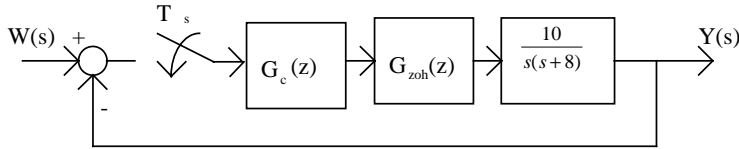


EE572 Solution to HW #17

1. Consider the system from HW#16:



Recall that we have already designed a lead compensator, $G_c(z)$, to meet the following specifications:

$$t_s \leq 0.4 \text{ sec and } M_p \leq 2\%$$

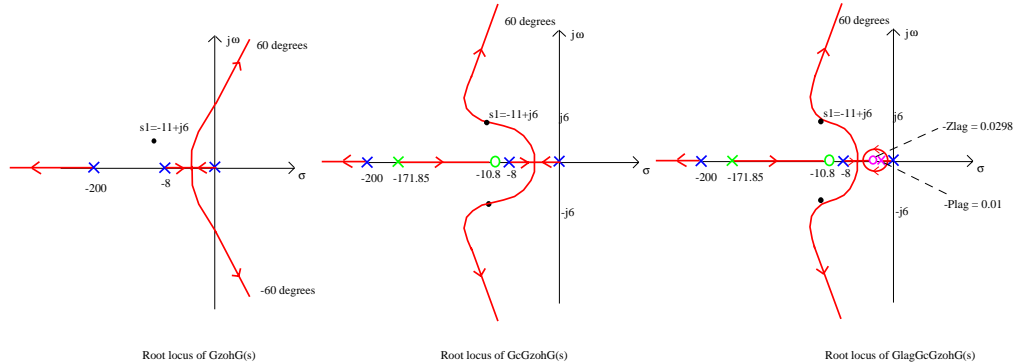
(see solution to HW#16 on the Web for one possible $G_c(z)$).

- a) Now design a $G_{lag}(z)$ to be put in series with $G_c(z)$ so that the compensated system meets the above specs plus the added spec that e_{ss} due to a unit ramp $\leq 1/50$.

Solution: using $G_c(s)$ from HW#15, we find that $K_v = \lim_{s \rightarrow 0} s G_c G_{zoh} G(s) = (213.08 \times 10.8 \times 10) / (171.85 \times 8) = 16.75$. Thus, we need to increase the steady-state gain by a factor of $50/16.75 = 2.986$ to meet our e_{ss} specs. Let $z_{lag}/p_{lag} = 2.986$ and pick $p_{lag} = 0.01$. Thus, $z_{lag} = 0.02986$. To find K_{lag} , use the magnitude condition: $K_{lag} = 1 / | \frac{s+0.02986}{s+0.01} G_c G_{zoh} G(s) |_{s=s_1} = 1.0014$. Therefore, the lag compensator which must be inserted is $G_{lag} = 1.0014(s+0.02986)/(s+0.01)$. Using the bilinear transformation to find $G_{lag}(z)$ produces $G_{lag}(z) = G_{lag}(s) \Big|_{s=\frac{2(z-1)}{T_s(z+1)}} = 1.0015(z-0.9997)/(z-0.9999)$.

- b) Sketch the compensated root locus.

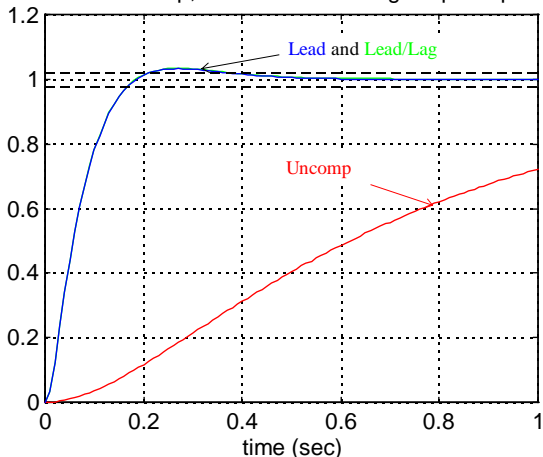
Solution: the original root locus of $G_{zoh}G(s)$, the root locus of the $G_c G_{zoh}G(s)$, and the root locus of $G_{lag} G_c G_{zoh}G(s)$ are shown below. Note that the lead/lag system root locus does not quite pass thru the desired dominant pole of $s_1 = -6 + j11$:



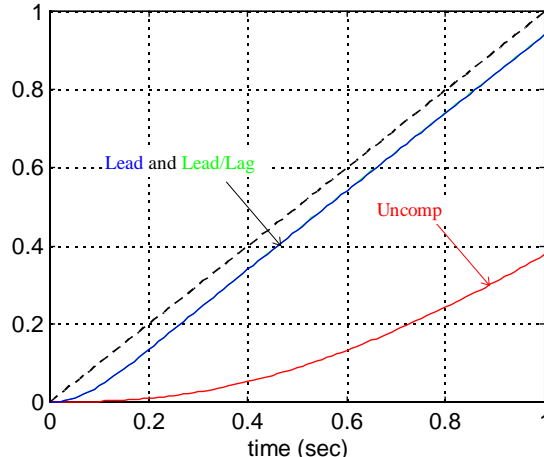
- c) Simulate (See Below)* a step and ramp response for your original closed-loop system, your lead compensated design from HW#16 (see back), and your new lead/lag system using MATLAB and lsim(). Measure t_s , M_p , and e_{ss} (both step and ramp) for all three cases. Does your design meet specs? (Hint: for the lead/lag system, you may have to increase your time axis to measure e_{ss})

Solution: (see Matlab code in appendix)

EE572 - Uncomp, Lead and Lead/Lag Step Responses

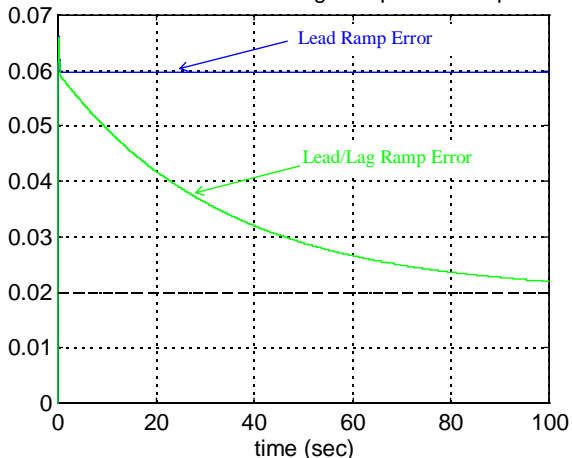


EE572 - Uncomp, Lead and Lead/Lag Ramp Responses



Looking at the step response, we see that the settling time for both the lead and the lead/lag compensated systems is about 0.38 seconds while the percent overshoot is just barely above 2%. However, the lead and the lead/lag step and ramp responses look nearly identical (recall that the steady-state ramp error should be about 2.99 times smaller for the lead/lag system). If we expand our time scale and plot the ramp error, we find:

EE572 - Lead and Lead/Lag Ramp Error Responses



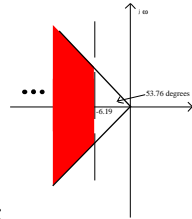
Here, with the time scale expanded to 100 seconds, we can clearly see that the steady-state ramp error for the lead/lag compensated system is tending to $1/50=0.02$ as expected. The reason this response requires so much time to reach steady state is that the lag compensator introduces a closed-loop pole at about $s=-0.03$ (see root locus). The settling time for this closed-loop pole is about $4/0.03 = 133$ sec.

2. a) Now design a compensator for the system in part a) that meets all of the following specifications (Hint: Think PID):

- $t_s \leq .65$ sec
- $M_p \leq 10\%$
- e_{ss} due to a ramp = 0
- e_{ss} due to a unit parabola $\leq 1/50$

Solution: from the first spec, we find that $t_s \leq 0.65 \Rightarrow \zeta\omega_n \geq 6.158$. From the second spec we find that $M_p = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} \times 100\%$. Or, solving this relationship for the damping coefficient we obtain, $\zeta = \sqrt{\frac{\ln(m)^2}{\pi^2 + \ln(m)^2}}$ where $m=M_p/100\% = 0.1$. Thus,

$$\zeta = \sqrt{\frac{\ln(0.1)^2}{\pi^2 + \ln(0.1)^2}} = 0.5912 = \cos \theta. \text{ Hence, } \theta = \cos^{-1}(\zeta) = \cos^{-1}(0.5912) = 53.76^\circ. \text{ Since we must have } M_p \leq 10\%, \text{ our constraint}$$



becomes $\theta \leq 53.76^\circ$. In the s-plane, we obtain the following region:

Let's pick desired dominant poles at $s_1 = -6.5 + j6$.

Next, from the two steady-state error specifications it is clear that we need a type 2 system. Since we only have a type one system, we must use a PID compensator to meet these specs! Thus, $G_c(s) = G_{PID}(s) = K_i/s + K_p + K_d s$. First, let's find K_i from the steady-state error spec: $K_a = \lim_{s \rightarrow 0} s^2 G_{PID} G_{zoh} G(s) = (K_i \times 10 \times 200) / (200 \times 8) = 1.25 K_i$. Since we want an acceleration error coefficient of $K_a = 50$, we

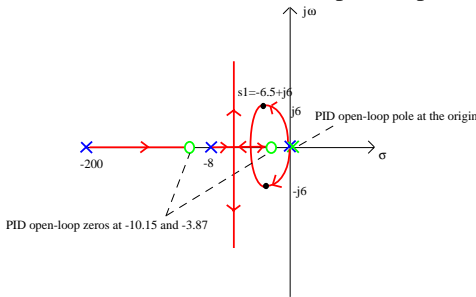
find that $K_i = 40$. Next, we can use the equation, $G_{PID} G_{zoh} G(s_1) = (K_i / s_1 + K_p + K_d s_1) G_{zoh} G(s_1) = -1$ to find K_p and K_d :

$K_p + K_d(-6.5 + j6) = -1 / G_{zoh} G(s_1) - K_i / s_1 = 7.659 + j6.1068$. Equating the imaginary parts of this equation we obtain $K_D = 6.1068/(6) = 1.0178$. Equating the real parts of the equation produces $K_P = 7.659 + 6.5K_D = 14.2747$. Thus, the final compensator is $G_{PID}(s) = 40/s + 14.2747 + 1.0178s$. In the z-plane (using the non-recommended bilinear transformation) we find that $G_{PID}(z) = G_{PID}(s) \Big|_{s = \frac{2(z-1)}{T_s(z+1)}} = 0.2(z+1)/(z-1) + 14.2747 + 203.56(z-1)/(z+1)$ (Note: we will learn in class today why this is a terrible method

of obtaining a digital PID controller!!)

b) Sketch the compensated root locus.

Solution: When we insert the PID compensator, $G_{PID}(s) = (1.0178s^2 + 14.2747s + 40)/s$ in the forward loop, we introduce an open-loop pole at the origin (to increase the system type number) as well as two zeros corresponding to the roots of $(1.0178s^2 + 14.2747s + 40)$. Using Matlab, we find the two open-loop zeros at $s = -10.1549$ and $s = -3.8701$. Thus, the compensated root locus looks like:

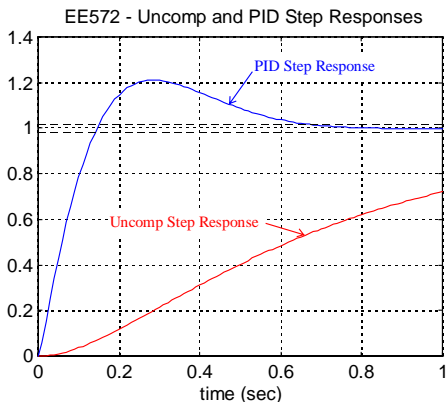


Root locus of GpidGzohG(s)

Note the compensated root-locus does indeed pass through the desired dominant poles.

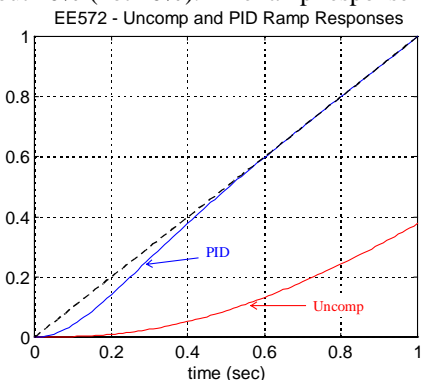
c) Simulate a step, ramp, and unit parabola responses for your PID compensated closed-loop system using MATLAB. Measure t_s , M_p , and e_{ss} (for step, ramp and parabola (a unit parabola is defined as $0.5t^2 u(t)$)).

Solution: (see Matlab code in appendix). The uncompensated and PID compensated step response looks like:

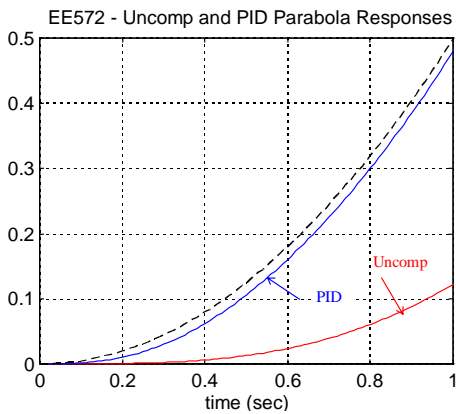


Note that the settling time is about 0.65 sec (which meets specs), but the overshoot is

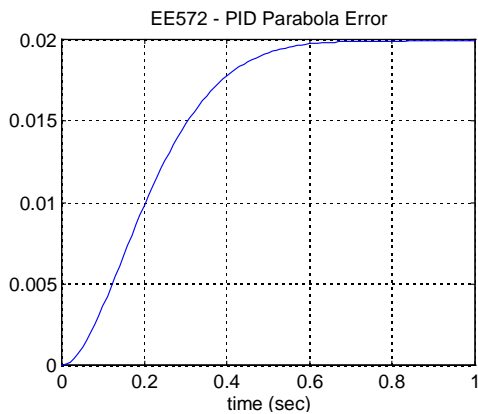
about 20% (not 10%). The ramp response looks like:



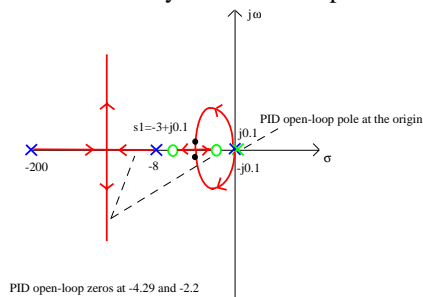
Note that the steady-state error is indeed zero! Finally, the parabola response looks like:



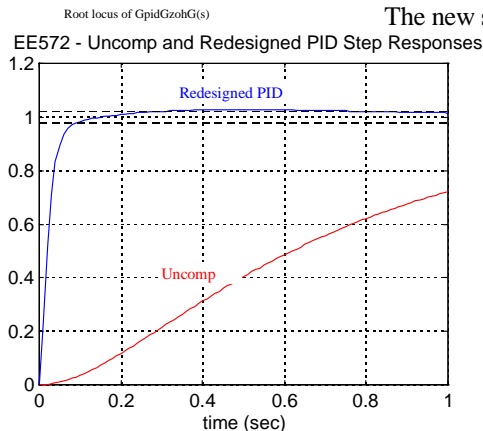
To verify that the parabola error does indeed go to $1/50=0.02$ let us plot it below:



Extra Design: Just for fun, let us redesign our PID controller to reduce the overshoot. Since the solution of the PID parameters (K_i , K_p , and K_D) is unique once the steady-state error and desired dominant poles are specified, we need to change our desired dominant poles to alter our PID design. Looking at the PID step response, we see that if we could reduce the initial overshoot to less than 2%, we would easily meet both the settling time and overshoot criteria! By experimenting with different dominant pole placements, we arrive at $s_1 = -3 + j0.1$. This means that $K_p = 27.55$ and $K_D = 4.25$. The new PID design introduces open-loop zeros at -4.29 and -2.2 which, in effect, are cancelled by the closed-loop dominant poles. The new root locus looks like:



The new step response looks like:



The new settling time is about 0.62 seconds and the overshoot is about 3%. Notice that because the open-loop zeros are, in effect, canceled by the closed-loop dominant poles, the system almost behaves like a first order system with a closed loop pole at about $s = -65$ (which would produce a settling time of about 0.06 seconds). Curiously, the closed-loop poles are at $-3 + j0.1$, $-3 - j0.1$, -64.6 , -137.4 . We will talk more about the effect of zeros today!

Appendix: Matlab Code for Simulation of Step, Ramp, and Parabola Responses:

```
numgzohg=200*10;
dengzohg=poly([0,-8,-200]);
numclgzohg=numgzohg;
denclgzohg=dengzohg+[0 0 0 numgzohg];
t=[0:.01:1];
step=ones(1,length(t));
ramp=t;
Ts=.01;
t=[0:100]*Ts;
yunstep=lsim(numclgzohg,denclgzohg,step,t);
yunramp=lsim(numclgzohg,denclgzohg,ramp,t);
numc=conv(numgzohg,213.08*[1 10.8]);
```

```

denc=conv(dengzohg,[1 171.85]);
numccl=numc;
denccl=denc+[0 0 0 numccl];
ycstep=lsim(numccl,denccl,step,t);
ycramp=lsim(numccl,denccl,ramp,t);
numlag=conv(numc,klag*[1 zlag]);
denlag=conv(denc,[1 plag]);
numcllag=numlag;
dencllag=denlag+[0 0 0 numlag];
ylagstep=lsim(numcllag,dencllag,step,t);
ylagramp=lsim(numcllag,dencllag,ramp,t);
plot(t,yunstep,t,ycstep,t,ylagstep,[0 1],'.98*[1 1],'-',[0 1],1.02*[1 1],'-')
grid;xlabel('time (sec)');ylabel('y,ylead,ylag')
title('EE572 - Uncomp, Lead and Lead/Lag Step Responses')
plot(t,yunramp,t,ycramp,t,ylagramp)
title('EE572 - Uncomp, Lead and Lead/Lag Ramp Responses')
grid;xlabel('time (sec)')
plot(t,yunramp,t,ycramp,t,ylagramp,t,t,'-')
title('EE572 - Uncomp, Lead and Lead/Lag Ramp Responses')
grid;xlabel('time (sec)')
t=[0:10000]*Ts;
ramp=t;
ylagramp=lsim(numcllag,dencllag,ramp,t);
ycramp=lsim(numccl,denccl,ramp,t);
plot(t,t-ycramp',t,t-ylagramp',[0 100],1/50*[1 1],'-')
title('EE572 - Lead and Lead/Lag Ramp Error Responses')
grid;xlabel('time (sec)')
numcpid=conv(numgzohg,[Kd Kp Ki]);
denpid=conv(dengzohg,[1 0]);
numclpid=numcpid;
denclpid=denpid+[0 0 0 numcpid];
parabola=0.5*t.^2;
ypidstep=lsim(numclpid,denclpid,step,t);
ypidramp=lsim(numclpid,denclpid,ramp,t);
ypidparabola=lsim(numclpid,denclpid,parabola,t);
yunparabola=lsim(numclgzohg,denclgzohg,parabola,t);
plot(t,yunstep,t,ypidstep,[0 1],'.98*[1 1],'-',[0 1],1.02*[1 1],'-')
title('EE572 - Uncomp and PID Step Responses')
grid;xlabel('time (sec)')

```