Chapter 3 HW Solution

Problem 1. Here you're given a *lead network*—often used in control systems to improve the transient response—which adds **around** 60° of phase angle at about $\omega_1 = 3$ rad/s. The lead network transfer function is

$$H(s) = \frac{s+1}{0.1s+1} \tag{1}$$

(a) The phase angle of the continuous system at 3 rad/sec is $\phi = 54.87°$. You can get the $H(z)$ and phase angles using MATLAB `c2d` and `bode` (except for the rectangular integration rules) or you can get the $H(z)$ "manually" and evaluate its angle with $z = e^{j\omega_1 T}$ with $T = 0.25$ ($z = 0.73 + j0.68$). My results were

(i) Forward rectangular (F):

$$s = \frac{z-1}{T} \implies H_F(z) = \frac{10(z-0.75)}{z+1.5} \text{ UNSTABLE!!} \tag{2}$$

(ii) Backward rectangular (B):

$$s = \frac{z-1}{Tz} \implies H_B(z) = \frac{3.5714(z-0.8)}{z-0.2857} \implies \phi_B = 38.92° \text{ (QUITE LOW)} \tag{3}$$

(iii) Trapezoidal (T):

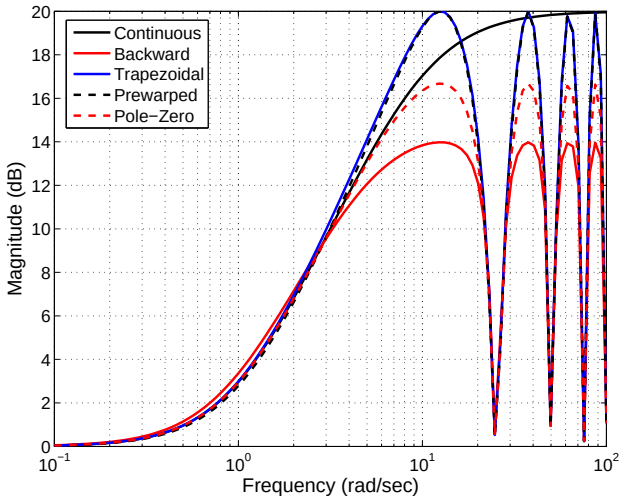
$$s = \frac{2}{T}\frac{z-1}{z+1} = 8\frac{z-1}{z+1} \implies H_T(z) = \frac{5(z-0.7777)}{z+0.1111} \implies \phi_T = 54.90° \text{ (VERY CLOSE!)} \tag{4}$$

(iv) Prewarped trapezoidal (P): here we have $A = \omega_1 / \tan(\omega_1 T/2) = 7.6214...$

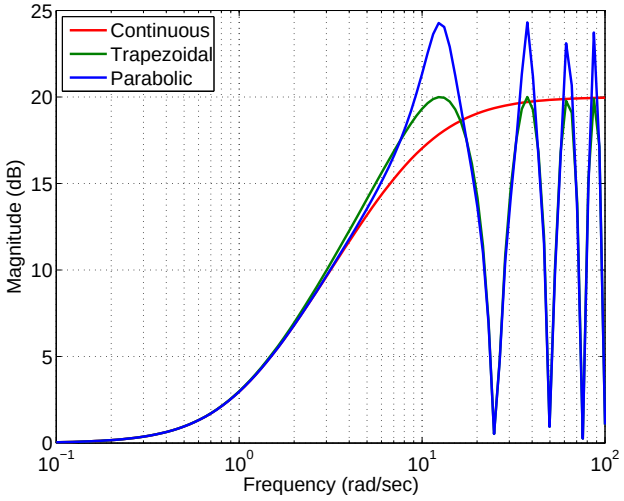
$$s = A\frac{z-1}{z+1} = 7.6214\frac{z-1}{z+1} \implies H_P(z) = \frac{4.8926(z-0.7680)}{z+0.1350} \implies \phi_P = 54.87° \text{ (EXACT!!)} \tag{5}$$

(v) Pole-Zero mapping (PZ):

$$\text{Map poles and zeros using } z = e^{sT} \implies H_{PZ}(z) = \frac{4.1496(z-0.7788)}{z-0.0821} \implies \phi_{PZ} = 47.57° \text{ (LOW)} \tag{6}$$



(a) Continuous and discrete versions.



(b) Continuous, trapezoidal, and parabolic.

Figure 1: Bode plots of the various lead networks.

Referring to Figure 1(a), the two best methods appear to be Trapezoidal and Prewarped, with Pole-Zero Mapping not too bad. It is no coincidence that MATLAB `c2d` supports these three methods, but *not* Forward

or BACKWARD rectangular. FORWARD in particular is dangerous, since it can result in an unstable discrete transfer function (it did in this problem).

(b) Bode amplitude plots of the original continuous system and all the simulations (except F) are also shown in Figure 1. Note the degeneration of all discrete systems at higher frequencies.

(c) Here I speculated on a "parabolic rule" based on our Chapter 2 HW problem on parabolic integration. Following the derivation of the trapezoidal rule in Section 3.2.3, we get a *parabolic* substitution rule as:

$$s \approx \frac{12}{T}\frac{z(z-1)}{5z^2+8z-1} \tag{7}$$

As expected, this rule is somewhat more complex than the other integration-based substitution rules. Also, I would not attempt solving (7) for $z = f(s)$.

Use of the parabolic rule of (7) in our lead network produces the discrete system

$$H_{Para}(z) = \frac{5.4082(z-0.7789)(z+0.0242)}{(z-0.1955)(z+0.522)} \implies \phi_{Para} = 55.67^\circ \text{ (CLOSE)} \tag{8}$$

For all the work involved in the parabolic rule, it's not quite as accurate (at least at $\omega = 3$) as the trapezoidal (unless I made a mistake—quite likely!) As another check, I compared the Bode magnitude plots of the continuous, trapezoidal, and parabolic in Figure 1(b). It appears that the parabolic rule "sticks with" the continuous a little longer than the trapezoidal rule, but...considering the additional hassle, the parabolic rule is probably not worth it!

Problem 2. The transfer function of a second-order system with $\zeta = 0.2$, $\omega_n = 50$ rad/s (≈ 8 Hz), and unity DC gain is:

$$G_C(s) = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2} = \frac{2500}{s^2+20s+2500} \tag{9}$$

(a) We can do the TRAPEZOIDAL and PREWARPED discrete simulations using MATLAB, since it supports these, but the MATLAB "MATCHED" (similar to POLE-ZERO mapping) is not quite the same as I present in my notes, so we have to do that one "manually." Using $T = 0.01$ (100 Hz sampling), my results are:

(i) TRAPEZOIDAL (T):

$$G_T(z) = \frac{0.0538z^2+0.1075z+0.0538}{z^2-1.6129z+0.8280} = \frac{0.0538(z^2+2z+1)}{z^2-1.6129z+0.8280} \text{ (poles } z = 0.8065 \pm j0.4214) \tag{10}$$

(ii) PREWARPED (P) at frequency ω_n:

$$G_P(z) = \frac{0.0559z^2+0.1117z+0.0559}{z^2-1.6016z+0.8250} = \frac{0.0559(z^2+2z+1)}{z^2-1.6016z+0.8250} \text{ (poles } z = 0.8008 \pm j0.4286) \tag{11}$$

(iii) POLE-ZERO mapping (PZ):

$$G_{PZ}(z) = \frac{0.0555z^2+0.1110z+0.0555}{z^2-1.5968z+0.8187} = \frac{0.0555(z^2+2z+1)}{z^2-1.5968z+0.8187} \text{ (poles } z = 0.7984 \pm j0.4257) \tag{12}$$

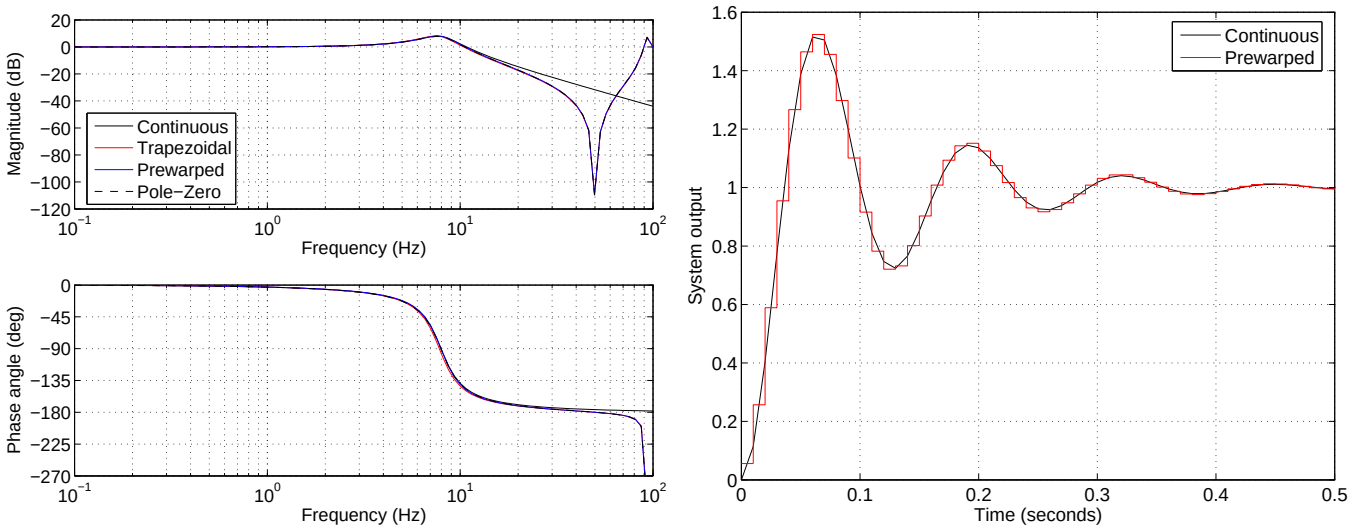
The three simulations are quite similar, with only minor deviations in numerator gain and pole locations. The zero locations of all three are exactly the same!

(b) The Bode amplitude plots of the continuous and all three discrete simulations from 0.1 to 100 Hz are shown in Figure 2(a). They all work pretty well (relatively fast sampling), but again the discrete systems go crazy at higher frequencies. The resonant peak of the continuous system is

$$|G_C|_{peak} = \frac{1}{\sqrt{2\zeta^2}} = 3.5355 \implies 10.97 \text{ dB} \tag{13}$$

All three discrete simulations show a resonant peak which is virtually identical to the continuous system.

(c) Unit step responses of the continuous and prewarped are shown in Figure 2(b). Pretty good agreement.



(a) Bode plots.

(b) Step responses.

Figure 2: Bode plots and step responses of Problem 2.

Problem 3. A *notch filter* is used to remove a given frequency from a signal. Since power-line noise is a common component, consider a notch filter that removes 60 Hz from the input. The transfer function of a second-order notch filter at frequency ω_o is

$$H_C(s) = \frac{Y(s)}{U(s)} = \frac{s^2 + \omega_o^2}{s^2 + 2\zeta\omega_o s + \omega_o^2} \tag{14}$$

where the damping ratio is $\zeta = \sqrt{2}/2 = 0.707$.

(a) Frequency $f_o = 60$ Hz corresponds to $\omega_o = 2\pi f_o = 376.9911$ rad/s. The notch filter at this frequency is simply equation (14) evaluated at this ω_o and the given ζ. This $H_C(s)$ should be entered into MATLAB.

(b) Sampling frequency $f_s = 500$ Hz corresponds to $T = 0.002$ seconds, or 2 msec. The prewarped substitution is

$$s = A\frac{z-1}{z+1}, \quad A = \frac{\omega_o}{\tan\frac{\omega_o T}{2}} = 952.1709 \quad \left(\frac{2}{T} = 1000 \text{ for comparison}\right) \tag{15}$$

When this substitution is used (or use MATLAB `c2d` using `'prewarp'`), the resulting $H_P(z)$ is given by

$$H_P(z) = \frac{0.6738z^2 - 0.9824z + 0.6738}{z^2 - 0.9824z + 0.3477} = \frac{0.6738(z^2 - 1.4579 + 1)}{z^2 - 0.9824z + 0.3477} \tag{16}$$

It is interesting to examine the zero and pole locations of $H_P(z)$, which are

$$\text{Zeros: } z = 0.7290 \pm j0.6845, \quad \text{Poles: } z = 0.4912 \pm j0.3262 \tag{17}$$

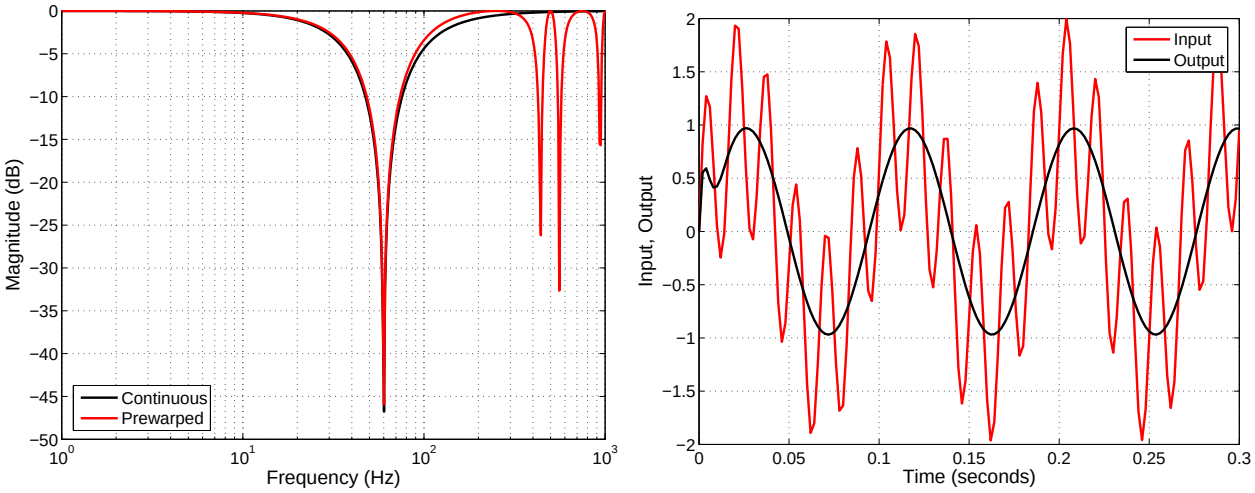
The zeros are right on the unit circle, corresponding to the the $j\omega$ axis in the s-plane. This is the **notch**! The angle of these zeros is $\theta = 0.7540$ rad, and so

$$\theta = 0.7540 = \omega T \implies \omega = \frac{\theta}{T} = 376.9911 \tag{18}$$

which corresponds *exactly* to 60 Hz.

(c) Bode magnitude plots of both $H_C(s)$ and $H_P(z)$ over $1 \le f \le 1000$ Hz are shown in Figure 3(a). The prewarped $H_P(z)$ displays almost exactly the same notch as the original continuous system, with inaccuracy at higher frequencies. It would perform very well at filtering out the 60 Hz frequency component.

(d) To test the notch filter, create an input consisting of 11 Hz sinusoid plus a 60 hz sinusoid, both of unity magnitude. I assumed that three periods of the 11 Hz sinusoid would be long enough to reach steady-state, and that's about 0.3 seconds. Thus I created a time vector of 0.3 seconds in length with a time step of T (0.002). The complete MATLAB code is shown below (including the discretization of the notch filter):



(a) Bode plots showing 60 Hz notch.

(b) Passing 11 Hz, filtering out 60 Hz.

Figure 3: Behavior of notch filter.

```
>> fo = 60;  % Notch frequency in Hz
>> wo = 2*pi*fo;  % Convert notch frequency to rad/sec
>> zeta = sqrt(2)/2;  % Damping ratio
>> numc = [1 0 wo^2];  % Continuous numerator s^2+wo^2
>> denc = [1 2*zeta*wo wo^2];  % Continuous denominator s^2+2*zeta*wn*s+wn^2
>> Hc = tf(numc,denc);  % Form continuous LTI notch filter
>> T = 0.002;  % Define sampling period
>> Hd = c2d(Hc,T,'prewarp',wo);  % Get prewarped simulation of notch filter
>> tmax = 0.3;  % Maximum time of simulation
>> t = 0:T:tmax;  % Time vector from 0 -> tmax at spacing T
>> f1 = 11;  % Frequency (Hz) of first input component
>> f2 = 60;  % Frequency (Hz) of second input component
>> u = sin(2*pi*f1*t)+sin(2*pi*f2*t);  % Construct the input (frequencies in rad/s)
>> y = lsim(Hd,u,t);  % Get the output using the MATLAB lsim function
>> plot(t,u,'r-',t,y,'k-');  % Plot input (red) and output (black)
>> xlabel('Time (seconds)');  % Label X axis
>> ylabel('Input, Output');  % Label Y axis
```

(e) It so happens that the notch filter transfer function $H_P(z)$ is in *exactly* the same form as the example in the notes in equation (3.33). So the block diagram for optimal realization of the notch filter will be exactly like Figure 3.7, and the "pseudo-code" to generate this filter is exactly that on page 48. I'll reproduce both of those here for the sake of completeness.

The notch filter transfer function is

$$H_P(z) = \frac{0.6738z^2 - 0.9824z + 0.6738}{z^2 - 0.9824z + 0.3477} = \frac{0.6738 - 0.9824z^{-1} + 0.6738z^{-2}}{1 - 0.9824z^{-1} + 0.3477z^{-2}} = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 - a_1z^{-1} - a_2z^{-2}} \tag{19}$$

and the (observer canonical) block diagram is (next page):

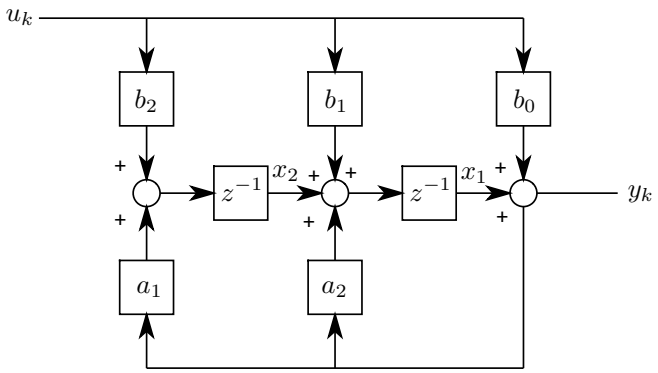


Figure 4: Observer canonical block diagram of notch filter.

```
10  read difference equation parameters and initialize variables;
20  set FLAG = 0;
30  select sample period, start clock;
40  if (FLAG == 1) print ("sampling period too short") and exit;
50  while (FLAG == 0) wait;
60  u = adc_in();
70  y = x1 + b0*u;
80  dac_out(y);
90  x1 = x2 + b1*u + a1*y;
100 x2 = b2*u + a2*y;
130 goto 40;
```

The code above from line 60 to line 130 contains the following operations:

- A/D conversion — varies depending on resolution, but around 10 μsec.
- D/A conversion — also dependent on resolution, but much faster, say 2 μsec.
- 5 floating point multiplies — on my "ancient" Apple 1.5 GHz G4 PowerBook `FMUL` = 9 nsec.
- 4 floating point additions — on the G4 PowerBook `FADD` = 9 nsec.
- 3 floating point assignments — same computer, `FASSIGN` = 8 nsec.
- Miscellaneous other operations (integer $\rightarrow$ float & float $\rightarrow$ integer conversion, *etc.*)

So I could compute this notch filter algorithm *much faster* than 500 Hz (the IBM G4 PowerPC processor has pretty fast floating-point operations). Most of the time is spent in A/D conversion. There are very fast A/D converters – they just cost more. Also, DSP chips are *built* for fast floating point multiplies and adds. These days, many industrial and consumer communication and control devices (cell phones, military communications, *etc.*) have embedded DSP chips. It's a different world than in 1979 when I first developed this course. But some things never change...