ECE-520: Discrete-Time Control Systems
Homework 7

Due: Tuesday January 31

In this homework we will simulate and study the prediction full order observer coupled with state variable feedback, and then we will add on integral control. As you hopefully remember, having an integrator in our system is generally preferable to using a prefilter for achieving a zero steady state error for a step input, since it will help overcome errors in the plant model. In this homework you will see little difference between the two methods, but in lab utilizing the ECP system you will hopefully see the benefit.

Some of the things we will do in this homework are a bit weird, but they are being done for the sake of demonstrating that your programs are working. Specifically, having the initial estimates of the states so large is quite unrealistic, but it does allow us to see if the estimated states are converging to the true states. However, these initial large states force us to utilize very small closed loop poles for the state feedback and the integrators. Strange as it may seem, we need to use these small poles to keep the system from overwhelming the motor (having the control effort reach saturation).

For the first four problems, utilize the torsional system models available from the class website. This system has been sampled with Ts = 0.05 seconds.

1a) Copy your program **DT_sv1_driver.m** to a new file **DT_sv1_observer_driver.m,** then copy the Simulink file **DT_sv1.mdl** to **DT_sv1_observer.mdl .** Modify these ***new*** files to implement state variable feedback using a full order observer to estimate the states for a one degree of freedom system. Your programs should be able to use either **place** or **acker** to place the closed loop poles directly, or **dlqr** to place the poles using the LQR algorithm. Your programs should write all of the true states and all of the estimated states to the workspace. Note that if our system has no integrator in it, there should be a prefilter. Be sure $u(k)$ is taken to be the signal just before the limiter.

b) Modify **DT_sv1_observer_driver.m** to plot the first two true states (not the delayed control signal) and estimated states on the same graph, as well as the system output and the control effort.

c) Place both the state feedback and observer poles all at 0.5 (yes, this is a bad design) The output of the system is the position of the first disk and this is also the output available to the observer ($C = C_y$ in this case). The disk is initially estimated to be at 5 degrees (be careful of units here! The model works in radians.). The initial estimates of the velocity and delayed control signal are zero. The real system starts at rest (all initial conditions are zero.) For a 10 degree step input you should get the graph shown in Figure 1. Turn in your plot.

d) Now place the observer poles all at 0.3, then at 0.2, then at 0. Rerun the system and turn in your plots. You should see the estimated states converging mores quickly to the real states as the observer poles get closer to the origin.

e) Now assume the same condition as in part **c**, but place the observer poles at 0.45, 0.5, and 0.55 (you will need to use the **place** command to place the poles.) Assume both the position of the first disk (state 1) and the delayed control signal (state 3) are available to the observer. *(Hint: C is a 2x3 matrix)* However, the output is still the position of the first disk. Simulate your system, you should get a result like that in Figure 2. Note that the observer has more information to work with here, so it does a better job of estimating the states. Turn in your plot.

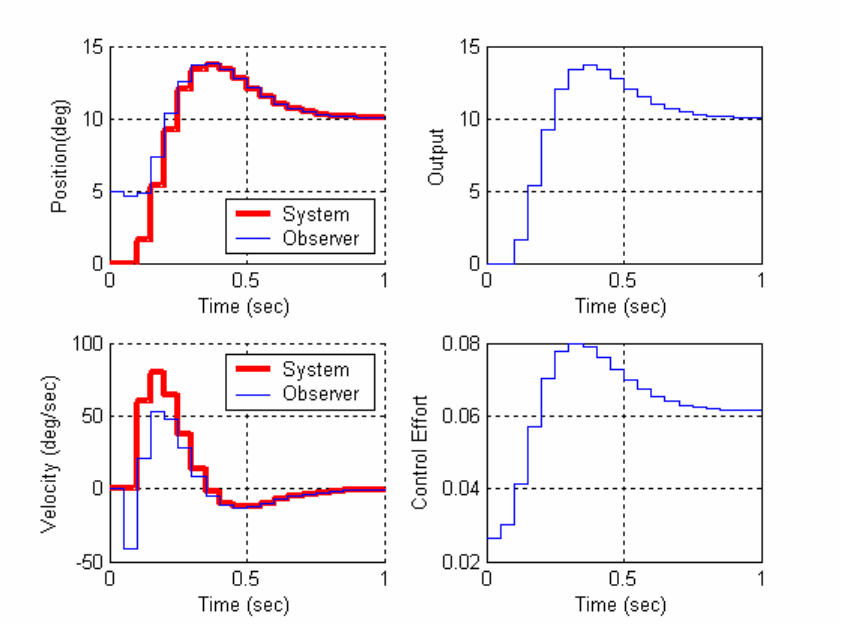

Figure 1. Results for problem **1c**.

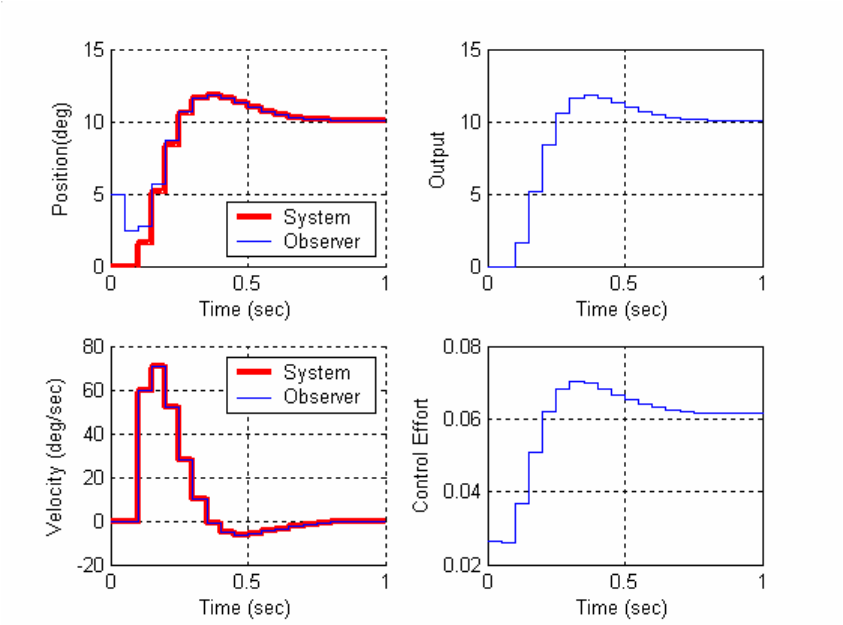

Figure 2. Results for part **1e**.

2a) Copy your program **DT_sv1_observer_driver.m** to a new file **DT_sv2_observer_driver.m,** then copy the Simulink file **DT_sv1_observer.mdl** to **DT_sv2_observer.mdl .** Modify these ***new*** files to implement both state variable feedback using a full order observer to estimate the states for a two degree of freedom system. Your programs should be able to use either **place** or **acker** to place the closed loop poles directly, or **dlqr** to place the poles using the LQR algorithm. Your programs should write all of the true states and all of the estimated states to the workspace. Note that if our system has no integrator in it, there should be a prefilter. Be sure $u(k)$ is taken to be the signal just before the limiter.

b) Modify **DT_sv2_observer_driver.m** to use the first four states for the ECP system. Specifically, you need to change get_desired_states as follows:

```
get_desired_states = [1 0 0 0 0 0 0 0;
                      0 1 0 0 0 0 0 0;
                      0 0 1 0 0 0 0 0;
                      0 0 0 1 0 0 0 0];
```

c) Modify **DT_sv2_observer_driver.m** to plot the first four true states (not the delayed control signal) and estimated states on the same graph, as well as the system output and the control effort.

d) Place all the state feedback poles at 0.2, and the observer poles at 0.06 0.08 0.1 0.12 0.14. You will probably need to use the **place** command to place the observer poles here. Assume the positions of both the first and second disk are available to the observer and the system output is the position of the first disk. The velocities of the disks and the delayed control signals are initially estimated to be zero. The estimated position of the first disk is assumed to be -5 degrees and the estimated position of the second disk is assumed to be 5 degrees. The real system is starts at rest. For a 10 degree step input you should get the graph shown in Figure 3. Turn in your plot.

e) For the same conditions as part d, but assume the observer can only use the position of the second disk. You should get the results shown in Figure 4. Turn in your plot.

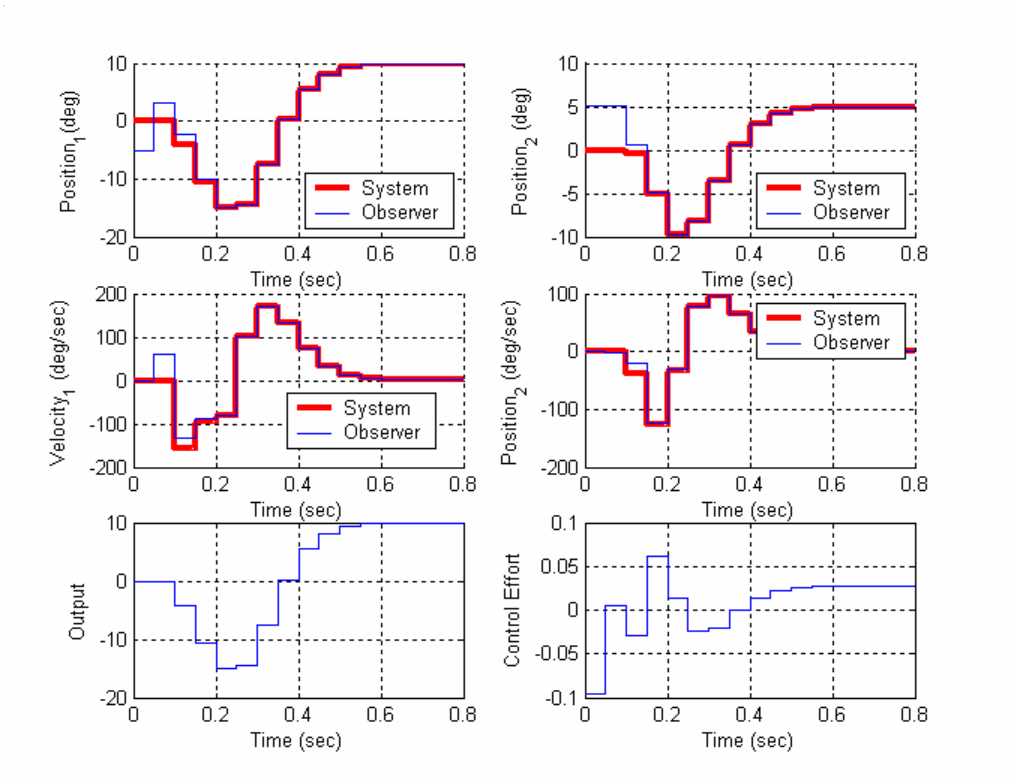

Figure 3: Results for problem **2d**.

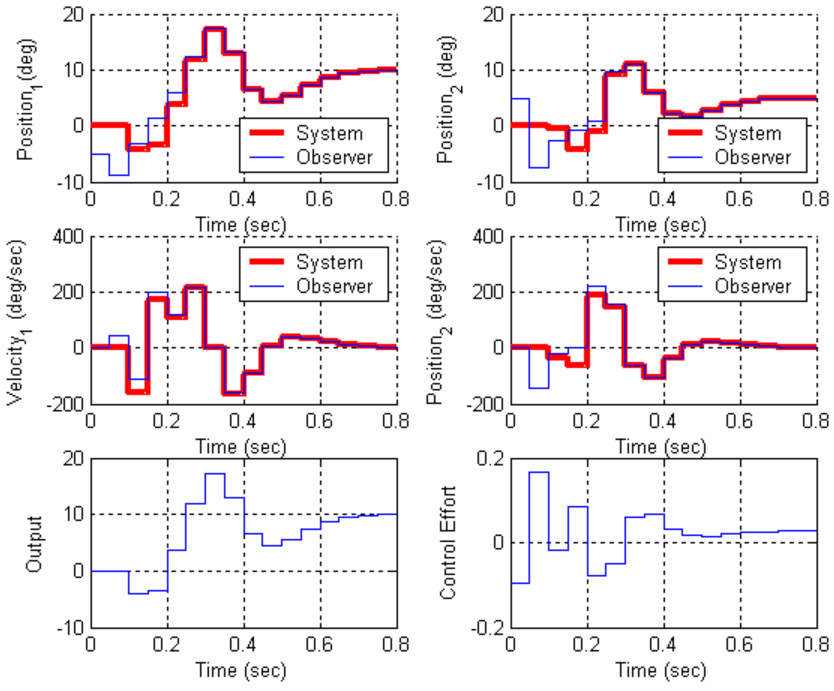

Figure 4: Results for problem **2e**.

3) Now we want to combine the observer and state feedback with integral control for the one degree of freedom system. We will continue to utilize the torisional model we have been using.

a) Copy your program **DT_sv1_observer_driver.m** to a new file **DT_sv1_obs_servo_driver.m,** then copy the Simulink file **DT_sv1_observer.mdl** to **DT_sv1_obs_servo.mdl .** Modify these ***new*** files to implement both state variable feedback using a full order observer to estimate the states and integral control.

b) Place all the state feedback poles at 0.4 and all of the observer poles at 0.2. The output of the system is the position of the first disk and this is also the output available to the observer ($C = C_y$ in this case). The disk is initially estimated to be at 5 degrees, the velocity is initially estimated to be -5 degrees/sec, and the delayed control signal is estimated to be 1 degree. The true system is initially at rest (all initial states are zero) For a 10 degree step input you should get the graph shown in Figure 5. Turn in your plot.

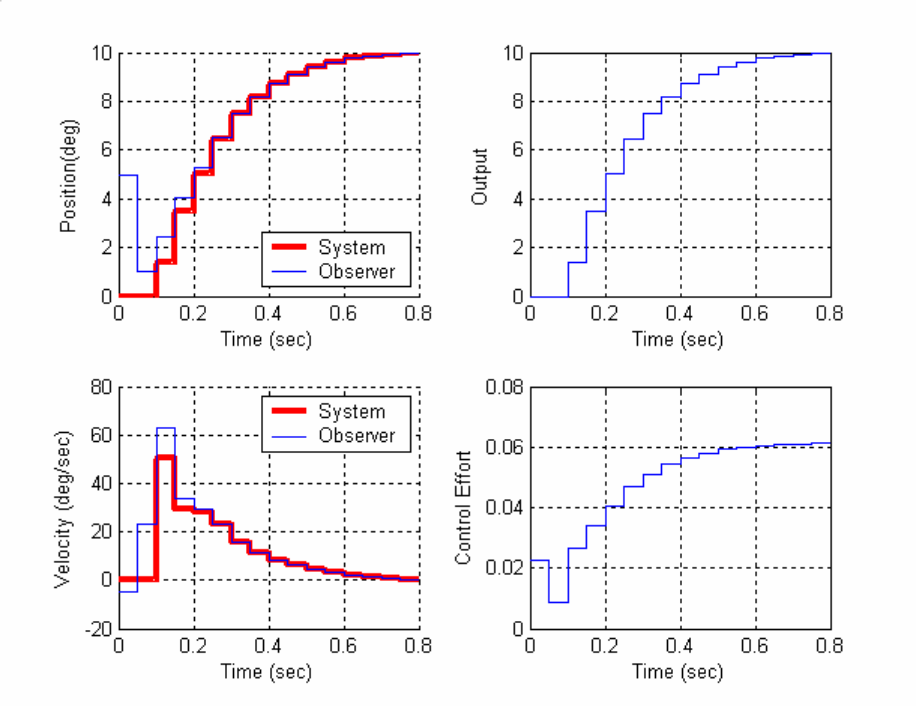

Figure 5: Results for problem **3b**.

4) Finally, we want to combine the observer and state feedback with integral control for the two degree of freedom system. We will continue to utilize the torisional model we have been using.

a) Copy your program **DT_sv2_observer_driver.m** to a new file **DT_sv2_obs_servo_driver.m,** then copy the Simulink file **DT_sv2_observer.mdl** to **DT_sv2_obs_servo.mdl .** Modify these ***new*** files to implement both state variable feedback using a full order observer to estimate the states and integral control. Note that if our system has no integrator in it, there should be a prefilter. Be sure $u(k)$ is taken to be the signal just before the limiter.

b) Place all the state feedback poles at 0.2, and the observer poles at 0.06 0.08 0.1 0.12 0.14. You will probably need to use the **place** command to place the observer poles here. Assume the position of the second disk isavailable to the observer and the system output is the position of the second disk. The estimated position of the first disk is assumed to be 5 degrees and the estimated position of the second disk is assumed to be -5 degrees. All other initial estimates are zero. The real system is starts at rest. The velocity and control inputs are assume to be zero. For a 10 degree step input you should get the graph shown in Figure 6. Turn in your plot.

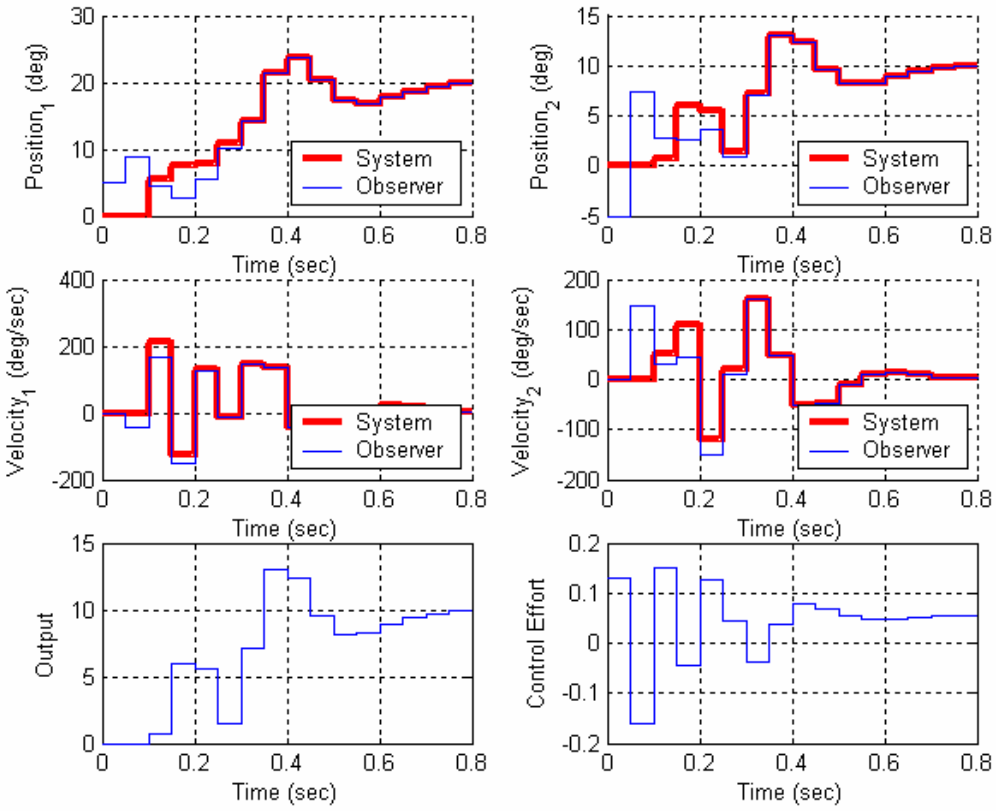

Figure 6: Results for problem **4b**.

Preparation for Lab

5) In each of the following, you should utilize whatever discrete-time model you think works best at modeling your torsional systems (one model for the one degree of freedom system and one model for the two degree of freedom system. Assume both your system and the observer states are all zero at the beginning of the simulation. (You will most likely not be able to see any difference between the estimated and true states.) Your system should meet the following contstraints:

For a 10 degree step input:

- the settling time for your system is less than 1 s
- the percent overshoot for your system is less than 20%
- the control effort does not hit a limiter (does not saturate)
- the steady state error is zero

a) For your one degree of freedom model, utilizing state variable feedback and a *prediction* full order observer (no integral control). Assume the position of the disk is available to the observer.

b) For your one degree of freedom model, utilizing state variable feedback, a *prediction* full order observer, and integral control. Assume the position of the disk is available to the observer.

c) For your two degree of freedom model, utilizing state variable feedback and a *prediction* full order observer (no integral control). Assume the positions of both disks are available to the observer, and you want to control the position of the first disk.

d) For your two degree of freedom model, utilizing state variable feedback, a *prediction* full order observer, and integral control. Assume the positions of both disks are available to the observer, and you want to control the position of the second disk.

Turn in all four plots, as well as a copy of the Simulink model for part **d** and the Matlab code for part **d**.