


LAB 7: DC MOTOR POSITION CONTROL

Course Number	MENG 3510
Course Title	Control Systems
Semester/Year	Winter / 2025

Lab/Tutorial Report No.	Lab 7
Report Title	DC Motor Position Control
Section No.	ONA
Group No.	2
Submission Date	2025-03-09
Due Date	2025-03-09

Student Name	Signature*	Total Mark
Joshua Mongal		/50
Mikaeel Khanzada	<i>Mikaeel Khanzada</i>	/50
Michael McCorkell	<i>Michael McCorkell</i>	/50
		/50

* By signing the above, you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a ZERO on the work or possibly more severe penalties.

<https://academic-regulations.humber.ca/2021-2022/17.0-ACADEMIC-MISCONDUCT>

LAB 7 Grading Sheet

Student Name: Joshua Mongal	Student Name: Mikaeel Khanzada
Student Name: Michael McCorkell	Student Name:
Part 1: Proportional Position Control	/15
Part 2: PD Position Control	/20
PART 3: PID Position Control	/10
General Formatting: Clarity, Writing style, Grammar, Spelling, Layout of the report	/5
Total Mark	/50

LAB 7: DC MOTOR POSITION CONTROL

OBJECTIVES

- To study position control of DC servo motor under proportional control, PD and PID control
- To find the closed-loop transfer function and time-domain characteristics
- To study the control systems design according to the time-domain specifications

DISCUSSIONS OF FUNDAMENTALS

SERVO MODEL

The servo motor **voltage-to-position** transfer function is,

$$G(s) = \frac{\Theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)} \quad (1)$$

where $\Theta_m(s)$ is the Laplace transform of **motor position**, $V_m(s)$ is the Laplace transform of **applied motor voltage**, K is the **steady-state gain** or the **DC gain** of the model, and τ is the **time-constant** of the model.

Note: The numerical value of K and τ have been identified in **Lab 4**. Use an appropriate model from Lab 4.

PID CONTROL

A **PID controller** is a feedback control mechanism that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculate an error value $e(t)$ as the difference between reference input $r(t)$ and actual output $y(t)$:

$$e(t) = r(t) - y(t) \quad (2)$$

and generates a control signal based on **proportional**, **integral**, and **derivative** terms (denoted by P, I, and D. respectively). The complete PID control signal can be expressed mathematically as a sum of the three terms:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int e(\tau) d\tau + T_d \frac{de(t)}{dt} \right] \quad (3)$$

The PID controller can also be described by the transfer function:

$$G_c(s) = \frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s} + T_d s \right) \quad (4)$$

and the corresponding block diagram can be implemented as below:

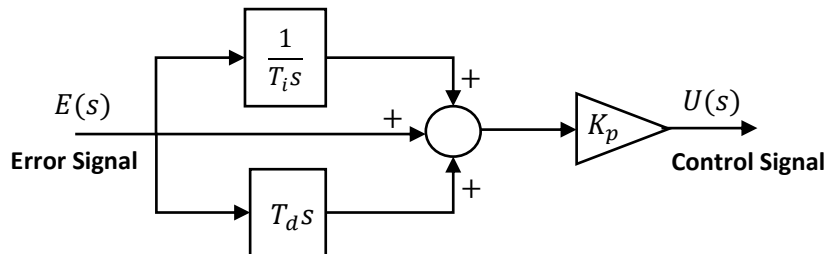


Figure 1: Block diagram of PID controller

The functionality of the PID controller can be summarized as follows: The **proportional** term is based on the present error, the **integral** term depends on past errors, and the **derivative** term is an anticipation of future errors.

SECOND-ORDER STEP RESPONSE

The **standard second-order** system transfer function has the form,

$$G(s) = \frac{Y(s)}{R(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (5)$$

where $Y(s)$ and $R(s)$ are the Laplace transform of the **output** and **input** signals, $\omega_n > 0$ is the **undamped natural frequency**, ζ is the **damping ratio**, and K is the **steady-state gain** or **DC gain**.

Consider an **underdamped** ($0 < \zeta < 1$) second-order system, subjected to a step input $r(t)$ with the amplitude of $R_0 = r_{max} - r_{min} = 1.5 - 0 = 1.5$,

$$r(t) = R_0 \rightarrow R(s) = \frac{R_0}{s}$$

The step response to this input is shown in Figure 2, where the step input is $r(t)$ and the output response is $y(t)$. Note the step input occurs at $t = t_0$.

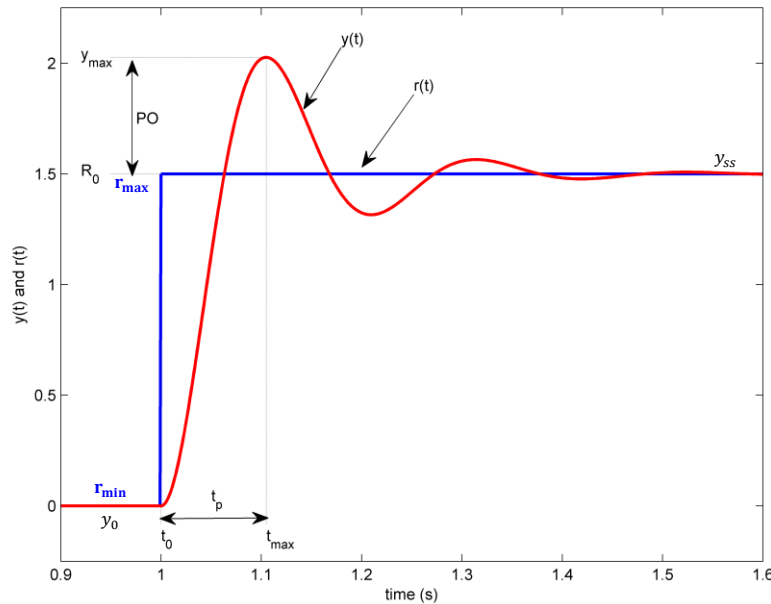


Figure 2: Step response of the second-order system

The **percentage of overshoot** and the **peak time** of this step response is obtained from the **graph** as follows,

$$\%O.S. = \frac{y_{max} - y_{ss}}{y_{ss} - y_0} \times 100\%, \quad t_p = t_{max} - t_0 \quad (6)$$

For an **underdamped second-order** system, the **overshoot** depends solely on the **damping ratio** ζ as,

$$M_p = y_{ss} e^{-\pi\zeta/\sqrt{1-\zeta^2}} \rightarrow \%O.S. = \frac{M_p}{y_{ss}} \times 100\% \quad (7)$$

The **peak time** depends on both the **damping ratio** ζ and the **undamped natural frequency** ω_n as,

$$t_p = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}} \quad (8)$$

Generally speaking, the **damping ratio** affects the **shape** of the response while the **natural frequency** affects the **speed** of the response

PART 1: Proportional Position Control

In this part, we will examine how a **proportional controller** can be used to control DC motor position. The block diagram model of the control system is shown in Figure 3,

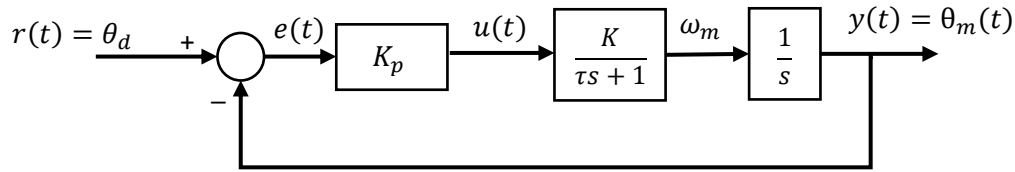


Figure 3: Proportional control of servo position

The proportional (P) control has the following structure,

$$u(t) = K_p(r(t) - y(t)) = K_p e(t) \quad (9)$$

where K_p is the proportional gain, $r(t) = \theta_d(t)$ is the set point or reference motor angle for the position control, $y(t) = \theta_m(t)$ is the measured load shaft angle, and $u(t) = v_m(t)$ is the control input (applied motor voltage).

The closed-loop **transfer function** $\frac{Y(s)}{R(s)}$ of the servo motor is determined as follows,

$$\frac{Y(s)}{R(s)} = \frac{\Theta_m(s)}{\Theta_d(s)} = \frac{K_p K}{\tau s^2 + s + K_p K} \quad (10)$$

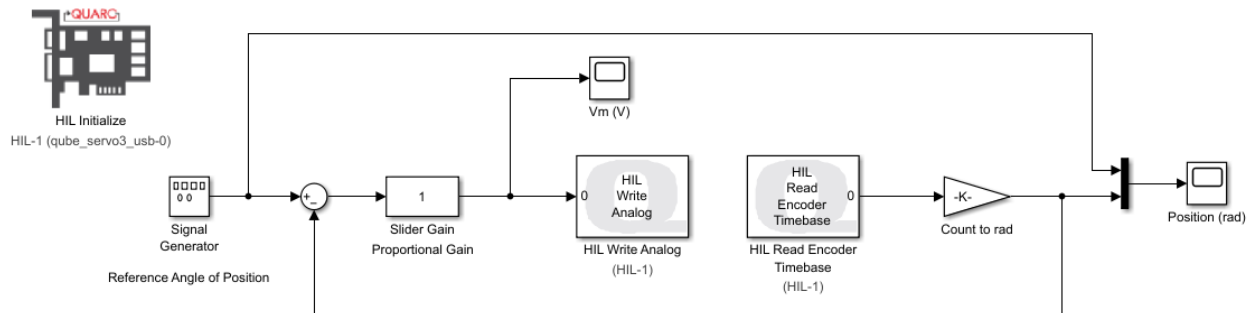
The **percentage of steady-state error** e_{ss} of the closed-loop system is defined as the difference between the reference input and the output signals after the system response has settled.

$$\%e_{ss} = \frac{\Delta r - \Delta y_{ss}}{\Delta r} \times 100\% \quad (11)$$

where $\Delta r = r_{max} - r_{min}$ is the total change of the input and $\Delta y_{ss} = y_{ss} - y_0$ is the total change of the output at the steady-state value.

1. Create the following **Simulink** diagram to implement the **proportional position control** of the servo motor.

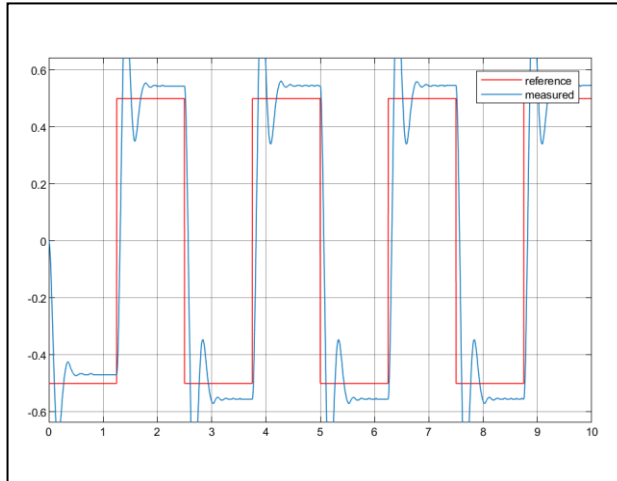
Remark: We are using a conversion gain of $2\pi/2048$ to convert the number of counts to radian at the encoder output.



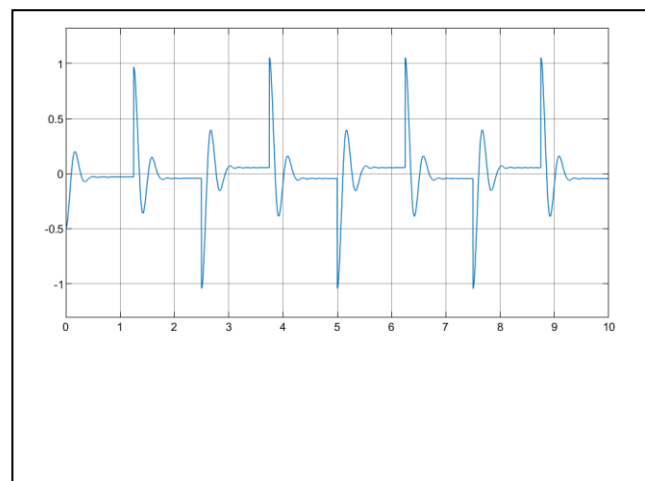
2. Open the **HIL Initialize** block and set the **Board type** to **qube_servo3_usb**.
3. Click on the **Model Settings** icon in the **MODELING** tab to open the **Configuration Parameters** window. Click on the **Solver** drop down menu and select the **Type of Fixed step** and set the **Solver** to **ode1(Euler)** solver. Then click **OK**.

- Use a **Slider Gain** block to model the proportional control gain. Set the **Slider Gain** to change from **Low = 1** to **High = 5** and set the value to $K_p = 1$. Click **OK**.
- Set the **Signal Generator** to generate a **square** waveform with an **amplitude** of **0.5rad** and **frequency** of **0.4Hz**.
- Save** the Simulink file as **Lab7.slx**. **Run** your code for **10 seconds**.
- Double-click and open the **Scope** blocks. Provide the plots with **white background** below:

Reference Input & Motor Position (rad)



Applied Voltage to Motor (V)



- Use the **Scope Cursor Measurement tools** to measure the **percentage of overshoot** ($\%O.S.$) and **peak-time** (t_p) using the provided method in **Eqn. (6)**. Insert the **measured** values in **Table 1**.

$$\%O.S. = \frac{y_{max} - y_{ss}}{y_{ss} - y_0} \times 100\% = \frac{0.888 - 0.544}{0.544 - -0.56} * 100 = 31.2\%$$

$$t_p = t_{max} - t_0 = 3.921 - 3.746 = 0.18 \text{ s}$$

- Is there a steady-state error? If so, measure the **percentage of steady-state error** ($\%e_{ss}$). Insert the value in **Table 1**. Explain why the steady-state error is **nonzero**.

$$\%e_{ss} = \frac{\Delta r - \Delta y_{ss}}{\Delta r} \times 100\% = \frac{1 - 0.98}{1} * 100 = 2\%$$

The steady state error is non zero because there is no feed back system where it can adjust the input and reduce that error.

Table 1: Control System with Proportional Control

Applied Input	Percent Overshoot (%O.S.)	Peak-Time (sec)	Percent of Steady-State Error (%e _{ss})
± 0.5 rad	31.2%	0.18 s	2%

10. Set the **Stop time** of Simulation to **inf**. **Run** the code.
11. Open the **Slider Gain** block. Vary K_p between **1** and **5** and observe the output signal. Explain how increasing the proportional gain affects the servo position control response in terms of the **overshoot**, **peak-time** and the **steady-state error**.

When increasing the proportional gain, it affects the servo position where the rise time decreases, overshoot increases, peak time decreases, steady state error decreases and settling time increases

12. **Stop** the code.
13. Set the **Slider Gain** to change from **Low = 0.1** to **High = 1** and set the value to $K_p = 1$. Click **OK**.
14. Set the **Stop time** of Simulation to **inf**. **Run** the code.
15. Open the **Slider Gain** block. Vary K_p between **1** and **0.1**. Explain how decreasing the proportional gain affects the servo position control response in terms of the **overshoot**, **peak-time** and **steady-state error**.

When decreasing the proportional gain, the servo position control response is affected whereby the rise time increases, overshoot decreases, peak time decreases, sse increases and settling time decreases.

16. Describe the response for very small gains of K_p (i.e. below **0.2**). Does the system respond as expected? Explain the effect of the **Static friction** in the steady-state error of servo position control.

The system does not respond as expected as it is not able to overcome the forces of static friction to create movement. Hence there is a dead zone that exists at below 0.2 Kp.

PART 2: Proportional – Derivative (PD) Position Control

In this part, we will implement a **PD control** to **decrease the overshoot**. The block diagram model of PD control system is shown in Figure 4,

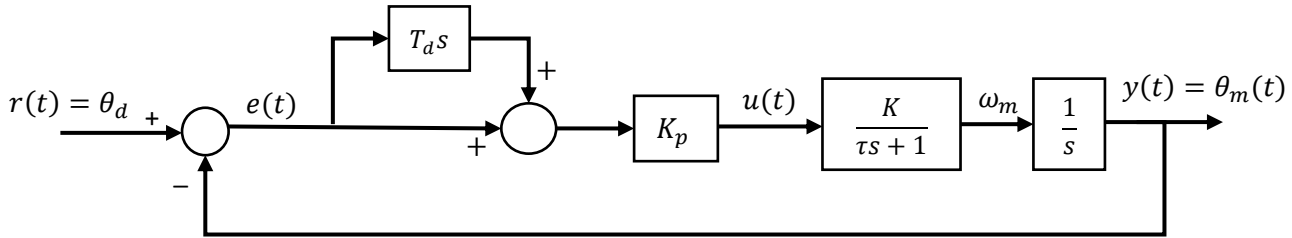


Figure 4: Proportional-Derivative (PD) control of servo position

The PD control has the following structure,

$$u(t) = K_p \left[e(t) + T_d \frac{de(t)}{dt} \right] \rightarrow U(s) = K_p (1 + T_d s) E(s) \quad (12)$$

where K_p is the proportional gain, T_d is the derivative time-constant, $e(t)$ is the error signal, and $u(t) = v_m(t)$ is the control input (applied motor voltage).

The closed-loop **transfer function** $\frac{Y(s)}{R(s)}$ of the servo motor is determined as follows,

$$\frac{Y(s)}{R(s)} = \frac{\Theta_m(s)}{\Theta_d(s)} = \frac{K_p K (1 + T_d s)}{\tau s^2 + (1 + K_p K T_d) s + K_p K} = \frac{K_p K (1 + T_d s) / \tau}{s^2 + \left((1 + K_p K T_d) / \tau \right) s + K_p K / \tau} \quad (13)$$

This is a **second-order transfer function with a real zero**. Recall the **standard second-order** transfer function,

$$\frac{Y(s)}{R(s)} = \frac{K_{dc} \omega_n^2}{s^2 + 2\zeta \omega_n s + \omega_n^2} \quad (14)$$

We can find the proportional gain K_p and the derivative time constant T_d in terms of the ζ and ω_n as below:

$$K_p = \frac{\omega_n^2 \tau}{K}, \quad T_d = \frac{2\zeta \omega_n \tau - 1}{K_p K} \quad (15)$$

The following steps show you how to find the PD controller parameters K_p and T_d and implement the controller. The goal is to design a PD controller to a step response with **peak-time** less than **0.2sec** and a **percent overshoot** less than **2.5%**.

First, use **Eqn. (16)** to determine the **natural frequency** ω_n and **damping ratio** ζ needed to match the desired percent overshoot and peak time specifications.

$$\zeta = -\frac{\ln(O.S.)}{\sqrt{\pi^2 + (\ln(O.S.))^2}} \quad \text{and} \quad \omega_n = \frac{\pi}{t_p \sqrt{1 - \zeta^2}} \quad (16)$$

Next, use **Eqn. (15)** to calculate the PD controller parameters K_p and T_d based on the obtained **natural frequency** ω_n and **damping ratio** ζ .

Remark: Use the DC motor model parameters K and τ based on an appropriate model from **Lab 4**.

17. Run the following code in **MATLAB** to calculate the damping ratio, natural frequency and the PD controller parameters. **Assign the K and τ values based on the DC motor model from Lab 4.**

```
K = % DC-gain
T = % Time constant (sec)
OS = 0.025; % Desired overshoot
tp = 0.2; % Desired peak-time (sec)

z = -log(OS)/sqrt(pi^2+log(OS)^2) % Damping ratio
wn = pi/(tp*sqrt(1-z^2)) % Undamped Natural Frequency (rad/sec)

Kp = (T*wn^2)/K % Proportional Gain
Td = (2*z*wn*T-1)/(K*Kp) % Derivative Time Constant
```

Provide the code execution from **MATLAB Command** window and complete **Table 2**:

```
>> K = 29
K =
    29

>> T = 0.096
T =
    0.0960

>> OS = 0.025; % Desired overshoot
>> tp = 0.2; % Desired peak-time (sec)
>> z = -log(OS)/sqrt(pi^2+log(OS)^2) % Damping ratio|
z =
    0.7613

>> wn = pi/(tp*sqrt(1-z^2)) % Undamped Natural Frequency (rad/sec)
wn =
    24.2268

>> Kp = (T*wn^2)/K % Proportional Gain
Kp =
    1.9430

>> Td = (2*z*wn*T-1)/(K*Kp) % Derivative Time Constant
Td =
    0.0451
```

Table 2: PD Controller Parameters

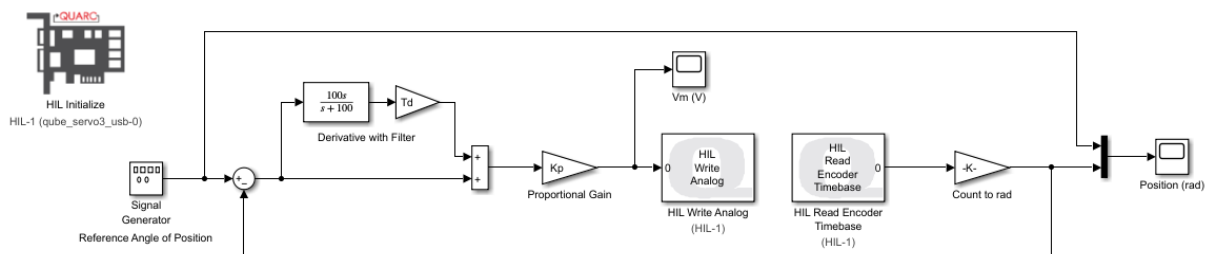
Damping Ratio (ζ)	Natural Frequency (ω_n)	Proportional Gain (K_p)	Derivative Time Constant (T_d)
0.7613	24.2268	1.9430	0.0451

18. Provide the transfer function of the designed PD controller,

$$G_c(s) = K_p(1 + T_d s) = 1.94(1 + 0.045s)$$

19. Modify the **Simulink** diagram as shown below to implement the PD controller.

Remark: Differentiating a noisy measured signal (i.e. in the derivative control) can produce a control signal with high frequency components that yields an undesirable response and, over time, may even damage the actuator (e.g. DC motor). We are using a derivative with a low-pass filter model to implement the derivative term to reduce the high-frequency measurement noise.

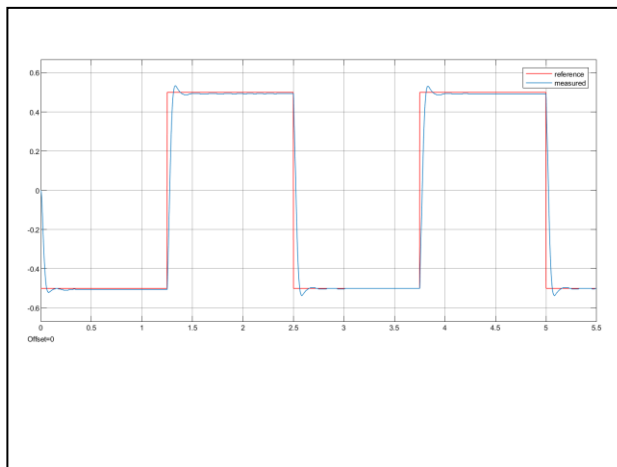


20. Set the PD controller parameters K_p and T_d based on the designed values in **Step 18**.

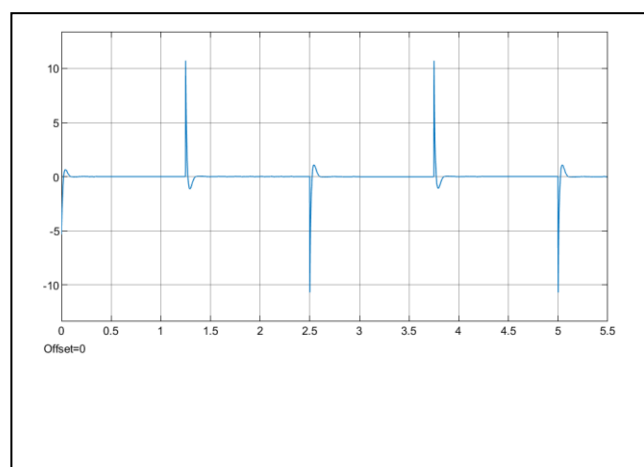
21. **Save** the Simulink file. **Run** your code for **5.5 seconds**.

22. Double-click and open the **Scope** blocks. Provide the plots with **white background** below:

Reference Input & Motor Position (rad)



Applied Voltage to Motor (V)



23. Use the **Scope Cursor Measurement tools** to measure the **percentage of overshoot** (%*O.S.*), **peak-time** (*t_p*) and the **percent of steady-state error** (%*e_{ss}*). Insert the values in **Table 3**. Do they match the desired percent overshoot and peak-time specifications?

$$\%O.S. = \frac{y_{max} - y_{ss}}{y_{ss} - y_0} \times 100\% = \frac{0.533 - 0.494}{0.494 - -0.506} * 100 = 3.9\%$$

$$t_p = t_{max} - t_0 = 1.331 - 1.249 = 0.082$$

$$\%e_{ss} = \frac{\Delta r - \Delta y_{ss}}{\Delta r} \times 100\% = \frac{1 - 0.998}{1} \times 100 = 0.2\%$$

They do not match the desired percent overshoot specification but it does match peak time being less than 0.2 seconds.

Table 3: Control System with PD Controller

Applied Input	Percent Overshoot (%O.S.)	Peak-Time (sec)	Percent of Steady-State Error (%e _{ss})
± 0.5 rad	3.9%	0.082	0.2%

PART 3: Proportional – Integral – Derivative (PID) Position Control

The **steady-state error** can be eliminated by adding an **integrator term** to the PD controller.

24. Run the following code in **MATLAB** to calculate the required **integral time constant** *T_i* based on the location of the dominant poles of the closed-loop system with PD controller in **Eqn. (13)**.

$$T_i = \frac{5}{|\text{Re}\{p_{cl}\}|}$$

```
sys_PD = tf([Kp*K*Td Kp*K],[T 1+Kp*K*Td Kp*K])
p = pole(sys_PD)
Ti = 5/abs(real(p(1)))
```

```
% Closed-loop system with PD
% Closed-loop system poles
% Integral Time Constant
```

Provide the code execution from **MATLAB Command** window.

```
sys_PD =

      2.541 s + 56.35
      -----
      0.096 s^2 + 3.541 s + 56.35

Continuous-time transfer function.

p =

-18.4444 +15.7080i
-18.4444 -15.7080i

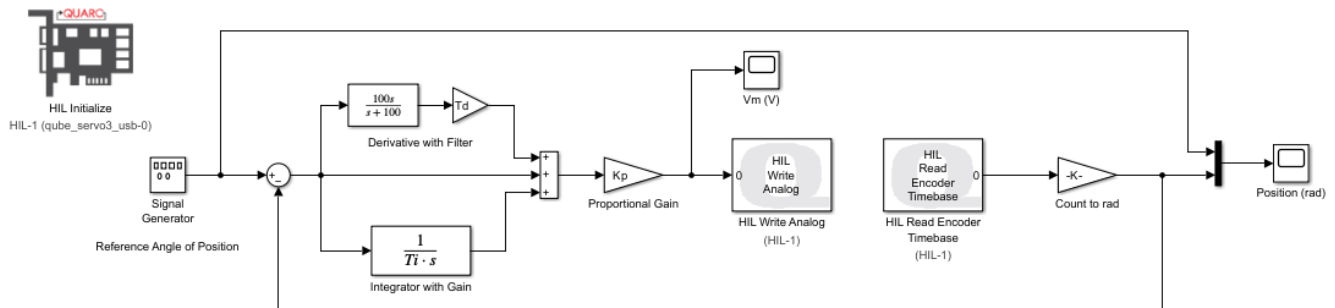
Ti =

0.2711
```

25. Provide the transfer function of the obtained **PID controller**,

$$G_c(s) = K_p \left(1 + T_d s + \frac{1}{T_i s} \right) = 1.94 \left(1 + 0.045s + \frac{1}{0.2711s} \right)$$

26. Modify the **Simulink** diagram as shown below to implement the **PID controller**.



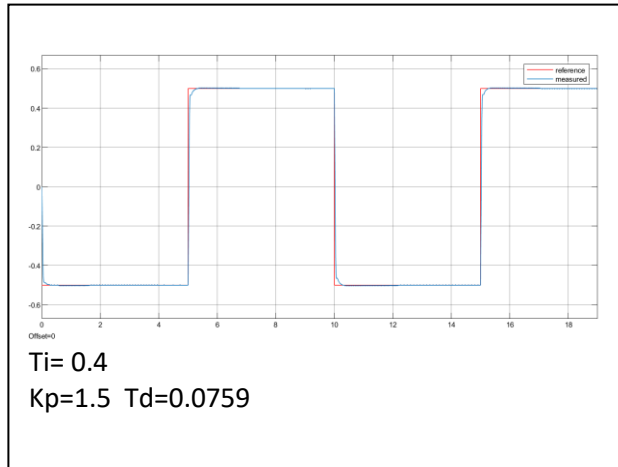
27. Set the PID controller parameters K_p , T_i and T_d based on the designed values in **Step 25**.

28. Set the **Signal Generator** to generate a **square** waveform with an **amplitude** of **0.5rad** and **frequency** of **0.1Hz**.

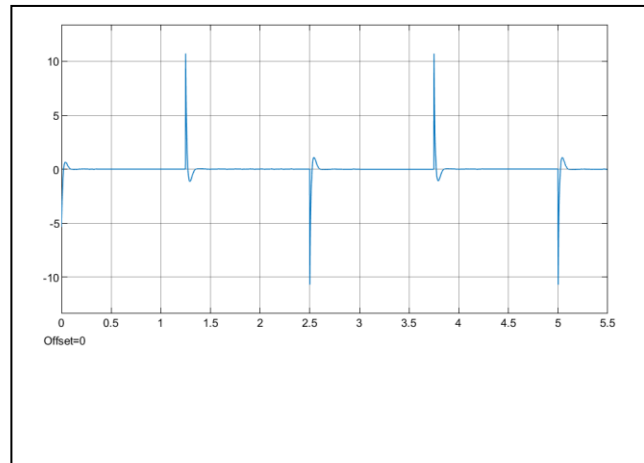
29. **Save** the Simulink file. **Run** your code for **19 seconds**.

30. Double-click and open the **Scope** blocks. Provide the plots below:

Reference Input & Motor Position (rad)



Applied Voltage to Motor (V)



31. Use the **Scope Cursor Measurement tools** and measure the **percent of steady-state error** ($\%e_{ss}$). Does the PID controller eliminate the steady-state error?

$$\%e_{ss} = \frac{\Delta r - \Delta y_{ss}}{\Delta r} \times 100\% = \frac{1-1}{1} * 100 = 0\%$$

Yes, the PID controller eliminates the steady-state error.

32. **Stop** the code.

33. **Power OFF** the QUBE-Servo 3 system.