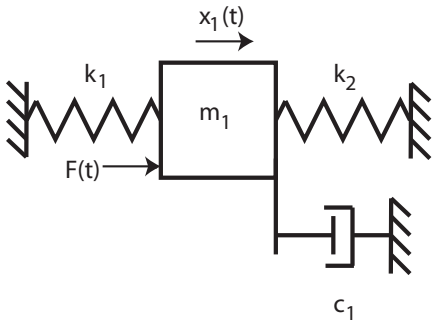
ECE-320 Lab 1
System Identification and Model Matching Control of a 1 dof Rectilinear System

In this lab you will be modeling and controlling a one degree of freedom rectilinear system. You will first fit the parameters of a transfer function model of your syste using time-domain analysis and frequency domain analysis. The steps we will go through in this lab are very commonly used in system identification (determining the transfer function) of unknown systems. We will utilize these models in later labs so do a good job in this lab, your results in later labs will be affected by how well you perform in this lab.

A one degree of freedom rectilinear mass-spring-damper system



can be modeled as

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

Here 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 to match the model to your system.

You will need to set up a folder for this lab and copy and unzip all of the files from ***Lab1 files.rar*** *from the class website. You may need to review the file* ***shortguide.pdf.*** *You may also need to set the* ***base address*** *for your system.*

Your memo should include a detailed descriptions of your system (so you can set them up again), the name of your model file. a ***table*** *comparing the estimated values of K, ω_n and ζ using the time domain and frequency domain estimates, 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 6 graphs (log-decrement and frequency response graphs for the rectilinear system, and four model matching results), each with a Figure number and caption. You should also include the data used for estimating the static gain. However, do not include the data for constructing the Bode plot.*

Part A: Time Domain Modelling of a One degree of Freedom System

Step 1: Set Up the System. Only the first cart should move, all other carts should be fixed. You need to have at least one spring connecting the first cart to the second cart (you may also have an additional spring between the motor and the first cart) and at least two masses on the cart. Do not use the damper. *If you use two springs, the stiffer of the two springs should be between the first cart and the motor!* ***Be sure you write down all of the information you need to duplicate this configuration. You need to fill out the data sheet indicating each configuration on the last page of the lab and turn it in! You also need to include this information in your memo.***

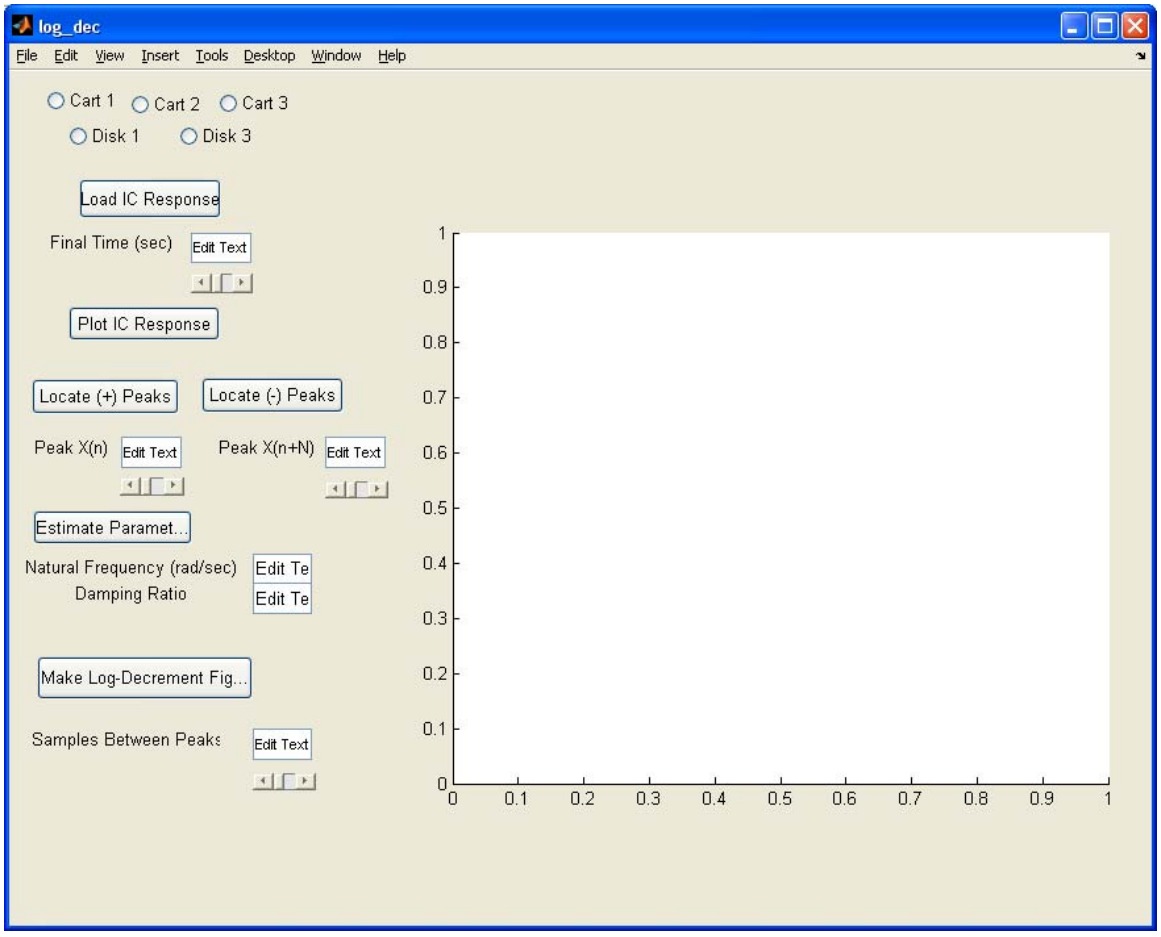
Step 2: Preparing to Use the ECP System with Simulink. Be sure the connector box (black and gray box on the top of the shelf) is off before connecting the system. *Do not force the connectors. If they don't seem to fit, ask for help!*

- Read the file **shortguide.pdf** and change the *Base Address* of both **ESCPDSPReset.mdl** and **Model210_Openloop.mdl** if necessary.
- Be sure to save the files after changing the *Base Address.*
- *Be sure to load the correct controller personality file for the ECP system !!!*

Step 3: Log Decrement Estimate of ζ *and* ω_n

The log decrement method is a way of estimating the natural frequency ω_n and damping ratio ζ of a second order system. You will go through the following steps:

- 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 the first mass, and hold it.
- Start **(play) Model210_Openloop.mdl** and let the mass go.
- Run the m-file **log_dec.m.** (be sure the file **log_dec.fig** is in the same folder**)**. The program **log_dec** comes up with the following GUI:



You need to

- Select **Cart 1**
- Select **Load IC (initial condition) Response** (the variables *time* and *x1* 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 include this figure in your memo.*

Step 4: Estimating the Static Gain K

You will go through the following steps:

- Reset the system using **ECPDSPresetmdl.mdl.**
- Modify **Model210_Openloop.mdl** so the input is a step. You may have to set the mode to **Normal**.
- Set the amplitude to something small, like 0.01 or 0.02 cm.
- Compile **Model210_Openloop.mdl,** if necessary.
- 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 may have to recompile after the change.
- You only need to run the system until it comes to steady state, then stop it.

Estimate the static gain as $K = \frac{x_{1,ss}}{A}$ where $x_{1,ss}$ is the steady state value of the cart position, and A is the input amplitude. You should determine the value $x_{1,ss}$ in Matlab, **don't use the X-Y Graph**. The variables *x1* and *time* should be in your workspace. You should use **three different input amplitudes** and produce three different estimates for the static gain. Try and make your systems move so the steady state values are between 0.5 and 1.0 cm. Average your three estimates to produce a final estimate of the static gain. Your static gain should generally be between 10 and 30. If yours is not, it is likely you did not use the same units for A and $x_{1,ss}$.

Part B: Frequency Domain Modelling of a One degree of Freedom System

Step 1: 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 our 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. 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 or the system shutting off. This amplitude will probably vary with each frequency. *The frequency needs to be entered in radians/sec!*
- 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 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. Not that the output may not be a sine wave symmetric about zero. Hence you need the average of the positive and negative values. The Matlab file **get_B.m** will help with this.

Enter the values of f , A, and B into the program **process_data_1dof.m** (you need to edit the file)

At the Matlab prompt, type **data = process_data_1dof**;

Step 2: Fitting Your Model to the Transfer Function

Run the program **model_1dof.m** at the Matlab prompt**.** There are four input arguments to this program:

- data, the measured data as determined by **process_data_1dof.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_1dof.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_1dof.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_1dof** you will load these matrices into your workspace.

Step 3: Improving the Model

Add additional data points so you have ***at least*** 4 points close to the resonant peak of the transfer function. It is important that this be well defined before you go on. If your damping ratio is less than 0.01 you definitely need more points near the peak!

Step 4: Renaming your Model

You should save and rename the files **process_data_1dof.m,** and **state_model_1dof.mat** in a way that you will be able to identify them later. *Write these names on the data sheet at the end of this lab and include theses names in your memo.* These are the files that contain a state variable model of your system and will be used in nearly all future labs!

Part C: Model Matching Control of a One degree of Freedom System

We would like our final design to meet the following design requirements:

- Settling time less than 0.5 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
- Percent Overshoot less than 10%

You should start with the lower step amplitudes, and work your way up to the higher step amplitudes. Not all systems or controllers will work for a 1 cm step. 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.

Step 1: Modify **closedloop_driver.m** to read in the correct model file (from part B-4).

Step 2: Modify **closedloop_driver.m** to use the correct *saturation_level* for the system you are using.

Step 3: Second Order ITAE Model Matching Control. (Be sure to record the value of ω_o you use.)

- Using a second order ITAE system, 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. Y*ou should run the simulation until the system is clearly at a steady state value, but at least one second.*

- Open **Model210_Closedloop.mdl** and be sure the *Base Address* is correct for your system.

- Compile **Model210_Closedloop.mdl**. The parameters this file needs are in the workspace. The yellow block is what makes the ECP system work, and replaces your model of the system with the real system.

- Reset the system by running **ECPDSPReset.mdl**.

- Connect **Model210_Closedloop.mdl** to the system and then run it.

- 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 modify this file to get the plot you want. *You need to include this graph in your memo.*

Step 4: Third Order ITAE Model Matching Control. (Be sure to record the value of ω_o you use.)

- Using a third order ITAE system, vary ω_o until you meet the design specs with the *simulation **(closedloop_driver.m)***. Y*ou should run the simulation until the system is clearly at a steady state value, but at least one second.*

- Open **Model210_Closedloop.mdl** and be sure the *Base Address* is correct for your system.

- Compile **Model210_Closedloop.mdl**. The parameters this file needs are in the workspace. The yellow block is what makes the ECP system work, and replaces your model of the system with the real system.

- Reset the system by running **ECPDSPReset.mdl**.

- Connect **Model210_Closedloop.mdl** to the system and then run it.

- 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 need to include this graph in your memo.*

Step 5: Second Order Deadbeat Model MatchingControl. (Be sure to record the value of ω_o you use.)

- Using a second order deadbeat system, vary ω_o until you meet the design specs with the *simulation **(closedloop_driver.m)***. Y*ou should run the simulation until the system is clearly at a steady state value, but at least one second.*

- 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 need to include this graph in your memo*.

Step 6: Third Order Deadbeat Model MatchingControl. (Be sure to record the value of ω_o you use.)

- Using a third order deadbeat system, vary ω_o until you meet the design specs with the *simulation **(closedloop_driver.m)***. Y*ou should run the simulation until the system is clearly at a steady state value, but at least one second.*

- 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 need to include this graph in your memo*.

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. There should be two figures per page, and they should be large enough I can read them.

Be sure in include the number of the ECP system you are using. These sometimes get moved around and you need to be able to find the exact same one again. They are not interchangeable.

Name(s) _____

1 dof rectilinear systems (model 210)

Left Spring: stiff/light/none Spring Number =
Number of Large Masses: 1 2 3 4
Number of Small Masses: 1 2 3 4
Right Spring: stiff/light/none Spring Number =

state_model_1dof.mat *is now named :*

process_data_1dof.m *is now named:*

Instructor Verification and Time _____