LAB 6: DC MOTOR SPEED CONTROL

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

Lab/Tutorial Report No.	**Lab 6**
Report Title	**DC Motor Speed Control**
Section No.	ONA
Group No.	2
Submission Date	02-23-2025
Due Date	02-23-2025

Student Name	**Signature***	**Total Mark**
Joshua Mongal	[illegible]	**/50**
Mikaeel Khanzada	Mikaeel Khanzada	**/50**
Michael McCorkell	Michael McCorkell	**/50**
		/50

LAB 6 Grading Sheet

Student Name: Joshua Mongal	**Student Name:** Mikaeel Khanzada
Student Name: Michael McCorkell	**Student Name:**

Part 1: Proportional Speed Control	**/20**
Part 2: Proportional-Integral Speed Control	**/20**
Part 3: Load Disturbance Effect	**/5**
General Formatting: **Clarity, Writing style, Grammar, Spelling, Layout of the report**	**/5**
Total Mark	**/50**

LAB 6: DC MOTOR SPEED CONTROL

OBJECTIVES

- To study speed control of DC servo motor under proportional control and PI control
- To find the closed-loop transfer function and time-domain characteristics
- To study the effect of load disturbance under proportional control and PI control

DISCUSSIONS OF FUNDAMENTALS

SERVO MODEL

The **QUBE-Servo 3** voltage-to-velocity transfer function is,

$$G(s) = \frac{\Omega_m(s)}{V_m(s)} = \frac{K}{\tau s + 1} \tag{1}$$

where Ω_m is the ***motor velocity***, V_m is the ***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 determined 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 calculates an error value $e(t)$ as the difference between reference input $r(t)$ and actual output $y(t)$:

$$e(t) = r(t) - y(t) \tag{2}$$

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:

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

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

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

and the corresponding block diagram can be implemented as below:

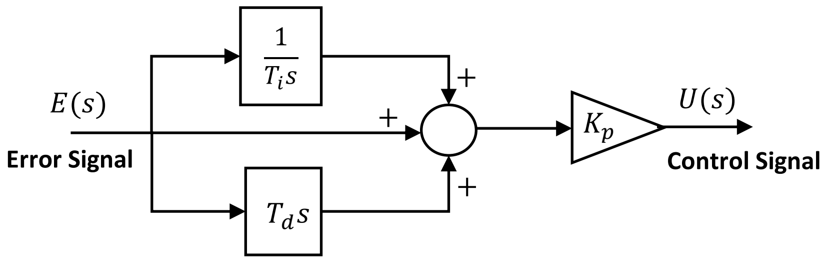


Figure 1: Block diagram of PID controller

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.

PART 1: Proportional Speed Control

In this part, we will examine how a **proportional controller** can be used to control DC motor speed. The block diagram model of control system is shown in Figure 2,

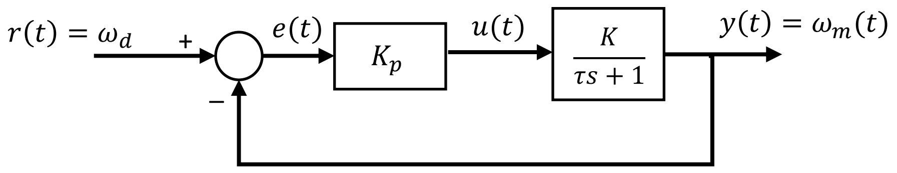


Figure 2: Proportional control of servo speed

The proportional (P) control has the following structure,

$$u(t) = K_p\big(r(t) - y(t)\big) = K_p\, e(t) \tag{5}$$

where K_p is the proportional gain, $r(t) = \omega_d$ is the reference angular velocity, $y(t) = \omega_m(t)$ is the measured angular velocity, and $u(t) = v_m(t)$ is the control input (applied motor voltage).

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

$$\frac{Y(s)}{R(s)} = \frac{K_p K}{\tau s + 1 + K_p K} \tag{6}$$

The **transfer function** $\boldsymbol{\frac{E(s)}{R(s)}}$ that represents the dynamics between a desired angular velocity and the error signal.

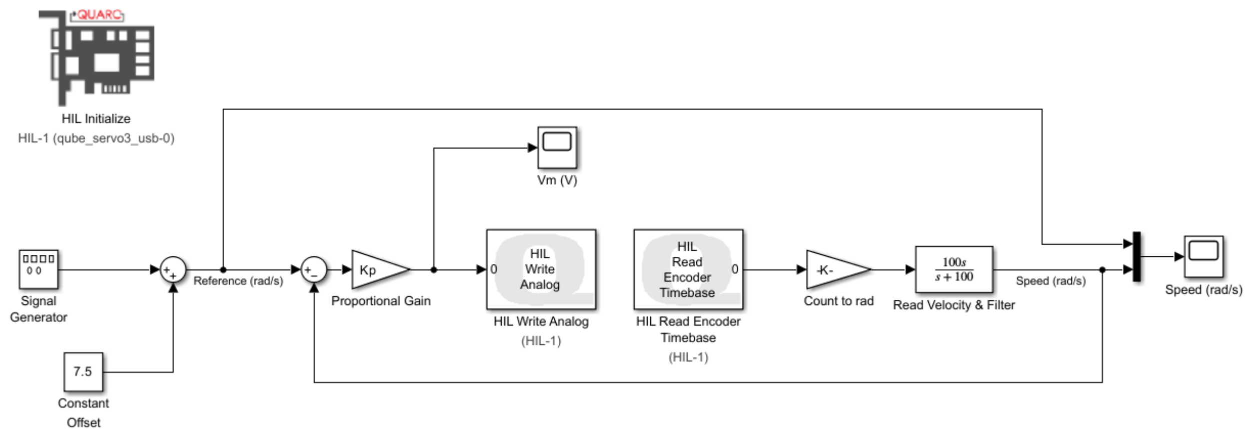
$$\frac{E(s)}{R(s)} = \frac{\tau s + 1}{\tau s + 1 + K_p K} \tag{7}$$

The **steady-state error $\boldsymbol{e_{ss}}$** of the closed-loop system for the reference step input $R(s) = \frac{R}{s}$ is determined using the ***Final-Value Theorem*** as below,

$$e_{ss} = \lim_{s\to 0} sE(s) = \lim_{s\to 0} s\left(\frac{\tau s + 1}{\tau s + 1 + K_p K}\right)\left(\frac{R}{s}\right) = \frac{R}{1 + K_p K} \tag{8}$$

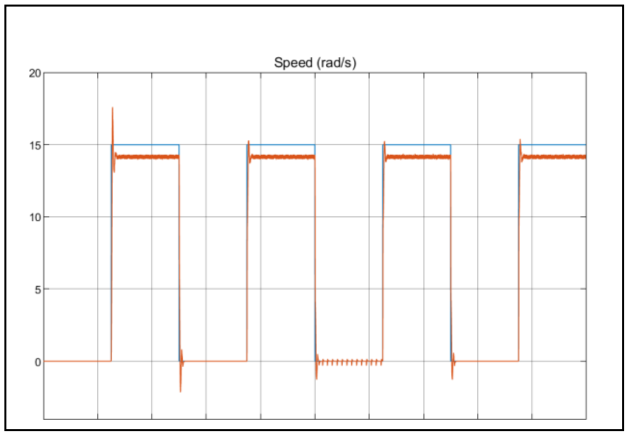
1. Create the following **Simulink** diagram to implement the **proportional speed control** of servo motor.

   ***Remark**: We are using a conversion gain of $2\pi/2048$ to convert the number of counts to radian at the encoder output, and a differentiator s cascade with a low-pass filter $100/(s+100)$ to obtain the speed of the motor and filter the quantization noise.*

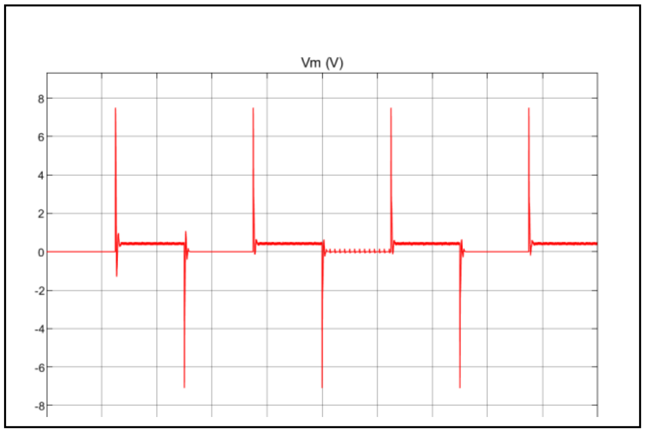


2. Open the **HIL Initialize** block and set the **Board type** to **qube_servo3_usb**.
3. Click on the **Model Settings** icon in the **MODELING** tab to open the **Configuration Parameters** window. Click on the **Solver** drop down menu and select the **Type** of **Fixed step** and set the **Solver** to **ode1** solver. Then click **OK**.
4. Set the **Signal Generator** to generate a **square** waveform with an **amplitude** of **7.5rad/s** and **frequency** of **0.4Hz**.
5. Set the **Constant** block to **7.5**, to create an offset of **7.5** to ensure that reference command is applied from **0** to **15rad/s**.
6. Set the proportional control gain to $\boldsymbol{K_p = 0.5}$.
7. **Save** the Simulink file as **Lab6.slx**. **Run** your code for **10 seconds**.
8. Double-click and open the **Scope** blocks. Provide the plots with **white background** below:

Reference Input & Motor Speed (rad/s)



Applied Voltage to Motor (V)



9. Use Scope *Cursor Measurement tools* to measure the **steady-state error** ($\boldsymbol{e_{ss} = r - y_{ss}}$). Insert the **measured** value of $\boldsymbol{e_{ss}}$ in **Table 1**.

Table 1

Applied Input	Steady-State Error (Measurement)	Steady-State Error (Calculation)	DC Motor Model from Lab 4
0 to 15 rad/s	15-14.154= 0.846	1.14	$\frac{24.25}{0.102s+1}$

10. Calculate the **steady-state error** using **Eqn. (8)** if the proportional gain is $\boldsymbol{K_p = 0.5}$, the step amplitude is $\boldsymbol{R = 15\ rad/s}$. Set the $\boldsymbol{K}$ value based on the DC Motor model obtained from **Lab 4**. Show your calculations below,

$$e_{ss} = \frac{R}{1+K_pK} = \quad = \frac{15}{13.125} = 1.14$$

11. Does the **calculated** steady-state error in **Step 9** match with the **measured** value in **Step 8**? If they do not match, give a reason why there would be a difference.

No, the calculated steady state error does not match with the measured value in step 8 because the calculated steady state error s based off a model where there were external disturbances present like friction or vibration while the measured state was based on a Simulink model where there are less to no disturbances.

12. Using the **Eqn. (8)** determine the **required proportional gain K_p** to decrease the steady-state error to the **half of the calculated value in Step 10**. Show your calculations and the K_p value below,

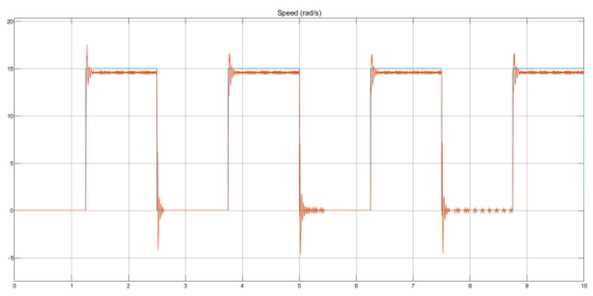
$$\frac{R}{1+K_pK} = e_{ss} \;=>\; \frac{15}{1+K_p * 24.25} = 0.57 \;=>\; \frac{1+K_p * 24.25}{15} = 1.754$$
$$=>\; 1+K_p * 24.25 = 26.32$$
$$=> K_p = \frac{25.32}{24.25} = 1.04$$

13. Change the proportional gain K_p in your Simulink diagram to the calculated value in **Step 12**. Run the code and validate the results by measuring the steady-state error from the graph. Show your calculations.

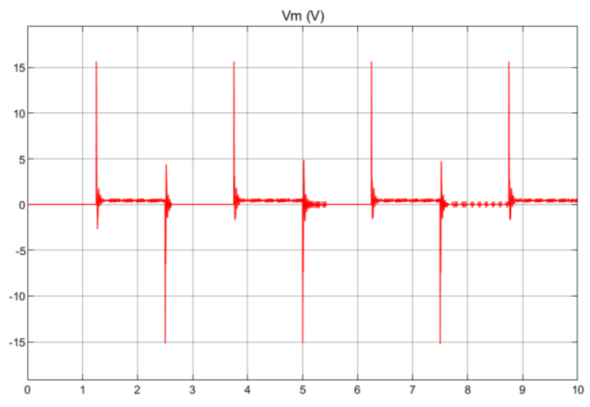
$e_{ss} = r - y_{ss} =$ 15-14.577 = 0.423

Provide the graphs with **white background** below,

Reference Input & Motor Speed (rad/s)



Applied Voltage to Motor (V)



PART 2: Proportional – Integral (PI) Speed Control

In this part, we will implement a **PI control** to **eliminate the steady-state error.** The block diagram model of PI control system is shown in Figure 3,

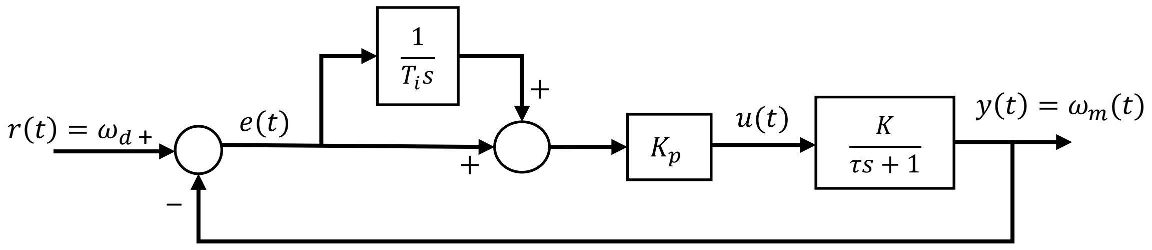


Figure 3: Proportional-Integral (PI) control of servo speed

The PI control has the following structure,

$$u(t) = K_p\left[e(t) + \frac{1}{T_i}\int e(\tau)d\tau\right] \quad \rightarrow \quad U(s) = K_p\left(1 + \frac{1}{T_i s}\right)E(s) \tag{9}$$

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

The following steps show you how to find the PI controller parameters K_p and T_i and implement the controller.

14. Set the K_p to the calculated value in **Step 12**.
15. Use **Eqn. (6)** to find the closed-loop transfer function $\frac{Y(s)}{R(s)}$ of the system with proportional controller, if K_p is the calculated value in **Step 12**, set the K, and τ values based on the obtained DC Motor model from **Lab 4**. Then find the **closed-loop pole, p_{cl}** location. Provide the transfer function model, and the closed-loop pole p_{cl} values below,

$$\frac{Y(s)}{R(s)} = \frac{K_p K}{\tau s + 1 + K_p K} = \frac{1.04 * 24.25}{0.102s + 1 + 25.22} = \frac{25.22}{0.102s + 26.22}$$

Poles is s= -257.0588

16. Determine the parameter $\boldsymbol{T_i}$ based on the <u>dominant closed-loop poles</u> p_{cl} location using the following equation. Show your calculations.

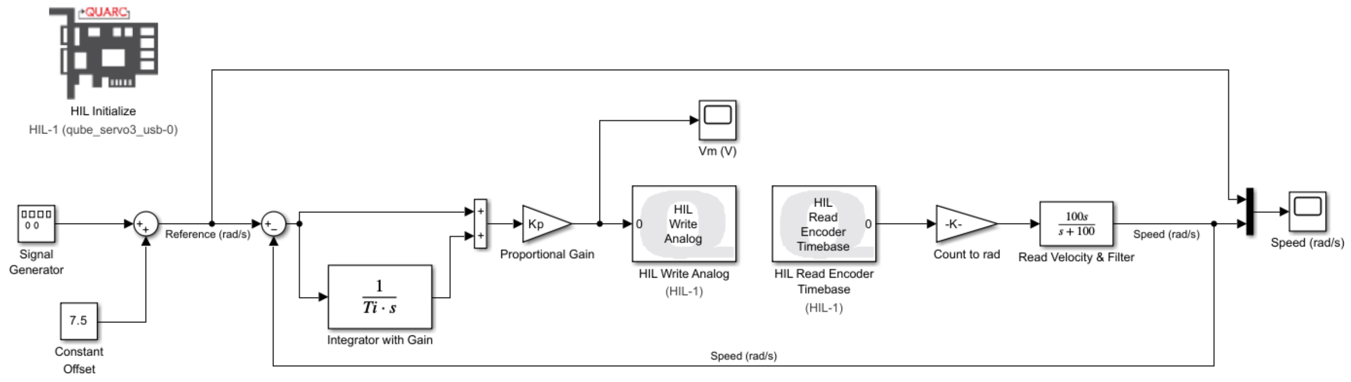
$$\boldsymbol{T_i} = \frac{\mathbf{10}}{|\mathbf{Re}\{\boldsymbol{p_{cl}}\}|}$$

$$\boldsymbol{T_i} = \frac{\mathbf{10}}{|\mathbf{Re}\{\boldsymbol{p_{cl}}\}|} = \frac{10}{|-257.06|} = 0.039$$

17. Provide the transfer function of the designed PI controller,

$$\boldsymbol{G_c(s) = K_p\left(1+\frac{1}{T_i s}\right) = \quad 1.04\left(1+\frac{1}{0.039s}\right)}$$

18. Modify the **Simulink** diagram as shown below to implement the PI controller.

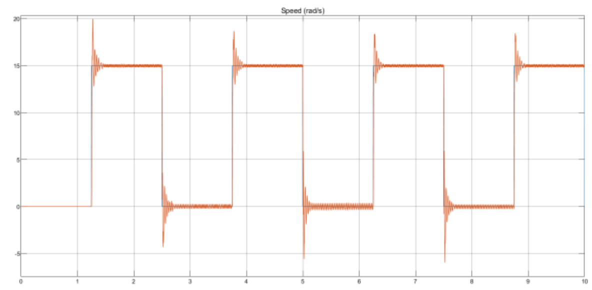


19. Set the PI controller parameters K_p and T_i based on the designed values in **Step 16**.

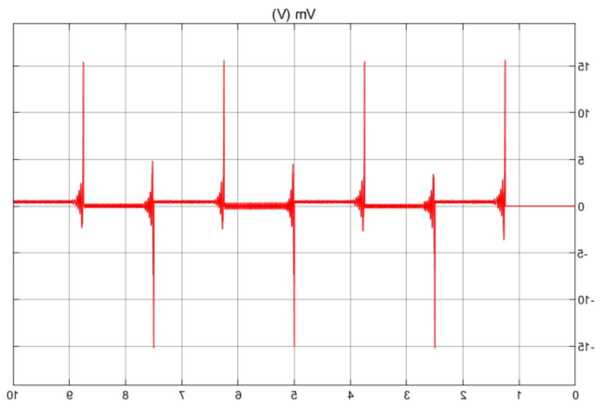
20. **Save** the Simulink file. **Run** your code for **10 seconds**.

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

Reference Input & Motor Speed (rad/s)



Applied Voltage to Motor (V)



22. Using the **Scope** *Cursor Measurement tools* to measure the **steady-state error** $(\boldsymbol{e_{ss} = r - y_{ss}})$. Does the PI controller eliminate the steady-state error?

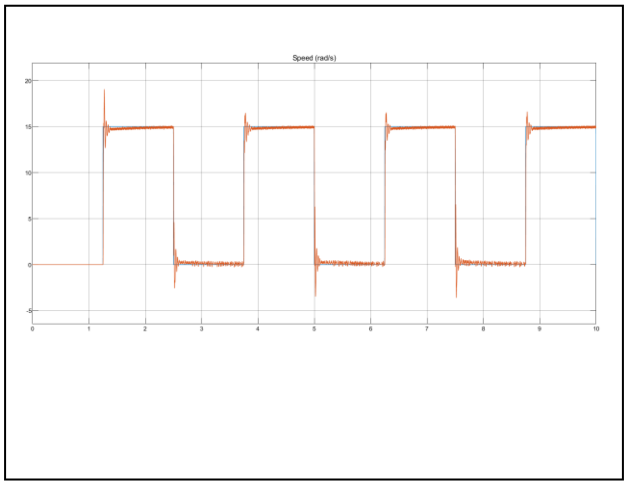
$e_{ss} = r - y_{ss} =$ 15-14.972 =0.028

Yes, since the measured steady state error value is very small in comparison to the previous errors, the PI controller does eliminate the steady state error.

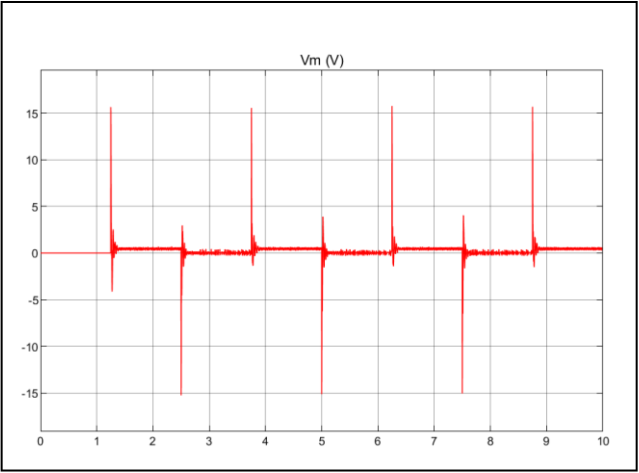
23. Fine tune (**if required**) the PI controller parameters (K_p and T_i) to achieve a better performance in terms of settling time and the overshoot. Provide your **final PI controller** and the **Scope** plots.

$G_c(s) = K_p\left(1+\frac{1}{T_i s}\right) = \mathbf{1.04}\left(\mathbf{1}+\frac{\mathbf{1}}{0.78s}\right)$

Reference Input & Motor Speed (rad/s)



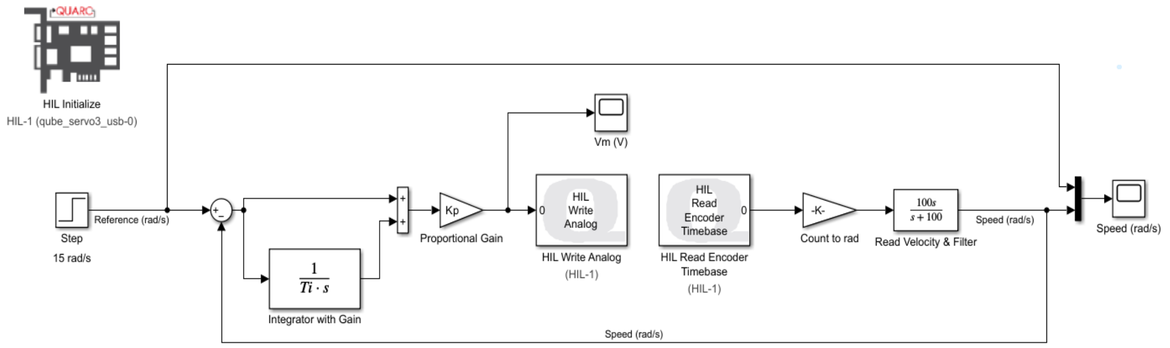
Applied Voltage to Motor (V)



PART 3: Load Disturbance Effect

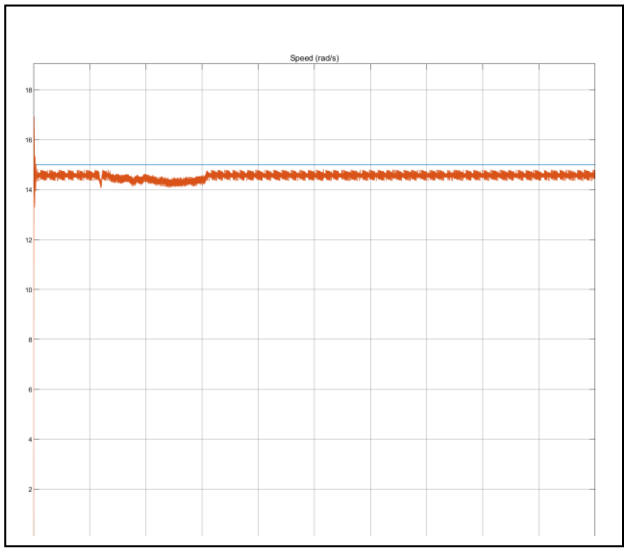
The effect of a disturbance can be replicated by applying torque to the inertia disk. A disturbance can be made by gently touching the inertia disk as it is spinning. This will not be a constant torque but is useful for quantitative analysis.

24. Modify the **Simulink** diagram as shown below by replacing the **Signal Generator** and **Constant Offset** blocks with a **Step** block.

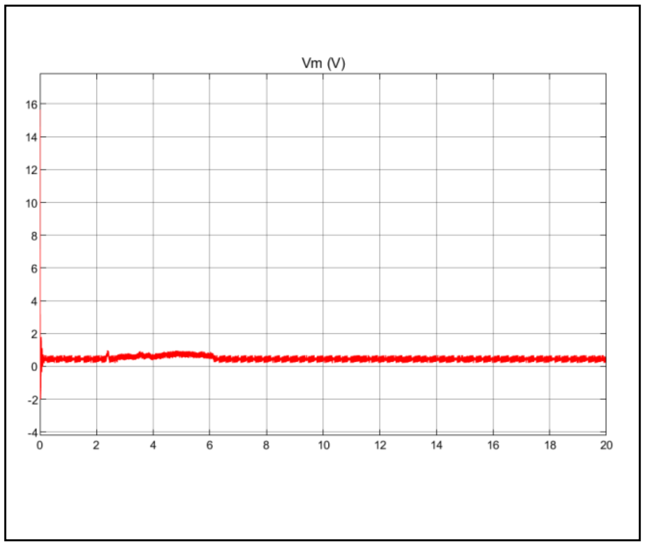


25. Set the **Step** input **Step time** to **0** and the **Final value** to **15rad/s**.
26. **<u>Disconnect</u> the output of the integral block**. This makes the controller only a **proportional control**.
27. **Save** the Simulink file. **Run** your code for **20 seconds**.
28. **<u>Gently touch</u>** the inertia disk as it is spinning for about **5 seconds** and release. This will decrease the speed of the rotation. (**do not stop the disk**)
29. Double-click and open the **Scope** blocks. Provide the plots with **white background** below:

Reference Input & Motor Speed (rad/s)



Applied Voltage to Motor (V)

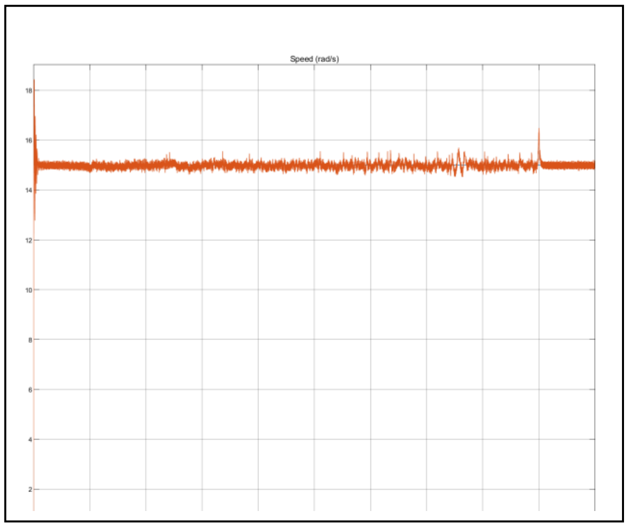


30. Explain the effect of load disturbance in **proportional control**. Does the proportional controller successfully reject the load disturbance effect?

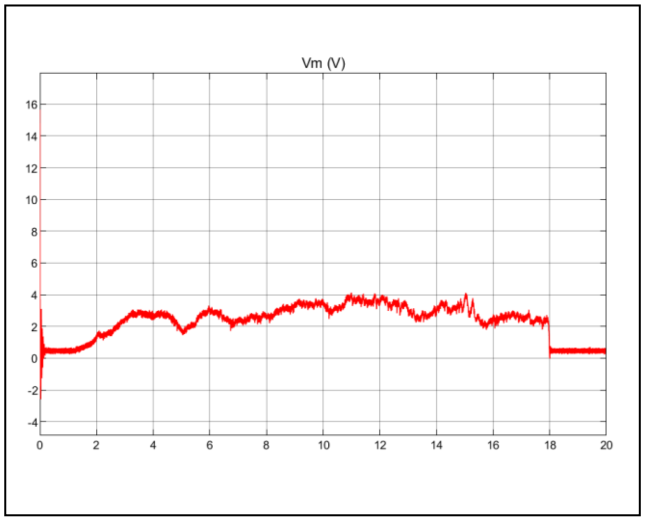
The effect of the load disturbance in proportional control is that it causes an immediate drop in speed and the system does not fully make up for the disturbance since it does not keep record of all the previous errors. So no, it doesn't reject the load disturbance effect successfully.

31. **<u>Connect</u> the output of the integral block**. This makes the controller a **PI control**.
32. **Save** the Simulink file. **Run** your code for **20 seconds**.
33. **<u>Gently touch</u>** the inertia disk as it is spinning for about **5 seconds** and release. This will decrease the speed of the rotation.
34. Double-click and open the **Scope** blocks. Provide the plots with white background below:

Reference Input & Motor Speed (rad/s)



Applied Voltage to Motor (V)



35. Explain the effect of load disturbance in **PI control**. Does the PI controller successfully reject the load disturbance effect?

The effect of the load disturbance in PI control is that maintains the speed and the system does fully make up for the disturbance since it keeps record of all the previous errors. So no, it does reject the load disturbance effect successfully.

36. **Stop** the code.

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