



**Prifysgol Cymru**  
Y Drindod Dewi Sant  
**University of Wales**  
Trinity Saint David

# TRANSIENT THERMAL DIFFUSION MODEL USING THE FINITE DIFFERENCE METHOD

Advanced Computational Methods

BEng Motorsport Engineering

27/01/2020

M6X00408

***VIJAY SINGH KUNER LACARRERE (P150659)***

## Table of Contents

|                                                                         |                                     |
|-------------------------------------------------------------------------|-------------------------------------|
| <b>1. INTRODUCTION .....</b>                                            | <b>Error! Bookmark not defined.</b> |
| <b>2. HEAT TRANSFER .....</b>                                           | <b>Error! Bookmark not defined.</b> |
| 2.1 Types of Heat Transfer .....                                        | 4                                   |
| 2.2 Conduction .....                                                    | 4                                   |
| <b>3. Finite Difference .....</b>                                       | <b>5</b>                            |
| 3.1 Finite Difference Form of the Heat Equation .....                   | 5                                   |
| 3.2 The Explicit Finite Difference Method .....                         | 6                                   |
| 3.3 Gaussian/Normal Distribution of the North Boundary .....            | 7                                   |
| 3.4 South, West, East, Bottom Right and Bottom Left boundaries .....    | 8                                   |
| <b>4. Script .....</b>                                                  | <b>9</b>                            |
| 4.1 Material Properties .....                                           | 9                                   |
| 4.2 User Inputs .....                                                   | 9                                   |
| 4.3 North Boundary Gaussian Distribution .....                          | 10                                  |
| 4.4 Finite Difference Method till Thermal Equilibrium is achieved ..... | 10                                  |
| 4.5 Information given to user .....                                     | 12                                  |
| <b>5. Analysis of the script .....</b>                                  | <b>13</b>                           |
| 5.1 Grid spacing difference .....                                       | 14                                  |
| 5.3 Time step difference .....                                          | 15                                  |
| <b>6. Conclusion .....</b>                                              | <b>16</b>                           |
| <b>7. Further Work .....</b>                                            | <b>16</b>                           |
| <b>8. References .....</b>                                              | <b>17</b>                           |
| <b>9. Appendix A (Excel Spreadsheet) .....</b>                          | <b>18</b>                           |
| <b>10. Appendix B (Script) .....</b>                                    | <b>20</b>                           |

## 1. Introduction

Use of Matlab to create a script, to showcase heat transfer along an Aluminium plate, with insulated boundary conditions and Normal/Gaussian Distribution.

The conditions given are the following, they can also be seen in Figure 1:

- Aluminium Properties:
  - Conductivity:  $143 \text{ W/mC}$
  - Density:  $2800 \text{ Kg/m}^3$
  - Specific Heat:  $795 \text{ J/KgC}$
- Boundary Conditions:
  - Normal/Gaussian Distribution of the North Boundary between  $10^\circ\text{C}$  and  $60^\circ\text{C}$ .
  - South, East and West boundaries are perfectly well insulated, no heat flow through those boundaries.
  - The initial temperature of the plate within the domain is of  $0^\circ\text{C}$

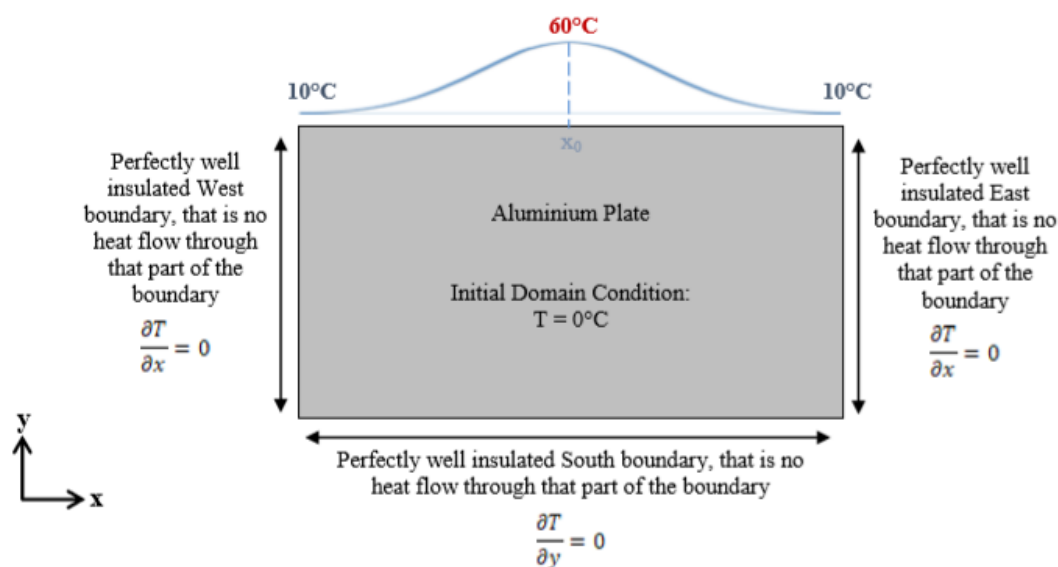


Figure 1 – Boundary conditions (Marotin, 2019)

Using this initial data, the heat transfer along the plate can be calculated using the Explicit Finite Difference Method on a 2D surface, within Matlab. Furthermore, it is also done in excel to cross examine results.

## 2. Heat Transfer

### 2.1 Types of Heat Transfer

According to (Theodore L. Bergman, 2011), "Heat transfer is thermal energy in transit due to a spatial temperature difference"

There are 3 types of heat transfer:

- Conduction: looking at this from a molecular level, this is the transfer of energy from a higher temperature molecule to the other when they are colliding. These random collisions create a transfer of energy, which can be called diffusion of energy. (Theodore L. Bergman, 2011)
- Convection: As well as transfer through random molecular motion (diffusion), energy is also transferred in bulk. This means, that many molecules are moving as a collective and transferring energy. (Theodore L. Bergman, 2011)
- Radiation: Is energy travelling through space. This kind of energy unlike conduction and convection, does not require the presence of a material medium. (World Nuclear Association, 2020)

This report will only cover Conduction of heat, as no convection or radiation are happening on the medium.

### 2.2 Conduction

Conduction in solids, can be attributed to atomic activity in the form of lattice vibrations. In an electrical non-conductor, the energy transfer only happens through these vibrations, while in a conductor, it is also due to the translational motion of the free electrons. (Theodore L. Bergman, 2011)

To calculate the conduction in time across a 2D plane the following diffusion equation, Equation 1, is used:

$$\nabla^2 T = \frac{1}{\alpha} \frac{\partial T}{\partial t} \quad \text{Equation 1}$$

Where  $\nabla^2 T$  is the Laplacian Operator,  $\alpha$  is the thermal diffusivity,  $T$  is the temperature and  $t$  is time.

$$\alpha = \frac{\text{Thermal Conductivity}}{\text{Density} \times \text{Specific Heat}} \quad \text{Equation 2}$$

$$\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \quad \text{Equation 3}$$

Where  $\nabla^2 T$  is defined in 3 dimensions in time.

Combining these equations, the following equation is obtained:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{\alpha} \frac{\partial T}{\partial t} \quad \text{Equation 4}$$

This equation assumes that there is no internal heat on the plate, the properties of the plate are uniform and non-changing, also due to the heat transfer happening on a 2D element the z axis is removed.

### 3. Finite Difference

#### 3.1 Finite Difference Form of the Heat Equation

Temperature distribution numerically dictates that an appropriate conservation equation shall be written for each node with unknown temperature. These equations will consequently be solved simultaneously for the temperature at each node. (Theodore L. Bergman, 2011)

In this process, the nodes are equidistant from each other, while the central temperature node is the average of the nodes around it (North, South, East and West).

Replacing derivative terms with linear equations by means of Taylor series, which iteratively solves and replaces the second order partial differential equation.

The following Taylor series can be obtained:

$$f'(x_0) = \frac{f(x_0+h)-f(x_0)}{h} * \text{Error } \propto h \quad \text{Equation 5: Forward Difference Approximation}$$

$$f'(x_0) = \frac{f(x_0)-f(x_0-h)}{h} * \text{Error } \propto h \quad \text{Equation 6: Backward Difference Approximation}$$

$$f'(x_0) = \frac{f(x_0+h)-f(x_0-h)}{2h} * \text{Error } \propto h^2 \quad \text{Equation 7: Central Difference Approximation}$$

$$f''(x_0) = \frac{f(x_0+h)-2f(x_0)+f(x_0-h)}{h^2} * \text{Error } \propto h^2 \quad \text{Equation 8: Second Order Central Difference Approximation}$$

In the equations (5, 6, 7 and 8),  $x_0$  is the initial point, while  $x_0 + h$  is a nearby point.

Analysing these equations, as h gets smaller the more accurate results will be obtained. In short, the smaller the distance between nodes the accuracy of the heat transfer will be greater.

### 3.2 The Explicit Finite Difference Method

The following equations are explicit because unknown nodal temperatures for the new time are determined by previous known nodal temperatures. In this way, transient temperature distribution is obtained using intervals of time,  $\Delta t$ . (Theodore L. Bergman, 2011)

Using Equations 4 and 7 the following can be found:

$$\frac{\partial T}{\partial t} = \frac{T_{i,j}^{k+1} - T_{i,j}^k}{\Delta t} \quad \text{Equation 9}$$

Where  $k+1$ , is the temperature in the new time and  $k$  is the temperature in the present time. While  $i, j$  specifies the position of the node it is obtaining energy from, as follows:

- $(i, j)$  Centre node.
- $(i + 1, j)$  East node.
- $(i - 1, j)$  West node.
- $(i, j + 1)$  North node.
- $(i, j - 1)$  South node.

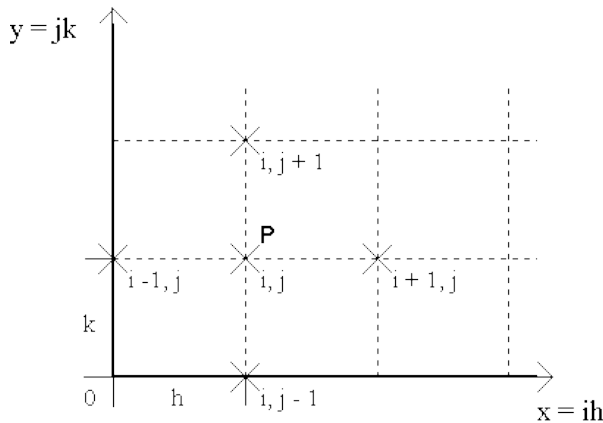


Figure 2 – Nodes (Rocha, 2008)

Substituting into Equation 9 into Equation 4:

$$\frac{1}{\alpha} \frac{T_P^{k+1} - T_P^k}{\Delta t} = \frac{T_E^k + T_W^k - 2T_P^k}{(\Delta x)^2} + \frac{T_N^k + T_S^k - 2T_P^k}{(\Delta y)^2} \quad \text{Equation 10}$$

If  $h = \Delta x = \Delta y$  the following equation can be done:

$$T_P^{k+1} = F_o (T_N^k + T_S^k + T_W^k + T_E^k) - \left(4 - \frac{1}{F_o}\right) T_P^k \quad \text{Equation 11}$$

Where  $F_o$  is the finite difference form of the Fourier number:

$$F_o = \frac{\alpha \Delta t}{h^2} \quad \text{Equation 12}$$

The accuracy of the finite difference method can be improved by decreasing the values of  $h=\Delta x=\Delta y$  by giving the matrix vastly more nodes. The drawback is that the smaller  $h$  is the more computing time it will take for it to reach thermal equilibrium. The choice for  $h$  is usually a compromise between computation time and accuracy. Once  $h$  is chosen, the time step  $\Delta t$  cannot be chosen independently, it must follow a stability criterion. (Theodore L. Bergman, 2011)

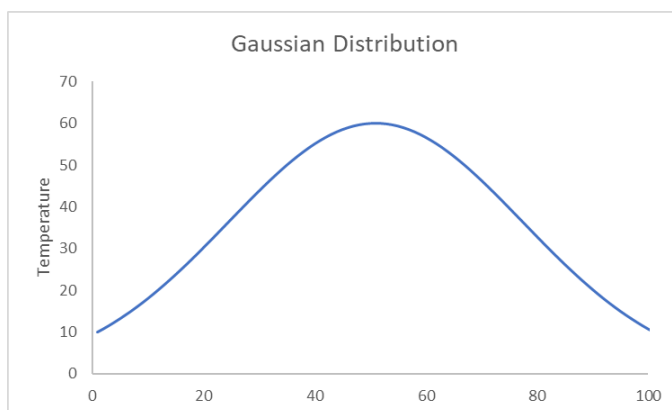
An undesirable feature of the explicit method is that it is not unconditionally stable. In a transient problem, the final nodal temperatures should continuously approach steady state values with increasing time. However, this method is characterised by numerically induced oscillations, these oscillations may become unstable, causing the solution to diverge. To avoid these errors,  $\Delta t$  must be kept under a certain value, which depend on  $h$  and other parameters. This is called stability criterion, the criterion is determined by requiring that the coefficient associated with the node of interest at the previous time is greater than or equal to zero. (Theodore L. Bergman, 2011)

For a 2-dimensional node this criterion would be:

$$F_o \leq \frac{1}{4} \quad \text{Equation 13}$$

### 3.3 Gaussian/Normal Distribution of the North Boundary

The North boundary of the plate is a normal distribution of heat with an amplitude of 60°C and a minimum of 10°C, normal distribution can be plotted as a bell curve like in graph 1



Graph 1- Gaussian Distribution

The Gaussian distribution function is the following:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(-x-\mu)^2}{(2\sigma^2)}} \quad \text{Equation 14 (Weinstein, 2020)}$$

Where  $\sigma$  is the scale parameter and  $\mu$  is the location parameter. (Engineering Statistics Handbook, 2013)<sup>o</sup>

$\frac{1}{\sigma\sqrt{2\pi}}$  is equal to the temperature at the centre of the curve, in this case 60°C. The standard deviation of temperature along the north boundary of the matrix will change depending on matrix size, in which case Equation 14 must be solved to  $\sigma$ , in the following way:

$$\sigma = \sqrt{\frac{-(x-x_0)^2}{2\ln\left(\frac{P(x)}{A}\right)}} \quad \text{Equation 15}$$

Where  $A = 60^\circ\text{C}$ ;  $P(x) = 10^\circ\text{C}$ ;  $x=1$ ;  $x_0 = \frac{\text{rows in matrix}+1}{2}$

Knowing  $\sigma$  it can be implemented into Equation 14, and the following equation is used:

$$\text{Temperature} = 80 * e^{\frac{-\left(\text{Row} - \frac{\text{rows in matrix}+1}{2}\right)^2}{2\sigma^2}} \quad \text{Equation 16}$$

Using Equations 15 and 16 the north boundary from where the heat transfer starts can be set in Matlab.

### 3.4 South, West, East, Bottom Right and Bottom Left boundaries

All other three boundaries, apart from the North boundary, are perfectly insulated. Equation 11 is changed accordingly to accommodate the nodes located at these three boundaries and two corners in the following manner:

$$T_P^{k+1} = F_o(T_N^k + T_N^k + T_W^k + T_E^k) - \left(4 - \frac{1}{F_o}\right) T_P^k \quad \text{Equation 17 – South Boundary}$$

$$T_P^{k+1} = F_o(T_N^k + T_S^k + T_E^k + T_E^k) - \left(4 - \frac{1}{F_o}\right) T_P^k \quad \text{Equation 18 – West Boundary}$$

$$T_P^{k+1} = F_o(T_N^k + T_S^k + T_W^k + T_W^k) - \left(4 - \frac{1}{F_o}\right) T_P^k \quad \text{Equation 19 – East Boundary}$$

$$T_P^{k+1} = F_o(T_N^k + T_N^k + T_E^k + T_E^k) - \left(4 - \frac{1}{F_o}\right) T_P^k \quad \text{Equation 20 – Bottom Left Corner}$$

$$T_P^{k+1} = F_o(T_N^k + T_N^k + T_W^k + T_W^k) - \left(4 - \frac{1}{F_o}\right) T_P^k \quad \text{Equation 21 – Bottom Right Corner}$$

These equations, as well as Equation 11 will iteratively calculate the heat transfer across the plate.

## 4. Script

### 4.1 Material Properties

The start of the script showcases the material properties and the calculation of alpha with equation 2

```
%% Material Properties
name=('Aluminium');           % Material name
conductivity = 143;           % Thermal conductivity(W/m.K)
specific_heat = 795;          % specific heat (J/kg K)
density = 2.8*10^3;           % density (kg/m^3)
alpha = conductivity/(specific_heat*density); % alpha (1/sec)
```

### 4.2 User Inputs

In this section the script asks the user to input the Dimensions of the plate, the distance between nodes and the time step. If the data the user inputs is negative the script will give the user a warning with the fprintf command and ask them to input a new piece of data with the conditions the script gives them, this will continue by means of a while loop till the script is satisfied with the users input. Furthermore, if the user hits the enter key by accident while inputting empty data, the isempty command will tell the script that no value has been entered and will keep the user in the while loop till the user does input a value.

The user is also given a suggestion on the size of the grid space using a fprintf command to make sure the size of the grid space is less than half of the X or Y dimension of the plate.

The time stability condition is also used within the while loop to when inputting the time step, with an fprintf command suggesting to the user what option would give the best results. Larger the time step, the larger the Fourier number will be and therefore the larger the amount of heat transfer per time step.

```
dt=0;                               % Time step
dtmin=0;
dtmax=1/(2*alpha*((1/dx^2)+(1/dy^2))); %test the stability condition
while dt<=dtmin || dt>=dtmax
    fprintf('For most accurate results please input time step closest to %f \n',1/(2*alpha*((1/dx^2)+(1/dy^2))))
    dt=input('Enter time step: ');
    if isempty(dt)                    %Replacing empty value with 0
        dt=0;
    end
    if dt<=dtmin || dt>=dtmax
        fprintf('Error, the stability condition is not met\nPlease choose a "dt" smaller than %f \n',1/(2*alpha*((1/dx^2)+(1/dy^2))))
    end
```

end

The number of nodes

### 4.3 North Boundary Gaussian Distribution

Initially a matrix Tf is created, from this matrix a north boundary can be implemented through Gaussian Distribution, then an if statement is used to specify that on row one is where the Gaussian distribution will be implemented.

Implementing Equation 15 and 16 into the script as follows:

```
%% Gaussian Distribution North Boundary
Tf=zeros(Ny,Nx);
normal=sqrt(-(1-((Nx+1)/2))^2)/(2*log(10/60));           %Normal or Gaussian Distribution

for r=1:Nx
    Tf(1,r)=60*exp(-((r-((Nx+1)/2))^2/(2*normal^2)));      %North Normal distribution boundary
condition
    r=r+1;
end
```

### 4.4 Finite Difference Method till Thermal Equilibrium is achieved

Initial Temperature (Tf) is the plate at present time, while Final Temperature (Tf) represents the plate at the next time step +1. A tolerance of 0.0001 is used which tells the script that the plate is at thermal equilibrium once it has a difference in temperature between nodes as the tolerance says, the lower the tolerance the more accurate the final answer will be but the more processing power is needed, so 0.0001 gives the user the best computing to accuracy ratio.

The while loop specifies that i will represent the second row to the last row in the Ny, which is the number of nodes in the Y axis. And j will represent the first column to the last column in the Nx, which is the number of nodes in the X axis.

By using equations (11, 17 , 18 , 19, 20 and 21) within the while loop using if and elseif statements it is possible to specify the location of these nodes and tell the script what to do at these nodes.

The West nodes it specifies in the if statement that they are the ones in the first column (j==1) except the one node on the last row (i<Ny), then equation 18 is used. Furthermore, the East nodes are in the last column (j==Nx), as well as the West node the last row is taken away (i<Ny), then equation 19 is used.

The Bottom Right node it specifies by saying it is in the last column ( $j=N_x$ ) and the last row ( $i=N_y$ ) then equation 21 is used, while for the Bottom left node the column changes to the first ( $j=1$ ) and equation 20 is used.

The South nodes are specified by saying they are in the last row ( $i=N_y$ ) and between the first and last column ( $j>1; j<N_x$ ).

And then all the central nodes use equation 11

```
%% Finite Difference Method till Thermal Equilibrium
Ti=0.1*Tf;                                %iteration criteria
k=0;                                       %iteration counter
tolerance=0.0001;                         %temperature tolerance

while max(max(abs(Tf-Ti)))>tolerance
    Ti=Tf;
    for i=2:Ny
        for j=1:Nx
            if j==1 && i<Ny                %West Nodes
                Tf(i,j)=Fo*(Ti(i+1,j)+Ti(i-1,j)+Ti(i,j+1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            elseif j==Nx && i==Ny          %Bottom Right Node
                Tf(i,j)=Fo*(Ti(i-1,j)+Ti(i-1,j)+Ti(i,j-1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            elseif j==1 && i==Ny           %Bottom Left Node
                Tf(i,j)=Fo*(Ti(i-1,j)+Ti(i-1,j)+Ti(i,j+1)+Ti(i,j+1)-(4-(1/Fo))*Ti(i,j));
            elseif j==Nx && i<Ny           %East Nodes
                Tf(i,j)=Fo*(Ti(i+1,j)+Ti(i-1,j)+Ti(i,j-1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            elseif i==Ny && j>1 && j<Nx    %South Nodes
                Tf(i,j)=Fo*(Ti(i-1,j)+Ti(i-1,j)+Ti(i,j+1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            else                            %Centre Nodes
                Tf(i,j)=Fo*(Ti(i+1,j)+Ti(i-1,j)+Ti(i,j+1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            end
            j=j+1;
        end
        i=i+1;
    end
    error=max(max(abs(Tf-Ti))); %Temperature difference between iterations
    k=k+1;                     %Number of iterations
end
```

To finish the while loop keeps going through iterations, till the tolerance is met.

## 4.5 Information given to user

While the while loop is running a fprintf command is used to communicate to the user at what stage iteration and the temperature difference between iterations till it reaches the tolerance value of 0.0001, in real time. Once the while loop finishes, within the command window it is possible to see the final matrix at thermal equilibrium.

A contour plot is created showcasing the final heat transfer at thermal equilibrium, an animation of such plot could be made showing each operation, but this drastically increases computing time, and it was chosen to just show the final iteration of the heat transfer at thermal equilibrium.

```
%% Contour plot of the heat transfer
figure(1)
contourf (Tf,100,'linecolor','none')
colormap jet
colorbar 'Eastoutside'
caxis([10 60])
title(['Plate at thermal equilibrium'])
view(180,90)
```

Using an if statement it is also possible to tell the user how much time it would take the plate to reach thermal equilibrium, which is showcased in the command window, as well as the total amount of iterations to reach thermal equilibrium and that the script has ended.

```
%% Time till thermal equilibrium is reached
Time_seconds=dt*k;
if Time_seconds<3600
    fprintf('Seconds taken to reach thermal equilibrium are: %f \n',Time_seconds)
    %Seconds
elseif Time_seconds>3600 && Time_seconds<86400
    Time_hours=Time_seconds/3600;
    fprintf('Hours taken to reach thermal equilibrium are: %f \n',Time_hours)
    %Hours
elseif Time_seconds>86400
    Time_days=Time_seconds/86400;
    fprintf('Days taken to reach thermal equilibrium are: %f \n',Time_days)
    %Days
end
fprintf('Total amount of iterations to reach thermal equilibrium: %f \n',k)
fprintf('End of Process Reached')
%End
```

## 5. Analysis of the script

To make sure the data given by the matlab code is correct it was cross examined using the iteration option on excel, to compare the two the following inputs where used:

- $L_x = 1$  (X dimension)
- $L_y = 0.9$  (Y dimension)
- $h = 0.1$  (grid space)
- $dt = 31.838$  (Time step closest to the stability condition)

Table 1 – Percentage difference

| 10.000 | 19.061 | 31.479 | 45.045 | 55.850 | 60.000 | 55.850 | 45.045 | 31.479 | 19.061 | 10.000 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0.276  | 0.276  | 0.276  | 0.276  | 0.276  | 0.276  | 0.276  | 0.276  | 0.276  | 0.276  | 0.276  |
| 0.544  | 0.544  | 0.544  | 0.544  | 0.544  | 0.544  | 0.544  | 0.544  | 0.544  | 0.544  | 0.544  |
| 0.795  | 0.795  | 0.795  | 0.795  | 0.795  | 0.795  | 0.795  | 0.795  | 0.795  | 0.795  | 0.795  |
| 1.023  | 1.023  | 1.023  | 1.023  | 1.023  | 1.023  | 1.023  | 1.023  | 1.023  | 1.023  | 1.023  |
| 1.219  | 1.219  | 1.219  | 1.219  | 1.219  | 1.219  | 1.219  | 1.219  | 1.219  | 1.219  | 1.219  |
| 1.378  | 1.378  | 1.378  | 1.378  | 1.378  | 1.378  | 1.378  | 1.378  | 1.378  | 1.378  | 1.378  |
| 1.495  | 1.495  | 1.495  | 1.495  | 1.495  | 1.495  | 1.495  | 1.495  | 1.495  | 1.495  | 1.495  |
| 1.567  | 1.567  | 1.567  | 1.567  | 1.567  | 1.567  | 1.567  | 1.567  | 1.567  | 1.567  | 1.567  |
| 1.591  | 1.591  | 1.591  | 1.591  | 1.591  | 1.591  | 1.591  | 1.591  | 1.591  | 1.591  | 1.591  |

Table 1 shows the percentage difference between the excel data and the Matlab data. In the Appendix the excel spreadsheet can be seen and the process used to get this data. This low temperature difference means that the matlab code is running in a correct manner and that it can be trusted.

Figure 3 showcases the nearly identical data on a contour plot.

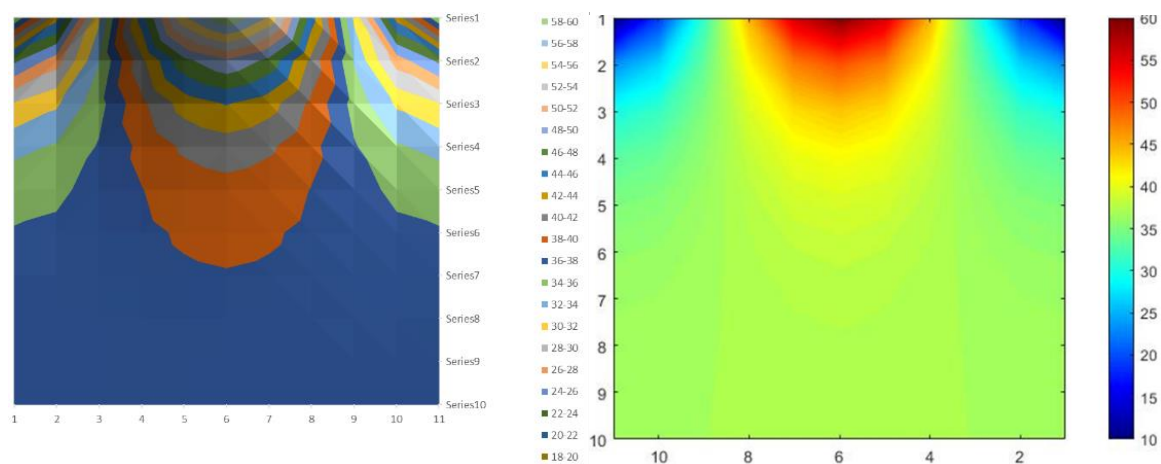


Figure 3 – On the Left, Excel contour plot at thermal equilibrium. On the Right, Matlab contour plot at thermal equilibrium

## 5.1 Grid spacing difference

Using a 5x5m plate as the starting point different grid spacing was tested for comparison.

Table 2 – Grid spacing comparison

|    | Lx (m) | Ly (m) | dt (sec) | h (m) | Iteration number | Total time |
|----|--------|--------|----------|-------|------------------|------------|
| 01 | 5      | 5      | 50       | 2     | 21244            | 12.29 days |
| 02 | 5      | 5      | 50       | 1.5   | 13453            | 7.78 days  |
| 03 | 5      | 5      | 50       | 1     | 15863            | 9.17 days  |
| 04 | 5      | 5      | 50       | 0.5   | 15835            | 9.16days   |
| 05 | 5      | 5      | 50       | 0.25  | 15828            | 9.15 days  |
| 06 | 5      | 5      | 50       | 0.2   | 15827            | 9.15 days  |
| 07 | 5      | 5      | 37.4     | 0.1   | 19932            | 8.62 days  |

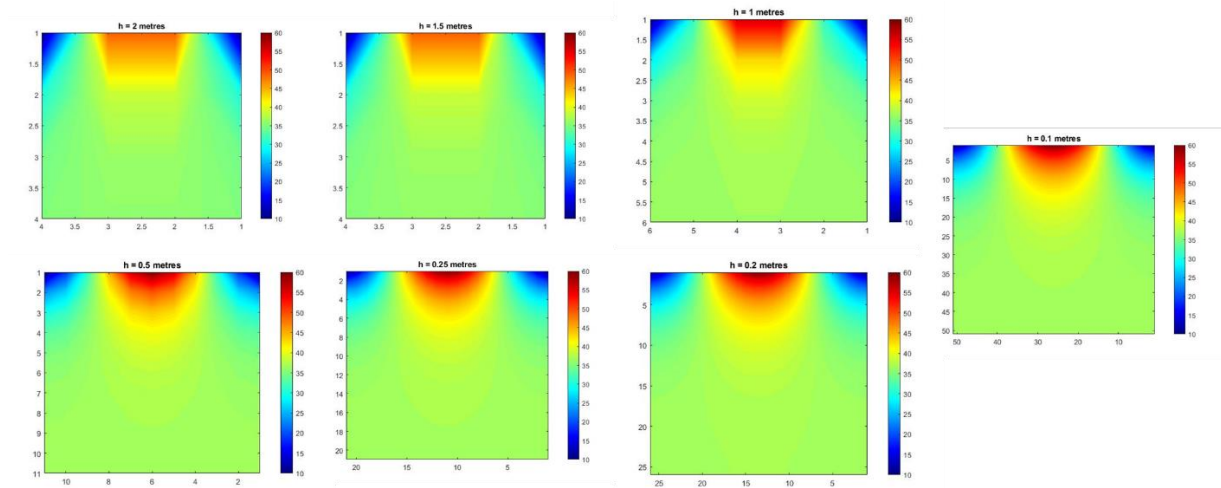


Figure 4 – Contour plots: different grid spacing

Looking at table 2 and at Figure 4, plots 04, 05 and 06 start to converge and are their optimal state. While a grid space too high will not give an accurate read out.

### 5.3 Time step difference

Using a 5x5m plate as the starting point different time steps where tested for comparison.

Table 3 – Time step comparison

|    | Lx (m) | Ly (m) | dt (sec) | h (m) | Iteration number | Total time |
|----|--------|--------|----------|-------|------------------|------------|
| 01 | 5      | 5      | 125      | 0.2   | 7486             | 10.83 days |
| 02 | 5      | 5      | 80       | 0.2   | 10818            | 10.01 days |
| 03 | 5      | 5      | 50       | 0.2   | 15827            | 9.15 days  |
| 04 | 5      | 5      | 30       | 0.2   | 23692            | 8.22 days  |
| 05 | 5      | 5      | 20       | 0.2   | 32341            | 7.48 days  |
| 06 | 5      | 5      | 10       | 0.2   | 53746            | 6.22 days  |
| 07 | 5      | 5      | 1        | 0.2   | 174110           | 2.02 days  |

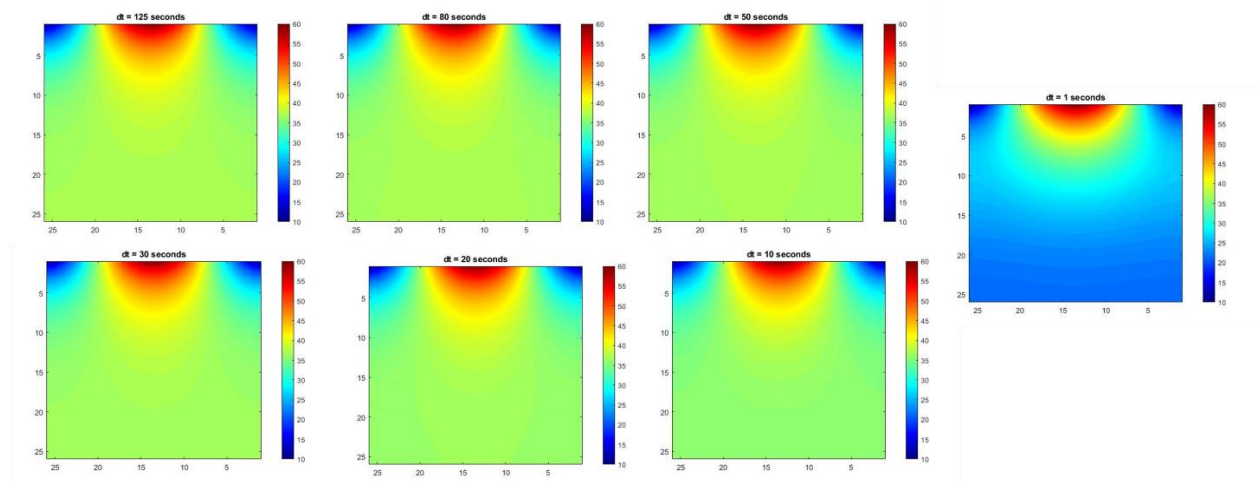


Figure 5 – Contour plots: different time steps

Surprisingly, the smaller time iterations give less accurate results to the higher time iterations, this might be due to the finite difference method equation used, according to (Theodore L. Bergman, 2011) the following equation is used for explicit finite difference approximation:

$$T_P^{k+1} = F_o(T_N^k + T_S^k + T_W^k + T_E^k) + (1 - 4F_o)T_P^k \quad \text{Equation 22 (Theodore L. Bergman, 2011)}$$

But when inputting this into the code, the same issue arises.

The most accurate results are achieved by inputting the closest value to the stability condition.

## 6. Conclusion

The code is very much restricted by the stability condition and using finite difference approximations will never get an exact value as to what the heat transfer will be but will give the user a pretty close approximation from which assumptions can be made.

A limitation of the model is the thermal conductivity is constant, According to the Wiedemann–Frantz law, it states that the ratio of the electrical and thermal conductivities of any metal is a universal constant at a given temperature, which in turn would change the Fourier number along the plate. (Mizutani, 2003)

The model is also limited by it not being realistic in the distribution of heat across the plate, as over time the insulated boundaries would take energy from the neighbouring nodes, at a slower rate than the interior nodes.

It would take a very long time for matlab to compute a very fine mesh at a very low grid space, such as a 10x10m plate with a 0.1m grid space, due to the stability condition limiting the time step to 38.14seconds.

## 7. Further Work

A code which would accommodate convection and radiation, in this case, the Biot number would have to be implemented into the code for convection with the following formula:

$$Bi = \frac{h \cdot \Delta x}{k} \quad \text{Equation 23 (Theodore L. Bergman, 2011)}$$

Where k is the thermal conductivity.

This Biot number plays a vital roll in conduction problems with surface convection effects. (Theodore L. Bergman, 2011)

The script can also be updated to accommodate one dimensional or three-dimensional problems.

To reduce computing time drastically the implicit finite difference method can be used, as it has the advantage over the explicit form of being unconditionally stable. Which means there are no restrictions on grid space and time step. (Theodore L. Bergman, 2011)

## 8. References

- Engineering Statistics Handbook. (2013). *Location and Scale Parameters*. Retrieved from Engineering Statistics Handbook:  
<https://www.itl.nist.gov/div898/handbook/eda/section3/eda364.htm#FORMULAS>
- Marotin, A. (2019). *Heat Transfer Assignment Brief*. Swansea: UWTSD.
- Mizutani, U. (2003). *Introduction to the electron theory of metals*. Cambridge University Press.
- Rocha, K. M. (2008). *Numerical Techniques for Real Options*. Retrieved from MarcoAGD:  
<http://marcoagd.usuarios.rdc.puc-rio.br/katia-num.html>
- Theodore L. Bergman, A. S. (2011). *Fundamentals of Heat and Mass Transfer*. USA: John Wiley and Sons.
- Weinstein, E. W. (2020). *Normal Distribution*. Retrieved from Mathworld-Wolfram:  
<http://mathworld.wolfram.com/NormalDistribution.html>
- World Nuclear Association. (2020). *What is Radiation?* Retrieved from World Nuclear Association:  
<https://www.world-nuclear.org/nuclear-basics/what-is-radiation.aspx>

## 9. Appendix A (Excel Spreadsheet)

Figure 5 shows in the red box the iteration option in excel used.

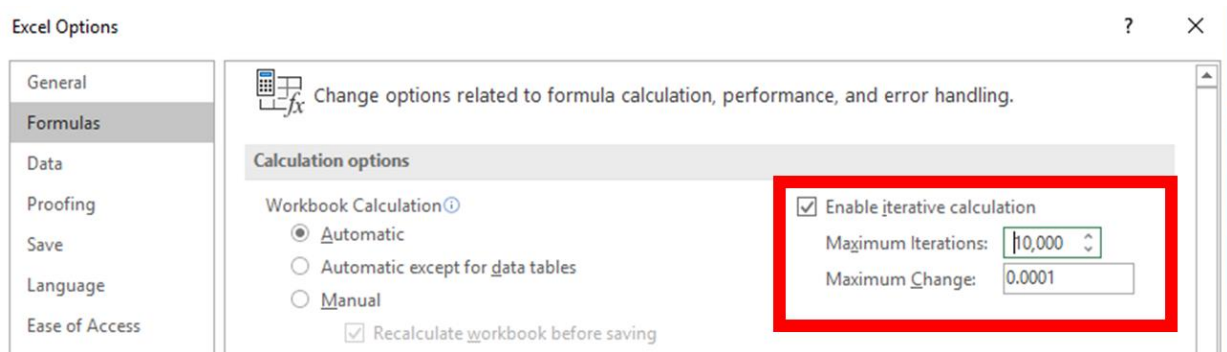


Figure 6 – Appendix: Iteration calculation excel

$$=(K4+J5+L5+K6)/4$$

|   | F | G | H | I | J      | K      | L     | M     | N     | O    |
|---|---|---|---|---|--------|--------|-------|-------|-------|------|
|   |   |   |   |   |        |        |       |       |       |      |
|   |   |   |   |   | 10     | 19.061 | 31.48 | 45.05 | 55.85 | 60   |
|   |   |   |   |   | 23.516 | 27.006 | 33.81 | 41.47 | 47.5  | 49.7 |
| 0 |   |   |   |   | 30.051 | 34     | 35.29 | 39.53 | 42.88 | 44.1 |
|   |   |   |   |   | 33.414 | 34.205 | 36.16 | 38.49 | 40.34 | 41.0 |
|   |   |   |   |   | 35.194 | 35.607 | 36.66 | 37.94 | 38.96 | 39.3 |
| 0 |   |   |   |   | 36.147 | 36.369 | 36.94 | 37.64 | 38.2  | 38.4 |
|   |   |   |   |   | 36.657 | 36.778 | 37.09 | 37.48 | 37.79 | 37.9 |

$$=(J3+K4+K4+J5)/4$$

|   | F | G | H | I | J      | K      | L     | M     |
|---|---|---|---|---|--------|--------|-------|-------|
|   |   |   |   |   |        |        |       |       |
|   |   |   |   |   | 10     | 19.061 | 31.48 | 45.05 |
|   |   |   |   |   | 4      | 27.006 | 33.81 | 41.47 |
| 0 |   |   |   |   | 30.051 | 31.637 | 35.29 | 39.53 |
|   |   |   |   |   | 33.414 | 34.205 | 36.16 | 38.49 |
|   |   |   |   |   | 35.194 | 35.607 | 36.66 | 37.94 |
| 0 |   |   |   |   | 36.147 | 36.369 | 36.94 | 37.64 |

$$=(J11+J11+K12+K12)/4$$

|   | F | G | H | I | J      | K      | L    |
|---|---|---|---|---|--------|--------|------|
|   |   |   |   |   |        |        |      |
|   |   |   |   |   | 10     | 19.061 | 31.4 |
|   |   |   |   |   | 23.516 | 27.006 | 33.8 |
| 0 |   |   |   |   | 30.051 | 31.637 | 35.2 |
|   |   |   |   |   | 33.414 | 34.205 | 36.1 |
|   |   |   |   |   | 35.194 | 35.607 | 36.6 |
| 0 |   |   |   |   | 36.147 | 36.369 | 36.9 |
|   |   |   |   |   | 36.657 | 36.778 | 37.0 |
|   |   |   |   |   | 36.924 | 36.994 | 37.1 |
| 0 |   |   |   |   | 37.052 | 37.097 | 37.2 |
|   |   |   |   |   | K12/4  | 37.127 | 37.2 |

$$=(T11+S12+T11+S12)/4$$

|   | F | G | H | I | J      | K      | L     | M     | N     | O     | P     | Q     | R     | S     | T     |
|---|---|---|---|---|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|   |   |   |   |   |        |        |       |       |       |       |       |       |       |       |       |
|   |   |   |   |   | 10     | 19.061 | 31.48 | 45.05 | 55.85 | 60    | 55.85 | 45.05 | 31.48 | 19.06 | 10    |
|   |   |   |   |   | 23.515 | 27.006 | 33.81 | 41.47 | 47.5  | 49.79 | 47.5  | 41.47 | 33.81 | 27.01 | 23.52 |
| 0 |   |   |   |   | 30.051 | 31.637 | 35.29 | 39.53 | 42.88 | 44.15 | 42.88 | 39.53 | 35.29 | 31.64 | 30.05 |
|   |   |   |   |   | 33.414 | 34.205 | 36.16 | 38.49 | 40.34 | 41.04 | 40.34 | 38.49 | 36.16 | 34.2  | 33.41 |
|   |   |   |   |   | 35.194 | 35.607 | 36.66 | 37.94 | 38.96 | 39.34 | 38.96 | 37.94 | 36.66 | 35.61 | 35.19 |
| 0 |   |   |   |   | 36.147 | 36.369 | 36.94 | 37.64 | 38.2  | 38.42 | 38.2  | 37.64 | 36.94 | 36.37 | 36.15 |
|   |   |   |   |   | 36.657 | 36.778 | 37.09 | 37.48 | 37.79 | 37.91 | 37.79 | 37.48 | 37.09 | 36.78 | 36.66 |
|   |   |   |   |   | 36.924 | 36.994 | 37.18 | 37.4  | 37.58 | 37.65 | 37.58 | 37.4  | 37.18 | 36.99 | 36.92 |
| 0 |   |   |   |   | 37.052 | 37.097 | 37.21 | 37.36 | 37.48 | 37.52 | 37.48 | 37.36 | 37.21 | 37.1  | 37.05 |
|   |   |   |   |   | 37.09  | 37.127 | 37.23 | 37.35 | 37.45 | 37.48 | 37.45 | 37.35 | 37.23 | 37.13 | 37.05 |

$$=(T3+S4+S4+T5)/4$$

|   | F | G | H | I | J      | K      | L     | M     | N     | O     | P     | Q     | R     | S     | T     |
|---|---|---|---|---|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|   |   |   |   |   |        |        |       |       |       |       |       |       |       |       |       |
|   |   |   |   |   | 10     | 19.061 | 31.48 | 45.05 | 55.85 | 60    | 55.85 | 45.05 | 31.48 | 19.06 | 10    |
|   |   |   |   |   | 23.515 | 27.006 | 33.81 | 41.47 | 47.5  | 49.79 | 47.5  | 41.47 | 33.81 | 27.01 | 23.52 |
| 0 |   |   |   |   | 30.051 | 31.637 | 35.29 | 39.53 | 42.88 | 44.15 | 42.88 | 39.53 | 35.29 | 31.64 | 30.05 |
|   |   |   |   |   | 33.414 | 34.205 | 36.16 | 38.49 | 40.34 | 41.04 | 40.34 | 38.49 | 36.16 | 34.2  | 33.41 |

## 10. Appendix B (Script)

```
clear; close all; clc;
%% Material Properties
name=('Aluminium');           % Material name
conductivity = 143;           % thermal conductivity (W/m.K)
specific_heat = 795;          % specific heat (J/kg K)
density = 2.8*10^3;           % density (kg/m^3)
alpha = conductivity/(specific_heat*density); % alpha (1/sec)

%% Input Section
% X dimension
Lx=0;
while Lx<=0
    Lx=input('Enter x dimension (m): ') % plate x dimension m
    if isempty(Lx)
        Lx=0;
    end
    if Lx<=0
        fprintf('Lx must be a positive value')
    end
end

% Y dimension
Ly=0;
while Ly<=0
    Ly=input('Enter y dimension (m): ') % plate y dimension m
    if isempty(Ly)
        Ly=0;
    end
    if Ly<=0
        fprintf('Ly must be a positive value')
    end
end

% Grid spacing X direction
h=0;
while h<=0
    fprintf('The grid spacing must be smaller than %f \n',(Lx/2))
    fprintf('and %f \n',(Ly/2))
    h=input('Enter grid spacing (m): ') %grid spacing m
    if isempty(h)
        h=0;
    end
    if h<=0
        fprintf('h must be a positive value')
    end
end

Nx=round((Lx/h)+1);           %number of nodes on x
Ny=round ((Ly/h)+1);          %number of nodes on y
dx=Lx/Nx;                     % delta x
```



```
dy=Ly/Ny; % delta y

% Time step
dt=0;
dtmin=0;
dtmax=1/(2*alpha*((1/dx^2)+(1/dy^2))); %test the stability condition
while dt<=dtmin || dt>=dtmax
    fprintf('For most accurate results please input time step closest to %f \n',1/(2*alpha*((1/dx^2)+(1/dy^2))))
    dt=input('Enter time step: ');
    if isempty(dt) %Replacing empty value with 0
        dt=0;
    end
    if dt<=dtmin || dt>=dtmax
        fprintf('Error, the stability condition is not met\nPlease choose a "dt" smaller than %f \n',1/(2*alpha*((1/dx^2)+(1/dy^2))))
    end
end

Fo=alpha*dt/h^2 %Fourier number

%% Gaussian Distribution North Boundary
Tf=zeros(Ny,Nx);
normal=sqrt(-((1-((Nx+1)/2))^2)/(2*log(10/60)));
%Normal or Gaussian Distribution

for r=1:Nx
    Tf(1,r)=60*exp(-((r-((Nx+1)/2))^2/(2*normal^2)));
    %North Normal distribution boundary condition
    r=r+1;
end

%% Finite Difference Method till Thermal Equilibrium
Ti=0.1*Tf; %iteration criteria
k=0; %iteration counter
tolerance=0.0001; %temperature tolerance

while max(max(abs(Tf-Ti)))>tolerance
    Ti=Tf;
    for i=2:Ny
        for j=1:Nx
            if j==1 && i<Ny %West Nodes
                Tf(i,j)=Fo*(Ti(i+1,j)+Ti(i-1,j)+Ti(i,j+1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            elseif j==Nx && i==Ny %Bottom Right Node
                Tf(i,j)=Fo*(Ti(i-1,j)+Ti(i-1,j-1)+Ti(i,j-1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            elseif j==1 && i==Ny %Bottom Left Node
                Tf(i,j)=Fo*(Ti(i-1,j)+Ti(i-1,j)+Ti(i,j+1)+Ti(i,j+1)-(4-(1/Fo))*Ti(i,j));
            elseif j==Nx && i<Ny %East Nodes
                Tf(i,j)=Fo*(Ti(i+1,j)+Ti(i-1,j)+Ti(i,j-1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            elseif i==Ny && j>1 && j<Nx %South Nodes
                Tf(i,j)=Fo*(Ti(i-1,j)+Ti(i-1,j)+Ti(i,j+1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            else %Centre Nodes
                Tf(i,j)=Fo*(Ti(i+1,j)+Ti(i-1,j)+Ti(i,j+1)+Ti(i,j-1)-(4-(1/Fo))*Ti(i,j));
            end
        end
    end
end
```

```
j=j+1;
end
i=i+1;
end
error=max(max(abs(Tf-Ti))); %Temperature difference between iterations
k=k+1; %Number of iterations
fprintf('The temperature difference between iterations is the following: %f \n',error);
fprintf('You are on iteration: %f \n',k);
end

fprintf('Below is the matrix at thermal equilibrium')
%Matrix at thermal equilibrium
Tf

%% Contour plot of the heat transfer
figure(1)
contourf (Tf,100,'linecolor','none')
colormap jet
colorbar 'Eastoutside'
caxis([10 60])
title(['Plate at thermal equilibrium'])
view(180,90)

%% Time till thermal equilibrium is reached
Time_seconds=dt*k;
if Time_seconds<3600
    fprintf('Seconds taken to reach thermal equilibrium are: %f \n',Time_seconds)
    %Seconds
elseif Time_seconds>3600 && Time_seconds<86400
    Time_hours=Time_seconds/3600;
    fprintf('Hours taken to reach thermal equilibrium are: %f \n',Time_hours)
    %Hours
elseif Time_seconds>86400
    Time_days=Time_seconds/86400;
    fprintf('Days taken to reach thermal equilibrium are: %f \n',Time_days)
    %Days
end
fprintf('Total amount of iterations to reach thermal equilibrium: %f \n',k)
fprintf('End of Process Reached') %End
```