Control Laboratory
Experiment 6 - System Identification of a Mass-Spring-Damper System

We will investigate the effects of varying the parameters of a physical spring mass damper system, and see how its behavior is different from the lumped parameter model.

Objectives:

The objectives of this lab are to:

--Estimate mass, stiffness, damping, and static gain of a system
--Compare experimental data of system impulse response to a theoretical simulation.
--Explain some of the limitations of lumped parameter models.

Analysis:

For the data from Test 2 determine ζ and ω_n using log decrement

1) Measure the period, T, of the experimental response and determine the damped natural frequency (ω_d) knowing that the period is given by Eq. 1.

$$T = \frac{2\pi}{\omega_d} \tag{1}$$

2) Using the log decrement determine the damping ratio

$$\delta = \frac{1}{n} ln \frac{x_0}{x_n} \tag{2}$$

Knowing δ you can determine the damping ratio from Eq. 3.

$$\zeta = \frac{\delta}{\sqrt{\delta^2 + 4\pi^2}} \tag{3}$$

3) Determine the natural frequency knowing

$$\omega_d = \omega_n \sqrt{1 - \zeta^2} \tag{4}$$

After completing these calculations for Test2a, Test2b, and Test2c fill out Table 1 in the Worksheet for Experiment #6.

Determine the equivalent mass of the cart

1) Use the results in Table 1 estimate the actual cart and damper mass. You can use a procedure as follows:

   a) Generate three equations by applying the known values of damped natural frequency and damping ratio to Eq. 1

$$\omega_{di} = \sqrt{\frac{k}{m_i}}\sqrt{1-\zeta_i^2} \tag{1}$$

b) These equations can be manipulated to the following form which is linear in the unknowns (k and $m_{eq,cart}$) as shown in Eqns. 2-5.

$$\left(m_{eq,cart}\right)\omega_{d_1}^2 - k\left(1-\zeta_1^2\right) = 0 \tag{2}$$
$$\left(m_{eq,cart}+1\right)\omega_{d_2}^2 - k\left(1-\zeta_2^2\right) = 0 \tag{3}$$
$$\left(m_{eq,cart}+1.5\right)\omega_{d_3}^2 - k\left(1-\zeta_3^2\right) = 0 \tag{4}$$

Equations 2-4 can be written in matrix form as shown in Eq. 5.

$$\begin{bmatrix} \omega_{d_1}^2 & \zeta_1^2-1 \\ \omega_{d_2}^2 & \zeta_2^2-1 \\ \omega_{d_3}^2 & \zeta_3^2-1 \end{bmatrix} \begin{Bmatrix} m_{eq,cart} \\ k \end{Bmatrix} = \begin{Bmatrix} 0 \\ -1\omega_{d_1}^2 \\ -1.5\omega_{d_3}^2 \end{Bmatrix} \tag{5}$$

The estimates of mass, and stiffness can be obtained using Eq. 5 above. The way to use Matlab to solve an overdetermined set of equation is shown at the end of this document.

The damping constant c can be found by matching coefficients between the expected system model as shown in Eq. 6 and standard form of the model as shown in Eq. 7.

$$\frac{m}{k}\ddot{x} + \frac{c}{k}\dot{x} + x = \frac{1}{k}f(t) \tag{6}$$

$$\frac{1}{\omega_n^2}\ddot{x} + \frac{2\zeta}{\omega_n}\dot{x} + x = Kf(t) \tag{7}$$

This comparison results in the following relationship between m, k, and ω_n:

$$\omega_n = \sqrt{\frac{k}{m}} \tag{8}$$

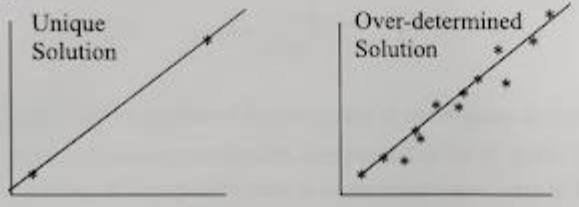
Solving for the damping constant gives

$$c_i = \frac{2\zeta k_i}{\omega_{n_i}} \tag{9}$$

Over-determined equations and Matlab:

What is an over-determined system?
Recall in many of your classes, you were taught that if you have three unknowns, you need three equations to solve them. As long as the equations are independent, you can find a unique solution. In an over-determined system you have more than data (or equations) than is needed to determine a unique solution. In fact, there may be so much data, that you may not be able to achieve a unique solution.

It's possible to define a solution for a set of equations for when there are more equations than there are unknowns. One common method which uses this type of solution is called the Least Squares Method. As you might recall, least squares is often used to find the equation of a straight line through a number of points. However, you really only need two points to give a line equation. If you measure more than two points you can still determine a line to fit the points, but it may not exactly pass through all the points. It is defined in such a way to give the best overall fit. This is usually done by trying to minimize the overall error of all the points with a line that passes through them.



In this same sense, we can find a best fit solution to a set of over-determined equations. The solution probably won't exactly fit any one of the equations, but it will give a solution that is close to all.

Implementing this type of solution in Matlab is pretty easy:
For example, let's say we have three equations for two unknown variables:

$$3x + 4y = 6$$
$$4x + 5y = 7$$
$$5x + 5y = 8$$

The following Matlab code solves this over-determined equation set, [A][X] =[B].

```
A = [3  4; 4  5; 5  5]
B = [6; 7; 8]
X = A\B
```

The solution matrix X is given by (x ,y) = (0.6471, 0.9412)

If you plug this solution back in notice the equalities are not exact, but close.

$$3*0.6471 + 4*0.9412 = 5.706 \approx 6$$
$$4*0.6471 + 5*0.9412 = 7.294 \approx 7$$
$$5*0.6471 + 5*0.9412 = 7.940 \approx 8$$

We will now use several techniques to do a full system identification for Test 2.
We will only perform a complete system identification for the run "Test 2". Assume the equivalent mass of the cart is the value determine above.

Method 1 – Log decrement method

Using the $m_{eq.cart}$ determined earlier and shown in Table 2 determine the spring stiffness, k, and viscous damping coefficient, c, for Test 2 using the frequency and damping ratio determined using the log decrement.

Method 2 – Using a performance index in Excel

Generally, analytical solutions and experimental measurements differ. The theoretical curve can be made to more closely (though usually not exactly) predict the experimental measurements by adjusting the model parameters (in this case, the mass, stiffness and damping). A measure of experimental/theoretical closeness is the *performance index J* given by

$$J = \frac{1}{n}\sum_{i=1}^{n}\left(x_i^{\text{model}} - x_i^{\text{test}}\right)^2 \tag{10}$$

where:

x_i^{model} = cart position predicted by the model at the i^{th} point in time
x_i^{data} = cart position experimentally determined at the i^{th} point in time
n = number of data points used in comparing experiment to theory

A second order system forced with an impulse is shown in Eq. 11

$$\frac{\ddot{x}}{\omega_n^2} + \frac{2\zeta}{\omega_n}\dot{x} + x = \frac{F}{k}\delta(t) \tag{11}$$

With zero initial conditions Eq. 11 has a solution given by Eq. 12 (see page 314 in the Text)

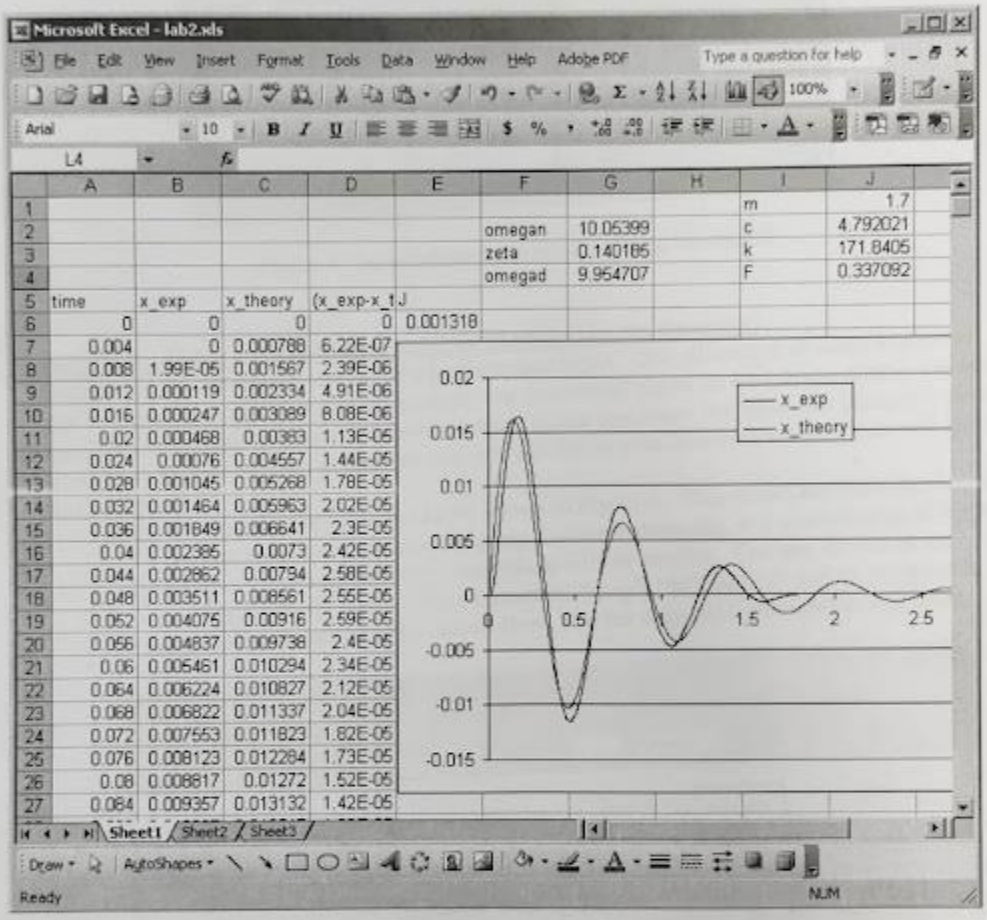
$$x(t) = \frac{Fe^{-\zeta\omega_n t}}{m_{eq}\omega_d}\sin(\omega_d t) \tag{12}$$

Your task is to find the value of F, ζ and ω_n that minimizes J. The easiest way to do this is in Excel, although Matlab can be also be used. Let's first look at how to do this in Excel.

Finding parameters using Excel

Format a spreadsheet as shown in Figure 4. Columns A and B contain your experimental data. Be sure your experimental data is in meters and not centimeters. Column C contains the theoretical prediction (your analytical solution of the ODE) which is affected by your choice of parameters. The formula entered in Column C should reference the cells containing omegan, zeta and omegad which in turn reference F, k, m and c. Column D is the square of the difference

between theoretical and experimental. Cell E6 contains the sum of Column D. Cells J2, J3, and J4 contains your initial guesses of c, k and F respectively (use the values you found using log decrement and a least squares solution as a first guess).



	A	B	C	D	E	F	G	H	I	J
1									m	1.7
2						omegan	10.05399		c	4.792021
3						zeta	0.140185		k	171.8405
4						omegad	9.954707		F	0.337092
5	time	x_exp	x_theory	(x_exp-x_t	J					
6	0	0	0	0	0.001318					
7	0.004	0	0.000788	6.22E-07						
8	0.008	1.99E-05	0.001567	2.39E-06						
9	0.012	0.000119	0.002334	4.91E-06						
10	0.016	0.000247	0.003089	8.08E-06						
11	0.02	0.000468	0.00383	1.13E-05						
12	0.024	0.00076	0.004557	1.44E-05						
13	0.028	0.001045	0.005268	1.78E-05						
14	0.032	0.001464	0.005963	2.02E-05						
15	0.036	0.001849	0.006641	2.3E-05						
16	0.04	0.002395	0.0073	2.42E-05						
17	0.044	0.002862	0.00794	2.58E-05						
18	0.048	0.003511	0.008561	2.55E-05						
19	0.052	0.004075	0.00916	2.59E-05						
20	0.056	0.004837	0.009738	2.4E-05						
21	0.06	0.005461	0.010294	2.34E-05						
22	0.064	0.006224	0.010827	2.12E-05						
23	0.068	0.006822	0.011337	2.04E-05						
24	0.072	0.007553	0.011823	1.82E-05						
25	0.076	0.008123	0.012284	1.73E-05						
26	0.08	0.008817	0.01272	1.52E-05						
27	0.084	0.009357	0.013132	1.42E-05						

Figure 4: Spreadsheet example for tuning the model.

Use Solver to minimize the sum of the square error by varying the value of your parameters. If you do not have Solver under your "tools" menu go to "Add-ins" and add it. To use Solver set the "target cell" as the cell in which the cost function is computed and set the "by changing cell" to the cell in which the parameters k, c and mag are located. To start the optimization, click on the Solve key. A sample Solver window is shown in Figure 5.

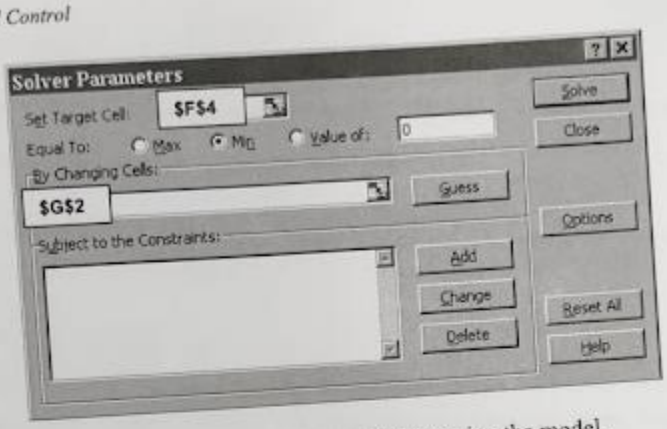


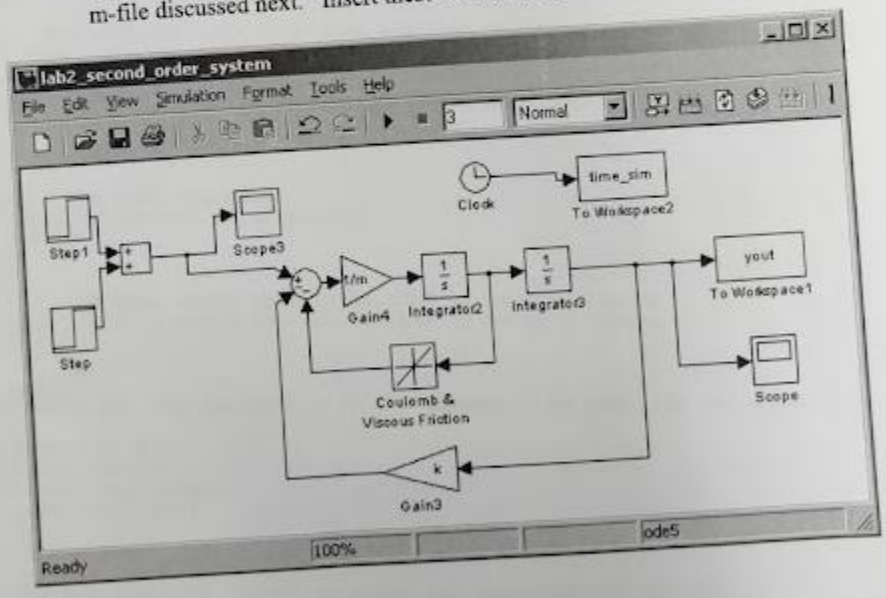
Figure 5: Spreadsheet example for tuning the model.

Method 3 – Using a performance index in Matlab (viscous damping)

Matlab can also be used to minimize a performance index. One advantage of using Matlab is that it allows us to use Simulink to generate our theoretical model so we can use a more complex model that may not have a closed form solution. Since you know the input was actually a pulse of width 0.05 s rather than an impulse you can use this in your Simulink model. You will need three files:

File 1: You need a Simulink model as shown in Figure 6. This model has been developed so that it can have viscous damping, Coulomb damping, or a combination of the two.

a. Use a fixed times step of magnitude 0.01 or smaller. You can do this in the "Simulation" menu. The parameters, "mag", "fd", and "c" will be assigned in an m-file discussed next. Insert these into the step and Coulomb/viscous block.



File 2: A Matlab m-file to set initial parameters and call "fminsearch" as shown below.

```
% m-file to define initial parameters and call fminsearch

Clear
load data2b      % this loads the data file with variables time and x1
m = 1.7;         % this is the equivalent mass determined earlier plus the added 1 kg
t1 = time;
x1 = x1/100;  % convert x1 to meters

% some initial values
k = 100;     % spring constant
c = 2;       % viscous damping coefficient
fd =0;       % Coulomb damping constant
mag=2.4;     % magnitude of the rectangular pulse input (make pulse 0.1 second long)

sim('lab2_second_order_system',[0 3]);   % this runs the Simulink model for 3 seconds
time_sim0= time_sim;     % save time using initial parameters
yout_0=yout;             % save displacement using initial parameters

%  Let's now minimize a performance index
% x0 = [k c fd mag];  % this is what you'd use if you vary k, c, fd and mag

x0 = [k c mag];  %fd not varied in this example

% the next lines set up the inputs for fminsearch
options=optimset(@fminsearch)
options=optimset(options,'Display','iter');

% run fminsearch to minimize a cost function J.  The function is defined
% by the m-file "lab2.m" and the outputs are given in the variable coeff

coeff=fminsearch(@lab2,x0,options)

k = coeff(1);
c = coeff(2);
mag = coeff(3);
sim('lab2_second_order_system',[0 3]);  % run the Simulink model with final values

plot(time,x1,'b+',time_sim0,yout_0,'k',time_sim,yout,'r')
legend('raw data','initial guess','best fit')
```

File 3: A function routine the calculates the performance index

```
function J = lab2(x)

load data2b      % load the data
x1 = x1/100;     % convert x1 to meters

k = x(1);
c = x(2);
% fd = x(3);     % this would need to be used if using 4 inputs
% mag = x(4);    % this would need to be used if using 4 inputs
mag = x(3);
fd = 0;

% the next lines put the parameters in the workspace so Simulink can acces
% them
assignin('base','k',k);
assignin('base','c',c);
assignin('base','mag',mag);
assignin('base','fd',fd);

sim('lab2_second_order_system',[0 3]);

% Interpolate the Simulink output to match the experimental points
```

```
y_interp = interp1(time_sim,yout,time);

% calculate the performance index
J=sum((y_interp-x1).^2);
```

Method 4 – Using a performance index in Matlab (Coulomb damping)
Modify the files shown above assuming that there is only Coulomb damping. Be careful to modify both of the m-files.

Method 5 – Using viscous and Coulomb damping
Modify the files shown above assuming that there is Coulomb and viscous damping. You may need to try various initial conditions. Try to get the best combination of c and F_d that minimized the performance index. Make sure your final values make sense.

Reporting the Lab:
Fill out the worksheet for today's lab.

Notes:

References

[1] www.rose-hulman.edu/~cornwell/courses/em406/labs/EM406_Lab2_system%20ID_2006.pdf
[2] System Dynamics, Ogata
[3] Vibration Text, Inman

Prepared by: **Dr. Musa Abdalla**
First Established on: **Fall 2015**

Experiment #6 Worksheet – System Identification of SDOF systems

Name and ID: _____ _____

Name and ID: _____ _____

Date: _____

Measurements and Calculations:

Table 1 – Summary of log decrement results for Test

Run	T_p [s]	ω_d [rad/s]	Log Decrement δ	Damping Ratio, ζ	ω_d^2	ζ^2-1	ω_n [rad/s]
Test 1							
Test 2							
Test 3							

Table 2 – Calculated values from over-determined equation set solution

$m_{eq,cart}$ [kg]	k [N/m]

Table 3 – Summary of results from Test 2 using the equivalent mass determined using the log decrement data.

Method	Equivalent spring constant (k)	Viscous damping coefficient, (c)	Input magnitude	Dry friction coefficient (F_d)	Performance Index (J)
Log decrement			na	na	na
Performance Index – viscous damping (Excel)				na	
Performance Index - viscous damping (Matlab)				na	
Performance Index – Coulomb damping (Matlab)		na			
Performance Index – Coulomb damping (Matlab)					

Observations:

Please attach any Figures you feel help explain your results. Hint: You may want one or more Figures showing the results from the various methods. If it is difficult to tell the difference between them, then plotting the difference between the models and the experimental results may be helpful