ME5422: Computer Control & Application

Comparative Analysis of Digital Controllers: Lead-Lag Compensation and PID Controllers for Dynamic System Control

April 2025

Kukadia Parthiv Vinubhai (A0304932J)

Department of Mechanical Engineering
National University of Singapore

1 Introduction

Digital control systems play a crucial role for applications in modern engineering, where its essential to have precise and efficient control of dynamic systems. Among the control strategies that are commonly employed, Proportional-Integral-Derivative (PID) controllers and Lead-Lag Compensation controllers are widely used due their effectiveness in improving transient response, steady-state performance, and stability.

A PID controller is a classical feedback control technique that is designed using a combination of proportional, integral, and derivative actions to regulate a system's output. It has a wide range of applications across various industries - a result of its simplicity and effectiveness in minimizing steady-state error and improving transient response [1]. However, there is a challenge that PID controllers struggle with, in systems with high-order dynamics or significant delays, there are issues with phase lag and stability.

In contrast, a lead-lag compensation controller can be used to provide a structured approach to enhance both phase margin and steady-state error characteristics. By increasing the phase margin, a lead compensation can improve transient response, while reducing the stead-state error allows the lag compensation to enhance steady-state accuracy [2]. The technique is often used in cases when fine-tuned stability and improved dynamic performance are required beyond what a standard PID controller can achieve.

This project focuses on designing, simulating, and implementing both a lead-lag compensation controller and a PID controller for a given dynamic system. A comparative analysis will be conducted to evaluate their effectiveness in terms of response time, stability, steady-state error, and robustness under varying operating conditions. The study aims to determine which control strategy offers superior performance for the given system constraints, considering the practical limitations of implementing digital-control in the real-world.

2 Objectives

The project will provide a quantitative and qualitative comparison between PID and lead-lag compensation controllers, by offering insights into their respective advantages and limitations in digital control system design. The objectives of this project are:

1. Design and Implementation
    - Develop a PID controller and a lead-lag compensation controller for the given dynamic system.
    - Implement the controllers in a digital control framework, considering practical constraints such as sensor resolution and actuator saturation limits.
2. Theoretical Analysis
    - Analyze the mathematical formulation of both control strategies.
    - Derive the system response characteristics using discrete-time modeling and root locus, bode plot, or frequency response methods.
3. Simulation and Performance Analysis
    - Simulate both controllers in MATLAB/Simulink and evaluate their response in terms of: Rise time, settling time, overshoot, steady-state error, stability margins, and robustness.
    - Investigate the impact of parameter tuning and system constraints on performance.
4. Comparative Study and Conclusion
    - Compare the two controllers based on simulation results and theoretical insights.
    - Identify trade-offs between control performance and implementation complexity.
    - Provide recommendations on the suitability of each controller for real-world applications.

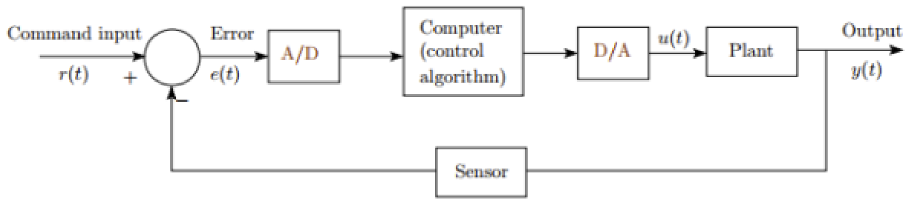


Figure 1: High-Level Block Diagram of Digital Controller [4]

3 Methodology

3.1 System Design

The system developed is a servo control system, where the goal is to regulate the position $\theta(t)$ of a rotating actuator based on a reference input while accounting for practical constraints such as sensor resolution, actuator saturation, and noise. The system consists of the following components:

1. Actuator (DC Motor/Servo Motor)
    - Convert electrical signals into mechanical motion (rotation).
    - The motor dynamics are governed by electromechanical equations, which describe the relationship between input voltage u(t) and output position $\theta(t)$.
    - The motor introduces damping effects and inertia, which influences system stability.
2. Sensor (Rotary Encoder)
    - Measures the angular position $\theta(t)$ and provides discrete pulses.
    - Resolution is selected as n=1000 pulses per revolution, meaning each full rotation (360°) is divided into 1000 discrete steps.
    - The discrete nature of the sensor introduces quantization error, affecting precision.
3. Controller (Digital Implementation: PID and Lead-Lag Compensator)
    - Processes the error signal and computes control inputs.
    - Implemented in MATLAB/Simulink and executed on a micro-controller or digital platform.
4. Actuator Driver (Power Amplifier or PWM Signal Generator)
    - Convert low-power control signals into high-power commands for the motor.
    - Ensures the control signal remains within the saturation limits:
      $-5 \leq u(t) \leq 5$.
5. Feedback Loop
    - Compares measured position θ with the reference input.
    - The controller adjusts the actuator's behavior based on real-time feedback.

A closed-loop system allows for real-time adjustments to the control effort to help maintain stability and achieve desired performance. A high-level system block diagram can be shown as:
Some considerations and constraints to take into account for our system design include:

1. Actuator Saturation

- The control signal u(t) is limited to a range of [-5,5] volts due to hardware constraints.
- If the controller generates an excessively large control input, it will be clipped to the saturation limits, potentially introducing nonlinear effects such as integral windup in PID controllers.
- Mitigation: Implement an anti-windup mechanism to handle saturation effects.

2. Encoder Resolution and Quantization
   - The encoder converts continuous angular position into discrete steps (n = 1000 pulses per revolution).
   - This introduces quantization error, which affects control precision.
   - Mitigation: The controllers are designed considering discrete resolution effects and apply interpolation techniques if necessary.
3. Servo Velocity is Not Directly Measured
   - The velocity $\omega(t)$ is not measured by can be estimated using numerical differentiation of position.
   - Mitigation: Implement a state observer (e.g., Kalman filter or finite difference approximation) to estimate velocity.
4. Digital Implementation and Sampling Time
   - The system is implemented digitally with a sampling time T_s.
   - The selection of T_s is such that it should be small enough to capture system dynamics (e.g., 10 ms to 20 ms), and if it is too small it will increase computational load, and if it is too large it will cause poor system response and aliasing.
5. Noise and Disturbance
   - Sensor noise and external disturbances affect system performance.
   - Mitigation: Use low-pass filtering techniques to remove high-frequency noise while preserving control bandwidth.

The implementation methodology will include:

1. Simulation Environment
   - The entire system is first modeled and tested in MATLAB/Simulink.
   - Discrete-time controllers are implemented using difference equations.
2. Micro-controller (Not applicable)
   - Controllers can be deployed on a micro-controller for real-world testing.
   - PWM signals are used to drive the motor.
3. Testing and Validation
   - Controllers are evaluated based on rise time, settling time, overshoot, steady-state error, and robustness.
   - Comparison between PID and Lead-Lag control strategies is conducted under various conditions.

Overall, following the implementation mentioned above will ensure that both controllers (PID and lead-lag) are implemented effectively within the physical and practical constraints of the servo system. The comparative analysis provided in Section 4 will give insight into their strengths and weaknesses, enabling an informed control strategy selection for real-world applications.

3.2 Control Algorithm

This section covers the design of the two control algorithms: a PID controller and a Lead-Lag compensator, for the discrete-time implementation of the system. these controllers are chosen to evaluate their performance in terms of tracking accuracy and stability for the given plant model.

PID Controller Design

The Proportional-Integral-Derivative (PID) controller is one of the most widely used control strategies due to its simplicity and effectiveness in a variety of systems. The PID controller is developed to minimize the error between the reference input and the system output by adjusting the control signal based on 3 terms:

1. **Proportional (P)**: Directly proportional to the error to ensure that the control effort increases with the magnitude of the error.
2. **Integral (I)**: Sum of the errors over time to address any steady-state error by eliminating it.
3. **Derivative (D)**: Accounting for the rate of change of the error, helping to dampen oscillations and improve the transient response.

In my digital controller design, the PID controller is tuned using MATLAB's "pidtune" function, which is an inbuilt optimizer for the PID parameters (K_p, K_i, K_d) based on the discrete-time system.

```matlab
% PID Controller - Adjusted tuning for better performance
C_pid = pidtune(G_discrete, 'PID', 2); % Initial tuning
% Further adjustment of Parameters
Kp_pid = C_pid.Kp * 16;
Ki_pid = C_pid.Ki * 2.75; % Increased Ki
Kd_pid = C_pid.Kd * 4.5;
C_pid = pid(Kp_pid, Ki_pid, Kd_pid, Ts); % Re-create C_pid with modified Ki
C_pid.Tf = 1/100; % Set derivative filter coefficient (N)
C_pid.Ts = Ts; % Explicitly set the sample time
```

The resulting PID parameters are used to implement the controller in the Simulink model. The discrete PID controller block is configured such that the Proportional gain (K_p) adjusts the magnitude of the control signal, the integral gain (K_i) corrects for accumulated errors over time, and the derivative gain (K_d) improves the transient behavior by reacting to error changes. The Simulink Diagram for the PID controller can be seen in Figure 2, the same design was replicated with a ramp input instead of step to accurately track ramp error as shown in Figure 3.

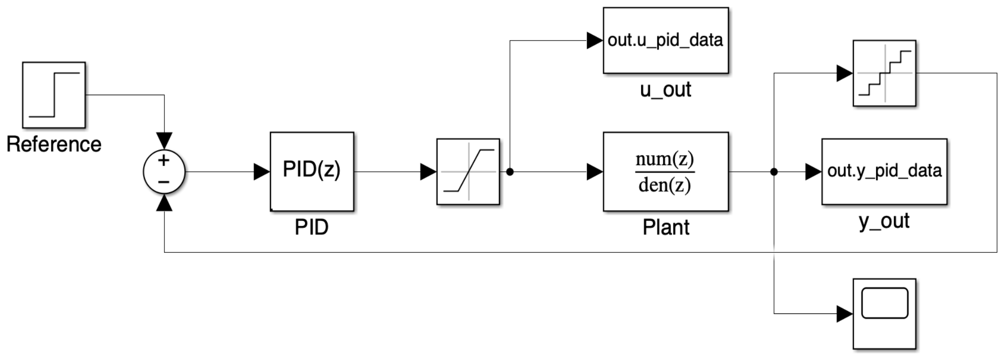


Figure 2: PID Step Controller Design

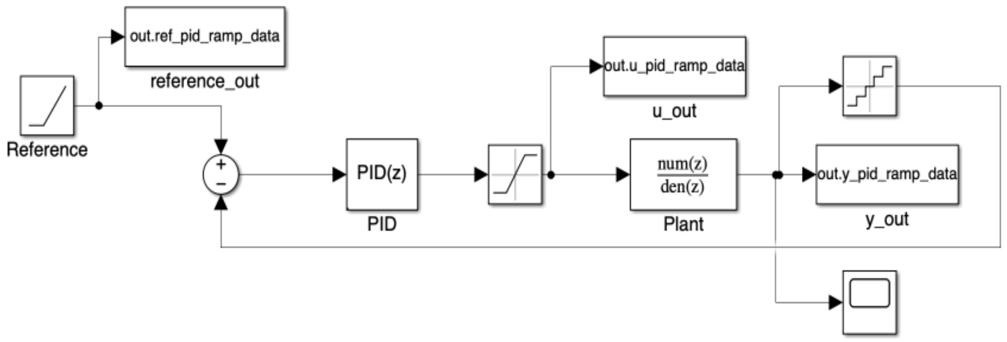


Figure 3: PID Ramp Controller Design

Lead-Lag Compensator Controller Design

A Lead-Lag Compensator is a type of controller designed to improve the system's phase margin and reduce steady-state error. It is particularly useful in systems that require both frequency and time-domain improvement. The compensator is built from two components:

1. **Lead Compensator**: Increases phase margin by adding a positive phase at higher frequencies, which will help improve the system's transient response.

2. **Lag Compensator**: Reduces steady-state error by providing low-frequency gain, which helps in stabilizing the system and improve accuracy.

The parameters of the compensator are chosen based on the desired phase margin and cutoff frequency. The compensator is derived from the plant's frequency response such that it meets a specified phase margin (PM) and cutoff frequency (ω_c).

```
PM = 70; wc = 5.5; % Slightly higher PM, lower wc
[mag,phase] = bode(G_continuous, wc);
phase_needed = PM - (180 + phase);
alpha = (1 - sind(phase_needed))/(1 + sind(phase_needed));
T_lead = 1/(wc*sqrt(alpha));
K_lead = sqrt(alpha)/mag;
K_lead = K_lead * 8; % Increase K_lead
beta = 25; % Further increased beta for lower ramp error
T_lag = 5/(wc/beta);
C_leadlag = c2d(K_lead*(1 + T_lead*s)/(1 + alpha*T_lead*s) * (1 + T_lag*s)/(1 + beta*
    T_lag*s), Ts, 'tustin');
```

The Lead-Lag Compensator is then discretized using the Tustin method and implemented in the digital domain, with the system's discrete-time transfer function G_discrete. The result is a controller that improves the system's performance by enhancing both accuracy and stability The Simulink Diagram for the Lead-Lag Compensator controller can be seen in Figure 4, an identical system was designed to accurately track Ramp Error as shown in Figure 5.

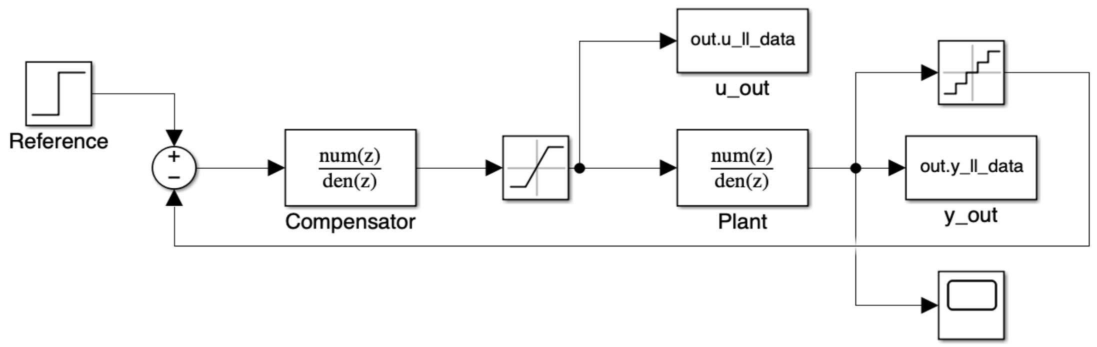


Figure 4: Lead-Lag Compensator Step Controller Design

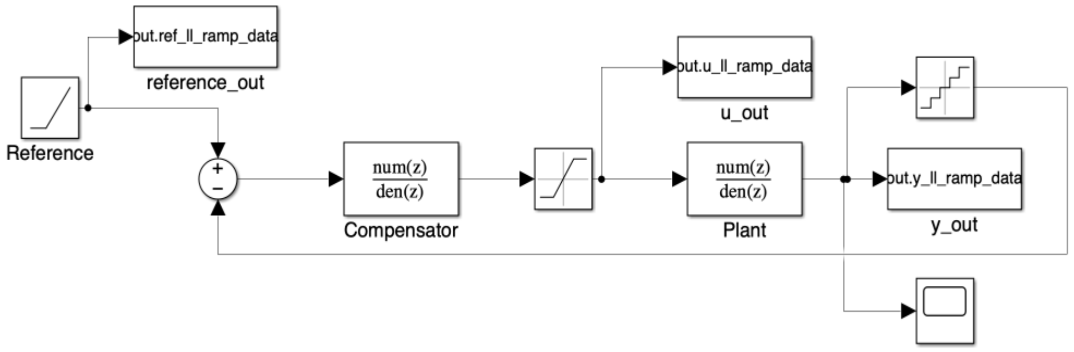


Figure 5: Lead-Lag Compensator Ramp Controller Design

Control Algorithm Implementation in Simulink

The algorithms are implemented in Simulink, where the system and controller are represented by a set of interconnected blocks:

1. **Reference Input**: A step input is provided as the reference signal, which the system aims to track.
2. **Controller**: The PID or Lead-Lag controller is used to adjust the control signal based on the error between the reference input and the system output.
3. **Plant**: The discrete-time transfer function G_discrete represents the plant being controlled, with the system dynamics modeled and processed in discrete time using a sample time (T_s).
4. **Quantization**: A quantizer is used to discretize the control signal, to ensure that the output is consistent with the resolution of the system (in my case a 1000-count encoder).
5. **Saturation**: The control signal is constrained within specified limits to prevent excessive control inputs.
6. **Feedback Loop**: The error is then calculated by comparing the output of the plant with the reference input, and this error is used to adjust the control signal in real-time.

These components are put together to work together to implement the control strategy, which provides the system with a way to track reference input while ensuring stability and performance according to the desired specification. The implementation can be seen below in 3.4 Software Implementation.

3.3 Software Implementation

A code was used to develop both controllers, starting with defining the transfer function, defining the parameters of both controllers, developing the model in simulink, running the simulation of the controllers for the system, and providing the necessary performance metrics and graphs that will be discussed in section 4. To access the code, please go to the linked github repository and download the `Digital_Controller.m` file - which when run will provide you with 4 .slx files (the 4 simulink systems - 2 for PID and 2 for Lead-Lag as mentioned above), and the performance metrics and graphs provided below: GitHub Repository for Code

This section details the software implementation of the Proportional-Integral-Derivative (PID) and Lead-Lag controllers within the MATLAB environment. The implementation leverages the Control System Toolbox for defining and analyzing the transfer functions and their frequency responses.

System Definition

The continuous-time plant is defined as a second-order system with a transfer function $G_c(s)$:

$$G_c(s) = \frac{1}{s(s+1)}$$

This continuous-time plant is then discretized using the Zero-Order Hold (ZOH) method with a sampling time of $T_s = 0.01$ seconds, resulting in the discrete-time plant $G_d(z)$. The discretization process is implemented in MATLAB using the 'c2d' function:

```
G_continuous = (1/(s+1))*(1/s);
Ts = 0.01;
G_discrete = c2d(G_continuous, Ts, 'zoh');
```

3.3.1 PID Controller Implementation

The PID controller is initially tuned using the 'pidtune' function in MATLAB, which provides initial gain values based on the dynamics of the discrete-time plant. The PID controller in the Laplace domain has the form:

$$C_{PID}(s) = K_p + \frac{K_i}{s} + K_d s$$

For digital implementation, this is discretized and often includes a derivative filter to reduce noise amplification. The 'pid' object in MATLAB represents this discrete-time PID controller. The 'pidtune' function uses the following form:

$$C_{pid} = Kp + Ki \cdot \frac{T_s}{z-1} + Kd \cdot \frac{z-1}{T_s}$$

and the gains provided from 'pidtune' are:

$$K_p = 2.36, K_i = 0.603, K_d = 2.06, T_s = 0.01$$

The 'pidtune' gains were not robust enough for the system, so the gains are further adjusted from the initial tuning with the resulting PID controller gains being:

$$K_p = 16 \times K_{p,tuned}$$

$$K_i = 2.75 \times K_{i,tuned}$$

$$K_d = 4.5 \times K_{d,tuned}$$

where $K_{p,tuned}$, $K_{i,tuned}$, and $K_{d,tuned}$ are the gains obtained from the 'pidtune' function. The derivative filter time constant is set such that $N = 1/T_f = 100$.

Lead-Lag Compensator Implementation

The Lead-Lag compensator is designed in the continuous-time domain based on desired phase margin (PM_{des}) and gain crossover frequency (ω_c). The design involves calculating the necessary phase lead and determining the parameters of the lead and lag sections.

The lead compensator has the form:

$$C_{lead}(s) = K_{lead}\frac{1 + T_{lead}s}{1 + \alpha T_{lead}s}, \quad \alpha < 1$$

where α is related to the maximum phase lead, T_{lead} determines the location of the zero, and K_{lead} is the gain.

The lag compensator has the form:

$$C_{lag}(s) = \frac{1 + T_{lag}s}{1 + \beta T_{lag}s}, \quad \beta > 1$$

where β determines the attenuation at high frequencies, and T_{lag} determines the location of the pole and zero at lower frequencies to improve steady-state error.

The design equations used are:

$$\begin{aligned}
\text{Phase needed} &= PM_{des} - (180^\circ + \angle G_c(j\omega_c)) \\
\alpha &= \frac{1 - \sin(\phi_{max})}{1 + \sin(\phi_{max})}, \quad \text{where } \phi_{max} \geq \text{Phase needed} \\
T_{lead} &= \frac{1}{\omega_c\sqrt{\alpha}} \\
K_{lead} &= \frac{\sqrt{\alpha}}{|G_c(j\omega_c)|} \times \text{Gain Adjustment Factor} \\
T_{lag} &= \frac{5}{\omega_c/\beta}
\end{aligned}$$

The MATLAB implementation for the Lead-Lag compensator is provided above, and the controller parameters, after several iterations were:

$$\text{Phase Margin (PM)} = 70^\circ$$
$$\text{Gain Crossover Frequency } (\omega_c) = 5.5 \text{ rad/s}$$

From the Bode plot of the continuous-time plant $G_{continuous}$ at ω_c:

$$|G_{continuous}(j\omega_c)| = \text{mag}$$
$$\angle G_{continuous}(j\omega_c) = \text{phase}$$

The required phase lead, ϕ_{needed}, the parameter α for the lead compensator, the time constant T_{lead} of the lead compensator, is calculated as shown above. The gain K_{lead} of the lead compensator is increased for better performance:

$$K_{lead} = \frac{\sqrt{\alpha}}{\text{mag}} \times 8$$

For the lag compensator, the parameter β is chosen to further reduce the ramp error:

$$\beta = 25$$

The time constant T_{lag} of the lag compensator is:

$$T_{lag} = \frac{5}{(\omega_c/25)}$$

The continuous-time Lead-Lag controller, $C_{leadlag\_continuous}(s)$, is then given by:

$$C_{leadlag\_continuous}(s) = K_{lead} \cdot \frac{1 + T_{lead}s}{1 + \alpha T_{lead}s} \cdot \frac{1 + T_{lag}s}{1 + \beta T_{lag}s}$$

Finally, the discrete-time Lead-Lag controller, $C_{leadlag}(z)$, is obtained by discretizing $C_{leadlag\_continuous}(s)$ using Tustin's method with a sampling time T_s:

$$C_{leadlag}(z) = \mathcal{Z}\{C_{leadlag\_continuous}(s)\}\Big|_{s=\frac{2}{T_s}\frac{z-1}{z+1}} = \text{c2d}(C_{leadlag\_continuous}, T_s, \text{'tustin'})$$

The continuous-time Lead-Lag compensator $C_{leadlag\_continuous}(s)$ is then discretized using Tustin's method (bilinear transformation) with the same sampling time T_s to obtain the discrete-time Lead-Lag compensator $C_{leadlag}(z)$.

The parameters for the controller are then used in the simulink models designed to run a PID controller simulation and a Lead-Lag controller simulation, and their system response, performance metrics, and bode plots are extracted for comparative analysis.

4 Results and Discussion

Below, I will discuss my results from designing a PID controller and a Lead-Lag Compensator controller under the constraints provided above. The controllers were simulated for a total of 10 seconds and a performance metric table was created as well a system response plot and control signal diagram.

4.1 Comparative Analysis

To evaluate controller performance, both PID and Lead-Lag compensator were simulated using identical plant models and sampling times. Figure 6 shows the response of the system over the simulated time frame, showing how both controllers behaved.

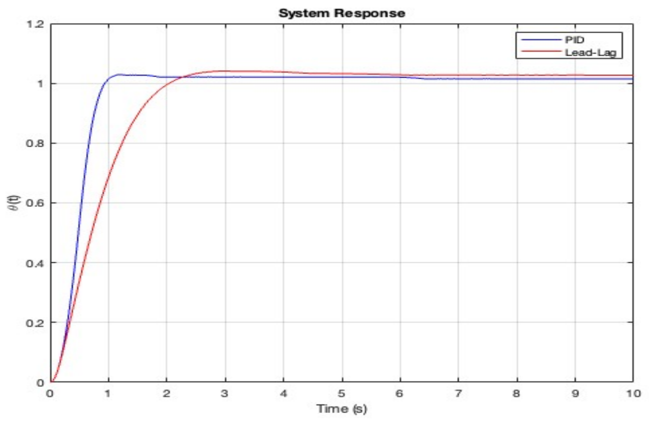


Figure 6: System Response

Figure 6 shows that the PID controller reached steady state quicker than the Lead-Lag controller, but they both seemed to have similar responses to the system. Table 1 shows the performance metrics that were computed for both systems:

When analysing the different metrics, the following comparison can be made:

Table 1: Performance Metrics Comparison between PID and Lead-Lag Controllers

Performance Metric	**PID Values**	**Lead-Lag Values**
Overshoot (%)	1.3444	1.3474
Ramp Error	0.0077447	0.2672
Settling Time (s)	0.92109	2.0997
Rise Time (s)	0.56622	1.3899
Peak Time (s)	1.19	2.97
Peak Value	1.0284	1.0406
Phase Margin (deg)	68.324	72.486
Gain Margin (dB)	26.37	40.752

1. Overshoot (%)
    - Both controllers are shown to exhibit extremely low overshoot, of around 1.34%, indicating that both systems are able to settle to the desired value with minimal oscillation beyond the target. The Lead-Lag controller shows a slightly higher overshoot of 1.3474%, compared to the PID controller with 1.344%, but the difference is essentially negligible.
2. Ramp Error
    - There is visibly a significant difference in the ramp error between both controllers. The PID controller shows an extremely low ramp error of 0.0077447, which suggests that the tracking of a ramp input with minimal steady-state error is excellent.
    - On the other hand, the Lead-Lag controller exhibits a considerably higher ramp error in comparison, with a value of 0.2672, implying that the system will have a noticeable steady-state error when subjected to a ramp input, but the value is still quite small in comparison to other methods, showing a well tuned controller.
3. Settling Time (s)
    - The PID controller is shown to have a much faster settling time of 0.91 seconds, in comparison to the Lead-Lag controller which has a 2.1 second settling time.
    - This means that the PID-controlled system is able to reach and stay within a certain percentage (between 2-5% of its final value) much quicker, showing a robust system.
4. Rise Time (s)
    - Similar to the settling time, the rise time for the PID controller is significantly faster at 0.566 seconds vs. 1.39 seconds of the Lead-Lag (nearly 3 times quicker).
    - This indicated that the PID-controlled system is able to reach its final value more rapidly, and is quicker at controlling the system and achieving steady state in comparison to the Lead-Lag Controller.
5. Peak Time (s)
    - The peak time is the time taken for the response to reach its first peak, which is also seen to be shorter for the PID controller with 1.19 seconds vs. the lead-lag controller with 2.97 seconds.
    - These results align with the faster overall response observed through the PID controller, showing a more robust system.
6. Peak Value
    - The peak value, which is the maximum value reached by the system output is slightly higher for the Lead-Lag controller in comparison to the PID controller, higher by 0.02.

- This result is consistent with the slightly higher overshoot that we observe with the Lead-Lag compensator, meaning that before settling, it reaches a higher value, which may result in potential errors in the real-world implementation, (e.g., a high peak current could stress a motor).

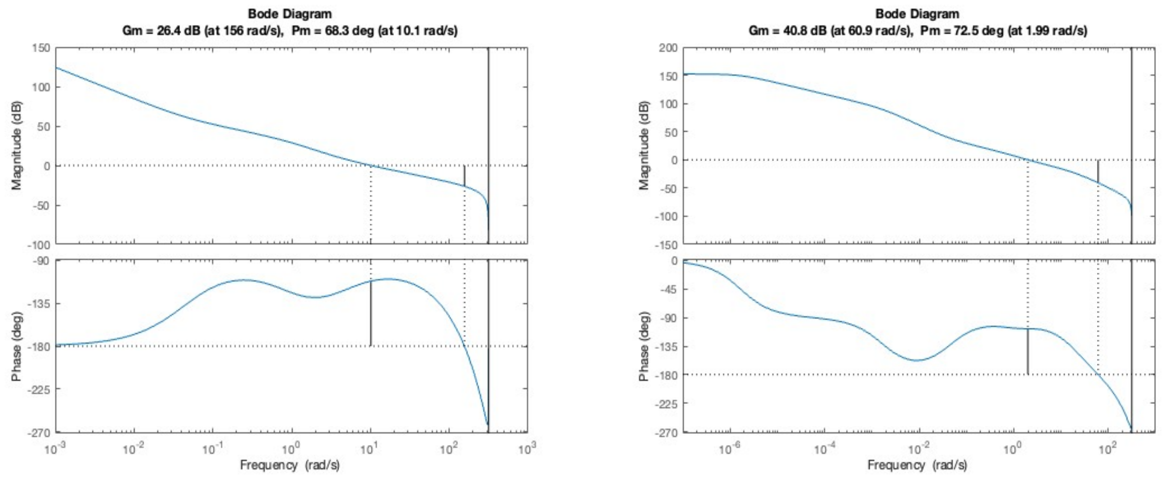
7. Phase Margin (deg)

   - Both controllers are shown to exhibit good phase margins, which is an indication of stable closed-loop systems. The Lead-Lag controller has a higher phase margin of 72.5 degrees compared to the PID controller having 68.3 degrees.
   - The general rule of thumb is that a higher phase margin (usually between 50-60 degrees) suggests better relative stability and damping, which both controllers exceed, but overall, they are both well within a stable range.

8. Gain Margin (dB)

   - The Lead-Lag controller once again, shows a larger gain margin of 40.75 dB compared to the PID controller with 26.37 dB.
   - A higher gain margin is an indication of greater tolerance to changes in system gain before instability occurs, which suggests better robustness to gain variations for the Lead-Lag compensated system.

These performance metrics allow us to draw a number of conclusions. The PID controller is shown to provide a faster and more transient response, while exhibiting significantly lower settling time, rise time, and ramp error. The overshoot is minimal and comparable to the Lead-Lag controller, with the phase and gain margins showing a robust controller - though they are slightly lower than the Lead-Lag controller. The Lead-Lag controller shows to offer better stability margins, showing greater robustness to parameter variations and disturbances. However, this also results in a slower transient response with significantly higher settling time, rise time, and a steady-state error for ramp inputs. The overshoot is marginally higher than PID, but it is still very low.



(a) (PID) (b) (Lead-Lag)

Figure 7: Bode Diagram

The bode plot shown in figure 7 provides a graphical representation of the frequency response of the open-loop system after implementing both controllers. The plot can help in understanding the stability and performance characteristics of the closed-loop control system. The magnitude plot shows the gain of the open-loop system in decibels as a function of frequency, with a higher magnitude indicating a larger amplification of the input signal at that frequency. The phase plot shows the phase shift that is introduced by the open-loop system in degrees as a function of frequency, where a negative phase shift means the output signal lags behind the input signal at that frequency.

Comparing the bode plots and the extracted stability margins for both controllers, we are able to see that they both closed-loop systems are stable - shown through the positive gain and phase margins. With the Lead-Lag having a significantly higher gain margin compared to the PID, the Lead-Lag is a more robust system against increases n the open-loop gain before it becomes unstable. The Lead-Lag also has a higher phase margin, which would indicate that it has better damping and less oscillatory behavior in the closed-loop response. The PID controller is seen to have a much higher gain crossover frequency (by nearly 5x - 10.1 rad/s vs. 1.99 rad/s) which would show that the PID controller has a faster closed-lop response and a wider bandwidth - as the performance metrics show. Therefore, the bode plots can confirm the trends that were observed through Table 1, where the PID controller is shown to provide a faster transient response, and the Lead-LAG is able to provide greater stability and robustness.

Overall, the choice between a PID controller and a Lead-Lag compensator depends on the specific requirements of the application. If a fast and accurate transient response and good tracking of ramp inputs are critical, the PID controller appears to be the better choice. However, if robustness to system variations and disturbances is the primary concern, and a slower response with a potential steady-state error for ramp inputs is acceptable, the Lead-Lag compensator might be preferred.

It is important to note that these are simulated results, and the actual performance in a real-world system might vary due to factors not included in the simulation. The trade-off is between having a robust system to noise (focusing on gains) vs. focusing on a system that responds faster - which will vary based on the specific application of the system. Tuning of the controller parameters will be necessary to optimize the performance of the controller for said applications.

4.2 Challenges

Several challenges were faced during the design of these 2 controllers, such as tuning the controllers, reducing ramp error, and ensuring data was logged accurately.

1. Discretization Method Selection: While the PID controller used MATLAB's internal discrete PID tuning tools, the Lead-Lag compensator had to be manually discretized using the Tustin method. Early trials using Zero-Order Hold (ZOH) yielded poor transient responses due to lag dominance and phase distortion, especially at higher frequencies. Understanding how each method affects frequency response was crucial to ensure a stable and accurate controller could be designed.

2. Tuning Parameters: Adjusting the PID controller required nonlinear scaling of Kp, Ki, and Kd gains to suppress overshoot and improve ramp performance. However, aggressive tuning led to control signal saturation and instability. Similarly, for the Lead-Lag compensator, increasing K_lead and β to minimize ramp error risked reducing phase margin and increasing sensitivity to noise.

3. Simulink Clutter: I programmatically added and connected blocks, which resulted to cluttered diagrams, making it hard to understand if the design was correct. To mitigate this an arrangesystem function had to be applied to ensure that the system was shown in a clean manner as shown in Figure 2 and Figure 4.

4. Data Logging: To ensure that post-simulation analysis was possible, the correct To Workspace block configuration was essential. Initial errors were caused by using the default setting of the block, which caused data to not get logged, and required manual debugging to identify the issue, and understand how to fix it.

5. Continuous Sampling Time: Ensuring that the sample time Ts=0.01 was uniformly applied across all discrete components in both Simulink models and controller definitions was critical. Initial mismatches led to the code not running, having errors being thrown regarding blocks being unable to be connected, and unexpected behavior such as delayed response or instability. To fix this, manual setting of the SampleTime property in every block and controller declaration was required, such that I was overwriting any default inputs.

6. Model Synchronization: To ensure a fair comparison between both controllers, the designs had to be synchronized in terms of plant dynamics and initial conditions. This meant that I had to clone the models, and ensure identical step input parameters, while verifying all plant parameters were the same across the system - especially quantization and saturation.

7. Quantization Effects: Quantizing sensor outputs using a simulated encoder resulted in nonlinear behavior, especially when I was fine-tuning controller gains. This revealed how limited resolution and discretization could degrade real-world control accuracy, placing emphasis on the importance of accounting for practical hardware constraints.

8. Stability and Margins: Designing the system to ensure sufficient phase and gain margins in the discrete domain was non-trivial. After discretization, ensuring a consistent phase boost from the lead components while avoiding excess phase lag from the lag components required iterative frequency response analysis, which was done using bode and margin functions.

9. System Response Time vs. Robustness: Designing a system that is quick and robust is difficult reducing settling time, rise time, and ramp error results in a lower gain and phase margin, which is the trade-off that needs to be taken into account for practical applications. A reduction in stability margins translates to a less robust system, more susceptible to instability due to parameter variations, disturbances, and modeling inaccuracies. There is an inverse relationship between system response time and robustness, with practical engineering often necessitating finding a suitable compromise. Therefore finding the right balance between response time and robustness meant iteratively reconfiguring the controller parameters.

These challenges showed the constraints that arise in implementing controllers in real-world applications and helped in providing ways to debug and fix issues that may arise with hardware constraints. Overall, these challenges provided a learning curve in understanding the design of a controller and how providing limiting constraints can affect the design process and the output.

4.3 Future Work

In the future, to improve the controller designs, a number of different methods can be implemented to make the controllers more robust. Incorporating an anti-windup strategy in the PID controller can help in further improving the system by preventing saturation-related issues. Additionally, integrating real-time adaptive gains or self-turning control can help in handling changing plant dynamics. Furthermore, the system's performance can be put to further test by evaluating the performance under sensor noise and load disturbances to assess its robustness and see how it may respond in the real-world if implemented on some sort of hardware. Finally, implementing a method of system identification integration can help in automatically estimating model parameters from real plant data to close the loop between the model-based and data-driven control design, which could help in validating if the discrete plant truly reflect a real digital implementation of $G\_continous$, which in-turn could allow me to later replace the analytical model with a real-world equivalent when moving from simulation to deployment.

Acknowledgments

Parthiv Kukadia would like to thank Professor Hong Geok Soon for the necessary knowledge to design a controller and ensure it is stable and accurate while also providing comparative metrics.

References

[1] K. J. Åström and T. Hägglund, Advanced PID Control. Research Triangle Park, NC, USA: ISA, 2006.

[2] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, Feedback Control of Dynamic Systems, 8th ed. Upper Saddle River, NJ, USA: Pearson, 2019.

[3] K. Ogata, Modern Control Engineering, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[4] ”Digital Control Systems,” Gayatri Vidya Parishad College of Engineering for Women, Visakhapatnam, India. [Online]. Available: https://www.gvpcew.ac.in/LN-CSE-IT-22-32/EEE/4-Year/DCS.pdf