Lab 1

Time Domain Analysis of a 1DOF Rectilinear System

Objective: Become familiar with the ECP control system and MATLAB interface
Collect experimental data from a 1DOF rectilinear system
Identify the parameters of the 1DOF system using time domain analysis

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment if necessary.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors.

Prelab: Read this laboratory theory and procedure thoroughly. Submit your answer to the following question on engineering paper at the beginning of the lab.

a. *The waveform in Figure 1 is the unit step response for a mass-cart-spring system, estimate the peak time, damping frequency, percent overshoot, damping ratio and static gain. Write the equation, G(s) that models this system.*

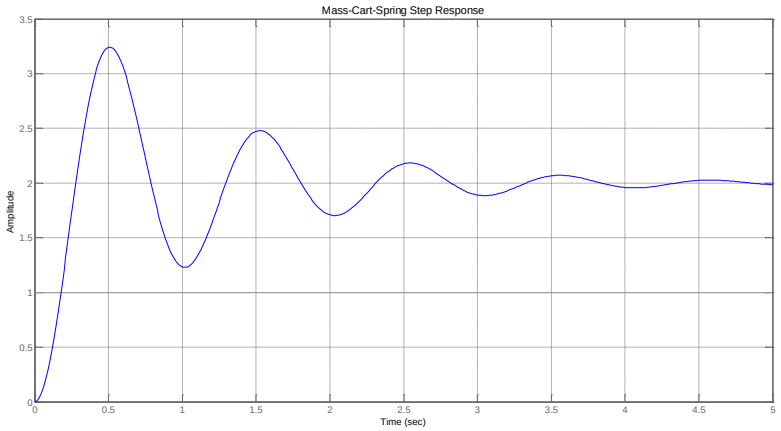


Figure 1: Mass-cart-spring underdamped step response

b. *The waveform in Figure 2 is the underdamped natural response for a mass-cart-spring system, estimate the natural frequency and damping ratio using the log-decrement method.*

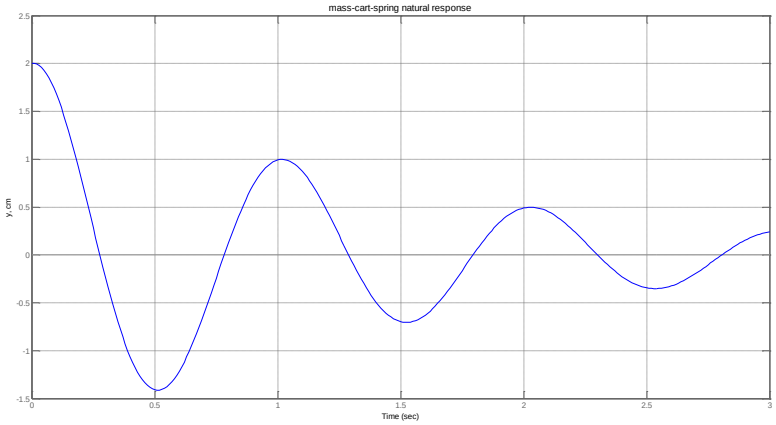


Figure 2: Mass-Cart-Spring Natural Response

Go to the Angel course folder, download and read "shortguide.pdf" to become familiar with how to set up the Education Controls Product (ECP) Model 210. If there are any questions, please ask your instructor.

Theory:

The one degree of freedom rectilinear mass-spring-damper system can be modeled by a second order differential equation. Figure 3 shows an example of a one degree of freedom rectilinear mass-spring-damper system.

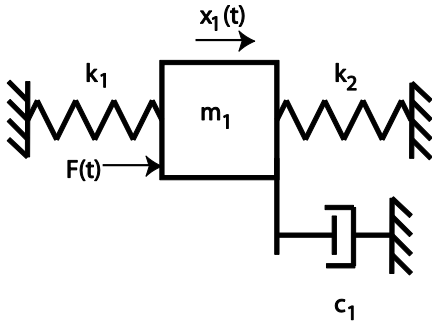


Figure 3: One DOF rectilinear mass-spring-damper system

The equation that models this system is given by

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

where K is the static gain, ω_n is the natural frequency, and ζ is the damping ratio. These are the parameters that must be determined to match the model to your system. Figure 4 provides a

typical under-damped second order system response. You will be creating a similar plot for the mass-spring system in lab in order to identify the system modeling parameters.

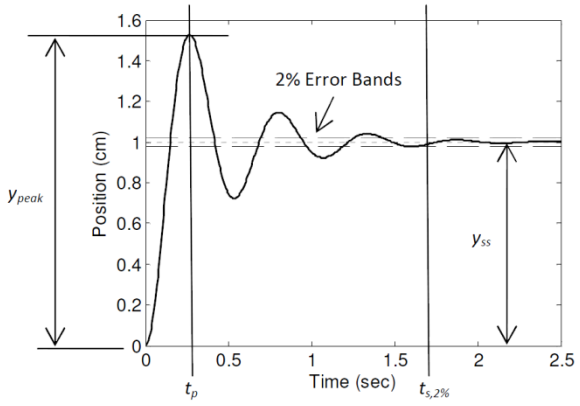


Figure 4: Second-order system under-damped step response

Recall that the second order equation is given by $\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = K\omega_n^2 \cdot f(t)$. You can use this plot and the following equations to estimate the system parameters.

Peak time	$t_p = \frac{\pi}{\omega_d}$
Damping frequency	$\omega_d = \omega_n \cdot \sqrt{1-\zeta^2}$
Percent overshoot	$\%OS = \frac{y_{peak} - y_{ss}}{y_{ss}} \; x \; 100$
Damping ratio	$\zeta = \sqrt{\frac{\ln^2(OS)}{\ln^2(OS) + \pi^2}}$
Static Gain	$K = \frac{y_{ss}}{A}$

There is an alternate method for finding the damping ratio and natural frequency of an under damped system. The **log-decrement method** can be applied to the natural (initial condition) response of a second order system in the time domain. **Logarithmic decrement**, δ, is the natural log of the amplitudes of any two successive peaks:

$$\delta = \frac{1}{n} ln \frac{x_o}{x_n}$$

where x_o is the greater of the two amplitudes and x_n is the amplitude of a peak n periods away. The damping ratio is then found from the logarithmic decrement:

$$\zeta = \frac{1}{\sqrt{1 + \left(\frac{2\pi}{\delta}\right)^2}}$$

The damping ratio can then be used to find the natural frequency ω_n.

Procedure:

A. System Setup
   1. Every week you will need to start **ECP 3.2->Download the Controller Personality File->Reset the Controller**
   2. Create a folder under Documents\ECE320\<your initials\Lab1
   3. Download the **LAB1.rar** files from the Angel class folder to the directory you created in step 1.
   4. Start MATLAB and change the current directory to the one created in step 1.
   5. Start SIMULINK by typing **simulink** in the MATLAB window
   6. Follow the procedure in the "shortguide.pdf" to reset the ECP system using SIMULINK and run the open loop model.
   7. Be sure to save the **ESCPDSPReset.mdl** and **Model210_Opennloop.mdl** files after changing the Base Address.

B. Set up the Mass-Cart-Spring system
   1. Use the Allen wrenches provided in the tool box to lock all of the carts in place except the one closest to the motor.
   2. Put two 500 gram masses on the cart closest to the motor.
   3. Select two springs, put the stiffer one between the cart and the motor and the other one between the first and second cart.
   4. You will need this set up for future labs so record the spring numbers, spring location, number and type of 500g masses. It may be helpful to draw a diagram that you can replicate later. You must provide a summary of the detailed set up in the lab memo. Figure 7 provides an example of a detailed diagram of the system setup.
   5. Be sure to also record the number of the ECP system, these sometimes get moved around and you must use the same one every week, they are not interchangeable.

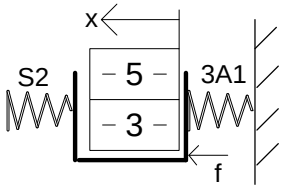


Figure 7: Mass-cart-spring system setup diagram

C. Step Response

1. Open the **Model210_Opennloop.mdl** and change the source block to a step, change the step time to "0" and the final value to "0.1" (0.1 cm). You should run the simulation until the system reaches steady state so select a reasonable value for the simulation stop time between 1 and 5 seconds. The simulation stop time is located in the main simulink window at the top next to the compile button.
2. Compile, connect and run **Model210_Opennloop.mdl.** In order to avoid compiling the open loop model every time you change the step amplitude or simulation time, you can make them both variables such as *amp* or *simtime* in the simulink block diagram and vary them in the MATLAB command window.
3. The simulation will create a graph of the step response and save the data to the *time* and *x1* variables on the MATLAB workspace. You should use these variables to create a plot of the step response (`plot(time,x1)`. Make sure to turn on the grid and put a descriptive label and title. You will estimate the system parameters from this graph. This plot and the estimated parameters should be included in your lab memo.
4. Note that these computers are old and the data acquisition may be faster than it can handle and you will get an error pop up that states too fast for hardware. Simply ignore this error and continue to work. However, if MATLAB starts to crash from these errors, you should increase the sample time in the **Model210_Opennloop.mdl** to approximately 0.004 in the ECP DAC yellow model block.
5. Repeat steps 1 through 4 for four masses on the cart. Don't forget to record the mass numbers used for inclusion in your memo and so that you can set the system up again in the future. Also, make sure to use Simulink to reset the ECP after each change in step amplitude as well as change to the setup to make sure to zero the initial conditions. Note: You don't have to re-compile **ESCPDSPReset.mdl** after each change to the system but you should connect and run.

D. Alternate estimation of static gain, K

1. There is an alternate method for estimating the static gain, by reading the data directly from the *x1* and *time* variables.
2. Set up the mass-cart-spring system with two masses as you did in part B. Collect data on the step response for an amplitude, A = 0.04, 0.06 and 0.08. Make sure to set the simulation stop time so that the system reaches steady state. See step C2 for an alternate method for setting these values to avoid re-compiling after every change.
3. View *x1* after each simulation and use the data to calculate the static gain.
4. Average the three estimates to produce a final estimate of the static gain, K. A reasonable value should be between 5 and 30. If your value is not in this range make

sure that you use the correct units for the step input amplitude and the steady-state value.

5. Repeat steps 1 through 4 for four masses on the cart. Make sure you use the masses in the same order as you did in part C. Make sure to use Simulink to reset the ECP after each change to the setup to make sure to zero the initial conditions.
6. You should include the three different estimates of the static gain, K in your lab memo. It should be in a table and compared to what you got for part C with part D as the nominal value. Comparison means to use the error formula

$$\%\ error = \frac{measured - nominal}{nominal}\ x\ 100$$

E. Log Decrement Estimate of ζ and ω_n

1. There is an alternate method for estimating the damping ratio and natural frequency by using the log decrement method.
2. Set up the mass-cart-spring system with two masses as you did in part B.
3. Modify the **Model210_Opennloop.mdl** so that the step input has an amplitude of "0"**.**
4. Compile and connect to the ECP system. (The mode should be external)
5. Displace the cart and hold it.
6. Click "play" on **Model210_Opennloop.mdl** and let the cart go.
7. The XY graph of the natural response should be generated on the screen
8. Run the m-file **log_dec.m** that and the GUI in Figure 8 should open.

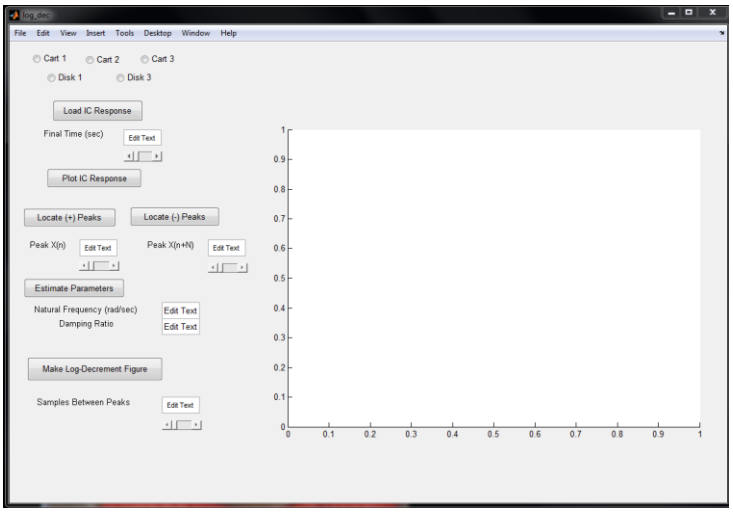


Figure 8: Log Decrement GUI Tool

9. Select **Cart 1**
10. Select **Load IC** (initial condition) Response (the variables *x1* and *time* will be loaded from workspace)
11. Set/modify the **Final Time** to be the same as the simulation stop time

12. Select **Plot IC Response** to plot the initial condition response
13. 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 (see Figure 9).

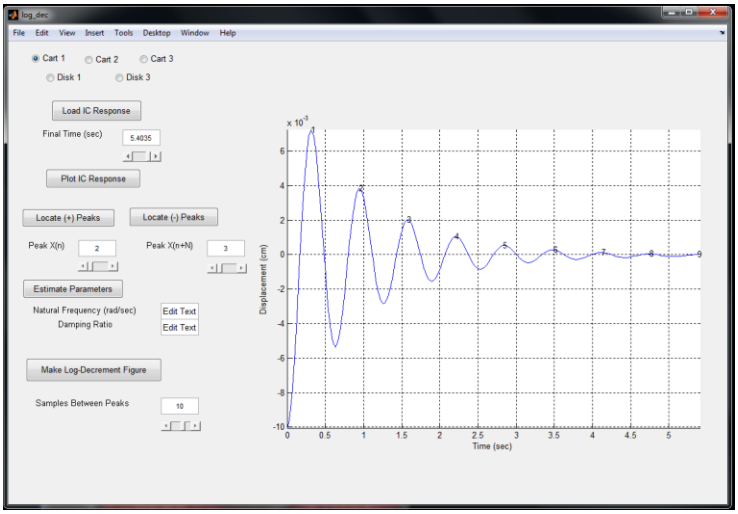


Figure 9: Locate positive peaks on the Log Decrement GUI Tool

14. 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 (i.e. 1 and 2 or 2 and 3).
15. Select **Estimate Parameters** to get the initial estimates of ζ and ω_n.
16. Select **Make Log-Decrement Figure** to get a plot and summary of the results. You need to include this figure in your memo (see Figure 10).

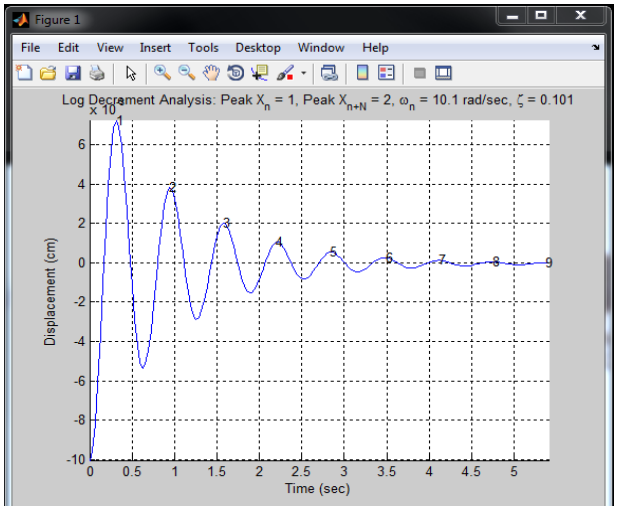


Figure 10: Locate Decrement Figure with estimates

17. Repeat steps 1 to 16 for four masses on the cart.
18. You should include the different estimates of ζ and ω_n for the two systems in a table and compare them to part C with part E as the nominal value.

Submission:

The lab memo will be due at the beginning of the next lab session. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, number of the ECP system (a diagram is preferable)
- A table comparing the estimated values of K, ζ, and ω_n using the two time domain techniques (note that the damping ratio may have a large error but the other two parameters should be close)
- All required figures with number and caption and they should be referenced in the text
- The discussion should include a compare and contrast of the two and four mass models and how the difference affects the gain, damping ratio and natural frequency.

Lab 2

Frequency Domain Analysis of a 1DOF Rectilinear System

Objective: Collect experimental data from a 1DOF rectilinear system
Identify the parameters of the 1DOF system using frequency domain analysis

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors

Prelab: Read the lab theory and procedure thoroughly.
Submit the answer to the following question on engineering paper the day before lab.

The waveform in Figure 1 is the magnitude Bode diagram for a mass-cart-spring system, estimate the static gain, K_o, maximum gain, K, natural frequency, ω_n and damping ratio, ζ. Write the transfer function, G(s) that models this system.

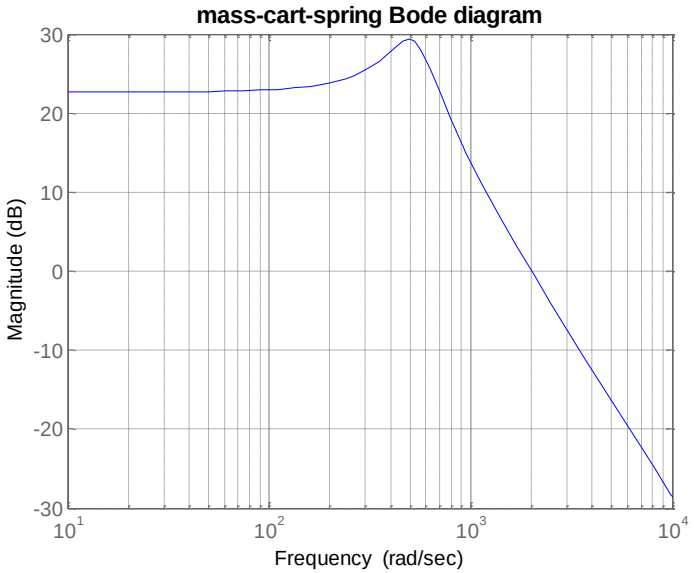


Figure 1: Mass-cart-spring magnitude Bode diagram

If there are any questions, please ask your instructor.

Theory:

Last week the time domain characteristics of the one degree of freedom rectilinear mass-cart-spring system were found. This week the frequency response of the same system will be determined. The frequency response involves constructing the magnitude portion of the Bode plot and fitting this measured frequency response to an expected transfer function to determine K, ζ and ω_n. The input to the system is a sinusoid, $x(t) = A\cos(\omega t)$, where A is measured in centimeters and the frequency, $\omega = 2\pi f$ is varied. After a transition period (transient response), the steady state output will be $y(t) = B\cos(\omega t + \theta)$, where B is also measured in cm. Since, the magnitude portion of the Bode plot will be constructed, we will ignore the phase angle, θ. Figure 2 presents the relationship between the input and output of a plant in the frequency domain.

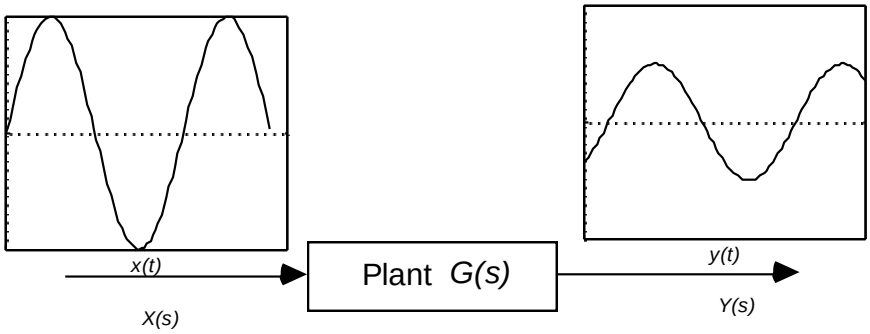


Figure 2: Physical Meaning of Frequency Response of a Linear System

The frequency response has maximum amplitude at resonance. The center frequency is ω_n at resonance and the amplitude is $20\log_{10}(K)$. Note that this K is **not** the static gain, K_o, found in lab 1, this is the maximum amplitude of the system response in the frequency domain. The bandwidth is the difference between the two frequencies where the maximum amplitude drops by 3 dB. The narrowness of the peak is defined by the quality factor (Q) or the damping ratio, $\zeta = BW/(2\omega_n)$. Since the plant is unknown, there will be some iteration involved in fitting the theoretical Bode plot to the experimental frequency response data to achieve the best fit. Since the resonance frequency is so important for finding system characteristics, it is very important to have several data points around this point. Finally, the Bode plot will be used to determine an appropriate transfer function model. Figure 3 presents the Bode diagram for the transfer function,

$$T(s) = \frac{7.73(1089)}{s^2 + 1.6s + 1089}$$

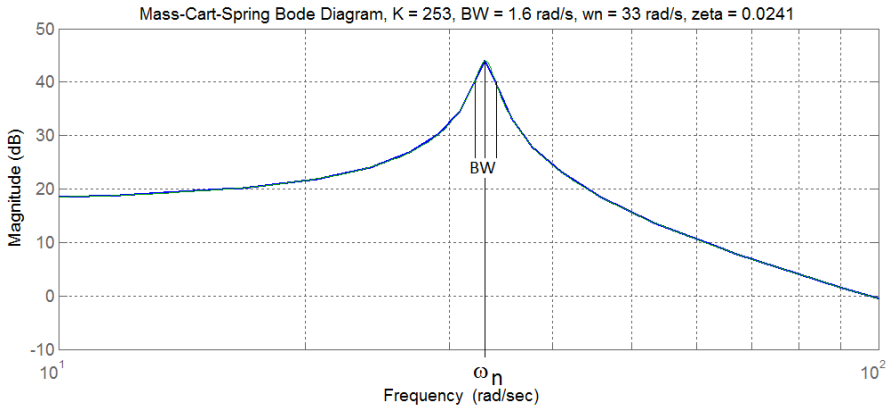


Figure 3: Mass-cart-spring underdamped step response

Procedure:

A. System Setup

1. Create a folder under Documents\ECE320\<your initials\Lab2
2. Copy all of the MATLAB files from the directory you made last week to the folder created in step 1.
3. Start MATLAB and change the current directory to the one created in step 1.
4. Set up the mass-cart-spring system with two masses that you built last lab with all of the same spring and mass values in the same location.
5. Every week you should run ECP3.2->Download Controller Personality File->Reset Controller, first.

B. Frequency Response

1. Modify **Model210_Openloop.mdl** so the input is a sinusoid. You may have to set the mode to **Normal**.
2. Set the frequency to 0.5 Hz, note that in MATLAB the frequency is rad/s so you need to input (2*pi*0.5). *The frequency must be entered in radians/sec!*
3. Set the amplitude of the sinusoid to a small value like 0.03 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 may vary with the frequency as you approach resonance.
4. Set the simulation stop time to run for between 1 and 10 seconds just enough so that the system can reach steady state.
5. Compile **Model210_Openloop.mdl,** connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.) To avoid compiling **Model210_Openloop.mdl** every time you make a change to the amplitude, frequency or simulation stop time, you should make them all variables such as *amp, simtime, freq* in the Simulink block diagram and then vary the values in the MATLAB command window.

6. 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.
7. Use Excel to create a data table and record the input frequency (f), input amplitude (A), and the output amplitude (B) when the system is in steady state.
8. Note that the output may not be a sine wave symmetric about the time axis and you will need to get the average of the positive and negative values. You can use the MATLAB file **get_B.m** to determine this value. After you run **get_B.m,** the average amplitude of *x1* will be displayed in the command window.
9. Repeat steps 1 through 8 for f = 1, 1.5, 2, 2.5, 3 ….7.5 Hz. Note that as the frequency gets larger, MATLAB may crash because of the data acquisition and you need to adjust the input sinusoid amplitude or slightly increase sample steps in the ECP Model 210 DAQ block to minimize this (see Figure 4). You should reset the system by connecting and running **ESCPDSPReset.mdl** to zero the initial conditions after each change in frequency and/or amplitude. You should only have to compile **ESCPDSPReset.mdl** the first time you use it.

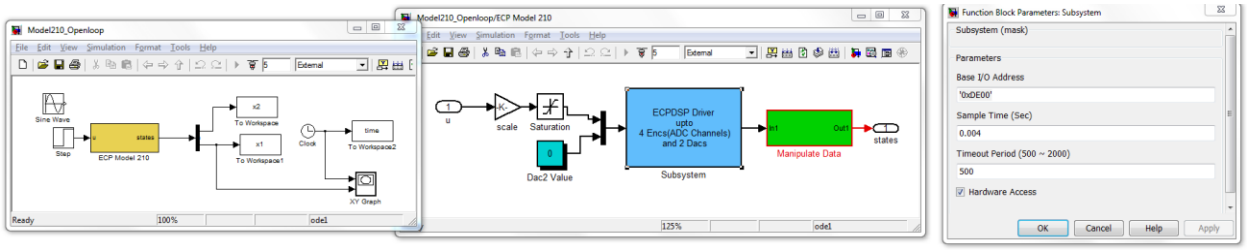


Figure 4: Adjusting the sample time in the ECP Model 210 DAQ block

10. Modify **process_data_1dof.m** m-file by entering the measured data of f, A, and B**.** Note that you can copy and paste it directly from Excel as long as you format the matrix appropriately.
11. At the MATLAB prompt, type **data = process_data_1dof**

C. Fit your model to the transfer function

1. 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, found in Lab 1
   - ω_n, the estimated natural frequency from the log decrement analysis, found in Lab 1
   - ζ, the estimated damping ratio from the log decrement analysis (i.e. `model_1dof(data, K, wn, zeta)`);
2. The program **model_1dof.m** will produce the following:
   - A graph indicating the fit of the identified transfer function to the measured data
   - 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. See Figure 5 for an example of the experimental Bode magnitude plot with theoretical transfer function.

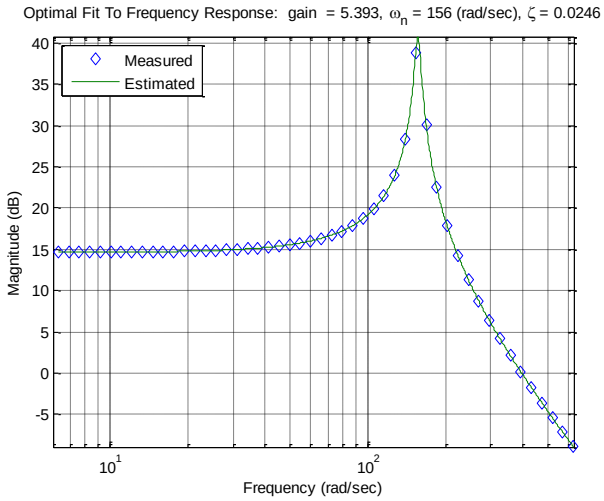


Figure 35: Experimental Bode magnitude plot with best fit theoretical transfer function

D. Improve the model

1. Add **at least** 5 additional data points close to the resonant peak of the transfer function. This is necessary to obtain the best model possible. Note that the horizontal axis is in rad/s so you need to determine the frequency in Hz near resonance to enter into the simulation.
2. Identify the 5 frequencies and repeat parts B and C of the procedure to improve the model. Remember you are near resonance so you need to make the amplitude small around 0.01 and be ready to stop the simulation if necessary.
3. Save the **process_data_1dof.m** file as **Lab2_<your initials>_<#ofmasses>_data.m.**
4. Save the **state_model_1dof.mat** file as **Lab2_<your initials>_<#ofmasses>_model.mat.**
5. These files contain the data and state variable model for the mass-cart-spring system and will be used in all future labs. If you subsequently type **load state_model_1dof**, you will load these matrices into your workspace.
6. REPEAT PARTS A – D for the 4 masses on the cart system that you built in Lab 1. Make sure to use the same masses in the same order as you did in Lab 1.
7. You must include the improved graphs of the identified transfer function to the measurement data for both system set ups in your lab memo.
8. Your memo should also include the state equation matrices and transfer function for the 2 plants.

Submission:

The lab memo will be due at the beginning of the next lab session. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, include ECP system number
- All required figures with number and caption and they should be referenced in the text
- All required data in tables with error analysis referenced in the text, if necessary
- The discussion should compare and contrast the results of the time domain analysis to the frequency domain analysis.
- The discussion should also compare and contrast the system parameters for the two mass versus the four mass system

Lab 3
Analysis of a 2DOF Rectilinear System

Objective: Collect experimental data from a 2DOF rectilinear system
Identify the parameters of the 2DOF system using frequency and time domain analysis

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment if necessary.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors

Prelab: Read the lab theory and procedure thoroughly. Submit the answer to the following question on engineering paper the day before lab.

Figure 1 is the model of a 2DOF mass-cart-spring damper system. Derive the state-space representation and transfer function for the translational mechanical system where $x_1(t)$ is the output. Assume that $m_1 = m_2 =$ 1 kg, $k_1 = 1$ N/m and $b_1 = b_2 = 1$ N-s/m.

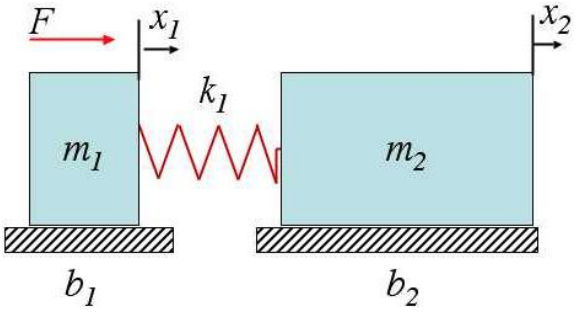


Figure 1: 2DOF mass-cart-spring-damper system

If there are any questions, please ask your instructor.

Theory:

The fundamental theory for frequency and time domain analysis of a 2DOF rectilinear system is the same as for a 1DOF system. The two degrees of freedom means that two of the carts are able to move. Similar to the 1DOF system, each of the carts will have a **resonant frequency** where the Bode magnitude plot reaches a local maximum. The frequency response analysis involves finding the magnitude of the Bode diagram as the frequency is varied and then matching it to a transfer function model similar to the prior lab. The time domain analysis,

involves using a step response and initial condition response to find the characteristic parameters. If we assume the surface is frictionless, the equations of motion for the 2 DOF system are

$$\ddot{x}_1 + 2\zeta_1\omega_1\dot{x}_1 + \omega_1^2 x_1 = \frac{k_1}{m_1}x_2 + \frac{1}{m_1}F \quad (1)$$

$$\ddot{x}_2 + 2\zeta_2\omega_2\dot{x}_2 + \omega_2^2 x_2 = \frac{k_1}{m_2}x_1 \quad (2)$$

If there is no applied force (F = 0) and the second cart is fixed in place ($x_2 = 0$), the characteristic equation of the first cart (equation (1)) becomes

$$s^2 + 2\zeta_1\omega_1 s + \omega_1^2 = 0 \quad (3)$$

If the second cart is free to move and the first cart is fixed in place ($x_1 = 0$), the characteristic equation of the second cart (equation (2)) becomes

$$s^2 + 2\zeta_2\omega_2 s + \omega_2^2 = 0 \quad (4)$$

Thus, the log decrement can be used to estimate ζ_1, ζ_2, ω_1, and ω_2.

Procedure:

A. System Setup

1. Create a folder under Documents\ECE320\<your initials\Lab3
2. Download the **LAB3.rar** files from the Angel class folder to the directory you created in step 1.
3. Lock the third cart from the motor in place.
4. For the frequency response analysis, the first two carts in the system should move. In the time domain analysis, each of the first two carts should move alone for the log decrement or time domain analysis.
5. This week you will decide how to set up the 2DOF system based upon the following criteria:
   - 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.
6. Make sure to record a detailed description of the system set up with mass numbers, locations, spring numbers and locations, damper to include in your lab manual. Creating a diagram to include in your lab memo would be advisable.

B. Log-Decrement (time domain analysis)

1. Reset the system using **ECPDSPresetmdl.mdl**.
2. Modify **Model210_Openloop.mdl** so the input has zero amplitude.
3. Compile **Model210_Openloop.mdl** and connect to the ECP system. (The mode should be External.)
4. Lock the second cart and displace the first cart and hold it.
5. Click **start (play) Model210_Openloop.mdl** and let the cart go. The XY graph of the initial condition response should be opened on the screen. Set the simulation stop time so that it runs until the system reaches steady state.
6. Run the m-file **log_dec.m** and select the cart that moved, if it is cart 1 the position variable is *x1*, if it is cart 2 the position variable is *x2*, and the time is labeled *time*.
7. Select **Cart 1**
8. Select **Load IC (initial condition) Response**
9. Select **Plot IC Response** to plot the initial condition response
10. Choose to identify the positive 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.
11. 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 (i.e. 1 and 2 or 2 and 3).
12. Select **Estimate Parameters** to get the initial estimates of ζ_1 and ω_1.
13. Select **Make Log-Decrement Figure** to get a plot and summary of the results.
14. You must include the estimates of the natural frequency and damping ratio and the **log_dec** figure in the lab memo.
15. Repeat part B steps 1 – 14 with cart 1 locked and cart 2 free to move.

C. Estimating the gains, K_1 and K_2

1. Be sure both carts are free to move, the cart farthest from the motor is locked in place.
2. Reset the system using **ECPDSPresetmdl.mdl**.
3. Modify **Model210_Openloop.mdl** so the input is a step. Set the amplitude to something small, like 0.01 or 0.02 cm.
4. Compile **Model210_Openloop.mdl** and connect to the ECP system. (The mode should be External.)
5. 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. If the first cart does not move by at least 1.5 cm then you should increase the input amplitude.
6. To determine the steady-state value of the cart position, don't use the graph but view the x_1 and x_2 variables in the MATLAB workspace.
7. Estimate the static gains by using the following formulas

$$K_1 = \frac{x_{1ss}}{A} \qquad K_2 = \frac{x_{2ss}}{A} \tag{5}$$

8. Repeat steps 1 through 7 for at least 3 different input amplitudes and average the estimated gains. You must include the estimated gains in your lab memo.

D. Frequency Domain Analysis

Similar to Lab 2, you will construct the magnitude portion of the Bode plot for the 2DOF rectilinear system and fit the measured frequency response to the frequency response of the expected transfer function to determine the system parameters. For each frequency, $\omega = 2\pi f$, there is an input $u(t) = A\cos(\omega t)$, where A is measured in centimeters. After the transient 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 you will create the magnitude portion of the Bode plot, the phase angles (θ_1 and θ_2) will be ignored.

1. Reset the system using **ECPDSPresetmdl.mdl**.
2. Modify **Model210_Openloop.mdl** so the input is a sinusoid. You may have to set the mode to Normal.
3. Set the frequency to 0.5 Hz and small amplitude for the sinusoid such as 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.
4. Compile **Model210_Openloop.mdl**, connect **Model210_Openloop.mdl** to the ECP system. (The mode should be External.)
5. 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.
6. Use Excel to create a data table to record the input frequency (f), the amplitude of the input (A), and the amplitude of the outputs (B_1 and B_2) when the system is in steady state. Use the program **get_Amp2.m** to get the amplitudes accurately. Be sure the plot from **get_Amp2** shows the system in steady state. As always, review the code and make sure that you understand how it works before you use it!! Sometimes the code may have to be modified to be applicable to your needs.
7. Enter the values of f, A , B_1 and B_2 into the program **process_data_2dof.m** under the *measured_data* variable
8. Repeat steps 1 through 7 for the following frequencies: 1, 1.5, 2, 2.5, 3, 3.5, …. 6.5,7, 7.5 Hz

E. Create and Improve the Model

1. At the Matlab prompt, type **data = process_data_2dof;**
2. Run the program **model_2dof.m**, which has the following ten arguments:

- data, the measured data as determined by **process_data_2dof.m**
- the estimated value of K_2, the gain of the second cart system
- 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 ζ_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 ζ_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)

(i.e. **model_2dof(data, K_2, ω_a, ζ_a, ω_b, ζ_b, ω_1, ζ_1, ω_2, ζ_2)**)

3. 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
   - A graph indicating the fit of the identified transfer function to the measured data for the second cart
   - 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.
4. You need to be sure you have 5 points close to the resonant peaks and nulls of the transfer functions. After creating the improved model, include the final graph of the curve fit of the measured data to the identified transfer function in your lab memo for the first and second cart.
5. Save the **state_model_2dof.mat** file as **Lab3_<your initials>_2dofmodel.mat**
6. Save the **process_data_2dof.m** file as **Lab3_<your initials>_data.m**
7. Your memo should include an error analysis of the gains, damping ratio and natural frequencies for cart 1 and cart 2 with frequency domain analysis as the nominal value.
8. Your memo should also include the state equation matrices and transfer function for the plants.

Submission:

The lab memo will be due at the beginning of the next lab session. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner

- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, include ECP system number
- All required figures with number and caption and they should be referenced in the text
- All required data in tables with error analysis referenced in the text, if necessary
- The discussion should include a comparison of the multiple methods used to find the gain, damping ratio and natural frequency

Lab 4
Model Matching

Objective: Use model matching to select a controller for a 1DOF and 2DOF system
Use the model to design the control system to meet certain design requirements such as settling time, steady state error and percent overshoot
Collect data to compare the designed response to the actual response

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment if necessary.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors

Prelab: Read the lab theory and procedure thoroughly. Submit the answer to the following question on engineering paper at the beginning of lab.

Assume that the closed loop system in Figure 1 has a plant, $G_p(s)$, that represents the mass-cart-spring system that you will control in lab. You will design the controller, $G_c(s)$ to meet certain system characteristics.

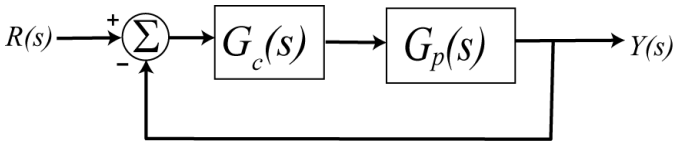


Figure 1. Closed-loop control system

The transfer function for the closed loop system is

$$T(s) = \frac{Y(s)}{R(s)} = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)}$$

One method to select the controller is to make the closed loop system match a transfer function hence, the name "model matching". Assume that the desired closed loop transfer function is $G_o(s)$. The plant and closed loop transfer function can be written in terms of numerators and denominators as

$$G_o(s) = \frac{N_o(s)}{D_o(s)} \quad G_p(s) = \frac{N_p(s)}{D_p(s)}$$

1. Show that if the controller is stable and proper it can be written as

$$G_c(s) = \frac{N_o(s)D_p(s)}{N_p(s)\left[D_o(s) - N_o(s)\right]}$$

2. *If the plant is $G_p(s) = \frac{1}{s+1}$, design the controller, $G_c(s)$, so that the closed loop control system matches a second - order **ITAE** (Integral of Time and Absolute Error) optimal system with the following closed loop transfer function,*

$$G_0(s) = \frac{\omega_0^2}{s^2 + 1.4\omega_0 s + \omega_0^2}$$

Note in your final answer that there is a pole/zero cancellation between the controller and the plant and there is a pole at zero in the controller.

3. *If the plant is $G_p(s) = \frac{1}{s+1}$, design the controller, $G_c(s)$, so that the closed loop control system matches a third-order **deadbeat** system with the following closed loop transfer function,*

$$G_0(s) = \frac{\omega_0^3}{s^3 + 1.90\omega_0 s^2 + 2.20\omega_0^2 s + \omega_0^3}$$

Note in your final answer that there is a pole/zero cancellation between the controller and the plant and there is a pole at zero in the controller.

If there are any questions, please ask your instructor.

Theory:

Model matching is the process of selecting a controller such that a closed loop control system matches a given model to satisfy certain design characteristics. For this lab, the two closed loop models are the ITAE and deadbeat systems. A **deadbeat response** is a system with a rapid response, minimal overshoot, and zero steady-state error for a step input. A system is considered an **optimum control system** when the system parameters are adjusted so that the index reaches an extremum, commonly a minimum value. A **performance index** is a quantitative measure of the performance of a system and a suitable one is the integral of the square of the error, ISE defined as

$$ISE = \int_0^T e^2(t)dt$$

Another performance criterion is the integral of the absolute magnitude of the error, IAE defined as

$$IAE = \int_0^T |e(t)|dt$$

To reduce the contribution of the large initial error to the value of the performance integral, as well as to emphasize errors occurring later in the response. The integral of time multiplied by the absolute error, ITAE is define by

$$ITAE = \int_0^T t|e(t)|dt$$

Coefficients of the closed loop transfer function that minimize the ITAE performance criterion for a step input have been determined and are summarized in Table 1. Note that the transfer function has n poles and no zeros. The coefficients for the deadbeat system have also been determined in Table 1.

n	ITAE	Deadbeat
2	$s^2 + 1.4\omega_o s + \omega_o^2$	$s^2 + 1.182\omega_o s + \omega_o^2$
3	$s^3 + 1.75\omega_o s^2 + 2.15\omega_o^2 s + \omega_o^3$	$s^3 + 1.9\omega_o s^2 + 2.20\omega_o^2 s + \omega_o^3$
4	$s^4 + 2.1\omega_o s^3 + 3.4\omega_o^2 s^2 + 2.7\omega_o^3 s + \omega_o^4$	$s^4 + 2.2\omega_o s^3 + 3.5\omega_o^2 s^2 + 2.8\omega_o^3 s + \omega_o^4$

Table 1: Optimum coefficients of T(s) based on the Deadbeat and ITAE criterion for a step input

One goal of this lab is to use model matching to select a controller so that the overall system meets certain design criteria. If the closed loop transfer function is known such as *Deadbeat* or *ITAE* then the controller can be found from the prior equation. However, there are some limitations on this type of model matching given by the following set of rules

1. The controller is a proper rational transfer function
2. The controller is stable
3. The degree of the closed loop denominator –the degree of the closed loop numerator ≥ the degree of the plant denominator – the degree of the plant numerator
4. All of the right half plane zeros of the plant must also be zeros in the closed-loop transfer function
5. The closed loop transfer function is stable

We will assume that all of these conditions are satisfied for the lab procedure implementation.

Procedure:

A. File Setup

1. Create a folder under Documents\ECE320\<your initials\Lab4
2. Copy **all** of the MATLAB files from Lab 3 to the folder created in step 1
3. Copy the 3 1DOF MATLAB model files (Lab2_<your initials>_0_model.mat., Lab2_<your initials>_2_model.mat., Lab2_<your initials>_4_model.mat.) from Lab 2 to the folder created in step 1

B. Model Matching of a 1DOF system

1. You will design a closed loop control system 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%
2. Set up the 1DOF rectilinear system with two masses just as you did in Lab 2, refer to your memo or system diagram.
3. Modify **closedloop_driver.m** to read in the lab 2 model file for a 1DOF rectilinear system with two masses (i.e. **load Lab2_<your initials>_0_model**)
4. Use the *second order zero position error ITAE system* and vary ω_o until you meet the design specifications with the simulation (**closedloop_driver.m**). This means to uncomment the MATLAB code with this transfer function (Go) and comment out the rest. Note that 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. At this point all of the variables you should need are in your current MATLAB workspace. You should run the simulation until the system is clearly at a steady state value for at least one second. Be sure you do not reach the limiter on the control effort. When the control effort saturates, this is where the motor must provide more torque than it can produce. This is to be avoided at all costs! Figure 2 presents an example of a simulation output from the closed loop driver.

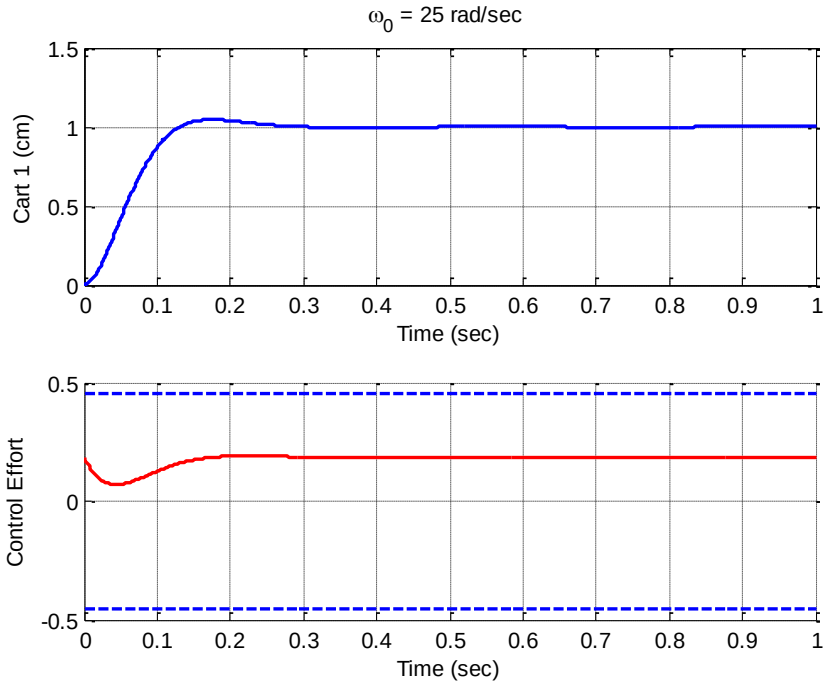


Figure 2: Closed Loop Driver Simulation Results

5. Open **Model210_Closedloop.mdl** and be sure the Base Address is correct for your system.
6. Compile **Model210_Closedloop.mdl**, the parameters this file needs are in the workspace after **closedloop_driver.m** runs. The yellow block is what makes the ECP system work, and replaces your model of the system with the real system.
7. 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.
8. Reset the system by running **ECPDSPReset.mdl**, connect **Model210_Closedloop.mdl** to the system and then run it.
9. 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. Figure 3 presents an example of the results of the **compare1.m** program. Try your best to get similar results.
10. *NOTE: You should be prepared that since the controller is designed based upon your model, it can only be as accurate as your model. Sometimes it may be difficult to meet the design specifications. If this happens, there will be some tradeoffs and you should do your best to meet one or two (%OS, e_{ss}, T_s) and come close to meeting the other.*

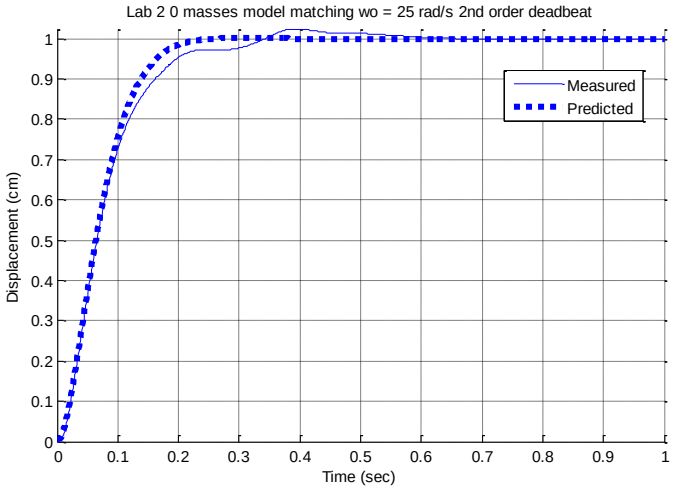


Figure 3: Comparison of closed loop simulation and actual results

11. Repeat steps 4 through 9 for the *third order zero position error ITAE system*, *second order deadbeat system*, and *third order deadbeat system*.
12. Repeat steps 2 through 11 for the lab 2 model file for a 1DOF rectilinear system with 4 masses.

C. Model Matching of a 2DOF system

1. You will design a closed loop control system to meet the following design requirements:
   - Settling time less than 1.0 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%
2. Set up the 2DOF rectilinear system exactly as you did in Lab 3, refer to your memo or system diagram.
3. Modify **closedloop_driver.m** to read in the lab 3 model file for a 2DOF rectilinear system (i.e. **load Lab3_<your initials>_2dofmodel**)
4. Use the *fourth order zero position error ITAE system* and vary ω_o until you meet the design specifications with the simulation (**closedloop_driver.m**). Note that 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. At this point all of the variables you should need are in your current MATLAB workspace. You should run the simulation until the system is clearly at a steady state value for at least one second. Be sure you do not reach the limiter on the control effort. When the control effort saturates, this is where the motor must provide more torque than it can produce. This is to be avoided at all costs!
5. Open and compile **Model210_Closedloop.mdl**.

6. Reset the system by running **ECPDSPReset.mdl**, connect **Model210_Closedloop.mdl** to the system and then run it.
7. 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.
8. Repeat steps 4 through 7 for the *fourth order deadbeat system*.

Your memo should compare the difference between the predicted response (from the model) and the real response for each of the systems. Your memo should also compare the results and performance of the different types of systems (2^{nd}, 3^{rd}, 4^{th}, deadbeat or ITAE).

Submission:

The lab memo will be due at the beginning of the next lab session. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, include ECP system number
- All required figures with number and caption and they should be referenced in the text
- All required data in tables with error analysis referenced in the text, if necessary

Lab 6
Root Locus

Objective: Determine the root locus for a system plant by varying proportional controller gains
Describe the relationship between controller gains the damping ratio and damped frequency

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment if necessary.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors

Prelab: Read the lab theory and procedure thoroughly. Submit the answer to the following question on engineering paper in class the day before lab.

Assume that the closed loop system in Figure 1 has a plant, $G_p(s)$, that represents the mass-cart-spring system that you will control in lab and a proportional controller, $G_c(s) = K$.

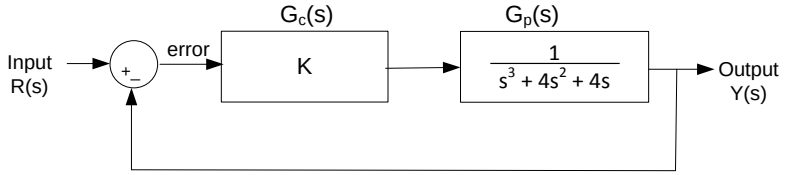


Figure 1. Closed-loop control system

1. *Use the Routh-Hurwitz criterion to determine the range of K for system stability*
2. *Create a plot of the closed loop poles on the s-plane as the controller gain K is varied from 0 to 18.*
3. *Use the* `rlocus(tf(num,den))` *MATLAB command to create the root locus plot for the open loop transfer function in Figure 1. Note that you must use the open loop transfer function with K = 1 in the MATLAB command.*

If there are any questions, please ask your instructor.

Theory:

Root locus analysis is a technique for plotting the closed loop poles of a control system as the controller gain, K, is varied using the open loop transfer function. This plot can be used to quickly identify the ranges for system stability as well as pole locations. In addition, the root locus can be used to design a system to meet certain design specifications.

Procedure:

A. File Setup

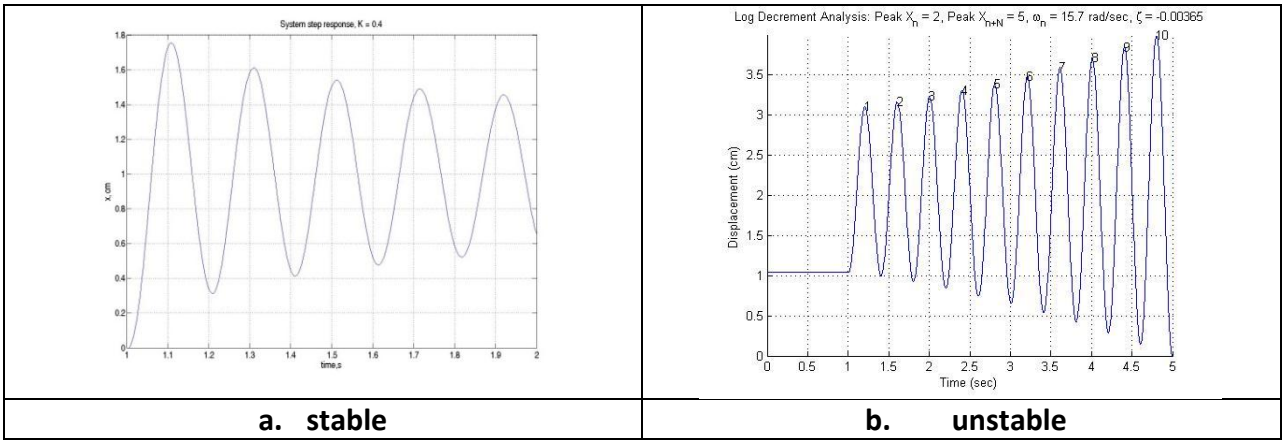
1. Create a folder under Documents\ECE320\<your initials\Lab6
2. Copy **all** of the MATLAB files from Lab 1 to the folder created in step 1
3. Download the **Model210_PControl.zip** file from the Angel course folder to the folder created in step 1

B. System Setup

1. Remove all springs from the system and lock all of the carts except for the first one.
2. Put four 500 gram masses on the first cart.
3. Connect the damper and insert the plug with 2 full turns.

C. Step Response

1. Reset the system by running **ECPDSPReset.mdl**
2. Use the Simulink model, **Model210_PControl.mdl** to get the step response data. First, find the largest gain value to cause marginal stability or slight instability then choose twenty values below this critical gain. **Be careful when you get close to the critical gain; be prepared to stop the simulation immediately so that you don't damage the system!** As a point of reference, for my system the range of values was between 0.02 and 0.5. Figure 1 provides two examples of the step response right before and right after it becomes unstable.
3. You should set the value of K in the MATLAB command window and then run **Model210_PControl.mdl.** You should run the simulation until the system reaches steady state most likely between 0 and 15 seconds.
4. Save the workspace after each step response so that you have access to the *x1* and *time* variables for later analysis.
5. Make sure to use **ECPDSPReset.mdl** to reset the system after each data set.



a. stable **b. unstable**

Figure 1: System Sample Step Responses

D. System Analysis

1. Plot all of the step response on two sets of axes with no more than 5 step responses per axes. Make sure the plot has a title, axes labeled and a legend to make it clear which gain corresponds to each step response. See Figure 2 for an appropriate step response graph. You should include these figures in your lab memo submission.

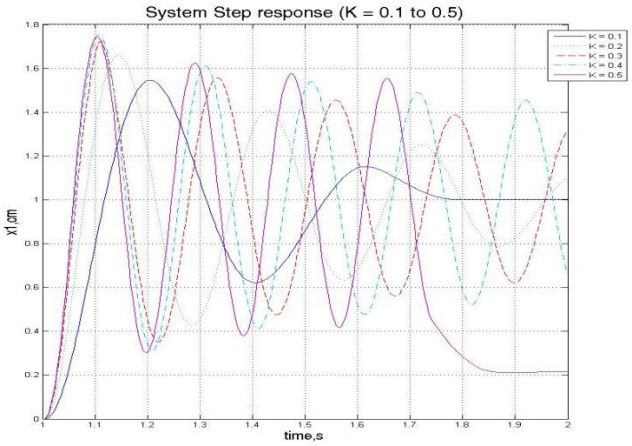


Figure 2: Step responses for K = 0.1 to 0.5

2. For each controller gain, K, use the log-decrement method by running the m-file, **log_dec.m** to determine the damping ratio and natural frequency.
3. Use Excel to create a table of controller gains (K), damping ratio (ζ), natural frequency (ω_n), damping factor (σ_d) and damped frequency (ω_d).
4. Create a plot of the damping ratio (ζ) versus gain to include your memo and describe the relationship between the two (see Figure 3).
5. Create a plot of the damped natural frequency (ω_d) versus gain to include in your memo and describe the relationship between the two (see Figure 3).

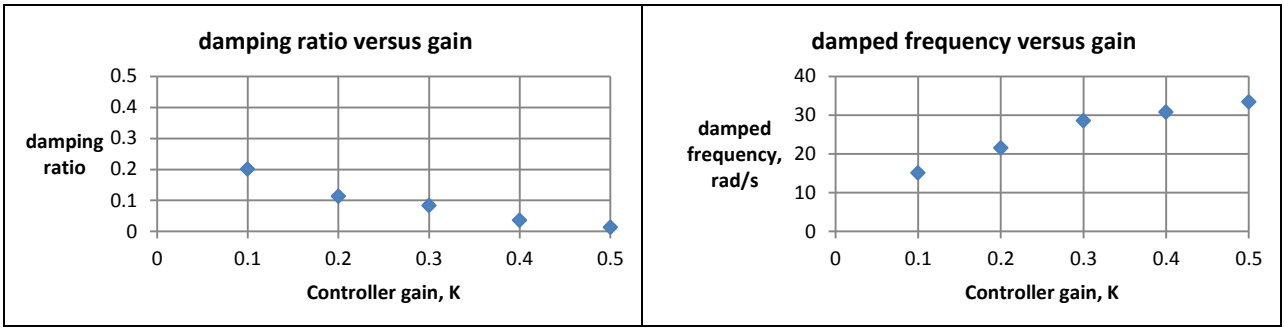


Figure 3: Damping ratio and damped natural frequency versus gain

6. Recall that the closed-loop poles of the system are given by $-\sigma_d \pm j\omega_d = -\zeta\omega_n \pm \omega_d$. Create a plot of the root locus of the experimentally determined closed loop poles as a function of controller gain (see Figure 4). This plot should be included in your lab memo submission.

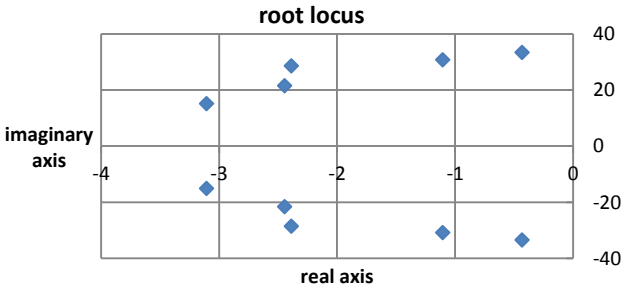


Figure 4: Experimental Root Locus

7. Save all of your 1DOF and 2DOF state models to a thumb drive or email them to yourself because you need them to complete prelab 7.

Submission:

The lab memo will be due at the beginning of the next lab session. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, include ECP system number
- All required figures with number and caption and they should be referenced in the text
- All required data in tables with error analysis referenced in the text, if necessary

Lab 7
PID control

Objective: Design a PID and PD controller for the one degree of freedom system to satisfy certain system characteristics
Evaluate the performance differences between a PD versus PID controller

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment if necessary.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors

Prelab:

*Read the lab theory and procedure thoroughly. Review the **Root Locus for Controller Design** and **Introduction to Sisotool** document posted in the Angel Course Folder. Use the instructions in this document to practice using **sisotool** not for submission. However, you should submit the following in class the day before lab.*

1. You and your lab partner should select different state variable models from the 1 degree of freedom systems created in Lab 2.
1. Use the **closedloop_driver.m** file from Lab 2 to load the correct state model into the system. This driver simulates the designed controller in the **closedloop.mdl**.
2. Comment out all of the other controllers and add the following lines:

```
kp = 0.5;            % dummy value for proportional gain
ki = 5.0;      % dummy value for integral gain
kd = 0.01;           % dummy value for derivative gain
Gc = tf(kp,1) + tf(ki,[1 0]) + tf([kd 0],[1/50 1]);
Go = feedback(Gc*Gp,1);
```

3. Note that we have modified the derivative controller so that it is in series with a one pole lowpass filter with pole at 50 (about 8 Hz). This will help smooth out the derivatives and avoid the noise at the beginning of the step.
4. You will need to be able to determine the PID controller coefficients from the controller. This is easiest done by equating coefficients. For example, if the PID controller is given by

$$G_c(s) = \frac{a(s^2 + bs + c)}{s}$$

show that the coefficients are determined by $k_d = a$, $k_p = ab$, and $k_i = ac$.

5. Use Matlab's **sisotool** to design **two** PID controllers (one with complex conjugate zeros, one with real zeros) for your system. Initially limit your gains as you will in the lab to

$$\begin{aligned} k_p &\leq 0.5 \\ k_i &\leq 5 \\ k_d &\leq 0.01 \end{aligned}$$

6. Your resulting design must have a settling time of 1.0 seconds or less and must have a percent overshoot of 25% or less. Note that **sisotool** defaults to an input of 1 which is acceptable for design purposes.
7. If you don't know how to get the correct plant transfer function, run **closedloop_driver.m** (with the correct model file) and it will put the correct transfer function Gp(s) into your Matlab workspace.
8. Implement the PID controller for the gains found in **sisotool** by putting them in the **closedloop_driver.m** file. Use a step with amplitude 0.5 cm in your **closedloop_driver.m** file.
9. Run the **closedloop_driver.m,** plot the control effort for approximately 0.2 seconds since the control effort is usually largest near the initial time. If your control effort reaches its limits, you need to go back to **sisotool** and modify your designs. See Figure 1 for an example of bad control effort.

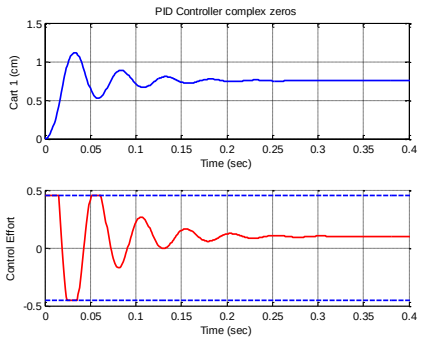


Figure 1: Bad Control Effort Example

10. If your control effort is not near the limit, you can increase the gains, particularly the derivative gain and simulate again. You can repeat this process until you are happy with the design.
11. Run your simulations for 2.0 seconds. Plot both the system output (from 0 to 2 seconds) and the control effort (from 0 to 0.2 seconds). Put a title on your plot to identify kp, ki, and kd.

Look at previous code to determine how to do this. Submit your plot as part of your prelab submission.

12. Our new transfer function has introduced finite zeros into the closed loop transfer function. We now want to use a **dynamic prefilter** to eliminate these zeros, so long as they are in the left half plane. We also need $G_{pf}(0)=1$. Hence we have

$$G_{pf}(s)=\frac{D_o(0)}{N_o(s)}$$

13. You should set the dynamic prefilter in MATLAB by commenting out your old prefilter code and creating code to implement the dynamic prefilter. This will cancel out the zeros of the closed loop system. Your numerator polynomial for the closed loop transfer function, No(s) should be second order. You can use the following code to create the dynamic prefilter:

```
[num_Go,den_Go] = tfdata(Go,'v');
num_Gpre = den_Go(end);
den_Gpre = num_Go;
```

14. Run your simulation again with the dynamic prefilter and compare the results and discuss similarities and differences.
15. You should submit 4 plots for your prelab including PID real zeros with and without the dynamic prefilter, PID complex zeros with and without the dynamic prefilter.

If there are any questions, please ask your instructor.

Theory:

PID control can be used to adjust a system to meet given steady-state and transient characteristics. Proportional control as well as a prefilter can be used to adjust the steady-state error. Derivative control adjusts the transient characteristics such as settling time and rise time. Integral control adjusts the steady-state characteristics such as stability and steady-state error. Figure 1 provides a closed loop transfer function with a prefilter, which is used to condition an input signal by changing the units or to fix the final value of the output.

$$T(s)=\frac{Y(s)}{R(s)}=\frac{G_{pf}G_c(s)G_p(s)}{1+\ G_{pf}G_c(s)G_p(s)}$$

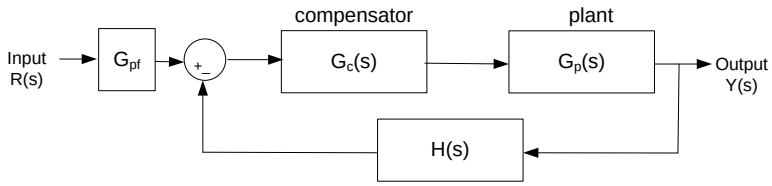


Figure 1: Closed loop control system with prefilter

Procedure:

A. File Setup

1. Create a folder under Documents\ECE320\<your initials>\Lab7
2. Copy **all** of the MATLAB files from Lab 2 to the folder created in step 1
3. Modify **closedloop_driver.m** to read in the correct model file. You may have to copy this model file to the current folder.

B. System Setup

1. Start by building the one degree of freedom mass-cart-spring system with two masses exactly as you did in lab 2
2. For each of the one-degree of freedom systems, you will try to adjust the controller parameters until you have achieved the following:
   - Settling time less than 1.0 seconds.
   - Steady state error less than 0.1 cm for a 1 cm step
   - Percent Overshoot less than 25%

C. PID Control (complex conjugate zeros)

1. Design a PID controller with complex conjugate zeros using **sisotool** to meet the design specifications. You should have already completed some of these designs in **sisotool** for your prelab. Use a **constant prefilter** (i.e., the number 1).
2. Implement the correct gains into the **closedloop_driver.m** which simulates the **closedloop.mdl**.
3. Run **closedloop_driver.m** to simulate the system for 1.5 seconds. If the design constraints are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)
4. Reset the system by running the **ECPDSPReset.mdl**
5. Compile and run **Model210_Closedloop.mdl** and connect to the system to run the actual hardware response
6. Use the **compare1.m** 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. Be sure to include an informative title, x and y labels and the values of K_p, K_i and K_d in your memo.
7. Change the **prefilter** to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation and compare the predicted with the measured response. You also need to include this graph in your memo. Be sure to include an informative title, x and y labels and the values of K_p, K_i and K_d in your memo.
8. Repeat steps 1 through 7 for the one degree of freedom system with four masses.

D. PID Control (real zeros)

2. Design a PID controller with real zeros using **sisotool** to meet the design specifications. You should have already completed some of these designs in **sisotool** for your prelab. Use a **constant prefilter** (i.e., the number 1).
3. Implement the correct gains into the **closedloop_driver.m.**
4. Run **closedloop_driver.m** to simulate the system for 1.5 seconds. If the design constraints are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)
5. Reset the system by running **ECPDSPReset.mdl**
6. Compile and run **Model210_Closedloop.mdl** and connect to the system to run the actual hardware response
7. Use the **compare1.m** 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. Be sure to include an informative title, x and y labels and the values of K_p, K_i and K_d in your memo.
8. Change the **prefilter** to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation and compare the predicted with the measured response. You also need to include this graph in your memo. Be sure to include an informative title, x and y labels and the values of K_p, K_i and K_d in your memo.
9. Repeat steps 1 through 7 for the one degree of freedom system with four masses.

E. PD Control

1. Design a PD controller with a real zero using **sisotool** to meet the design specifications. You may not be able to meet the percent overshoot constraint but do the best you can. You can use a **constant prefilter** for the design but the value probably won't be one. Use the constant prefilter to correct the steady-state error.
2. Implement the correct gains into the **closedloop_driver.m.**
3. Run **closedloop_driver.m** to simulate the system for 1.5 seconds. If the design constraints are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)
4. Reset the system by running **ECPDSPReset.mdl**
5. Compile and run **Model210_Closedloop.mdl** and connect to the system to run the actual hardware response
6. Use the **compare1.m** 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. Be sure to include an informative title, x and y labels and the values of K_p, K_i and K_d in your memo.
7. Change the **prefilter** to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation and compare the predicted with the measured

response. You also need to include this graph in your memo. Be sure to include an informative title, x and y labels and the values of K_p, K_i and K_d in your memo.

8. Repeat steps 1 through 7 for the one degree of freedom system with four masses.

*Your memo should include **six** graphs for each of the 1 dof systems you used (two different PID controllers and one PD controllers with and without dynamic prefilters.) Be sure to include the values of k_p, k_i, and k_d and whether the PID controller had real zeros or complex conjugate zeros in the captions for each figure. 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. How does the use of a dynamic prefilter change the response?*

Submission:

The lab memo will be due at the beginning of the next lab session. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, include ECP system number
- All required figures with number and caption and they should be referenced in the text
- All required data in tables with error analysis referenced in the text, if necessary
- The discussion should include a comparison and contrast of the performance of the different PID and PD controllers with and without real or complex conjugate zeros with and without dynamic prefilters
- The discussion should also compare the designed controllers and response for the two and four mass systems

Lab 8
PID control

Objective: Control the one degree of freedom system previously modeled using PI-D and I-PD controllers with and without dynamic prefilters
Evaluate the performance differences between PI-D and I-PD controllers

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment if necessary.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors

Prelab:
Read the lab theory and procedure thoroughly and complete the following prelab for submissions in class the day before lab.

For each of the one degree of freedom systems, try to adjust the parameters until you have achieved the following

- Settling time less than 1.0 seconds.
- Steady state error less than 0.1 cm for a 1 cm step
- Percent Overshoot less than 25%

As a start, you should initially limit your gains as follows:

$$\begin{aligned} k_p &\leq 0.5 \\ k_i &\leq 5 \\ k_d &\leq 0.01 \end{aligned}$$

1. Use **sisotool** to determine the values of k_p, k_i and k_d for the PI-D and I-PD controllers by first setting the control architecture to get the proper configuration (see Figure 1).

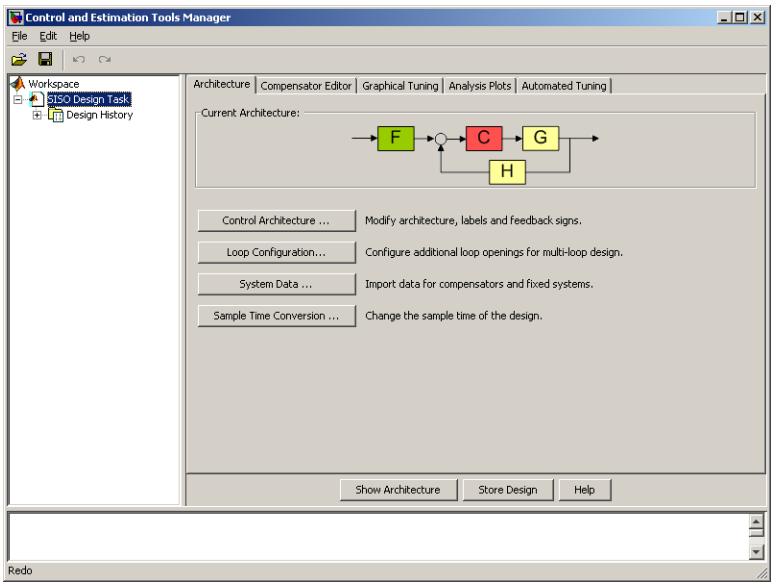


Figure 1: Control and Estimation Tools Manager

2. Select the correct architecture and make sure that **sisotool** uses negative feedback for both loops (see Figure 2).

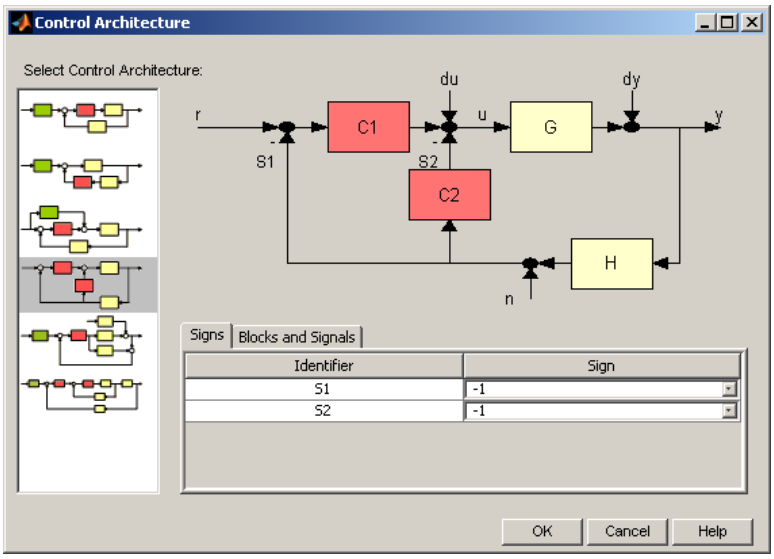


Figure 2: Control Architecture

3. Click OK, and then click **Loop Configuration** on the **Control and Estimation Tools Manager**
4. On the drop down menu select **Open Loop-Output of C1** and check **Open** for **Output of Block C2.**
5. On the drop down menu select **Open Loop-Output of C2** and check **Open** for **Output of Block C1** (see Figure 3)

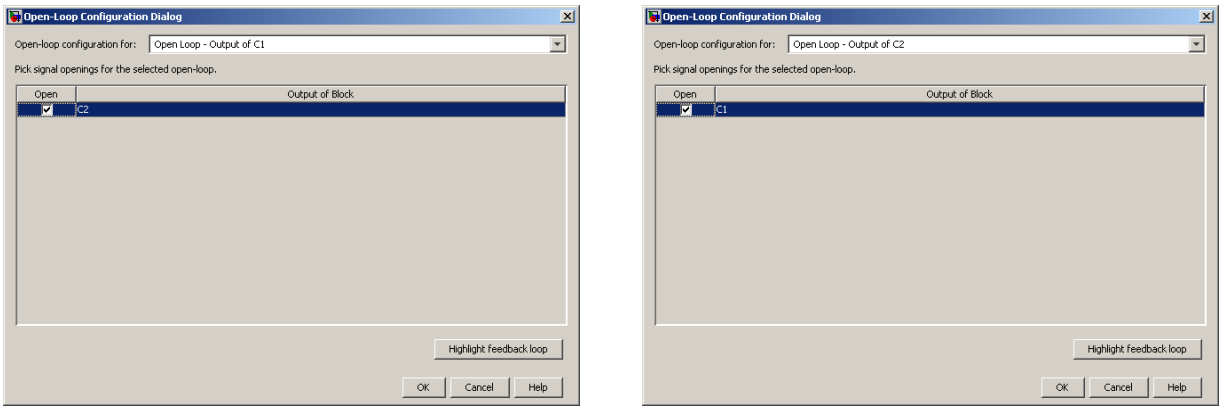


Figure 3: Loop Configuration

6. Go back to the **Control and Estimation Tools Manager** and select the **Graphical Tuning** tab.
7. Choose to plot the **Root Locus** plot for both **Open Loop 1** and **Open Loop 2** as shown in Figure 4.

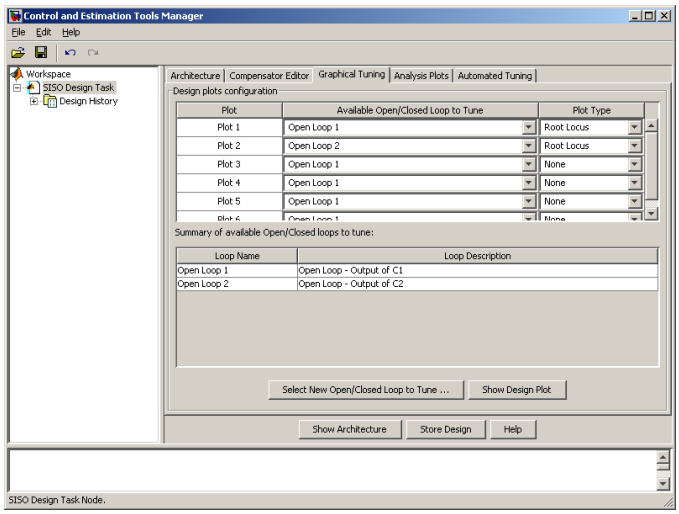


Figure 4: Design Plot Configurations

8. There will now be a design window with 2 empty root locus plots. For practice assume that the plant is $G_p(s) = \dfrac{938.4}{s^2 + 1.25s + 329.8}$
9. After this plant is imported into sisotool, there will be a design window and two identical proportional controllers shown in Figure 5.

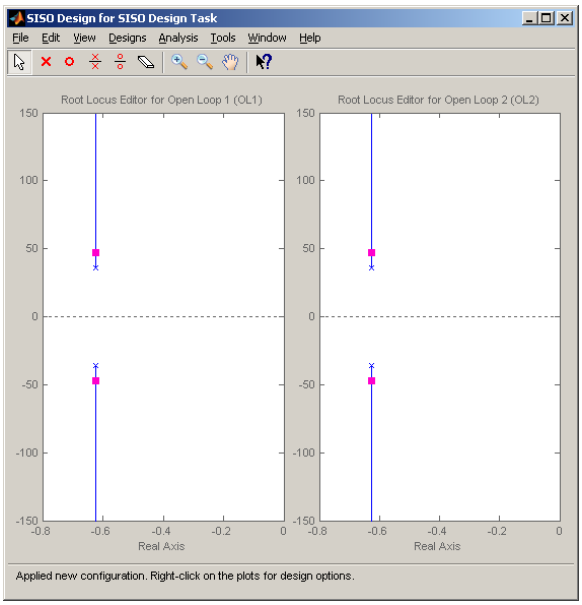


Figure 5: SISO Design for SISO Design Task

10. To design the PI-D controller, let $C_2(s) = k_d s$ and don't worry about the lowpass filter at this time. Assume that $k_d = 0.01$; to assign this to the controller C2 select **Designs**, **Edit Compensator**, then select **C2** in the compensator window and enter the controller.
11. To design the PI part of the controller into $C_1(s)$, assume that $C_1(s) = \dfrac{0.15(s+15)}{s}$ and enter this into **C1.** If you have done this correctly, you should get the root locus plots in Figure 6.

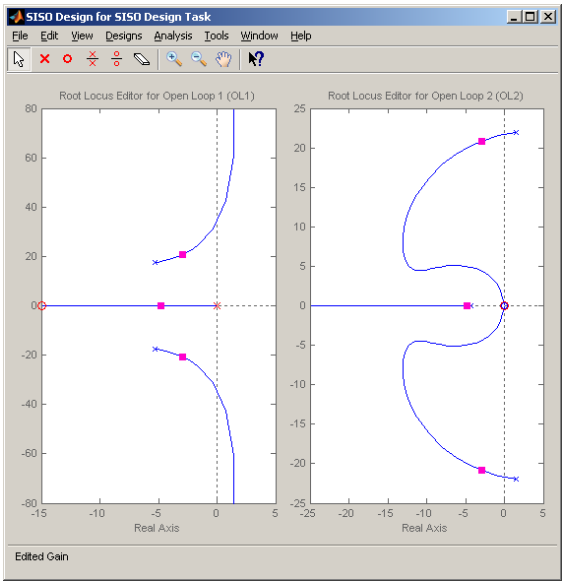


Figure 6: Root locus for PI-D controller

12. The step response for this controller configuration is shown in Figure 7.

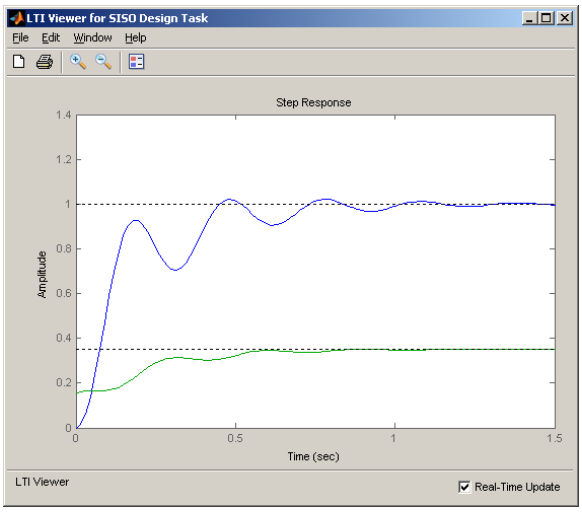


Figure 7: Step response for PI-D controller

13. Note that, compared to a normal PID controller, the control effort is not infinite at 0, and actually builds as time goes on (like an integral controller). At this point we might want to go back to modify the controllers, C1 and C2, to get acceptable performance. Note that it is possible to modify the two controllers (and gains) independently.
14. Now assume that you will control the same plant but use an I-PD controller. Assume that the controllers have the form

$$C_1(s) = \frac{1.5}{s}, C_2(s) = 0.01(s+1)$$

15. If you entered the controllers correctly, you should get the root locus plot shown in Figure 8.

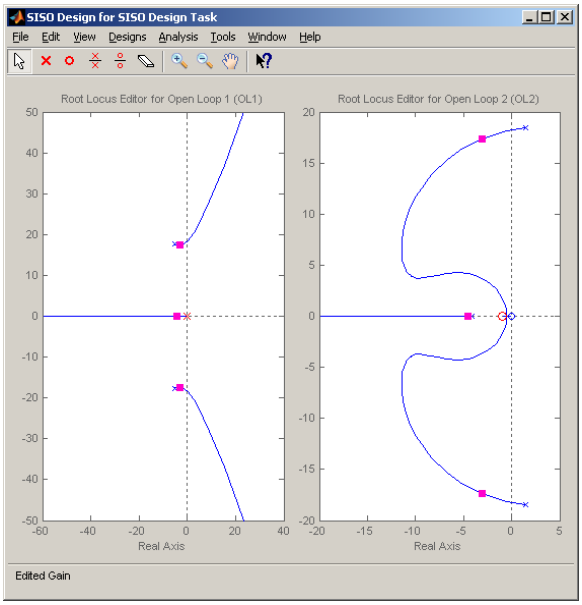


Figure 8: Root Locus for I-PD controller

16. The step response for the I-PD controller is shown in Figure 9.

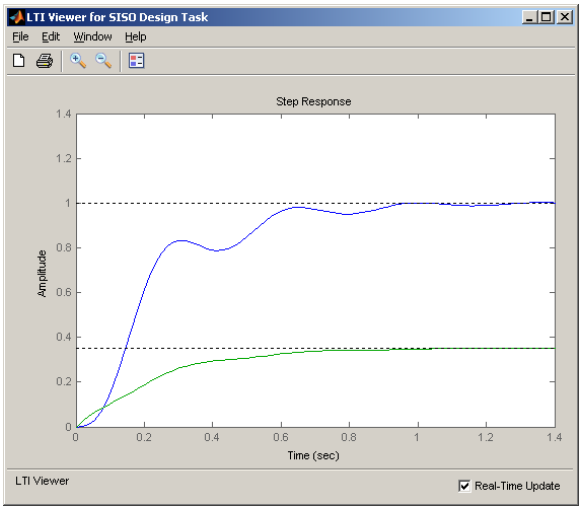


Figure 9: Step Response for I-PD controller

17. Now use **sisotool** to design an I-PD and I-PD controller for the 3 one degree of freedom systems you built in Lab 1 to meet the design specifications. If you cannot meet the design specifications exactly, do the best you can. All of your designs will use a constant prefilter, although you will also implement the dynamic prefilter in lab.
18. You should submit **four** graphs for each of the systems (PI-D root locus, PI-D step response, I-PD root locus, I-PD step response). Make sure that you provide a descriptive title that includes k_p, k_i and k_d and label the axes.

If there are any questions, please ask your instructor.

Theory:

While PID controllers are very versatile, they have a number of drawbacks. One of the major drawbacks is that for a unit step input, the control effort $u(t)$ can be infinite at the initial time. This is referred to as a *set-point kick*. There are two commonly used configurations of PID controls schemes that utilize a different structure, the PI-D and the I-PD controllers. These are a bit more difficult to model using Matlab's ***sisotool***, but it can be done.

The PI-D controller avoids the set-point kick by putting the derivative in the feedback path, while the I-PD controller avoids the set-point kick by placing both the derivative and proportional terms in the feedback path. Both types of controllers can be implemented using the Simulink model shown in Figure 10, which you will construct and name appropriately.

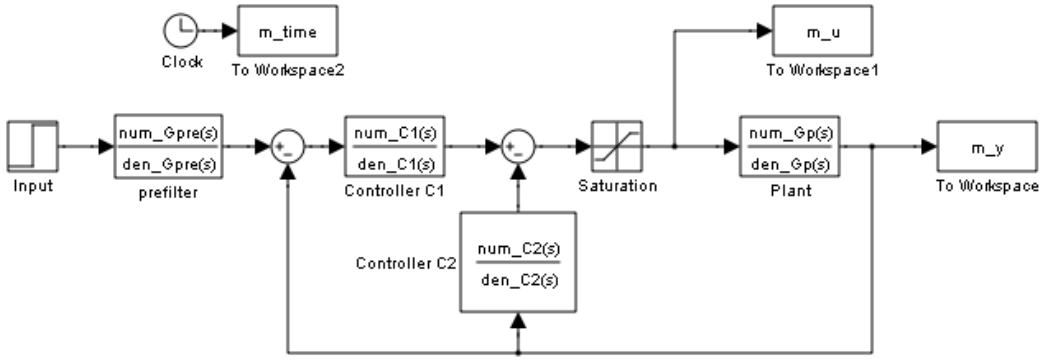


Figure 10: PI-D and I-PD Simulink model

For the PI-D controller, the controller transfer functions are:

$$C_1(s) = k_p + \frac{k_i}{s}, \quad C_2(s) = \frac{k_d s}{\frac{1}{50}s + 1}$$

For the I-PD controller, the transfer functions are: we have

$$C_1(s) = \frac{k_i}{s} \quad , C_2(s) = k_p + \frac{k_d s}{\frac{1}{50}s + 1}$$

Note that for both controllers, the lowpass filter with a cutoff of 50 rad/sec is put in series with the differentiator to clean up the signal. For both of these controllers, if we ignore the prefilter and assume it is unity then the transfer function from input to output is

$$\frac{Y(s)}{R(s)} = \frac{C_1(s)G_p(s)}{1 + C_2(s)G_p(s) + C_1(s)G_p(s)}$$

Procedure:

A. File Setup

1. Create a folder under Documents\ECE320\<your initials>\Lab8
2. Copy **all** of the MATLAB files from Lab 7 to the folder created in step 1
3. Modify **closedloop_driver.m** to read in the correct model file and implement the new structure. Make sure to comment out the old PID controller. In particular, it must now create C1(s) and C2(s), as well as the closed loop transfer function similar to the following:

```
C1 = tf(kp,1) + tf(ki,[1 0]);
[num_C1,den_C1] = tfdata(C1,'v') ;
C2 = tf([kd 0],[1/50 1]);
[num_C2,den_C2] = tfdata(C2,'v') ;
```

```
Go = C1*Gp/(1 + C2*Gp + C1*Gp);
Go = minreal(Go);
```

4. Rename the ECP simulation file, **closedloop.mdl** as **closedloop_PID_<your initials>.mdl**. Modify the new simulation model by creating the block diagram shown in Figure 10. Change the **`sim`** function in the closedloop_driver.m to call the new simulation file, **closedloop_PID_<your initials>.mdl.**
5. Rename the ECP driver file, **Model_210_Closedloop.mdl** as **Model210_PID_<your initials>.mdl**. Modify the new hardware model by creating the block diagram shown in Figure 10.

B. PI-D control

1. Set up the 1 DOF mass-cart spring system with two masses exactly the way you did when you determined the model parameters.
2. Design a PI-D controller to meet the design specifications. You may have already done this in the prelab. Use a constant prefilter (i.e., a number, most likely the number 1) and be sure to observe the limits on the controller gains.
3. Implement the correct gains into **closedloop_driver.m**
4. Simulate the system for 1.5 seconds, if the design constraints are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)
5. Once you are happy with the simulated design, reset the system by running **ECPDSPReset.mdl**
6. Compile and run **Model210_PID_<your initials>.mdl** and connect to the system to run the actual hardware response
7. Use the **compare1.m** file 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. Make sure to add a descriptive title that includes the values of K_p, K_d and K_i and axis labels.
8. Change the **prefilter** to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation and compare the predicted with the measured response. You also need to include this graph in your memo. Be sure to include an informative title, x and y labels and the values of K_p, K_i and K_d in your memo. Note that you will have to create the closed loop transfer function and then use that to create the **dynamic prefilter**.
9. Repeat steps 1 through 8 for the one DOF system four mass system.

I-PD Control

1. Set up the 1 DOF mass-cart spring system with two masses exactly the way you did when you determined the model parameters.

2. Design a I-PD controller to meet the design specifications. You may have already done this in the prelab. Use a constant prefilter (i.e., a number, most likely the number 1) and be sure to observe the limits on the controller gains.
3. Implement the correct gains into **closedloop_driver.m**
4. Simulate the system for 1.5 seconds, if the design constraints are not met, or the control effort hits a limit, redesign your controller (you might also try a lower input signal)
5. Once you are happy with the simulated design, reset the system by running **ECPDSPReset.mdl**
6. Compile and run **Model210_PID_<your initials>.mdl** and connect to the system to run the actual hardware response
7. Use the **compare1.m** file 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. Make sure to add a descriptive title that includes the values of K_p, K_d and K_i and axis labels.
8. Change the **prefilter** to cancel the zeros of the closed loop system and still have a steady state error of zero. Rerun the simulation and compare the predicted with the measured response. You also need to include this graph in your memo. Be sure to include an informative title, x and y labels and the values of K_p, K_i and K_d in your memo.
9. Repeat steps 1 through 8 for the one DOF system with 4 masses.

*Your memo should include **four graphs** for each of the 1 DOF systems you used (one PI-D and one I-PD controller with and without dynamic prefilters.) Be sure to include the values of k_p, k_i, and k_d in the captions for each figure. Your memo should compare the difference between the predicted response (from the model) and the actual response (from the real system) for each of the systems. How does the use of a dynamic prefilter change the response?*

Submission:

The lab memo will be due at the beginning of the next lab session. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, include ECP system number
- All required figures with number and caption and they should be referenced in the text

- All required data in tables with error analysis referenced in the text, if necessary
- The discussion should involve a comparison and contrast of the PI-D and I-PD controllers with and without the dynamic prefilter.
- The discussion should also include the differences in performance between the two and four mass systems

Lab 9
State Variable Feedback – Inverted Pendulum

Objective: Control the one degree of freedom system using state-variable feedback
Control the two degree of freedom system using state-variable feedback
Model, simulate and control a regular and inverted pendulum on a cart

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment if necessary.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors

This week's prelab and lab procedure are very long. Therefore, it is very important that you start early and prepare as much as possible in advance in order to finish the lab in the allotted time.

Prelab:
Read the lab theory and procedure thoroughly. Submit the following prelab on engineering paper at the beginning of lab.

1DOF State Variable Feedback

The one degree of freedom Simulink model (**Basic_1dof_State_Variable_Model.mdl**) implements a state variable feedback control model for a one degree of freedom system. This model uses the Matlab code **Basic_1dof_State_Variable_Model_Driver.m** to control it. These files are available in the Angel course folder under Lab 9 (**Lab9.zip**).

a) One lab partner should complete the prelab on the two mass 1DOF system and the 2DOF system models built in Labs 2 and 3. The other lab partner should complete the prelab on the four mass 1DOF and the 2DOF system models built in Labs 2 and 3.
b) Set the **saturation_level** to the correct level for the rectilinear system (model 210) and comment out the model 205 saturation level. Assume a step input of 1 cm (`Amp = 1.0;`) and comment out the amplitude in radians, this is for the torsional system.
c) Set the final simulation time to 2 seconds (`Tf = 2.0;`)
d) Modify **Basic_1dof_State_Variable_Model_Driver.m** to load the correct 1DOF state model into the system (`load 2_mass_model;`).

e) Design a state variable feedback system using **pole placement** for the one degree of freedom rectilinear system (`p=[p1 p2]; K = place(A,B,p);`). Determine the characteristic equation and closed loop poles for the step response (output) of the system to have:
   - A settling time less than or equal to 0.5 seconds
   - A percent overshoot less than or equal to 10%

f) You should include the derivation of the characteristic equation and closed loop poles on engineering paper with your prelab submission. Note that there may be some iteration on these values to meet the design specification during the simulation and lab procedure but this process gives you a starting point for the process.

g) To set the location of the closed loop poles, find the part of the code that assigns poles to the variable p, and change the elements of p. The biggest problem is to make sure the control effort is not too large. Your design should not saturate the system (control effort) and you should use a **constant prefilter** as defined in the driver m file, (`Gpf = 1.0;`).

h) Plot both the system output (from 0 to 2 seconds) and the control effort (from 0 to 0.2 seconds). Plot the control effort only out to 0.2 seconds since the control effort is usually largest near the initial time. If your control effort reaches its limits (±0.4), you need to go back and modify your design. Submit your plot with a descriptive title, the axes labeled and the closed loop poles and gains printed on it.

2DOF State Variable Feedback

The two degree of freedom Simulink model (**Basic_2dof_State_Variable_Model.mdl**) implements a state variable model for a two degree of freedom system. This model uses the Matlab code **Basic_2dof_State_Variable_Model_Driver.m** to drive it.

a) Modify **Basic_2dof_State_Variable_Model_Driver.m** to load the correct state model into the system.

b) Set the **saturation_level** to the correct level for the model 210 rectilinear system and assume a step input of 1 cm, similar to what you did for the 1DOF system.

c) Design a state variable feedback system using **pole placement** for the two degree of freedom rectilinear system. Determine the characteristic equation and closed loop poles for the step response (output) of the system to have:
   - A settling time less than or equal to 0.5 seconds
   - A percent overshoot less than or equal to 10%

d) You will need to place 4 poles but the dominant poles affect the settling time and damping ratio. The second two poles should be at least 5 times away from the dominant poles. You

should include the derivation of the characteristic equation and closed loop poles on engineering paper with your prelab submission. Note that there may be some iteration on these values to meet the design specification during the simulation and lab procedure but this provides a starting point.

e) To set the location of the closed loop poles, find the part of the code that assigns poles to the variable p, and change the elements of p. The biggest problem is to make sure the control effort is not too large. Your design should not saturate the system (control effort) and you should use a **constant prefilter** as defined in the driver m file**.**

f) Run your simulation for 2.0 seconds. Plot both the system output (from 0 to 2 seconds) and the control effort (from 0 to 0.2 seconds). Plot the control effort only out to 0.2 seconds since the control effort is usually largest near the initial time. If your control effort reaches its limits (±0.4), you need to go back and modify your design. Submit your plot with a descriptive title, the axes labeled and the closed loop poles and gains printed on it.

Pendulum State Variable Feedback

The simulink model, **Basic_2dof_State_Variable_Model_pend.mdl**, implements a state variable model for a pendulum. This model uses the MATLAB code, **Basic_2dof_State_Variable_Model_Driver_pend.m** to control the system. You will modify **Basic_2dof_State_Variable_Model_Driver_pend.m** to work with the **regular** pendulum model. Specifically, you need to

- Load the regular model for the pendulum (`load regular_model;`)
- Since we are trying to hold the pendulum in place, this is a *regulator problem* so set the input to zero (`Amp=0;`) .
- Next, set the initial angle of the pendulum to 0.05 radians and all other initial conditions to zero (`ic_x1 = 0;ic_x1_dot=0;ic_theta=0.05;ic_theta_dot=0;`).
- Use the pole placement method to control the position of the pendulum and the cart. The goal is to keep the pendulum pointing straight down and keep the cart from moving more than about 2.5 cm in each direction. The control effort should also be less than 0.4 and the system should come to steady state in less than 1.0 second.
- Once you have a satisfactory design, you should plot the system output and control effort from 0 to 2 seconds and submit it with a descriptive title, the axes labeled and the closed loop poles and gains printed on it.

Use the results of simulation of the regular pendulum to model the ***inverted*** pendulum. Use the same model and driver files but you need to change the state model. Specifically, you need to

- Load the inverted model for the pendulum (`load inverted_model;`)
- Since this is still a *regulator problem*, the input should be zero

- Set the initial value of the pendulum to 0.05 radians and all other initial conditions to zero.
- Use the pole placement method to control the position of the pendulum and the cart. The goal is to keep the pendulum pointing straight up and keep the cart from moving more than about 2.5 cm in each direction. The control effort should also be less than 0.4 and the system should come to steady state in less than 1.0 seconds. Limiting the cart motion is usually the most difficult part. Often your controller for the regular pendulum will work for this part too, but not always.
- Once you have a satisfactory design, you should plot the system output and control effort from 0 to 2 seconds and submit it with a descriptive title, the axes labeled and the closed loop poles and gains printed on it.

Theory:

Review your lecture notes and study guide for the essential theory on pole placement for state-variable feedback.

Pendulum – State Variable Feedback

In this derivation we will make a state variable model for a regular pendulum (a pendulum hanging down) attached to the first cart. It will be easier to measure the parameters for a regular pendulum since it is a stable system. In the lab we will initially try and control the regular pendulum. Once this is working, we will try to control an inverted pendulum. Note that the inverted pendulum is an unstable system. To go from the model of a regular pendulum to the model of an inverted pendulum we use the substitution $l \rightarrow -l$. Consider the configuration for the regular pendulum shown in Figure 1.

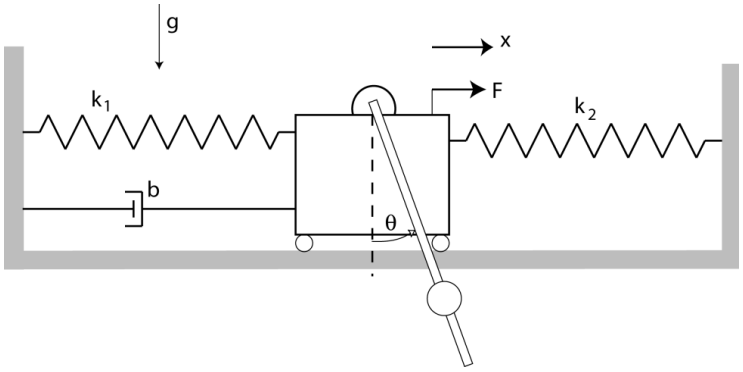


Figure 1: Pendulum on a cart system

The equations of motion for the regular pendulum can be written as

$$\begin{aligned}(J+ml^2)\ddot{\theta}+ml\cos(\theta)\ddot{x}+mgl\sin(\theta) &= 0\\ \left(M+m\right)\ddot{x}+ml\ddot{\theta}\cos(\theta)-ml\dot{\theta}^2\sin(\theta)+c\dot{x}+kx &= F\end{aligned}$$

The mass of the cart is M, center of mass of the pendulum is m, the moment of inertia of the pendulum about its center of mass is J, L is the length of the pendulum, and l is the distance from the pivot to the center of mass of the pendulum. The angle θ is measured counterclockwise from straight up, x is the displacement of the first cart (positive to the right), and g is the gravitational constant.

Using a small angle/small velocity assumption, we can approximate the above equations of motion as

$$\begin{aligned}(J+ml^2)\ddot{\theta}+ml\ddot{x}+mgl\theta &\approx 0\\ \left(M+m\right)\ddot{x}+ml\ddot{\theta}+c\dot{x}+kx &\approx F\end{aligned}$$

The first equation can be rewritten as $\frac{1}{\omega_\theta^2}\ddot{\theta}+\ \frac{1}{g}\ddot{x}+\theta\ =\ 0$

If we assume the cart is fixed, then $\ddot{x}=0$ and we have $\ddot{\theta}+\ \omega_o^2\theta=0$

This is the equation for a simple pendulum. If the pendulum is deflected a small angle and released, it will oscillate with frequency, ω_θ. It is possible to measure the period of oscillations, T_θ, and use that to find ω_θ.

The second equation of motion can be rewritten as $\frac{1}{\omega_1^2}\ddot{x}+\ K_1\ddot{\theta}+\ \frac{2\zeta_1}{\omega_1}\dot{x}+x\ =\ K_2F$

If we assume there is no input (F = 0) and the pendulum does not move very much ($\ddot{\theta}\approx 0$) then we can use the log-decrement method to get initial estimates of ω_1 and ζ_1. Assuming we apply a step input of amplitude A to the cart then the steady state value is $K_2=\frac{x_{ss}}{A}$.

The transfer function between the cart position and pendulum angle can be found from the first equation of motion as

$$\frac{\Theta(s)}{X(s)}=-\frac{\omega_\theta^2}{g}\left(\frac{s^2}{s^2+\omega_\theta^2}\right)$$

To measure the gravitational constant in cm, the transfer function between the cart position and input force in cm is given by

$$\frac{X(s)}{F(s)} = \frac{\omega_1^2 K_2 (s^2 + \omega_\theta^2)}{\left(1 - K_1 \omega_1^2 \frac{\omega_\theta^2}{g}\right) s^4 + (2\zeta_1 \omega_1) s^3 + (\omega_1^2 + \omega_\theta^2) s^2 + (2\zeta_1 \omega_1 \omega_\theta^2) s + \omega_1^2 \omega_\theta^2}$$

This expression can be use do to determine K_1 and get better estimates of ω_1 and ζ_1.

The linearized dynamical equations are

$$\begin{aligned} \ddot{\theta} &\approx -\frac{1}{g}\omega_\theta^2 \ddot{x} - \omega_\theta^2 \theta \\ \ddot{x} &\approx -2\zeta_1 \omega_1 \dot{x} - \omega_1^2 x - K_1 \omega_1^2 \ddot{\theta} + K_2 \omega_1^2 F \end{aligned}$$

By substituting the second equation into the first equation, we get

$$\ddot{\theta} \approx \frac{1}{\Delta}\left(\frac{\omega_1^2 \omega_\theta^2}{g}\right) x + \frac{1}{\Delta}\left(\frac{2\zeta_1 \omega_1 \omega_\theta^2}{g}\right) \dot{x} + \frac{1}{\Delta}\left(-\omega_\theta^2\right) \theta + \frac{1}{\Delta}\left(-\frac{\omega_1^2 \omega_\theta^2 K_2}{g}\right) F$$

where

$$\Delta = 1 - \left(\frac{K_1 \omega_1^2 \omega_\theta^2}{g}\right)$$

and substituting the first equation into the second equation we get

$$\ddot{x} \approx \frac{1}{\Delta}\left(-\omega_1^2\right) x + \frac{1}{\Delta}\left(-2\zeta_1 \omega_1\right) \dot{x} + \frac{1}{\Delta}\left(K_1 \omega_1^2 \omega_\theta^2\right) \theta + \frac{1}{\Delta}\left(K_2 \omega_1^2\right) F$$

Defining $q_1 = x$, $q_2 = \dot{x}$, $q_3 = \theta$ and $q_4 = \dot{\theta}$, we get the following state equations

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\left(\frac{\omega_1^2}{\Delta}\right) & -\left(\frac{2\zeta_1 \omega_1}{\Delta}\right) & \left(\frac{K_1 \omega_1^2 \omega_\theta^2}{\Delta}\right) & 0 \\ 0 & 0 & 0 & 1 \\ \left(\frac{\omega_1^2 \omega_\theta^2}{g\Delta}\right) & \left(\frac{2\zeta_1 \omega_1 \omega_\theta^2}{g\Delta}\right) & -\left(\frac{\omega_\theta^2}{\Delta}\right) & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \left(\frac{K_2 \omega_1^2}{\Delta}\right) \\ 0 \\ -\left(\frac{\omega_1^2 \omega_\theta^2 K_2}{g\Delta}\right) \end{bmatrix} F$$

If we now want to model the inverted pendulum ($l \rightarrow -l$), how do the terms change in the matrices above? In lab, we will see that when we try to fit the frequency response data, you will get an unusual response. To understand this response you will obtain values of

$$K_1 = \frac{ml}{k}$$
$$\omega_1^2 = \frac{k}{M+m}$$
$$\omega_\theta^2 = \frac{mgl}{J+ml^2}$$

If we assume that the mass of the cart and pendulum attachment is much larger than the mass at the center of mass of the pendulum, then we have M is much larger than m. Secondly, J is the moment of inertia about the center of mass of the pendulum, m is the mass at the center of mass of the pendulum, and l is the distance from the pivot to the center of mass of the pendulum. For our systems, the pendulum bars have negligible mass and all of the mass is essentially concentrated at the center of mass hence ml^2 is much larger than J. Using these two assumptions yields $\Delta \approx 1$ and the system transfer function becomes

$$\frac{X(s)}{F(s)} \approx \frac{\omega_1^2 K_2}{s^2 + 2\zeta_1\omega_1 s + \omega_1^2}$$

That is, there is a pole/zero cancellation! This is the effect you will often see in lab.

Procedure:

A. File Setup

1. Create a folder under Documents\ECE320\<your initials>\Lab9
2. Extract all of the **Lab9.zip** files from the Angel course folder to the folder created in step 1. Make sure to use the version of the files you have modified for the prelab.
3. Copy **all** of the 1DOF and 2DOF system models from Labs 2 and 3 into the folder created in step 1.

B. 1DOF Pole Placement Design

1. Set up the 1DOF system with <u>two</u> masses exactly as you did when you determined its model parameters.
2. Modify **Basic_1dof_State_Variable_Model_Driver.m** to read in the correct model file and to use the correct **saturation_level** for the Model 210 system. The input is a 1 cm step.
3. Design a state variable controller using pole placement to control the position of the cart to meet the design specifications. You should have already done this for the prelab.
4. Simulate the system for 1.5 seconds, if the design constraints are not met, or the control effort hits a limit; redesign your controller (you might also try a lower input signal). You may have already done this in the prelab.

5. Open **Model210_sv1.mdl** and confirm that the base address is correct. Then compile the closed loop ECP Simulink driver, **Model210_sv1.mdl**, connect to the system, and run the simulation
6. Use the **compare1.m** file (or a modification of it) to plot the results of both the simulation and the real system on two nice, neatly labeled graphs using the subplot command. It is two graphs because **you need to compare both the position and the velocity of the cart** (see Figure 2 for an example). You need to include these graphs in your memo submission. (*Note: the steady state error is likely to be off. Placing the poles farther away can reduce this error for these systems.*)
7. Repeat steps 1 through 6 for the 1DOF system with 4 masses. You should compare and contrast the results of these analyses in your memo.

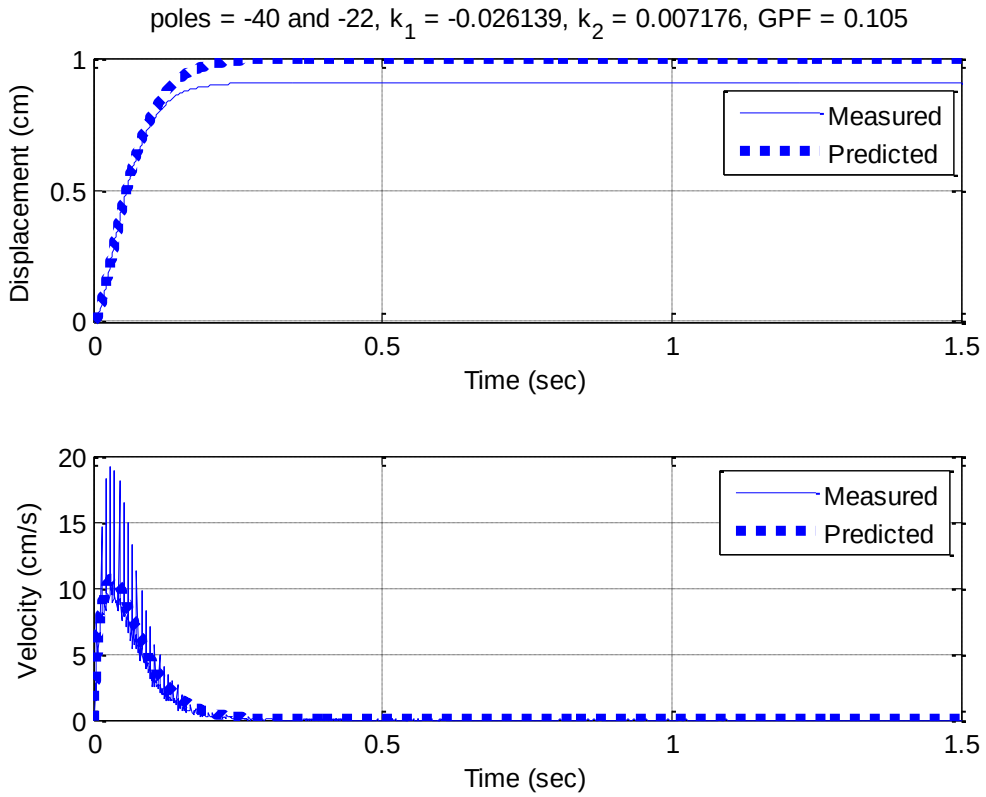


Figure 2: 1DOF Pole Placement Simulation Results

C. 2DOF Pole Placement Design

1. Set up the 2DOF system exactly as you did when you determined its model parameters
2. Modify **Basic_2dof_State_Variable_Model_Driver.m** to read in the correct model file and to use the correct **saturation_level** for the Model 210 system. The input is a 1 cm step.

3. Design a state variable controller using pole placement to control the position of the **first cart** to meet the design specifications (`C = [1 0 0 0];`). You may have already done this for the prelab.
4. Simulate the system for 1.5 seconds, if the design constraints are not met, or the control effort hits a limit; redesign your controller (you might also try a lower input signal). You may have already done this in the prelab.
5. Open **Model210_sv2.mdl** and confirm that the base address is correct. Then compile the closed loop ECP Simulink driver, **Model210_sv1.mdl**, connect to the system, and run the simulation
6. Use the **compare1.m** file (or a modification of it) to plot the results of both the simulation and the real system on four nice, neatly labeled graphs using the subplot command. **You need to compare both the position and the velocity of both of the carts**. You need to include these graphs in your memo.
7. Repeat the above steps to design a state variable controller using pole placement to control the position of the **second cart** to meet the design specifications (`C = [0 0 1 0];`).

D. Regular Pendulum Pole Placement Design

a. Set up the system

1. The state equations for the regular pendulum were derived in the prelab. All of the quantities must be identified to get the A and B matrices for the state variable description. For this system, D = 0 and C is determined by what you want the output to be (i.e. cart (`C = [1 0 0 0];`) or pendulum (`C = [0 0 1 0];`)).
2. Lock all of the carts except the one closest to the motor, place two large masses on the first cart and one heavy (stiff) spring between the first and second cart. There is no spring between the motor and the cart and the damper is not attached.
3. The pendulum should be securely fastened to the first cart and rest on top of the masses and be securely tightened.
4. The mass on the pendulum should be within about 2 or three inches from the pivot. Remember that the cart must be able to get under the center of mass of the pendulum in order to right it, so if the center of mass of the pendulum is too far away the cart will never be able to get under it. The ECP system should be moved to the edge of the bench, so that the pendulum is completely free to swing without hitting the bench.
5. The wire to measure the position of the pendulum position encoder should be securely attached (with the screws) and the cart and pendulum should be free to move (see Figure 2).

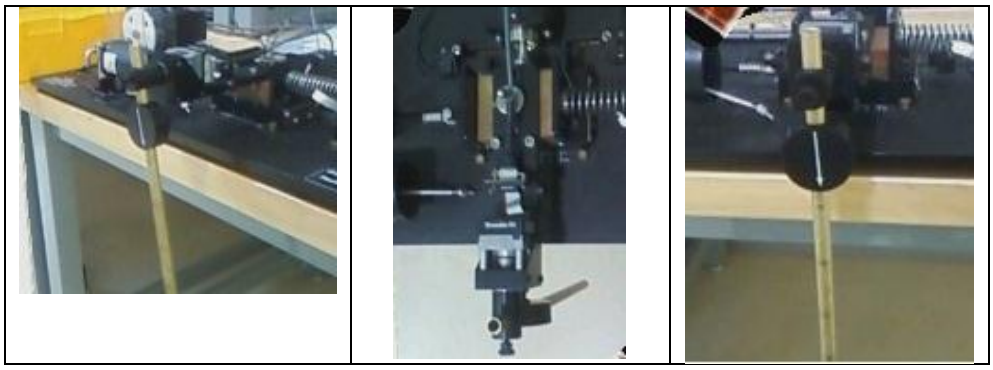


Figure 2: Regular Pendulum Set Up

b. Estimate ω_θ

1. From the equations of motion, if we assume the cart is fixed, then $\ddot{x}$ = 0 and we have $\ddot{\theta} + \omega_\theta^2 \theta = 0$. This is the equation for the simple pendulum, if the pendulum is deflected and released it will oscillate with a frequency of ω_θ.
2. To measure this frequency, set the amplitude of the input in **Model210_Openloop.mdl** to "0" and confirm that the base address is correct.
3. Displace the pendulum and let it go. Since we are using a small angle assumption, the pendulum should not be displaced too far.
4. Using MATLAB, plot the displacement of the pendulum versus time. This is the variable *theta* in the MATLAB workspace. Determine the period (T_θ) of the pendulum oscillations and use the period to calculate ω_θ.
5. You should include a graph of the pendulum oscillation (`plot(time, theta))` with a descriptive title in your memo submission (see Figure 3).

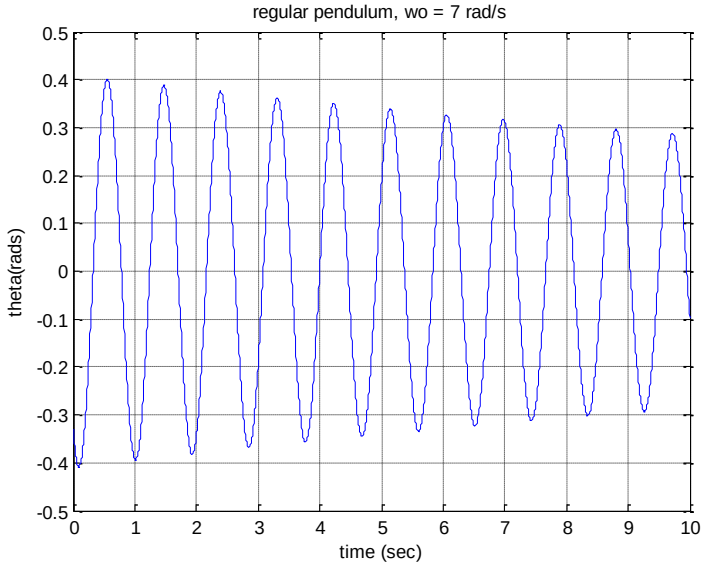


Figure 3: Estimation of ω_θ.

c. Estimate ω_1 and ζ_1 using log-decrement analysis

1. If we assume that there is no input to the cart (F = 0) and the pendulum does not move very much ($\ddot{\theta} \approx 0$) then we have

$$\frac{1}{\omega_1^2}\ddot{x} + \frac{2\zeta_1}{\omega_1}\dot{x} + x = 0$$

2. Use the log-decrement method to get estimates of ω_1 *and* ζ_1 by displacing the cart and running ***Model210_Openloop.mdl***
3. Lastly, run the ***log_dec.m*** MATLAB file to generate the estimate of the system parameters. Note that you may have to copy the ***log_dec.m*** and ***log_dec.fig*** files from one of the earlier lab folders.

d. Estimation of K_2

1. Modify ***Model210_Openloop.mdl*** to have amplitude of "0.1", apply a step input of amplitude A to the cart and estimate K_2 by using the formula,

$$K_2 = \frac{x_{ss}}{A}$$

2. Compile and run ***Model210_Openloop.mdl*** and view the variable *x1* in the MATLAB command window to find x_{ss}. Calculate the gain using the formula in step 1.
3. Reset the system to have zero initial conditions by running ***ESCPDSPReset.mdl***
4. Repeat steps 2 through 5 for two more different input amplitudes.
5. Calculate K_2 as the average of the 3 gains found.

e. Fit the estimated frequency response to the measured frequency response

1. This method will be used to determine K_1 and to get better estimates of ω_1 and ζ_1. At this point, all of the variables needed for the state variable model will be determined. The magnitude portion of the Bode plot will be constructed and fitted to the measured frequency response of the expected transfer function to determine these parameters.
2. For frequencies f = 1, 1.5, 2, 2.5, …. 5 Hz modify ***Model210_Openloop.mdl*** so the input is a sinusoid with amplitude of "0.1". As you approach resonance, you may have to decrease the input amplitude to keep the cart from hitting the stops by moving more than 2.5 cm. Make sure you also have at least 2 or 3 data points near the pendulum's natural frequency, f_θ.
3. Compile **Model210_Openloop.mdl**, connect to the ECP and run the model. If the cart does not move much, increase the amplitude of the input sinusoid. If the cart hits the stops, you need to adjust the pendulum. Note that you can make the input amplitude a variable such as `amp` in the SIMULINK block diagram and make the frequency a variable such as `freq` in the SIMULINK block diagram and set the values in the MATLAB command window to keep

from recompiling ***Model210_Openloop.mdl*** after each run. *Make sure that the system reaches steady state before you measure the amplitude!* You should change the simulation stop time if it does not reach steady state.

4. Create an Excel file to 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. Use the MATLAB program, ***get_B.m***, to get the average of the output amplitude.
5. You will probably notice that the output does not look quite as sinusoidal as usual. This is because we are not really giving the pendulum enough time to reach steady state. Enter the values of *f*, *A*, and *B* into the program ***process_data_pendulum.m***
6. At the Matlab prompt, type ***data = process_data_pendulum;***

f. Modeling the Regular Pendulum

1. Run the program **model_pendulum_full.m**; there are 5 input arguments to this program:
   - ***data***, the measured data as determined by ***process_data_pendulum.m***
   - $\mathbf{K_2}$, the estimated value from part d
   - ω_θ, the estimated frequency of the pendulum, in Hz from part b
   - ζ_1, the estimated damping ratio of the cart from part c
   - ω_1, the estimated natural frequency of the cart, in Hz from part c

2. The program ***model_pendulum_full.***m will produce the following:
   - A graph of the fitted transfer function of the position of the cart to the input to the measured frequency response data. The optimal estimates of all parameters is(written at the top of the graph (see Figure 4).
   - 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. Rename this file to something similar to **regular_pend_model.mat.**
   - A list of the poles and zeros of the estimated transfer function are printed to the MATLAB command window. This allows you to see how close to a pole/zero cancellation you have (this is 6.6133 rad/s for Figure 4).
   - You need to be sure you have 4 points close to the resonant peaks of the transfer function. This is particularly true if you have very small values of ζ (<0.1), which correspond to very sharp peaks. In addition, you should add a couple of points near the frequency of the pole/zero cancellation to clean things up and simulate the system at the natural frequency of the pendulum. **At a minimum you should add 6 additional points.**
   - You should also compare the final estimates of the parameters with your initial estimates (i.e. error analysis). The values for all of the frequencies and gains should be fairly close to the final values however, the damping ratios may be quite different.

Create a data table with gridlines in the lab memo submission that shows the measured, nominal, and percent error for the system parameters.

- You should include the final improved graph with a descriptive title as well as the state variables (A, B, C, D) and transfer function in your lab memo submission.

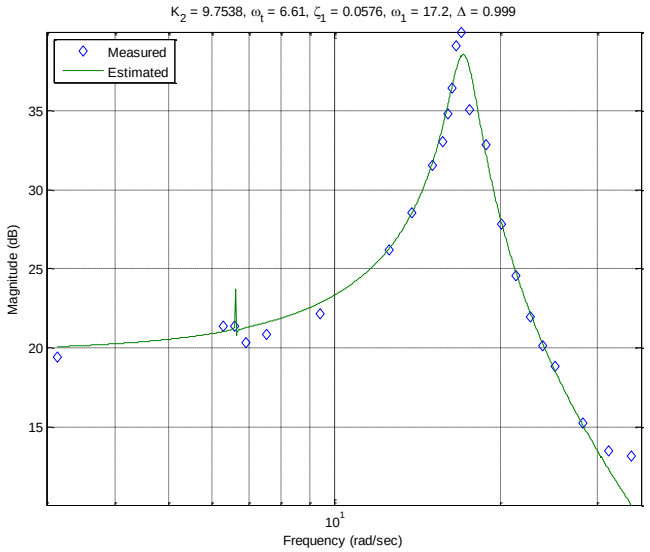


Figure 4: Regular Pendulum Model with pole/zero cancelation at ω_θ = 6.6133 rad/s

g. Control the ECP system with Simulink for the Regular Pendulum

1. We will now try to control the regular pendulum that you just modeled. This is actually a **regulator problem** in that you will try to maintain a set point (i.e., keep the pendulum pointing down).
2. Using the program **Basic_2dof_State_Variable_Model_Driver_pend.m** and the corresponding state variable model file (**regular_pend_model.mat**), simulate the system with zero input, and all initial conditions set to zero except for the initial pendulum displacement. Set the initial pendulum displacement to something like 0.01 radians (`ic_theta = 0.01;`)
3. Use state variable feedback so that the first cart moves to keep the pendulum pointing down (the pendulum angle is zero degrees) with a settling time of less than 1.0 seconds. Use the direct pole placement method to determine the feedback gain values. You also need to be sure the control effort is less than 0.4 and the cart does not move more than about 2.5 cm. You can use your prelab design as a good starting point for the pole placement. You can also try to place the poles at -5, -7, -9, -11. Try to make sure that the first two gains are less than 0.2 if possible. See Figure 5 for an example of an acceptable design from the simulation.

4. Once you are happy with your design, you can set the simulation final time to 20 seconds (`Tf = 20;`).
5. Confirm that the pendulum is pointing straight down and is **at rest**. Reset the system using **ECPDSPReset** (this is how the ECP finds its zero).
6. Now try to compile and run **Model210_Regular_Pendulum.mdl** to control the ECP system with these same parameters. Since there is no input, the system should do nothing. Use a pencil or ruler to gently nudge the pendulum. The cart should move so the pendulum returns to an angle of zero degrees. If your system does not seem to work very well, do not go on to the next part until you've figured out what is wrong.

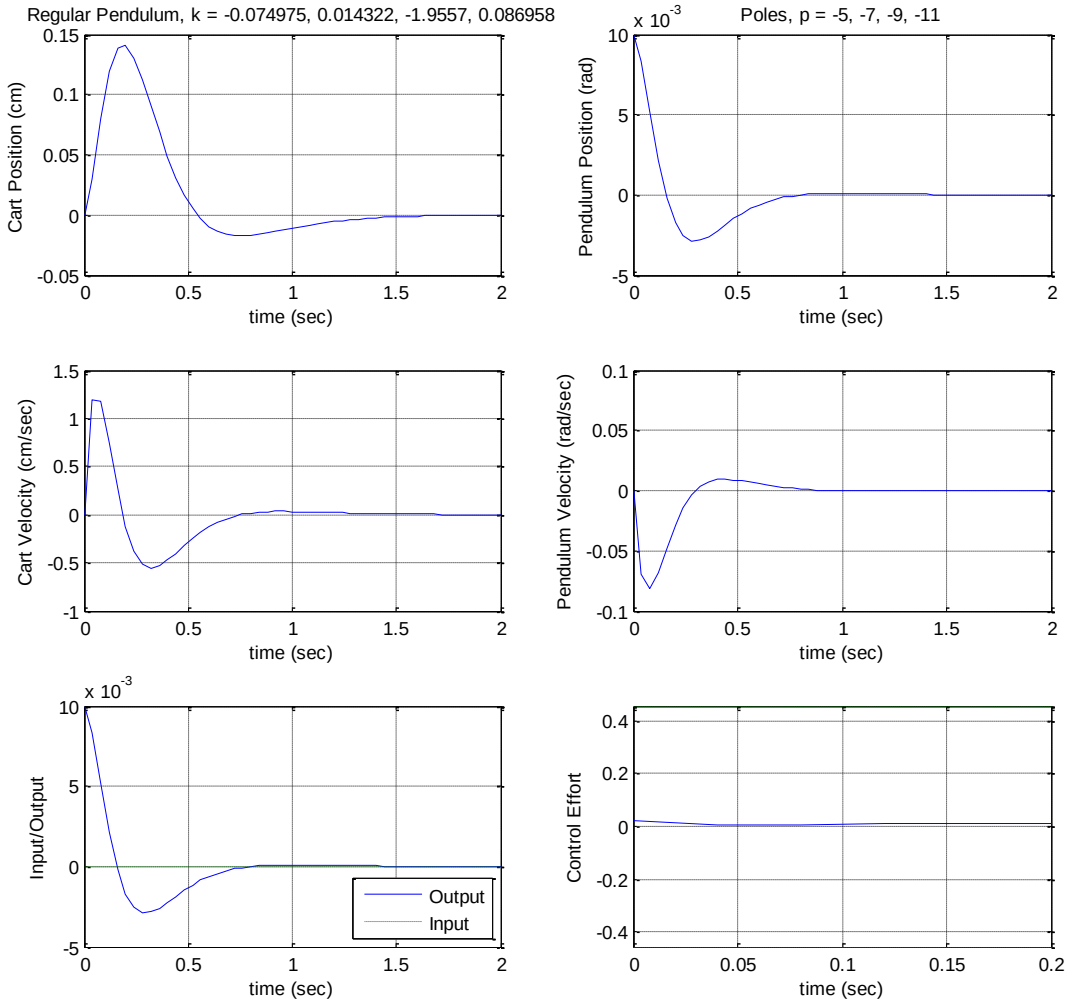


Figure 5: Regular Pendulum State Variable Feedback Design Simulation Results

h. Modeling the Inverted Pendulum

1. Run the program **model_inverted_pendulum_full.m;** there are 5 input arguments to this program:
   - **data**, the measured data as determined by **process_data_pendulum.m**
   - the estimated value of $\mathbf{K_2}$
   - ω_θ, the estimated frequency of the pendulum, in radians/sec
   - ζ_1, the estimated damping ratio of the cart.
   - ω_1, the estimated natural frequency of the cart, in radians/sec

2. The program **model_inverted_pendulum_full.m** will produce the same output as the program **model_pendulum_full.m** produced. The only difference will be the state variable model. You should rename this model to sometime like **inverted_pend_model.mat.** You should include the state variables (A, B, C, D) and transfer function in your lab memo submission.

i. Control the ECP system with Simulink for the Inverted Pendulum

1. This is much more difficult than controlling the regular pendulum, since the system is inherently unstable. You should expect failure until you get the hang of it. The biggest problem is that you must start holding the pendulum nearly vertical. If you are off even slightly in holding the pendulum vertical the system will not work. In addition, it is very important to complete the following steps explicitly. **Make sure you read all of the steps before you run the inverted pendulum system!**
2. Simulate the system with zero input, and all initial conditions set to zero except for the initial pendulum displacement (0.01 rads). Use state variable feedback so the first cart moves to keep the pendulum pointing up with a settling time of less than 1.0 seconds. You can use your prelab values as a good starting point for the pole placement.
3. Confirm that the pendulum is pointing straight down and is a rest. Reset the system using **ECPDSPReset** (this is how the ECP zeros). ***You must always do this step before you try and run the inverted pendulum!***
4. To control the inverted pendulum through the ECP system, you must use the file **Model210_Inverted_Pendulum.mdl**. Compile this with your parameters by running **Basic_2dof_State_Variable_Model_Driver_pend.m**. Make sure that you modify this file to load the inverted pendulum model (`load inverted_pend_model;`).
5. Connect to the ECP system, but do not start the simulation. Gently rotate the pendulum up to vertical and hold it loosely between your fingers. Run the simulation while still holding the pendulum loosely. If the system moves rapidly when started you need to let go of the

pendulum and stop the system. If the system does not move much, gently let go of the pendulum.

6. While the pendulum is vertical, gently blow on it. This should provide some external disturbance which the system should be able to overcome. This is feedback control in action. See Figure 6 for an example of the working system.
7. 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 (**Basic_2dof_State_Variable_Model_Driver_pend.m**) of the system to get all the necessary parameters in the workspace before compiling the ECP system.
8. Demonstrate the working inverted pendulum to your instructor.

Figure 6: Inverted Pendulum State Variable Feedback Control

Submission:

The lab memo will be due at the beginning of the next lab session. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, include ECP system number
- All required figures with number and caption and they should be referenced in the text
- All required data in tables with error analysis referenced in the text, if necessary

Lab 10

Integral Control and Observers – State Variable Feedback

Objective: Control the one degree of freedom system with a full order observer
Control the one degree of freedom system with full order observer and integral control
Control the two degree of freedom system with a full order observer
Control the two degree of freedom system with full order observer and integral control

Caution: The ECP systems are expensive and can be damaged!
Have one member of your group ready to push the hardware shut off button when you are running an experiment if necessary.
Do not touch the system while it is running.
Turn off the ECP power before changing connectors

Prelab:

Read the lab theory and procedure thoroughly and complete the following prelab for submission in class the day before lab. Note that this prelab is very long and it is recommended that you start early and complete it in its entirety to save time during the lab session. Since the prelab is such an integral part of this week's lab, it will be worth 40% of the lab grade.

In this prelab, we will simulate and study integral control, full order observers and then the fully integrated system with state variable feedback. This type of integrator in the system is preferable to a prefilter in order to achieve zero steady state error for a step input, since it will help overcome errors in the plant model. Many of these steps of this prelab will be implemented to confirm that the programs are working correctly.

1) For all of the problems in the prelab, use the state models for the rectilinear systems posted in the Angel course folder (***prelab10.rar***).
2) Confirm in SIMULINK that you use a variable step solver, and the maximum time step is approximately 1e-5. You can do this by going to the following: Simulation-> Configuration Parameters. Also, set the initial conditions on the integrators to a 0 (a scalar).
3) ***You need to save plots of every case you are run. You can paste them into a Word document with appropriate figure captions and submit it via email.***

Integral Control 1DOF Model

1) Create a Lab 10 folder on your hard drive.
2) Copy the program, **Basic_1dof_State_Variable_Model_driver.m,** from a prior lab to the Lab 10 folder and rename it**, sv1_integral_driver.m.**
3) Copy the Simulink file **Basic_1dof_State_Variable_Model.mdl** to the Lab 10 folder and rename it, **sv1_integral.mdl.**
4) Modify **sv1_integral_driver.m** to implement state variable feedback using integral control. This includes changing the load command to add the A, B, C, and D matrices from the prelab model file (**state_model_1dof_rect.mat**). This also includes defining the A and B matrix for the integral controller in order to place the poles,

$$A_I = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \qquad B_I = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

5) You should also set all of the initial conditions on the integrators to zero (`ic_x=0;ic_x_dot=0;`). Lastly change the simulation to the new model file (`sim('sv1_integral.mdl');`). You may also have to define the C matrix dependent upon whether you want the output to be the cart position or velocity.
6) Modify **sv1_integral.mdl** to implement state variable feedback using integral control. The basic integral control scheme is shown below in Figure 1. This includes adding a summing block after the step input that also subtracts the output y. It also involves adding an integrator after the first summing block and multiplying by the integrator gain, *KI*. The output of the integrator is the input to the original summing block and the rest of the model is the same as in lab 9. Note that both integrators should have initial conditions of zero set in the model or in the m-file.

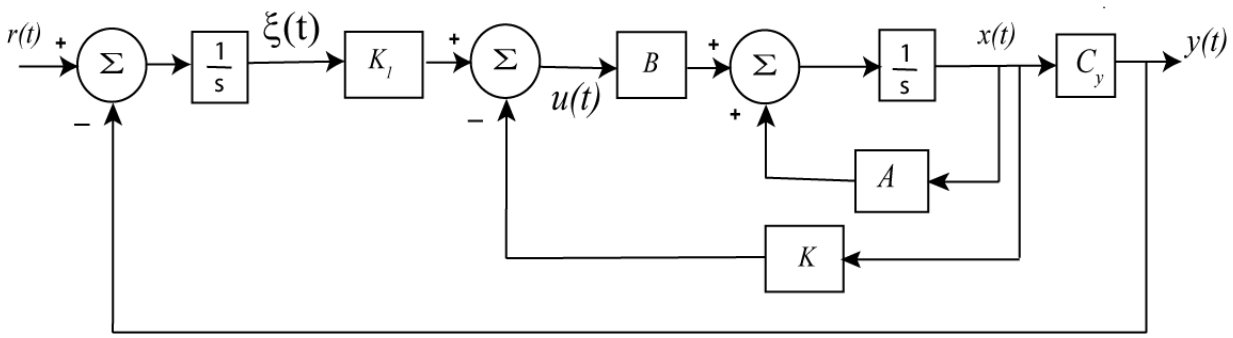


Figure 1: Integral control block diagram

7) Modify **sv1_integral_driver.m** to plot the two states and the control effort. Set the step amplitude to 1.0, the final time to 2.5 seconds, place the poles at -5, -10, -15, and run the simulation. You should get results like those shown in Figure 2.

8) Assume that this system must be designed to settle in less than 2 seconds, have less than 15% overshoot and zero steady-state error.

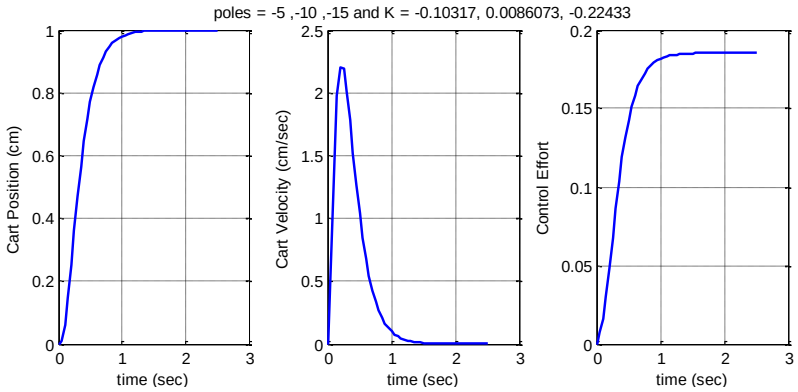


Figure 2: Initial results for the 1 DOF system with integral control

9) At this point there is no real reason for us to use integral control, since we know everything exactly and there is zero steady-state error. Let's assume your lab partner is has derived a state model that is way off from the actual system. In your code, just before your run the simulation (just before the **sim** command), type `A = A*0.75`, which basically makes the A state matrix way off from the designed model. If you make this change just before the simulation, your gains will be chosen assuming your model is correct, and we will be running the simulation using a different system. You should get results like those shown in Figure 3, which look pretty hopeless, but we can improve them.

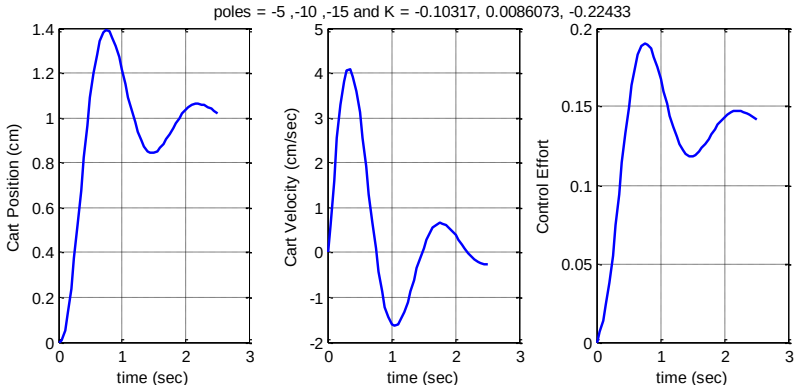


Figure 3: Integral control with a bad system model

10) Systems with these integrators have an interesting characteristic that they correct the error and high frequency noise. Change the poles to -5, -10, and -35 and rerun the system; you should get the results shown in Figure 4, which are much better. Note that the control effort is also much better. **Before you go on, be sure to remove the line A = A*0.75!**

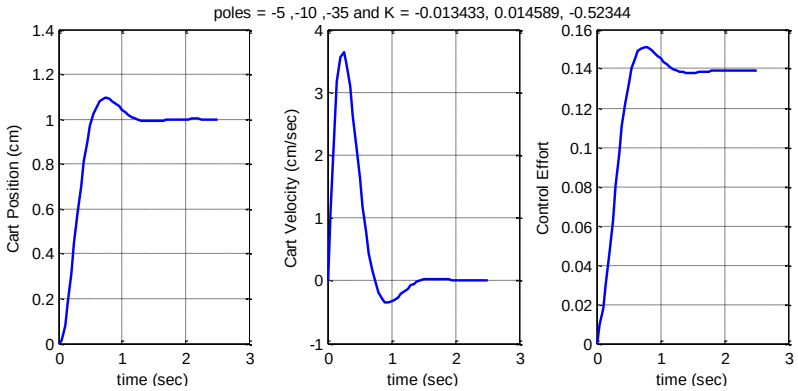


Figure 4: Integral control with a bad system model and better gains

Integral Control 2DOF Model

1) In this problem we will incorporate integral control for a two degree of freedom system and run some test cases to verify your code is working.
2) Copy your program **Basic_2dof_State_Variable_Model_driver.m** to the Lab 10 folder and rename it **sv2_integral_driver.m.** Then copy the Simulink file **Basic_2dof_State_Variable_Model.mdl** to the Lab 10 folder and rename it, **sv2_integral.mdl.**
3) Modify these ***new*** files to implement state variable feedback using integral control. For **sv2_integral_driver.m** this includes loading the 2DOF model for the rectilinear system (`load state_model_2dof_rect`). This also includes defining the A and B matrix for the integral controller in order to place the poles,

$$A_I = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \qquad B_I = \begin{bmatrix} B \\ 0 \end{bmatrix}$$

4) You should also set all of the initial conditions on the integrators to zero (`ic_x1=0;ic_x2=0;ic_x1_dot=0;ix_x2_dot=0;`). Lastly change the simulation to the new model file (`sim('sv2_integral.mdl');`). You may also have to define the C matrix dependent upon whether you want the output to be the first or second cart position or velocity.
5) The process to modify **sv2_integral.mdl** will be similar to that completed for the 1DOF system. You will need to add a summer after the step input that subtracts the output y. Then you will add an integrator with zero initial condition and gain `KI` after the summer. This is now the new input to the summer in the original model and everything else remains the same.
6) Modify **sv2_integral_driver.m** to plot the four states, output and the control effort.

7) Set the amplitude to 1.0 and the final time to 2.5 seconds, and place the poles at -5, -10, -15,-20, and -5. Run the simulation and you should get results similar to those shown in Figure 5.

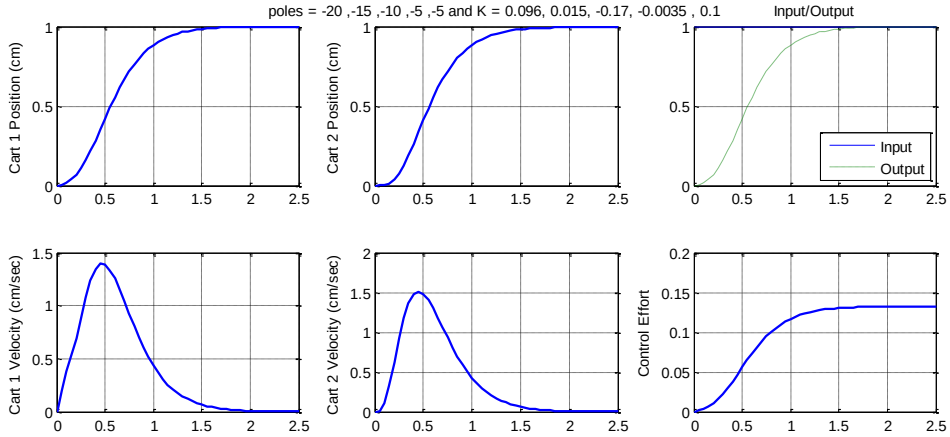


Figure 5: State feedback control of a 2DOF rectilinear system

8) Assume your lab partner made a mistake in the system analysis and your model is not accurate. You will simulate this error by setting the A matrix to **`0.75*A`** after the gains have been determined for the poles in step 6. Run the simulation for 3 seconds and the results should look similar to Figure 6. Note that it does not meet the settling time and percent overshoot requirements anymore.
9) The benefit of integral control is that this bad model can still be designed to improve the response by increasing the integral controller gain. Set the poles to -5, -10, -15,-20, and -25 and run the simulation for 2s again. You should get results similar to those in Figure 7. **<u>If your results agree, remove the line A = 0.75*A from your code.</u>**

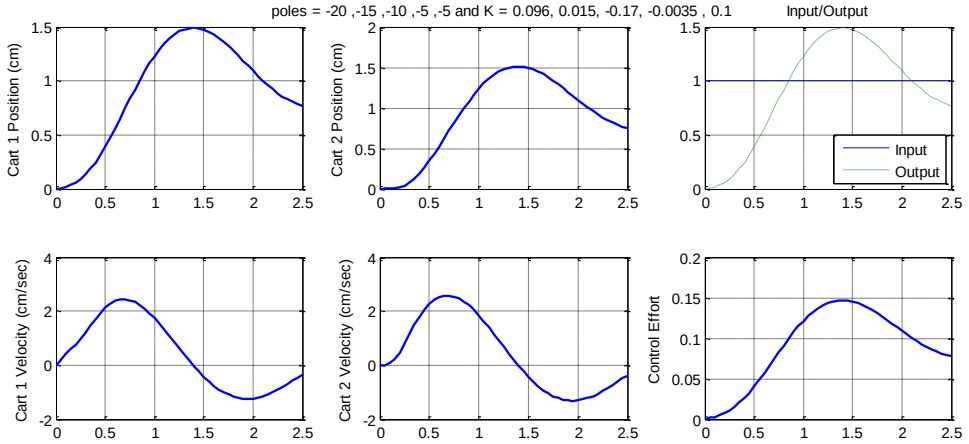


Figure 6: State feedback control of a 2DOF rectilinear system with bad data

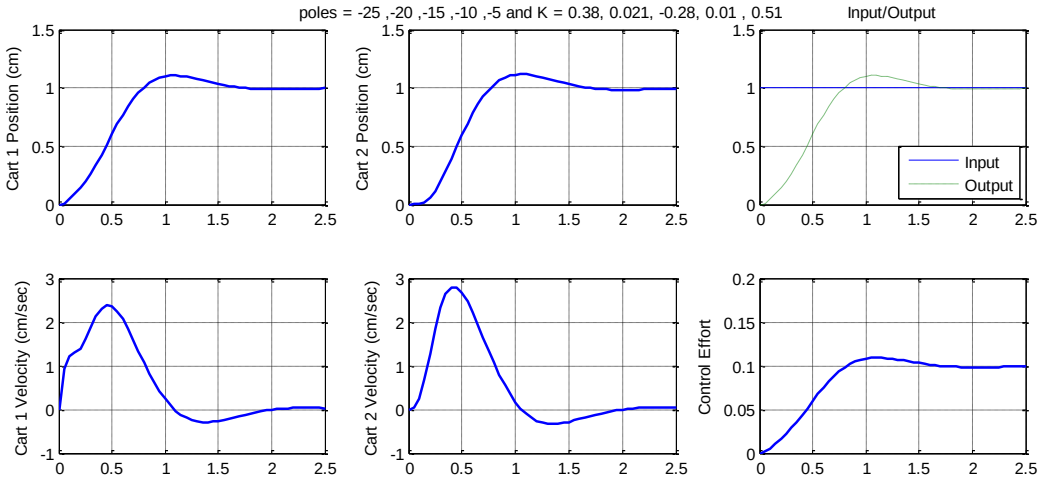


Figure 7: Improved state feedback control of a 2DOF rectilinear system with bad data

Full order observer state variable feedback for a 1DOF Model

1. In this problem we will incorporate a full order observer for a one degree of freedom system and run some test cases to verify your code is working. The full order observer state variable feedback control scheme is shown in Figure 8.

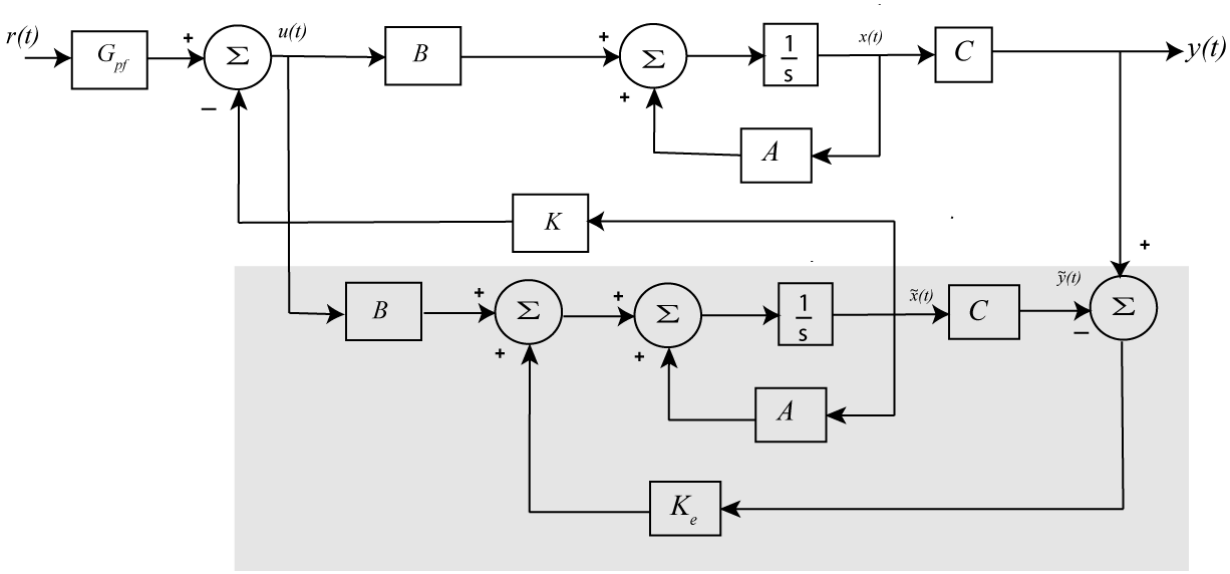


Figure 8: Full order observer state variable feedback structure

2. Copy the Simulink file, **Basic_1dof_State_Variable_Model.mdl** to the Lab 10 folder and rename it, **sv1_observer.mdl.** Modify **sv1_observer.mdl** to implement the full order

observer state variable feedback in Figure 8. This modification includes breaking the feedback from the output to the summer and inserting the observer. It also involves adding three summers, the multiplier for the A, B, and C matrices for the observer, an integrator and the gain, `Ke`. Add `To Workspace` blocks for the variables `m_x1_hat` and `m_x1_dot_hat`.

3. Copy the MATLAB file, **Basic_1dof_State_Variable_Model_driver.m** to the Lab 10 folder and rename it, **sv1_observer_driver.m**. Modify **sv1_observer_driver.m** to implement the full order observer state variable feedback in Figure 8. Set the correct saturation level for the Model 210 and load the **state_model_1dof_rect** file. The prefilter gain should also be set to `Gpf = -1/(C*inv(A-B*K)*B);`. Then find the observer feedback gains by using `Ke = place(A',C',10*p)';`. Set all of the initial conditions for the state variable model and observer to zero. Set the simulation to run the **sv1_observer.mdl** Simulink model. Finally, modify **sv1_observer_driver.m** to plot each of the true and estimated states on the same graph. See Figure 9 for an example of doing this.
4. Now verify that you have built the full order observer state feedback correctly by setting the input to 1 cm, the final time to 1 second, the state feedback poles to p = [-10 -12] and the observer to respond <u>five times</u> faster. You should get a graph similar to Figure 9.

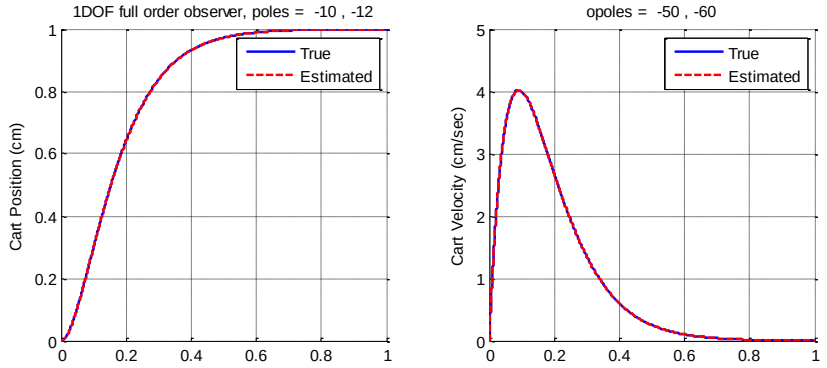


Figure 9: Results for full order observer for 1DOF system

5. Now we will try and observe what happens when there is a difference between our model and the real system. Modify the observer subsystem in the model (**.mdl**) file, so the A matrix in the observer is called **`AA`**. Just before you run the model file, enter the command **`AA=A*1.5;`**. You should get results like those in Figure 10.
6. Change the observer poles so they are <u>ten times</u> further from the imaginary axis than the state feedback poles (observer poles at `10*p`). You should get the results shown in Figure 11. If you place the observer poles <u>50</u> times away from the state feedback poles, you get the result in figure 12. Note that as the observer poles get further away, the estimates get better. **<u>Be sure to set AA=A before you go on.</u>**

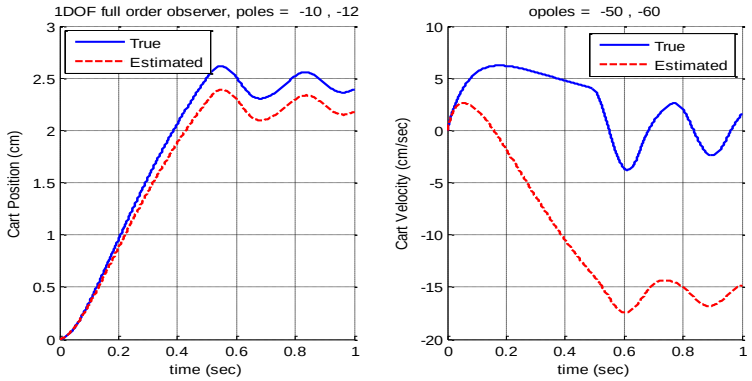


Figure 10: Full order observer for a model error for 1DOF system

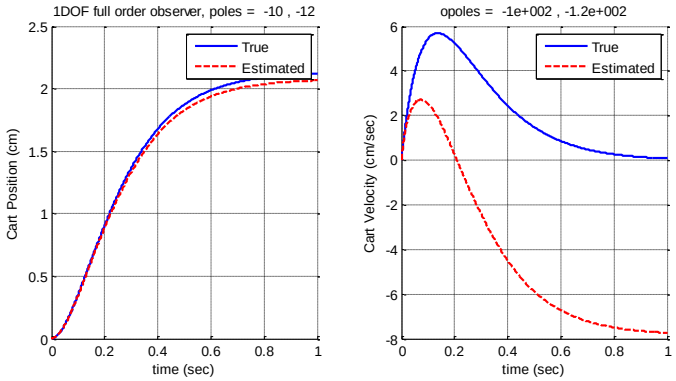


Figure 11: Full order observer for a model error for 1DOF system with 10x state feedback poles

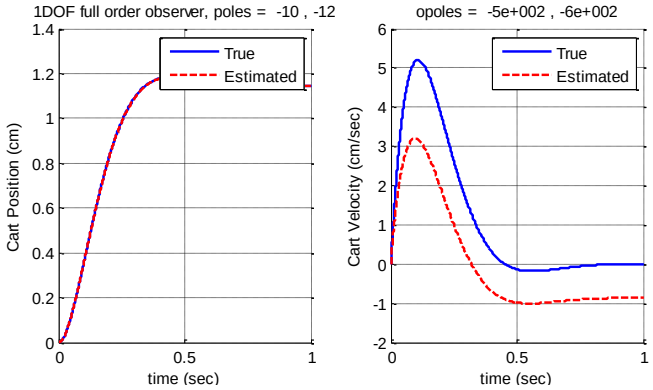


Figure 12: Full order observer for a model error for 1DOF system with 50x state feedback poles

Full order observer state variable feedback for a 2DOF Model

1. In this problem we will incorporate a full order observer for a two degree of freedom system and run some test cases to verify your code is working.
2. Copy the Simulink file **Basic_2dof_State_Variable_Model.mdl** to the Lab 10 folder and rename it, **sv2_observer.mdl.** Modify **sv2_observer.mdl** to implement full order observer state feedback for a 2DOF system. This includes adding the observer A, B, C matrices, the observer gain, an integrator and 2 summers. You should set the C matrices so that the output and estimated output are the position of cart 2.
3. Copy your program **Basic_2dof_State_Variable_Model_driver.m** to the Lab 10 folder and rename it, **sv2_observer_driver.m.** Modify **sv2_observer_driver.m** to implement state variable feedback using integral control. This includes loading the **state_model_2dof_rect** and setting the C matrix to read the cart 2 position and also setting Cy to track the cart 2 position. Adding the code to find the observer feedback gains and set the initial conditions to zero. The prefilter gain should also be set to `Gpf = -1/(Cy*inv(A-B*K)*B);`. Make sure to change the simulation to run **sv2_observer**. Finally, modify **sv2_observer_driver.m** to plot each of the true and estimated states on the same graph. See Figure 13 for an example.
4. For an input of 1 cm and a final time of 2.0 seconds, set both the state feedback and observer poles to p = [-10, -12, -14, -16] with the input to the observer the position of the second cart. You should get a graph like that in Figure 13.

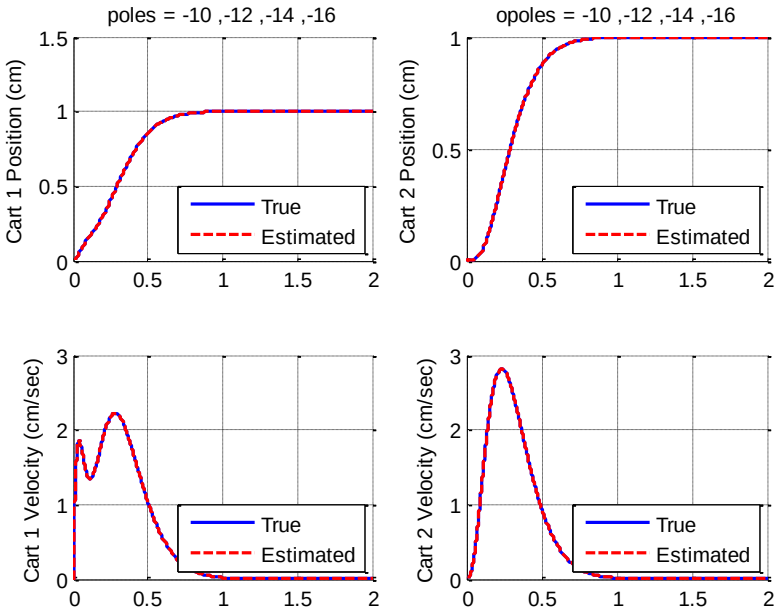


Figure 13: Full order observer for a 2DOF system

5. Now we will try and observe what happens when there is a difference between our model and the real system. Modify the observer system in the model (**.mdl**) file, so the A matrix is called AA. Just before you simulate the model file, enter the command **AA = A*1.01**. Note in Figure 14 that the estimated state for cart 1 is not as accurate and the system is almost marginally stable.
6. Now place the state feedback poles at twice their original values (at `2*p`), and change the observer poles to be ten times their original values (`10*p`). You should get results like those in Figure 15. Note that if we do not change the pole locations, we will get an unstable system.

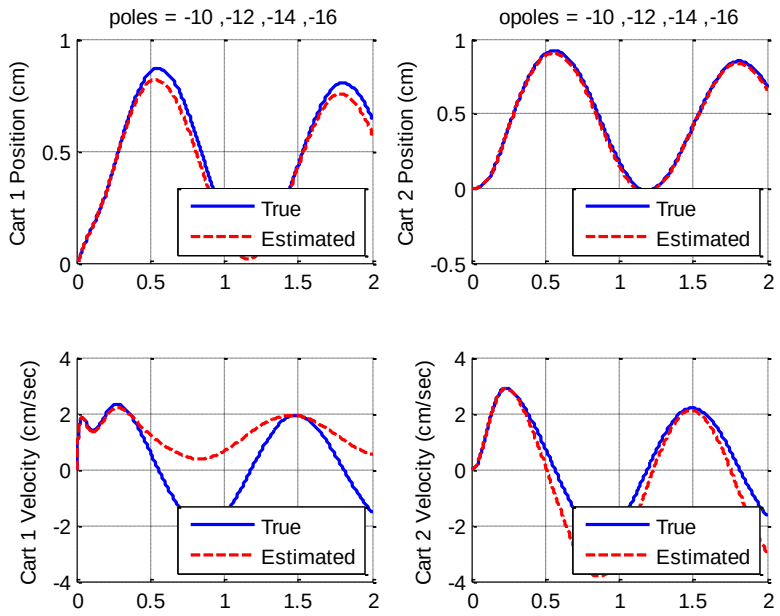


Figure 14: Full order observer for a 2DOF system with a bad model

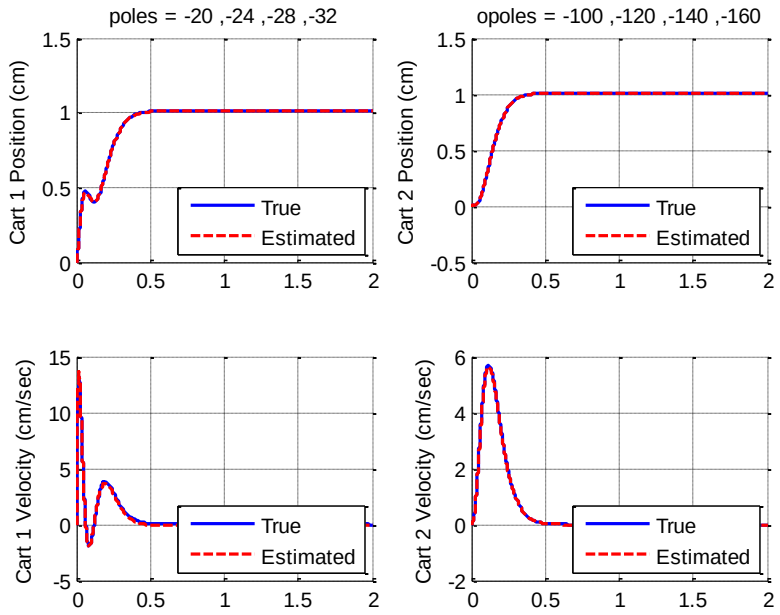


Figure 15: Full order observer for a 2DOF system with a bad model

7. Before you go on, set **AA = A**, or change your model back to its original form. Note that with the observers in the system, we do not have zero steady state error if we do not use the prefilter to cancel out the error. However, this is not the ideal solution because feedback control systems are very sensitive to parameter changes for blocks outside of the feedback loop. An alternate solution would be to add an integrator or increase the system to Type 1 and this will be reviewed next.

Type 1 Full Order Observer for a 1DOF Model

1. In this problem we will incorporate a full order observer and an integrator for a one degree of freedom system and run some test cases to verify your code is working.
2. Copy your program **sv1_observer_driver.m** to a new file **sv1_observer_int_driver.m.** Then copy the Simulink file **sv1_observer.mdl** to **sv1_observer_int.mdl.** Modify these ***new*** files to implement state variable feedback with an observer using integral control. The basic control scheme is shown below in Figure 16. Note that the prefilter is not needed in order to get zero steady state error anymore.

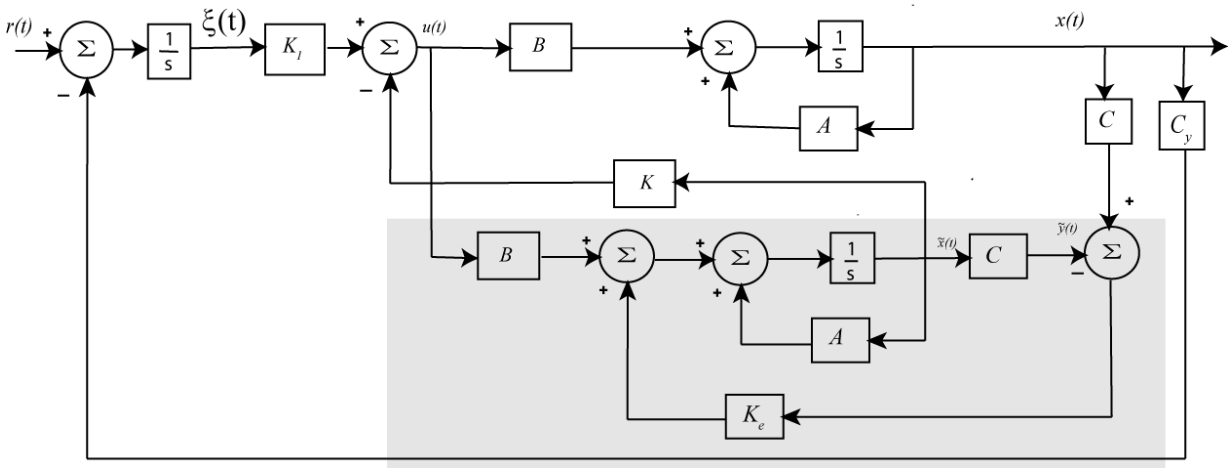


Figure 16: Full order observer state variable feedback structure with integral control.

3. For an input of 1 cm, a final time of 1 second, place the state feedback poles at `ps = [ -35, -20+10j, -20-10j]` and observer poles at `po = [-25 -30]`, you should get a graph like that in Figure 17.
4. Now we will try and observe what happens when there is a difference between our model and the real system. Modify the observer block in the model file, so the A matrix is called AA. Just before you simulate the model file, enter the command **`AA = A*1.5`**. You should get results like those in Figure 18.

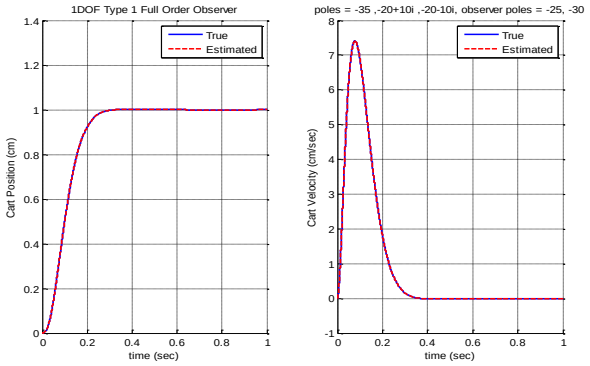


Figure 17: Full order observer state variable feedback with integrator

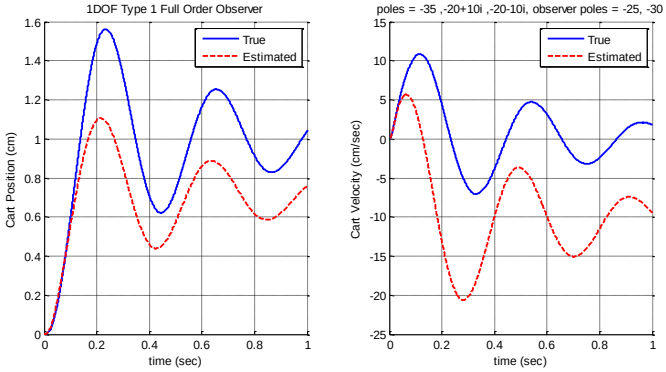


Figure 18: Full order observer with integrator with a bad model

5. Now change the observer poles so they are 10 larger than before (`10*po`). You should get results like those in Figure 19. Note that in both Figures 17 and 19, the steady state error is zero, even if the observer is not converged. Make sure to let **`AA = A`** before you move on.

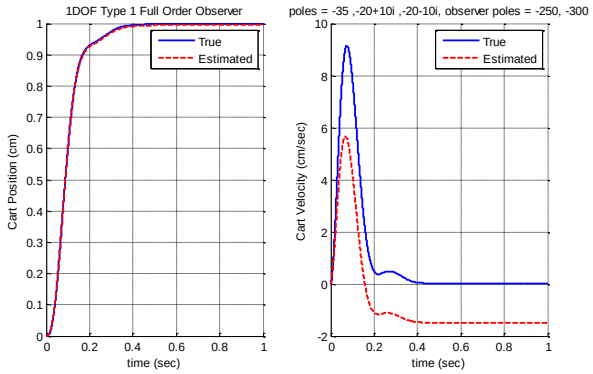


Figure 19: Full order observer with integrator with a bad model

Type 1 Full Order Observer for a 2DOF Model

1. In this problem we will incorporate a full order observer and an integrator for a two degree of freedom system and run some test cases to verify your code is working.
2. Copy your program **sv2_observer_driver.m** to a new file **sv2_observer_int_driver.m,** then copy the Simulink file **sv2_observer.mdl** to **sv2_observer_int.mdl.** Modify these ***new*** files to implement state variable feedback with an observer using integral control.
3. For an input of 1 cm, a final time of 2 seconds, set the state feedback poles at ps = [-10, -12, -14, -16, -18], the observer poles at po = [-10 -12 -14 -16], the output of the system is the position of the second cart and the input to the observer is the position of both carts, you should get a graph like that in Figure 20.
4. Now we will try and observe what happens when there is a difference between our model and the real system. Modify the observer block in the model file, so the A matrix is called

AA. Just before you run the model file, enter the command **`AA = A*1.1`**. You should get results like those in Figure 21.

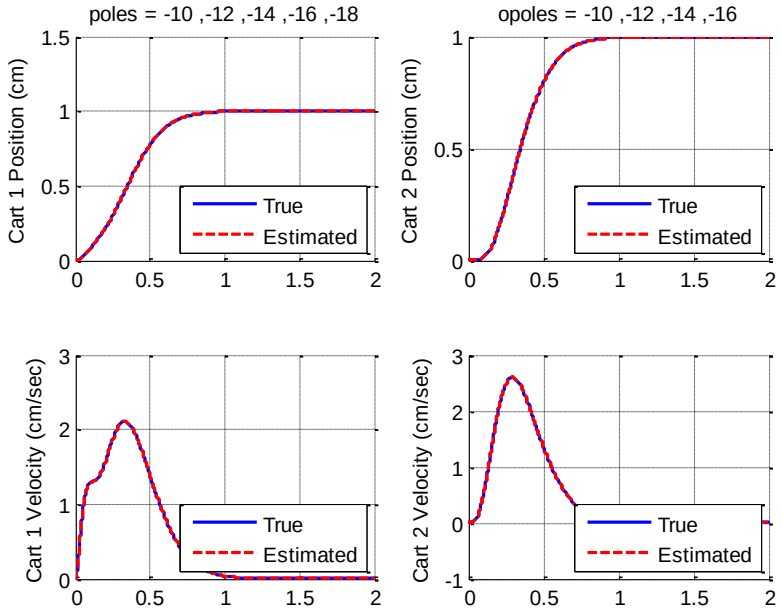


Figure 20: State feedback, full order observer, and integral control for a 2DOF system

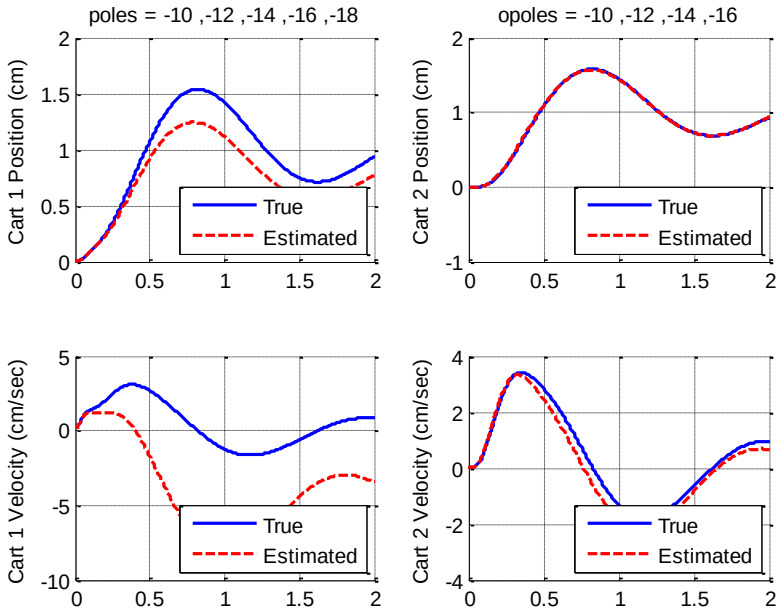


Figure 21: State feedback, full order observer, and integral control for the 2DOF system with a bad model

5. Now change the observer poles to be 10 times larger than before (`10*po`). You should get the figure like that in Figure 22.
6. Before you go on, set **AA = A**.

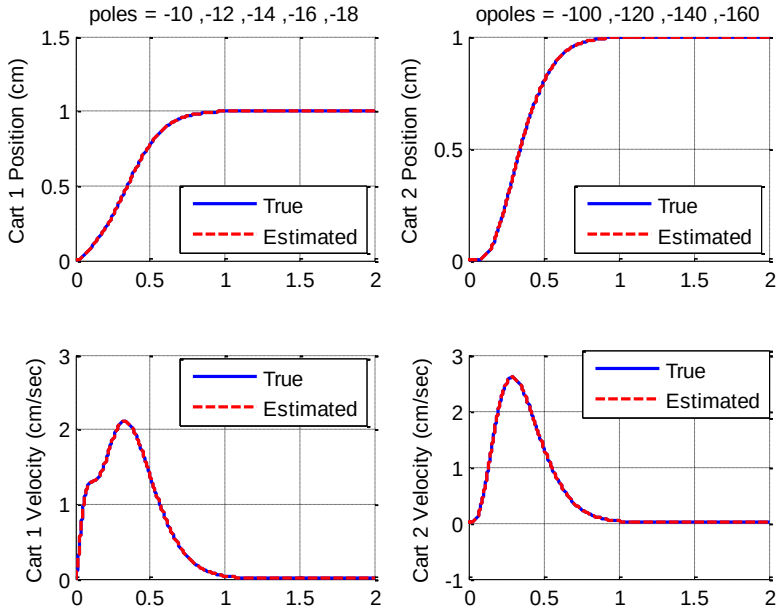


Figure 22: Using an observer to improve a Type 1 state feedback, full order observer for a 2DOF system with a bad model

*You need to turn in all **19** of the graphs you produced and also the **3** 2DOF SIMULINK models for integral control, full order observer control and Type 1 full order observer. You can copy all of these figures into a Word document and email it to your instructor. Make sure you have figure captions or labels for each.*

Theory:

Please review the lecture notes and study guide for the essential theory related to integral control, state variable feedback control and observers.

Procedure:

For this lab, you will use integral control, state variable feedback control and observers to design your 1DOF and 2DOF systems to have less than a 2 second settling time and less than a 15% overshoot. Select the state feedback controller gains and observer gains to meet this design requirement. Typically, you should make the observer at least 2 to 10 times faster than the state feedback controller. Try to limit the closed loop poles to values between -1 and -50. If you hear a grinding noise from the ECP motors when you run the system, you should move

the closed loop poles closer to the origin. Lastly, verify that the controller effort does not reach the limits.

A. File Setup

1. Create a folder under Documents\ECE320\<your initials>\Lab10
2. Copy **all** of your state variable models from Labs 2 and 3 to the folder created in step 1
3. Copy **all** of the files used and created in the prelab to the folder created in step 1
4. Copy the **Model210_sv1.mdl** from Lab 9 to the folder created in step 1 and rename it **Model210_observer_sv1.mdl**. Edit this file to match the Simulink simulation file created in the prelab for the 1DOF full order observer (**sv1_observer.mdl**) except the ECP model 210 yellow block replaces the A and B state matrices (simulated state model).
5. You may also need to copy any other required files from prior lab folders such as the ECP reset model and the **compare1.m** file.

While it may seem easier to just copy and paste the yellow ECP block, this often leads to really strange errors, ***so don't do it!***

Note: Be sure all initial estimates in your observers and integrators are set to zero before you try and control the ECP systems, and all of your simulated system errors have been removed.

B. 1DOF Full Order Observer

1. Set up your 1DOF rectilinear system with 2 masses exactly as you did when you created the model.
2. Modify the **sv1_observer_driver.m** to load the correct model for this 1DOF rectilinear system.
3. Select poles to meet the design requirements. Simulate the system with a state feedback controller and a full order observer using **sv1_observer.mdl**. You may need to vary the location of the state feedback and observer poles until you get a good response.
4. Once the design has been completed, use the driver (**sv1_observer_driver.m**) to use the full order observer to control the actual ECP system by simulating **Model210_observer_sv1.mdl**
5. Compare the actual states (from the ECP system) and the estimated states (from the observer implemented using the state model). You will have to modify **compare1.m** to do this. Be sure to change the legend command and use different line types so the results will be acceptable with black and white printers. Only plot until the system reaches steady state. You will probably have a non-zero steady state error.
6. Include the graph from the comparison in your lab memo submission.
7. Repeat part B for the 1DOF rectilinear system with 4 masses and compare the results to the system with 2 masses

C. Type 1 1DOF Full Order Observer

1. Copy the **Model210_sv1.mdl** from Lab 9 to the Lab 10 folder and rename it **Model210_observer_int_sv1.mdl**. Edit this file to match the Simulink simulation file created in the prelab for the Type 1 1DOF full order observer (**sv1_observer_int.mdl**) except the ECP model 210 yellow block replaces the A and B state matrices (simulated state model).
2. Modify **sv1_observer_int_driver.m** to load the correct state model for the 1DOF rectilinear system built.
3. Select poles to satisfy the design requirements. Simulate your rectilinear system with a state feedback controller and a full order observer, with a configuration that uses integral control. Note that you may not be able to meet a 2 second settling time requirement. **Do not try for a particularly fast response, it will overwhelm the system!** Initially put the observer poles and the state feedback poles at the same location. Note that there will be one more closed loop pole than observer pole. Verify that the controller does not reach its limits! Vary the location of the state feedback and observer poles until you get a good response.
4. Once the design has been completed, use the driver (**sv1_observer_int_driver.m**) for the full order observer with integral control to drive the actual ECP system by simulating **Model210_observer_int_sv1.mdl**.
5. Compare the actual states (from the ECP system) and the estimated states (from the observer implemented using the state model). You will have to modify **compare1.m** to do this. Be sure to change the legend command and use different line types so the results will be acceptable with black and white printers. Only plot until the system reaches steady state. Your system should have a near zero steady state error.
6. Include the graph from the comparison in your lab memo submission.
7. Repeat part C for the system with 4 masses and compare it to the results for the system with 2 masses.

D. 2DOF Full Order Observer

1. Set up your 2DOF rectilinear system as you did when you created the model.
2. Copy the **Model210_sv2.mdl** from Lab 9 to the Lab 10 folder and rename it **Model210_observer_sv2.mdl**. Edit this file to match the Simulink simulation file created in the prelab for the Type 1 2DOF full order observer (**sv2_observer.mdl**) except the ECP model 210 yellow block replaces the A and B state matrices (simulated state model).

3. Modify the **sv2_observer_driver.m** to load the correct model for the 2DOF rectilinear system. Assume the position of both carts is available to the observer and that we want to control cart 2.
4. Design the system to satisfy the design requirements. Initially, place the observer and closed loop poles at the same location and simulate the system with a state feedback controller and a full order observer using **sv2_observer.mdl**. Verify that the controller effort does not reach the limits. You may need to vary the location of the state feedback and observer poles until you get a good response. Recall that you typically want the observer to respond faster than the state variable feedback controller.
5. Once the design has been completed, use the driver (**sv2_observer_driver.m**) to use the Type 1 full order observer to control the actual ECP system by simulating **Model210_observer_sv2.mdl**
6. Compare the actual states (from the ECP system) and the estimated states (from the observer implemented using the state model). You will have to modify **compare1.m** to do this. Be sure to change the legend command and use different line types so the results will be acceptable with black and white printers. Only plot until the system reaches steady state. You will probably have a non-zero steady state error.
7. Include the graph from the comparison in your lab memo submission.

E. Type 1 2DOF Full Order Observer

1. Copy the **Model210_sv2.mdl** from Lab 9 to the Lab 10 folder and rename it **Model210_observer_int_sv2.mdl**. Edit this file to match the Simulink simulation file created in the prelab for the Type 1 2DOF full order observer (**sv2_observer_int.mdl**) except the ECP model 210 yellow block replaces the A and B state matrices (simulated state model).
2. Modify **sv2_observer_int_driver.m** to load the correct state model for the 2DOF rectilinear system built and simulate your rectilinear system with a state feedback controller and a full order observer, with a configuration that uses integral control.
3. Design the system to satisfy the design requirements. Initially try to place the closed loop poles and observer poles near each other but remember that there will be on additional closed loop pole**. Do not try for a particularly fast response, it will overwhelm the system!** Select closed loop pole values between -1 and -50. Verify that the controller does not reach its limits! Vary the location of the state feedback and observer poles until you get a good response.
4. Once the design has been completed, use the driver (**sv2_observer_int_driver.m**) for the full order observer with integral control to drive the actual ECP system by simulating **Model210_observer_int_sv2.mdl**.

5. Compare the actual states (from the ECP system) and the estimated states (from the observer implemented in the ECP system). You will have to modify **compare1.m** to do this. Be sure to change the legend command and use different line types so the results will be acceptable with black and white printers. Only plot until the system reaches steady state. Your system should have a near zero steady state error.
6. Include the graph from the comparison in your lab memo submission.

Submission:

The lab memo will be due by 5 pm on the last day of finals week. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written initials beside your name(s)
- Written in first person from you and your lab partner
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment
- A detailed description of the system setup including number of carts, spring numbers and types, number of masses and labels, include ECP system number
- All required figures with number and caption and they should be referenced in the text
- All required data in tables with error analysis referenced in the text, if necessary
- The discussion should compare and contrast the full order observer and Type 1 full order observer results for the 1DOF and 2DOF systems.
- The discussion should also compare the results of the two 1DOF system responses