ME1060 Final Project: Solving Unsteady Heat Equation in 2D

Shane Riley

Department of Mechanical Engineering and Materials Science,
University of Pittsburgh, Pittsburgh, PA 15261, USA

Abstract

In this work, the partial differential equation (PDE) describing transient heat flow through a 2-dimensional plate is solved numerically using a forward-in-time central-in-space differentiation scheme. To this end, a three point central differencing scheme and the explicit Euler's method are employed. A Neumann stability analysis, as well as a grid study, are performed to determine the effect of varying spatial and temporal step sizes and stability and results. All routines (expect matrix inversion) are self-performed. A smooth steady-state profile is found using this method.

Keywords: Partial Differential Equation, Explicit Finite Difference, Stability

Nomenclature

T	Temperature $[K]$
$k\rho$	Thermal Conductivity $[W/mK]$
ρ	Density $[kg/m^3]$
Cp	Specific Heat $[J/kgK]$
h	Step size in space $[m]$
dt	Step size in time $[s]$

Subscripts

p	from previous frame

1. Problem Statement

Consider two-dimensional heat equation:

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

where $\alpha = \frac{k}{\rho C_p}$. Here k denotes the thermal conductivity and ρ and C_p represent density and specific heat, respectively.

*Author, Phone: 412-402-8402
Email address: shane.riley@pitt.edu ()

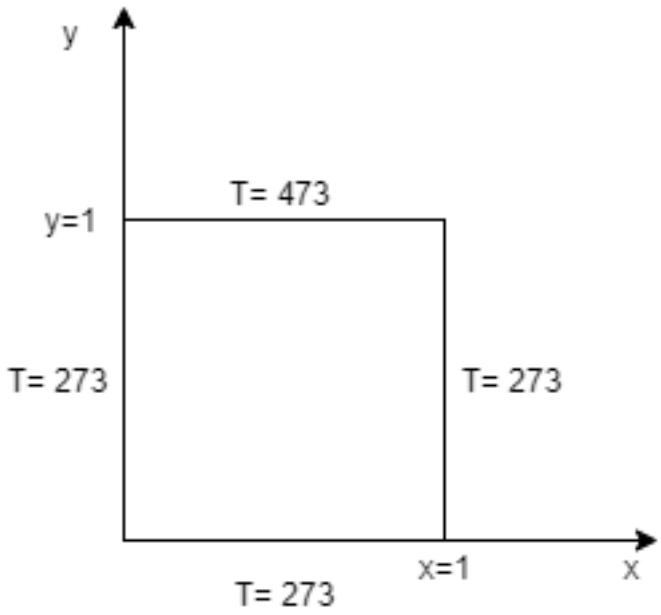

Figure 1: Boundary conditions (temperatures are given in Kelvin).

Solve Eq. 1 for pure Aluminium with material properties presented in Table 1 and Boundary conditions are constant values (Dirichlet boundary conditions) and shown in Fig. 1.

Material	Thermal Conductivity ($J/m.K.s$)	Density (kg/m^3)	Specific Heat ($J/kg.K$)
Pure Aluminium	220	2707	896

Table 1: Material properties

2. Introduction

Literature review, problem statement, your assumption, and the details of the numerical method go here.

3. Methodology

3.1. Discretization

In order to apply a differentiation scheme to solve this problem, we must first discretize the space and and the span of time. This discrete space is represented as a stack of node grids, one grid for each timestep. Since the problem is being solved forward in time, the node grids will be solved inductively, one at a time. The lateral distance between nodes in the node grid is represented as h, and the time difference between node grids (frames) is dt.

Temperatures for each frame are folded and stored in a one-dimensional vector. This is done by giving every node in the space an assigned number, counting upward horizontally left-to-right. For example: a $1[m^2]$

square plate with $h = 0.1[m]$ would have 11 nodes to a side, resulting in a state vector of length 11^2. While this method is great for storage, it requires a system of finding adjacent nodes from a given node number. This is easily expressed using function handles:

```
% Get node above (+y)
n_up = @(n) n + nodes_per_row;

% Get node below (-y)
n_down = @(n) n - nodes_per_row;

% Get node to left (-x)
n_left = @(n) n - 1;

% Get node to right (+x)
n_right = @(n) n + 1;
```

Listing 1: Node connectivity

Additionally, it becomes necessary to find all of the top and wall nodes, such that the boundary conditions can be applied more easily:

```
bottom_nodes = 1:nodes_per_row;
top_nodes = (num_nodes - nodes_per_row + 1):num_nodes;
right_nodes = nodes_per_row:nodes_per_row:num_nodes;
left_nodes = 1:nodes_per_row:num_nodes;
wall_nodes = union(union(bottom_nodes, right_nodes), left_nodes);
```

Listing 2: Wall nodes

3.2. Applying the PDE

In order to solve 1 numerically, it must be expressed as an equation of constants and a linear combination of nodal temperatures. We will apply Euler's explicit method in time and 3-point central differentiation in space to represent the terms. n is the node number, or index in the temperature vector. The node numbers are sub-scripted when an adjacent node is necessary. T_p is the vector of the temperatures from the previous frame, which will already be solved when evaluating the current frame.

$$\frac{\partial T}{\partial t} \approx \frac{T(n) - T_p(n)}{dt} \tag{2}$$

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T(n_{x-1}) + T(n_{x+1}) - 2T(n)}{h^2} \tag{3}$$

$$\frac{\partial^2 T}{\partial y^2} \approx \frac{T(n_{y-1}) + T(n_{y+1}) - 2T(n)}{h^2} \tag{4}$$

Substituting into 1 and rearranging yields the following, where $n_{adjacent}$ represents the 4 laterally connected nodes (found using the function handles):

$$T(n)(\frac{1}{dt} + \frac{4\alpha}{h^2}) - \frac{\alpha}{h^2}(\sum T(n_{adjacent})) = \frac{T_p(n)}{dt} \tag{5}$$

3.3. Running the Solution

This equation is populated into each row of a linear system of order equal to the number of spatial nodes. When the matrix is inverted, the nth frame is found and the timestep moves forward. For the first frame, an initial condition of $273[K]$ is applied on the internal space, and the Dirichlet boundaries are applied on every timestep as well.

```
% Coeffs
c_n = (1/dt) + (4*alpha)*((1/h^2));
c_w = -alpha/h^2;

% Iterate over timesteps
for t=2:num_steps

    % delT/delt
    b = zeros(num_nodes, 1);

    % alpha((del^2 T/delx^2) + (del^2 T/dely^2))
    A = zeros(num_nodes);

    % Build the A matrix
    for n=1:num_nodes

        % Check for walls
        if ismember(n,top_nodes)
            A(n,n) = 1;
            b(n) = T_H;
        elseif ismember(n,wall_nodes)
            A(n,n) = 1;
            b(n) = T_W;
        else
            % Inside
            A(n,n) = c_n;
            A(n,n_down(n)) = c_w;
            A(n,n_up(n)) = c_w;
            A(n,n_left(n)) = c_w;
            A(n,n_right(n)) = c_w;
            % Add to b matrix
            b(n) = T(n,t-1)/dt;
        end
```

```
    end

    % Find new temp
    T(:,t) = (A\b);

end
```

Listing 3: Solve for frames.

3.4. Postprocessing

The result from this calculation is stored in an $m * n$ matrix, where m is the number of nodes and n is the number of timesteps. A reshaping function is used to grab a timestep and reshape it for plotting with x and y coordinates. In the script, the entire solution is wrapped in a user-defined function to allow for easy runtime analysis.

4. Results and Discussion

For creating results for plotting, the scheme is employed with $(h, dt) = (0.05, 1)$ over a timespan from 0 to $1000[s]$. The boundaries discussed in the problem statement are applied, and a uniform temperature $T = 273[K]$ initial condition is arbitrarily applied. It is worth noting that due to the presence of only Dirichlet boundary conditions, the steady-state temperature profile will not depend on the initial condition. The solution is plotted over time in Figure 2, and the same solution is recorded as a video in the attached documentation:

It is intuitively clear from the plots in Figure 2 that the temperature profile rapidly approaches a steady-state value for ever-increasing time.

4.1. Neumann Stability and Error

Since the differentiation schemes arise from Taylor Series approximations, we can quantify the truncation error:

$$TE = O(dt, h^2) \tag{6}$$

As step sizes decrease, the round off error will overtake truncation error. With an analytical solution for comparison, this tradeoff could be examined by finding the effect of these discretization constants on error. This is taken as future work.

The numerical implementation also features a stability criterion of its own–Neumann stability. This stability is expressed as a conditional:

$$\frac{\alpha * dt}{h^2} \leq \frac{1}{2} \tag{7}$$

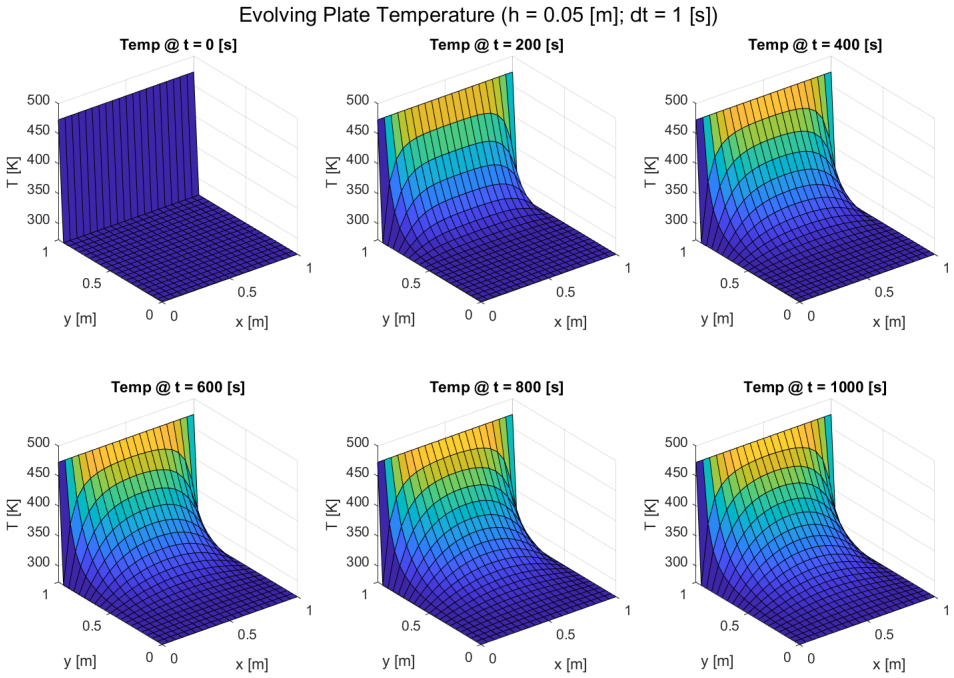

Figure 2: Temperature over Time.

Given pure aluminum and a minimum spatial step of $h = 0.025[m]$, a maximum timestep is found:

$$dt_{max} = \frac{h^2}{2\alpha} \tag{8}$$

Using this equation, $dt_{max} = 3.45[s]$. For the smallest used spatial step, keeping timestep under this value will ensure Neumann stability.

4.2. Runtime Analysis

A runtime study is performed by sampling a space of $h = 0.1, 0.05, 0.025$, and $dt = 2, 1, 0.5$. The setup and solution of each solution is timed. The plot is shown in Figure 3. As expected, the runtime scales linearly with the inverse of dt, and much more extremely with the inverse of h (the size of the A matrix grows quadratically with the spatial step, since the problem is two-dimensional).

5. Conclusions

Using MATLAB, a forward-in-time centered-in-space is successfully implemented to solve the transient heat equation. The simulation is relatively time and space-efficient. To this end, a nodal storage scheme is employed to effectively handle the multi-dimensional problem The FTCS scheme features a linear runtime over timespan, and quadratic time over space (2-dimensional space). As expected, the temperature profile

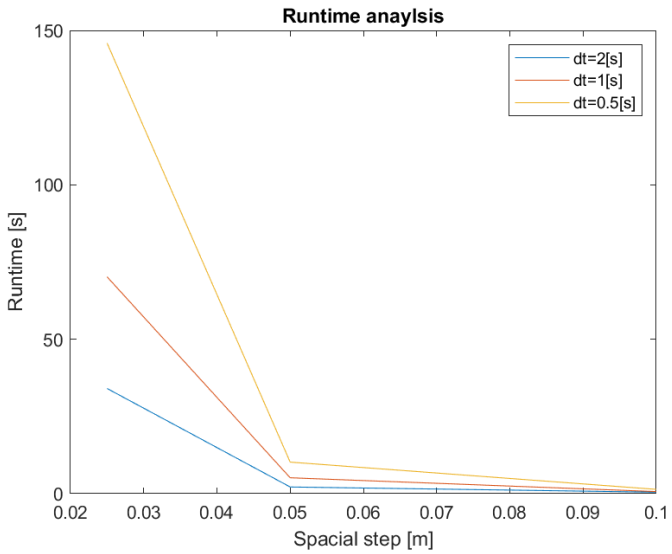

Figure 3: Runtime analysis.

reaches a smooth steady-state profile rather quickly. The truncation error is $O(dt, h^2)$, and Neumann stability is guaranteed.