Introduction to Computers for Engineers:

Recitation #7

Learning Objectives

- Understand how to load data from external files (e.g. .csv files)
- Continue to develop an understanding of loops
- Continue to develop an understanding of arrays
- Continue to develop an understanding of MATLAB's plot functionality

Quiz 4:

- Question:

Please write a function that takes two numerical arrays, in1 and in2, as inputs. in1 and in2 both have the same length, n.

Your function should return one output, out1. out1 is also an array of length n.

Each element of out1 should be the absolute value of the sum of the elements at the same index of in1 and in2. Recall that the abs function will return the absolute value of a number. Recall that the length command can be used to find the number of elements in an array. **Please use a loop to accomplish this.**

- Most seen (incorrect) solution:

```
function [out1] = array(in1, in2)
n= length(in1) ;
for i = 1:n
    out1(i) = abs(in1(i) +in2(i)) ;
end
end
```

We will be working with data!

- The data that we will be working with is the **average sunlight in Newark, NJ for a given month**
  - The measurement of this data is the **Global Horizonal Irradiance**
  - GHI is the total amount of shortwave radiation received from above by a surface horizontal to the ground
  - GHI's units are Watts/Meter Squared
- For the years 2014 through 2017
- The data is in 4 files (one for each year)
- Months are sequential
  - Index 1 is January, 12 is December

Activity 1: Loading in data

- **Download each of the files for today from the Recitations/Recitation 7 folder on Canvas**
  - Be sure to download them **into the same folder as your MATLAB code.**
  - Be sure that the file extension does not get modified in the download process. It should be .csv
  - Sometimes Safari likes to do weird things to .csv files. Please let us know if you are using Safari and need extra help
  - If you use MATLAB Online, you will need to upload the files to use them
- Use csvread to load the data into MATLAB
  - Example: `data1 = csvread('sunData2014.csv');`
- **In your groups, discuss:**
  - What is the data type of the data?
  - **What important information must be understood by the programmer?**

Activity 2: Working with data

- We want to compute an array containing the **average sunlight in a given month for the years 2014 to 2017 using a for-loop.**
- **Write a script that calculates an array with the average sunlight in each month over the 4 year period**
  - The resulting array should be 1x12
  - Index 1 should be the average for all Januarys, and index 12 the average for all Decembers
- Your script should do the following:
  - Initialize an array that will hold the averages values (the resulting array)
  - Contain a loop that counts through indices
  - Index each array using the loops counting variable
  - Fill in your initialized array
- **In your group, discuss:**
  - In what other useful ways could we process this data?
  - How would we modify our script to perform that data processing?

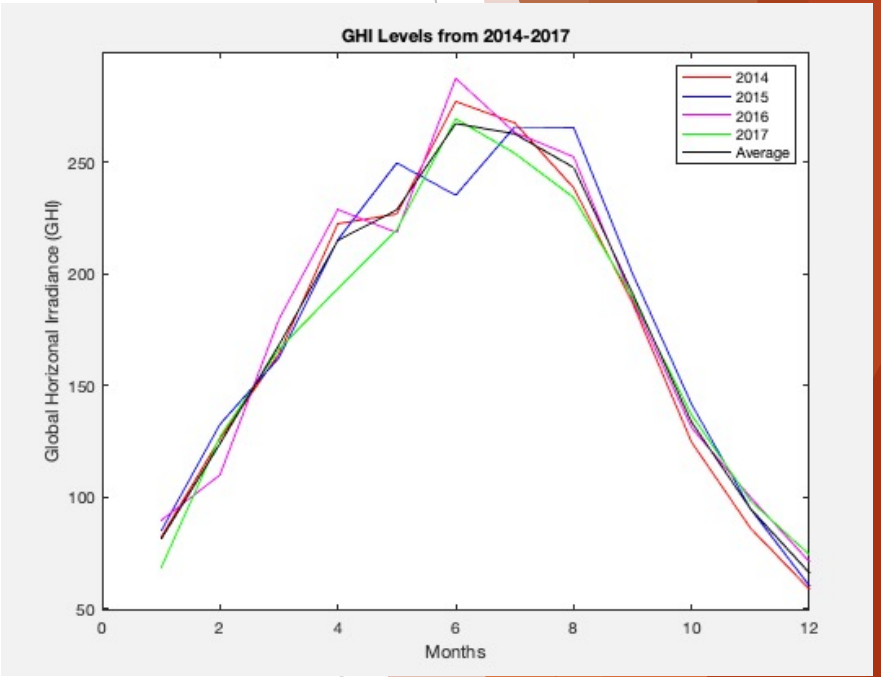
Activity 2: Solution

```
% create array of zeros
average_array = zeros(1, length(data_2014));

% looping through each month
for i=1:length(average_array)
    month_avg = (data_2014(i) + data_2015(i) + data_2016(i) + data_2017(i)) / 4;
    average_array(i) = month_avg;

end
```

Activity 3: Plotting data in MATLAB

- We want to **create a plot to compare our averaged array to the data from each year** to see how much variation there is
- On the same set of axes, plot the 5 different lines using the `hold on` command
  - Example:

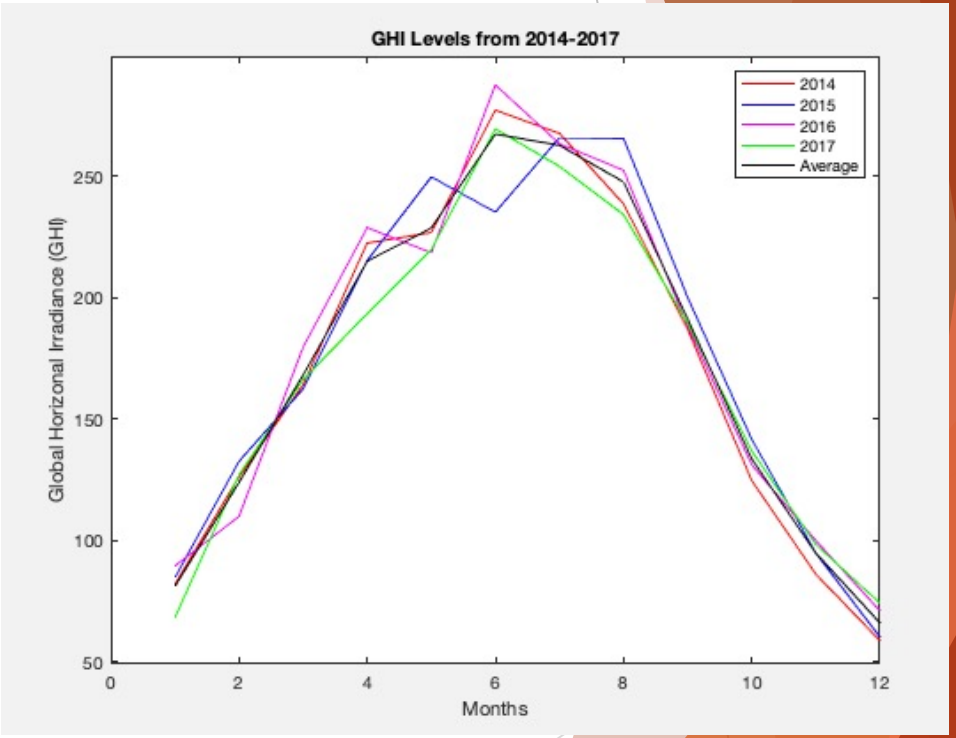
```
% plotting
plot(x_axis, data_2014, 'r')

hold on
plot(x_axis, data_2015, 'b')
```



- Your plot should look like figure on the right:
- We can label and put legends using the commands:
  - title – use the command title, as in title('This is a title')
  - axes labels – use the commands xlabel and ylabel, as in xlabel('month')
  - legend – use the command legend, as in legend('Plot1', 'Plot2'….)

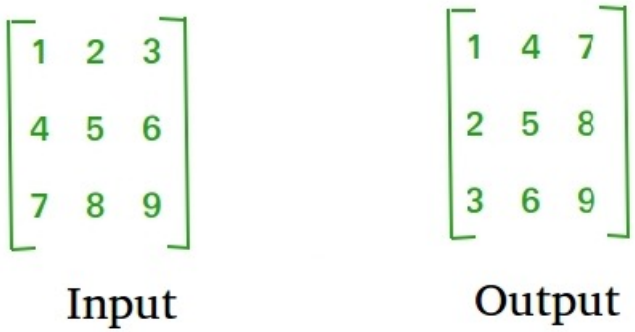
Activity 3: Solution

```
x_axis = 1:12;

% plotting
plot(x_axis, data_2014, 'r')

hold on
plot(x_axis, data_2015, 'b')

hold on
plot(x_axis, data_2016, 'm')

hold on
plot(x_axis, data_2017, 'g')

hold on
plot(x_axis, average_array, 'k')
xlabel('Months')
ylabel('Global Horizonal Irradiance (GHI)')
title('GHI Levels from 2014-2017')
legend('2014', '2015', '2016', '2017', 'Average')
```



Activity 4: More nested for-loops

- We want to write a function called matrixTranspose, with the following:
  - Input: my_matrix (2D array)
  - Output: transposed_matrix (2D array)
- The purpose of this function is to transpose your input matrix, which is doing the following:



- We will only be working with **symmetric matrices** (e.g. 3x3 matrices). For symmetric matrices, we can just put the (i,j)-th element in the (j, i)-th place.
- You want to initialize your output matrix as a matrix of zeros (same size as your input) and fill them in **using a nested for-loop**.

Activity 4: Solution

```
function [transposed_matrix] = matrixTranspose(my_matrix)

[rows, cols] = size(my_matrix);
transposed_matrix = zeros( size(my_matrix) );

for i=1:rows
   for j=1:cols
   transposed_matrix(j, i) = my_matrix(i, j);

   end

end


end
```

```
input_matrix =

     1     2     3
     4     5     6
     7     8     9

>> output_matrix = matrixTranspose(input_matrix)

output_matrix =

     1     4     7
     2     5     8
     3     6     9
```

Activity 5: Even more nested for-loops

- We want to write a function called **maxElement**, with the following:
  - Input: my_matrix (2D array)
  - Output: max_val (double)
- The purpose of this function is get the maximum value of your input matrix.
- You want to initialize your maximum value (max_val) as the first element of your input matrix and then **using a nested for-loop,** replace that value with an if-statement.
- Test your function using test cases.