ECE-420: Discrete-Time Control Systems
Homework 7

Due: Friday October 26 in class

1) For the following matrix

$$A = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & 1 & 0 & 2 \end{bmatrix}$$

a) Find a set of vectors that form a basis for the null space of A.

b) Is the vector $\underline{n} = \begin{bmatrix} 2 & 2 & -2 & 2 \end{bmatrix}^T$ in the null space of A? That is, can you represent this vector as a linear combination of your basis vectors?

2) For the following matrix

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 2 & 2 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

a) Find a set of vectors that form a basis for the null space of A.

b) Is the vector $\underline{n} = \begin{bmatrix} 2 & 6 & -2 & -1 \end{bmatrix}^T$ in the null space of A? That is, can you represent this vector as a linear combination of your basis vectors?

3) For the following matrix

$$A = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 2 & 1 & 3 & 5 \end{bmatrix}$$

a) Find the rank of A (the number of linearly independent rows or columns).

b) Determine two vectors that span the null space of A.

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

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

a) Assuming state variable feedback, use Akermann's formula to find a state variable feedback matrix K to place the closed loop poles at 0 and 0.1.

b) Use Ackermann's formula to find a state variable feedback matrix K that will result in deadbeat control.

c) Assuming state variable feedback, use the direct eigenvalue assignment method to find a state variable feedback matrix K to place the closed loop poles at 0 and 0.1.

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

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

a) Assuming state variable feedback, use Ackermann's formula to find a state variable feedback matrix K to place the closed loop poles at 0 and 0.1.

b) Use Akermann's formula to find a state variable feedback matrix K that will result in deadbeat control.

c) Assuming state variable feedback, use the direct eigenvalue assignment method to find a state variable feedback matrix K to place the closed loop poles at 0 and 0.1.

6) 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, use the direct eigenvalue placement method to find two different state variable feedback matrices K to place the closed loop poles at -0.1 and -0.2.

b) Use the direct eigenvalue placement method to find a state variable feedback matrix K that will result in deadbeat control.

7) The one degree of freedom discrete –time Simulink model **DT_sv1.mdl** implements a state variable model with state feedback. This model uses the Matlab code **DT_sv1_driver.m** to drive it. Both of these programs are available on the course website. The blue and yellow blocks represent the *model* of the system which would be replaced with the ECP driver/real system if we had a lab. Note that the Matlab driver program assumes a one unit delay between the input and output.

a) Load the continuous time one degree of freedom ***torsional*** state variable description, **bobs_1dof_model205.mat.** The program will allow you to change the sampling rate if you need to.

b) Modify the Matlab code (the poles and the **place** command) to place the poles of the closed loop system in such a way so that for a 15 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

c) *Simulate the system and turn in your graphs.* Try at least two different sampling rates (do not make the sampling interval less than 0.01) so you can see how the system responds as the sampling rate is increased, and how the control effort is usually increased as the sampling interval is decreased.

d) Modify the Matlab code so your system has deadbeat response, if possible. You may need to use the **acker** command instead of place to do this. You may have to increase the sampling interval for this to work. *Turn in your graphs.*

8) Copy **DT_svl.mdl** and **DT_sv1_driver.m** to **DT_sv2.mdl** and **DT_sv2_driver.m,** respectively, and then modify the new files to work with a two degree of freedom system. All of the states except the last (delayed input) state must be plotted, as well as the control effort and the output state (y). You should use subplot(3,2) so each state has its own plot. Assume we want to control the position of the second disk (You may have to change the C matrix to do this.) You will need to modify both the initial conditions on the delay block, and you will need 5 states output from the **demux** (x1, x1_dot, x2, x2_dot, and delayed u). Then

a) Load the continuous time two degree of freedom ***torsional*** model, **bobs_2dof_model205.mat.**

b) Modify the Matlab code (the poles and the **place** command) to place the poles of the closed loop system in such a way so that for a 15 degree step input, *for the first disk*:

- 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

Simulate the system *and turn in your graph*.

c) If you do not like your response, you may change the sampling interval, but don't make the sampling interval smaller than 0.01 seconds.

d) Repeat part b (and c, if necessary) to control the position of the second disk. *Turn in your graph.*

e) Modify the Matlab code so your system has deadbeat response, if possible. You may need to use the **acker** command instead of place to do this. You may control the position of either disk*. Turn in your graph.*