Solution to HW4

AP5.1 We are given a system with closed loop transfer function

\[ T(s) = \frac{Y(s)}{R(s)} = \frac{96(s + 3)}{(s + 8)(s^2 + 8s + 36)} \]  \hspace{1cm} (1)

and asked to determine (a) the steady-state error corresponding to a unit step input, (b) the overshoot and 2\% settling time computed under the assumption of dominant complex poles, and (c) the actual system response, which we are asked to compare to the estimates obtained in (b).

Solution:

(a) We assume that \( E(s) \) is defined to be \( R(s) - Y(s) = R(s) - T(s)R(s) \) and note that \( T(0) = 96(3)/(8(36)) = 1 \). Then the steady state error due to a step disturbance \( R(s) = 1/s \) is, by the final value theorem,

\[ e_{ss} = \lim_{s \to 0} s R(s)[1 - T(s)] = 0. \]  \hspace{1cm} (2)

(b) As always, when asked to make the assumption that the complex poles dominate, we should first check to see whether this assumption is justified. Referring to the criterion on page 254 of the text, we note that here, \( \omega_n\zeta = 8/2 = 4 \) so

\[ 8 = \left| \frac{1}{(\pi)} \right| \not\geq 10|4| = 40 \]  \hspace{1cm} (3)

so the required assumption is NOT satisfied and we do not expect the predictions made for the settling time and percent overshoot to be accurate. However, since we were asked to make them, we use the method given on pp. 254–255 in Section 5.4 of the text. We note that \( \zeta = 8/(2\omega_n) = 4/6 = 2/3 \) and that \( a = 3 \) is the zero order coefficient in the factor \((s + 3)\) in the denominator so from the table in Figure 5.13(a) we find that with \( a/(\omega_n\zeta) = 3/4 = 0.75 \) and \( \zeta = 0.67 \) we expect approximately 40\% overshoot. We use the usual equation to estimate the settling time, obtaining

\[ T_{s,2\%} \approx \frac{4}{4} = 1. \]  \hspace{1cm} (4)

(c) The step response is shown in figure Figure 1. It was generated using the commands shown in the transcript below. Note that we determined the settling time by checking first to see whether the response deviated by more than 2\% over the time interval from \( t = 1 \) sec. to \( t = 1.5 \) sec. then seeing that it did not, checking backward from time \( t = 1 \) sec. to find the time at which the value of \( y \) first fell within 2\% of the final value. We find that \( T_{s,2\%} = 0.9375 \). The percent overshoot is found to be 33\% (rounding up to the nearest percent). We see that the settling time estimate obtained under the assumption of dominant second order poles is no substitute for actually calculating the values using Matlab.
```matlab
>> den = conv([1 8],[1 8 36])

  den =
       1   16  100  288

>> num = 96*[1 3]

  num =
       96   288

>> [y,t] = step(tf(num,den));
>> plot(t,y)
>> grid
>> title('AP5.1c Step Response')
>> xlabel('Time (s)')
>> ylabel('y(t)')
>> print -deps AP5.1c.eps
>> t(78)

  ans =
       1.0026

>> t(116)

  ans =
       1.4974

>> min(y([78:116]))

  ans =
       0.9829

>> y(70:78)'

  ans =

       Columns 1 through 7
       1.0356  1.0300  1.0248  1.0199  1.0154  1.0113  1.0075

       Columns 8 through 9
```
1.0041  1.0010

>> t(73)

ans =

0.9375

>> max(y)

ans =

1.3236

Figure 1: Plot of Step Response for AP5.1
AP5.5 In this problem we are given a block diagram of a system in which the zero of the controller can be varied. We are asked to (a) determine the steady state error due to a step input for the cases where the controller zero \( \alpha \) is (i) zero and (ii) nonzero, and (b) plot the response of the system for a step disturbance with \( \alpha = 0, 10, \) and 100. We are then asked to compare the results and choose the best of the given values for \( \alpha \).

**Solution:** Note that if \( \alpha = 0 \), the controller consists of a gain of one.

**Case (i), \( \alpha = 0 \):**

The closed loop transfer function is

\[
T(s) = \frac{Y(s)}{R(s)} = \frac{50(s + 2)}{(s + 3)(s + 4) + 50(s + 2)}. \tag{5}
\]

The error \( E(s) = R(s) - Y(s) \) is then \( E(s) = R(s)[1 - T(s)] \) and \( T(0) = 100/(12 + 100) = 25/28 \) so applying the final value theorem we obtain

\[
e_{ss} = \lim_{s \to 0} s \left( \frac{1}{s} \right) \left[ 1 - \frac{25}{28} \right] = \frac{3}{28}. \tag{6}
\]

**Case (ii) \( \alpha \neq 0 \):**

The closed loop transfer function is

\[
T(s) = \frac{Y(s)}{R(s)} = \frac{50(s + 2)(s + \alpha)}{s(s + 3)(s + 4) + 50(s + 2)(s + \alpha)}. \tag{7}
\]

The error \( E(s) = R(s) - Y(s) \) is then \( E(s) = R(s)[1 - T(s)] \) and \( T(0) = 1 \) so applying the final value theorem we obtain

\[
e_{ss} = \lim_{s \to 0} s \left( \frac{1}{s} \right) [1 - 0] = 0. \tag{8}
\]

Thus we see that a nonzero value of \( \alpha \) is preferable from the standpoint of steady-state error in the response of the closed loop system to a step input.

**Case (b):** The transfer function from the disturbance to the output is

\[
\frac{Y(s)}{D(s)} = \frac{50s(s + 2)}{s(s + 3)(s + 4) + 50(s + 2)(s + \alpha)}. \tag{9}
\]

so a step input of \( D(s) = 1/s \) leads to a steady state error of

\[
e_{ss} = y_{ss} = \lim_{s \to 0} s \left( \frac{1}{s} \right) \frac{50s(s + 2)}{s(s + 3)(s + 4) + 50(s + 2)(s + \alpha)} = 0. \tag{10}
\]

The transcript below generates the plots shown in Figure 2 which indicate that \( \alpha = 100 \) gives the smallest overshoot and fastest settling time.

```matlab
>> alpha=0; sys0 = feedback(tf(50*[1 2],[1 7 12]),tf([1 alpha],[1 0]))
```

**Transfer function:**

\[ 50 \ s^{-2} + 100 \ s \]
\[ s^3 + 57 s^2 + 112 s \]

\[ \Rightarrow \alpha = 10; \text{sys10} = \text{feedback}(\text{tf}(50*[1 \ 2],[1 \ 7 \ 12]),\text{tf}([1 \ \alpha],[1 \ 0])) \]

Transfer function:
\[ \frac{50 s^2 + 100 s}{s^3 + 57 s^2 + 612 s + 1000} \]

\[ \Rightarrow \alpha = 100; \text{sys100} = \text{feedback}(\text{tf}(50*[1 \ 2],[1 \ 7 \ 12]),\text{tf}([1 \ \alpha],[1 \ 0])) \]

Transfer function:
\[ \frac{50 s^2 + 100 s}{s^3 + 57 s^2 + 5112 s + 10000} \]

\[ \Rightarrow \text{step(sys0,'-'),sys10,'--',sys100,'--')} \quad \%\% \text{find a good range for the x-axis} \]
\[ \Rightarrow t=[0:.01:2]; \quad \%\% \text{by examining plot obtained with} \]
\[ \Rightarrow y0=\text{step(sys0,t)}; \quad \%\% \text{the step command with no output} \]
\[ \Rightarrow y10=\text{step(sys10,t)}; \quad \%\% \text{(it generates a plot instead)} \]
\[ \Rightarrow y100=\text{step(sys100,t)}; \]
\[ \Rightarrow \text{plot(t,y0,'b-',t,y10,'g-.',t,y100,'r--');} \]
\[ \Rightarrow \text{legend('\alpha=0','\alpha=10','\alpha=100')} \]
\[ \Rightarrow \text{grid} \]
\[ \Rightarrow \text{xlabel('Time (s)')} \]
\[ \Rightarrow \text{ylabel('y(t)'}) \]
\[ \Rightarrow \text{title('AP5.5b Response to Step Disturbance')} \]
\[ \Rightarrow \text{print -depsc AP5.5b.eps} \]

**AP5.6** We’re given the block diagram of an armature-current-controlled DC motor in Figure AP5.6. We are asked to (a) determine the steady state error resulting from a ramp input \( r(t) = t, t \geq 0 \) in terms of the gain constants \( K, K_b, \) and \( K_m \). We are then asked (b) to find a value of \( K \) such that with \( K_m = 10 \) and \( K_b = 0.05 \), the steady state error equals 1. Finally we are asked to (c) plot the responses to a unit step and a unit ramp input over a period of 20 seconds and state whether the responses are acceptable.

**Solution:**

(a) The transfer function is
\[
T(s) = \frac{Y(s)}{R(s)} = \frac{K K_m}{s(s + 0.1 + K_b K_m) + K K_m}.
\]  

The steady state error resulting from a ramp input is then
\[
e_{ss} = \lim_{s \to 0} s \left( \frac{1}{s^2} \right) [1 - T(s)] = \frac{K_b K_m + 0.01}{K K_m}.
\]
(b) Substituting for $K_m$ and $K_b$ in $e_{ss} = 1$ and solving for $K$ yields $K = 0.051$.

(c) The step and ramp responses obtained using the Matlab code below are shown in Figure 3. Given no other information as to what would constitute acceptable responses here, we judge the responses acceptable since the overshoot is not unduly large and the settling time is not unduly long. Note that to compute the ramp response, we used the step command on the system consisting of $1/s$ in series with the original closed loop transfer function since there is no “ramp” command in Matlab.

```matlab
>> Kb = 0.05; Km = 10; K = 0.051; sys = tf([K*Km],[1 Kb*Km+0.01 K*Km])
Transfer function:
0.51
-------------------
 s^2 + 0.51 s + 0.51

>> [ys,t]=step(sys);
```
>> [yr, t] = step(series(tf([1], [1 0]), sys), t);
>> subplot(2, 1, 1)
>> plot(t, ys)
>> ylabel('y(t)')
>> grid
>> title('AP5.6c Step and Ramp Responses')
>> subplot(2, 1, 2)
>> plot(t, yr)
>> grid
>> xlabel('Time (s)')
>> ylabel('y(t)')
>> print -deps AP5.6c.eps

Figure 3: Plot of Step and Ramp Responses for AP5.6.
DP5.4 We are given the system shown in the block diagram in Figure DP5.4 which has closed loop transfer function
\[
\frac{Y(s)}{R(s)} = T(s) = \frac{10K}{(s + 90)(s + 9)(s + 1) + 10K} = \frac{10K}{s^3 + 100s^2 + 909s + 810 + 10K} \quad (13)
\]

We are asked to (a) find a second order model for the closed loop system, (b) select a gain \(K\) yielding less than 15% overshoot and less than 12% steady state error in response to a unit step input, and (c) test the resulting design on the full third order system.

**Solution:** First I tried to approach this by obtaining expressions for the roots of the cubic equation obtained by setting the denominator equal to zero. I quickly determined that this was a bad idea as the expressions for the roots of a cubic are quite complicated even without one of the coefficients containing an unknown parameter \(K\).

(a) The next approach was to consider some sample values for \(K\) and see what happens. I chose sample values 1, 10, 100, and 1000. For these, I obtain using Matlab, the roots of the denominator of the closed loop system, then checked to see whether the criterion on page 254 was satisfied. Here’s a fragment of the code, for the case \(K = 10\).

```matlab
>> K = 10; den = conv([1 90],[1 10 9])+[0 0 0 10*K]
den =
    1   100   909   910
>> roots(den)
ans =
   -90.0139
   -8.8429
   -1.1432
```

```matlab
>> conv([1 -ans(2)],[1 -ans(3)]) % find the quadratic factor of (5.18)
ans =
    1.0000   9.9861  10.1096
>> ans(2)/2 % find the product zeta*omega_n
ans =
    4.9931
```

What I found was that it didn’t matter which value of \(K\) I used. The criterion was always satisfied, and the value of \(K\) didn’t have much influence on the value of
Accordingly, I concluded that I could substitute a second order approximation for the third order open loop transfer function, allowing me to simplify subsequent calculations. (Of course in the end, the verification in part (c) will tell whether this assumption is reasonable.)

We now write the open loop transfer function as

$$G_c(s)G(s) = \frac{10K}{(s + 90)(s + 1)(s + 9)} = \frac{10K/90}{(s/90 + 1)(s + 1)(s + 9)}$$

(14)

which has second order approximation, neglecting the fast pole,

$$G_c(s)G(s) \approx \frac{K/9}{(s + 1)(s + 9)}$$

(15)

leading to closed loop transfer function

$$T(s) \approx \frac{K/9}{s^2 + 10s + K/9}.$$  

(16)

(b) From Figure 5.8, page 251, we determine that we obtain overshoot less than 15% if we choose $\zeta \geq 0.52$ approximately. Solving $10 \leq 2\zeta \omega_n = 2(0.52)\sqrt{9 + K/9}$ for $K$ yields $100/1.1^2 \leq 9 + K/9$ or $K \leq 662$.

The steady state error associated with a step input will be

$$e_{ss} = \lim_{s \to 0} s \left(\frac{1}{s}\right) \left[1 - T(s)\right] = 1 - \frac{K}{81 + K}$$

(17)

so to obtain $e_{ss} < 12\%$ we need $81/(81 + K) < 0.12$ which requires (solving for $K$) $K \geq 594$.

Combining these requirements we have $594 < K < 662$. To be conservative, we choose $K = 628$.

(c) We test the behavior of the system with gain $K = 628$ in the following Matlab transcript, plotting the results in Figure 4, and determining that, for a step input, the percent overshoot of the third order system is 14.86% and the steady state error is 11.42%.

```
>> K = 628; sys = feedback(tf([K*10],conv([1 90],[1 10 9])),1)
Transfer function:
6280
-------------------------
 s^3 + 100 s^2 + 909 s + 7090

>> t=[0:.01:5];
>> [y,t]=step(sys,t);
>> plot(t,y)
>> grid
>> print -deps DP5.4c.eps
```
>> size(t)
ans =
   501    1
>> y(501)
ans =
   0.8858
>> 1-y(501)
ans =
   0.1142
>> (max(y)-y(501))/y(501)
ans =
   0.1486

MP5.3 We are given an open loop transfer function

\[ T(s) = \frac{\omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2} \]  

and asked to obtain and plot impulse responses for four pairs of values of \( \omega_n \) and \( \zeta \).

**Solution:** A transcript of the Matlab session is given below and the plots produced are shown in Figure 5. We notice that the two cases in which \( \zeta = 0 \) yield undamped oscillations of amplitude \( \omega_n^2 \) and that the two cases in which \( \zeta \neq 0 \) yield damped oscillations. Comparing our plots to those in Figure 5.17, page 260 we see that our case (i) \( \omega_n = 2 \) and \( \zeta = 0 \) corresponds to the 3rd plot in the 1st row (imaginary roots of magnitude greater than 1); case (ii) \( \omega_n = 2 \) and \( \zeta = 0.1 \) corresponds to the 1st plot in the 1st row\(^1\); case (iii) corresponds to the 3rd plot in the 2nd row; and case (iv) corresponds to the 2nd plot in the 2nd row\(^2\).

\[^1\]I cheated here. I cannot see much of a difference between the first two plots in the first row so I solved for the roots of the denominator for case (ii), finding them to be \(-2 \pm 1.99j\) which would be closer to the triangle for the 1st plot than that for the 2nd plot in the 1st row.

\[^2\]Roots are \(-2 \pm 0.9798j\).
Figure 4: Step Response for AP5.4.

Transfer function:
\[ \frac{4}{s^2 + 4} \]

```
>> [y,t]=impulse(sys);plot(t,y)
>> subplot(2,2,2)
>> wn=2;ze=0.1;sys=tf([wn^2],[1 2*ze*wn wn^2])
```

Transfer function:
\[ \frac{4}{s^2 + 0.4s + 4} \]

```
>> [y,t]=impulse(sys);plot(t,y)
>> subplot(2,2,3)
```
>> wn=1;ze=0;sys=tf([wn^2],[1 2*ze*wn wn^2])

Transfer function:
1
--------
s^2 + 1

>> [y,t]=impulse(sys);plot(t,y)
>> subplot(2,2,4)
>> wn=1;ze=0.2;sys=tf([wn^2],[1 2*ze*wn wn^2])

Transfer function:
1
--------------
s^2 + 0.4 s + 1

>> [y,t]=impulse(sys);plot(t,y)
>> title('\omega_n=1,\zeta=0.2')
>> subplot(2,2,1);title('\omega_n=2,\zeta=0')
>> subplot(2,2,2);title('\omega_n=2,\zeta=0.1')
>> subplot(2,2,3);title('\omega_n=1,\zeta=0')
>> for index=1:4,subplot(2,2,index);grid;end
>> print -deps MP5.3.eps

MP5.6 We are given a block diagram in Figure MP5.6, and asked to compare two different controllers, \( G_{c,a}(s) = 2 \) and \( G_{c,b} = 2 + 1/s \).

Solution:

(a) The first controller is implemented in the closed-loop transfer function \( tf1 \), which gives the transfer function from the input \( R(s) \) to the error \( E(s) \), and \( tf2 \), which gives the transfer function from the input \( R(s) \) to the output \( Y(s) \) in the Matlab script below. The plots are shown in Figure 6 The attitude error at 10 seconds is found to be 0.3001 rad.

(b) The second controller is implemented in the closed-loop transfer function \( tf3 \), which gives the transfer function from the input \( R(s) \) to the error \( E(s) \), and \( tf4 \), which gives the transfer function from the input \( R(s) \) to the output \( Y(s) \) in the Matlab script below. The plots are shown in Figure 7 The attitude error at 10 seconds is found to be -0.0074 rad. We see that the more complex controller improved the error.

>> \%\% MP5.6

>> forward = tf([1],[1])

Transfer function:
1
Figure 5: Plots for MP5.3.

\[ \omega_n = 2, \zeta = 0 \]

\[ \omega_n = 2, \zeta = 0.1 \]

\[ \omega_n = 1, \zeta = 0 \]

\[ \omega_n = 1, \zeta = 0.2 \]

\[ \text{>> reverse} = \text{series}(\text{tf([-20],[1 10]),tf([-1 -5],[1 3.5 6 0]))} \]

Transfer function:

\[ \frac{20 s + 100}{s^4 + 13.5 s^3 + 41 s^2 + 60 s} \]

\[ \text{>> tf1=} \text{feedback(forward,reverse)} \]

Transfer function:

\[ \frac{s^4 + 13.5 s^3 + 41 s^2 + 60 s}{s^4 + 13.5 s^3 + 41 s^2 + 80 s + 100} \]

\[ \text{>> tf2} = \text{feedback(reverse,forward)} \]
Figure 6: Plots for MP5.6a.

MP5.6a Ramp Responses, $\theta_d(t) = t/2$

Transfer function:
\[
\frac{20 s + 100}{s^4 + 13.5 s^3 + 41 s^2 + 80 s + 100}
\]

```matlab
>> t = [0:.01:10];
>> y = lsim(tf2,0.5*t,t);
>> e = lsim(tf1,0.5*t,t);
>> subplot(2,1,1);plot(t,y);grid;ylabel('y(t)');
>> subplot(2,1,2);plot(t,e);grid;ylabel('e(t)');
>> xlabel('Time (s)');
>> subplot(2,1,1);title('MP5.6a Ramp Responses, $\theta_d(t) = t/2$');
>> print -deps MP5.6a.eps
>> size(t)
```
ans =

    1001   1

>> e(1001)

ans =

    0.3001

>> reverse = series(tf([-20 1],[1 10 0]),tf([-1 -5],[1 3.5 6 0]))

Transfer function:

    20 s^2 + 99 s - 5
-------------------------------------
    s^5 + 13.5 s^4 + 41 s^3 + 60 s^2
>> reverse = series(tf(-10*[2 1],[1 10 0]),tf([-1 -5],[1 3.5 6 0]))

Transfer function:
    20 s^2 + 110 s + 50
--------------------
    s^5 + 13.5 s^4 + 41 s^3 + 60 s^2

>> tf3 = feedback(forward, reverse)

Transfer function:
    s^5 + 13.5 s^4 + 41 s^3 + 60 s^2
-----------------------------
    s^5 + 13.5 s^4 + 41 s^3 + 80 s^2 + 110 s + 50

>> tf4 = feedback(reverse, forward)

Transfer function:
    20 s^2 + 110 s + 50
-----------------------------
    s^5 + 13.5 s^4 + 41 s^3 + 80 s^2 + 110 s + 50

>> y = lsim(tf4,0.5*t,t);
>> e = lsim(tf3,0.5*t,t);
>> figure(2); subplot(2,1,1); plot(t,y); grid; ylabel('y(t)')
>> subplot(2,1,2); plot(t,e); grid; ylabel('e(t)')
>> subplot(2,1,1); title('MP5.6b Ramp Responses, \theta_d(t) = t/2')
>> subplot(2,1,2); xlabel('Time (s)')
>> print -deps MP5.6b.eps
>> e(1001)

ans =

    -0.0074