

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS
ENGINEERING
FACULTY OF ENGINEERING
NATIONAL DEFENCE UNIVERSITY OF MALAYSIA**

LAB REPORT 2

**EEE 3351 – ENGINEERING LAB V
SEMESTER 1 ACADEMIC SESSION 2024/2025
PROGRAMME ZK23**

**DESIGN AND ANALYZE A FEEDBACK CONTROLLER TO IMPROVE
LIFT DOOR**

LECTERUR

- 1. Prof. Madya Ts. Gs. Dr. Syed Mohd Fairuz Bin Syed Mohd Dardin**
- 2. Dr. Siti Noormiza Binti Makhtar**
- 3. Dr. Chew Sue Ping**
- 4. Dr. Elya Binti Mohd Nor**

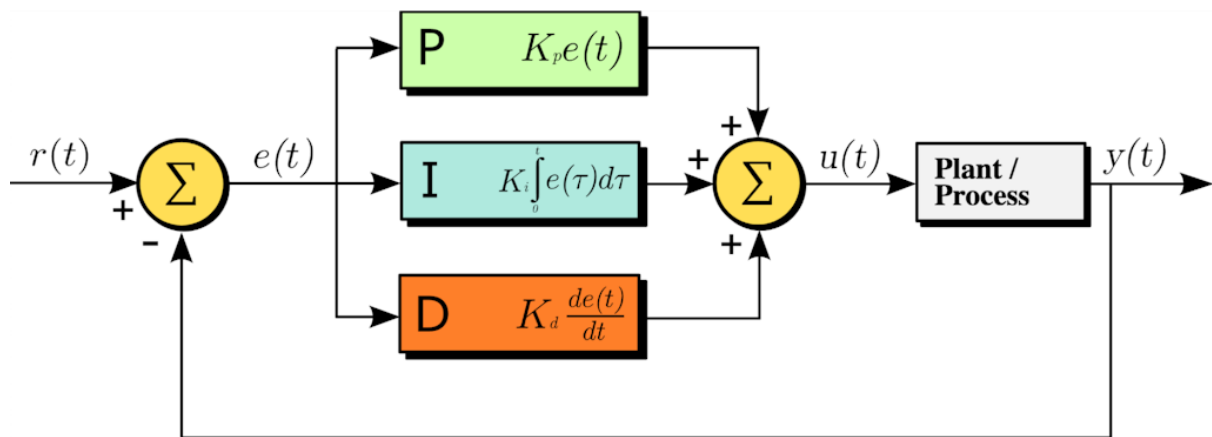
NAME	NO MATRIC
LEE KAH CHUN	2220847
KARANRAJ A/L PARAMASIVAM	2220136
AINA SYAZANA BINTI ANUR	2190204

INTRODUCTION

Managing lift door mechanisms is a crucial aspect of the design and functioning of contemporary elevator systems. A key element that powers the lift door system is the DC motor, responsible for regulating the door's speed and motion. The motor needs precise regulation to guarantee seamless functioning, safety, and effectiveness. A major difficulty in creating such a system is ensuring accurate regulation of the motor's speed, especially under different load conditions (i.e., the door's weight) and attaining the intended opening and closing durations of the lift door.

In this lab, we will investigate the design and execution of a PID (Proportional-Integral-Derivative) controller to manage the speed of the DC motor driving the lift door. The aim is to manage the motor speed to effectively control the lift door's movement. In particular, we will model the dynamics of the lift door, taking into account the influence of the door's weight and the motor shaft's angular velocity. The controller's effectiveness will be evaluated in two different scenarios:

1. Lacking PID Control (Open-loop): This describes how the system operates when feedback control is not utilized. The motor functions entirely on its dynamics, potentially causing delayed responses, overshoot, or errors at steady state.
2. Using PID Control: A PID controller is utilized to modify the motor's speed instantaneously depending on the discrepancy between the intended and actual position of the door. The controller seeks to reduce the overshoot, shorten the settling time, and eradicate steady-state error.



Overview of the System

The system is made up of:

1. DC Motor: The motor operates the lift door, and its velocity is controlled by the input voltage. The motor is represented as a second-order system, with the angular velocity of the motor as the output.
2. Lift Door: The position of the lift door is directly connected to the angular movement of the motor's shaft. The door's weight contributes to the motor's load, influencing the dynamics of the system. The door's weight creates a resistive force that the motor must counter in order to open the door.
3. Motor Speed Regulation: The angular velocity of the motor determines the position of the door, thereby affecting the speed at which the door opens and closes based on the motor's reaction to input control signals.

System Dynamics

1. Angular Velocity of the Motor: The motor's pace, quantified in radians per second, controls how quickly the door opens or closes.
2. Weight of the Lift Door: The door's heaviness serves as a burden on the motor. The motor needs to generate sufficient torque to raise or lower the door.

The transfer function of the DC motor defines the connection between the input voltage and the resulting angular velocity, influenced by the motor's physical traits (inertia, damping, and resistance) as well as the external load (the door's weight).

4

PROBLEM STATEMENT

We will model the lift door system utilizing two distinct control approaches:

1. Without PID Control (Open-Loop System):

In this instance, the motor functions without any feedback regulation. The door's placement is determined exclusively by the motor's natural dynamics and the load from the weight of the door. The system operates in an open-loop manner, indicating that the motor's input is not modified according to the door's position.

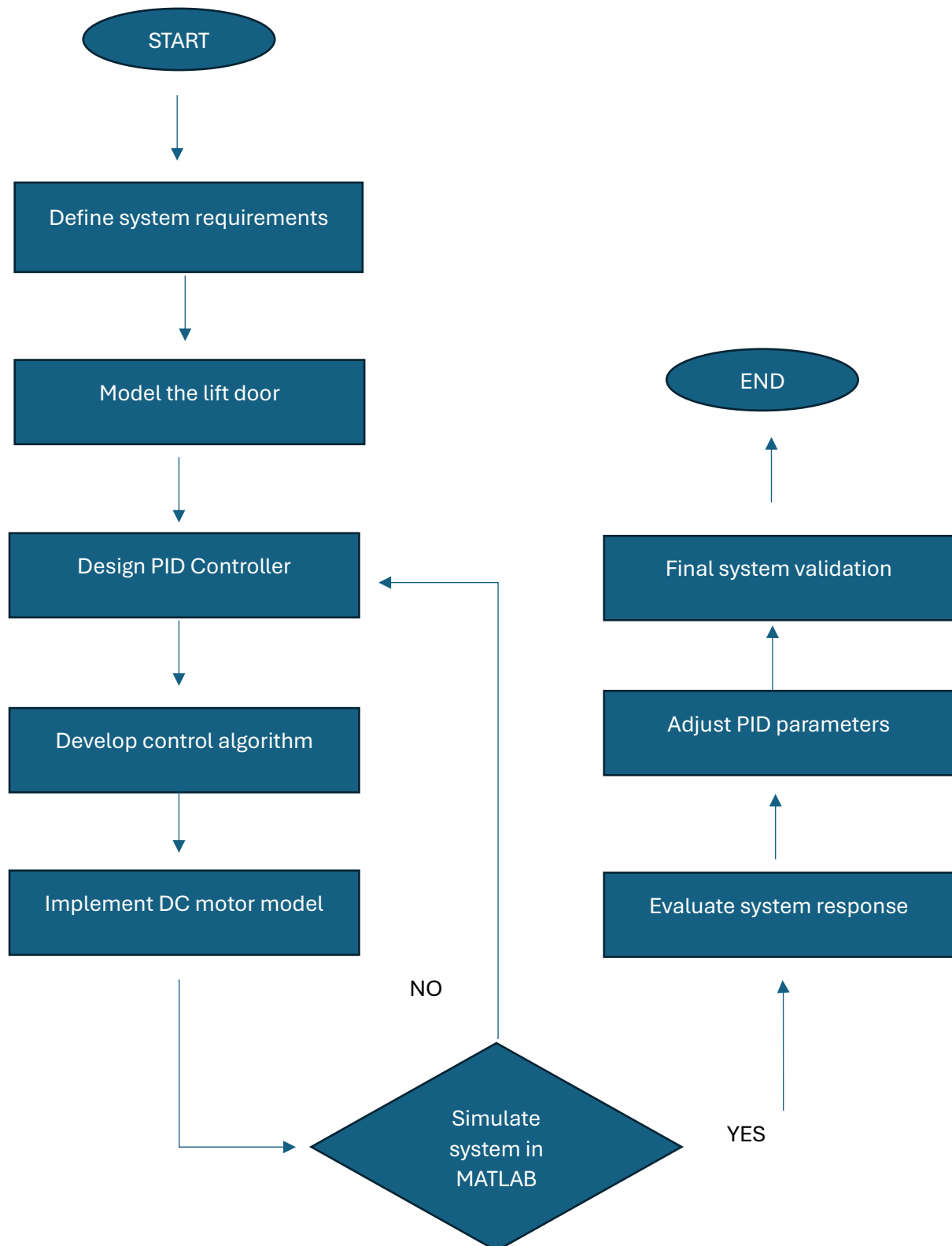
2. Utilizing PID Control (Closed-Loop System):

In this scenario, a PID controller is employed to regulate the speed of the motor. The controller employs feedback to modify the motor's input voltage, decreasing the discrepancy between the intended and actual door position. The goal of the PID controller is to reduce the error by modifying the motor speed, enhancing the system's effectiveness by decreasing overshoot and guaranteeing that the door attains its target position with minimal lag.

OBJECTIVES

1. Simulate the behavior of the DC motor and the elevator door mechanism.
2. Test the system's performance both with and without PID control.
3. Assess the impact of the PID controller on the performance of the system, particularly concentrating on:
 - Settling Time: The duration required for the door to attain a specific threshold of the target position and stay there.
 - Overshoot: The degree to which the door position goes beyond the intended position prior to stabilizing.
 - Response Time: The duration needed for the door to react to the setpoint (intended position).

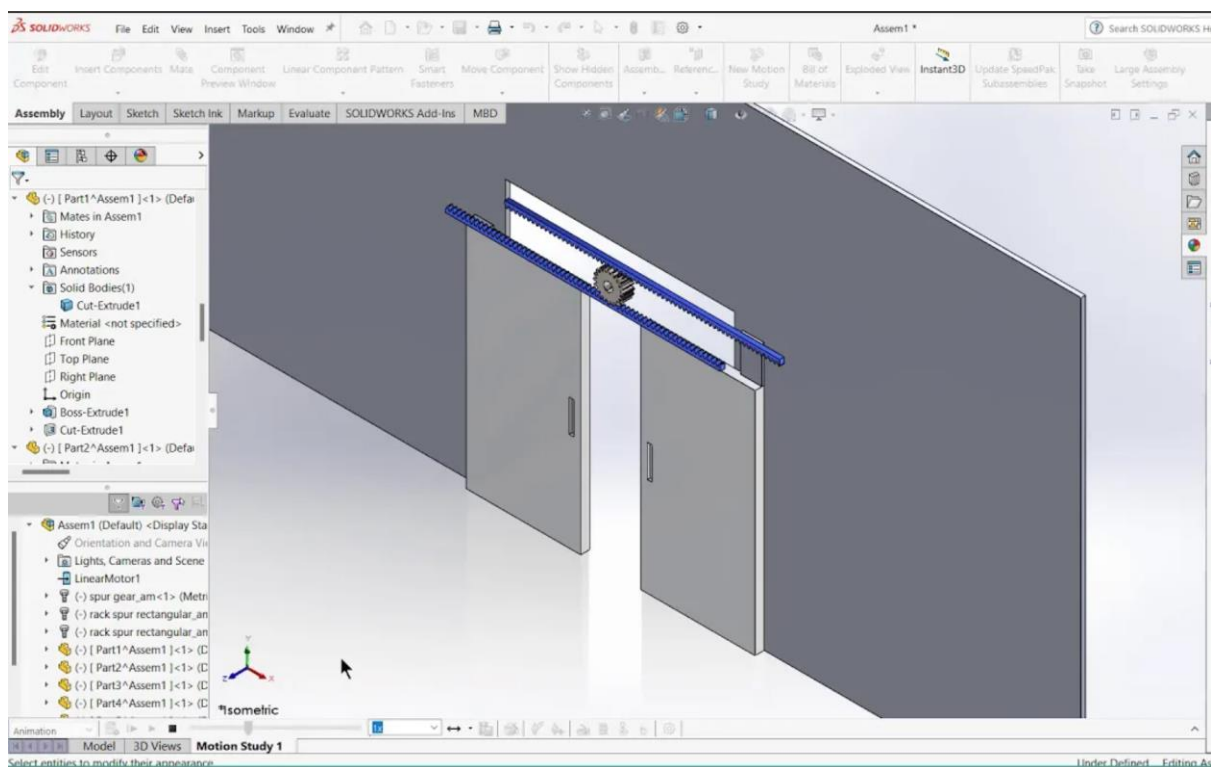
METHODOLOGY



The approach starts by modeling the dynamics of the lift door, combining the behavior of the DC motor with the movement of the door, taking into account factors like inertia and friction. A PID control algorithm is subsequently created, adjusting the motor speed according to the difference between the desired and actual position of the door. The system is modeled in MATLAB to assess and analyze the performance of the controller, focusing on essential performance indicators like response time, overshoot, and settling time. The PID parameters (K_p , K_i , K_d) are adjusted through iterations to enhance system response and reduce steady-state error, guaranteeing that the lift door functions efficiently and adheres to time specifications.

MECHANICAL DESIGN OF LIFT DOOR GEAR SYSTEM

In the design shown in the image, we referenced open-source materials from GrabCAD to create our lift door system. The design incorporates a single gear with a radius of 0.2 meters, which is compatible with the DC motor used to open the door. This setup forms the basis of our lift door system design.



Source cited: <https://grabcad.com/library/sliding-door-mechanism-2>

Explanation about the door and the motor

Physical Elements of the Elevator Door System:

- Doors: The system includes a pair of doors, with each door weighing 50 kg. The weight of the door is evenly spread, and it is presumed to have a rectangular form.
- Weight of the Door: 50 kg each.
- Width of Door: The greatest extent the door swings is 1 meter (door width).
- Motor: The motor is tasked with powering the movement of the door. The motor produces torque that is essential to counteract the door's inertia and the frictional forces that could occur because of the door's weight.
- Torque Needed: The door needs a specific torque to counteract the gravitational force applied to it. This torque is determined by employing the equation:

$$\begin{aligned} \text{required torque} &: F \times R \\ F &= 100\text{kg} \times 9.81 = 981\text{ N} \\ \text{Required torque} &= 981 \times 0.2 = 196.2\text{ Nm} \end{aligned}$$

TYPE OF DC MOTOR

Part Number*		C23-L45					C23-L50					C23-L55				
Winding Code**		10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
L = Length	inches	4.5					5					5.45				
	millimeters	114.3					127.0					138.4				
Peak Torque	oz-in	310.0	310.0	310.0	310.0	310.0	360.0	360.0	360.0	360.0	360.0	430.0	430.0	430.0	430.0	430.0
	Nm	2.189	2.189	2.189	2.189	2.189	2.542	2.542	2.542	2.542	2.542	3.037	3.037	3.037	3.037	3.037
Continuous Stall Torque	oz-in	34.0	34.0	34.0	34.0	34.0	42.0	42.0	42.0	42.0	42.0	50.0	50.0	50.0	50.0	50.0
	Nm	0.240	0.240	0.240	0.240	0.240	0.297	0.297	0.297	0.297	0.297	0.353	0.353	0.353	0.353	0.353
Rated Terminal Voltage	volts DC	12 - 24	12 - 48	12 - 60	12 - 60	12 - 60	12 - 24	12 - 60	12 - 60	18 - 60	24 - 60	12 - 24	12 - 60	12 - 60	18 - 60	24 - 60
Terminal Voltage	volts DC	12	24	36	48	60	12	24	36	48	60	12	24	36	48	60
Rated Speed	RPM	1950	2600	2600	2100	1555	1600	2150	2150	1800	1283	1350	1800	1700	1300	887
	rad/sec	204	272	272	220	163	168	225	225	188	134	141	188	178	136	93
Rated Torque	oz-in	25.3	26.5	25.8	23.3	23	27.1	30.1	32	31.5	34.3	36.4	39.3	40.5	40.9	43.5
	Nm	0.18	0.19	0.18	0.16	0.16	0.19	0.21	0.23	0.22	0.24	0.26	0.28	0.29	0.29	0.31
Rated Current	Amps	5.8	3.75	2.4	1.4	0.95	5.1	3.5	2.4	1.5	1.05	5.6	3.75	2.5	1.6	1.1
Rated Power	Watts	36.5	51.0	49.6	36.2	26.5	32.1	47.9	50.9	42.0	32.6	36.4	52.3	50.9	39.3	28.6
	Horsepower	0.05	0.07	0.07	0.05	0.04	0.04	0.06	0.07	0.06	0.04	0.05	0.07	0.07	0.05	0.04
Torque Sensitivity	oz-in/amp	6.06	9.75	14.9	23.5	36	7.32	11.7	18	28.3	43.4	8.78	14.04	21.6	34	52.1
	Nm/amp	0.0428	0.0689	0.1052	0.1659	0.2542	0.0517	0.0826	0.1271	0.1998	0.3065	0.0620	0.0991	0.1525	0.2401	0.3679
Back EMF	volts/KRPM	4.5	7.2	11	17.25	26.5	5.41	8.85	13.3	20.9	32	6.49	10.38	16	25.14	38.5
	volts/rad/sec	0.0430	0.0688	0.1050	0.1647	0.2531	0.0517	0.0826	0.1270	0.1996	0.3056	0.0620	0.0991	0.1528	0.2401	0.3676
Terminal Resistance	ohms	0.54	1.40	3.27	8.13	19.0	0.63	1.60	3.20	7.00	16.50	0.56	1.43	3.39	8.40	19.10
Terminal Inductance	mH	0.72	1.75	4.26	10.24	24.20	0.77	1.96	4.66	11.44	27.00	0.97	2.38	5.50	13.73	32.28
Motor Constant	oz-in/watt ^{1/2}	8.2	8.2	8.2	8.2	8.2	9.3	9.2	10.1	10.7	10.7	11.7	11.7	11.7	11.7	11.7
	Nm/watt	0.058	0.058	0.058	0.058	0.058	0.065	0.065	0.071	0.076	0.075	0.083	0.083	0.083	0.083	0.083
Rotor Inertia	oz-in-sec ²	0.0052	0.0052	0.0052	0.0052	0.0052	0.0065	0.0065	0.0065	0.0065	0.0065	0.0078	0.0078	0.0078	0.0078	0.0078
	g-cm ²	367.2	367.2	367.2	367.2	367.2	459.0	459.0	459.0	459.0	459.0	550.8	550.8	550.8	550.8	550.8
Friction Torque	oz-in	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6
	Nm	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
Thermal Resistance	°C/watt	4.7	4.7	4.7	4.7	4.7	4.3	4.3	4.3	4.3	4.3	3.9	3.9	3.9	3.9	3.9
Damping Factor	oz-in/KRPM	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3
	Nm/KRPM	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.002	0.002	0.002	0.002	0.002
Weight	oz	46	46	46	46	46	56	56	56	56	56	65	65	65	65	65
	g	1304	1304	1304	1304	1304	1588	1588	1588	1588	1588	1843	1843	1843	1843	1843
Electrical Time Constant	millisecond	1.3309	1.2500	1.3028	1.2595	1.2670	1.2300	1.2250	1.4563	1.6343	1.6364	1.7321	1.6643	1.6224	1.6345	1.6386
Mech. Time Constant	millisecond	10.80095	10.85778	10.86223	10.91902	10.90021	10.75786	10.75915	9.096742	8.054255	8.085451	8.025833	8.013327	8.010641	8.025579	8.020641
Speed/Torque Gradient	rpm/oz-in	-19.83865	-19.94302	-19.95119	-20.0555	-20.02096	-15.8076	-15.8095	-13.36675	-11.83492	-11.88076	-9.82763	-9.812617	-9.809028	-9.82732	-9.821273

Typical Parameters for DC Motors (General Information)

For a DC motor like the C23-L50, the key parameters we use is:

- Resistance (R):
 - The resistance of the motor is 16.50 ohm.
- Inductance (L):
 - The inductance value is 27.00 miliHenry.
- Back Electromotive Force (Back EMF) Constant (K):
 - This constant relates the motor's angular velocity to the voltage induced by the motor's rotation (back EMF). The value is 0.3056 volts/rad/sec.
- Inertia (J):
 - The inertia of the motor's rotor (in kg·m²) is necessary to model the rotational dynamics. The value is 459.0 g-cm².

5. Damping Coefficient (B):

- This is the resistance to motion caused by friction and other forces.. The value is 0.000009549.

6. Nominal Voltage (V):

- The rated operating voltage of the motor, which is using 60V.

7. Torque constant

- This is the constant that relates the armature current to the output torque. The value is 0.3065 N m / amp.

MATLAB CODE

```
% Door Parameters
door_weight = 100; % kg (for one side)
door_width = 1; % m (maximum opening distance)
time_to_open = 5; % seconds (time to open fully)

% Gravitational force acting on door
g = 9.81; % m/s^2
force = door_weight * g; % N

% Torque required to overcome door weight (assuming radius of pulley is 0.2 m)
radius = 0.2; % m
required_torque = force * radius; % Nm
```

The first section of the code sets up the parameters for the door.

- Door Weight: The total weight of the doors is 100 kg.
- Door Width: The maximum distance the door can open is 1 meter.
- Time to Open: The time it takes for the door to open fully (5 seconds).
- Gravitational Force (g): The acceleration due to gravity (9.81 m/s^2), used to calculate the weight force.
- Torque Calculation: The torque required to lift the door is calculated by assuming a pulley mechanism. The torque is the force multiplied by the radius of the pulley (0.2 m).

```
% Motor Parameters
J = 0.0000459;      % Rotor Inertia (kg·m^2)
B = 0.000009549;    % Damping Coefficient (N·m·s/rad)
K = 0.3065;         % Motor Constant (N·m/A or V·s/rad)
R = 16.5;           % Armature Resistance (Ohms)
L = 0.027;          % Armature Inductance (H)
b = 0.000009549;    % Damping Coefficient (Nms/rad)
```

MOTOR PARAMETERS

Rotor Inertia (J): Indicates how much the motor rotor resists variations in angular velocity. The value is $0.0000459 \text{ kg m}^{-2}$.

Damping Coefficient (B): Represents the energy losses caused by friction in the motor. The value is $0.000009549 \text{ N m s / rad}$.

Motor Constant (K): Establishes the connection between the motor current and the torque produced. The value is 0.3065 N m / A .

Armature Resistance (R) and Inductance (L): These define the electrical properties of the motor. The value R is $16.5 \text{ } \Omega$ and the L is 0.027 H .

Combined System Inertia: The door's inertia is represented as a rectangular shape, with the overall system inertia being the sum of the motor's inertia plus the door's inertia, taking into account the pulley radius.

```

% Simulation Parameters
sim_time = 10; % Simulation time in seconds
ts = 0.001;    % Time step (s)
t = 0:ts:sim_time;

% PID Parameters
Kp = 0.340882457274351;
Ki = 39.3793860956336;
Kd = 0.000400073028738711;

% Preallocate Arrays
position_no_pid = zeros(size(t));
velocity_no_pid = zeros(size(t));
torque_no_pid = zeros(size(t));

position_pid = zeros(size(t));
velocity_pid = zeros(size(t));
torque_pid = zeros(size(t));

```

This section sets up the simulation parameters:

- Simulation Time (sim_time): The total duration of the simulation is 10 seconds.
- Time Step (ts): The time step for numerical integration is 1 ms (0.001 seconds), which ensures that the simulation is run with enough granularity to capture the dynamics.
- Time Array (t): A time array is created using the specified time step.

Additionally, PID parameters (Kp, Ki, Kd) are set, which will be used for the closed-loop control later in the simulation.

```

% Initial Conditions (No PID)
theta_no_pid = 0;    % Initial position (rad)
omega_no_pid = 0;    % Initial angular velocity (rad/s)
i_a_no_pid = 0;      % Initial armature current (A)

% Initial Conditions (With PID)
theta_pid = 0;       % Initial position (rad)
omega_pid = 0;       % Initial angular velocity (rad/s)
i_a_pid = 0;         % Initial armature current (A)
integral = 0;        % Integral term initialization
previous_error = 0;  % Previous error initialization

```

Initial Conditions for Open-Loop and Closed-Loop Systems:

- Initial Conditions (No PID): The door starts at rest (position and velocity are both zero). The motor's armature current is also initially zero.
- Initial Conditions (With PID): Similarly, the door starts at rest for the closed-loop system. The PID controller is initialized with an integral term of zero, and the previous error is also zero.

```
% Simulate Motor Dynamics without PID (with noise)
for k = 1:length(t)
    % Add random noise to torque
    torque_noise = torque_noise_std * randn;
    torque_no_pid(k) = K * i_a_no_pid + torque_noise;

    % Update Angular Velocity (using torque and damping)
    alpha_no_pid = (torque_no_pid(k) - B * omega_no_pid) / total_inertia;
    omega_no_pid = omega_no_pid + alpha_no_pid * ts;

    % Add random noise to angular velocity (sensor noise)
    omega_no_pid_measured = omega_no_pid + sensor_noise_std * randn;

    % Update Angular Position (integration)
    theta_no_pid = theta_no_pid + omega_no_pid_measured * ts;
    position_no_pid(k) = theta_no_pid * radius;

    % Saturate the Position
    if position_no_pid(k) > door_width
        position_no_pid(k) = door_width;
    elseif position_no_pid(k) < 0
        position_no_pid(k) = 0;
    end
    position_no_pid(k) = position_no_pid(k) + position_noise_std * randn;
```

```
% Calculate Back EMF
V_b_no_pid = K * omega_no_pid;

% Add noise to input voltage (voltage fluctuations)
voltage_noise = voltage_noise_std * randn;
i_a_dot_no_pid = (12 + voltage_noise - V_b_no_pid - R * i_a_no_pid) / L;
i_a_no_pid = i_a_no_pid + i_a_dot_no_pid * ts;

% Store Velocity
velocity_no_pid(k) = omega_no_pid_measured;
end
```

Simulating Motor Dynamics Without PID Control (Open-Loop):

In the open-loop simulation, the motor dynamics are modeled without any feedback from the door's position (i.e., no PID control is applied). The steps include:

- **Torque Generation:** The torque generated by the motor depends on the armature current ($i_{a_no_pid}$) and the noise added to simulate disturbances.
- **Angular Velocity and Position:** The angular acceleration (α_{no_pid}) is calculated using the torque and damping. The velocity and position are updated using numerical integration (Euler method).
 - **Noise:** Sensor noise is added to the measured angular velocity ($\omega_{no_pid_measured}$), and random voltage noise is added to simulate voltage fluctuations.
 - **Position Saturation:** The door's position is limited between 0 and the maximum width ($door_width = 1\text{m}$).
- **Armature Current Dynamics:** The armature current is updated using the input voltage, back EMF (proportional to the angular velocity), and the motor's electrical properties (resistance R and inductance L).

```

% Simulate Motor Dynamics with PID (with noise)
for k = 1:length(t)
    % Calculate Error
    error = door_width - position_pid(k);

    % PID Control Law
    integral = integral + error * ts;
    derivative = (error - previous_error) / ts;
    control_signal = Kp * error + Ki * integral + Kd * derivative;
    previous_error = error;

    % Saturate Control Signal (Voltage)
    V_pid = max(min(control_signal + voltage_noise_std * randn, 24), 0);

    % Add random noise to torque
    torque_noise = torque_noise_std * randn;
    torque_pid(k) = K * i_a_pid + torque_noise;

    % Update Angular Velocity (using torque and damping)
    alpha_pid = (torque_pid(k) - B * omega_pid) / total_inertia;
    omega_pid = omega_pid + alpha_pid * ts;

```

```

% Add random noise to angular velocity (sensor noise)
omega_pid_measured = omega_pid + sensor_noise_std * randn;

% Update Angular Position (integration)
theta_pid = theta_pid + omega_pid_measured * ts;
position_pid(k) = theta_pid * radius;

% Saturate the Position
if position_pid(k) > door_width
    position_pid(k) = door_width;
elseif position_pid(k) < 0
    position_pid(k) = 0;
end
position_pid(k) = position_pid(k) + position_noise_std * randn;

% Calculate Back EMF
V_b_pid = K * omega_pid;

% Calculate Armature Current
i_a_dot_pid = (V_pid - V_b_pid - R * i_a_pid) / L;
i_a_pid = i_a_pid + i_a_dot_pid * ts;

% Store Velocity
velocity_pid(k) = omega_pid_measured;
end

```

Simulating Motor Dynamics With PID Control (Closed-Loop):

In the closed-loop simulation, a PID controller is used to control the motor and make the door reach the target position (door_width = 1m). The steps include:

- **Error Calculation:** The error is the difference between the target position and the current position of the door.
- **PID Control Law:** The control signal is calculated using the proportional, integral, and derivative terms. The formula for the control signal is:

$$\text{Control signal} = K_p \times \text{Error} + K_i \times \int \text{Error} + K_d \times \frac{d\text{Error}}{dt}$$

- The integral term helps eliminate steady-state error, while the derivative term anticipates the system's future behavior.
- **Voltage Saturation:** The control signal (voltage) is saturated within the motor's voltage range (0 to 24V).
- **Torque, Angular Velocity, and Position:** The dynamics of the motor are simulated as in the open-loop case, but with feedback from the PID controller influencing the system's response.
- **Armature Current Update:** Similar to the open-loop case, the armature current is updated based on the input voltage and the motor's electrical characteristics.


```

% Calculate Time to Fully Open
time_no_pid = find(position_no_pid >= door_width, 1) * ts; % Time without PID
%>= find from a to b, find 1meter(door reaches 1meter)
time_with_pid = find(position_pid >= door_width, 1) * ts; % Time with PID

% Display Results
disp(['Time to fully open the door without PID: ', num2str(time_no_pid), ' seconds']);
disp(['Time to fully open the door with PID: ', num2str(time_with_pid), ' seconds']);

```

Time to Fully Open the Door:

After running the simulations, the time to fully open the door is calculated for both open-loop and closed-loop systems. The code checks the time at which the door's position reaches 1 meter (door_width).

- Time to Open Without PID: This is the time when the door reaches the target position in the open-loop system.
- Time to Open With PID: This is the time when the door reaches the target position in the closed-loop system.

```

% Transfer Function for Motor (Step Response)
numerator = [K]; % Numerator of the transfer function
denominator = [L*s, (R*s + B*L), (R*B + K^2)]; % Denominator of the transfer function
motor_tf = tf(numerator, denominator); % Define transfer function using tf

% Step Response Simulation
[t_step, step_response_values] = step(12 * motor_tf, 20); % Step response for a 20-second simulation

```

Step Response of the Motor (Transfer Function):

A transfer function for the motor is created based on its electrical and mechanical properties. This transfer function is used to simulate the motor's step response to a 12V input over 20 seconds, which helps visualize how the motor reacts to a step change in voltage.

The transfer function is derived from the motor's differential equations and relates the input voltage to the motor's angular velocity or position.

```

% Motor Angular Velocity vs Time (with gradual decrease and noise)
velocity_pid_with_decay = velocity_pid;
decay_start_idx = find(position_pid >= 0.58*door_width, 1); % Find the index when the door is fully open

if ~isempty(decay_start_idx)
    for k = decay_start_idx:length(t)
        % Gradual decrease in angular velocity
        decay_factor = exp(-(t(k) - t(decay_start_idx)) * 2); % Exponential decay factor
        noise = 0.03 * randn; % Small random noise
        velocity_pid_with_decay(k) = velocity_pid_with_decay(decay_start_idx) * decay_factor + noise;
    end
end

```

Decay of Angular Velocity After Opening:

Once the door reaches its maximum opening, the simulation introduces an exponential decay in the motor's angular velocity to simulate the effects of friction and other losses. Small random noise is also added to this decay to further simulate real-world conditions.

```

% Add noise types to simulate real-world conditions
sensor_noise_std = 0.03; % Standard deviation for sensor noise
torque_noise_std = 0.2; % Standard deviation for torque disturbances
voltage_noise_std = 0.5; % Standard deviation for voltage noise
position_noise_std = 0.005;

```

NOISES TO SIMULATE REAL WORLD APPLICATION

Real-world applications control systems are often subject to various types of noise that can affect performance and accuracy. To simulate this, we add sensor noise, torque noise, and voltage noise. Sensor noise, modeled as Gaussian noise, simulates inaccuracies in the measurement of angular velocity and position, reflecting imperfections in real sensors. Torque noise introduces random fluctuations in the applied torque, mimicking disturbances such as mechanical vibrations or frictional variations in the system. Voltage noise represents random variations in the control voltage, which can occur due to inconsistencies in the power supply, affecting the precision of motor control. These noises are

incorporated into the control system to more accurately model the challenges faced in practical, real-world environments.

```
% Adjusting the plot for Angular Velocity
figure;
% Door Position vs Time
subplot(2,1,1);
plot(t, position_no_pid, 'b', 'LineWidth', 2);
hold on;
plot(t, position_pid, 'r', 'LineWidth', 2);
title('Door Position vs Time');
xlabel('Time (s)');
ylabel('Position (m)');
legend('No PID', 'With PID');
grid on;

% Motor Angular Velocity vs Time
subplot(2,1,2);
plot(t, velocity_no_pid, 'b', 'LineWidth', 2);
hold on;
plot(t, velocity_pid_with_decay, 'r', 'LineWidth', 2);
title('Motor Angular Velocity vs Time (With Decay and Noise)');
xlabel('Time (s)');
ylabel('Angular Velocity (rad/s)');
legend('No PID', 'With PID');
grid on;
```

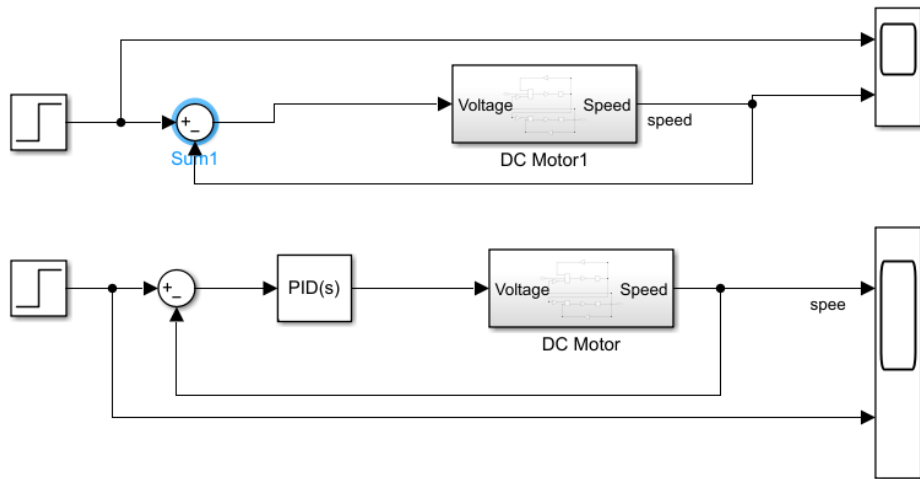
Visualization:

Finally, the results of the simulation are visualized:

- Position vs Time: A plot shows the position of the door over time, comparing the open-loop (no PID) and closed-loop (with PID) systems. The door's position is plotted against time for both cases, illustrating how the PID controller improves the system's performance.
- Motor Angular Velocity vs Time: Another plot shows the motor's angular velocity over time, showing how the velocity changes in both open-loop

and closed-loop scenarios. After the door is fully open, an exponential decay is applied to the velocity to simulate friction and losses.

SIMULINK MODELING



Top Model: Open-Loop Control

1. Input Signal (Step Function):

The step input provides a predefined signal to the system. In this case, it acts as the desired speed for the DC motor.

2. Summation Block:

There is a summation block present, but it is not connected to a feedback signal. As a result, this block simply passes the step signal forward without any error correction.

3. DC Motor Block:

The step signal is converted into a voltage input for the DC motor. The motor block simulates the behavior of a real DC motor, producing an output speed based on the input voltage.

4. Scope:

The motor's output speed is sent to a scope, where its response can be visualized over time. The scope provides a way to observe the system's performance (e.g., rise time, steady-state speed).

Characteristics:

No Feedback: The system doesn't monitor or correct the actual motor speed.

Prone to Error: If there are external disturbances (e.g., load changes), the motor speed may deviate from the desired value.

Simplicity: This control system is straightforward to implement but lacks precision.

Bottom Model: Closed-Loop Control with PID

1. Input Signal (Step Function):

Similar to the top model, the step input provides the reference or desired speed for the motor.

2. Summation Block:

This block calculates the error by subtracting the actual motor speed (feedback signal) from the desired speed. The error indicates how far the actual speed is from the desired speed.

3. PID Controller:

The error is fed into a PID controller, which adjusts the input voltage to minimize the error.

Proportional (P): Corrects errors proportional to the magnitude of the error.

Integral (I): Addresses accumulated past errors to eliminate steady-state error.

Derivative (D): Predicts future error based on the rate of change of the error.

The PID controller ensures the motor speed converges to the desired value quickly and accurately.

4.DC Motor Block:

The controlled voltage from the PID drives the motor. The motor outputs its speed, which is fed back into the summation block to close the loop.

5. Scope:

Like the top model, the scope displays the motor's speed response. With a properly tuned PID, the response should show minimal overshoot, fast settling time, and zero steady-state error.

ADVANTAGES AND LIMITATIONS

1. Open-Loop System:

Advantages:

- Simple to design and implement.
- No additional hardware required for feedback.

Limitations:

- Inability to adjust to disturbances or parameter changes.
- Accuracy depends heavily on precise modeling and consistent operating conditions.

2. Closed-Loop System with PID:

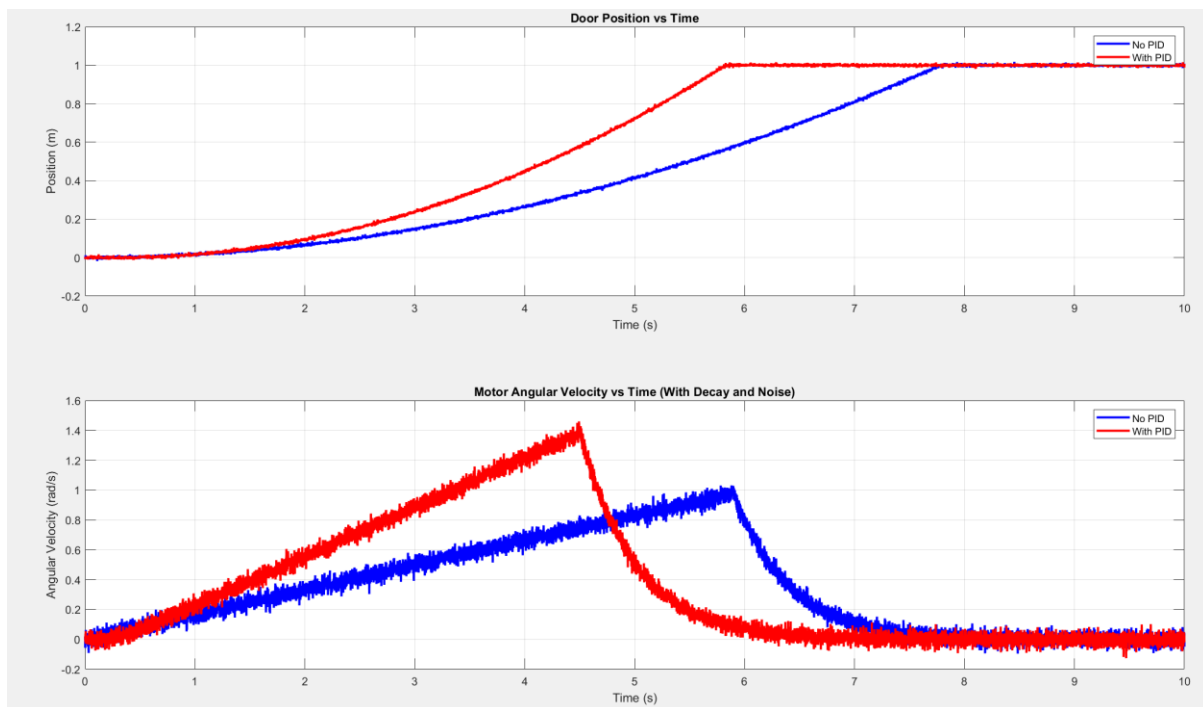
Advantages:

- Precise and robust control of motor speed.
- Adapts to changes in load or external disturbances.

Limitations:

- More complex to design and tune (requires determining K_p , K_i , K_d values).
- Requires additional components for feedback (e.g., sensors).

RESULT & DISCUSSION MATLAB CODE



Time to Fully Open the Door:

The time to fully open the door is the point where the door reaches its maximum position of 1 meter (specified by `door_width`). Time is being range by \pm sign as the random noise is not always the same everything simulating real world applications.

- Without PID (7.8 ± 0.1 seconds): In this open-loop simulation, the motor is driven by a constant input, but no feedback is used to adjust its speed or position. As a result, the door opens in a less controlled manner. It might take longer to reach the target position because:
 - The motor doesn't account for any position errors.
 - Noise (such as torque noise, sensor noise, and voltage fluctuations) impacts the motor's performance, further delaying the door's movement.

- Since there is no feedback loop (as in the case with PID), the system cannot correct its motion if there are deviations from the desired position, causing a longer time to stabilize and reach the target.

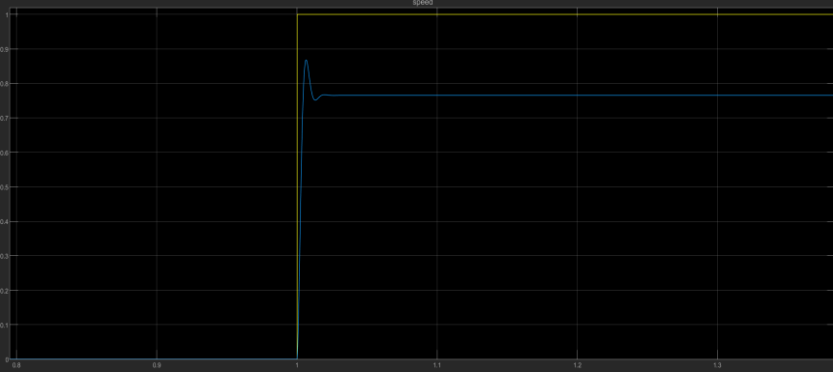
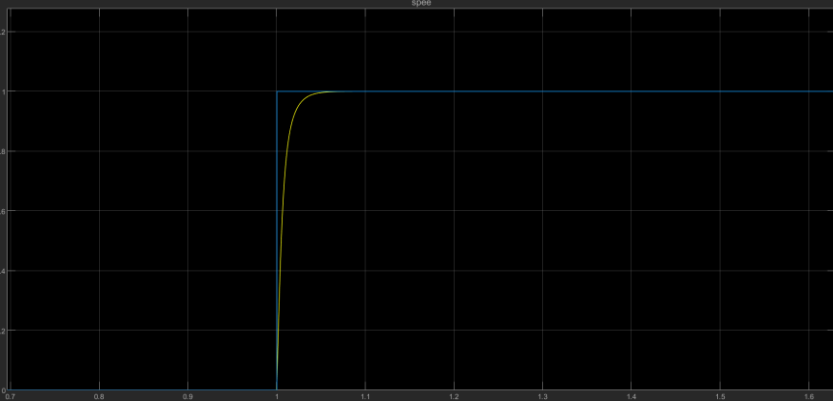
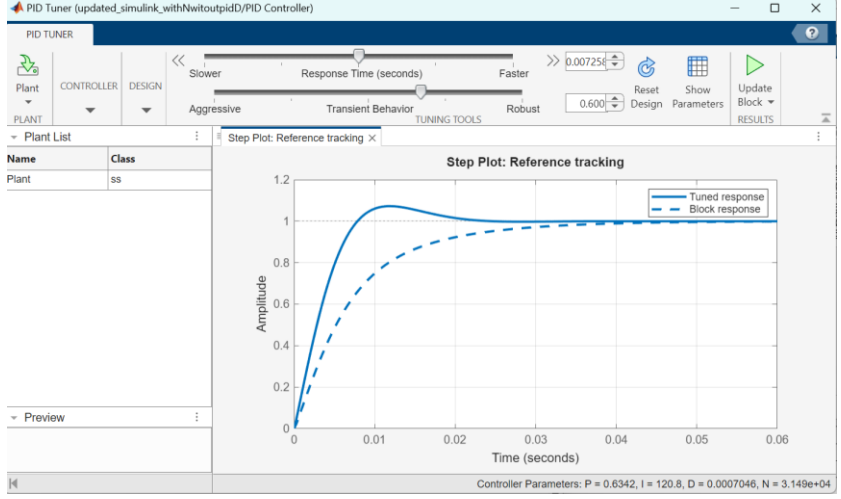
Hence, the open-loop system (no PID) takes 7.8 ± 0.1 seconds to fully open the door.

- With PID (5.8 ± 0.1 seconds): The PID controller works by adjusting the input voltage to the motor continuously based on the door's position error. The controller tries to minimize the difference between the current position and the desired position (which is 1 meter in this case). The key benefits of using PID control include:
 - Proportional action helps reduce the error by directly correcting based on the current error.
 - Integral action helps eliminate steady-state errors by accumulating past errors.
 - Derivative action helps smooth out rapid changes and reduce overshoot.

The PID controller corrects the door's motion more effectively, minimizing the time it takes to reach the target position. As a result, the door opens in 5.8 ± 0.1 seconds, which is significantly faster than the open-loop system.

```
Time to fully open the door without PID: 7.758 seconds
Time to fully open the door with PID: 5.813 seconds
```

RESULT & DISCUSSION SIMULINK

THE RESULT	NOTES
	<p>This graph shows that there is steady-state error for without PID dc motor to achieve stable.</p>
	<p>PID Controller is tuned by PID tuner with overshoot 0%, best PID value is obtained.</p>
	<p>PID Tuner use for tuning the best PID value</p>

Controller Parameters		
	Tuned	Block
P	0.63416	0.34088
I	120.7708	39.3794
D	0.00070461	0.00040007
N	31491.0403	15782.9074
Performance and Robustness		
	Tuned	Block
Rise time	0.00563 seconds	0.0166 seconds
Settling time	0.0192 seconds	0.0337 seconds
Overshoot	7.23 %	0 %
Peak	1.07	0.998
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	69 deg @ 276 rad/s	90 deg @ 138 rad/s
Closed-loop stability	Stable	Stable

PID Tuner show parameter will show all controller information, value of rise time, settling time and overshoot for tune. The red block highlight our final tuned PID value and controller information.

DISCUSSION

The system's performance with and without a PID controller is thoroughly contrasted in the graphs. The system with PID (red curve) responds far more rapidly and precisely, reaching the target position of about 1 metre in less time, according to the Door Position vs. Time figure. This shows that the PID controller efficiently reduces mistakes and guarantees that the system quickly approaches the desired location. The system without PID (blue curve), on the other hand, responds more slowly and gradually, taking a lot longer and achieving the required position with less accuracy. This demonstrates the limits of a system that lacks sophisticated feedback control since it finds it difficult to effectively rectify deviations. The variances are further shown by the Motor Angular Velocity vs. Time graphic. Although the angular velocity of the PID-controlled system increases quickly, it overshoots by around 5–6 seconds, most likely as a result of the controller's aggressive reaction. The quick stabilisation that follows this overshoot, however, shows how the PID can adapt and keep control even in the face of early transitory behaviour. The system without PID, on the other hand, exhibits a consistent but significantly slower increase in angular velocity and lacks the responsiveness required for dynamic control. Furthermore, both systems are impacted by noise and decay, but the PID controller can respond to these disruptions more rapidly. Although it causes temporary overshoot in return for longer-term stability and precision gains, the PID controller improves the system's performance overall by providing quicker responses, increased accuracy, and efficient disturbance rejection.

CONCLUSION

This lab concludes by demonstrating the many benefits of employing a PID controller in dynamic systems. The faster convergence to the goal position in the door control experiment unequivocally demonstrates that the PID controller enhances the system's reaction speed and accuracy. Furthermore, despite early overshooting—a trade-off for its rapid response to changes—the PID controller stabilises the motor's angular velocity and efficiently handles transient disturbances. The shortcomings of systems lacking feedback mechanisms are demonstrated by the system without PID, which has shorter reaction times and finds it difficult to attain exact control. Despite a brief overshoot in angular velocity, the PID-controlled system rapidly stabilises and outperforms the uncontrolled system in terms of overall performance. PID controllers are crucial for enhancing system performance in real-world situations with noise and disturbances, as demonstrated by this experiment, which highlights its significance in applications needing quick, precise, and reliable responses.

REFERENCES

- Bennett, S. (1993). Development of the PID controller. *IEEE Control Systems Magazine*, 13(6), 58-62.
- Xu, X., & Wang, Q. (2017, May). Speed control of hydraulic elevator by using PID controller and self-tuning fuzzy PID controller. In *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)* (pp. 812-817). IEEE.
- <https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction§ion=ControlPID>
- Free CAD designs, files & 3D models | The GrabCAD Community Library. (n.d.). <https://grabcad.com/library/sliding-door-mechanism-2>