round(Nx_test))<1e-12 && abs(Ny_test-round(Ny_test))<1e-12
        Nx = round(Nx_test);
        Ny = round(Ny_test);
        nx = Nx + 1;
        ny = Ny + 1;
        break;
    else
        fprintf('Invalid h. Must divide dimensions evenly.\n');
    end
end

fprintf('Grid: %d x %d nodes\n\n', nx, ny);
```

Figure 4: Grid spacing input and validation MATLAB code

5.4. Time Step Selection and Stability Enforcement

The time step is defined by the user and is checked against the stability condition for the explicit scheme:

$$r = \frac{\alpha \Delta t}{h^2} \leq 0.25$$

As shown in Figure 5, the value of the Fourier number r is calculated within the code, and the user is prompted to re-enter the time step if the stability condition is not satisfied.

```
%%      Time Step
dt_max = h^2/(4*alpha);

while true
    dt = input('Enter time step dt (s): ');

    if dt <= 0
        fprintf('dt must be positive.\n');
        continue;
    end

    r = alpha*dt/h^2;

    fprintf('r = %.4f (must be <= 0.25)\n', r);
    fprintf('Maximum stable dt = %.3e\n', dt_max);

    if r <= 0.25
        fprintf('Time step accepted.\n\n');
        break;
    else
        fprintf('Time step too large → UNSTABLE. Try again.\n\n');
    end
end
```

Figure 5: Time step input and stability check MATLAB code

2D Transient Heat Equation Solved with the Explicit Difference Method

This validation is essential because the EFDM is conditionally stable. If the time step is too large, numerical errors will grow with each iteration, leading to divergence and non-physical results.

5.5. Initial and Boundary Condition Implementation

The temperature field is initialised to zero across all nodes, representing the initial condition of the plate. Boundary conditions are then applied such that the west, east, and south boundaries are fixed at zero temperature, while the north boundary is set to a constant temperature excluding the corner nodes.

The implementation of these conditions is shown in Figure 6, where the top row is first reset before applying the interior boundary temperature. This ensures that the corner nodes remain consistent with the adjacent boundary conditions.

```
%%      Create Grid ---
x = linspace(0, Xdim, nx);
y = linspace(0, Ydim, ny);
[X, Y] = meshgrid(x, y);

%%      Initialise Temperature
T = zeros(ny, nx);
T_new = T;

% Boundary conditions
T(1,:) = 0;           % South
T(:,1) = 0;           % West
T(:,end) = 0;         % East
T(end,:) = 0;         % Reset top row
T(end,2:end-1) = T0;  % Top interior
```

Figure 6: Grid creation and boundary condition implementation MATLAB code

These boundary conditions are reapplied at each iteration to maintain consistency throughout the simulation.

5.6. Iterative Solution Procedure

The temperature field is updated iteratively using the explicit finite difference formulation. At each time step, the temperature at each interior node is calculated using the values of its four neighbouring nodes from the previous time step.

This update is implemented within nested loops, as shown in Figure 7, where the new temperature is computed directly using the finite difference equation. This reflects the physical process of heat diffusion, where energy is transferred between neighbouring points in the domain.

2D Transient Heat Equation Solved with the Explicit Difference Method

```
%%      EFDM Solver
tol = 1e-6;
max_iter = 5e5;

for it = 1:max_iter

    for j = 2:ny-1
        for i = 2:nx-1
            T_new(j,i) = T(j,i) + r*( ...
                T(j+1,i)+T(j-1,i)+T(j,i+1)+T(j,i-1)-4*T(j,i));
        end
    end

    % Reapply BCs
    T_new(1,:) = 0;
    T_new(:,1) = 0;
    T_new(:,end) = 0;
    T_new(end,:) = 0;
    T_new(end,2:end-1) = T0;

    if max(abs(T_new(:)-T(:))) < tol
        fprintf('Converged in %d iterations\n', it);
        break;
    end

    T = T_new;
end

if it == max_iter
    fprintf('Warning: Max iterations reached\n');
end
```

Figure 7: EFDM MATLAB code

The explicit nature of the method allows the temperature field to be updated without solving a system of equations, simplifying implementation.

5.7. Convergence Criterion

The simulation proceeds iteratively until the maximum difference between successive temperature fields falls below a specified tolerance. This condition is used to approximate thermal equilibrium.

As shown in Figure 7, the convergence check evaluates the maximum absolute change across all nodes. When this value becomes sufficiently small, the solution is assumed to have reached steady state.

Although this does not represent an exact physical equilibrium, it provides a practical and computationally efficient stopping criterion.

5.8. Post-Processing and Visualisation

After convergence, the temperature distribution is visualised using contour plots to provide a clear representation of heat diffusion across the plate. A consistent colour scale is applied to ensure that temperature variations are accurately represented.

2D Transient Heat Equation Solved with the Explicit Difference Method

The resulting numerical solution is shown in Figure 8, illustrating the temperature gradient from the heated boundary into the interior of the domain.

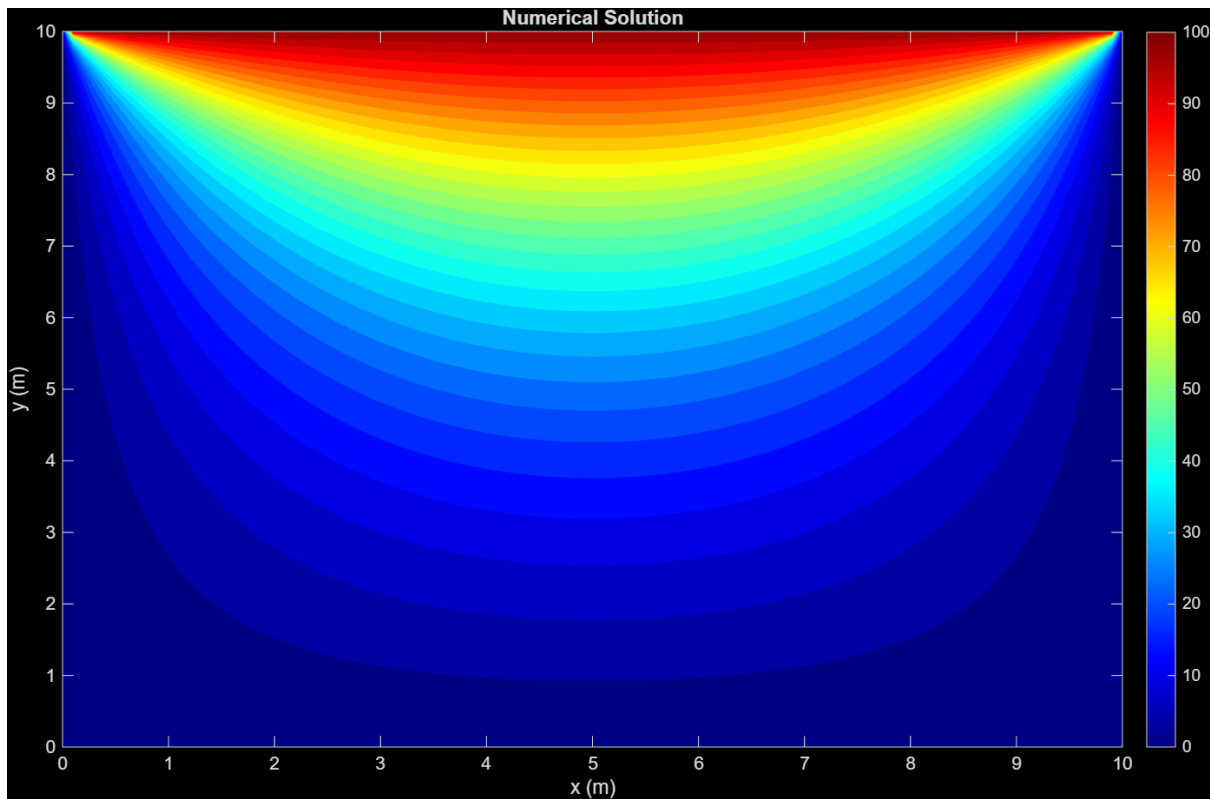


Figure 8: Numerical temperature distribution

6. Validation of Numerical Model

6.1. Validation Approach

The numerical model is validated by comparing the steady-state EFDM solution with the analytical solution for the same boundary conditions (Carslaw, H.S.; Jaeger, J.C., 1959). Agreement is assessed using both visual comparison of temperature fields and quantitative error metrics.

6.2. Validation Case Definition

The validation was performed for a square aluminium plate with dimensions $X_{dim} = 10$ m and $Y_{dim} = 10$ m. A uniform grid spacing of $h = 0.1$ m and time step $\Delta t = 5$ s were used.

The north boundary temperature was set to $T_0 = 100^\circ\text{C}$, while the west, east, and south boundaries were maintained at 0°C , with the corner nodes also fixed at 0°C . The chosen time step satisfies the stability condition of the explicit scheme.

6.3. Comparison of Temperature Fields

The analytical and numerical temperature distributions are shown in Figure 9, respectively. Both solutions exhibit identical overall behaviour, with temperature decreasing smoothly from the heated north boundary into the interior of the plate.

2D Transient Heat Equation Solved with the Explicit Difference Method

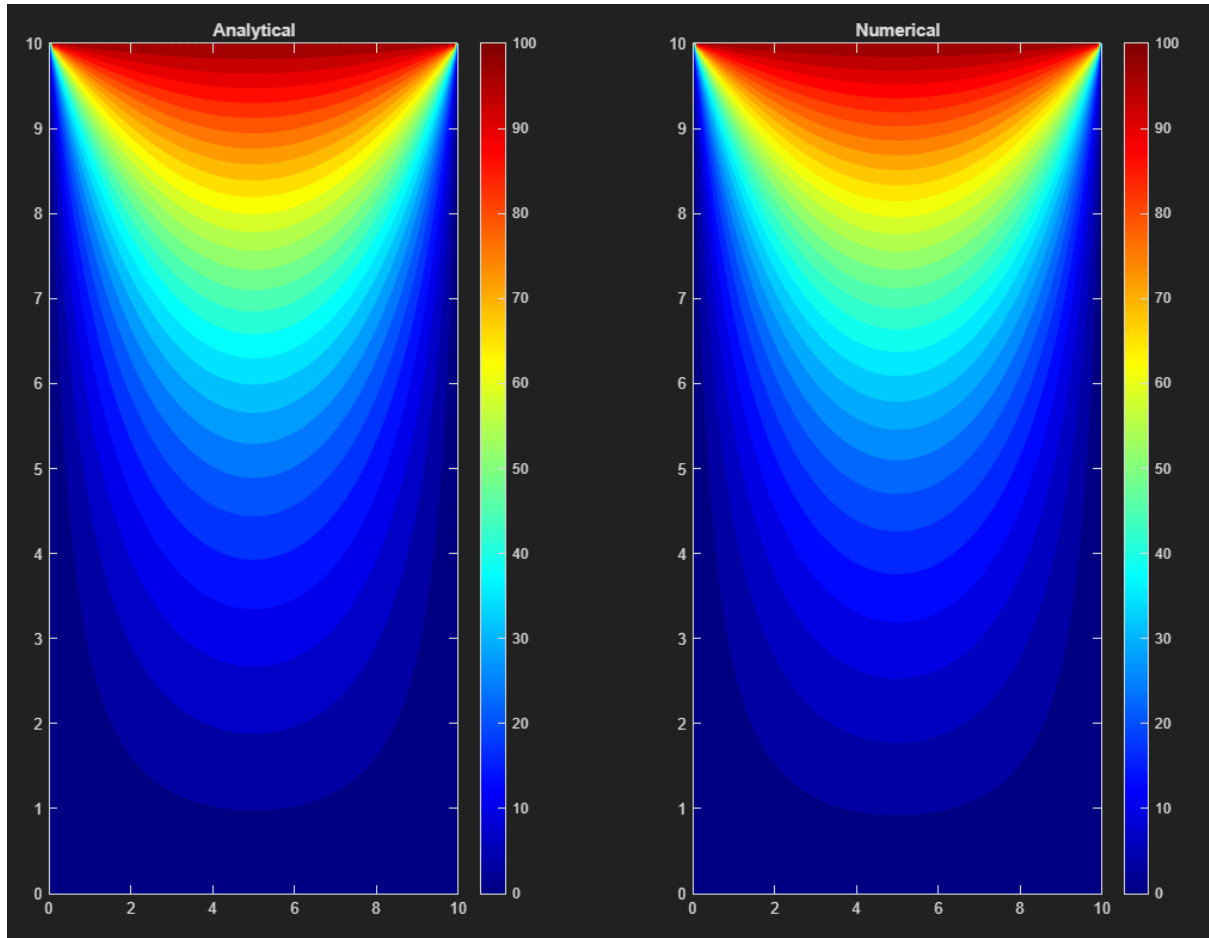


Figure 9: Analytical and numerical temperature distribution

Symmetry about the vertical centreline is observed in both cases, and the influence of the zero-temperature boundaries is clearly represented. The close visual agreement across the majority of the domain indicates that the numerical model accurately captures the physical heat conduction process.

6.4. Error Distribution

The spatial distribution of absolute error is shown in Figure 10. The error is minimal across most of the plate, with values close to zero throughout the interior region.

2D Transient Heat Equation Solved with the Explicit Difference Method

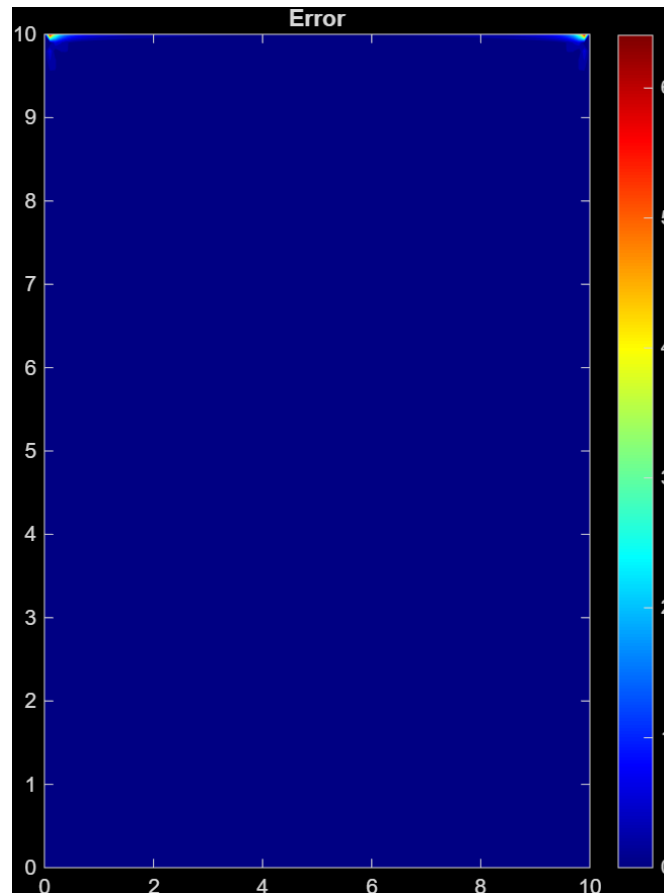


Figure 10: Error difference between analytical and numerical temperature distribution

However, localised regions of higher error are observed at the top corners of the domain. These areas correspond to locations where the boundary conditions introduce sharp temperature gradients and discontinuities.

The numerical method approximates spatial derivatives using neighbouring nodes, and this approximation becomes less accurate in regions where the temperature field changes rapidly over short distances. As a result, the largest discrepancies occur at these corner locations.

6.5. Quantitative Error Analysis

The following error metrics were obtained:

- Mean absolute error: $1.486 \times 10^{-2} \text{ } ^\circ\text{C}$
- Maximum absolute error: $6.62 \text{ } ^\circ\text{C}$

The low mean error confirms that the numerical solution is highly accurate across the majority of the domain. The relatively large maximum error reflects the localised discrepancies at the top corners, rather than a global inaccuracy in the model.

6.6. Convergence Behaviour

The numerical solution converged after 123,839 iterations, corresponding to a computed time to equilibrium of 619,195 s. The high number of iterations is a consequence of the refined grid

and the explicit time-stepping approach, which advances the solution gradually toward steady state.

6.7. Validation Conclusion

The EFDM model demonstrates strong agreement with the analytical solution, both visually and quantitatively. The very low mean error indicates that the numerical method accurately represents the temperature field across the plate.

The presence of higher error at the top corners is expected due to steep temperature gradients and the limitations of finite difference approximations in such regions. This does not significantly affect the overall solution accuracy.

Overall, the results confirm that the numerical implementation is correct, stable, and convergent.

7. Sensitivity Analysis of Numerical Parameters

7.1. Purpose of Sensitivity Analysis

A sensitivity analysis is conducted to evaluate how the numerical solution responds to variations in key parameters, namely grid spacing, time step, and convergence tolerance. This analysis is essential for assessing the robustness and reliability of the model, as well as understanding the trade-offs between accuracy and computational cost.

7.2. Effect of Grid Spacing

The influence of spatial discretisation is investigated by comparing solutions obtained using three different grid spacings:

- $h = 0.1$ m
- $h = 0.25$ m
- $h = 0.5$ m

The numerical temperature distributions for 0.1, 0.25 and 0.5 grid spacings are shown in Figure 11, Figure 12 and Figure 13.

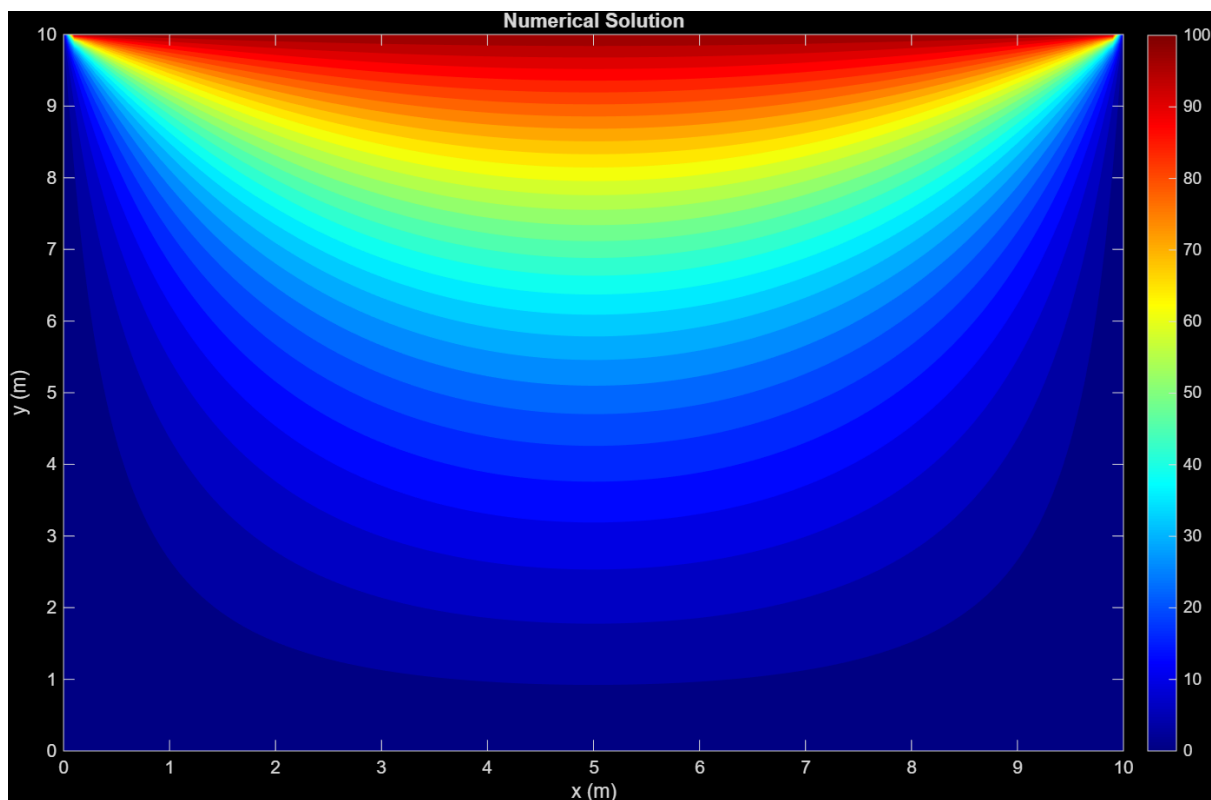


Figure 11: Numerical solution at 0.1 grid spacing

2D Transient Heat Equation Solved with the
Explicit Difference Method

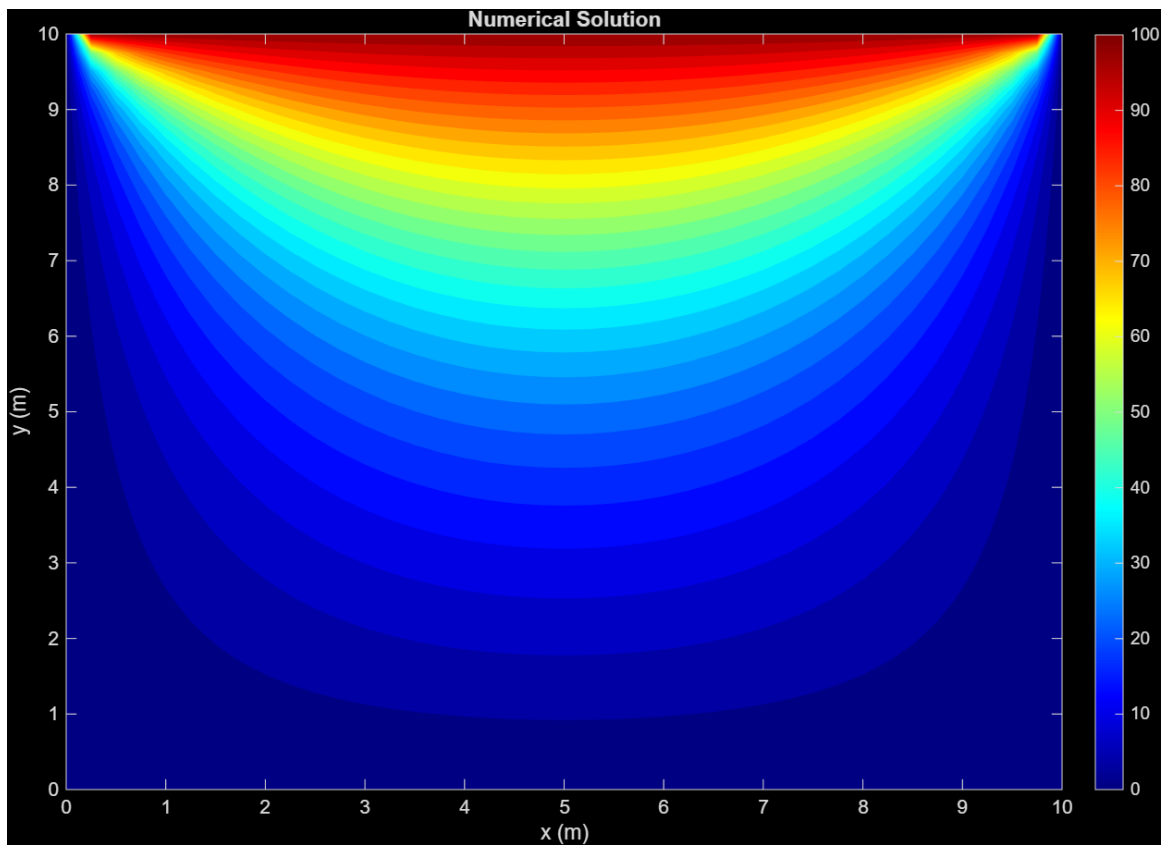


Figure 12: Numerical solution at 0.25 grid spacing

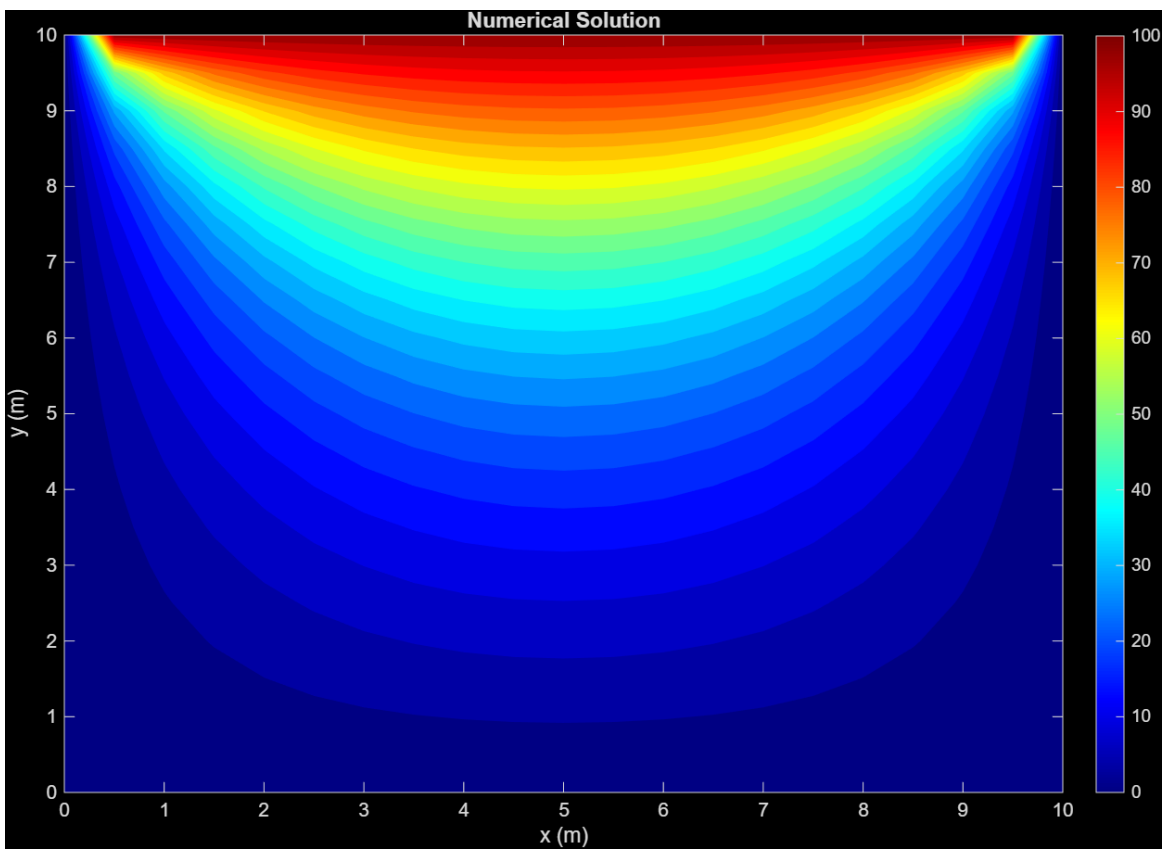


Figure 13: Numerical solution at 0.5 grid spacing

2D Transient Heat Equation Solved with the Explicit Difference Method

Visual comparison shows that all cases produce the same overall temperature profile, with heat diffusing from the north boundary into the plate. However, the solution becomes progressively smoother and more refined as the grid spacing decreases.

The finer grid ($h = 0.1$) produced the most accurate representation of the temperature field, particularly in regions of high curvature near the top boundary.

7.3. Quantitative Effect on Accuracy

The effect of grid refinement on numerical accuracy is summarised below:

Grid Spacing (m)	Mean Error (°C)	Max Error (°C)
0.5	6.161×10^{-2}	1.355
0.25	2.269×10^{-2}	1.355
0.1	1.486×10^{-2}	6.62

Figure 14: Mean and max error at variable grid spacing

The results show a clear reduction in mean error as grid spacing decreases, confirming that spatial refinement improves overall solution accuracy. This behaviour is consistent with finite difference theory, where smaller grid spacing reduces truncation error in the approximation of second derivatives (LeVeque, R.J., 2007).

However, the maximum error does not follow the same trend. While it remains relatively constant between $h = 0.5$ and $h = 0.25$, it increases significantly for $h = 0.1$. This is due to the error being highly localised at the top corner nodes, where steep temperature gradients and boundary discontinuities occur.

7.4. Interpretation of Error Behaviour

The reduction in mean error demonstrates that the numerical solution is converging towards the analytical solution as the grid is refined. This indicates that the EFDM implementation is consistent and correctly formulated.

The increase in maximum error for the finest grid highlights an important limitation of using a single global error metric. Since the error is concentrated in a very small region of the domain (the top corners), it does not significantly affect the overall accuracy of the solution.

This behaviour is also influenced by the analytical solution itself, which is approximated using a finite number of series terms. Near boundary discontinuities, this approximation can introduce additional localised error.

7.5. Summary of Findings

The sensitivity analysis demonstrates that grid spacing is the dominant factor influencing numerical accuracy. Reducing h leads to improved agreement with the analytical solution, confirming convergence of the method.

While localised errors remain near boundary discontinuities, the overall solution is highly accurate for sufficiently refined grids. This highlights the importance of selecting an appropriate grid resolution to balance accuracy and computational efficiency. The results demonstrate numerical convergence of the EFDM solution with respect to grid refinement.

8. Limitations of the Numerical Model

8.1. Numerical Accuracy Limitations

The finite difference method introduces truncation error through the approximation of derivatives using discrete grid points (LeVeque, R.J., 2007). As shown in section 7, this error is reduced with decreasing grid spacing but cannot be eliminated entirely without significantly increasing computational cost.

Additionally, the results in section 6 showed that error is not uniformly distributed, with the largest discrepancies occurring at the top corners of the domain. These localised errors arise from steep temperature gradients and boundary discontinuities, where the finite difference approximation is less accurate.

8.2. Stability and Efficiency Constraints

The explicit formulation is conditionally stable and requires the Fourier number to satisfy $r \leq 0.25$ (Mojumder, M.S.H.; Rahman, M.M.; Islam, M.S., 2023). This restricts the time step size and leads to a large number of iterations to reach convergence.

Although the model remains stable, the results showed that over 120,000 iterations were required even for refined grids. This highlights a key limitation of the EFDM, where computational efficiency is reduced as accuracy requirements increase.

8.3. Convergence Approximation

Thermal equilibrium is determined using a numerical tolerance based on the maximum change in temperature between iterations. This provides a practical stopping condition but does not guarantee that the solution has reached a true steady state. As a result, the final solution represents an approximation of equilibrium, dependent on the selected tolerance value.

8.4. Modelling Assumptions

The model assumes constant material properties, a uniform grid, and a simple rectangular geometry. These assumptions simplify the problem but limit applicability to more complex engineering scenarios, where properties may vary with temperature and geometries may be irregular.

8.5. Summary

The EFDM model provides accurate results for the defined problem but is limited by discretisation error, stability constraints, and computational cost. These limitations highlight the importance of selecting appropriate numerical parameters and justify the need for more advanced methods in complex applications.

9. Suggested Improvements and Future Developments

9.1. Numerical Method Efficiency

The high iteration count required to reach equilibrium highlights the inefficiency of the explicit scheme for this problem. Replacing the EFDM with an implicit method, such as the Crank–Nicolson scheme, would remove the stability constraint and allow significantly larger time steps (Crank, J.; Nicolson, P., 1947). This would reduce computational time while maintaining accuracy.

9.2. Spatial Resolution Strategy

The sensitivity analysis showed that reducing grid spacing improves accuracy, particularly in regions of high temperature gradient near the top boundary. Instead of refining the entire domain, an adaptive grid could be used to increase resolution only in these regions. This would improve accuracy where required without a proportional increase in computational cost.

9.3. Boundary Region Accuracy

The validation results indicated that the largest errors occur at the top corner nodes. This is due to steep gradients and limitations of the central difference approximation near boundaries. Applying higher-order finite difference schemes or local grid refinement at the boundaries would reduce these localised errors and improve overall solution quality.

9.4. Convergence Definition

The current equilibrium condition is based on a numerical tolerance, which may not fully represent physical steady state. A more robust approach would involve monitoring residual reduction or ensuring negligible energy change within the system, providing a more physically meaningful convergence criterion.

9.5. Model Applicability

The current model is restricted to a simplified domain and constant properties. Extending the model to include temperature-dependent properties or more complex geometries would improve its relevance to practical engineering applications.

9.6. Summary

The model can be improved by increasing computational efficiency, refining spatial accuracy in critical regions, improving boundary treatment, and adopting more physically representative convergence criteria. These changes would enhance both accuracy and applicability without fundamentally altering the modelling approach.

10. Conclusion

This study developed and validated a numerical model to simulate transient heat conduction in a two-dimensional aluminium plate using the Explicit Finite Difference Method. The governing heat equation was successfully discretised and implemented in MATLAB, with appropriate handling of boundary conditions and numerical stability.

Validation against the analytical solution showed strong agreement across the majority of the domain. The low mean error confirmed that the model accurately represents the temperature distribution, while the observed localised errors at the top corners were attributed to steep gradients and boundary discontinuities.

The sensitivity analysis demonstrated that grid spacing is the primary factor influencing accuracy. Refining the grid significantly reduced the mean error, confirming convergence of the numerical solution. In contrast, the time step mainly affected computational efficiency rather than the final steady-state solution.

The results also highlighted key limitations of the explicit method, particularly the high number of iterations required to reach equilibrium and the dependence on discretisation parameters. These limitations suggest that alternative numerical approaches and targeted grid refinement could improve both efficiency and accuracy.

Overall, the model provides a valid and reliable representation of heat diffusion for the defined problem and demonstrates the effectiveness of the EFDM when applied with appropriate parameter selection.

References

Carslaw, H.S.; Jaeger, J.C., 1959. *Conduction of Heat in Solids*, Oxford: Oxford University Press.

Crank, J.; Nicolson, P., 1947. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Proceedings of the Cambridge Philosophical Society*, 43(1).

Croft, D.R.; Lilley, D.G., 1977. *Heat Transfer Calculations Using Finite Difference Equations*, London: Applied Science Publishers.

Incropera, F.P.; DeWitt, D.P.; Bergman, T.L.; Lavine, A.S., 2007. *Fundamentals of Heat and Mass Transfer*. 6th ed. Hoboken: Wiley.

LeVeque, R.J., 2007. *Finite Difference Methods for Ordinary and Partial Differential Equations*, Philadelphia: SIAM.

Mojumder, M.S.H.; Rahman, M.M.; Islam, M.S., 2023. Efficient finite difference methods for the numerical analysis of the heat equation. *Journal of Applied Mathematics and Physics*, Issue 11.

Ozisik, M.N., 1993. *Heat Conduction*. 2nd ed. New York: Wiley.

Appendix

```
clc;

clear;

close all;

%%    Material Properties

k = 143;

rho = 2.8e3;

cp = 795;

alpha = k/(rho*cp);

fprintf('Thermal diffusivity alpha = %.6e m^2/s\n\n', alpha);

%%    User Inputs

Xdim = input('Enter plate width Xdim (m): ');

Ydim = input('Enter plate height Ydim (m): ');

T0 = input('Enter top boundary temperature T0 (C): ');

%%    Grid Setup

while true

    h = input('Enter grid spacing h (m): ');

    Nx_test = Xdim / h;

    Ny_test = Ydim / h;

    if h > 0 && abs(Nx_test-round(Nx_test))<1e-12 && abs(Ny_test-round(Ny_test))<1e-12

        Nx = round(Nx_test);

        Ny = round(Ny_test);

        nx = Nx + 1;

        ny = Ny + 1;
```

2D Transient Heat Equation Solved with the
Explicit Difference Method

```
        break;
    else
        fprintf('Invalid h. Must divide dimensions evenly.\n');
    end
end

fprintf('Grid: %d x %d nodes\n\n', nx, ny);

%% Time Step
dt_max = h^2/(4*alpha);

while true
    dt = input('Enter time step dt (s): ');

    if dt <= 0
        fprintf('dt must be positive.\n');
        continue;
    end

    r = alpha*dt/h^2;

    fprintf('r = %.4f (must be <= 0.25)\n', r);
    fprintf('Maximum stable dt = %.3e\n', dt_max);

    if r <= 0.25
        fprintf('Time step accepted.\n\n');
        break;
    else
        fprintf('Time step too large → UNSTABLE. Try again.\n\n');
    end
end
```

2D Transient Heat Equation Solved with the Explicit Difference Method

end

%% Create Grid ---

x = linspace(0, Xdim, nx);

y = linspace(0, Ydim, ny);

[X, Y] = meshgrid(x, y);

%% Initialise Temperature

T = zeros(ny, nx);

T_new = T;

% Boundary conditions

T(1,:) = 0; % South

T(:,1) = 0; % West

T(:,end) = 0; % East

T(end,:) = 0; % Reset top row

T(end,2:end-1) = T0; % Top interior

%% EFDM Solver

tol = 1e-6;

max_iter = 5e5;

for it = 1:max_iter

for j = 2:ny-1

for i = 2:nx-1

T_new(j,i) = T(j,i) + r*(...

T(j+1,i)+T(j-1,i)+T(j,i+1)+T(j,i-1)-4*T(j,i));

end

end

2D Transient Heat Equation Solved with the Explicit Difference Method

```
% Reapply BCs

T_new(1,:) = 0;
T_new(:,1) = 0;
T_new(:,end) = 0;
T_new(end,:) = 0;
T_new(end,2:end-1) = T0;

if max(abs(T_new(:)-T(:))) < tol
    fprintf('Converged in %d iterations\n', it);
    break;
end

T = T_new;
end

if it == max_iter
    fprintf('Warning: Max iterations reached\n');
end

%% Time to Equilibrium

time_eq = it * dt;
fprintf('Time to equilibrium = %.3f s\n', time_eq);

%% Numerical Plot

figure;
contourf(X,Y,T,30,'LineColor','none');
colorbar;
colormap(jet);
clim([0 T0]);
```

2D Transient Heat Equation Solved with the
Explicit Difference Method

```
title('Numerical Solution');

xlabel('x (m)');
ylabel('y (m)');

%% Analytical Solution

n_terms = 150;

T_ana = zeros(ny,nx);

for j = 1:ny
    for i = 1:nx

        xv = X(j,i);
        yv = Y(j,i);

        sum_term = 0;
        for n = 0:n_terms-1
            m = 2*n + 1;

            num = exp((m*pi*(yv - Ydim))/Xdim) - ...
                exp((-m*pi*(yv + Ydim))/Xdim);
            den = 1 - exp((-2*m*pi*Ydim)/Xdim);

            sum_term = sum_term + (1/m)* ...
                sin((m*pi*xv)/Xdim)*(num/den);
        end

        T_ana(j,i) = (4*T0/pi)*sum_term;
    end
end
```

2D Transient Heat Equation Solved with the Explicit Difference Method

%% Error Analysis

```
err = abs(T - T_ana);
```

```
fprintf('Mean error = %.3e C\n', mean(err(:)));
```

```
fprintf('Max error = %.3e C\n\n', max(err(:)));
```

%% Comparison Plots

```
figure;
```

```
subplot(1,3,1);
```

```
contourf(X,Y,T_ana,30,'LineColor','none');
```

```
colorbar; colormap(jet); clim([0 T0]);
```

```
title('Analytical');
```

```
subplot(1,3,2);
```

```
contourf(X,Y,T,30,'LineColor','none');
```

```
colorbar; colormap(jet); clim([0 T0]);
```

```
title('Numerical');
```

```
subplot(1,3,3);
```

```
contourf(X,Y,err,30,'LineColor','none');
```

```
colorbar;
```

```
title('Error');
```