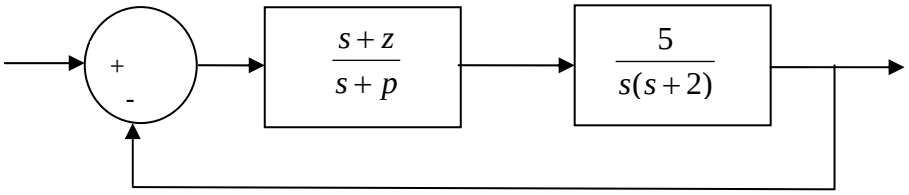
ECE-320: Linear Control Systems
Homework 6

Due: Tuesday April 26 at 10 AM

Warning: Problem 7 may take a long time. ***Do not*** *wait until the night before to attempt this problem.*

1) For the system shown below, with the lag compensator ($z > p$):



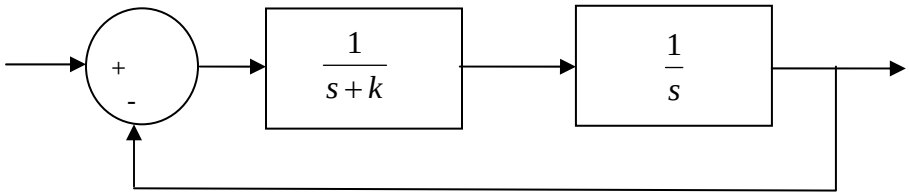
a) Show that without the lag compensator, $K_v = \frac{5}{2}$ and $e_v = \frac{2}{5}$.

b) Include the lag compensator, with z = 0.1, so the steady state error will be $0.01.\ e_v = 0.01$. (Answer: p = 0.0025).

2) Standard root locus form for determining the poles of the closed loop transfer function is

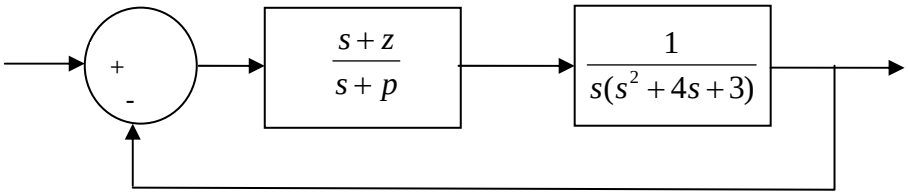
$$1 + kG(s) = 0$$

If we want to use the root locus to determine the possible pole locations for the following system,



what is $G(s)$?

3) For the system shown below:

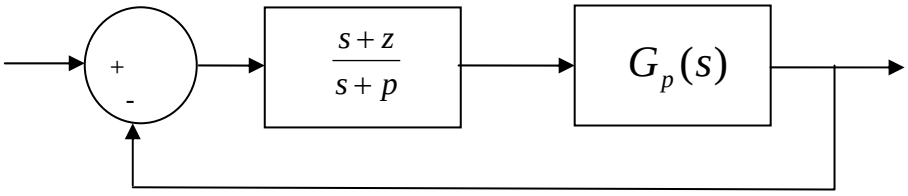


Assume we want to use the lag compensator so that $e_v = 0.1$ We will be varying the locations of the pole and zero of the lag compensator to accomplish this, and will look at the effects of these changes on both the unit step response and the unit ramp response. For each of the simulations below, run the simulation to 35 seconds. For *z* = 0.1, 0.01, and 0.001

- Find the correct value for *p* to produce the required velocity error.
- Using Matlab, simulate the unit step response for the original system (without the lag compensator) and with the lag compensator. Plot both results on one graph, as well as the input signal, using different line styles and a legend. Use the subplot command to put this on the top of the page.
- Using Matlab, simulate the unit ramp response for both the original system and the system with the lag compensator. Plot both results on one graph, as well as the input signal, using different line styles and a legend. Use the subplot command to put this on the bottom of the page.

You should notice that the large the value of z, the quicker the steady state velocity error is reduced. However, this is at the expense of large changes in the step response.

4) We have used lag compensators for obtaining the desired velocity error. It is possible to use a lag compensator to achieve a desired position error, but this is seldom done since the lag compensator tends to slow down the system. In this problem we will consider the following system

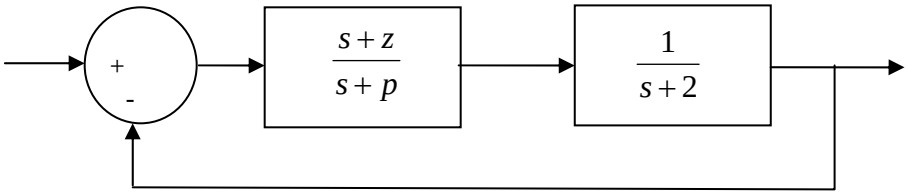


Assume that without the lag compensator, we have a known position error constant K_p.

a) Now we add the lag compensator. Show that to achieve the desired position error e_p, p and z are related by

$$p = \frac{e_p K_p z}{1 - e_p}$$

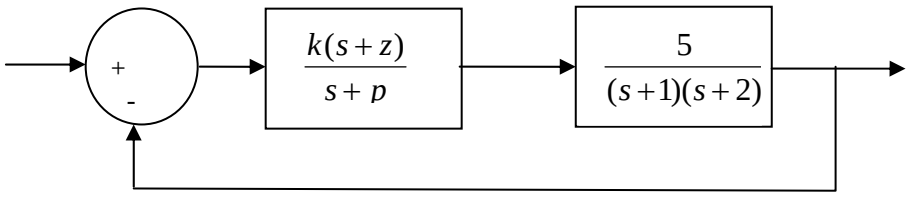
b) Using Matlab, determine the unit step response of the following system without the lag compensator.



c) For the system above, assume we want e_p = 0.1 for a unit step input. For z = 1 and 0.1 determine the corresponding value of p and simulate the system to determine the unit step input. Plot all of your results on one graph, along with the original (no lag) step response. Be sure to use different line types and a legend. Run the simulation two times, first for a final time of 2 seconds (to see the initial shape of the response is not really changing) and then for 100 seconds, to verify that the system has reached 0.9.

d) For this system, we can actually get reasonable results if we make z large enough. However, not we are using the lag controller to significantly alter the root locus. For z = 10 and 100 determine the corresponding value of p and simulate the system to determine the unit step input. Plot all of your results on one graph, along with the original step response. Be sure to use different line types and a legend. Run the simulation for a final time of 2 seconds.

5) *A different approach to the root locus...*
Consider the following closed loop control system with a lead controller ($p > z$), where *k*, *p*, and *z* are values to be determined. We would like the **dominant poles** of the closed loop system to be at $-6 \pm 2j$ and the **position error** (for a unit step) to be less than or equal to 0.08. We will assume *k*, *p*, and *z* are *positive* values.



a) Show that the closed loop poles are roots of the equation

$$1 + k\left(\frac{s+z}{s+p}\right)\left(\frac{5}{(s+1)(s+2)}\right) = 0$$

b) Since s is a complex variable, we can write the results for part (a) as a magnitude condition and a phase condition. Show that these conditions are given as

magnitude condition:

$$k\frac{|s+z|}{|s+p|}\frac{5}{|s+1||s+2|} = 1$$

phase condition:

$$\angle(s+z) + \angle 5 - \angle(s+p) - \angle(s+1) - \angle(s+2) = \pm 180^o$$

c) Since we want the point $-6+2j$ to be on the root locus, show that when we evaluate the above expressions using $s = -6+2j$ we get the equations

magnitude condition:

$$k = 4.817\sqrt{\frac{(p-6)^2+4}{(z-6)^2+4}}$$

phase condition:

$$\tan^{-1}\left(\frac{2}{z-6}\right) - \tan^{-1}\left(\frac{2}{p-6}\right) = 131.6^o$$

Note that once we choose p and z that meet the phase condition, we can always find a k to meet the magnitude condition. Hence the phase condition is the most difficult to meet.

d) Now use the "identity"

$$\tan^{-1}(r_1) - \tan^{-1}(r_2) = \tan^{-1}\left(\frac{r_1 - r_2}{1 + r_1 r_2}\right)$$

to show we can write the phase condition as

$$p = \frac{7.775z - 40}{z - 4.224}$$

Note that this identity does not take into account the two argument nature of the arctangent function. Hence the results can sometimes be off by 180 degrees. However, for most cases it is quite helpful.

e) The numerator of the above function is zero when $z = 5.144$, and the denominator is zero when $p = 4.224$. Usually places where the numerator or denominator is zero indicate where there is some type of change. To answer the following questions, if may prove useful to put in a value of z and find the corresponding p value to determine what is wrong. Remember we want a *lead controller* and the *dominant poles* at $-6 \pm 2j$

i) If $z > 5.144$ we will not get an acceptable lead controller. Why?

ii) If $4.224 < z < 5.144$ we will not get an acceptable controller. Why?

iii) If $z < 2$, we will not get an acceptable lead compensator. Why? (Hint: Sketch a root locus plot)

f) Show that the condition to meet the position error requirement can be written as

$$\frac{kz}{p} > 4.6$$

g) We must determine k, p, and z to satisfy the requirements

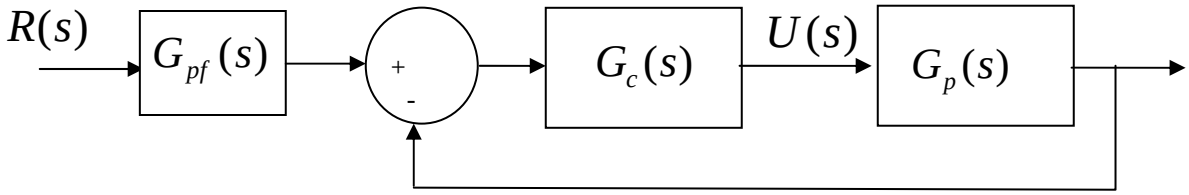
$$2 < z < 4.224 \quad p = \frac{7.775z - 40}{z - 4.224} \quad k = 4.817\sqrt{\frac{(p-6)^2 + 4}{(z-6)^2 + 4}} \qquad \frac{kz}{p} > 4.6$$

Make a table showing your guess for z, the corresponding p and k, and then kz/p. If you think about your results this should not take very long.

h) Determine the location of all of the closed loop poles and verify that the dominant poles are at $-6 \pm 2j$ (the Matlab command **rlocus** may help)

6) One of the things that will be coming up in lab more and more is the limitation of the amplitude of the control signal, or the *control effort*. This is also a problem for most practical systems. In this problem we will do some simple analysis to better understand why Matlab's sisotool won't give us a good estimate of the control effort for some types of systems, and why dynamic prefilters can often really help us out here (see Problem 7 also).

a) For the system below,



show that U(s) and R(s) are related by

$$U(s) = \frac{G_c(s)G_{pf}(s)}{1+G_p(s)G_c(s)}R(s)$$

b) For many types of controllers, the maximum value of the control signal is just after the step is applied, at $t = 0^+$. Although most of the time we are concerned with steady state values and use the final value Theorem in the *s*-plane, in this case we want to use the initial value Theorem, which can be written as

$$\lim_{t\to 0^+} u(t) = \lim_{s\to\infty} sU(s)$$

If the system input is a step of amplitude *A*, show that

$$u(0^+) = \lim_{s\to\infty} \frac{AG_c(s)G_{pf}(s)}{1+G_p(s)G_c(s)}$$

This result shows very clearly that the initial control signal is directly proportional to the amplitude of the input signal, which is pretty intuitive.

c) Now let's assume

$$G_c(s) = \frac{N_c(s)}{D_c(s)}$$

$$G_p(s) = \frac{N_p(s)}{D_p(s)}$$

$$G_{pf}(s) = \frac{N_{pf}(s)}{D_{pf}(s)}$$

If we want to look at the initial value for a unit step, we need to look at

$$u(0^+) = \lim_{s\to\infty} \frac{sG_c(s)G_{pf}(s)}{1+G_c(s)G_p(s)} \frac{1}{s} = \lim_{s\to\infty} \frac{G_c(s)G_{pf}(s)}{1+G_c(s)G_p(s)}$$

Let's also then define

$$\tilde{U}(s) = \frac{G_c(s)G_{pf}(s)}{1+G_c(s)G_p(s)}$$

so that

$$u(0^+) = \lim_{s\to\infty} \tilde{U}(s)$$

Show that

$$\tilde{U}(s) = \frac{N_{pf}(s)}{\left(\frac{D_c(s)}{N_c(s)}\right)D_{pf}(s) + \left(\frac{N_p(s)}{D_p(s)}\right)D_{pf}(s)}$$

and

$$\deg \tilde{U} = \deg N_{pf} - \max\left[\deg D_c - \deg N_c + \deg D_{pf}, \deg N_p - \deg D_p + \deg D_{pf}\right]$$

where $\deg Y$ is the degree of polynomial Y.

d) Since we are going to take the limit as $s \to \infty$, we need the degree of $\tilde{U}(s)$ to be less than or equal to zero for a step input to have a finite $u(0^+)$. Why?

For our 1 dof systems in lab, we have $\deg N_p = 0$ *and* $\deg D_p = 2$. *Use this for the remainder of this problem*

e) If the prefilter is a constant, show that in order to have a finite $u(0^+)$ we must have

$$\deg D_c \geq \deg N_c$$

f) If the numerator of the prefilter is a constant, then in order to have a finite $u(0^+)$ we must have

$$\deg D_c - \deg N_c + \deg D_{pf} \geq 0 \text{ or } -2 + \deg D_{pf} \geq 0$$

g) For P, I, D, PI, PD, PID, and lead controllers, determine if $u(0^+)$ is finite if the prefilter is a constant.

Note: Although it may appear that the control effort is sometimes infinite, in practice this is not true since our motor cannot produce an infinite signal. This large initial control signal is referred to as a *set- point kick*. There are different ways to implement a PID controller to avoid this, and if I really loved you I'd show you how.

Preparation for Lab 7

7) 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 7, so come prepared!

a) Get the state variable model files for one rectilinear (model 210) and one rotational (model 205) systems you modeled in lab (only the 1 dof systems). *Since you will be implementing these controllers during lab 7, if you have any clue at all you and your lab partner will do at least one different system!*

*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 a PID controller (with complex conjugate zeros) for your *rectilinear* (Model 210) 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 0.6 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 controller 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 design.. If your control effort is not near the limit, you can increase the gains, particularly the derivative gain.

g) 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). 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 controller makes the system a type 1 system, we don't need a prefilter to have a position error of zero. Changing the prefilter will also not have any effect on the velocity error. 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 $G_{pf}(0) = 1$. Hence we have

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

For us, we set the prefilter Gpf(s) to num_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 Go(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,den_Gp] = ss2tf(A,B,C,D);
%
% you are not responsible for the following voo doo
%
tol = 1e-6;
 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 plot. How have the results changed?

i) Using Matlab's sisotool, design a PID controller (with complex conjugate zeros) for your torsional (model 205) system. Use a step amplitude of 15 degrees in your

closedloop_driver.m file. *Remember to convert to radians!* Your resulting design must have a settling time of 1.0 seconds or less and must have a percent overshoot of 25% or less. Repeat steps e-h for this system.

Turn in your final code!