Lab 4: Heat Maps

A. Background:

Often times in engineering, you will be collecting data that is two dimensional in that you are looking at the value of some property across a surface. For instance, you might be looking at the heat gradient across a surface, such as a circuit board to identify components in need of a heat sink, or you might be looking at the forces acting on a roof surface of a structure you're designing due to a load. In either of these cases, and in many more, you no longer have a single independent and dependent variable, as we've been dealing with so far this semester. Instead, you now have two independent variables (x and y location) and one dependent variable corresponding to the magnitude of the attribute you are measuring. So how do we visualize this type of data?

One useful method is using a heat map (http://en.wikipedia.org/wiki/Heat_map, http://xkcd.com/1138/). This plot displays an array as in image where each value is represented by a color. In general, larger values are red (hot) and smaller values are blue (cold). You have all seen examples of heat maps. One example is thermal imaging. In this case, higher temperatures are associated with red colors and cooler colors are blue. The example below shows a heat map of a circuit board, where the hottest color is actually white.

Another example of heat maps is in medical imaging. The example below is of a PET scan, where the patient is given a substance that emits positrons which can be picked up by specialized equipment. Often this substance is tied to glucose, so that areas of the body with high rates of metabolism (such as cancers) can be identified by higher concentrations of the substance.

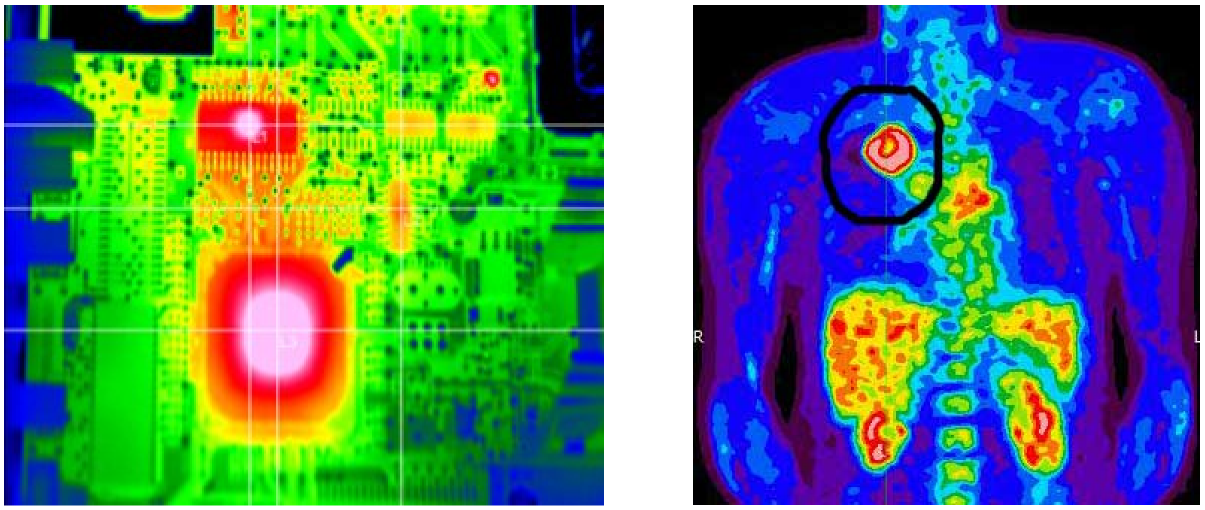
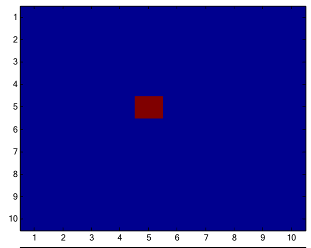


Figure 1: Examples of Heat Maps used for Thermal Imaging (left) and Medical Imaging (right)

In this lab, you will be expanding on the work you did last week for the heat transfer along a 1-D rod to instead look at the heat transfer on a 2-D plate.

B. Heat Map Practice

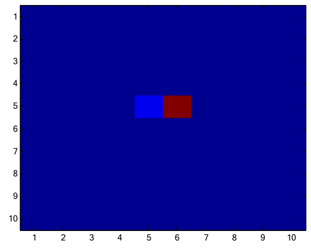
One way to create heat maps in MATLAB is to use the ***imagesc*** command, which takes an array and scales it so that the largest values display as red, the smallest values display as blue, and all values in between follow a rainbow pattern (Red, Orange, Yellow, Green, Blue).



Try the commands below at the MATLAB command window:

```
>> pic = zeros(10,10);
>> pic(5,5) = 1;
>> imagesc(pic);
```

You should see a dark blue box (background of all zeros) with a red box (the value of 1 that was added) in the middle.



Next, try this:

```
>> pic(5,6) = 10;
>> imagesc(pic);
```

Again, a dark blue box will appear, but (5,6) will be red and (5,5) will be light blue. This is because the value at (5,6) has a much larger value than the value at (5,5), which makes the value at (5,6) red and the value at (5,5) closer to the background blue color.

Try changing the values stored at locations (5,5) and (5,6) above and notice how the colors change in the resulting image. However, what will not change is that the largest value is always red and the background is always blue.

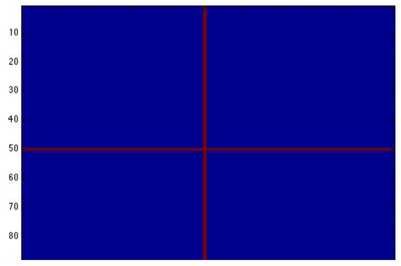
C. Creating Heat Maps

Use ***imagesc*** and **row/column addressing techniques** to create the following images.

(a) 101x101 blue image with a red cross centered in the middle.

MATLAB Commands:

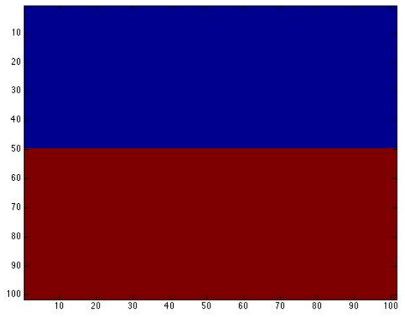
```
clear;clc;close all;
pic=zeros(101,101);
pic(:,50) = 10;
pic(50,:) = 10;
imagesc(pic);
```



(b) 100x100 blue image with the bottom half red.

MATLAB Commands:

```
clear;clc;close all;
pic=zeros(100,100);
pic(51:100,:) = 10;
imagesc(pic);
```



D. Understand the Process

As mentioned previously, we will be using heat maps to show the temperature gradient on a 2-D plate, similar to what was done last week with the temperature gradient along the 1-D rod. For this simulation, we will again assume that as time progresses, the temperature in a given section is the average of the surrounding sections. However, the calculations for this process are slightly more complicated as we now have several different locations we need to deal with:

- A corner section
- A side section
- A middle section

For a corner section, the new temperature will be computed by averaging the corner section with the two adjacent values. For instance, the equation for the upper left corner would be:

$T_{(1,1)\text{ new}}$	$T_{(1,2)\text{old}}$
$T_{(2,1)\text{old}}$	

$$T_{(1,1)\text{ new}} = \frac{\left(T_{(1,1)\text{old}} + T_{(1,2)\text{old}} + T_{(2,1)\text{old}}\right)}{3}$$

Similar equations can be created for the other three corners. For a side section, the new temperature will be computed be averaging the side section with the three adjacent values. In this case, the equation for a section along the left side would be:

$T_{(4,1)\text{old}}$	
$T_{(5,1)\text{ new}}$	$T_{(5,2)\text{old}}$
$T_{(6,1)\text{old}}$	

$$T_{(5,1)\text{ new}} = \frac{\left(T_{(5,1)\text{old}} + T_{(4,1)\text{old}} + T_{(6,1)\text{old}} + T_{(5,2)\text{old}}\right)}{4}$$

Again, similar equations can be created for the other three sides. For a middle section, the new temperature will be computed by averaging the middle section with all four adjacent values. The equation for a section in the middle would be:

	$T_{(4,5)\text{old}}$	
$T_{(5,4)\text{old}}$	$T_{(5,5)\text{ new}}$	$T_{(5,6)\text{old}}$
	$T_{(6,5)\text{old}}$	

$$T_{(5,5)\text{ new}} = \frac{\left(T_{(5,5)\text{old}} + T_{(5,4)\text{ old}} + T_{(5,6)\text{old}} + T_{(4,5)\text{old}} + T_{(6,5)\text{old}}\right)}{5}$$

Using the methods described above, complete the table before for the first iterations, given the temperature gradient on the left.

Original Gradient				**Iteration 1**				**Iteration 2**		
50	45	40		45	43.75	40		42.75	41.8875	40
40	40	35		42.5	38.8	36.25		41.075	39.46	37.0125
40	34	30		38	36	33		38.83	36.45	35.083

E. Heat Map of a 2-D Temperature Gradient

Download the five .mat files from the Blackboard metasite. The four Gradient .mat files have the original temperature distribution across the plate. The Heatmap_Colormap.mat file will be used to display the changes in the temperature gradient over time.

Write a script that will:

- Ask the user to choose which Gradient .mat file to use then `load` that file. Note: each file has a different temperature gradient but all four .mat files use the same variable name for the temperature gradient: `Plate_Temp_Gradient`.
- Determine the range (difference between the maximum value in Plate_Temp_Gradient and the minimum value in Plate_Temp_Gradient) and create a variable called `scaling` equal to 255/range.
- Create a variable called `offset` and set it equal to the value you would need to add to the Plate_Temp_Gradient to make the minimum value in the array equal to 0. *Note: the scaling and offset variables will be useful in properly displaying the changes in the temperature gradient over time.*
- Load the Heatmap_Colormap.mat file.
- Create two new 2-d arrays exactly the same size as Plate_Temp_Gradient. The first array will hold the old temperature data and should initially be set equal to Plate_Temp_Gradient. The second array will hold the updated temperature data and can be initialized however you would like prior to the loop – all zeros works well.
- The script should then implement the process described in Part D. The process should continue until equilibrium has been reached; that is, when the range in temperatures across the plate is less than 0.5 ^oF. It is recommended that you use a pause of 0.05 seconds for the animation.
- To display the temperature changes over time, add the following commands after you update the temperature data:

```
image(scaling*(New_Array + offset));
colormap(heatmap)
```

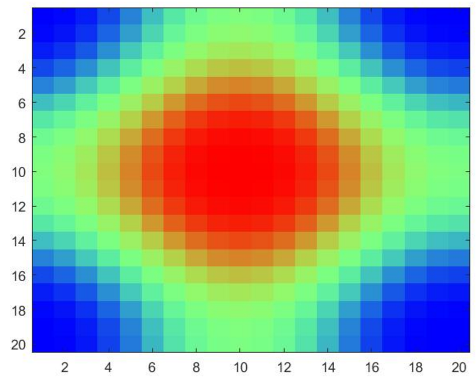
  Note: New_Array is the variable name for the array with the updated temperatures. Change it to whatever you called that array in your code. The variables scaling and offset were determined based on the original temperature data and should not be changed.

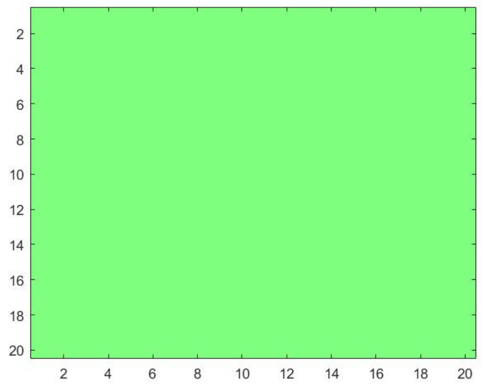
- Once equilibrium is reached, your script should display at the command prompt the number of iterations it took to reach equilibrium as well as the equilibrium temperature (the average temperature of your final temperature gradient).
- Your script should also plot the original and final distribution of temperatures along the plate. You will need to do these in separate figure windows – either use the `subplot` command or use the `figure` command to accomplish this.

Once you have your script completed, use it to complete the following table and include the plots of the original and final temperature distributions.

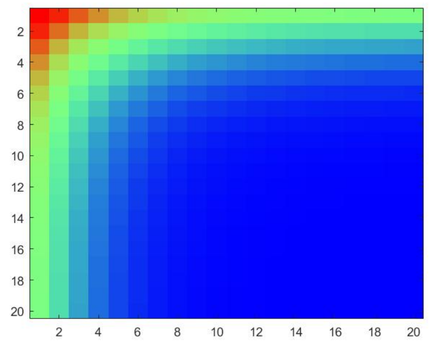
Data File	**Number of Iterations**	**Equilibrium Temperature (°F)**
Gradient_1.mat	760	2. 03
Gradient_2.mat	1003	34. 06
Gradient_3.mat	789	64. 63
Gradient_4.mat	4018	68. 69

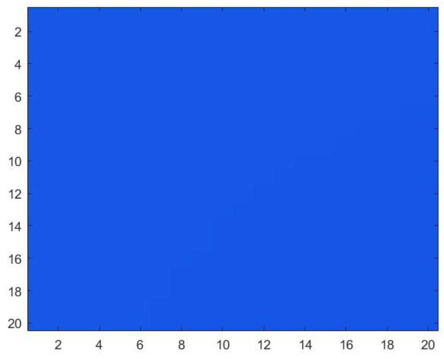
MATLAB Plots of Initial and Final Temperature Distributions for Distribution_1.mat:



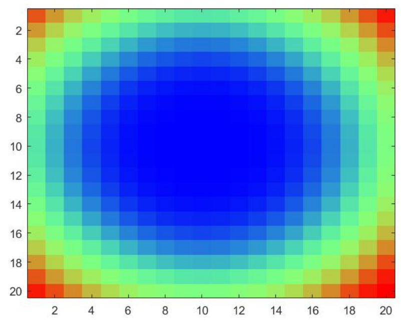


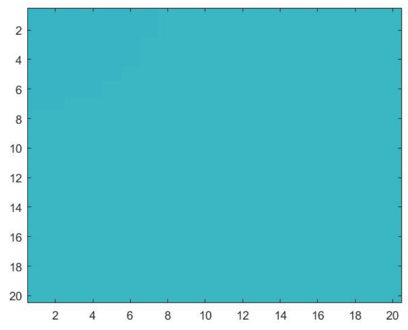
MATLAB Plots of Initial and Final Temperature Distributions for Distribution_2.mat:



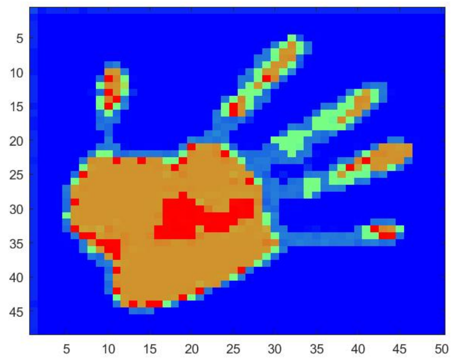


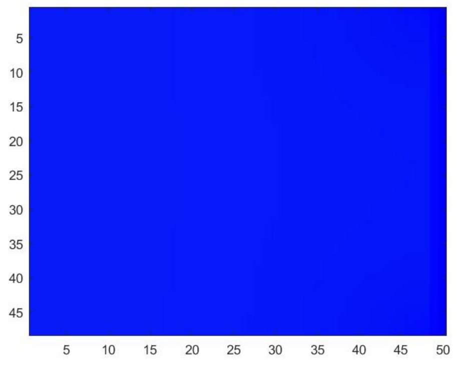
MATLAB Plots of Initial and Final Temperature Distributions for Distribution_3.mat:





MATLAB Plots of Initial and Final Temperature Distributions for Distribution_4.mat:





PASTE YOUR SCRIPT FILE HERE:

```
%% LAB4
% Name: Horace
% Date: 26 Feb 2019

%% Code
%clear processor
clear; close all; clc; commandwindow;
pick = menu('Choose','Gradient 1','Gradient 2','Gradient
3','Gradient 4');
switch pick
    case 1
        load Gradient_1;
    case 2
        load Gradient_2;
    case 3
        load Gradient_3;
    case 4
        load Gradient_4;
end
```

```
load Heatmap_Colormap;

scaling = 255/(max(max(Plate_Temp_Gradient)) -
min(min(Plate_Temp_Gradient)));

offset = -min(min(Plate_Temp_Gradient));
Old_Array = Plate_Temp_Gradient;
New_Array = zeros(size(Plate_Temp_Gradient));
[rows,cols] = size(Plate_Temp_Gradient);
count = 0;

while max(max(Old_Array)) - min(min(Old_Array)) > 0.5
    count = count + 1;
    for r = 1:rows
        for c = 1:cols
            if r == 1 && c == 1 % Top Left Corner
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r,c+1)+Old_Array(r+1,c))/3;

            elseif r == rows && c == 1 %Bottom Left Corner
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r-1,c) + Old_Array(r,c+1))/3;

            elseif r == 1 && c == cols  % Top Right Corner
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r,c-1) + Old_Array(r+1,c))/3;

            elseif r == rows && c == cols % Bottom Right
Corner
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r, c-1) + Old_Array(r-1,c))/3;

            elseif r == 1   % Top Row but not in corner
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r,c1)+Old_Array(r,c+1)+Old_Array(
r+1,c))/4;

            elseif r == rows % Bottom Row but not in corn
er
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r,c1)+Old_Array(r,c+1)+Old_Array(
r-1,c))/4;

            elseif c == 1   % First Column
```

```
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r1,c)+Old_Array(r+1,c)+Old_Array(
r,c+1))/4;

            elseif c == cols  % Last Column
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r1,c)+Old_Array(r+1,c)+Old_Array(
r,c-1))/4;

            else  % Middle Section
                New_Array(r,c) =
(Old_Array(r,c)+Old_Array(r,c1)+Old_Array(r,c+1)+Old_Array(
r-1,c)+Old_Array(r+1,c))/5;                     end
end     end

    image(scaling*(New_Array + offset));
    colormap(heatmap)
    pause(0.05)
    Old_Array = New_Array;
end
fprintf('The number of iterations is %i \n',count);
fprintf('The mean equilibrium temperature is %0.2f \n',
mean(mean(New_Array)));
subplot(1,2,1);
image(scaling*(Plate_Temp_Gradient+offset));
colormap(heatmap);
title('Original Temperature Gradient');
subplot(1,2,2); image(scaling*(New_Array+offset));
colormap(heatmap);
title('Final Temperature Gradient');
```

F. To be turned in:

This word document with all tables and figures included and the m-file for your script.