Homework Assignment #1 (ME-190, Fall 2016)

Due date: Thursday, Sept. 8, 2016, 2:00 pm.

Problem 1- Develop a script (and a function) for sine signal identification.

Applications: Experimental data analysis; development of frequency response data.

Description: The Excel file "SineWaveData.xlsx" contains information of a noisy sine signal with low number of samples per cycle. The first column is the time vector in [seconds], and the second column contains the measured signal information in [Volts]. The frequency of the signal is **10 Hz**, and is given. However, the magnitude and the phase of the signal are unknown. We would like to develop a simple code that enables us to find the magnitude and phase of the signal with a reasonable accuracy.

1. Create a function file with:

   Input (Argument) x, where x(1) represents the magnitude and x(2) represents the phase in radians.

   Output: Cumulative quadratic error between the measured data and simulated sine wave as follows:

$$e = \sum_{k=1}^{N} (y_{m,k} - y_k)^2$$

   Where *N* is the number of samples (elements of measured signal), y_m is the measured signal, and

$$y_k = x(1) * \sin(2\pi f \; t_{m,k} + x(2))$$

   Where *f* is the frequency of the signal in Hz (10 Hz in this case), *y* is the simulated sine signal, and t_m is the time data associated with the *measured* signal (the first column of the Excel file).

   To access the measured time and signal data inside the function, you need to either read the data inside the function directly from the Excel file, or use MATLAB's "evalin" command and read data from workspace. In the latter case, the data must be first read in the main scrip which will call the function.

   In the main script we read data from Excel:

   ExpData = xlsread(filename);

   In the function file we access ExpData using "evalin":

   function CumError = CalcCumError(x)
   evalin … Access ExpData here
   CumError = … Calculate function output.

This function must return smallest error value when x(1) and x(2) match the measured data magnitude and phase. However, we don't know these values. We can perhaps guess a range for these values and an initial guestimate.

Let's assume x(1) ∈ [5 10] and x(2) ∈ [0 π/2].

2. Create nested for loops which vary x(1) and x(2) within their given ranges with 100 linearly spaced increments:

```
x1 = 5:dx1:10;  %you need to find dx1 and dx2
x2 = 0:dx2:pi/2;
for i = ? % need to replace ? with an appropriate range
        for j = ?
                CumError(i,j) =  CalcCumError([x1(i) x2(j)]);
        end
end
```

   Write a piece of code that finds the minimum value of CumError and corresponding x1 and x2 values (x1_opt and x2_opt). Use the "disp" and "num2str" commands to display the results in the command window and published report. For example:

```
disp(['Estimated amplitude is: '  num2str(x1_opt) '[V]'])
disp …
```

   Now we would like to sketch a 3-D plot using "surf" command:

```
figure(1); surf(x1,x2,CumError)
```

   This must give you a plot showing the cumulative quadratic error between the sine wave model and measured signal, as a function of x1 and x2.

   Now, let's plot the estimated sine function on top of the measured data to see if they match. To create the estimated sine signal, we can define a new time vector, t1, with finder time intervals, e.g., 1ms:

```
t_sim = [0:0.001:(ExpData(end,1))];
y_sim = x1_opt*sin(2*pi*10*t_sim + x2_opt);

plot simulated and experimental data on top of each other here.
```

3. We would like to repeat the same System ID process, except that we would like to use an optimization function instead of using incremental "for" loops.

For this, we need to use an optimization function from MATLAB's optimization toolbox. One of the best optimization algorithms for unconstraint parameter ID is the Nelder-Mead simplex search algorithm, which requires an initial estimate for the parameters. Let's use the middle of the range as the initial values:

x1_initial = 7.5;
x2_initial = pi/4.

Then we can use "fminsearch" command which represents the Nelder-Mead algorithm:

[x_opt f_opt] = fminsearch('CalcCumError',[x1_initial x2_initial]);

Now, you can compare the resulted x_opt values and the corresponding function value. Use "disp" command to display results on the command window for publishing the data. How do the optimized x values compare to the ones you obtained from the nested for loops?

4. The optimization function, fminsearch, evaluates the function many times to arrive at the optimal solution, but it doesn't show the trace of x values and the corresponding function values. We can trace the optimization from the initial to the final state and plot the function value over optimization iteration to see how it is minimized. For this we can use "evalin" and "assignin" commands inside the function:

```
function CumError = CalcCumError(x)
…
CumError  = …

% Reading trace of CumError from workspace and updating back to workspace
CumErrorTrace = evalin('base','CumErrorTrace');
CumErrorTrace = [CumErrorTrace CumError];
assignin('base','CumErrorTrace',CumErrorTrace);
```

This will keep adding the new CumError value to the existing CumErrorTrace vector every time the function is called by fminsearch. We will only need to initialize CumErrorTrace by an empty element inside the main script (outside the function):

CumErrorTrace = [];
fminsearch(…

Now, we can plot and see the optimization result:
Figure(2); plot(CumErrorTrace);

How many function evaluations did it take to arrive to a good error value? How does this compare with the nested for loops?

Use tic toc commands to see how much each process takes.

tic
Nested for loops code
toc

tic
Nelder-mead optimization code
toc

Document your codes and results to the above three problem sections, and turn in the hard copy including your scripts, plots, results, and conclusions at the due time. Make sure your plots include representative labels and titles. Use Matlab's "Publish" feature to create the report. Include function code as an Appendix at the end of the report.