Experiment 2

Control System CAD and CAS using Matlab

Sahand Sabour

EEE220

Instrumentation and Control System

Student ID

1614650

Xi'an Jiaotong Liverpool University

$y_c =$

Problem 1

As the first step of solving this problem, given the following equation

$$\ddot{y} + 3\dot{y} + 2y = u \tag{1}$$

The Laplace transform of both sides of the equation would be calculated as follows

$$L\{\ddot{y} + 3\dot{y} + 2y\} = L\{u\}$$

$$s^2Y(s) - sy(0) - \dot{y}(0) + 3sY(s) - 3y(0) + 2Y(s) = \tfrac{1}{s}$$

$$s^2Y(s) + 3sY(s) + 2Y(s) = \tfrac{1}{s}$$

$$(s+1)(s+2)Y(s) = \tfrac{1}{s}$$

which gives

$$Y(S) = \frac{1}{s(s+1)(s+2)} = \frac{1}{s^3 + 3s^2 + 2s} \tag{2}$$

This equation can be further simplified as follows:

$$Y(S) = \frac{a}{s} + \frac{b}{s+1} + \frac{c}{s+2} \tag{3}$$

Where a,b, and c are calculated to be $\frac{1}{2}$, -1, and $\frac{1}{2}$ respectively. Consequently, by applying reverse Laplace transform on both sides of equation 3, $y(t)$ can be obtained.

$$y(t) = \frac{e^{-2t}}{2} - e^{-t} + \frac{1}{2} \tag{4}$$

The MATLAB code for this problem is as follows:

```
%Code for Problem 1 --> part 1
sys= tf([1],[1 3 2 0]);
t= 0:1:10;
y= 0.5*exp(-2*t)-exp(-t)+0.5;

subplot(3,1,1), plot(t,y),xlabel('time (seconds)'),ylabel('magnitude')
,title('Analytical solution'),grid on;
subplot(3,1,2), plot(impulse(t,sys)),xlabel('time (seconds)')
,ylabel('magnitude'),title('Impulse response'),grid on;
subplot(3,1,3), plot(step(t,sys)),xlabel('time (seconds)')
,ylabel('magnitude'),title('Step response'),grid on;
```

The following figure would be obtained as the result of compiling the above code:

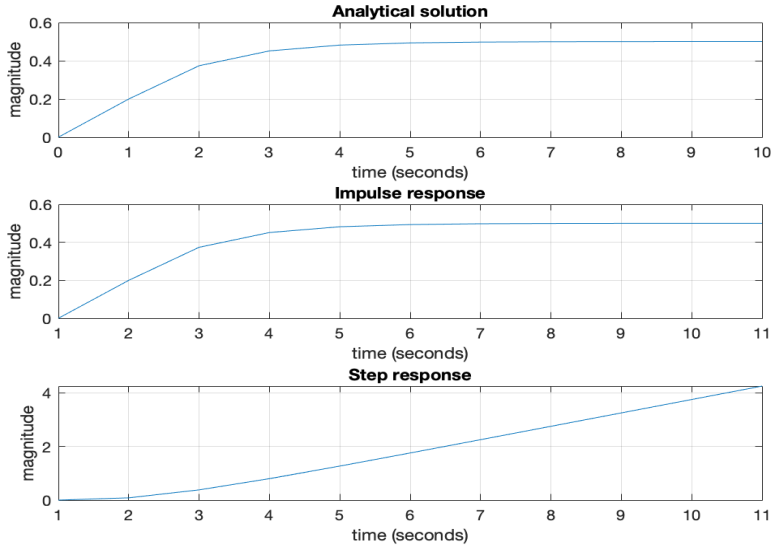

Figure 1: Co-plot of the analytical solution, impulse and step

In order to further verify the results, the following code can be executed; this would graph the impulse response and the analytical solution on the same plane (check Figure 2).

```
%Code for Problem 1 --> part 2
sys= tf([1],[1 3 2 0]);
t= 0:0.1:10;
y= 0.5*exp(-2*t)-exp(-t)+0.5;

plot(t,y,'xr')
hold on;
impulse(sys)
legend('Analytical solution', 'Impluse response')
grid on;
hold off;
```

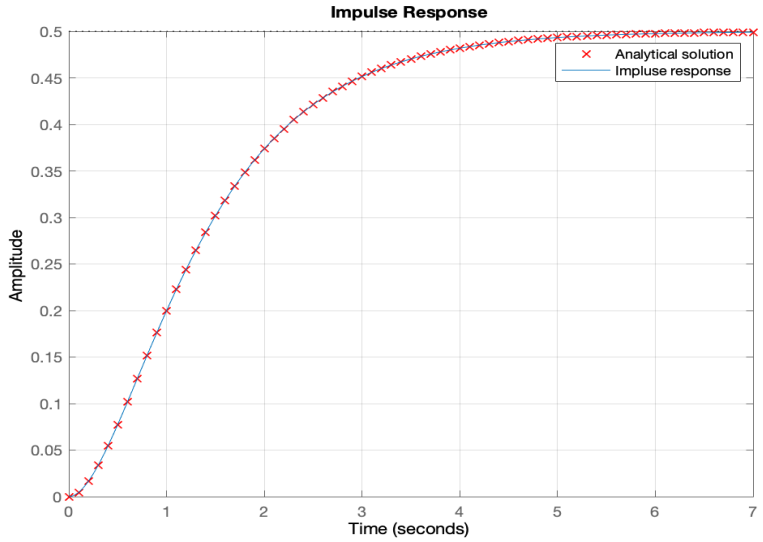

Figure 2: Graph of the analytical solution and the impulse response

Hence, the results of this section are successfully verified.

Problem 2

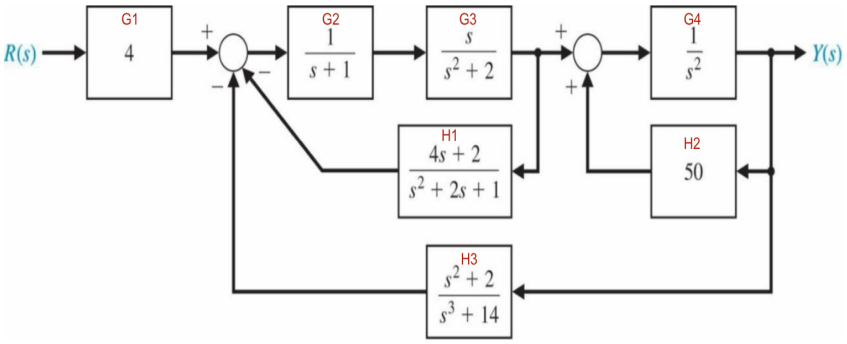

Figure 3: Figure for Problem 2

Given the above figure, the MATLAB code for this problem would be as follows:

```
%Code for Problem 2
%--part a
G1=tf([4], [1])
G2=tf([1],[1 1])
G3=tf([1 0],[1 0 2])
```

```
G4=tf([1],[1 0 0])
H1=tf([4 2],[1 2 1])
H2=tf([50],[1])
H3= tf([1 0 2],[1 0 0 14])

G23= series(G2, G3)
G23H1=feedback(G23, H1, -1)
G4H2= feedback(G4, H2, 1)

sys= series(G1, feedback(series(G23H1, G4H2), H3, -1))

%--part b
%This would draw the pole zero map
pzmap(sys)
%This would display poles and zeros, which is used for verification
[P,Z]= pzmap(sys)

%--part c
poles= pole(sys)
zeros= zero(sys)
```

Executing the above could would provide the following results:

a) the closed-loop transfer function:

$$\frac{4s^6+8s^5+4s^4+56s^3+112s^2+56s}{s^{10}+3s^9-45s^8-125s^7-200s^6-1177s^5-2344s^4-3485s^3-7668s^2-5598s-1400}$$

b) pole-zero map of the closed-loop transfer function:

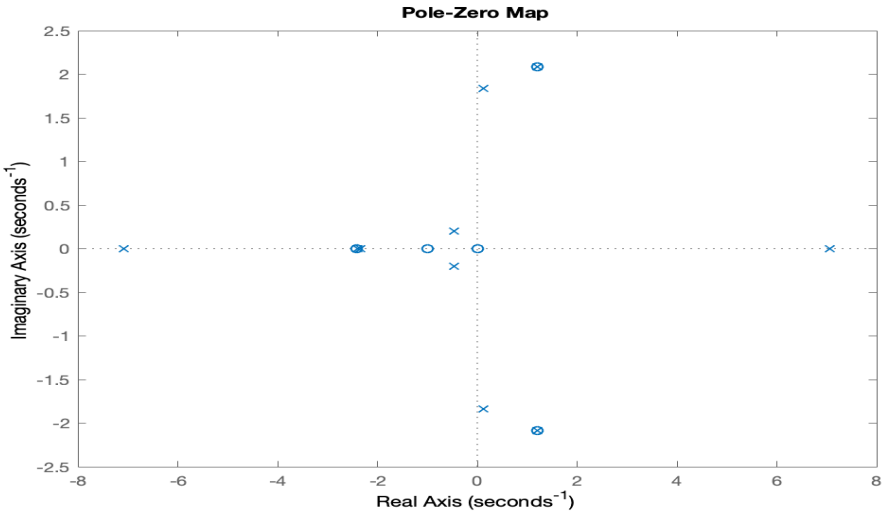

Figure 4: pole-zero map of sys

```
P =                          Z =

   7.0709 + 0.0000i             0.0000 + 0.0000i
  -7.0713 + 0.0000i             1.2051 + 2.0872i
   1.2051 + 2.0863i             1.2051 - 2.0872i
   1.2051 - 2.0863i            -2.4101 + 0.0000i
   0.1219 + 1.8374i            -1.0000 + 0.0000i
   0.1219 - 1.8374i            -1.0000 - 0.0000i
  -2.3933 + 0.0000i
  -2.3333 + 0.0000i
  -0.4635 + 0.1997i
  -0.4635 - 0.1997i
```

Figure 5: poles and zeroes based on pzmap function

c) Poles and zeros of the closed-loop transfer function:

```
poles =                      zeros =

   7.0709 + 0.0000i             0.0000 + 0.0000i
  -7.0713 + 0.0000i             1.2051 + 2.0872i
   1.2051 + 2.0863i             1.2051 - 2.0872i
   1.2051 - 2.0863i            -2.4101 + 0.0000i
   0.1219 + 1.8374i            -1.0000 + 0.0000i
   0.1219 - 1.8374i            -1.0000 - 0.0000i
  -2.3933 + 0.0000i
  -2.3333 + 0.0000i
  -0.4635 + 0.1997i
  -0.4635 - 0.1997i
```

Figure 6: poles and zeroes based on pole and zero functions

As displayed in figures 4, 5, and 6, the obtained poles and zeroes in sections b and c display the same results, which verifies the obtained values.

Problem 3

The MATLAB code for this problem is as follows:

```
%Code for Problem 3
%--part a
Ga= tf([1],[1 12])
state_var_Ga = ss(Ga)

%--part b
Gb= tf([1 5 3],[1 0 8 5])
state_var_Gb = ss(Gb)
```

```
%---part c
Gc=tf([1  4  1],[1  0  3  3  1])
state_var_Gc= ss(Gc)
```

Executing the provided code would give the solution for each of the following sections respectively.

a) $A = \begin{bmatrix}-12\end{bmatrix}, B = \begin{bmatrix}1\end{bmatrix}, C = \begin{bmatrix}1\end{bmatrix}$, and $D = \begin{bmatrix}0\end{bmatrix}$

b) $A = \begin{bmatrix}0 & -2 & -1.25\\ 4 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}, B = \begin{bmatrix}2\\0\\0\end{bmatrix}, C = \begin{bmatrix}0.5 & 0.625 & 0.375\end{bmatrix}, D = \begin{bmatrix}0\end{bmatrix}$

c) $A = \begin{bmatrix}0 & -1.5 & -1.5 & -0.5\\ 2 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}, B = \begin{bmatrix}2\\0\\0\\0\end{bmatrix}, C = \begin{bmatrix}0 & 0.25 & 1 & 0.25\end{bmatrix}, D = \begin{bmatrix}0\end{bmatrix}$

Problem 4

The MATLAB code for this problem is as follows:

```
%Code for Problem 4
%---part a
A=[0  1  0;0  0  1;-3  -2  -5];
B=[1;0;1];
C=[1  0  0];
D=[0];
sys= ss(A,B,C,D)

%---part b
t= 0:0.001:10;
u=t*0;
x0=[0;  -1;  1];
[y,T,  x]=lsim(sys,u,t,x0);
plot(T,y,t,u,'---'),xlabel('time'),ylabel('rad'),grid,
title('System response')

%---part c
dt=10
phi= expm(A*dt)
x= phi*x0
```

problem solutions

a) Transfer function:

$$T(S) = \frac{Y(S)}{U(S)} = \frac{s^2 + 5s + 3}{s^3 + 5s^2 + 2s + 3} \tag{5}$$

b) Graph of response of the system to initial condition:

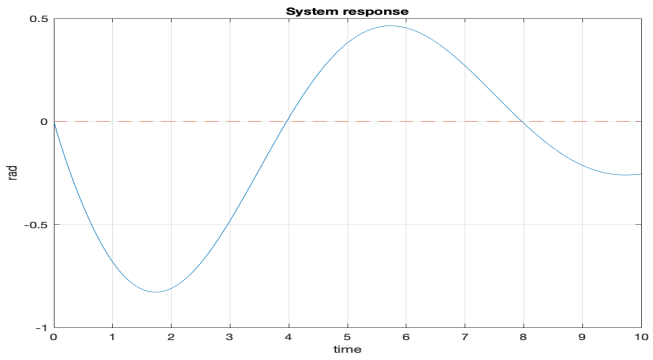

Figure 7: System Response to x_0 for $0 \geq t \geq 10$

c) State transition matrix and x(t) at t=10 for the initial condition:

```
phi =

    0.0853    0.3183    0.0637
   -0.1911   -0.0421   -0.0003
    0.0010   -0.1905   -0.0405
```

(a) State transition matrix

```
x =

   -0.2545
    0.0418
    0.1500
```

(b) x(t) at t=10

Problem 5

The MATLAB code for this problem is as follows:

```
%Code for Problem 5
sys= tf([1],[1 2 2 4 1 2]);
%-- part a
pzmap(sys);
```

```
%-- part b
poles= pole(sys);

%-- part c
step(sys), grid;
```

a) The characteristics equation for the given system is

$$q(s) = s^5 + 2s^4 + 2s^3 + 4s^2 + s + 2 = 0$$

Consequently, the Routh-Hurwitz array would be as follows:

S^5	1	2	1
S^4	2	4	2
S^3	ϵ	ϵ	0
S^2	2	2	0
S^1	ϵ	0	0
S^0	2		

where $\epsilon \to 0$.

There are two auxiliary polynomials at the s^4-line $(2s^4 + 2s^2 + 2)$ and s^2-line $(2s^2 + 2)$ respectively. There are no sign changes in the first column; however, there are two zero rows, which indicates presence of two roots on the imaginary axis ($j\omega$ axis). Therefore, the system is unstable.

One of the roots can be found by trial and error (s=2). Consequently, the characteristics equation can be written as the following:

$$q(s) = (s + 2)(s^4 + 2s^2 + 1) = (s + 2)(s^2 + 1)^2 \tag{6}$$

Which can be further simplified to the following equation:

$$q(s) = (s + 2)(s + j\omega)^2(s - j\omega)^2 \tag{7}$$

Hence,

there is a pole on $s = -2$,

two repeating poles on $s = -j\omega$, and

two repeating poles on $s = j\omega$.

b) The poles of T(S):

```
poles =

  -2.0000 + 0.0000i
  -0.0000 + 1.0000i
  -0.0000 - 1.0000i
   0.0000 + 1.0000i
   0.0000 - 1.0000i
```

Figure 9: poles of the transfer function

The above values were obtained as a result of executing the provided code. Moreover, the pole-zero map graph of the system was also plotted (Figure 10).

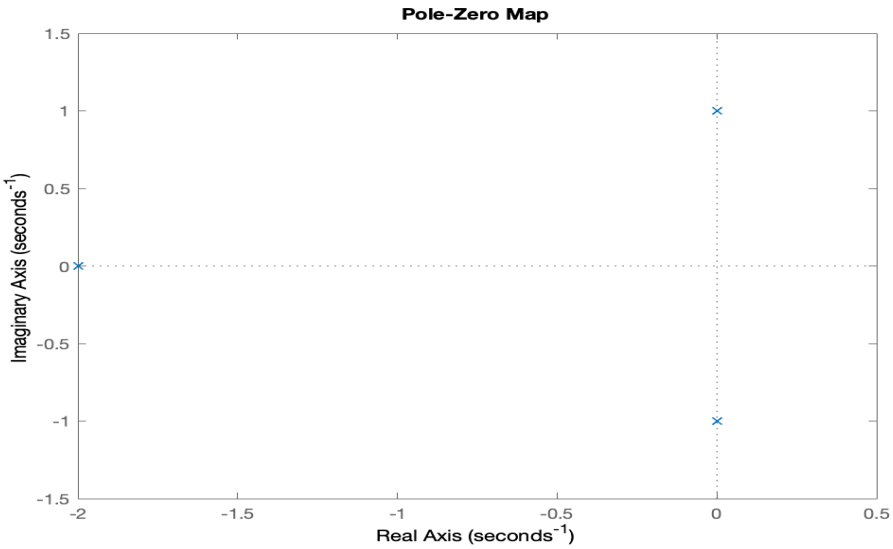

Figure 10: poles of the transfer function

According to both the graph and the obtained results (Figures 9 and 10 respectively), it can be seen that the calculated results of part (a) are successfully verified.

c) The unit step response:

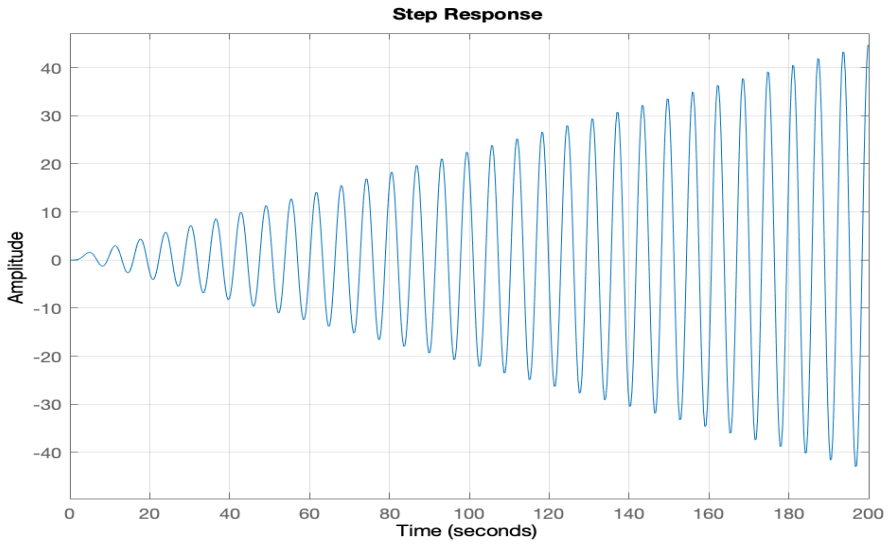

Figure 11: Unit step response

The obtained step response of the system illustrates distortions as well as no boundary for the output, which suggests the instability of the system. Hence, the results from the previous sections are verified.

Problem 6

The MATLAB code for this problem is divided into two sections.

```
%Code for Problem 6 --> part 1
Gcs= tf([1 4 4],[1 30 200]);
Gs= tf([1],[1 3 3.5 0]);

sys= series(Gcs, Gs);
rlocus(sys), grid;
```

Firstly, the root locus of the system is drawn (Figure 12). According to the figure, an arbitrary point whose Percent Overshoot (P.O) was slightly less that ten was chosen. Accordingly, K would be assumed to be equal to this value.

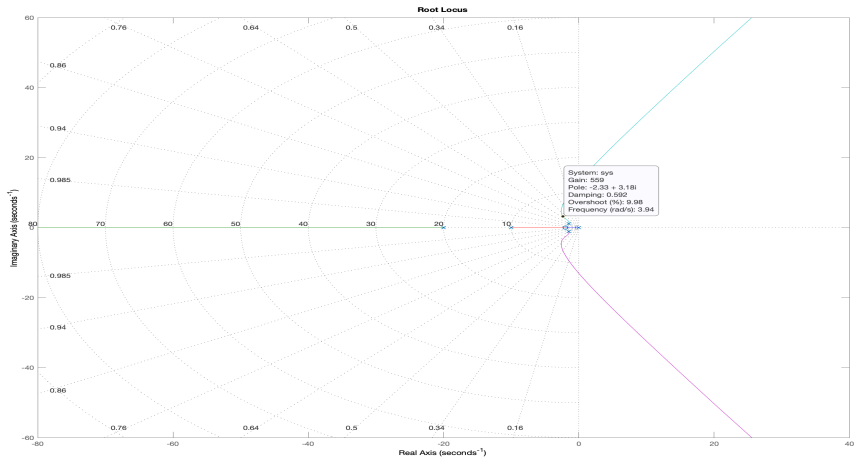

Figure 12: Root locus of the system

In order to verify the value of K, the step response of the system for K=559 was plotted using the following code.

```
%Code for Problem 6 --> part 2
Gcs= tf([1 4 4],[1 30 200]);
Gs= tf([1],[1 3 3.5 0]);

sys= series(Gcs, Gs);
sys1= feedback(series(559, sys),1);
step(sys1)
S = stepinfo(sys1)
```

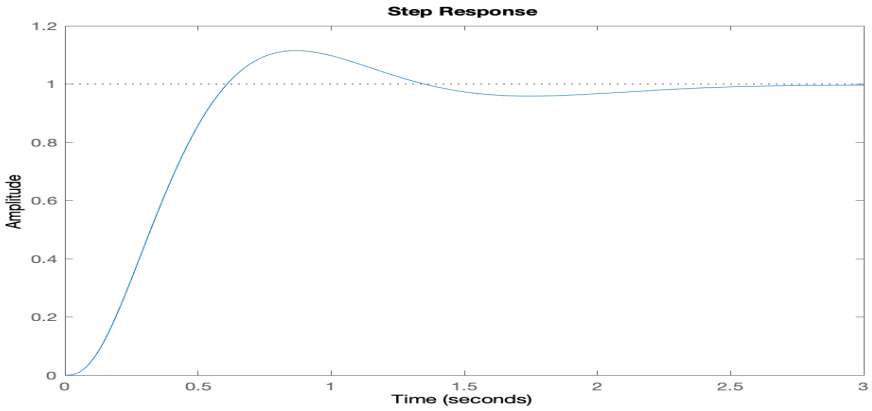

Figure 13: Step response of system for K=559

```
S =

  struct with fields:

      RiseTime: 0.3907
  SettlingTime: 2.2464
   SettlingMin: 0.9048
   SettlingMax: 1.1147
     Overshoot: 11.4730
    Undershoot: 0
          Peak: 1.1147
      PeakTime: 0.8608
```

Figure 14: Step response information for K=559

As shown in the above figure, the settling time is 2.2564 seconds, which satisfies the initial condition. Therefore, K=559 is verified to be a correct value.

Problem 7

The MATLAB code for this problem is as follows:

```
%Code for Problem 7
t= 0:0.01:10;
%-- part b => k=5
k=5;
sys1= tf([k k],[1 16 21.5+k 97.5+k]);
[y1, T1]= step(sys1, t);
%-- part b => k=10
k=10;
sys2= tf([k k],[1 16 21.5+k 97.5+k]);
[y2, T2]= step(sys2, t);
%-- part b => k=50
k=50;
sys3= tf([k k],[1 16 21.5+k 97.5+k]);
[y3, T3]= step(sys3, t);

plot(T1, y1, T2, y2, T3, y3),xlabel('time (seconds)')
,ylabel('magnitude')
title('System response corresponding to K'),grid on;
legend('K=5','K=10','K=50');
```

a) The closed loop transfer function:

$$T(s) = \frac{Y(s)}{U(s)} = \frac{G_c(s)G(s)}{1 + G_c(s)G(s)} \tag{8}$$

Where

$$G_c(s) = K\frac{s+1}{s+15} \tag{9}$$

and

$$G(s) = \frac{1}{s^2+s+6.5} \tag{10}$$

Hence, the transfer function would be obtained as follows:

$$T(s) = \frac{Y(s)}{U(s)} = \frac{G_c(s)G(s)}{1+G_c(s)G(s)} \tag{11}$$

which gives

$$T(s) = \frac{K(s+1)}{s^3+16s^2+(21.5+K)s+97.5+K} \tag{12}$$

b) The response of the closed-loop system corresponding to different values of K is displayed in the below figure.

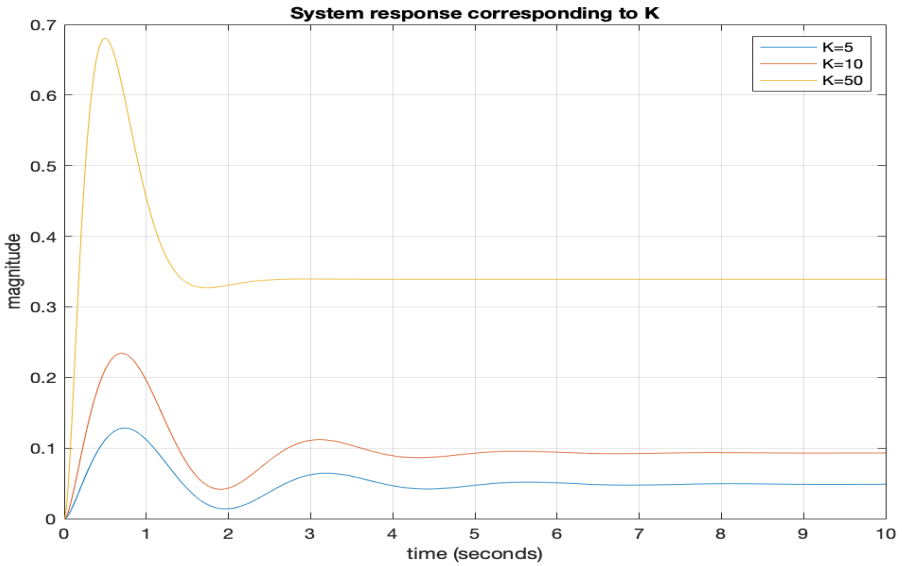

Figure 15: system response for K= 5, 10, and 50

c) Given that $K = 10$, $R(s) = 0$, and $T_d(s) = \frac{1}{s}$, the output $Y(S)$ of the system can be written as the following equation:

$$Y(s) = \frac{G(s)}{1+G_C(s)G(s)}T_d(s) = \frac{s+15}{s^3+16s^2+31.5s+107.5} \times \frac{1}{s} \tag{13}$$

Based on the Final Value Theorem, we have that

$$\text{steady-state value} = \lim_{s\to\infty} y(t) = \lim_{s\to 0} sY(s) \tag{14}$$

Which gives

$$\text{steady-state value} = \lim_{s\to 0} \frac{s+15}{s^3+16s^2+31.5s+107.5} = \frac{15}{107.5} \approx 0.1396 \tag{15}$$

Problem 8

The MATLAB code for this problem is as follows:

```
%Code for Problem 8
%--part a
sys_Ga= tf([30 12], [1 14 43 30])
rlocus(sys_Ga)
[r,K]= rlocus(sys_Ga)

%--part b
sys_Gb= tf([1 4 0 10 6 4], [1 7 4 1 1 10 1])
rlocus(sys_Gb)
[r,K]= rlocus(sys_Gb)
```

The required graph for each section of this problem are provided respectively below:

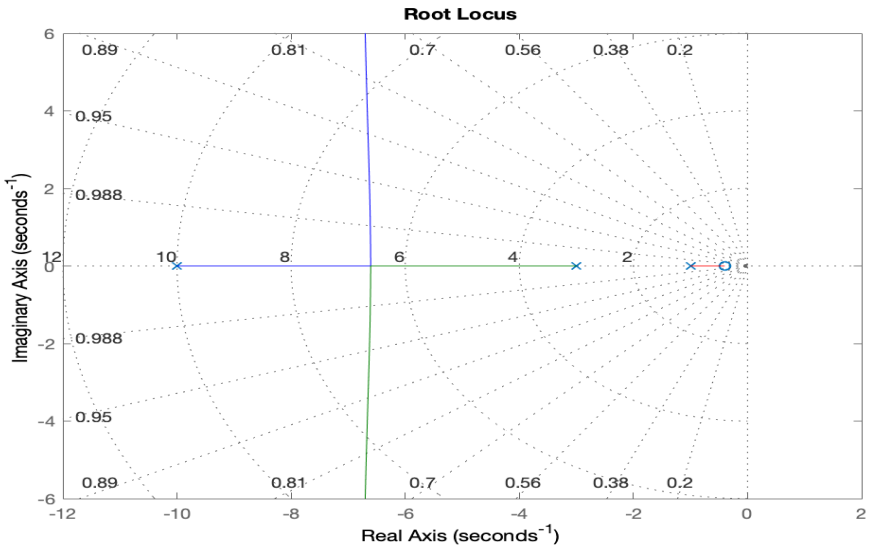

Figure 16: Root locus for part a

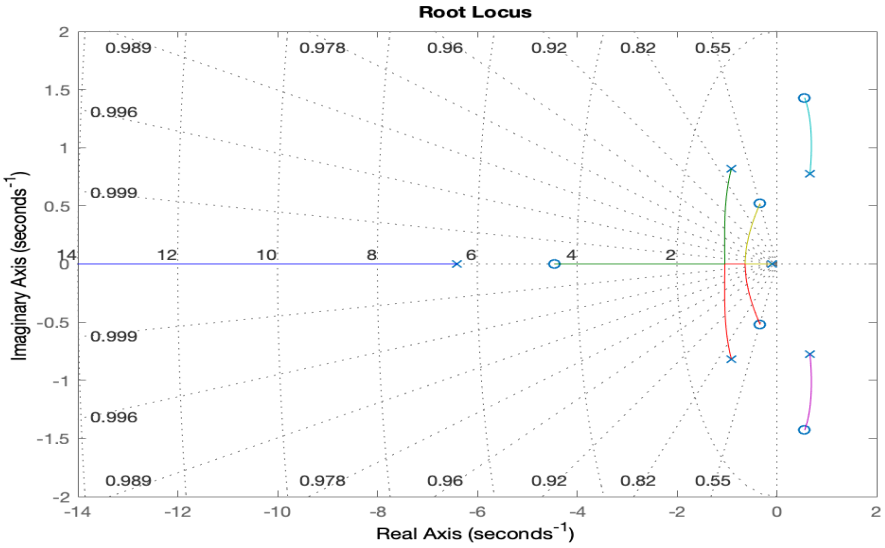

Figure 17: Root locus for part b

Problem 9

a) The MATLAB code for part a of this problem is as follows:

```
%Code for Problem 9
%-- part a
controller = tf([2],[1]);
servo= tf([-10], [1 10]);
model= tf([-1 -5],[1 3.5 6 0]);

G1= series(controller, servo);
G2 = series(G1, model);

sys= feedback(G2, 1);
t= 0:0.1:10;
thetaD= 0.5*t;

[y,T]= lsim(sys, thetaD,t);
plot(T, y, t, thetaD),xlabel('time (seconds)')
,ylabel('rads'),grid on;
legend('output','input');
```

The ramp response corresponding to the above code is displayed in the below figure.

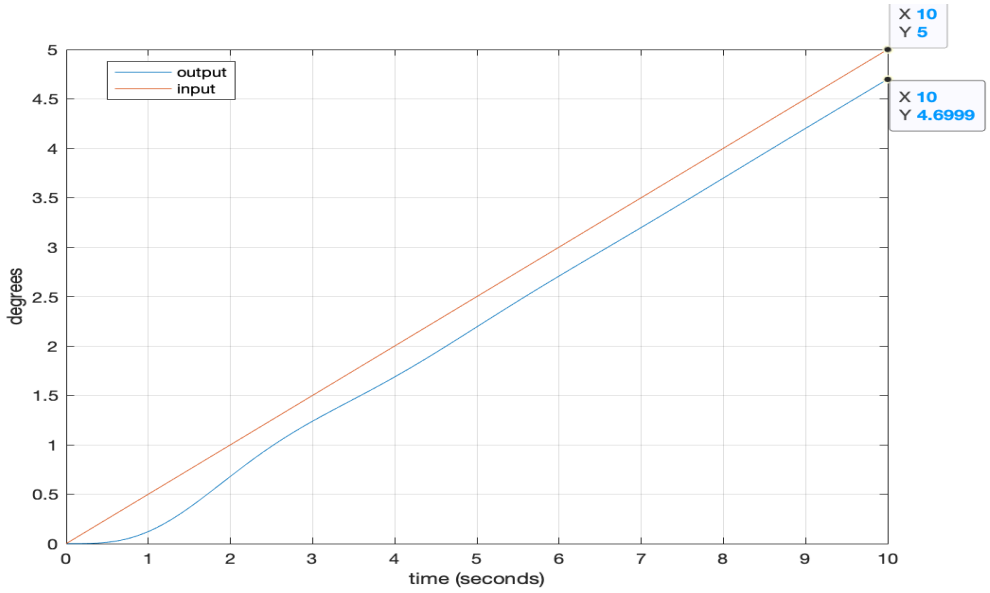

Figure 18: Ramp response for part a

b) The MATLAB code for part a of this problem is as follows:

```
%Code for Problem 9
%-- part b
controller = tf([2 1],[1 0]);
servo= tf([-10], [1 10]);
model= tf([-1 -5],[1 3.5 6 0]);

G1= series(controller, servo);
G2 = series(G1, model);

sys= feedback(G2, 1);
t= 0:0.1:10;
thetaD= 0.5*t;

[y,T]= lsim(sys, thetaD,t);
plot(T, y, t, thetaD),xlabel('time (seconds)')
,ylabel('rads'),grid on;
legend('output','input');
```

The ramp response corresponding to the above code is displayed in the below figure.

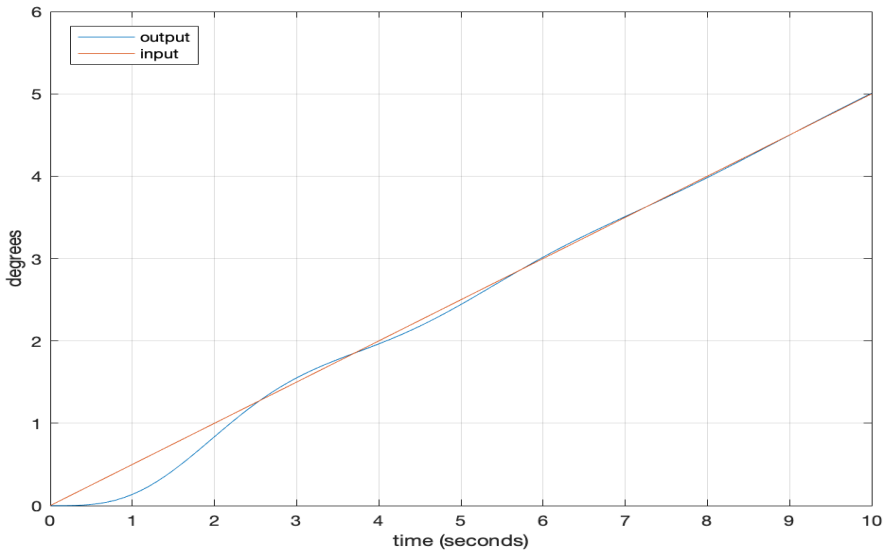

Figure 19: Ramp response for part b

Since this is a type-1 system, its steady-state error can be calculated via the following equation:

$$e_{ss} = \frac{A}{K_v} = \frac{0.5}{\lim\limits_{s\to 0} sG_C(s)G(s)} \tag{16}$$

Subsequently, the steady-state errors for each part can be obtained respectively as follows:

a)

$$e_{ss} = \frac{0.5}{\lim\limits_{s\to 0} s(2)(\frac{-10}{s+10})(\frac{-s-5}{s^3+3.5s^2+6s})} = 0.3 \tag{17}$$

b)

$$e_{ss} = \frac{0.5}{\lim\limits_{s\to 0} s(\frac{2s+1}{s})(\frac{-10}{s+10})(\frac{-s-5}{s^3+3.5s^2+6s})} = 0 \tag{18}$$

The obtained results show significantly less error in the second section where a more complex controller is implemented; This is verified by the graphs in figures 18 and 19.

Problem 10

The MATLAB code for part a of this problem is as follows:

```
%Code for problem 10
Gs= tf([1],[1 5 6]);
% --part a
Gc1= tf([1],[1]);
sys1= series(Gc1, Gs);
rlocus(sys1), grid, ylim([-5, 5]);
sys1_1= feedback(series(11.9, sys1),1);
S1 = stepinfo(sys1_1);

% --part b
Gc2= tf([1],[1 0]);
sys2= series(Gc2, Gs);
rlocus(sys2), grid;
sys2_1= feedback(series(4.45, sys2),1);
S2 = stepinfo(sys2_1);

% --part c
Gc3= tf([1 1],[1 0]);
sys3= series(Gc3, Gs);
rlocus(sys3), grid;
sys3_1= feedback(series(10, sys3),1);
S3 = stepinfo(sys3_1);

% --part d
t= 0:0.01:12;
subplot(3,1,1), plot(step(sys1_1, t)), grid,
xlabel('time (miliseconds)'), ylabel('amplitude'),
title('G_c(s)= K');
subplot(3,1,2), plot(step(sys2_1, t)), grid,
xlabel('time (miliseconds)'), ylabel('amplitude'),
title('G_c(s)= K/s');
subplot(3,1,3), plot(step(sys3_1, t)), grid,
xlabel('time (miliseconds)'), ylabel('amplitude'),
title('G_c(s)= K(1+1/s)');

% --part e
plot(t, step(sys1_1, t), t, step(sys2_1, t), t,step(sys3_1, t)),
legend('G_c(s)= K','G_c(s)= K/s', 'G_c(s)= K(1+1/s)'),
xlabel('Time (seconds)'), ylabel('amplitude'),
title('Graph of step response for the three controllers');
```

For sections a-c, the root locus of the system corresponding to the $G_C(s)$ of that section is plotted as the first step. Accordingly, an arbitrary point on the graph which has approximately the same percent overshoot as the required limit is used.

In this problem, the percent overshoot is required to be less than 10%. Hence, a point on each graph whose percent overshoot was approximately 10% was chosen. The gain of these points would be equal to K. Accordingly, the step response information of the system corresponding to the obtained value of K is printed to verify the results.
The root locus graphs as well as step response information for each of these three sections are provided respectively below:

a) When $G_C(s) = K$

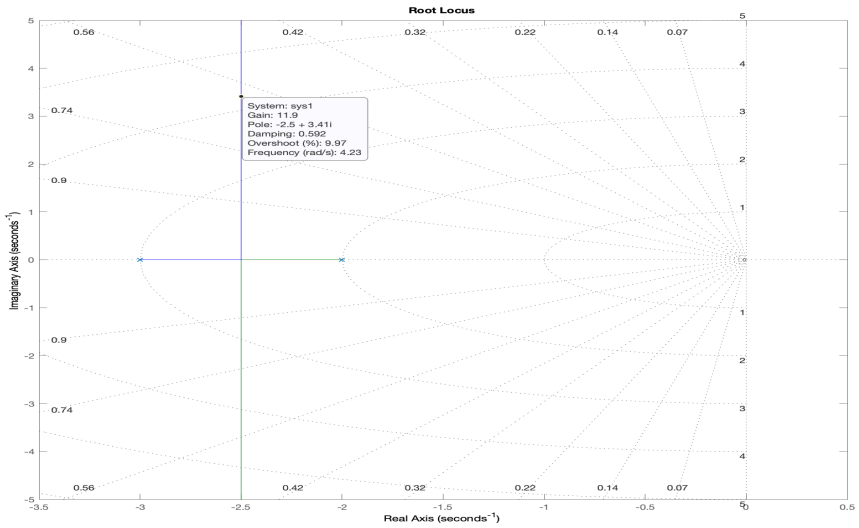

Figure 20: Root locus of the system when $G_C(s) = K$

```
S1 =

  struct with fields:

        RiseTime: 0.4334
    SettlingTime: 1.4005
     SettlingMin: 0.6003
     SettlingMax: 0.7314
       Overshoot: 10.0153
      Undershoot: 0
            Peak: 0.7314
        PeakTime: 0.9210
```

Figure 21: System response information when $G_C(s) = K$

Therefore, $K = 11.9$ is acceptable for when $G_C(s) = K$.

b) When $G_C(s) = K/s$

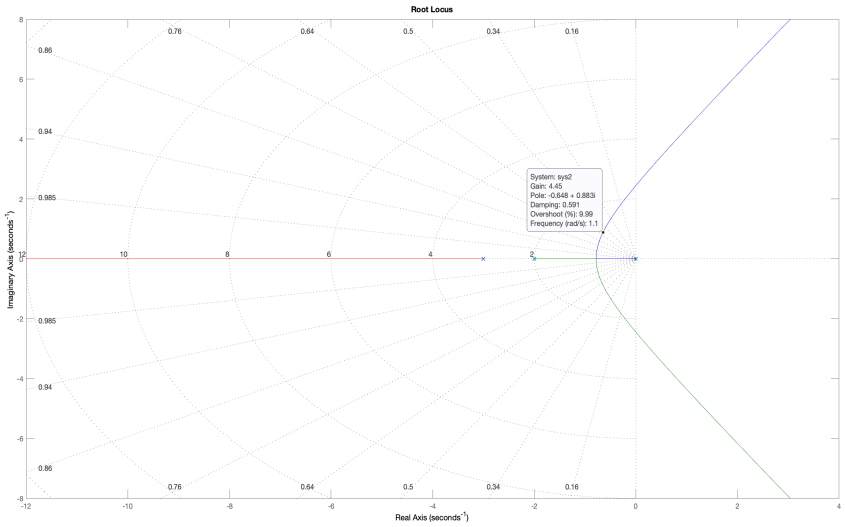

Figure 22: Root locus of the system when $G_C(s) = K/s$

```
S2 =

  struct with fields:

       RiseTime: 1.7651
   SettlingTime: 5.6980
    SettlingMin: 0.9192
    SettlingMax: 1.0948
      Overshoot: 9.4778
     Undershoot: 0
           Peak: 1.0948
       PeakTime: 3.8396
```

Figure 23: System response information when $G_C(s) = K/s$

Therefore, $K = 4.45$ is acceptable for when $G_C(s) = K/s$.

c) When $G_C(s) = K(1 + 1/s)$

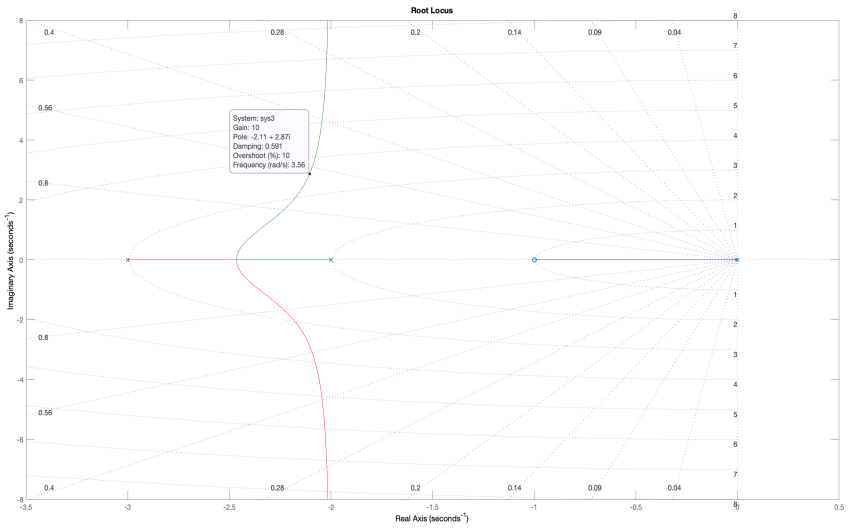

Figure 24: Root locus of the system when $G_C(s) = K(1 + 1/s)$

```
S3 =

  struct with fields:

      RiseTime: 0.7132
  SettlingTime: 3.2480
   SettlingMin: 0.9021
   SettlingMax: 0.9996
     Overshoot: 0
    Undershoot: 0
          Peak: 0.9996
      PeakTime: 8.2671
```

Figure 25: System response information when $G_C(s) = K(1 + 1/s)$

Therefore, $K = 10$ is acceptable for when $G_C(s) = K(1 + 1/s)$.

d) Co-plot of the unit step response for the closed loop system of each of the above controllers (a-c) if provided below:

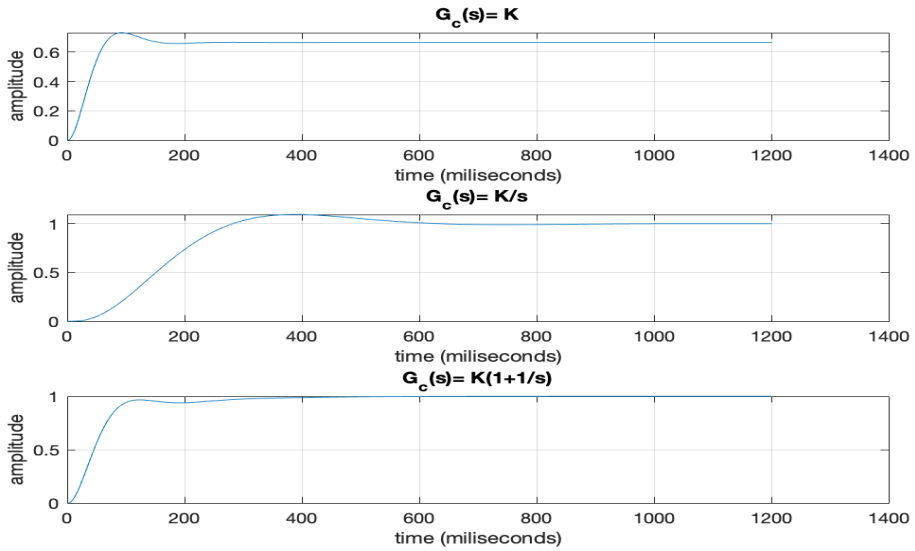

Figure 26: Co-plot of the unit step response for the closed loop system

e) In order to compare and contrast the three controllers, the step response of each controller is plotted (Figure 17).

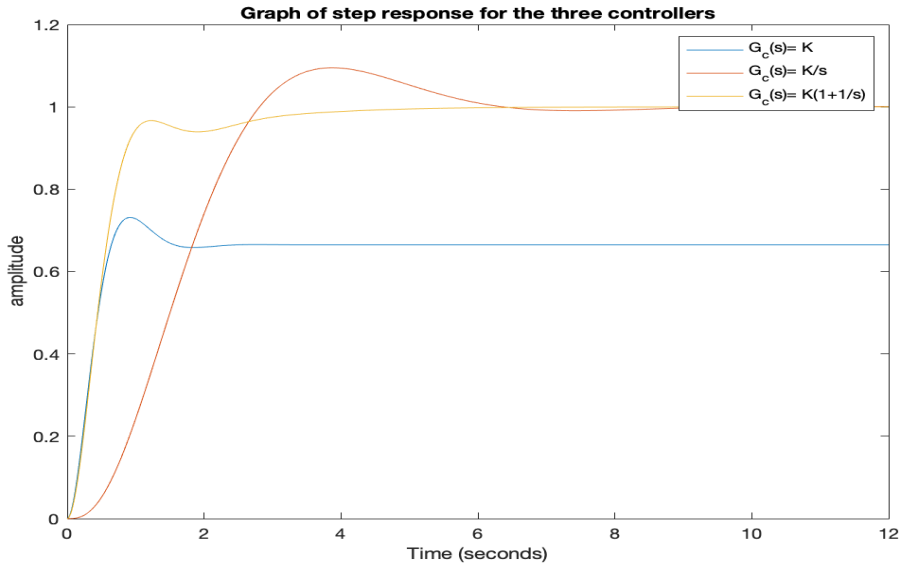

Figure 27: Co-plot of the unit step response for the closed loop system

As shown in the above figures, the settling times in sections a and c are considerably less than the settling time in section b (approximately half). This can be verified by checking the values in Figures 21, 23, and 25. Hence, it is suggested that the settling time in proportional controllers is much less than the settling time in integral controllers.

Furthermore, the steady-state error e_{ss} for a step input $\frac{A}{s}$, where A is the magnitude of the step input, can be obtained via the following equation:

$$e_{ss} = \lim_{s\to 0} s\frac{\frac{A}{s}}{1 + G_C(s)G(s)} = \frac{A}{1 + \lim_{s\to 0} G_C(s)G(s)} \tag{19}$$

Hence, the steady-state error for each of the above controllers can be calculated:

a)

$$e_{ss} = \frac{A}{1 + \lim_{s\to 0} K(\frac{1}{s^2+5s+6})} = \frac{A}{1 + \lim_{s\to 0} \frac{11.9}{s^2+5s+6}} \approx 0.33 \times A \tag{20}$$

b)

$$e_{ss} = \frac{A}{1 + \lim_{s\to 0}(\frac{K}{s})(\frac{1}{s^2+5s+6})} = \frac{A}{1 + \lim_{s\to 0} \frac{4.45}{s^3+5s^2+6s}} = 0 \tag{21}$$

c)

$$e_{ss} = \frac{A}{1 + \lim_{s\to 0}(\frac{K(s+1)}{s})(\frac{1}{s^2+5s+6})} = \frac{A}{1 + \lim_{s\to 0} \frac{10s+10}{s^3+5s^2+6s}} = 0 \tag{22}$$

Therefore, it is shown that the steady state error in integral controllers is considerably less than the error in proportional controllers.

Lastly, by reviewing the system response information corresponding to each control (Figures 21, 23, and 25), it can be noticed that the overshoot in the proportional integral controller is 0, which is significantly less than the overshoot in the other two controllers.

In conclusion, implementing a proportional integral (PI) controller would decrease the settling time of an integral controller, decrease the steady-state error of a proportional controller, and decrease the overshoot of the two mentioned controllers likewise. Hence, implementation of this controller would be highly beneficial.