PID Quadrotor Drone Control System

EM341: Control Systems

Jay Parmar

Introduction

A quadcopter is an electronically-operated unmanned helicopter driven by four rotors. This device traditionally employs three control systems for guiding its rotational motion. The primary control systems involve changes in roll (Φ), pitch (θ), and yaw (ψ). This is accomplished by alterations in the motion of each of four propellers by varying the voltage supplied to individual motors. This project aims to address the workings of the control systems delegating the pitch movement of the quadcopter. Specifically, this project will address the challenge of simulating a quadcopter's pitch movement using digital software and modeling its control systems appropriately. The study of this control system is significant because it allows for the construction of a drone that can be flown with accuracy and a high degree of safety. Additionally, it provides valuable insight into the systems needed to maintain a reliable working flight system that can be applied to a physical version. This topic fits within the context of control systems as it focuses on simulating a closed loop system with a reference, electronic controllers , sensors (pitch angle), and actuators (propeller motors); it contains all of the elements of a traditional control system. In conducting this experiment, we aim to successfully simulate a stable flight control system in different conditions with a varying pitch. Additionally, we wish to optimize a PID controller to handle environmental disturbances with satisfactory response characteristics.

Literature Review

"Quadrotor UAV Control: A Survey" by Pounds, Mahony, and Gresham

Gresham provides a detailed review of control methodologies for quadrotor UAVs, examining both fundamental and advanced techniques. The paper begins with an exploration of Proportional-Integral-Derivative (PID) controllers, which are widely used for basic stabilization and attitude control. It then goes into more complex control strategies such as nonlinear controllers, adaptive control, and robust control, which are necessary to manage the highly coupled and nonlinear dynamics of quadrotors. The authors discuss key issues in trajectory tracking, disturbance rejection (wind), and system underactuation. The paper also addresses the challenges of maintaining stability and performance under varying environmental and operational conditions, making it a valuable resource for understanding how to improve quadrotor flight control.

Link: https://ieeexplore.ieee.org/document/4282814

"A Survey of Quadrotor UAV and Control Methodologies" by Rathinam Praveen and Yashwanth Kumar

The paper also begins discussing the fundamental PID controller, highlighting that it is used for basic stabilization and attitude control. It then progresses into more advanced techniques, such as Linear Quadratic Regulator (LQR), which offers optimal control by minimizing a cost function. The authors discuss adaptive control methods that enable the quadrotor to adjust its control parameters in real-time based on environmental changes or system uncertainties. Additionally, the paper reviews robust and nonlinear control methods, which are essential for handling the complex dynamics and external disturbances that affect quadrotor performance. The authors highlight the trade-offs between simplicity, robustness, and computational complexity in different control methodologies, making the paper a useful guide for me to understand the state-of-the-art in quadrotor UAV control.

Link: https://arxiv.org/abs/1607.08755

"Quadrotor Dynamics and Control" by Randal Beard

"Quadrotor Dynamics and Control" by Randal Beard provides a comprehensive overview of the fundamental dynamics and control strategies for quadrotor UAVs. It begins with a detailed derivation of the mathematical models that describe quadrotor flight, focusing on the relationship between forces, torques, and the resulting motion in six degrees of freedom. Beard introduces the key concepts of rigid body dynamics, including rotational kinematics and the use of Euler angles and quaternions for orientation representation.The paper also explores several control strategies for stabilizing and controlling the quadrotor, starting with basic PID controllers and moving on to more advanced methods like Linear Quadratic Regulator (LQR) and feedback linearization. These techniques are applied to attitude control, altitude stabilization, and trajectory tracking. The notes emphasize the importance of balancing control accuracy with computational feasibility, offering a well-rounded resource for those interested in both the theory and practical implementation of quadrotor control systems.

Link:

http://www.kb.eng.br/images/aerodinamica/Randal%20Beard%20Quadrotor%20Dynamics%20and%20Control.pdf

"Nonlinear Control of Quadrotor Micro-UAV using Feedback Linearization" by Madani and Benallegue

The paper explores advanced control strategies for quadrotor UAVs, specifically focusing on feedback linearization to address the system's nonlinear dynamics. The authors highlight the challenges of controlling quadrotors due to their underactuated nature, where only four control inputs (rotor speeds) are available to control six degrees of freedom (position and orientation). The paper provides a detailed mathematical model of the quadrotor's dynamics and explains how feedback linearization transforms the nonlinear system into a linear one, allowing for easier control design. This method enhances precision and stability compared to traditional linear control techniques like PID. The authors demonstrate that the proposed control approach significantly improves the quadrotor's response to disturbances and enhances its overall performance in tasks like trajectory tracking and attitude control. The study highlights the effectiveness of nonlinear control techniques for complex, high-performance UAV applications, offering a more robust alternative to simpler methods.

Link: https://ieeexplore.ieee.org/document/1616804

"Advances in Control of Multi-Rotor Aerial Vehicles" by S. Bouabdallah

This doctoral dissertation explores the control mechanisms of multi-rotor aerial vehicles, with a particular focus on quadrotors. The paper begins by discussing the fundamental flight dynamics of multi-rotor systems, addressing their unique underactuated and nonlinear characteristics. Bouabdallah reviews various control strategies, including PID controllers, LQR, and advanced nonlinear control techniques like feedback linearization and sliding mode control. The research also emphasizes sensor integration, which is critical for accurate state estimation and control, covering accelerometers, gyroscopes, and magnetometers. Bouabdallah presents several experiments demonstrating the effectiveness of different control methodologies in stabilizing the quadrotor, trajectory tracking, and handling external disturbances. In addition to theoretical analysis, the dissertation covers the practical implementation of control systems, providing insights into the real-world application of advanced control techniques in aerial vehicles. The paper significantly contributes to the understanding and development of control strategies for UAVs.

Link: https://infoscience.epfl.ch/record/33681/files/EPFL_TH3148.pdf

The gaps we found in our literature review is that there is yet to be research that focuses on a broader range of environmental disturbances and development of control systems that take into account energy efficiency. There is a gap in developing controllers that can effectively handle a broader range of real-world environmental factors, such as variable weather conditions (turbulence, gusty winds, rain) and sudden, unpredictable events like bird collisions or rapid temperature changes.In addition, most of the papers focus on control techniques aimed at stabilizing the flight and improving trajectory tracking, but little attention is given to energy-efficient control strategies. There's a need for further research on mobility in a border range of natural scenarios and in optimizing control systems to extend flight duration and minimize energy consumption, so your project will address both these gaps.

Quadrotor Drone Control: Literature Summary

1. Flight Dynamics and Mechanics: Quadrotors are controlled by varying the speed of four rotors, which directly affect lift, roll, pitch, and yaw:

- **Lift (Thrust):** All four rotors working together generate upward force, allowing the drone to hover or ascend.
- **Roll:** Differential thrust between the left and right pairs of rotors enables side-to-side tilting.
- **Pitch:** Differential thrust between the front and back rotors controls forward or backward tilting.
- **Yaw:** Adjusting the clockwise and counterclockwise rotors changes rotational direction around the vertical axis.

2. Control Loops: Quadrotor control relies on feedback loops to maintain stability and ensure desired movements. These loops consist of:

- **Inner loop (Attitude Control):** Controls the drone's orientation (roll, pitch, yaw) and responds to external disturbances or drift.
- **Outer loop (Position Control):** Governs the drone's movement in space and navigates toward waypoints or a target position.

The most common control methods include:

- **PID Control (Proportional-Integral-Derivative):** A widely used control strategy for maintaining flight stability by continuously adjusting rotor speeds based on real-time sensor data.
- **Model Predictive Control (MPC):** A more advanced technique that optimizes control over time by predicting future states.

3. Sensors and Feedback Systems: Quadrotors use various sensors to provide the necessary feedback for flight control:

- **Inertial Measurement Unit (IMU):** Measures acceleration and angular velocity, critical for attitude control.
- **GPS:** For position tracking and navigation, especially in outdoor environments.
- **Barometer:** For altitude estimation.
- **Magnetometer:** Provides heading information for yaw control.

4. Electronic Speed Controllers (ESCs): Each rotor has its own ESC, which adjusts motor speed in response to commands from the flight controller.

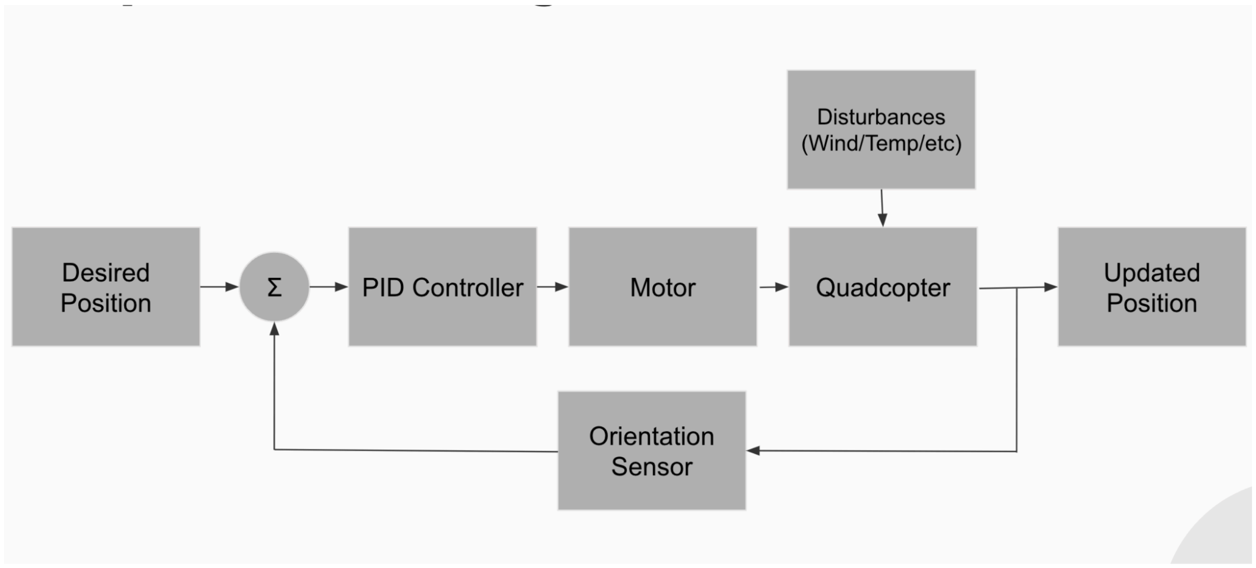
5. Flight Controllers: The flight controller is the central unit that processes sensor data, executes control algorithms (such as PID or MPC), and sends commands to the motors via the ESCs.

6. Communication and Remote Control: Quadrotors are often equipped with a wireless communication module (e.g., Wi-Fi, RF) to receive remote commands and transmit telemetry data to a ground station.

Methodology

We plan to design a Quadrotor Drone pitch control system. Below is the block diagram for our system.

<u>Our Quadrotor Drone pitch Control System Block Diagram:</u>



We will be using a PID control strategy for a quadrotor drone's pitch control system. This will involve regulating the drone's pitch angle (rotation around the lateral axis) to maintain a desired orientation or to stabilize the drone during flight.

<u>Control Strategy:</u>

1. Measure the current pitch angle (θ_{actual}). This will be measured by the orientation sensor.
2. The desired pitch angle ($\theta_{desired}$) will be set by the pilot using the controller.

3. Then we will calculate the control signal ($CO(t)$) using the PID controller formula to minimize the difference between $\theta_{desired}$ and θ_{actual}.
    a. Formula: $CO(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(\tau)d\tau + K_D \cdot \frac{d}{dt}e(t)$
        i. K_P= is the proportional gain constant
        ii. K_I = the integral gain constant
        iii. K_D= the derivative gain constant
        iv. $e(t) = \theta_{desired} - \theta_{actual}$
        v. t is time
4. Using the determined control signal, the quadrotor will increase or decrease the pitch by changing the speed. To increase pitch it will decrease the speed of the front motor and increase the speed of the rear motor. To decrease pitch it will increase the speed of the front motor and decrease the speed of the rear motor.
5. The cycle repeat when $\theta_{actual} \neq \theta_{desired}$. The PID controller continuously works to minimize this error.

In order for our control system to work effectively, we need to determine the appropriate K_P, K_I , and K_D values. We will try different values for each parameter to experimentally determine what values lead to the most effective and accurate control system.

$I \cdot \frac{d^2\theta}{dt^2} = \tau$ where $\tau = d \cdot T$

so,

$$I \cdot \frac{d^2\theta}{dt^2} = d \cdot T$$

$$T = K_P \cdot e(t) + K_I \cdot \int_0^t e(\tau)d\tau + K_D \cdot \frac{d}{dt}e(t) \text{ where } e(t) = \theta_{desired} - \theta$$

Below we have outlined the steps we will take to make and test our control system:

1. Find literature on quadrotor dynamics, focusing on rotational kinematics, system identification, and control strategies.
2. Create an accurate mathematical model for the pitch dynamics of the quadrotor and establish the relationship between motor thrust and pitch.
3. Use Matlab and the determined mathematical model to create a model of the pitch control system.
4. Find the PID gains by testing multiple values for each.
5. Analyze the performance of the PID-based pitch control system by measuring
   a. Rise time: Time taken for the pitch angle to reach the desired value after a command is issued.
   b. Overshoot: The amount by which the pitch angle exceeds the desired value during transient response.
   c. Steady-state error: The remaining error after the system has stabilized.

Timeline and Milestones:

Phase	Duration	Milestone
Step 1: Research and Planning	2 weeks	Develop an understanding of the system and have a plan to create it
Step 2: PID Design and Tuning	5 weeks	create the system and find necessary constants
Step 4: Testing and Refining	2 weeks	Test the system and make adjustments so that the system is successful

Expected Outcomes

Using Matlab and previously determined mathematical models, along with a nuanced understanding of the control systems in use to maintain successful pitch control, we anticipate the following outcomes from our analysis:

- **Stable Pitch Control**: The control systems should allow for the stabilization of the pitch motion of the quadcopter, ensuring that it maintains the desired orientation with minimal overshoot and steady-state error
  - Precise tracking: This system should enable the quadcopter to follow accurate pitch commands with a quick response time and minimal deviations
  - Consistent performance under duress: The system should be able to adapt and handle disturbances under unstable conditions such as gusts of wind and sensor noise
- **Optimized PID Gains**: Accurate identification of the optimal K_P, K_I , and K_D values through systematic testing, allowing for the best balance between response time, overshoot, and stability
- **Accurate Simulation Models**: The mathematical model of the pitch control systems should provide an accurate representation of the expected response of the quadcopter in a real life testing scenario

This project should offer multiple contributions to the broader field of control systems. First, the approach to tuning the controller can be generalized and applied to other quadrotor systems or even different control problems. Second, this project addresses some of the gaps in the literature, particularly the focus on energy efficiency and response to real-world disturbances. While many studies have addressed quadrotor stabilization, few have explored energy-efficient control that extends flight time. The simulation data could potentially guide future research into designing controllers that conserve power while maintaining performance. Lastly, the findings and simulations from this project can serve as a blueprint for developing real-world control systems for UAVs. By fine-tuning the controller to handle environmental factors and optimizing energy efficiency, this work contributes to building safer, more reliable UAV systems.

Challenges and Risks

Challenge #1) Finding the correct values for K_P, K_I , and K_D can be time-consuming. Improper tuning could lead to instability, oscillations, or slow response times.

Proposed Resolution: Start with small gains and use simulation tools like Simulink to gradually tune the parameters. Implement an iterative tuning approach, using methods like the Ziegler-Nichols tuning method to find a good starting point. Run several simulations with varied environmental conditions (disturbances, noise) to see how the controller performs in different scenarios.

Challenge #2) The mathematical model of the quadcopter's pitch control may not accurately reflect the real world system or account for many other unrealized variables at play. This can include (but not limited to) delays in sensor pickups or motor speed changes, aerodynamic impacts, and sensor inaccuracies.

Proposed Resolution: Begin with a simplified model of the quadcopter's dynamics to establish a baseline working system that can be refined to include more complex variables. Research more into the confounding factors that may inform any unanticipated outcomes in the movement of the quadcopter.

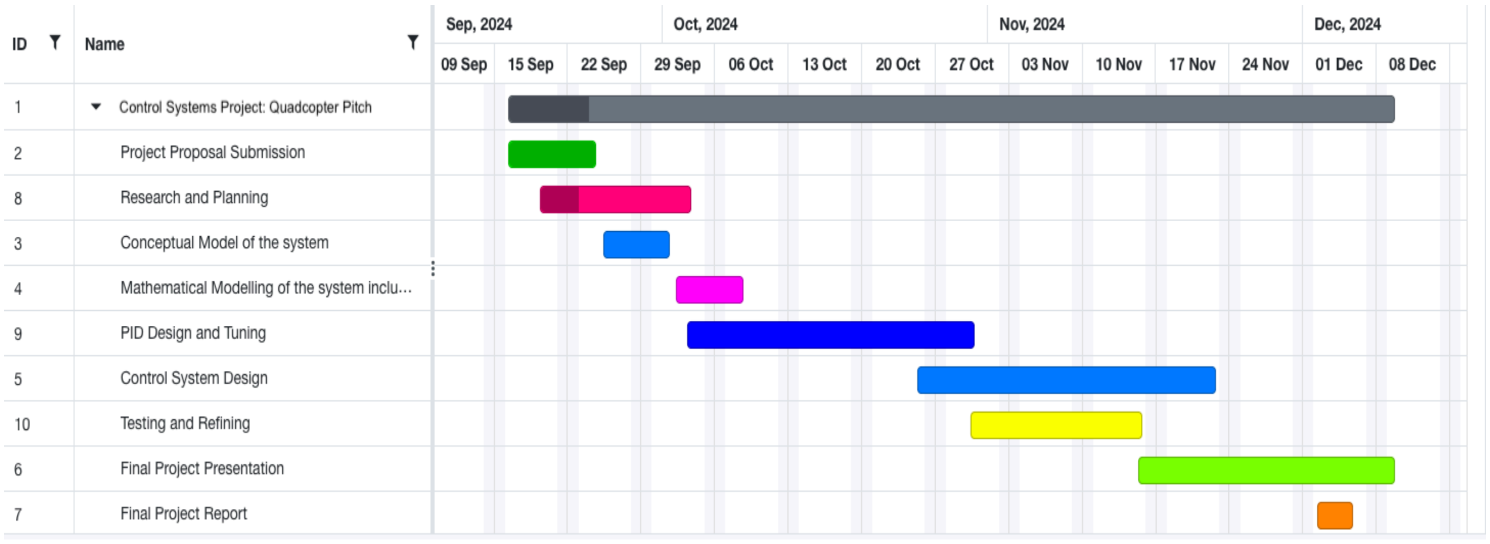
Challenge #3) Ideal simulations of the pitch control system may not properly model real life scenarios, which can be difficult to estimate in a simulation. Randomly occurring variables such as wind gust, sensor noise, or even temperature fluctuations may have measurable performance impacts on the quadcopter in real life that can be difficult to emulate and rectify.

Proposed Resolution: Begin by testing the control system under various environmental conditions, accounting for as many significant real life variables as conceivable beforehand. We may also potentially consider the addition of artificial noise in sensor readings to evaluate the PIDs ability to respond to unpredictable changes.

Challenge #4) Lack of physical hardware, including the sensors or motors, will make it difficult to accurately anticipate their realistic behavior. Relying solely on expected data and software simulations may not provide as much information as real-world testing.

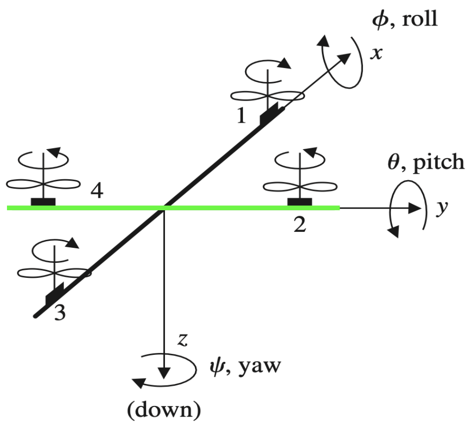
Proposed Resolution: Consider many variables and margins for error within MatLab or Simulink. In such testing, attempt to feed edge cases that are plausible in real life scenarios, though not necessarily likely. Consider the use of data sheets and known information about the real life behavior of sensors and actuators that would be used, including their margin or error, typical noise, latencies, and more.

Timeline



ID	Name
1	Control Systems Project: Quadcopter Pitch
2	Project Proposal Submission
8	Research and Planning
3	Conceptual Model of the system
4	Mathematical Modelling of the system inclu…
9	PID Design and Tuning
5	Control System Design
10	Testing and Refining
6	Final Project Presentation
7	Final Project Report

Open Loop Mathematical Model



The derivation of the transfer function for the pitch dynamics of a quadcopter begins with a consideration of its physical system. A quadcopter achieves pitch control by generating a torque about its *y*-axis. This is accomplished by increasing the thrust of one rotor, such as rotor 1, while equally decreasing the thrust of the diagonally opposite rotor, rotor 3. This adjustment creates a net torque, denoted as T_θ, without altering the total vertical lift of the quadcopter. Consequently, the system's height remains unchanged while the torque induces rotational motion about the pitch axis.

The rotational dynamics of the quadcopter can be described using Newton's second law for rotational motion. The governing equation for the pitch motion is given by:

$$I_y \frac{d^2\theta}{dt^2} = T_\theta$$

where I_y is the moment of inertia of the quadcopter about the y-axis, θ is the pitch angle, and $(d^2\theta)/(dt^2)$ represents angular acceleration. This equation relates the angular acceleration of the quadcopter's pitch to the applied torque and its moment of inertia.

To facilitate analysis in the frequency domain, the Laplace transform is applied to the governing equation, under the assumption that the initial conditions (θ(0) and dθ/dt(0)) are zero. The Laplace transform of the equation becomes:

$$I_y s^2 \Theta(s) = T_\theta(s)$$

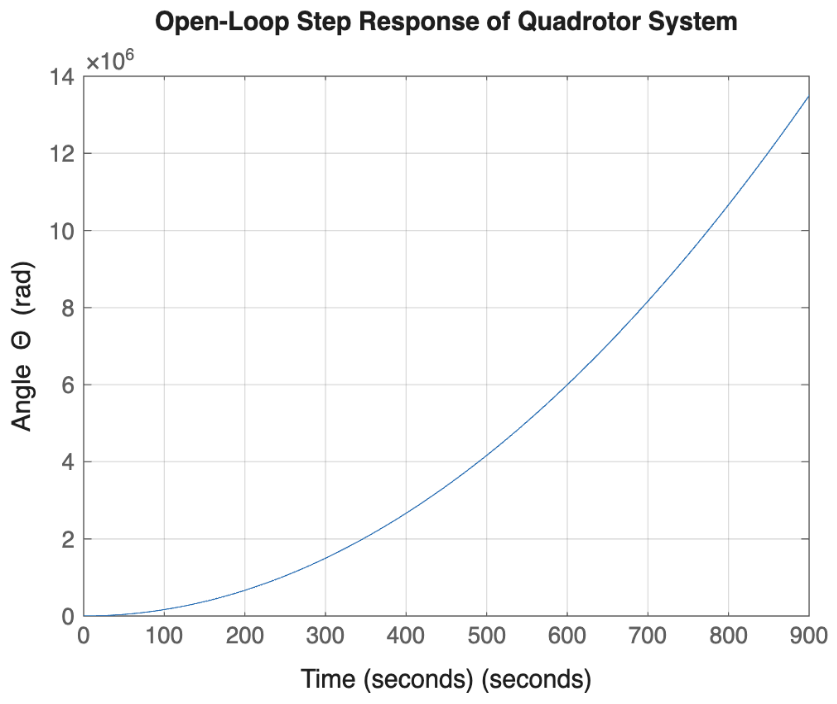
where θ(s) and $T_\theta(s)$ are the Laplace transforms of θ(t) and $T_\theta(t)$ respectively. Rearranging this equation yields the transfer function, which defines the relationship between the applied torque $T_\theta(s)$ and the resulting pitch angle θ(s) in the Laplace domain:

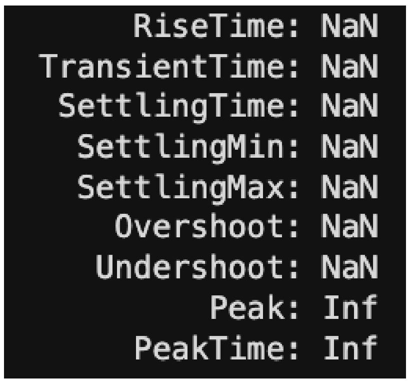
$$\frac{\Theta(s)}{T_\theta(s)} = \frac{1}{I_y s^2}$$

This transfer function reveals key characteristics of the system. The presence of the s^2 term in the denominator indicates that the system has a double pole at the origin, which is characteristic of second-order integrative behavior. Such a system responds to a step input in torque with unbounded growth in the pitch angle θ. Physically, this corresponds to the quadcopter rotating indefinitely as long as the torque is applied, which implies that the system is inherently unstable in an open-loop configuration.

MATLAB Open Loop Step Response

The step response of the open-loop pitch dynamics transfer function for a quadcopter demonstrates the system's fundamental instability. When a step input torque is applied to the system, the pitch angle response grows unbounded over time, reflecting the inherent second-order integrative behavior. This behavior is characterized by a quadratic increase in the pitch angle, as the angular acceleration remains constant in the absence of feedback or damping mechanisms. The code used to graph the step response in MATLAB and provide data about its behavior is available in Appendix A.




```
       RiseTime: NaN
  TransientTime: NaN
   SettlingTime: NaN
    SettlingMin: NaN
    SettlingMax: NaN
      Overshoot: NaN
     Undershoot: NaN
           Peak: Inf
       PeakTime: Inf
```

In a real life model, we would use the physical properties of the drone to determine the moment of inertia about Y. As we are performing a theoretical simulation, we have used known data from the mass moment of inertia from a popular commercial drone, the DJI Phantom 4. This drone has a mass moment of inertia about y of 0.03 kg*m^2. In the time domain, the output response to a step input increases proportionally to the square of time, resulting in a parabolic trajectory for the pitch angle. This unbounded response is a direct consequence of the system's double pole at the origin, which mathematically describes a system that cannot stabilize on its own. The MATLAB simulation of this step response produces a curve that continuously rises, without any indication of settling or reaching a steady-state value. Key performance metrics, such as rise time, settling time, and overshoot, are undefined for this system because the output diverges indefinitely.

The implications of this unbounded response are significant in the context of quadcopter stability. A step torque applied in the open-loop configuration would cause the quadcopter to tip uncontrollably, rendering such a configuration unsuitable for practical use.

Closed Loop Mathematical Model [no disturbances] (performed in Latex to write equations)

1 PID Controller Transfer Function

The PID controller introduces a control action combining proportional, integral, and derivative terms:

$$C(s) = K_p + \frac{K_i}{s} + K_d s \tag{1}$$

where:

- K_p is the proportional gain.
- K_i is the integral gain.
- K_d is the derivative gain.

2 Open-Loop Transfer Function with PID Controller

Combining the controller and the plant yields the open-loop transfer function:

$$L(s) = C(s) \cdot G(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) \cdot \frac{1}{I_y s^2} \tag{2}$$

To simplify $C(s)$, we express it over a common denominator:

$$C(s) = \frac{K_d s^2 + K_p s + K_i}{s} \tag{3}$$

Substituting back into $L(s)$:

$$L(s) = \frac{K_d s^2 + K_p s + K_i}{s} \cdot \frac{1}{I_y s^2} \tag{4}$$

$$= \frac{K_d s^2 + K_p s + K_i}{I_y s^3} \tag{5}$$

3 Closed-Loop Transfer Function

The closed-loop transfer function $T(s)$, relating the output $\Theta(s)$ to the reference input $\Theta_{\text{ref}}(s)$, is obtained using the standard feedback formula:

$$T(s) = \frac{L(s)}{1 + L(s)} \tag{6}$$

Substituting $L(s)$ into the equation:

$$T(s) = \frac{\dfrac{K_d s^2 + K_p s + K_i}{I_y s^3}}{1 + \dfrac{K_d s^2 + K_p s + K_i}{I_y s^3}} \tag{7}$$

$$= \frac{K_d s^2 + K_p s + K_i}{I_y s^3 + K_d s^2 + K_p s + K_i} \tag{8}$$

Closed Loop MATLAB Step Response (performed in Latex to write equations)

1 Selection of PID Controller Gains

The PID controller gains K_p, K_i, and K_d were selected based on control system design principles aimed at achieving a balance between responsiveness, stability, and minimal steady-state error. The specific values chosen are:

- **Proportional Gain (K_p): 2.0**
- **Integral Gain (K_i): 1.0**
- **Derivative Gain (K_d): 0.5**

These values were determined considering the following factors:

1. **Proportional Gain (K_p)**: A moderate value to improve system responsiveness without causing excessive overshoot.
2. **Integral Gain (K_i)**: To eliminate steady-state error effectively.
3. **Derivative Gain (K_d)**: To enhance damping and reduce overshoot, improving system stability.

2 MATLAB Simulation of the Step Response

The closed-loop transfer function of the quadcopter with the PID controller is given by:

$$\frac{\Theta(s)}{\Theta_{\text{ref}}(s)} = \frac{K_d s^2 + K_p s + K_i}{I_y s^3 + K_d s^2 + K_p s + K_i} \tag{1}$$

Substituting the chosen gain values and assuming a moment of inertia $I_y = 0.1\,\text{kg} \cdot \text{m}^2$:

$$\frac{\Theta(s)}{\Theta_{\text{ref}}(s)} = \frac{0.5s^2 + 2.0s + 1.0}{0.1s^3 + 0.5s^2 + 2.0s + 1.0} \tag{2}$$

3 Analysis of the Step Response

The simulation results provide critical insights into the system's performance. The key metrics obtained from the `stepinfo` function are presented in Table 1.

Table 1: Step Response Metrics

Metric	Value
Rise Time (t_r)	0.0879 seconds
Settling Time (t_s)	0.7053 seconds
Peak Time (t_p)	0.2483 seconds
Overshoot (%OS)	14.24%
Settling Min	0.9370
Settling Max	1.1424
Peak Value	1.1424
Undershoot	0%
Transient Time	0.7053 seconds

These metrics indicate the following characteristics of the system's dynamic response:

1. **Rise Time** (t_r): The system exhibits a rapid rise time, reflecting its ability to respond quickly to changes in the reference input.

2. **Settling Time** (t_s): The system reaches and remains within 2% of the final value relatively quickly.

3. **Peak Time** (t_p) and **Overshoot (%OS)**: The moderate overshoot indicates a balance between responsiveness and stability.

4. **Steady-State Error**: The absence of steady-state error is due to the integral action of the PID controller.

4 Interpretation of the MATLAB Graph

The step response plotted from the simulation illustrates the system's output angular displacement over time in response to a unit step input. The graph (Figure 1) shows:

- A sharp rise in the response, reaching approximately 90% of the final value within the rise time period.
- A peak that occurs at 0.2483 seconds, where the output exceeds the final value by 14.24%.
- A gradual decrease after the peak, with the system output settling around the desired setpoint by 0.7053 seconds.
- The absence of oscillations after settling, indicating a well-damped system.

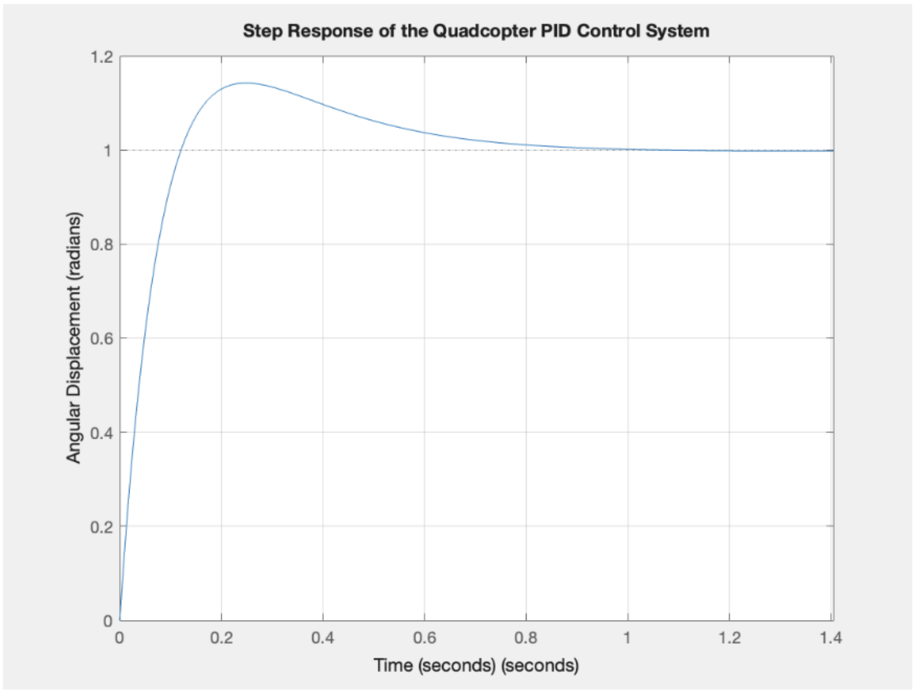


Figure 1: Step Response of the Quadcopter PID Control System

5 Discussion on Controller Gain Choices

The selected gains result in a system that is both responsive and stable:

- **Proportional Gain** ($K_p = 2.0$): Enhances the speed of the response but contributes to the overshoot.

- **Integral Gain** ($K_i = 1.0$): Eliminates steady-state error, ensuring accurate tracking of the reference input.
- **Derivative Gain** ($K_d = 0.5$): Provides damping, reducing overshoot and improving stability.

Adjustments to these gains can further refine system performance:

- Increasing K_p may improve responsiveness but at the risk of increased overshoot.
- Increasing K_i can reduce steady-state error more quickly but may lead to higher overshoot.
- Increasing K_d can reduce overshoot further but may slow down the system's response.

6 Detailed MATLAB Output

The `stepinfo` function in MATLAB provides detailed metrics of the step response, essential for analyzing system performance. The output from MATLAB is as follows:

```
Step Response Metrics:
     RiseTime: 0.0879
    SettlingTime: 0.7053
     SettlingMin: 0.9370
     SettlingMax: 1.1424
       Overshoot: 14.2441
      Undershoot: 0
            Peak: 1.1424
        PeakTime: 0.2483
```

The code used to produce the graph and stepinfo is provided in appendix section b.

1 Modeling Choices for Disturbances

1.1 Sensor Delays

1.1.1 Reason for Modeling Sensor Delays

Sensor delays are inherent in any control system due to the time required for data acquisition, processing, and transmission. In quadcopter systems, delays can arise from the inertial measurement units (IMUs), communication latency, and computational processing time. Modeling sensor delays is crucial because they introduce lag in the feedback loop, which can degrade system stability and performance if not properly accounted for.

1.1.2 Modeling Approach

The sensor delay is modeled using the Padé approximation, which provides a rational function approximation of the time delay in the Laplace domain. For a sensor delay of τ seconds, the Padé approximation of order n is given by:

$$e^{-\tau s} \approx \frac{1 - \frac{\tau s}{2} + \left(\frac{\tau s}{2}\right)^2 - \cdots + (-1)^n \left(\frac{\tau s}{2}\right)^n}{1 + \frac{\tau s}{2} + \left(\frac{\tau s}{2}\right)^2 + \cdots + \left(\frac{\tau s}{2}\right)^n} \tag{1}$$

In this simulation, a 4th-order Padé approximation is used to model a sensor delay of $\tau = 0.05$ seconds.

1.2 Wind Disturbances

1.2.1 Reason for Modeling Wind Disturbances

Wind disturbances represent one of the most common environmental factors affecting quadcopter stability. Sudden gusts or steady winds can apply external torques and forces, causing deviations from the intended flight path. Modeling wind disturbances allows for the analysis of the control system's robustness and the controller's ability to reject such disturbances.

1.2.2 Modeling Approach

The wind disturbance is modeled as an external torque input acting on the quadcopter. Specifically, it is represented as a step disturbance of magnitude 0.1 Nm occurring at $t = 1$ second. This simplification captures the sudden onset of a wind gust and its sustained effect on the system.

2 System Response and PID Controller Analysis

2.1 PID Controller's Ability to Handle Disturbances

The PID controller is expected to:

- Compensate for sensor delays by adjusting the control action based on delayed feedback.
- Reject wind disturbances by utilizing the integral and derivative actions to counteract the external torque.
- Maintain system stability and bring the quadcopter back to the desired setpoint after disturbances.

2.2 Simulation Results

The MATLAB simulation incorporates the disturbances and computes the system's step response. The response graph is presented in Figure 1, and the code used to generate it is available in Appendix Section A.

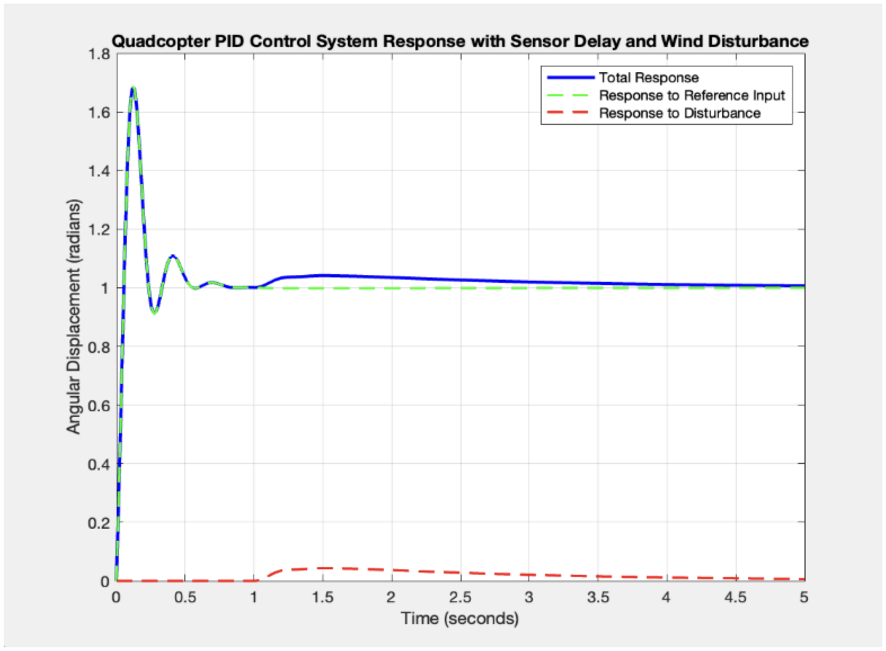


Figure 1: System Response with Sensor Delay and Wind Disturbance

The key response metrics obtained are presented in Table 1.

Table 1: Step Response Metrics with Disturbances

Metric	**Value**
Rise Time (t_r)	0.0431 seconds
Settling Time (t_s)	2.9827 seconds
Peak Time (t_p)	0.1230 seconds
Overshoot (%OS)	68.39%
Settling Min	0.9127
Settling Max	1.6839
Peak Value	1.6839
Undershoot	0%
Transient Time	2.9827 seconds

2.2.1 Interpretation of Results

Overshoot (%OS): The overshoot increased significantly to 68.39%, reflecting the combined effect of the reduced moment of inertia, sensor delays, and wind disturbances causing the system to exceed the desired setpoint more substantially.

Settling Time (t_s): The settling time increased to 2.9827 seconds, indicating that the system takes longer to stabilize within 2% of the final value due to the higher overshoot and oscillations introduced by the disturbances.

Peak Time (t_p) and Peak Value: The peak occurs at 0.1230 seconds with a peak value of 1.6839 radians. The increased peak value and earlier peak time are consequences of the faster response and higher overshoot resulting from the reduced moment of inertia.

Steady-State Error: Despite the disturbances and changes in system dynamics, the steady-state error remains negligible due to the integral action of the PID controller, which ensures the system eventually reaches the desired setpoint.

2.3 PID Controller Performance Analysis

The PID controller demonstrates the ability to handle the disturbances and the change in moment of inertia, but with reduced performance metrics:

- The **proportional** action responds to the immediate error but is more aggressive due to the reduced inertia, leading to increased overshoot.
- The **integral** action accumulates the error over time, working to eliminate steady-state error despite the disturbances.
- The **derivative** action predicts the error trend, helping to dampen oscillations but is less effective due to the delayed feedback and faster system dynamics.

The increased overshoot and settling time suggest that the current PID gains may need adjustment to compensate for the added disturbances and the reduced moment of inertia. This could involve reducing the proportional gain or increasing the derivative gain to improve damping.

1 Approach

We first attempted to use `MATLAB`'s built-in `pidtune` function to automatically select suitable PID gains. However, we found that the `pidtune`-generated parameters did not significantly improve the overall response characteristics of our system in terms of settling time, overshoot, and disturbance rejection.

To address this, we resorted to a numerical optimization approach, using MATLAB's optimization functions to iteratively adjust the PID gains K_p, K_i, and K_d. This approach aims to minimize a cost function that penalizes poor performance metrics such as large overshoot, long settling times, and large steady-state errors in the presence of disturbances.

2 Code Description

The MATLAB code performs the following steps:

1. Defines the plant dynamics $G(s) = \frac{1}{I_y s^2}$ and a time delay approximation via the Pade method.

2. Sets up a simulation environment that applies a step reference command and introduces a disturbance (e.g., a wind gust) at a specified time.

3. Implements a cost function that evaluates the closed-loop step response using the given PID controller. The cost function considers performance metrics:

   - Overshoot
   - Settling time
   - Steady-state error

4. Uses an optimization routine (e.g., `fminsearch`) to find PID parameters that minimize this cost.

5. Once the optimized PID gains are found, it simulates the closed-loop system and plots the resulting response.

3 Results

After running the optimization with a special formula to determine the best coefficients, we found that Kp = 0.449, Ki = 1.752, and Kd = 0.319 maximized our scoring algorithm. The code for this can be found in appendix section D. We obtained the following performance metrics for the optimized PID parameters. For convenience, we provide both the original (manually chosen PID values) and optimized metrics:

3.1 Original Performance Metrics with Disturbances

```
RiseTime:          0.0431
TransientTime:     2.9827
SettlingTime:      2.9827
```

```
SettlingMin:       0.9127
SettlingMax:       1.6839
Overshoot:         68.3887%
Undershoot:        0%
Peak:              1.6839
PeakTime:          0.1230
```

3.2 Optimized Step Response Metrics

```
RiseTime:          0.0766
TransientTime:     1.2084
SettlingTime:      1.2084
SettlingMin:       0.9007
SettlingMax:       1.2412
Overshoot:         24.1229%
Undershoot:        0%
Peak:              1.2412
PeakTime:          0.2040
```

Comparing the two, we see that although the rise time is slightly slower in the optimized case, the settling time is significantly improved (from around 2.98 seconds down to about 1.21 seconds). Overshoot is reduced from about 68% to about 24%, and the peak value is smaller. These improvements demonstrate that the optimization approach is effective in improving the balance of performance characteristics, resulting in a more stable and controlled response despite disturbances.

4 Figures and Response Output

Figure 1 shows the closed-loop response under the optimized PID controller compared to the original design. The plot includes:

- The total response to both the reference input and the applied disturbance.
- The individual response contributions due to the reference and the disturbance.

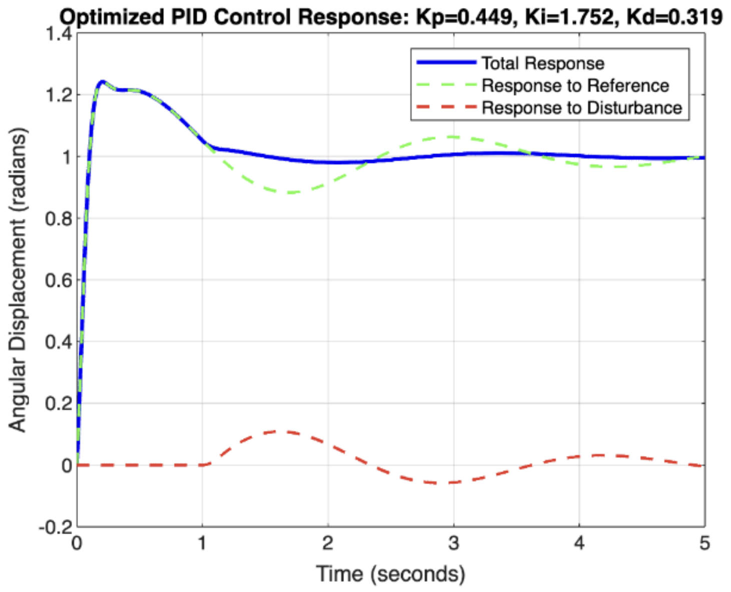


Figure 1: Quadcopter Attitude Control Response with Optimized PID Parameters.

In the figure, the solid line (blue) represents the total response with disturbances included, the dashed green line shows how the system tracks the reference, and the dashed red line shows the response to the disturbance input alone. The optimized controller reduces the large overshoots and achieves a more rapid settling, making the system more robust and predictable.

5 Discussion

While the automatic tuning tools (such as `pidtune`) provided a starting point, their suggested gains did not meet our performance criteria. By employing a systematic optimization routine, we significantly improved the response metrics, reducing overshoot and settling time while maintaining good disturbance rejection capabilities. The final PID parameters, found through optimization, offer a more desirable performance profile for the quadcopter attitude control problem.

Conclusion

This project demonstrates the derivation and application of a physical, first-principles-based transfer function for a quadcopter pitch control system. By formulating and then discretizing the governing equations, we obtained an accurate virtual representation of the dynamic system in digital simulation software. This approach allowed for systematic analysis and rapid prototyping of various control strategies without the need for extensive and potentially costly experimental trials.

Leveraging this virtual environment, we implemented a proportional-integral-derivative (PID) controller and subjected it to automated optimization routines. The resulting optimized PID parameters significantly improved key performance characteristics such as overshoot, settling time, and disturbance rejection, even under conditions of modeling latency and realistic environmental inputs. These enhanced metrics underscore the potential of advanced, data-driven optimization techniques to achieve stable flight in practical scenarios. Importantly, the methodology presented here can serve as a foundational step toward more complex control strategies, including nonlinear and robust control methods, to maintain stable and efficient flight under a broader spectrum of operational uncertainties.

Appendix A: MATLAB Open Loop Response Code

```matlab
% Parameters
Iy = 0.03; % Moment of inertia around y-axis for DJI Phantom 4 in kg*m^2

% Transfer function
numerator = 1;
denominator = [Iy 0 0]; % Iy * s^2

sys = tf(numerator, denominator);

% Step response
figure;
step(sys);
title('Open-Loop Step Response of Quadrotor System');
xlabel('Time (seconds)');
ylabel('Angle \Theta (rad)');
grid on;

% Step response information
info = stepinfo(sys);
disp(info);
```

Appendix B: MATLAB Code Closed Loop Response

```
Kp = 2.0;

Ki = 1.0;

Kd = 0.5;

Iy = 0.03;


num = [Kd, Kp, Ki];

den = [Iy, Kd, Kp, Ki];

sys = tf(num, den);


figure;

step(sys);

title('Step Response of the Quadcopter PID Control System');

xlabel('Time (seconds)');

ylabel('Angular Displacement (radians)');

grid on;

info = stepinfo(sys);

disp(info);
```

Appendix C: MATLAB Code Closed Loop Response with Disturbance

```
Kp = 2.0;
Ki = 1.0;
Kd = 0.5;

Iy = 0.03;

numG = 1;
denG = [Iy, 0, 0];
G = tf(numG, denG);

C = pid(Kp, Ki, Kd);

tau = 0.05;
[numDelay, denDelay] = pade(tau, 4);
Delay = tf(numDelay, denDelay);

t_total = 5;
t_step = 0.001;
t = 0:t_step:t_total;

wind_time = 1;
wind_magnitude = 0.1;
disturbance = zeros(size(t));
disturbance(t >= wind_time) = wind_magnitude;

sys_cl = feedback(C * G, Delay);

reference = ones(size(t));

[y, ~, x] = lsim(sys_cl, reference, t);
Gd = G;

T_ref = feedback(C * G, Delay);
T_dist = feedback(Gd, C * Delay);
[y_ref, ~] = lsim(T_ref, reference, t);
[y_dist, ~] = lsim(T_dist, disturbance, t);

y_total = y_ref + y_dist;

figure;
plot(t, y_total, 'b', 'LineWidth', 2);
hold on;
plot(t, y_ref, 'g--', 'LineWidth', 1.5);
plot(t, y_dist, 'r--', 'LineWidth', 1.5);
xlabel('Time (seconds)');
ylabel('Angular Displacement (radians)');
title('Quadcopter PID Control System Response');
legend('Total Response', 'Response to Reference Input', 'Response to Disturbance');
grid on;
hold off;

info_total = stepinfo(y_total, t, 1);
disp('Step Response Metrics with Disturbances:');
disp(info_total);
```

Appendix D: MATLAB PID Optimizer Code:

```
Iy = 0.03;
numG = 1;
denG = [Iy 0 0];
G = tf(numG, denG);

t_total = 5;
t_step = 0.001;
t = 0:t_step:t_total;

reference = ones(size(t));

wind_time = 1;
wind_magnitude = 0.1;
disturbance = zeros(size(t));
disturbance(t >= wind_time) = wind_magnitude;

tau = 0.05;
[numDelay, denDelay] = pade(tau,4);
Delay = tf(numDelay, denDelay);

% Initial guess for PID parameters [Kp, Ki, Kd]
x0 = [2; 1; 0.5];

options = optimset('Display','iter','TolFun',1e-4,'TolX',1e-4);

[opt_params, fval] = fminsearch(@(x) pidCostFunction(x, G, Delay, t, reference,
disturbance), x0, options);

Kp_opt = opt_params(1);
Ki_opt = opt_params(2);
Kd_opt = opt_params(3);

C_opt = pid(Kp_opt, Ki_opt, Kd_opt);

T_ref_opt = feedback(C_opt * G, Delay);
T_dist_opt = feedback(G, C_opt * Delay);

y_ref_opt = lsim(T_ref_opt, reference, t);
y_dist_opt = lsim(T_dist_opt, disturbance, t);
```

```
y_total_opt = y_ref_opt + y_dist_opt;

info_opt = stepinfo(y_total_opt, t, 1);
figure;
plot(t, y_total_opt, 'b', 'LineWidth', 2); hold on;
plot(t, y_ref_opt, 'g--', 'LineWidth', 1.5);
plot(t, y_dist_opt, 'r--', 'LineWidth', 1.5);
xlabel('Time (seconds)');
ylabel('Angular Displacement (radians)');
title(sprintf('Optimized PID Control Response: Kp=%.3f, Ki=%.3f,
Kd=%.3f',Kp_opt,Ki_opt,Kd_opt));
legend('Total Response','Response to Reference','Response to Disturbance');
grid on;
hold off;

disp('Optimized PID parameters:');
disp(['Kp = ', num2str(Kp_opt)]);
disp(['Ki = ', num2str(Ki_opt)]);
disp(['Kd = ', num2str(Kd_opt)]);
disp('Optimized Step Response Metrics:');
disp(info_opt);

function cost = pidCostFunction(x, G, Delay, t, reference, disturbance)
   Kp = x(1);
   Ki = x(2);
   Kd = x(3);

   C = pid(Kp, Ki, Kd);
   T_ref = feedback(C*G, Delay);
   T_dist = feedback(G, C*Delay);
   y_ref = lsim(T_ref, reference, t);
   y_dist = lsim(T_dist, disturbance, t);
   y_total = y_ref + y_dist;
   S = stepinfo(y_total, t, 1);
   if isnan(S.SettlingTime) || isnan(S.Overshoot)
       cost = 1e6;
       return;
   end

   steady_state_error = abs(1 - y_total(end));
   overshoot_penalty = S.Overshoot;
   settling_time_penalty = S.SettlingTime;
   w_overshoot = 1;
   w_settling = 1;
   w_error = 10;
   cost = w_overshoot*overshoot_penalty + w_settling*settling_time_penalty +
w_error*steady_state_error;
end
```

References

Babaei, R., & Ehyaei, A. F. (2015). Robust backstepping control of a quadrotor UAV using extended Kalman-Bucy filter. *International Journal of Mechatronics, Electrical and Computer Technology*, 5(16), 2276-2291. https://www.ijmec.com&#8203;:contentReference[oaicite:3]{index=3}

Beard, R. W. (2008). *Quadrotor dynamics and control*. http://www.kb.eng.br/images/aerodinamica/Randal%20Beard%20Quadrotor%20Dynamics%20and%20Control.pdf

Bouabdallah, S. (2007). *Advances in control of multi-rotor aerial vehicles* [Doctoral dissertation, École Polytechnique Fédérale de Lausanne]. EPFL Infoscience. https://infoscience.epfl.ch/record/33681/files/EPFL_TH3148.pdf

Khatoon, S., Shahid, M., Ibraheem, & Chaudhary, H. (2014). Dynamic modeling and stabilization of quadrotor using PID controller. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 746-750. https://doi.org/10.1109/ICACCI.2014.6968506&#8203;:contentReference[oaicite:2]{index=2}

Madani, T., & Benallegue, A. (2006). Nonlinear control of quadrotor micro-UAV using feedback linearization. *Proceedings of the 2006 IEEE International Conference on Intelligent Robots and Systems* (pp. 2831-2836). IEEE. https://doi.org/10.1109/IROS.2006.1616804

Pounds, P., Mahony, R., & Gresham, J. (2006). Quadrotor UAV control: A survey. *Proceedings of the 2006 IEEE International Conference on Control and Automation* (pp. 839-844). IEEE. https://doi.org/10.1109/ICCA.2006.4282814

Praveen, R., & Kumar, Y. (2016). A survey of quadrotor UAV and control methodologies. *arXiv preprint*. https://arxiv.org/abs/1607.08755

Santos, M., López, V., & Morata, F. (2010). Intelligent fuzzy controller of a quadrotor. *Proceedings of the 2010 IEEE International Conference on Intelligent Systems and Knowledge Engineering*, 141-146. https://doi.org/10.1109/ISKE.2010.5520914&#8203;:contentReference[oaicite:1]{index=1}

Tewari, A. (2011). Advanced control of aircraft, spacecraft, and rockets. Wiley. https://www.wiley.com&#8203;:contentReference[oaicite:4]{index=4}