ECE-520: Linear Control Systems
Homework 6

Due: Thursday January 19 at 5 PM **Exam 1, Monday January 23 in Class**

1) For the discrete-time state variable system given by

$$x(k+1) = \begin{bmatrix} 0.1 & 0.2 \\ 0.2 & 0.1 \end{bmatrix} x(k) + \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(k)$$

a) Assuming state variable feedback, find two different state variable feedback matrices K to place the closed loop poles at -0.1 and -0.2.

b) Find a state variable feedback matrix K that will result in deadbeat control.

2) (*LQR Design*) In addition to direct plot placement methods, we can also use the linear quadratic regulator (LQR) method for choosing the state feedback gain K. The linear quadratic regulator finds the gain K to minimize

$$J = \sum_{k=0}^{\infty} \left[x^T(k) Q x(k) + u(k)^T R u(k) \right]$$

where

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ u(k) &= -Kx(k) \end{aligned}$$

For $x(k) \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^m$, Q is an $n \times n$ positive semi-definite matrix, and R is an $m \times m$ matrix. Since our inputs are scalars, R will be a scalar for us. In addition, we will use a diagonal matrix for Q. If we were to use the LQR method for one of our 2 degree of freedom systems, we could rewrite J as

$$J = \sum_{k=0}^{\infty} \left[q_1 x_1^2(k) + q_2 \dot{x}_1^2(k) + q_3 x_2^2(k) + q_4 \dot{x}_2^2(k) + q_5 u_d^2(k) + R u^2(k) \right]$$

A large value of R penalizes a large control signal, a large value of q_1 will penalize the position of the first cart/disk, a large value of q_2 will penalize a large value of the velocity of the first cart/disk, a large value of q_3 will penalize the position of the second cart/disk, while a large value of q_4 penalizes the velocity of the second cart/dusk. All of the q_i should be zero or positive.

For discrete-time systems it is easiest to find K using the following command in Matlab:

$$K = dlqr(G, H, diag([q_1 \quad q_2 \quad q_3 \quad q_4 \quad q_5]), R)$$

You generally have to try different values of the q_i to find an acceptable controller. Later in the course we will see how Matlab determines these gains.

a) Modify the code **DT_sv1_driver.m** to utilize the LQR algorithm to determine the state feedback gain matrix. Comment out the code previously used (using either **place** or **acker** to find the state feedback gain matrix.)

b) Utilizing your one degree of freedom ***rectilinear*** model that was obtained by discretizing your continuous time model, determine a *Q* and *R* matrix so that your system with state variable feedback meets the following design criteria (you will have to simulate your system and iterate on *Q* and *R*):

   For a 1 cm 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

c) Simulate the system and turn in your (final) graph. Write on your graph the *q* and *R* values (you don't have to do this in Matlab.)

d) Modify the code **DT_sv2_driver.m** (or whatever you called it) to utilize the LQR algorithm to determine the state feedback gain matrix. Comment out the code previously used (using either **place** or **acker** to find the state feedback gain matrix.)

e) Utilizing your two degree of freedom ***rectilinear*** model that was obtained by discretizing your continuous time model, determine a *Q* and *R* matrix so that your system with state variable feedback meets the following design criteria (you will have to simulate your system and iterate on *Q* and *R*):

   For a 1 cm step input (and assuming the position of the first cart is the output):
   - 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

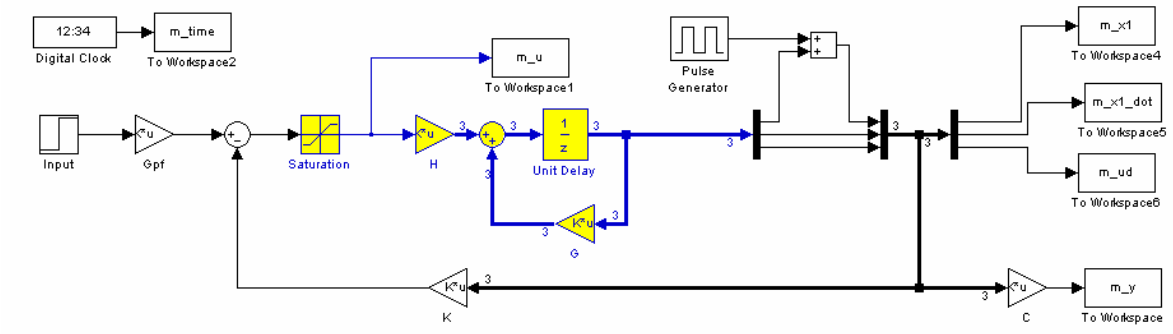
f) Simulate the system and turn in your (final) graph. Write on your graph the *q* and *R* values (you don't have to do this in Matlab.)

3) (*Regulators*) A regulator is a kind of control system that attempts to keep the system at a certain *set point*. One example of a regulator is a thermostat, which is used to keep a home/apartment/cave etc. at a certain fixed temperature. Of course, many system operate as both control systems (with inputs) and then as regulators (once the set point has been reached).

a) Using the feedback gains for the one degree of freedom system developed in problem 3, assume the input amplitude of the step is zero, but the position of the first cart is at 1 cm (you will incorporate these initial conditions in the $\frac{1}{z}$ block in the Simulink model). Run the simulation to show how the cart returns to the zero position and turn in your plot.

b) Using the feedback gains for the two degree of freedom system developed in problem 3, assume the input amplitude of the step is zero, but the position of the first cart is at 1 cm and the position of the second cart is at zero. Run the simulation to show how the cart returns to the zero position and turn in your plot.

c) Set the initial conditions to zero for parts **a** and **b** above, and set the input to a 1 cm step. Add a disturbance to the position of the first cart, as shown below for a one degree of freedom system.



For the pulse generator *disturbance*, set the Amplitude to 0.2 cm, the Period to 2 seconds, the Pulse Width to 10%, and the Phase Delay to 1 second.

Run (both) simulations (the one and two degree of freedom systems) for four seconds and turn in your plots. You should see the carts return to the 1 cm position after a short time.

4) (*Servo System with Integrator. Read Ogata pages 460-464*) Copy your program **DT_sv1_driver.m** to a new file **DT_sv1_servo_driver.m,** then copy the Simulink file **DT_sv1.mdl** to **DT_sv1_servo.mdl .** Modify these ***new*** files to implement both state variable feedback and an integrator control (*See Figure 6-18*). 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. Note that if our system has an integrator in it, there should be no prefilter.

Using this code and the LQR algorithm, show that for your one degree of freedom ***rectilinear*** system you can meet the following design constraints:

For a 1 cm 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

Turn in your graph.

Using this code and the **acker** command, design and simulate a deadbeat response for your one degree of freedom rectilinear system. Don't worry if your system saturates the control effort, you have no control over this.

Turn in your graph.

Copy your program **DT_sv1_servo_driver.m** to a new file **DT_sv2_servo_driver.m,** then copy the Simulink file **DT_sv1_servo.mdl** to **DT_sv2_servo.mdl .** Modify these ***new*** files to implement both state variable feedback and an integrator control (*See Figure 6-18*). 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. Note that if our system has an integrator in it, there should be no prefilter.

Using this code and the LQR algorithm, show that for your two degree of freedom ***rectilinear*** system you can meet the following design constraints:

For a 1 cm step input (assuming the second cart is the output):

- 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

Turn in your graph.

Using this code and the **acker** command, design and simulate a deadbeat response for your two degree of freedom rectilinear system. Don't worry if your system saturates the control effort, you have no control over this.

Turn in your graph.

Finally, turn in your code for the 2 dof system and a copy of the corresponding Simulink model file.