

University of Wales Trinity Saint David School of Engineering

*Transient Thermal Diffusion Model using the
Finite Difference Method*

*Advanced Computational Methods
M6X00408*

Nicholas David Thomas

1702567

27/01/2020

Contents

1	Introduction	i
2	Equations	ii
3	Finite Difference Method	iii
3.1	Taylor Series	iii
3.2	Central Difference Method	iii
3.3	Forward Difference Method	iv
3.4	Backward Difference Method	v
3.5	Explicit method	v
3.6	Implicit method	vi
4	Boundary Conditions Implemented	vi
4.1	Linear Distribution	vi
4.1	Gaussian Curve	vii
4.2	Whilst Running	viii
5	Script	viii
6	Analysis of Code	xi
6.1	Changing Time Step	xi
6.2	Grid Spacing	xii
6.3	Constant Fourier Number Different Grid Spacing	xiv
6.4	Changing Dimensions	xvi
7	Gaussian Plate	xix
8	Limitations of Model and Improvements	xx
9	Conclusion	xx
10	References	xxi
11	Appendices	xxii

Figure 1-Example of Gaussian Distribution.....	i
Figure 2-Example of the different difference method.....	v
Figure 3-Script to run Linear Distribution.....	vi
Figure 4-Script to Run a Gaussian Distribution.....	vii
Figure 5-Master Script	viii
Figure 6-Script of Input	viii
Figure 7- Script of Checking Grid Spacing.....	ix
Figure 8-Menu for Type of Distribution.....	ix
Figure 9-Script for creating base plate	ix
Figure 10-Script to calculate Fourier number.....	x
Figure 11-Results from Changing Time Step.....	xi
Figure 12-Results from changing Grid Spacing	xiii
Figure 13-Results from Constant Fourier number and changing Grid Spacing	xv
Figure 14-Results from Change in Dimensions	xviii
Figure 15-Results from Gaussian Plate	xix

1 Introduction

The problem investigated within this report is of the time-dependant heat propagation across a two-dimensional plate. The plate is split into finite elements evenly distributed across it with a spacing of h and is solved in the computer programme MATLAB (Mathworks, 2020). The problem is based on an aluminium plate which starts at 0°C which then has a heat source which has a Gaussian distribution of heat, ranging from 60°C to 10°C . To add to that the edges of the plate are also insulated to negate the effects of heat loss or gain from the sides. The plate has a thermal conductivity (k) of 143 W/m.C , density (ρ) of 2.8×10^3 and a specific heat capacity (c) of 795 J/kg.C . The plate will be solved using cartesian format using the explicit method. To validate the equations used within the script there will also be a plate which has a linearly spaced heat on the boundary conditions as this mean that there should be clearly defined temperatures in each column. This plate will be analysed so to make sure it works in different conditions to be sure that the gaussian results will be what is expected.

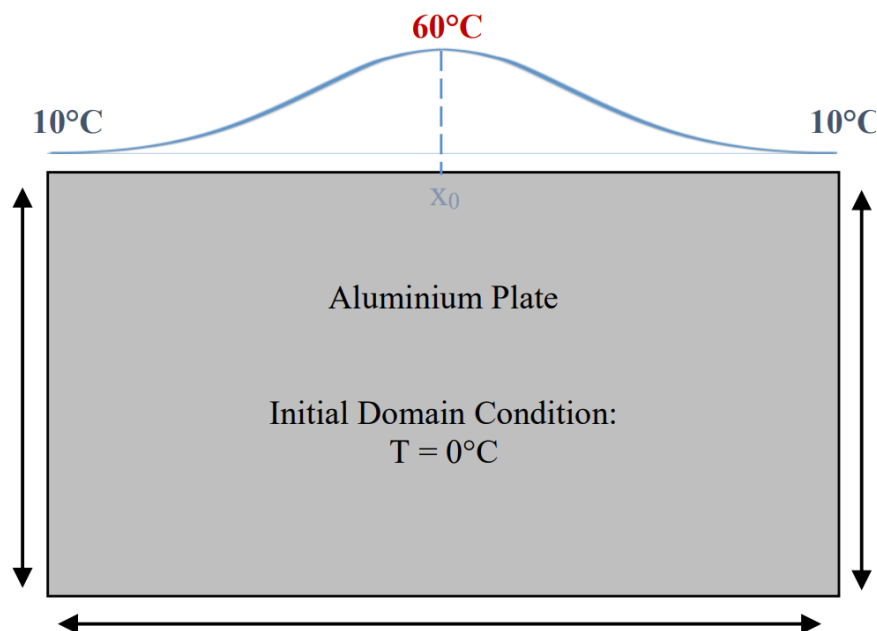


Figure 1-Example of Gaussian Distribution

2 Equations

$$\alpha = \frac{K}{\rho C}$$

Equation 1- Thermal Diffusivity

- α =Thermal Diffusivity (m^2s^{-1})
- K =Conductivity ($\text{Wm}^{-1}\text{k}^{-1}$)
- P =Density (kgm^{-3})
- C =Specific Heat Capacity (Jkgk^{-1})

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

Equation 2- Fourier Number

- F_o =Fourier Number
- α =Thermal Diffusivity (m^2s^{-1})
- t =Time Step (s)
- h =Grid Spacing (m)

$$\nabla^2 T = \frac{1}{\alpha} \times \frac{\partial T}{\partial y}$$

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{\alpha} \times \frac{\partial T}{\partial y}$$

$$T_p^{t+\Delta t} = F_o \left(T_E^t + T_W^t + T_N^t + T_S^t - \left(4 - \frac{1}{F_o} \right) T_p \right)$$

Equation 3- Heat Transfer Equation

- T_p =Temperature at a point P
- F_o =Fourier Number
- T_z =Temperature at position North, East, South, West

3 Finite Difference Method

The author used a time-dependent thermal diffusion model using the finite difference method. This model is used in solids or fluids when they are not moving. For this to happen the systems have to be in direct contact so that conduction and therefore heat transfer can occur. Conduction in solids can be represented with the following equation.

$$\nabla^2 T = \frac{1}{\alpha} \times \frac{\partial T}{\partial y}$$

If you then substitute you then get this term:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{\alpha} \times \frac{\partial T}{\partial y}$$

The FDM method in itself converts differential equations into a number of equations which can be calculated using matrix algebra (Lewis, et al., 1996). Because it uses matrix algebra this method is best used using a software such as MATLAB as to hand calculate it would take a very long time and would have human errors introduced.

3.1 Taylor Series

The Taylor series is used as an expansion function into an infinite sum of terms (mathisfun, 2017) that are calculated from the derivatives at another point. Essentially it makes an estimate of what a function will look like by taking derivatives of the function then adding them together and therefore simplifying a partial differential equation. Theoretically if there was an infinite number of derivatives then the exact function can be found (Bi, 2018).

$$f(x) = F(0) + f'(0)x + \frac{f''(0)}{2!}x^2 \dots + \frac{f^{(n)}(0)}{n!}x^n$$

Equation 4-Taylor Series

This can be written in a more compact version by using summation:

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

Equation 5-Summation of Taylor Series

3.2 Central Difference Method

The central difference method is used in this case to simplify the Taylor series in respect to grid spacing. It does this by taking values from the sides of a point to estimate the value of that point. So, in this case it will be North, East, South, West. In this model the author used

this to the second derivative. However, because this equation makes an estimate this does introduce errors. These errors seen in red are the truncation errors, in this case the second order.

Raw Equation:

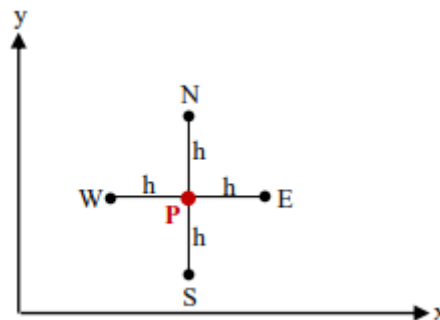
$$f'(x_0) \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} - f''(x_0) \frac{h^2}{3!} - f^{(5)}(x_0) \frac{h^4}{5!}$$

Equation 6- Central Difference Method Equation

For our purpose:

$$\frac{\partial^2 T_p}{\partial x^2} = \frac{T_e^t - 2 \times T_p^t + T_w^t}{h^2}$$

Equation 7-Central Difference Method for this case



3.3 Forward Difference Method

The forward difference method calculates the value by taking a value in front (x_0) and then calculating the value by adding the step to the original point then taking away the value of the original point then dividing by the step (equation below).

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

Equation 8-Forward Difference Method

Using the Taylor Series:

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

Equation 9-Forward Difference Method Adapted with the Taylor Series

For this model it uses this method for calculating the change in time e.g. what will the values be in t seconds.

$$\frac{\partial T}{\partial t} \approx \frac{T_p^{t+\Delta t} - T_p^t}{\Delta t}$$

3.4 Backward Difference Method

The backward method is the same as the forward just in reverse (equation below). In this case this method is not used as it is not a variable that is needed as the heat propagation will always move forward in time.

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

Equation 10-Backward Difference Method

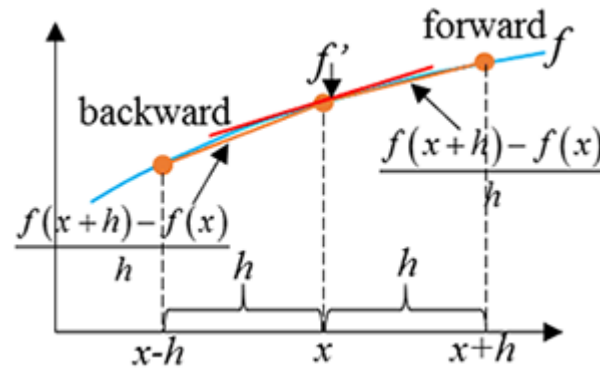


Figure 2-Example of the different difference method

(Liu, 2018)

3.5 Explicit method

Using the equations from central and forward difference methods to calculate in terms of grid spacing and time (Bui, 2009). This method calculates using values which are independent of other values. This means that there is only one variable to calculate at each iteration. This is the method the author is using as this means that it is computationally inexpensive.

However, it does have its drawbacks as it is less accurate than other methods as it introduces errors along the way. To add to that it is limited on what time steps it can use as the Fourier number must be below 0.25 (Croft & Stone, 1977) for the equation to work as any higher and it would have issues with stability. Below is the equation with only one unknown (in red).

$$\frac{T_E^t - 2 \times T_p^t + T_W^t}{h^2} + \frac{T_N^t - 2 \times T_p^t + T_S^t}{h^2} = \frac{1}{\alpha} \times \frac{T_p^{t+\Delta t} - T_p^t}{\Delta t}$$

$$T_p^{t+\Delta t} = \frac{\alpha \Delta t}{h^2} \left(T_E^t + T_W^t + T_N^t + T_S^t - \left(4 - \frac{1}{F_o} \right) T_p^t \right)$$

$$F_o = \frac{\alpha \Delta T}{h^2}$$

Final Equation:

$$T_p^{t+\Delta t} = F_o \left(T_E^t + T_W^t + T_N^t + T_S^t - \left(4 - \frac{1}{F_o} \right) T_p^t \right)$$

3.6 Implicit method

This is much more accurate but computationally more expensive. It is also more stable so larger time steps can be used in comparison to explicit. This is because over large time periods there is an issue with stability criterion. If this case it was done with implicit the neighbouring points around the value at the new time you want would need to be calculated. Explicit just needs the neighbouring temperatures only at the old-time step. This is why, it is so computationally more expensive as it has to do these calculations simultaneously (De Bortoli, et al., 2015).

Equation for heat transfer if implicit method used:

$$-\frac{1}{F_0}T_p^n = T_0^{n+1} + T_1^{n+1} + T_E^{n+1} + T_E^{n+1} + T_W^{n+1} + T_N^{n+1} + T_S^{n+1} - \left(6 + \frac{1}{F_0}\right) \cdot T_P^{n+1} + \frac{h^2 \cdot Q'}{k}$$

Equation 11-Heat Transfer equation if using Implicit Method

4 Boundary Conditions Implemented

4.1 Linear Distribution

One method of making the boundary conditions was the linear distribution. This was used as a test case to check the validity of the equations used. Essentially the base case is where the temperature are evenly distributed between the two sides as seen below.

```
%Creating T0 using even distribution
cr=linspace(Htemp,0,xpoint);
%Putting the rest of the boundary conditons in
To(1,:)=cr;
To(end,:)=cr;
To(:,1)=Htemp;
```

Figure 3-Script to run Linear Distribution

50	45	40	35	30	25	20	15	10	5	0
50	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0
50	45	40	35	30	25	20	15	10	5	0

Table 1- Example of Linear Distributed Plate

4.1 Gaussian Curve

A Gaussian Curve also known as a bell curve is a normalized frequency distribution which is symmetrical about a point (Morris & Langari, 2012). It uses the equation below.

$$T(x) = Ae^{\frac{-(x-x_0)^2}{2\sigma^2}}$$

Equation 12- Gaussian Distribution Equation

Using this the author then had to calculate the sigma value by rearranging the above equation.

$$\frac{T(x)}{A} = e^{\frac{-(x-x_0)^2}{2\sigma^2}}$$

$$\ln\left(\frac{T(x)}{A}\right) = \frac{-(x-x_0)^2}{2\sigma^2} \ln e$$

$$\ln\left(\frac{T(x)}{A}\right) = \frac{-(x-x_0)^2}{2\sigma^2}$$

```

Ltemp=input('Define Lower Temperature: ');
%Finds standard deviation (Sigma)
Xo=(xpoint+1)/2;
sig=sqrt(((xpoint-Xo)^2)/(2*log(Ltemp/Htemp)));
%Calculating T0 using gaussian distribution
row=1;
while row<=xpoint
    To(1,row)=(Htemp*exp(-(row-Xo)^2/(2*(sig^2))));
    row=row+1;
end
end

```

Figure 4-Script to Run a Gaussian Distribution

$$\sigma = \sqrt{\frac{-(x-x_0)^2}{2 \ln\left(\frac{T(x)}{A}\right)}}$$

Equation 13- Sigma Calculation

4.2 Whilst Running

Whilst running, the Gaussian plate has a slightly different set of boundary conditions. As on the edges when it is calculating there is no value on the other sides to calculate. Therefore, to make the model work the edges have their own calculation so that it takes the value from the opposite side that it is not using the 0 value affecting the result.

10.0000	19.0605	31.4787	45.0450	55.8503	60	55.8503	45.0450	31.4787	19.0605	10.0000
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Table 2- Gaussian Distribution

5 Script

The script used has one master script this runs each other script in order. Doing this has meant that each step off the system can be split up nicely.

Figure 5-Master Script

The values script is most of the inputs. So, it asks what the dimension of the plate and what grid spacing.

```
X make X axis hot side if gaussian. If even distribution make Y hot side
Enter X axis Dimension (m):
Enter Y axis Dimension (m):
Enter Grid Spacing:
```

Figure 6-Script of Input

It then goes on if the grid spacing does not divide equally into the x and y dimensions then it will ask to re-enter the value. It does this by dividing the dimensions by the grid spacing then checks if it is an integer by using the rem function. If it is not, then it will ask to re input value for grid spacing.

<pre>%% Checking grid spacing is correct if rem(xaxis,h)==0 && rem(yaxis,h)==0 To=zeros(ypoint,xpoint); else warning('Number does not divide evenly in') h=input('Re-input Grid Spacing: '); xpoint=(xaxis/h)+1; ypoint=(yaxis/h)+1; To=zeros(ypoint,xpoint); end</pre>	<pre>%Making sure that grid spacing is correct %making grid of zeroes %Brings up a warning %Inputs the grid spacing %Makes a table of 0's the same size as the number of points</pre>
---	---

Figure 7- Script of Checking Grid Spacing

After this the script brings up a pop-up menu of what type of distribution you want, either even (linear distribution) or Gaussian distribution. Whatever is picked will then have some constants made and will be asked what the highest temperature will be. Once this is decided it goes onto making the plate at time zero

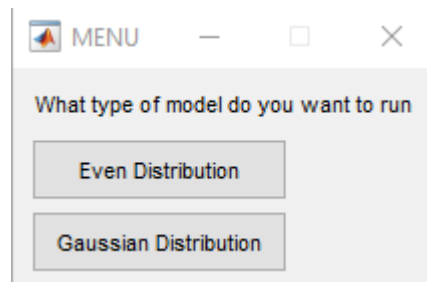


Figure 8-Menu for Type of Distribution

<pre>%% Creating Plate base values % Making an if statement to select which model you want to use and then creating it % Choosing which type of distribution to use if D==1 %Creating T0 using even distribution cr=linspace(Htemp,0,xpoint); %Putting the rest of the boundary conditions in To(1,:)=cr; To(end,:)=cr; To(:,1)=Htemp; else Ltemp=input('Define Lower Temperature: '); %Finds standard deviation (Sigma) Xo=(xpoint+1)/2; sig=sqrt(((xpoint-Xo)^2)/(2*log(Ltemp/Htemp))); %Calculating T0 using gaussian distribution row=1; while row<=xpoint To(1,row)=(Htemp*exp(-((row-Xo)^2)/(2*(sig^2)))); row=row+1; end end clear xaxis clear yaxis clear row</pre>	<pre>%Getting user to define lower temperature</pre>
--	--

Figure 9-Script for creating base plate

Initially the script makes a grid of zeroes which depending on the distribution will then add the rest of then values. If the gaussian distribution was selected an extra input will be asked of adding a lower temperature.

As can be seen the script also calculates the sigma value before the plate starts to be created. Once the base plate is created it then goes onto calculating the Fourier number. It does this by first asking for the time step and then calculating alpha, then the Fourier number. If the value of the Fourier number is above 0.25 a warning appears, and the script will not continue without the correct time step being entered. Once the value is re-entered the Fourier number is calculated again.

<pre> tstep=input('Enter Time Step(s): '); a=K/(rho*c); Fo=((a*tstep)/(h^2)); %% Checking if the Fourier number is below 0.25 if not re-enter time step and recalculates while Fo>0.25 warning('Fourier number too small retry!') tstep=input('Redefine time step: '); Fo=((a*tstep)/(h^2)); end </pre>	<pre> %Calculates the Alpha value %Fourier Number Calculation </pre>
--	--

Figure 10-Script to calculate Fourier number

After this it then goes on to the calculations as mentioned earlier. The way the script works is that the calculation will keep going until the exit criteria is met. It will keep going until there is not change which is less than 0.9999 difference. It does this by if that node does meet the criteria then it will put a 1 in stopcheck. Once all of the stopcheck is filled it will stop the calculation. This works on both distributions but the gaussian has extra stop checks as there are extra calculations for the edges and the corners. At each iteration the script will also display the plate so that the heat transfer can be seen over time. Both of these can be seen in appendices.

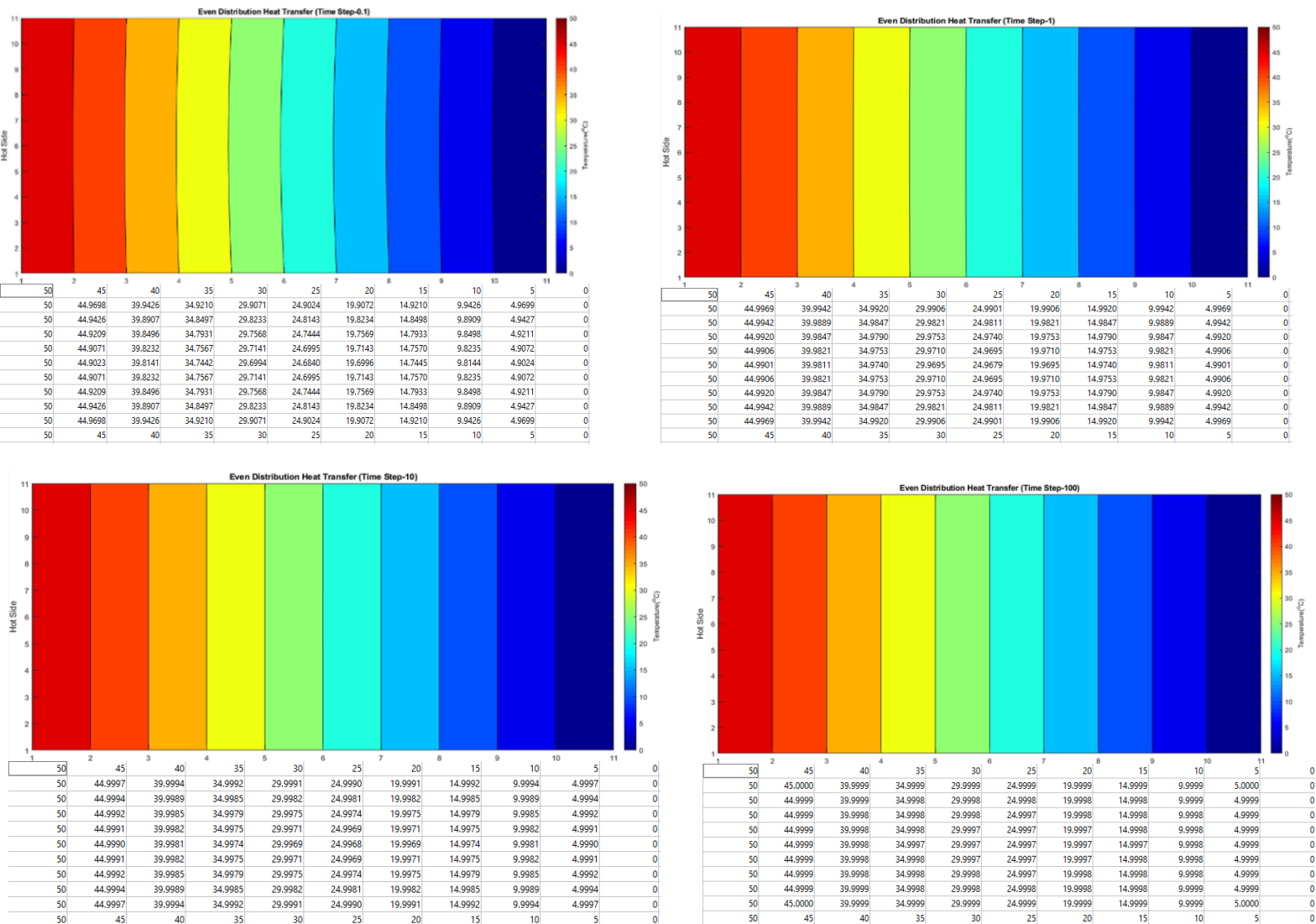
6 Analysis of Code

In this section the author will analyse the effect of changing variables in the code. This is to test that the calculations are still accurate when changing variables such as time step, grid spacing and dimensions. The author expects the results to be neat columns which are the same value all along it.

6.1 Changing Time Step

Time Step	Gid Spacing	Time To Equilibrium	No. Iterations	X Dimension	Y Dimension
0.1	0.5	96159.6	961596	5	5
1	0.5	141615	141615	5	5
10	0.5	187320	18732	5	5
100	0.5	232700	2327	5	5
1000	0.5	0	0	5	5

Table 3-Changing Time Step



Warning: Fourier number too large retry!

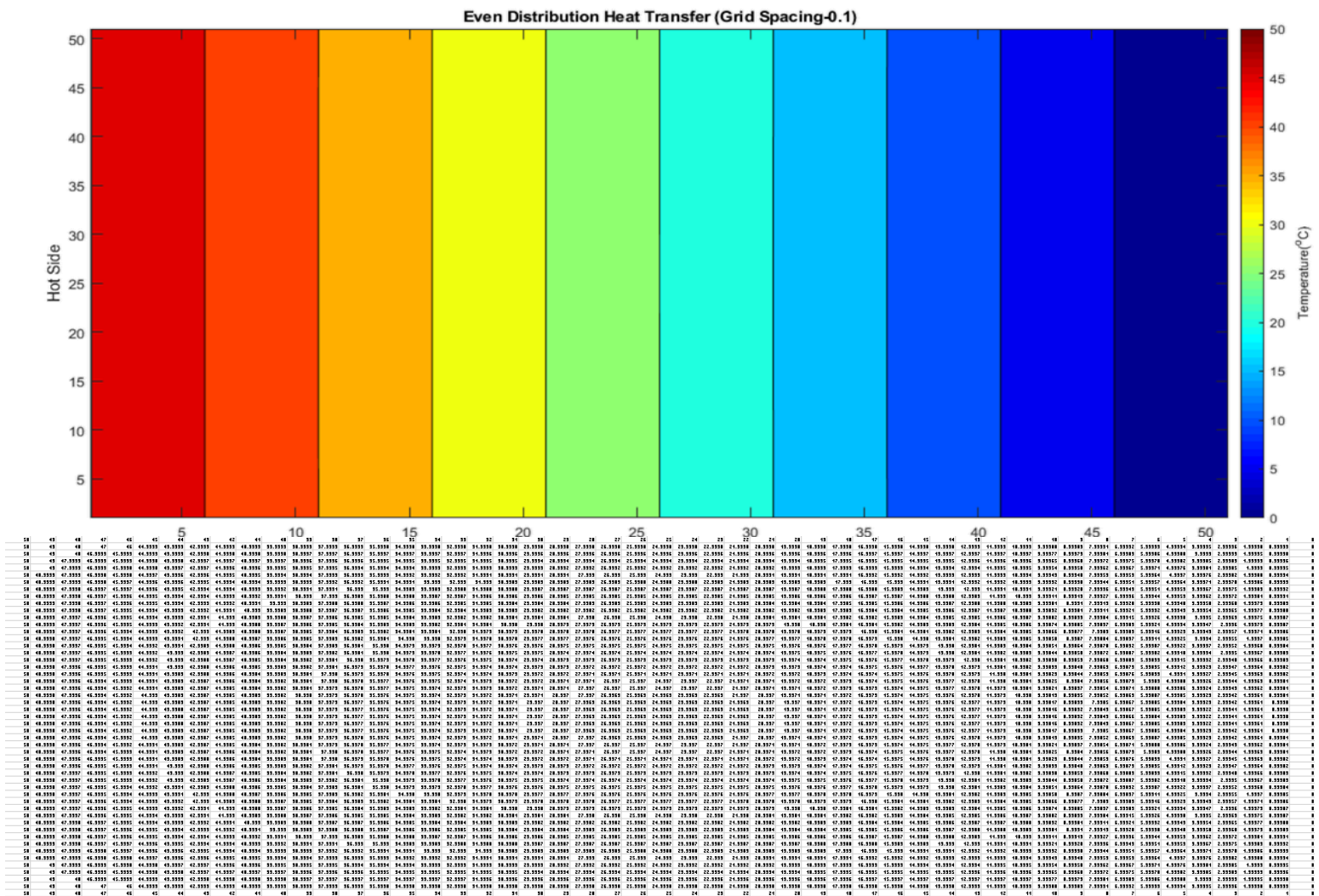
Figure 11-Results from Changing Time Step

As can be seen the higher the time step the more accurate the model becomes. This is down to the errors introduced by each iteration therefore with less iterations with larger time steps the more accurate it is. Also, when the Fourier number is too large (above 0.25) then a warning works to show that the time step needs to be changed to make it work. When the time step is at 0.1 the error is at its maximum. So much so that the shape of the distribution had started to warp. To get over this the stop criterion could be extended to more decimal places, but this would extend the calculation time.

6.2 Grid Spacing

Grid Spacing	Time Step	Time To Equilibrium	No. Iterations	X Dimension	Y Dimension
0.1	10	186630	18663	5	5
0.5	10	187320	18732	5	5
1	10	188440	18844	5	5
5	10	0	0	5	5

Table 4-Change in Grid Spacing



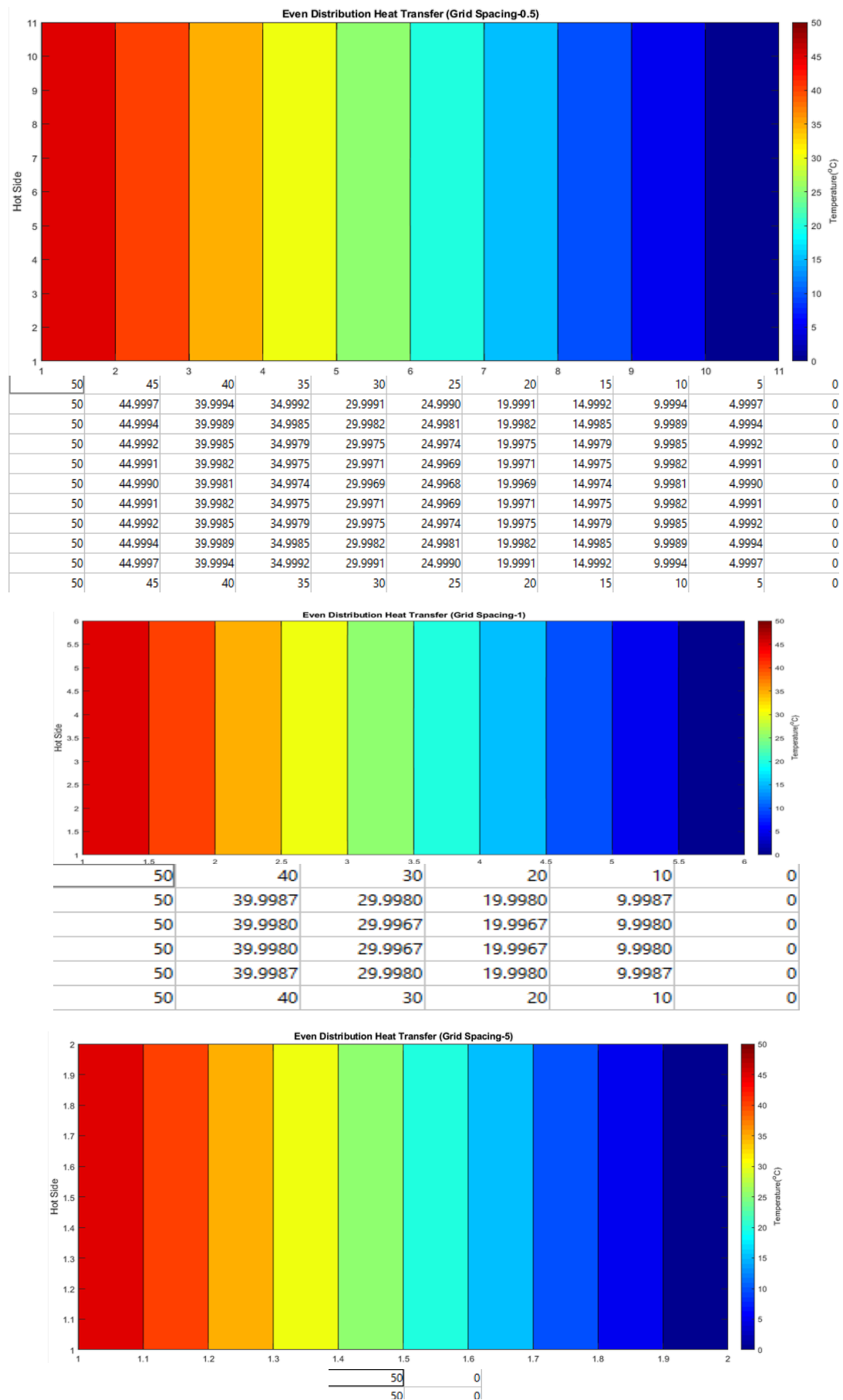


Figure 12-Results from changing Grid Spacing

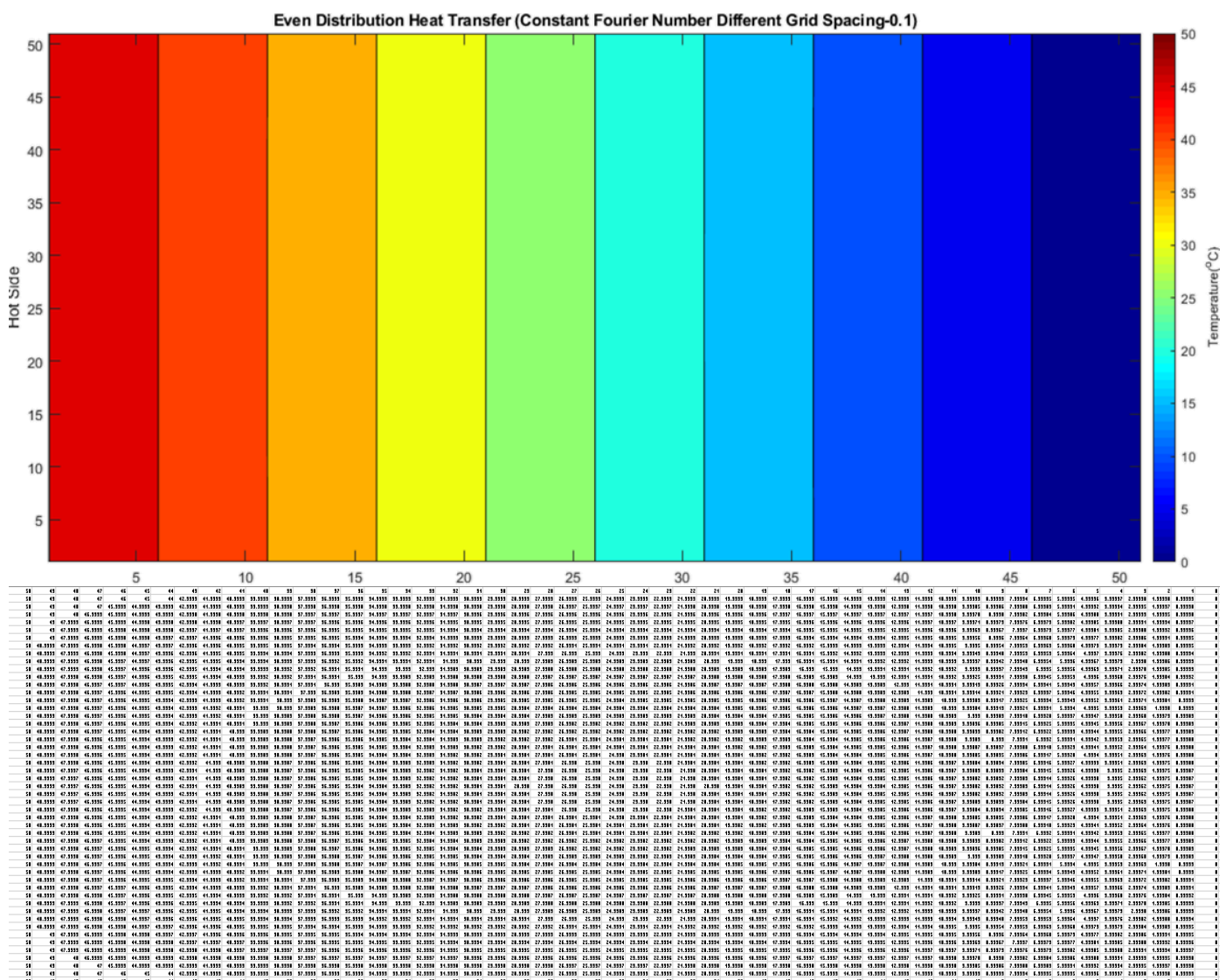
The accuracy of the model does not change all that much when changing the grid spacing.

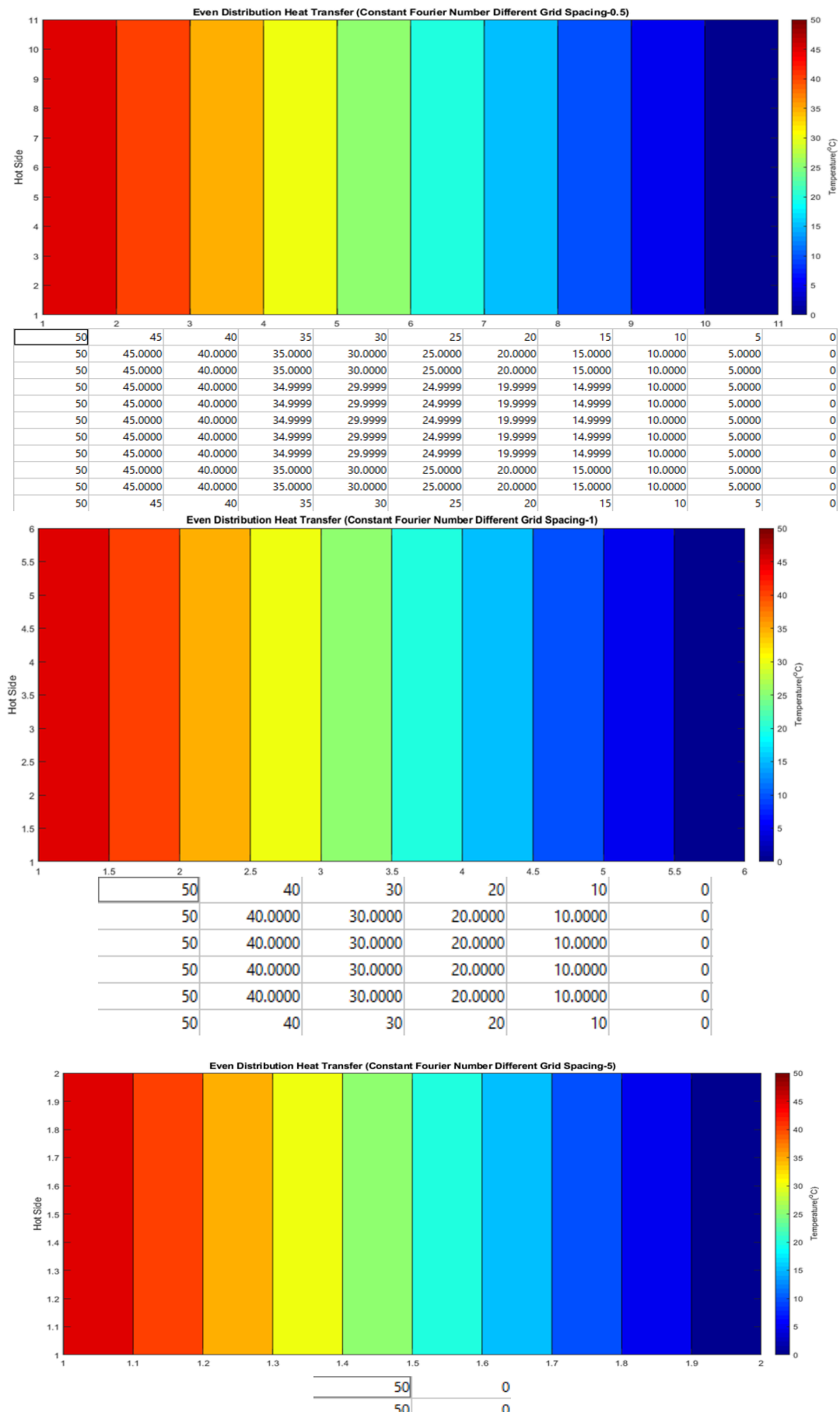
This is down to it being the part of the equation which is derived from the central difference method rather than the forward. The central difference method is more accurate than forward as it uses more points of data to calculate the answer rather than just one. However, interestingly the time to get to equilibrium increased as the grid spacing is getting larger.

6.3 Constant Fourier Number Different Grid Spacing

Fo=0.1					
Grid Spacing	Time Step	Time To Equilibrium	No. Iterations	X Dimension	Y Dimension
0.1	15.57	195341.22	12546	5	5
0.5	389.16	258013.08	663	5	5
1	1556.64	281751.84	181	5	5
5	38916.08	38916.08	1	5	5

Table 5-Change in Constant Fourier Number and changing Grid Spacing





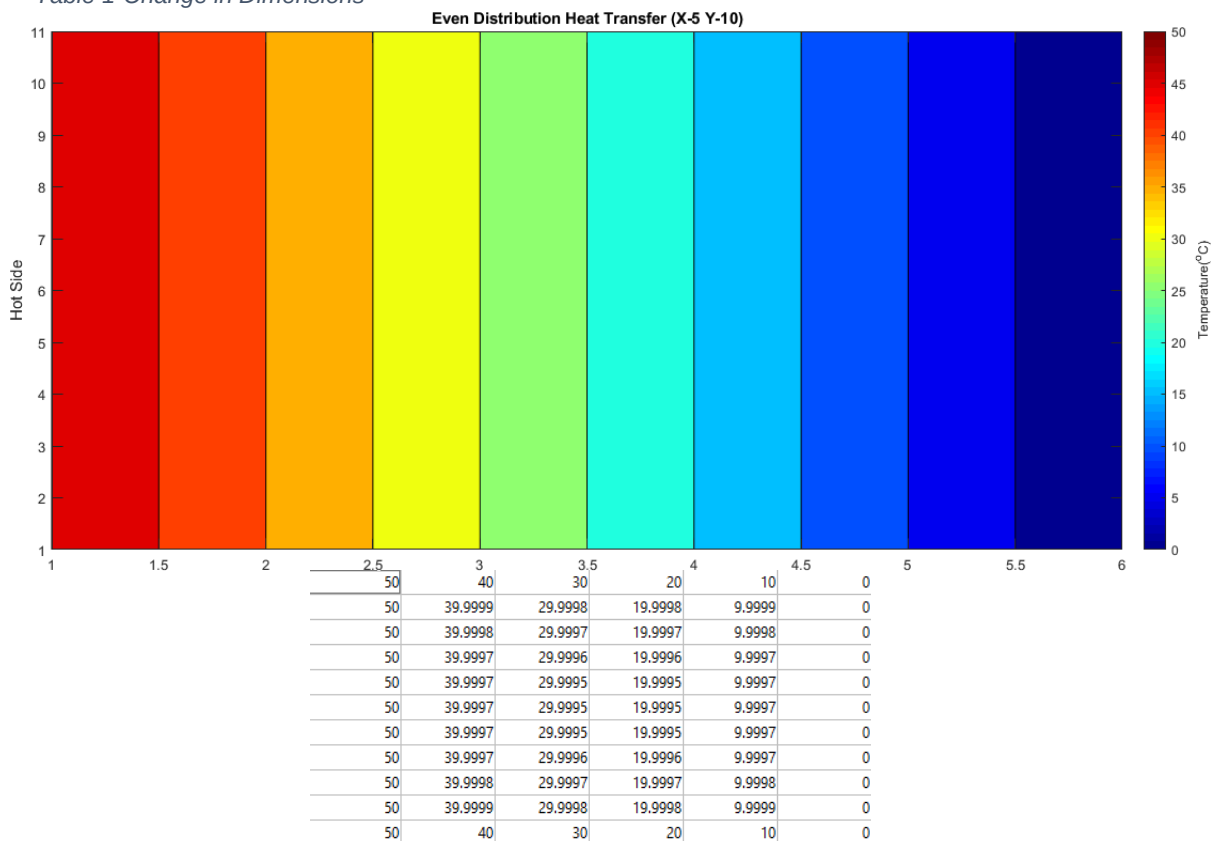
As can be seen the accuracy of the model is the same through the runs except when the grid spacing is 0.1. The accuracy being the same is expected as the Fourier number is the main affecter of the accuracy as it is derived from the forward difference part. The 0.1 being inaccurate is an interesting part. This may be down to the fact that the actual time step is a reoccurring number and the author rounded it so there may be a slight error introduced there.

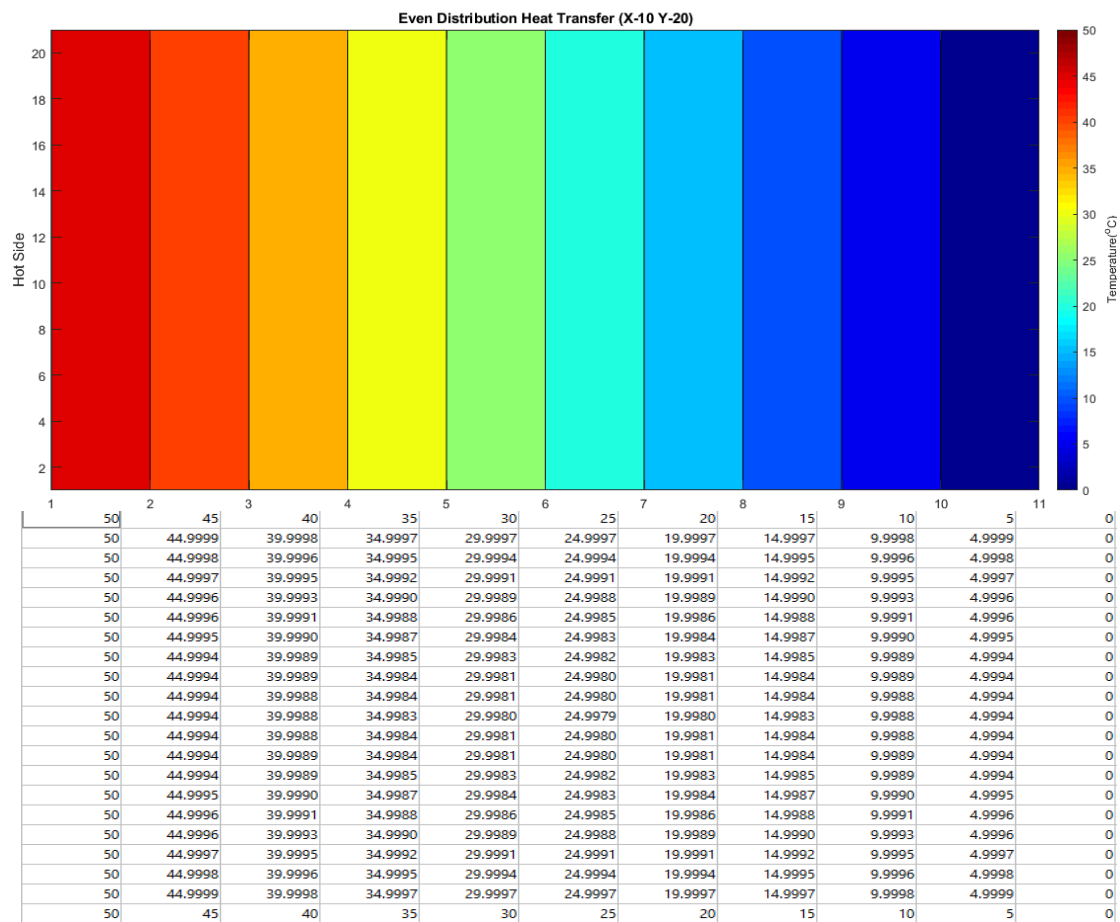
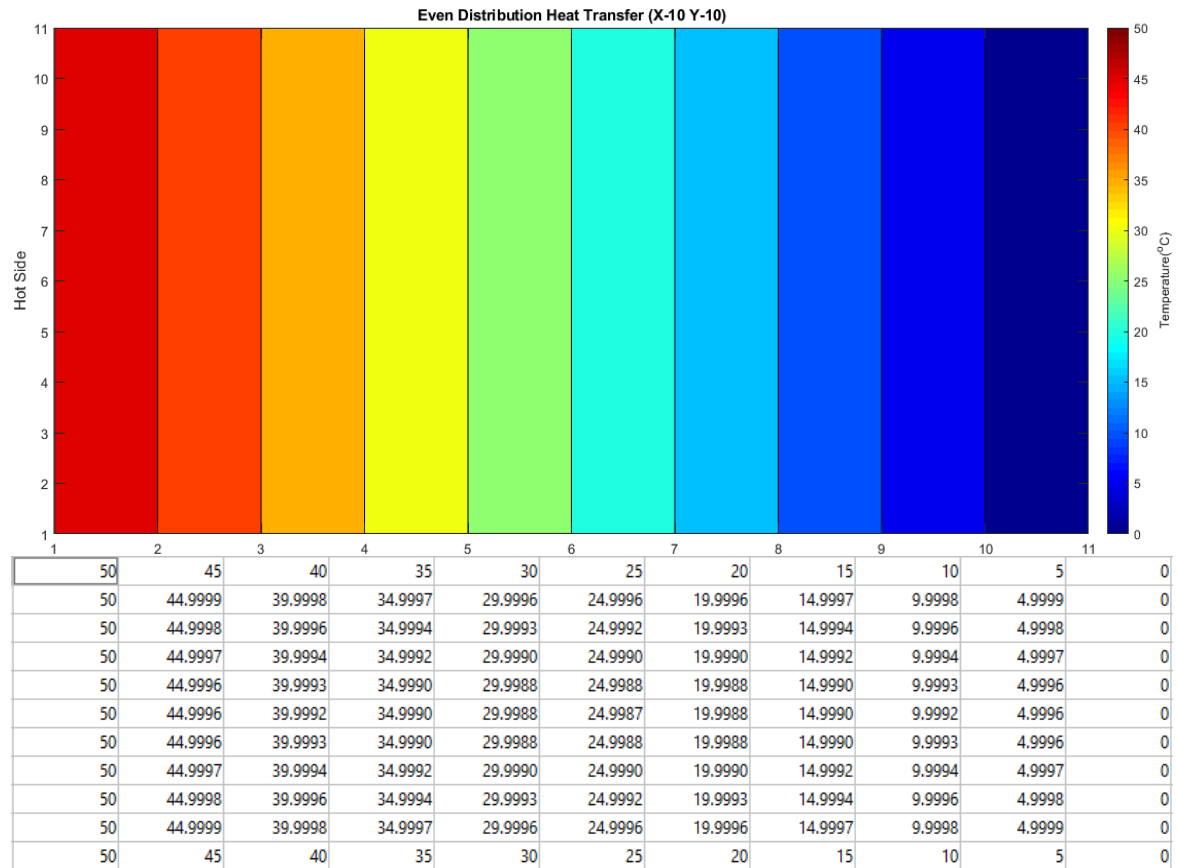
6.4 Changing Dimensions

For ease of comparison the grid spacing is kept as 1 as it will work for all sizes. To add to that the time step is kept at 100 as this is a fairly safe value which will mean that the Fourier number is not too large.

X Dimension	Y Dimension	Time To Equilibrium	No. Iterations	Grid Spacing	Time Step
5	10	361700	3617	1	100
10	10	821900	8219	1	100
10	20	1254900	12549	1	100
20	20	2840200	28402	1	100

Table 1-Change in Dimensions





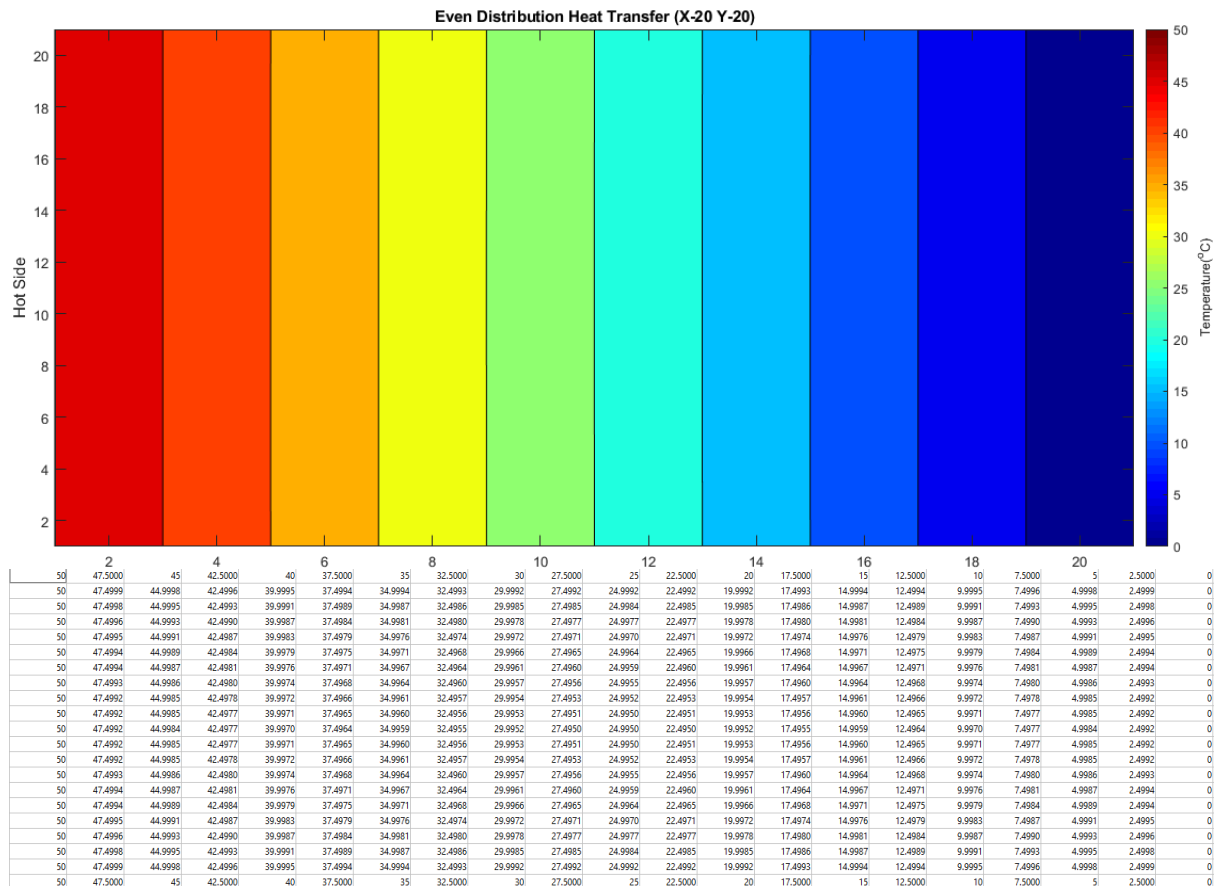


Figure 14-Results from Change in Dimensions

As can be seen the accuracy does not change all that much. This is because the Fourier number is staying constant. However, expectedly the time to reach equilibrium is increasing as the plate size increases. This is because there is more plate for the heat to propagate across. To add to that the calculation time also increased as plate size increased as expected as there is a need for more calculations as there are more elements. The fact that the accuracy does not change is a good thing as it means that the model works in different plate scenarios.

7 Gaussian Plate

As the code was seen to be accurate for the linear distribution plate the author assumes that it will be accurate for the Gaussian plate.

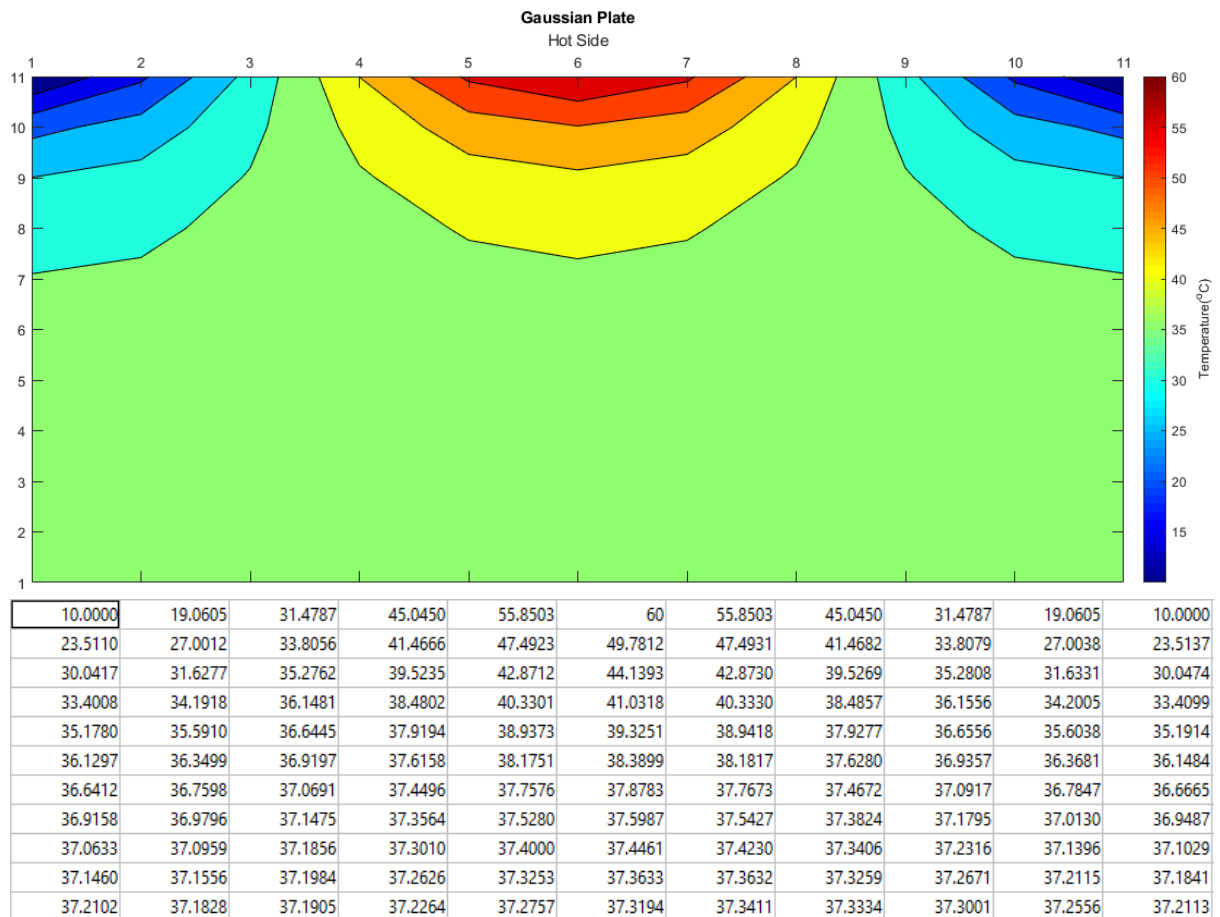


Figure 15-Results from Gaussian Plate

As can be seen this is not fully symmetrical whereas the author would expect it to be fully symmetrical. This maybe down to the calculations starting at one end of the plate and moving across. So, as the calculation works across it introduces errors into the code build up. However, it maybe that the code on the edges are incorrect and are introducing errors. As the author assumed that the values on the outside of the plate are the same as the inside this may have introduced the error. However, the inaccuracies are to 1 decimal place. This suggests that for a basic model it works but if more precision is needed the model will have to be adapted.

8 Limitations of Model and Improvements

One limitation of the model is the Fourier number as it has to be below 0.25 so that the model is accurate. This then encompasses the grid spacing and time step as these both have an effect on the Fourier number these are limited into a certain range. To add to that the grid spacing is also limited as the grid spacing must divide equally into the dimensions. This means that if the plate is an awkward value such as with decimal places then it will not be able to run. A way of overcoming this is maybe have a different grid spacing for the X and Y dimensions so that if the plate is still a whole number then it will be able to make sure it divides equally in both directions. To add to that there could also be a bit of code which if the plate is a decimal place then it can round the value of the plate so that the plate will be able to divide easier.

Another limitation is the type of model itself. As the explicit method is used there is naturally going to be error within the calculations at each iteration. If an implicit method were used, then it could mean that there would be a massive reduction in error. However, the calculations for this would be much more complex so the running time would be increased and to chase that 0.0001°C would be questionable if it was worth it. Also, as the most accurate thermometers can only measure to 0.01°C (Aron, 2013) there would be no way to validate the results to that level of accuracy. However, the model could be more accurate if the stop criterion went on to more decimal places, but this would increase computer processing time.

9 Conclusion

In conclusion the author has found that the model is accurate to at least 2 decimal places. The author believes this is close enough as a thermometer is only able to measure to this. However, it could be improved by using the implicit method to overcome the issues of accuracy. There could also be improvements on the grid spacing by being able to make it so that it can split more complicated dimensions. But the writer believes that the equations work well enough to believe the results of the gaussian distribution as well as the linear distribution.

10 References

- Aron, J., 2013. *Newscientist*. [Online]
Available at: <https://www.newscientist.com/article/mg21929254-000-musical-argon-is-most-accurate-thermometer-ever/>
[Accessed 13 January 2020].
- Bi, Z., 2018. *Finite Element Analysis Applications*. s.l.:Academic Press.
- Bui, T., 2009. *Explicit and Implicit Methods In*. s.l.:Uconn Library.
- Croft, D. & Stone, J., 1977. *Heat Transfer Calculations using Finite Difference Equations*. Sheffield: Applied Science Publishers.
- De Bortoli, A. L., Andreis, G. S. & Pereira, F. N., 2015. Chapter 6- Numerical Methods for Reactive Flows. In: *Modeling and Simulation of Reactive Flows*. s.l.:Elsevier, pp. 123-169.
- Lewis, R., Morgan, K., Thomas, H. & Seetharamu, K., 1996. *The Finite Element Method in Heat Transfer Analysis*. Chichester: John Wiley & Sons Ltd.
- Liu, D. Z., 2018. *multiphysics*. [Online]
Available at: <https://www.multiphysics.us/FDM.html>
[Accessed 14 January 2020].
- mathisfun, 2017. *mathisfun.com*. [Online]
Available at: <https://www.mathsisfun.com/algebra/taylor-series.html>
[Accessed 07 January 2020].
- Mathworks, 2020. *Mathworks*. [Online]
Available at: <https://uk.mathworks.com/products/matlab.html>
[Accessed 16 January 2020].
- Morris, A. S. & Langari, R., 2012. *Measurement and Instrumentation Theory and Application*. s.l.:Elsevier Inc.

11 Appendices

11.1 Master Script

```
clear
clc
close all
%% Runs each .m file in order
values
T0
fourier_number
calculation_g
calculation_e
```

11.2 Values Input

```
%% Inputs
% Getting User to define dimensions and time steps
disp('X make X axis hot side if gaussian. If even distribution make Y hot side') % Displays Text
xaxis=input('Enter X axis Dimension (m): ');
%getting user to define x axis
yaxis=input('Enter Y axis Dimension (m): ');
%getting user to define y axis
h=input('Enter Grid Spacing: ');
%Getting user to enter grid spacing
xpoint=(xaxis/h)+1;
%Creates the number of points on the x-axis
ypoint=(yaxis/h)+1;
%Creates the number of points on the Y-axis
%% Checking grid spacing is correct
if rem(xaxis,h)==0 && rem(yaxis,h)==0
%Making sure that grid spacing is correct
    To=zeros(ypoint,xpoint);
%making grid of zeroes
else
    warning('Number does not divide evenly in')
%Brings up a warning
    h=input('Re-input Grid Spacing: ');
%Inputs the grid spacing
    xpoint=(xaxis/h)+1;
    ypoint=(yaxis/h)+1;
    To=zeros(ypoint,xpoint);
%Makes a table of 0's the same size as the number of points
end
%% More inputs
D=menu('What type of model do you want to run','Even Distribution','Gaussian Distribution'); %Brings up a selection menu to choose which type of plate to run
Htemp=input('Define Highest Temperature: ');
%Inputs the highest temperature
%% Constants
K=143;
rho=2800;
c=795;
```

11.3 Fourier Number Calculation

```
tstep=input('Enter Time Step(s): ');
a=K/(rho*c);
%Calculates the Alpha value
```

```

Fo=((a*tstep)/(h^2));
%Fourier Number Calculation
%% Checking if the Fourier number is below 0.25 if not re-enter time step
and recalculates
while Fo>0.25
    warning('Fourier number too large retry!')
    tstep=input('Redefine time step: ');
    Fo=((a*tstep)/(h^2));
%Fourier Number Calculation
end

```

11.4 Equal Spacing Calculation

```

clc
set(0,'defaultfigurewindowstyle','docked')
if D==1
    Tnew=To;
    stopcheck=ones([ypoint,xpoint]); %Creates table of 1's
    stopcheck(2:end-1,2:end-1)=0; %Making the edges of the plate 0's
    exitcriteria=ones([ypoint,xpoint]); %Creates table of 1's
    stopper=0;
    stop=0.9999999;
    Niteration=0;
    %% Calculation
    while sum(stopcheck,'all')<sum(exitcriteria,'all') %Runs until the sum
of the stopcheck < exitcriteria
        Told = Tnew;
        for row=2:ypoint-1 %Row Loop
            for col=2:xpoint-1 %Column Loop
                Tnew(row,col)=Fo*(Told(row-1,col)+Told(row+1,col) ...
                    +Told(row,col-1)+Told(row,col+1)-(4-
(1/Fo))*Told(row,col)); %Actual Calculation

                stopper=Told(row,col)/Tnew(row,col);
%Checking if there is a difference between old and new plates
                if stopper>=stop
                    stopcheck(row,col)=1;
                    % When the difference is smaller than 0.99999 it
changes that cell in stop check to 0 once the total of the stopchek is
higher than the exitcriteria.
                    % Then the iterations finish
                end
            end
        end
    end
%End of 2nd IF Statement

    end
%End of Col for
end
%End of Row for
%% Plotting
%showing the plate changing over time
contourf(flipud(Tnew))
title('Even Distribution Heat Transfer')
ylabel('Hot Side')
c=colorbar;
c.Label.String='Temperature(^{o}C)';
%Labeling colourbar
colormap(jet)
%Setting Colourmap
drawnow

```

```

        Niteration=Niteration+1;
    end
%End of while

```

```

end
%End of 1st IF Statement

```

11.5 Calculation for Gaussian Plate

```

clc
if D==2
%If the menu clicked gaussian goes to this place
    Tnew=To;
    stopcheck=ones([ypoint,xpoint]);
%Creates table of 1's
    stopcheck(2:end-1,2:end-1)=0;
%Making the edges of the plate 0's
    exitcriteria=ones([ypoint,xpoint]);
%Creates table of 1's
    stopper=0;
    stop=0.9999999;
    co=1;
    ro=1;
    Niteration=0;
    %% Calculation
    tic
    while sum(stopcheck,'all')<sum(exitcriteria, 'all')
%Runs until the sum of the stopcheck < exitcriteria
        Told = Tnew;

        for row=2:ypoint-1
%row Loop
            for col=2:xpoint-1
%column Loop
                %Calculation for middle
                Tnew(row,col)=Fo*(Told(row-1,col)+Told(row+1,col) ...
                    +Told(row,col+1)+Told(row,col-1)-(4-(1/Fo)) ...
                    *Told(row,col));

                %Calculation for Right edge
                for ro=2:ypoint-1
                    Tnew(ro,end)=Fo*(Told(ro-1,end)+Told(ro+1,end) ...
                        +Told(ro,end-1)+Told(ro,end-1)-(4-(1/Fo)) ...
                        *Told(ro,end));
                    %Checks if it has stopped changing
                    stopper=Told(ro,end)/Tnew(ro,end);
                    if stopper>=stop
                        stopcheck(ro,end)=1;
                    end
                end
                %Calculation for Left edge
                for ro=2:ypoint-1
                    Tnew(ro,1)=Fo*(Told(ro+1,1)+Told(ro-1,1)+Told(ro,2) ...
                        +Told(ro,2)-(4-(1/Fo))*Told(ro,1));
                    %Checks if it has stopped changing
                    stopper=Told(ro,1)/Tnew(ro,1);
                    if stopper>=stop
                        stopcheck(ro,1)=1;
                    end
                end
            end
        end
    end
end

```

```

%calculation for bottom edge
for co=2:xpoint-1
    Tnew(end,co)=Fo*(Told(end-1,co)+Told(end-1,co)...
        +Told(end,co-1)+Told(end,co-1)-(4-(1/Fo))...
        *Told(end,co));
    %Checks if it has stopped changing
    stopper=Told(end,co)/Tnew(end,co);
    if stopper>=stop
        stopcheck(end,co)=1;
    end
end
%calculation for corners
Tnew(end,end)=Fo*(Told(row-1,col)+Told(row-1,col)...
    +Told(row,col-1)+Told(row,col-1)-(4-(1/Fo))...
    *Told(row,col));
Tnew(end,1)=Fo*(Told(row-1,col)+Told(row-1,col)...
    +Told(row,col+1)+Told(row,col+1)-(4-(1/Fo))...
    *Told(row,col));

%Checking the change in value
stopper=Told(row,col)/Tnew(row,col);
if stopper>=stop
    stopcheck(row,col)=1;
end
% When the difference is smaller than 0.99999 it changes
that cell in stop check to 0 once the total of the stopchcek is higher than
the exitcriteria.
% Then the iterations finish
%end of IF Statement 2
end
%end of col for
end
%end of row for
%% Plotting
if rem(row,10)==0 && rem(col,20)==0
    %Showing the Plate over time
    contourf(flipud(Tnew))
    c=colorbar;
    c.Label.String='Temperature(^{o}C)';
%Giving title to colourbar
xlabel('Hot Side')
set(gca, 'XAxisLocation', 'top')
%Moving the Xaxis to the top
colormap(jet)
drawnow
Niteration=Niteration+1
end
end
%End of while

toc
end
%End of 1st IF Statement

```