Phillip Krigbaum

ME 450 Sect 01

Project 1 Part 2

Due 3/24/23

Introduction

This report presents the numerical solution to the 2D steady state problem in a rectangular domain. I started this project with a simple 2D steady state problem in a rectangular domain that only had conductivity occurring as the mode of heat transfer. I was able to confirm my numerical solution using an analytical solution. After confirming that my numerical solution was accurate, I was able to modify the solution to the requirements of part 2. For this part, the rectangular domain contained a circle that contained heat generation. It also introduced convection to the northern surface of the domain and made the east and west surfaces adiabatic, Figure 1 demonstrates the design requirements. Using the model that was generated in part 1 I was able to modify it to meet the design requirements of this part. After doing this I calculated the error that occurs as you increased the number of nodes in the domain.

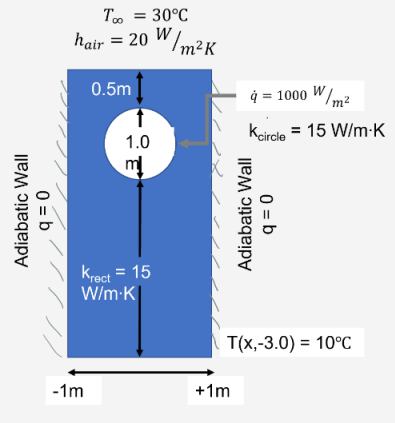


Figure 1 Design Requirements

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 1000 in the circle but 0 throughout the rest of the domain. Due to this, we can reduce the above equation into the following:

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

Using this as our base, we can 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. Due to the heat generation of this problem that is located in the circle, these 9 classifications can have another classification on top of their original classification. They can be labeled as "heat generating nodes" if they are in the circle of our problem.

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 can 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:

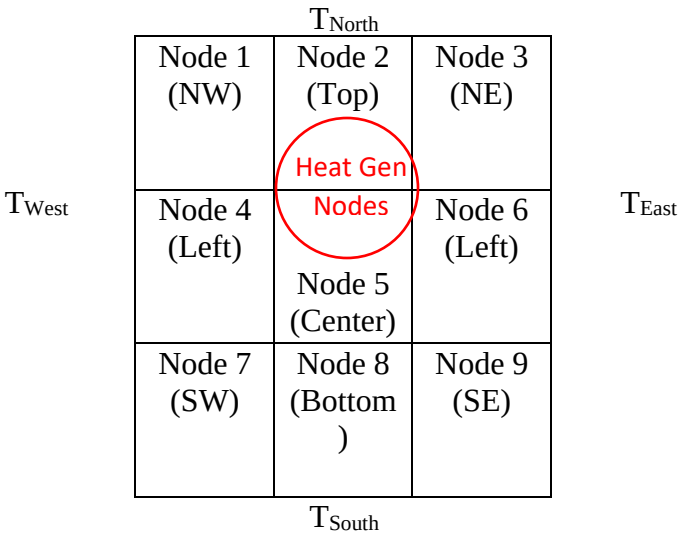


Figure 2 Node Matrix Set Up

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], the interior node is [5] and any node that is located in the heat generation circle is a "heat generation node". After classifying the nodes, we are able to define the heat transfer coming through each side of every node. If it is a southern corner node or a surface node, the difference in temperature from the southern direction is based on the southern surface temperature. This gives us an equation such as:

$$q_{Bottom} = -k\frac{\left(T_{South} - T(7)\right)}{dy/2}(dx \cdot 1) \qquad \text{(Eqn. 4)}$$

This equation shows the heat transfer through node seven from the bottom of the node. As you can see because node seven is a corner node on the southern surface the temperature difference is based on the southern 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 7 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 can 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:

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

Now we can set up the equation to find the heat transfer from the left of node 7. This is the equation shown below.

$$q_{Left} = 0 \qquad \text{(Eqn. 6)}$$

The heat transfer coming from the left of node 7 is 0 because the west surface is adiabatic. This means that no heat transfer occurs across this boundary. This means that for all corner nodes and east and west surface nodes, the q coming from the outer wall is 0 due to the fact that the walls are adiabatic.

Equations 5 and 6 show us how to deal with nodes closest to the east, west and south 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.

On the northern surface of the domain there is convection that occurs from the surrounding air. This requires a different equation than the one we used for conduction. We use equation 8 for the q_{Top} on all of the northern surface, and northern corner nodes.

$$q_{Top} = \frac{h}{k}\Big(T_{inf} - T(1)\Big) \cdot (dx \cdot 1) \qquad \text{(Eqn. 8)}$$

The next step is to analyze the affect that the $\dot{q}_{gen}$ has on the heat transfer. To do this we find which nodes are located in the circle and add equation 9 to the total heat transfer of this node.

$$q_{gen} = \frac{\dot{q}_{gen} \cdot (dx \cdot dy)}{k} \qquad \text{(Eqn. 9)}$$

Using equations 5, 6, 7, 8 and 9 as a reference, I was able to build the heat transfer equations coming from all four directions for every node and find the heat transfer coming from the heat generation.

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 $\dot{E}_{in} - \dot{E}_{out} + \dot{E}_{gen}$ in equation 2. When we plug that sum into equation 2 it is equal to 0. We can see this in equation 10 below. This is what allowed us to divide out the k in the earlier equation.

$$q_{Right} + q_{Left} + q_{Top} + q_{Bottom} + q_{gen} = 0 \qquad \text{(Eqn. 10)}$$

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 11 below.

$$\left[\left(-\frac{h}{k} \cdot dx\right) - 2\right] T(1) + 1T(2) + 1T(4) = -\frac{h}{k} \cdot T_{\infty} \cdot dx \qquad \text{(Eqn. 11)}$$

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 11 is referring to the heat transfer through node 1, so the first row in matrix A would go as follows: $\left[\left(-\frac{h}{k} \cdot dx - 2\right)\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\right]$. 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 11 and place that into matrix b. For nodes that lie in the heat generation circle, we need to subtract q_{gen} from that node's matrix b position. Repeat this for every node.

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 12:

$$T \cdot A = b \qquad \text{(Eqn. 12)}$$

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.

Temperature Solutions

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 3.

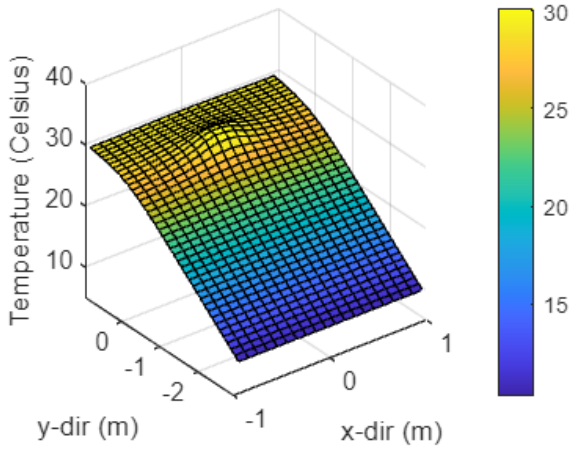


Figure 3 Temperature Surface Plot

This plot allows us to verify that the temperature makes sense based on what is given in the problem. For our problem, there is convection the north surface with a free stream temperature of 30 degrees Celsius, adiabatic east and west sides and a southern surface temperature of 10 degrees. There is also heat generation at (0,0). Figure 3 verifies these conditions by showing a southern surface temperature of 10 degrees Celsius and a "hump" in the graph at (0,0). The hump represents the heat generation that is occurring in the circle at this location. The graph shows that the temperature disperses and lowers as you move away from (0,0) in any direction. Overall, Figure 3 verifies the conditions presented in our problem.

In order to gain more insight, I have also plotted T as a heatmap plot. This plot is the plate in 2 dimensions. The figure 4 allows us to visualize how the heat is diffusing through the plate and verify that our FDM model is accurate.

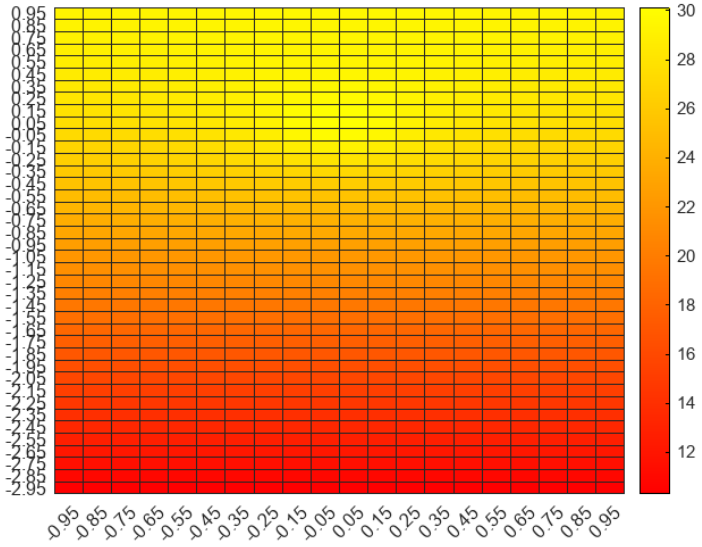


Figure 4 Heat Map of Temperature

Figure 4 confirms what we saw in figure 3. We see that there is a high spot of temperature at (0,0) and the heat dissipates evenly through the surface to be at 10 degrees Celsius on the southern border.

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

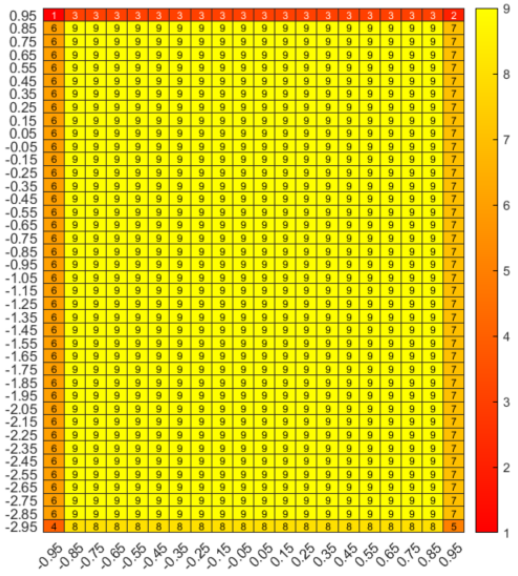


Figure 5 Node Identification

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.

Heat Rate Solution

After verifying visually that the FDM model works properly, I needed calculate the heat rate coming from the top, right, bottom, left side and generation 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 another way. Another visual representation. Figure 6 shows the heat rate at each node.

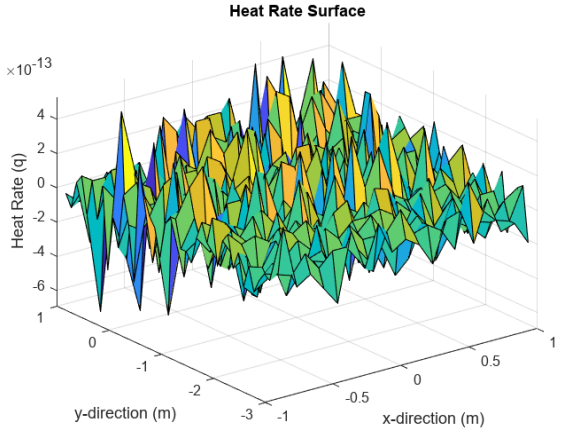


Figure 6 Heat Rate Surface Map

This again verifies our model because it shows that the heat rate is higher where there is a higher temperature difference, such as in the heat generation circle. 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 the heat transfer up across all the nodes. We find that the total heat transfer is 3.233e-13 when we have 200 nodes generated. It is important to note that the total heat rate should sum up to zero according to the energy balance equation shown in equation 2. This graph is set to a scale of 10^{-13} this means that the total heat transfer at each node is very close to zero. This satisfies the thermal energy balance expression shown in equation 2.

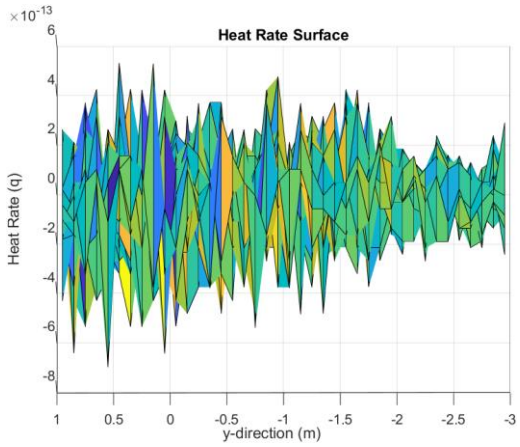


Figure 7 Heat Rate Surface y-direction

Figure 7 is the same as figure 6 but it is oriented to only show the heat rate as you move across the y axis. This allows us to see that the magnitude of q is larger at the heat generation circle and it gets slightly smaller as we move to the right (southern face). This again proves what we expect to see.

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 2 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.

	Surface	q
1	"North"	22.3503
2	"East"	0
3	"South"	-187.6396
4	"West"	0

Table 1 Heat Rate Out of Domain Surfaces

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 southern surface, with the east and west surfaces being 0 as they are adiabatic.

The next step was to develop the error that is caused by doubling the discretization. To do this I ran the code under the first discretization of 10x20 and then looped the code to increase in node size each iteration until it reached 20x40. This allows us to compare the total heat transfer from each of the different discretization sizes. The results of this error is demonstrated in figure 8.

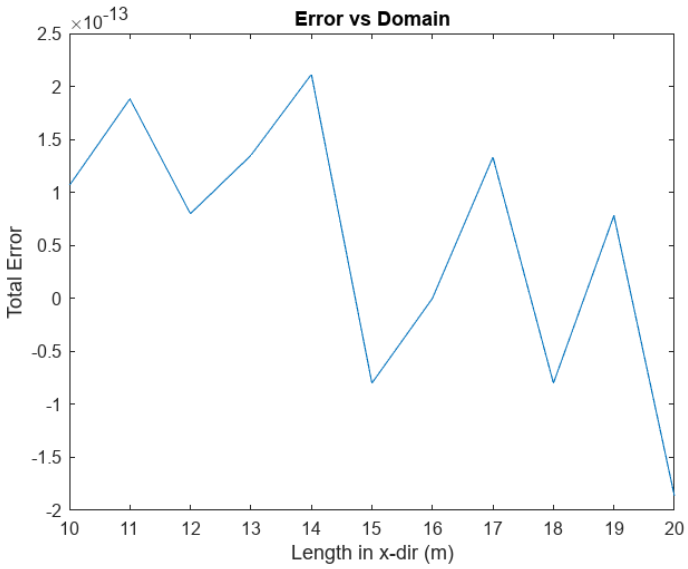


Figure 8 Error for Discretization

Figure 8 shows us that the error fluctuates as we change the discretization, but it follows an overall downward trend. This makes sense due to the fact that the heat generation is only in a small portion of the domain. The cause of the fluctuations is due to how many nodes land in the circle. When the ratio of $\frac{heat\ generation\ node}{non\ heat\ generation\ node}$ is higher the q is slightly higher which cause an increase in error. However, the trend shown in this graph shows us that the mode nodes we add the more accurate the model becomes.

Discussion

This numerical project taught me how to approach a problem by solving a simpler version of a problem and verify that your model works before moving onto a more difficult problem that cannot be as easily verified. I learned how to apply the finite difference method to solve the steady-state temperature profile and heat rate for a complex configuration with non-uniform boundary conditions. By using the framework developed in Part I, I was able to modify the code to handle the new domains and boundary conditions. Overall, this project taught me the importance of choosing appropriate numerical methods for solving complex heat transfer problems, as well as the importance of verifying the numerical solution to ensure its accuracy. I also gained practical experience in using MATLAB to develop and implement numerical solutions to complex problems.

Appendix A

ME 450

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

Date: March 24, 2023

```
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]
d = 1; %diameter of the circle [m]
Area = pi*(d/2)^2; %Area of the circle [m^2]


%Material/Domain Properties
q_dot_circle = 1000; %heat generation of the circle [W/m2]
k = 15; %Thermal conductivity of the solid domain, [W/mK]


%Boundary Conditions


T_inf = 30; %North boundary temperature, [C]
h = 20; % Thermal convection coefficient of the air [W/m2K]
T_bot = 10; %South boundary temperature, [C]
```

Find: Temperature Profile

Assumption: 2D flow, SS, constant k

Solve:

Discretize the Problem

```
%Problem Discretization - needs to be FLEXIBLE
n_x = 10:1:20;
for j = 1:length(n_x)
n_y = (W/L)*n_x(j); %discretized units in the y-direction
%discretization in x and y-direction are the same because the geometries
%are equivalent.
```

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


N = n_x(j)*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 = (-L/2)+(dx/2):dx:(L/2)-(dx/2); % x-coordinates of T values (left to right)
y = (d)-(dy/2):-dy:(-W+d)+(dy/2);


[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) <= -L/2+dx && yy(i) >= (d-dy)  %NW Corner


        A(i,i) = (-(h/k)*dx)-2; %T(i) %temperature central to the nodal equation
        A(i,i+1) = 1; % right
        A(i,i+n_x(j)) = 1; % bottom, one row down
```

```
        b(i) = (-(h/k)*T_inf*dx); %known constants for T(i = 1)
        c(i) = 1; %identfies node type



    elseif xx(i) > (L/2 - dx) && yy(i) > (d-dy)% NE Corner
        A(i,i) = (-(h/k)*dx)-2;
        A(i,i-1) = 1; %left
        A(i,i+n_x(j)) = 1; %below
        b(i) = (-(h/k)*T_inf*dx); %knowns
        c(i) = 2; %node type


    elseif yy(i) > (d- dy) %top edge - Dirchlet
        A(i,i) = (-(h/k)*dx)-3; %temperature central to the nodal equation
        A(i,i-1) = 1; %left
        A(i,i+1) = 1; %right
        A(i,i+n_x(j)) = 1; % bottom
        b(i) = (-(h/k)*T_inf*dx); %known values
        c(i) = 3; %node type




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




    elseif xx(i) >= (L/2-dx) && yy(i) <= (-W+d+dy) %SE Corner
        A(i,i) = -4; %temp central to nodal eqn
        A(i,i-1) = 1; %left
        A(i,i-n_x(j)) = 1; %top
        b(i) = -2*T_bot; %knowns
        c(i) = 5; %node type



    elseif xx(i) < -(L/2)+dx %West Surface - Dirchlet
        A(i,i) = -3; %temp central to nodal eqn
```

```
        A(i,i+1) = 1; %right
        A(i,i-n_x(j)) = 1; %top
        A(i,i+n_x(j)) = 1; %below
        b(i) = 0; %knowns
        c(i) = 6; %node type




    elseif xx(i) > L/2- dx %East Surface - Convective BC
        A(i,i) = -3; %temp central to nodal eqn
        A(i,i-1) = 1; %left
        A(i,i-n_x(j)) = 1; %top
        A(i,i+n_x(j)) = 1; %below
        b(i) = 0; %knowns
        c(i) = 7; %node type



     elseif yy(i) < (-W+d)+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(j)) = 1; %top
        b(i) = -2*T_bot; %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(j)) = 1; %top
        A(i,i+n_x(j)) = 1; %below
        b(i) = 0; %knowns
        c(i) = 9; %node type
    end
    if sqrt((xx(i))^2+(yy(i))^2)<=(d/2)^2
         b(i) = b(i) - ((q_dot_circle*(dx*dy))/k);
    end
end
```

Solve unknown vector T(i)

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

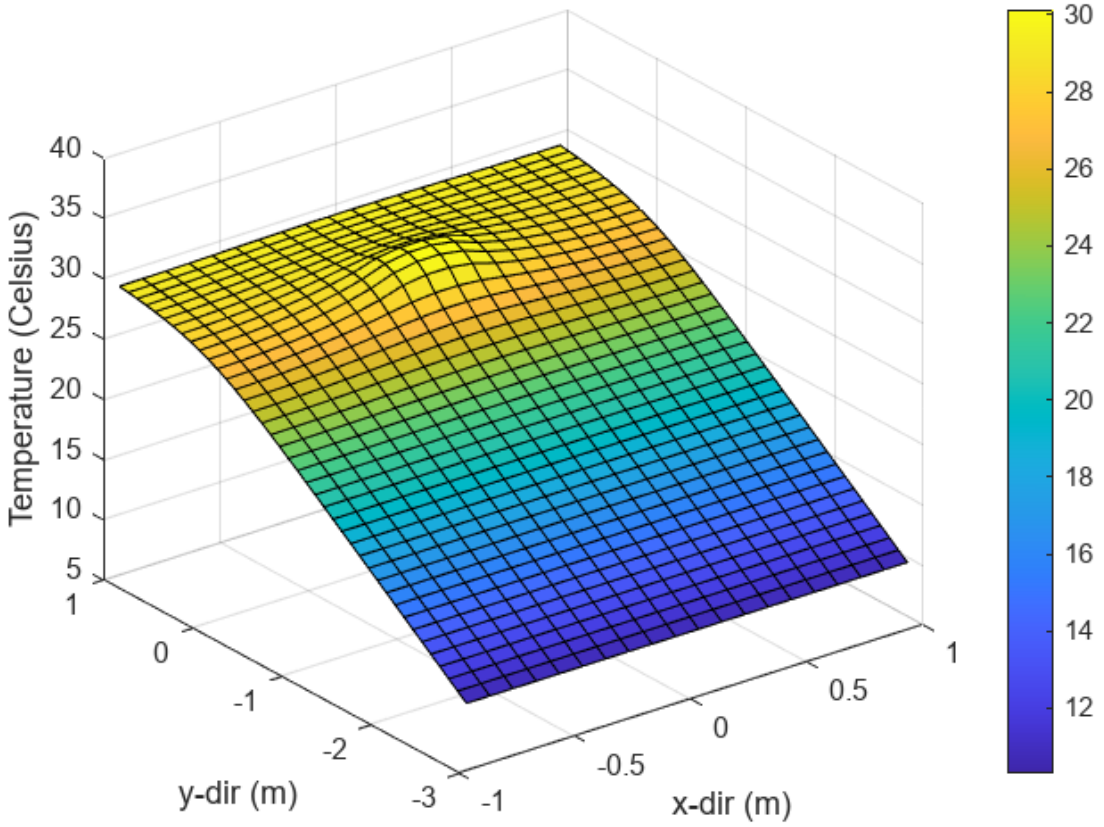
Visualize the temperature distribution

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

```
T_FD= reshape(T,n_x(j),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
```

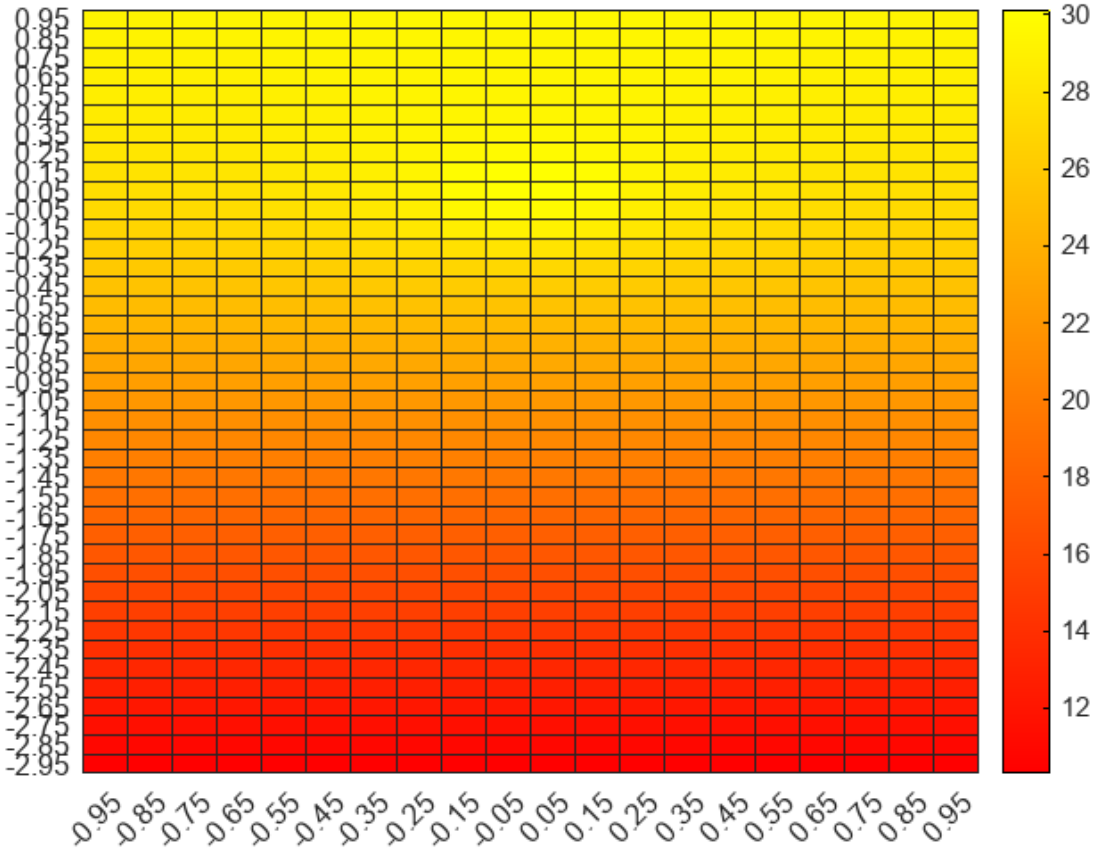



```
xlabel('x-dir (m)')
ylabel('y-dir (m)')
zlabel('Temperature (Celsius)')
```

```
zlim([5 40])
colorbar
```

4) Visualize temperature data as a heat map

```
figure(2)
heat = heatmap(x,y,T_FD,'Colormap',autumn);
```




```
% Note, a heat map requires the vectors of x and y, not the X,Y arrays
```

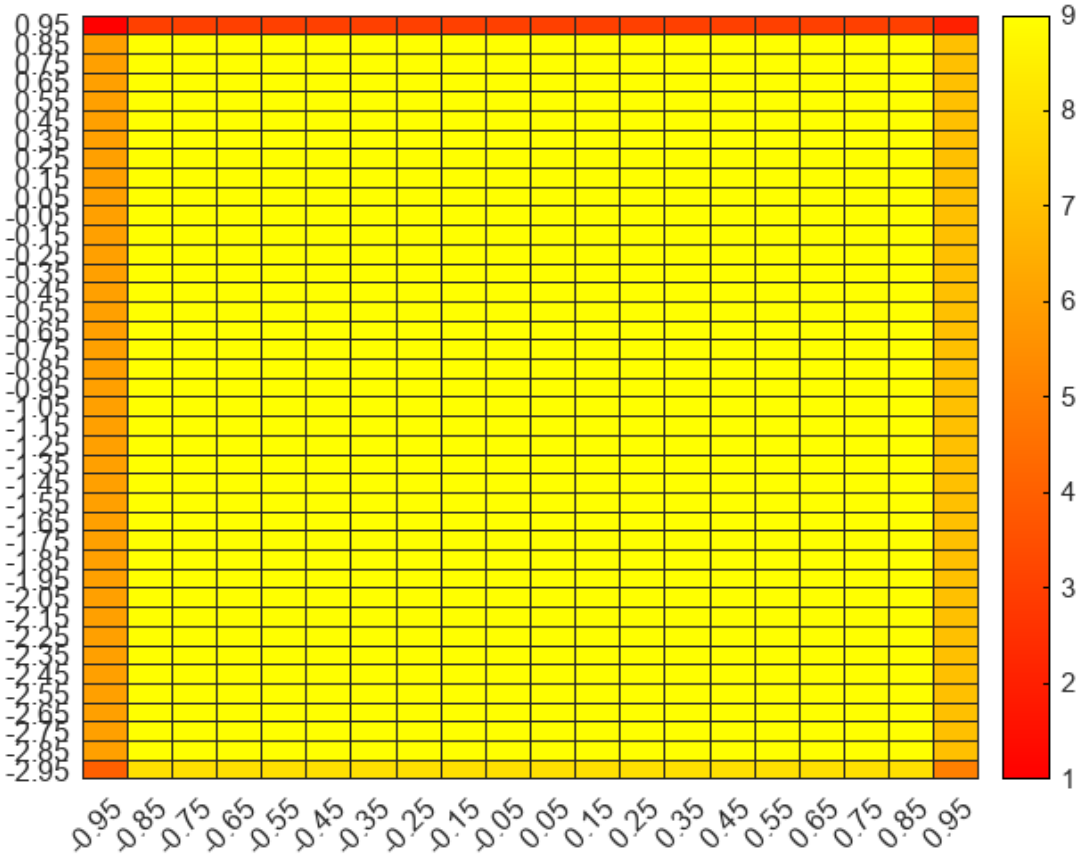
5) Visualize temperature data as a contour map

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

```
ylabel('y-Direction');
```

6) Check that nodes were assigned correctly

```
nodes = reshape(c,n_x(j),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
```

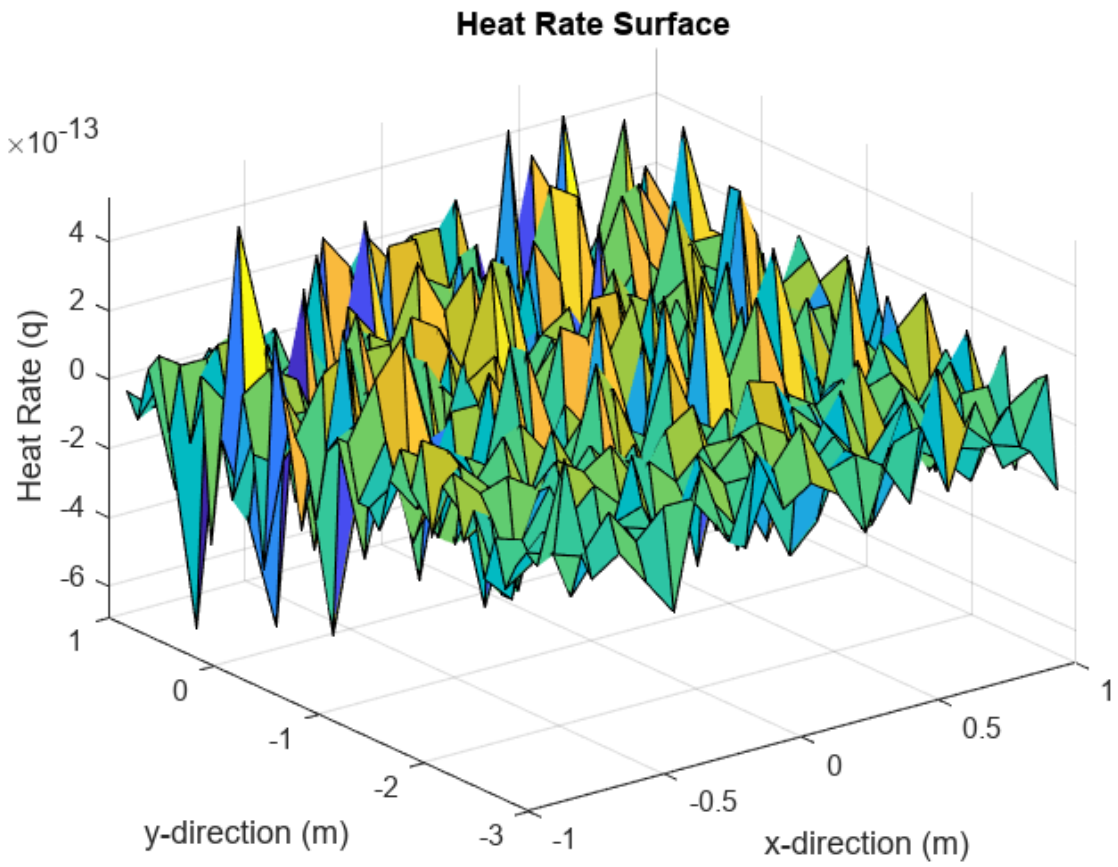
```
qwest  = zeros(N,1); %Net energy flow through the west surface


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


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


    if xx(i) <= -(L/2)+dx && yy(i) >= d-dy  %NW Corner


        qtop   = h * (dx*1)*(T_inf - T(i));  % Top Boundary Flow
        qright = k * (dy*1)*(T(i+1) - T(i))/dx;  % Right Boundary Flow
        qleft  = 0;  % Left Boundary Flow
        qbott  = k * (dx*1)*(T(i+n_x(j)) - 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/2) - dx) && yy(i) > (d-dy)% NE Corner
        qtop   = h * (dx*1)*(T_inf - T(i));  % Top Boundary Flow
        qright = 0;  % Right Boundary Flow
        qleft  = k * (dy*1)*(T(i-1) - T(i))/(dx);  % Left Boundary Flow
        qbott  = k * (dx*1)*(T(i+n_x(j)) - 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) > (d - dy) %top edge - Dirchlet
        qtop   = h * (dx*1)*(T_inf - T(i));  % Top Boundary Flow
        qright = k * (dy*1)*(T(i+1) - T(i))/dx;  % Right Boundary Flow
        qleft  = k * (dy*1)*(T(i-1) - T(i))/(dx);  % Left Boundary Flow
        qbott  = k * (dx*1)*(T(i+n_x(j)) - 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) < -(L/2)+dx && yy(i) < (-W+d)+ dy  %SW Corner
    qtop   = k * (dx*1)*(T(i-n_x(j)) - T(i))/(dy);  % Top Boundary Flow
    qright = k * (dy*1)*(T(i+1) - T(i))/dx;  % Right Boundary Flow
    qleft  = 0;  % Left Boundary Flow
    qbott  = k * (dx*1)*(T_bot - 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/2)-dx && yy(i) <= (-W+d+dy) %SE Corner
    qtop   = k * (dx*1)*(T(i-n_x(j)) - T(i))/(dy);  % Top Boundary Flow
    qright = 0;  % Right Boundary Flow
    qleft  = k * (dy*1)*(T(i-1) - T(i))/(dx);  % Left Boundary Flow
    qbott  = k * (dx*1)*(T_bot - 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) < -(L/2)+dx %West Surface - Dirchlet
    qtop   = k * (dx*1)*(T(i-n_x(j)) - T(i))/(dy);  % Top Boundary Flow
    qright = k * (dy*1)*(T(i+1) - T(i))/dx;  % Right Boundary Flow
    qleft  = 0;  % Left Boundary Flow
    qbott  = k * (dx*1)*(T(i+n_x(j)) - 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/2)- dx %East Surface - Convective BC
       qtop   = k * (dx*1)*(T(i-n_x(j)) - T(i))/(dy);  % Top Boundary Flow
       qright = 0;  % Right Boundary Flow
       qleft  = k * (dy*1)*(T(i-1) - T(i))/(dx);  % Left Boundary Flow
       qbott  = k * (dx*1)*(T(i+n_x(j)) - 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) < (-W+d)+dy %South Surface - Adiabatic BC
       qtop   = k * (dx*1)*(T(i-n_x(j)) - T(i))/(dy);  % Top Boundary Flow
       qright = k * (dy*1)*(T(i+1) - T(i))/dx;  % Right Boundary Flow
       qleft  = k * (dy*1)*(T(i-1) - T(i))/(dx);  % Left Boundary Flow
       qbott  = k * (dx*1)*(T_bot - 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 * (dx*1)*(T(i-n_x(j)) - T(i))/(dy);  % Top Boundary Flow
       qright = k * (dy*1)*(T(i+1) - T(i))/dx;  % Right Boundary Flow
       qleft  = k * (dy*1)*(T(i-1) - T(i))/(dx);  % Left Boundary Flow
       qbott  = k * (dx*1)*(T(i+n_x(j)) - 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
   if sqrt((xx(i))^2+(yy(i))^2)<=(d/2)^2
        qtotal(i) = qtotal(i) + (q_dot_circle*dy*dx);
   end
end
```

Heat Rate Solution:

```
figure(4)
qtotall = reshape(qtotal,n_x(j),n_y)';
surf(X,Y,qtotall)
```




```
title('Heat Rate Surface');
xlabel('x-direction (m)');
ylabel('y-direction (m)');
zlabel('Heat Rate (q)');


qt = sum(qtotal)
qt = 3.2330e-13
qt = -1.4396e-12
qt = -1.9131e-12
qt = -3.9108e-12
qt = -7.8071e-13
```

```
qt = -1.4952e-12
qt = -8.4377e-13
qt = -5.8357e-12
qt = 4.2255e-12
qt = -8.4518e-12
qt = -5.5493e-12
qnorth = sum(qnorth);
qeast = sum(qeast);
qsouth = sum(qsouth);
qwest = sum(qwest);
```

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

```
Surfaces = 4×2 table
```

	Surface	q
1	"North"	25.8065
2	"East"	0
3	"South"	-185.8065
4	"West"	0

```
Surfaces = 4×2 table
```

	Surface	q
1	"North"	22.3503
2	"East"	0
3	"South"	-187.6396
4	"West"	0

```
Surfaces = 4×2 table
```

	Surface	q
1	"North"	57.1429
2	"East"	0
3	"South"	-168.2540
4	"West"	0

```
Surfaces = 4×2 table
```

	Surface	q
1	"North"	-8.3571
2	"East"	0
3	"South"	-204.6607
4	"West"	0

`Surfaces = 4×2 table`

	Surface	q
1	"North"	-28.7895
2	"East"	0
3	"South"	-216.1084
4	"West"	0

`Surfaces = 4×2 table`

	Surface	q
1	"North"	25.6228
2	"East"	0
3	"South"	-185.6228
4	"West"	0

`Surfaces = 4×2 table`

	Surface	q
1	"North"	8
2	"East"	0
3	"South"	-195.5000
4	"West"	0

`Surfaces = 4×2 table`

	Surface	q
1	"North"	12.8342
2	"East"	0
3	"South"	-192.7650
4	"West"	0

`Surfaces = 4×2 table`

	Surface	q
1	"North"	1.5779
2	"East"	0
3	"South"	-199.1088
4	"West"	0

```
Surfaces = 4×2 table
```

	Surface	q
1	"North"	-20.8757
2	"East"	0
3	"South"	-211.8113
4	"West"	0

```
Surfaces = 4×2 table
```

	Surface	q
1	"North"	25.5319
2	"East"	0
3	"South"	-185.5319
4	"West"	0

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

```
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
This Satisfies our Energy Balance Equation
```

Error Calculation

```
clear sum
```

```
for i = 1:N
    error(j) = sum(qtotal(i));
end


end


figure(6)
plot(n_x,error)
ylim("auto")
xlabel('Length in x-dir (m)')
ylabel('Total Error')
title('Error vs Domain')
```