CIEG 675
LAB #3 Due **Tuesday January 26, 2021**

*Please make your lab as a word document now.* Include the MATLAB code for each problem and if the output is a graph, save it to a file (`print`), and insert the figure along with the code used to generate it into the document with some description of the figure(s). If the problem asks you to create a function, paste the entire function into the document for that problem as well. I ask this because from now on you will often be making functions and thus turning in the output from one script does not make sense anymore. However, if you are still working with a script for some problems separate the problems into sections using (%%). Print out the word document and turn in. SUPPRESS output where not needed to save "paper", if using publishing for part of what you are handing in.

In a script (m-file), do the following and verify it works by running your script.

1) Write a function that will take an arbitrary, but smooth, vector of data and locate ALL the local maxima and minima in the vector (call it **`min_max`**, as "`minmax`" is a built-in function in newer MATLAB versions starting with R2014b). It should return, as output, the indices of the input vector where these maxima and minima occur. Test your function on the following data set that you should make quite smooth by using small increments in $x$, where $x$ should go from 0 to 10:

$$y = x^{1.01} + 4\cos(3\pi x/4) - 2\sin(2\pi x/3) - 0.25$$

In the same figure, plot $y$ vs $x$, with the maxima and minima (computed by your function, **`min_max`**) on top of the curve as different symbols (remember you can use `>> help plot` to see a list of available line/marker specifiers). DO NOT USE a built in function to find local minima or maxima. I want you to write the code to do it.

2) Create a 1 x 2 structure array with 3 fields of your choosing. The only stipulation is that the 1st field must be a string (class: char), the 2nd field must be a cell array (class: cell), and the 3rd field should be an array of numeric values (class: double).

3) Create a 12 x 1 cell array whose contents are the twelve months of the year, in order, starting with `'January'`.

4) Write a function called **`day2sec`** that takes an input vector of MATLAB dates and converts it first to time relative to the 1st date value in the input vector (i.e. the first value in the output vector should be zero) and then from days to seconds. The output vector should be the number of seconds that have elapsed since the 1st value in the input vector.

5) Download the file from Canvas called "*AtlanticCity_TemperatureData.csv*". Load the data into MATLAB using `dlmread` (or some other function). The 1st column contains dates in MATLAB time. The 2nd column is air temperature and the 3rd column is water temperature, both in degrees Celsius. The data span all of 2015. Subtract the first date value from all other dates, then plot air temperature and water temperature versus time, on the same axes. Add a legend, labels, etc. Finally, change the `'xticklabel'` values of the current axes to be your cell array from Problem 3, where the `'xtick'` locations should be the number of days after January 1st, for each first day of the month (i.e. [1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335]). Resize the figure to be wider, so that all the month labels on the *x*-axis are not overlapping.

6) Another way to load files into MATLAB without knowing/specifying the format of data in advance is using a `while` loop, with `feof` and `fgetl`, as follows:

```
ct = 0;  % a counter
fid = fopen('somefile.txt', 'r'); % open file for reading
% Keep running a loop until EndOfFile
%   (feof(fid) = 1 if end of the file is reached)
while feof(fid) ~= 1
    ct = ct + 1; % increase counter by 1
    data{ct} = fgetl(fid); % grab entire line of text from file,
                             store as string in the ct^th cell
end
fclose(fid); % close file
%
% ct should be equal to the number of lines of text in the file
```

Download the file on Canvas called "*surprise.txt*", which contains multiple lines of text, each with a different number of characters. First, using the method outlined above, load each line of the file into a cell array in MATLAB. Then, write a code (it shouldn't be long…mine is only 7 lines) that will print each line of text into the command window (fprintf), **one character at a time**, and after each full line has been printed, it should then print the newline character combination in MATLAB, `fprintf('\n')`, in order to jump to the next line.

7) Develop a data set that is 10,000 points long composed of random numbers from a normal distribution. Make a histogram of this data set and then overlay a Gaussian curve on top. You will have to determine how to normalize your plot or histogram to get the y-axis to scale appropriately.

8) Perform a power spectral density (PSD) calculation using pwelch

[Pxx,F] =PWELCH(X,WINDOW,NOVERLAP,NFFT,Fs)

from the data you will generate as follows: Make a time vector, t, out to 10000 with spacing of 0.01.

$$y = 4\cos(2\pi t) + \sin(2\pi t/0.2) + 0.01 randn(1, length(t))$$

Mess with the different parameters in pwelch to see what they do to change the resulting PSD. We talked briefly about them in class. WARNING: DO NOT change more than one parameter at a time. WARNING: you cannot change Fs. That is the frequency of your input time series and is 100 Hz for the time vector I asked you to create.

To see data from pwelch, you normally plot on a semilogy plot with Pxx as a function of F (frequency). Tell me what the plot tells you with respect to the given function y. This will involve several plots with accompanying text to **describe** what you see. I can envision a plot with several curves where you have modified just one of the parameters. Then another plot with several curves where you have modified a different parameter.