ECE-320: Linear Control Systems
Homework 7

Due: Tuesday October 18 at 2 PM

1) For the plant

$$G_p(s) = \frac{1}{s(s+2)}$$

show that the first order controller that put the closed loop poles at $-1 \pm j$ and -3 is $G_c(s) = 2$.

2) For the plant

$$G_p(s) = \frac{1}{s+1}$$

a) Determine a controller so the closed loop pole is at -10, then determine a prefilter so the steady state error for a unit step is zero.

b) Determine a controller $G_c(s)$ so the system is a type 1 system and the closed loop poles are at $-2 \pm j$. Show that the resulting closed loop transfer function is

$$G_o(s) = \frac{3s+5}{s^2+4s+5}$$

and determine the steady state error for a unit ramp.

3) For the plant

$$G_p(s) = \frac{s-1}{s+1}$$

a) Determine a **strictly proper** controller $G_c(s)$ so the closed loop poles are at $-2 \pm j$, determine the closed loop transfer function, and find a constant prefilter so the steady state error for a unit step is zero. Show that the resulting closed loop transfer function (without the prefilter) is

$$G_o(s) = \frac{-s+1}{s^2+4s+5}$$

b) Determine a **strictly proper** controller $G_c(s)$ so the closed loop poles are at $-2 \pm j$ and -5 ($D_o(s) = s^3 + 9s^2 + 25s + 25$), and the system is a type one system. Show that the resulting closed loop transfer function is

$$G_o(s) = \frac{(-25-21s)(s-1)}{s^3+9s^2+25s+25}$$

Preparation for Lab 7

4) In this problem we are going to be adding a diophantine equation solver to your **closedloop_driver.m ,** then reproducing the results from problems 1 and 2 (This solver won't work for problem 3, since there are some unusual requirements in that problem.)

a) Download the function **solve_diophantine.m** from the class web site.

b) Make the plant transfer function the same as in problem 1 and set the input amplitude to 0.5.

c) Comment out all of the other controllers, and add the lines

```
p = [-1+j -1-j -3];      % desired closed loop pole locations
m = 1;                   % the order of the controller

Gc = solve_diophantine(Gp,p,m);
if (isempty(Gc))
  return;
end;
```

d) Set the **saturation_level** to something like 100, *temporarily*.

e) Run the code to check your answers for problem 1. Turn in your plot. Be sure to run the simulation long enough to reach steady state, and be sure your **constant** prefilter is set correctly to give you a steady state error of 0.

5) Modify the code to duplicate the results from problem 2. Run the code and check your answers for both parts of problem 2. Turn in your plots. Again use an input amplitude of 0.5. Be sure to run the simulation long enough to reach steady state, and be sure your **constant** prefilter is set correctly to give you a steady state error of 0.

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

a) Get the state variable model files for your systems. *Since you will be implementing these controllers during lab 7, you and your lab partner are to simulate different systems!*

You will need to have ***closedloop_driver.m*** *load the correct state model into the system and set the* **saturation level** *back to what it should have been! You should have an input of 0.5 cm or 10 degrees (converted to radians). Your output should be in degrees if you are using the Model 205.*

b) Design a type 0 controller for your system using the diophantine equation method. 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 1.0 seconds or less and must have a percent overshoot of 25% or less. Your design should not saturate the system (control effort), your controller must be stable, and you should use a **constant prefilter**.

c) Run your simulation for 2.0 seconds. Plot both the system output (from 0 to 2 second) 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 and your controller (you can just write this on your plot).

d) Modify your design from part b to implement a type 1 controller, with all other conditions the same (you may have to modify the closed loop pole locations). The prefilter should be a **constant** and the controller should be stable. Turn in your plot.

e) (*Potentially Tricky!*) Modify your design from part d to use a **dynamic** prefilter to cancel the zeros of the closed loop transfer function and produce a steady state error of zero. Here the denominator of the prefilter is the numerator of the closed loop transfer function, and the numerator is set for a position error of zero (Think!) Turn in your plot, your controller, and your prefilter (you can write the controller and prefilter on the graph). ***The prefilter must be stable***, hence the numerator must have all of its zeros in the LHP. You may have to modify your pole locations from part d. For my systems, I had some complex conjugate poles with imaginary parts larger than the real parts, but this may not help you. You will probably need to iterate here.

Turn in your code and plots.