Lab 10: Modeling, Simulation, and Control of a 2 Degree of Freedom Inverted Pendulum System

Overview In this lab you will model, simulate, and control both a regular pendulum on a cart and an inverted pendulum on a cart. Note that the system we are controlling is unstable!

You will be given all of the files you need for this lab, except for the 2 dof files you created last week (and used in the prelab). I have the files you need on a memory stick, so ask for them when you are ready.

For the two degree of freedom inverted pendulum system configuration from the prelab, with $q_1 = x$, $q_2 = \dot{x}$, $q_3 = \theta$, and $q_4 = \dot{\theta}$, we get the following state equations

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

We need to identify all of these quantities to get the A and B matrices for the state variable description. For our system $D = 0$ and C is determined by whatever we want the output to be.

Step 1: Set Up the System. Only the first cart should be able to move. In addition:

- There should be *at least two masses* on the first cart.
- There should be *at least one spring* in the system. If there are two springs the stiffer spring should be between the cart and the motor.
- The pendulum should be securely fastened to the first cart. It should rest on top of the masses and be securely tightened.
- 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.
- 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.

Step2a) Estimate of ω_θ

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 a simple pendulum. If the pendulum is deflected a small angle and released, it will oscillate with frequency ω_θ.

To measure this:

- Set the input in **Model210_Openloop.mdl** to 0
- Set the X-Y graph in **Model210_Openloop.mdl** to measure the position of the pendulum. You may want to change the y-min and y-max values in the X-Y graph. We are measuring angles in radians, not degrees.
- Displace the pendulum and let it go. Since we are using a small angle assumption, the pendulum should not be displaced too far.
- Using Matlab, plot the displacement of the pendulum versus time, and determine the period of the pendulum, and determine ω_θ.

Step 2b) Estimation of ω_1 *and* ζ_1

If we assume there is not input ($F = 0$) and the pendulum does not move very much ($\ddot{\theta} \approx 0$) then we have

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

Use the log-decrement method to get estimates of ω_1 and ζ_1. Not that we don't really have the case of $\ddot{\theta} \approx 0$, but this approximation is not too far off. You will need to include these log-decrement results in your memo.

Step 2c) Estimation of K_2

Applying a step input of amplitude A to the cart, estimate

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

Step 2d) Fitting the Estimated Frequency Response to the Measured Frequency Response

The transfer function between the input and the position of the first cart is given by

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

We will use this expression to determine K_1 and get better estimates of ω_1 and ζ_1, then we will have all of the parameters we need for our state variable model. We will be constructing the magnitude portion of the Bode plot and fitting this measured frequency response to the frequency response of the expected transfer function to determine these parameters. For each frequency $\omega = 2\pi f$ we have as input $F(t) = A\cos(\omega t)$ where, for out systems, A is measured in centimeters. After a transition period, the steady state output will be $x(t) = B\cos(\omega t + \theta)$ for the position of the first cart

Since we will be looking only at the magnitude portion of the Bode plot, we will ignore the phase angles.

You will go through the following steps

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

- Make sure the first cart is free to move.
- Modify **Model210_Openloop.mdl** so the input is a sinusoid. To make any changes to **Model210_Openloop**.mdl, the mode must be **Normal**.
- Set the frequency and amplitude of the sinusoid. Try a small amplitude to start, like 0.1
- Compile **Model210_Openloop.mdl** *if necessary.* This is usually not necessary so only do it when you have to (the program will let you know.)
- Connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.)
- Run **Model210_Openloop.mdl.** If the cart does not seem to move much, increase the amplitude of the input sinusoid. If the cart moves too much, decrease the amplitude of the input sinusoid. Note that if the cart hits the stops you will probably need to adjust the pendulum. *Be sure the system reaches steady state before you measure the amplitude!*
- 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. You will probably want to use the file **get_A.m**.

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** (you need to edit the file)

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

Step 3a) Modeling the Regular Pendulum

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**
- the estimated value of 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

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

- A graph indicating the fit of the transfer function from the input to the position of the cart to the measured frequency response data.

- The optimal estimates of all parameters (written at the top of the graphs)
- A file **state_model.mat** in your directory**.** This file contains the A, B, C, and D matrices for the state variable model of the system. If you subsequently type **load state_model** you will load these matrices into your workspace.
- A list of the poles and zeros of the estimated transfer function. This allows you to see how close to a pole/zero cancellation you have.

You need to be sure you have 4 points close to the resonant peaks of the transfer functions. This is particularly true if you have very small values of ζ (which correspond to very sharp peaks.) In addition, you should add points near the frequency of the pole/zero cancellation to clean things up. You may also want to simulate the system at the natural frequency of the pendulum.

You should also compare the final estimates of the parameters with your initial estimates. The values for the all of the frequencies and gains should be fairly close to the final values. The damping ratios may be quite different.

Step 3b) Controlling the ECP System with Simulink for the Regular Pendulum

a) We want first of all to try and control a regular pendulum. This is actually a *regulator* problem in that we will be trying to maintain a set point (i.e., to keep the pendulum pointing down.) Using the program **Basic_2dof__State_Variable_Model_Driver.m** and the corresponding model Simulink model file, simulate the system with zero input, and all initial conditions set to zero except for the initial pendulum displacement. Be sure to use the model file that goes with your system. Set the initial pendulum displacement to something like 0.01 rad. Utilize state variable feedback so 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. Utilize either direct pole placement or the LQR algorithm to determine the feedback gain values (I used pole values of -5 -7 -9 -11, so you might want to try with these.) You also need to be sure the control effort is less then 0.4 and the cart does not move more than about 2.5 cm. You will have to simulate your **model** a number of times to get the performance you want before you try it on the ECP system. Try to keep the first two gains (k_1 and k_2) less than 0.2 if you can.

b) In the Matlab workspace, set the final time to 20 seconds by typing Tf = 20;

c) Now try to compile **Model210_sv2.mdl** to control the ECP system with these same parameters.

d) Be sure the pendulum is pointing straight down and is at rest. Reset the system using **ECPDSPReset** (this is how the ECP finds out where zero is).

e) Connect to the system and start the system. Since there is no input the system should do nothing. Gently nudge the pendulum (or use a pencil to nudge it) and 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.***

3c) Modeling the Inverted Pendulum

Run the program **model_inverted_pendulum_full.m.** *(Note that this is a different program than you used before!)* There are 5 input arguments to this program:

- **data**, the measured data as determined by **process_data_pendulum.m**
- the estimated value of 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

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.

Step 3d) Controlling the ECP System with Simulink for the Inverted Pendulum

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 follow the following steps:

Read through the following steps before you try and run the inverted pendulum system!

a) Simulate the system with zero input, and all initial conditions set to zero except for the initial pendulum displacement. Set this to something like 0.01 rad. Utilize state variable feedback so the **first cart** moves to keep the pendulum pointing up with a settling time of less than 1.0 seconds. A really good idea is to try and use the same LQR algorithm (or feedback pole locations) as you used for the regular pendulum.

b) Be sure the pendulum is free to swing without hitting the workbench.

c) In the Matlab workspace, set the final time to 20 seconds by typing Tf = 20;

d) To control the inverted pendulum through the ECP system you must use the file **Model210_2dof_Inverted_Pendulum_Model.mdl.** Compile this with your parameters.

e) Be sure the pendulum is pointing straight down and is a rest. Reset the system using **ECPDSPReset** (this is how the ECP finds out where zero is). ***You must always do this step before you try and run the inverted pendulum!***

f) Connect to the ECP system, but do not start the Simulink. Gently rotate the pendulum up to vertical and hold it loosely between your fingers. Start the Simulink 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 (you'll know this), gently let go of the pendulum.

g) While the pendulum is vertical, gently blow on it. This should provide some external disturbance which the system should be able to overcome (***yeah feedback!***).

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

There is no memo due, but you must get me or a designated representative to sign below, and turn this in at the end of lab:

Lab partners:_____

I actually saw this think working: _____