ECE-320: Linear Control Systems
Homework 5

Due: **Thursday** January 22 at the beginning of class

1) Consider the continuous-time plant with transfer function

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

We want to determine the discrete-time equivalent to this plant, $G_p(z)$, by assuming a zero order hold is placed before the continuous-time plant to convert the discrete-time control signal to a continuous time control signal.

Show that if we assume a sampling interval of T, the equivalent discrete-time plant is

$$G_p(z) = \frac{z(0.5 - e^{-T} + 0.5e^{-2T}) + (0.5e^{-T} - e^{-2T} + 0.5e^{-3T})}{(z - e^{-T})(z - e^{-2T})}$$

Note that we have poles were we expect them to be, but we have introduced a zero in going from the continuous time system to the discrete-time system.

2) At the end of this homework is a brief review of sisotool. We will be using sisotool in this problem to design discrete-time controllers. For a discrete-time controller, all of the closed loop poles must be inside the unit circle for stability and in general, the close to the origin the poles are the fast the system moves (again, the dominant poles are those furthest from the unit circle.)

The basic transfer function form of the components of a discrete-time PID controller are as follows:

Proportional (P) term : $C(z) = K_p E(z)$ Integral (I) term: $C(z) = \frac{K_i}{1 - z^{-1}} = \frac{K_i z}{z - 1}$

Derivative (D) term : $C(z) = K_d(1 - z^{-1}) = \frac{K_d(z-1)}{z}$

PI Controller: To construct a PI controller, we add the P and I controllers together to get the overall transfer function:

$$C(z) = K_p + \frac{K_i z}{z-1} = \frac{(K_p + K_i)z - K_p}{z-1}$$

In *sisotool* this will be represented as $C(z) = \frac{K(z^2 + az)}{z(z-1)} = \frac{K(z+a)}{(z-1)}$

In order to get the coefficients we need out of the *sisotool* format we equate coefficients to get:

$$K_p = -Ka, \quad K_i = K - K_p$$

PID Controller: To construct a PID controller, we add the P, I, and D controllers together to get the overall transfer function:

$$C(z) = K_p + \frac{K_i z}{z-1} + \frac{K_d(z-1)}{z} = \frac{K_p z(z-1) + K_i z^2 - K_d(z-1)^2}{z(z-1)} = \frac{(K_p + K_i + K_d)z^2 + (-K_p - 2K_d)z + K_d}{z(z-1)}$$

In *sisotool* this will be represented as $C(z) = \frac{K(z^2 + az + b)}{z(z-1)}$

For the following parts you are to plot your step responses and include the controller you used (just write it on the plot) and turn it in. Be sure to make sure your controller forms match up to those shown above (you controller should also be in the zero/pole/gain form, as they are above.)

a) For a system with plant $G_p(z) = \dfrac{z}{z^2 + 0.5z + 0.1}$ and sampling interval $T_s = 0.1$

(enter this in Matlab as Gp = tf([1 0],[1 0.5 0.1],0.1

i. Design an I controller with a settling time less than 0.9 seconds
ii. Design a PI controller with a settling time less than 0.4 seconds
iii. Design a PID controller with complex conjugate zeros with a settling time of less than 0.2 seconds

b) For a system with plant $G_p(z) = \dfrac{0.5z + 0.2}{z^2 + 0.1}$ and sampling interval $T_s = 0.1$

(enter this in Matlab as Gp = tf([0.5 0.2],[1 0 0.1],0.1

i. Design an I controller with a settling time less than 0.6 seconds
ii. Design a PI controller with a settling time less than 0.5 seconds
iii. Design a PID controller with complex conjugate zeros with a settling time of less than 0.5 seconds

3) Consider a system with closed loop transfer function $G_o(s) = \dfrac{\alpha k_p}{s + \alpha + k_p}$. The *nominal* values for the parameters are $k_p = 1$ and $\alpha = 2$.

a) Determine an expression for the sensitivity of the closed loop system to variations in k_p. Your final answer should be written as numbers and the complex variable *s*.
b) Determine an expression for the sensitivity of the closed loop system to variations in α. Your final answer should be written as numbers and the complex variable *s*.
c) Determine expressions for the <u>*magnitude*</u> of the sensitivity functions in terms of frequency, ω
d) As $\omega \to \infty$ the system is more sensitive to which of the two parameters?

4) The file **zoh_files.slx**and **zoh_driver.m** illustrate how to model discrete-time transfer function systems from continuous-time transfer function systems in both Matlab and Simulink.

- The first system in **zoh_files.slx** illustrates how to utilize a sample and hold and a zero order hold to allow modelling a continuous-time system as a discrete-time system.
- The second system is the equivalent discrete-time constructed in Matlab. If you run the file **zoh_driver.m** you should see the results of the two simulations lie on top of each other.

For this problem turn in your plot of the outputs (the Matlab and Simulink should be the same) and print your **zoh_files.slx** file (so I can see it). This problem should be very simple, but I want you to have to see how to do some of these things in Matlab and Simulink.

5) Consider the plant

$$G_p(s) = \frac{\alpha_0}{s+\alpha_1} = \frac{3}{s+0.5}$$

where 3 is the nominal value of α_0 and 0.5 is the nominal value of α_1. In this problem we will investigate the sensitivity of closed loop systems with various types of controllers to these two parameters. We will assume we want the settling time of our system to be 0.5 seconds and the steady state error for a unit step input to be less than 0.1.

a) (*ITAE Model Matching*) Since this is a first order system, we will use the first order ITAE model,

$$G_o(s) = \frac{\omega_o}{s+\omega_o}$$

i) For what value of ω_o will we meet the settling time requirements and the steady state error requirements?

ii) Determine the corresponding controller $G_c(s)$.

iii) Show that the closed loop transfer function (using the parameterized form of $G_p(s)$ and the controller from part ii) is

$$G_o(s) = \frac{\frac{8}{3}\alpha_0(s+0.5)}{s(s+\alpha_1)+\frac{8}{3}\alpha_0(s+0.5)}$$

iv) Show that the sensitivity of $G_o(s)$ to variations in α_0 is given by $S_{\alpha_0}^{G_0} = \frac{s}{s+8}$

v) Show that the sensitivity of $G_o(s)$ to variations in α_1 is given by $S_{\alpha_1}^{G_o} = \frac{-0.5s}{s^2+8.5s+4}$

b) (*Proportional Control*) Consider a proportional controller, with $k_p = 2.5$.

i) Show that the closed loop transfer function is $G_o(s) = \frac{2.5\alpha_0}{s+\alpha_1+2.5\alpha_0}$

ii) Show that the sensitivity of $G_o(s)$ to variations in α_0 is given by $S_{\alpha_0}^{G_0} = \frac{s+0.5}{s+8}$

iii) Show that the sensitivity of $G_o(s)$ to variations in α_1 is given by $S_{\alpha_1}^{G_o} = \frac{-0.5}{s+8}$

c) (*Proportional+Integral Control*) Consider a PI controller with $k_p = 4$ and $k_i = 40$.

i) Show that the closed loop transfer function is $G_o(s) = \frac{4\alpha_0(s+10)}{s(s+\alpha_1)+4\alpha_0(s+10)}$

ii) Show that the sensitivity of $G_o(s)$ to variations in α_0 is given by $S_{\alpha_0}^{G_0} = \dfrac{s(s+0.5)}{s^2+12.5s+120}$

iii) Show that the sensitivity of $G_o(s)$ to variations in α_1 is given by $S_{\alpha_1}^{G_o} = \dfrac{-0.5s}{s^2+12.5s+120}$

d) Using Matlab, simulate the unit step response of each type of controller. Plot all responses on one graph. Use different line types and a legend. Turn in your plot and code. ***Do not*** *make separate graphs for each system!*

e) Using Matlab and subplot, plot the sensitivity to α_0 for each type of controller on ***one graph*** at the top of the page, and the sensitivity to α_1 on one graph on the bottom of the page. Be sure to use different line types and a legend. Turn in your plot and code. Only plot up to about 8 Hz (50 rad/sec) using a semilog scale with the sensitivity in dB (see next page). ***Do not*** *make separate graphs for each system!*

In particular, these results should show you that the model matching method, which essentially tries and cancel the plant, are generally more sensitive to getting the plant parameters correct than the PI controller for low frequencies. However, for higher frequencies the methods are all about the same.

Hint: If $T(s) = \dfrac{2s}{s^2+2s+10}$, *plot the magnitude of the frequency response using:*

```
T = tf([2 0],[1 2 10]);
w = logspace(-1,1.7,1000);
[M,P]= bode(T,w);
Mdb = 20*log10(M(:));
semilogx(w,Mdb); grid;
xlabel('Frequency (rad/sec)');
ylabel('dB');
```

Sisotool (Brief) Summary

Getting Started

- Type **sisotool** in the command window
- Click **close** when the help window comes up
- Click on **View**, then **Design Plots Configuration**, and turn off all plots except the **Root Locus** plot (set the **Plot Type** to **Root Locus** for **Plot 1**, and set the **Plot Type** to **None** for all other Plots)

Loading the Transfer Function

- In the **SISO Design** window, Click on **file → import.**
- We will usually be assigning *Gp(z)* to block *G* (the plant). Under the **System** heading, click on the line that indicates **G**, then click on **Browse**.
- Choose the available Model that you want assigned to G (Click on the appropriate line) and then click on **Import**, and then on **Close**.
- Click **OK** on the System Data (Import Model) window
- Once the transfer function has been entered, the root locus is displayed. Make sure the poles and zeros of your plant are where you think they should be.

Generating the Step Response

- Click on **Analysis → Response to Step Command** (the system is unstable at this point)
- You will probably have two curves on your step response plot. To just get the output, type **Analysis → Other Loop Responses**. If you only want the output, then only *r* to *y* is checked, and then click **OK**. However, sometimes you will also want the *r* to *u* output, since it shows the control effort for P, I, and PI controllers.
- You can move the location of the pole in the root locus plot by putting the cursor over the pink button and holding the left mouse button down as you move the pole locations. You should note that the step response changes as the pole locations change.
- The bottom of the root locus window will show you the closed loop poles corresponding to the cursor location if you hold down the left mouse button. However, if you need all of the closed loop poles you have to look at all of the branches.

Entering a Compensator (controller):

- Click on **Designs**, then **Edit Compensators**.
- Right click in the **Dynamics** window to enter real poles and zeros. You will be able to changes these values very easily later.

- You can again see how the step response changes with the compensator by moving the locations of the zero (grab the pink dot and slide it) and moving the gain of the system (grab the squares and drag them). Remember we need all poles of the closed loop system to be inside the unit circle for stability!

Adding Constraints

- Right Click on the Root Locus plot, and choose **Design Requirements** then either **New** to add new constraints, or **Edit** to edit existing constraints.
- At this point you have a choice of various types of constraints.
- *Remember these constraints are only exact for ideal second order systems!!!!!*

Printing/Saving the Figures:

To save a figure **sisotool** has created, click **File → Print to Figure**

Odds and Ends :

You may want to fix the axes. To do this,

- Right click on the Root Locus Plot
- Choose **Properties**
- Choose **Limits**
- Set the limits and turn the **Auto Scale** off

You may also want to put on a grid, as another method of checking your answers. To do this, right click on the Root Locus plot, then choose **Grid**

It is easiest if you use the zero/pole/gain format for the compensators. To do this click on **Edit → SISO Tool Preferences → Options** and click on **zero/pole/gain**.