Attitude Control for Quadcopters: A PD-Based Approach for Stabilization

Akash Ajin, Akhilesh Menon, Paul Choi, Shiv Kanade
MAT292: Ordinary Differential Equations

Fall 2025

Abstract

This project investigates the attitude dynamics of a quadcopter, an inherently unstable system that requires continuous feedback control for stable flight. We derive a 12-state nonlinear rigid-body model based on the Newton–Euler equations of motion and formulate it as a coupled system of ordinary differential equations (ODEs). Using this model, we design a Proportional–Derivative (PD) control architecture for attitude (roll, pitch, yaw) and altitude. Numerical simulations using a Runge–Kutta ODE solver (`scipy.integrate.solve_ivp`) show that the tuned controllers stabilize the vehicle from rest, track step commands in roll and yaw with small overshoot and short settling times, and maintain altitude with negligible steady-state error. These results illustrate how ODE modeling and numerical integration directly enable the analysis and design of feedback controllers for real engineering systems.

1 Introduction

1.1 Motivation

A quadcopter is an unmanned aerial vehicle (UAV) whose flight is controlled by four motors. This UAV's navigational agility has made it popular in fields from photography to logistics. However, quadcopters are inherently unstable. Without a constant stream of adjustments from a control system, minor disturbances would cause them to tumble. The system's dynamics are also highly non-linear and coupled, making it a very fitting problem for an ODE course.

1.2 Project Goal

The primary objective of this project was to model the unstable flight dynamics of a quadcopter, of mass 500 grams and arm length 0.225 meters, as a system of coupled, nonlinear ordinary differential equations, and to use this model to design and tune a stabilizing feedback controller. We implement a PD architecture for attitude and altitude control, and evaluate their performance in numerical simulation. The project provides a clear, practical

demonstration of how the tools of an ODE course (modeling, linear algebra, and numerical solvers) are used to solve a fundamental problem in modern robotics. In this work we restrict attention to hover and small-angle attitude maneuvers, which allows us to focus on deriving and simulating the underlying ODEs without the added complexity of full 3D trajectory tracking. Hover is treated as an equilibrium point of the nonlinear system, and the goal of the feedback controller is to locally stabilize this equilibrium in the presence of small disturbances. The central takeaway of this project is that modeling the quadcopter using the nonlinear equations of motion highlights coupling effects and behaviors that are not captured by simplified or linearized models about hover. In particular, the nonlinear model provides clearer insight into how attitude and altitude dynamics interact during disturbance rejection and controller response. Controller performance is evaluated quantitatively in simulation using standard time-domain metrics such as rise time, settling time, and overshoot.

2 Mathematical Model and Theoretical Foundation

The quadcopter's motion is modeled as a rigid body in 3D space, using a fixed inertial frame (Earth, E) and a rotating body frame (Body, B) attached to the vehicle [1], [2].

2.1 State Vector

The system is described by a 12-element state vector $\mathbf{x} \in \mathbb{R}^{12}$:

$$\mathbf{x} = [\ \underbrace{x, y, z}_{\substack{\text{position} \\ \text{(Frame E)}}}\ ,\quad \underbrace{\phi, \theta, \psi}_{\substack{\text{attitude (Euler)} \\ \text{(Frame E)}}}\ ,\quad \underbrace{\dot{x}, \dot{y}, \dot{z}}_{\substack{\text{lin. velocity} \\ \text{(Frame E)}}}\ ,\quad \underbrace{p, q, r}_{\substack{\text{ang. velocity} \\ \text{(Frame B)}}}\ ]^T$$

This state is a hybrid, as linear motion is tracked in the inertial frame E while rotational motion is tracked in the body frame B. This requires transformation matrices to couple the dynamics.

2.2 Translational Dynamics

In the inertial frame, Newton's 2nd Law governs translational motion:

$$\mathbf{F}_{\text{net},E} = m\ddot{\mathbf{p}}_E = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

The net force is the sum of gravity $\mathbf{F}_{grav,E} = [0, 0, -mg]^T$ and the total thrust $\mathbf{F}_{thrust,B} = [0, 0, T]^T$. Thrust is generated in the body frame (acting along the quadcopter's z_B-axis) and must be rotated into the inertial frame:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m} \left(\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \right)$$

where $T = T_1 + T_2 + T_3 + T_4$, m is the mass(0.5kg), and R is the $Z - Y - X$ body-to-inertial rotation matrix. R is constructed as $R = R_z(\psi)R_y(\theta)R_x(\phi)$:

$$R = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

where $c\alpha = \cos(\alpha)$ and $s\alpha = \sin(\alpha)$. The first six ODEs are thus: $\dot{x} = \dot{x}$, $\dot{y} = \dot{y}$, $\dot{z} = \dot{z}$, and the three translational acceleration equations above $(\ddot{x}, \ddot{y}, \ddot{z})$.

2.3 Rotational Dynamics

Rotational motion is described by the Newton–Euler equations in the body frame [1], [3]:

$$\tau_B = I\dot{\omega}_B + \omega_B \times (I\omega_B)$$

where $\tau_B = [\tau_\phi, \tau_\theta, \tau_\psi]^T$ are the net torques, I is the 3×3 inertia tensor, and $\omega_B = [p, q, r]^T$. Assuming a diagonal inertia tensor $I = \text{diag}(I_{xx}, I_{yy}, I_{zz})$, the ODEs for the angular rates are:

$$\begin{aligned} \dot{p} &= (1/I_{xx})\,(\tau_\phi - (I_{zz} - I_{yy})qr) \\ \dot{q} &= (1/I_{yy})\,(\tau_\theta - (I_{xx} - I_{zz})pr) \\ \dot{r} &= (1/I_{zz})\,(\tau_\psi - (I_{yy} - I_{xx})pq) \end{aligned}$$

The Euler angle derivatives (in frame E) are related to the body rates (in frame B) by the transformation W:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = W(\phi, \theta) \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

These nine equations form the complete system of 12 first-order ODEs $(\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}, \ddot{x}, \ddot{y}, \ddot{z}, \dot{p}, \dot{q}, \dot{r})$, which can be written in the form $\dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{u})$.

3 Methodology

3.1 Overall Control Architecture

The controller is organized in two layers. An **inner attitude loop** uses three PD controllers to track desired Euler angles $(\phi_d, \theta_d, \psi_d)$ by commanding body-frame torques $(\tau_\phi, \tau_\theta, \tau_\psi)$. An **altitude loop** uses a PD controller to track the desired height z_d by commanding the total thrust T. Together, these four controllers produce the virtual control vector

$$\mathbf{u}_v = [T, \tau_\phi, \tau_\theta, \tau_\psi]^T,$$

which is then mapped to the individual motor thrusts via the inverse mixing matrix M^{-1} described below. In the simulations we chose simple step commands (for example, $z_d = 1.0\,\text{m}$ and attitude steps of $\phi_d = 10°$, $\psi_d = 30°$ over a finite time window), while holding $\theta_d = 0°$ to test cross-coupling.

3.2 Control Strategy: PID Controller

To stabilize the dynamics, we implemented four independent controllers following the standard PID control law [4], [5], which computes a corrective action

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

The proportional term $K_p e(t)$ responds to the current error, the integral term accumulates past error to eliminate steady-state drift, and the derivative term anticipates future error to damp oscillations. For all four axes (ϕ, θ, ψ, z) we ultimately set $K_i = 0$ and used a PD structure, relying on gravity feed-forward compensation for the altitude controller to eliminate steady-state error.

3.3 Control Allocation

The four PID outputs, known as virtual controls, $\mathbf{u}_v = [T, \tau_\phi, \tau_\theta, \tau_\psi]^T$, must be mapped to the four individual motor thrusts $\mathbf{T}_m = [T_1, T_2, T_3, T_4]^T$. Based on the quadcopter configuration and the final invertible mixing matrix used:

$$\begin{aligned}
T &= T_1 + T_2 + T_3 + T_4 \\
\tau_\phi &= L(-T_1 + T_3) \quad \text{(Roll torque)} \\
\tau_\theta &= L(-T_2 + T_4) \quad \text{(Pitch torque)} \\
\tau_\psi &= k_m(-T_1 + T_2 - T_3 + T_4) \quad \text{(Yaw torque)}
\end{aligned}$$

Where k_m is the yaw torque coefficient (0.01 Nm/N), and L is the arm length (0.225 m). This required the inverse mapping $\mathbf{T}_m = M^{-1}\mathbf{u}_v$, where M^{-1} is:

$$M^{-1} = \begin{bmatrix} 1/4 & -1/(2L) & 0 & -1/(4k_m) \\ 1/4 & 0 & -1/(2L) & 1/(4k_m) \\ 1/4 & 1/(2L) & 0 & -1/(4k_m) \\ 1/4 & 0 & 1/(2L) & 1/(4k_m) \end{bmatrix}$$

3.4 Numerical Simulation

The simulation was implemented in Python using the `scipy.integrate.solve_ivp` function [6], which employs a Runge–Kutta 45 method to solve the 12 ODEs [7]. We used a main loop that calls `solve_ivp` at each discrete control step with sampling period $\Delta t = 0.01\,\text{s}$. At each step k, the controller computes the motor thrusts from the current state $\mathbf{x}(t_k)$, and `solve_ivp` integrates the ODEs from t_k to $t_{k+1} = t_k + \Delta t$ using these thrusts as constant inputs. This mimics a digital flight controller running at 100 Hz interacting with continuous-time vehicle dynamics. The initial condition is a vehicle at rest near the origin, which highlights the need for active stabilization.

4 Results

4.1 PID Tuning

The goal of tuning was to achieve a near *critically damped* response for all four axes: fast tracking with minimal overshoot or oscillation.

We initially attempted the Ziegler–Nichols Ultimate Cycle Method [8] to establish a theoretical baseline. However, due to the highly non-linear nature and cross-coupling present in the full ODE model, this approach yielded aggressive gains that resulted in significant under-damped oscillations and, critically, the integral term K_i consistently triggered integral windup, leading to instability in the attitude axes.

Therefore, the final stable design for the controllers uses a Proportional–Derivative (PD) structure ($K_i \approx 0$). The final tuned gains are presented in Table 1. For these gains, the altitude response exhibits a rise time of roughly $t_r \approx 2.0\,\text{s}$ and settles within a small band around the target by about $t_s \approx 3.0\,\text{s}$ with negligible overshoot. The roll and yaw responses have rise times on the order of 1.0 s and settle without sustained oscillations, which is consistent with a nearly critically damped design.

Table 1: Final Controller Gains for Critically Damped Response

Controller	K_p	K_i	K_d
Altitude (z)	3.00	0.00	2.280
Roll (ϕ)	0.08	0.00	0.043
Pitch (θ)	0.08	0.00	0.043
Yaw (ψ)	1.00	0.00	0.057

4.2 Hover Stabilization

The simulation commanded the quadcopter to maintain a hover at $z = 1.0\text{m}$. The Altitude controller performance is shown in Figure 1.

4.3 Attitude Disturbance Rejection

The controller's ability to track commands and stabilize against disturbances was tested by commanding a 10-degree Roll and a 30-degree Yaw from $t = 2\text{s}$ to $t = 8\text{s}$. The attitude controller performance is shown in Figure 2.

5 Discussion

The primary objective of stabilizing the quadcopter using an ODE-based model and feedback control was successfully met.

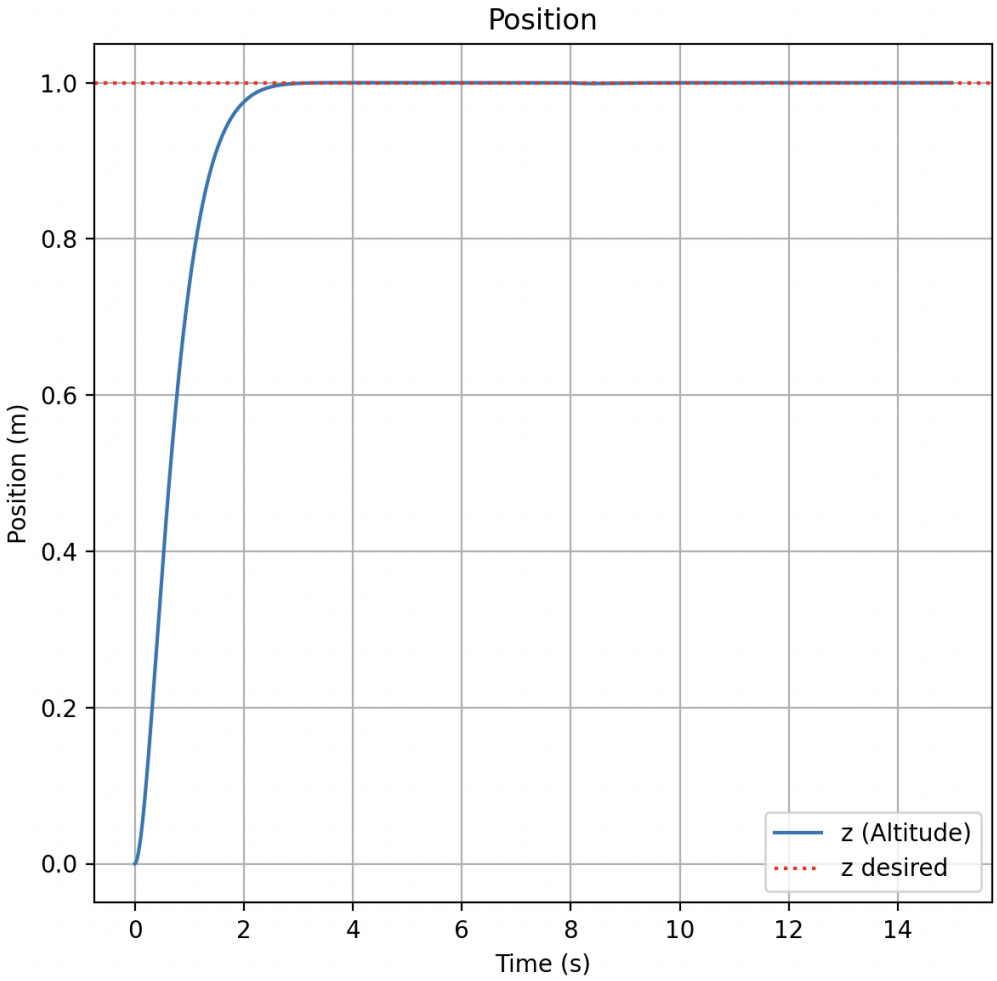

Figure 1: Altitude (z-axis) response to a 1.0m step command. The PD controller brings the quadcopter to the desired altitude in approximately 3 seconds with zero overshoot.

5.1 Interpretation of Results

As shown in Figure 1, the altitude controller (PD) achieves near-perfect tracking of the $z = 1.0\,\mathrm{m}$ setpoint with no visible overshoot and a short settling time. The gravity-compensation feed-forward successfully eliminates the steady-state error that would otherwise be present due to gravity and small modeling offsets.

Figure 2 demonstrates the performance of the attitude controllers (PD). The roll and yaw axes track their respective command steps (10° and 30°) with fast rise times and settle without the destructive oscillations observed during the initial tuning phase. When pitch is held at 0°, the response remains close to zero throughout the maneuver, confirming the robustness of the mixing matrix $\mathbf{M}^{-1}$ and the effectiveness of the decoupled control design. Overall, the time-domain behavior observed in simulation is consistent with the design goal of a nearly critically damped response.

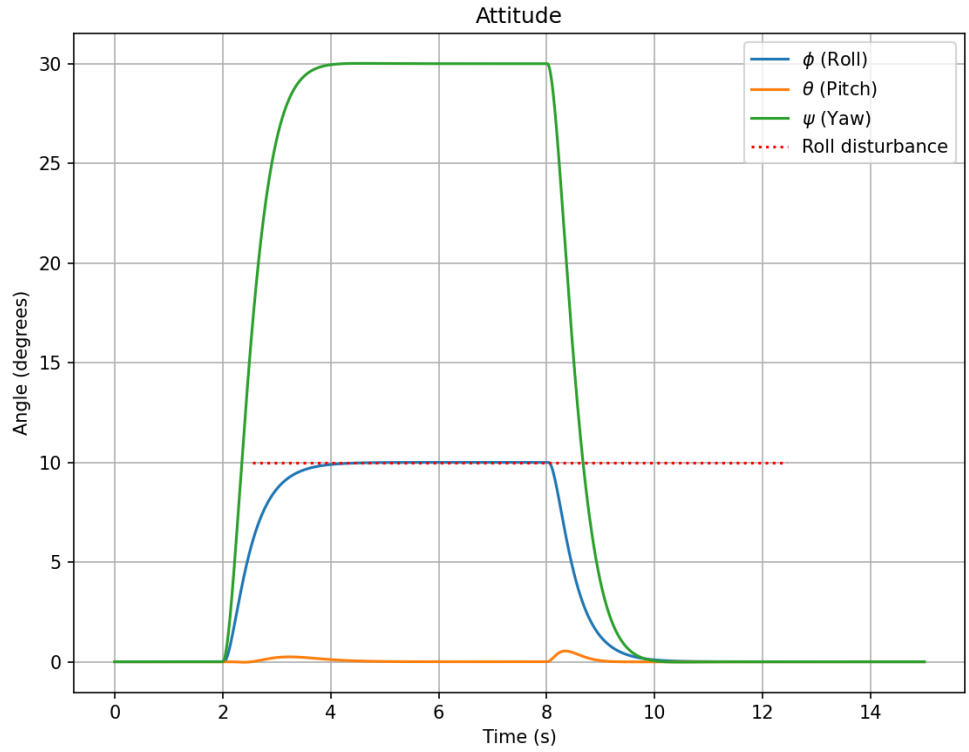

Figure 2: Attitude (roll, pitch, yaw) response to simultaneous step commands (Roll=10°, Yaw=30°). The PD controller achieves fast rise time with minimal overshoot, and the Pitch axis remains stable at 0°, demonstrating excellent **cross-coupling rejection.**

5.2 Limitations and Future Work

The model currently has several simplifying assumptions:

- **No aerodynamic drag:** We assume drag on the body is negligible. This is accurate for low speeds but, at higher speeds, drag forces would need to be added to the translational ODEs in $\mathbf{F}_{\text{net},E}$, potentially changing the optimal gains.
- **Perfect control inputs:** We assume motors react instantly and linearly, ignoring actuator saturation and time delays. Including motor dynamics would require additional states and ODEs and would make the closed-loop system higher dimensional.
- **Attitude vs. position control:** The current controller regulates altitude and attitude but not horizontal position. A natural extension is a cascaded architecture where an outer-loop PID generates desired roll and pitch angles from (x, y) position errors, enabling full 3D trajectory tracking.

These limitations suggest clear directions for future work: extend the ODE model to include additional physics and actuators, and design corresponding controllers that preserve stability and performance.

6 Conclusion

We derived a 12-DOF non-linear ODE system governing quadcopter dynamics and implemented a stabilizing PD control architecture solved numerically with Runge–Kutta methods (`scipy.integrate.solve_ivp`). The simulations show that the inherently unstable open-loop system can be stabilized, with step responses that exhibit small overshoot and short settling times in both altitude and attitude. This project demonstrates how the core ideas from an ODE course, formulating physical laws as differential equations and solving them numerically, translate directly into the analysis and design of practical control systems [5], [7].

References

[1] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor robots," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012. DOI: 10.1109/MRA.2012.2206474

[2] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *AIAA Guidance, Navigation and Control Conference*, AIAA, 2007.

[3] D. T. Greenwood, *Classical Dynamics*. Dover Publications, 2003.

[4] K. J. Åstr"om and T. H"agglund, *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, 1995.

[5] K. Ogata, *Modern Control Engineering*, 5th ed. Prentice Hall, 2010.

[6] P. Virtanen, R. Gommers, T. E. Oliphant, et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2

[7] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.

[8] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Transactions of the ASME*, vol. 64, no. 8, pp. 759–768, 1942.