Lab 3: Temperature Gradient Along a 1-D Rod

A. Background Information

Have you ever cooked on a stove with a pan with a metal handle? If you have, you may have noticed that when you start cooking, the handle is cool and everything is great. However, as you leave the pan on the stove, the handle gradually warms up. If you leave the pan on the stove for long enough, eventually the handle will be as hot as the stove itself!

This is a phenomenon known as heat transfer. Just like water likes to flow from areas of higher elevation (and higher potential energy) to areas of lower elevation (and lower potential energy), heat energy tries to travel from areas of high temperature (higher energy) to areas of lower temperature (lower energy). In addition, heat energy attempts to equalize the temperature along an entire medium. This is why the handle of your pan will eventually reach the same temperature as the stove: as more heat energy is added to the pan that energy transfers throughout the pan in an attempt to equalize the temperature throughout.

For our purposes, we are going to investigate the temperature gradient along a 1-D rod where we are not continually adding heat to the system. As you know, if there was a real rod we were working with, there would be an infinite number of points along the length of the rod and thus an infinite number of temperature values to deal with. As we know from our DAQ experiments, the computer cannot handle an infinite number of points. Instead, we need to break the length of the rod into a finite set of sections to estimate how the temperature changes along the length of the rod. We simply assume that the temperature in each section is the average temperature in that region.

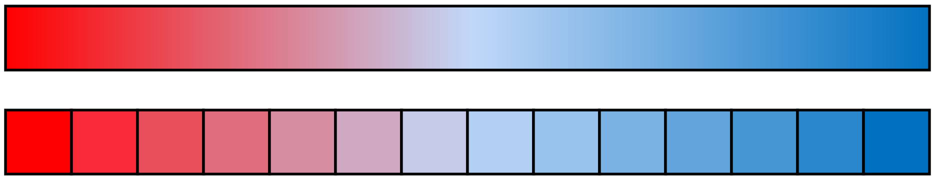


Figure 1: Continuous Temperature Gradient (top) versus Discrete Temperature Gradient (bottom)

If we are going to model this situation in MATLAB, we need a structure that is similar. For this, we will use a 1-D array. Each element in the array will represent one of the sections along the length of the 1-D rod and the value of the array will represent the temperature of the rod for that section. This is a common practice in engineering when performing simulations of systems. A more complicated version of this same basic procedure is called Finite Element Analysis (FEA), in which a complicated object is broken down into a number of smaller sections (usually a grid). The interaction between each smaller section is known and can be easily calculated. The object can then be run through simulations by performing calculations based on the smaller sections rather than trying to perform a complex calculation on the overall object. This technique is used in a wide variety of application areas, from simulating automotive crashes, stress and strain forces on buildings and structural members, and predicting electromagnetic radiation patterns.

B. Understand the Process

In order to model this process, we will assume that as time progresses, the temperature in a given section changes as the average of the surrounding sections. There are two distinct regions of the rod that we need to pay attention to, which will require us to calculate the new temperature in different ways:

- A middle section
- An end section

For a middle section, we will simply take the average of the temperatures of the sections on either side and the middle section:

$$T_{n_{new}} = \frac{T_{n+1_{old}} + T_{n_{old}} + T_{n-1_{old}}}{3}$$

where n is the location of the current section

For an end section, we will take the average of the temperatures in the end section and the neighboring section:

$$T_{n_{new}} = \frac{T_{n_{old}} + T_{n-1_{old}}}{2} \qquad (1)$$

$$T_{n_{new}} = \frac{T_{n_{old}} + T_{n+1_{old}}}{2} \qquad (2)$$

where n is the location of the current section, equation (1) corresponds to the right side of the rod, and equation (2) corresponds to the left side of the rod.

Every time the temperatures of the rod are updated within our simulation, we will need to use previous temperature values for all sections. To see how this works, complete the table below for the relatively simple temperature distribution.

	Section 1	**Section 2**	**Section 3**	**Section 4**	**Section 5**
Original Distribution	82 ^{o}F	78 ^{o}F	76 ^{o}F	71 ^{o}F	67 ^{o}F
First Update	80 ^{o}F	78.67 ^{o}F	75 ^{o}F	71.33 ^{o}F	69 ^{o}F
Second Update	79.335 ^{o}F	77.89 ^{o}F	75 ^{o}F	71.78 ^{o}F	70.165 ^{o}F

C. Animation in MATLAB

One useful skill to develop when creating scripts in MATLAB is the ability to animate graphs or other graphics. The process is relatively simple, and is something you actually use every day. In fact, you're using it right now!

Whenever you view an animated image on a display, be it a television, computer screen, or in a movie theater, you are actually viewing a series of slightly different still images. By showing this set of still images at a quick enough rate, your eye tricks you into seeing an animated scene. This is the same principle used for hand drawn cartoons or those flip books you used to create as a kid (or maybe still do!). The refresh rate you see on the specifications for your television or computer monitor refers to how quickly the image is updated on the screen.

In MATLAB, we can create animations by doing the same thing: we display a graphic, make a change to it, display it again, and so on. If we can do so at a quick enough rate, it will appear that our graphic is moving.

Copy and paste the following code into a script and run it.

```
for k = 0:pi/50:2*pi
    t = 0:pi/100:2*pi;          % calculate time values
    y = sin(t+k);               % calculate y values
    plot(t,y);                  % plot y values
    axis([0 2*pi -1 1]);        % set axis
    pause(0.05);                % pause for animation
end
```

Notice that as the script runs, it looks like the sine wave is moving across the figure window. How does this code work? If we look at the code, there are three main parts: the for loop, the computations and plotting, and the pause statement. The computations and plotting are pretty straight forward: we create a t vector, use that vector to compute some values (in this case, a sine wave), and plot the data.

The for loop repeats this process 101 times and also changes the phase of the sinusoidal wave, based on the value of k. This is what creates the animation. Each time the loop repeats and the data is plotted, the y values will have been calculated for a slightly different set of input values.

The last component, the pause statement, is important for two reasons. First, it allows MATLAB enough time to create and display the graph before moving on to the next iteration of the loop. Without the pause statement, MATLAB doesn't have enough time to display the graph before the next graph is ready to be displayed, so the animation will not occur. The other role the pause statement plays is to adjust the speed of the animation.

Try changing the value in the pause statement and re-running the code. What happens when you increase the value? What happens when you decrease the value?

The sine wave moves more slowly as I increase the value.
The sine wave moves faster as I decrease the value.

D. Implement the Process

Write a script in MATLAB that implements the process above. Your script will start with an initial temperature distribution and will continue to update the temperatures along the length of the rod until you reach equilibrium. For our case, we will assume that equilibrium has been reached once the range in temperatures along the rod is less than 0.5 ^{o}F.

Your script will need to do the following:

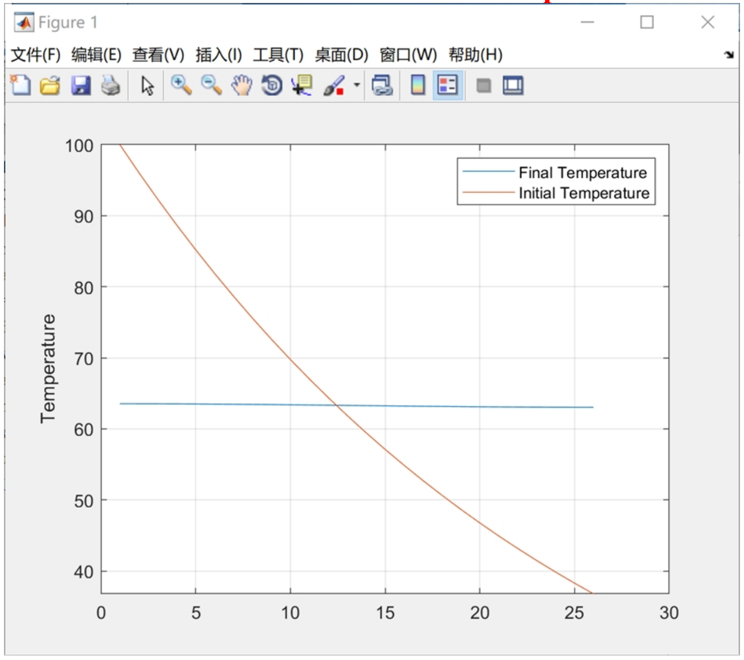
- Load a .mat file containing an initial temperature gradient
    - There are several files available on Canopy that you will need to download (Distribution_1.mat, Distribution_2.mat, Distribution_3.mat, and Distribution_4.mat)
    - In each file, the name of the vector containing the initial temperature gradient of the rod is called Rod_Temp_Gradient
- Create two new vectors exactly the same size as Rod_Temp_Gradient. The first vector will hold the old temperature data and should initially be set equal to Rod_Temp_Gradient. The second vector will hold the updated temperature data and can be initialized however you would like prior to the loop – all zeros works well.
- Use a nested loop structure to implement the process of equalizing the temperature along the rod
    - You will need to use one inner loop to update the temperatures along the rod using the averaging algorithm described in Part B.
    - You will need to use one outer loop to determine whether you have reached equilibrium (when the difference between the maximum temperature and the minimum temperature in the rod is less than 0.5^{o}F).
      Note: Remember that once you start a new iteration of temperature updating, your new values from the previous iteration become your old values for the current iteration!
- Create an animation of the temperature distribution by plotting the distribution after each update
    - The plot and pause statement should be placed inside your outer loop and outside of your inner loop. **Note: use pause(0.001) otherwise some of the distributions will take a long time to run.**
    - You will also need a ylim command so that your window stays at the same size. Set your y-axis so that the lower end stays at your initial minimum temperature and the upper end stays at your initial maximum temperature:

      ```
      ylim([min(Rod_Temp_Gradient) max(Rod_Temp_Gradient)]);
      ```
- 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 vector). MATLAB has a function called `mean` to compute an average.
- Your script should also plot the original and final distribution of temperatures along the length of the rod

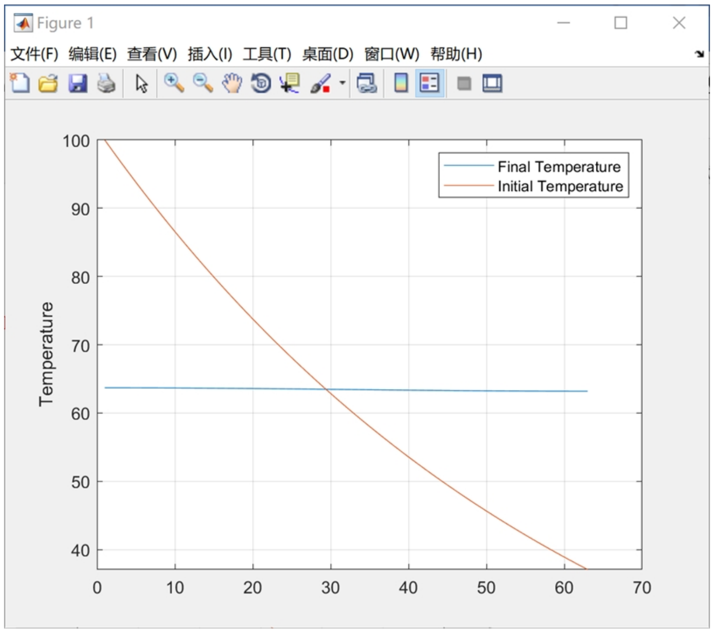
Once you have your script completed, use it to complete the following table and answer the questions.

Data File	Number of Iterations	Equilibrium Temperature (ºF)
Distribution_1.mat	902	63.29
Distribution_2.mat	5448	63.45
Distribution_3.mat	2421	0.08
Distribution_4.mat	3168	45.00

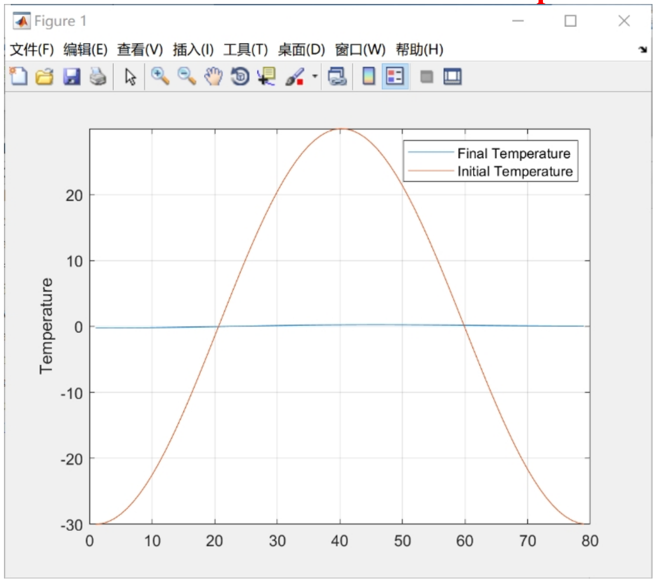
MATLAB Plot of Initial and Final Temperature Distributions for Distribution_1.mat:



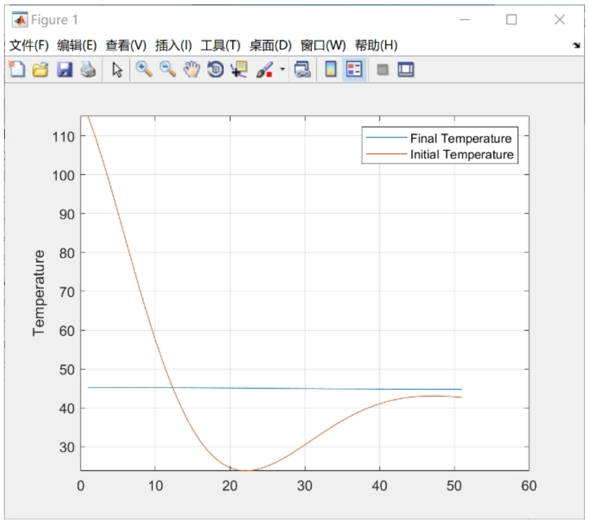
MATLAB Plot of Initial and Final Temperature Distributions for Distribution_2.mat:



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



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



PASTE YOUR SCRIPT FILE HERE:

```
%% LAB3
% Name: Horace
% Date: 26 Feb 2019

%% Code
%clear processor
clc;clear;close all;
load Distribution_4.mat
n = length(Rod_Temp_Gradient);
old = Rod_Temp_Gradient;
new = zeros(1,n);
x_axis = 1:1:n;
min_initial = min(Rod_Temp_Gradient);
max_initial = max(Rod_Temp_Gradient);
count = 0;
while max(old)-min(old) >= 0.5
    for i = 1:n
        if i == 1
            new(i) = (old(i)+old(i+1))/2;
        elseif  (i>=2)&&(i<=n-1)
            new(i) = (old(i-1)+old(i)+old(i+1))/3;
        else
            new(i) = (old(i)+old(i-1))/2;
        end
    end
    count = count+1;
    old = new;
    plot(x_axis,old)
    ylim([min_initial max_initial]);
    grid on
    pause(0.001)
end
hold on
plot(x_axis,Rod_Temp_Gradient)
ylabel('Temperature')
legend('Final Temperature','Initial Temperature')
fprintf(['number of iterations = %i\n'...
         'average of final temperature =
%0.2f\n'],count,mean(old))
```

If you look at the initial temperature distribution, where does the final temperature distribution end up? What shape is it?

The final temperature distribution end up at the average temperature. The shape is a line.

Distribution_1.mat and Distribution_2.mat contain the same distribution of temperatures along the rod, but Distribution_2.mat uses more sections. How does the number of sections used affect the number of iterations?

The number of iterations increase as the number of sections used increase.

Again looking at the results for Distribution_1.mat and Distribution_2.mat, how does the number of sections used affect the final equilibrium temperature?

The final equilibrium temperature can be more accurate as the number of section used increase.

E. To be turned in:

- You will need to upload this word document with all tables, questions, and figures included and the m-file for your script.