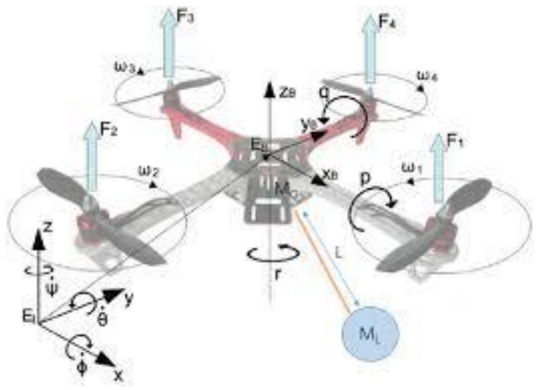
MODELLING, CONTROL AND STATE ESTIMATION OF A QUADROTOR



By: Dieudonne Fonyuy YUFONYUY

Introduction

Autonomous aerial vehicles, particularly quadrotors, have gained widespread attention due to their high manoeuvrability, compact design, and versatility in applications ranging from surveillance and inspection to mapping and delivery. However, their inherently unstable dynamics, nonlinear behaviour, and susceptibility to environmental disturbances pose significant challenges for robust control and accurate state estimation.

This project presents a complete workflow for the **modelling**, **control**, and **state estimation** of a quadrotor unmanned aerial vehicle (UAV) using **MATLAB** and **Simulink**. It begins with a detailed derivation of the **6-degree-of-freedom (6-DOF) nonlinear dynamics** using Newton-Euler equations, followed by a linearisation around a hover equilibrium point. This sets the foundation for designing **Linear Quadratic Regulator (LQR)** controllers to stabilize the system and track trajectories.

To overcome limitations of steady-state errors in LQR, the model is extended with **integral action**, improving accuracy during trajectory tracking tasks. A particular focus is given to robustness against **external disturbances** such as **wind gusts and variations in payload or mass properties.**

Given that not all states are directly measurable in real-world implementations (e.g., linear velocities or orientation angles), the project implements a **Kalman Filter** for the linearised system model and an **Extended Kalman Filter (EKF)** to reconstruct the full state vector from noisy measurements of position and angular rates. This EKF-based observer is integrated with the nonlinear model in Simulink, enabling closed-loop simulation of estimation and control under realistic sensor limitations and process uncertainties.

A wide range of simulation scenarios are presented, including:

- Stabilisation from initial perturbations
- Tracking of **3D circular, spiral, and rectangular trajectories, Linear**
- Robustness evaluation under **sensor noise** and **parameter uncertainty, and strong wind disturbances.**

This work bridges theoretical modelling with practical simulation and control, providing a modular and extensible testbed for researchers and engineers interested in UAV control, estimation, and autonomous navigation.

1. Quadcopter

The aerial platform utilized in this project is a quadcopter UAV. To effectively model its behaviour and design an LQR controller, it is necessary to understand its dynamic properties, derive the governing equations of motion, and perform system linearisation—an essential step for applying linear control techniques such as LQR.

1.1 Dynamics

A quadcopter, or quadrotor, is a type of multirotor UAV equipped with four rotors. These rotors are typically arranged in either a plus (+) or cross (×) configuration. This project adopts the plus configuration, where the pitch and roll axes align with the main body frame of the UAV, making the system easier to model and more intuitive for analysis.

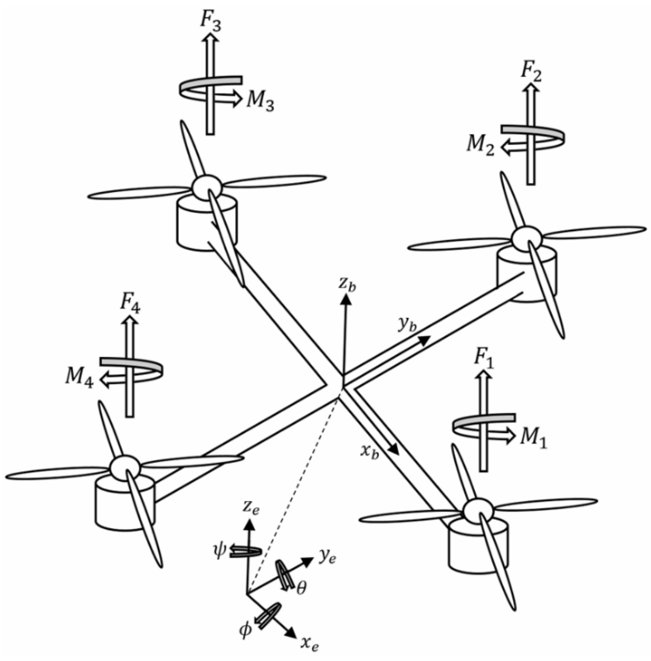


Figure 1. Axes, torques and forces of a quadcopter.

As depicted in Figure 1, the rotors are configured such that one pair spin clockwise (CW) and the opposing diagonal pair spins counterclockwise (CCW). This arrangement balances the reactive torques produced by the propellers, allowing the vehicle to maintain a steady yaw without undesired rotation.

The quadcopter exhibits inherently nonlinear dynamics, characterized by six degrees of freedom: three translational motions (along x, y, and z) and three rotational motions (roll, pitch, and yaw). These dynamics govern the complex motion of the vehicle in 3D space and form the foundation for control and estimation design.

The control of the quadcopter is achieved by varying the rotational speeds (RPM) of its four rotors. These changes directly influence the generated thrust and torque, enabling the vehicle to rotate and translate in different directions. Since the quadcopter has six degrees of freedom (6DOF) but only four independent control inputs, it is classified as an underactuated system. As illustrated in Figure 2.4, each rotor produces both a thrust force (F_i) and a torque moment (M_i), which together govern the vehicle's motion.

In the Earth-fixed reference frame, rotational motion around the $\mathbf{z}$-axis is known as yaw (ψ), rotation around the $\mathbf{x}$-axis is referred to as roll (ϕ), and rotation around the $\mathbf{y}$-axis is termed pitch (θ). These angular motions are defined relative to the center of the quadcopter. To produce pitch and roll, the thrust center must be adjusted, while yaw is controlled by manipulating the net torque.

These movements are achieved by adjusting rotor thrusts as follows:

- Vertical movement (ascend/descend): Increase or decrease all rotor thrusts (F_1, F_2, F_3, F_4) equally.
- Roll (rotation about x-axis): Keep F_1 and F_3 constant, while increasing or decreasing F_2 and doing the opposite to F_4.
- Pitch (rotation about y-axis): Keep F_2 and F_4 constant, while adjusting F_1 and F_3 in opposite directions.
- Yaw (rotation about $\mathbf{z}$-axis): Increase the torque from $\mathbf{M}_1$ and $\mathbf{M_3}$ while decreasing $\mathbf{M}_2$ and $\mathbf{M}_4$, or vice versa.

Each rotor creates two key effects: thrust and torque, which are modeled by the equations:

$$\begin{aligned} |F_i| &= k_F\omega_i^2 \\ |M_i| &= k_M\omega_i^2 \end{aligned} \qquad (1.1)$$

Where:

- ω_i is the angular velocity (RPM) of rotor i,
- k_F and k_M are thrust and torque coefficients.

The rotation speed ω of the propeller determines the thrust it generates, with the constant k_F accounting for environmental factors like air density and back-EMF. Propellers work by creating a pressure difference as they spin, which results in thrust. Similarly, torque (or moment) depends on the rotation rate and an empirically determined constant k_M. Torque acts in the opposite direction to the propeller's rotation, meaning that a clockwise (CW) spinning propeller induces a counterclockwise (CCW) torque on the quadcopter's body. This principle aligns with Newton's third law: every action has an equal and opposite reaction. While the rate of change of rotor speed affects the dynamics, this powertrain behaviour is often neglected in control system design for simplicity.

When building a mathematical model of a quadcopter's dynamics, many forces can influence motion. However, to keep the model practical and solvable, simplifications are necessary. The primary forces considered are gravity, air resistance, thrust, and torque. Because thrust and torque can be modulated by varying rotor speeds, they serve as the system's control inputs. The quadcopter's motion can be described in terms of four control actions: altitude, roll, pitch, and yaw. These control inputs are defined as:

$$\begin{aligned} U_1 &= F_1 + F_2 + F_3 + F_4 = k_F(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_2 &= L(F_2 - F_4) = Lk_F(\omega_2^2 - \omega_4^2) \\ U_3 &= L(F_3 - F_1) = Lk_F(\omega_3^2 - \omega_1^2) \\ U_4 &= M_1 - M_2 + M_3 - M_4 = k_M(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{aligned} \qquad (1.2)$$

If the quadcopter is treated as a perfectly rigid body in space, its motion can be modeled using Euler's equations for rigid body dynamics. Together with Newton's second law (which governs linear acceleration due to applied force), Euler's equations enable a detailed description of the quadcopter's motion. The combined system of differential equations describes both translational and rotational dynamics.

Equations (1.3) represent Newton's second law and Euler's rotational dynamics, respectively:

$$\begin{aligned} F &= ma \\ M &= I\dot{\omega} + \omega \times (I\omega) \end{aligned} \qquad (1.3)$$

where:

- F is the total force,
- m is the mass,
- a is linear acceleration,
- M is the applied torque,
- I is the inertia matrix (distinct from the identity matrix),
- ω is angular velocity,
- $\dot{\omega}$ is angular acceleration.

To describe the quadcopter's motion and orientation accurately, coordinate systems play a crucial role. Typically, two reference frames are used: the World frame and the body frame. These frames help facilitate simulation, control, and interpretation of the quadcopter's behavior. Without a fixed reference frame, it would be difficult to describe the vehicle's pose and orientation in a consistent way.

Sensors such as gyroscopes, accelerometers, and GPS units are essential for determining the quadcopter's motion and position. The data from these devices must be interpreted correctly with respect to the chosen reference frames. In aerospace, Tait-Bryan angles are often used to represent orientation, providing a practical method to describe rotation in three dimensions.

Tait-Bryan angles represent orientation with respect to the Earth frame. Unlike Euler angles, which use zero degrees for vertical alignment, Tait-Bryan angles define zero degrees as aligned with the horizontal plane. These angles translate the body frame's orientation to the Earth frame using a specific rotation order.

In this context, the ZYX convention is applied, meaning the rotations are carried out in the order: z -axis, then y-axis, and finally x-axis. This sequence first applies yaw (around the z-axis), then pitch (around the y-axis), and finally roll (around the x -axis). The corresponding rotation matrices are derived accordingly.

The elementary rotation matrices define the rotation about the principal axes. They are given by: Rotation about the x-axis (Roll):

$$R_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{1.4}$$

Rotation about the y-axis (Pitch):

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{1.5}$$

Rotation about the z-axis (Yaw):

$$R_Z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.6}$$

The total rotation matrix from the body frame to the Earth frame is obtained by combining the three individual rotations:

The matrices $R_X(\phi), R_Y(\theta)$, and $R_Z(\psi)$ shown in Equations (2.16) to (2.18) are the basic rotation matrices around the x^{-}, y^{-}, and z-axes, respectively. In these expressions, c and s represent the cosine and sine of the respective angles.

$$R_b^G = R_X(\phi)R_Y(\theta)R_Z(\psi) = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\theta)s(\phi) - c(\phi)s(\psi) & c(\psi)c(\phi)s(\theta) + s(\psi)s(\phi) \\ c(\theta)s(\psi) & s(\psi)s(\theta)s(\phi) + c(\phi)c(\psi) & c(\phi)s(\psi)s(\theta) - c(\theta)s(\phi) \\ s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \quad (1.7)$$

Here, $c(\cdot)$ and $s(\cdot)$ denote cosine and sine functions, respectively. This rotation matrix R_b^G is also called a transformation matrix, as it transforms a vector v_e from the Earth frame to a vector v_0 in the body frame. If the body-frame linear velocity is given by $[u \quad v \quad w]^T$, then its equivalent in the Earth frame is:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_b^G \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$= \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\theta)s(\phi) - c(\phi)s(\psi) & c(\psi)c(\phi)s(\theta) + s(\psi)s(\phi) \\ c(\theta)s(\psi) & s(\psi)s(\theta)s(\phi) + c(\phi)c(\psi) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \begin{bmatrix} u \\ \mathrm{v} \\ w \end{bmatrix} \quad (1.8)$$

This leads to the expanded equations:

$$\begin{aligned} \dot{x} &= u(c(\psi)c(\theta)) + v(c(\psi)s(\theta)s(\phi) - c(\phi)s(\psi)) + w(c(\psi)c(\phi)s(\theta) + s(\psi)s(\phi)) \\ \dot{y} &= u(c(\theta)s(\psi)) + v(s(\psi)s(\theta)s(\phi) + c(\phi)c(\psi)) + w(c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi)) \\ \dot{z} &= u\big(-s(\theta)\big) + v\big(c(\theta)s(\phi)\big) + w\big(c(\theta)c(\phi)\big) \end{aligned} \quad (1.9)$$

These equations describe how the linear velocity in the Earth frame is derived from the velocities in the body frame through coordinate transformation using the rotation matrix R_b^C.

To represent the full rotation from the body frame to the inertial (Earth) frame, the combined rotation matrix R_b^G is formed by multiplying the three individual matrices in the order $R_X(\phi)R_Y(\theta)R_Z(\psi)$, as shown in Equation (2.19). This matrix is also referred to as the transformation matrix because it converts a vector v_e expressed in the Earth frame to the equivalent vector v_b in the body frame using the relation $v_b = R_b^G v_e$. If the body-frame linear velocity is given by the vector $[u \quad v \quad w]^T$, then applying the transformation matrix R_b^G yields the velocity in the inertial frame $[\dot{x} \quad \dot{y} \quad \dot{z}]^T$, as shown in Equation (2.20). The resulting expressions for the inertial velocity components are derived in Equation (2.21), where:

- $\dot{x}$ is computed using the combination of body-frame velocities u, v, w and the trigonometric terms involving the roll, pitch, and yaw angles;
- $\dot{y}$ and $\dot{z}$ are derived similarly by projecting the body-frame velocities through the rotation matrix.

These equations represent how Newton's second law is extended to rotational systems by applying rotation transformations between coordinate frames.

In the body reference frame, the acceleration can be described using Newton's second law, $F = ma$. For a quadcopter, this becomes:

$$f_b = m \cdot a_b \tag{1.10}$$

Here, a_b represents the total acceleration in the body frame. It includes both linear acceleration $\dot{v}_b$ and the angular acceleration induced by rotation, $\omega_b \times v_b$. Combining these gives:

$$f_b = m(\omega_b \times v_b + \dot{v}_b) \tag{1.11}$$

Rearranging the expression to isolate the linear acceleration:

$$\dot{v}_b = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{f_b}{m} - \omega_b \times v_b = \frac{f_b}{m} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \frac{f_b}{m} - \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} \tag{1.12}$$

The net force f_b can be broken into components along the three body axes as $f_b = [f_{bx} \quad f_{by} \quad f_{bz}]$. Additionally, to reflect the effect of aerodynamic drag (which opposes the velocity), we introduce drag forces proportional to the body velocities. These act in each direction and are represented by d_x, d_y, d_z as drag coefficients. The equations become:

$$\begin{cases} \dot{u} = \frac{f_{bx}}{m} - qw + rv - \frac{d_x}{m} u \\ \dot{v} = \frac{f_{by}}{m} - ru + pw - \frac{d_y}{m} v \\ \dot{w} = \frac{f_{bz}}{m} - pv + qu - \frac{d_z}{m} w \end{cases} \tag{1.13}$$

The external forces acting on the quadcopter can be split into gravitational force components aligned with the body axes. Thrust acts in the body z-direction and opposes gravity. The gravitational terms are influenced by the orientation angles ϕ and θ, whereas the yaw angle ψ has no direct effect.

- In the $\mathbf{x}$-direction, the gravity component is:

$$mg(s(\phi)s(\theta))$$

- In the y-direction:

$$-mg(c(\phi)s(\theta)) \tag{1.14}$$

- In the $\mathbf{z}$-direction (including thrust):

$$-mg(c(\theta)c(\phi)) + U_1$$

This extended formulation accounts for not only gravitational and control forces, but also realistic aerodynamic drag, offering a more complete dynamic model of the quadcopter.

The linear acceleration components of the quadcopter in the body frame can be further expanded by incorporating the gravitational forces and control inputs. These forces are influenced by the orientation angles ϕ and θ, as previously explained. Assuming the external body forces are purely due to gravity and thrust, the resulting equations become:

$$\begin{cases} \dot{u} = g(s(\theta)) - qw + rv - \frac{d_x}{m}u \\ \dot{v} = -g(c(\theta)s(\phi)) - ru + pw - \frac{d_y}{m}v \\ \dot{w} = -g(c(\theta)c(\phi)) + \frac{U_1}{m} - pv + qu - \frac{d_z}{m}w \end{cases} \tag{1.15}$$

These expressions account for:

- Gravitational components derived from orientation,
- Thrust contribution $\frac{U_1}{m}$ along the body z-axis,
- Aerodynamic drag in each direction,
- Coriolis terms from cross-product of angular and linear velocity.

To model the rotational motion of the quadcopter, we define the angular velocity vector in the body frame as $\dot{\omega} = [\dot{p}\dot{q}\dot{r}]^T$. Applying Euler's rigid body rotation equation from (2.15), and including the effects of the generated control moments from each rotor, the rotational dynamics are:

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{1.16}$$

This formulation incorporates the inertial response of the vehicle (via the inertia matrix I) and the gyroscopic effects (through the $\omega \times (I\omega)$ term), capturing the full 3D rotational dynamics of the quadcopter.

The term I refers to the quadcopter's inertia matrix, which encapsulates its resistance to rotational acceleration. The parameter L represents the fixed distance from the vehicle's center of mass to the center of each rotor. Because the quadcopter is assumed to be symmetrical, this distance is constant for all arms. The inertia matrix is diagonal and composed of the values I_x, I_y, and I_z, corresponding to the principal axes.

The torque matrix $M_{\text{TorqueMatrix}}$ in Equation (2.27) reflects the fact that the torque about the z -axis results from the total moment generated by all four rotors. For example, torque around the $\mathbf{x}$-axis is produced by the difference in thrust between rotors 2 and 4 , so this yields the term $L(F_2 - F_4)$. Similarly, torque around the $\mathbf{y}$-axis arises from rotors 3 and 1. The signs of these terms are dictated by rotor orientation (i.e., direction of rotation). For instance, if $L(F_2 - F_4)$ is positive, then $L(F_4 - F_2)$ would correspond to an equal but opposite torque. The same interpretation applies for the $\mathbf{y}$-axis torque from rotors 1 and 3..

The connection between angular velocities in the body frame $[pqr]^T$ and the angular rates of Euler angles $[\dot{\phi}\dot{\theta}\dot{\psi}]^T$ is described by:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{1.17}$$

This transformation arises by sequentially considering rotational transitions between the Earth and body frames. Each individual rotation (roll, pitch, yaw) is introduced one at a time, accumulating their effects.

The full mapping from angular body rates $[pqr]^T$ to Euler angle rates can be constructed by applying the composition of rotational matrices:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = R_x(\phi)R_y(\theta)\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R_x(\phi)\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{1.18}$$

1.2 System Equations

The earlier subsection presented the theoretical derivation of the quadcopter's dynamics. Building on that foundation, we can now formulate the system equations. If the expressions for the time derivatives of position $[\dot{x}, \dot{y}, \dot{z}]$ are required. To determine the angular rates of the Euler angles-roll (ϕ), pitch (θ), and yaw (ψ)-we simplify Equation into the following form:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \tag{1.19}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ q\cos(\phi) - r\sin(\phi) \\ q\sin(\phi)\sec(\theta) + r\cos(\phi)\sec(\theta) \end{bmatrix}$$

To derive the rate of angular velocity, we start from Euler's equation (Equation 2.27). By pre-multiplying both sides with the inverse of the inertia matrix, we obtain:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \left(\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} - \begin{bmatrix} d_p p \\ d_q q \\ d_r r \end{bmatrix} \right) I^{-1} \tag{1.20}$$

Here, the terms $d_p p, d_q q$, and $d_r r$ represent rotational aerodynamic drag acting against roll, pitch, and yaw, respectively.

Writing this with the expanded inertia matrix:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \left(\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} - \begin{bmatrix} d_p p \\ d_q q \\ d_r r \end{bmatrix} \right) I^{-1} \tag{1.21}$$

This expression includes both the gyroscopic coupling from angular momentum and the rotational damping due to aerodynamic drag, which is critical for accurate simulation and control of a real quadrotor system.

After expanding the gyroscopic cross-product term from Euler's equation and applying the inverse of the inertia matrix, we arrive at the angular acceleration in component form:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \left(\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} qrI_z - rqI_y \\ rpI_x - prI_z \\ pqI_y - qpI_x \end{bmatrix} \right) I^{-1} \tag{1.22}$$

This simplifies further as:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{L}{I_x}(F_2 - F_4) + qr \cdot \frac{I_y - I_z}{I_x} \\ \frac{L}{I_y}(F_3 - F_1) + rp \cdot \frac{I_z - I_x}{I_y} \\ \frac{1}{I_z}(M_1 - M_2 + M_3 - M_4) + pq \cdot \frac{I_x - I_y}{I_z} \end{bmatrix} \tag{1.23}$$

Next, we substitute the force and moment equations:

$$F_i = k_F \omega_i^2 \text{ and } M_i = k_M \omega_i^2$$

This yields:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{L}{I_x} k_F(\omega_2^2 - \omega_4^2) + qr \cdot \frac{I_y - I_z}{I_x} \\ \frac{L}{I_y} k_F(\omega_3^2 - \omega_1^2) + rp \cdot \frac{I_z I_x}{I_y} \\ \frac{1}{I_z} k_M(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) + pq \cdot \frac{I_x - I_y}{I_z} \end{bmatrix} \tag{1.24}$$

This yields:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{L}{I_X} k_F(\omega_2^2 - \omega_4^2) + qr \cdot \frac{I_y - I_z}{I_x} \\ \frac{L}{I_y} k_F(\omega_3^2 - \omega_1^2) + rp \cdot \frac{I_z - I_x}{I_y} \\ \frac{1}{I_z} k_M(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) + pq \cdot \frac{I_x - I_y}{I_z} \end{bmatrix} \tag{1.25}$$

Finally, we incorporate aerodynamic drag torques $d_p p, d_q q, d_r r$ acting about each axis. These are damping terms that represent energy lost due to air resistance:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{L}{I_x} k_F(\omega_2^2 - \omega_4^2) + qr \cdot \frac{I_y - I_z}{I_x} - \frac{d_p}{I_d} p \\ \frac{L}{I_y} k_F(\omega_3^2 - \omega_1^2) + rp \cdot \frac{I_z - I_x}{I_y} - \frac{d_q}{I_y} q \\ \frac{1}{I_z} k_M(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) + pq \cdot \frac{I_x - I_y}{I_z} - \frac{d_r}{I_z} r \end{bmatrix} \tag{1.26}$$

This completes the final form of the rotational dynamics, now accounting for both inertial coupling and aerodynamic damping, making the model more physically accurate for simulation and control.

Aerodynamic drag is a resistive force that acts in opposition to motion. It is typically modeled as a quadratic function of the body-frame velocity components:

$$F_{D,i} = -\frac{1}{2}\rho C_d A_i v_i |v_i|, \text{ for } i \in \{x, y, z\} \tag{1.17}$$

Where:

- ρ : Air density $(\mathrm{kg/m^3})$
- C_d : Drag coefficient (dimensionless)

- A_i : Cross-sectional area along axis i
- v_i : Velocity component u, v, w along the respective axis

This drag force introduces deceleration terms in the translational acceleration equations of the quadcopter.

Including aerodynamic drag, the body-frame translational accelerations become:

$$\begin{aligned} \dot{u} &= g\sin(\theta) - qw + rv - \tfrac{1}{2m}\rho C_d A_x u|u| \\ \dot{v} &= -g\cos(\theta)\sin(\phi) - ru + pw - \tfrac{1}{2m}\rho C_d A_y v|v| \\ \dot{w} &= -g\cos(\theta)\cos(\phi) + \tfrac{U_1}{m} - pv + qu - \tfrac{1}{2m}\rho C_d A_z w|w| \end{aligned} \tag{1.28}$$

1.3 Kinematics and Dynamics (Earth + Body Frames)

$$\begin{gathered} \dot{x} = u \cdot (c_\psi c_\theta) + v \cdot (c_\psi s_\theta s_\phi - c_\phi s_\psi) + w \cdot (c_\psi c_\phi s_\theta + s_\psi s_\phi) \\ \dot{y} = u \cdot (s_\psi c_\theta) + v \cdot (s_\psi s_\theta s_\phi + c_\psi c_\phi) + w \cdot (s_\psi c_\phi s_\theta - c_\psi s_\phi) \\ \dot{z} = -u \cdot \sin(\theta) + v \cdot \cos(\theta)\sin(\phi) + w \cdot \cos(\theta)\cos(\phi) \\ \dot{\phi} = p + q \cdot \sin(\phi)\tan(\theta) + r \cdot \cos(\phi)\tan(\theta) \\ \dot{\theta} = q \cdot \cos(\phi) - r \cdot \sin(\phi) \\ \dot{\psi} = q \cdot \sin(\phi)\sec(\theta) + r \cdot \cos(\phi)\sec(\theta) \\ \dot{u} = g\sin(\theta) - qw + rv - \tfrac{1}{2m}\rho C_d A_x u|u| \\ \dot{v} = -g\cos(\theta)\sin(\phi) - ru + pw - \tfrac{1}{2m}\rho C_d A_y v|v| \\ \dot{w} = -g\cos(\theta)\cos(\phi) + \tfrac{U_1}{m} - pv + qu - \tfrac{1}{2m}\rho C_d A_z w|w| \end{gathered} \tag{1.29}$$

Rotational Dynamics

$$\begin{aligned} \dot{p} &= \tfrac{U_2}{I_x} + qr \cdot \tfrac{I_y - I_z}{I_x} \\ \dot{q} &= \tfrac{U_3}{I_y} + rp \cdot \tfrac{I_z - I_x}{I_y} \\ \dot{r} &= \tfrac{U_4}{I_z} + pq \cdot \tfrac{I_x - I_y}{I_z} \end{aligned} \tag{1.30}$$

Legend (Symbol Definitions)

- x, y, z : Linear position in inertial (Earth) frame
- ϕ, θ, ψ : Angular position (roll, pitch, yaw)
- u, v, w : Linear velocities in body frame
- p, q, r : Angular velocities in body frame
- U_1, U_2, U_3, U_4 : Collective thrust and torques (control inputs)

- C_d : Drag coefficient
- A_x, A_y, A_z : Projected cross-sectional areas
- I_x, I_y, I_z : Inertias
- ρ : Air density

The values of the parameters are given by Table 1.0 below. A test bed can be developed and used for conducting the values, but in this case the data is obtained from [2]

Description	Parameter	Value [SI unit]
Quadcopter mass	m	$0.547[\mathrm{~kg}]$
Acceleration of gravity	g	$9.81[\mathrm{~m/s^2}]$
Arm length	L	$0.17[\mathrm{~m}]$
Thrust coefficient	k_F	$1.5 \cdot 10^{-7}[\mathrm{~N} \cdot \mathrm{RPM}^{-2}]$
Torque coefficient	k_M	$3.75 \cdot 10^{-7}[\mathrm{Nm} \cdot \mathrm{RPM}^{-2}]$
Air resistance coefficient	C_d	$1.0[-]$
Inertia for x_b	I_x	$3.3 \cdot 10^{-3}[\mathrm{~kg} \cdot \mathrm{~m}^2]$
Inertia for y_b	I_y	$3.3 \cdot 10^{-3}[\mathrm{~kg} \cdot \mathrm{~m}^2]$
Inertia for z_b	I_z	$5.8 \cdot 10^{-3}[\mathrm{~kg} \cdot \mathrm{~m}^2]$
Cross-sectional area for x_b	A_x	$0.011[\mathrm{~m}^2]$
Cross-sectional area for y_b	A_y	$0.011[\mathrm{~m}^2]$
Cross-sectional area for z_b	A_z	$0.022[\mathrm{~m}^2]$

Table 1.0: The values of the parameters used in the model.

2.0 Linearisation and State-Space Representation

To implement a Linear Quadratic Regulator (LQR) controller, it is necessary to express the nonlinear dynamics of the quadcopter in a linear state-space form. This requires defining a state vector and linearizing the nonlinear system around a stable operating point.

Let the state vector $\boldsymbol{\chi}$ (Greek letter chi) be defined as:

$$\chi = [x \quad y \quad z \quad \phi \quad \theta \quad \psi \quad u \quad v \quad w \quad p \quad q \quad r]^T \tag{2.0}$$

and the standard state-space representation is written as:

$$\dot{\boldsymbol{\chi}} = A\boldsymbol{\chi} + Bu \tag{2.1}$$

To linearize, we expand the nonlinear dynamics around a stationary (hover) equilibrium point ($\boldsymbol{\chi}_s, \boldsymbol{u}_s$), sometimes denoted as $(\boldsymbol{\chi}_e, \boldsymbol{u}_e)$ or $(\boldsymbol{\chi}_0, \boldsymbol{u}_0)$. The system behaves approximately linearly about this point.

For hover, we assume the quadcopter is upright, experiencing no rotation or lateral motion, and undergoing only vertical movement. This lets us apply the small-angle approximation:

$$\cos(\cdot) \approx 1,\ \sin(\cdot) \approx \cdot,\ \tan(\cdot) \approx$$

Linearized System Equations (with Small-Angle Assumptions and Drag) Substituting these into the full nonlinear model yields the following linearized expressions:

$$\begin{aligned}
\dot{x} &= u + v(\theta\phi - \psi) + w(\theta + \psi\phi) \\
\dot{y} &= u\psi + v(\psi\theta\phi + 1) + w(\psi\theta - \phi) \\
\dot{z} &= -u\theta + v\phi + w \\
\dot{\phi} &= p + q\phi\theta + r\theta \\
\dot{\theta} &= q - r\phi \\
\dot{\psi} &= q\phi + r \\
\dot{u} &= g\theta - qw + rv - \frac{1}{m}\rho C_d A_x u \\
\dot{v} &= -g\phi - ru + pw - \frac{1}{m}\rho C_d A_y v \\
\dot{w} &= -g + \frac{U_1}{m} - pv + qu - \frac{1}{m}\rho C_d A_z w \\
\dot{p} &= \frac{U_2}{I_x} + qr \cdot \frac{I_y - I_z}{I_x} \\
\dot{q} &= \frac{U_3}{I_y} + rp \cdot \frac{I_z - I_x}{I_y} \\
\dot{r} &= \frac{U_4}{I_z} + pq \cdot \frac{I_x - I_y}{I_y}
\end{aligned} \tag{2.2}$$

2.1 Linearisation of the aerodynamical drag term

The aerodynamic drag terms $\frac{1}{2m}\rho C_d A_i v_i |v_i|$ are nonlinear. To incorporate them in the linearized model, we approximate them using their Jacobian (first-order Taylor expansion):

Let us define drag-related damping coefficients:

$$\epsilon_x = \frac{1}{m}\rho C_d A_x |u|,\ \epsilon_y = \frac{1}{m}\rho C_d A_y |v|,\ \epsilon_z = \frac{1}{m}\rho C_d A_z |w| \tag{2.3}$$

These are small but nonzero around hover (since small but finite velocities may occur during maneuvers). The linearized damping terms for the A matrix are:

$$\frac{\partial \dot{u}}{\partial u} = -\epsilon_x,\ \frac{\partial \dot{v}}{\partial v} = -\epsilon_y,\ \frac{\partial \dot{w}}{\partial w} = -\epsilon_z \tag{2.4}$$

These terms act as linear damping along each axis and are incorporated symbolically into the state-space model as negative diagonals in the A matrix.

To include them symbolically, we introduce them as linear damping terms.

$$\hat{x} = f(\chi, u) = \begin{bmatrix} u + v(\theta \cdot \phi - \phi) + w(\theta + \psi \cdot \phi) \\ u - \psi + v(\psi - \theta \cdot \phi + 1) + w(\psi \cdot \theta - \phi) \\ u \cdot (-\theta) + v \cdot \phi + w \\ p + q \cdot \phi \cdot \theta + r \cdot \theta \\ q - r \cdot \phi \\ q - \phi + r \\ g - \theta - qw + rv\ - \frac{1}{m}\rho C_d A_x u, \\ -g \cdot \phi - ru + pw\ - \frac{1}{2m}\rho C_d A_y v, \\ -g + \frac{U_1}{w} - pv + qu\ - \frac{1}{2m}\rho C_d A_z w, \\ \frac{U_2}{T_1} + qr\frac{l_y - I_z}{l_x} \\ \frac{U_1}{l_y} + rp\frac{l_s - l_x}{l_y} \\ \frac{U_4}{T_z} + pq\frac{l_x - l_z}{I_z} \end{bmatrix} \tag{2.5}$$

Defining $\delta X = X - Xx$ and $\delta u = u - u_x$ as small perturbations, a Taylor series expansion around the stationary point (while only keeping the linear terms, thus neglecting the term $f(X_x, u_s)$ and all high order terms) yields

$$\delta \dot{X} = \frac{\partial f(X_x, u_x)}{\partial_X}\delta X + \frac{\partial f(X_s, u_s)}{\partial u}\delta u. \tag{2.6}$$

The partial derivative (δ) represents the Jacobian matrix. If χ_s and u_s are stationary (equilibrium) solutions, then $x = f(\chi_x, u_x)$ becomes zero. This implies that for equilibrium point χ_s, the input u_x leads to the solution of the algebraic system $f(\chi_s, u_s) = 0$. As mentioned earlier, the stationary point is an arbitrary point in space where the quadcopter hovers, which means that the state vector can be written as

$$x_x = [x_s \quad y_s \quad z_s \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \tag{2.7}$$

To achieve hovering straight up only the thrust force for each rotor needs to change equally, which means that $U_1 = mg$ (to counteract gravity) and $U_2 = U_3 = U_4 = 0$. Hence, u_x is

$$U_s = [mg \quad 0 \quad 0 \quad 0]^T$$

Examining Equation (2.5) (which is in the form of a linear time-invariant state equation $\hat{X} = A\chi + Bu$), one can observe that for the linear approximation system, the matrix A (the Jacobian of $f(\chi, u)$ with respect to χ) and B (the Jacobian of $f(\chi, u)$ with respect to u) are expressed as

$$A = \frac{\partial f(\chi,u)}{\partial \chi}\bigg|_{\chi=\chi_s,u=u_s} = \left.\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} & \frac{\partial f_1}{\partial \phi} & \cdots & \frac{\partial f_1}{\partial r} \\ \frac{\partial f_2}{\partial \dot{x}} & \frac{\partial f_2}{\partial \dot{y}} & \frac{\partial f_2}{\partial \dot{z}} & \frac{\partial f_2}{\partial \phi} & \cdots & \frac{\partial f_2}{\partial r} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{12}}{\partial \dot{x}} & \frac{\partial f_{12}}{\partial \dot{y}} & \frac{\partial f_{12}}{\partial \dot{z}} & \frac{\partial f_{12}}{\partial \phi} & \cdots & \frac{\partial f_{12}}{\partial r} \end{bmatrix}\right|_{\chi=\chi_s,u=u_s}, \tag{2.8}$$

$$B = \frac{\partial f(\chi,u)}{\partial u}\bigg|_{\chi=\chi_s,u=u_s} = \left.\left|\begin{bmatrix} \frac{\partial f_1}{\partial U_1} & \frac{\partial f_1}{\partial U_2} & \frac{\partial f_1}{\partial U_3} & \frac{\partial f_1}{\partial U_4} \\ \frac{\partial f_2}{\partial U_1} & \frac{\partial f_2}{\partial U_2} & \frac{\partial f_2}{\partial U_3} & \frac{\partial f_2}{\partial U_4} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{12}}{\partial U_1} & \frac{\partial f_{12}}{\partial U_2} & \frac{\partial f_{12}}{\partial U_3} & \frac{\partial f_{12}}{\partial U_4} \end{bmatrix}\right|\right|_{\chi=\chi_s,u=u_s}. \tag{2.9}$$

Solving for A and B matrices results

$$A = \left.\left|\begin{bmatrix} 0 & 0 & 0 & v\theta + w\psi & v\phi + w & -v + w\phi & 1 & \theta\phi - \psi & \theta + \psi\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & v\psi\theta - w & v\psi\phi + w\psi & u + v\theta\phi + w\theta & \psi & \psi\theta\phi + 1 & \psi\theta - \phi & 0 & 0 & 0 \\ 0 & 0 & 0 & v & -u & 0 & -\theta & \phi & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & q\theta & q\phi + r & 0 & 0 & 0 & 0 & 1 & \phi\theta & \theta \\ 0 & 0 & 0 & -r & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\phi \\ 0 & 0 & 0 & q & 0 & 0 & 0 & 0 & 0 & 0 & \phi & 1 \\ 0 & 0 & 0 & 0 & g & 0 & -\epsilon_x & r & -q & 0 & -w & v \\ 0 & 0 & 0 & -g & 0 & 0 & -r & -\epsilon_y & p & w & 0 & -u \\ 0 & 0 & 0 & 0 & 0 & 0 & q & -p & -\epsilon_z & -v & u & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r\frac{I_y - I_z}{I_x} & q\frac{I_y - I_z}{I_z} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r\frac{I_z - I_x}{I_y} & 0 & p\frac{I_z - I_x}{I_y} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q\frac{I_x - I_y}{I_z} & p\frac{I_x - I_y}{I_z} & 0 \end{bmatrix}\right|\right|_{\chi=\chi_s,u=U_s} \tag{2.30}$$

Where,

- $\epsilon_x = \frac{1}{m}\rho C_d A_x$
- $\epsilon_y = \frac{1}{m}\rho C_d A_y$ (2.31)
- $\epsilon_z = \frac{1}{m}\rho C_d A_z$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix}_{\chi=\chi_s, u=u_s} . \tag{2.32}$$

2.2 Linearised system at hover position

Upon applying the hover equilibrium:

- $\phi = \theta = \psi = 0$
- $u = v = w = p = q = r = 0$
- $\mathrm{U}_1 = \mathrm{mg}, \mathrm{U}_2 = \mathrm{U}_3 = \mathrm{U}_4 = 0$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \tag{2.33}$$

Now the SSR ($\dot{\chi} = A\chi + Bu$) of the linear system is obtained, where matrices A and B are defined by Equations (2.43) and (2.44), respectively. The expression for $\dot{\chi}$ is as follows

$$\begin{cases} \dot{x} = u \\ \dot{y} = v \\ \dot{z} = w \\ \dot{\phi} = p \\ \dot{\theta} = q \\ \dot{\psi} = r \\ \dot{u} = g\theta \\ \dot{v} = g\phi \\ \dot{w} = -\frac{1}{m} \\ \dot{p} = \frac{U_2}{I_x} \\ \dot{q} = \frac{U_3}{I_y} \\ \dot{r} = \frac{U_4}{I_z} \end{cases}, \qquad \dot{\chi} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \\ g\theta - \epsilon_x u \\ g\phi - \epsilon_y v \\ -\frac{1}{m}U_1 - \epsilon_z w \\ \frac{1}{I_x}U_2 \\ \frac{1}{I_y}U_3 \\ \frac{1}{I_z}U_4 \end{bmatrix} \tag{2.34}$$

Physical Interpretation of Each Equation

Equation	Physical Meaning
$\dot{x} = u$	x -position changes with forward velocity (along body x -axis)
$\dot{y} = v$	y-position changes with lateral velocity (along body y -axis)
$\dot{z} = w$	z-position changes with vertical velocity (along body z-axis)
$\dot{\phi} = p$	Roll angle changes at the rate of angular velocity around x -axis
$\dot{\theta} = q$	Pitch angle changes at the rate of angular velocity around y -axis
$\dot{\psi} = r$	Yaw angle changes at the rate of angular velocity around z -axis
$\dot{u} = g\theta - \epsilon_x u$	Forward acceleration due to pitch angle, reduced by drag
$\dot{v} = g\phi - \epsilon_y v$	Lateral acceleration due to roll angle, reduced by drag
$\dot{w} = -\frac{U_1}{m} - \epsilon_z w$	Vertical acceleration from thrust, opposed by drag and gravity
$\dot{p} = \frac{U_2}{I_x}$	Roll acceleration caused by torque about x -axis
$\dot{q} = \frac{U_3}{I_y}$	Pitch acceleration caused by torque about y-axis
$\dot{r} = \frac{U_4}{I_z}$	Yaw acceleration caused by torque about z-axis

Table 2.0: Physical Interpretation of Each Equation

2.3 Symbolic Linearization Approach

To design a controller using the LQR framework, we first express the nonlinear quadrotor dynamics in a linear state-space form. This is done by symbolically computing the Jacobian matrices of the system equations with respect to the state and input vectors:

$$\dot{x} = f(x, u) \tag{2.35}$$

To linearize, we apply a first-order Taylor series expansion around the hovering equilibrium point (x_s, u_s), where:

- The quadrotor hovers at a fixed position with zero velocity and zero angular rotation.
- The thrust input balances gravity: $U_1 = mg$, and torques are zero: $U_2 = U_3 = U_4 = 0$.

We define the perturbation variables:

$$\delta x = x - x_s,\ \delta u = u - u_s \tag{2.36}$$

Then the linearized system becomes:

$$\delta\dot{x} = A\delta x + B\delta u \tag{2.37}$$

Where:

- $A = \frac{\partial f(x,u)}{\partial x}\Big|_{x_s,u_s}$ (2.38)
- $B = \frac{\partial f(x,u)}{\partial u}\Big|_{x_s,u_s}$

These Jacobians were computed symbolically in MATLAB using the Symbolic Math Toolbox. This approach ensures that all partial derivatives, including those related to aerodynamic drag terms (nonlinear in velocity), are correctly captured.

2.3 Controllability and Observability

With the linearised system established, the implementation of the LQR controller can now proceed. MATLAB's built-in commands such as ctrb (for controllability), obsv (for observability), and rank() can confirm whether the system is fully controllable and observable. This is validated by ensuring that both the controllability and observability matrices are full rank—that is, they have rank equal to the number of system states (12).

```
A = jacobian(f, x);
B = jacobian(f, u);
C = eye(12); D = zeros(12,4);

ctrb_matrix = ctrb(A,B);
obsv_matrix = obsv(A,C);
rank_ctrb = rank(ctrb_matrix);  % Should be 12
rank_obsv = rank(obsv_matrix);  % Should be 12
```

To ensure that the designated controller can influence and that observer can observe all system states, we verify controllability and observability of the linearized model. To ensure that the designated controller can influence and observe all system states, we verify controllability and observability of the linearised model.

In this project, both matrices were confirmed to be full rank, indicating that the system is suitable for state-feedback control and state estimation using observers or Kalman filters and Extended Kalman Filters.

3.0 LQR Control Implementation

3.1 Objective

The goal of this section is to design and implement an LQR (Linear Quadratic Regulator) controller to stabilize the quadcopter at a hovering state. This controller works based on the previously derived linearized model that includes aerodynamic drag effects. It regulates all 12 states of the system and corrects deviations caused by external disturbances.

3.2 State-Space Model

The dynamics around the hover condition are modeled as a linear time-invariant system:

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad (3.0)$$

Where:

- $x(t) \in \mathbb{R}^{12}$: The state vector, consisting of positions, Euler angles, linear velocities, and angular rates.
- $u(t) \in \mathbb{R}^{4}$: The input vector representing collective thrust and control torques.
- A, B : Matrices derived from symbolic linearisation, representing the linearised system dynamics, coupling between states, and aerodynamic drag.

The LQR controller minimizes the following performance index:

4.3 LQR Gain Design

The LQR controller minimizes the following performance index:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{3.1}$$

Where:

- $Q = \text{diag}\,(10,10,10,100,100,100,1,1,1,1,1,1)$: Emphasizes precise control over position and orientation.
- $R = 0.1 \cdot I_4$: Applies a uniform penalty on all control inputs to limit actuation effort.

The feedback gain matrix $K \in \mathbb{R}^{4\times12}$ is computed using MATLAB's lqr() function.
4.4 Simulink Implementation

The closed-loop system was modeled in Simulink using the following components:

- State-Space Block: Encodes the system matrices A, B, C, and D.
- LQR Gain Block: Computes the control law $u(t) = -Kx(t)$.
- Initial Condition Block: Applies an initial deviation (e.g., offset in position or orientation) to test the controller's performance.
- Autonomous Response: No external inputs are applied after initialization; the system autonomously returns to the hover condition.