Loopshaping Example

Construct a controller for the plant G(s) so that the closed-loop satisfies the following design requirements:

1. Closed-loop is stable
2. Speed of Response: Loop gain crossover frequency `|L(j*wc)|` =1 at wc = 10 rad/sec
3. Good Reference Tracking: `|S(jw)|` < 0.01 for w<0.5 rad/sec
4. Good Noise Rejection: `|T(jw)|` < 0.01 for w>200 rad/sec
5. Good Robustness: Phase Margin of approximately +/- 45deg

Intepretation: Note that the sensitivity function S(s) is the closed-loop transfer function from reference r to tracking error e. Thus requirement R3 means that if the reference is a sinusoid r(t)=sin(wt) with frequency w<0.5 then the steady-state tracking error will be e(t)=|S(jw)| sin(wt+ angle(S(jw)). Thus the amplitude of the error will be small (<0.01) if requirement R3 is satisfied. Earlier in the course we specified tracking requirements in terms of the steady-state error due to step reference commands. It is important to realize that w=0rad/sec corresponds to constant inputs. In particular, if the reference is a unit step, r(t)=1, then the steady-state tracking error is ess=S(0). Thus requirement R3 implies a steady-state error <0.01 for unit step reference commands.

The noise rejection requirement (R4) has similar sinusoidal frequency response interpretations.

Contents

Plant Model

```
Gnum = [0.4 16];
Gden = [1 3 2];
G = tf(Gnum,Gden);
```

Step 0: Requirements on Loop

The first step of the loopshaping design process is to convert the design requirements into specifications on the loop transfer function, L=G*K.

1. Closed-loop is stable: For now, we'll simply check for closed-loop stability at the end of our design. We'll show later, using the Nyquist stability theorem, that our loop-shaping design process will yield closed-loop stability.
2. Speed of Response: This requirement is already specified in terms of the loop as `L(j*wc)` =1 at wc = 10 rad/sec.
3. Good Reference Tracking: The requirement on the sensitivity function is: `S(jw)` = | 1/(1+L(jw)) | < 0.01 for w<0.5 rad/sec Cross-multiply to obtain an equivalent condition on the loop: `1+L(jw)` > 100 This is approximately equivalent to: `L(jw)` > 100 for w<0.5 rad/sec

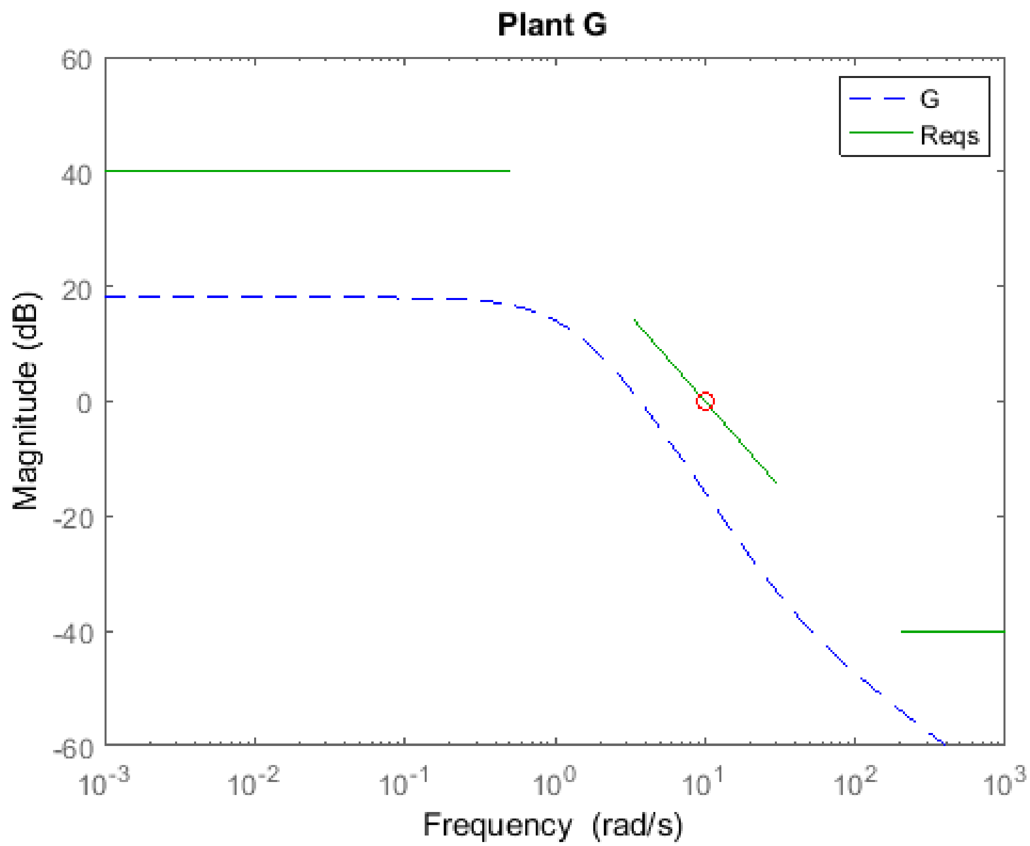
4. Good Noise Rejection: `T(jw)` < 0.01 for w>200 rad/sec The requirement on the complementary sensitivity function is: `T(jw) = L(jw) / (1+L(jw))` < 0.01 for w>200 rad/sec This is approximately equivalent to: `L(jw)` < 0.01 for w>200 rad/sec
5. Good Robustness: Phase Margin of approximately +/- 45deg Assume our loop only has zeros and poles in the left-half of the complex plane. Then the Bode Gain/Phase relation implies that the phase margin requirement is satisfied if the slope of `L(jw)` >= -30 dB/dec for frequencies near wc. Note that 1/s rolls-off at -20dB/dec and 1/s^2 rolls off at -40dB/dec. Thus a slope of -30dB/dec roughly corresponds to 1/s^(3/2).

Store Requirements on L: The syntax FRD(Data,Frequency) stores frequency response data. The syntax LINSPACE(X1, X2, N) generates N points linearly spaced between X1 and X2.

```
wc = 10;
R2 = frd(1,wc);

N=500;
wlow = linspace(1e-3,0.5);
R3 = frd( tf(100), wlow);

whigh = linspace(200,1e3,N);
R4 = frd( tf(0.01), whigh);

wmid = linspace(wc/3,wc*3,N);
R5 = frd( (wc./wmid).^(1.5),wmid);
```

Bode plot of plant G with requirements The plant itself satisfies the noise rejection requirement (R4). However, G=1 at w=3.7 rad/sec and hence G does not satisfy the gain crossover frequency requirement (R2). Moreover, G does not satisfy the tracking requirement (R3). The loopshaping procedure will be used to construct a controller so that L=G*K satisfies all requirements.

```
figure(1);
bodemag(G,'b--',R3,'g',R4,'g',R5,'g',R2,'ro');
title('Plant G');
legend('G','Reqs','Location','Best');
ylim([-60 60]);
```



Step 1: Design K1 to achieve desired crossover (R2)

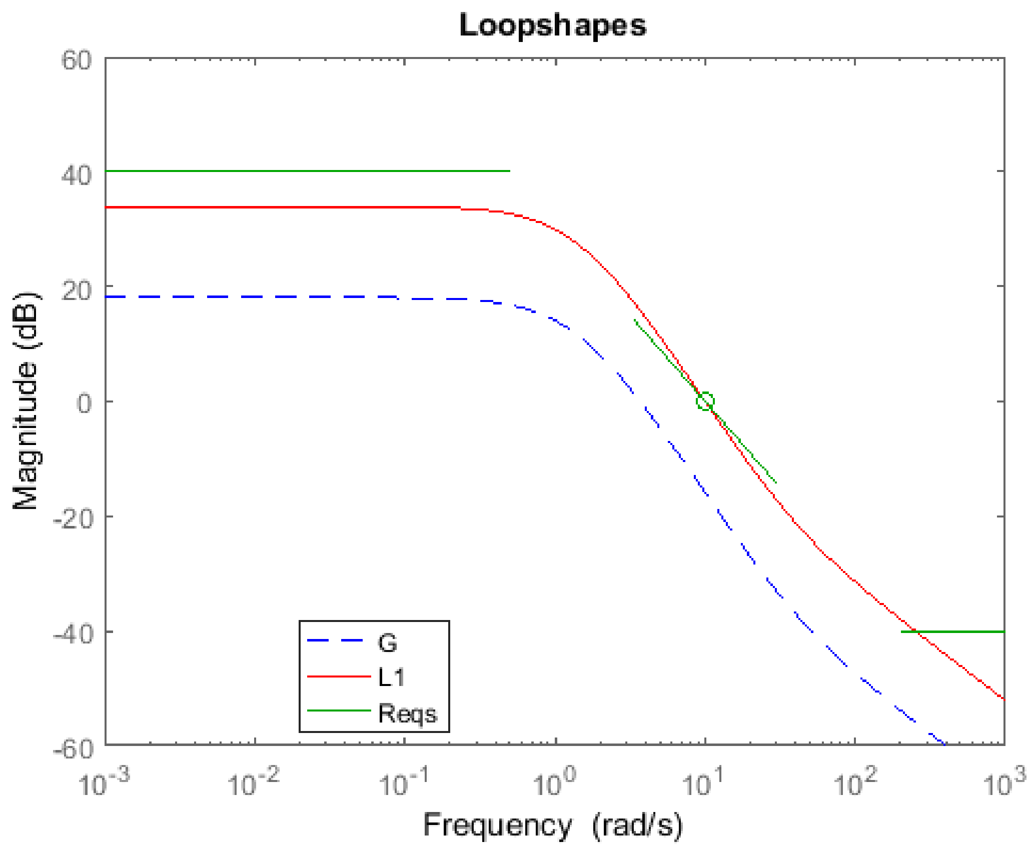
Our first step will be to design a proportional controller K1 so that the first loopshape L1=G*K1 has the desired gain crossover frequency: 1=|L(jwc)| = `G(jwc)*K1` This can be achieved by choosing the proportional gain `K1` = 1/`G(jwc)` The FREQRESP command can be used to evaluate G(jwc).

Construct loopshape, L1

```
K1 = 1/abs( freqresp(G,wc) );
L1 = G*K1;
```

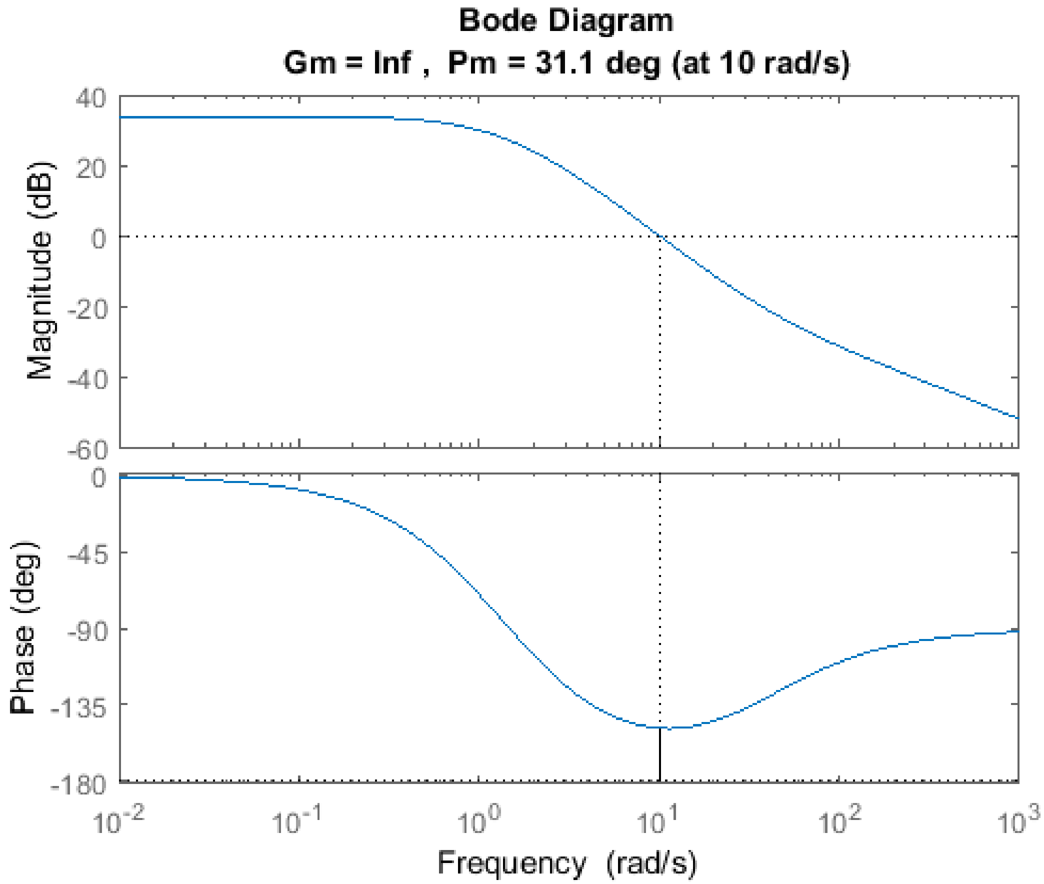
Bode plot of loopshape L1 with requirements L1 now satisfies the gain crossover frequency requirement (R2). However, it no longer satisfies the noise rejection requirement (R4) because we increased the gain. The loop also fails to satisfy the reference tracking (R3) and robustness requirements (R5).

```
figure(2);
bodemag(G,'b--',L1,'r',R3,'g',R4,'g',R5,'g',R2,'go');
title('Loopshapes');
legend('G','L1','Reqs','Location','Best');
ylim([-60 60]);
```



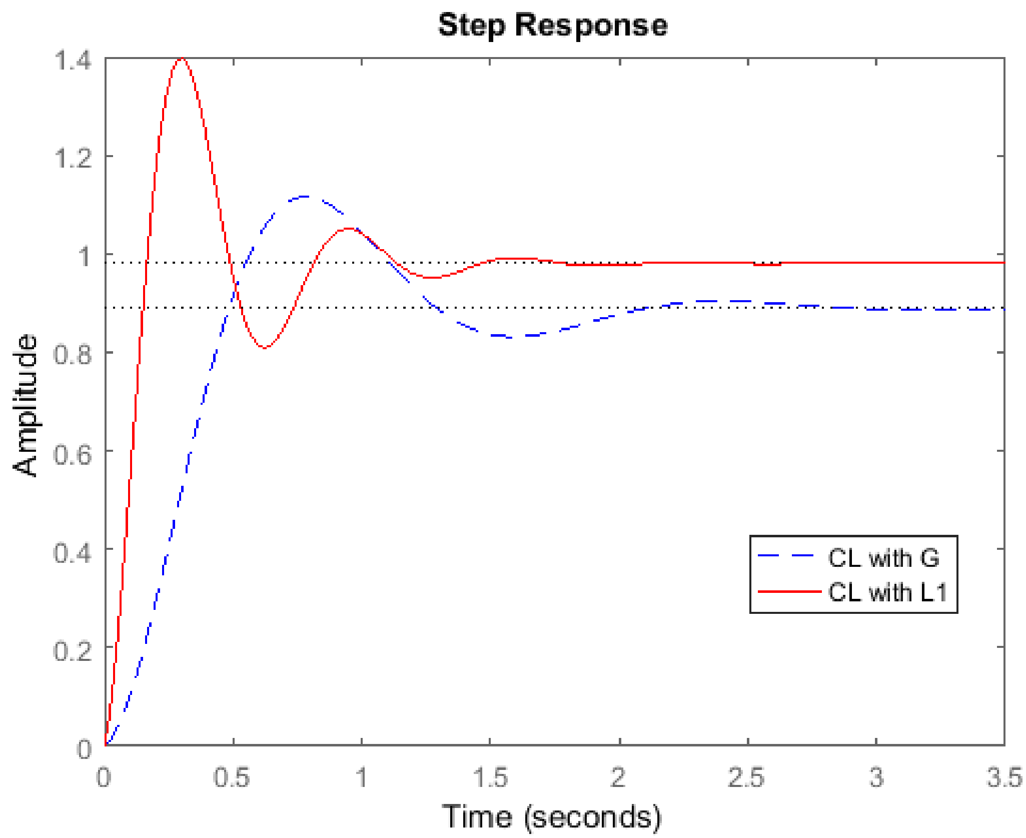
Phase Margin for L1 The MARGIN command can be used to compute the gain and phase margins for a system. This command shows that L1 has 31degs of phase margin at the cross-over frequency. This confirms that L1 does not satisfy the robustness requirement (R5).

```
figure(3);
margin(L1)
```



Unit Step Response The original plant G has a gain crossover at w=3.7rad/sec. The loopshape L1 has a gain crossover at w=10rads/sec. Thus we expect L1 to have a faster **closed-loop** response than G because it has a higher **open-loop** crossover frequency. The step responses show that the closed-loop with L1 has a faster rise time and settling time. However, the response with L1 has a larger overshoot. The large overshoot is related to the small phase margin achieved by L1.

```
figure(4);
CL0 = feedback(G,1);
CL1 = feedback(L1,1);
step(CL0,'b--',CL1,'r');
legend('CL with G','CL with L1','Location','Best');
```



Step 2: Design K2 to achieve desired reference tracking (R3)

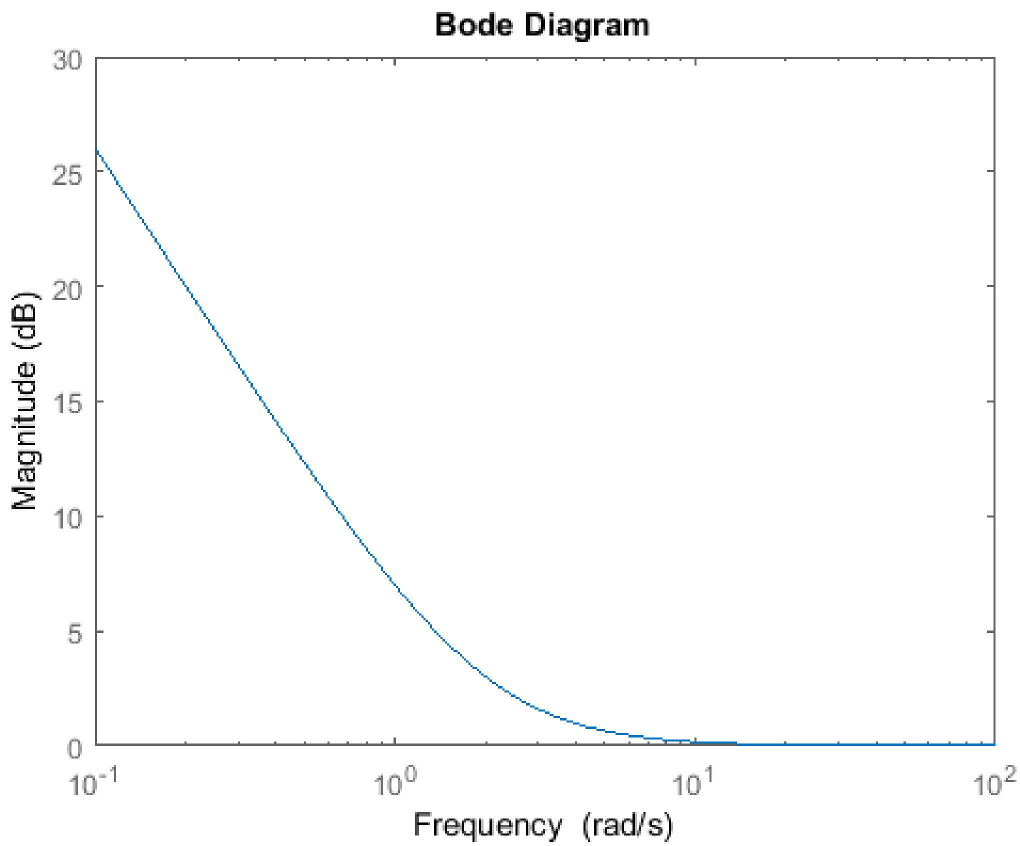
Our loopshape L1 has a low frequency gain of about 50 but we have a requirement that `L(jw)` >100 for w < 0.5 rad/sec. Thus we need to increase the loop gain by a factor of 2 at low frequencies. We can use an integral boost to increase the low-frequency gain. Kib(s) = (s+wbar)/s The integral boost Kib(s) has a high frequency gain of 1 and a corner frequency of wbar. Thus Kib(s) has very little effect at high frequencies but it will increase the gain at low frequencies. As a rough rule of thumb, choose the corner frequency wbar to be at least a factor of 4-5 below the gain crossover frequency wc. This will ensure that Kib has small effect on the cross-over and high frequency behavior. A little of trial and error will quickly lead to a value of wbar such that the next loopshape L2 meets the low frequency tracking requirement.

Construct loopshape, L2

```
wbar = 2;
Kib = tf([1 wbar],[1 0]);
K2 = K1*Kib;
L2 = G*K2;
```

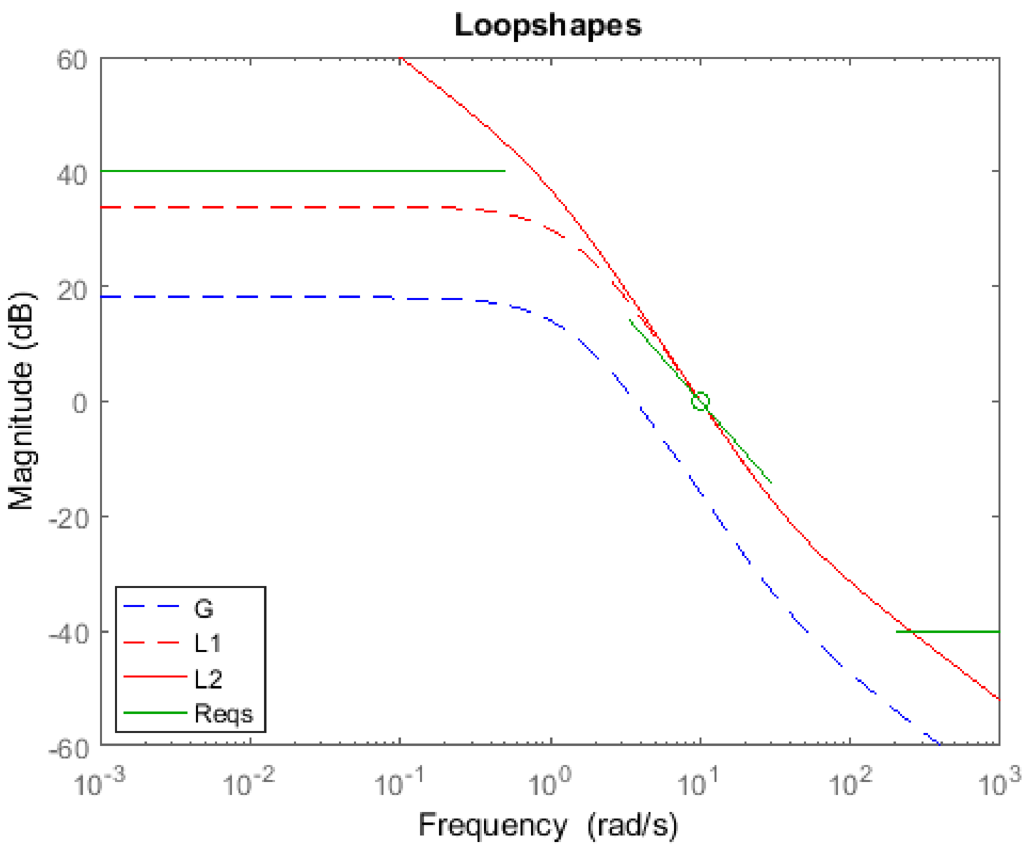
Bode plot of integral boost, Kib Note that Kib has large gain at low frequencies but gain of 1 (=0dB) at high frequencies.

```
figure(5);
bodemag(Kib);
```



Bode plot of loopshape L2 with requirements L2 now satisfies tracking requirement (R3) and crossover requirement (R2). It does not satisfy the noise rejection (R5) and robustness requirements (R5). Also, notice that the loopshape L2 is very similar to the loopshape L1 at high frequencies. The integral boost has increased the low frequency gain with small effect at higher frequencies.

```
figure(6);
bodemag(G,'b--',L1,'r--',L2,'r',R3,'g',R4,'g',R5,'g',R2,'go');
title('Loopshapes');
legend('G','L1','L2','Reqs','Location','Best');
ylim([-60 60]);
```



Step 3: Design K3 to achieve desired noise rejection (R4)

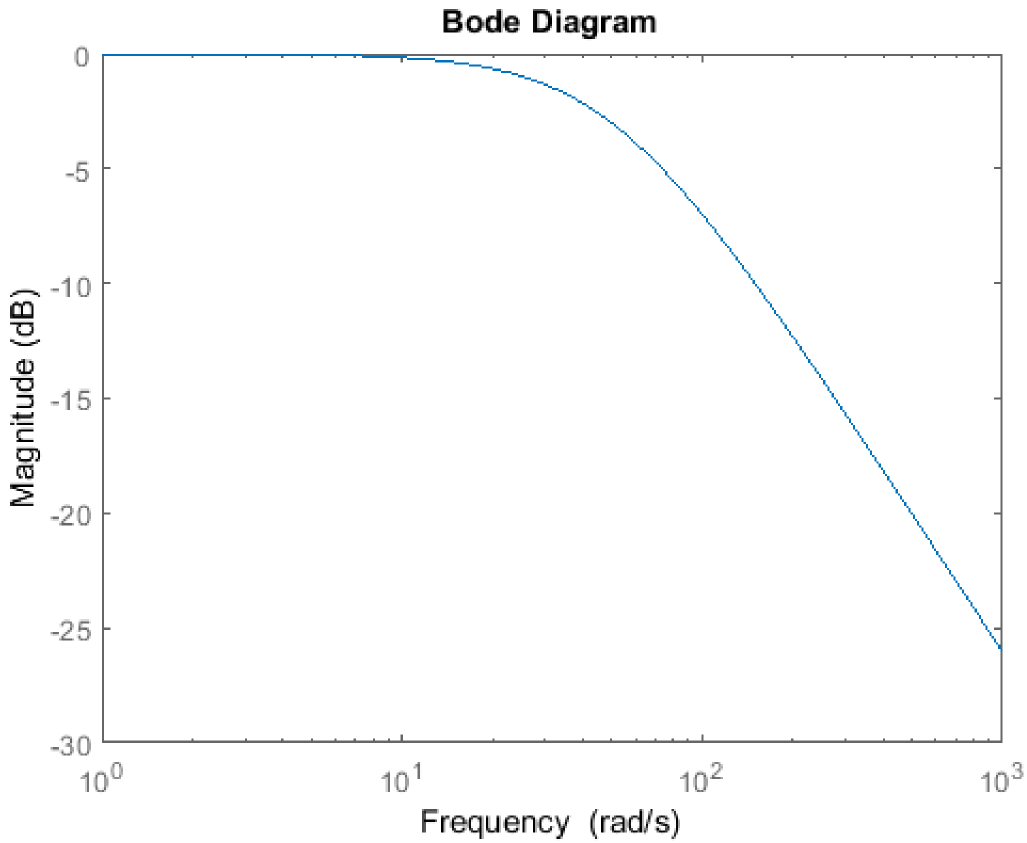
Our loopshape L2 does not satisfy the high frequency noise rejection requirement. Specifically, we need to reduce the high frequency loop gain so that it satisfies `L(jw)` < 0.01 for w>200 rad/sec We can use a roll-off to decrease the high-frequency gain. Kro(s) = wbar/(s+wbar) The roll-off Kro(s) has a low frequency gain of 1 and a corner frequency of wbar. Thus Kro(s) has very little effect at low frequencies but it will decrease the gain at high frequencies. As a rough rule of thumb, choose the corner frequency wbar to be at least a factor of 4-5 above the gain crossover frequency wc. This will ensure that Kro has small effect on the cross-over and low frequency behavior. A little of trial and error will quickly lead to a value of wbar such that the next loopshape L3 meets the high frequency noise rejection requirement.

Construct loopshape, L3

```
wbar = 50;
Kro = tf(wbar,[1 wbar]);
K3 = K2*Kro;
L3 = G*K3;
```

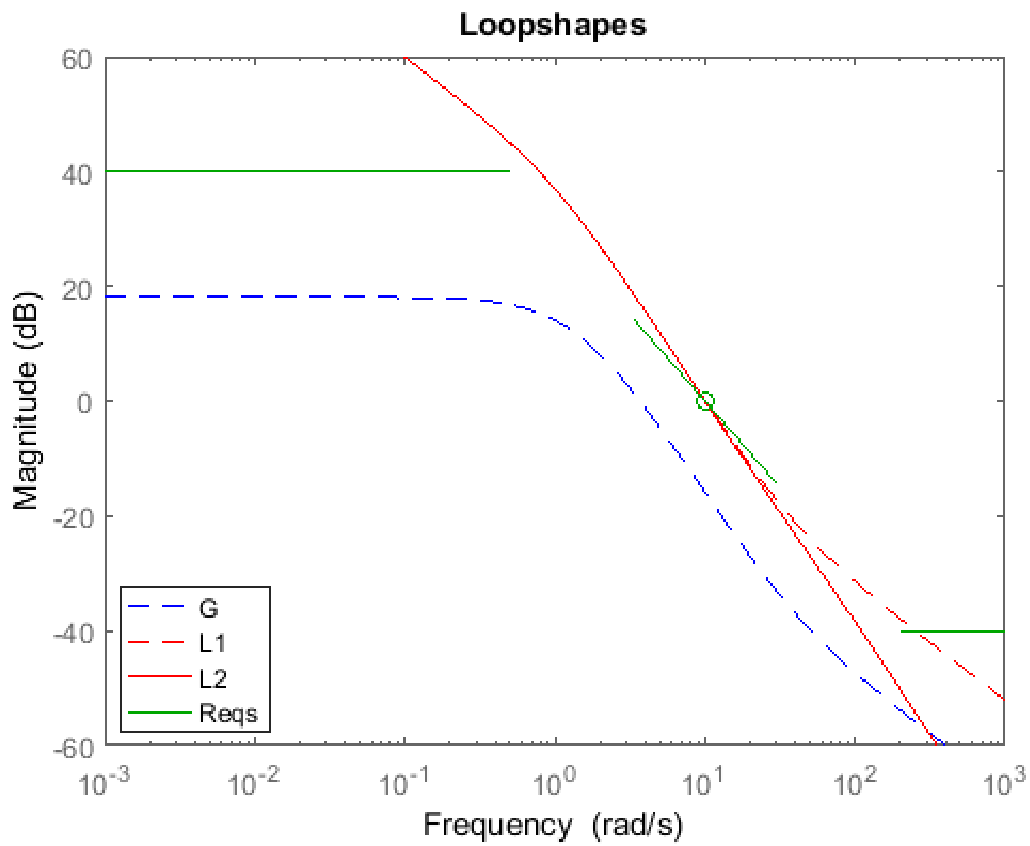
Bode plot of roll-off, Kro Note that Kro has small gain at high frequencies but gain of 1 (=0dB) at low frequencies.

```
figure(7);
bodemag(Kro);
```



Bode plot of loopshape L3 with requirements L3 now satisfies all requirements except the robustness requirement (R5). Also, notice that the loopshape L3 is very similar to the loopshape L2 at low frequencies. The roll-off decreases the high frequency gain with small effect at lower frequencies.

```
figure(8);
bodemag(G,'b--',L2,'r--',L3,'r',R3,'g',R4,'g',R5,'g',R2,'go');
title('Loopshapes');
legend('G','L1','L2','Reqs','Location','Best');
ylim([-60 60]);
```

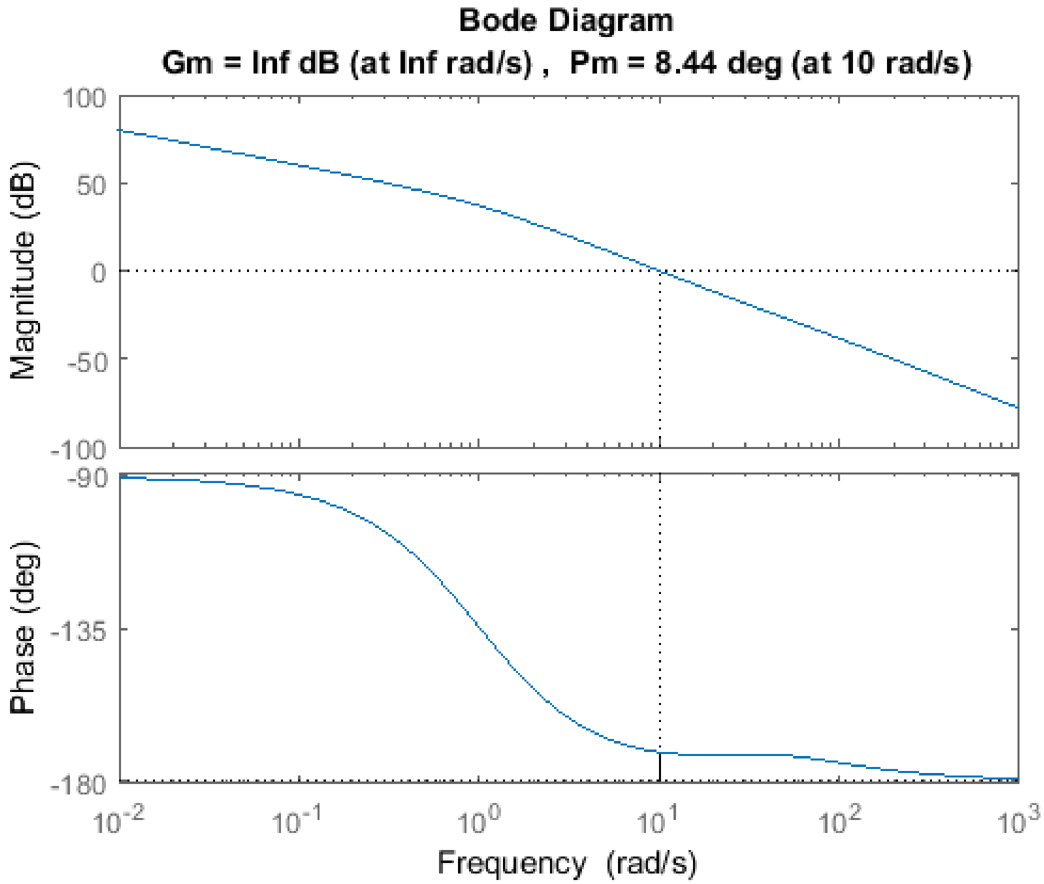


Step 4: Design K4 to achieve desired robustness (R5)

Our loopshape L3 does not satisfy the robustness requirement (R5) of +/-45deg of phase margin. On the magnitude plot, the loopshape L3 has a slope that is too steep at the gain cross-over frequency. We can use a lead controller to make the the slope more shallow which equivalently increase the phase at crossover. The lead controller has the form: Kl(s) = (beta*s+wbar)/(s+beta*wbar) We get to choose beta >1 and wbar. The lead Kl(s) has a high frequency gain of beta, a low frequency gain of 1/beta, and `Kl(jwbar)` =1. We'll choose wbar = wc so that Kl does not affect the loopgain at the desired cross-over wc. Larger values of beta has the benefit that it will give larger values of phase. However, larger values of beta have the drawback that Kl will decrease the low frequency gain and increase the high frequency gain. After a bit of trial and error, the robustness reuqirement is satisfied for a choice of beta=2.

Phase Margin for L3 Use the MARGIN command again to compute the gain and phase margins for L3. This command shows that L3 only has 8degs of phase margin at the cross-over frequency. This confirms that L1 does not satisfy the robustness requirement (R5).

```
figure(9);
margin(L3)
```

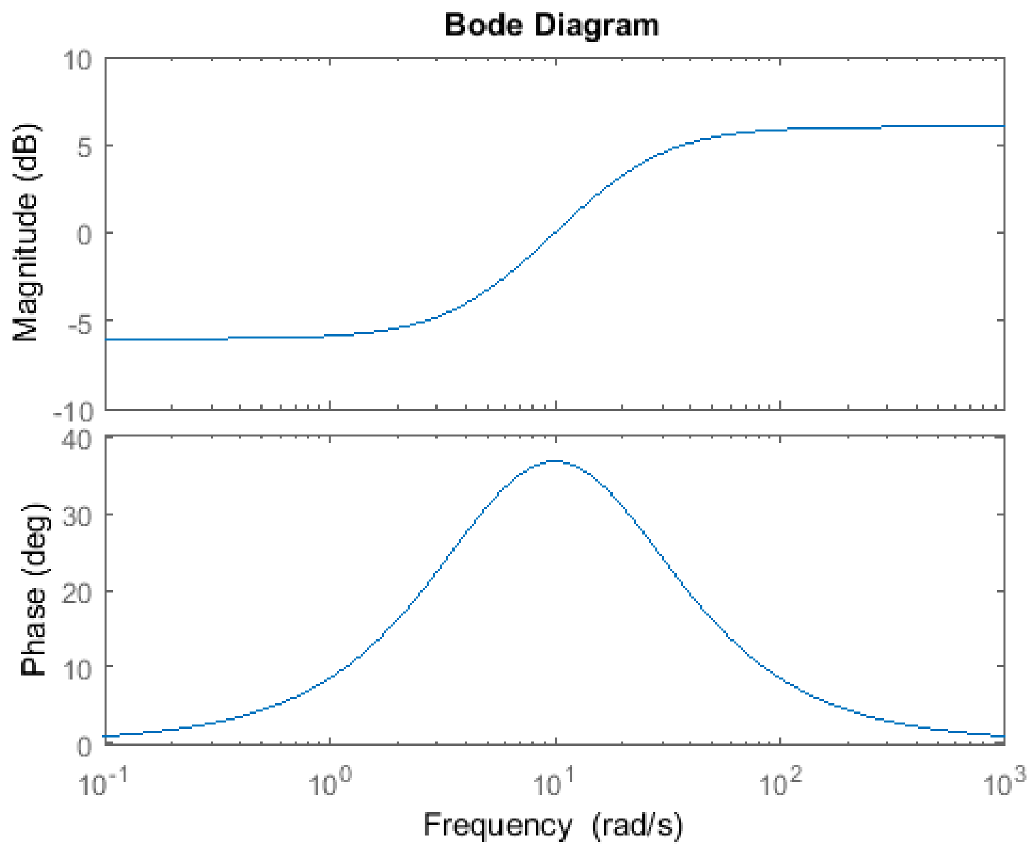


Construct loopshape, L4

```
beta = 2;
Kl = tf([beta wc],[1 beta*wc]);
K4 = K3*Kl;
L4 = G*K4;
```

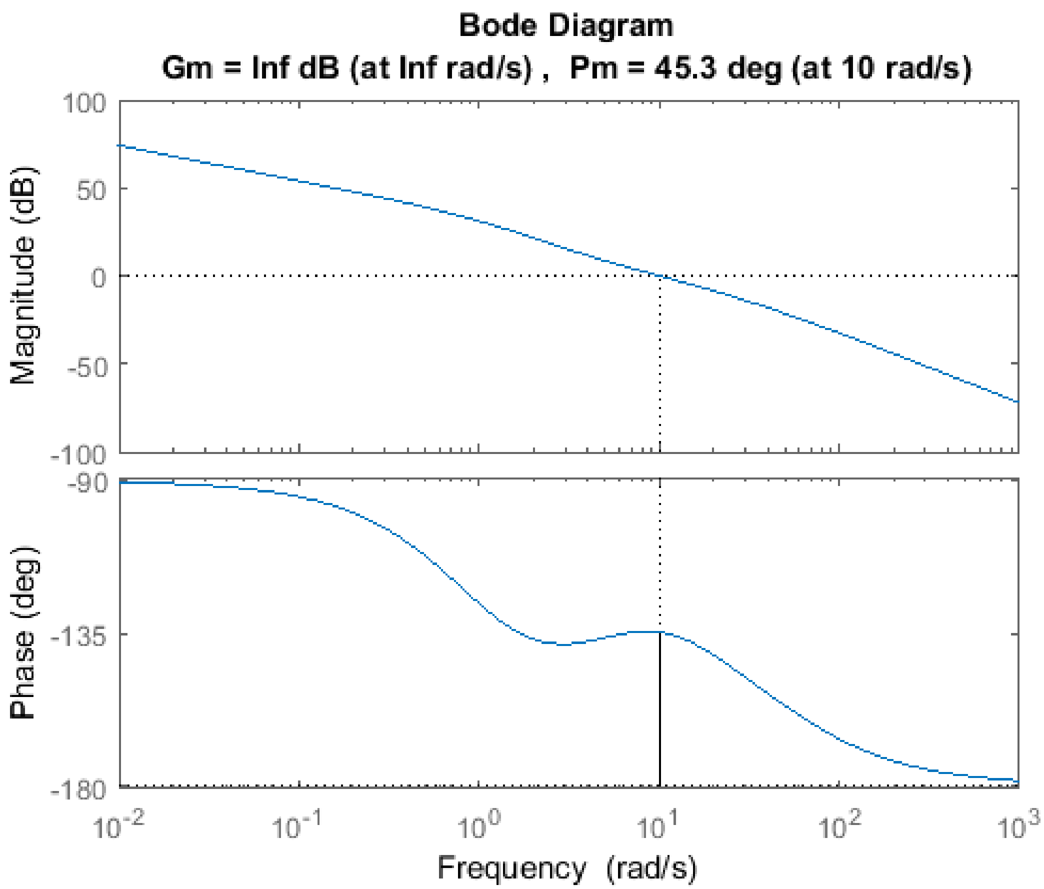
Bode plot of lead control, Kl Note that Kl has a gain of 1 (=0dB) and a positive slope at wc. The lead has a gain 1/beta<1 at low frequencies and a gain of beta>1 at high frequencies. The lead has a postive phase of 37 deg at wc.

```
figure(10);
bode(Kl);
```



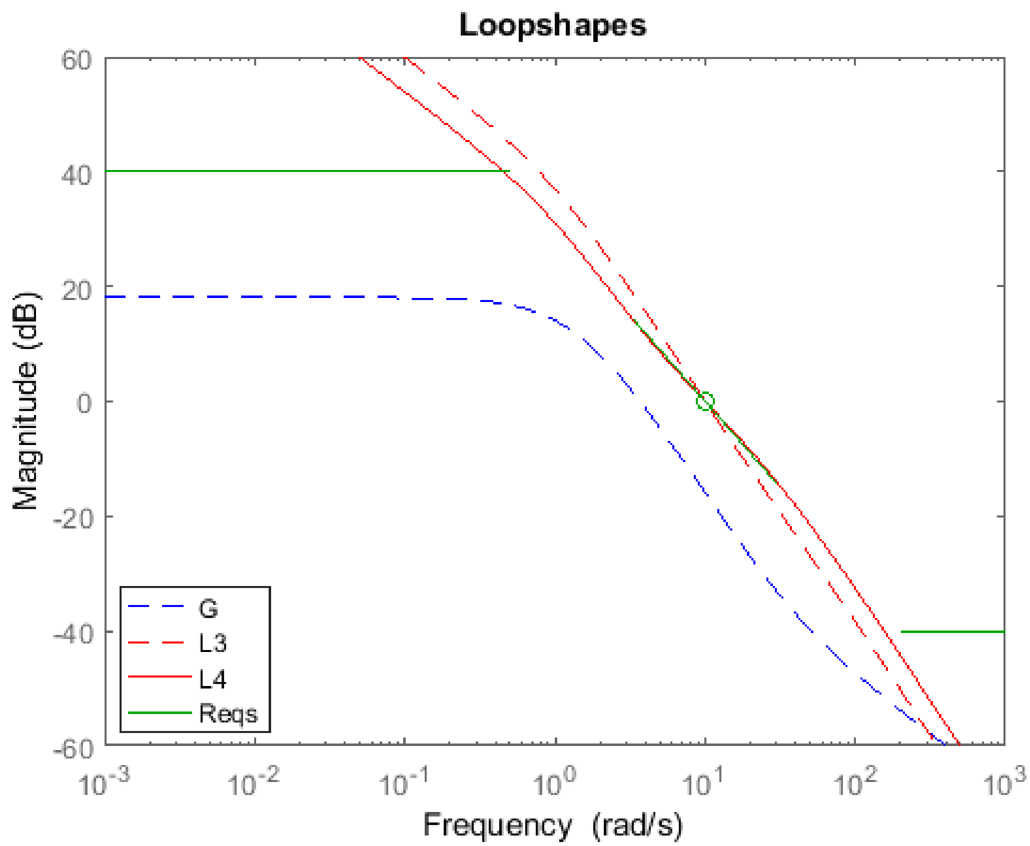
Phase Margin for L4 The new loopshape L4 now has 45deg of phase margin and hence the loop satisfies the robustness requirement (R5). Notice the 45deg of phase margin for L4 is equal to the 8deg phase margin of L3 plus the additional 37degs of phase contributed by the lead controller Kl.

```
figure(11);
margin(L4)
```



Bode plot of loopshape L4 with requirements L4 now satisfies the robustness margin requirement. However, the lead controller has decreased the low frequency gain. As a result L4 slightly violates the low frequency tracking requirement (R3). We can go back and iterate on our various choices to ensure that all requirements are satisfied. However, we'll go ahead with this design since it almost satisfies all requirements.

```
figure(12);
bodemag(G,'b--',L3,'r--',L4,'r',R3,'g',R4,'g',R5,'g',R2,'go');
title('Loopshapes');
legend('G','L3','L4','Reqs','Location','Best');
ylim([-60 60]);
```



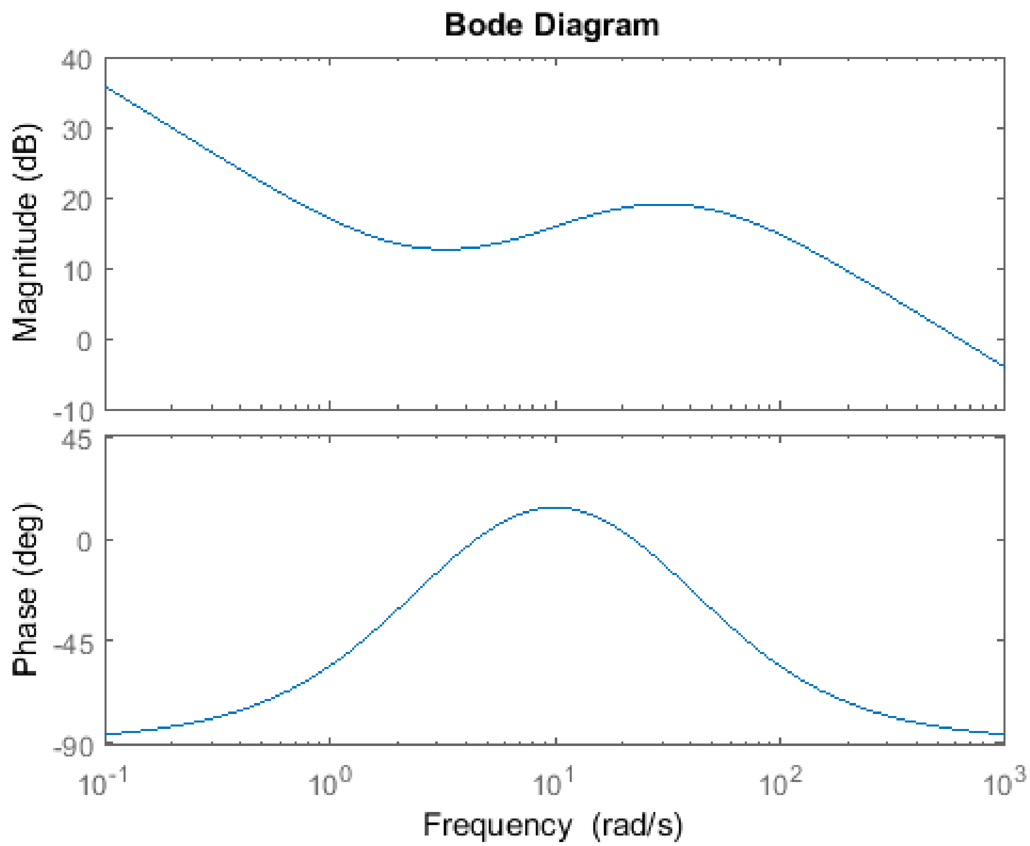
Performance Assessment

We can assess the performance of our final control design K4 in both the time and frequency domains.

Final Controller The final controller K4 is third order (3 poles) with two zeros. We constructed this controller in four stages: 1) Proportional, 2) Integral Boost, 3) Roll-off, 4) Lead Controller. The Bode plot of K4 shows the integral behavior at low frequencies, the lead behavior (positive slope) at mid frequencies and roll-off at high frequencies.

```
figure(13)
bode(K4)
K4
```

```
K4 =

   621.4 s^2 + 4350 s + 6214
  -------------------------
     s^3 + 70 s^2 + 1000 s

Continuous-time transfer function.
```



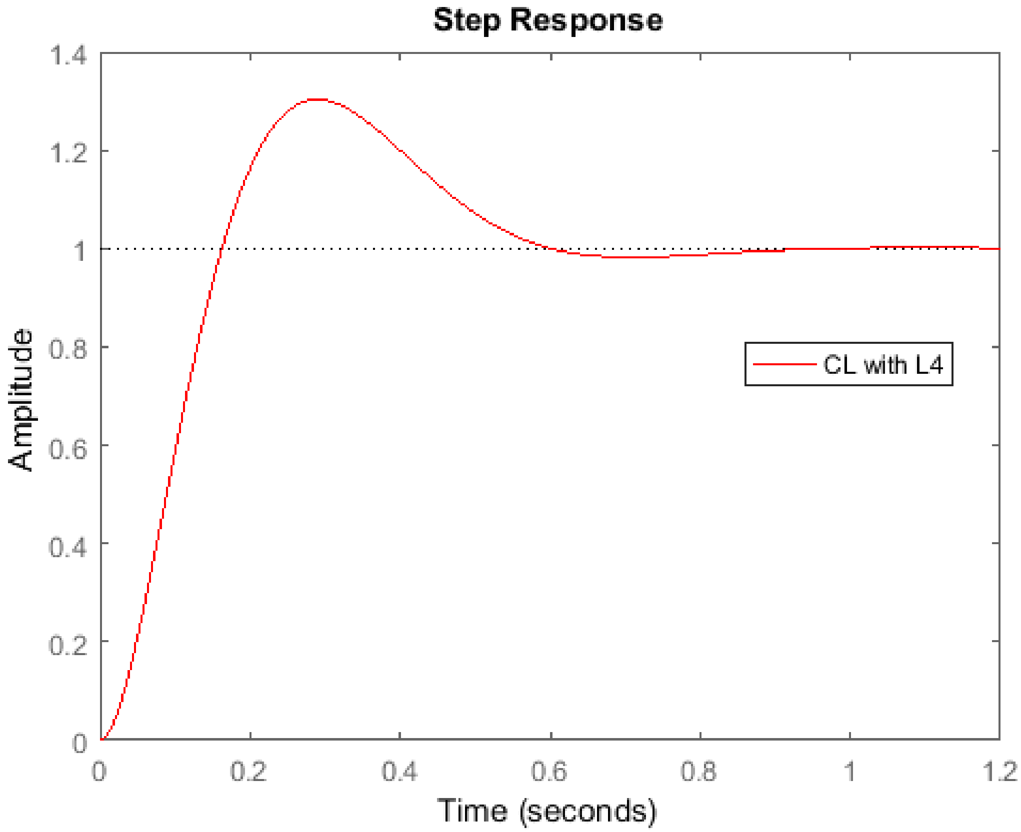
Closed-loop stability We can verify that the final controller K4 results in closed-loop stability. Formally we need to check the stability of all closed-loop transfer functions. For simplicity, we'll check the poles of the complementary sensitivity function. This closed-loop transfer function has all poles in the left half of the complex plane and hence it is stable.

```
CL4 = feedback(L4,1);
pole(CL4)
```

```
ans =
 -48.5920 + 0.0000i
  -6.4162 + 8.1041i
  -6.4162 - 8.1041i
  -9.5757 + 0.0000i
  -2.0000 + 0.0000i
```

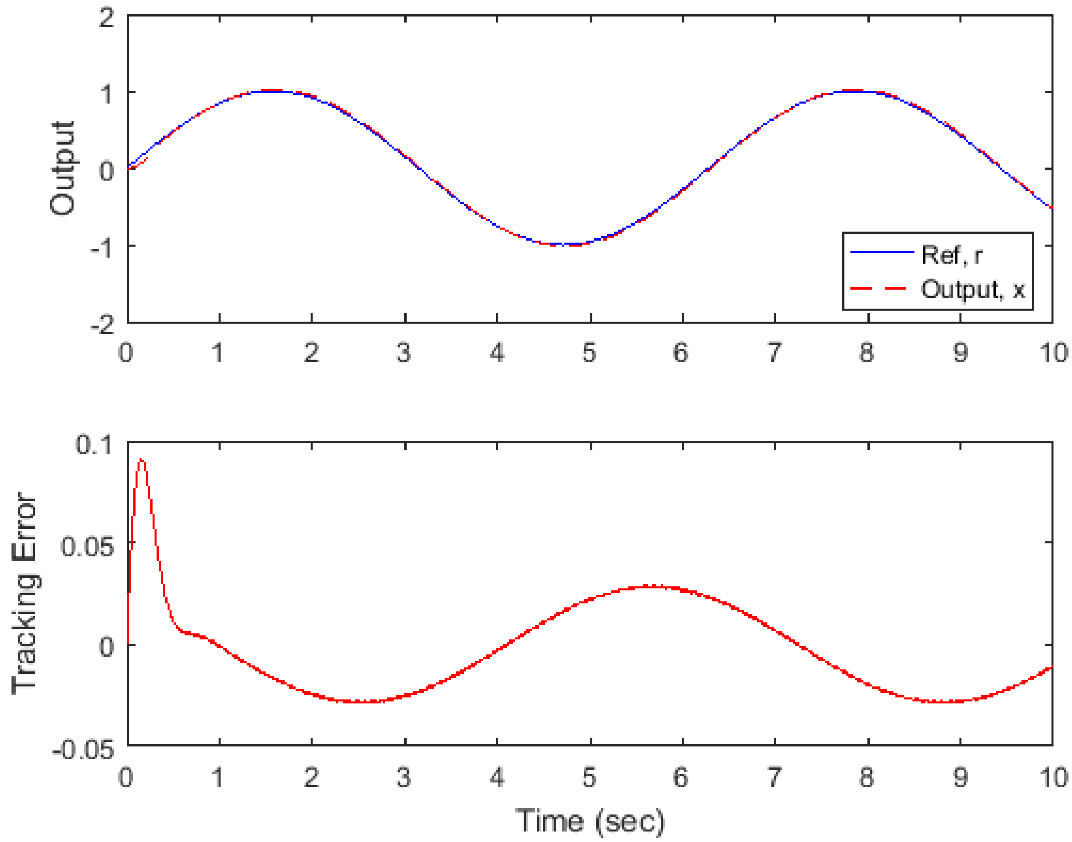
Unit Step Response The figure below shows the closed-loop response x(t) for a unit step reference command. The response converges to a steady-state value xss(t)=1, i.e. the closed-loop has zero steady-state tracking error. This zero steady-state error is a consequence of the the integral control in K4. The 5% settling time is about 0.5sec which is roughly equal to 5/wc.

```
figure(14);
step(CL4,'r');
legend('CL with L4','Location','Best');
```



Simulations We can also run a simulation with a sinusoidal reference command and noise input: r(t) = Ar*sin(wr*t) and n(t)=An*sin(wnt). The simulation is performed using Simulink. Note that the actual tracking error is r-x while the measured tracking error in the model is e=r-(x+n).

```
Ar = 1;
wr = 1;
An = 0.1;
wn = 200;

Knum = K4.Num{1};
Kden = K4.Den{1};
Tf = 10;
sim('LoopshapingSim',Tf)

figure(15)
subplot(211);
plot(time,r,'b',time,x,'r--')
ylabel('Output');
legend('Ref, r','Output, x','Location','Best');

subplot(212);
plot(time,r-x,'r')
ylabel('Tracking Error');
xlabel('Time (sec)');
```



Linear Analysis The simulation results can be verified using frequency response analysis. In the Laplace domain the output is related to the reference command and noise by: X(s) = T(s)*R(s) - T(s)*N(s) where T is the complementary sensitivity function. For linear systems, these two effects add linearly and we can analyze them separately.

First, consider the effect of the noise on the output: X(s) = -T(s)*N(s) For the sinusoidal noise in the model x(t) = -An*|T(jwn)|*sin(wn*t + angle(T(jwn))) We designed the loop so that T is small at high frequencies. As a result, the steady state output due to the noise is approximately x(t) = -An*|L(jwn)|*sin(wn*t + angle(T(jwn))) `L(jwn)` is very small so the noise will have a very small effect on the output.

```
abs( freqresp(L4,wn) )
```

```
ans =
    0.0061
```

Next, consider the effect of the reference on the tracking error: E(s) = S(s)*R(s) For the sinuoidal reference command, the steady-state error is e(t) = Ar*|S(jwr)|*sin(wr*t + angle(S(jwr))) We designed the loop so that S is small at low frequencies. As a result, the steady state error is approximately e(t) = (Ar/|L(jwr)|)*sin(wr*t + angle(S(jwr))) 1/|L(jwr)| is very small at low frequencies. You can verify that the tracking error in the simulation plot is roughly Ar/|L(jwr)|.

```
ess = Ar*1/abs( freqresp(L4,wr) )

figure(15)
subplot(212);
hold on;
plot([time(1) time(end)],[ess ess],'g--');
```

```
plot([time(1) time(end)],[-ess -ess],'g--');
hold off;
```

```
ess =
    0.0279
```