

ENGR-UH 1001
Computer Programming for Engineers

Assignment 4 - Mechanical Engineering Case Study
(Temperature Distribution Simulation)

Muhammad Zunair Viqar
Mzv205
Fall 2019

a) Step 1: Problem Identification and Statement

The Objective is to write the program code for a software that models the temperature distribution in a thin metal plate with a constant (isothermal) temperature on each side, using a 2D grid with N number of rows and M number of columns. The software asks the user to input the number of rows and columns, the temperature of the four sides of the metal plate and the specific threshold for equilibrium. Using this information, going across the rows of the matrix, the software models an animation on the temperature distribution across the metal plate until the temperature difference between the previous and the current value at all points becomes less than the specific threshold.

b) Step 2: Gathering Information and I/O Description

In order to model the temperature distribution in a thin metal plate using a 2D grid, we use the following working described with the variables provided along with the input from the user and gather all missing information in one place.

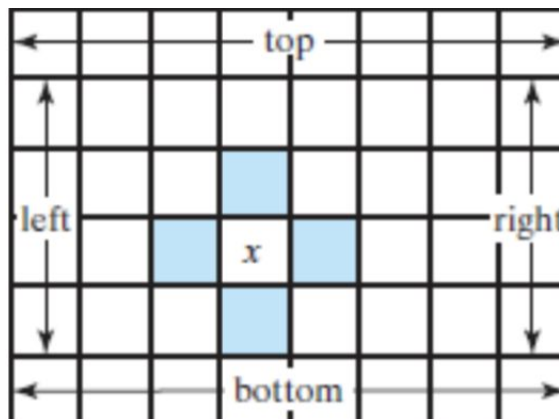
- User Inputs and Constants:
 - Number of Rows: N
 - Number of Columns: M
 - Temperature of the Top of the Metal Plate: temp(1)
 - Temperature of the Bottom of the Metal Plate: temp(2)
 - Temperature of the Left of the Metal Plate: temp(3)
 - Temperature of the Right of the Metal Plate: temp(4)
 - Tolerance Value: tv
- Matrices:
 - A NxM matrix to store the current temperatures: tempgrid(N,M)
 - A NxM matrix to store the previous temperatures: x(N,M)
 - A NxM matrix to store the difference between the temperatures: diff(N,M)
- Other variables:
 - Time: time

We initialise the tempgrid matrix to 0 and then use the Temperatures of the four sides of the metal plate, input by the user, and assign these values to the respective rows and columns of the tempgrid matrix as follows:

(The attached image is only to show how the tempgrid matrix should be at the beginning)

		N - Number of Rows													
		←-----temp(1)-----→													
M Number Of Columns	↑ ↓	←-----temp(3)-----→													
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
				←-----temp(2)-----→											

Following this as a starting point, we use the concept that the temperature of the interior points changes according to the temperature around them. The temperature of an interior point can be computed as the average of the four adjacent temperatures; the points shaded in the figure below represent the adjacent temperatures for the pointed labeled x in the grid.



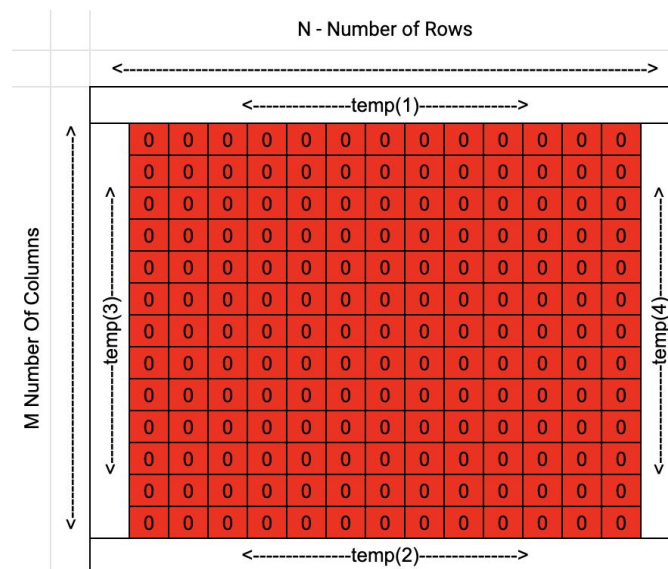
Each time that the temperature of a point in the matrix changes, the temperatures of the points adjacent to it changes. This continues until a thermal equilibrium is achieved and all temperatures

become constant. Thermal Equilibrium is when the difference between the previous temperature at a point in the matrix and the current temperature at a point in the matrix is less than tv (tolerance value).

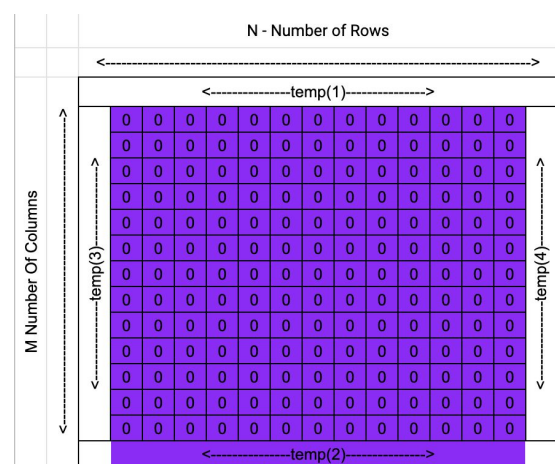
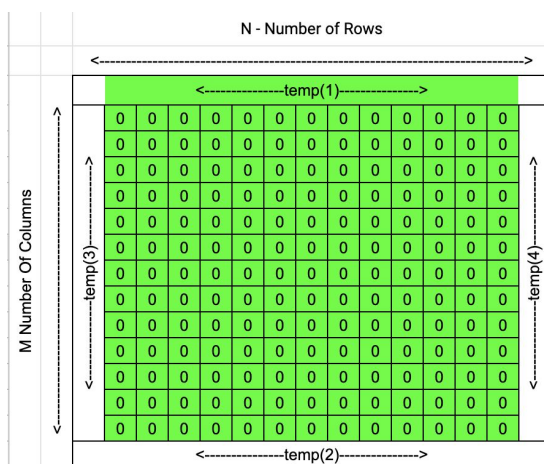
In order to calculate the difference to check for the thermal equilibrium, we copy all the values stored in the matrix to another matrix called x. Then we perform the following Matrix Addition:

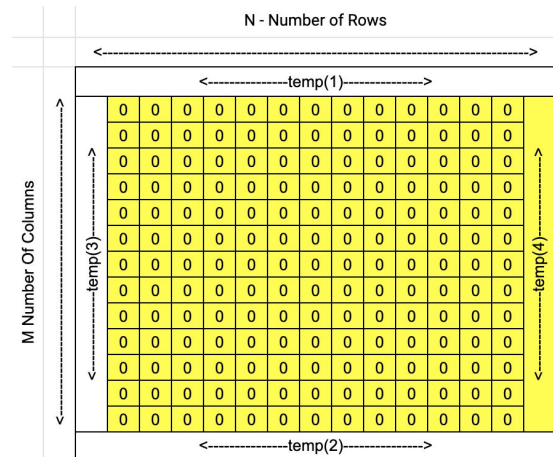
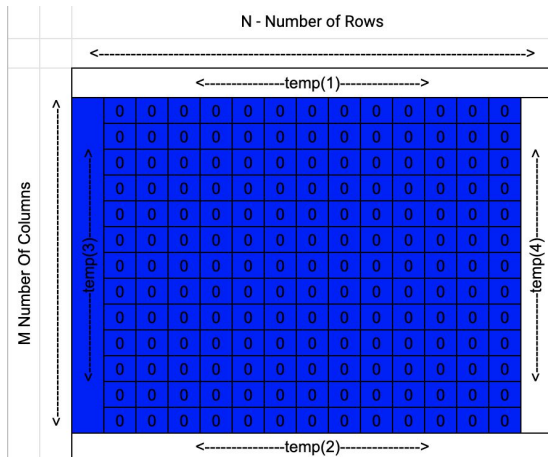
(The following is shown in accordance with the first iteration of the loop only)

- We use the red highlighted section of the matrix shown below as the destination for the matrix addition result:



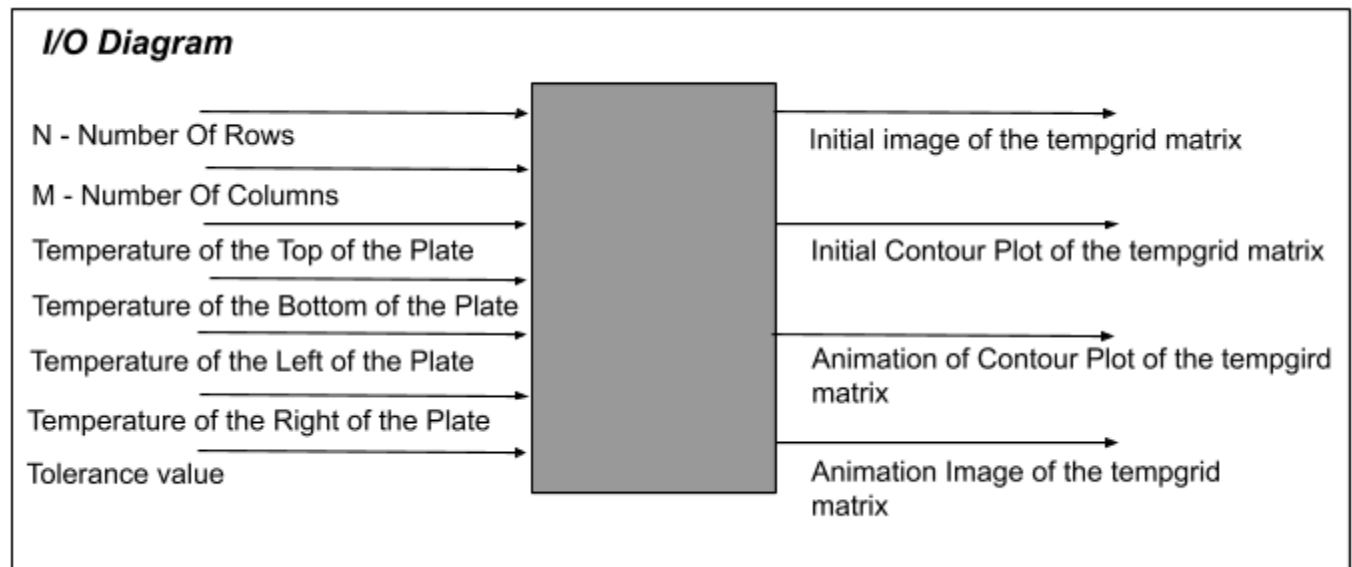
- Then, we add the green, purple, blue and yellow sections of the matrix together to get the required result of adding the adjacent values to each point:





As we store the newly calculated value in the tempgrid matrix, we then calculate the difference between the previous value and the current value at every point and store the result in the diff matrix.

The above shown matrix addition calculations keep on happening with the updated values of the matrices until every single value in the diff matrix becomes less than the tolerance value that is input by the user.



c) Step 3: Test Cases and algorithm

- Test Cases:

Test cases use sample input data that is used in manual calculations by the user to produce expected output data in order to verify the functionality of the program. For Test Cases, different ranges of data are used as follows;

Test Case 1:

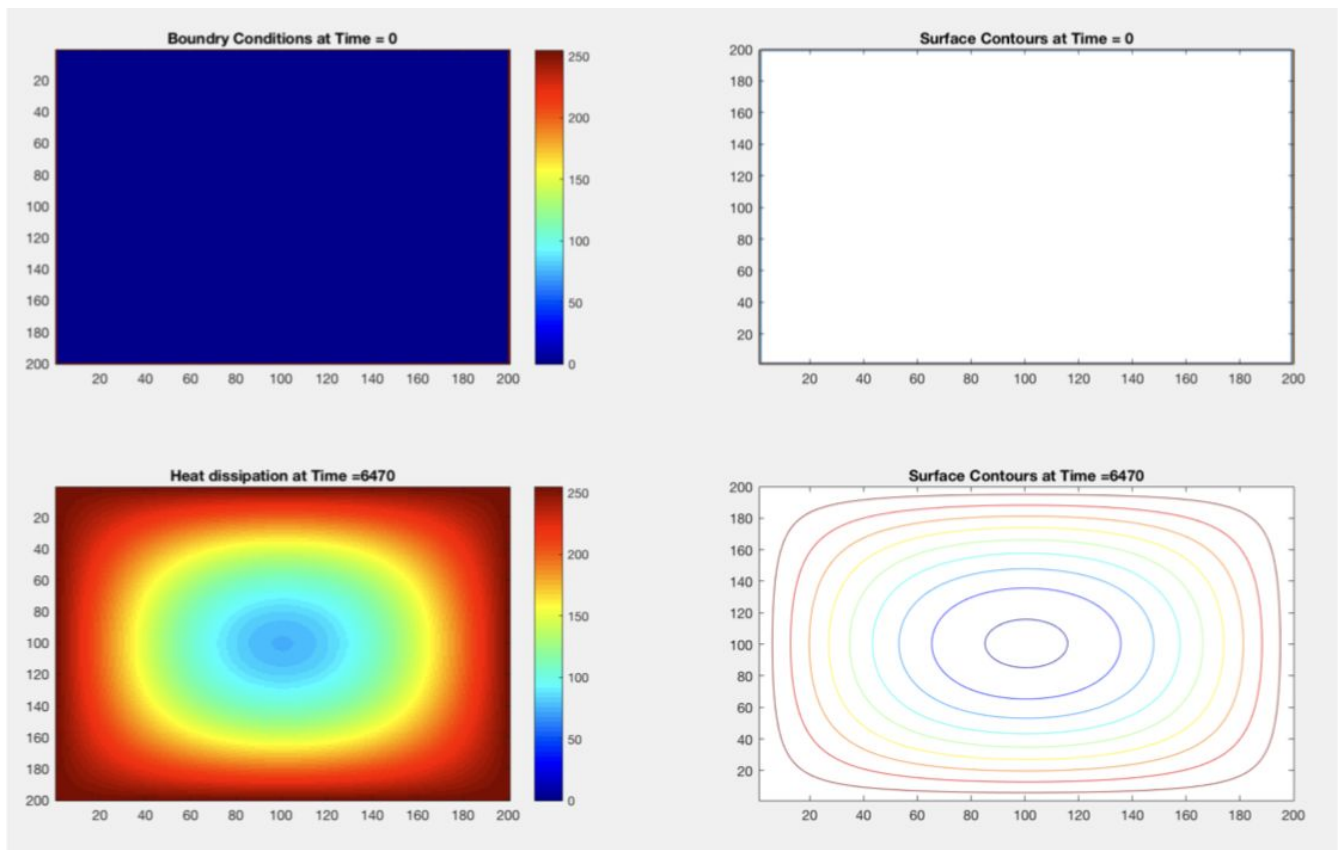
(Hot Edges)

Dimensions of grid: 200 x 200

Boundary Conditions: 255, 255, 255, 255

Simulation Threshold: 0.02

The desired result should be as follows:



Test Case 2:

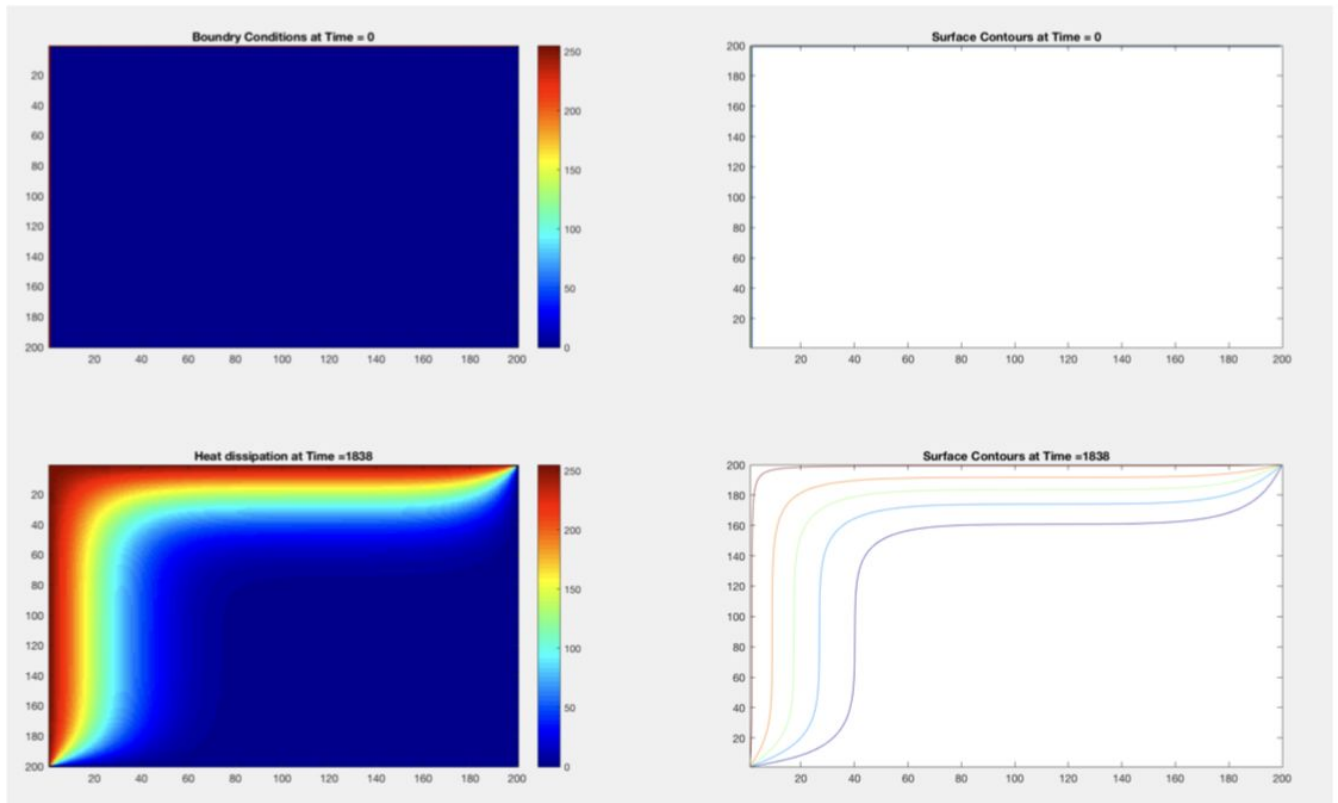
(Mixed Boundary Conditions)

Dimensions of grid: 200 x 200

Boundary Conditions: 255, 0, 255, 0

Simulation Threshold: 0.05

The desired result should be as follows:



Test Case 3:

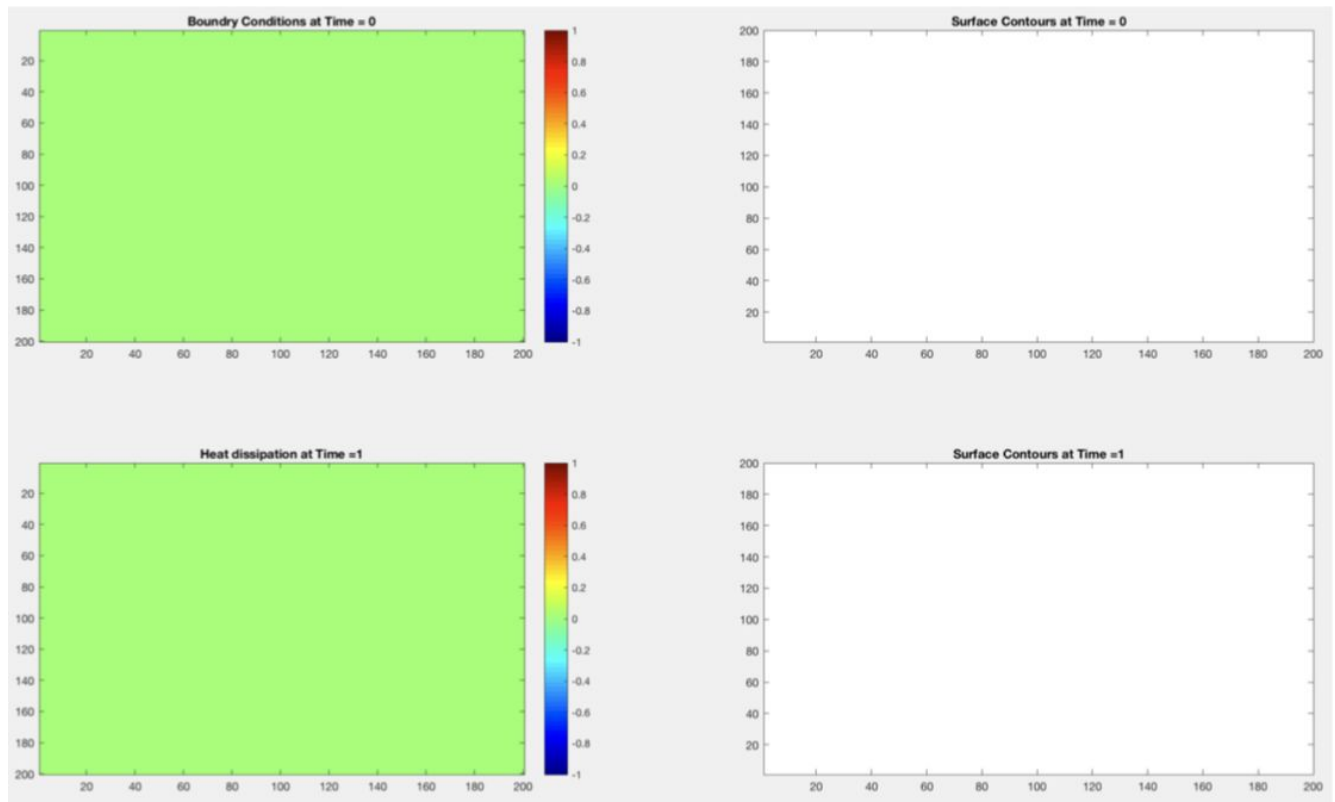
(Zero Boundary Conditions)

Dimensions of grid: 200 x 200

Boundary Conditions: 0, 0, 0, 0

Simulation Threshold: 0.01

The desired result should be as follows:



Test Case 4:

(Invalid Values)

Dimensions of grid: -20

Should Prompt the user to input again!

Test Case 5:

(Invalid Values)

Dimensions of grid: 200 x 40

Should Prompt the user to input again!

Test Case 6:

(Invalid Values)

Dimensions of grid: 200 x 200

Boundary Conditions: -10

Should Prompt the user to input again!

Test Case 7:

(Invalid Values)

Dimensions of grid: 200 x 200

Boundary Conditions: 255, -100

Should Prompt the user to input again!

Test Case 8:

(Invalid Values)

Dimensions of grid: 200 x 200

Boundary Conditions: 255, 255, 500

Should Prompt the user to input again!

Test Case 9:

(Invalid Values)

Dimensions of grid: 200 x 200

Boundary Conditions: 255, 255, 255, 1000

Should Prompt the user to input again!

Test Case 10:

(Invalid Values)

Dimensions of grid: 200 x 200

Boundary Conditions: 255, 255, 255, 255

Simulation Threshold: 0.06

Should Prompt the user to input again!

- Algorithm:

Main Function()

Valid validval values are in the range from 50 to 200 in steps of 1

Print "Enter The Number Of Rows: "

Read value into N

Repeat while N is not equal to validval

Print "Invalid! Enter Number Of Rows Again: "

Read value into N

Print "Enter The Number Of Columns: "

Read value into M

Repeat while M is not equal to validval

Print "Invalid! Enter Number Of Columns Again: "

Read value into M

Assign 0 to tempgrid(N,M)

Assign 1 to i

Repeat while i is less than 5

If i is equal to 1

Print "Enter the Temperature for Top of the Metal Plate: "

Read value into temp(i)

Repeat while temp(i) is less than 0 or greater than 255

Print "Invalid! Enter the Temperature for Top of the Metal Plate Again:"

Otherwise if i is equal to 2

Print "Enter the Temperature for Bottom of the Metal Plate:"

Read value into temp(i)

Repeat while temp(i) is less than 0 or greater than 255

Print "Invalid! Enter the Temperature for Bottom of the Metal Plate Again:"

Otherwise if i is equal to 2

Print "Enter the Temperature for Left of the Metal Plate:

Read value into temp(i)

Repeat while temp(i) is less than 0 or greater than 255

Print "Invalid! Enter the Temperature for Left of the Metal Plate Again:"

Otherwise

Print "Enter the Temperature for Right of the Metal Plate:"

Read value into temp(i)

Repeat while temp(i) is less than 0 or greater than 255

Print "Invalid! Enter the Temperature for Right of the Metal Plate Again:"

Increment i

Print "Enter The Tolerance Value"

Read value into tv

Repeat while tv is less than 0 or greater than 0.05

Print "Invalid! Enter The Tolerance Value Again: "

Read value into tv

Assign 1 to a

Repeat while a is less N+1

Assign temp(3) to tempgrid(a,1)

Increment a

Assign 1 to b

Repeat while b is less N+1

Assign temp(4) to tempgrid(b,M)

Increment b

Assign 1 to c

Repeat while c is less M+1

Assign temp(3) to tempgrid(1,c)

Increment c

Assign 1 to d

Repeat while d is less M+1

Assign temp(3) to tempgrid(N,d)

Increment d

Assign 0 to x(N,M)
Assign 1 to diff(N,M)
Assign 0 to time

Assign a 2x2 grid to the figure
Assign 1st Position in grid to the next plot
Print an image of the matrix tempgrid
Print Title "Boundry Conditions at time = ", time
Assign the jet colormap to the image
Assign colorbar to the image

Assign 2nd position in grid to the next plot
Print a contour plot of the matrix tempgrid
Print Title "Surface Contours at time = ", time

While true

For i = 2 to N-1
For j = 2 to M-1
Assign tempgrid(i,j) to x(i,j)

Assign (tempgrid(2:(N-1),1:(M-2)) + tempgrid(2:(N-1),3:M) + tempgrid(1:(N-2),2:(M-1)) + tempgrid(3:N,2:(M-1)))/4 to
tempgrid(2:(N-1),2:(M-1))

For m= 2 to N-1
For n= 2 to M-1
Assign (tempgrid(i,j) - x(i,j)) to diff(i,j)

Assign 1 to e
Repeat while a is less N+1
Assign 0 to diff(a,1)
Increment a

Assign 1 to f
Repeat while b is less N+1
Assign 0 to diff(f,M)
Increment b

Assign 1 to g
Repeat while c is less M+1
Assign 0 to diff(1,g)
Increment c

Assign 1 to h
Repeat while d is less M+1
Assign 0 to diff(N,h)
Increment d

Assign (time + 1) to time

Assign 3rd Position in grid to the next plot
Print an image of the matrix tempgrid
Print Title "Heat Dissipation at time = ", time
Assign the jet colormap to the image
Assign colorbar to the image
Print the image as the loop is running on 20 frames per second

Assign 4th position in grid to the next plot
Print a contour plot of the matrix tempgrid
Print Title "Surface Contours at time = ", time
Print the image as the loop is running on 20 frames per second

If all the values in the diff matrix are less than tv
Break the loop

d) Step 4: Code or implementation

% The Software models the temperature distribution in a thin metal plate with constant (isothermal) temperatures on each side, using a 2D grid (matrix).

validNandM = 50:1:200;

% Range for the valid values of N and M

N = input('Enter The Number Of Rows: ');

% Prompts the user to input a value for N

while ~any(N==validNandM)

 N = input('Invalid! Enter Number Of Rows Again: ');

end

%loop to validate that the value input is within the range

M = input('Enter The Number of Columns: ');

while ~any(M==validNandM)

 M = input('Invalid! Enter Number Of Columns Again: ');

end

tempgrid = zeros(N,M);

% creates and initialises a NxM matrix called tempgrid with all zeros

for i = 1:4

 if i ==1

 temp(i) = input ('Enter the Temperature for Top of the Metal Plate: ');

 %Prompts the user to input a value

 while (temp(i)<0) || (temp(i)>255)

 temp(i) = input('Invalid! Enter the Temperature for Top of the Metal Plate Again: ');

 end

 %loop to validate that the value input is within the range

 elseif i ==2

 temp(i) = input ('Enter the Temperature for Bottom of the Metal Plate: ');

 while (temp(i)<0) || (temp(i)>255)

 temp(i) = input('Invalid! Enter the Temperature for Bottom of the Metal Plate Again : ');

 end

 elseif i ==3

 temp(i) = input ('Enter the Temperature for Left of the Metal Plate: ');

 while (temp(i)<0) || (temp(i)>255)

 temp(i) = input('Invalid! Enter the Temperature for Left of the Metal Plate Again: ');

 end

 else

 temp(i) = input ('Enter the Temperature for Right of the Metal Plate: ');

 while (temp(i)<0) || (temp(i)>255)

 temp(i) = input('Invalid! Enter the Temperature for Right of the Metal Plate Again: ');

 end

 end

end

% loop runs 4 times to prompt the user for input in each iteration

tv = input('Enter The Tolerance Value: ');

while (tv<0) || (tv>0.05)

 tv = input('Invalid! Enter The Tolerance Value Again: ');

end

tempgrid(:,1) = temp(3);

%Sets the entire first column of matrix tempgrid with temperature for left of the metal plate

tempgrid(:,M) = temp(4);

%Sets the entire last column of matrix tempgrid with temperature for right of the metal plate

tempgrid(1,:) = temp(1);

%Sets the entire first row of matrix tempgrid with temperature for top of the metal plate

tempgrid(N,:) = temp(2);

%Sets the entire last row of matrix tempgrid with temperature for bottom of the metal plate

```

x = zeros(N,M);
% creates and initialises a NxM matrix called x with all zeros
diff = ones(N,M);
% creates and initialises a NxM matrix called diff with all ones
time = 0;
% initialises time to 0

subplot(2,2,1)
% subplot(m,n,p) divides the current figure into an m-by-n grid and creates axes in the position specified by p
% this creates a 2x2 grid and positions the following image in the 1st position
image(tempgrid)
%prints an image of the matrix tempgrid
title(['Boundary Conditions at time = ',num2str(time)])
%creates a title for the image
colormap(jet), colorbar
%defines the colors for the image and shows the colorbar

subplot(2,2,2)
% positions the following contour graph on the 2nd position
contour(tempgrid)
% prints a contour plot for the matrix tempgrid
title(['Surface Contours at time = ',num2str(time)])

while 1
x(2:(N-1),2:(M-1)) = tempgrid(2:(N-1),2:(M-1));
%stores the previous values of the matrix tempgrid into the matrix x
tempgrid(2:(N-1),2:(M-1)) = (tempgrid(2:(N-1),1:(M-2)) + tempgrid(2:(N-1),3:M) + tempgrid(1:(N-2),2:(M-1)) + tempgrid(3:N,2:(M-1)))/4;
%Uses Matrix Algebra to add the adjacent values of each point altogether
diff(2:(N-1),2:(M-1)) = tempgrid(2:(N-1),2:(M-1)) - x(2:(N-1),2:(M-1));
%calculates the difference between the previous and current values of the matrix tempgrid and stores into another matrix called diff
    diff(1,:) = 0;
    diff(N,:) = 0;
    diff(:,1) = 0;
    diff(:,M) = 0;
time = time + 1;
%increases the time with each iteration

subplot(2,2,3)
image(tempgrid)
title(['Heat Dissipation at time = ',num2str(time)])
colormap(jet), colorbar
drawnow limitrate;
% updates the figure as it is being processed and limits the number of updates to 20 frames per second

subplot(2,2,4)
contour(flipud(tempgrid))
title(['Surface Contours at time = ',num2str(time)])
drawnow limitrate;

if all(diff<tv)
    break;
end
%ends the iterations when the difference for all the grid values is less than the tolerance value
end

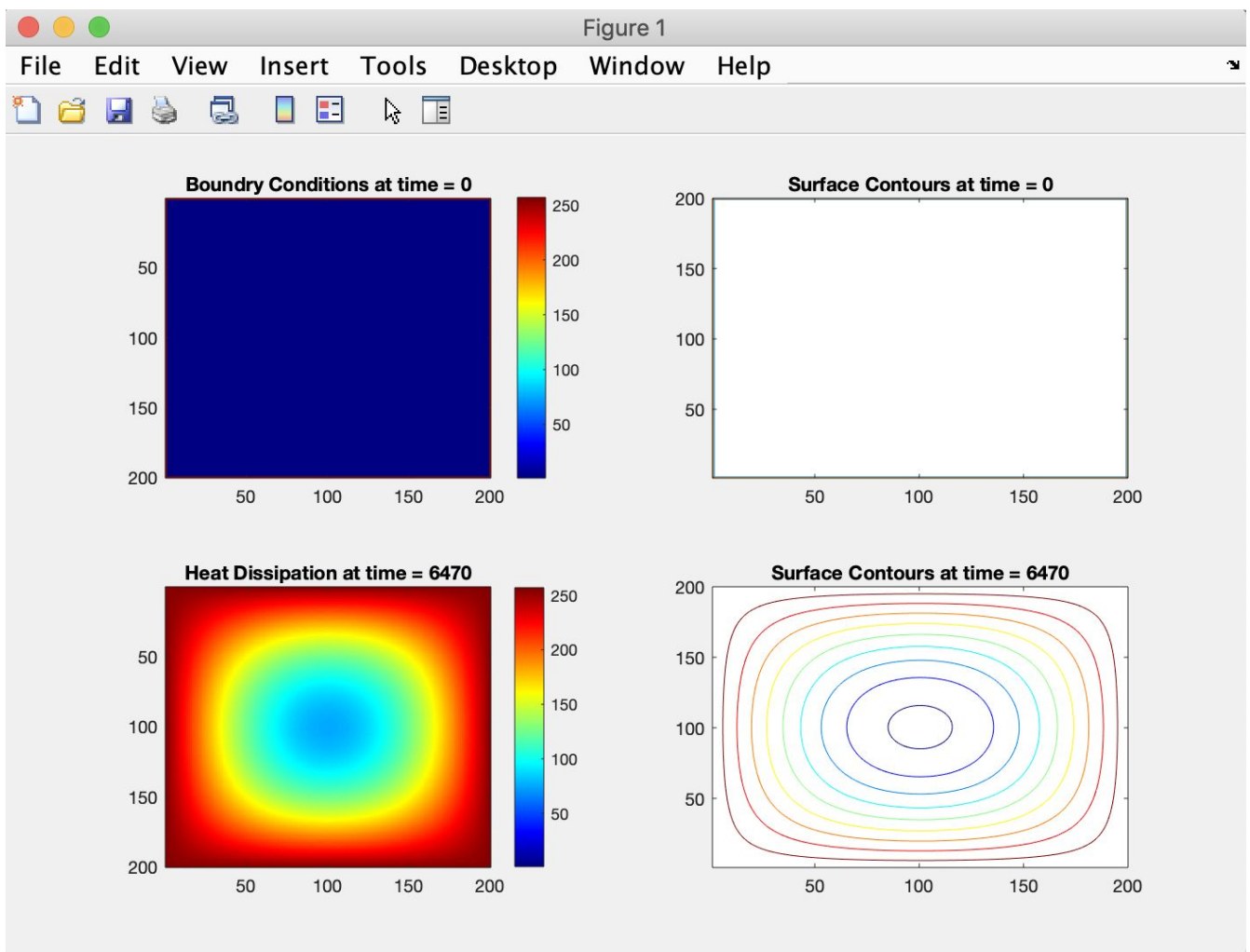
```

e) Step 5: Test and Verification

This step ensures if the final source code contains any errors or not. All the test cases in Step 3 are tried by running the program and the expected outcomes are compared with the actual outcomes.

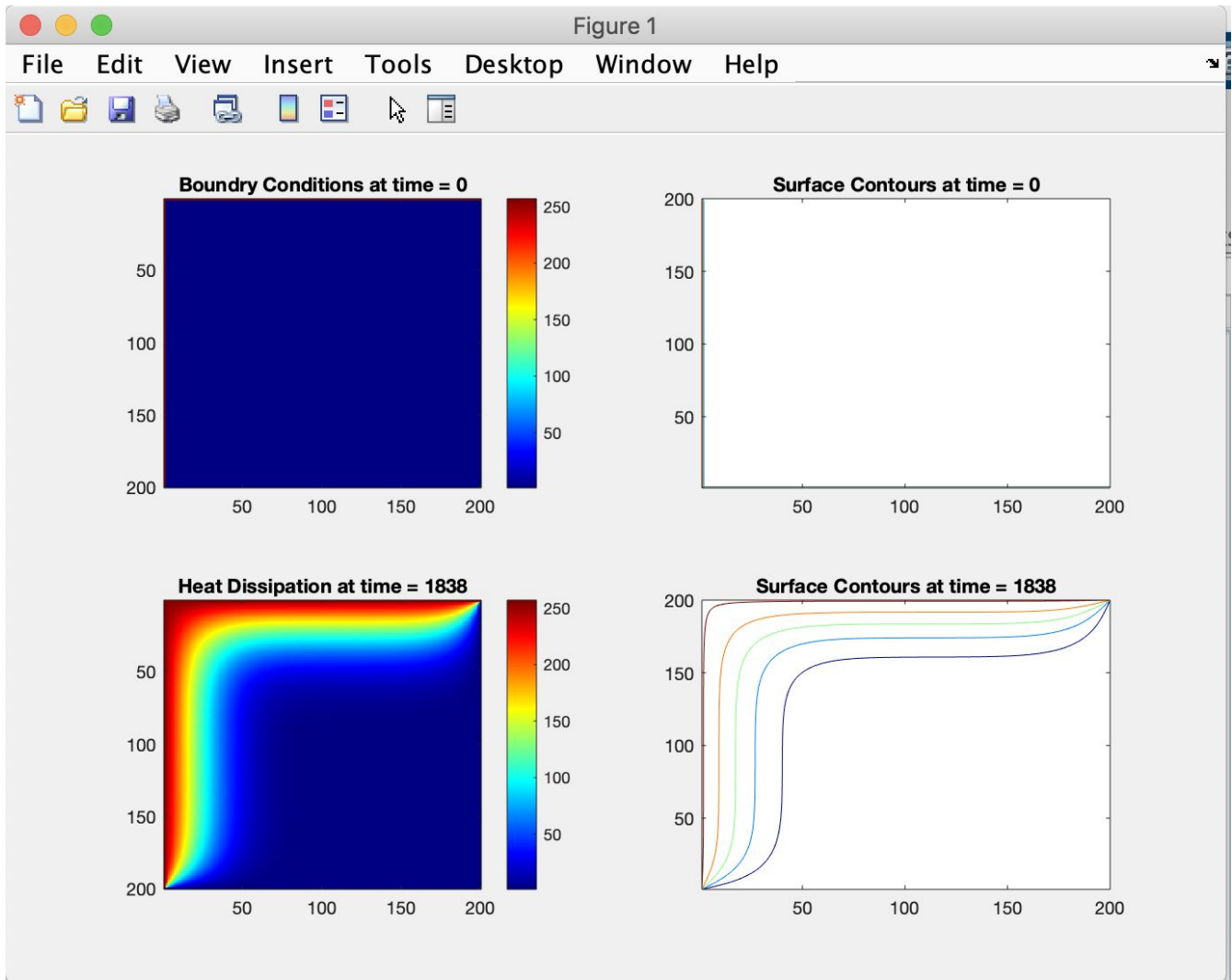
Test Case 1:

```
Editor - ComputerProgrammingAssignment4Code.m  Command Window
Enter The Number Of Rows: 200
Enter The Number of Columns: 200
Enter the Temperature for Top of the Metal Plate: 255
Enter the Temperature for Bottom of the Metal Plate: 255
Enter the Temperature for Left of the Metal Plate: 255
Enter the Temperature for Right of the Metal Plate: 255
Enter The Tolerance Value: 0.02
fx >>
```



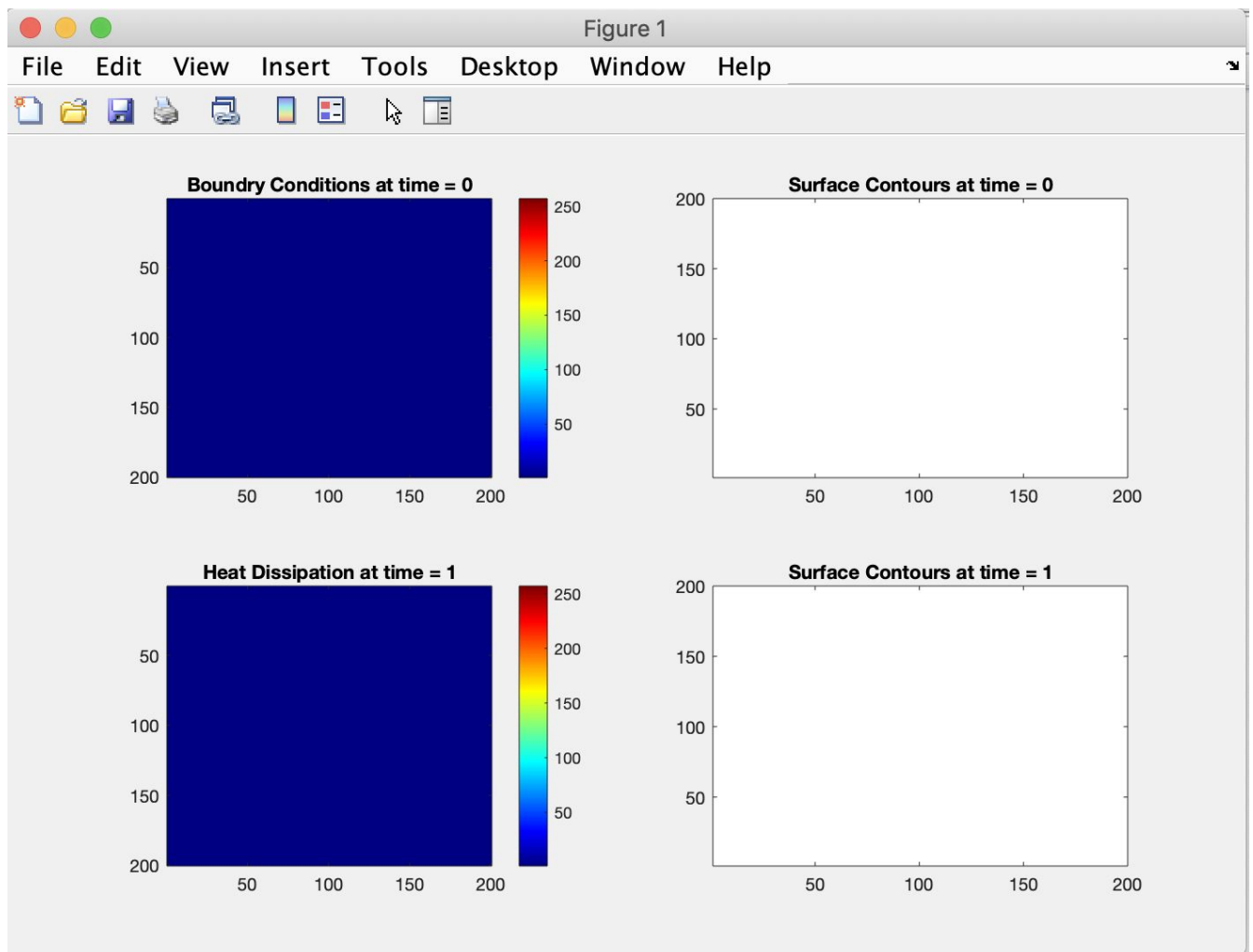
Test Case 2:

```
Editor - ComputerProgrammingAssignment4Code.m  Command Window
Enter The Number Of Rows: 200
Enter The Number of Columns: 200
Enter the Temperature for Top of the Metal Plate: 255
Enter the Temperature for Bottom of the Metal Plate: 0
Enter the Temperature for Left of the Metal Plate: 255
Enter the Temperature for Right of the Metal Plate: 0
Enter The Tolerance Value: 0.05
fx >> |
```



Test Case 3:

```
Enter The Number Of Rows: 200
Enter The Number of Columns: 200
Enter the Temperature for Top of the Metal Plate: 0
Enter the Temperature for Bottom of the Metal Plate: 0
Enter the Temperature for Left of the Metal Plate: 0
Enter the Temperature for Right of the Metal Plate: 0
Enter The Tolerance Value: 0.01
Warning: Contour not rendered for constant ZData
Warning: Contour not rendered for constant ZData
fx >> |
```



Test Case 4:



Editor - ComputerProgrammingAssignment4Code.m Command Window

```
Enter The Number Of Rows: -20
fx Invalid! Enter Number Of Rows Again:
```

Test Case 5:



Editor - ComputerProgrammingAssignment4Code.m Command Window

```
Enter The Number Of Rows: 200
Enter The Number of Columns: 40
fx Invalid! Enter Number Of Columns Again: |
```

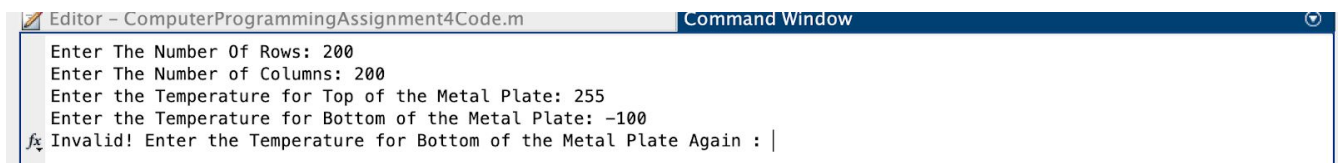
Test Case 6:



Editor - ComputerProgrammingAssignment4Code.m Command Window

```
Enter The Number Of Rows: 200
Enter The Number of Columns: 200
Enter the Temperature for Top of the Metal Plate: -10
fx Invalid! Enter the Temperature for Top of the Metal Plate Again:
```

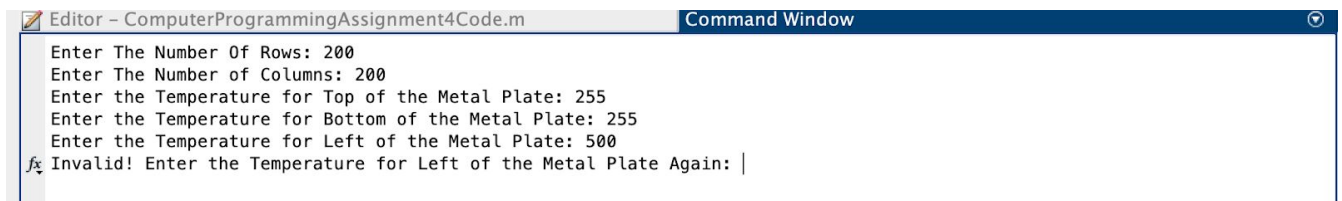
Test Case 7:



Editor - ComputerProgrammingAssignment4Code.m Command Window

```
Enter The Number Of Rows: 200
Enter The Number of Columns: 200
Enter the Temperature for Top of the Metal Plate: 255
Enter the Temperature for Bottom of the Metal Plate: -100
fx Invalid! Enter the Temperature for Bottom of the Metal Plate Again : |
```

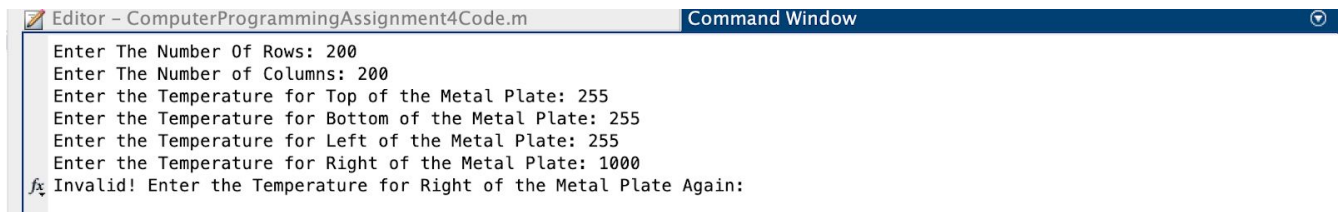
Test Case 8:



Editor - ComputerProgrammingAssignment4Code.m Command Window

```
Enter The Number Of Rows: 200
Enter The Number of Columns: 200
Enter the Temperature for Top of the Metal Plate: 255
Enter the Temperature for Bottom of the Metal Plate: 255
Enter the Temperature for Left of the Metal Plate: 500
fx Invalid! Enter the Temperature for Left of the Metal Plate Again: |
```

Test Case 9:

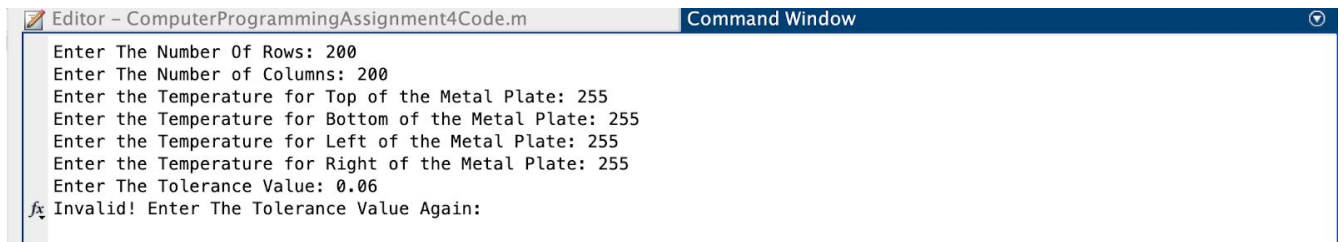


The image shows a MATLAB interface with two windows. The 'Editor' window displays the file 'ComputerProgrammingAssignment4Code.m' with the following code:

```
Enter The Number Of Rows: 200
Enter The Number of Columns: 200
Enter the Temperature for Top of the Metal Plate: 255
Enter the Temperature for Bottom of the Metal Plate: 255
Enter the Temperature for Left of the Metal Plate: 255
Enter the Temperature for Right of the Metal Plate: 1000
fx Invalid! Enter the Temperature for Right of the Metal Plate Again:
```

The 'Command Window' is currently empty.

Test Case 10:



The image shows a MATLAB interface with two windows. The 'Editor' window displays the file 'ComputerProgrammingAssignment4Code.m' with the following code:

```
Enter The Number Of Rows: 200
Enter The Number of Columns: 200
Enter the Temperature for Top of the Metal Plate: 255
Enter the Temperature for Bottom of the Metal Plate: 255
Enter the Temperature for Left of the Metal Plate: 255
Enter the Temperature for Right of the Metal Plate: 255
Enter The Tolerance Value: 0.06
fx Invalid! Enter The Tolerance Value Again:
```

The 'Command Window' is currently empty.

User Guide

This software can be used by Mechanical Engineers for modelling an animation of the temperature distribution in a thin metal plate and show a demonstration of how heat dissipates across the metal plate. The software considers a constant temperature of the boundaries of the metal plate that does not change and is only input by the user. The user can input different boundary temperatures, use a different size of the matrix being used to demonstrate and also use different thresholds for the equilibriums. The Software outputs four image plots on the same figure to allow for comparison. One is an image plot at time = 0, second is a contour plot at time = 0. The remaining two are animations of the image and contour plots as the equilibrium is being reached from the starting point.