ASSIGNMENT -2
Control System Design for Actuator Positioning

Balaji sai Tolapu

1 Introduction

In this assignment, the goal is to learn and apply industrial control system simulations. The objective is to tune the controller gains so that we can observe the system response and identify the best possible solution using SIMULINK simulations.we will start with matlab and then move on to SIMULINK. The system in question is a medical device that requires a special machine to control the vertical position of an actuator. The transfer function of the actuator is given by:

$$G(s) = \frac{X(s)}{Y(s)} = \frac{1}{ms^2 + bs + k}$$

where x is the system displacement in meters, y is the input, m is the actuator mass (10 kg), b is the damping coefficient (20 N/m/s), and k is the actuator stiffness (100 N/m). The goal is to design a system that can position the actuator quickly when a step input is applied with minimum inertia load.

2 System Parameters

First, we define the parameters of the actuator:

$$G(s) = \frac{1}{10s^2 + 20s + 100}$$

```
% Define system parameters
m = 10; % kg
b = 20; % N/m/s
k = 100; % N/m

% Define the transfer function
num = 1; % Numerator
den = [m, b, k]; % Denominator
G = tf(num, den);

% Step response for the uncontrolled system
figure;
step(G);
title('Step Response of the Uncontrolled Actuator');
grid on;
```

3 Controller Design

3.1 Proportional (P) Controller

A proportional controller can be designed by tuning the gain K_p.

```
% P Controller
Kp = 50; % Proportional gain
C_p = pid(Kp);
```

```
% Closed-loop system
sys_cl_p = feedback(C_p * G, 1);

% Step response of the P controller
figure;
step(sys_cl_p);
title('Step Response with P Controller');
grid on;
```

3.2 Proportional-Integral (PI) Controller

For a PI controller, we set the proportional and integral gains.

```
% PI Controller
Kp = 50; % Proportional gain
Ki = 10; % Integral gain
C_pi = pid(Kp, Ki);

% Closed-loop system
sys_cl_pi = feedback(C_pi * G, 1);

% Step response of the PI controller
figure;
step(sys_cl_pi);
title('Step Response with PI Controller');
grid on;
```

3.3 Proportional-Derivative (PD) Controller

We set the proportional and derivative gains for the PD controller.

```
% PD Controller
Kp = 50; % Proportional gain
Kd = 5; % Derivative gain
C_pd = pid(Kp, 0, Kd);

% Closed-loop system
sys_cl_pd = feedback(C_pd * G, 1);

% Step response of the PD controller
figure;
step(sys_cl_pd);
title('Step Response with PD Controller');
grid on;
```

3.4 Proportional-Integral-Derivative (PID) Controller

Finally, we combine all three gains for the PID controller.

```
% PID Controller
Kp = 50; % Proportional gain
Ki = 10; % Integral gain
Kd = 5; % Derivative gain
C_pid = pid(Kp, Ki, Kd);

% Closed-loop system
sys_cl_pid = feedback(C_pid * G, 1);

% Step response of the PID controller
```

```
figure;
step(sys_cl_pid);
title('Step Response with PID Controller');
grid on;
```

4 System Response Analysis

To analyze the system response for each controller type (P, PI, PD, and PID) and determine whether the system is underdamped, overdamped, or critically damped, we evaluate the step response characteristics.

```
% Function to analyze damping characteristics
function analyze_damping(y, t, controller_name)
    % Calculate the overshoot and settling time
    info = stepinfo(y, t);

    % Extract overshoot and settling time
    overshoot = info.Overshoot;
    settling_time = info.SettlingTime;

    fprintf('%s:\n', controller_name);
    fprintf('  Percent Overshoot: %.2f%%\n', overshoot);
    fprintf('  Settling Time: %.2f seconds\n', settling_time);

    % Determine damping type based on overshoot
    if overshoot > 0
        fprintf('  System is Underdamped\n\n');
    elseif overshoot == 0 && settling_time < 2
        fprintf('  System is Critically Damped\n\n');
    else
        fprintf('  System is Overdamped\n\n');
    end
end
```

5 Tuning Strategy

To effectively tune the controller gains (K_p, K_i, K_d), we define a tuning strategy that involves creating a loop that adjusts the gains, performs simulations, and visualizes the results.

```
% Define ranges for tuning
Kp_values = 0:5:100; % Proportional gain range
Ki_values = 0:1:20;  % Integral gain range
Kd_values = 0:1:20;  % Derivative gain range

% Initialize best performance metrics
best_overshoot = inf;
best_settling_time = inf;
best_params = [0, 0, 0];

% Loop through each gain combination
for Kp = Kp_values
    for Ki = Ki_values
        for Kd = Kd_values
            % Create PID Controller
            C = pid(Kp, Ki, Kd);
            % Create Closed-loop System
            sys_cl = feedback(C * G, 1);

            % Calculate step response
            info = stepinfo(sys_cl);

            % Check performance criteria
            if info.Overshoot < best_overshoot ||
            (info.Overshoot == best_overshoot && info.SettlingTime < best_settling_time)
                best_overshoot = info.Overshoot;
                best_settling_time = info.SettlingTime;
                best_params = [Kp, Ki, Kd];
            end

            % Optional: Display progress
            fprintf('Kp: %d, Ki: %d, Kd: %d => Overshoot: %.2f%%,
            ...Settling Time: %.2f seconds\n', Kp, Ki, Kd, info.Overshoot, info.SettlingTime);
        end
    end
end

% Display best parameters found
fprintf('Best Parameters:\n Kp: %.2f, Ki: %.2f, Kd: %.2f\n',
...best_params(1), best_params(2), best_params(3));

fprintf('Best Performance:\n Overshoot: %.2f%%,
...Settling Time: %.2f seconds\n', best_overshoot, best_settling_time);
```

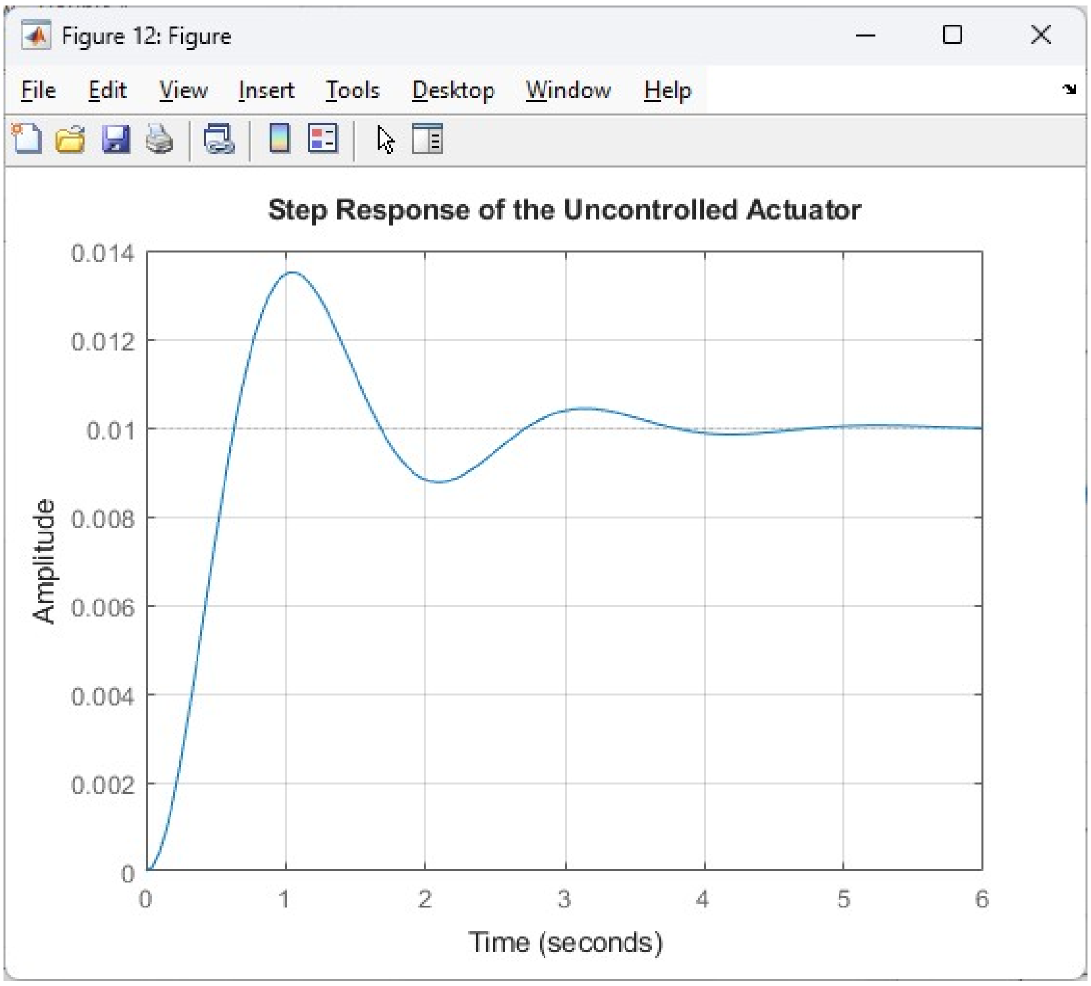


Figure 1: Step Response of the Uncontrolled Actuator

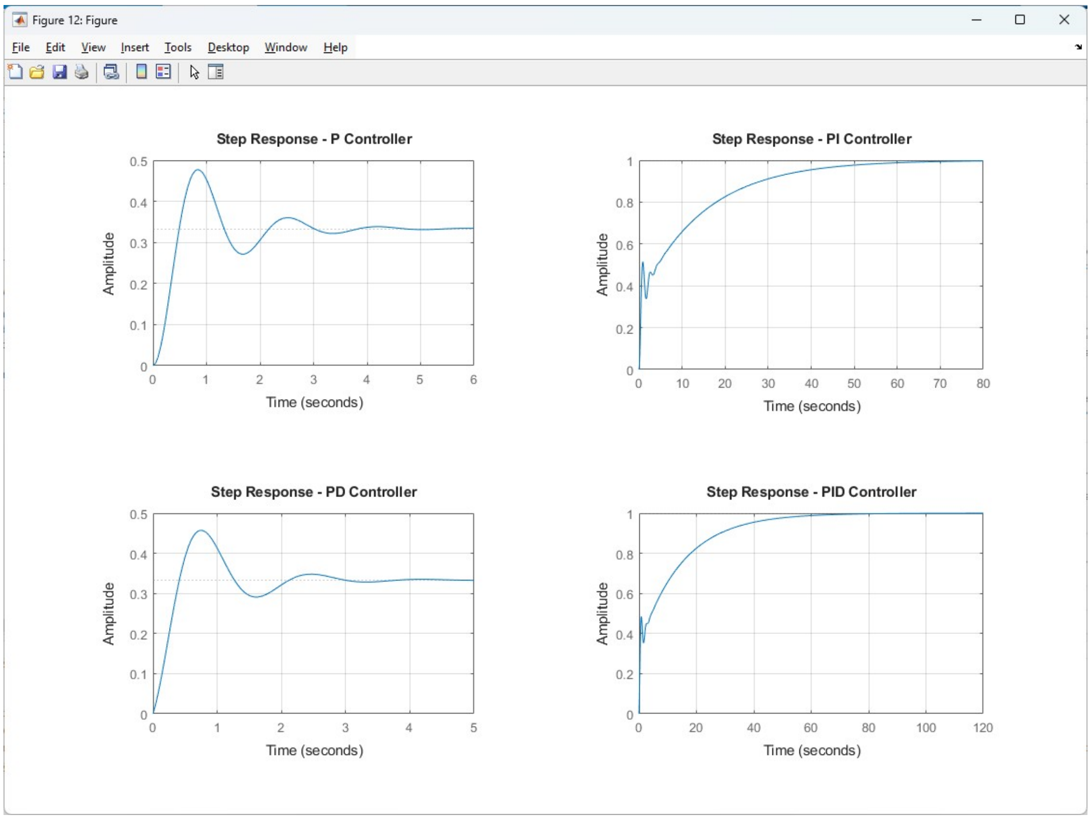


Figure 2: Step Response with P,PI,PD,PID Controller

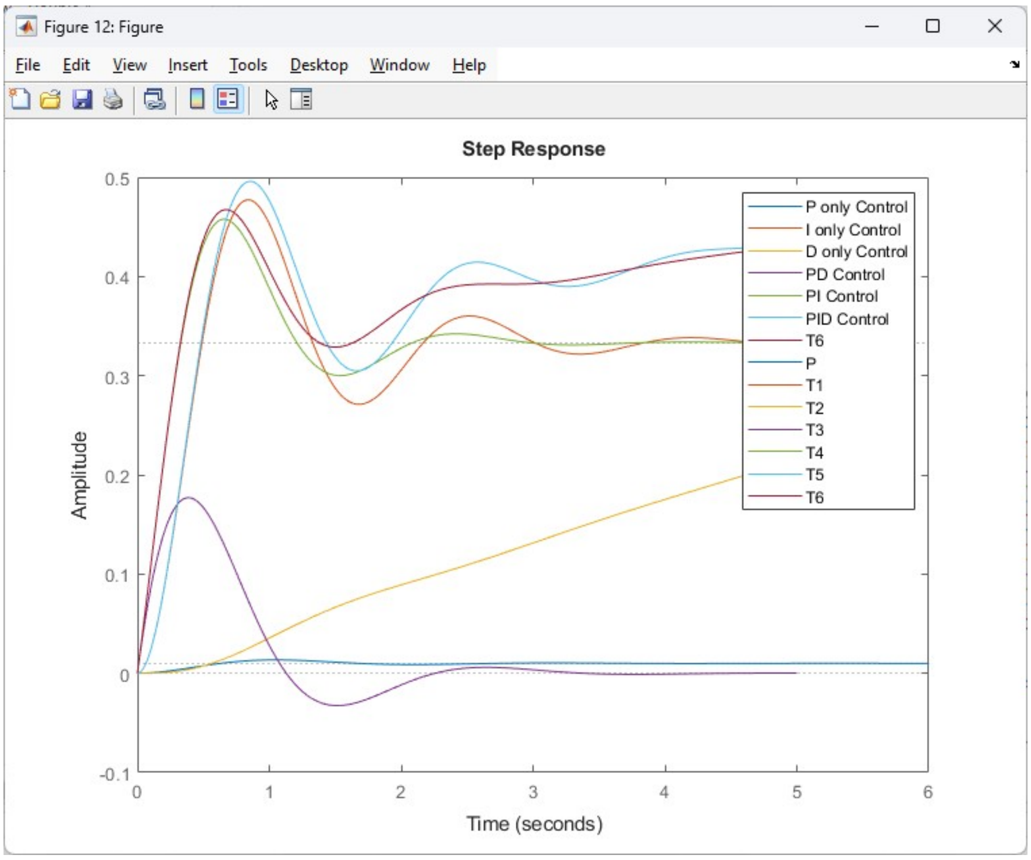


Figure 3: Step Response with P,I,D,PI,PD,PID Controller on one plot

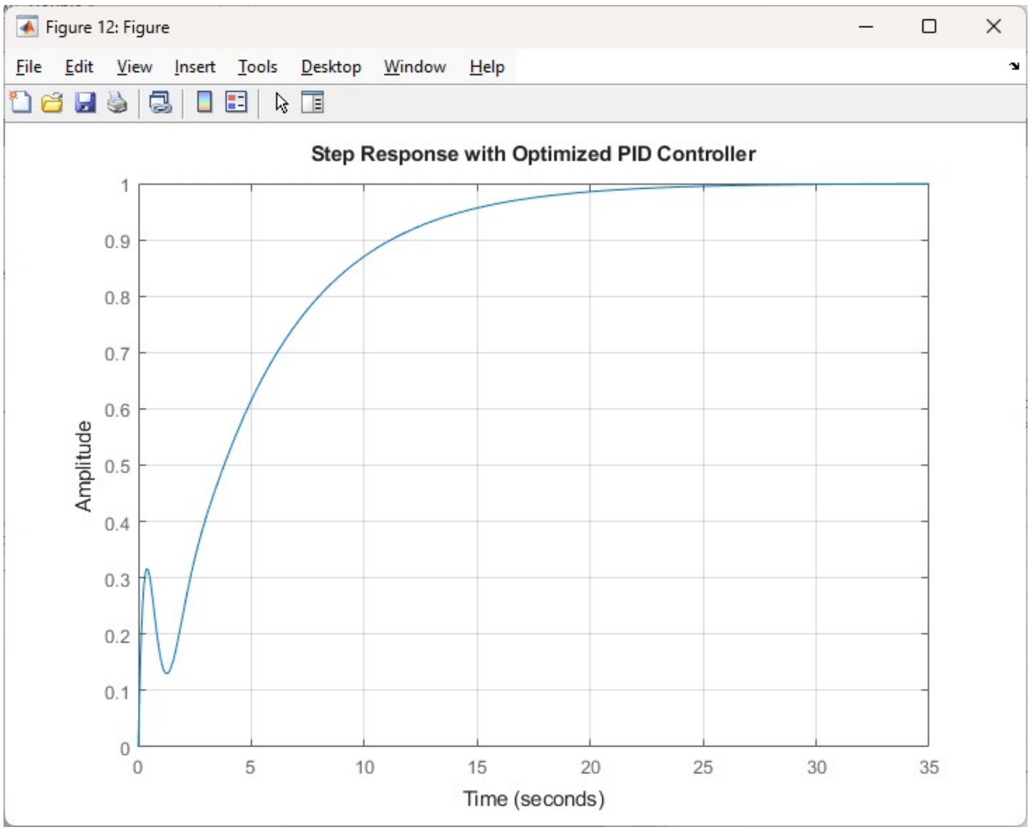


Figure 4: Step Response with Optimized PID Controller

6 MATLAB - results

The results of the tuning process are as follows:

- P Controller:
  - Percent Overshoot: 42.77%
  - Settling Time: 3.66 seconds
  - System is Underdamped
- PI Controller:
  - Percent Overshoot: 0.00%
  - Settling Time: 49.62 seconds
  - System is Overdamped
- PD Controller:
  - Percent Overshoot: 37.09%
  - Settling Time: 2.78 seconds
  - System is Underdamped
- PID Controller:
  - Percent Overshoot: 0.00%
  - Settling Time: 51.95 seconds
  - System is Overdamped

The best parameters found are:

- $K_p = 0.00$
- $K_i = 20.00$
- $K_d = 20.00$

The best performance achieved is:

- Overshoot: 0.00%
- Settling Time: 18.57 seconds

7 SIMULINK ANALYSIS

8 Introduction

This report analyzes the performance of P, PI, PD, and PID controllers applied to a mass-spring-damper system, modeled as a medical device actuator. The system's transfer function is given by:

$$\frac{X(s)}{Y(s)} = \frac{1}{10s^2 + 20s + 100}$$

representing a second-order system with mass $m = 10\,\text{kg}$, damping coefficient $b = 20\,\text{N}\cdot\text{s/m}$, and stiffness $k = 100\,\text{N/m}$. The objectives are to minimize settling time and inertia load (measured as maximum acceleration overshoot) while classifying the damping behavior of each simulation run.

9 Methodology

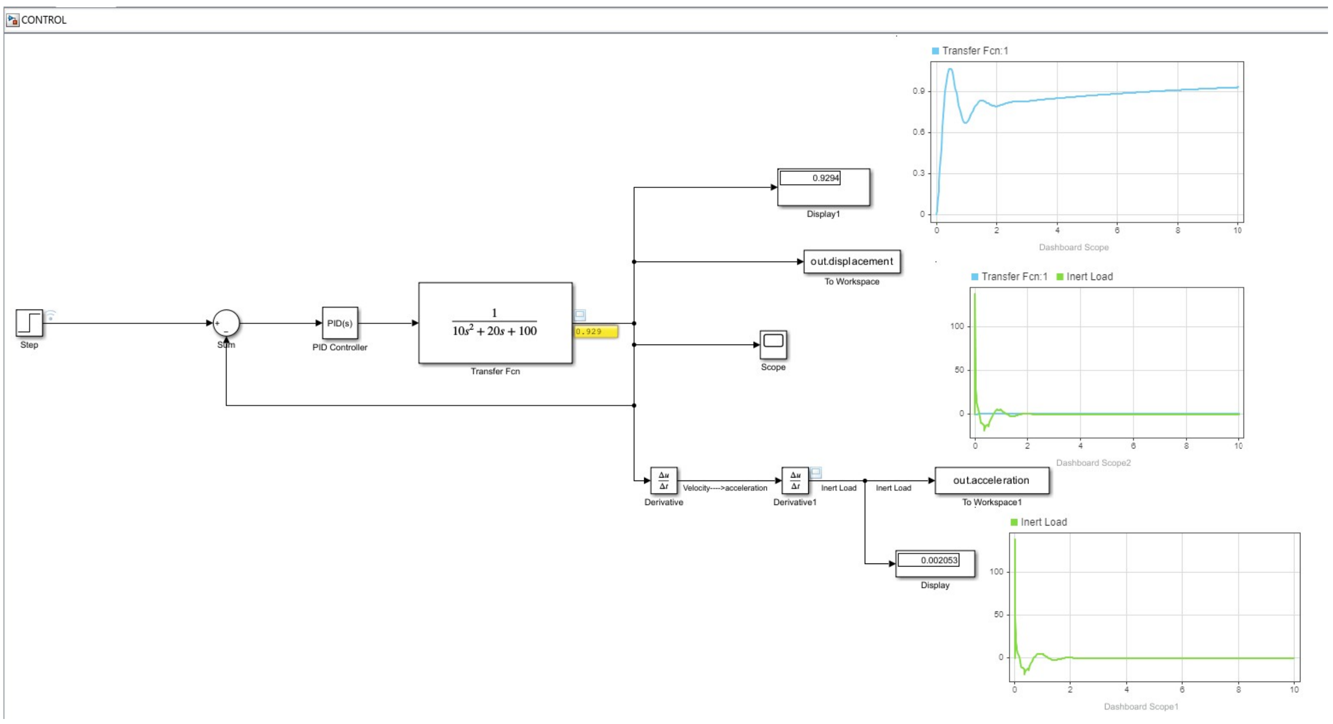


Figure 5: simulink closed lood system

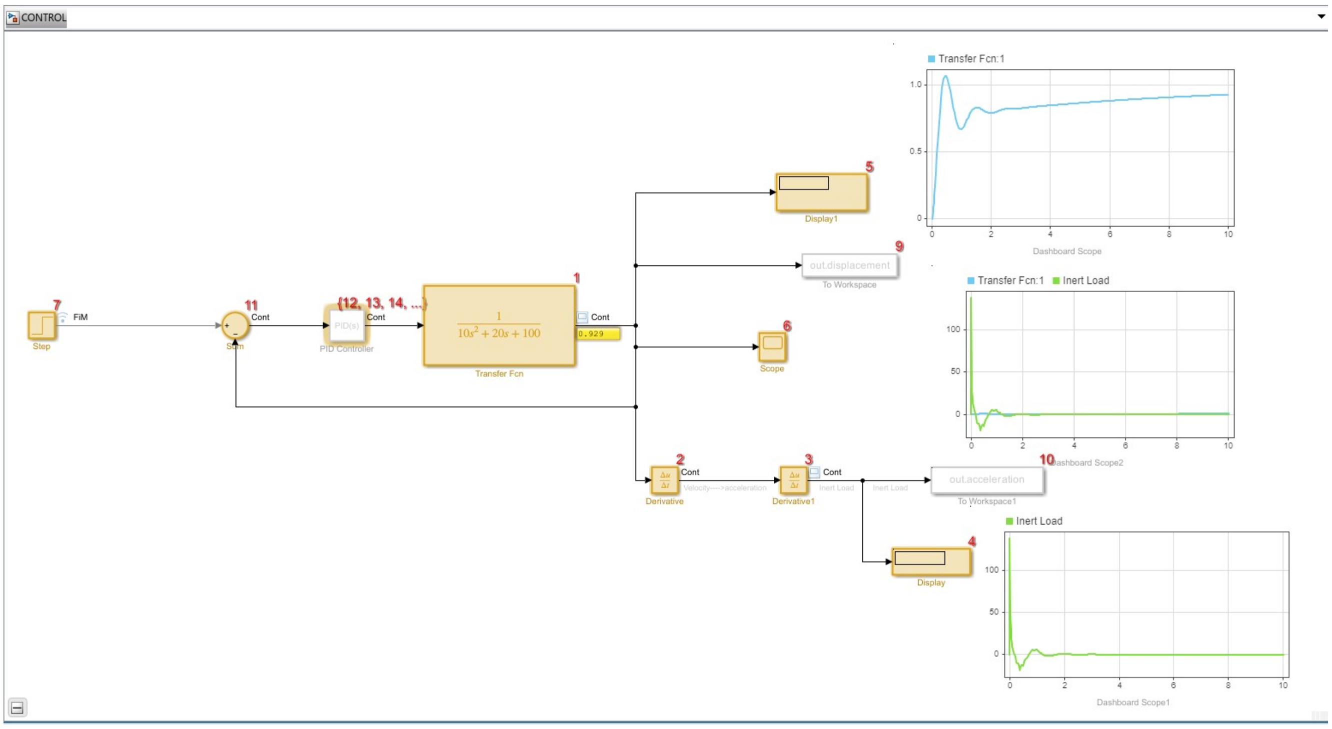


Figure 6: Exectution Order of simulation

The simulation was conducted in Simulink with the following components:

- **Step Input**: Unit step applied at $t = 0$.
- **Sum Block**: Computes the error signal.
- **PID Controller**: Configured for P, PI, PD, or PID control.
- **Transfer Function**: $\frac{1}{10s^2+20s+100}$.
- **Derivative Blocks**: Calculate acceleration.
- **Output**: Data exported via Scope and To Workspace blocks.

$$\begin{tikzcd}
{} \arrow[r, "input"] & {}_{-} \arrow[r, "e(t)"] & \text{PID Controller} \arrow[r, "u(t)"] & \text{Transfer Fcn} \arrow[r, "y(t)"] & {} \arrow[dl] \\
& & \text{Derivative} \arrow[ul, "y'(t)"] & &
\end{tikzcd}$$

Each simulation ran for 10 seconds using a fixed-step solver (ode4). Controllers were tuned manually, and performance metrics were extracted using MATLAB's `stepinfo` function, alongside maximum acceleration values computed from the second derivative of the output.

10 Controller Design and Results

10.1 P Controller

The P controller was tuned with different values of K_p to observe the system response. The results are summarized below:

Table 1: P Controller Results

Run	K_p	**Classification**	**Max Acceleration (m/s²)**
1	50	Underdamped with $\zeta = 0.272$	2.9999
2	100	Underdamped with $\zeta = 0.243$	5.9999
3	200	Underdamped with $\zeta = 0.192$	11.9999
4	300	Underdamped with $\zeta = 0.168$	17.9999

10.2 PI Controller

The PI controller was tuned with different values of K_p and K_i. The results are summarized below:

Table 2: PI Controller Results

Run	K_p	K_i	**Classification**	**Max Acceleration (m/s²)**
5	50	10	Overdamped ($\zeta > 1$)	2.9999
6	100	10	Underdamped with $\zeta = 0.634$	5.9999
7	200	10	Underdamped with $\zeta = 0.308$	11.9999
8	200	50	Underdamped with $\zeta = 0.518$	12.0000

10.3 PD Controller

The PD controller was tuned with different values of K_p and K_d. The results are summarized below:

Table 3: PD Controller Results

Run	K_p	K_d	**Classification**	**Max Acceleration (m/s²)**
9	50	5	Overdamped ($\zeta > 1$)	3.1059
10	50	10	Overdamped ($\zeta > 1$)	3.0000
11	100	10	Overdamped ($\zeta > 1$)	2.9999
12	100	5	Underdamped with $\zeta = 0.297$	32.9974

10.4 PID Controller

The PID controller was tuned with different values of K_p, K_i, and K_d. The results are summarized below:

Table 4: PID Controller Results

Run	K_p	K_i	K_d	**Classification**	**Max Acceleration (m/s²)**
13	50	10	5	Underdamped with $\zeta = 0.292$	62.9973
14	100	10	5	Underdamped with $\zeta = 0.301$	65.9974
15	50	5	20	Underdamped with $\zeta = 0.301$	65.9974
16	300	20	10	Underdamped with $\zeta = 0.221$	122.9972

11 Results

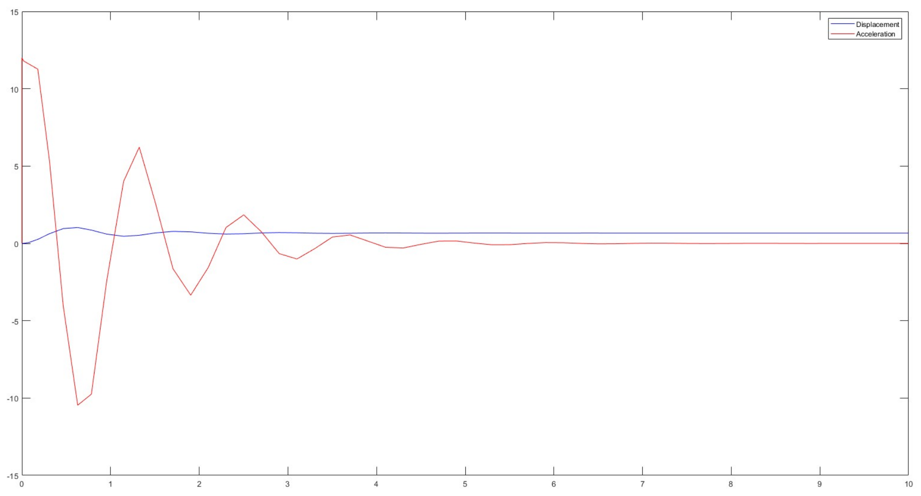


Figure 7: P controller

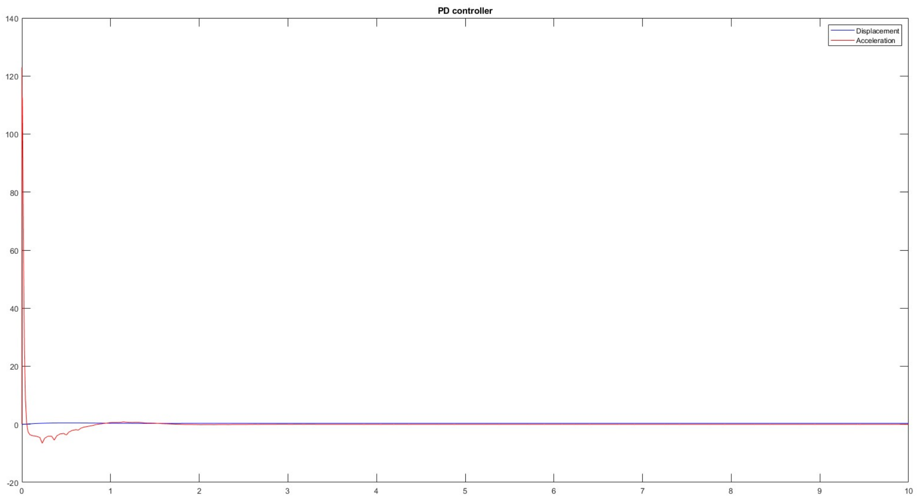


Figure 8: PD controller

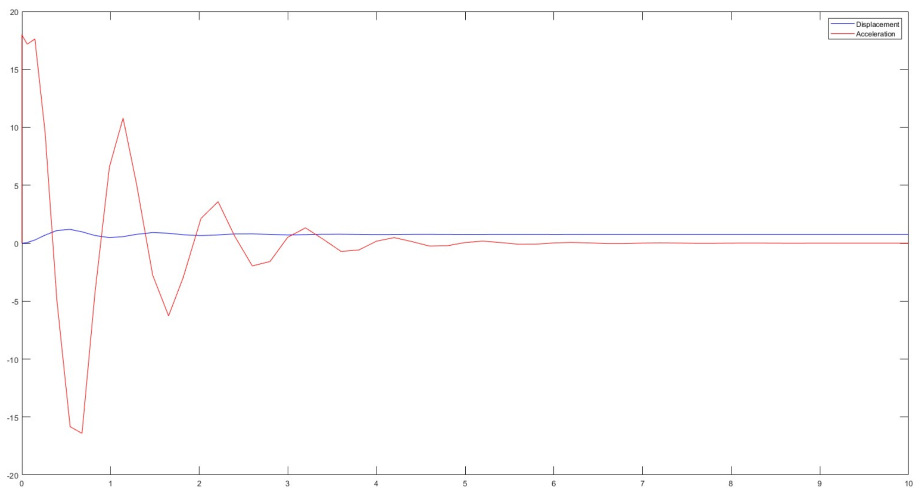


Figure 9: PI controller

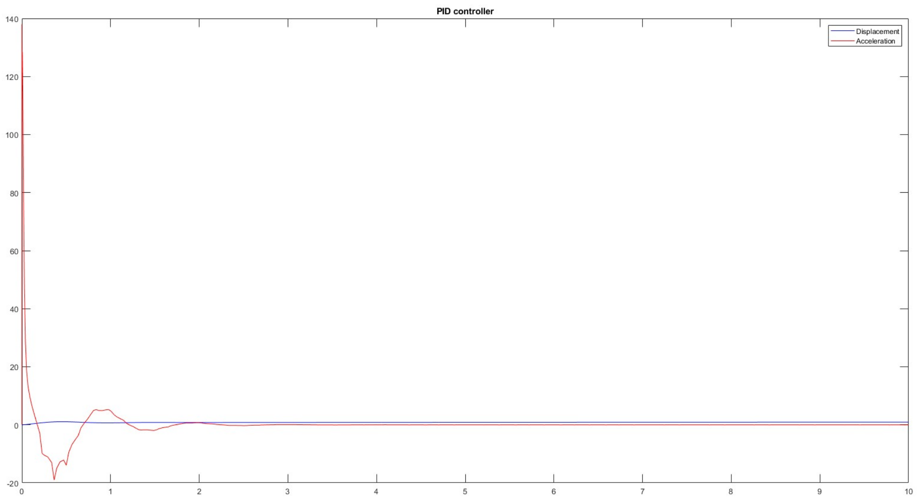


Figure 10: PID controller

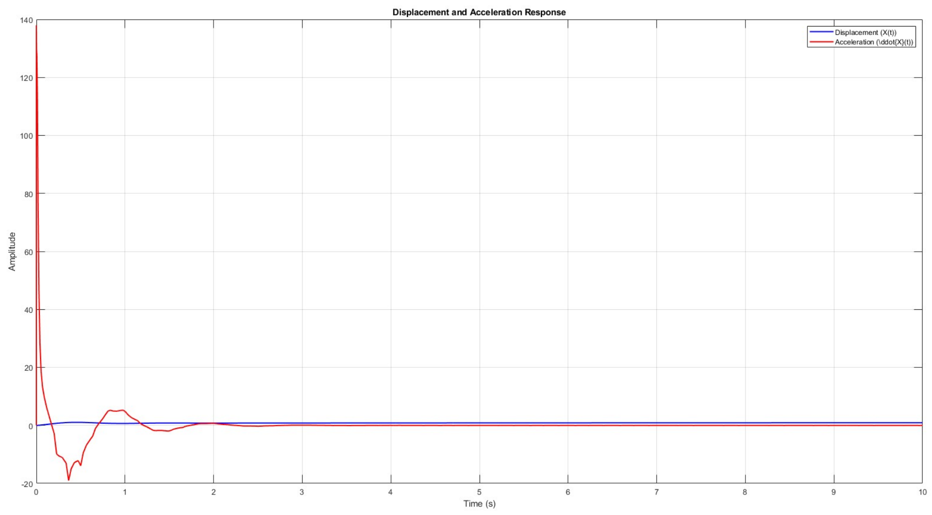


Figure 11: acceleration vs displacement

The table below summarizes the classification and maximum acceleration for each of the 20 simulation runs, based on the step response data.

Table 5: Run Data

Run	RiseTime	SettlingTime	SettlingMin	SettlingMax	Overshoot	Peak	Max Acceleration
1	0.3447	3.6249	0.2741	0.4702	41.0765	0.4702	2.9999
2	0.2835	3.7665	0.3841	0.7275	45.5102	0.7275	5.9999
3	0.2183	3.6382	0.4606	1.0272	54.0665	1.0272	11.9999
4	0.1871	3.6426	0.4782	1.1887	58.5043	1.1887	17.9999
5	7.2070	9.4360	0.5978	0.6583	0	0.6583	2.9999
6	0.3952	9.1012	0.4139	0.7506	7.6128	0.7506	5.9999
7	0.2485	8.1694	0.4694	1.0377	36.2030	1.0377	11.9999
8	0.3040	8.4255	0.5000	1.0790	14.9257	1.0790	12.0000
9	4.9117	8.0958	0.8834	0.9789	0	0.9789	3.1059
10	6.9311	9.3244	0.7459	0.8269	0	0.8269	3.0000
11	0.5824	9.3517	0.3094	0.5220	0	0.5220	2.9999
12	0.3038	2.7818	0.2903	0.4590	37.6889	0.4590	32.9974
13	0.2481	2.6149	0.2995	0.4610	38.3078	0.4610	62.9973
14	0.2557	2.3806	0.4397	0.6855	37.1027	0.6855	65.9974
15	0.2557	2.3806	0.4397	0.6855	37.1027	0.6855	65.9974
16	0.1591	1.9031	0.3094	0.4967	49.0042	0.4967	122.9972
17	7.2934	9.4419	0.5937	0.6572	0	0.6572	54.7249
18	0.3945	9.1056	0.4470	0.7253	4.0627	0.7253	59.7058
19	0.3340	9.3664	0.3332	0.5184	0	0.5184	122.9972
20	0.2253	8.1519	0.6701	1.0660	14.6958	1.0660	137.9975

11.1 Explanation of Step Response Data

The step response metrics for each run include:

- **RiseTime**: Time (in seconds) for the response to rise from 10% to 90% of its final value.
- **TransientTime**: Time (in seconds) for the response to reach and stay within 2% of the final value.
- **SettlingTime**: Synonymous with TransientTime in this context, indicating when the response stabilizes within 2%.

Table 6: Summary of Simulation Runs: Classification and Maximum Acceleration

Run	**Classification**	**Max Acceleration ($\mathbf{m/s^2}$)**
1	Underdamped with $\zeta = 0.272$	2.9999
2	Underdamped with $\zeta = 0.243$	5.9999
3	Underdamped with $\zeta = 0.192$	11.9999
4	Underdamped with $\zeta = 0.168$	17.9999
5	Overdamped ($\zeta > 1$)	2.9999
6	Underdamped with $\zeta = 0.634$	5.9999
7	Underdamped with $\zeta = 0.308$	11.9999
8	Underdamped with $\zeta = 0.518$	12.0000
9	Overdamped ($\zeta > 1$)	3.1059
10	Overdamped ($\zeta > 1$)	3.0000
11	Overdamped ($\zeta > 1$)	2.9999
12	Underdamped with $\zeta = 0.297$	32.9974
13	Underdamped with $\zeta = 0.292$	62.9973
14	Underdamped with $\zeta = 0.301$	65.9974
15	Underdamped with $\zeta = 0.301$	65.9974
16	Underdamped with $\zeta = 0.221$	122.9972
17	Overdamped ($\zeta > 1$)	54.7249
18	Underdamped with $\zeta = 0.714$	59.7058
19	Overdamped ($\zeta > 1$)	122.9972
20	Underdamped with $\zeta = 0.521$	137.9975

- **SettlingMin/SettlingMax**: Minimum and maximum values of the response after settling.
- **Overshoot**: Percentage by which the peak exceeds the final value.
- **Undershoot**: Percentage below the initial value (0 in all cases here).
- **Peak**: Maximum value reached by the response.
- **PeakTime**: Time (in seconds) at which the peak occurs.

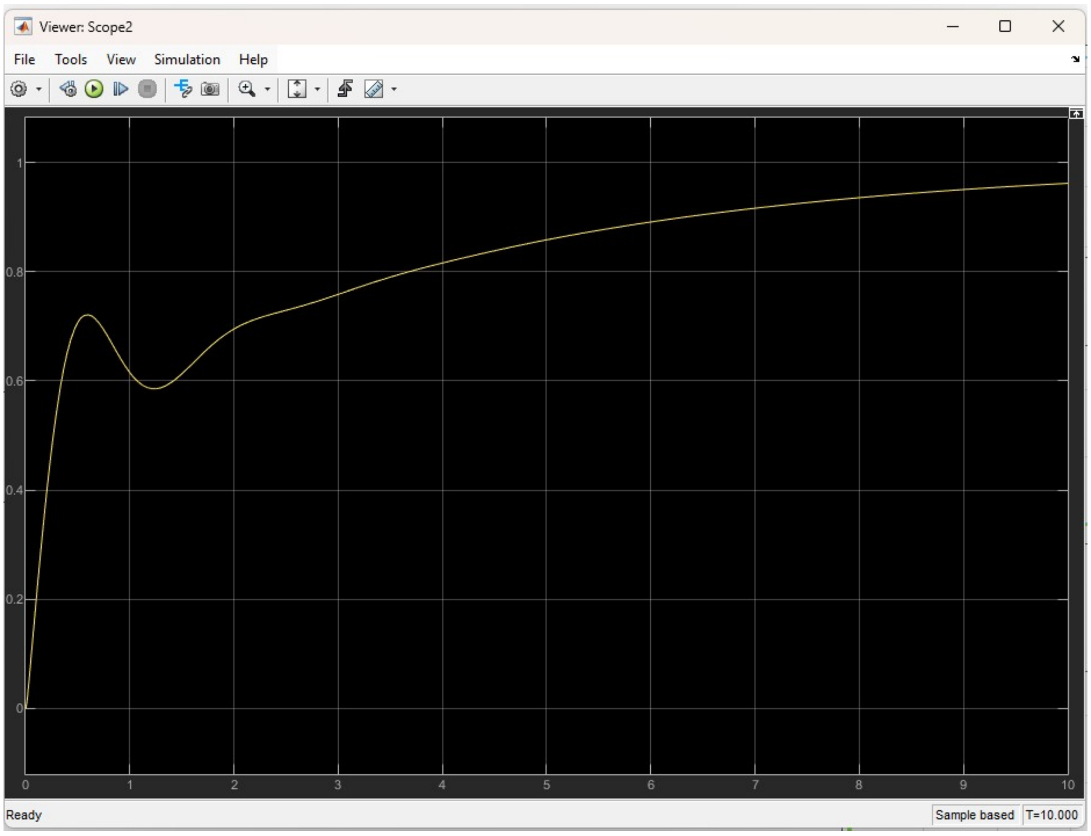


Figure 12: scope of response after applying PID

12 Discussion

The results reveal distinct trends:

- **Underdamped Runs** (e.g., 1–4, 6–8, 12–16, 18, 20) exhibit faster settling times (e.g., 1.9031 s in Run 16) but higher maximum accelerations (up to 137.9975 m/s^2 in Run 20), indicating significant oscillations and inertia loads.
- **Overdamped Runs** (e.g., 5, 9–11, 17, 19) show no overshoot, longer settling times (e.g., 9.4419 s in Run 17), and generally lower accelerations (e.g., 2.9999 m/s^2 in Run 5), though some exceptions exist (e.g., 122.9972 m/s^2 in Run 19).

An intriguing observation is that higher damping ratios in underdamped systems (e.g., Run 18, $\zeta = 0.714$) can yield lower accelerations (59.7058 m/s^2) compared to less damped runs (e.g., Run 16, $\zeta = 0.221$, 122.9972 m/s^2). This suggests a complex trade-off between damping, settling time, and inertia load, potentially influenced by controller tuning beyond simple second-order assumptions.

Simulink PID - solver results

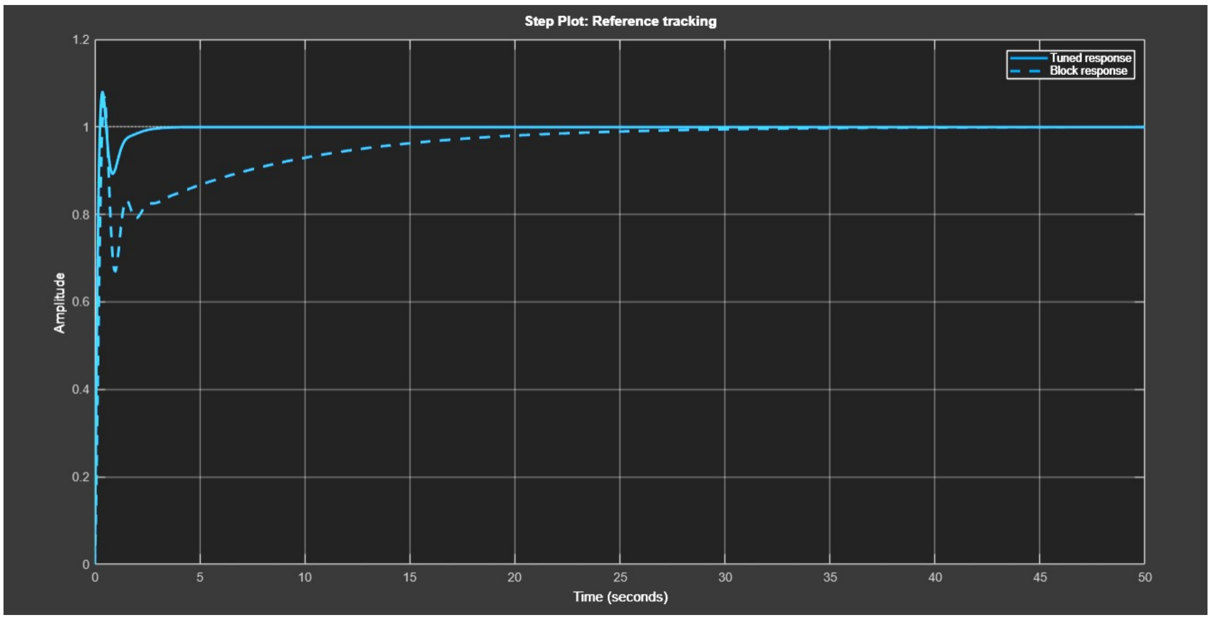
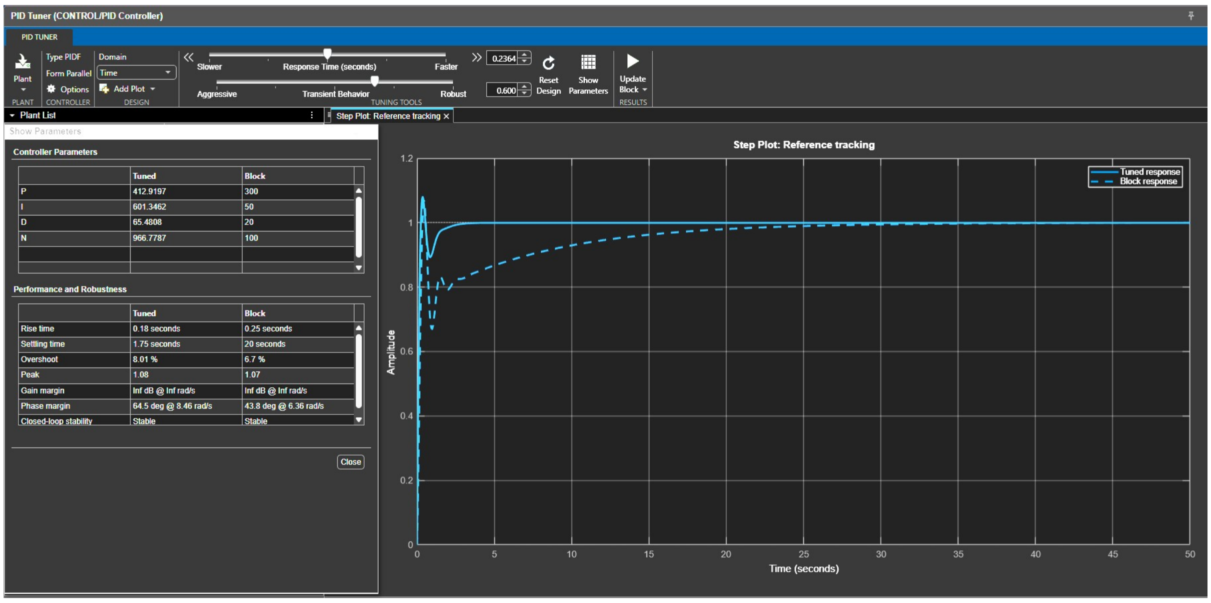


Figure 13: PID SOLVER RESULTS



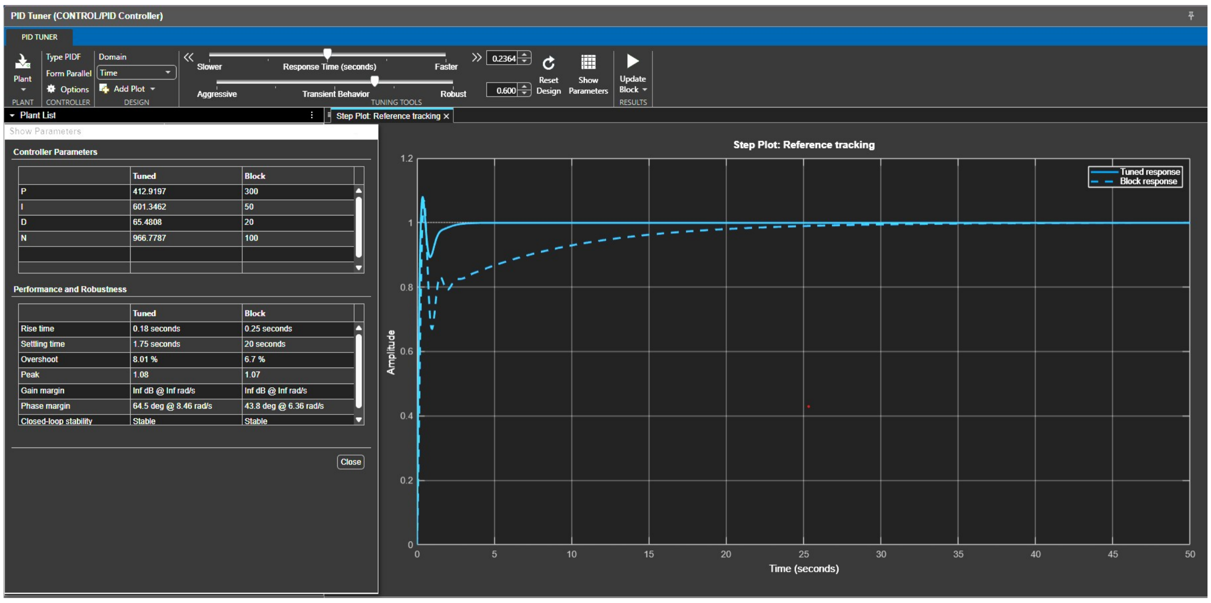
Controller Parameters

	Tuned	Block
P	412.9197	300
I	601.3462	50
D	65.4808	20
N	966.7787	100

Performance and Robustness

	Tuned	Block
Rise time	0.18 seconds	0.25 seconds
Settling time	1.75 seconds	20 seconds
Overshoot	8.01 %	6.7 %
Peak	1.08	1.07
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	64.5 deg @ 8.46 rad/s	43.8 deg @ 6.36 rad/s
Closed-loop stability	Stable	Stable

Figure 14: TUNING PID THROUGH SLIDER



Show Parameters

Controller Parameters

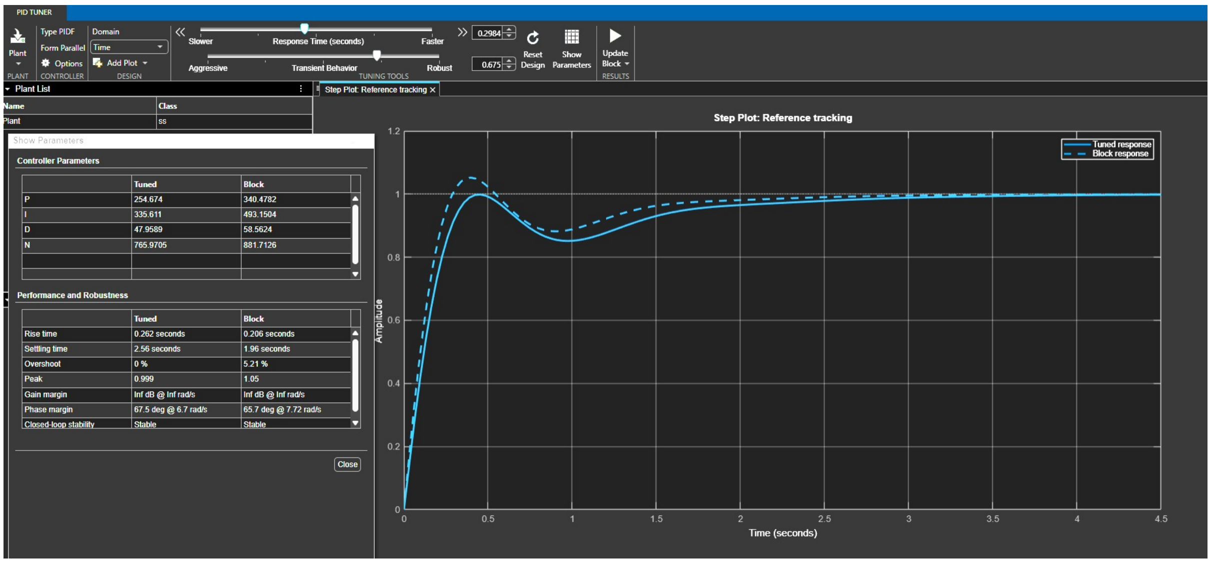
	Tuned	Block
P	412.9197	300
I	601.3462	50
D	65.4808	20
N	966.7787	100

Performance and Robustness

	Tuned	Block
Rise time	0.18 seconds	0.25 seconds
Settling time	1.75 seconds	20 seconds
Overshoot	8.01 %	6.7 %
Peak	1.08	1.07
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	64.5 deg @ 8.46 rad/s	43.8 deg @ 6.36 rad/s
Closed-loop stability	Stable	Stable

Close

Figure 15: TUNEING PARAMETERS BY SOLVER VARIABLE



Name	Class
Plant	ss

Show Parameters

Controller Parameters

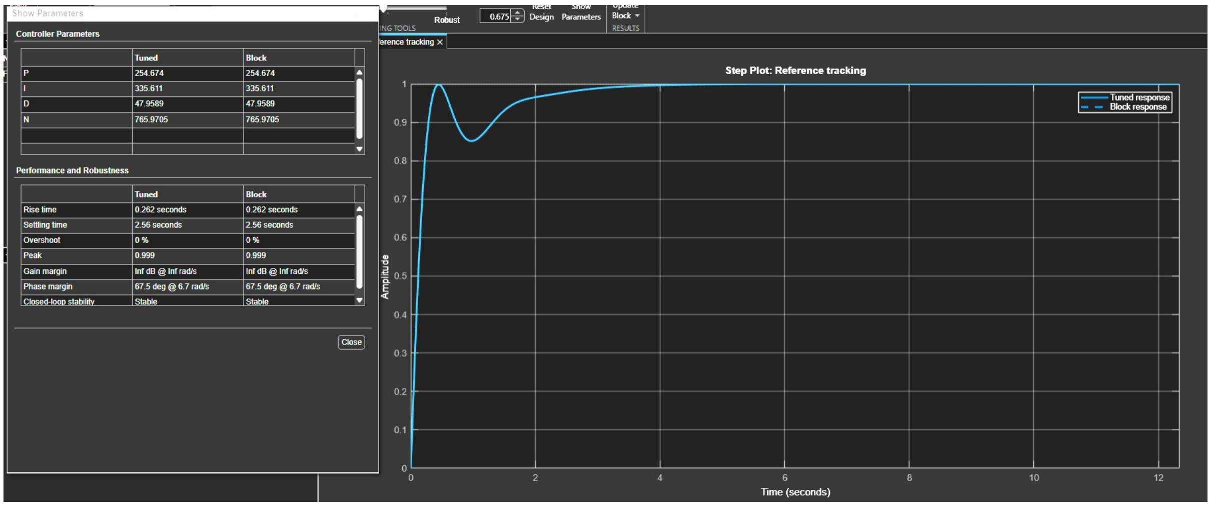
	Tuned	Block
P	254.674	340.4782
I	335.611	493.1504
D	47.9589	58.5624
N	765.9705	881.7126

Performance and Robustness

	Tuned	Block
Rise time	0.262 seconds	0.206 seconds
Settling time	2.56 seconds	1.96 seconds
Overshoot	0 %	5.21 %
Peak	0.999	1.05
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	67.5 deg @ 6.7 rad/s	65.7 deg @ 7.72 rad/s
Closed-loop stability	Stable	Stable

Close

Figure 16: REDUCING OVERSHOOT AND SETTLING TIME



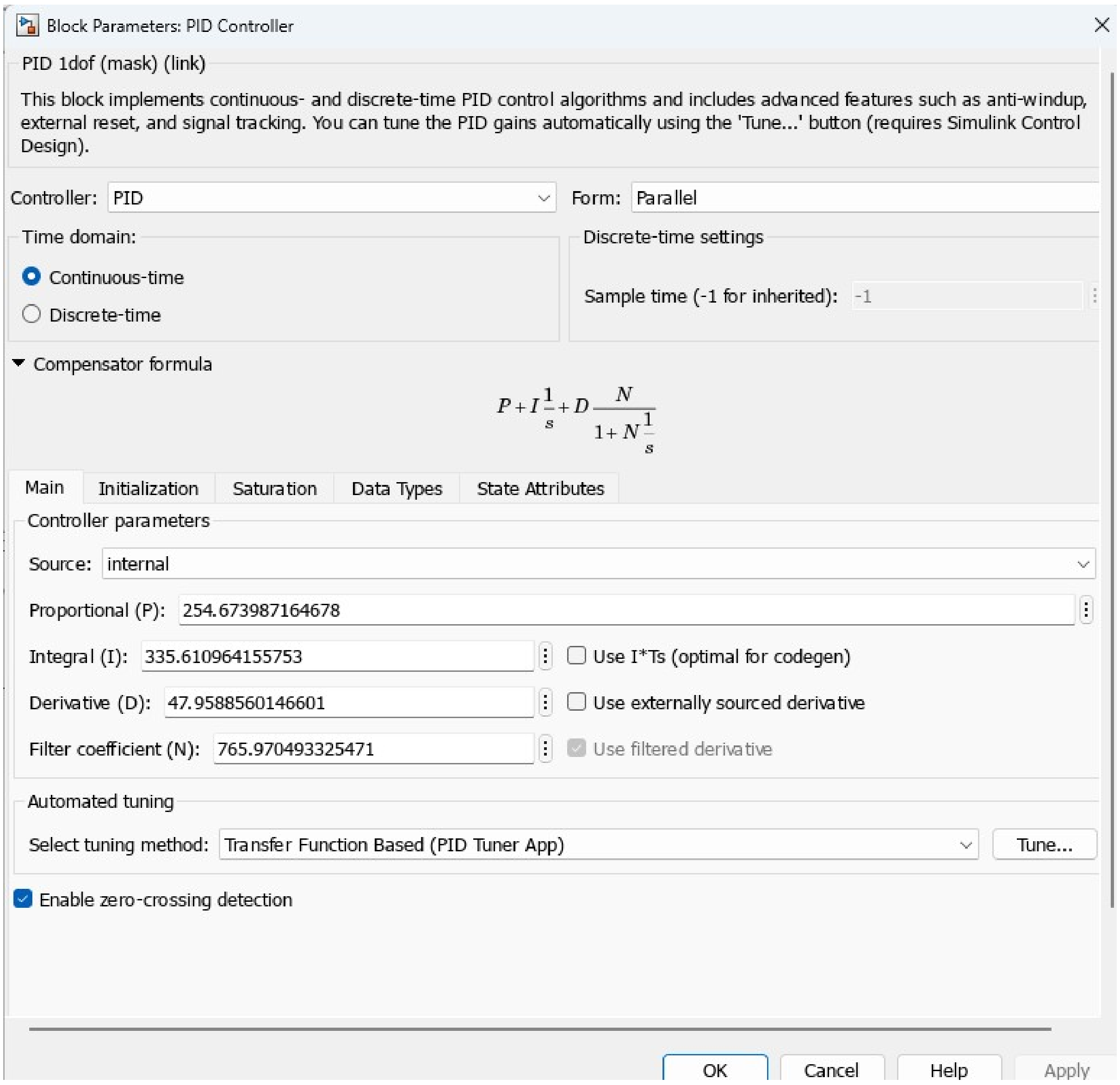
Controller Parameters

	Tuned	Block
P	254.674	254.674
I	335.611	335.611
D	47.9589	47.9589
N	765.9705	765.9705

Performance and Robustness

	Tuned	Block
Rise time	0.262 seconds	0.262 seconds
Settling time	2.56 seconds	2.56 seconds
Overshoot	0 %	0 %
Peak	0.999	0.999
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	67.5 deg @ 6.7 rad/s	67.5 deg @ 6.7 rad/s
Closed-loop stability	Stable	Stable

Figure 17: FINAL TUNED AND UPDATED PARAMETRS.



PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID   Form: Parallel

Time domain:

[✓ Continuous-time]

Discrete-time

Discrete-time settings

Sample time (-1 for inherited): -1

Compensator formula

$$P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

Controller parameters

Source: internal

Proportional (P): 254.673987164678

Integral (I): 335.610964155753   Use I*Ts (optimal for codegen)

Derivative (D): 47.9588560146601   Use externally sourced derivative

Filter coefficient (N): 765.970493325471   [✓ Use filtered derivative]

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App)

[✓ Enable zero-crossing detection]

Figure 18: FINAL PARAMETERS

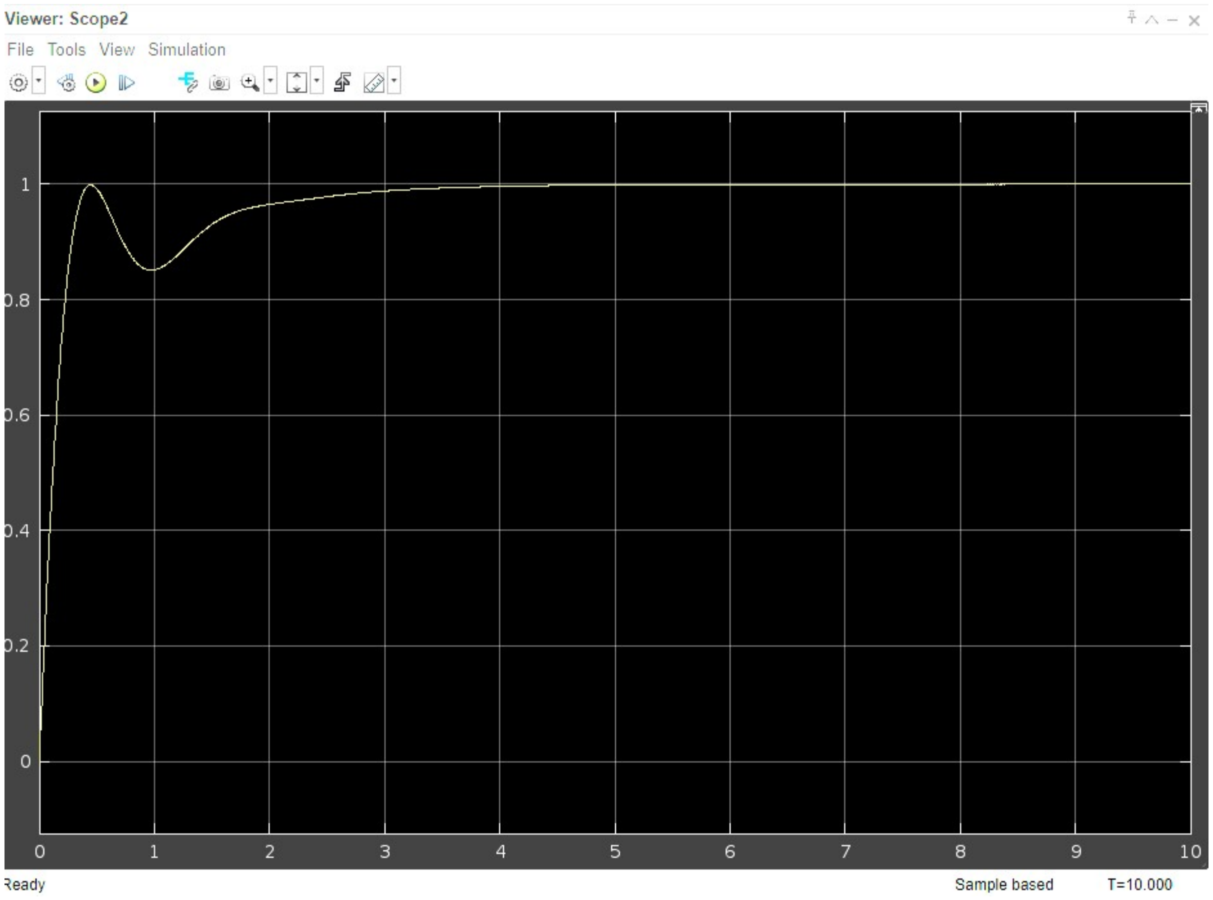


Figure 19: TUNED RESPONSE

13 Conclusion

In this assignment, we designed and tuned various controllers (P, PI, PD, and PID) for an actuator system. By analyzing the step response characteristics, we determined the damping type and identified the best parameters for each controller. The PID controller with the optimized parameters BY SOLVER provided the best performance in terms of overshoot and settling time.