ECE-320 Lab 3Model Matching Control, and System Identification of a 2 Degree of Freedom System

Overview

In the first part of this lab you will be controlling the systems you previously modeled using an ITAE, and a deadbeat controller (one of each type of controller.). For each system you modeled, you need to plot and examine the difference between how the model predicts the response and how the system actually responds. Then you will obtain a system model for one two degree of freedom system.

*It will be easiest to do this Lab in the same folder you used for Lab 2. You will also need the files **closed_loop_driver.m** and **closedloop.mdl** from Lab 1 and the additional files from the class website. You may also have to change the base address of your ECP Simulink driver files.*

Special note: When comparing the predicted response and the actual response, I do not want to see a graph that shows very little. For example, if both the predicted response and the real response have settled within 0.5 seconds, I do not want to see a graph that goes out to 4 seconds.

Design Specifications: *Not all systems or controllers will work for a 1 cm step or a 15 degree step. If your system will not work for these values, use the lower value in the design specification. For each of your systems controlled using model matching, you should try and adjust your parameters until the real (ECP) system has achieved the following:*

Torsional Systems (Model 205)

- Settling time less than 0.75 seconds.
- Absolute value of the steady state error less than 2 degrees for a 15 degree step, and less than 1 degree for a 10 degree step (*the input to the Model 205 must be in radians!*) Only one of these is required.
- Percent Overshoot less than 10%

Rectilinear Systems (Model 210)

- Settling time less than 0.75 seconds.
- Absolute value of the steady state error less than 0.1 cm for a 1 cm step, and less than 0.05 cm for a 0.5 cm step. Only one of these is required.
- Percent Overshoot less than 10%

You should start with the lower step amplitudes, and work your way up to the higher step amplitudes. Your real systems may oscillate a bit. If this happens try to reduce the input level, since this limits the allowed control effort. *It may not be possible to eliminate all of* *the oscillations, since these types of controllers depend on canceling the plant dynamics,* and if your model is not accurate enough the controller will not cancel the real plant well enough. Do the best you can*, as though your grade depended on it.*

PART A: *Changes to **closedloop_driver.m***

Note: You may not need to make all of the changes below. It depends on how much of this you did in Lab 1.

You should end up with a section of code which includes all the different types of controllers. You will comment out the types you are not using, and will be adding to these in the next few labs.

Step 1: Modify **closedloop_driver.m** to read in your state model file. For example, if the model file was named **state_model_1dof.mat** you need the following near the beginning of your Simulink file

```
load state_model_1dof
C = [1 0];
[num_Gp,den_Gp] = ss2tf(A,B,C,D);
```

Step 2: Be sure **closedloop_driver.m** contains the following lines.

```
num_Gp = (abs(num_Gp) > tol*ones(1,length(num_Gp))).*num_Gp;
den_Gp = (abs(den_Gp) > tol*ones(1,length(den_Gp))).*den_Gp;
```

Step 3: Modify the **saturation_level** variable in **closedloop_driver.m** for the appropriate system.

```
saturation_level = 1000/2196;  % (rectilinear system, Model 210)
saturation_level = 1000/2546;  % (torsional system, Model 205)
```

Comment out these lines until you need them. To comment out use a '%'

Step 4: Modify **closedloop_driver.m** for utilizing ITAE model matching control. Here, the desired closed loop transfer function is:

$$G_o(s) = \frac{\omega_o^2}{s^2 + 1.4\omega_o s + \omega_o^2}$$

You should enter this into **closedloop_driver.m**, but leave the frequency ω_o a variable.

Step 5: Modify **closedloop_driver.m** for utilizing deadbeat model matching control. Here the desired closed loop transfer function is:

$$G_o(s) = \frac{\omega_o^2}{s^2 + 1.82\omega_o s + \omega_o^2}$$

You should enter these into **closedloop_driver.m**, but leave the frequency ω_o a variable.

<u>Step 6:</u> Modify **closedloop_driver.m** to determine the controller. For model matching control, this will be

Gc = minreal(Go/(Gp*(1-Go))

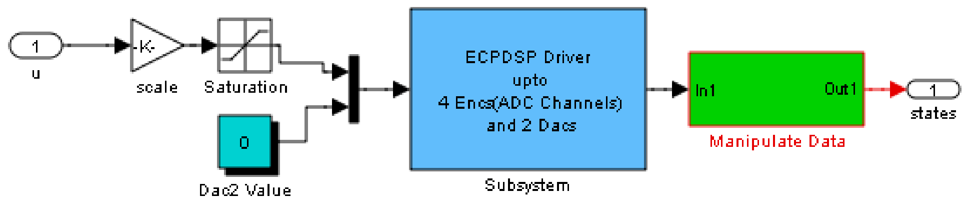
Be sure to use the **minreal** command.

<u>Step 7:</u> Be sure **closedloop_driver.m** computes the prefilter gain, Gpf. You did this in Lab 1.

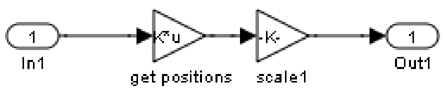
PART B: *Creating Closed Loop Simulink Systems for the ECP Systems*

We will go over how to change the Model 205 (torsional system), and you will need to follow the same steps for the Model 210 (rectilinear system). Open **Model205_Openloop.mdl** and save it as **Model205_Closedloop.mdl.**

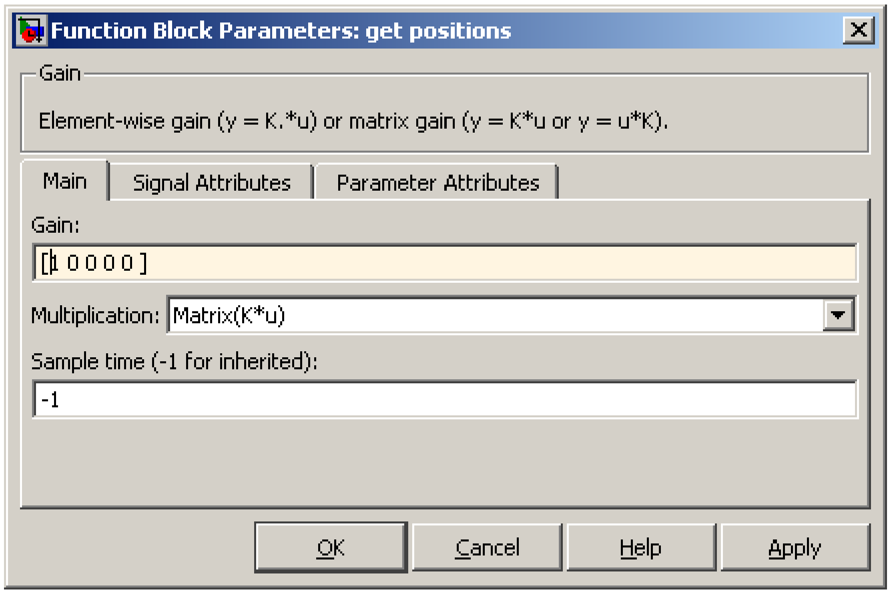
You will next have to modify the ECP block (the yellow block) to have only one output. To do this, click on the yellow block, and you should get the following inner block:



Next click on the *Manipulate Data* block, to get



Finally, click on the *get positions* block and change the *Gain:* so it looks like the following:


Function Block Parameters: get positions

Gain

Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).

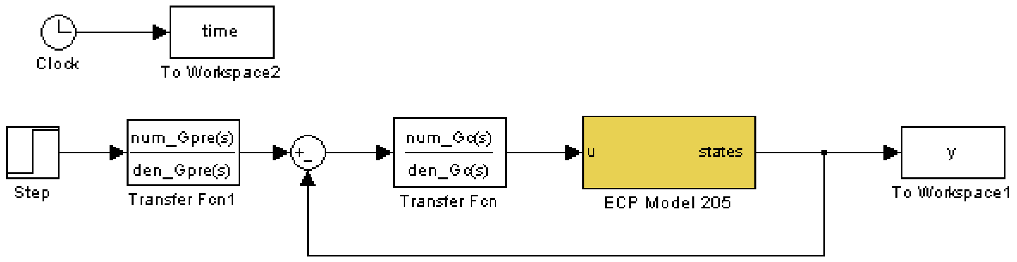
Gain: [1 0 0 0 0]

Multiplication: Matrix(K*u)

Sample time (-1 for inherited): -1

This tells the ECP block that the only output is the position of the first disk. Save and close all of the intermediate files.

Next, modify **Model205_Closedloop.mdl** so it looks like the following



This is a cross between **closedloop.mdl** and **Model205_Openloop.mdl.**, so you can pretty much just cut from **closedloop.mdl** and paste into **Model205_Closedloop.mdl.** In the **step** function, be sure the *Final value* is set to *Amp*, the *Step time* is set to *zero.* Also set the *Stop time* in the Simulink model file to *Tf* . Note that we have named the output *y*. It will be easier for you if the output from both the Model205 and Model210 are labeled *y*, so you should do this.

Now go through the same steps for the Model 210 system.

PART C: *Model Matching Control of One Degree of Freedom Systems*

For each of your two 1 dof systems, you will need to go through the following steps:

Step 1: Set up the 1 dof system exactly the way it was when you determined its model parameters.

Step 2: Modify **closedloop_driver.m** to read in the correct model file. You may have to copy this model file to the current folder.

Step 3: Modify **closedloop_driver.m** to use the correct *saturation_level* for the system you are using.

Step 4: ITAE Control. *(Be sure to record the value of ω_o you use.)*

- Vary ω_o until you meet the design specs with the *simulation* ***(closedloop_driver.m)***. The larger the value of ω_o, the faster your system will respond and the closer your model will predict the response of the real system. Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.

- Compile the correct closed loop ECP Simulink driver, connect to the system, and run the simulation.

- Use the **compare1.m** file (or a modification of it) to plot the results of both the simulation and the real system on one nice, neatly labeled graph. You may need to change this file so the output from the ECP system is labeled *y*, not *x1* or *theta1*. The results for the torsional systems ***must be displayed in degrees***. You need to include this graph in your memo.

Step 5: Deadbeat Control. *(Be sure to record the value of ω_o you use.)*

- Vary ω_o until you meet the design specs with the *simulation* ***(closedloop_driver.m)***. The larger the value of ω_o, the faster your system will respond and the closer your model will predict the response of the real system. Be sure you do not reach the limiter on the control effort, unless you really like restarting your computer. At this point all of the variables you should need are in your current Matlab workspace.

- Compile the correct closed loop ECP Simulink driver, connect to the system, and run the simulation.

- Use the **compare1.m** file (or a modification of it) to plot the results of both the simulation and the real system on one nice, neatly labeled graph. The results for the torsional systems ***must be displayed in degrees***. You need to include this graph in your memo.

PART D: *System Identification of a Two Degree of Freedom Systems*

You need to fill out the data sheet indicating the configuration on the last page of the lab and turn it in!

Step 0: Set Up the System. Both the first and second carts should move. The third cart should be fixed in place. In addition:

- Either the carts should have an equal amount of weight on them, or the first cart should have more weight than the second cart. You need at least one mass on each cart.

- Either all springs connecting carts should have equal stiffness, or the springs should get less stiff from left to right. You need to use at least two springs.

- If you want to use the active damper, unscrew the screw in the damper.

You will be using this configuration throughout the remainder of the course so be sure you write down all of the information you need to duplicate this configuration.

Step 1a) Initial Estimates of ω_1, ζ_1, ω_2, *and* ζ_2

From the equations of motion we have

$$\begin{aligned}\ddot{x}_1 + 2\zeta_1\omega_1\dot{x}_1 + \omega_1^2 x_1 &= \frac{k_2}{m_1}x_2 + \frac{1}{m_1}F \\ \ddot{x}_2 + 2\zeta_2\omega_2\dot{x}_2 + \omega_2^2 x_2 &= \frac{k_2}{m_2}x_1\end{aligned}$$

a) If there is no applied force ($F = 0$) and the second cart is fixed in place ($x_2 = 0$), we have

$$s^2 + 2\zeta_1\omega_1 s + \omega_1^2 = 0$$

Use the log decrement method to get our initial estimate of ω_1 and ζ_1.

b) If the second cart is free to move and the first cart is fixed in place ($x_1 = 0$), we have

$$s^2 + 2\zeta_2\omega_2 s + \omega_2^2 = 0$$

Use the log decrement method to get our initial estimate of ω_2 and ζ_2. For the log-decrement analysis you will go through the following steps for **each** cart:

- Be sure only one cart is free to move
- Reset the system using **ECPDSPresetmdl.mdl.**
- Modify **Model210_Openloop.mdl** so the input has ***zero*** amplitude.
- Compile **Model210_Openloop.mdl** if necessary.
- Connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.)
- Displace whichever cart is free to move and hold it.
- Start **(play) Model210_Openloop.mdl** and let the cart 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,* and the time is labeled *time*. (These are the defaults in **Model210_Openloop.mdl**.)

You will need to include these **two** log-decrement results in your memo.

Step 1b) Estimating the Gains K_1 *and* K_2

You will go through the following steps:

- Be sure both carts are free to move
- Reset the system using **ECPDSPresetmdl.mdl.**
- Modify **Model210_Openloop.mdl** so the input is a step. Set the amplitude to something small, like 0.01 or 0.02 cm.
- Compile **Model210_Openloop.mdl**
- Connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.)
- Run **Model210_Openloop.mdl.** If the carts do not seem to move much, increase the amplitude of the step. If the carts move too much, decrease the amplitude of the step. You may have to recompile.
- Estimate the static gains as

$$K_1 = \frac{x_{1,ss}}{A} \quad K_2 = \frac{x_{2,ss}}{A}$$

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

You should do this in Matlab, don't use the X-Y Graph. The variables *x1*, *x2,* and *time* should be in your workspace. You need to increase the value of the input amplitude until the first cart is moving about 1.5 cm or so Use the static gains associated with this input amplitude as your estimates of the static gain.

<u>Step 1c) Fitting the Estimated Frequency Response to the Measured Frequency Response</u>

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 the parameters we need. 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 $x_1(t) = B_1\cos(\omega t + \theta_1)$ for the first cart and $x_2(t) = B_2\cos(\omega t + \theta_2)$ for the second cart, where both B_1 and B_2 are also measured in cm. Since we will be looking only at the magnitude portion of the Bode plot, we will ignore the phase angles θ_1 and θ_2.

You will go through the following steps

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

- Make sure both carts are free to move, and the third cart is fixed.
- Modify **Model210_Openloop.mdl** so the input is a sinusoid. You may have to set the mode to **Normal**.
- Set the frequency and amplitude of the sinusoid. Try a small amplitude to start, like 0.01 cm. Generally this amplitude should be as large as you can make it without the system hitting a limit. This amplitude will probably vary with each frequency.
- Compile **Model210_Openloop.mdl,** if necessary. (Assume it is not necessary. The system will let you know if it is necessary.)
- Connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.)
- Run **Model210_Openloop.mdl.** If the carts do not seem to move much, increase the amplitude of the input sinusoid. If the carts move 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_1 and B_2) when the system is in steady state. Modify the program **get_Amp2.m** to help record these amplitudes accurately. Be sure the plot from **get_Amp2** shows the system in steady state. Be sure to look at the graph and understand what the code is doing before you use it!!!
-

Enter the values of f, A, B_1 and B_2 into the program **process_data_2dof.m** (you need to edit the file)

At the Matlab prompt, type **data = process_data_2dof**;

Run the program **model_2dof.m.** There are ten input arguments to this program:

- **data**, the measured data as determined by **process_data_2dof.m**
- the estimated value of K_2
- ω_a, the estimated frequency of the first resonance, when both carts are moving, in radians/sec
- ζ_a, the estimated first damping ratio when both carts are moving. Assume $\zeta_a = 0.1$.
- ω_b, the estimated frequency of the second resonance, when both carts are moving, in radians/sec
- ζ_b, the estimated second damping ratio when both carts are moving. Assume $\zeta_b = 0.1$.
- ω_1 the estimated natural frequency of the first cart when it is the only cart moving (from the log decrement analysis)
- ζ_1 the estimated damping ratio of the first cart when it is the only cart moving (from the log decrement analysis)
- ω_2 the estimated natural frequency of the second cart when it is the only cart moving (from the log decrement analysis)
- ζ_2 the estimated damping ratio of the second cart when it is the only cart moving (from the log decrement analysis)

The program **model_2dof.**m will produce the following:

- A graph indicating the fit of the identified transfer function to the measured data for the first cart (You need to include the *final* graph of this fit in your memo.)
- A graph indicating the fit of the identified transfer function to the measured data for the second cart (You need to include the *final* graph of this fit in your memo.)
- The optimal estimates of all parameters (written at the top of the graphs)
- A file **state_model_2dof.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_2dof** you will load these matrices into your workspace.

You need to be sure you have 4 points close to the resonant peaks of the transfer functions. This is particularly true if you have very small values of ζ (which correspond to very sharp peaks) At this point you probably should go back and add a few points near both the resonant peaks and nulls.

Your memo should include at least two graphs for each of the 1 dof systems you used for the model matching control. The values of ω_o used in the controller must be in the figure captions or as part of the graph (not handwritten). The figures should all be attachments, at least two per page. Your memo should compare the difference between the predicted response (from the model) and the real response (from the real system) for each of the systems. Attach your Matlab driver file ***closedloop_driver.m***

Your memo should also include a description of your two degree of freedom system (so you can set them it again), a table comparing the estimated values of the static gains, the natural frequencies, and the damping ratios using the two different methods (time domain and optimized frequency domain), and a brief comparison of the values. The damping ratios are often quite different, so that's OK. The other values should be close. You should include as attachments 4 additional graphs (2 log-decrement and 2 frequency response graphs), each with a figure number and caption. You should also include the data used for estimating the static gain.

<u>2 dof rectilinear systems (model 210)</u>

Damper:	Yes No	
Left Spring:	stiff/light/none	Spring Number =
Number of Large Masse on First Cart:	2 3 4	
Number of Small Masses on First Cart:	1 2 3 4	
Middle Spring :	stiff/light	Spring Number =
Number of Large Masse on Second Cart:	2 3 4	
Number of Small Masses on Second Cart:	1 2 3 4	
Right Spring :	stiff/light	Spring Number =

state_model_2dof.mat *is now named :*