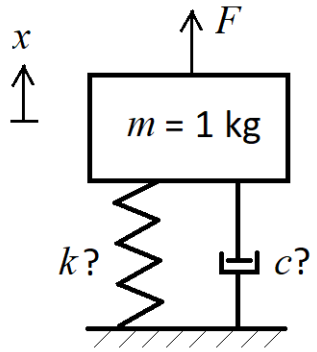
Homework Assignment #1 (ME-190, Fall 2017)

Due date: Thursday, Sept. 7, 2017, 1:30 pm.

Objective: Develop MATLAB codes for the identification of a mass-spring-damper system's parameters.

Applications: Experimental data analysis; System Identification.

Description: The below figure shows an underdamped mass-spring-damper system with a known mass of 1 kg, but unknown damping coefficient, *c*, and unknown spring constant, *k*. A unit impulse force is applied to the system at *t* = 0, and the response of the system is measured. It is desirable to identify the values of *c* and *k* from the measured data. A mathematical model for the unit impulse response of the system is given below:

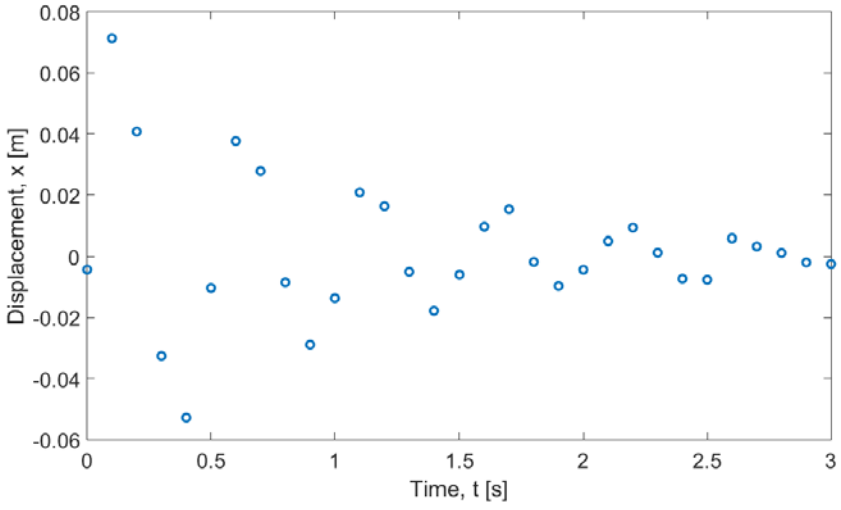


Unit impulse response ($F = \delta(t)$):

$$x(t) = \frac{1}{\omega_n\sqrt{1-\xi^2}} e^{-\xi\omega_n t}\sin(\omega_n\sqrt{1-\xi^2}\,t)$$

where $\omega_n = \sqrt{\frac{k}{m}}$, $\xi = \frac{c}{2\sqrt{km}}$

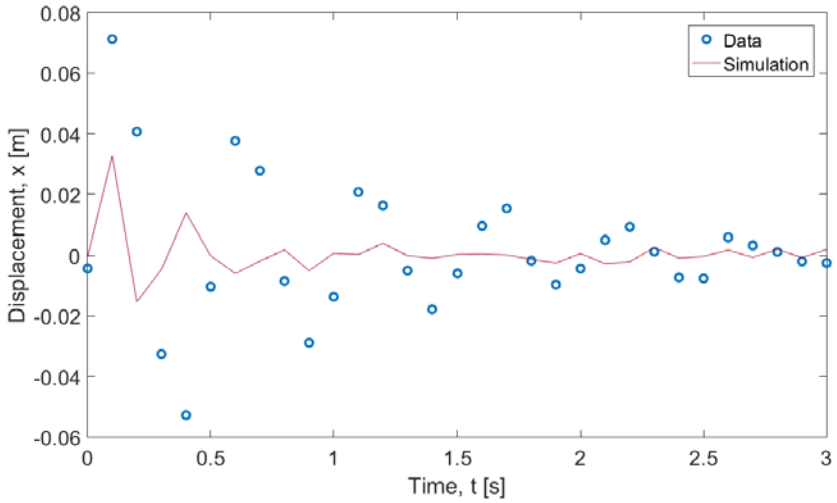
The Excel file "ImpulseResponseData.xlsx" contains time and system response information. The first column is the time vector in [seconds], the second column contains the measured output response in [m]. The figure below shows the output data vs time.



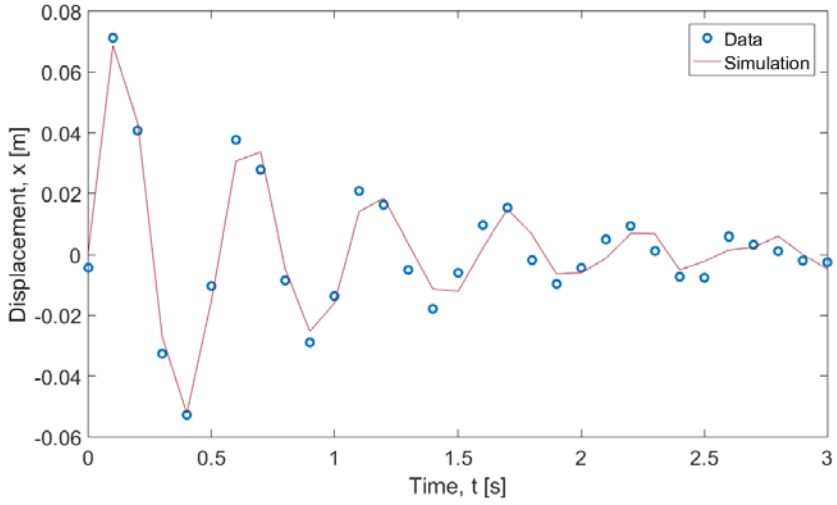
Problems: (Include all the codes/scrips and plots in your work).

Problem 1. Identification by trial and error

a. Write a script which simulates and plots the response of the system based on the provided formula, along with the data in the excel file on the same figure (see the below figure as an example). For this, you need to use the time information in the excel file. Use arbitrary values for parameters ξ and ω_n in your simulation. Make sure $0 < \xi < 1$ and $\omega_n > 0$.



b. Use trial and error for the parameter values (ξ and ω_n) to match the experimental data visually. No perfect match is expected. A reasonable match is sufficient (See figure below as an example). Provide the final values of ξ and ω_n along with your code and plot.



Problem 2. Error function development

a. Develop a function which receives ξ, ω_n, *t_data*, and *x_data* as input arguments and calculates the following error function:

$$e\_rms = \sum_{k=1}^{N} (x_{data,k} - x_{sim,k})^2$$

Where *N* is the number of samples (elements of the measured data), x_{data} is the provided output data in the excel file, and x_{sim} is the simulated output through the provided equation.

function e_rms = CalcRMSError (zeta, wn, t_data, x_data)
…
e_rms = … Calculate function output

b. Write a short script that calculates the RMS error using your developed function for the estimated values of ξ and ω_n from Problem 1.b. Report the value of the RMS error.

c. Create an exhaustive neighborhood search process for finding the values of ξ and ω_n which makes the error function output as small as possible. For this, use double "for" loops, one for ξ and one for ω_n. Set the range to be ±20% of the values you guessed in Problem 1.b:

$\xi \in \xi_{guess} \times$ [0.8:0.01:1.2]

$\omega_n \in \omega_{n,guess} \times$ [0.8:0.01:1.2].

```
zeta_vec = zeta_guess*[0.8:0.01:1.2];
wn_vec = wn_guess* [0.8:0.01:1.2];

for i = ? % need to replace ? with an appropriate range
        for j = ?
            E _Matrix(i,j) =  CalcRMSError(…);
        end
end
```

d. Write a piece of code that finds the minimum value of *E_Matrix* and the corresponding ξ and ω_n values. Use the "disp" and "num2str" commands to display the results in the command window and the published report. For example:

```
disp(['Optimal value of wn is: '  num2str(?) '[rad/s]'])
disp …
```

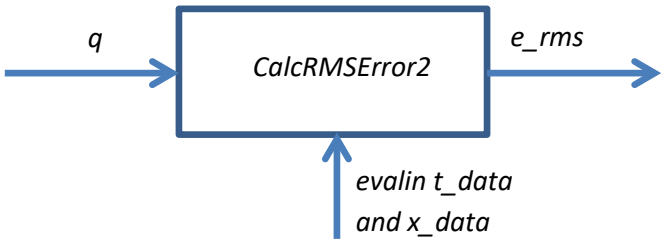
e. Sketch a 3-D plot of the error matrix as a function of *zeta_vec, wn_vec* using the "surf" command:
*figure(?); surf(*zeta_vec*,* wn_vec*,E_Matrix)*

This must give you a plot showing the RMS error between the model and the measured data, as a function of ξ and ω_n.

Problem 3. Automated optimization of the error

a. Develop a function which receives variable *q* as the only input argument*,* where $q = [\xi, \omega_n]$ and calculates the same error quantity as part *a*. For this, the function needs to access *t_data* and *x_data* using "evalin" command. The data must be placed in the workspace first.

function e_rms = CalcRMSError2 (q)
Use evalin to access t_data and x_data here
…
e_rms = … Calculate function output



b. Using the Nelder-Mean simplex search algorithm (Matlab's "*fminsearch"* command) minimize the RMS error. For the initial parameter values, use the guessed values of ξ and ω_n from Problem 1.b.

q0 = [zeta_guess, wn_guess]
[q_opt e_opt] = fminsearch('CalcRMSError2',q0);

c. Use the optimal values of ξ and ω_n to plot the simulated response of the system along with the measured data similar to Problem 1.a. Do you observe any improvement compared to the response obtained from the guessed values in Problem 1.b?

d. Plot a smoother response of the simulated output. Create a new time vector with 0.001 s time step within the same time range as the collected data, and re-compute the model output using the provided equation.

t_sim = [0:0.001:?];

compute the model (simulated) output based on the new time vector here
plot the simulated and the experimental data on top of each other here.

e. Compute the values of *c* and *k* based on the provided value of *m*, and the optimized values of ξ and ω_n. Are you satisfied with the response of the model vs the measurement samples? If so, congratulations! You have identified the unknown parameters of the systems. This is the *second* step toward developing a controller for a mechatronic system. What is the first step do you think?

Document your codes and results to the above three problem sections, and turn in the hard copy including all 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.