

**Analysis of heat flux in a
2m x 2m
plate through the use of
Steady-State, Unsteady-State (Explicit), and Transient (Implicit)
finite difference methods**

By

Jon Dibacco and Michael Tanja

ME 344

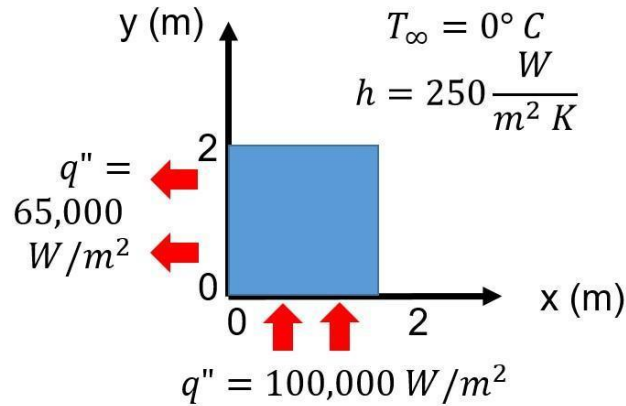
California Maritime Academy

4/22/18

Copywrite

Tanja, Dibacco, 2018

Part I: 2-D Steady State



Reporting

For our entire project, we focused on populating a square matrix of coefficient (usually referred to as matrix A or matrix C), then we inverted this matrix and multiplied it by a vector of constants (for this steady state case). To obtain the A matrix, we assume the following:

- the plate is so thin that when we calculate area of heat transfer, we use only the length of the sides of the square (as pictured below in figure [center node picture])
- each node is analyzed separated as a perfect square, that is $dx = dy$
- the material analyzed is Aluminum with a coefficient of conduction of 250 W/mK
- the surrounding temperature is $0^{\circ}C$ with a coefficient of convection of 250 W/m^2K
- the plate that we are analyzing is a $2\text{m} \times 2\text{m}$ plate

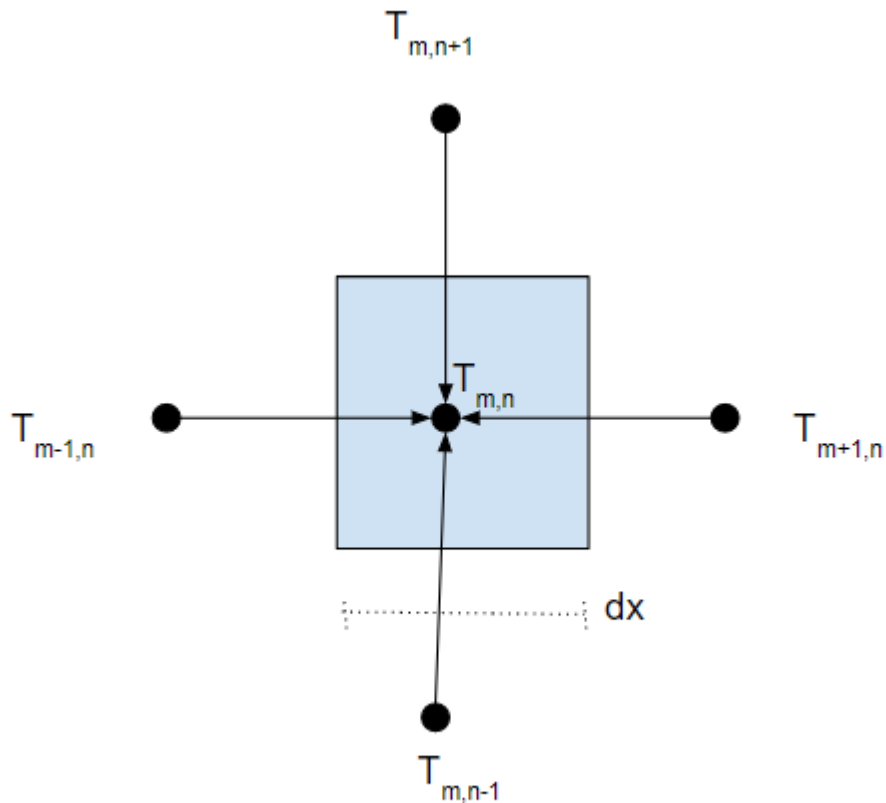


Figure 1 General schematic for node calculation

To be more specific to our group, we assumed that the heat for all of our nodes (Top, bottom, right, left, and all four corners) is flowing into the center node. This made our derivations look a little bit cleaner, and our code was therefore a little cleaner as well. In our derivations, we ordered the coefficients before all five of the points in our nodes in the following order:

1. Heat flow from the top ($T_{m,n+1}$)
2. Heat flow from the bottom ($T_{m,n-1}$)
3. Heat flow from the right ($T_{m+1,n}$)
4. Heat flow from the left ($T_{m-1,n}$)
5. Center point ($T_{m,n}$)

We would then add whatever constants we had in our derivations (i.e. surrounding temperatures and heat sources)

This scenario was defined as the sum of all of the heats from all four directions flowing into the center point of our node. For steady state, that means that the sum of the heat flows and whatever constants we have will be set equal to zero [please refer to scenario 1 derivations in the appendix]

After these derivations were made, they were coded into MATLAB and plotted. The system of equations for our coefficient matrix and constant vectors could be found in the calculations (circled in red ink)

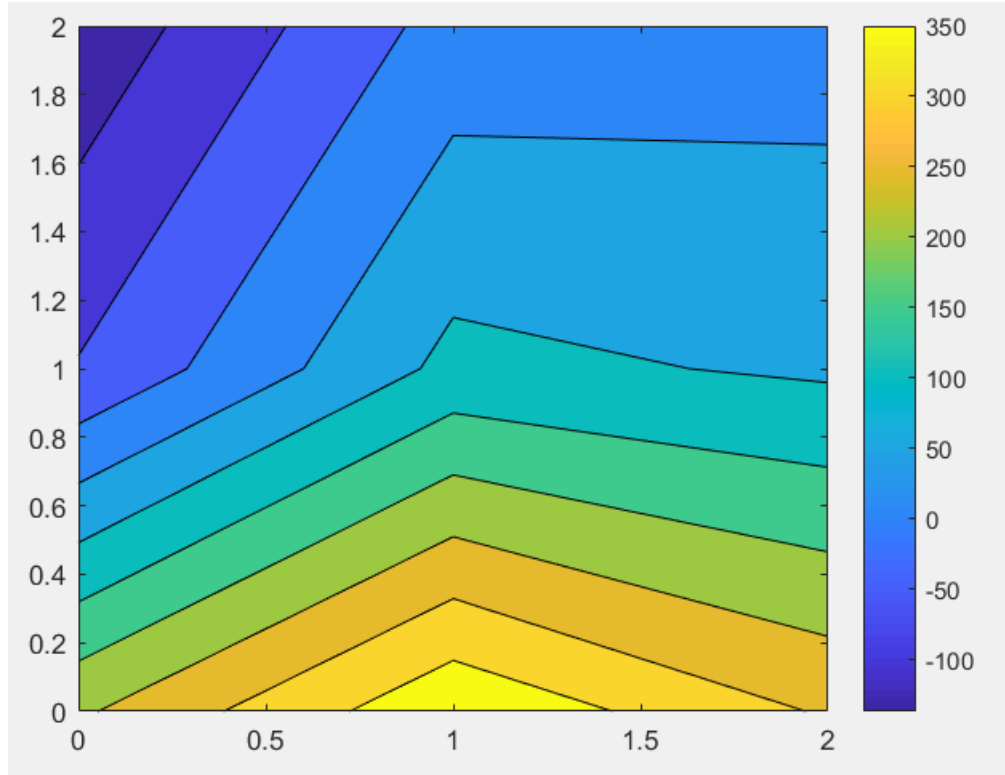
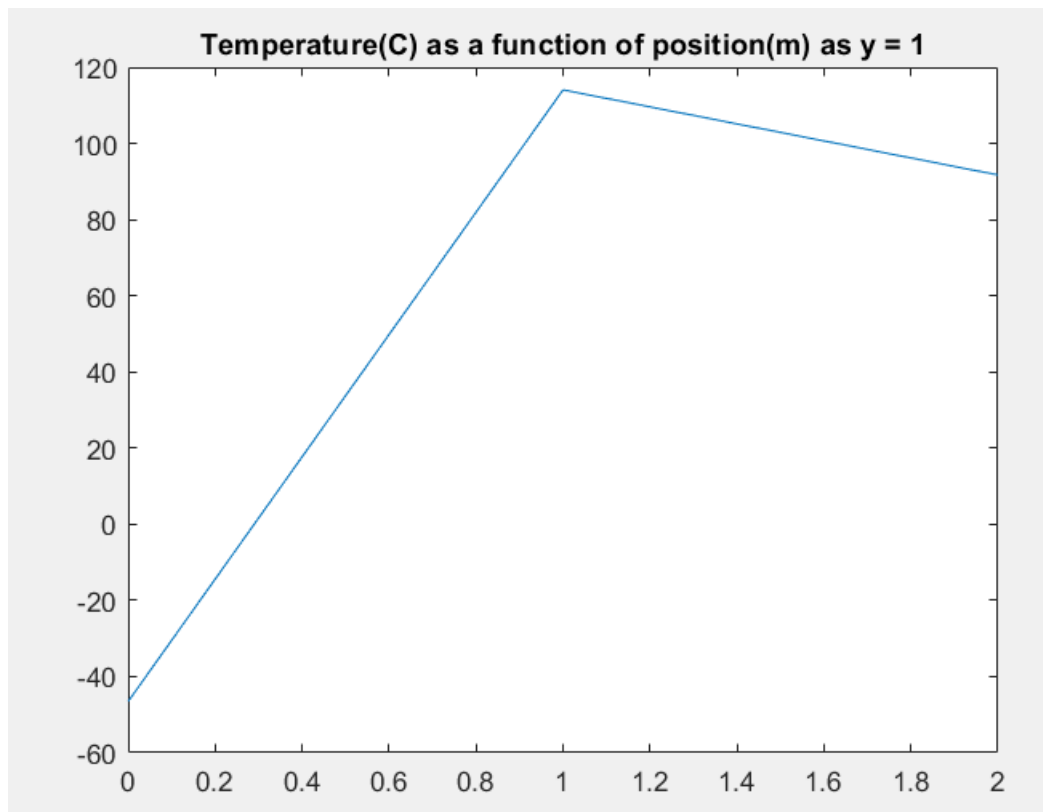
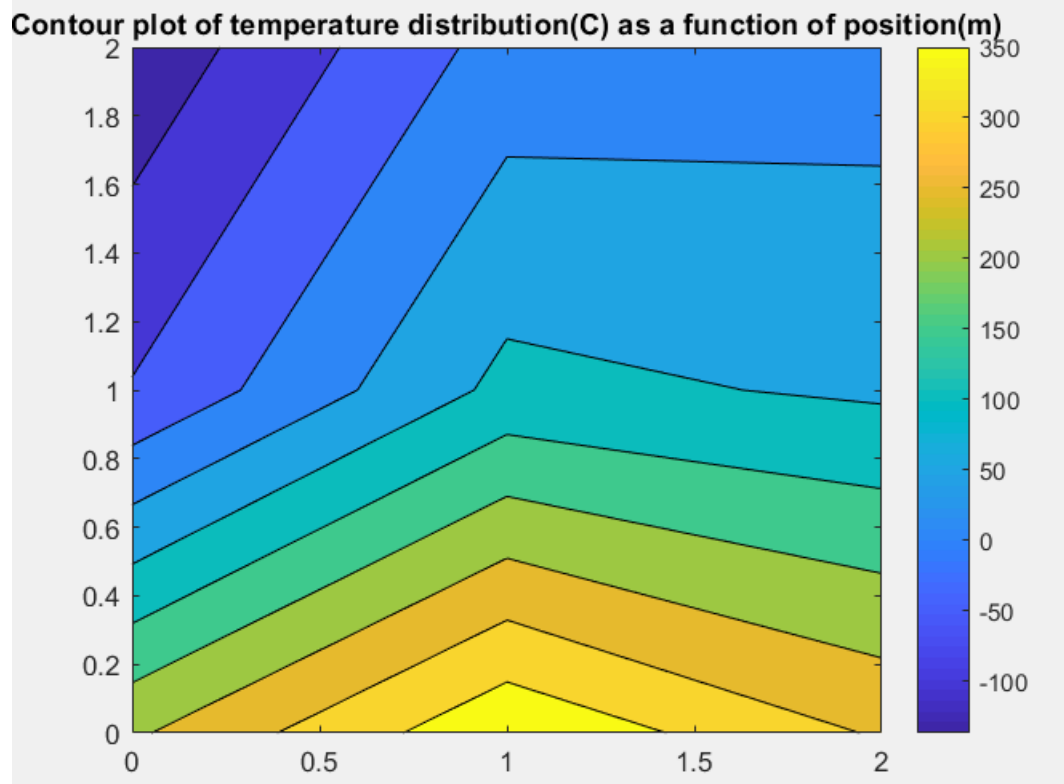


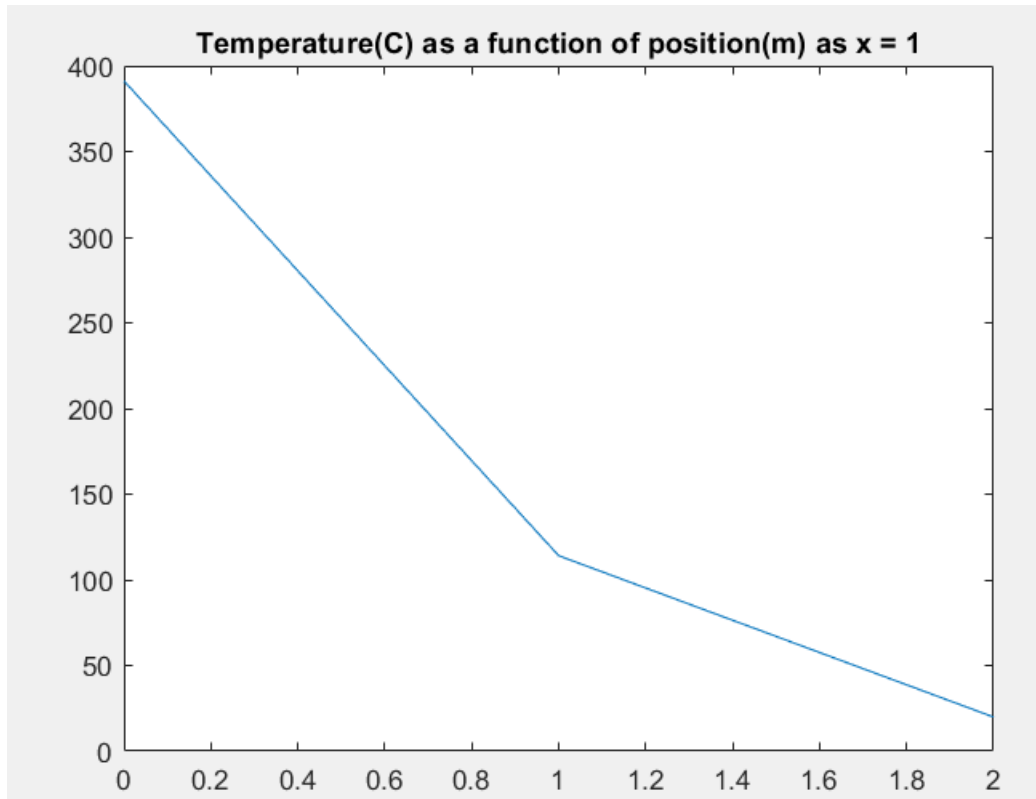
Figure 2 Case 1, Steady state

As we stated previously, the code was written in the order that we stated the coefficients before their respective temperatures. In our code, we did nine different “for” loops: four for the corners, four for the sides, and one for the center nodes. Each node was initiated by the code through different algebraic manipulations of the number of rows and columns determined by the user in the beginning of the code. For example, the bottom left corner, which we labeled as corner 3, is always going to be a value of one. The top right corner, which is known to us as corner 2, is “nNodes”, which is calculated as the number of rows and columns in the square. At the end of the calculations, line 162 in the M.file, the “T” matrix is constructed, and calculated as the inverse of “A” multiplied by our constant “b” vector. The M.file for part 1 can be found in our appendix.

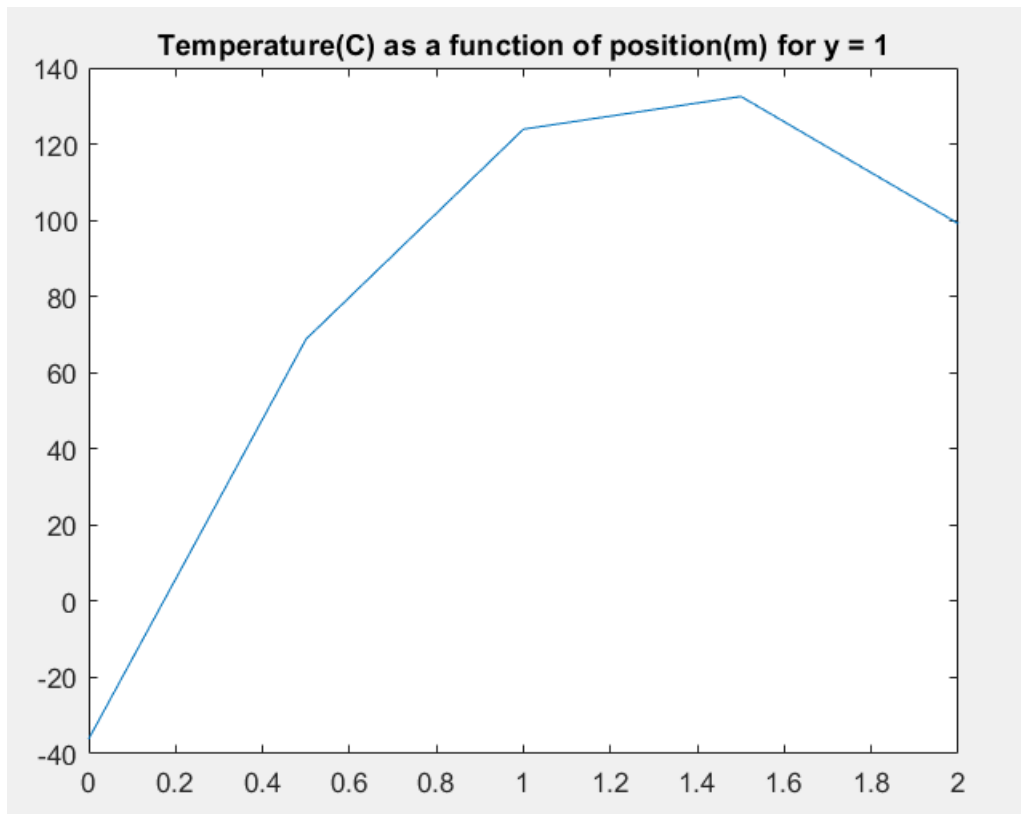
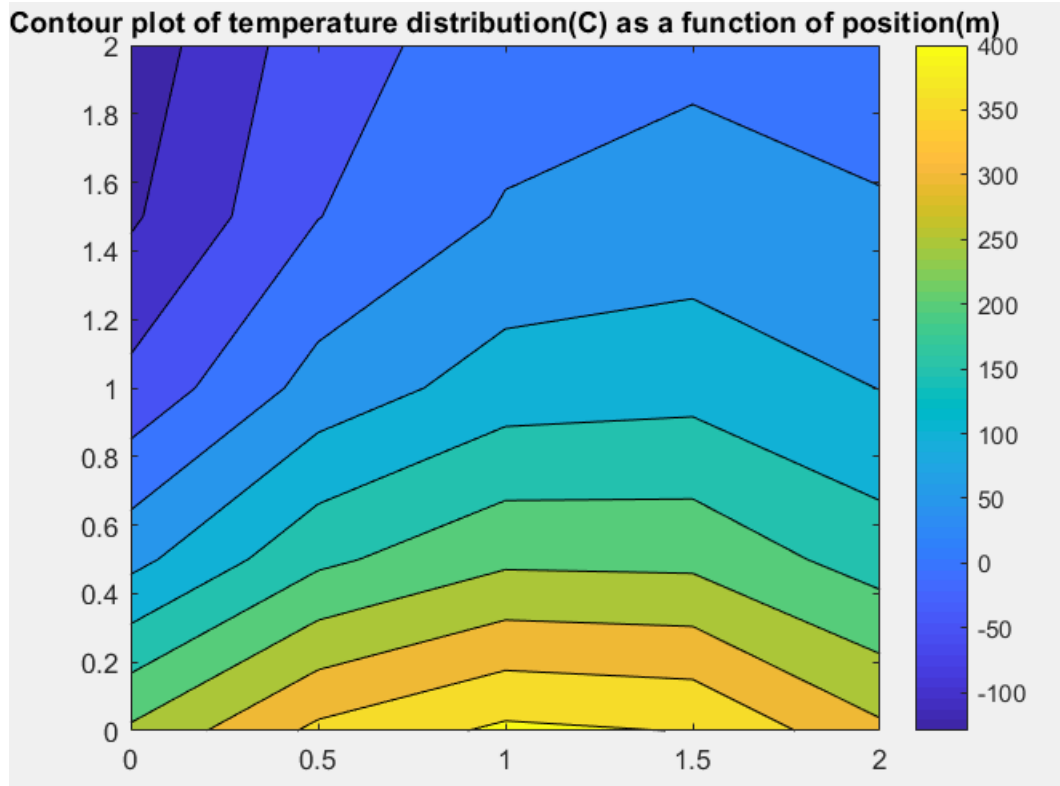
Result

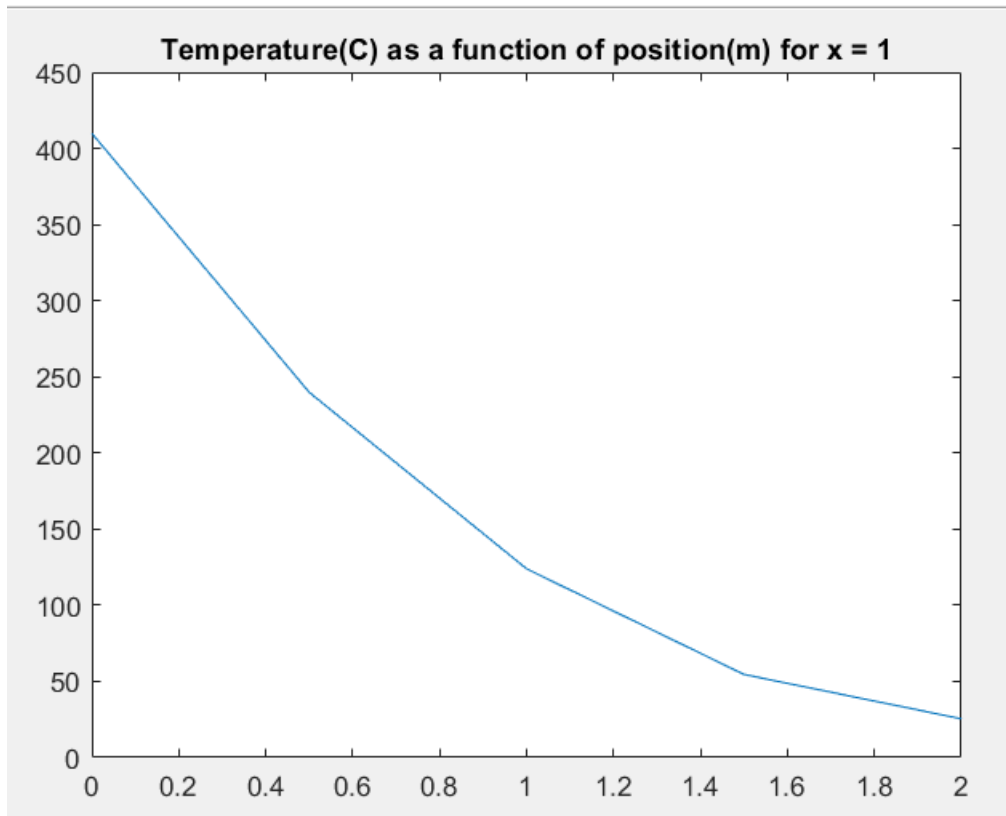
Case 1



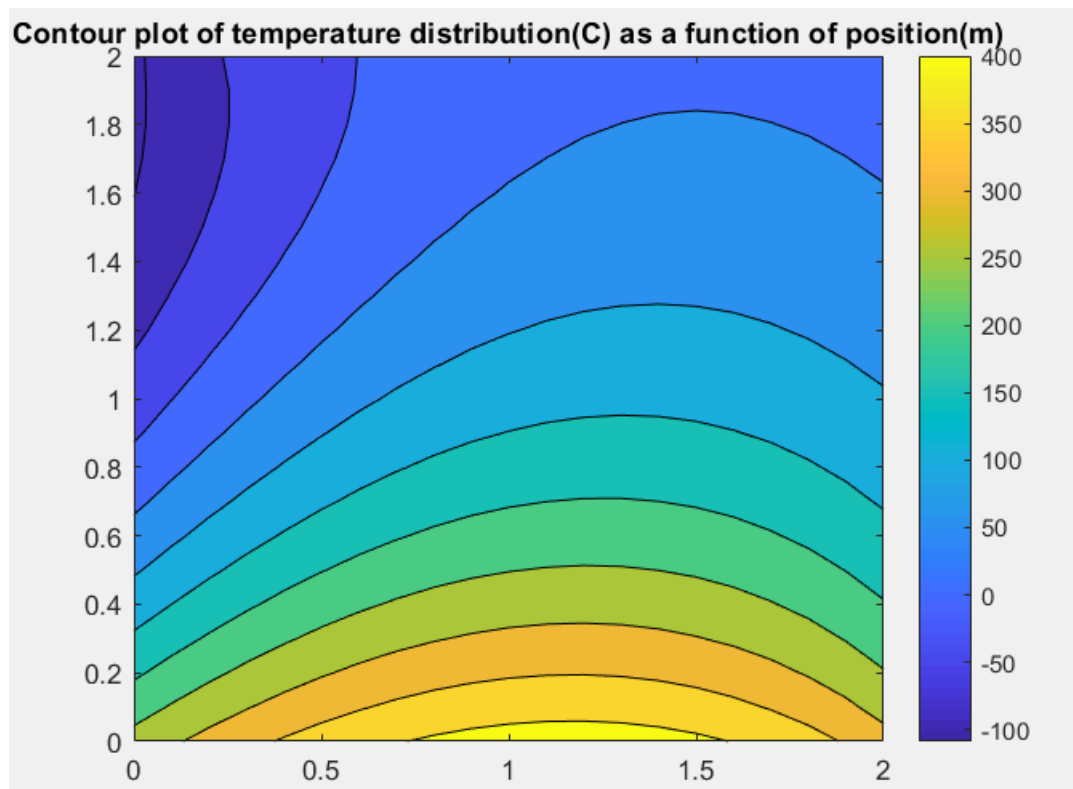


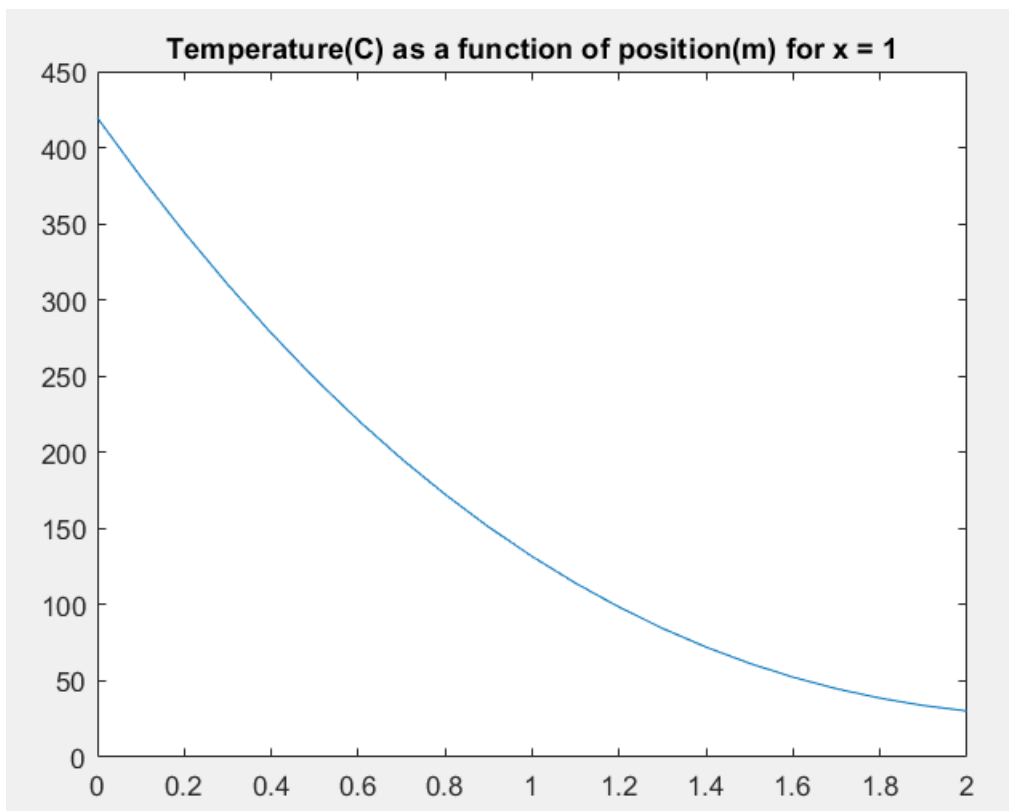
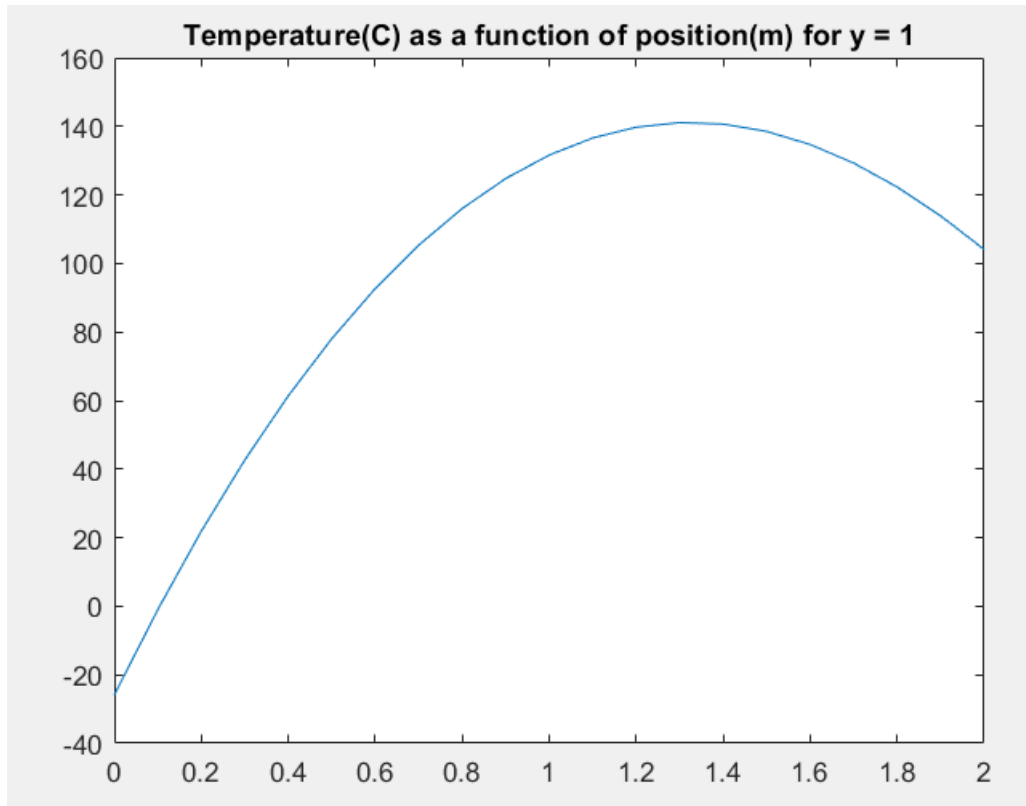
Case 2





Case 3





Discussion

As the figures suggest, the overall heat flow seems to be coming from the bottom nodes where the heat source of $100000 \text{ W/m}^2\text{K}$. As we go up along the y axis at any point 'x' in the contour plot, it gets colder and colder. The contour plot also suggests that the heat sink on the left is cooling the plate off at a faster rate than the surround temperature. Increasing the mesh size will increase the resolution of the contour plots.

for Section B our A matrix was not invertible we tried to troubleshoot but we couldn't identify the problem. our resultant A matrix looked exactly the same as some of our peers that we tried to compare against, but when we tried to take the inverse it just wouldn't work.

Part 2: Unsteady-State, Explicit Method

Setup

In this problem, we are analyzing the same plate. The difference is that we have a time variance, as opposed to just the positional variance that was present in the first problem. The overall setup for the numerical method is the explicit finite element analysis, which we learned last semester in Dr. Julie Simons' numerical methods class. Pictured below is the general method which we used to derive and solve this problem:

$$M \frac{\vec{u}^{k+1} - \vec{u}^k}{\Delta t} = A \vec{u}^k + \vec{b} \quad \text{explicit}$$

$$M \vec{u}^{k+1} = M \vec{u}^k - \Delta t A \vec{u}^k + \Delta t \vec{b}$$

$$\vec{u}^{k+1} = \vec{u}^k - \Delta t M^{-1} A \vec{u}^k + \Delta t \vec{b}$$

$$\vec{u}^{k+1} = \underbrace{(I - \Delta t M^{-1} A)}_{\text{matrix}} \underbrace{\vec{u}^k}_{\text{vector}} + \underbrace{\Delta t \vec{b}}_{\text{vector}} \quad \text{Explicit Scheme}$$

In our derivations, the coefficient matrix will be a combination and algebraic manipulation of our constants. We are given initial temperatures of 400 degrees celsius for the entire time, for time zero. In this section, we also assumed that the summation of all the heat in one node is equal to the lumped capacitance of a metal, which is calculated as:

$$\rho V c \frac{d\theta}{dt}$$

where $d\theta$ is simply our central node's future temperature minus our central node's present temperature. In our derivations for part 2 in the appendix, you can see that from the above equation, we can derive a fourier number term, F_o . An example of our derivation method can be seen in the picture below:

Corner 2

$$\frac{h\Delta x}{2}(T_{\infty}-T_{m,n}) + \frac{k\Delta x}{\Delta y^2}(T_{m,n-1}-T_{m,n}) + \frac{h\Delta x}{2}(T_{\infty}-T_{m,n}) + \frac{k\Delta x}{\Delta x^2}(T_{m-1,n}-T_{m,n}) = \frac{\rho\Delta x^2 c}{4\Delta t}(T'_{m,n}-T_{m,n})$$

$$\frac{h\Delta x}{k}(T_{\infty}-T_{m,n}) + (T_{m,n-1}-T_{m,n}) + \frac{h\Delta x}{k}(T_{\infty}-T_{m,n}) + (T_{m-1,n}-T_{m,n}) = \frac{2\rho\Delta x^2 c}{4k\Delta t}(T'_{m,n}-T_{m,n}) \quad \frac{1}{2F_0}$$

$$\frac{2F_0 h\Delta x}{k}(T_{\infty}-T_{m,n}) + 2F_0(T_{m,n-1}-T_{m,n}) + \frac{2F_0 h\Delta x}{k}(T_{\infty}-T_{m,n}) + 2F_0(T_{m-1,n}-T_{m,n}) + 2T_{m,n} = T'_{m,n}$$

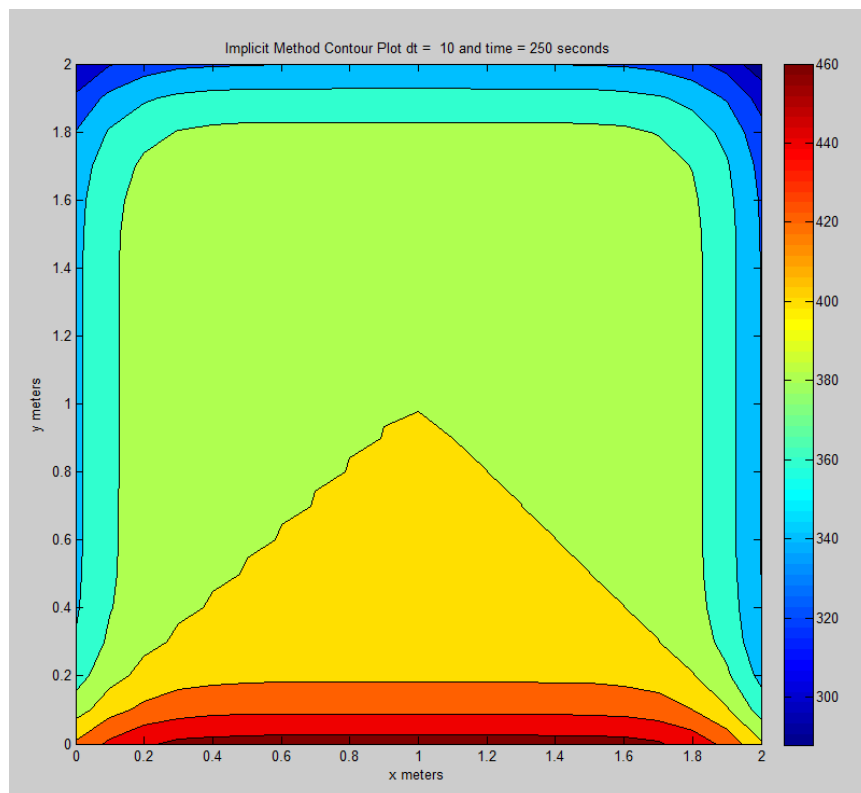
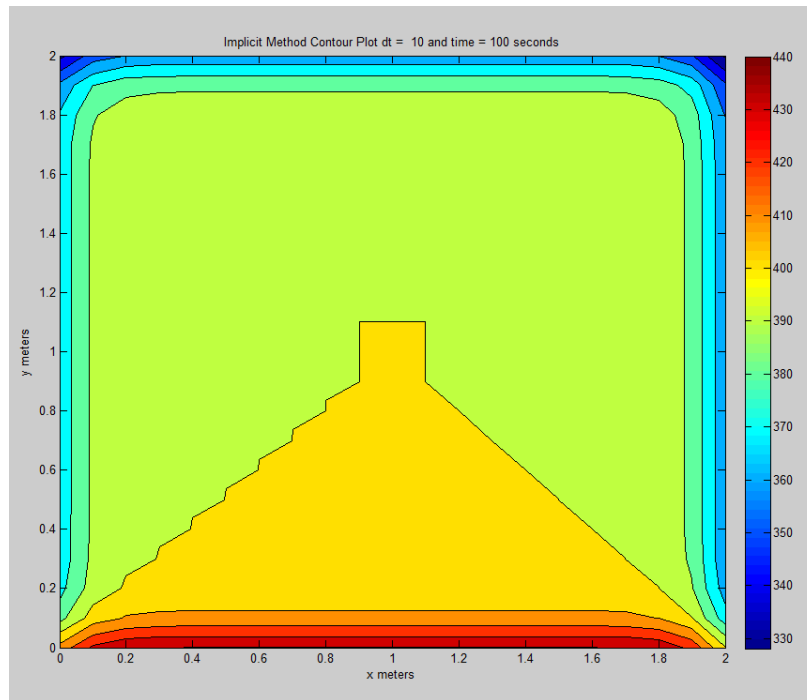
$$2T_{\infty} + 2F_0 T_{m,n-1} + 2T_{m,n} + 2F_0 T_{m,n} + \left(-\frac{4F_0 h\Delta x}{k} - 4F_0 + 1\right) + \frac{4F_0 h\Delta x}{k} T_{\infty} = T'_{m,n}$$

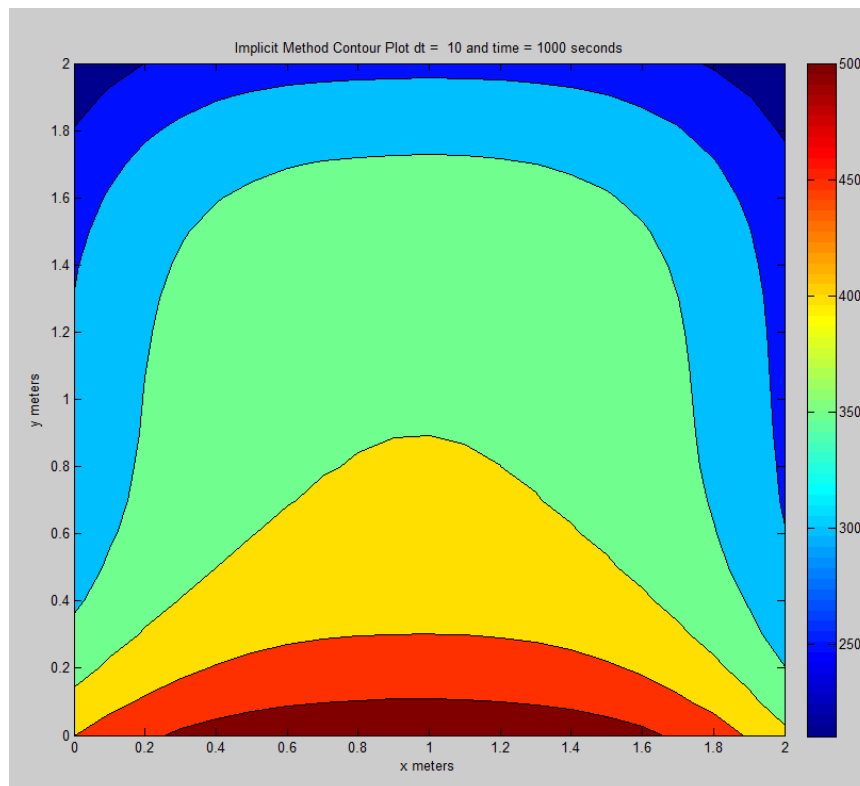
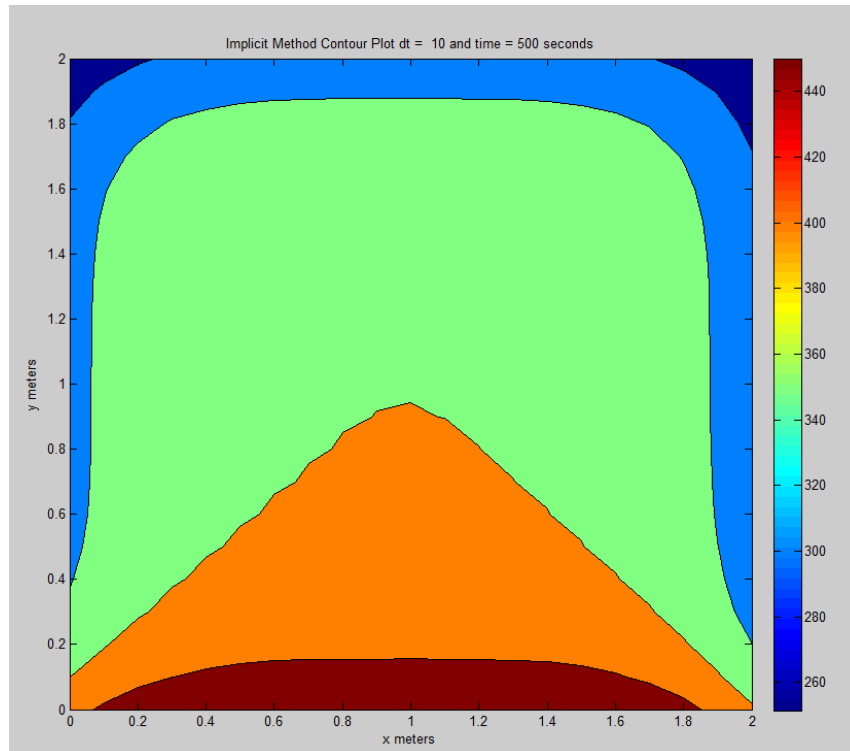
The explicit method allows us to solve one temperature value by using current temperature to solve for the future temperature. The future temperature can be seen in Dr. Simons' work as u^{k+1} , whereas the present temperature is u^k . The 'b' vector is made of constants of heat sources and the surrounding temperature, whereas the matrix multiplied by our u^k vector is made up of different heat coefficients multiplied or divided by our 'dx' or F_0 .

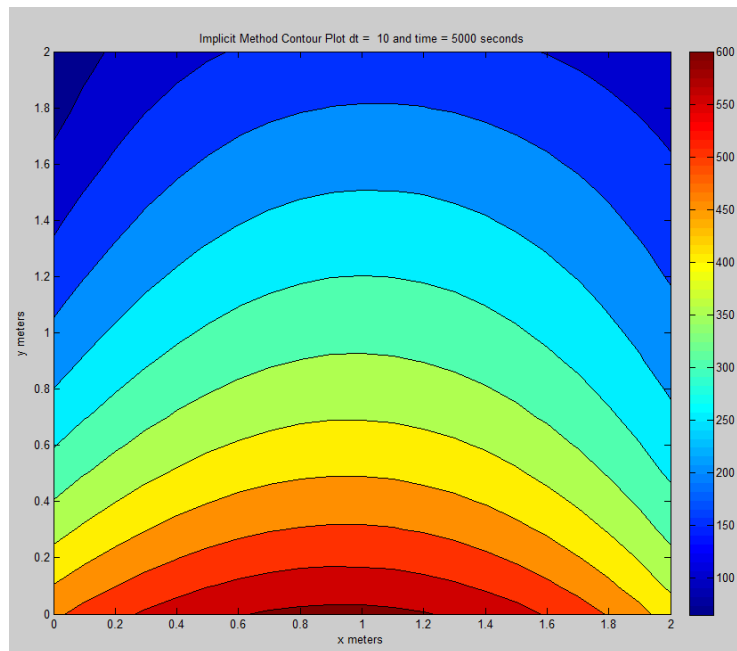
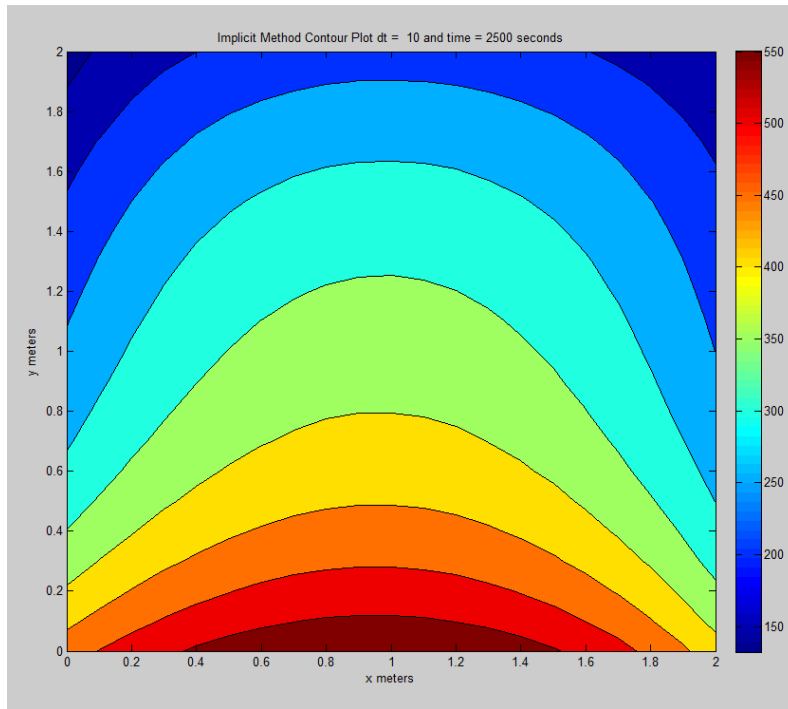
The limitation for our code seems to be at high resolution ($dx = 0.1$) and high time step ($dt = 50s$). This is because the Fourier number, F_0 , is equal to 0.5. According to the stability requirement for explicit methods, the Fourier number cannot exceed 0.25.

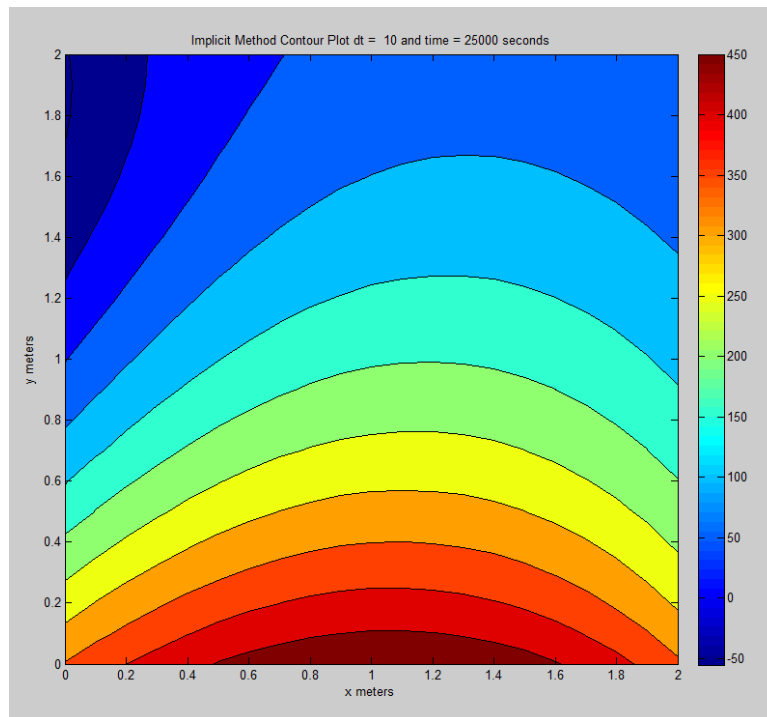
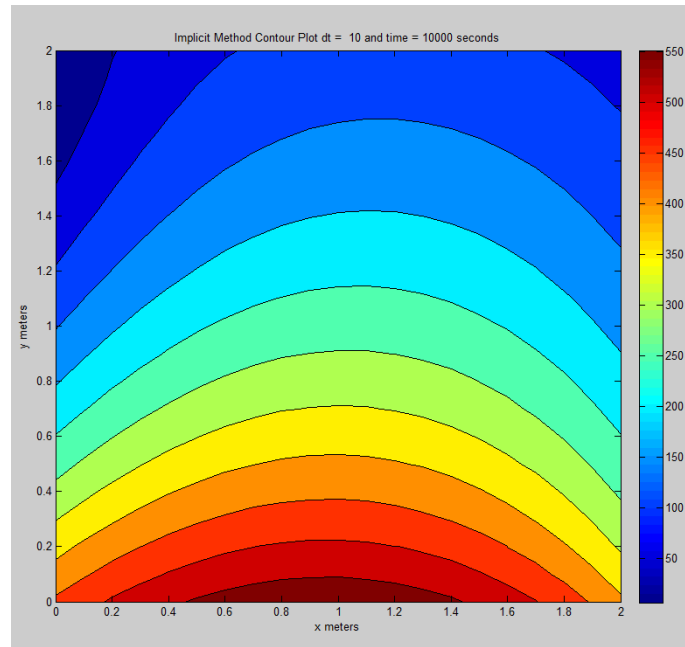
Our code is heavily based on the previous code. The orientation of the derivations and the method of how the code designated the notes remain the same. The difference is in the coefficients of each node. Now that there is a time step, that is taken into account when making the derivations. The constant 'b' vector is also made to be positive, whereas the 'b' vector in the previous problem was negative so the temperature equations were kept separate from the constant equations. In this scenario, the future term was kept separate from the present terms and the constant terms. An end time was set at 50000s and the dt could be set by the user. As the code ran, it would store a certain 'TMesh' temperature for all the nodes, and it would add another vector for every dt step it took. The first vector in 'TMesh' is a vector of 400, which corresponds to the entire plate starting at 400 degrees celsius.

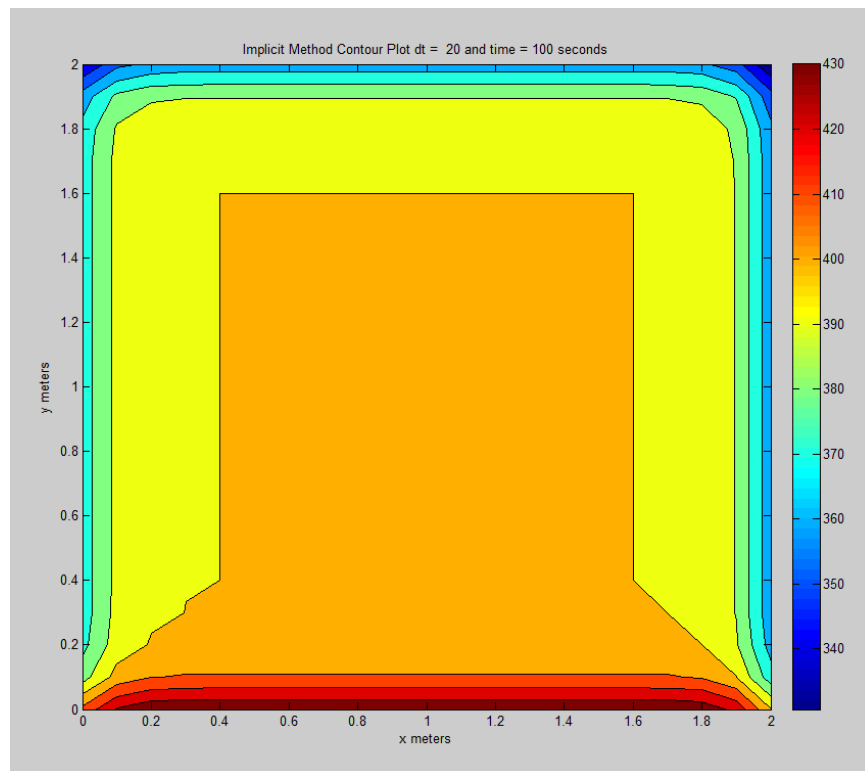
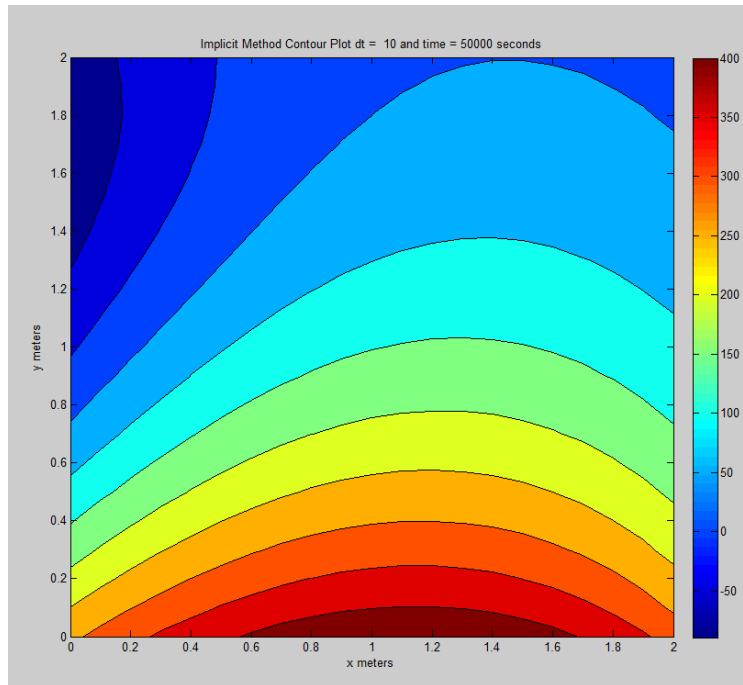
Results

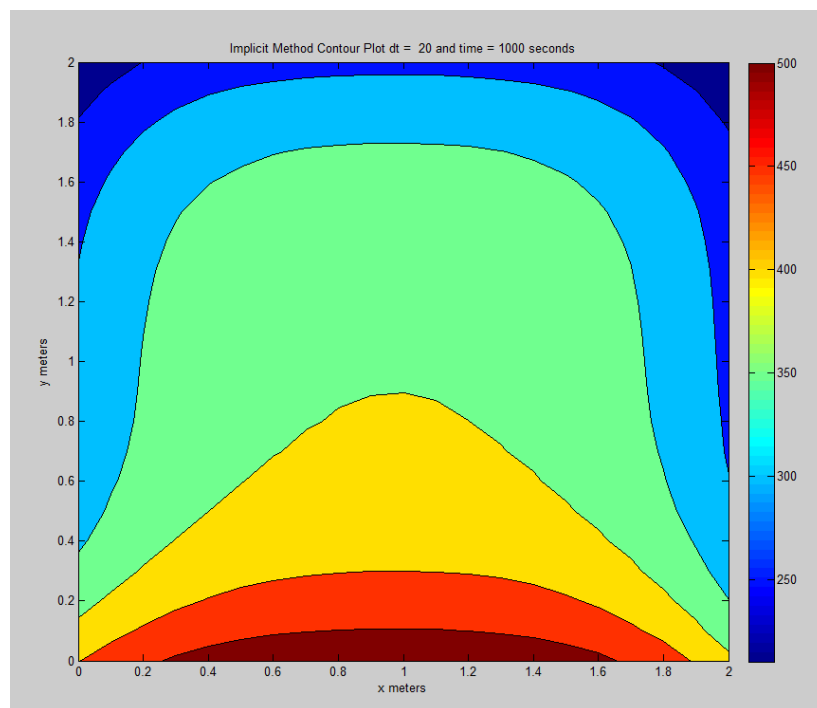
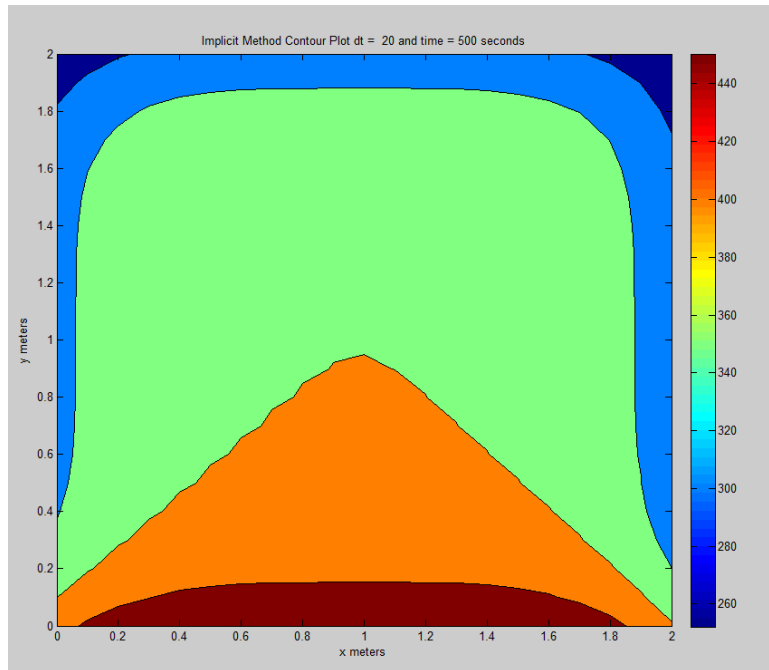


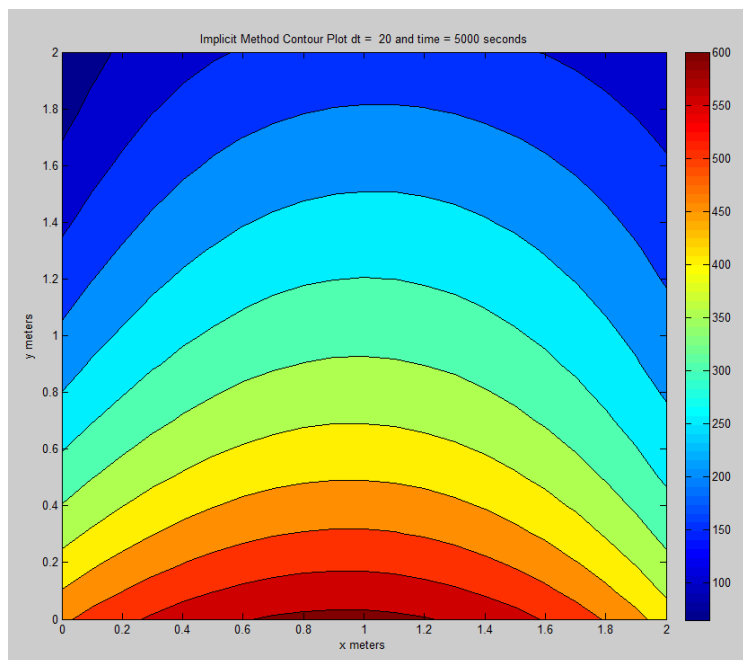
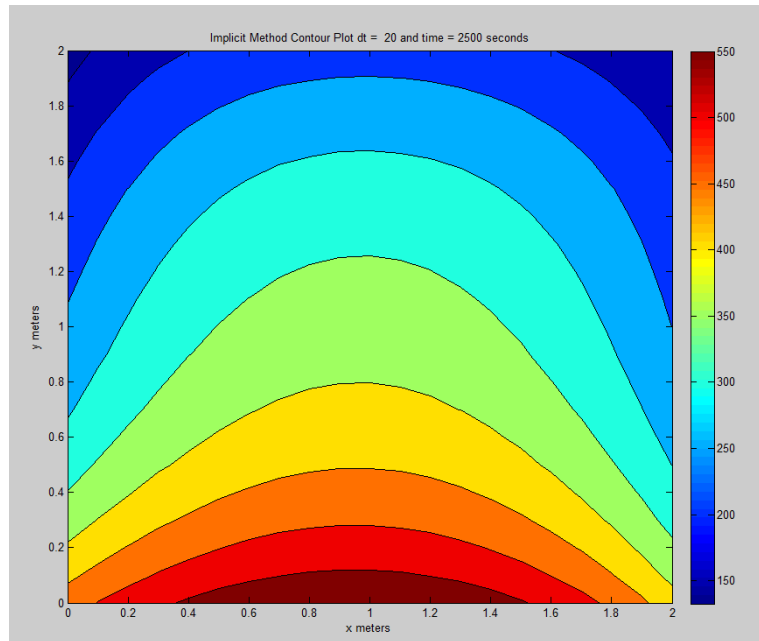


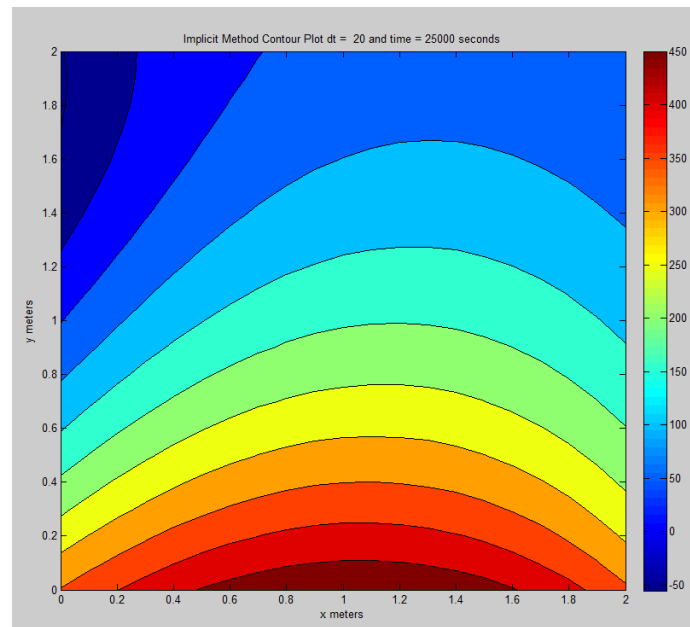
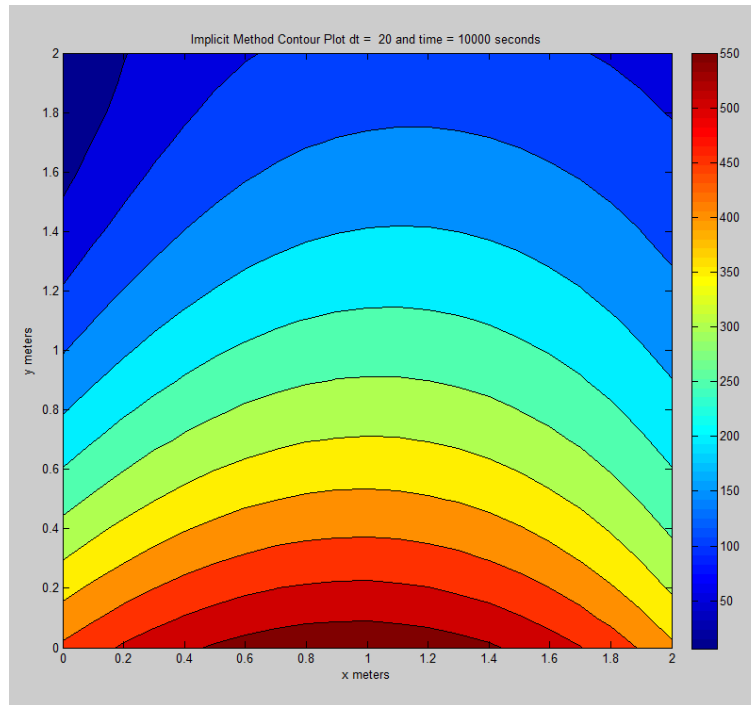


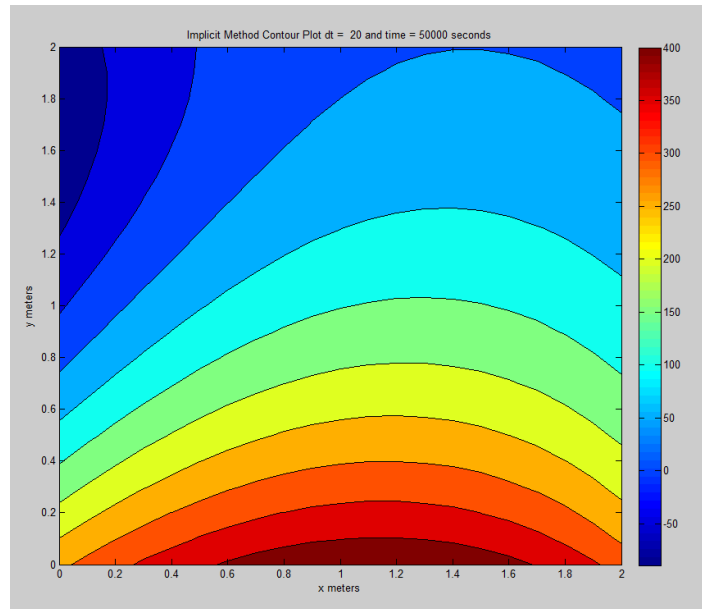




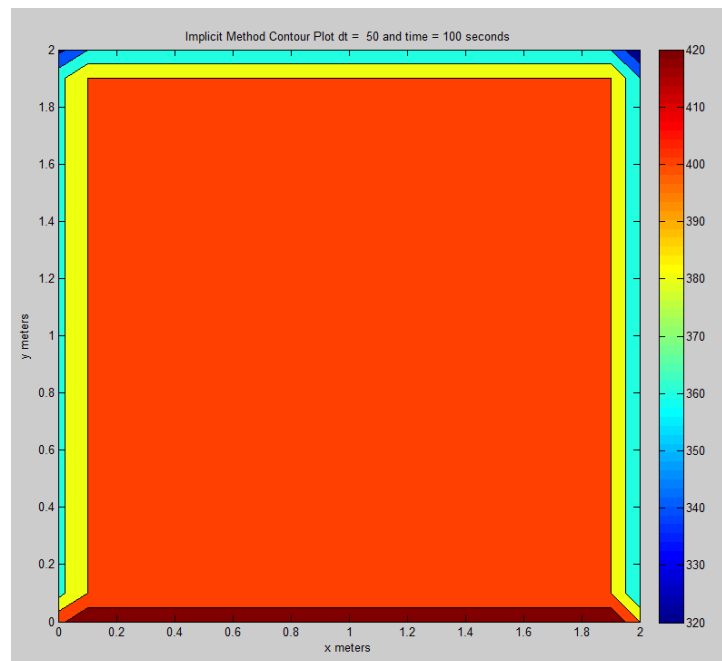


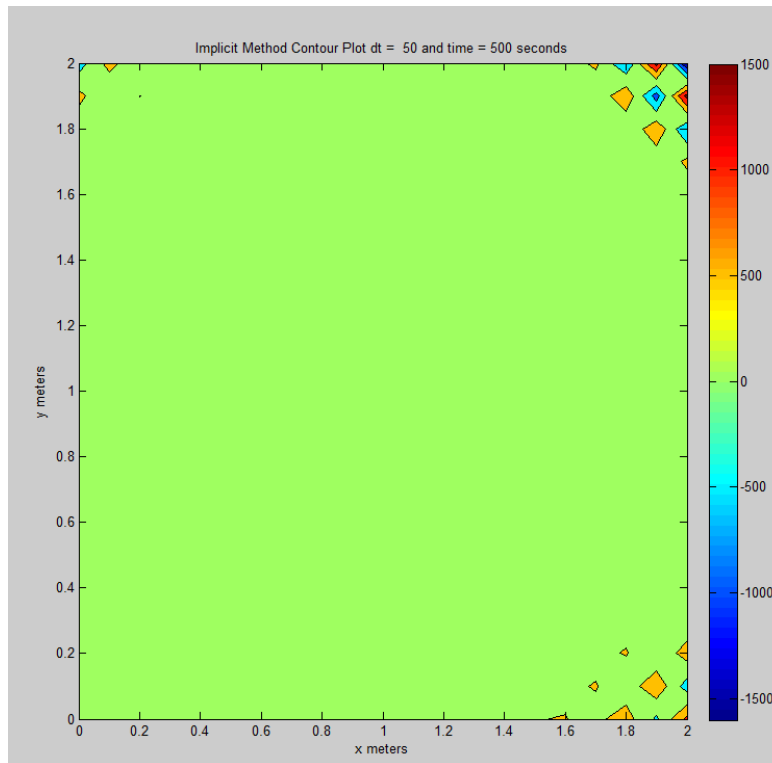
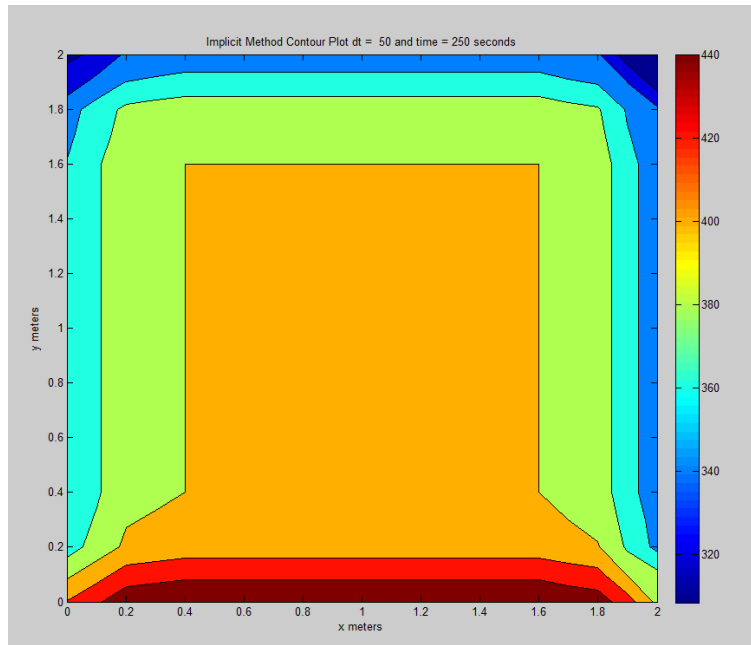


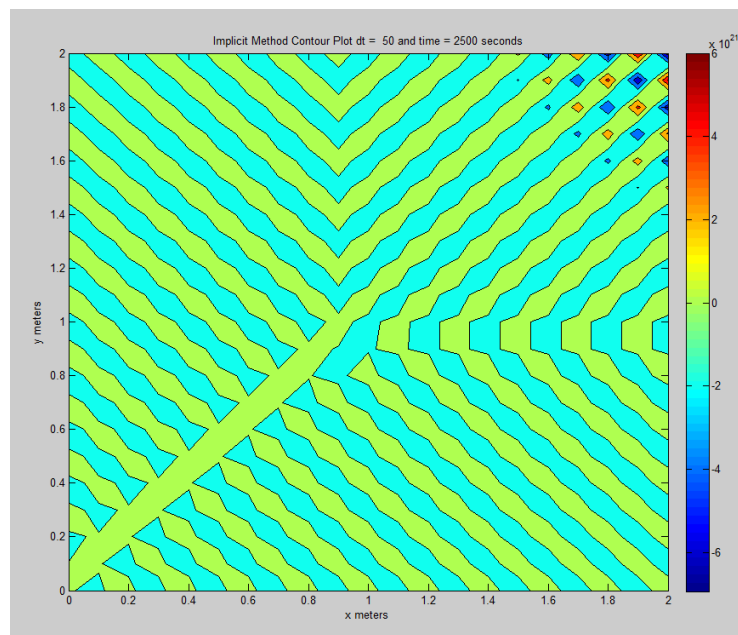
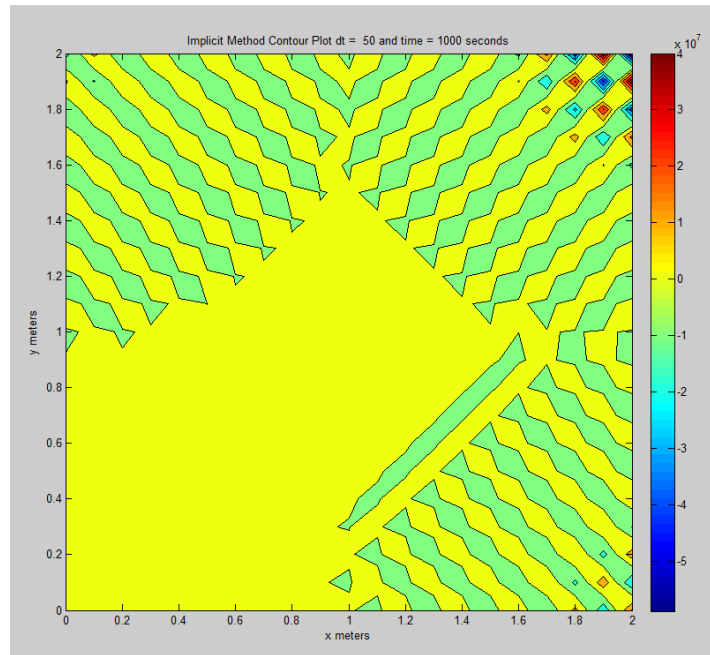


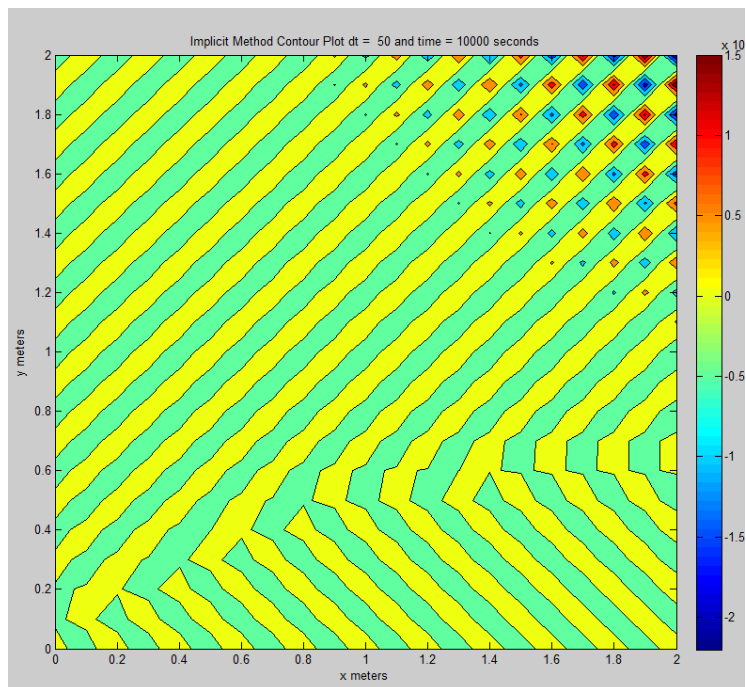
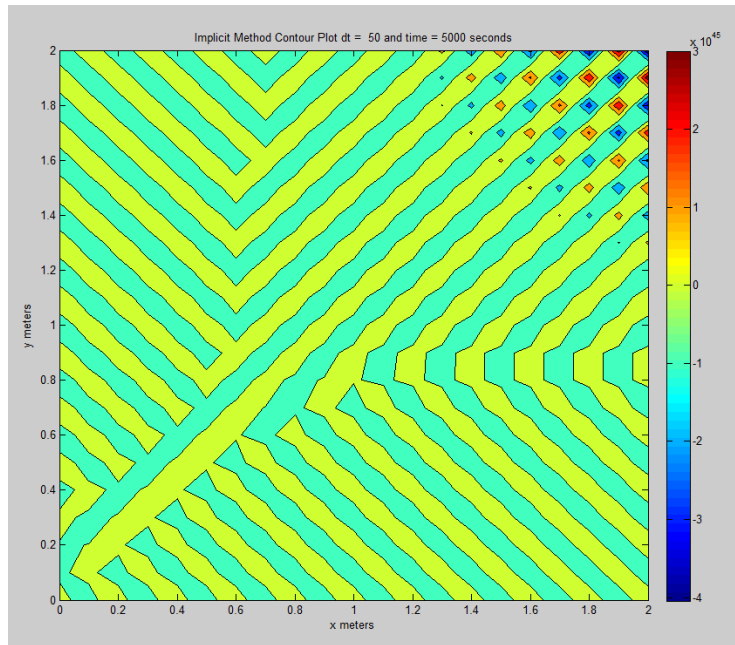


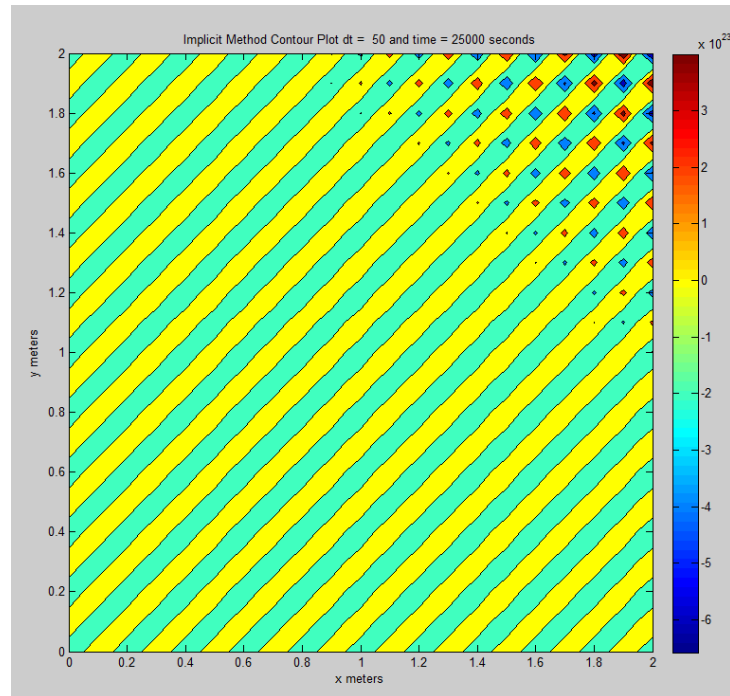
Below the following contour, the F_o starts to approach 0.25, and the contour begins to be unstable. We have no plot for $dt = 50$ and $endTime = 50000s$ because at that point, our F_o exceed 0.25 and the plot becomes unstable.











Discussion

When the code is stable, that is, when the Fourier number does not exceed 0.25, the heat flow seems to be similar to the first scenario. Again, we see that the heat source is generating a higher temperature throughout the plate, and the heat will flow into the top and right side of the plate, while the heat sink on the left will pull in most of the heat. Increasing dt will decrease the time it takes to solve a problem. If you have an x amount of time, and dt was increased every time you ran the code, it would increase the time intervals between 1 and your endtime, x . Increasing or decreasing dt will also affect the numerical value of your Fourier number, thus affecting the stability of the code.

Part 3: 2-D, Transient, Implicit

Setup

Similar to the previous problems, we are analyzing the same plate. And also similar to the last problem, we are using a time variance and a positional variance to analyze the behaviour of

heat in such a plate. The overall setup for the numerical method is the implicit finite element analysis, which was also a topic that was discussed in our numerical methods class. Pictured below is the general method which we used to derive and solve this problem:

Suppose $N=0$ for now.

$$M \frac{\vec{u}^{k+1} - \vec{u}^k}{\Delta t} = A \vec{u}^{k+1} + \vec{b}$$

$$\underbrace{(M - \Delta t A)}_K \vec{u}^{k+1} = M \vec{u}^k + \Delta t \vec{b}$$

$$\vec{u}^{k+1} = \underbrace{K^{-1} M}_{\text{matrix}} \vec{u}^k + \Delta t \underbrace{K^{-1}}_{\text{matrix}} \underbrace{\vec{b}}_{\text{vector}}$$

Implicit Scheme

As we can see from the scheme, the difference between the implicit method and the explicit method is the solving of a future temperature matrix by using a known vector value. The method used for constructing some coefficient matrix 'C' (or 'K' as Dr. Simons' notes suggest) is slightly altered. A pattern that we saw was that the coefficients that were positive in the 2nd problem are now negative numbers. An example of how we derived the coefficients for our 'C' matrix can be seen in the following figure:

Right

$$\frac{k\Delta x}{\Delta y^2}(T_1' - T_c') + \frac{k\Delta x}{\Delta y^2}(T_2' - T_c') + h\Delta y(T_{\infty} - T_c') + \frac{k\Delta x}{\Delta x}(T_4' - T_c') = \frac{\rho\Delta x^2 c}{2\Delta t}(T_c' - T_c^0)$$

$$(T_1' - T_c') + (T_2' - T_c') + \frac{2h\Delta x}{k}(T_{\infty} - T_c') + 2(T_4' - T_c') = \frac{1}{F_0}(T_c' - T_c^0)$$

$$F_0(T_1' - T_c') + F_0(T_2' - T_c') + \frac{2F_0h\Delta x}{k}(T_{\infty} - T_c') + 2F_0(T_4' - T_c') - T_c' = T_c^0$$

$$(-F_0)(T_1' - T_c') + (-F_0)(T_2' - T_c') + (-\frac{2F_0h\Delta x}{k})(T_{\infty} - T_c') + (-2F_0)(T_4' - T_c') + T_c' = T_c^0$$

$$(-F_0)T_1' + (-F_0)T_2' + T_3' + (-2F_0)T_4' + (4F_0 + \frac{2F_0h\Delta x}{k} + 1)T_c' + \left[-\frac{2F_0h\Delta x}{k}T_{\infty}\right] = T_c^0$$

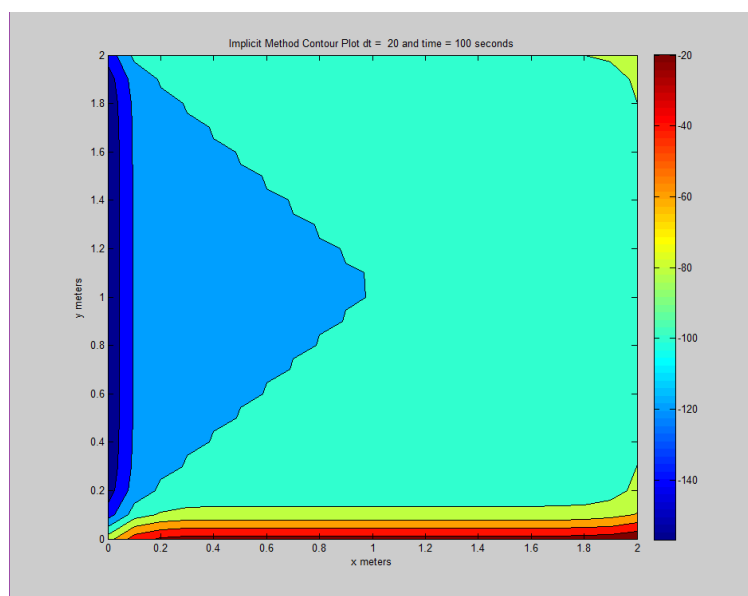
You can see the stark difference in this figure, as the 'C' matrix that was constructed of current temperatures in the last example, is now a matrix of future temperatures with negative coefficients.

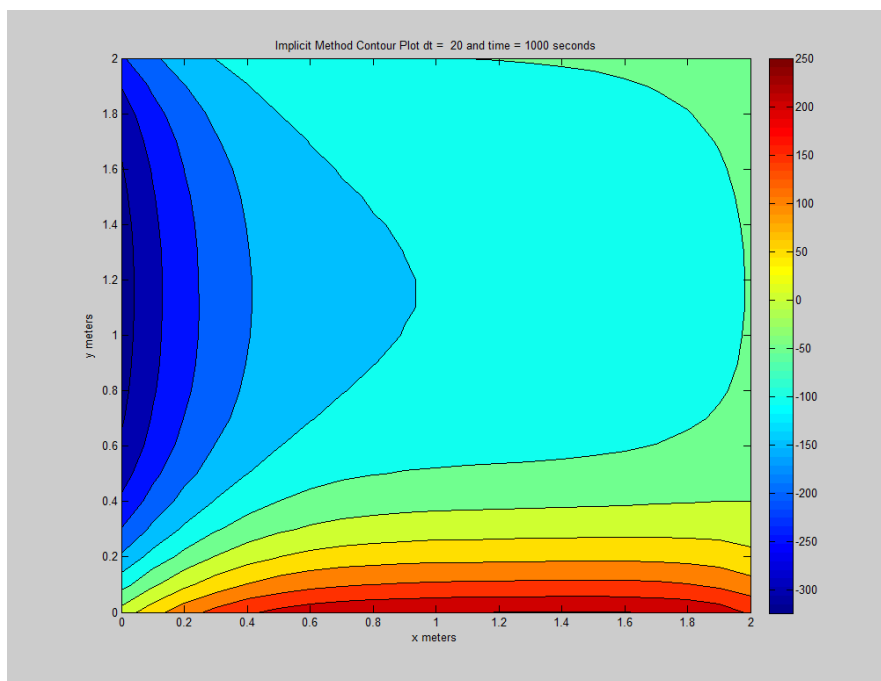
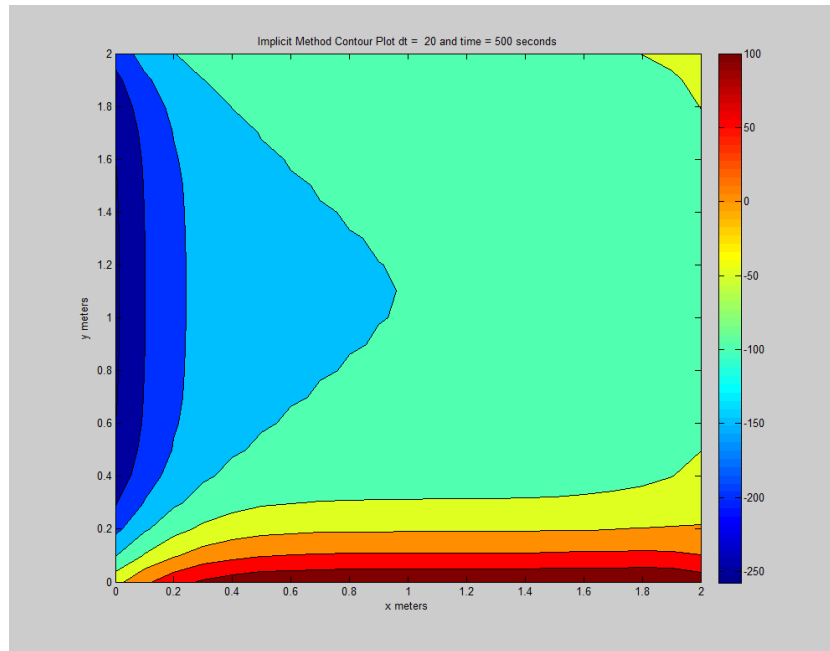
In our code, the setup was very similar to the previous example. The only things that changed are the coefficients in front of every node. As previously stated, they are the negative of their counterpart coefficients in part two. The last difference is that the calculation at the end inverts

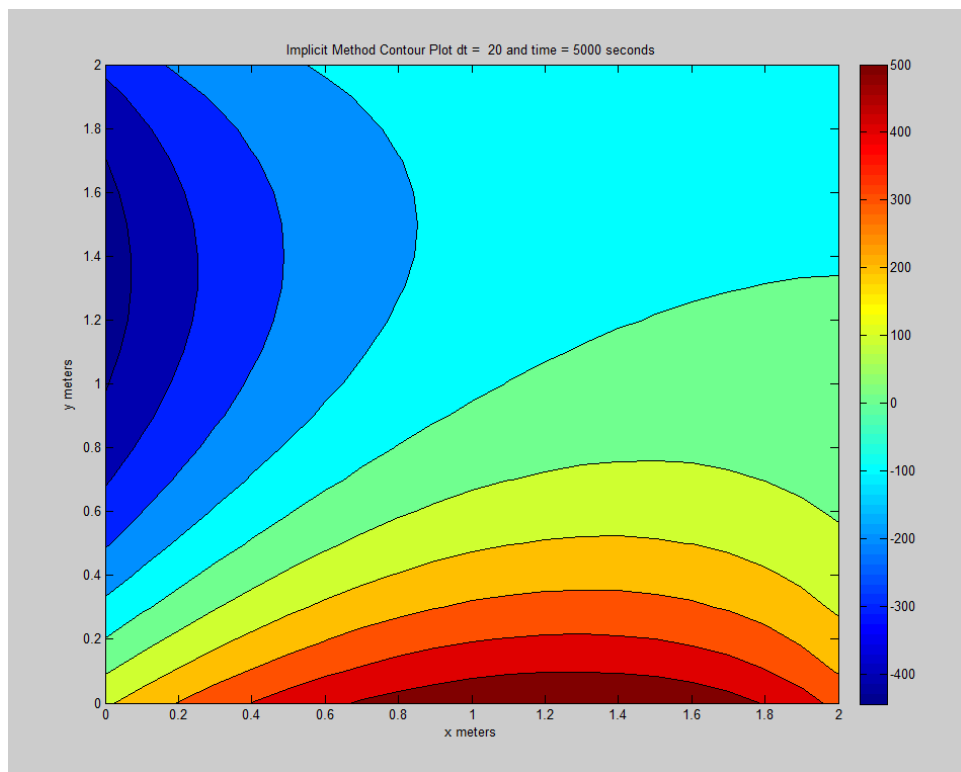
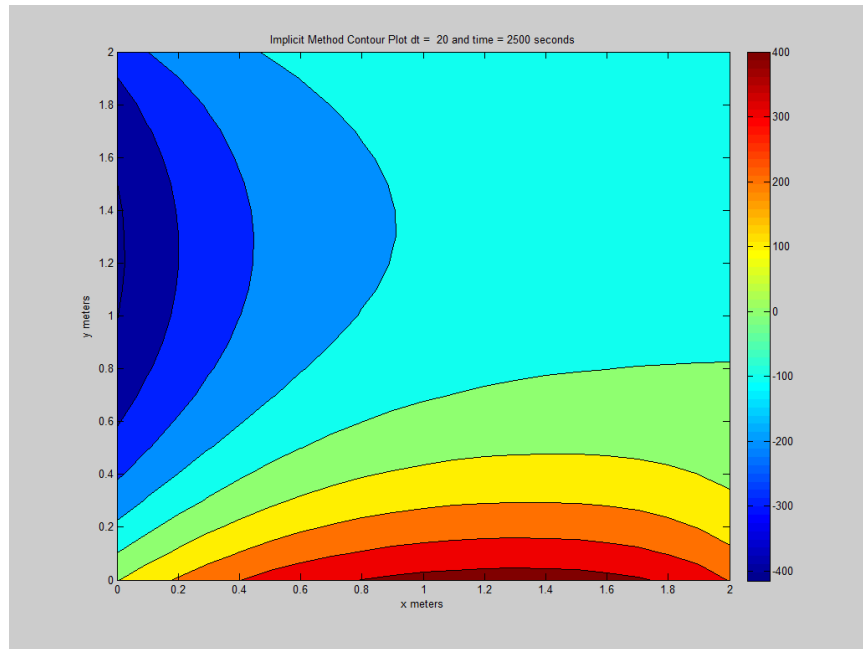
the 'C' matrix and distributes it to the present temperature vector u^k and the constant b vector. A picture of this matlab calculation can be seen below, and the complete matlab code can be referred to in the appendix:

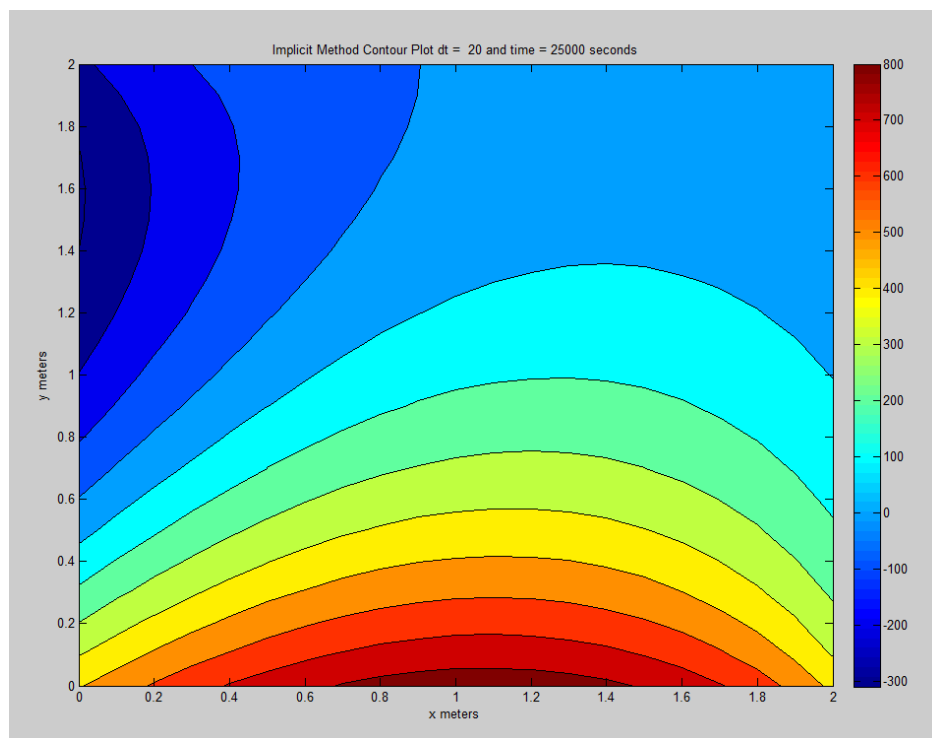
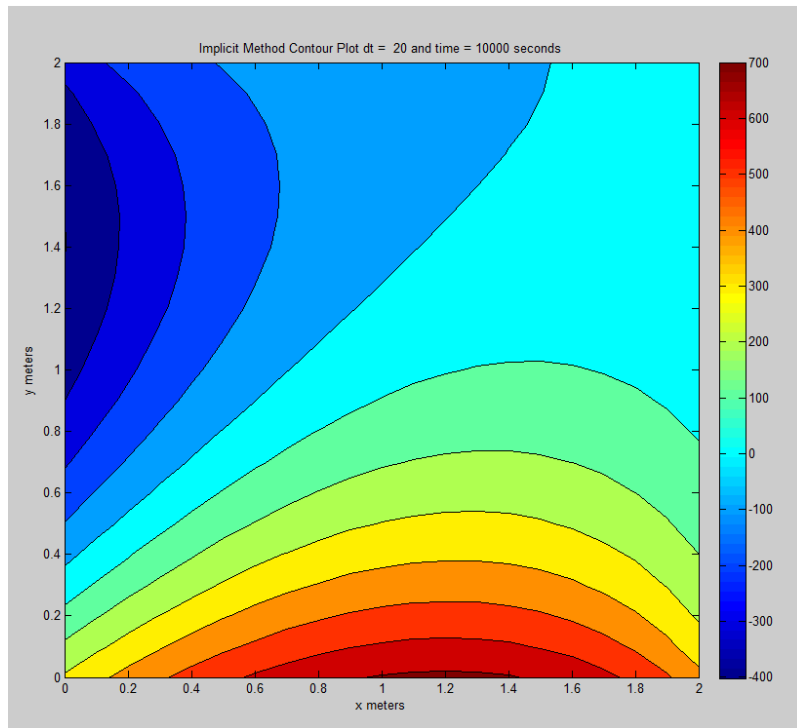
```
K          = inv(C);  
T(:,i+1) = K*T(:,i) + K*b;  
i         = i+1;
```

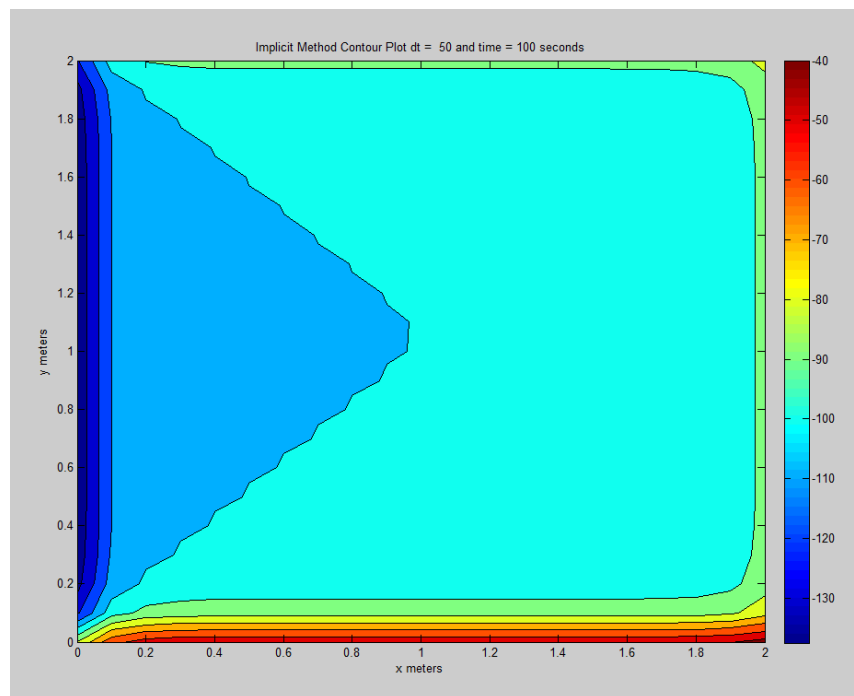
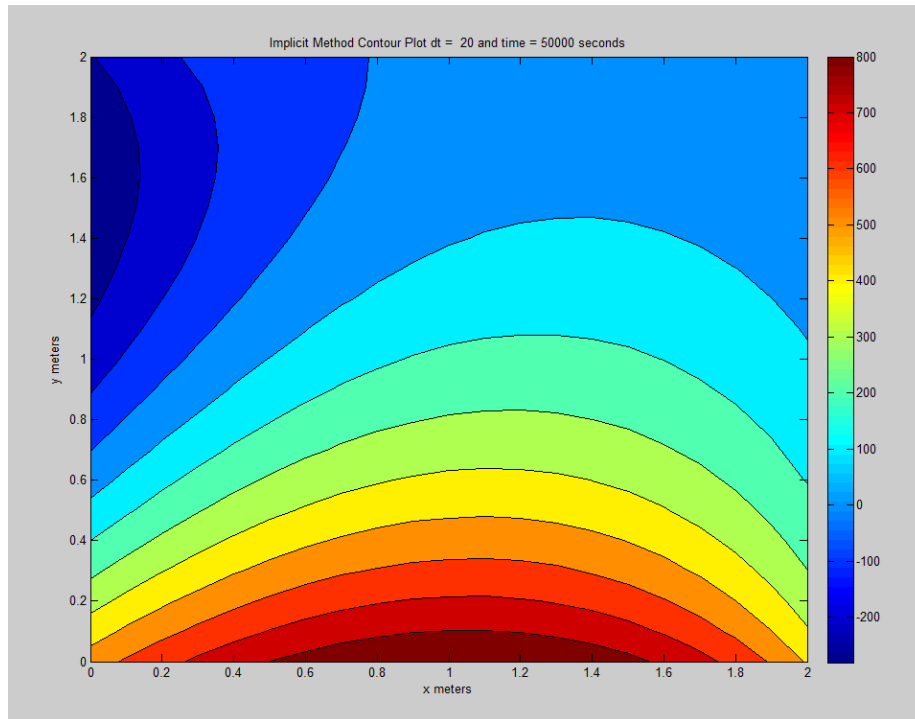
Results

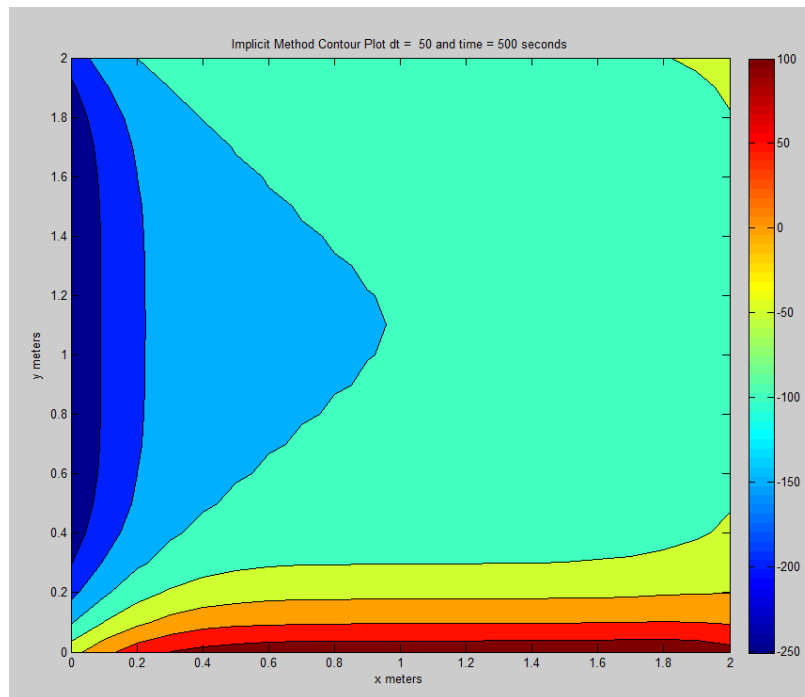
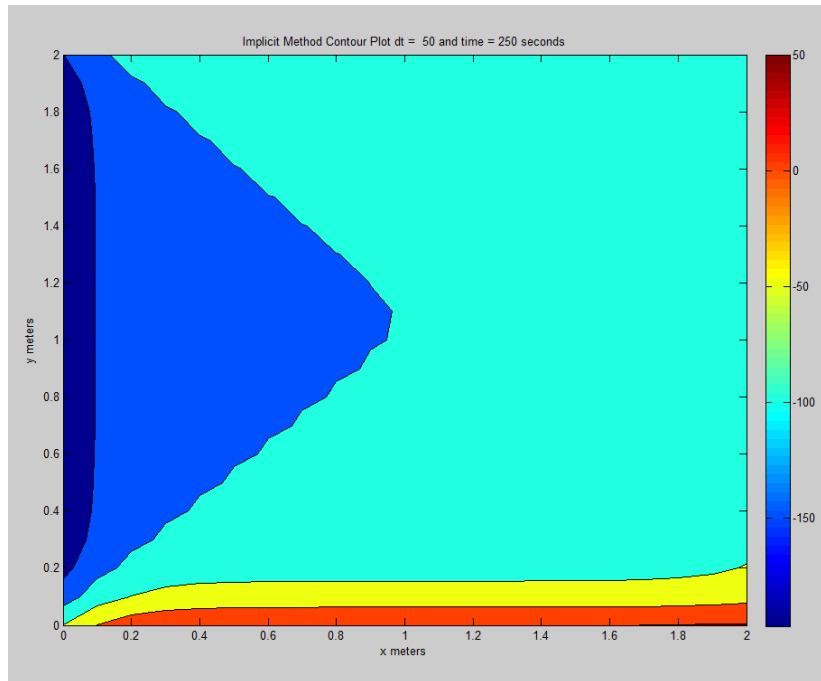


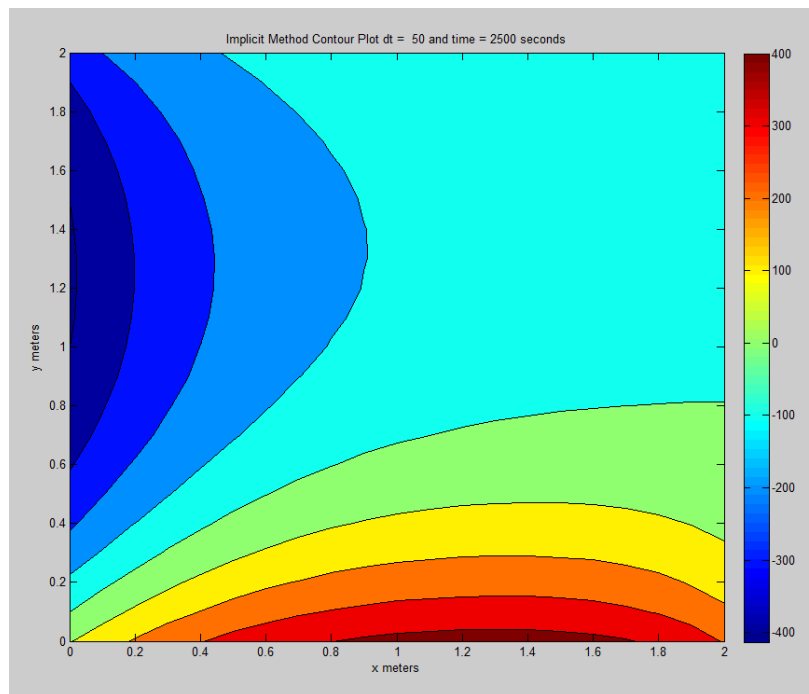
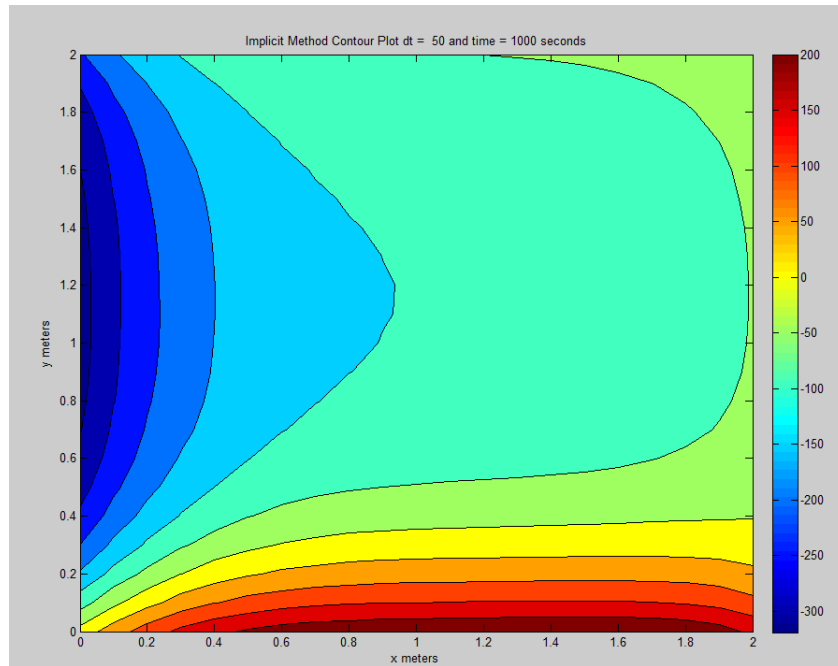


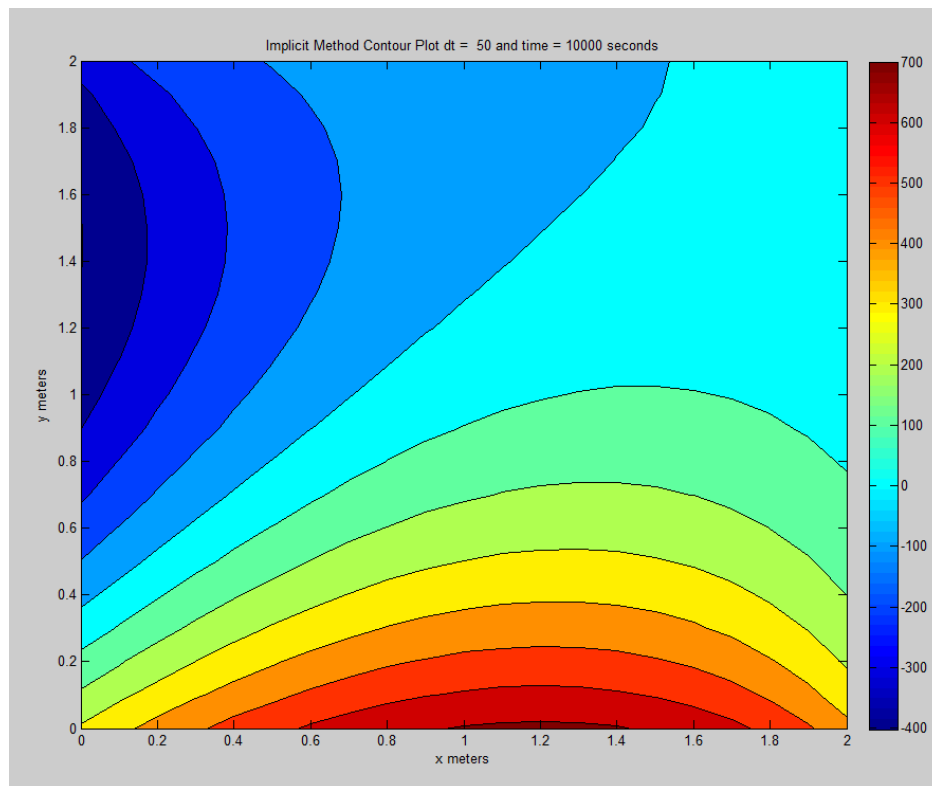
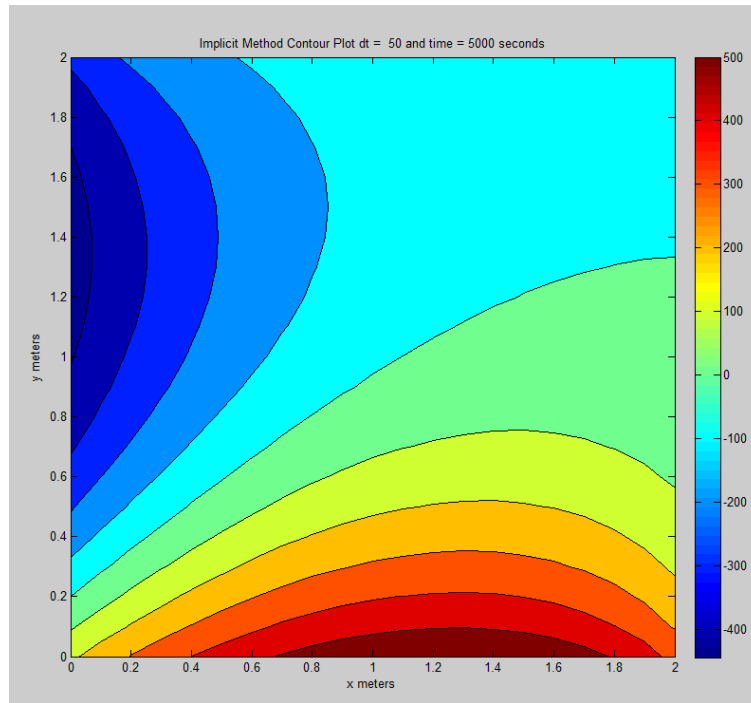


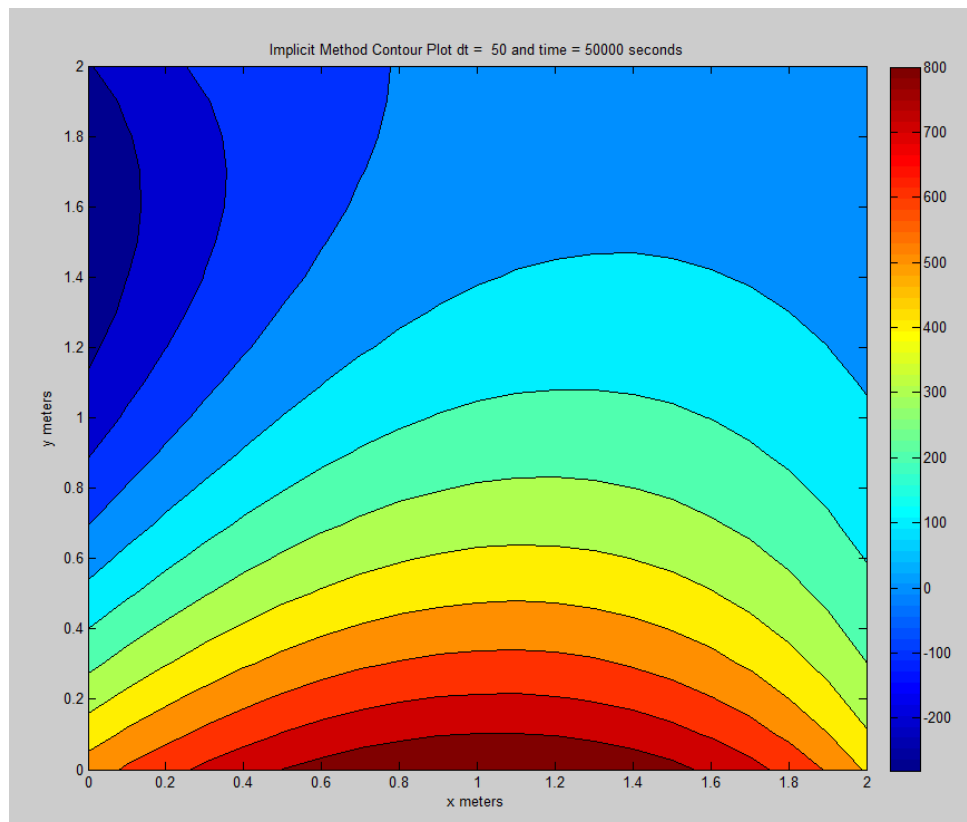
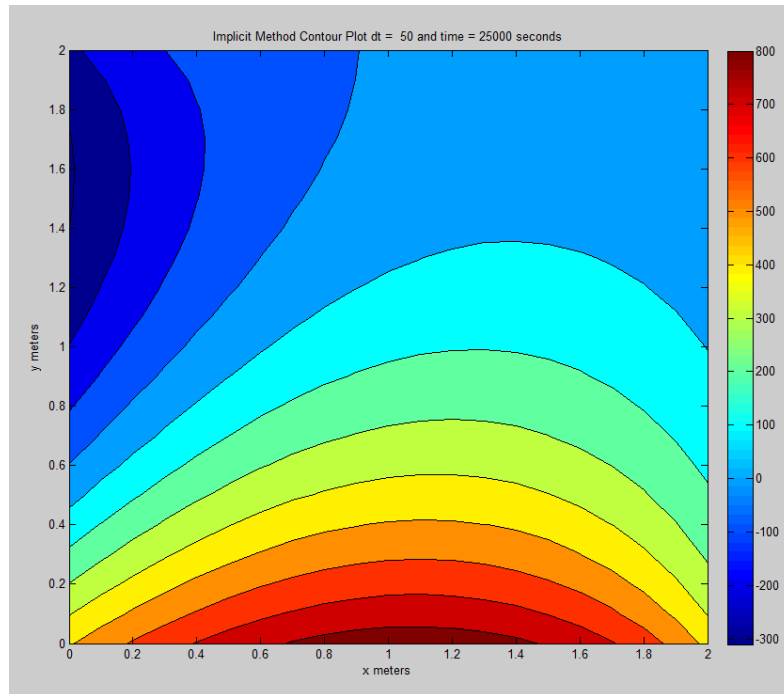


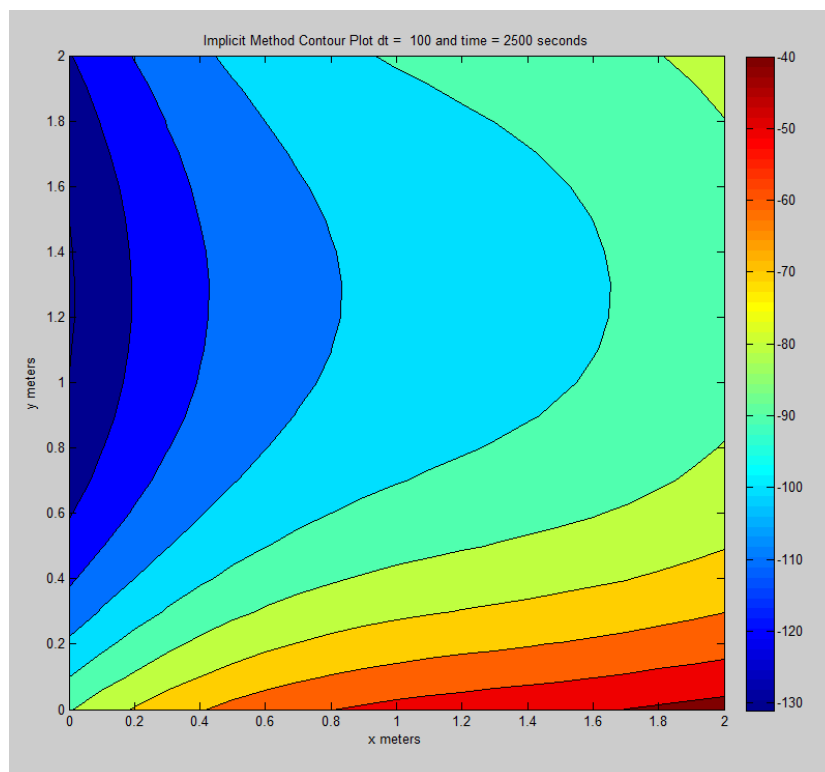
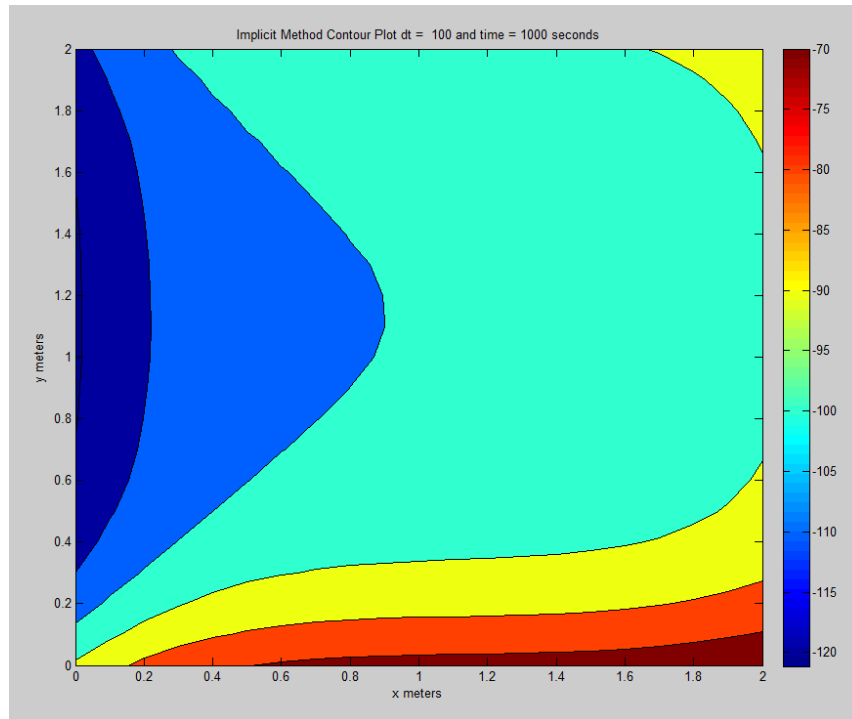


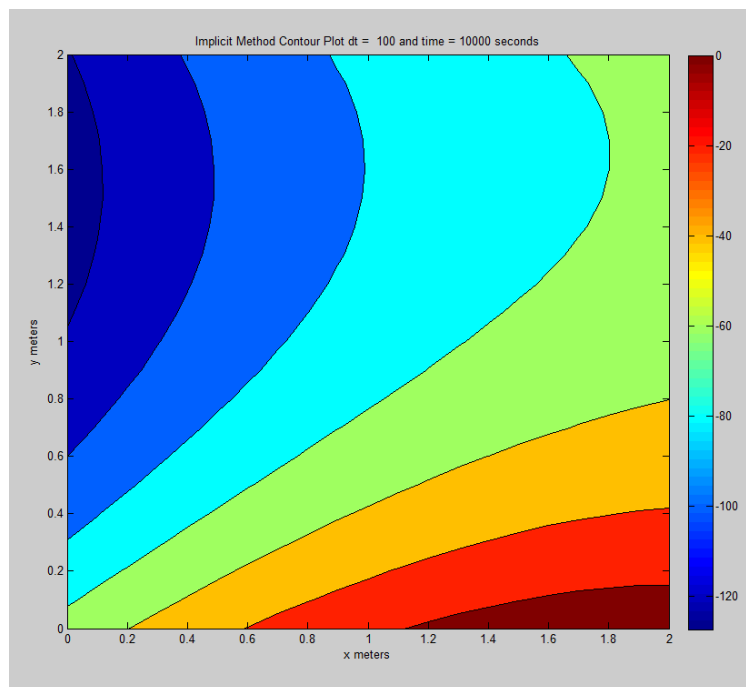
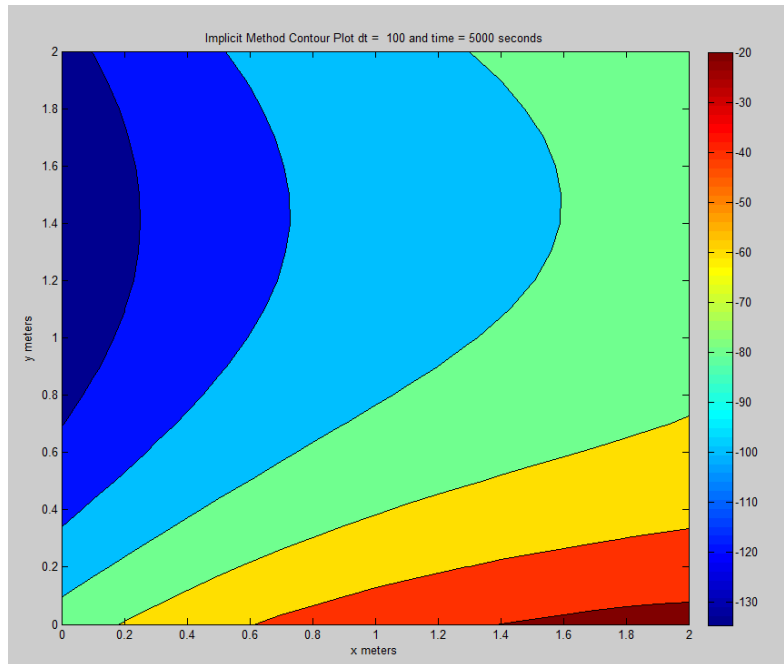


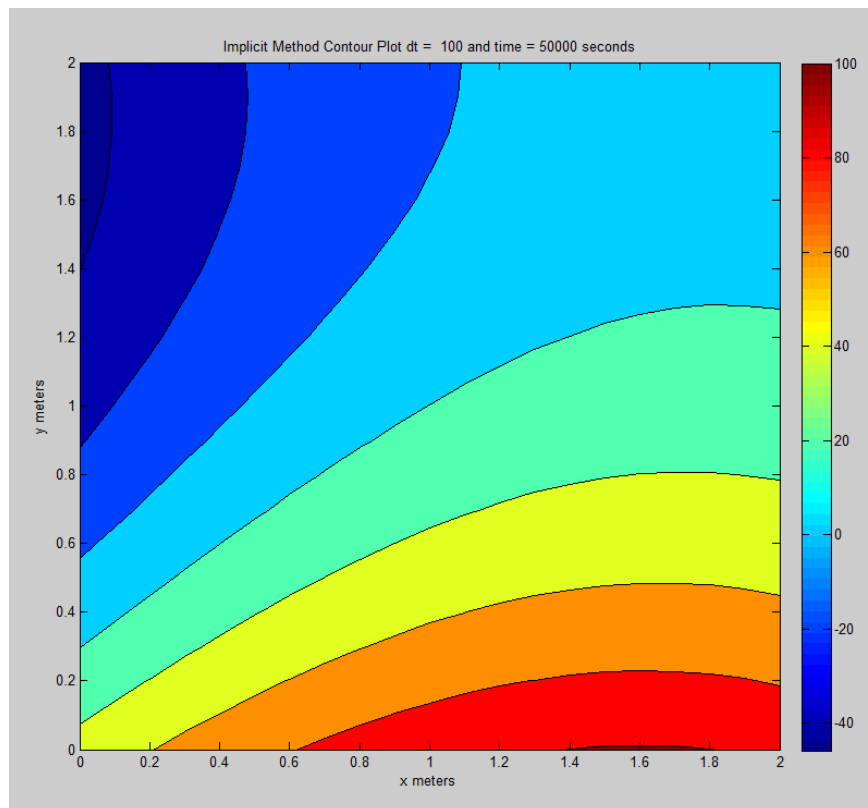
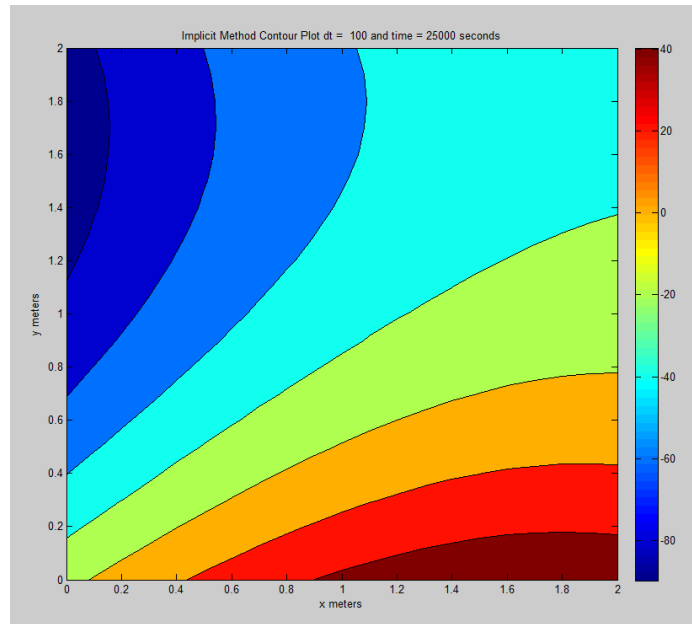












Discussion

As expected, the overall thermal behaviour was similar to that of the other problems. The heat source on the bottom of the plate would have its heat distributed along the plate in a fashion that makes it look like the heat sink on the left is “sucking in” the heat more than the surrounding temperature is. The time step variance in this module seems to have little to no destructive effect on the stability of the mesh. One of the lessons learned in Dr. Simon’s

numerical methods class is that the implicit method is very good for staying stable even with large time steps.

Appendix

Derivations for Part 1

Finite Difference Equations assuming $\Delta x = \Delta y$

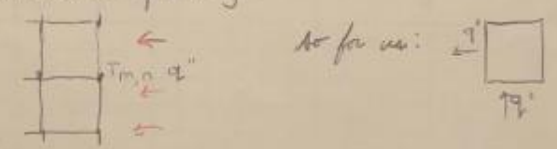
Center: (a) $T_{m,n+1} + T_{m,n-1} + T_{m+1,n} + T_{m-1,n} - 4T_{m,n} = 0$
 Tol 4.2

Surface with convection: (b) $T_{m,n+1} + T_{m,n-1} + 2T_{m+1,n} + 2T_{m-1,n} - 2\left(\frac{h\Delta x}{k} + 2\right)T_{m,n} + \frac{2h\Delta x}{k}T_\infty = 0$
 Tol 4.2

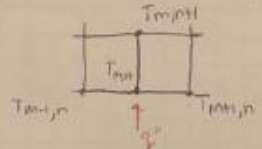
Node @ corner: (c) $T_{m,n+1} + T_{m,n-1} + T_{m+1,n} + T_{m-1,n} + (-2\left(\frac{h\Delta x}{k} + 1\right)T_{m,n} + \frac{2h\Delta x}{k}T_\infty = 0$
 Tol 4.2

Node @ plane: $T_{m,n+1} + T_{m,n-1} + 2T_{m+1,n} + 2T_{m-1,n} + (-4)T_{m,n} + \frac{2q''\Delta x}{k} = 0$
 surface with uniform heat flux
 Tol 4.2

Assume the following:



Modified:



So far we:

$$q_{m,n+1} + q_{m,n-1} + q_{m+1,n} + q_{m-1,n} + q = 0$$

$$k(T_{m,n+1} - T_{m,n}) + k(T_{m,n-1} - T_{m,n}) + \frac{k}{2}(T_{m+1,n} - T_{m,n}) + \frac{k}{2}(T_{m-1,n} - T_{m,n}) + q''(\Delta x) = 0$$

$$(T_{m,n+1} - T_{m,n}) + (T_{m,n-1} - T_{m,n}) + \frac{1}{2}(T_{m+1,n} - T_{m,n}) + \frac{1}{2}(T_{m-1,n} - T_{m,n}) + \frac{q''(\Delta x)}{k} = 0$$

$$\frac{2T_{m,n+1} - 2T_{m,n} + 2T_{m,n-1} - 2T_{m,n} + T_{m+1,n} - T_{m,n} + T_{m-1,n} - T_{m,n}}{2} + \frac{q''(\Delta x)}{k} = 0$$

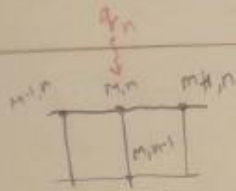
$$4T_{m,n+1} - 4T_{m,n} + 2T_{m+1,n} - 4T_{m,n} + T_{m-1,n} - T_{m,n} + \frac{2q''\Delta x}{k} = 0$$

$$q_{m,n+1} = \frac{k\Delta x}{2\Delta y}(T_{m,n+1} - T_{m,n})$$

$$q_{m+1,n} = \frac{k\Delta y}{2\Delta x}(T_{m+1,n} - T_{m,n})$$

$$q_{m-1,n} = \frac{k\Delta y}{2\Delta x}(T_{m-1,n} - T_{m,n})$$

$$q = q''(\Delta x)$$



$$\phi T_{m,n+1} + 2T_{m,n-1} + T_{m+1,n} + T_{m-1,n} + (-2)\left(\frac{h\Delta x}{k} + 2\right)T_{m,n} + \frac{2h\Delta x}{k}T_\infty = 0$$

Correct

1 2

$\Delta y = \Delta x$

1

2

$$\left(\frac{-h\Delta x}{k} - 2\right)$$

1) ~~scribbled out~~

$$\frac{2}{k}\left(\frac{-h\Delta x}{2} - k\right)$$

$$\left(\frac{h\Delta x}{k}T_\infty + \frac{q_n\Delta y}{k}\right)$$

$$\phi T_{m,n+1} + \frac{k}{2}T_{m,n-1} + \frac{k}{2}T_{m+1,n} + \phi T_{m-1,n} + \left(\frac{-h\Delta x}{2} - \frac{k}{2} - \frac{k}{2}\right)T_{m,n} + \left(\frac{h\Delta x}{2}T_\infty + \frac{q_n\Delta y}{2}\right) = 0$$

$$2) (0) \quad (1) \quad (1) \quad (0) \quad \left(\frac{-h\Delta x}{k} - 2\right) \quad \left(\frac{h\Delta x}{k}T_\infty + \frac{q_n\Delta y}{k}\right) = 0$$

$$\phi T_{m,n+1} + \frac{k}{2}T_{m,n-1} + \phi T_{m+1,n} + \frac{k}{2}T_{m-1,n} + \left(\frac{-2h\Delta x}{k} - 1\right)T_{m,n} + (h\Delta x)T_\infty = 0$$

$$3) \frac{k}{2}T_{m,n+1} + \phi T_{m,n-1} + \frac{k}{2}T_{m+1,n} + \phi T_{m-1,n} + (-k)T_{m,n} + \left(q_n\frac{\Delta x}{2} + q_n'\frac{\Delta y}{2}\right) = 0$$

$$4) \frac{k}{2}T_{m,n+1} + \phi T_{m,n-1} + \phi T_{m+1,n} + \frac{k}{2}T_{m-1,n} + \left(-k - \frac{h\Delta x}{2}\right)T_{m,n} + \left(\frac{h\Delta x}{2}T_\infty + \frac{q_n'\Delta y}{2}\right) = 0$$



$T_{m,n+1}$

$T_{m,n-1}$

$T_{m+1,n}$

$T_{m-1,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

$T_{m,n}$

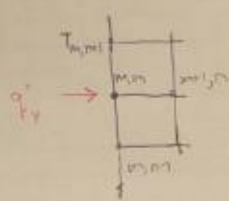
$T_{m,n}$

$$k(T_{m,n+1} - T_{m,n}) + \frac{k}{2}(T_{m+1,n} - T_{m,n}) + \frac{k}{2}(T_{m-1,n} - T_{m,n}) + q''(\Delta x) = 0$$

$$(T_{m,n+1} - T_{m,n}) + \frac{1}{2}(T_{m+1,n} - T_{m,n}) + \frac{1}{2}(T_{m-1,n} - T_{m,n}) + \frac{q''(\Delta x)}{k} = 0$$

$$2T_{m,n+1} - 2T_{m,n} + T_{m+1,n} - T_{m,n} + T_{m-1,n} - T_{m,n} + \frac{2q''(\Delta x)}{k} = 0$$

$$d \quad 2T_{m,n+1} + 0T_{m,n-1} + T_{m+1,n} + T_{m-1,n} + (-4)T_{m,n} + \frac{2q''(\Delta x)}{k} = 0$$



$$q_{m,n+1} + q_{m,n-1} + q_{m+1,n} + q_{m,n} + q_{m,n} + q_{m,n} = 0$$

$$\frac{k\Delta x}{2\Delta y}(T_{m,n+1} - T_{m,n}) + \frac{k\Delta x}{2\Delta y}(T_{m,n-1} - T_{m,n}) + \frac{k\Delta y}{\Delta x}(T_{m+1,n} - T_{m,n}) + q''_y(\Delta y) = 0$$

$$\frac{k}{2}(T_{m,n+1} - T_{m,n}) + \frac{k}{2}(T_{m,n-1} - T_{m,n}) + k(T_{m+1,n} - T_{m,n}) + q''_y(\Delta y) = 0$$

$$3(T_{m,n+1} - T_{m,n}) + (T_{m,n-1} - T_{m,n}) + 2(T_{m+1,n} - T_{m,n}) + \frac{2q''_y(\Delta y)}{k} = 0$$

$$e \quad T_{m,n+1} + T_{m,n-1} + 2T_{m+1,n} + 0T_{m-1,n} + (-4)T_{m,n} + \frac{2q''_y(\Delta y)}{k\Delta x} = 0$$

Matlab file, part 1

```
% Michael Tanja and Jonathan DiBacco

% The first segment of the code is where all the parameters are
stated.
% The only thing you will have to change is the NperRow and the
NperCol
% The 'dx' changes according to the inputs for NperRow/Col

clear all
close all

%%
%parameters and
%mesh properties
prompt1 = 'What resolution of dx would you want? ';
prompt2 = 'What is the length and width of this square plate? ';
L = input(prompt2);          %unsure about what this does, but it was
in the example
W = L;          %unsure about what this does, but it was in the
example
dx = input(prompt1);
qx = 100000;      %heat flux bottom
qy = -65000;      %heat flux side
NperRow = (L/dx) + 1; % Nodes per row
NperCol = NperRow; % Nodes per column
nNodes = NperRow * NperCol;
A = zeros(NperRow, NperCol);
b = zeros(NperRow,1);

k = 250;          %conduction coefficient
h = 250;          %convection coefficient
Tinf = 0;         %fluid temp in celsius

%next in line is node set up
%%
% Center Node/s

for ii = NperRow+2 : ((2*NperRow-1))
    for jj = ii:NperRow:nNodes-(NperRow+1)%NperRow+2 : (NperRow^2) -
(NperRow+1) %put this first in order, matlab reads top down

        A(jj, jj+NperRow) = 1; %upper
        A(jj, jj-NperRow) = 1; %lower
        A(jj, jj+1) = 1;      %right
        A(jj, jj-1) = 1;      %left
        A(jj, jj) = -4;       %center
        b(jj) = 0;           %boundary
```

```

end
end

%%
% Convection node/s on the right

for ii = 2*NperRow : NperRow : (NperRow^2) - NperRow
%needs to be more generalized
    jj=ii
    A(ii, jj+NperRow) = 1;           %upper
    A(ii, jj-NperRow) = 1;           %lower
                                     %no right
    A(ii, jj-1) = 2;                 %left
    A(ii, jj) = (-4-((2*h*dx)/k)); %center
    b(ii) = -((2*h*dx)/k)*Tinf;      %boundary
end

%%
% Convection node/s on top
for ii = (nNodes)-(NperRow-2) : (nNodes-1) %generalized
    jj= ii
                                     %no top term
    A(ii, jj-NperRow) = 2;           %lower
    A(ii, jj+1) = 1;                 %right
    A(ii, jj-1) = 1;                 %left
    A(ii, jj) = (-4-((2*h*dx)/k)); %center
    b(ii) = -(2*h*dx*Tinf)/k;        %boundary
end

%starting corners 1-4
%%
% Corner 1 (this boundary condition should have same initiation
% for any matrix size)
for ii = (NperCol*NperRow)-(NperRow-1) %generalized
    jj=ii
                                     %no top term
    A(ii, jj-NperRow) = 1;           %lower
    A(ii, jj+1) = 1;                 %right
                                     %no left term
    A(ii, jj) = (-2-((2*h*dx)/k)); %center
    b(ii) = -(((2*h*dx*Tinf)/k) + ((2*qy*dx)/k)); %boundary
                                     %EDIT: change flow of heat on boundary
condition
end

%%
% Corner 2 (This shouldn't change either)

for ii = nNodes
    jj=ii
                                     %no top term

```

```

    A(ii, jj-NperRow) = 1;           %lower term
                                     %no right term
    A(ii, jj-1) = 1;                 %left
    A(ii, jj) = (-(2*h*dx)/(k)) - 2; %center
    b(ii) = -(2*h*dx*Tinf)/k;        %boundary
end

%%
% Corner 3 (should stay the same for all meshes)

for ii = 1
    jj=ii
    A(ii, jj+NperRow) = 1; %upper
                           %no lower
    A(ii, jj+1) = 1;       %right
                           %no left term
    A(ii, jj) = -2;        %center
    b(ii) = -((dx/k)*(qy+qx)); %boundary
                           %EDIT: Changed flow of heat on the boundary condition
end

%%
% Corner 4 (should be the same for all matrices)

for ii = NperRow
    jj=ii
    A(ii, jj+NperRow) = 1; %upper
                           %no lower term
                           %no right term
    A(ii, jj-1) = 1;       %left term
    A(ii, jj) = -2-((h*dx)/k); %center
    b(ii) = -(((qx*dx) + (h*dx*Tinf))/k); %boundary
end

%corners are finished
%Start heat sources
%%
%heat source left side

for ii = NperRow+1 : NperRow : (nNodes)-(NperRow+1)
    %try to generalize for all matrices
    jj=ii
    A(ii, jj+NperRow) = 1; %upper
    A(ii, jj-NperRow) = 1; %lower
    A(ii, jj+1) = 2;       %right
                           %no left term
    A(ii, jj) = -4;        %center
    b(ii) = -(2*(qy)*dx)/k; %boundary
                           %EDIT: Changed flow of heat on the boundary condition,
for qy
end

```

```

%%
%heat source bottom

for ii = 2: (NperRow-1)    %generalized
    jj=ii
    A(ii, jj+NperRow) = 2; %upper
                        %no lower term
    A(ii, jj+1) = 1;      %right term
    A(ii, jj-1) = 1;      %left term
    A(ii, jj) = -4;        %center
    b(ii) = -(2*qx*dx)/k;  %boundary
end

%%
%Matrices are all set up, should be able to solve and graph now
v = b'; %turned array into vector to make dimensions agree
T = A\b;

%%
%creating the mesh plot

x = [0: dx :L]; %x-axis grid
y = [0: dx :W]; %y-axis grid

[X,Y] = meshgrid(x,y);
%Create a loop that converts the vector back into a mesh to plot
%takes the temperature vector and reorganizes into a matrix

nIndex = 1; %this index will count down the temperature
for aa = 1:NperRow %nNodes/NperRow
    for bb = 1:NperRow
        TMesh(aa,bb) = T(nIndex);
        nIndex = nIndex +1;
    end
end
%TMesh1 = abs(TMesh);
%recycled some of the example code
contourf(X,Y,TMesh)

colorbar

%contour looks ok,but heat might be flowing in the wring direction
%change the sign for heat flow on the right face of the figure
%^this should affect values for T1, T4, T7

```


Derivations for Part 2

Corner 1

$$\begin{aligned}
 & \frac{h\Delta x}{2}(T_w - T_{m,n}) + \frac{h\Delta x}{2}(T_{m,n-1} - T_{m,n}) + \frac{h\Delta x}{2}(T_{m+1,n-1} - T_{m,n}) + \frac{q_y''\Delta x}{2} = \frac{\rho\Delta x^2 c}{4\Delta t}(T_{m,n} - T_{m,n}^o) \\
 & \frac{h\Delta x}{k}(T_w - T_{m,n}) + (T_{m,n-1} - T_{m,n}) + (T_{m+1,n-1} - T_{m,n}) + \frac{q_y''\Delta x}{k} = \frac{\rho\Delta x^2 c}{4k\Delta t}(T_{m,n} - T_{m,n}^o) \quad \frac{1}{2F_o} \\
 & \frac{2F_o h\Delta x}{k}(T_w - T_{m,n}) + 2F_o(T_{m,n-1} - T_{m,n}) + 2F_o(T_{m+1,n-1} - T_{m,n}) + 2F_o \frac{q_y''\Delta x}{k} + T_{m,n} = T_{m,n}^o \\
 & 2T_{m,n+1} + 2F_o T_{m,n-1} + 2F_o T_{m+1,n-1} + 2T_{m,n} + (-4F_o - \frac{2F_o h\Delta x}{k} + 1) + \frac{2F_o q_y''\Delta x}{k} = T_{m,n}^o
 \end{aligned}$$

$$\begin{aligned}
 \Delta x^2 &= \Delta x^2 \\
 \Delta t &= (\frac{\Delta x}{2})^2
 \end{aligned}$$

Corner 2

$$\begin{aligned}
 & \frac{h\Delta x}{2}(T_w - T_{m,n}) + \frac{h\Delta x}{2}(T_{m,n-1} - T_{m,n}) + \frac{h\Delta x}{2}(T_w - T_{m,n}) + \frac{h\Delta x}{2}(T_{m+1,n-1} - T_{m,n}) = \frac{\rho\Delta x^2 c}{4\Delta t}(T_{m,n} - T_{m,n}^o) \\
 & \frac{h\Delta x}{k}(T_w - T_{m,n}) + (T_{m,n-1} - T_{m,n}) + \frac{h\Delta x}{k}(T_w - T_{m,n}) + (T_{m+1,n-1} - T_{m,n}) = \frac{\rho\Delta x^2 c}{4k\Delta t}(T_{m,n} - T_{m,n}^o) \quad \frac{1}{2F_o} \\
 & \frac{2F_o h\Delta x}{k}(T_w - T_{m,n}) + 2F_o(T_{m,n-1} - T_{m,n}) + \frac{2F_o h\Delta x}{k}(T_w - T_{m,n}) + 2F_o(T_{m+1,n-1} - T_{m,n}) + T_{m,n} = T_{m,n}^o \\
 & 2T_{m,n+1} + 2F_o T_{m,n-1} + 2T_{m+1,n-1} + 2F_o T_{m,n} + (-4F_o - \frac{4F_o h\Delta x}{k} - 4F_o + 1) + \frac{4F_o h\Delta x}{k} T_w = T_{m,n}^o
 \end{aligned}$$

Corner 3

$$\begin{aligned}
 & \frac{h\Delta x}{2}(T_{m,n-1} - T_{m,n}) + \frac{q_x''\Delta x}{2} + \frac{h\Delta x}{2}(T_{m+1,n-1} - T_{m,n}) + \frac{q_y''\Delta x}{2} = \frac{\rho\Delta x^2 c}{4\Delta t}(T_{m,n} - T_{m,n}^o) \\
 & (T_{m,n+1} - T_{m,n}) + \frac{q_x''\Delta x}{k} + (T_{m+1,n-1} - T_{m,n}) + \frac{q_y''\Delta x}{k} = \frac{\rho\Delta x^2 c}{2k\Delta t}(T_{m,n} - T_{m,n}^o) \quad \frac{1}{2F_o} \\
 & 2F_o(T_{m,n+1} - T_{m,n}) + \frac{2F_o q_x''\Delta x}{k} + 2F_o(T_{m+1,n-1} - T_{m,n}) + \frac{2F_o q_y''\Delta x}{k} + T_{m,n} = T_{m,n}^o \\
 & 2F_o T_{m,n+1} + 2F_o T_{m,n-1} + 2F_o T_{m+1,n-1} + 2T_{m,n} + (-4F_o - \frac{4F_o q_x''\Delta x}{k} + 1) + \frac{2F_o q_y''\Delta x}{k} = T_{m,n}^o \\
 & \frac{h\Delta x}{2}(T_{m,n+1} - T_{m,n}) + \frac{q_x''\Delta x}{2} + \frac{h\Delta x}{2}(T_w - T_{m,n}) + \frac{h\Delta x}{2}(T_{m+1,n-1} - T_{m,n}) = \frac{\rho\Delta x^2 c}{4\Delta t}(T_{m,n} - T_{m,n}^o) \\
 & (T_{m,n+1} - T_{m,n}) + \frac{q_x''\Delta x}{k} + \frac{h\Delta x}{k}(T_w - T_{m,n}) + (T_{m+1,n-1} - T_{m,n}) = \frac{\rho\Delta x^2 c}{2k\Delta t}(T_{m,n} - T_{m,n}^o) \quad \frac{1}{2F_o} \\
 & 2F_o(T_{m,n+1} - T_{m,n}) + \frac{2F_o q_x''\Delta x}{k} + \frac{2F_o h\Delta x}{k}(T_w - T_{m,n}) + 2F_o(T_{m+1,n-1} - T_{m,n}) + T_{m,n} = T_{m,n}^o \\
 & 2F_o T_{m,n+1} + 2F_o T_{m,n-1} + 2F_o T_{m+1,n-1} + 2F_o T_{m,n} + (-4F_o - \frac{2F_o h\Delta x}{k} + 1) + \frac{2F_o q_x''\Delta x}{k} + \frac{2F_o h\Delta x}{k} T_w = T_{m,n}^o
 \end{aligned}$$

$$-T + (T_{m,n+1} - T_{m,n}) + \frac{k\Delta x}{\Delta y} (T_{m,n+1} - T_{m,n}) + \frac{k\Delta y}{\Delta x} (T_{m+1,n} - T_{m,n}) + \frac{k\Delta y}{\Delta x} (T_{m-1,n} - T_{m,n}) = \frac{\rho V C}{\Delta t} (T_{m,n} - T_{m,n}^o) \quad \Delta = \Delta x^2$$

$$(T_{m,n+1} - T_{m,n}) + (T_{m,n+1} - T_{m,n}) + (T_{m+1,n} - T_{m,n}) + (T_{m-1,n} - T_{m,n}) = \frac{2\rho V \Delta x^2 C}{k \Delta t} (T_{m,n} - T_{m,n}^o)$$

$$\frac{F_o}{2} (T_{m,n+1} - T_{m,n}) + \frac{F_o}{2} (T_{m,n+1} - T_{m,n}) + \frac{F_o}{2} (T_{m+1,n} - T_{m,n}) + \frac{F_o}{2} (T_{m-1,n} - T_{m,n}) + T_{m,n}^o = T_{m,n}$$

$$\frac{F_o}{2} T_{m,n+1} + \frac{F_o}{2} T_{m,n+1} + \frac{F_o}{2} T_{m+1,n} + \frac{F_o}{2} T_{m-1,n} + (-4F_o + 1) T_{m,n}^o = T_{m,n}$$

$$\text{right} \quad \frac{h\Delta x}{\Delta y} (T_{m,n+1} - T_{m,n}^o) + \frac{k\Delta y}{\Delta x} (T_{m,n+1} - T_{m,n}^o) + \frac{h\Delta y}{\Delta x} (T_{m,n}^o - T_{m,n}^o) + \frac{k\Delta y}{\Delta x} (T_{m+1,n} - T_{m,n}^o) = \frac{\rho V \Delta x^2 C}{\Delta t} (T_{m,n} - T_{m,n}^o)$$

$$(T_{m,n+1} - T_{m,n}^o) + (T_{m,n+1} - T_{m,n}^o) + 2(T_{m+1,n} - T_{m,n}^o) = \frac{2\rho V \Delta x^2 C}{k \Delta t} (T_{m,n} - T_{m,n}^o) \quad \left[\text{is it } \frac{\Delta x^2}{2} \right]$$

$$\frac{F_o}{2} (T_{m,n+1} - T_{m,n}^o) + \frac{F_o}{2} (T_{m,n+1} - T_{m,n}^o) + \frac{F_o h \Delta x}{k} (T_{m,n}^o - T_{m,n}^o) + F_o (T_{m+1,n} - T_{m,n}^o) + T_{m,n}^o = T_{m,n}$$

$$\frac{F_o}{2} T_{m,n+1} + \frac{F_o}{2} T_{m,n+1} + 0 T_{m,n} + 2F_o T_{m+1,n} + (-4F_o - \frac{2F_o h \Delta x}{k} + 1) T_{m,n}^o + \frac{2F_o h \Delta x}{k} T_{m,n}^o = T_{m,n}$$

$$\text{step} \quad h\Delta x (T_{m,n}^o - T_{m,n}^o) + \frac{k\Delta x}{\Delta y} (T_{m,n+1} - T_{m,n}^o) + \frac{k\Delta y}{\Delta x} (T_{m+1,n} - T_{m,n}^o) + \frac{k\Delta y}{\Delta x} (T_{m-1,n} - T_{m,n}^o) = \frac{\rho V \Delta x^2 C}{\Delta t} (T_{m,n} - T_{m,n}^o) \quad \downarrow \text{assumed} \quad \frac{\Delta x^2}{2} \text{ instead}$$

$$\frac{2h\Delta x}{k} (T_{m,n}^o - T_{m,n}^o) + 2(T_{m,n+1} - T_{m,n}^o) + (T_{m+1,n} - T_{m,n}^o) + (T_{m-1,n} - T_{m,n}^o) = \frac{1}{F_o} (T_{m,n} - T_{m,n}^o)$$

$$\frac{F_o 2h\Delta x}{k} (T_{m,n}^o - T_{m,n}^o) + 2F_o (T_{m,n+1} - T_{m,n}^o) + F_o (T_{m+1,n} - T_{m,n}^o) + F_o (T_{m-1,n} - T_{m,n}^o) + T_{m,n}^o = T_{m,n}$$

$$0 T_{m,n+1} + 2F_o T_{m,n+1} + F_o T_{m+1,n} + F_o T_{m-1,n} + (-\frac{F_o 2h\Delta x}{k} - 4F_o + 1) T_{m,n}^o + \frac{F_o 2h\Delta x}{k} T_{m,n}^o = T_{m,n}$$

$$\text{left} \quad \frac{k\Delta x}{\Delta y} (T_{m,n+1} - T_{m,n}^o) + \frac{k\Delta x}{\Delta y} (T_{m,n+1} - T_{m,n}^o) + \frac{k\Delta y}{\Delta x} (T_{m+1,n} - T_{m,n}^o) + q''_y \Delta y = \frac{\rho V \Delta x^2 C}{\Delta t} (T_{m,n} - T_{m,n}^o)$$

$$(T_{m,n+1} - T_{m,n}^o) + (T_{m,n+1} - T_{m,n}^o) + 2(T_{m+1,n} - T_{m,n}^o) + \frac{2q''_y \Delta x}{k} = \frac{\rho V \Delta x^2 C}{k \Delta t} (T_{m,n} - T_{m,n}^o)$$

$$F_o (T_{m,n+1} - T_{m,n}^o) + F_o (T_{m,n+1} - T_{m,n}^o) + 2F_o (T_{m+1,n} - T_{m,n}^o) + \frac{2F_o q''_y \Delta x}{k} + T_{m,n}^o = T_{m,n}$$

$$F_o T_{m,n+1} + F_o T_{m,n+1} + 2F_o T_{m+1,n} + 0 T_{m-1,n} + (-4F_o + 1) T_{m,n}^o + \frac{2F_o q''_y \Delta x}{k} = T_{m,n}$$

$$\begin{aligned}
 & \left(T_{m,n+1} - T_{m,n} \right) + q'' \Delta x + \frac{k \Delta y}{\Delta x^2} (T_{m+1,n} - T_{m,n}) + \frac{h}{\Delta x^2} (T_{m,n} - T_{m,n-1}) = 2 \alpha \dot{T}_{m,n} \\
 & 2(T_{m,n+1} - T_{m,n}) + \frac{2q'' \Delta x}{K} + (T_{m+1,n} - T_{m,n}) + (T_{m-1,n} - T_{m,n}) = \frac{1}{F_0} (T_{m,n} - T_{m,n}) \\
 & 2F_0(T_{m,n+1} - T_{m,n}) + \frac{2F_0 q'' \Delta x}{K} + F_0(T_{m+1,n} - T_{m,n}) + F_0(T_{m-1,n} - T_{m,n}) + T_{m,n} = T_{m,n} \\
 & 2F_0 T_{m,n+1} + T_{m,n-1} + F_0 T_{m+1,n} + F_0 T_{m-1,n} + (-4F_0 + 1) T_{m,n} + \frac{2F_0 q'' \Delta x}{K} = T_{m,n}
 \end{aligned}$$

Matlab File Part 2

```
% Michael Tanja and Jonathan DiBacco

% The first segment of the code is where all the parameters are
stated.
% The only thing you will have to change is the NperRow and the
NperCol
% The 'dx' changes according to the inputs for NperRow/Col

clear all
close all

%%
%parameters and
%mesh properties
prompt1 = 'What resolution of dx would do you want?          : ';
prompt2 = 'What is the length and width of this square plate?: ';
L      = input(prompt2);          % unsure about what this does, but it
was in the example
W      = L;                      % unsure about what this does, but it
was in the example
dx     = input(prompt1);
qx     = 100000;                 % heat flux bottom
qy     = -65000;                 % heat flux side
NperRow = (L/dx) + 1;            % Nodes per row
NperCol = NperRow;               % Nodes per column
nNodes  = NperRow * NperCol;
C       = zeros(NperRow, NperCol);
b       = zeros(NperRow,1);

k       = 250;                   % conduction coefficient
h       = 250;                   % convection coefficient
Tinf    = 0;                     % fluid temp in celsius

dt      = input('what is the dt?                               : ');
        % delta t
endtime = input('how long should the simulation run?          : ');
        % setting end time
Tcol    = endtime/dt;            % number of temperature colloms
T       = zeros(nNodes,Tcol);    % initializing Temperature %Tcol
T(:,1)  = 400*ones(1, nNodes);  % setting initial condition to
Temperature = 400 at Time = 0

% got rid of ro and cp
```

```

a      = 1e-4;                                % thermal diffusivity
Fo      = (a*dt)/(dx*dx);                      %Fourier
i = 1;
%%
% setting up the time loop
for t = 1:dt:endtime

% next in line is node set up
%%
% Center Node/s

for ii = NperRow+2 : ((2*NperRow-1))
    for jj = ii:NperRow:nNodes-(NperRow+1)%NperRow+2 : (NperRow^2) -
(NperRow+1) %put this first in order, matlab reads top down

        C(jj, jj+NperRow) = Fo;                % upper
        C(jj, jj-NperRow) = Fo;                % lower
        C(jj, jj+1)      = Fo;                % right
        C(jj, jj-1)      = Fo;                % left
        C(jj, jj)        = (1-(4*Fo)); % center
        b(jj)            = 0;                % boundary
    end
end

%%
% Convection node/s on the right

for ii = 2*NperRow : NperRow : (NperRow^2) - NperRow
    %needs to be more generalized

    jj = ii;

    C(ii, jj+NperRow) = Fo;                    % upper
    C(ii, jj-NperRow) = Fo;                    % lower
                                                % no right
    C(ii, jj-1)      = 2*Fo;                    % left
    C(ii, jj)        = (((-2*Fo*h*dx)/k)+(-4*Fo)+1); % center
    b(ii)            = ((2*Fo*h*dx*Tinf)/k);    % boundary
end

%%

```

```

% Convection node/s on top
for ii = (nNodes)-(NperRow - 2) : (nNodes -1) %generalized

    jj = ii;

                                                                    % no top
term
    C(ii, jj-NperRow) = 2*Fo;                                % lower
    C(ii, jj+1)        = Fo;                                % right
    C(ii, jj-1)        = Fo;                                % left
    C(ii, jj)          = (-4*Fo)+((-2*Fo*h*dx)/(k)) + 1;    % center
    b(ii)              = (2*Fo*h*dx*Tinf)/(k);              % boundary
end

%starting corners 1-4
%%
% Corner 1 (this boundary condition should have same initiation
%   for any matrix size)
for ii = (NperCol*NperRow)-(NperRow-1)      %generalized

    jj = ii;

                                                                    % no
top term
    C(ii, jj-NperRow) = 2*Fo;                                %
lower
    C(ii, jj+1)        = 2*Fo;                                %
right
                                                                    % no
left term
    C(ii,jj)           = ((-4*Fo) + ((-2*Fo*h*dx)/k) + 1);    %
center
    b(ii)              = (((2*Fo*qy*dx)/k)+((2*Fo*h*dx*Tinf)/k)); %
boundary

                                                                    %EDIT: change flow of heat on boundary condition
end

%%
% Corner 2 (This shouldn't change either)

for ii = nNodes

    jj = ii;

                                                                    % no top
term

```

```

        C(ii, jj-NperRow) = 2*Fo; % lower
term
% no right
term
        C(ii, jj-1) = 2*Fo; % left
        C(ii, jj) = ((-4*Fo*h*dx)/(k)) + (-4*Fo) + 1; % center
        b(ii) = (-4*Fo*h*dx*Tinf)/k; % boundary
end

%%
% Corner 3 (should stay the same for all meshes)

for ii = 1

    jj = ii;

    C(ii, jj+NperRow) = 2*Fo; % upper
    % no lower
    C(ii, jj+1) = 2*Fo; % right
    % no left term
    C(ii, jj) = (-4*Fo) + 1; % center
    b(ii) = (((2*Fo*dx)/k)*(qx+qy)); % boundary

    %EDIT: Changed flow of heat on the boundary condition
end

%%
% Corner 4 (should be the same for all matrices)

for ii = NperRow

    jj = ii;

    C(ii, jj+NperRow) = 2*Fo; % upper
    % no
lower term
% no
right term
    C(ii, jj-1) = 2*Fo; % left
term
    C(ii, jj) = (-4*Fo) + ((-2*Fo*h*dx)/k) + 1; % center
    b(ii) = (((2*Fo*qx*dx) + (2*Fo*h*dx*Tinf))/k); %
boundary
end

```

```

%corners are finished
%Start heat sources
%%
%heat source left side

for ii = NperRow+1 : NperRow : (nNodes)-(NperRow+1)           %try to
generalize for all matrices

    jj = ii;

    C(ii, jj+NperRow) = Fo;                                     % upper
    C(ii, jj-NperRow) = Fo;                                     % lower
    C(ii, jj+1)        = 2*Fo;                                   % right
                                                                % no left term
    C(ii, jj)          = ((-4*Fo) + 1);                         % center
    b(ii)              = (2*Fo*qy*dx)/k;                       % boundary

    %EDIT: Changed flow of heat on the boundary condition, for
qy
end

%%
%heat source bottom

for ii = 2: (NperRow-1)    % generalized

    jj = ii;

    C(ii, jj+NperRow) = 2*Fo;                                     % upper
                                                                % no lower term
    C(ii, jj+1)        = Fo;                                     % right term
    C(ii, jj-1)        = Fo;                                     % left term
    C(ii, jj)          = (-4*Fo) + 1;                         % center
    b(ii)              = (2*Fo*qx*dx)/k;                       % boundary
end

%%
% here we are goint to actually solve for future T

T(:,i+1) = C*(T(:,i))+ b;
i        = i+1;

end

%%

```

```

% creating the mesh plot

x      = 0: dx :L;      % x-axis grid
y      = 0: dx :W;      % y-axis grid

[X,Y] = meshgrid(x,y);
%Create a loop that converts the vector back into a mesh to plot
%takes the temperature vector and reorganizes into a matrix

nIndex = 1; %this index will count down the temperature
for aa = 1:NperRow %nNodes/NperRow
    for bb = 1:NperRow
        TMesh(aa,bb) = T(nIndex,Tcol);
        nIndex = nIndex +1;
    end
end
%TMesh1 = abs(TMesh);
%recycled some of the example code

contourf(X,Y,TMesh)

colorbar

%contour looks ok,but heat might be flowing in the wring direction
%change the sign for heat flow on the right face of the figure
%^this should affect values for T1, T4, T7

```


57

$$(-\frac{2E_0 h \Delta x}{k})(T_0 - T_1) + (-2E_0)(T_2 - T_1) + (-E_0)(T_3 - T_2) + (-E_0)(T_4 - T_3) + \frac{1}{k} = T_2$$

$$T_c + \frac{k\Delta x}{h_2}(T_2 - T_c) + \frac{h\Delta x}{2}(T_\infty - T_c) + \frac{k\Delta x}{4\Delta x^2}(T_1 - T_c) = \frac{\rho\Delta x^2}{4} \frac{dT_c}{dt} - (T_2 - T_c^0)$$

$$\frac{h\Delta x}{k}(T_\infty - T_c) + (T_2 - T_c) + \frac{h\Delta x}{k}(T_\infty - T_c) + (T_1 - T_c) = \frac{1}{2h_2}(T_2 - T_c^0)$$

$$\frac{2F_0 h\Delta x}{k}(T_\infty - T_c) + 2F_0(T_2 - T_c) + \frac{2F_0 h\Delta x}{k}(T_\infty - T_c) + 2F_0(T_1 - T_c) = -T_c^0$$

$$\left(-\frac{2F_0 h\Delta x}{k}\right)(T_\infty - T_c) + (-2F_0)(T_2 - T_c) + \left(-\frac{2F_0 h\Delta x}{k}\right)(T_\infty - T_c) + (-2F_0)(T_1 - T_c) = T_c^0$$

$$\emptyset T_1 + (-2F_0)T_2 + \emptyset T_3 + (-2F_0)T_4 + \left(4F_0 + \frac{4F_0 h\Delta x}{k} + 1\right)T_c = \left[-\frac{4F_0 h\Delta x}{k}T_\infty\right] = T_c^0$$

annu 2.2

$$\frac{h\Delta x}{2}(T_\infty - T_c) + \frac{k\Delta x}{4\Delta x^2}(T_2 - T_c) + \frac{k\Delta x}{4\Delta x^2}(T_3 - T_c) = \frac{\rho\Delta x^2}{4} \frac{dT_c}{dt} - (T_2 - T_c^0) + q_2 \Delta x$$

$$\frac{h\Delta x}{k}(T_\infty - T_c) + (T_2 - T_c) + (T_3 - T_c) + \frac{q_2 \Delta x}{k} = \frac{1}{2h_2}(T_2 - T_c^0)$$

$$\frac{2F_0 h\Delta x}{k}(T_\infty - T_c) + 2F_0(T_2 - T_c) + 2F_0(T_3 - T_c) + \frac{2F_0 q_2 \Delta x}{k} - T_c = -T_c^0$$

$$\left(-\frac{2F_0 h\Delta x}{k}\right)(T_\infty - T_c) + (-2F_0)(T_2 - T_c) + (-2F_0)(T_3 - T_c) - \frac{2F_0 q_2 \Delta x}{k} + T_c = T_c^0$$

$$\emptyset T_1 + (-2F_0)T_2 + (-2F_0)T_3 + \emptyset T_4 + \left(\emptyset F_0 + \frac{2F_0 h\Delta x}{k} + 1\right)T_c = \left[-\frac{2F_0 q_2 \Delta x}{k}\right] = T_c^0$$

$$\frac{h\Delta x}{2}(T_1 - T_c) + \frac{q_1 \Delta x}{2} + \frac{k\Delta x}{4\Delta x^2}(T_2 - T_c) + \frac{q_2 \Delta x}{2} = \frac{\rho\Delta x^2}{4} \frac{dT_c}{dt} - (T_2 - T_c^0)$$

$$(T_1 - T_c) + \frac{q_1 \Delta x}{k} + (T_2 - T_c) + \frac{q_2 \Delta x}{k} = \frac{1}{2h_2}(T_2 - T_c^0)$$

$$2F_0(T_1 - T_c) + \frac{2F_0 q_1 \Delta x}{k} + 2F_0(T_2 - T_c) + \frac{2F_0 q_2 \Delta x}{k} - T_c = -T_c^0$$

$$4(-2F_0)(T_1 - T_c) - \frac{2F_0 q_1 \Delta x}{k} + (-2F_0)(T_2 - T_c) - \frac{2F_0 q_2 \Delta x}{k} + T_c = T_c^0$$

$$(-2F_0)T_1 + \emptyset T_2 + (-2F_0)T_3 + \emptyset T_4 + (4F_0 + 1)T_c = \left[\left(-\frac{2F_0 \Delta x}{k}\right)(q_1 + q_2)\right] = T_c^0$$

Matlab code part 3

```
% Jonathan DiBacco and Michael Tanja

% The first segment of the code is where all the parameters are
stated.
% The only thing you will have to change is the NperRow and the
NperCol
% The 'dx' changes according to the inputs for NperRow/Col

clear all
close all

%%
%parameters and
%mesh properties
prompt1 = 'What resolution of dx would do you want?          : ';
prompt2 = 'What is the length and width of this square plate?: ';
L      = input(prompt2);          % unsure about what this does, but it
was in the example
W      = L;                      % unsure about what this does, but it
was in the example
dx     = input(prompt1);
qx     = 100000;                 % heat flux bottom
qy     = -65000;                 % heat flux side
NperRow = (L/dx) + 1;            % Nodes per row
NperCol = NperRow;               % Nodes per column
nNodes  = NperRow * NperCol;
C       = zeros(NperRow, NperCol);
b       = zeros(NperRow,1);

k       = 250;                   % conduction coefficient
h       = 250;                   % convection coefficient
Tinf    = 0;                     % fluid temp in celsius

dt      = input('what is the dt?                                : ');
        % delta t
endtime = input('how long should the simulation run?           : ');
        % setting end time
Tcol    = endtime/dt;            % number of temperature colloms
T       = zeros(nNodes,Tcol);    % initializing Temperature %Tcol
T(:,1)  = 400*ones(1, nNodes);  % setting initial condition to
Temperature = 400 at Time = 0

% got rid of ro and cp
```

```

a      = 1e-4;                                % thermal diffusivity
Fo      = (a*dt)/(dx*dx);                      %Fourier
i = 1;
%%
% setting up the time loop
for t = 1:dt:endtime

% next in line is node set up
%%
% Center Node/s

for ii = NperRow+2 : ((2*NperRow-1))
    for jj = ii:NperRow:nNodes-(NperRow+1)%NperRow+2 : (NperRow^2) -
(NperRow+1) %put this first in order, matlab reads top down

        C(jj, jj+NperRow) = -Fo;                % upper
        C(jj, jj-NperRow) = -Fo;                % lower
        C(jj, jj+1)      = -Fo;                % right
        C(jj, jj-1)      = -Fo;                % left
        C(jj, jj)        = ((4*Fo)+1);         % center
        b(jj)            = 0;                  % boundary
    end
end

%%
% Convection node/s on the right

for ii = 2*NperRow : NperRow : (NperRow^2) - NperRow
    %needs to be more generalized

    jj = ii;

    C(ii, jj+NperRow) = -Fo;                    % upper
    C(ii, jj-NperRow) = -Fo;                    % lower
                                                % no right
    C(ii, jj-1)      = -2*Fo;                    % left
    C(ii, jj)        = ((2*Fo*h*dx)/k)+(4*Fo)+1); % center
    b(ii)            = ((2*Fo*h*dx*Tinf)/k);     % boundary
end

%%

```

```

% Convection node/s on top
for ii = (nNodes)-(NperRow - 2) : (nNodes -1) %generalized

    jj = ii;

    C(ii, jj-NperRow) = -2*Fo; % no top term
    C(ii, jj+1) = -Fo; % lower
    C(ii, jj-1) = -Fo; % right
    C(ii, jj) = (4*Fo)+((2*Fo*h*dx)/(k)) + 1; % left
    b(ii) = (2*Fo*h*dx*Tinf)/(k); % center
end % boundary

%starting corners 1-4
%%
% Corner 1 (this boundary condition should have same initiation
% for any matrix size)
for ii = (NperCol*NperRow)-(NperRow-1) %generalized

    jj = ii;

    C(ii, jj-NperRow) = -2*Fo; % no
    C(ii, jj+1) = -2*Fo; %
    C(ii, jj) = ((4*Fo) + ((2*Fo*h*dx)/k) + 1); %
    b(ii) = (((2*Fo*qy*dx)/k)+((2*Fo*h*dx*Tinf)/k)); %
    %EDIT: change flow of heat on boundary condition
end

%%
% Corner 2 (This shouldn't change either)

for ii = nNodes

    jj = ii;

    C(ii, jj-NperRow) = -2*Fo; % no top term
    % no right
    % no left
    % no bottom
    % no center
    % no boundary
end

```

```

        C(ii, jj-1)      = -2*Fo;                                % left
        C(ii, jj)        = ((4*Fo*h*dx)/(k)) + (4*Fo) + 1; % center
        b(ii)            = (4*Fo*h*dx*Tinf)/k;                  % boundary
    end

%%
% Corner 3 (should stay the same for all meshes)

for ii = 1

    jj = ii;

    C(ii, jj+NperRow) = -2*Fo;                                % upper
                                                                % no lower
    C(ii, jj+1)        = -2*Fo;                                % right
                                                                % no left term
    C(ii, jj)          = (4*Fo) + 1;                            % center
    b(ii)              = (((2*Fo*dx)/k)*(qx+qy)); % boundary

                                %EDIT: Changed flow of heat on the boundary condition
end

%%
% Corner 4 (should be the same for all matrices)

for ii = NperRow

    jj = ii;

    C(ii, jj+NperRow) = -2*Fo;                                %
upper                                                                % no
                                                                % no
lower term                                                            % no
right term
    C(ii, jj-1)      = -2*Fo;                                % left
term
    C(ii,jj)          = (4*Fo)+((2*Fo*h*dx)/k) + 1;          % center
    b(ii)            = (((2*Fo*qx*dx)+(2*Fo*h*dx*Tinf))/k); %
boundary
end

%corners are finished
%Start heat sources
%%

```

```

%heat source left side

for ii = NperRow+1 : NperRow : (nNodes)-(NperRow+1)           %try to
generalize for all matrices

    jj = ii;

    C(ii, jj+NperRow) = -Fo;           % upper
    C(ii, jj-NperRow) = -Fo;           % lower
    C(ii, jj+1)       = -2*Fo;         % right
                                % no left term
    C(ii, jj)         = ((4*Fo) + 1); % center
    b(ii)             = (2*Fo*qy*dx)/k; % boundary

    %EDIT: Changed flow of heat on the boundary condition, for
qy
end

%%
%heat source bottom

for ii = 2: (NperRow-1)    % generalized

    jj = ii;

    C(ii, jj+NperRow) = -2*Fo;           % upper
                                % no lower term
    C(ii, jj+1)       = -Fo;           % right term
    C(ii, jj-1)       = -Fo;           % left term
    C(ii, jj)         = (4*Fo) + 1;     % center
    b(ii)             = (2*Fo*qx*dx)/k; % boundary
end

%%
% here we are goint to actually solve for future T

K      = inv(C);
T(:,i+1) = K*T(:,i)+ K*b;
i       = i+1;

end

%%
% creating the mesh plot

```



```

x      = 0: dx :L;      % x-axis grid
y      = 0: dx :W;      % y-axis grid

[X,Y] = meshgrid(x,y);
%Create a loop that converts the vector back into a mesh to plot
%takes the temperature vector and reorganizes into a matrix

nIndex = 1; %this index will count down the temperature
for aa = 1:NperRow %nNodes/NperRow
    for bb = 1:NperRow
        TMesh(aa,bb) = T(nIndex,Tcol);
        nIndex = nIndex +1;
    end
end
%TMesh1 = abs(TMesh);
%recycled some of the example code

contourf(X,Y,TMesh)

colorbar

%contour looks ok,but heat might be flowing in the wring direction
%change the sign for heat flow on the right face of the figure
%^this should affect values for T1, T4, T7

```