EEE575

ASSIGNMENT 3

Q1

The paper "The Dilemma of PID Tuning" by Somefun et al. reviews the inherent challenges in designing and tuning proportional-integral-derivative (PID) controllers, a fundamental element of modern feedback control systems.

The PID control law originated in the early 20th century with Nicolas Minorsky's research on ship steering systems, which mimicked the "perception-action" principle of a skilled helmsman. Despite its computational simplicity, the PID controller remains the most widely used controller in industry due to its ubiquity and robust performance across a wide range of applications.

The paper primarily discusses that tuning the three PID parameters—proportional gain, integral time constant, and derivative time constant—is an NP-hard problem. This complexity presents significant challenges for engineers and designers, ultimately leading to costly and complex industrial automation solutions. The authors categorize existing tuning methods into three main categories: plant model-based methods, non-plant model-based methods, and hybrid methods.

A key limitation highlighted in the paper is that a large number of PID controllers in operation are either in manual mode or their performance degrades over time, requiring costly returning. This paper also explores the growing interest in fractional-order PID (FOPID) controllers, concluding that while they offer additional degrees of tuning freedom, their high computational cost and implementation complexity hinder their widespread industrial adoption. Furthermore, since fractional-order representations are often approximated using high-order integer filters, their purported advantages are questionable.

This paper demonstrates that while PID controllers are an enduring cornerstone of control theory, their practical application is not without significant challenges. The authors propose a forward-thinking solution: closed-loop input-output settling time can be used as a sufficient system property for control design. This highlights that finding more efficient and robust tuning methods remains an open problem even for classical control laws.

Q2

$$G = \frac{1}{s^2 + 2\gamma s + 1}$$

$$D = \frac{K(s+a)}{s+b}$$

A

Type 1: 1 pole at the origin

$$G \cdot D = \frac{1}{s^2 + 2\gamma s + 1} \cdot \frac{K(s+a)}{s+b} = \frac{K(s+a)}{(s^2 + 2\gamma s + 1)(s+b)}$$

$$GD = \frac{K(s+a)}{(s^2 + 2\gamma s + 1)(s+b)}$$

$$s^2 + 2\gamma s + 1 = 0$$

$$s = \frac{-2\gamma \pm \sqrt{4\gamma^2 - 4}}{2} = -\gamma \pm \sqrt{\gamma^2 - 1}$$

So, no matter what value γ takes, there will be two roots

$$s + b = 0$$

$$s = -b$$

For the system to be Type 1, we need the open loop transfer function to have exactly 1 pole at the origin.

If b=0, there is a peak at s=0 → Type 1

If a=0, the s in the numerator cancels out the s in the denominator, degenerating to Type 0.

$$b = 0, a \neq 0,\ \ k \neq 0$$

Therefore, must be used to ensure that the origin and poles are not canceled out,

Thus G(s)D(s) has exactly one extreme point at the origin returning the system to Type 1.

B

When b = 0

$$GD = \frac{K(s + a)}{s(s^2 + 2\gamma s + 1)}$$

$$T_d = \frac{G \cdot D}{1 + G \cdot D} = \frac{\frac{K(s + a)}{s(s^2 + 2\gamma s + 1)}}{1 + \frac{K(s + a)}{s(s^2 + 2\gamma s + 1)}}$$

$$T_d = \frac{K(s + a)}{s^3 + 2\gamma s^2 + (1 + K)s + Ka}$$

$$s^3 + 2\gamma s^2 + (1 + K)s + Ka = 0$$

1	s^3	1	1+K	0
2	s^2	2γ	Ka	0

3	s^1	b_1	0	0
4	s^0	Ka	0	0

$$
\begin{aligned}
b_1 &= -\frac{\det\begin{bmatrix} 1 & 1+K \\ 2\gamma & Ka \end{bmatrix}}{2\gamma} \\
&= -\frac{1\cdot Ka - 2\gamma\cdot(1+K)}{2\gamma} \\
&= -\frac{Ka - 2\gamma(1+K)}{2\gamma} \\
&= \frac{2\gamma(1+K) - Ka}{2\gamma} \\
&= \frac{2\gamma + 2\gamma K - Ka}{2\gamma} \\
&= \frac{2\gamma + K(2\gamma - a)}{2\gamma}
\end{aligned}
$$

So, we have

1>0

2γ>0, to γ>0

$$\frac{2\gamma + K(2\gamma - a)}{2\gamma} > 0$$

Ka>0

Because 2γ>0,

$$2\gamma + K(2\gamma - a) > 0$$

$$\frac{aK}{\gamma(1+K)} < 2$$

$$2\gamma(1+k) - ka > 0$$
$$2\gamma(1+k) > ka$$
$$1 + k > ka/2\gamma$$
$$(1-a)k > \frac{1}{2\gamma} - 1$$
$$k > \frac{1-2\gamma}{2\gamma(1-a)}$$

$$\begin{gathered} 2\gamma(1+k) - ka > 0 \\ 2\gamma(1+k) > ka \\ 2\gamma(1+k) > 0 \\ (1+k) > 0 \\ k > -1 \end{gathered}$$

So, the answer is

$$b = 0, \gamma > 0, K > -1, a \neq 0, Ka > 0, 2\gamma(1+K) > Ka$$

Q3

$$G = \frac{1}{(s+1)(5s+1)}$$

A

$$k_p = 19, k_i = 0.5, k_d = \frac{4}{19}$$

$$D = K_P$$

$$D = K_P + K_D S$$

$$D = K_p + K_L/S + K_D S$$

Case1:

$$D = K_P = 19$$

$$L = G \cdot D_{cl} = \frac{19}{(s+1)(5s+1)}$$

Step:

$$k_P = \lim_{s \to 0} L(s) = 19$$

$$e_{ss} = \frac{1}{1+19} = 0.05$$

Slope:

$$k_v = \lim_{s \to 0} sL(s) = 0$$

$$e_{ss} = \infty$$

Case2:

$$D = K_P + K_D S = 19 + \frac{4}{19}s$$

$$\begin{aligned} L(s) &= G(s)D(s) \\ &= \frac{1}{(s+1)(5s+1)}\left(19 + \frac{4}{19}s\right) \\ &= \frac{361+4s}{19(s+1)(5s+1)} \end{aligned}$$

$$L(s) = \frac{361+4s}{19(s+1)(5s+1)}$$

Step:

$$k_P = \lim_{s \to 0} L(s) = 19$$

$$e_{ss} = \frac{1}{1+19} = 0.05$$

Slope:

$$\begin{aligned} K_v &= \lim_{s \to 0} sL(s) \\ &= \lim_{s \to 0} s\frac{361+4s}{19(s+1)(5s+1)} \\ &= \lim_{s \to 0} \frac{361s+4s^2}{19(s+1)(5s+1)} \\ &= 0 \end{aligned}$$

$$K_v = 0$$

$$e_{ss} = \infty$$

Case3:

$$D = K_p + K_L/S + K_D S = 19 + \frac{1}{2s} + \frac{4}{19}s$$

$$L(s) = G(s)D(s) =$$
$$(19 + \frac{0.5}{s} + \frac{4}{19}s)(\frac{1}{(s+1)(5s+1)})$$
$$= \frac{0.210526s^2 + 19s + 0.5}{5s^3 + 6s^2 + s}$$

$$L(s) = \frac{0.210526s^2 + 19s + 0.5}{5s^3 + 6s^2 + s}$$

Step:

Still there is an integrator (Type 1)

$$e_{ss} = 0$$

Slope:

$$K_v = \lim_{s\to 0} sL(s) = \lim_{s\to 0} sL(s)$$
$$= \lim_{s\to 0} \frac{0.210526s^2 + 19s + 0.5}{5s^2 + 6s + 1} = \frac{0.5}{1} = 0.5$$
$$K_v = 0.5$$

$$e_{ss} = \frac{1}{K_v} = \frac{1}{0.5} = 2$$

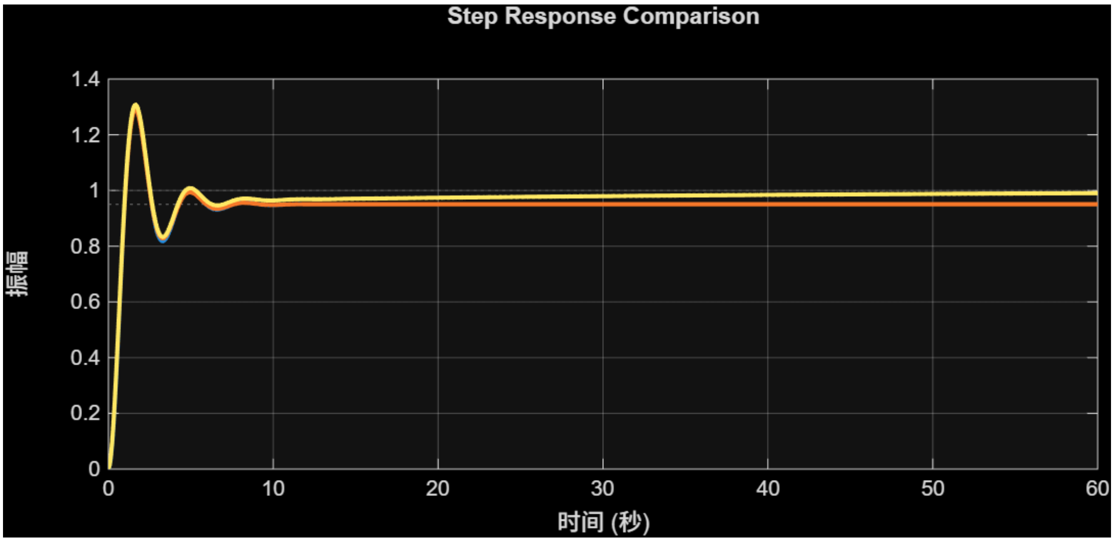
$$e_{ss} = 2$$

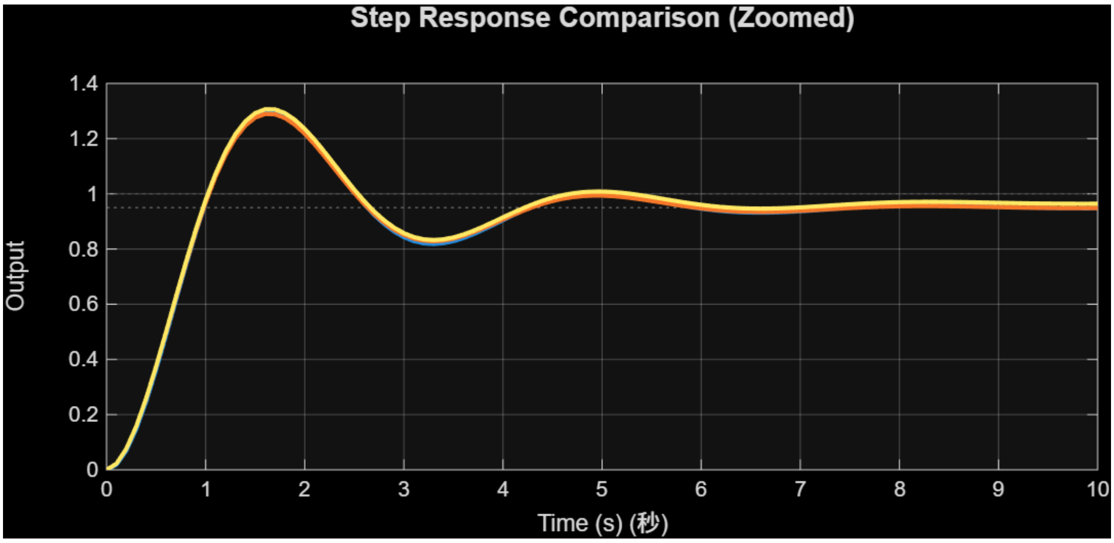
Controller	**System Type**	**Step Error**	**Ramp Error**	**Error Constant**
P $k_p = 19$	Type 0	0.05	∞	$k_p = 19$
PD $k_p = 19, k_i = 0.5$	Type 0	0.05	∞	$k_p = 19$
PID $k_p = 19, k_i = 0.5, k_d = \frac{4}{19}$	Type 1	0	2	$K_v = 0.5$

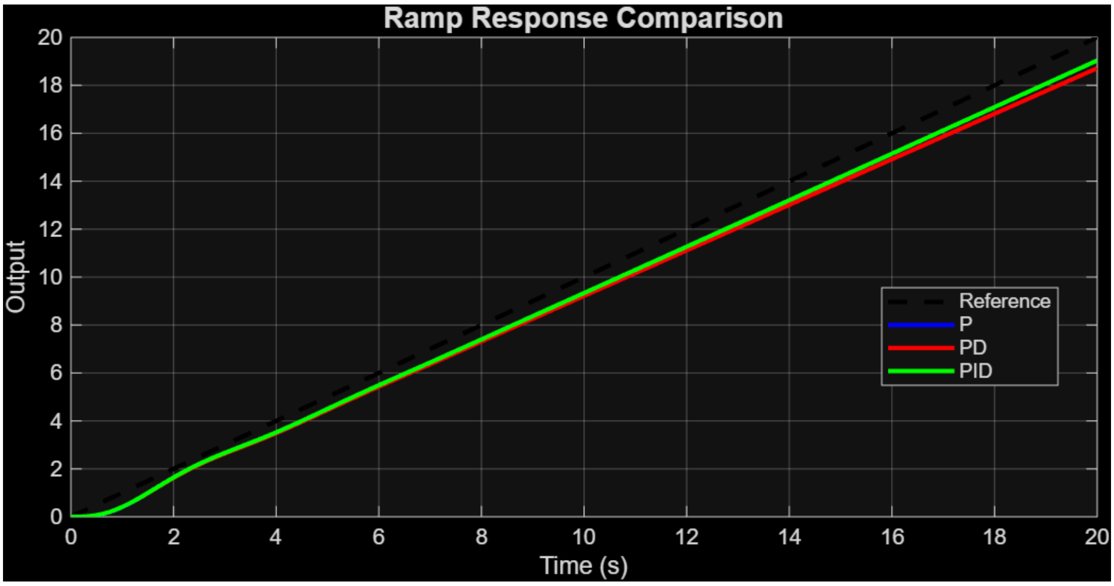
B

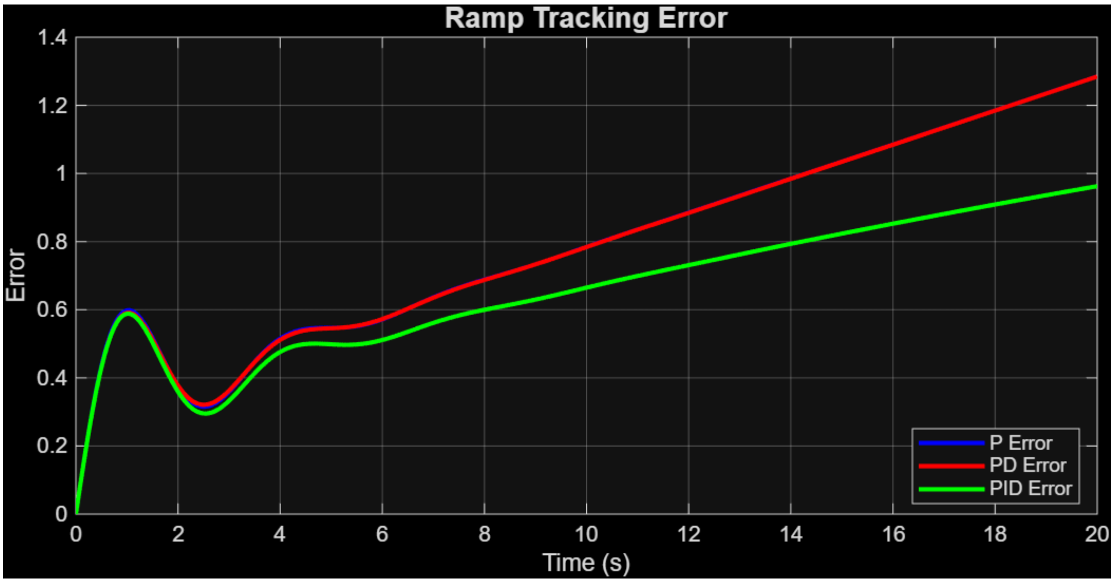
```
>> ASS33BF
Steady-state errors for step input:
P control: 0.0500
PD control: 0.0500
PID control: 0.0000
Ramp tracking errors:
P control: 1.2850
PD control: 1.2845
PID control: 0.9626
>>
```

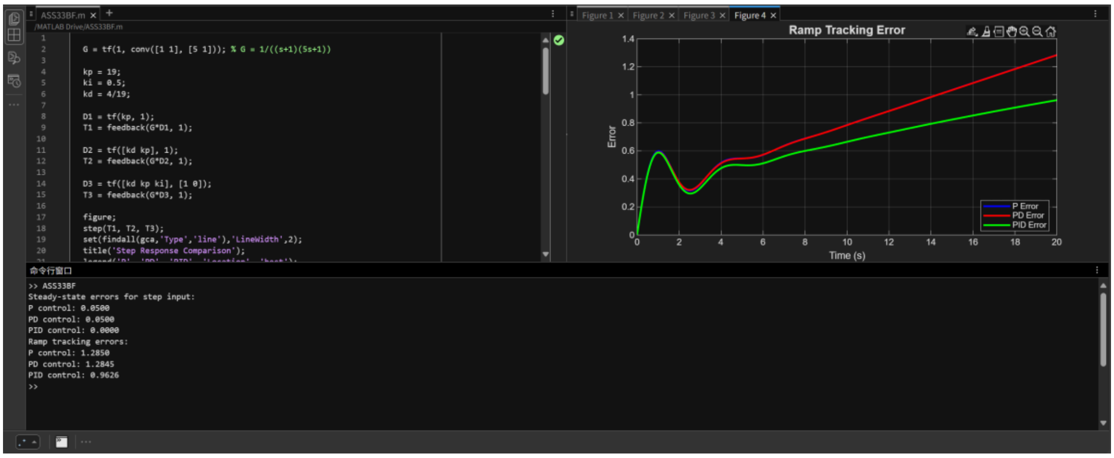
lotting unit-step and ramp response for tracking




Step Response Comparison (Zoomed)
Output
Time (s) (秒)


Ramp Response Comparison
Output
Time (s)
Reference
P
PD
PID


Ramp Tracking Error
Error
Time (s)
P Error
PD Error
PID Error



CODE

```
G = tf(1, conv([1 1], [5 1])); % G = 1/((s+1)(5s+1))

kp = 19;
ki = 0.5;
kd = 4/19;

D1 = tf(kp, 1);
T1 = feedback(G*D1, 1);

D2 = tf([kd kp], 1);
T2 = feedback(G*D2, 1);

D3 = tf([kd kp ki], [1 0]);
T3 = feedback(G*D3, 1);

figure;
step(T1, T2, T3);
set(findall(gca,'Type','line'),'LineWidth',2);
title('Step Response Comparison');
legend('P', 'PD', 'PID', 'Location', 'best');
grid on;



% --- Step response ---
figure;
step(T1, T2, T3, 10);
set(findall(gca,'Type','line'),'LineWidth',2);
title('Step Response Comparison
(Zoomed)','FontSize',13,'FontWeight','bold');
```

```
legend('P','PD','PID','Location','best');
xlabel('Time (s)'); ylabel('Output'); grid on;

% --- Ramp response ---
t = (0:0.05:20)';
u_ramp = t;
y1 = lsim(T1, u_ramp, t);
y2 = lsim(T2, u_ramp, t);
y3 = lsim(T3, u_ramp, t);

figure;
plot(t,u_ramp,'k--','LineWidth',2); hold on;
plot(t,y1,'b-','LineWidth',2);
plot(t,y2,'r-','LineWidth',2);
plot(t,y3,'g-','LineWidth',2);
title('Ramp Response Comparison','FontSize',13,'FontWeight','bold');
legend('Reference','P','PD','PID','Location','best');
xlabel('Time (s)'); ylabel('Output'); grid on;

% --- Ramp tracking error ---
figure;
plot(t,u_ramp-y1,'b-','LineWidth',2); hold on;
plot(t,u_ramp-y2,'r-','LineWidth',2);
plot(t,u_ramp-y3,'g-','LineWidth',2);
title('Ramp Tracking Error','FontSize',13,'FontWeight','bold');
legend('P Error','PD Error','PID Error','Location','best');
xlabel('Time (s)'); ylabel('Error'); grid on;

fprintf('Steady-state errors for step input:\n');
fprintf('P control: %.4f\n', 1 - dcgain(T1));
fprintf('PD control: %.4f\n', 1 - dcgain(T2));
fprintf('PID control: %.4f\n', 1 - dcgain(T3));
fprintf('Ramp tracking errors:\n');
fprintf('P control: %.4f\n', u_ramp(end) - y1(end));
fprintf('PD control: %.4f\n', u_ramp(end) - y2(end));
fprintf('PID control: %.4f\n', u_ramp(end) - y3(end));
```