Phillip Krigbaum

ME 450 Sect 01

Project 1 Part 1

Due 3/10/23

Introduction

This report presents an analytical and numerical solution to the 2D steady-state heat conduction problem in a rectangular domain. The problem is solved using the finite difference method (FDM) and compared to an analytical solution obtained through separation of variables. The results are analyzed and the error between the two solutions is calculated. The FDM solution is found to have a relatively low error, indicating good agreement with the analytical solution. The report concludes by discussing the strengths and weaknesses of each method and their suitability for different types of problems.

Methods

To start the finite difference method (FDM) we start with the energy balance equation:

$$\frac{dE}{dt} = 0 = \dot{E}_{in} - \dot{E}_{out} + \dot{E}_{gen} \qquad \text{(Eqn. 1)}$$

We set the energy balance equation equal to 0 because the energy is not changing in time. We have a constant temperature at each surface through time. For this problem we are told that the energy generation is also equal to zero. This means that there is no energy being created in the plate. Due to this, we are able to reduce the above equation into the following:

$$\dot{E}_{in} - \dot{E}_{out} = 0 \qquad \text{(Eqn. 2)}$$

Using this as our base, we are able to set up nodes throughout the plate in order to estimate the heat transfer through the plate. A node is just a point on the plate that we decide to look at. The more nodes we place on the plate, the more accurate the estimation becomes. Once we decide how many nodes we want on the plate we begin to analyze how the heat transfer through each node. To do this we must first classify each node. There are 9 classifications that a node can have. The first four classifications are “corner nodes” this means that the node is closest to the corner of the plate. There are four corner nodes because there are four corners in the array on nodes. The next 4 classifications are “surface nodes” these are the nodes that are closest to the surface of the plate and are in between 2 corner nodes. There are four surface nodes because there are 4 sides to the array of nodes. The final classification is an “interior node” this is a node that is surrounded by other nodes and is not closest to a surface.

In order to examine the heat flow through the plate, we look at how the heat flows through the nodes. To do this we examine the heat flow through each individual node coming from all 4 directions. We use the conduction heat transfer equation below to examine the heat flow through the node:

$$q = -kA_x \frac{dT}{dx} \qquad \text{(Eqn. 3)}$$

Using this equation, we are able to set up the heat transfer through a node from all four directions. For example, if we set up a 3x3 node matrix where each node is dx apart in the x direction and dy apart in the y direction as shown below:

		T_{North}		
	Node 1 (NW)	Node 2 (Top)	Node 3 (NE)	
T_{West}	Node 4 (Left)	Node 5 (Center)	Node 6 (Left)	T_{East}
	Node 7 (SW)	Node 8 (Bottom)	Node 9 (SE)	
		T_{South}		

Figure 1

We would be able to analyze the heat transfer through each node. First, we would need to classify each node. The corner nodes are [1,3,7,9], the surface nodes are [2,4,6,8] and the interior node is [5]. After classifying the nodes we are able to define the heat transfer coming through each side of every node. If it is a corner node or a surface node, the difference in temperature from at least one direction is based on the surface temperature. This gives us an equation such as:

$$q_{Top} = -k \frac{\left(T_{North} - T(1)\right)}{dy/2} (dx \cdot 1) \qquad \text{(Eqn. 4)}$$

This equation shows the heat transfer through node 1 from the top of the node. As you can see because node one is a corner node the temperature difference is based on the northern surface temperature. I have also substituted dx*1 for the cross-sectional area. This is based on the assumption that the plate is 1 meter thick. Also notice that the dx under the fraction becomes (dy/2) this is distance between node 1 and the wall. There is one more step to take to reduce this equation to its final form. That is dividing everything by k. We are able to do this because when we plug this equation into the energy balance equation (Eqn. 2), we will see that k is on both sides of the equation. This means that we can divide out the k on both sides of the equation. This reduces the equation to:

$$q_{Top} = -\frac{k}{k} \frac{\left(T_{North} - T(1)\right)}{dy/2} (dx \cdot 1) \qquad \text{(Eqn. 5)}$$

Now we can set up the equation to find the heat transfer from the left of node one. This is the equation shown below.

$$q_{Left} = -\frac{k}{k}\frac{\left(T_{West} - T(1)\right)}{dx/2}(dy \cdot 1) \qquad \text{(Eqn. 6)}$$

The only differences between equation 5 and equation 6 is that the temperature difference is based on the west surface temperature now. Because of this we are dividing by (dx/2) because the distance between the west wall and node 1 is along the x axis and is half of the distance between 2 nodes. The other difference is that the cross sectional area is based on dy instead of dx.

Equations 5 and 6 show us how to deal with nodes when they are closest to the surfaces, but what about when the next closest temperature point is another node? This can be seen in equation 7.

$$q_{Right} = -\frac{k}{k}\frac{\left(T(2) - T(1)\right)}{dx}(dy \cdot 1) \qquad \text{(Eqn. 7)}$$

In equation 7 the temperature difference references the node closest to the right of node 1. Due to this, we see that the change in x is no longer divided by 2. This is because we are now traveling the full distance between nodes.

Using equations 5, 6 and 7 as a reference, I was able to build the heat transfer equations coming from all four directions for every node.

After defining the heat transfer equations for every node, I then added the four equations up at each node. Adding the equations up tells us the total heat transfer into each node, the sum of these equations represents $\dot{E}_{in} - \dot{E}_{out}$ in equation 2. When we plug that sum into equation 2 it is equal to 0. We can see this in equation 8 below. This is what allowed us to divide out the k in the earlier equation.

$$q_{Right} + q_{Left} + q_{Top} + q_{Bottom} = 0 \qquad \text{(Eqn. 8)}$$

Now we will plug in the equations that we found for the heat transfer from each direction. Then we will subtract the surface temperatures to the right side of the equation. As an example, I have included the reduced equation of node 1 in equation 9 below.

$$-6T(1) + 1T(2) + 1T(4) = -2T_{North} - 2T_{West} \qquad \text{(Eqn. 9)}$$

Then this process is repeated for every node in the array. After all the nodes have been calculated, 2 matrices can be created. The first matrix, matrix A, is created by assigning the coefficient in front of each temperature value to a position in the matrix. If there are 9 nodes in a system then matrix A will be a 9x9 matrix. Each row in the matrix corresponds to a node, the first row corresponds to the first node. Then the coefficient that is in front of each temperature value will be placed in this row. The column corresponds to the node's temperature that follows the coefficient. For example, equation 9 is referring to the heat transfer through node 1, so the first row in matrix A would go as follows: $[-6\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$. Repeating this process to fill out the rest of matrix A allows us to move to the next step. Filling out matrix b is very easy. Just like in matrix A, the row corresponds to the node number. We will take the entire right side of equation 9 and place that into matrix b. Repeat this for every node and we will have filled in both matrix A and b.

This process is shown in the MATLAB code displayed in Appendix A. It is shown in the "Populate matrix A(i,j) & b(i)" section of the code. The code loops through each node and checks for the nodes position. Based on the position it generates the A matrix and b matrix values and identifies the node type.

After populating both matrix A and b we can plug those matrices into equation 10:

$$T \cdot A = b \qquad \text{(Eqn. 10)}$$

Solving for T gives us the Temperature at every node. This is completed in the "Solve for unknown vector T(i)" section of Appendix A.

After solving for the Temperature at each node we must verify our solution. We will do this a couple of different ways. The first method is to verify the T at each node visually. To aid in this, the MATLAB attached in appendix A creates a surface plot of the temperature. This plot is shown in Figure 2.

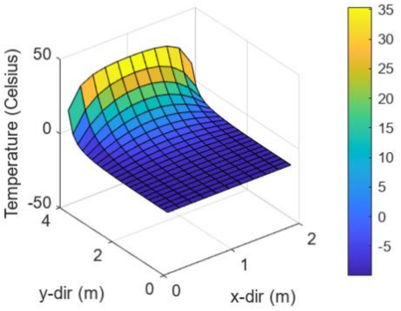


Figure 2

This plot allows us to verify that the temperatures make sense based on what is given in the problem. For our problem, the north surface has a temperature of 40 C while the other 3 surfaces have a temperature of -10 C. The surface plot shown in Figure 2 verifies this by showing a decrease in temperature as you move down the plat in both the x and y direction. There is a sharper decrease in temperature in the x direction. This makes sense because the x direction is shorter than the y direction in this problem.

In order to gain more insight, I have also plotted T as a heatmap and as a contour plot. Both plots shown in Figure 3 Figure 4 are the plate in 2 dimensions. These figures allow us to visualize how the heat is diffusing through the plate and they verify that our FDM model is accurate.

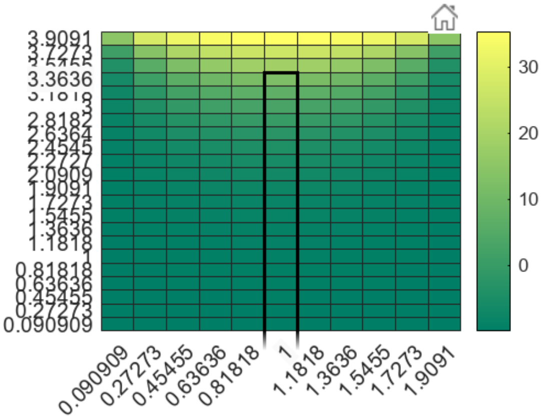


Figure 3

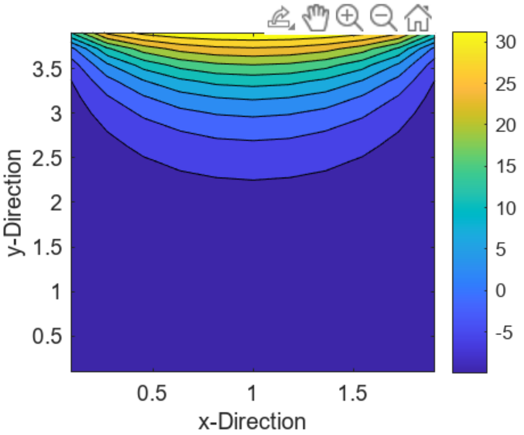
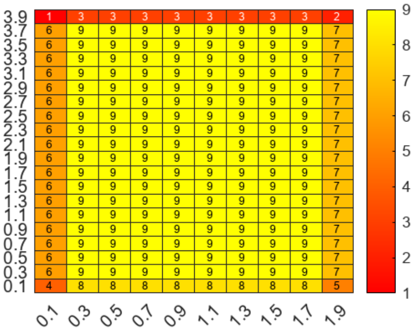


Figure 4

In order to ensure that our code accurately assigned each node's classification, I have displayed each node on a heat map shown in Figure 5



	0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
3.9	1	3	3	3	3	3	3	3	3	2
3.7	6	9	9	9	9	9	9	9	9	7
3.5	6	9	9	9	9	9	9	9	9	7
3.3	6	9	9	9	9	9	9	9	9	7
3.1	6	9	9	9	9	9	9	9	9	7
2.9	6	9	9	9	9	9	9	9	9	7
2.7	6	9	9	9	9	9	9	9	9	7
2.5	6	9	9	9	9	9	9	9	9	7
2.3	6	9	9	9	9	9	9	9	9	7
2.1	6	9	9	9	9	9	9	9	9	7
1.9	6	9	9	9	9	9	9	9	9	7
1.7	6	9	9	9	9	9	9	9	9	7
1.5	6	9	9	9	9	9	9	9	9	7
1.3	6	9	9	9	9	9	9	9	9	7
1.1	6	9	9	9	9	9	9	9	9	7
0.9	6	9	9	9	9	9	9	9	9	7
0.7	6	9	9	9	9	9	9	9	9	7
0.5	6	9	9	9	9	9	9	9	9	7
0.3	6	9	9	9	9	9	9	9	9	7
0.1	4	8	8	8	8	8	8	8	8	5

Figure 5

Figure 5 displays each node in its proper location in the nodal array. It also displays the classification that the node was given. This allows me to confirm that each node was given the proper classification and therefore the correct math was completed to fill both matrix A and b.

After verifying visually that the FDM model works properly, I needed to verify it analytically. To do this I calculated the heat rate coming from the top, right, bottom and left side of each node. Then totaled those up for each node. This gave me the total heat rate at each node. From that, I was able to verify my model two more ways. The first, another visual representation. Figure 6 shows the heat rate at each node.

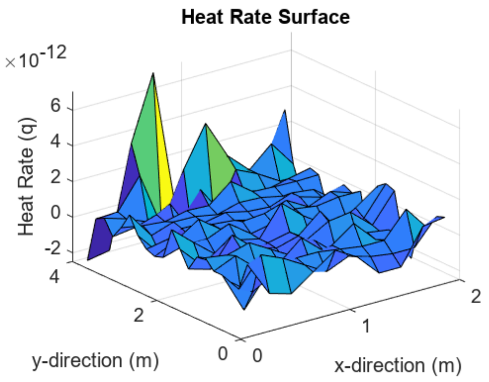
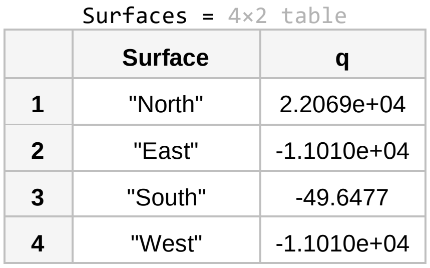


Figure 6

This again verifies our model because it shows that the heat rate is higher where there is a higher temperature difference. It also shows that there are "hills and valleys" of heat transfer throughout the plate. These hills and valleys end up canceling each other out when we sum all of the heat transfer up across all of the nodes. We find that the total heat transfer is 8.48e-12 when we have 200 nodes generated. This estimation of heat transfer becomes more accurate as you add more nodes to the system.

We also wanted to calculate how much heat transfer was occurring across each surface. To do this I set the appropriate side at each node that is along the surface equal to the matching surface heat transfer. I do this because if the energy is entering the node then it must be leaving from the surface. For example, in Figure 1 node 1 is in the Northwest corner. So, for node 1 I set $q_{North} = q_{Top}$. After doing this for every node I was able to sum up the heat transfer across each surface and interpret that data.



Surfaces = 4×2 table

	Surface	q
1	"North"	2.2069e+04
2	"East"	-1.1010e+04
3	"South"	-49.6477
4	"West"	-1.1010e+04

Table 1

Table 1 shows the heat transfer across each of the surfaces. This data also verifies that our model is working correctly. It shows that there is a positive heat transfer on the north surface and negative heat transfers on the remaining 3 surfaces, with the east and west surfaces matching.

After calculating the heat transfers, we are able to use equation 11 and 12 to calculate an analytical temperature at each node.

$$\theta(x,y) = \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^{n+1}+1}{n}\cdot\sin\left(\frac{n\pi x}{L}\right)\cdot\frac{\sinh\left({}^{n\pi y}/_{L}\right)}{\sinh\left({}^{n\pi W}/_{L}\right)} \qquad \text{(Eqn.11)}$$

$$T_a = \theta(x,y) * (T2 - T1) + T1 \qquad \text{(Eqn.12)}$$

After finding the analytical temperature I was able to generate another surface plot, this plot is shown in figure 7. This looks very similar to figure 2 which helps verify that the FDM model is correct.

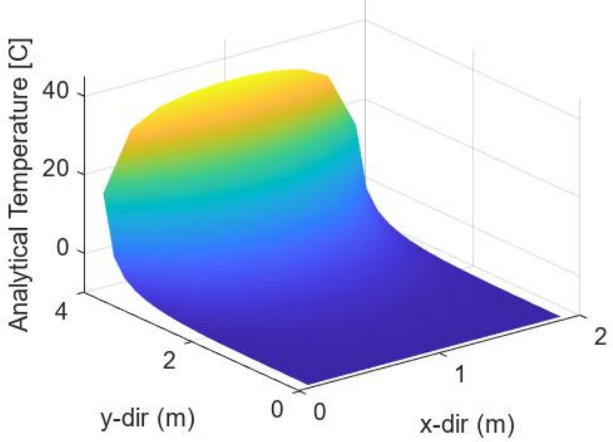


Figure 7

I then plotted the analytical temperature in a 2-dimensional plot. This is shown in figure 8. This plot looks very similar to figure 4 which further confirms that this model is correct.

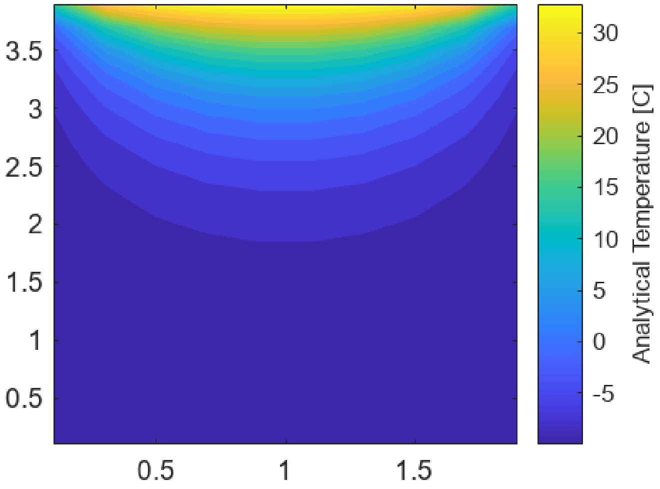


Figure 8

The final verification is calculating the error between the analytical and FDM solutions. To do this I used equation 13.

$$Error = (T_{FD} - T_a)^2 \qquad \text{(Eqn. 13)}$$

This equation takes the difference between the two temperature approximations at each node. I then sum up the nodal errors to get a total error of 6.37 degrees C. Figure 9 and 10 show this error graphically and we can see where the error lies.

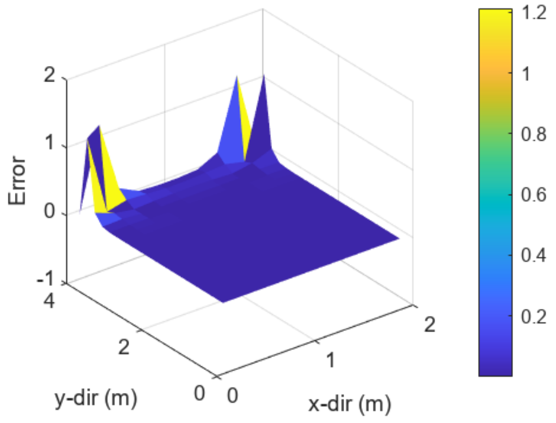


Figure 9

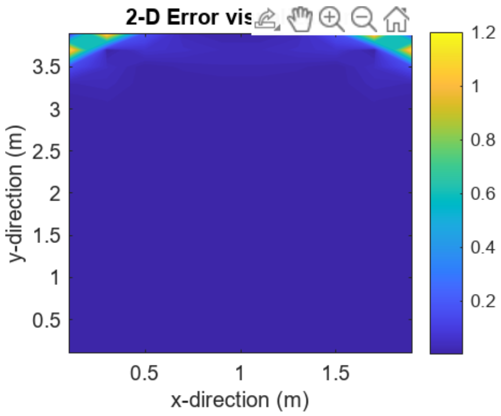


Figure 10

Both figure 9 and 10 show that most of the error occurs at the northeast and northwest corners. This makes sense that the error would occur in these locations because this is where the situation presents its flaws. The problem sets the northern surface to be 40 degrees Celsius and the remaining three sides to be -10 degrees Celsius. This will cause a problem at these two corners because the corners are simultaneously 40 degrees and -10 degrees. This causes issues to appear in our FDM model. However, the error is small enough that the FDM model can be verified by the analytical solution. Also, as shown in Figure 9 and 10, there is almost no error occurring throughout the rest of the plate. This means that the FDM model is verified by the analytical solution.

Appendix A

The 9 Node Sample Problem

ME 450

Programmer: Phillip Krigbaum adapting from J. Wade 2023, adapting from D. Willy 2020

Date: March 10, 2023

This code is demonstrating the

```
clear; clc; close all;   %Clearing/closing prior work
format short
```

Given:

```
%Problem Geometry
L = 2; %length of the domain in the x-direction, [m]
W = 4; %length of the domain in the y-direciton, [m]


%Material/Domain Properties


k = 20; %Thermal conductivity of the solid domain, [W/mK]


%Boundary Conditions


T2 = 40; %North boundary temperature, [C]
T1 = -10; %East, South and West boundary temperature, [C]
```

Find: Temperature Profile

Assumption: 2D flow, SS, qgen = 0, constant k

Solve:

Discretize the Problem

```
%Problem Discretization - needs to be FLEXIBLE
n_x =10 ;%discretized units in x-direction, [nd]
n_y = (W/L)*n_x; %discretized units in the y-direction
%discretization in x and y-direction are the same because the geometries
%are equivalent.
```

```
dx = L/n_x; %discretization size in x-direction, [m]
dy = W/n_y; %discretization size in y-direction, [m]


N = n_x*n_y; %Total unknown temperature nodes, [nd]


%Note formulating total unknown temperatures means I can vary mesh size w/o
%consequence to remaining equations.


x = dx/2:dx:L; % x-coordinates of T values (left to right)
y = (W-dy/2):-dy:0; % y-coordinates of T values (top to bottom)


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


xx = reshape(X',N,1);
yy = reshape(Y',N,1);
```

Initialize matrices and vectors

```
A = zeros(N); %Initilizs my matrix coefficient A based on total unknown nodes,
[C]
b = zeros(N,1); %Initializes the known constants based on total unknown nodes,
[C]
T = zeros(N,1); %Intializes unknown temperature, [C]


c = zeros(N,1); %Initializes a vector that identifies the node type (e.g. NW
corner, interior, etc)
```

Populate matrix A(i,j) & b(i)

```
for i = 1:N  %Run through all a(i,j) and b(i) coefficients, remember


    if xx(i) <= dx && yy(i) >= W-dy  %NW Corner


        A(i,i) = -6; %T(i) %temperature central to the nodal equation
        A(i,i+1) = 1; % right
        A(i,i+n_x) = 1; % bottom, one row down
        b(i) = -2*T2 - 2*T1; %known constants for T(i = 1)
        c(i) = 1; %identfies node type



    elseif xx(i) > (L - dx) && yy(i) > (W-dy)% NE Corner
        A(i,i) = -6;
        A(i,i-1) = 1; %left
        A(i,i+n_x) = 1; %below
        b(i) = -2*T1-2*T2; %knowns
        c(i) = 2; %node type


    elseif yy(i) > (W - dy) %top edge - Dirchlet
        A(i,i) = -5; %temperature central to the nodal equation
        A(i,i-1) = 1; %left
        A(i,i+1) = 1; %right
        A(i,i+n_x) = 1; % bottom
        b(i) = -2*T2; %known values
        c(i) = 3; %node type





    elseif  xx(i) < dx && yy(i) < dy  %SW Corner
        %xx(i) >= (L-dx/2) && yy(i) >= (W-dy/2) %NE Corner
        A(i,i) = -6; %temp central to nodal eqn
        A(i,i+1) = 1; %right
        A(i,i-n_x) = 1; %top
        b(i) = -4*T1; %knowns
        c(i) = 4; %node type





    elseif xx(i) >= (L-dx) && yy(i) <= (dy) %SE Corner
        A(i,i) = -6; %temp central to nodal eqn
        A(i,i-1) = 1; %left
        A(i,i-n_x) = 1; %top
        b(i) = -4*T1; %knowns
        c(i) = 5; %node type



    elseif xx(i) < dx %West Surface - Dirchlet
        A(i,i) = -5; %temp central to nodal eqn
        A(i,i+1) = 1; %right
        A(i,i-n_x) = 1; %top
        A(i,i+n_x) = 1; %below
        b(i) = -2*T1; %knowns
        c(i) = 6; %node type





    elseif xx(i) > L- dx %East Surface - Convective BC
        A(i,i) = -5; %temp central to nodal eqn
        A(i,i-1) = 1; %left
        A(i,i-n_x) = 1; %top
        A(i,i+n_x) = 1; %below
        b(i) = -2*T1; %knowns
        c(i) = 7; %node type



     elseif yy(i) < dy %South Surface - Adiabatic BC
        A(i,i) = -5; %temp central to nodal eqn
        A(i,i-1) = 1; %left
        A(i,i+1) = 1; %right
        A(i,i-n_x) = 1; %top
        b(i) = -2*T1; %knowns
        c(i) = 8; %node type


    else %Interior Node
        A(i,i) = -4; %temp central to nodal eqn
        A(i,i-1) = 1; %prior row
        A(i,i+1) = 1; %right
        A(i,i-n_x) = 1; %top
        A(i,i+n_x) = 1; %below
        b(i) = 0; %knowns
        c(i) = 9; %node type






    end
end
```

Solve unknown vector T(i)

```
T = A\b; %Matlab's Matrix Inversion Solver
```

Visualize the temperature distribution

1) Turn the (Nx1) T vector into a 2D (n_x) x (n_y) array using "reshape" function

```
T_FD= reshape(T,n_x,n_y)';  %Reshaping finite difference solved T_FD vector into 
the 2D array (n_x by n_y)
```

3) Visualize Temperature data as a surface map

```
figure(1)
surf(X,Y,T_FD)  %Note surface plot requires the X,Y coordinate arrays in order to 
plot
xlabel('x-dir (m)')
ylabel('y-dir (m)')
zlabel('Temperature (Celsius)')
zlim([-50 50])
% view(60,45)
colorbar
```

4) Visualize temperature data as a heat map

```
figure(2)
h = heatmap(x,y,T_FD,'Colormap',summer);
% Note, a heat map requires the vectors of x and y, not the X,Y arrays
```

5) Visualize temperature data as a contour map

```
figure(3)
contourf(X,Y,T_FD,10)
% view(90,90)
colorbar
xlabel('x-Direction');
ylabel('y-Direction');
```

6) Check that nodes were assigned correctly

```
nodes = reshape(c,n_x,n_y)';
nodecheck = heatmap(x,y,nodes,"colormap",autumn)
```

7) **Verify** the numerical simulation

```
%initializing thermal energy flow vectors for each node and each domain
```

```
%surface


qtotal = zeros(N,1); %Net energy flow through the nodal finite volume
qnorth = zeros(N,1); %Net energy flow through the north surface
qeast  = zeros(N,1); %Net energy flow through the east surface
qsouth = zeros(N,1); %Net energy flow through the south surface
qwest  = zeros(N,1); %Net energy flow through the west surface


for i = 1:N  %Run the energy balance through all nodes


    %%Solve qtotal at each node using the newly solved temperatures


    if xx(i) <= dx && yy(i) >= W-dy  %NW Corner


        qtop   = k * (T2 - T(i))/(dy/2);  % Top Boundary Flow
        qright = k * (T(i+1) - T(i))/dx;  % Right Boundary Flow
        qleft  = k * (T1 - T(i))/(dx/2);  % Left Boundary Flow
        qbott  = k * (T(i+n_x) - T(i))/dy;% Bottom Boundary Flow


        qtotal(i) = qtop + qright + qleft + qbott;
        qnorth(i) = qtop; %Net energy flow through the north surface
        qeast(i)  = 0; %Net energy flow through the east surface
        qsouth(i) = 0; %Net energy flow through the south surface
        qwest(i)  = qleft; %Net energy flow through the west surface


     elseif xx(i) > (L - dx) && yy(i) > (W-dy)% NE Corner
        qtop   = k * (T2 - T(i))/(dy/2);  % Top Boundary Flow
        qright = k * (T1 - T(i))/(dx/2);  % Right Boundary Flow
        qleft  = k * (T(i-1) - T(i))/(dx);  % Left Boundary Flow
        qbott  = k * (T(i+n_x) - T(i))/dy;% Bottom Boundary Flow


        qtotal(i) = qtop + qright + qleft + qbott;
        qnorth(i) = qtop; %Net energy flow through the north surface
        qeast(i)  = qright; %Net energy flow through the east surface
        qsouth(i) = 0; %Net energy flow through the south surface
        qwest(i)  = 0; %Net energy flow through the west surface

     elseif yy(i) > (W - dy) %top edge - Dirchlet
        qtop   = k * (T2 - T(i))/(dy/2);  % Top Boundary Flow
        qright = k * (T(i+1) - T(i))/dx;  % Right Boundary Flow
        qleft  = k * (T(i-1) - T(i))/(dx);  % Left Boundary Flow
```

```
       qbott  = k * (T(i+n_x) - T(i))/dy;% Bottom Boundary Flow


       qtotal(i) = qtop + qright + qleft + qbott;
       qnorth(i) = qtop; %Net energy flow through the north surface
       qeast(i)  = 0; %Net energy flow through the east surface
       qsouth(i) = 0; %Net energy flow through the south surface
       qwest(i)  = 0; %Net energy flow through the west surface





    elseif  xx(i) < dx && yy(i) < dy  %SW Corner
       qtop   = k * (T(i-n_x) - T(i))/(dy);  % Top Boundary Flow
       qright = k * (T(i+1) - T(i))/dx;  % Right Boundary Flow
       qleft  = k * (T1 - T(i))/(dx/2);  % Left Boundary Flow
       qbott  = k * (T1 - T(i))/(dy/2);% Bottom Boundary Flow


       qtotal(i) = qtop + qright + qleft + qbott;
       qnorth(i) = 0; %Net energy flow through the north surface
       qeast(i)  = 0; %Net energy flow through the east surface
       qsouth(i) = qbott; %Net energy flow through the south surface
       qwest(i)  = qleft; %Net energy flow through the west surface



    elseif xx(i) >= (L-dx) && yy(i) <= (dy) %SE Corner
       qtop   = k * (T(i-n_x) - T(i))/(dy);  % Top Boundary Flow
       qright = k * (T1 - T(i))/(dx/2);  % Right Boundary Flow
       qleft  = k * (T(i-1) - T(i))/(dx);  % Left Boundary Flow
       qbott  = k * (T1 - T(i))/(dy/2);% Bottom Boundary Flow


       qtotal(i) = qtop + qright + qleft + qbott;
       qnorth(i) = 0; %Net energy flow through the north surface
       qeast(i)  = qright; %Net energy flow through the east surface
       qsouth(i) = qbott; %Net energy flow through the south surface
       qwest(i)  = 0; %Net energy flow through the west surface



    elseif xx(i) < dx %West Surface - Dirchlet
       qtop   = k * (T(i-n_x) - T(i))/(dy);  % Top Boundary Flow
       qright = k * (T(i+1) - T(i))/dx;  % Right Boundary Flow
       qleft  = k * (T1 - T(i))/(dx/2);  % Left Boundary Flow
       qbott  = k * (T(i+n_x) - T(i))/dy;% Bottom Boundary Flow
```

```
        qtotal(i) = qtop + qright + qleft + qbott;
        qnorth(i) = 0; %Net energy flow through the north surface
        qeast(i)  = 0; %Net energy flow through the east surface
        qsouth(i) = 0; %Net energy flow through the south surface
        qwest(i)  = qleft; %Net energy flow through the west surface



     elseif xx(i) > L- dx %East Surface - Convective BC
        qtop   = k * (T(i-n_x) - T(i))/(dy);  % Top Boundary Flow
        qright = k * (T1 - T(i))/(dx/2);  % Right Boundary Flow
        qleft  = k * (T(i-1) - T(i))/(dx);  % Left Boundary Flow
        qbott  = k * (T(i+n_x) - T(i))/dy;% Bottom Boundary Flow


        qtotal(i) = qtop + qright + qleft + qbott;
        qnorth(i) = 0; %Net energy flow through the north surface
        qeast(i)  = qright; %Net energy flow through the east surface
        qsouth(i) = 0; %Net energy flow through the south surface
        qwest(i)  = 0; %Net energy flow through the west surface


     elseif yy(i) < dy %South Surface - Adiabatic BC
        qtop   = k * (T(i-n_x) - T(i))/(dy);  % Top Boundary Flow
        qright = k * (T(i+1) - T(i))/dx;  % Right Boundary Flow
        qleft  = k * (T(i-1) - T(i))/(dx);  % Left Boundary Flow
        qbott  = k * (T1 - T(i))/(dy/2);% Bottom Boundary Flow


        qtotal(i) = qtop + qright + qleft + qbott;
        qnorth(i) = 0; %Net energy flow through the north surface
        qeast(i)  = 0; %Net energy flow through the east surface
        qsouth(i) = qbott; %Net energy flow through the south surface
        qwest(i)  = 0; %Net energy flow through the west surface
     else %Interior Node
        qtop   = k * (T(i-n_x) - T(i))/(dy);  % Top Boundary Flow
        qright = k * (T(i+1) - T(i))/dx;  % Right Boundary Flow
        qleft  = k * (T(i-1) - T(i))/(dx);  % Left Boundary Flow
        qbott  = k * (T(i+n_x) - T(i))/dy;% Bottom Boundary Flow


        qtotal(i) = qtop + qright + qleft + qbott;
        qnorth(i) = 0; %Net energy flow through the north surface
        qeast(i)  = 0; %Net energy flow through the east surface
        qsouth(i) = 0; %Net energy flow through the south surface
        qwest(i)  = 0; %Net energy flow through the west surface
    end
end
```

Heat Rate Solution:

```
figure(4)
qtotall = reshape(qtotal,n_x,n_y)';
surf(X,Y,qtotall)
title('Heat Rate Surface');
xlabel('x-direction (m)');
ylabel('y-direction (m)');
zlabel('Heat Rate (q)');


qtotal = sum(qtotal)
qnorth = sum(qnorth);
qeast = sum(qeast);
qsouth = sum(qsouth);
qwest = sum(qwest);


Surfaces = table('Size',[4
2],'VariableTypes',{'string','double'},'VariableNames',{'Surface','q'});
Surfaces(1,:) = {'North',qnorth};
Surfaces(2,:) = {'East', qeast};
Surfaces(3,:) = {'South', qsouth};
Surfaces(4,:) = {'West', qwest}


if qtotal<10^(-10)
    disp ('This Satisfies our Energy Balance Equation')
else
    disp ('This does not satisfy our Energy Balance Equation')
end
```

Error Calculation

```


theta = zeros(length(x),length(y));


for i = 1:n_x
    for j = 1:n_y
        for n = 1:100
            lambda = n*pi/L;
            theta(i,j) = theta(i,j) + (2/pi).*(((-
1)^(n+1)+1)/n).*(sin(lambda.*x(i)).*(sinh(lambda.*y(j))/(sinh(lambda.*W))));
        end
    end
end
```

```
T_a = theta.*(T2-T1) + T1; %Solve for Temp from theta



figure(5)
surf(x,y,T_a','edgecolor','none')
xlabel('x-dir (m)')
ylabel('y-dir (m)')
zlabel('Analytical Temperature [C]')
shading interp
zlim([-10 45])




figure(6)
contourf(x,y,T_a',20,'edgecolor','none')
c = colorbar;
c.Label.String = 'Analytical Temperature [C]';
```

Calculating ERROR

```
Err = (T_FD - T_a').^2;
Error_total = sum(Err);
Error_total = sum(Error_total);


fprintf('The Total Error is %.2f degrees [C]',Error_total);




figure(3)
surf(x,y,Err,'edgecolor','none')
xlabel('x-dir (m)')
ylabel('y-dir (m)')
zlabel('Error')
zlim([-1 2])
colorbar


figure(8)
contourf(x,y,Err,100,'edgecolor','none')
xlabel('x-direction (m)')
ylabel('y-direction (m)')
title('2-D Error visualization')
colorbar
```