ECE-521

Lab 1: Modeling, Simulation, and Control of a 1 Degree of Freedom System

Overview In this lab, as in virtually all labs in this course, you will do the following:

1) Experimentally determine the frequency response of a system and use this to identify the system (find the transfer function or state space model)

2) Simulate the model (with your controller and/or observer system) using Simulink.

3) Implement the controller/observer on the ECP system by modifying your Simulink model

4) Controlling the ECP system with Simulink

5) Comparing the predicted response (from the model) with the actual response (from the ECP system)

Since this is the first lab, many of the tools you will need will be given to you. In the future you will be modifying the tools before lab (as part of your homework) and then using them on the real systems during lab.

Step 0: Set Up the System. Only the first cart should move, all other carts should be fixed. You need to have at least one spring connected to the cart and at least one mass on the cart. If you want to use the active damper, unscrew the screw in the damper. ***You will be using this configuration throughout most of the course so be sure you write down all of the information you need to duplicate this configuration.***

The transfer function between input and output for this configuration can easily be shown to be

$$G(s) = \frac{K}{\frac{1}{\omega_n^2}s^2 + \frac{2\zeta}{\omega_n}s + 1}$$

where K is the static gain, ω_n is the natural frequency, and ζ is the damping ratio. These are the parameters we need to determine for this model.

1a) Initial Estimate of Static Gain

We will obtain our initial estimate of the static gain by looping at the response of the open loop system to a step input.

You will go through the following steps:

- Reset the system using **ECPDSPresetmdl.mdl.**
- Modify **Model210_Openloop.mdl** so the input is a step. To make any changes to **Model210_Openloop**.mdl, the mode must be **Normal**.
- Set the amplitude to something small, like 0.01 or 0.02.
- Compile **Model210_Openloop.mdl**
- Connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.)
- Run **Model210_Openloop.mdl.** If the cart does not seem to move much, increase the amplitude of the step. If the cart moves too much, decrease the amplitude of the step. You must also recompile after any changes.
- Estimate the static gain as

$$K = \frac{x_{ss}}{A}$$

where x_{ss} is the steady state value of the cart position, and A is the input amplitude.

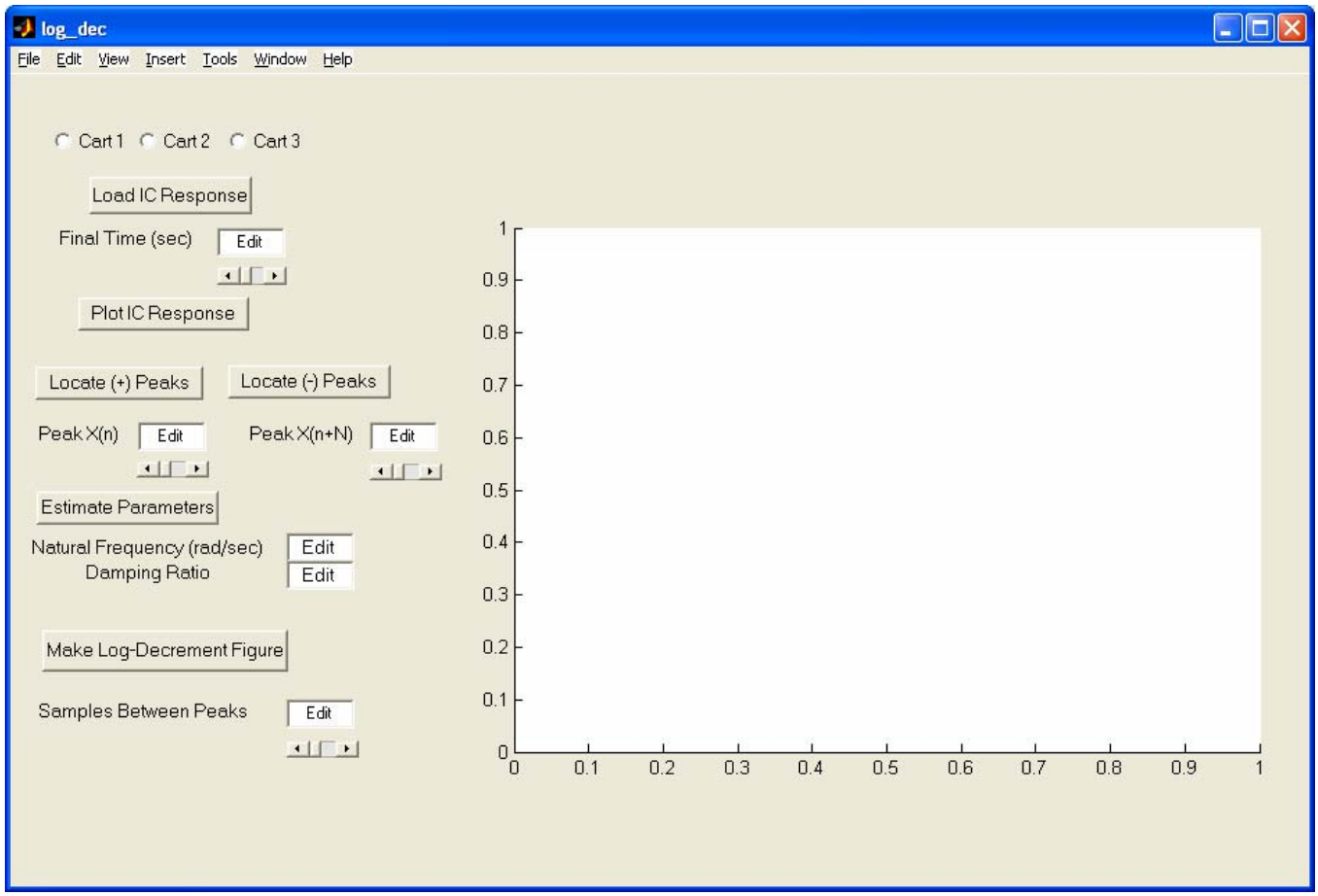
1b) Log Decrement Estimate of ζ *and* ω_n

As you recall, the log decrement method is a way of estimating the natural frequency ω_n and damping ratio ζ of a second order system. We will use these estimates as starting point for the optimization routines.

You will go through the following steps:

- Reset the system using **ECPDSPresetmdl.mdl.**
- Modify **Model210_Openloop.mdl** so the input has zero amplitude. To make any changes to **Model210_Openloop**.mdl, the mode must be **Normal**.
- Compile **Model210_Openloop.mdl**
- Connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.)
- Displace the first mass, and hold it.
- Start **(play) Model210_Openloop.mdl** and let the mass go.
- Run the m-file **Log_Dec.m.** This should be in the same directory as **Model210_Openloop.mdl** and **Log_Dec.fig**. This routine assumes the position of the first cart is labeled *x1*, the position of the second cart is labeled *x2*, the position of the third cart is labeled *x3*, and the time is labeled *time*. (These are the defaults in **Model210_Openloop.mdl**.)

The program **Log_Dec** comes up with the following GUI:



You need to

- Select the **cart** to be analyzed (cart one in this case)
- Select **Load IC (initial condition) Response** (the variables time and x1, x2, or x3 will be loaded from the workspace). At this point some initial estimates will be made.
- Set/modify the **Final Time**
- Select **Plot IC Response** to plot the initial condition response
- Choose to identify the positive peaks (**Locate + Peaks**) or negative peaks (**Locate - Peaks**) . If the peaks are not numbered consecutively, you need to decrease the **Samples Between Peaks** and try again until all peaks have been identified.
- Choose the initial peak (**Peak x(n)**) and final peak (**Peak x(n+N)**) to use in the log-decrement analysis. These should be fairly close to the beginning of the initial condition response. Don't try and use more than a few peaks.
- Select **Estimate Parameters** to get the initial estimates of ζ and ω_n
- Select **Make Log-Decrement Figure** to get a plot and summary of the results. You need to put this figure in your memo.

1c) Fitting the Estimated Frequency Response to the Measured Frequency Response

We will be constructing the magnitude portion of the Bode plot and fitting this measured frequency response to the frequency response of the expected transfer function to determine K, ζ, and ω_n. For each frequency $\omega = 2\pi f$ we have as input $u(t) = A\cos(\omega t)$ where, for out systems, A is measured in centimeters. After a transition period, the steady state output will be $x_1(t) = B\cos(\omega t + \theta)$, where B is also measured in cm. Since we will be looking only at the magnitude portion of the Bode plot, we will ignore the phase angle θ.

You will go through the following steps

For frequencies $f = 0.5,1,1.5...7.5$ Hz

- Modify **Model210_Openloop.mdl** so the input is a sinusoid. To make any changes to **Model210_Openloop**.mdl, the mode must be **Normal**.
- Set the frequency and amplitude of the sinusoid. Try a small amplitude to start, like 0.01
- Compile **Model210_Openloop.mdl**
- Connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.)
- Run **Model210_Openloop.mdl.** If the cart does not seem to move much, increase the amplitude of the input sinusoid.. If the cart moves too much, decrease the amplitude of the input sinusoid.
- Record the input frequency (f), the amplitude of the input (A), and the amplitude of the output (B) when the system is in steady state. In Matlab you can just type **plot(time,x1); grid;** once the system has stopped.

Enter the values of f, A, and B into the program **process_data.m** (you need to edit the file)

At the Matlab prompt, type **data = process_data**;

Run the program **model_1cart.m.** There are four input arguments to this program:

- data, the measured data as determined by process_data.m
- K the estimated static gain
- ω_n the estimated natural frequency (from the log decrement analysis)
- ζ the estimated damping ratio (from the log decrement analysis)

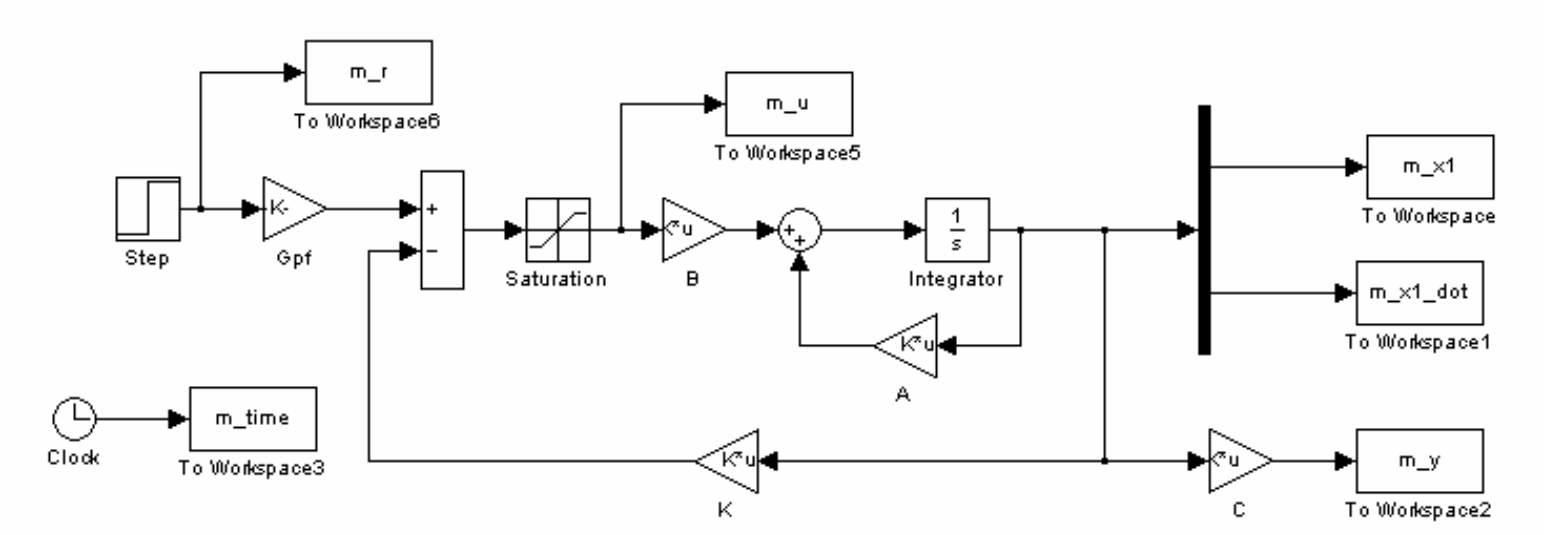
The program **model_1cart.**m will produce the following:

- A graph indicating the fit of the identified transfer function to the measured data. (You need to include this graph in your memo.)
- The optimal estimates of K, ζ, and ω_n (written at the top of the graph)
- A file **state_model.mat** in your directory**.** This file contains the A, B, C, and D matrices for the state variable model of the system. If you subsequently type **load state_model** you will load these matrices into your workspace.

You need to be sure you have 4 points close to the resonant peak of the transfer function. At this point you probably should go back and add a few points near the resonant peak.

2) Simulating the Model Using Simulink

You have been given the Simulink model **Basic_1_dof_State_Variable_Model.mdl,** which is shown below. This Simulink model implements a state variable feedback controller. We have assumed for our system that D = 0. It is important to note that virtually all of the values in the Simulink model are variables, and this model will be driven by Matlab. Hence all of the variables we use in the simulation of our system will be available in the Matlab workspace, so we can use them again when we want to use Simulink to control the ECP system.

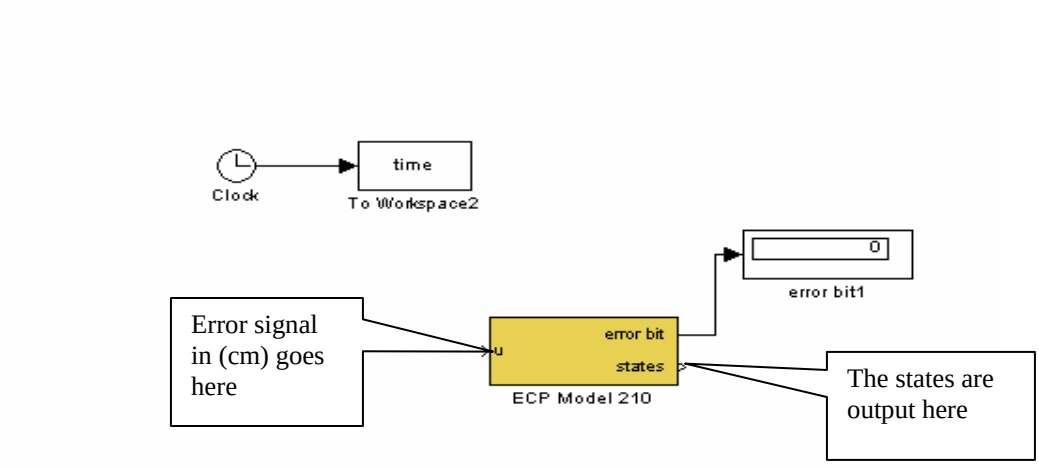


All of the output variable names begin with an m, such as **m_x1**. This is to distinguish the variables from the model with the actual values from the ECP system.

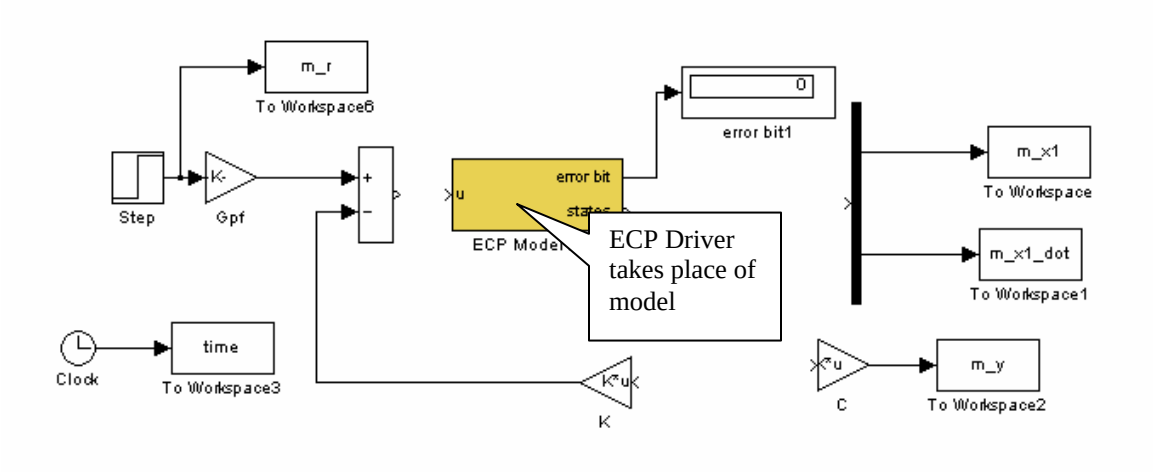
3) Implementing the Controller/Observer on the ECP System

At this point you need to be sure and save your Simulink model in a safe place (i.e. make a backup of the model by saving it with a different name), since we are going to be cutting and pasting.

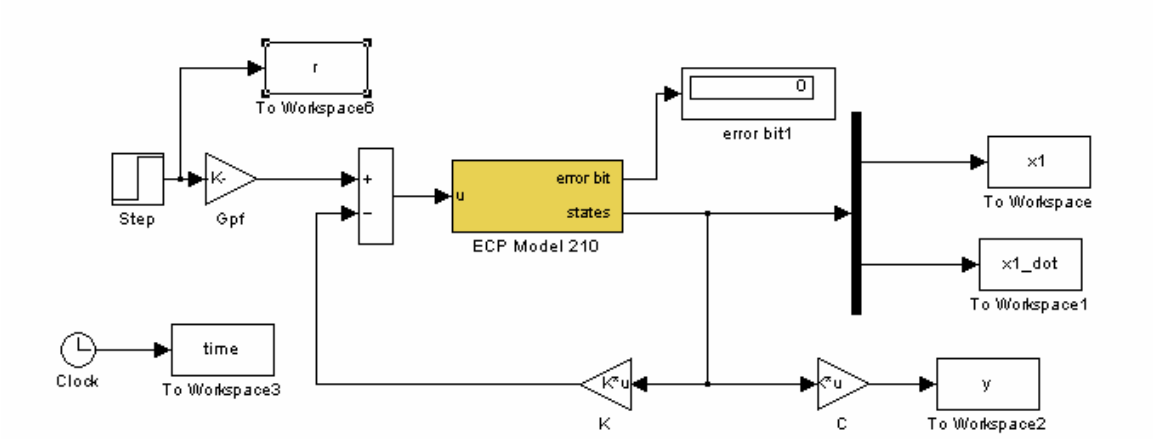
First we need to get the Simulink model **ECP210_Template.mdl**, as shown below.



The next step is to remove the model of the system and the output from your Simulink model of the system, and prepare to merge the two Simulink files, as shown below:



Then we finish connecting the systems and rename variables (so there is no overlab in names between the model and the real system) as shown on the next page:



Note that we need to use different names in the Simulink files for

- the time (time or m_time)
- the reference input (r or m_r)
- the position of the first cart (x1 or m_x1)
- the velocity of the first cart (x1_dot or m_x1_dot)

This is so we can compare the response of the model with the response of the real system.

4) Controlling the ECP System with Simulink

a) First you need to use your **model** to place the closed loop poles at -50 and -60 and try to track a 1 cm input. If the control effort is too large (reaches 0.5) you will need to move the poles in closer. (You will need to turn in your graph from this simulation.)
Once you have run your model you will have determined all of the parameters you need and they are available in Matlab's workspace.

Now try to compile your Simulink that will be driving the ECP system with these same parameters.

Run the Simulink driving the ECP system. If the system runs acceptably, produce plots comparing the position of the cart as a function of time with the predicted position of the cart as a function of time. Similarly, compare the velocity of the cart as a function of time with the predicted velocity of the cart as a function of time. (*You will be doing this alot, you may want to write an m-file to help you.*) Both of these plots should be well labeled using legends and different line types and included in your memo. How close is the predicted response to the real (measured) response?

If the ECP system does not work (or buzzes), first try resetting the system.
Then try making the closed loop poles closer to the origin. Be sure to rerun the **model** of the system to get all the necessary parameters in the workspace before compiling the ECP system.

b) Utilize the linear quadratic regulator algorithm to achieve the same performance (at least in terms of settling time) as the direct pole placement. You will have to simulate your **model** a number of times to get the performance you want before you try it on the ECP system. Record the weights you used.

Run the Simulink driving the ECP system. If the system runs acceptably, produce plots comparing the position of the cart as a function of time with the predicted position of the cart as a function of time. Similarly, compare the velocity of the cart as a function of time with the predicted velocity of the cart as a function of time. Both of these plots should be well labeled using legends and different line types and included in your memo. How close is the predicted response to the real (measured) response?

If the ECP system does not work (or buzzes), first try resetting the system.
Then try making the closed loop poles closer to the origin. Be sure to rerun the **model** of the system to get all the necessary parameters in the workspace before compiling the ECP system.