IE415

Control of Autonomous Systems

Project Report

Control System Analysis for Quadrotor Models: PID and LQR Approaches

202201207

Swayam Hingu

Assigned By - Proff. Sujay Kadam

Contents

1 INTRODUCTION

Quadcopters, a type of unmanned aerial vehicle powered by four rotors, have gained significant attention due to their growing applications in areas such as video surveillance, search and rescue missions, and remote sensing. The ability to fly in unknown and dynamic environments with precision is crucial, and this necessitates the development of effective control systems. For quadcopters, maintaining stable flight and maneuverability is achieved through robust control algorithms.

1.1 Objective

This project focuses on a simulation-based comparison between two control systems for a linearized quadrotor model: the Proportional Integral Derivative (PID) controller, a classical control method, and the Linear Quadratic Regulator (LQR), an optimal control method. The main objective is to derive a linear mathematical model for the quadrotor using Newtonian mechanics and Euler's laws, providing a simplified representation of its dynamics. This model serves as the foundation for implementing both the PID and LQR controllers. Simulations of both control strategies are conducted in Simulink under identical initial conditions, allowing a direct performance comparison.

1.2 Motivation for study

The motivation behind this study is driven by the desire to enhance quadcopter control in more unpredictable and complex environments. While PID controllers are widely used due to their simplicity and effectiveness, they require linearization for each test scenario, which can be limiting. In contrast, the LQR controller does not require linearization, making it potentially more versatile and robust. This research aims to determine whether the optimal control provided by LQR offers significant advantages in terms of performance and stability under varying conditions.

1.3 Scope and Future Applications

The scope of this project extends to the feasibility of implementing LQR controllers for quadcopters, with the potential for application in other unmanned aerial vehicles such as fixed-wing aircraft and vertical take-off and landing (VTOL) vehicles. The results of this work could have broad implications for enhancing autonomous flight control systems in various unmanned vehicles.

2 QUADROTOR MODEL

The quadrotor is an underactuated nonlinear system with four inputs (rotors) and six outputs (three translational and three rotational motions). It is controlled by varying the speeds of the four rotors, which produce thrust and torques for controlling the quadrotor's movement. The system's orientation is described using Euler angles (roll ϕ, pitch θ, yaw ψ) and its position and orientation are expressed in both fixed (inertial) and mobile (body-centered) reference frames.

2.1 System Dynamics

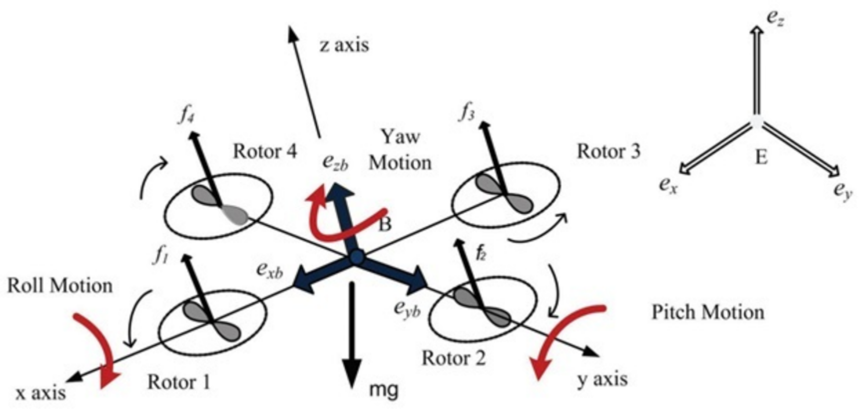


Figure 1: Quadrotor: a dynamic system

The dynamics of the quadrotor are governed by the following equations:
1. Translational motion:

$$\ddot{x} = (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\,\frac{F}{m}$$

$$\ddot{y} = (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\,\frac{F}{m}$$

$$\ddot{z} = -g + (\cos\phi\cos\theta)\,\frac{F}{m}$$

2. Rotational motion:

$$\dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}}qr - \frac{J_r}{I_{xx}}q\omega + \frac{u_2}{I_{xx}}$$

$$\dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}}pr - \frac{J_r}{I_{yy}}p\omega + \frac{u_3}{I_{yy}}$$

$$\dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}}pq + \frac{u_4}{I_{zz}}$$

Where:

$$
\begin{aligned}
x, y, z &: \text{Translational positions}\\
p, q, r &: \text{Angular velocities (roll, pitch, yaw)}\\
F &: \text{Total thrust}\\
m &: \text{Mass of the quadrotor}\\
g &: \text{Gravitational acceleration}\\
I_{xx}, I_{yy}, I_{zz} &: \text{Moments of inertia}\\
J_r &: \text{Rotor inertia}\\
u_2, u_3, u_4 &: \text{Control inputs for torques}
\end{aligned}
$$

2.2 Inputs

1. **Thrust:**

$$u_1 = K_f \cdot (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)$$

2. **Roll torque:**

$$u_2 = K_f \cdot (\omega_4^2 - \omega_2^2)$$

3. **Pitch torque:**

$$u_3 = K_f \cdot (\omega_1^2 - \omega_3^2)$$

4. **Yaw torque:**

$$u_4 = K_m \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)$$

Where:

$$
\begin{aligned}
K_f &: \text{Thrust factor}\\
K_m &: \text{Drag factor}\\
\omega_i &: \text{Angular velocity of the } i\text{th rotor}
\end{aligned}
$$

This model describes the fundamental dynamics of the quadrotor system, which is essential for control strategies and simulations.

3 STATE-SPACE MODELING

The state-space representation of the quadrotor system consists of 12 state variables, four input variables, and four output variables. The state variables represent the quadrotor's position, orientation, and their respective velocities. The input variables correspond to the thrust, roll, pitch, and yaw motions, while the output variables are the required state variables for stability analysis, namely the vertical displacement (z), roll (ϕ), pitch (θ), and yaw (ψ) angles.
The state-space model is represented by the following equations:

$$\dot{X} = AX + BU$$
$$Y = CX + DU$$

Where:

$$X^T = [x\, y\, z\, \phi\, \theta\, \psi\, \dot{x}\, \dot{y}\, \dot{z}\, p\, q\, r]$$
$$U^T = [u_1\, u_2\, u_3\, u_4]$$
$$Y^T = [z\, \phi\, \theta\, \psi]$$

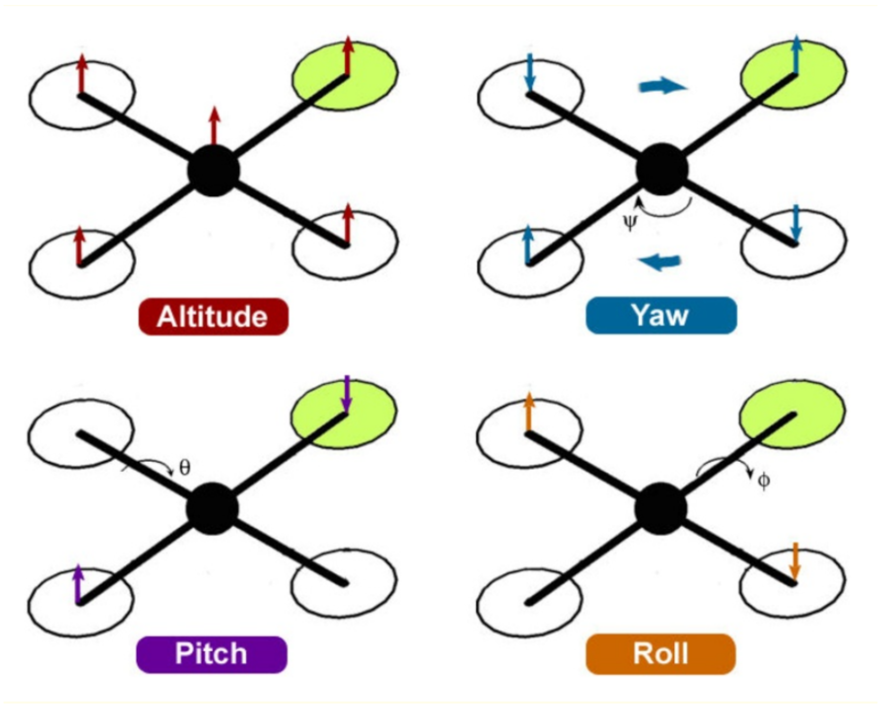


Figure 2: Output of Quadrotor dynamics system

Where:

- A is the 12×12 state matrix.
- B is the 12×4 input matrix.
- C is the 4×12 output matrix, where each column corresponds to the states needed for output.
- D is the 4×4 null matrix.

Once the state-space equations are obtained, a controller is designed for the quadrotor system. The next section discusses the action of a controller on the system.

4 PROPORTIONAL INTEGRAL DERIVATIVE (PID) CONTROL

4.1 Overview

The PID controller is a standard control model aimed at minimizing the error between the desired setpoint and the actual output. It uses three terms: proportional, integral, and derivative, each serving a specific purpose:

- **Proportional (P)**: Corrects error by scaling it with a proportional constant K_p. Larger K_p values lead to faster correction but can cause overshoot.
- **Integral (I)**: Sums past errors to eliminate steady-state errors, controlled by K_i. Too large K_i can cause instability.
- **Derivative (D)**: Predicts future error behavior by considering its rate of change, scaled by K_d, and helps dampen oscillations.

Proper tuning of K_p, K_i, and K_d ensures quick and stable system responses.

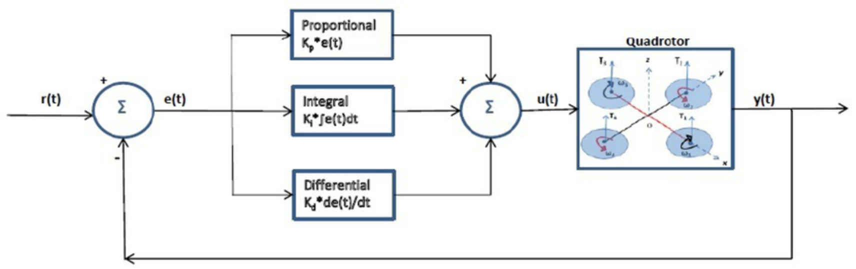


Figure 3: PID controller applied to the quadrotor

4.2 Equations

The control output is calculated using the following PID equation:

$$U(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{18}$$

Where:

- $U(t)$ is the control signal.
- $e(t)$ is the error, defined as the difference between the setpoint and the actual value.

The constants K_d and K_i are defined as:

$$K_d = K_p T_d \quad \text{and} \quad K_i = \frac{K_p}{T_i} \tag{19}$$

Where:

- T_d and T_i are the time constants for the derivative and integral actions, respectively.

The error $e(t)$ is defined as:

$$e(t) = \text{setpoint value} - \text{actual value} \tag{20}$$

4.3 Controller Design

In a quadrotor system, the state parameters (position and orientation) must be controlled to maintain stability. Small deviations from setpoint values can lead to significant motion variations. While the user has control, external disturbances may destabilize the system, necessitating an internal flight controller.
The four inputs—thrust, roll, pitch, and yaw—control six state parameters:

- **Thrust** controls the z-axis (vertical displacement).
- **Roll** controls the x-axis and the pitch angle θ.
- **Pitch** controls the y-axis and the roll angle ϕ.
- **Yaw** controls the yaw angle ψ.

Thus, six PID controllers are used:

- 1 for thrust,
- 2 for roll and pitch (cascaded loops for each),
- 1 for yaw.

The cascaded PID system has an inner loop for fast response and an outer loop for stability. Proper tuning ensures that the system maintains stability and minimizes errors even during disturbances.

4.4 Tuning PID Parameters

PID parameters K_p, K_i, and K_d are tuned to optimize system performance. These values are usually determined through trial-and-error or optimization methods to avoid instability and ensure the quadrotor responds efficiently.

5 LINEAR QUADRATIC REGULATOR (LQR)

5.1 Overview

The LQR controller, unlike the classic PID controller, employs an optimization approach to minimize a cost function that considers both the state and control inputs. The cost function is influenced by the Q and R matrices, which control the importance given to the states and inputs, respectively. By tuning these matrices, the controller can be adjusted for faster response or more efficient power consumption.

5.2 Cost Function and Optimization

The optimization problem involves minimizing the following cost function:

$$J = \int_0^\infty (X^T Q X + U^T R U)\, dt \tag{21}$$

Where:

- X represents the state vector,
- U represents the control input,
- Q and R are weight matrices for states and inputs.

To solve the problem, the Algebraic Riccati Equation (ARE) is used:

$$A^T S + SA - SBR^{-1}B^T S + Q = 0 \tag{22}$$

This equation gives the matrix S, which is used to compute the state feedback gain matrix K:

$$K = R^{-1}B^T S \tag{23}$$

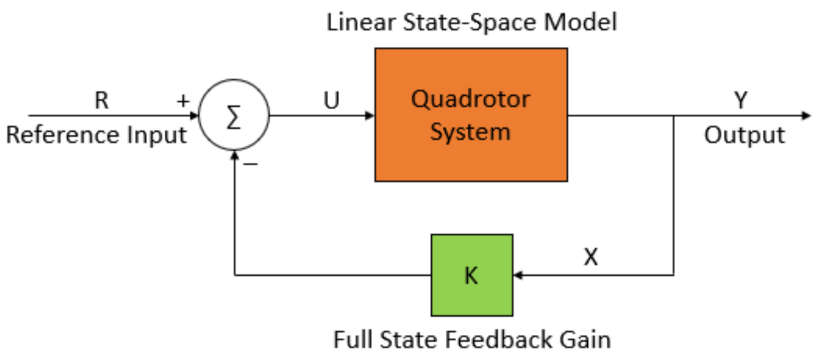


Figure 4: The closed loop quadrotor system with LQR controller

5.3 Control Input

The control input U is derived from the state feedback gain matrix K and the state vector X:

$$U = -KX \tag{24}$$

This equation signifies that the control input is a linear function of the state vector, where the matrix K determines the influence of each state on the control input. By adjusting K, the system can be optimized to achieve the desired setpoint while balancing performance and power efficiency.

5.4 Tuning and Results

The LQR controller's performance depends heavily on the tuning of the Q and R matrices. For the quadrotor system, the values of these matrices must be selected to balance the response time and power consumption.

6 LUENBERGER OBSERVER FOR QUADROTOR

6.1 Overview

A Luenberger Observer is a state estimation technique used to estimate unmeasured state variables of a system based on its mathematical model and measured outputs. For a UAV (Quadrotor), the observer is particularly useful when not all states (such as velocities or angles) can be directly measured due to sensor limitations or noise. The observer reconstructs the full state vector using the system's input, output, and its dynamics. **We need an observer when the full state of the system is not directly measurable, allowing us to estimate unmeasured states based on the available outputs and system model.**

6.2 State-Space Model

The UAV dynamics can be expressed in state-space form:

$$\dot{X}(t) = AX(t) + BU(t)$$

$$Y(t) = CX(t) + DU(t)$$

Where:

- $X(t)$ is the state vector (e.g., position, velocities, angles).
- $U(t)$ is the input vector (e.g., control inputs: thrust, roll, pitch, yaw).
- $Y(t)$ is the output vector (e.g., measured states such as position or angles).
- A, B, C, D are system matrices.

6.3 Observer Design

The Luenberger Observer introduces an estimated state vector $\hat{X}(t)$ and updates it using the system model and measured outputs. The observer equation is:

$$\dot{\hat{X}}(t) = A\hat{X}(t) + BU(t) + L\left(Y(t) - \hat{Y}(t)\right)$$

Where:

- $\hat{X}(t)$ is the estimated state vector.
- L is the observer gain matrix.
- $Y(t) - \hat{Y}(t)$ is the error between the measured and estimated outputs.

The observer gain matrix L is designed to place the eigenvalues of the observer error dynamics matrix $A - LC$ in a stable region, ensuring fast and accurate state estimation.

6.4 Error Dynamics

The estimation error $E(t)$ is defined as:

$$E(t) = X(t) - \hat{X}(t)$$

Differentiating $E(t)$:

$$\dot{E}(t) = (A - LC)E(t)$$

The matrix $A - LC$ determines the stability and convergence of the observer. Proper placement of eigenvalues ensures that the estimation error approaches zero.

6.5 Design Steps

1. **Linearize the Quadrotor Dynamics:** If the Quadrotor system is nonlinear, linearize it around a specific operating point.
2. **Choose Observer Gain L:** Use techniques like pole placement or optimal control methods to compute L.
3. **Simulate and Validate:** Verify the observer's performance by simulating the Quadrotor dynamics and comparing estimated states $\hat{X}(t)$ with actual states $X(t)$.

6.6 Applications for Quadrotor

- **Fault Detection:** Identify discrepancies between measured and estimated states.
- **State Reconstruction:** Estimate unmeasured variables like angular velocity or acceleration.
- **Improved Control:** Use state estimates for advanced control strategies like LQR or MPC.

7 RESULTS AND DISCUSSION

The initial conditions for the system are set as:

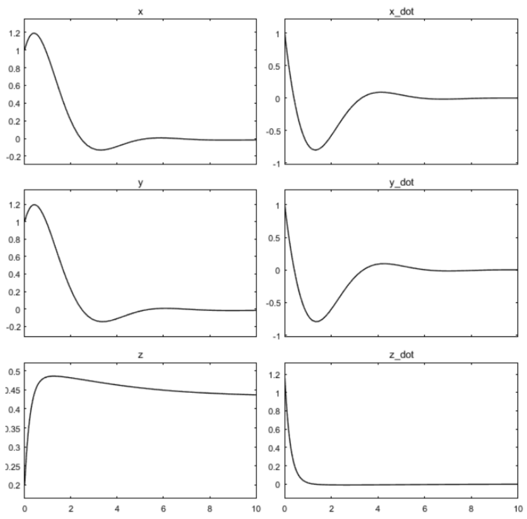
$$[1\,1\,0.2\,1\,1\,0\,1\,1\,1\,1\,1\,1]$$

These values represent the initial deviations of the state parameters from their desired setpoints. The input to the system is given as a pure thrust force. For this scenario, all state parameters, except for z, should converge to zero, as no additional rotor action is desired. The initial conditions indicate that the rotor is deviated from its setpoint, prompting the controller to act and bring the system back to equilibrium.
The performance of both the PID and LQR controllers is analyzed based on their responses. Both controllers achieve a similar settling time of approximately 7 seconds for all state parameters. However, the overshoot behavior differs between the two controllers:

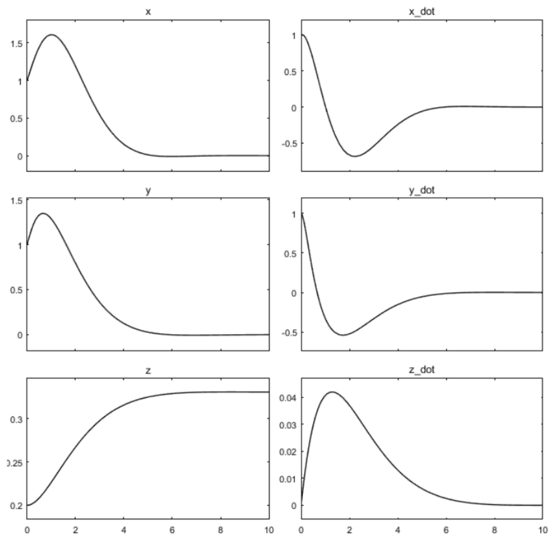
- The first overshoot of the PID controller is observed to be 1.2 units, which is lower than the LQR controller's first overshoot of 1.6 units for the x and y parameters.
- The PID controller exhibits a secondary overshoot in several state parameters, while the LQR controller shows a single overshoot peak.

The presence of multiple overshoot peaks in the PID controller indicates reduced stability compared to the LQR controller. The LQR controller provides a more stable and robust response, ensuring the system quickly converges to the desired state with minimal oscillations. This makes the LQR controller preferable for applications requiring higher stability and robustness.

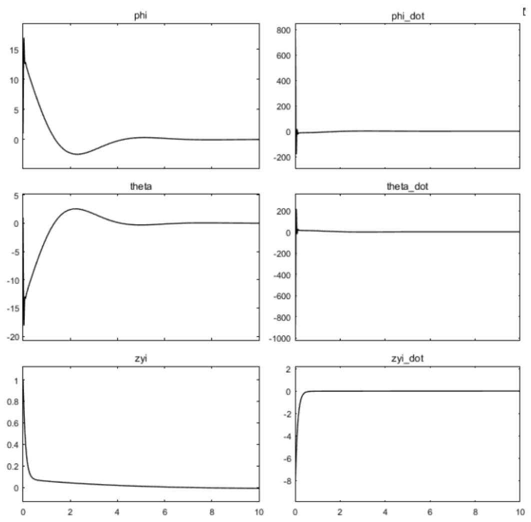
The results are summarized with the following visual representations:



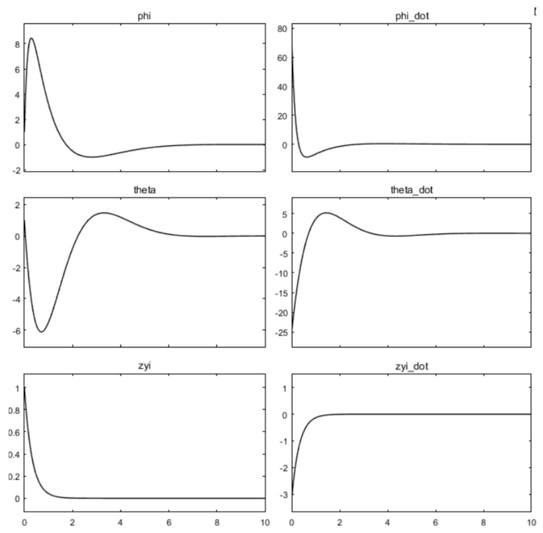
PID Controller Response: Translational States



LQR Controller Response: Translational States



PID Controller Response: Rotational States



LQR Controller Response: Rotational States

8 CONCLUSION

The two controllers, PID and LQR, both show similar performance in terms of settling time, meaning they take approximately the same amount of time to reach the desired state. However, the LQR controller has distinct advantages over the classical PID controller:

1. **Overshoot Behavior:** The PID controller exhibits two overshoot peaks in most cases, indicating a more aggressive response, which can result in instability or unwanted fluctuations. In contrast, the LQR controller shows a more stable behavior with only one overshoot peak, indicating better stability and smoother control.
2. **Tuning Complexity:** The PID controller requires tuning of six parameters for each of the six states, making it a tedious and time-consuming process. On the other hand, the LQR controller only requires tuning of two matrices (Q and R), which simplifies the design and tuning process.
3. **Computational Complexity:** The PID controller involves six feedback loops, making it computationally expensive and complex. In contrast, the LQR controller uses a single control loop, reducing the computational burden.

Thus, the LQR controller is better suited for quadrotor control systems, offering better stability, reduced complexity, and lower computation time compared to the PID controller. This makes LQR a more efficient and practical choice for real-time control applications in quadrotors.

9 SIMULATION RESULTS OF QUADROTOR SYSTEM

Source Code: https://github.com/Swayam-Hingu/IE415-Quadrotor_LQR_Control

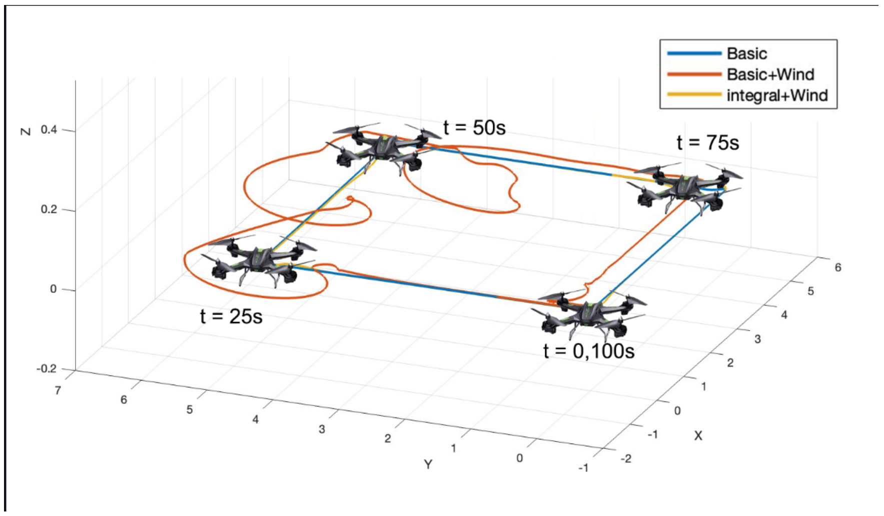


Figure 7: Trajectories of Quadrotor System with Basic Control, Basic Control with Wind, and Integral Control with Wind

This image displays the simulation results of a multi-quadrotor system under different control strategies. The drones are navigating in a 3D space, and their trajectories are compared for three scenarios: basic control, basic control with wind disturbances, and integral control with wind disturbances.

9.1 Control Strategies and Their Effects

- **Blue Path (Basic Control):** This represents the drone's trajectory when using a basic control system that does not account for wind disturbances. The drones follow a relatively straightforward path with small deviations.
- **Red Path (Basic + Wind):** Here, wind disturbances are introduced, and the drones' paths are altered. The drones show more significant deviations from the desired trajectory as they are affected by the wind. This highlights the limitation of basic control when dealing with external forces like wind.
- **Yellow Path (Integral + Wind):** In this case, an integral controller is used to correct the drone's trajectory in the presence of wind. The integral controller helps reduce steady-state errors caused by wind disturbances. The drones using this control strategy follow smoother and more accurate paths compared to the basic control with wind.

The simulation includes time markers at different moments ($t = 0, 25, 50, 75$) to show the drones' positions at specific intervals. These markers help illustrate how each control strategy performs over time.

- At $t = 0, 100\,\text{s}$, the drones start from their initial positions.
- At $t = 25\,\text{s}$, the drones begin to show the effects of the control strategies.

- At $t = 50\,\text{s}$ and $t = 75\,\text{s}$, the deviations due to wind are more pronounced in the basic control scenario (red path) but are corrected by the integral controller (yellow path).

9.2 Key Insights

1. **Wind Disturbances:** Wind causes noticeable deviations in the drones' paths, especially for basic control (red path).
2. **Control Strategy Performance:** The basic control struggles with wind disturbances, as seen in the red path, while the integral control adjusts for the errors more effectively, leading to smoother trajectories (yellow path).
3. **Integral Control Advantage:** The integral controller helps eliminate steady-state errors and offers a more robust solution in environments with disturbances like wind.

9.3 Conclusion

Overall, the simulation demonstrates the benefits of using an integral controller to improve drone performance in the presence of wind disturbances, making it more effective than basic control systems.

10 REFERENCES

- A Comparative Study Between a Classical and Optimal Controller for a Quadrotor click here
- Koopman-LQR Controller for Quad-rotor UAVs from Data click here
- Trajectory Control Simulation of Quadrotor Reference click here
- Robust control for a quadrotor UAV based on linear quadratic regulator click here