Mechatronic System Simulation and Control (EEN1048)

Assignment

Control Scheme Analysis of an Electric Power Steering System

Student Name:	**Student ID:**	**Programme:**
Ivan McCauley	21355886	ME4

Table of Contents

1. Introduction

This report presents an in-depth analysis of a PID-based control scheme used in an Electric Power Steering (EPS) system, based on the paper "Improved PID Control Design for Electric Power Steering DC Motor"[1] by A. Turan. The study looks at an improved PID tuning method that enhances system stability and responsiveness compared to traditional PID tuning techniques. The report is structured into three main sections: Selection and Justification, Review and Analysis, and Reflection. The selection section outlines the reason behind choosing this control scheme, followed by a review and analysis of the system, including simulations. Finally, the report also reflects on the key learning outcomes and challenges faced throughout the assignment.

Electric Power Steering (EPS) has become a popular alternative to traditional Hydraulic Power Steering (HPS) systems. Instead of relying on hydraulics, EPS uses an electric motor to assist with steering, making it more efficient, responsive, and controllable. [1] The shift towards EPS is largely driven by the need for better energy efficiency and overall improvements in vehicle safety.

2. Part A: Selection, Justification & Research Trail

2.1 Justification

2.1.1 What type of physical system does the model represent and how is that relevant to the module?

The selected control scheme represents an Electric Power Steering (EPS) system, a critical component in automotive engineering. EPS systems replace traditional Hydraulic Power Steering (HPS) by using an electric motor to provide steering assistance based on the input (read by a sensor) from the driver at the steering wheel.

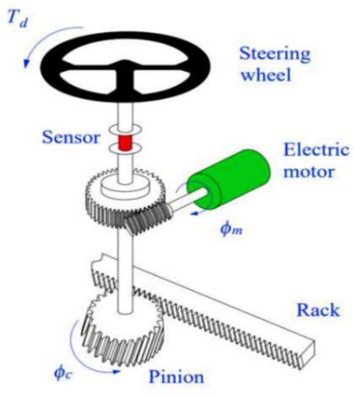

Figure 1: EPS system[1]

This system is directly relevant to the Mechatronic System Simulation and Control (EEN1048) module as it integrates mechanical, electrical, and control system principles EPS systems involve dynamic modelling, feedback control, and real-time system response analysis, which align with the learning outcomes of the module. The use of PID control to regulate motor torque and assist with steering involves concepts in closed-loop control and system stability, making it a suitable study for EEN1048. Analyzing the EPS system enhances understanding of key control concepts, including controllability, observability, and response characteristics such as overshoot, settling time and steady-state values.

Also, modelling the EPS system using MATLAB and Simulink is highly relevant to the module. Simulink provides a visual representation of system dynamics, enabling simulation of PID tuning and transient response characteristics. This aligns with the skills required in control system design, testing, and validation, making it well suited for this assignment.

2.1.2 How was the model derived?

The model for the Electric Power Steering (EPS) system was derived by incorporating both the mechanical dynamics of the steering system and the electrical behaviour of the DC motor. The model is derived using a structured mathematical approach, where the system is represented by its transfer function.

Transfer Function Representation of the EPS System

The dynamic behaviour of the EPS motor and steering system is represented using a third-order transfer function, which was obtained by analyzing the torque balance equations for the system. The general form of the open loop transfer function of the EPS system is given as:

$$G(s) = \frac{1.26s^2 + 27.86s + 52734.82}{1.47s^3 + 112.27s^2 + 38552.43s + 50915.38}$$ [1]

This function describes how the input voltage to the motor translates into angular displacement of the steering column A third-order transfer function was chosen because it accurately captures the dominant system dynamics without introducing unnecessary complexity. Higher-order models may be more computational without necessarily improving the PID performance that well.

PID Controller Model Formulation

The PID controller for the EPS system was designed using an improved tuning method, which optimizes the proportional (K_p), integral (K_i), and derivative (K_d) gains to enhance system performance. The equation for the controller is given by :

$$C(s) = \frac{K_d s^2 + K_p s + K_i}{s}$$ [1]

The closed-loop system with PID control is represented as:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}$$ [1]

where T(s) is the closed-loop transfer function of the system

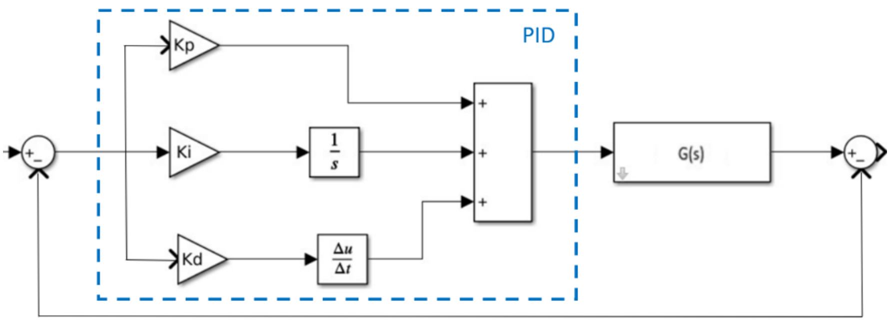

Figure 2: Block Diagram of closed loop system.

2.1.3 How was the model parameterized?

The EPS model was set up by assigning values to key electrical, mechanical, and control system parameters. These values were sourced from manufacturer specifications, experimental measurements, and system identification methods.

1. System Model
The system model represents the EPS system's behaviour, including motor dynamics, the steering rack, and vehicle interaction.

- **G(s):** The overall system model, capturing both mechanical and electrical properties like motor dynamics and steering mechanism.
- **GN(s):** The numerator of the transfer function of the overall system model G(s).
- **GD(s):** The denominator of the transfer function of the overall system model G(s).

2. PID Controller
The controller adjusts steering assistance based on feedback error between desired and actual steering angles.

- **C(s):** The PID controller that processes feedback to adjust steering input.
- $\boldsymbol{K_p}$**:** Proportional gain '...to improve the transient response rise time and settling time'[2]
- $\boldsymbol{K_i}$**:** Integral gain '...to improve steady-state response.' [2]
- $\boldsymbol{K_d}$**:** Derivative gain '...to improve the transient response by way of predicting error will occur in the future' [2] based on the rate of change.

3. Closed-Loop System
This models the system behavior with feedback applied to minimize error.

- **T(s):** The closed-loop system transfer function, combining the controller and system model.
- $\boldsymbol{T_N(s)}$ **and** $\boldsymbol{T_D(s)}$**:** The numerator and denominator of T(s).
- **ζ (Damping Ratio):** Determines the speed of oscillation decay, ensuring stability and transient response.
- $\boldsymbol{W_n}$ **(Natural Frequency):** Defines system oscillation frequency without damping.
- **Δ(s):** The target polynomial that shapes system dynamics to meet performance goals.

4. Optimization and Tuning
Optimization algorithms fine-tune parameters to achieve optimal performance.

- **GA (Genetic Algorithm):** '...a computational search technique for finding approximate solutions to optimize models and search problems.' [3]
- **ALO (Ant Lion Optimization):** Another optimization technique for tuning PID parameters to improve error minimization and stability.
- $\boldsymbol{Err_{min}}$**:** The minimum achievable error.

5. Residuals and Error Characteristics

Residuals and error characteristics ensure system accuracy and performance.

- **R(s):** Residue polynomial used for stability analysis after simplifying the system transfer function.
- **m:** The degree difference between the numerator and denominator polynomials.
- **x, y:** Coefficients related to proportional, integral, and derivative errors.

6. System Performance Metrics

Several metrics assess whether the EPS system meets performance goals.

- **SSE (Steady State Error):** Measures the difference between desired and actual output as time approaches infinity, which ensures a precise steering angle.
- **RPM (Revolutions Per Minute):** Indicates steering motor speed, affecting system responsiveness.
- **LQR (Linear Quadratic Regulator):** '...a stable control method based on specified performance weightings rather than eigenvalue placement.' [4]
- **LQG (Linear Quadratic Gaussian):** Extends LQR to account for noisy measurements.
- **MPC (Model Predictive Control):** '...is an advanced method of process control that is used to control a process while satisfying a set of constraints.' [5]

By carefully parameterizing the system model, PID controller, and closed-loop dynamics, we can achieve precise control over system performance.

Symbol	Quantity
CU	Control unit
PWM	Pulse width modulation
GA	Genetic algorithm
SSE	Steady state error
RPM	Revolutions Per Minute
LQR	Linear Quadratic Regulator
LQG	Linear Quadratic Gauss
MPC	Model Predictive Control
$G(s)$	System model
$G_N(s)$	Numerator of G(s)
$G_D(s)$	Denominator of G(s)
$C(s)$	Contreller system
$T(s)$	Closed loop system
ALO	Ant lion optimization
$T_N(s)$	Numerator of T(s)
$T_D(s)$	Denominator of T(s)
ζ	Damping ratio
w_n	Natural frequency
$R(s)$	Residue polynomial
$\Delta(s)$	Target polynomial of the closed loop system
m	Degree difference
Err_{min}	Minimum error
x,y	Coefficients of the total error
k_p	Ratio gain
k_i	Integral gain
k_d	Differential gain

Figure 3: Notations of parameters [1]

2.1.4 Why did you pick a control scheme with this model?

I chose this model because electric power steering (EPS) is a widely used system in modern vehicles, and understanding how it's controlled is both practical and relevant. As someone with an interest in cars, I found it particularly interesting to explore how EPS works and how control systems improve vehicle performance. EPS helps make steering smoother, more efficient, and safer, which is why it's so important in modern cars. Since controlling steering position and torque is a big part of how EPS works, I wanted to see how a PID-based approach helps keep it stable and responsive.

Another reason I chose this model is that it's already been validated in the IEEE paper, meaning I can compare my results with published data to check accuracy. The paper also provides realistic system parameters, like motor resistance, inertia, and damping, making it easier to implement in MATLAB and Simulink.

Overall, I picked this model because it's realistic, well-documented, and can be applied to the learning objectives of EEN1048. It also allowed me to combine my what I learn about in lectures with my personal interest in cars, giving me a better understanding of how real-world control systems are designed.

2.1.5 What is the format/order of the model?

The format of the EPS system is a third order transfer function that captures the dynamics of the system, including motor, steering torque, and assist control.

$$G(s) = \frac{1.26s^2 + 27.86s + 52734.82}{1.47s^3 + 112.27s^2 + 38552.43s + 50915.38}$$ [1]

2.1.6 What aspects of the model make it easy/difficult to work with for analysis/simulation/control design?

- **Easy Aspects:**

The use of transfer functions allows straightforward analysis using MATLAB tools such as bode plots, root locus, and step response analysis.

Integral, derivative and gain blocks are available in SIMULINK for modelling the PID controller

- **Difficult Aspects:**

The nonlinearities of mechanical components such as torsion bars and steering torque feedback can make designing the controller more difficult.

Factors such as road conditions can have a big impact on how the system behaves.

2.1.7 Is the model controllable/observable?

The open loop transfer function:

$$G(s) = \frac{1.26s^2 + 27.86s + 52734.82}{1.47s^3 + 112.27s^2 + 38552.43s + 50915.38}$$ [1]

needed to be converted to state space format so its controllability matrix could be attained. This was done using the tf2ss() command in MATLAB. From that the controllability matrix was found using ctrb(A,B).

There are two ways of checking controllability, one method is checking the rank of the controllability matrix using the rank() function, if the rank is the same as the smallest dimension of the controllability matrix, then the system is controllable. The other method is simply checking the determinant of the controllability matrix, if the controllability matrix is non-zero then its controllable.

```
% Initializing TF num and den
Gs_num = [1.26 27.86 52734.82]
Gs_den = [1.47 112.27 38552.43 50915.38]
[A,B,C,D] = tf2ss(Gs_num, Gs_den)
% Controllability matrix
CO = ctrb(A,B)
% Checking rank of CO
CO_rank = rank(CO)
% Checking determinant of CO
CO_det = det(CO)
```

Figure 4: MATLAB code for SS conversion and controllability

The rank was found to be 3, and the determinant was found to be 1, showing that the system is controllable.

Similarly, for determining observability, the observability matrix was found from the state space representation using obsv(A,C). The rank and determinant of the observability matrix was then calculated.

```
% Observability matrix
OB = obsv(A,C)
OB_rank = rank(OB)
OB_det = det(OB)
```

Figure 5: MATLAB code for determining observability

The rank and determinant were found to be 3 and -1.0088e+13 respectively, showing that the system is observable.

2.1.8 Is the model stable/unstable/marginally stable/conditionally stable etc.? Why?

To determine the stability of the system, the poles of G(s) can be found using the pole() and pzmap() function in MATLAB as follows:

```
% Stability analysis
Gs = tf([Gs_num],[Gs_den])
poles = pole(Gs)
pzmap(Gs)
```

Figure 6: Stability analysis of the model

The poles were found to be:

```
poles =

  -37.524 +     157.22i
  -37.524 -     157.22i
  -1.3257 +          0i
```

Figure 7: Poles of G(s)

The pole-zero map is shown below, with all poles in the right hand of the s-plane:

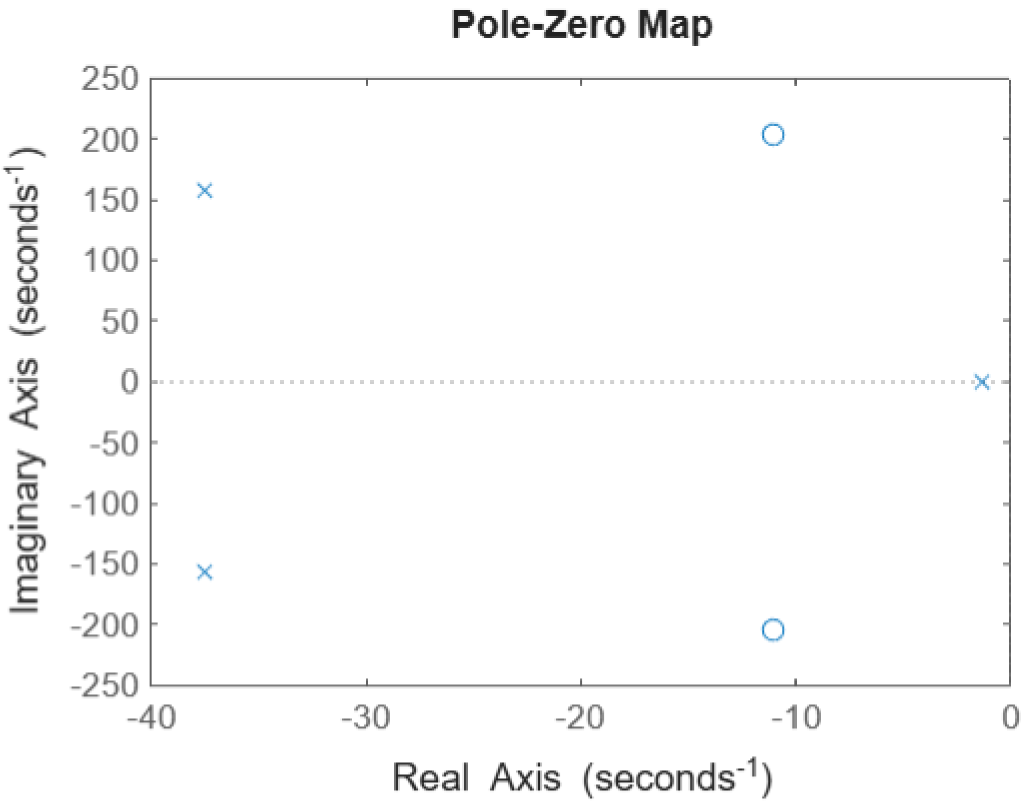

Figure 8: Pole-zero map of the model

All poles have negative real parts therefore the system is stable, meaning that the system's response will eventually settle to a steady state.

2.1.9 How does this source facilitate the rest of the assessment?

For part B (Review Analysis & Simulation):

- The model equations and parameters allow for it to be directly implemented in MATLAB/SIMULINK.
- The performance metrics (rise time, overshoot, settling time, steady-state error) help for comparison with experimental results.

For part C (Reflection):

- The study gives multiple tuning methods for the model, so a detailed evaluation can be carried out.

2.2 Research Trail

2.2.1 Reference 1:

Marouf, M. Djemai, C. Sentouh, and P. Pudlo, ''A new control strategy of an electric-power-assisted steering system,'' IEEE Trans. Veh. Technol., vol. 61, no. 8, pp. 3574–3589, Oct. 2012

This paper looks at an alternative control strategy for EPS, focusing on steering stability, torque estimation, and compensation techniques. It analyses how assist motor control and torque feedback influence vehicle handling. The study influenced the design of the improved PID controller in the selected IEEE paper, as it emphasized the importance of precise tuning for better steering feel and stability.

2.2.2 Reference 2:

X. Ma, Y. Guo, and L. Chen, ''Active disturbance rejection control for electric power steering system with assist motor variable mode,'' J. Franklin Inst., vol. 355, no. 3, pp. 1139–1155, Feb. 2018."

This study focuses on handling disturbances in EPS systems, such as vibrations. The research helped highlight the need for a more robust control method to keep steering smooth in different driving conditions. The IEEE paper took this into account when designing and tuning the PID controller, ensuring it could handle disturbances effectively.

2.2.3 Reference 3:

T. Yang, ''A new control framework of electric power steering system based on admittance control,'' IEEE Trans. Control Syst. Technol., vol. 23, no. 2, pp. 762–769, Mar. 2015

Yang's research looks at ways to improve steering responsiveness and energy efficiency using admittance control. The IEEE paper used this work as a reference when fine-tuning the PID controller, ensuring that the system remained stable while adapting to different driving conditions.

These references played a big role in designing the control strategy, disturbance handling, and the tuning process in the IEEE paper. They helped to make sure that the final PID design was stable and capable of handling real-world conditions.

3. Part B: Review, Analysis & Simulation

3.1 Control Scheme Review

The control scheme analysed in this report is a PID-based control system for an Electric Power Steering (EPS) system, as presented in the IEEE paper "Improved PID Control Design for Electric Power Steering DC Motor". The EPS system replaces traditional hydraulic power steering (HPS) by using an electric motor to provide steering assistance, improving efficiency and precision.

3.1.1 What is the control scheme required to achieve?

The primary goal of the PID control scheme in EPS is to ensure precise steering torque control while maintaining stability, responsiveness, and robustness. The control system is designed to:

- Reduce overshoot ($M_p = 5\%$) and settling time ($t_s = 0.21s$) and oscillations in steering response.
- Minimize steady-state error for accurate torque application.
- Improve system stability under varying road conditions and driver inputs.
- Provide consistent steering assistance while improving energy efficiency.

3.1.2 What constraints does the physical system impose?

The system is affected by many constraints that must be considered when designing the controller. Nonlinearities introduce uncertainties into the system, as factors like road conditions, friction, and external disturbances can impact steering behavior. The physical limitations of the motor and actuator also play a role, as high torque output or power consumption can lead to overheating and reduced efficiency. Lastly, sensitivity to environmental factors such as temperature changes, tire-road interactions, and unexpected disturbances can further affect the stability of the system.

3.1.3 Why was this controller used?

A PID controller was selected for EPS because it’s simple, effective and easy to simulate. The method used in this study improves performance by optimizing the proportional (K_p), integral (K_i), and derivative (K_d) gains for better stability and response compared to the traditional tuning methods. Also, there is a lot of information available on the PID controller as its commonly used in industry.

3.1.4 How does this controller work?

The controller works by constantly adjusting the motor torque based on the error signal between the desired and actual steering angle.

Proportional (P) Controller

The proportional (P) controller adjusts the torque based on the present error [2] by asjusting the output proportionally. The error signal is calculated by subtracting the actual steering angle from the desired steering angle. The output of the P controller is directly proportional to the error:

$$K_p * e(t)$$

where e(t) is the error signal. If we increase K_p we increase the response speed, however if its too high it can lead to overshoot and oscillations.

Integral (I) Controller

The integral (I) controller eliminates steady-state error by accumulating past errors. [2] The output of the I controller is given by:

$$K_i * \int e(t)dt$$

However, this controller can be 'sensitive to noise and prone to oscillations.' [6]

Derivative (D) Controller

The derivative (D) term predicts future errors by considering the rate of change. [2] The output of the D controller is given by:

$$K_d * \frac{d}{dt}e(t)$$

This controller dampens oscillations and therefore decreases settling time t_s.

All these controllers combined makes up the PID controller which makes sure that the EPS system has a fast response, stability, and precision with different driving conditions. The full output equation for the PID controller is given by:

$$K_p * e(t) + K_i * \int e(t)dt + K_d \cdot \frac{d}{dt}e(t)$$

3.1.5 How was the controller designed?

The PID controller for the EPS system was designed using an improved tuning method that optimizes proportional (K_p), integral (K_i), and derivative (K_d) gains to improve stability and performance compared to the traditional tuning methods. The goal was to reduce overshoot, improve transient response, and make the steering assist stable and responsive.

The design process involved several key steps:

1. **System Representation and Error Signal Generation**
The EPS system's model includes both the mechanical dynamics of the steering system and also the electrical behavior of the motor. The controller is put in series with the motor model, and a sum block is used to calculate the error signal. A sensor measures the actual steering angle, and the desired angle is subtracted from this to produce the error signal. This error is then fed into the PID controller, which produces action to adjust the motor output.

2. **Defining Performance Requirements**
The controller was designed to meet specific time-domain performance criteria, including:

    - **Settling Time (t_s):** The time required for the system to reach steady-state operation.
    - **Maximum Overshoot (M_p)**: The deviation from the response required.
    - **Steady-State Error (SSE)**: The final error between the desired and actual output.
    - **Rise Time (t_r)**: The time taken for the response to reach a certain percentage of its final value.

The design aimed to these factors while making sure that the system stays stable under different road conditions.

3. **PID Tuning Approach**
Unlike other methods like Ziegler-Nichols or Tyreus-Luyben, the paper used an improved tuning method. The tuning process involved:

    - Defining a target polynomial based on the desired performance values.

    $$\Delta(s) = s^2 + 2\zeta\omega n s + \omega_n^2$$

    Where ω_n is the natural frequency, and ζ is the damping ratio.

    - Adjusting the PID parameters to minimize error values (Err), calculated using a weighted combination of M_p and t_s:

    $$Err = x\, e_1 + y e_2$$ [1]

    where:

        - $e1 = \frac{M_p - M_{p,desired}}{M_p}$ [1] (overshoot error)
        - e2= $e2 = \frac{t_s - t_{s,desired}}{t_s}$ [1] (settling time error)
        - x and y are weighted coefficients ensuring a balance between overshoot and settling time. [1]

    - The range for the proportional gain K_p was chosen '...within the stable region of the control loop'. [1] PID parameters were iteratively adjusted within this region to minimize the total error Err.
    - The coefficients of the characteristic equation were matched to those of the target polynomial Δ(s), using the following equation:
    $$\left(\Delta\,(s) * R(s)\right)_{coeff} = TD(s)_{coeff}$$
    where $R(s)$ is the difference polynomial and $TD(s)$ is the characteristic equation of the system.

Design method	Parameters of PID controller		
	k_p	k_i	k_d
Ziegler-Nichols [1]	12.3536	0.034	0.0082
Tyreus – Luyben [1]	60	0.0278	0.0069
Haugen [1]	147.4193	0.07855	0.005641
IMC	15.11	85.17	0.2033
Proposed	217.3	4344.6	5.0314

Figure 9: All PID controller parameters[1]

4. **Forming the Closed-Loop System**
   The PID controller was integrated into a feedback control loop, which consists of:

   - The EPS system (plant)
   - The PID controller (C(s))
   - An input (from the driver)
   - A feedback signal (motor response)

   The closed-loop system transfer function is then:

$$T(s) = \frac{C(s)G(s)}{1+C(s)G(s)} \quad [1] \qquad \text{where: } C(s) = \frac{K_d s^2 + K_p s + K_i}{s} \quad [1]$$

3.1.6 How successful was the controller in achieving the objective of the control design?

The controller was designed with specific targets for the system's response:

- Settling time (t_s) to be minimized for faster stabilization.
- Maximum overshoot (M_p) to be low to maintain a smooth steering feel.
- Rise time (t_r) to be minimized to ensure a quick response to the driver’s input.
- Peak time (t_p) to be minimized to reduce the time for the system to reach its first peak value in the response.

The step response of the closed-loop system was analysed, and the results were compared with the other PID controllers that were designed (Ziegler-Nichols, Tyreus-Luyben, Haugen, and IMC). This method achieved the best performance with:

- **Settling time (t_s)**: 0.2291s
- **Rise time (t_r)**: 0.0101s
- **Overshoot (M_p)**: 4.999%
- **Peak time (t_p)**: 0.0512s
- A steady state error of zero.

Criteria	Ziegler-Nichols method [1]	Tyreus-Luyben method [1]	Haugen method [1]	IMC method	Proposed method
M_p (%)	0	21.341	6.989	9.4215	4.999
t_s (s)	1.268	0.477	0.3	0.52	0.2291
t_r (s)	0.037	0.013	0.013	0.0856	0.0101
t_p (s)	1.023	0.026	0.03	0.2252	0.0512
SSE	0.0065	0.015-0.003	0.0157	0.002	0

Figure 10: Performance results of PID controller for EPS system.[1]

These results clearly show that the controller successfully met the design objectives, with an overall improvement in all performance parameters compared to other methods.

Step Response

Below is the step response of the proposed design along with previous methods. It can be clearly seen that there is an improvement in the settling time, and overshoot compared to the other methods.

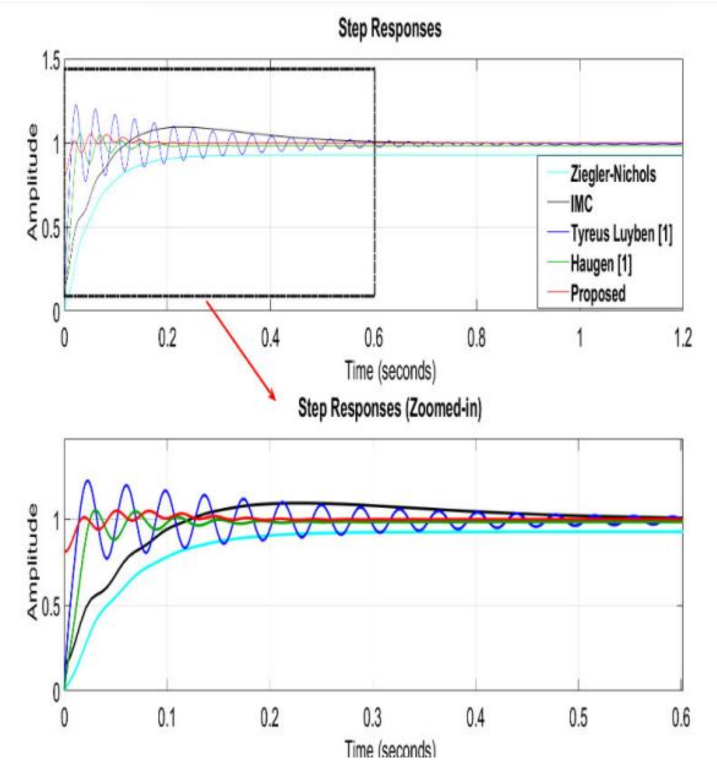

Figure 11: Step responses of EPS system with controller. [1]

Robustness Testing

Robustness testing was carried out to find how the controller performs under varying road conditions, motor speed changes, and external disturbances. It was tested using both pulse and random input signals.

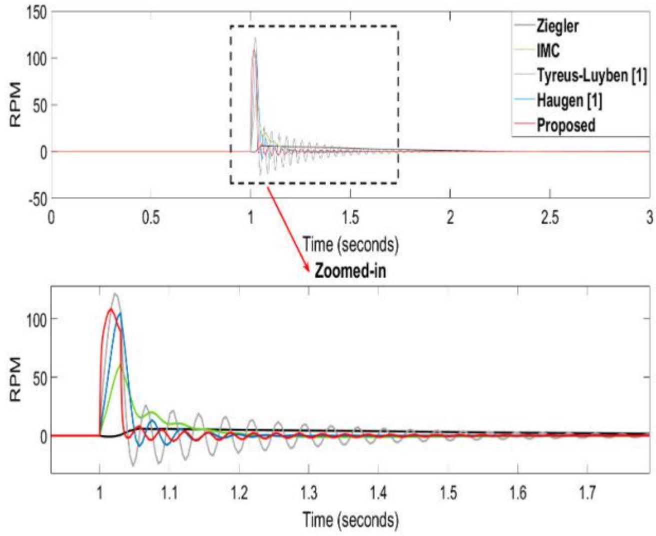

Figure 12: Robustness Test with Pulse Input[1]

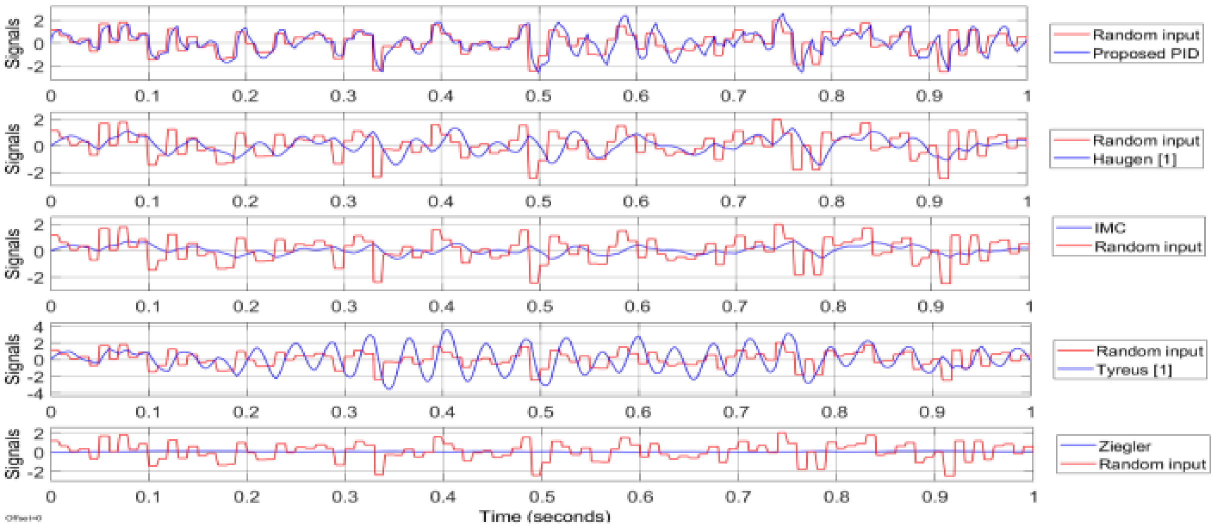

Figure 13: Robustness Test with Random Input. [1]

The test results show that the proposed method was the most robust, with the shortest time to return to a stable state after disturbance.

Criteria	Ziegler-Nichols method [1]	Tyreus-Luyben method [1]	Haugen method [1]	IMC method	Proposed method
Time (sec)	5.1	2.7	7.2	5.5	0.9

Figure 14: Robustness performance results of PID controller for EPS system. [1]

The controller designed for the EPS system met the success criteria objectives of minimizing settling time, peak time, and % overshoot, while also eliminating steady-state error. The performance and robustness tests further showed that it could adapt to varying conditions. Overall, the proposed PID control design was highly successful.

3.2 Control Scheme Analysis

This section focuses on evaluating the performance and stability of the PID controller designed for the EPS system. This includes a look into the closed-loop system, its order, stability, and its frequency response.

3.2.1 What is the format/order of the closed-loop system?

The closed-loop system is a fourth-order system, as found by the order of the transfer function's denominator ($T_D(s)$). The closed-loop transfer function is given by:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}$$ [1]

Where $C(s) = \frac{K_d s^2 + K_p s + K_i}{s}$ [1]

$$\Rightarrow \quad T(s) = \frac{G\_N(s)(k_d s^2 * k_p s + k_i)}{G_D(s)s + G_{N(s)}(k_d s^2 + k_p s + k_i)}$$ [1]

The numerical value of T(s) can be found in MATLAB as follows:

```
% 3.2 Analysis
kp = 217.3
ki = 4344.6
kd = 5.0314

Cs= tf([kd kp ki],[1 0])
Ts = feedback(Gs, Cs)
```

Figure 15: MATLAB code for solving CLTF

The resulting CLTF is:

$$T(s) = \frac{1.26s^3 + 27.86s^2 + 52734.82s}{7.81s^4 + 526.2s^3 + 315400s^2 + 11630000s + 229100000}$$

3.2.3 Is the closed-loop system stable/unstable/marginally stable/conditionally stable etc.? Why?

To determine the stability of the system, the poles of T(s) can be found using the pole() and pzmap() function in MATLAB as follows:

```
CLTFpoles=pole(Ts)
pzmap(Ts)
```

Figure 16: Stability analysis of the closed loop system

The poles of the CLTF were found to be:

```
CLTFpoles =

  -14.645 +     195.69i
  -14.645 -     195.69i
  -19.048 +     19.975i
  -19.048 -     19.975i
```

Figure 17: Poles of T(s)

The pole-zero map is shown below, with all poles in the right hand of the s-plane:

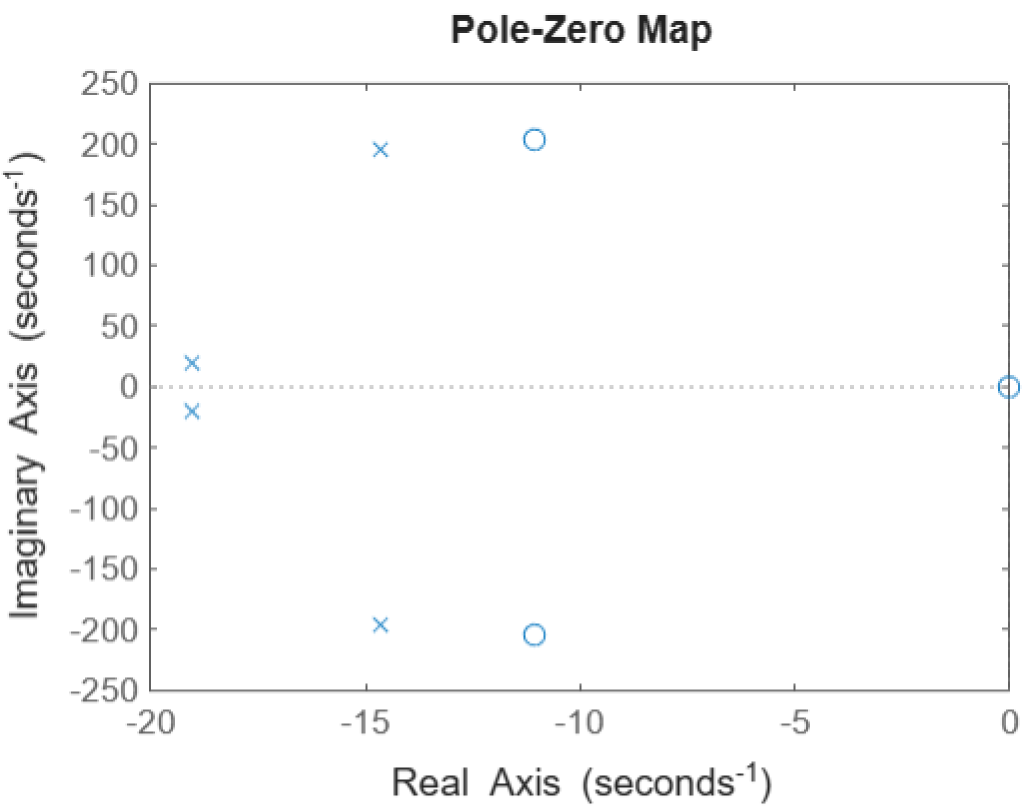

Figure 18: Pole-zero map of T(s)

All poles have negative real parts therefore the system is stable, meaning that the system's response will eventually settle to a steady state.

The stability can be confirmed by forming a Nyquist plot of the open loop transfer function, to check for stability on a Nyquist plot, the number of counterclockwise encirclements around the $(-1, 0i)$ point must equal the number of unstable poles. In this case we wouldn't want to have any encirclements as our system should have no unstable poles.

Firstly, the OLTF needs to be calculated as the product of G(s) and C(s), which can be found using the series() command in MATLAB. Then the nyquist() command can be used to produce the Nyquist plot:

```
OLTF = series(Gs, Cs)
nyquist(OLTF)
```

Figure 19: MATLAB code for Nyquist plot

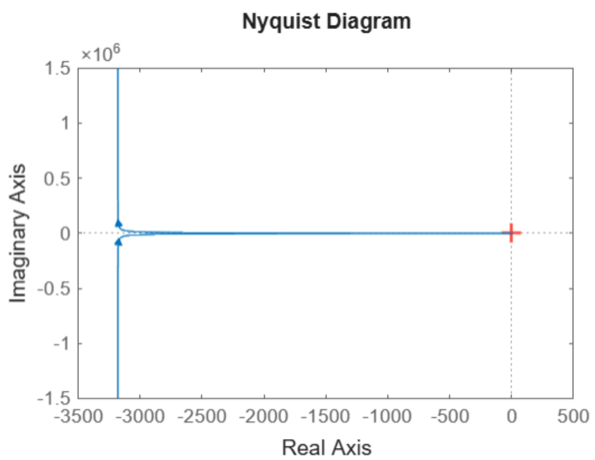

Figure 20: Nyquist plot

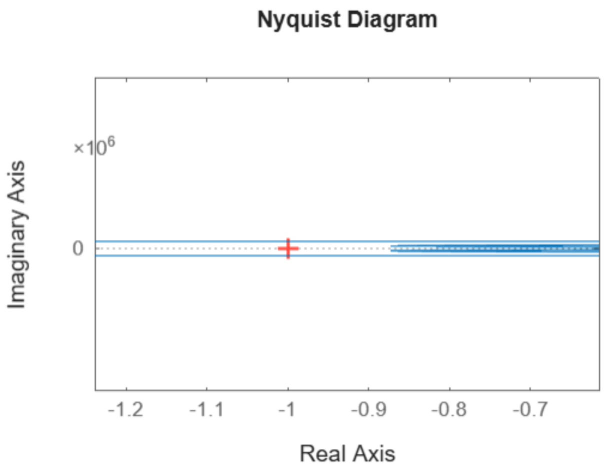

Figure 21: Nyquist plot (zoomed in)

From the Nyquist plot above there are 0 encirclements around the $(-1, 0i)$ point, (ie. N=0) and since there are 0 unstable poles in the OLTF (ie. P=0), N equals P which satisfies the Nyquist criterion for stability.

3.2.4 What are the poles/zeros/eigenvalues of the closed-loop system and how do these affect the behavior of the overall system?

The poles and zeros of the closed-loop system provide information on its time-domain performance and stability. As mentioned before, the system's poles are located in the left half of the s-plane, showing the system is stable.

To find the poles, zeros and eigenvalues of the system we can use the functions pole(), zero() and eig() respectively.

```
poles=pole(Ts)
zeros=zero(Ts)
CLTF_ss=ss(Ts);
eigenvalues=eig(CLTF_ss)
```

Figure 22: MATLAB code for finding poles, zeros and eigenvalues of the closed loop system

```
poles =

  -14.645 +     195.69i
  -14.645 -     195.69i
  -19.048 +     19.975i
  -19.048 -     19.975i


zeros =

        0 +          0i
  -11.056 +     204.28i
  -11.056 -     204.28i


eigenvalues =

  -14.645 +     195.69i
  -14.645 -     195.69i
  -19.048 +     19.975i
  -19.048 -     19.975i
```

Figure 23: Poles, zeros and eigenvalues of closed loop system

- **Poles/eigenvalues**: The poles are found from the roots of the denominator of the CLTF, and the eigenvalues can be found from the state space representation of the closed loop system '...by solving the characteristic equation: det(A- λI) = 0, where λ is the eigenvalue we are trying to solve for, and I is the identity matrix of the same size as A.'[7] The values for the poles and eigenvalues are the same in this case. They '...affect the stability margin and the damping characteristics of the system. The closer the pole is to the imaginary axis in the complex plane, the more it affects the system's stability and oscillations.' [8]
- **Zeros**: Zeros are the roots of the numerator of the CLTF. 'Zeros can improve system stability and transient response by effectively canceling out poles or altering the system's frequency response.'[8] A well-placed zero can help to reduce steady-state error, while improper placement of zeros could lead to complexity or instability.

The pole-zero map (shown in section 3.2.3) shows that the system's poles and zeros are positioned correctly to keep the system stable and responsive (poles are marked as an X and zeros as an O). By adjusting the PID controller, we can adjust these positions to get performance we want, such as reducing overshoot, achieving a faster settling time, and ensuring zero steady-state error.

3.2.5 What type of frequency response does the closed-loop system have? What does that tell about the overall system time-domain performance?

The frequency response of the closed-loop system, as shown in the Bode plot, reveals several important characteristics of the system's behavior. The plot consists of two parts: the magnitude response (in dB) and the phase response (in degrees), both plotted against frequency (in radians per second).

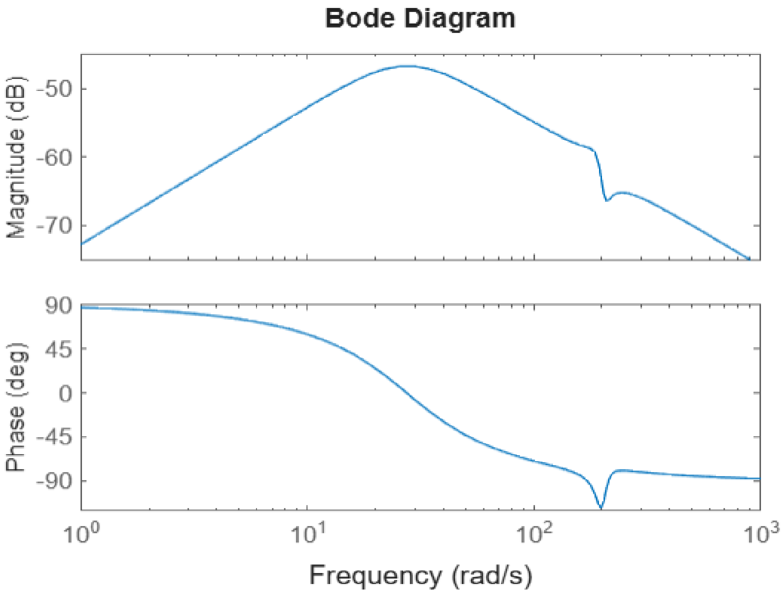

Figure 24: Bode plot for closed loop system

- **Magnitude Response**: The magnitude plot shows that the system shows a peak in the gain followed by a steep decline at higher frequencies. The peak suggests that the system is most responsive that frequency range, with the gain gradually decreasing as the frequency increases. This indicates that the system has a low-pass behavior, accepting low-frequency inputs while rejecting high-frequency disturbances or noise.
- **Phase Response**: The phase plot shows a decrease in phase when frequency increases, indicating that the system introduces a phase shift as it processes higher frequency signals. The phase reaches a value of approximately -90 degrees at 1000rad/s.

Time-Domain Performance:

The **time-domain performance** of the system can be seen from from the Bode plot in terms of both its stability and responsiveness. Looking at the magnitude response, it's clear that the system works well with low-frequency inputs, responding quickly and settling faster. However, for high-frequency disturbances, the system's response weakens due to the decrease in gain, which suggests that the system is optimized for steady-state behavior but might struggle with high-frequency noise or quick changes.

The phase response backs this up, showing that the system stays stable without any major phase shifts that could cause instability or unwanted oscillations in the system's response.

Overall, the Bode plot suggests that the system acts like a low-pass filter, tracking steady-state inputs well while filtering out high-frequency disturbances. This setup results in a stable and responsive system with fast settling times, minimal overshoot, and no noticeable steady-state error.

3.3 Simulation

3.3.1 Step Input Simulation/Verification of Paper Results

Uncompensated System:

The uncompensated system is simulated first to observe the behavior of the EPS system without any control applied. It was simulated with a unit step input over 10 seconds at a 0.02s sampling rate with the following modelling settings:

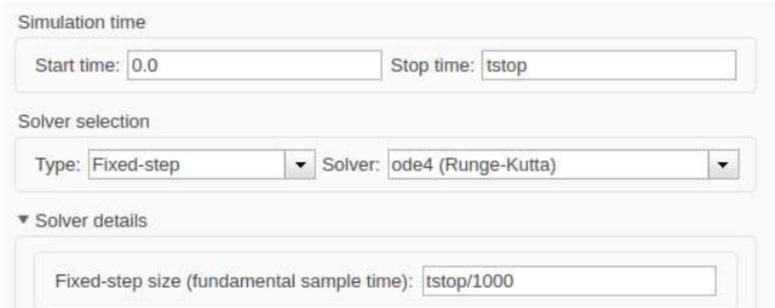

Figure x: Simulink solver settings

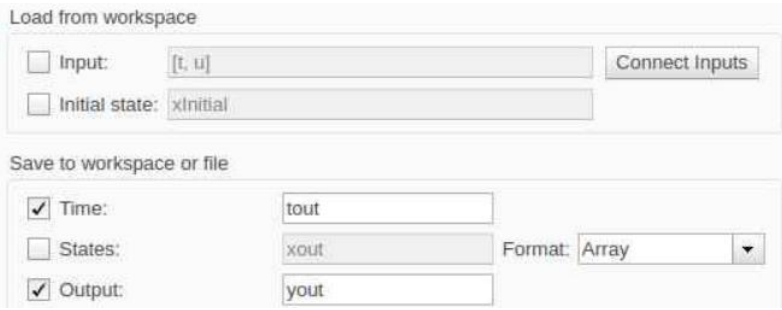

Figure 25: Simulink data import/export settings

The following and Simulink model MATLAB code was used to simulate the response.

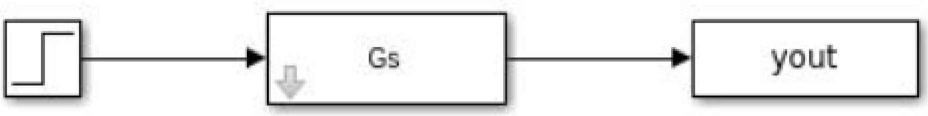

Figure 26: Simulink model for uncompensated system

```
Gs = tf([1.26 27.86 52734.82], ...
    [1.47 112.27 38552.43 50915.38])
tstop=10
sim('Assignment')
plot(tout, yout);
title('System Response');
xlabel('Time (s)');
ylabel('Output Value');
grid on;
```

Figure 27: MATLAB code for uncompensated system

The following step response was obtained from the simulation:

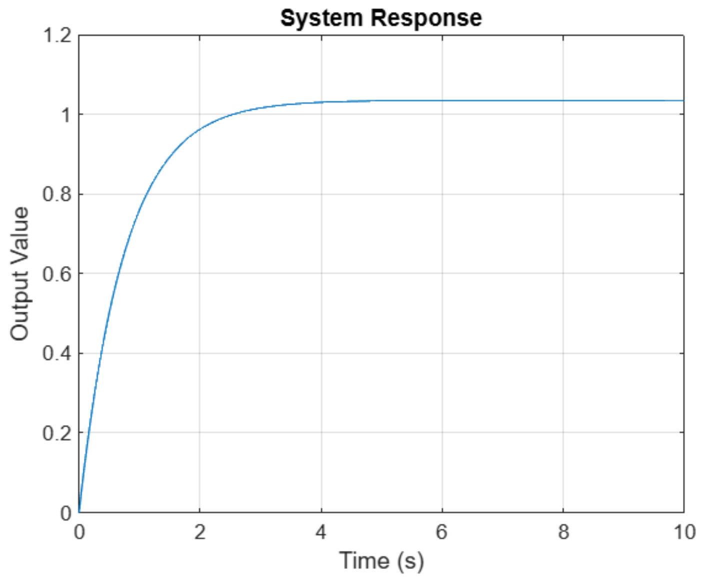

Figure 28: Unit step response of uncompensated system

Looking at the step response for the uncompensated system, it appears that the system is stable, but it has a slow rise time, and the output settles close to 1, but does not reach it exactly. This indicates that the system exhibits a steady-state error, meaning it can't fully eliminate the error between the desired and actual output for a step input. To address this, the PID controller is required to speed up the rise time and eliminate this error.

3.3.2 Compensated System

Once the uncompensated system was simulated, the designed PID controller is applied. The PID controller was implemented using a PID controller block. The controller gains K_p, K_i and K_d were selected based on the values given in the IEEE paper.

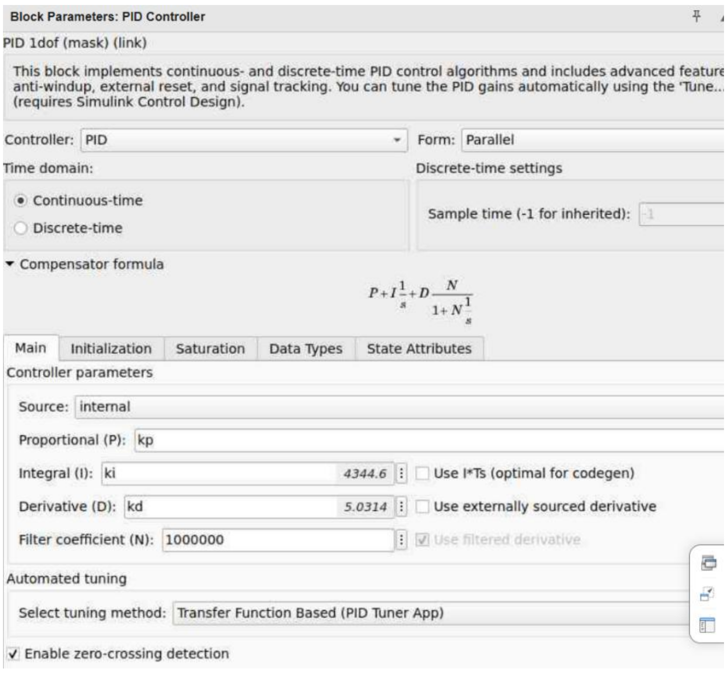

Figure 29: Parameter settings for PID Controller block

The feedback loop was created using a summer block in Simulink, and the complete block diagram is as follows:

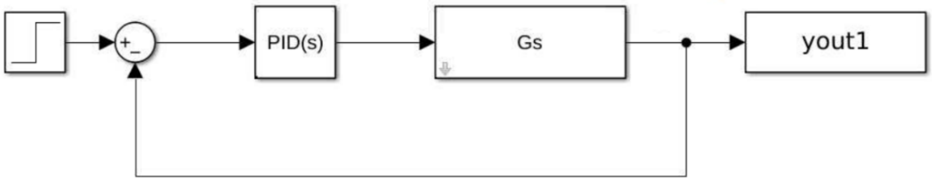

Figure 30: Simulink model for compensated system

For the simulation, the system was run with a unit step input for 10 seconds, however the solver selection type had to be changed to a variable step as an error kept occurring when trying to simulate the model. The results of the step response can be seen below:

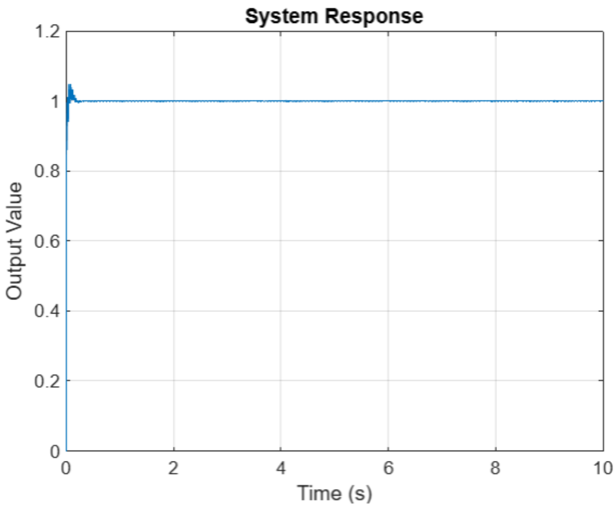

Figure 31: Step response of the compensated system

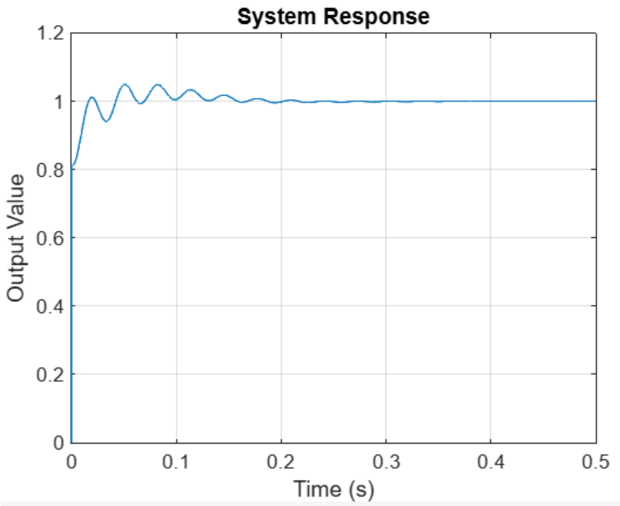

Figure 32: Step response of the compensated system (zoomed in)

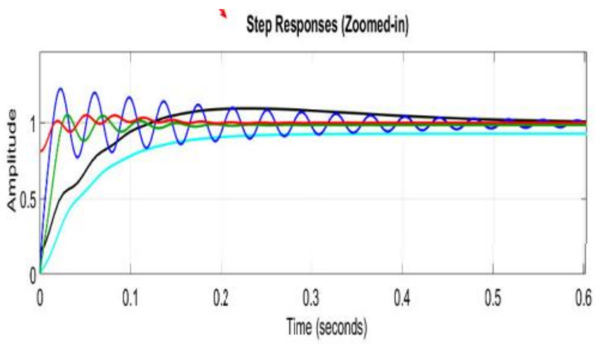

Figure 33: Step response from IEEE paper

The step response of the compensated system shows significant improvement over the uncompensated system. The compensated system quickly responds to the step input and settles near the desired value of 1, with no steady-state error. However, there is a slight overshoot followed by some oscillations, meaning that while the system is performing well, there is still room for improvement. The oscillations suggest that the system is slightly underdamped, which is common when the proportional gain is on the higher side. Despite this, the compensated system is a clear improvement in terms of and response time and eliminating steady state error compared to the uncompensated version, which. Overall, the PID controller has successfully enhanced the system, giving a more accurate and faster response to step inputs with little to no error.

To compute the performance metrics percent overshoot, settling time, peak time, rise time and steady state error, the following MATLAB code was exccecuted:

```
% Steady-state value
y_ss = yout1(end); % last value of yout1

% Percent Overshoot
y_max = max(yout1); % max value of the response
PO = ((y_max - y_ss) / y_ss) * 100;

% Settling Time
settling_idx = find(abs(yout1 - y_ss) <= 0.02 * y_ss, 1, 'first');
t_s = tout(settling_idx); % Time when response stays within 2% of ss

% Rise Time
t_r_10 = find(yout1 >= 0.1 * y_ss, 1, 'first'); % When reaches 10% of ss
t_r_90 = find(yout1 >= 0.9 * y_ss, 1, 'first'); % When reaches 90% of ss
t_r = tout(t_r_90) - tout(t_r_10); % Difference between 90% and 10%

% Peak Time
[~, peak_idx] = max(yout1); % Index of the peak value
t_p = tout(peak_idx); % Time of first peak

% Steady-State Error
desired_value = 1;
E_ss = y_ss - desired_value;

% Display Results
disp(['Percent Overshoot: ', num2str(PO), '%']);
disp(['Settling Time: ', num2str(t_s), ' s']);
disp(['Peak Time: ', num2str(t_p), 's']);
disp(['Rise Time: ', num2str(t_r), 's']);
disp(['Steady-State Error: ', num2str(E_ss)]);
```

Figure 34: MATLAB code for calculating step performance metrics

Now comparing the results to that of the IEEE paper:

```
Percent Overshoot: 4.8444%
Settling Time: 0.015806 s
Rise Time: 0.0092386s
Peak Time: 0.051452s
Steady-State Error: 0.0012586
```

Figure 35: Results of the simulation

Percent Overshoot	4.999%
Settling Time	0.2291s
Rise Time	0.0101s
Peak Time	0.0512s
Steady State error	0

Figure 36: Results found in the IEEE paper.

The simulation results closely align with those presented in the IEEE paper.

- **Percent Overshoot:** The simulation shows 4.8444%, which is almost identical to the paper's result of 4.999%, reflecting effective PID tuning.
- **Settling Time:** The simulation achieves a slightly faster settling time of 0.015806s, compared to 0.2291s in the paper, indicating better optimization of the PID controller. This result was the one that deviated the most between the two.
- **Rise Time:** The simulation rise time of 0.0092386s is very close to the paper's 0.0101s, meaning the transient response is very accurate.
- **Peak Time:** The peak time of 0.051452 s in the simulation is almost identical to the 0.0512 s in the paper.
- **Steady-State Error:** The simulation results in a steady-state error of 0.0012586, slightly higher than the paper's value of 0 which could possibly be due to rounding, but still within an acceptable range.

Overall, the simulation performs similarly to the results in the IEEE paper, with minor variations in settling time and steady-state error. The PID controller shows strong performance, with fast stabilization and minimal error.

3.3.3 Robustness test

Similarly to the IEEE paper, a robustness test was carried out to further assess the performance of the system and to test the similarity of results between this model and the paper. A pulse generator block was added to the closed loop system which adds a disturbance that is 'a 50 amplitude, 10ms pulse lasting 2s.'[1]

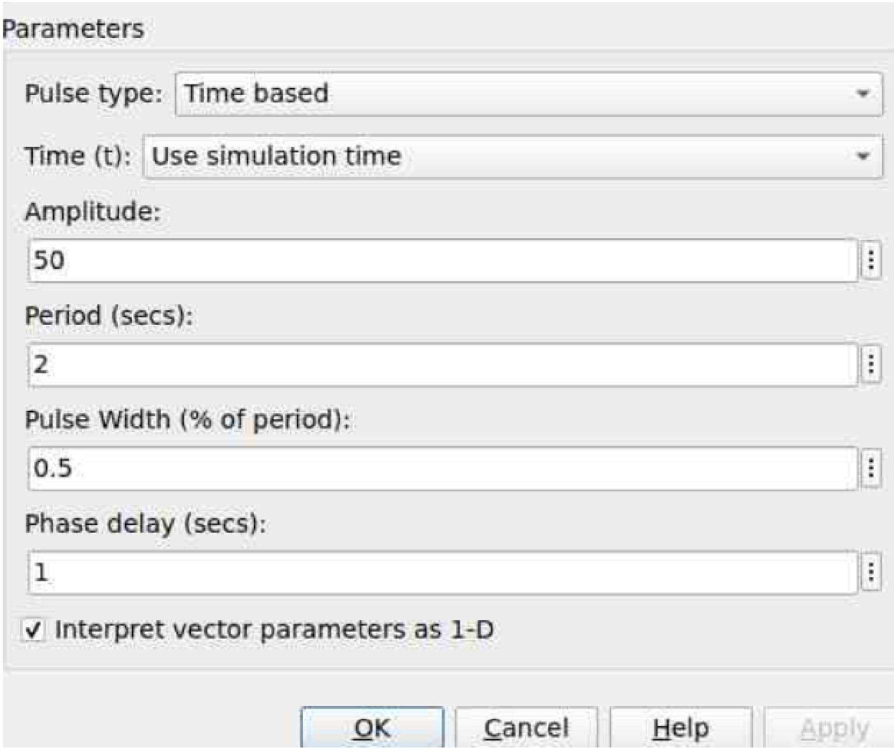

Figure 37: Pulse generator block parameters

The pulse generator was connected to the system input, and the response of the system was observed by measuring the output from the To Workspace block. The simulation time was set to 3 seconds to ensure the pulse disturbance and the system's response could be fully captured.

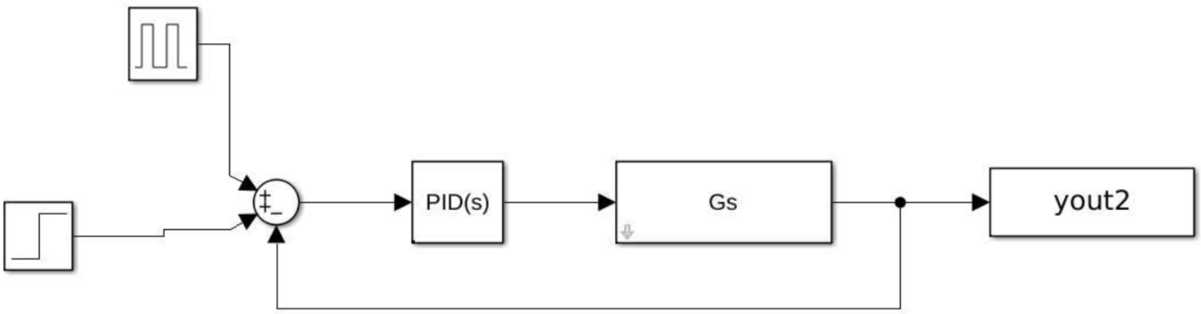

Figure 38: Block diagram with added disturbance

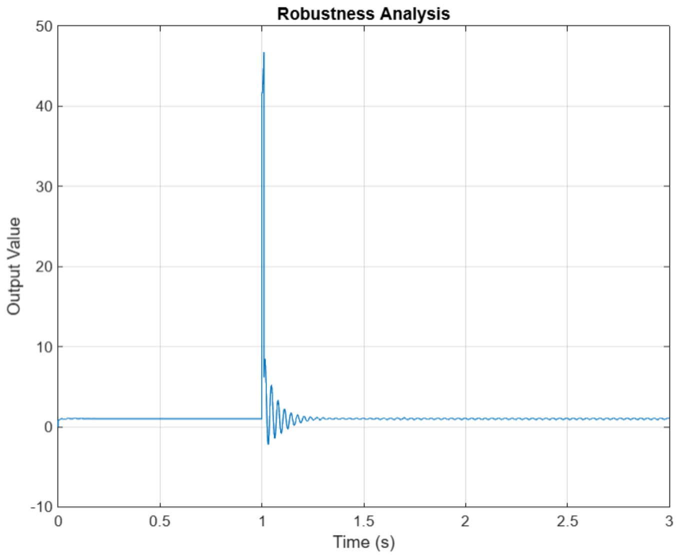

Figure 39: Response to 50 amplitude pulse disturbance

The observed behavior shows that the PID-controlled EPS system is able to recover very quickly from the pulse disturbance. The output initially experienced a sharp spike due to the pulse, but it quickly recovered, settling at the steady-state value in under roughly a quarter of a second. This quick return to normal shows that the system is robust to disturbances and can maintain its performance. In the case of real world external disturbances like road bumps or sudden steering inputs, the result of this test shows that the system would be able handle these changes with little delay.

3.3.6 Conclusion

In conclusion, the simulations of both the uncompensated and compensated systems demonstrate the improvement achieved with the addition of the PID controller. The uncompensated system showed stability but was unable to eliminate steady-state error and had a slow rise time. After introducing the PID controller, the system's performance was greatly improved, with a much faster rise time, reduced steady-state error, and a quick return to the desired output.

The comparison of the simulation results with those from the IEEE paper shows that the PID controller was tuned accurately, with only minor variations in settling time and steady-state error. The system performed similarly to the published results, confirming the accuracy of the designed system.

Additionally, the robustness test showed that the system is capable of handling external disturbances effectively. After the pulse disturbance was applied, the system quickly returned to its steady-state value, highlighting the PID controller's ability to maintain stability. These results suggest that the PID-controlled EPS system would respond well to real-world disruptions. Overall, the simulations and robustness test confirm that the PID controller greatly improves the system's performance, achieving fast stabilization, minimal error, and robust performance with disturbances.

4 Part C: Reflection

4.1 What did you learn?

Throughout this assignment, I learned about the application of PID control to an Electric Power Steering (EPS) system. More specifically, I learned how to design and tune a PID controller to improve the system's performance, focusing on reducing overshoot, minimizing steady-state error, and improving the transient response. This was very beneficial to me because I found the topic of PID control slightly confusing during the module EEN1047 Control Systems Analysis last semester, this assignment really strengthened my understanding of it. After designing the controller, I improved my learning on how to perform stability analyses, calculate performance metrics (such as settling time and rise time), and carry out tests to see how the system handles disturbances. I also became a lot more familiar with Simulink and MATLAB practices.

4.2 How did you learn it?

I learned these concepts through hands-on experience and also my own online research during the assignment. Writing scripts in MATLAB and running simulations on Simulink helped me get more used to creating models and also helped me understand the elements of a PID controller by playing around with the gain values and seeing how they affect the system. During the assignment I had many questions that I had to look online for answers, especially on the topic of PID control. This research strengthened my understanding of the topic.

4.3 Was it useful? Why/why not?

This assignment was extremely useful as it allowed me to apply what I learned in class to a real life practical problem, this broadened my understanding as it wasn't just general block diagrams and equations I was working with, I could relate it to the EPS system. The opportunity to design and simulate a PID-controlled system improved my overall understanding of PID massively. It also helped me understand the importance of testing a system under different conditions like I did with the robustness testing.

4.4 Did you have a plan for carrying out the project? If so, what was it? If not, why not?

At first, I did not have a very detailed plan as I didn't know what system I was going to pick, but I gradually developed one as I read through my chosen paper. My plan was to first model the EPS system, then implement the PID controller, and follow this by simulating the system and analyzing the results. The procedure in the paper also helped me to follow a step by step plan for assessing the system. I also planned to compare my simulation results with those presented in the IEEE paper. While I did not have a fully written out plan at the start, the process of working through each task step-by-step helped me work it out.

4.5 How much time did you spend on particular activities, was this appropriate?

I think a lot of time was spent in the beginning looking for the paper I was going to use. I felt that I didn't know what I was looking for and spent hours assessing different models, seeing if they fit the requirements of the assignment. Even though having to look for the paper myself was a good learning experience I did feel like there could've been more guidance on what to look out for. I spent a significant amount of time on setting up the Simulink model and troubleshooting simulation issues, particularly related to solver settings. This was necessary, as resolving these issues made sure that the simulation could run correctly and produce reliable results. I also spent large amount of time writing the report itself, but this was also neccessary. The time I spent on most of these tasks was appropriate, but I do feel that the search for the right paper took a lot more time than it should have.

4.6 What went right? What went wrong? What would you do differently?

What went right was the successful simulation of the systems, with results that were consistent with the paper. The PID controller provided a clear improvement in system performance, and the robustness test demonstrated the system's resilience to disturbances. What went wrong were the initial difficulties with solver settings, which caused errors during simulation. Once I switched to the variable-step solver, the issues were fixed. Also, if I were to do the assignment again, I would allocate more time to look at different disturbance scenarios and see how the system would react.

4.7 Was there any aspect of the assignment that you found interesting? Was there a control scheme/physical system you think would be interesting to know more about?

The robustness testing was the most interesting part of the assignment for me. It was interesting to see how the PID controller got the system to recover so quickly from the disturbance. As for future exploration, I would be interested in learning more about PID control, as before I wasn't as familiar with it but this assignment has sparked my interest in it.

References

[1] "Improved PID Control design for electric power steering DC motor," IEEE Journals & MagazineIEEE Xplore, 2025. https://ieeexplore.ieee.org/document/10818684 [Accessed 18/02/2025]

[2] T. K. Priyambodo, A. Dharmawan, O. A. Dhewa, and N. A. S. Putro, "Optimizing control based on fine tune PID using ant colony logic for vertical moving control of UAV system," *AIP Conference Proceedings*, Jan. 2016, doi: 10.1063/1.4958613. [Accessed 27/02/2025]

[3] "Genetic Algorithm" ScienceDirect, from Applied Energy 2025. https://www.sciencedirect.com/topics/engineering/genetic-algorithm [Accessed 27/02/2025]

[4] "Linear Quadratic Regulator" ScienceDirect, from Encyclopaedia of Physical Science and Technology (Third Edition), 2003 https://www.sciencedirect.com/topics/physics-and-astronomy/linear-quadratic-regulator [Accessed 27/02/2025]

[5] Wikipedia contributors, "Model predictive control," *Wikipedia*, Dec. 17, 2024. https://en.wikipedia.org/wiki/Model_predictive_control [Accessed 27/02/2025]

[6] Testbook, "Integral Controller: Know Definition, block diagram, Derivation & Applications," *Testbook*, Aug. 22, 2024. https://testbook.com/electrical-engineering/integral-controller [Accessed 01/03/2025]

[7] "The eigen values of linear system are the location ofa)zeros of the systemb)poles of the systemc)both (a) and (b)d)finite poles and zerosCorrect answer is option 'B'. Can you explain this answer? EduRev Electrical Engineering (EE) Question," *EDUREV.IN*. https://edurev.in/question/1758702/The-eigen-values-of-linear-system-are-the-location-ofa-zeros-of-the-systemb-poles-of-the-systemc-bot [Accessed 04/03/2025]

[8] B. Avcı, "Navigating the Influence of Poles and Zeros in Control Systems: A Deep Dive with Python," *Medium*, Nov. 27, 2023. [Online]. Available: https://medium.com/@mbugraavci38/navigating-the-influence-of-poles-and-zeros-in-control-systems-a-deep-dive-with-python-bac85aa85ac4 [Accessed 04/03/2025]