SE420 Laboratory Assignment 10
Control of the Furuta Pendulum Using a Full Order Observer

Goals for this Lab Assignment:

1. Implement a full order observer to estimate the velocity states of the Furuta pendulum

Matlab Functions Used:

- Place.

Prelab:

Read through the entire lab assignment.

Controller Design and Simulation (Must be Done with your Lab Partner):

You will design a full order observer to estimate the velocity states of the Furuta pendulum. Using this observer along with the controller designed in Lab 9, simulate your controller and observer with the given non-linear model of the Furuta pendulum. You must complete this design and demonstrate it to your TA before proceeding to the implementation exercise.

This lab is designed to give you an introduction to designing and implementing an observer to estimate the unmeasured states of your control system. The Furuta pendulum experiment only has two measured states θ_1 and θ_2. So instead of using our velocity approximations from the previous labs, you will design a full order observer to estimate the velocities given the linear equations of motion for the Furuta about its balancing point. Again the only design specification is to stabilize the Furuta pendulum in the inverted position. You can use the same full state feedback controller that you designed in Lab 9.

The system equations remain the same as in Lab 9 with the exception of C, which now is

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

indicating that only θ_1 and θ_2 are measured. Using this along with your discrete A and B matrices from Lab 9, use pole-placement to design a full order observer with poles much faster then the closed loop poles of (A-B*K). You will find your observer works best if all the poles are placed to be faster than 0.5 in the z-domain.

The simulation for this experiment should build off the one you developed for Lab 9 and to speed things up for this lab is given to you in the link below. The simplest way to implement an observer in Simulink is to use a **Discrete State-Space** block. The block should be set up such that

A_{obs}= [A_d-L*C]

B_{obs}= [B_d L]

C_{obs}= 4x4 Identity matrix (Don't get confused with this C. This is just telling the **Discrete State-Space** block to output all four of the estimated states. This is a different C then the one you used to design the observer L gain).

D_{obs}= 4x3 zeros matrix (to match the dimensions of B_{obs} and C_{obs})

The three inputs to this state space block are the **saturated** control effort, **u**, δx_1, and δx_3. The order of these three inputs is defined by the order of B and L in B_{obs}. Since a state space block in Simulink can only have one input, those two signals will have to be multiplexed into one signal. Note- **Mux** is an abbreviation for a multiplexer. The **Mux** block is found

in the **Signal Routing** section of the Simulink library. The output of the state space block is the estimate of the four delta states of the linearized system. **u** is then calculated by multiplying **–K** by these state estimates. These are the same K gains you found in Lab 9. The Simulink model can be found at http://coecsl.ece.illinois.edu/se420/FurutaObserver.zip and is shown in Figure 1. Unzip this file in a folder and change Matlab's current directory to that folder. Your main job here is the find the correct L value that creates an observer that is able to balance the Furuta's pendulum. We will consider your simulation finished when the Furuta simulation can withstand of Pulse Disturbance of **amplitude 1.5**. Note that this Simulink model of the Furuta Pendulum already has the parameters that were given in Lab 9.

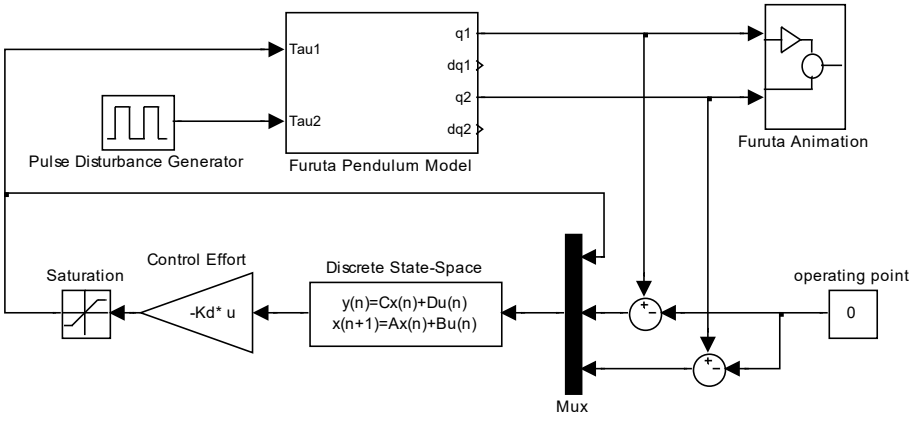

Figure 1: Simulink Model implementing Furuta Observer

Laboratory Exercise

Implement the observer and full state feedback controller on the F28379D. Use Lab 9's code as a starting point because you will be asked to plot some of the previous states in real-time. Actually, you will need to keep almost all of the lab 9 code. You will add the observer equations and then just change the balance control calculation u = -K*xhat. Remember that the state equations you need to implement in your control code are:

$$\delta\hat{x}_{k+1} = A_d\delta\hat{x}_k + B_d u_k + L(y_k - C\delta\hat{x}_k)$$
$$u_k = -K\delta\hat{x}_k$$

which we are going to implement as

$$\delta\hat{x}_{k+1} = A_{obs}\delta\hat{x}_k + B_{obs}\begin{bmatrix} u_k \\ y_k \end{bmatrix} \quad \text{where} \quad u_k = -K\delta\hat{x}_k \quad \text{and} \quad y_k = \begin{bmatrix} \theta_1 - setpt \\ \theta_2 \end{bmatrix}$$

Also leave the equations that you used to approximated velocity in Lab 9 in your code. You will use these approximations as the "actual" velocities to compare to your observed velocity states.

To help you enter your observer matrices in your C code we have written a small M-file, **matrixtoCformat,** to format the data into an array format and then you can cut and paste the arrays into your code. Unfortunately there is no matrix multiplication built into the C language. There are libraries you can find but that will take more time understanding the library then typing out the equations. Plus typing out the equations is a better educational exercise so you understand the complexity of the observer equations. Below is part of the first line of code you need to type to perform the observer equations. Also don't forget to save the past states before you exit the Timer ISR. Also make sure to saturate uk between

-10 and 10 torque command units. The uk applied to the observer equations has to be the same as what is applied to the real system and the real system is saturated at +/- 10.

```
uk = ?;  // Keep in mind here that setpt is already in x1hK1 because it entered at y1k.
ukunsat = uk;
//saturate uk +/- 10;
x1hK1 = Aobs[0][0]*x1hK+Aobs[0][1]*x2hK + ? + ? + Bobs[0][0]*uk + Bobs[0][1]*(th1-setpt) + Bobs[0][2]*th2mod;
x2hK1 = ?;
x3hK1 = ?;
x4hK1 = ?;
//save old states x1hK, x2hK, x3hK, x4hK.
//All this before your swing-up if statements  !! Make sure to use ukunsat in the two uk checks of swing-up algorithm
```

Lab Check Off:

1. Demo your working Simulink simulation using the non-linear Furuta model.
2. Demo your working full order observer controller.
3. Demo a plot of x4 estimated verses x4 actual (lab 9's approximation of velocity). Notice the convergence of the estimated state when a tap is applied to the link. When plotting here you do not have to plot the simulation. Just the two calculated velocities.