Quadrotor Drone Control System

Optimizing Quadrotor stability through basic Control Theory

Author: Nil Trujillo Albert

ntrujialb@gmail.com

January 29, 2025

1 Introduction

Quadrotor drones have gained significant attention in both research and industry due to their versatility, agility, and wide range of applications, including aerial surveillance, package delivery, and autonomous navigation. However, their unstable dynamics require an effective control strategy to maintain stability and ensure accurate movement. Controlling a quadrotor involves solving a complex nonlinear system where multiple control inputs must be adjusted in real time to regulate a desired position.

The primary goal of this project is to design, model, and simulate a stable system for a quadrotor drone using an appropriate controller. The focus is on ensuring that the drone can maintain a stable altitude and orientation (roll, pitch, and yaw) under different input conditions, serving as a fundamental step toward an advanced trajectory control.

However, this project does not yet address trajectory tracking, which remains a key challenge in quadrotor control. The stabilization system developed here lays the groundwork for future improvements, where trajectory planning and autonomous navigation can be explored as the next step.

This project show how even basic control techniques can be used to stabilize a complex dynamic system ensuring stable flight. Future work can focus on integrating path-following algorithms and real-time adaptability, ultimately moving toward a fully autonomous quadrotor control system.

2 3D Modeling of the quadrotor drone

The 3D modeling of the quadrotor drone serves as a fundamental step in the physical design along with the mathematical models and simulations required for the development of this project. This model provides a representation of the drone's structure, enabling accurate calculations of key parameters such as mass distribution, moments of inertia, and aerodynamic properties. These parameters are key inputs for the mathematical controller proposed in this document designed to stabilize the drone's yaw, roll, pitch, and altitude.

The quadrotor drone has been modeled using SolidWorks as a preliminary design that closely resembles the real physical structure but is not intended as the final design for manufacturing. This 3D model serves as a foundation to accurately simulate and analyze the drone's behavior, providing the necessary physical parameters for the precise and efficiency of the controller.

Figure 1: Drone assembly from Solid Works

While the model captures the essential characteristics of the drone, its primary purpose is to support the development and testing of the yaw, roll, pitch, and altitude controllers. These controllers are the main focus of this project, and the 3D model ensures that their design is based on realistic and reliable data, bridging the gap between theoretical modeling and practical implementation.

Components	**Material**	**Mass**
Central frame	Aluminium 6061-T4	0.76 Kg
Propeller guards	Aluminium 6061-T4	0.07 Kg
Propellers	ABS	0.01 Kg

Table 1: List of materials

I_{xx}	I_{yy}	I_{zz}
0.01	0.01	0.01

Table 2: Principal moments of inertia $[m^4]$

3 Drone Dynamics

To describe the dynamics of a drone, we define two coordinate systems:

- **Global (Inertial) Frame A:** This frame remains fixed and is used to describe the position and orientation of the drone from a determined position (eg. ground). It is denoted as (x, y, z).
- **Local (Body-Fixed) Frame B:** This frame moves with the drone and is centered at its center of mass (CM). It is used to describe forces, torques, and angular velocities acting on the drone. It is denoted as (X', Y', Z').

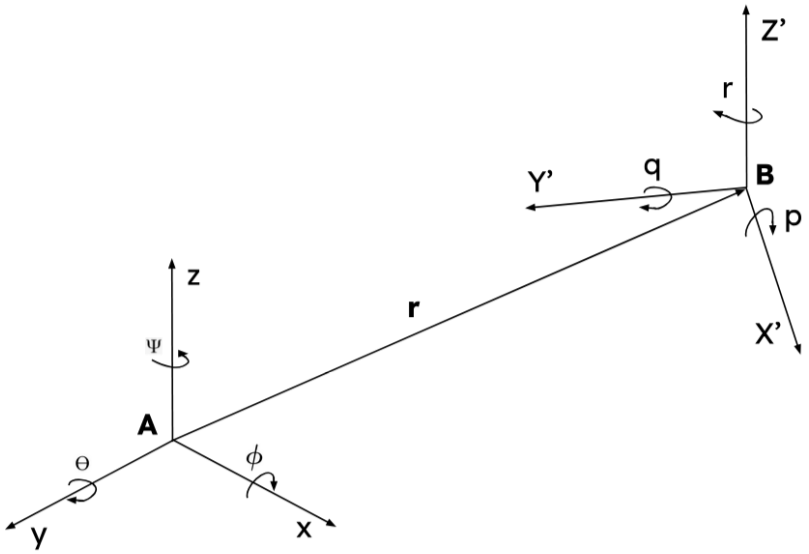

Figure 2: A and B coordinate systems

The state of the drone is represented by the generalized coordinates:

$$\vec{q} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]$$

where:

- x, y, z: Position of the drone's center of mass (CM) respect to the global frame.
- ϕ, θ, ψ: Angles of rotation (roll, pitch and yaw) that describe the orientation of the drone relative to the global frame.
- $\dot{x}, \dot{y}, \dot{z}$: Translational velocities in the local frame.
- p, q, r: Angular velocities in the local frame.

The orientation of the drone is described using a ZYX Euler-angle rotation. The rotation matrix from the local (body-fixed) frame to the global (inertial) frame is given by:

$$\boldsymbol{R}_B^A = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi), \tag{1}$$

where the individual rotation matrices are:

$$\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2}$$

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3}$$

$$\boldsymbol{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Combining (2), (3), (4), the full rotation matrix $\boldsymbol{R}_B^A$ is:

$$\boldsymbol{R}_B^A = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{5}$$

where $c_\phi = \cos\phi$, $s_\phi = \sin\phi$, $c_\theta = \cos\theta$, $s_\theta = \sin\theta$, $c_\psi = \cos\psi$, and $s_\psi = \sin\psi$.

The angular velocities in the local frame, denoted by (p, q, r), are related to the rates of change of the Euler angles $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ through the following relationship:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{W} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{6}$$

where $\mathbf{W}$ is given by:

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \tag{7}$$

3.1 Equations of Motion using the Newton-Euler approach

Using Newton's second law, the translational equations of motion for the drone are expressed as:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \boldsymbol{R}_B^A \begin{bmatrix} 0 \\ 0 \\ F_T \end{bmatrix} - m \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{8}$$

where:

- $F_T = \sum_{i=1}^{4} F_i$: Total thrust generated by the propellers.
- m: Mass of the drone.
- g: Gravitational acceleration.

Using Euler's equation, the rotational dynamics are described by:

$$\boldsymbol{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \boldsymbol{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{9}$$

where:

- $\boldsymbol{I}$: Inertia matrix of the drone with symmetry.
- M_x, M_y, M_z: Moments about the roll, pitch, and yaw axes, respectively.
- p, q, r: Angular velocities in the local (body-fixed) frame.

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

The total thrust and torques with respect to the local coordinate system can be calculated as follows:

$$F_T = \sum_{i=1}^{4} F_i = F_1 + F_2 + F_3 + F_4 \tag{10}$$

where F_i represents the thrust generated by the four propellers.

$$\sum M_x = 0 \implies M_x = L(F_2 - F_4) \tag{11}$$

$$\sum M_y = 0 \implies M_y = L(F_3 - F_1) \tag{12}$$

$$\sum M_z = 0 \implies M_z = L(F_1 - F_2 + F_3 - F_4) \tag{13}$$

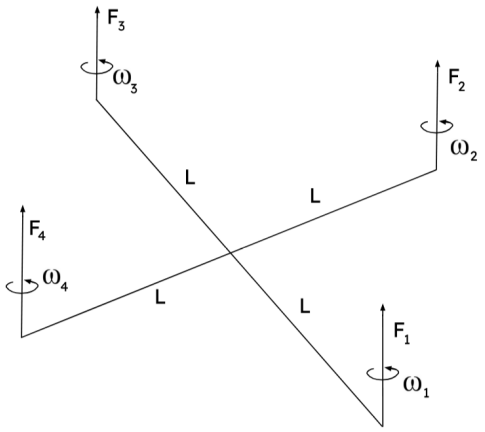

Figure 3: Thrust diagram of the drone

- L: Distance from the center of mass to each rotor in the local coordinate system.

The relationship between the components $[F_T, M_x, M_y, M_z]^T$ and the individual rotor forces $[F_1, F_2, F_3, F_4]^T$ can be expressed as:

$$\begin{bmatrix} F_T \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ L & -L & L & -L \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{14}$$

3.2 Equations of Motion Using the Euler-Lagrange Approach

To design an appropriate controller, the equations of motion can be expressed in a compact form using the Euler-Lagrange approach. The governing equation is:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_{i,\mathrm{NC}} \tag{15}$$

where $L = T - V$ is the Lagrangian, T is the kinetic energy, V is the potential energy, and $Q_{i,\mathrm{NC}}$ represents the non-conservative forces. Expanding the Euler-Lagrange equation, we get:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = Q_{i,\mathrm{NC}} \tag{16}$$

The potential and kinetic energy of the system are given by:

$$T = \frac{1}{2}m\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) + \frac{1}{2}\left[I_{xx}p^2 + I_{yy}q^2 + I_{zz}r^2\right] \tag{17}$$

$$V = -mgz \tag{18}$$

where:

- m: Mass of the drone.
- g: Gravitational acceleration.

Using the Lagrangian approach, the translational equations of motion are derived as:

$$m\ddot{x} = 0, \quad m\ddot{y} = 0, \quad m\ddot{z} + mg = F_T \tag{19}$$

which can be compactly written as before:

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \boldsymbol{R}_B^A\begin{bmatrix}0\\0\\F_T\end{bmatrix} - \begin{bmatrix}0\\0\\mg\end{bmatrix} \tag{20}$$

Substituting the components of the rotation matrix $\boldsymbol{R}_B^A$, we get:

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = F_T\begin{bmatrix}c_\psi s_\theta c_\phi + s_\psi s_\phi\\ s_\psi s_\theta c_\phi - c_\psi s_\phi\\ c_\theta c_\phi\end{bmatrix} - \begin{bmatrix}0\\0\\mg\end{bmatrix} \tag{21}$$

For the rotational motion, Euler-Lagrange's equation gives us:

$$I_{xx}\dot{p}\frac{d}{dt}\frac{\partial p}{\partial \dot{q}_i} + I_{yy}\dot{q}\frac{d}{dt}\frac{\partial p}{\partial \dot{q}_i} + I_{zz}\dot{r}\frac{d}{dt}\frac{\partial p}{\partial \dot{q}_i} - I_{xx}p\frac{\partial p}{\partial q_i} + I_{yy}q\frac{\partial q}{\partial q_i} + I_{zz}r\frac{\partial r}{\partial q_i} = M_i \tag{22}$$

where M_i represents the moments about the respective axes.

This expands into individual equations for ϕ, θ, and ψ.
Reorganizing these, the translational and rotational dynamics can be expressed as a second order differential equation with matrix form:

$$\boldsymbol{A}\frac{d^2\boldsymbol{\chi}}{dt^2} + \boldsymbol{B}\frac{d\boldsymbol{\chi}}{dt} = \boldsymbol{M} \tag{23}$$

where:

$$\boldsymbol{\chi} = \begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix}M_x\\ M_y\\ M_z\end{bmatrix} \tag{24}$$

and the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are defined as:

$$\boldsymbol{A} = \begin{bmatrix} I_{xx} & 0 & -I_{xx}S_\theta \\ 0 & I_{yy}C_\phi^2 + I_{zz}S_\phi^2 & (I_{yy} - I_{zz})C_\phi S_\phi C_\theta \\ -I_{xx}S_\theta & (I_{yy} - I_{zz})C_\phi S_\phi C_\theta & I_{xx}S_\theta^2 + I_{yy}S_\phi^2 C_\theta^2 + I_{zz}C_\phi^2 C_\theta^2 \end{bmatrix} \tag{25}$$

$$\boldsymbol{B} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{26}$$

The elements C_{ij} of $\boldsymbol{B}$ are detailed as:

$$C_{11} = 0 \tag{27}$$

$$C_{12} = (I_{yy} - I_{zz})(\dot{\phi}C_\phi S_\phi + \dot{\psi}S_\phi^2 C_\phi) + (I_{zz} - I_{yy})\dot{\psi}C_\phi^2 S_\phi - I_{xx}\dot{\psi}C_\theta \tag{28}$$

$$C_{13} = (I_{zz} - I_{yy})\dot{\psi}C_\phi S_\phi C_\theta \tag{29}$$

$$C_{21} = (I_{zz} - I_{yy})(\dot{\phi}C_\phi S_\phi + \dot{\psi}S_\phi C_\theta) + (I_{yy} - I_{zz})\dot{\psi}C_\phi^2 C_\theta + I_{xx}\dot{\psi}C_\theta \tag{30}$$

$$C_{22} = (I_{zz} - I_{yy})\dot{\phi}C_\phi S_\phi \tag{31}$$

$$C_{23} = -I_{xx}\dot{\psi}S_\theta C_\theta + I_{yy}\dot{\psi}S_\phi^2 S_\theta C_\theta + I_{zz}\dot{\psi}C_\phi^2 S_\theta C_\theta \tag{32}$$

$$C_{31} = (I_{yy} - I_{zz})\dot{\psi}C_\phi S_\phi - I_{xx}\dot{\theta}C_\theta \tag{33}$$

$$\begin{aligned} C_{32} = {} & (I_{zz} - I_{yy})(\dot{\phi}C_\phi S_\phi + \dot{\psi}S_\phi^2 C_\theta) \\ & + (I_{yy} - I_{zz})\dot{\phi}C_\phi^2 C_\theta + I_{xx}\dot{\theta}S_\theta C_\theta \\ & - I_{yy}\dot{\psi}S_\phi^2 S_\theta C_\theta - I_{zz}\dot{\psi}C_\phi^2 S_\theta C_\theta \end{aligned} \tag{34}$$

$$C_{33} = (I_{yy} - I_{zz})\dot{\phi}C_\phi S_\phi C_\theta^2 - I_{yy}\dot{\theta}S_\phi^2 C_\theta S_\theta - I_{zz}\dot{\theta}C_\phi^2 C_\theta S_\theta + I_{xx}\dot{\theta}C_\theta S_\theta \tag{35}$$

These equations provide a compact and structured representation of the drone's dynamics, making them suitable for designing our desired controller. The matrix form allows for efficient numerical computation and facilitates integration.

4 PID Controller design

In control engineering, the PID (Proportional-Integral-Derivative) controller is one of the most commonly used methods for ensuring system stability and precision. It consists of three components: a proportional term (P) that reacts to the current error, an integral term (I) that corrects accumulated past errors, and a derivative term (D) that anticipates future errors by analyzing the rate of change. The combination of these three terms allows the system to adjust dynamically to changes and disturbances, making it ideal for our system that require real-time stability and fast response.

A PID controller operates by continuously calculating the error $e(t)$, which represents the difference between the desired setpoint $r(t)$ and the actual system output $y(t)$:

$$e(t) = r(t) - y(t) \tag{36}$$

The control signal $u(t)$, which determines the input to the system (e.g., motor thrust adjustments), is computed as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{37}$$

where:

- K_p is the proportional gain, which scales the present error.
- K_i is the integral gain, which accumulates past errors to eliminate steady-state error.
- K_d is the derivative gain, which anticipates future errors by evaluating the rate of change of the error signal.

Using the Laplace transform, the PID equation can be rewritten in the frequency domain:

$$U(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) E(s) \tag{38}$$

Expressing this as a transfer function:

$$C(s) = K_p + \frac{K_i}{s} + K_d s \tag{39}$$

The closed-loop transfer function of the system, assuming a unity feedback configuration, is given by:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)} \tag{40}$$

where $G(s)$ is the plant transfer function representing the system dynamics.

In this project, PID controllers are applied separately to control the four main state variables of the quadrotor.
The parameters K_p, K_i, and K_d are adjusted using trial and error, adjusting parameters iteratively based on performance metrics like overshoot, settling time, and steady-state error.

4.1 Controlling the Drone's Position and Orientation

The quadrotor drone operates by continuously monitoring its state, detecting deviations from the desired trajectory, and applying corrective actions through thrust adjustments in each rotor. The system begins by obtaining sensor data to determine its current state:

$$x(t) = [\phi, \theta, \psi, z]$$

where ϕ (roll), θ (pitch), and ψ (yaw) define the drone's orientation, while z represents its altitude. The desired reference state is given by:

$$x_d(t) = [\phi_d, \theta_d, \psi_d, z_d]$$

The error is then computed as:

$$e(t) = x_d(t) - x(t),$$

which serves as the input to the PID controller:

$$F_T = u_z, \quad M_x = u_{roll}, \quad M_y = u_{pitch}, \quad M_z = u_{yaw}.$$

The controller calculates the corrective action necessary to minimize this error using equation (37) computing the required forces and torques. These forces and torques are then introduced into the equations of motion of the quadrotor governed by equations (21) , (23). The updated orientation and position of the quadrotor are obtained by numerically integrating these equations.

Once the total thrust F_T and moments (M_x, M_y, M_z) are calculated, they must be distributed among the four rotors. The force-moment relationship of the quadrotor is given by equation (14) where each rotor force F_i is related to the corresponding angular velocity ω_i by:

$$F_i = K_f \omega_i^2$$

where $K_f = \frac{1}{2}\rho A C_t R^2$ is the aerodynamic force coefficient, with ρ representing air density, A the blade rotor area, C_t the thrust coefficient, and R the radius of the rotor.

The necessary rotor speeds are thus given by:

$$\omega_i = \sqrt{\frac{F_i}{K_f}}$$

The computed rotor speeds ω_i are sent as commands to the motors, adjusting their rotation rates accordingly. This process runs iteratively in a continuous feedback loop, where the drone continuously updates its position and orientation, detects new errors, recalculates forces and torques, and adjusts rotor speeds to maintain stability. Through this iterative correction, the PID controller effectively minimizes the error and ensures smooth stabilization of the drone.

5 Simulation results

To evaluate the performance of the implemented control system, the quadrotor dynamics were simulated using MATLAB/Simulink. The simulation framework was designed to numerically solve the equations of motion, incorporating the forces and torques computed by the control algorithm. This allowed an easy visualization and analysis of the drone's response under different input conditions and disturbances.

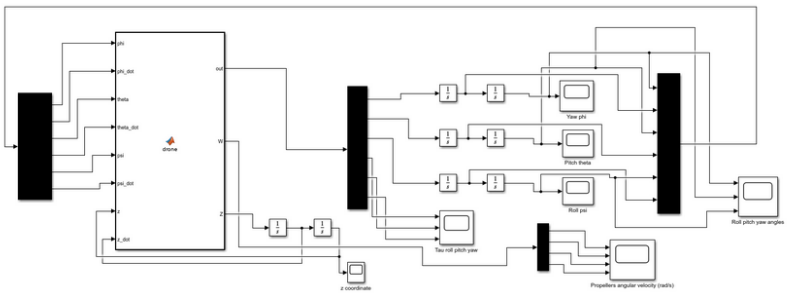

Figure 4: Simulink diagram

To test the effectiveness of our control strategy, the system was subjected to different test cases, including:

- **Step response evaluation:** Where the quadrotor is commanded to reach a specific altitude or orientation.
- **Sinusoidal tracking:** To test the system's ability to follow periodic reference inputs.
- **Pulse generator response:** Where the drone is exposed to periodical perturbations to evaluate how well the controller restores stability.

The simulations revealed that the Proportional-Integral (PI) controller fails to stabilize the system due to the accumulation of the integral term, which leads to oscillatory and unstable behavior.

On the other hand, the Proportional-Derivative (PD) controller provides the best performance, offering a fast response with minimal overshoot and strong disturbance rejection. The PID controller performs slightly worse than the PD, as the integral term introduces additional correction that is not required, slightly reducing the system's responsiveness.

Based on these results, the PD controller is chosen as the optimal control strategy for this quadrotor system, achieving rapid stabilization and smooth tracking of desired trajectories.

Controller	Overshoot (%)	Settling Time (s)	Stability
P	High	Fast but oscillatory	Unstable
PI	Very High	Slow, oscillatory	Unstable
PD	Low	Fast	Stable
PID	Moderate	Slightly slower than PD	Stable

Table 3: Comparison of control strategies for quadrotor stabilization

Coordinate	Kp	Kd
Z (Altitude)	45	10
Roll	45	12
Pitch	45	12
Yaw	45	12

Table 4: PD Controller Gains for Each Coordinate

The following figures illustrate the simulation results obtained for different input signals, allowing us to test the effectiveness of the system's response. By analyzing these results, we can evaluate the controller's ability to track various reference movements, stabilize the system, and minimize errors under different conditions.

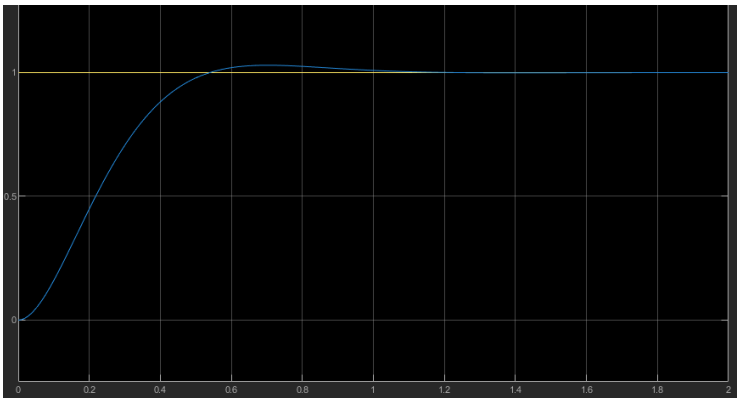

Figure 5: Step response for the z coordinate

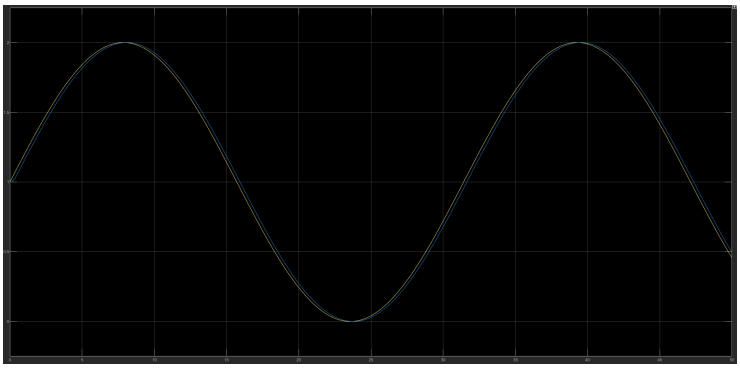

Figure 6: Sine wave response for the z coordinate

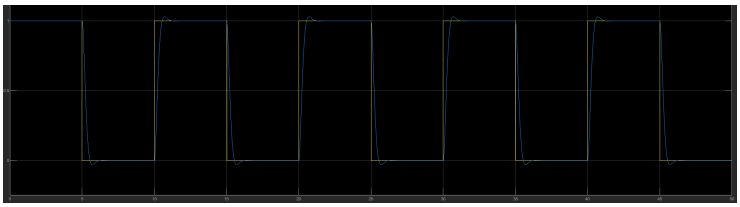

Figure 7: Sine wave response for the z coordinate

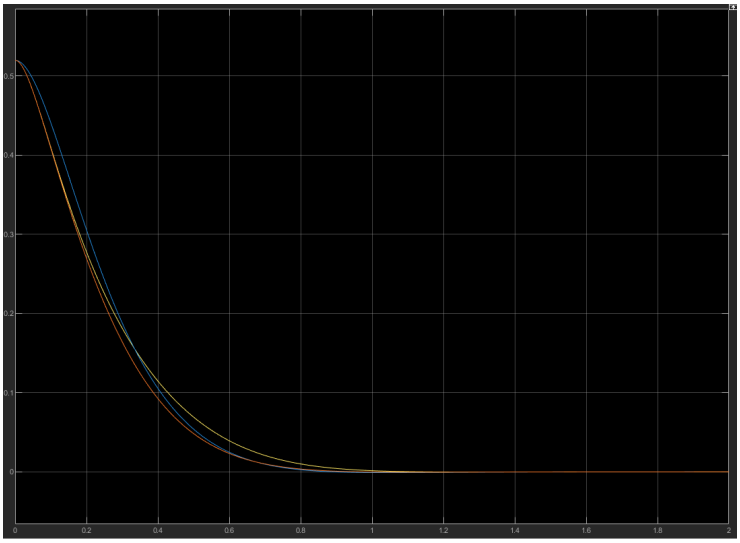

Figure 8: From 30^{o} to 0^{o} for roll pitch yaw angles

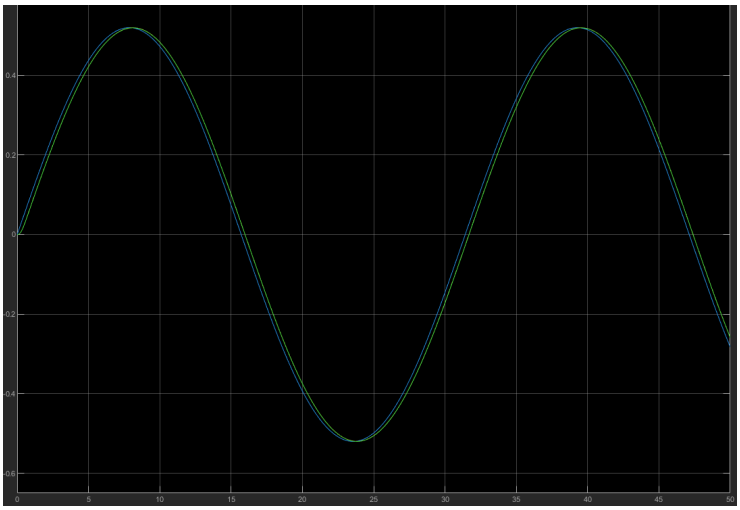

Figure 9: Sine wave response for roll pitch yaw angles

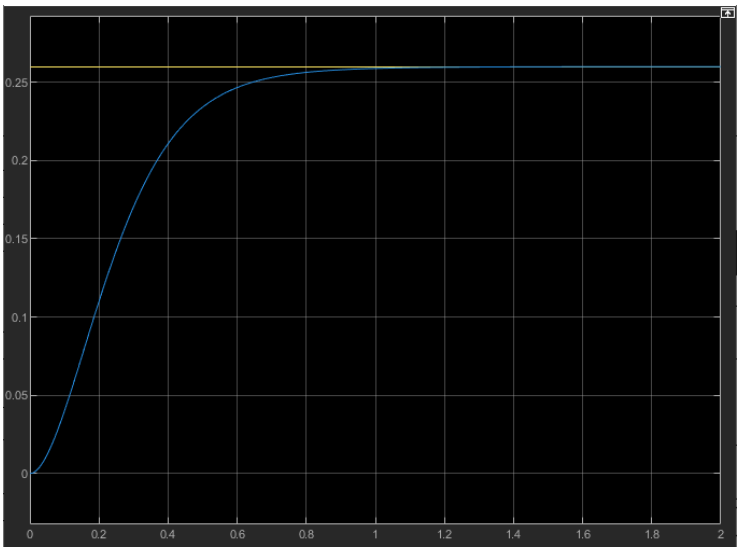

Figure 10: Step response from 0^{o} to 15^{o} for phi (roll)

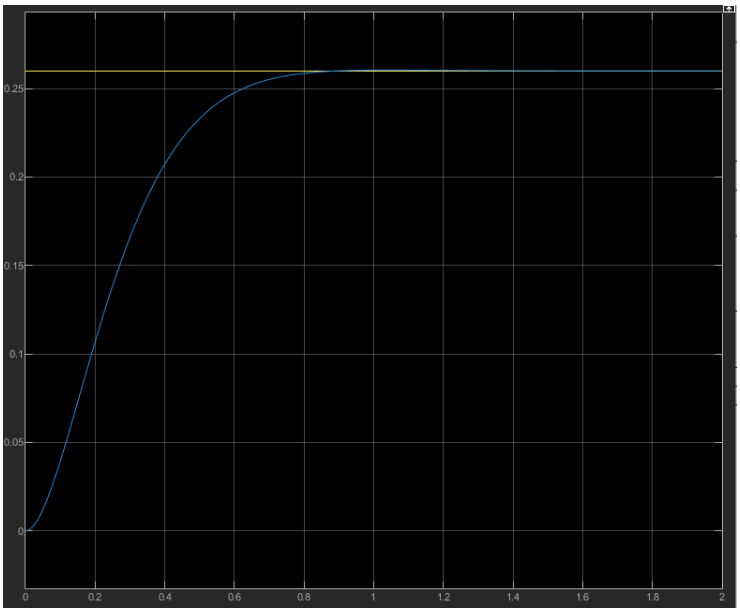

Figure 11: Step response from 0^{o} to 15^{o} for theta (pitch)

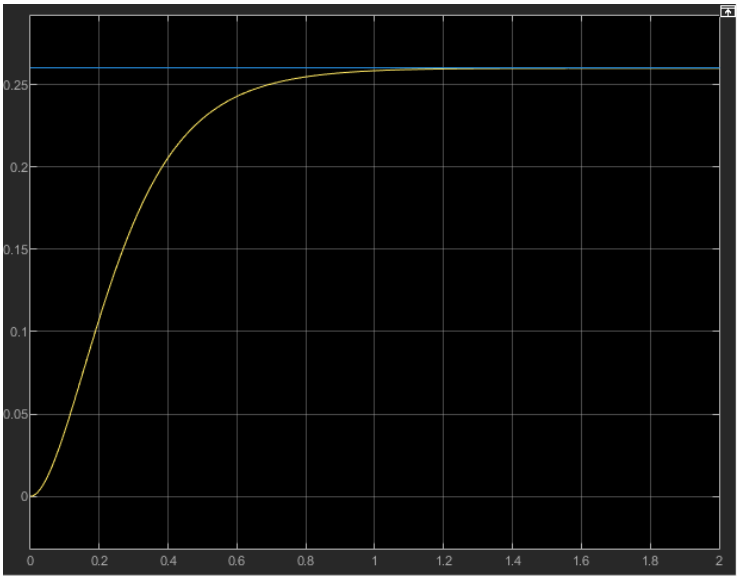

Figure 12: Step response from 0^{o} to 15^{o} for psi (yaw)

Coordinate	Overshoot (%)	Peak Time (s)	Settling Time (s)	Steady State Error
Z (Altitude)	1.49%	0.697	1.2	0
Roll (ϕ)	0%	0.85	0.85	0
Pitch (θ)	0%	0.9	0.9	0
Yaw (ψ)	0%	1.2	1.2	0

Table 5: Performance metrics of the PD controller for altitude and angular stabilization

The simulation results demonstrate the effectiveness of the PD controller in stabilizing the quadrotor's altitude and angular coordinates. The system achieves good performance across all axes, with minimal overshoot, fast peak times, and short settling times, ensuring a stable and responsive flight behavior.

For the z coordinate, the system exhibits an overshoot of only 1.5%, which is within acceptable limits for precise height control. The peak time is 0.697s, meaning the system reacts quickly to altitude changes, and it fully stabilizes within 1.2 seconds, with zero steady-state error. This confirms that the selected K_p and K_d values provide a well-balanced response, avoiding excessive oscillations while maintaining fast tracking.

In the angular coordinates (ϕ, θ, ψ) the controller achieves zero overshoot, which is ideal for smooth and controlled drone movements. The peak times vary slightly reflecting the different control characteristics of each axis. All angles stabilize within 1.2s, demonstrating an efficient response with no residual oscillations.

6 Conclusions

The goal of this project was to apply a basic yet effective PD controller to regulate a highly dynamic and complex system through numerical simulations in MATLAB/Simulink which i've accomplished. I validated the system's response under different input conditions, demonstrating the utility of using a proportional-derivative approach for quadrotor stabilization.

The results confirmed that the PD controller successfully stabilizes the system, ensuring fast response times, minimal overshoot, and zero steady-state error in angular stabilization. This performance shows that a well-tuned PD controller can effectively manage the drone's orientation and altitude without requiring more complex control strategies at this stage.

Although many advanced control methods exist such as nonlinear controllers, adaptive techniques and model predictive control, the purpose of this project was to establish a solid foundation for stabilization using a simple approach. The PD controller was chosen intentionally for its ease of implementation, low computational cost, and robustness in handling small disturbances. However, this work represents only the first step in a wider control strategy.

The next stage of this project would be focusing on trajectory control, allowing the quadrotor to autonomously follow predefined paths while maintaining stability. This will involve additional control layers and possibly more sophisticated algorithms to optimize tracking performance.

7 References

References

[1] Nicolás Monteagudo Duro, "Modelado y control de un cuadricóptero AR.DRONE," 2016

[2] Juan Gabriel Jaramillo Bucheli, Fernando Alonso Vaca De La Torre, "Implementación de un sistema de control robusto para seguimiento de trayectoria de tres cuadricópteros en formación," 2018

[3] Nicolás Parra Ballesteros, "Modelado y simulación de sistema de vuelo de un dron con detección de fallos," 2020