**ECE-320**: Linear Control Systems
Homework 4

Due: **Friday March 29 by 5 PM (no lab this week)**

**1)** Consider the plant

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

where 3 is the nominal value of $\alpha_0$ and 0.5 is the nominal value of $\alpha_1$. 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 $\omega_o$ 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 $\alpha_0$ is given by $S_{\alpha_0}^{G_0} = \frac{s}{s+8}$

v) Show that the sensitivity of $G_o(s)$ to variations in $\alpha_1$ 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 $\alpha_0$ 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 $\alpha_1$ 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 $G_o(s) = \dfrac{4\alpha_0(s+10)}{s(s+\alpha_1)+4\alpha_0(s+10)}$

ii) Show that the sensitivity of $G_o(s)$ to variations in $\alpha_0$ 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 $\alpha_1$ 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.

e) Using Matlab and subplot, plot the sensitivity to $\alpha_0$ for each type of controller on ***one graph*** at the top of the page, and the sensitivity to $\alpha_1$ 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 below). ***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');
```

**2)** (*sisotool problem*) For the plant modeled by the transfer function

$$G_1(s) = \frac{6000}{s^2 + 4s + 400}$$

You are to design a PI controller, a PID controller with **complex conjugate zeros**, and a PID controller with **real zeros** that meet the following specifications

$$PO \le 10\%$$
$$T_s \le 2.5\,\text{sec}$$
$$k_p \le 0.5$$
$$k_i \le 5$$
$$k_d \le 0.01$$

In *sisotool*, in the LTI viewer, if you right click on the graph and select **Characteristics** you can let *sisotool* find the settling time. You should copy your step response and root locus plots to a word document, as well as including your values of the controller coefficients.

**3)** (*sisotool problem*) For the plant modeled by the transfer function

$$G_2(s) = \frac{6250}{s^2 + 0.5s + 625}$$

You are to design a PI controller, a PID controller with **complex conjugate zeros**, and a PID controller with **real zeros** that meet the following specifications

$$PO \le 10\%$$
$$PI\ T_s \le 15.0\ sec, PID\ T_s \le 0.5\ sec$$
$$k_p \le 0.5$$
$$k_i \le 5$$
$$k_d \le 0.01$$

In *sisotool*, in the LTI viewer, if you right click on the graph and select **Characteristics** you can let *sisotool* find the settling time. You should copy your step response and root locus plots to a word document, as well as including your values of the controller coefficients.

**Preparation for Lab 3 (Prelab to be turned in as part of your homework)**

**4)** In this problem we are going to be adding a PID controller to your **closedloop_driver.m** file. Once the PID controller is implemented, we can easily form any of the common controllers (P,I, PI, and PD) by settling coefficients to zero.

*You will be using this code and these designs in Lab 3, so come prepared!*

a) Get the state variable model files for one of your 1 degree of freedom systems. *Since you will be implementing these controllers during lab 3, if you have any clue at all you and your lab partner will do different systems!*

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

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

```
kp = 0.2;      % just a dummy value
ki = 0.02;     % and even dummer value
kd = 0.002;  % way stupid value

Gc = tf(kp,1) + tf(ki,[1 0]) + tf([kd 0],[1/50 1]);
```

Note that we have modified the derivative controller so that it is in series with a one pole lowpass filter with pole at 50 (about 8 Hz). This will help smooth out the derivatives.

c) You will need to be able to determine the PID controller coefficients from the controller. This is easiest done by equating coefficients. For example, if the PID controller is given by

$$C(s) = \frac{a(s^2 + bs + c)}{s}$$

show that the coefficients are determined by $k_d = a$, $k_p = ab$ , and $k_i = ac$ .

d) Using Matlab's **sisotool**, design ***two*** PID controllers (with complex conjugate zeros, one with real zeros) for your system. Initially limit your gains as in the lab

$$\begin{aligned} k_p &\leq 0.5 \\ k_i &\leq 5 \\ k_d &\leq 0.01 \end{aligned}$$

Your resulting design must have a settling time of 1.0 seconds or less and must have a percent overshoot of 25% or less. Note that **sisotool** defaults to an input of 1, that's OK for design purposes. *If you don't know how to get the correct plant transfer function, run* ***closedloop_driver.m*** *(with the correct model file) and it will put the correct transfer function Gp(s) into your Matlab workspace.*

e) Implement the PID controllers in **closedloop_driver.m**. Be sure the saturation limits are set appropriately for your ECP system (rectilinear or torsional). Use a step with amplitude 0.5 cm in your **closedloop_driver.m** file.

f) Simulate the system. 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 to part (d) and modify your designs. If your control effort is not near the limit, you can increase the gains, particularly the derivative gain.

g) Run your simulations for 2.0 seconds. Plot both the system output (from 0 to 2 seconds) and the control effort (from 0 to 0.2 seconds). Put a title on your plot to identify kp, ki, and kd. Look at previous code to determine how to do this. Turn in your plot.

Since the PID controllers makes the system a type 1 system, we don't need a prefilter to have a steady state error of zero. However, sometimes we can use the prefilter to make the transient response a bit nicer, or reduce the control effort. However, this is done at the expense of a block outside the control loop, which may be bad. Never the less, we continue anyway...

h) Our new transfer function has introduced finite zeros into the closed loop transfer function. We now want to use a **dynamic prefilter** to eliminate these zeros, so long as they are in the left half plane. We also need $G_o(0) = 1$. Hence we have

$$G_{pf}(s) = \frac{D_o(0)}{N_o(s)}$$

For us, we set the prefilter Gpf(s) to den_Go(end)/num_Go. (This should all be done in Matlab! Comment out your old code and add this new code.) This will cancel out the zeros of the closed loop system. Your numerator polynomial for $G_0(s)$, which is denoted as $N_o(s)$, should be second order. If it is not, be sure you have not removed the lines

```
num_Gp = (abs(num_Gp) > tol*ones(1,length(num_Gp))).*num_Gp;
 den_Gp = (abs(den_Gp) > tol*ones(1,length(den_Gp))).*den_Gp;
```

Rerun part (g) with the **dynamic** prefiler and turn in your plots. How have the results changed?

*Turn in your final code! You should have 4 plots to turn in: Two for the PID with real zeros (with and without dynamic prefilters) and two for the PID with complex zeros (with and without prefilters).*