ECE-320: Linear Control Systems
Homework 9

Due: Tuesday November 3, 2009 at the beginning of class

1) Consider the following characteristic equations

$$\Delta(s) = s^2 + bs + 1$$
$$\Delta(s) = s^3 + bs^2 + cs + 1$$
$$\Delta(s) = s^4 + bs^3 + cs^2 + ds + 1$$

a) Show that for the 2nd order system we need $b > 0$ for no RHP poles

b) Show that for the 3rd order system we need $b > 0$, $c > 0$, and $bc - 1 > 0$ for no RHP poles

c) Show that for the 4th order system we need $b > 0$, $c > 0$, $d > 0$ and $bcd - d^2 - b^2 > 0$ for no RHP poles

2) For $\Delta(s) = s^3 + s^2 + 2s + 2$,

a) determine if there are any poles in the RHP

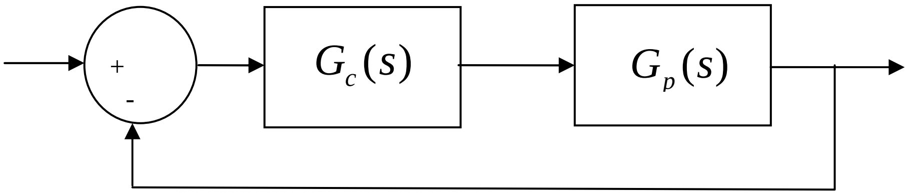
b) if possible factor the characteristic equation and determine all of the poles

3) For $\Delta(s) = s^4 + 2s^3 + 4s^2 + 6s + 3$,

a) determine if there are any poles in the RHP

b) if possible factor the characteristic equation and determine all of the poles

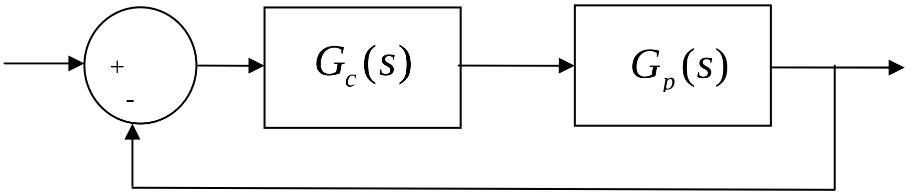
4) Consider the following control system with plant $G_p(s) = \dfrac{1}{s^2 + s + 1}$



a) For the integral controller $G_c(s) = \dfrac{k}{s}$, use the Routh array to show that there are no poles in the RHP for $0 < k < 1$. Verify your results using Matlab (either sisotool or the rlocus command).

b) For the PI controller $G_c(s) = \dfrac{k(s+z)}{s}$, with $z > 0$, show that for no RHP poles we must have $z < 1$ and $k > 0$, or for $z > 1$ we must have $0 < k < \dfrac{1}{z-1}$. Determine the factors of $\Delta(s)$. Verify your results using Matlab (either *sisotool* or the *rlocus* command).

5) Consider the following control system with plant $G_p(s) = \dfrac{1}{s^2 + 2s + 1}$



a) For the integral controller $G_c(s) = \dfrac{k}{s}$, use the Routh array to show that there are no poles in the RHP for $0 < k < 2$. Verify your results using Matlab (either sisotool or the rlocus command).

b) For the PI controller $G_c(s) = \dfrac{k(s+z)}{s}$, with $z > 0$, show that for no RHP poles we must have $0 < z < 2$ and $k > 0$, or for $z > 2$ we must have $0 < k < \dfrac{2}{z-2}$ Determine the factors of $\Delta(s)$. Verify your results using Matlab (either *sisotool* or the *rlocus* command).

6) For the plant

$$G_p(s) = \frac{K}{\frac{1}{\omega_n^2}s^2 + \frac{2\zeta}{\omega_n}s + 1}$$

a) If the plant input is $u(t)$ and the output is $x(t)$, show that we can represent this system with the differential equation

$$\frac{1}{\omega_n^2}\ddot{x}(t) + \frac{2\zeta}{\omega_n}\dot{x}(t) + x(t) = Ku(t)$$

b) Assuming we use states $q_1(t) = x(t)$ and $q_2(t) = \dot{x}(t)$, and the output is $x(t)$, show that we can write the state variable description of the system as

$$\frac{d}{dt}\begin{bmatrix} q_1(t) \\ q_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}\begin{bmatrix} q_1(t) \\ q_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ K\omega_n^2 \end{bmatrix}u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} q_1(t) \\ q_2(t) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix}u(t)$$

or

$$\dot{q}(t) = Aq(t) + Bu(t)$$
$$y(t) = Cq(t) + Du(t)$$

Determine the A, B, C and D matrices.

c) Assume we use state variable feedback of the form $u(t) = G_{pf}r(t) - kq(t)$, where $r(t)$ is the new input to the system, G_{pf} is a prefilter (for controlling the steady state error), and k is the state variable feedback gain vector. Show that the state variable model for the closed loop system is

$$\dot{q}(t) = (A - Bk)q(t) + (BG_{pf})r(t)$$
$$y(t) = (C - Dk)q(t) + (DG_{pf})r(t)$$

or

$$\dot{q}(t) = \tilde{A}q(t) + \tilde{B}r(t)$$
$$y(t) = \tilde{C}q(t) + \tilde{D}r(t)$$

d) Show that the transfer function (matrix) for the closed loop system between input and output is given by

$$G(s) = \frac{Y(s)}{R(s)} = (C - Dk)(sI - (A - Bk))^{-1}BG_{pf} + DG_{pf}$$

and if D is zero this simplifies to

$$G(s) = \frac{Y(s)}{R(s)} = C(sI - (A - Bk))^{-1} BG_{pf}$$

e) Assume $r(t) = 1$ and $D = 0$. Show that, in order for $\lim_{t \to \infty} y(t) = 1$, we must have

$$G_{pf} = \frac{-1}{C(A - Bk)^{-1} B}$$

Note that the prefilter gain is a function of the state variable feedback gain!

If matrix P is given as

$$P = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

then

$$P^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

and the determinant of P is given by $ad - bc$. This determinant will also give us the characteristic polynomial of the system.

Preparation for Lab 9

7*) You will be using this code and the following designs in Lab 9, so come prepared!*

a) The one degree of freedom Simulink model (**Basic_1dof_State_Variable_Model.mdl**) implements a state variable model for a one degree of freedom system. This model uses the Matlab code **Basic_1dof_State_Variable_Model_Driver.m** to drive it. Both of these programs are available on the course website.

a) Get the state variable model files for the systems you modeled in lab. *Since you will be implementing these controllers during lab 9, if you have any clue at all you and your lab partner will do different systems!*

*You will need to have **Basic_1dof_State_Variable_Model_Driver.m** load the correct state model into the system!*

b) You need to set the **saturation_level** to the correct level for the rectilinear (model 210) or torsional (model 205) system. Assume we have an input step of 1 cm or 15 degrees (be sure to convert to radians!)

c) Design a state variable feedback system using **pole placement** for either your torsional or your rectilinear system. For this method, we basically guess the pole locations and simulate the system. To set the location of the closed loop poles, find the part of the code that assigns poles to the variable p, and change the elements of p. You will need to choose the closed loop pole locations (This is a guess and check sort of thing. The biggest problem is making sure the control effort is not too large.) Your resulting design must have a settling time of 0.5 seconds or less and must have a percent overshoot of 10% or less. Your design should not saturate the system (control effort) and you should use a **constant prefilter**.

d) Run your simulation for 2.0 seconds. Plot both the system output (from 0 to 2 seconds) and the control effort (from 0 to 0.2 seconds). Plot the control effort only out to 0.2 seconds since the control effort is usually largest near the initial time. If your control effort reaches its limits, you need to go back and modify your design. Turn in your plot with your closed loop poles and your gains (you can just write these on your plot).

e) An alternative method for determining the feedback gains is based on what is called a **linear quadratic regulator**. The linear quadratic regulator finds the gain K to minimize

$$J = \int_0^\infty \left[x^T(t)Qx(t) + u(t)^T Ru(t) \right] dt$$

where

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ u(t) &= -Kx(t) \end{aligned}$$

For our one degree of freedom systems, Q is a 2x2 positive definite matrix, and R is a scalar. Since we will use a diagonal matrix for Q and for our system $u(t)$ is a scalar, we can rewrite J as

$$J = \int_0^\infty \left[q_1 x_1^2(t) + q_2 \dot{x}_1^2(t) + Ru^2(t) \right] dt$$

This is very similar to the quadratic optimal control we already discussed in class for transfer functions. A large value of R penalizes a large control signal, a large value of q_1 will penalize the position of the first cart, while a large value of q_2 will penalize a large value of the velocity of the first cart. All of the q_i should be zero or positive.

It's easiest to find K using the following command in Matlab: $K = lqr(A, B, diag([q_1 \quad q_2]), R)$

Try different values of the q_i to find an acceptable controller. Turn in your plot with your closed loop poles and your gains (you can just write these on your plot).

Turn in your plots and your code. Be sure your plots are accurately labeled!