GJU

الجامعة الألمانية الأردنية

German Jordanian University

Quadcopter model and control

School of applied technical sciences

Mechatronics engineering

German Jordanian University

Bachelors thesis

Students:

Amro Habboush 20181102011

Rami Abu Al Nadi 20181102047

Supervisor:

Dr. Ghaith Alrefai

Contents

Figures:

Tables:

Nomenclature

- ξ: Quadcopter's linear location
- ξ': Quadcopter's linear velocity
- ξ'': Quadcopter's linear acceleration
- η: Euler angles
- η': Euler angles velocity
- η'': Euler angles acceleration
- ϕ: Roll angel (Rotation around X axis)
- θ: Pitch angel (Rotation around Y axis)
- ψ: Yaw angel (Rotation around Z axis)
- ϕ': Roll angel velocity
- θ': Pitch angel velocity
- ψ': Yaw angel velocity
- ϕ'': Roll angel acceleration
- θ'': Pitch angel acceleration
- ψ'': Yaw angel acceleration
- X: Linear position (on X axis)

- Y: Linear position (on Y axis)
- Z: Linear position (on Z axis)
- X': Linear velocity (on X axis)
- Y': Linear velocity (on Y axis)
- Z': Linear velocity (on Z axis)
- X'': Linear acceleration (on X axis)
- Y'': Linear acceleration (on Y axis)
- Z'': Linear acceleration (on Z axis)
- M: mass
- G: Gravity
- R: Rotational matrix
- T: Thrust
- C: Cosine
- S: Sine
- D/dt: Time derivative
- $W^{-1}\eta$: Transformation matrix η from the body frame to the inertial frame.
- V: Angular velocity
- V': Angular acceleration
- L: Lagrangian energy
- Q: The linear and angular position vectors
- I: Identity matrix
- Z: Height
- A: drag force coefficients for velocities
- E(t): Error (difference)
- Xd(t): Desired state
- X(t): Current state
- U(t): The control input
- KP: Parameters for the proportional
- KI: Parameters for the integral
- KD: Parameters for the derivative
- W: Angular velocity (system input)
- K: Thrust coefficient
- τ: Torque
- L: Distance from the center of the quadcopter to each motor
- B: Torque coefficient (drag) Tb: Thrust (body frame)
- Ax: Drag force coefficients for velocities (on X axis)
- Ay: Drag force coefficients for velocities (on Y axis)

- Az: Drag force coefficients for velocities (on Z axis)
- Dx, Dy, Dz: Effective acceleration over each axis
- $\dot{x}$: Rate of change of the state variables.
- A: The system matrix.
- X: The state vector.
- B: The input matrix.
- U: The control input vector.
- D: The disturbance.
- Y: The output vector.
- C: The output matrix.
- Q_mat: The cost associated with our system variables.
- R_mat: The cost associated with our system inputs.
- K_mat: Gain matrix.

1. Introduction

Aim of the Thesis

The thesis aims to comprehensively investigate the dynamics, modeling, and control aspects of quadcopters, focusing on understanding their mathematical model and developing effective methods for stabilization and trajectory control. In addition to the PID controller part, the thesis will include an LQR controller part, with a plan to compare the performance of these two controllers.

Methodology

The dynamics and control of quadcopters are thoroughly examined as part of the research for this thesis. First, the differential equations regulating the behavior of quadcopters are derived using the Newton-Euler and Euler-Lagrange equations in a methodical manner. This basic simulation offers a strong foundation for understanding the dynamics of quadcopters. Next, the study concentrates on trajectory control and stabilization, using flight simulations to examine the behavior of the model. A Proportional-Derivative (PD) controller, renowned for its efficiency and simplicity, is used to accomplish stabilization. Furthermore, a heuristic approach to trajectory control is shown, incorporating a PD controller to mitigate the effect of stochastic external pressures. A linearization of the PID controller will be carried out to improve this methodology and enable a comparison with the Linear Quadratic Regulator (LQR).

2. Mathematical model of quadcopter

Figure 1 shows the quadcopter's structure along with the forces, torques, and angular velocities produced by its four rotors, which are numbered 1 through 4.

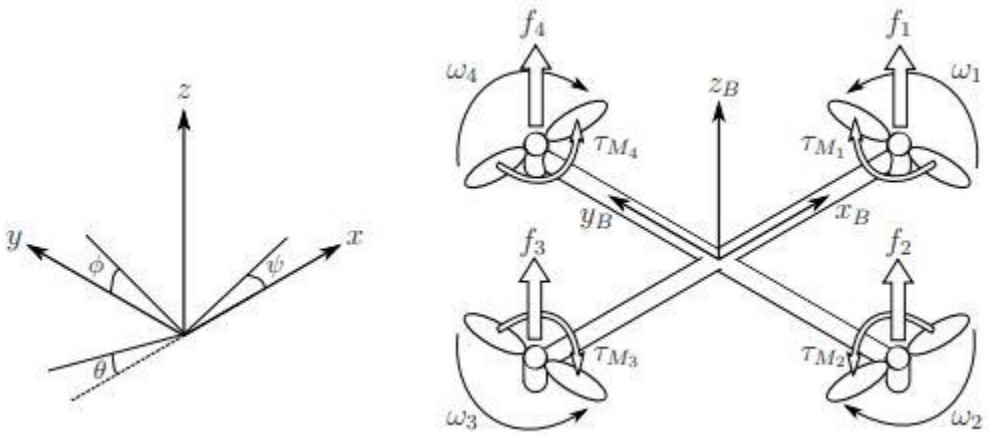

Figure 1: Inertial frame and body frame of a quadcopter

Using ξ, the quadcopter's absolute linear location is specified in the x, y, and z axes of the inertial frame. Three Euler angles, η, are used in the inertial frame to establish the attitude, or angular position. The rotation of the quadcopter around the y-axis is determined by its pitch angle, θ. The rotation around the x-axis is determined by roll angle ϕ and around the z-axis by yaw angle ψ. The linear and angular position vectors are contained in vector q.

$$\xi = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \eta = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} \qquad q = \begin{bmatrix} \xi \\ \eta \end{bmatrix} \qquad (1)$$

The quadcopter's center of mass is where the body frame originates. V_B and V determine the linear and angular velocities in the body frame, respectively.

$$V_B = \begin{bmatrix} Vx, B \\ Vy, B \\ Vz, B \end{bmatrix} \qquad V = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad (2)$$

Where p represents the velocity around φ, q around θ and r around ψ.

The rotation matrix $R_{\text{body to inertial}}$ that transforms coordinates from the body fixed frame to the inertial frame using Euler angles ψ, θ, and φ is given by:

Rotation about the z-axis (ψ):

$$\begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation about the y-axis (θ):

$$\begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix}$$

Rotation about the x-axis (φ):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & C\phi & -S\phi \\ 0 & S\phi & C\phi \end{bmatrix}$$

$$R = Rz(\psi)\cdot Ry(\theta)\cdot Rx(\varphi)$$

$$= \begin{bmatrix} C\psi C\theta & C\psi S\theta S\varphi - S\psi C\varphi & C\psi S\theta C\varphi + S\psi S\varphi \\ S\psi C\theta & S\psi S\theta S\varphi + C\psi C\varphi & S\psi S\theta C\varphi - C\psi S\varphi \\ -S\theta & C\theta S\varphi & C\theta C\varphi \end{bmatrix} \quad (3)$$

Here, Cψ and Sψ denote the cosine and sine of ψ, Cθ and Sθ denote the cosine and sine of θ, and Cφ and Sφ denote the cosine and sine of φ.

The rotation matrix R is orthogonal, implying that its inverse, denoted as R^{-1} , is equal to its transpose, R^T. This matrix facilitates the transformation from the body frame to the inertial frame.

For angular velocities, the transformation matrix from the inertial frame to the body frame is W_η and $W^{-1}{}_\eta$ from the body frame to the inertial frame.

$$\dot{\eta} = W^{-1}{}_\eta \, \nu$$

$$= \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S\varphi T\theta & C\varphi T\theta \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi/C\theta & C\varphi/C\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (4)$$

If we divide both sides on $W^{-1}{}_\eta$ we get:

$$\nu = W_\eta \, \dot{\eta}$$

$$= \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S\theta \\ 0 & C\varphi & C\theta S\varphi \\ 0 & -S\varphi & C\theta C\varphi \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (4)$$

The four arms of the quadcopter are supposed to be symmetrically arranged with respect to the x and y axes of the body. Therefore, diagonal matrix I, where Ixx = Iyy, is the inertia matrix.

$$I = \begin{bmatrix} Ixx & 0 & 0 \\ 0 & Iyy & 0 \\ 0 & 0 & Izz \end{bmatrix} \quad (5)$$

The rotation speed of rotor i, represented by ωi, generates a force fi aligned with the rotor axis. Additionally, the angular velocity and acceleration of the rotor contribute to the generation of torque τ_{Mi} around the rotor axis.

$$f_i = k\,\omega^2_i \qquad \tau_{Mi} = b\,\omega^2_i + I_M\,\dot{\omega}_i \quad (6)$$

where the rotor's inertia moment is IM, the lift constant is k, and the drag constant is b. Usually the effect of $\dot{\omega}_i$ is considered small and thus it is omitted.

Thrust T is produced in the body's Z axis direction by the combined forces of the rotors. The torques τ_ϕ, τ_θ and τ_ψ in the direction of the respective body frame angles make up torque τ_B.

$$T = \sum^4_{i=1} f_i = k \sum^4_{i=1} \omega^2_i \qquad T_B = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \quad (7)$$

$$\tau_B = \begin{bmatrix} \tau\varphi \\ \tau\theta \\ \tau\psi \end{bmatrix} \quad (8)$$

where $\tau\varphi = Lk\,(-\omega^2_2 + \omega^2_4)$ $\tau\theta = Lk\,(-\omega^2_1 + \omega^2_3)$ $\tau\psi = \sum^4_{i=1} \tau_{Mi}$

in which L is the distance between the rotor and the center of mass of the quadcopter. Thus, the roll movement is acquired by decreasing the 2nd rotor velocity and increasing the 4th rotor velocity. Similarly, the pitch movement is acquired by decreasing the 1st rotor velocity and increasing the 3rd rotor velocity. Yaw movement is acquired by increasing the angular velocities of two opposite rotors and decreasing the velocities of the other two.

2.1. Newton-Euler equations

Since the quadcopter is thought to be a rigid body, its dynamics may be explained using the Newton-Euler equations. The gravity R^TG and the total thrust of the rotors T_B are equal to the force needed for the acceleration of mass m $\dot{V}_B$ and the centrifugal force $v \times (mV_B)$ in the body frame.

$$m\dot{V}_B + v \times (mV_B) = R^TG + T_B. \quad (9)$$

The centrifugal force is zero in the inertial frame. Therefore, the only factors accelerating the quadcopter are gravity and the strength and direction of the thrust.

$$m\ddot{\xi} = G + RT_B$$

$$= \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + T/m \begin{bmatrix} C\psi S\theta C\varphi + S\psi S\varphi \\ S\psi S\theta C\varphi - C\psi S\varphi \\ C\theta C\varphi \end{bmatrix} \quad (10)$$

Within the body frame, the external torque τ is equal to the angular acceleration of inertia Iv˙, centripetal forces v × (Iv), and gyroscopic forces Γ.

$$Iv^{\cdot} + v \times (Iv) + \Gamma = \tau$$

$$= \quad v^{\cdot} = I^{-1}\left(- \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} Ixx\ p \\ Iyy\ q \\ Izz\ r \end{bmatrix} - I_r \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_\Gamma + \tau \right)$$

$$= \quad \begin{bmatrix} p^{\cdot} \\ q^{\cdot} \\ r^{\cdot} \end{bmatrix} = \begin{bmatrix} (Iyy - Izz)\ q\ r/Ixx \\ (Izz - Ixx)\ p\ r/Iyy \\ (Ixx - Iyy)\ p\ q/Izz \end{bmatrix} - I_r \begin{bmatrix} q/Ixx \\ -p/Iyy \\ 0 \end{bmatrix} \omega_\Gamma + \begin{bmatrix} \tau\varphi/Ixx \\ \tau\theta/Iyy \\ \tau\psi/Izz \end{bmatrix} \qquad (11)$$

where ωΓ is equal to ω1 − ω2 + ω3 − ω4.

The transformation matrix W^{-1} and its time derivative are then used to attract the angular accelerations in the inertial frame from the body frame accelerations.

$$\eta^{\cdot\cdot} = \quad d/dt\ (W^{-1}{}_\eta\ v) = \quad d/dt\ (W^{-1}{}_\eta)\ \ v + W^{-1}{}_\eta\ v^{\cdot}$$

$$\begin{bmatrix} 0 & \dot{\phi}C_\phi T_\theta + \dot{\theta}S_\phi/C_\theta^2 & -\dot{\phi}S_\phi C_\theta + \dot{\theta}C_\phi/C_\theta^2 \\ 0 & -\dot{\phi}S_\phi & -\dot{\phi}C_\phi \\ 0 & \dot{\phi}C_\phi/C_\theta + \dot{\phi}S_\phi T_\theta/C_\theta & -\dot{\phi}S_\phi/C_\theta + \dot{\theta}C_\phi T_\theta/C_\theta \end{bmatrix} \boldsymbol{\nu} + \boldsymbol{W}_\eta^{-1}\dot{\boldsymbol{\nu}}. \qquad (12)$$

2.2. Euler-Lagrange equations

The translational and rotational energies, Etrans and Erot minus the potential energy, Epot, equals the Lagrangian L.

$$L\ (q, q^{\cdot}) = Etrans + Erot - Epot$$

$$= \quad (m/2)\ \dot{\xi}^T \dot{\xi} + (1/2)\ v^T Iv - mgz \qquad (13)$$

The external forces and torques in the Euler-Lagrange equations are:

$$\begin{bmatrix} f \\ \tau \end{bmatrix} = d/dt\ (\partial L/\partial q^{\cdot}) - (\partial L/\partial q)\ . \qquad (14)$$

The linear and angular components do not depend on each other thus they can be studied separately. The linear external force is the total thrust of the rotors. The linear Euler-Lagrange equations are:

$$f = R\ T_B = m\xi^{\cdot\cdot} + mg \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad (15)$$

The Jacobian matrix J (η) from ν to $\dot{\eta}$ is:

$$J(\eta) = J = W_{\eta}^{T} I W_{\eta} =$$

$$\begin{bmatrix} I_{xx} & 0 & -I_{xx}S_{\theta} \\ 0 & I_{yy}C_{\phi}^{2} + I_{zz}S_{\phi}^{2} & (I_{yy} - I_{zz})\, C_{\phi}S_{\phi}C_{\theta} \\ -I_{xx}S_{\theta} & (I_{yy} - I_{zz})\, C_{\phi}S_{\phi}C_{\theta} & I_{xx}S_{\theta}^{2} + I_{yy}S_{\phi}^{2}C_{\theta}^{2} + I_{zz}C_{\phi}^{2}C_{\theta}^{2} \end{bmatrix}. \quad (16)$$

In this case the Jacobian matrix can be denoted as the identity matrix (I) for the internal frame.

Thus, the rotational energy Erot can be expressed in the inertial frame as:

$$E_{rot} = (1/2)\, \nu^{T} I \nu = (1/2)\, \ddot{\eta}^{T} J \ddot{\eta}. \quad (17)$$

The torques of the rotors make up the external angular force. Their Euler-Lagrange angular equations are:

$$\tau = \tau_{B} = J\ddot{\eta} + d/dt\,(J)\,\dot{\eta} - (1/2)\,(\partial/\partial\eta)\,(\dot{\eta}^{T} J \dot{\eta}) = J\ddot{\eta} + C(\eta, \dot{\eta})\,\dot{\eta}. \quad (18)$$

in which the matrix C (η, $\dot{\eta}$) is the Coriolis term, containing the gyroscopic and centripetal terms.

$$C(\eta, \dot{\eta}) = \begin{bmatrix} C11 & C12 & C13 \\ C21 & C22 & C23 \\ C31 & C32 & C33 \end{bmatrix} \quad (19)$$

$C11 = 0$

$C12 = (Iyy - Izz)(\dot{\theta} C_{x} S_{\phi} + \dot{\psi} S_{\phi}^{2} C_{\theta}) + (Izz - Iyy)(\dot{\psi} C_{\phi}^{2} C_{\theta} - Ixx\, \dot{\psi} C_{\theta})$

$C13 = (Izz - Iyy)\, \dot{\psi} C_{\phi} S_{\phi} C_{\theta}^{2}$

$C21 = (Izz - Iyy)(\dot{\theta} C_{x} S_{\phi} + \dot{\psi} S_{\phi} C\theta) + (Iyy - Izz)(\dot{\psi} C_{\phi}^{2} C_{\theta} + Ixx\, \dot{\psi} C_{\theta})$

$C22 = (Izz - Iyy)\dot{\phi} C_{\phi} S_{\phi}$

$C23 = -Ixx\, \dot{\psi} S_{\theta} C_{\theta} + Iyy\, \dot{\psi} S_{\phi}^{2} S_{\theta} C_{\theta} + Izz\, \dot{\psi} C_{\phi}^{2} S_{\theta} C_{\theta}$

$C31 = (Iyy - Izz)\, \dot{\psi} C_{\theta}^{2} S_{\phi} C_{\phi} - Ixx\, \dot{\theta} C_{\theta}$

$C32 = (Izz - Iyy)(\dot{\theta} C_{x} S_{\phi} S_{\theta} + \dot{\phi} S_{\phi}^{2} C_{\theta}) + (Iyy - Izz)\, \dot{\phi} C_{\phi}^{2} C_{\theta} + Ixx\, \dot{\psi} S_{\theta} C_{\theta} - Iyy\, \dot{\psi} S_{\phi}^{2} S_{\theta} C_{\theta} - Izz\, \dot{\psi} C_{\phi}^{2} S_{\theta} C_{\theta}.$

$C33 = (Iyy - Izz)\dot{\phi} C_{\phi} S_{\phi} C_{\theta}^{2} - Iyy\, \dot{\theta} S_{\phi}^{2} S_{\theta} C_{\theta} - Izz\, \dot{\theta} C_{\phi}^{2} S_{\theta} C_{\theta} + Ixx\, \dot{\theta} C_{\theta} S_{\theta}$

Equation (18) leads to the differential equations for the angular accelerations which are equivalent with Equations (11) and (12):

$$\ddot{\eta} = J^{-1} (\tau_{B} - C(\eta, \dot{\eta})\, \dot{\eta}). \quad (20)$$

2.3. Aerodynamical effects

Complex dynamic interactions are simplified in the model that before. The drag force produced by the air resistance is incorporated to drive the quadcopter to behave more realistically. Equations (10) and (15) are utilized in this construction, and the diagonal coefficient matrix links the linear velocities to the force causing the motion to slow down.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + T/m \begin{bmatrix} C\psi S\theta C\varphi + S\psi S\varphi \\ S\psi S\theta C\varphi - C\psi S\varphi \\ C\theta C\varphi \end{bmatrix} - 1/m \begin{bmatrix} Ax & 0 & 0 \\ 0 & Ay & 0 \\ 0 & 0 & Az \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \quad (21)$$

The drag force coefficients for velocities in the corresponding directions of the inertial frame are represented by the letters Ax, Ay, and Az.

One might incorporate multiple additional aerodynamic factors into the model. For instance, research on the effects of airflow disturbances, blade flapping, and angle of attack on thrust. Aerodynamical impacts have a complex influence and are challenging to model. Additionally, some effects are only significantly affected at high velocities. As a result, the stated straightforward model is applied, and these impacts are removed from the model.

3. PID Simulation

Using the Matlab computer language, the quadcopter's mathematical model is constructed for simulation in Matlab 2010. The table below displays the parameter values that are utilized in the simulations. The quadcopter will slow down and come to a halt when angles ϕ and θ stabilize at zero values, which is why the values of the drag force coefficients Ax, Ay, and Az were chosen.

Parameter	Value	Unit
g	9.81	$\mathrm{m/s^2}$
m	0.468	kg
l	0.225	m
k	$2.980 \cdot 10^{-6}$	
b	$1.140 \cdot 10^{-7}$	
I_M	$3.357 \cdot 10^{-5}$	$\mathrm{kg\ m^2}$

Parameter	Value	Unit
I_{xx}	$4.856 \cdot 10^{-3}$	$\mathrm{kg\ m^2}$
I_{yy}	$4.856 \cdot 10^{-3}$	$\mathrm{kg\ m^2}$
I_{zz}	$8.801 \cdot 10^{-3}$	$\mathrm{kg\ m^2}$
A_x	0.25	kg/s
A_y	0.25	kg/s
A_z	0.25	kg/s

Table 1: Parameter values for simulation

Using the following example instance, a quadcopter is simulated to test the mathematical model. At first, the quadcopter is in a stable condition where all position and angle values are zero and its body frame is in line with the inertial frame. The thrust equals gravity and the total thrust equals the hover thrust. The simulation runs at intervals of 0.0001 seconds, resulting in a two-second total elapsed duration.

The quadcopter used the hover thrust to increase all of its rotor velocities, which allowed it to rise for the first 0.25 seconds. After that, the ascent is halted by sharply lowering the rotor velocities for the next 0.25 seconds. As a result, in the first 0.5 seconds, the quadcopter rose 0.1 meters. Following the ascent, the quadcopter regains stability.

After then, the quadcopter is forced into a roll motion by accelerating the fourth rotor and decelerating the second rotor for a quarter of a second. By reducing the fourth rotor's velocity and raising the second rotor's velocity for 0.25 seconds, the acceleration of the roll motion is stopped. Based on this, the roll angle ϕ increased by approximately 25 degrees after 0.5 seconds of roll action. The quadcopter accelerated in the direction of the negative y-axis because to the roll angle.

Next, by raising the third rotor's velocity and lowering the first, a pitch motion is produced, much like the roll motion. By raising the first rotor's velocity and decreasing the third rotor's velocity, the motion is stopped. Pitch movement caused the pitch angle θ to increase by about 22 degrees. Pitch angle is what drives the quadcopter's acceleration in the direction of the positive x-axis.

Ultimately, the first and third rotor velocities are increased while the second and fourth rotor velocities are decreased to turn the quadcopter in the direction of the yaw angle ψ. By raising the velocities of the second and fourth rotors and decreasing the velocities of the first and third rotors, the yaw motion is stopped. As a result, the yaw angle (ψ) rises by around 10 degrees.

The rotors' total thrust stayed relatively close to the starting total thrust throughout the whole simulation. Consequently, the thrust value in the z-axis direction is reduced by the roll and pitch angle deviations from zero. As a result, the quadcopter descends and accelerates along the negative z-axis.

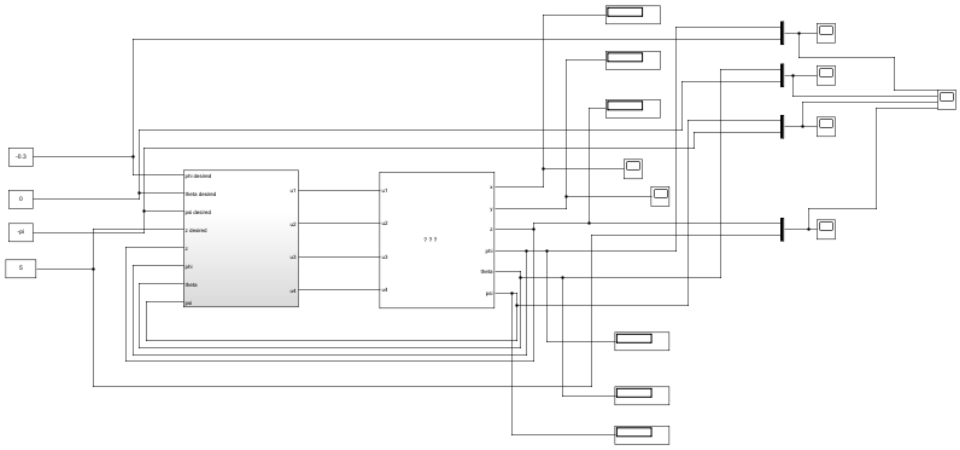

Figure 2: Quadcopter model using PID

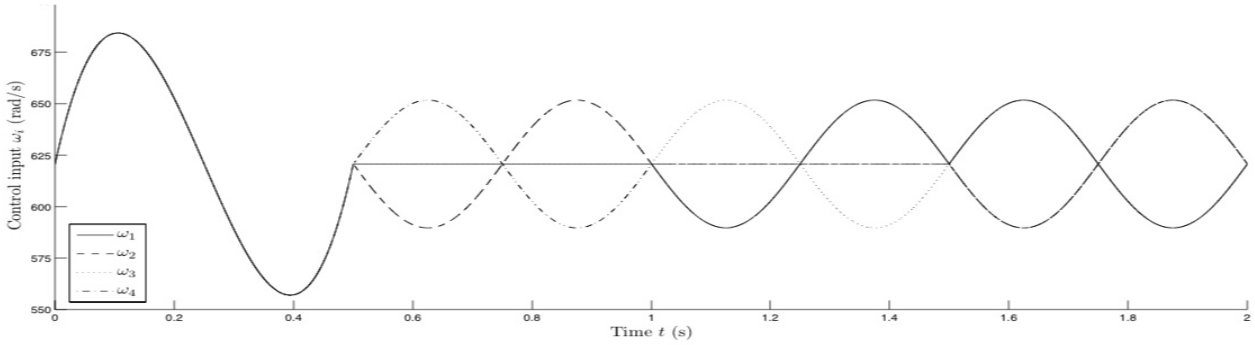

Figure 3: Control inputs ωi

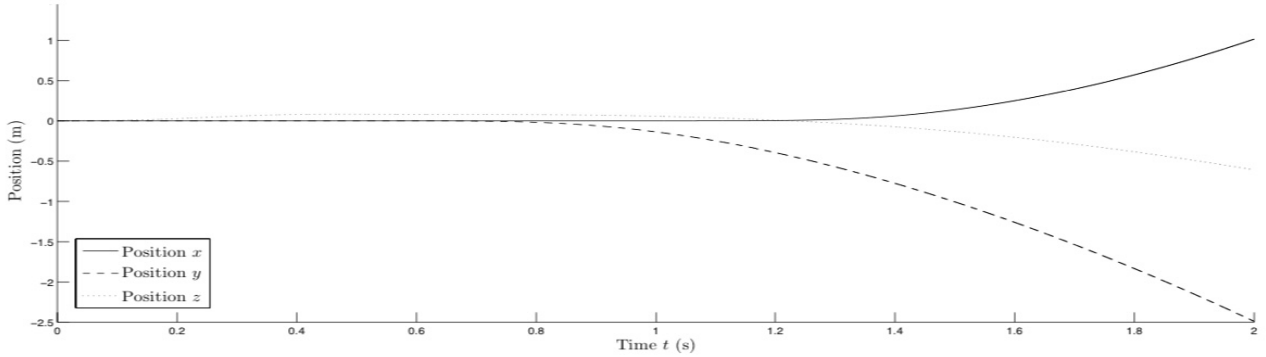

Figure 4: Positions x, y, and z

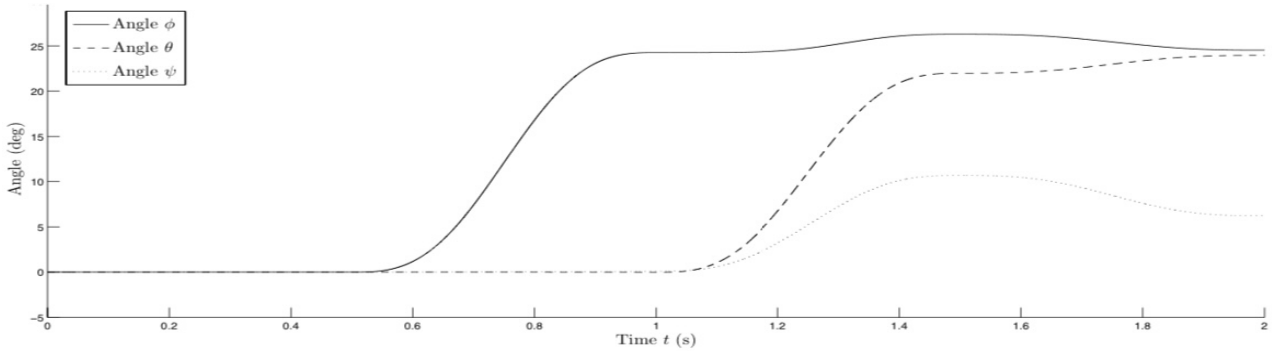

Figure 5: Angles ϕ , θ, and ψ

4. Stabilization of quadcopter

A PID controller is used to keep the quadcopter stabilized. The straightforward design and straightforward implementation of the PID controller is its advantages. The PID controller's standard configuration is:

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

$$u(t) = K_P\, e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D\, de(t)/dt \qquad (22)$$

where K_P, K_I, and K_D are the parameters for the proportional, integral, and derivative parts of the PID controller, and u(t) is the control input. e(t) represents the difference between the desired state, xd(t), and the current state, x(t).

A quadcopter has four control inputs (the angular velocities of the four rotors, ωi), but only six states (positions ξ and angles η). Equations (10), (11), and (12) define the quadcopter dynamics, which show the interactions between the states and the overall thrust T and the torques τ produced by the rotors. The quadcopter is held aloft by the overall thrust T, which also influences acceleration along the z-axis. Angle ϕ's acceleration is influenced by torque τ_ϕ, angle θ's acceleration is influenced by torque τ_θ, and angle ψ's acceleration is aided by torque τ_ψ.

As a result, the quadcopter's PD controller is selected as:

$$T = (g + K_{z,D}(\dot{z}_d - \dot{z}) + K_{z,P}(z_d - z))\, m/C_\phi C_\theta$$

$$\tau_\phi = (K_{\phi,D}(\dot{\phi}_d - \dot{\phi}) + K_{\phi,P}(\phi_d - \phi))\, Ixx$$

$$\tau_\theta = (K_{\theta,D}(\dot{\theta}_d - \dot{\theta}) + K_{\theta,P}(\theta_d - \theta))\, Iyy \qquad (23)$$

$$\tau_\psi = (K_{\psi,D}(\dot{\psi}_d - \dot{\psi}) + K_{\psi,P}(\psi_d - \psi))\, Izz$$

where the quadcopter's mass m, moments of inertia I, and gravity, g, are also taken into account.

Equations (7) and (8) can be used to obtain the correct angular velocities of the rotors ωi using data from Equation (23).

$$\omega^2_1 = (T/4k) - (\tau_\theta/2kl) - (\tau_\psi/4b)$$

$$\omega^2_2 = (T/4k) - (\tau_\phi/2kl) + (\tau_\psi/4b)$$

$$\omega^2_3 = (T/4k) + (\tau_\theta/2kl) - (\tau_\psi/4b) \qquad (24)$$

$$\omega^2_4 = (T/4k) + (\tau_\phi/2kl) + (\tau_\psi/4b)$$

The stabilization of a quadcopter is used to assess the PD controller's functionality. The table below displays the parameters of the PD controller. Position $\xi = [0\ 0\ 1]^T$ in meters and angles $\eta = [10\ 10\ 10]^T$ in degrees are the quadcopter's beginning conditions. $Z_d = 0$ is the target altitude position. Since steady hovering is the goal of the stabilization, $\eta_d = [0\ 0\ 0]^T$.

Parameter	Value
$K_{z,D}$	2.5
$K_{\phi,D}$	1.75
$K_{\theta,D}$	1.75
$K_{\psi,D}$	1.75

Parameter	Value
$K_{z,P}$	1.5
$K_{\phi,P}$	6
$K_{\theta,P}$	6
$K_{\psi,P}$	6

Table 2: Parameters of the PD controller

After five seconds, the angles and altitude are stabilized to zero. However, because the angles' values were non-zero, the positions x and y differed from the zero values. The quadcopter has travelled more than one meter in the positive x direction and 0.5 meters in the negative y direction before it is stabilized to hover. This is a result of the PD controller's control technique failing to take the accelerations in the x and y axes into account. As a result, a different control mechanism needs to be developed to provide control over all of the quadcopter's locations and angles.

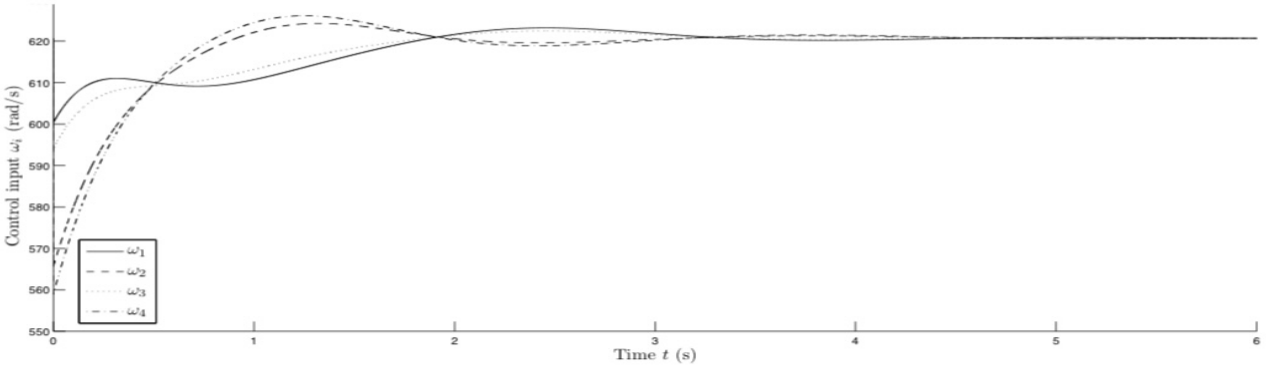

Figure 6: Control inputs ωi

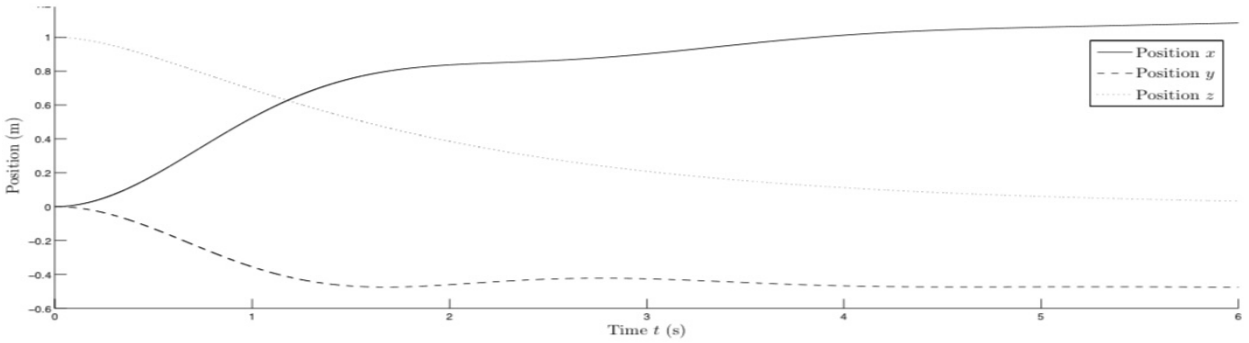

Figure 7: Positions x, y, and z

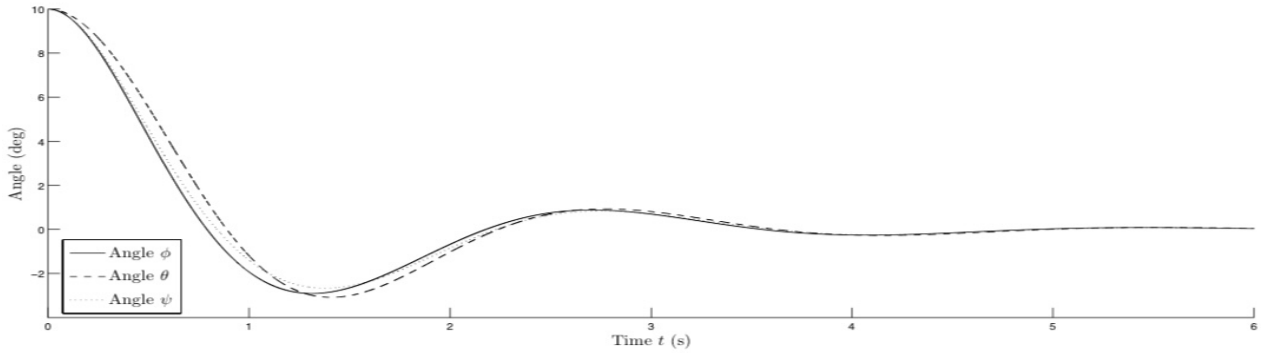

Figure 8: Angles ϕ, θ, and ψ

5. Trajectory control

Trajectory control is used to manipulate the quadcopter's rotor velocities in order to move it from its initial location to the intended destination. Due to its complicated dynamics, determining a quadcopter's ideal trajectory is a challenging undertaking. Still, the quadcopter can be adequately controlled with a straightforward control scheme. Thus, our study and development focus on a heuristic method.

The study of the relationships and dependencies between states, state derivatives, and control inputs forms the foundation for the creation of a control mechanism. Equations 7, 8, 20, and 21 define these dependencies and interactions.

The torques τϕ, τθ and τψ are defined by the supplied control inputs ωi, along with the overall thrust T. Depending on the current angles and angular velocities, the torques have an impact on the angular accelerations. The angular velocities $\dot{\eta}$, which are integrated from the angular accelerations $\ddot{\eta}$, can be used to integrate the angles η. The linear velocities $\dot{\xi}$, the angles η, and the total thrust Tall affect the linear accelerations $\ddot{\xi}$. From the linear accelerations $\ddot{\xi}$ through the linear velocities $\dot{\xi}$, the linear position ξ is integrated.

For given states ξ, therefore, this line of reasoning has to be carried out in reverse order to identify appropriate control inputs ωi.

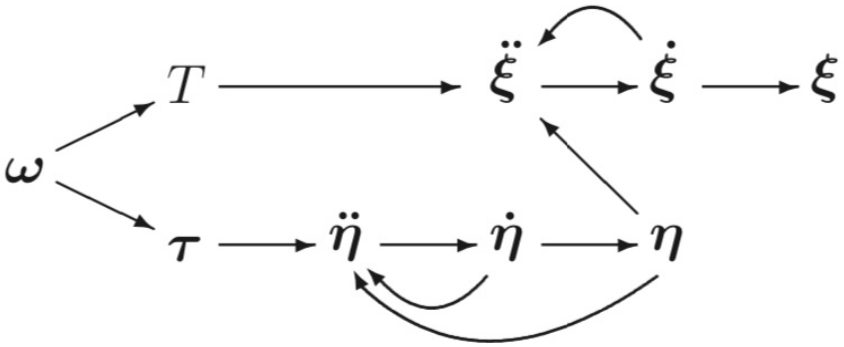

Figure 9: Interactions between states, state derivatives, and control inputs

One way to achieve the desired trajectory according to positions x, y, and z for each time T is to generate linear accelerations. Three equations are obtained from equation (21).

$$T_B = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} = R^T \left(m \left(\ddot{\xi} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right) + \begin{bmatrix} Ax & 0 & 0 \\ 0 & Ay & 0 \\ 0 & 0 & Az \end{bmatrix} \dot{\xi} \right) \qquad (25)$$

where the unknown variables to be addressed are the total thrust T, angles Ά and θ, and desired trajectory values ξ, ξ, and ψ.

This equation can be used to compute the total thrust T for each time t, as well as the necessary angles ϕ and θ.

$$\phi = \arcsin\left((d_x S_\psi - d_y C_\psi) / (d^2_x + d^2_y + (d_z + g)^2) \right)$$

$$\theta = \arctan\left((d_x C_\psi + d_y S_\psi) / (d_z + g) \right) \qquad (26)$$

$$T = m \left(d_x (S_\theta C_\psi C_\phi + S_\psi S_\phi) + d_y (S_\theta S_\psi C_\phi - C_\psi S_\phi) + (d_z + g) C_\theta C_\phi \right)$$

in which

$$d_x = \ddot{x} + (A_x \dot{x}/m)$$

$$d_y = \ddot{y} + (A_y \dot{y}/m) \qquad (27)$$

$$d_z = \ddot{z} + (A_z \dot{z}/m)$$

Simple deduction can be used to compute the angular velocities and accelerations from the known values of the angles ϕ and θ. Equation (20) can be used to solve for the torques τ using the angular velocities and accelerations. Equation (24), when the torques and thrust are known, can be used to compute the control inputs ωi.

5.1. Heuristic method for trajectory generation

It is challenging to generate appropriate accelerations $\dddot{\xi}$ because the third and fourth derivatives of the position, jerk and jounce, must have a decent composition. The composition of the control inputs ωi shows how the jounce values have an impact. The jounces must be carefully taken into account while generating the accelerations since high jounce values correspond to high control input values.

To produce jounce values, a heuristic approach might be applied. To regulate the derivatives, the approach makes use of a symmetric structure in the jounce function f (t). Three sine functions define one influential element of the function as follows:

$$f(t)=\begin{cases} a\sin\left(\left(\frac{1}{b}\right)\pi t\right) & 0 \le t \le b \\ -a\sin\left(\left(\frac{1}{b}\right)\pi t-\pi\right) & b \le t \le 3b \\ a\sin\left(\left(\frac{1}{b}\right)\pi t-3\pi\right) & 3b \le t \le 4b \end{cases} \quad (28)$$

Figure 10 illustrates the function's structure. A smooth function is produced by using the sine functions. These three sine functions combine to create a function whose second half reduces acceleration down to zero after raising it to a predetermined value in the first half. Constant velocity is produced by this acceleration. Reversing the velocity to zero can be accomplished by using the function's mirror image. The final location is determined by the duration c between the jounce's accelerating and decelerating parts, which are mirror images of each other, and the parameters a and b of the sine functions, which are given in Equation (28).

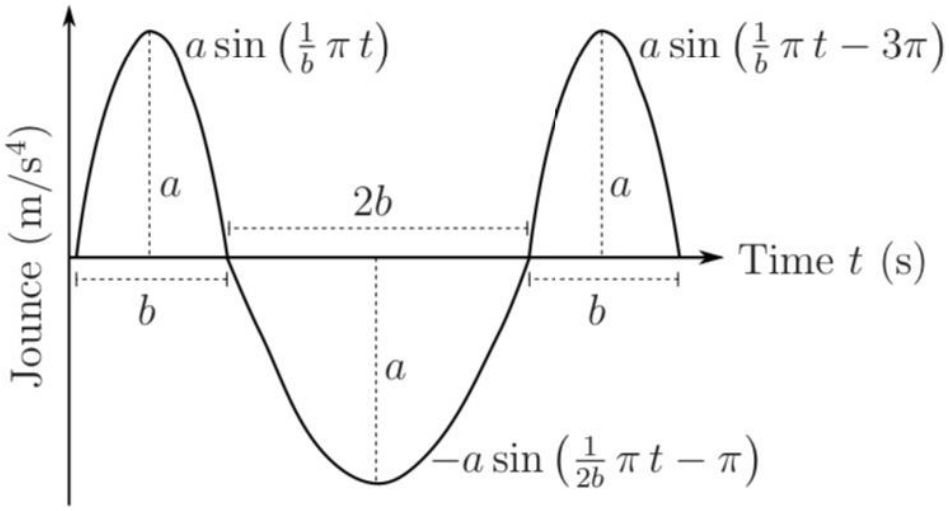

Figure 10: Heuristic method for the generation of jounce functions

Regretfully, the approach fails to provide ideal paths. Therefore, to determine the quadcopter's ideal trajectory, an appropriate algorithm and a dynamic optimization model would be required. Nonetheless, the suggested procedure is simple to apply in order to produce appropriate jounce values that will result in the desired trajectory.

Using a sample simulation, the method's functionality is examined. Equation (28) is used to construct the jounce of position x, with the values a = 1, b = 0.5, and c = 2. Figure 11 displays the position x and its derivatives as they relate to the proposed jounce. It is also possible for y and z to generate the jounce concurrently. But in this case, just position x is taken into account since it is easier to see how the angle θ is controlled by the relationship between the jounce of position x and the control inputs ω1 and ω3.

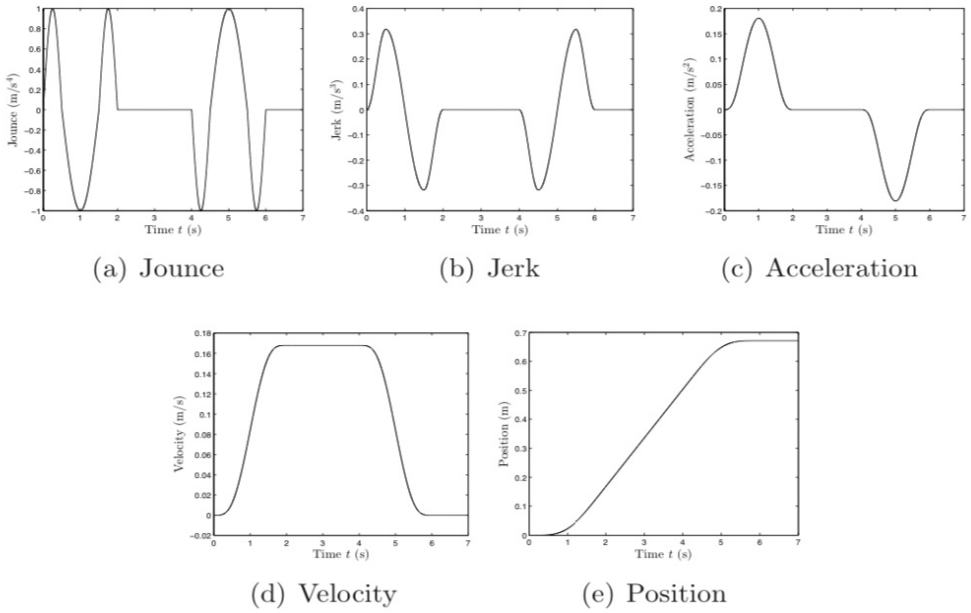

Figure 11: Planned position x and its derivatives with given jounce

The example case simulation is run using the position x accelerations and velocities that were previously provided. For any time, t, the planned value of angle ψ is zero. First, for each time t, the necessary angles ϕ and θ and thrust T are solved using Equation (26). From the solved angles with derivation, the angular velocities and accelerations are computed. After that, the angular velocities and accelerations are used to solve the torques. The control inputs are finally resolved. After that, the simulation is run using the supplied control inputs.

Figure 12 displays the control inputs that have been calculated. Figure 13 displays the simulated positions ξ, whereas Figure 14 displays the simulated angles η. Figure 13 shows that the values of the positions y and z remain zero and that the simulated position x is the same as the planned position x in Figure 11(e). In order to generate the constant acceleration needed to offset the drag force brought on by the intended constant velocity, the angle θ increases throughout the acceleration and then stabilizes to a constant value. In order to bring the quadcopter to a stop, the angle is finally adjusted in the opposite way.

The anticipated jounce in Figure 11(a) and the estimated control inputs ω1 and ω3 in Figure 12 have a similar form. The projected acceleration $\ddot{x}$ and the simulated angle θ have the same form. Because the drag force brought on by the velocity needs to be adjusted for, the shapes of the control inputs and the angles deviate from the planned values of the jounce and the acceleration.

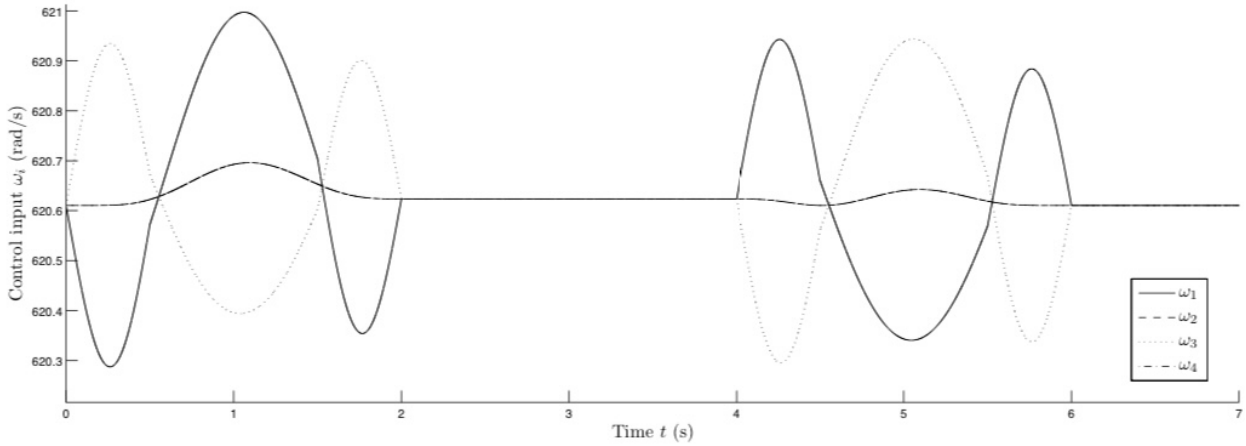

Figure 12: Control inputs ωi

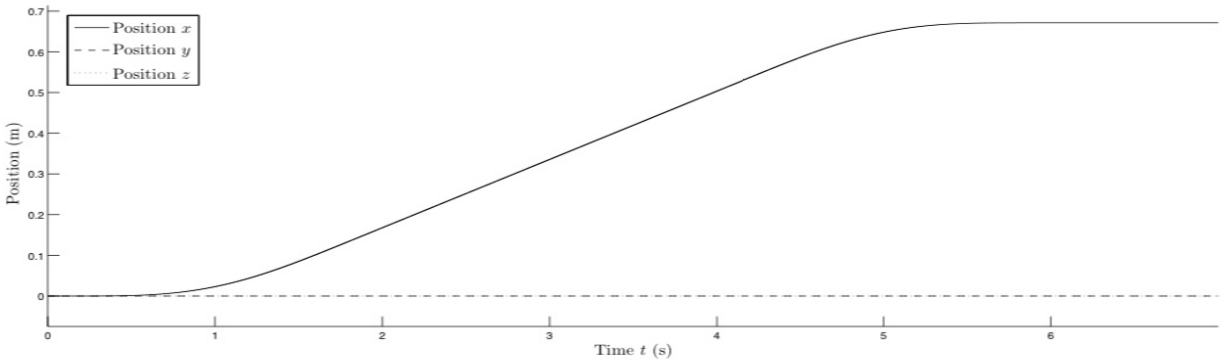

Figure 13: Positions x, y, and z

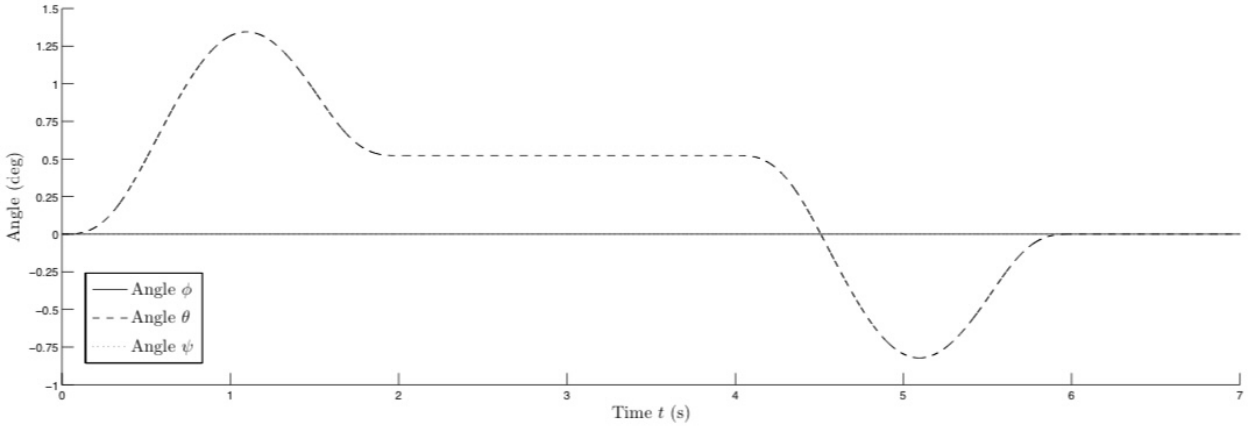

Figure 14: Angles ϕ, θ, and ψ

The technique can also be applied in the presence of un modeled linear forces, like as wind, which have an impact on the quadcopter's position via affecting its linear accelerations. It is feasible to rectify the trajectory with a new computation from the present, but erroneous, location to the next checkpoint if the trajectory is calculated over shorter distances. A demonstration of this technique is shown in Figure 15. The solid arrows show the achieved trajectory, whereas the arrows with dashed lines show the anticipated trajectory in the x y plane. The start and finish places are shown by black squares, and the trajectory's checkpoints are indicated by white squares.

The quadcopter's current location is used to determine the control inputs to the next checkpoint; however, due to unpredictable and un modeled forces, the actual position—shown by X—differs from the intended position. The target checkpoint is moved to the next one and fresh control inputs are computed if the quadcopter gets close enough to it. The quadcopter arrives at its target after traveling through each checkpoint and repeating this process.

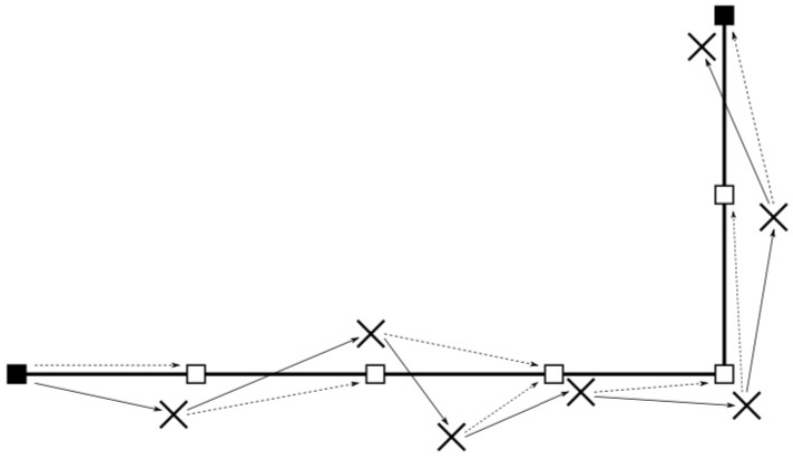

Figure 15: Example of checkpoint flight pattern with external disturbances

The main flaw in the suggested approach is that it only functions as intended when the quadcopter takes off from a stable attitude, the angles ϕ and θ, as well as their derivatives, are zero, and the attitude is not affected by outside forces while in flight. Minor variations in the angles might lead to a significant divergence in the trajectory. A potential solution to this issue is to use the previously suggested PD controller to stabilize the quadcopter at each checkpoint, or alternatively, use the heuristic method to angles. The effect of interim stabilization is transient, though, if the angular disturbances are constant.

5.2. Integrated PD controller

Including a PD controller in the heuristic approach provides an additional way to account for potential deviations in the angles. The PD controller determines the necessary values (dx, dy, and dz) in Equation (26) by taking into account the differences between the present and desired values (subscript d) of the locations (ξ), velocities ($\dot{\xi}$), and accelerations ($\ddot{\xi}$).

$$d_x = K_{x,P}\,(x_d - x) + K_{x,D}\,(\dot{x}_d - \dot{x}) + K_{x,DD}\,(\ddot{x}_d - \ddot{x})$$

$$d_y = K_{y,P}\,(y_d - y) + K_{y,D}\,(\dot{y}_d - \dot{y}) + K_{y,DD}\,(\ddot{y}_d - \ddot{y}) \qquad (29)$$

$$d_z = K_{z,P}\,(z_d - z) + K_{z,D}\,(\dot{z}_d - \dot{z}) + K_{z,DD}\,(\ddot{z}_d - \ddot{z})$$

Equation (26) provides the commanded angles ϕ_c and θ_c as well as the thrust T. The PD controller in Equation (30) regulates the torques τ, just as it does the locations ξ, velocities $\dot{\xi}$, and accelerations $\ddot{\xi}$, with the intended values (subscript d) in mind. Formula (23). Equation (24) can be used to solve the control inputs using the calculated thrust and torques.

$$\tau_\phi = (K_{\phi,P}\,(\phi_c - \phi) + K_{\phi,D}\,(\dot{\phi}_c - \dot{\phi}))\,I_{xx}$$

$$\tau_\theta = (K_{\theta,P}\,(\theta_c - \theta) + K_{\theta,D}\,(\dot{\theta}_c - \dot{\theta}))\,I_{yy} \qquad (30)$$

$$\tau_\psi = (K_{\psi,P}\,(\psi_c - \psi) + K_{\psi,D}\,(\dot{\psi}_c - \dot{\psi}))\,I_{zz}$$

An example case that shows the PD controller's performance has all of the positions x, y, and z, as well as their derivatives, having the same values as shown in Figure 11. The PD parameters listed in Table 3 are used to run the simulation.

Variable	Parameter value		
i	$K_{i,P}$	$K_{i,D}$	$K_{i,DD}$
x	1.85	0.75	1.00
y	8.55	0.75	1.00
z	1.85	0.75	1.00
ϕ	3.00	0.75	-
θ	3.00	0.75	-
ψ	3.00	0.75	-

Table 3: Parameters of the PD controller

Figures 16 - 18 show the simulation's outcomes. Figure 16 displays the simulated control inputs; Figure 17 shows the simulated locations; and Figure 18 shows the simulated angles. After six seconds, the quadcopter's location is quite near to its intended position, but it continues to fluctuate for a few seconds. Throughout the simulation, the angles vary significantly to provide the desired locations, velocities, and accelerations. During the acceleration, the control input values fluctuated, but after that, their behavior stabilized.

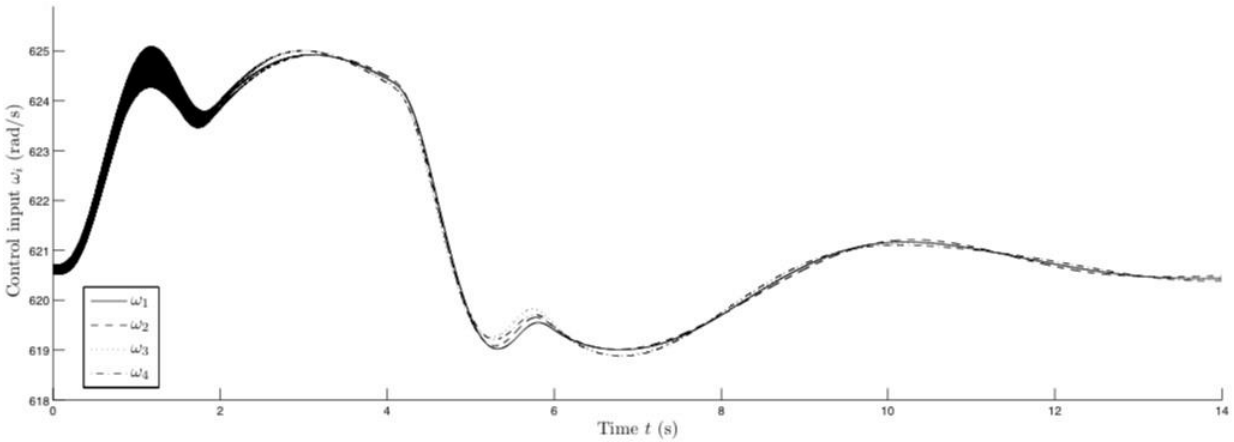

Figure 16: Control inputs ωi

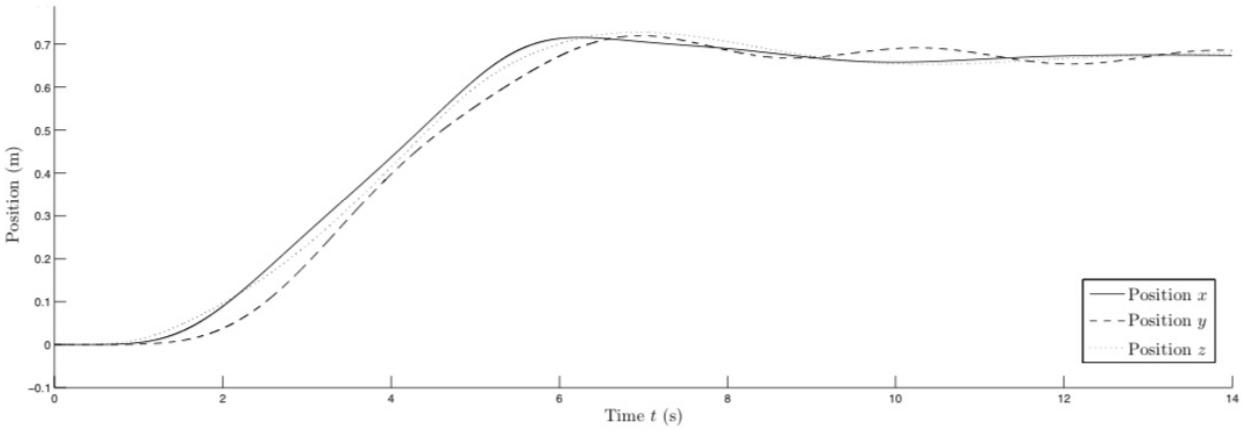

Figure 17: Positions x, y, and z

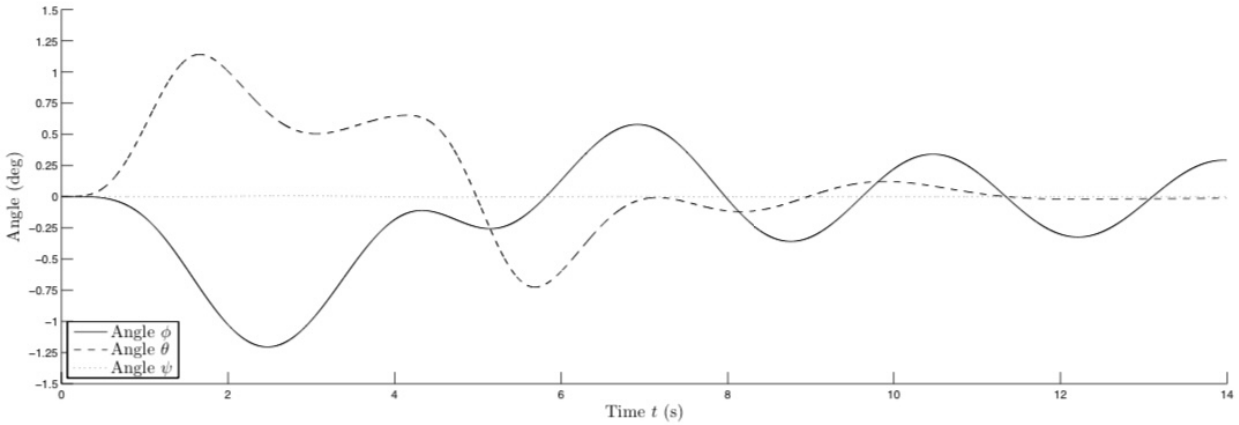

Figure 18: Angles ϕ, θ, and ψ

In the sample scenario, the suggested integrated PD controller operated effectively. Nonetheless, the parameter settings have a significant impact on the controller's performance. The controller won't react fast enough to follow the intended trajectory if the parameter values are tiny. The use of equations taking into account the torques and the control inputs requires a method to calculate the best feasible torques and from them the best control inputs. If the parameter values are significant, the quadcopter cannot perform the required drastic changes in the angular velocities of the rotor and the control inputs, and it may not be feasible with certain torques. Although it is quite challenging, another approach that might be used is to vary the PD parameters in accordance with the current locations, angles, and their derivatives.

6.LQR

For the best control of linear systems, the Linear Quadratic Regulator (LQR) is a potent control method that is extensively utilized in robotics and engineering. The process involves integrating the ideas of linear systems theory and optimal control to create controllers that minimize a quadratic cost function. Based on a linear model of the system dynamics and a cost function that penalizes departures from intended states and control attempts, the LQR controller calculates the best control inputs. This methodology produces controllers that are resilient, effective, and able to manage intricate systems with numerous inputs and outputs. Applying LQR techniques to quadcopter control can result in better overall performance, more accurate trajectory tracking, and more stability when compared to more conventional control strategies like PID controllers.

6.1. LQR Linearization

To transition from the Proportional-Derivative (PID) control method to the Linear Quadratic Regulator (LQR) technique, we employ MATLAB's LQR function. This function provides us with the LQR gain necessary for stabilizing the quadcopter system.

K= lqr (A,B,Q,R)

Based on the linearization equations:

$\dot{x} = Ax + Bu + Du$

Y= Cx + Du

Where:

- $\dot{x}$ is the rate of change of the state variables over time
- A is the system matrix
- x is the state vector representing the quadcopter's state variables, such as position, velocity, and orientation.
- B is the input matrix
- u is the control input vector, which includes PID control signals for each degree of freedom.
- D is the disturbance, accounting for external forces or uncertainties affecting the system.
- Y is the output vector
- C is the output matrix.

However, before applying LQR, we must obtain the linearized A and B matrices, which serve as input matrices for our LQR Control algorithm. These matrices are derived from the linearized model of the quadcopter's dynamics, capturing the system's behavior around an operating point. The goal of using LQR is to achieve optimized control with minimum cost, where the cost function represents the discrepancy between the desired system state and the current state based on the input provided by the controller. This transition allows us to explore the benefits of optimal control strategies in enhancing quadcopter stabilization and trajectory tracking performance.

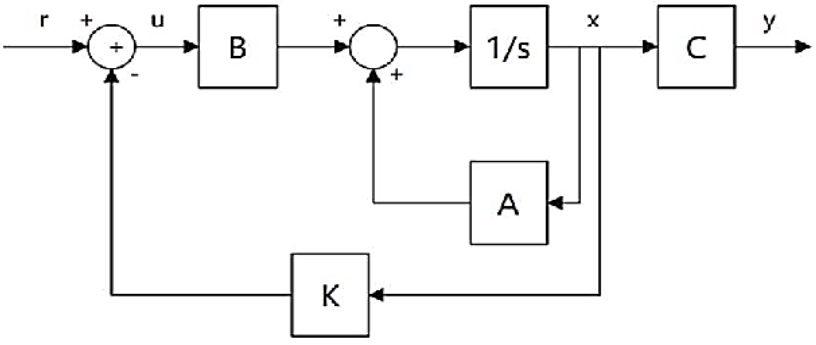

Figure 19: LQR system overlook

The quadcopter's dynamics are captured by the system matrix A in our state-space description, which focuses on the equations controlling angular velocity and angular position. These variables are important because they reflect the quadcopter's orientation dynamics and rotational motion, which have a big impact on how it behaves and controls itself.

In state space equations we only consider first differential equations:

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S\varphi T\theta & C\varphi T\theta \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi/C\theta & C\varphi/C\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \text{ and } \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + T/m \begin{bmatrix} C\psi S\theta C\varphi + S\psi S\varphi \\ S\psi S\theta C\varphi - C\psi S\varphi \\ C\theta C\varphi \end{bmatrix}$$

We simplify the system matrix in our study by applying small angle approximations utilizing linearized equations. By concentrating on the quadcopter system's linearized dynamics, our method enables us to produce a more understandable matrix representation. Our goal is to reduce complexity and capture the basic dynamics of the system by utilizing small angle approximations. This will make the process of analysis and control design simpler. In this case matrix A will be:

X'
ϕ'
θ'
ψ'
ϕ''
θ''
ψ''
x''
y''
z''
x'
y'
z'

ϕ'	θ'	ψ'	ϕ''	θ''	ψ''	x''	y''	z''	x'	y'	z'	X
0	0	0	0	0	0	1	0	0	0	0	0	ϕ
0	0	0	0	0	0	0	1	0	0	0	0	θ
0	0	0	0	0	0	0	0	1	0	0	0	ψ
0	0	0	0	0	0	0	0	0	1	0	0	ϕ'
0	0	0	0	0	0	0	0	0	0	1	0	θ'
0	0	0	0	0	0	0	0	0	0	0	1	ψ'
0	0	0	0	0	0	0	0	0	0	0	0	x'
0	0	0	0	0	0	0	0	0	0	0	0	y'
0	0	0	0	0	0	0	0	0	0	0	0	z'
0	-g	0	0	0	0	1	0	0	0	0	0	x
g	0	0	0	0	0	0	1	0	0	0	0	y
0	0	0	0	0	0	0	0	1	0	0	0	z

This matrix reflects the dynamic nature of the quadcopter system's mobility by showing how angular and linear velocity directly affect it. In particular, variations in the angular and linear velocities directly affect the behavior of the system, emphasizing their important function in regulating the quadcopter's motion.

Additionally, the matrix shows that pitch and roll dynamics are affected by gravity since these motions require the quadcopter to change its orientation in relation to gravity. But yaw rotation, which revolves around a vertical axis, is independent of gravity. This difference in the way that different rotating axes are affected by gravitational forces offers important new information for comprehending and simulating the behavior of the quadcopter in flight.

Moving to matrix B:

X’	U1	U2	U3	U4
ϕ'	0	0	0	0
θ'	0	0	0	0
ψ'	0	0	0	0
ϕ''	0	0	0	0
θ''	0	0	0	0
ψ''	1/m	0	0	0
x''	0	0	0	0
y''	0	0	0	0
z''	1/m	0	0	0
x'	0	1/I_xx	0	0
y'	0	0	1/I_yy	0
z'	1/m	0	0	1/I_zz

Here U is given as:

- U1: Total upward force on the quadrotor along Z axis (T-mg)
- U2: Roll torque (about X axis)
- U3: Pitch torque (about Y axis)
- U4: Yaw torque (about Z axis)

This emphasizes how crucial it is to counteract rotating torques in order to preserve stability and avoid positional shifts along their respective axes. To guarantee steady rotation without deviating from their intended positions along their axes, the quadcopter's rotational torques must overcome inertia.

Furthermore, the control input U1 is essential for counteracting the mass of the quadcopter, especially when overcoming gravity to achieve upward movement. This means that in order to control the quadcopter's dynamics and maintain stable flying performance, a careful balancing of control inputs is required.

Moving to matrix C:

ϕ'	θ'	ψ'	ϕ''	θ''	ψ''	x''	y''	z''	x'	y'	z'	X
1	0	0	0	0	0	0	0	0	0	0	0	ϕ
0	1	0	0	0	0	0	0	0	0	0	0	θ
0	0	1	0	0	0	0	0	0	0	0	0	ψ
0	0	0	1	0	0	0	0	0	0	0	0	ϕ'
0	0	0	0	1	0	0	0	0	0	0	0	θ'
0	0	0	0	0	1	0	0	0	0	0	0	ψ'
0	0	0	0	0	0	1	0	0	0	0	0	x'
0	0	0	0	0	0	0	1	0	0	0	0	y'
0	0	0	0	0	0	0	0	1	0	0	0	z'
0	0	0	0	0	0	0	0	0	1	0	0	x
0	0	0	0	0	0	0	0	0	0	1	0	y
0	0	0	0	0	0	0	0	0	0	0	1	z

We have chosen to incorporate the identity matrix into our method's system representation. We can directly observe every output that our quadcopter system produces thanks to this decision. We hope to preserve an accurate and complete picture of the system's outputs by using the identity matrix, which will enable careful analysis and efficient control tactics.

For this model as we will not consider any additional disturbances so we shall take D matrix as 0.

6.2. LQR Simulation

For A matrix:

```
A_mat=  [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 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 -g 0 0 0 0 1 0 0 0 0 0;
         g 0 0 0 0 0 0 1 0 0 0 0;
         0 0 0 0 0 0 0 0 1 0 0 0];
```

Figure 20: A matrix Matlab

For B, C and D matrices:

```
B_mat=   [0   0    0    0
0    0    0    0
0    1/I_xx   0    0
0    0    1/I_yy   0
1/m  0    0    1/I_zz
0    1    0    0
0    0    1    0
0    0    0    1
1/m  0    0    0
0    1/I_xx   0    0
0    0    1/I_yy   0
1/m  0    0    1/I_zz];

C_mat= eye(12);

D_mat=zeros(12,4);
```

Figure 21: B,C and D matrices Matlab

for LQR also need to create the Q and R matrices where Q matrix are the cost associated with our system variables and R matrix are the cost associated with our system inputs. For Q matrix we define it as identity of 12 and for R we shall use identity matrix of 4 to signify the 4 main inputs. Hence, we can also finally get our K gain matrix for the LQR system which is used to minimize our cost function, as in:

```
Q_mat=eye(12);

R_mat=eye(4);

K_mat=lqr(A_mat,B_mat,Q_mat,R_mat);
```

For K matrix (Gain):

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.4580	-92.4275	-0.8680	-0.0816	-0.0957	0.4804	-31.6753	-494.9630	0.7358	0.8633	0.2351	2.5026
2	-14.3867	-9.6918	-0.0040	0.9435	-0.3190	0.0895	-7.2527	-10.2370	-0.1690	1.4233	-0.9087	0.3689
3	2.6731	-28.5365	-0.2536	0.3195	0.8841	-0.2278	-46.9616	-153.2006	-0.8800	0.2548	1.3259	-0.1402
4	3.4545	24.2263	0.4269	0.0327	0.3277	0.8422	-2.3952	103.1675	0.5427	-0.4341	-0.0211	0.5888

Figure 22: K matrix Matlab

the final LQR model:

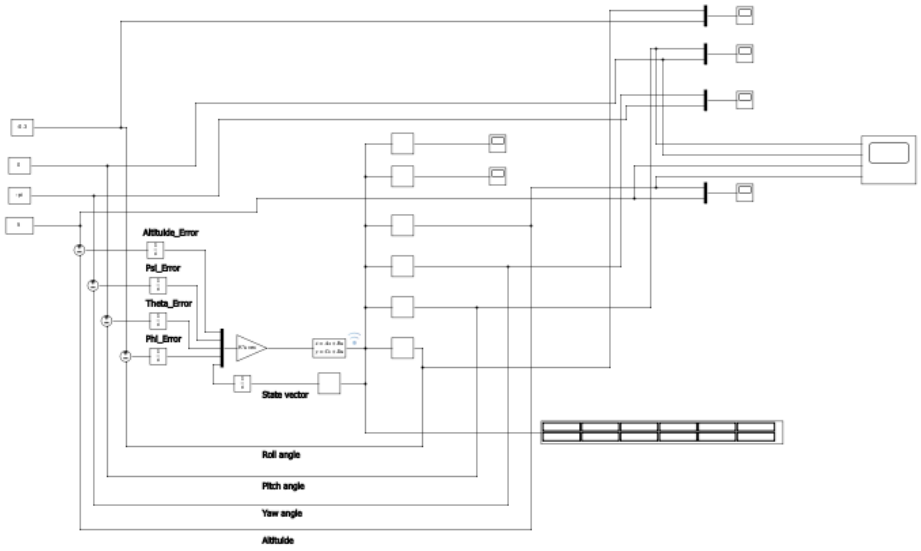

Figure 23: Quadcopter model using LQR

As part of our methodology, we apply the LQR gain matrix K as negative feedback to stabilize the system using a linearized model. To remove steady-state defects, integrators are used. In addition, we separate our desired outputs from the state vector using selectors. These can be subtracted from the intended numbers to get the error, which allows the controller to stabilize the system and get the right results.

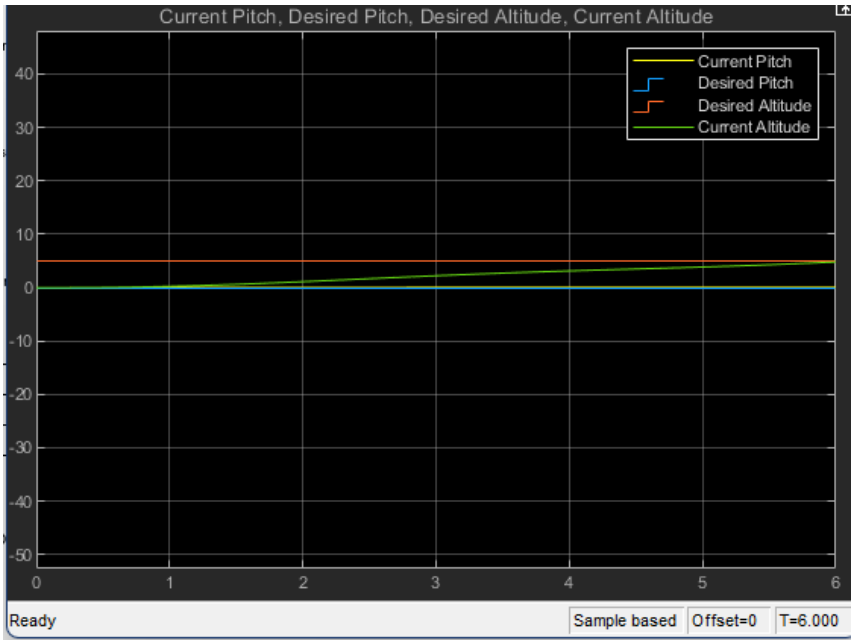

Figure 24: LQR outputs

This graph shows how the controller feedback causes our readings to ultimately approach the intended levels. Pitch and altitude are kept within the designated range by the quadcopter. In particular, the graph shows that our quadcopter successfully maintains its orientation by staying stable and not fall from its design position.

7. Noise

We may verify our system's stability in the presence of mistakes or inconsistent sensor calibrations by adding noise to it. Since hardware is the cause of these problems, we have no direct influence over them. To improve overall system robustness, we can, nevertheless, modify our control procedures to minimize or reduce the additional mistakes observed in our signals.

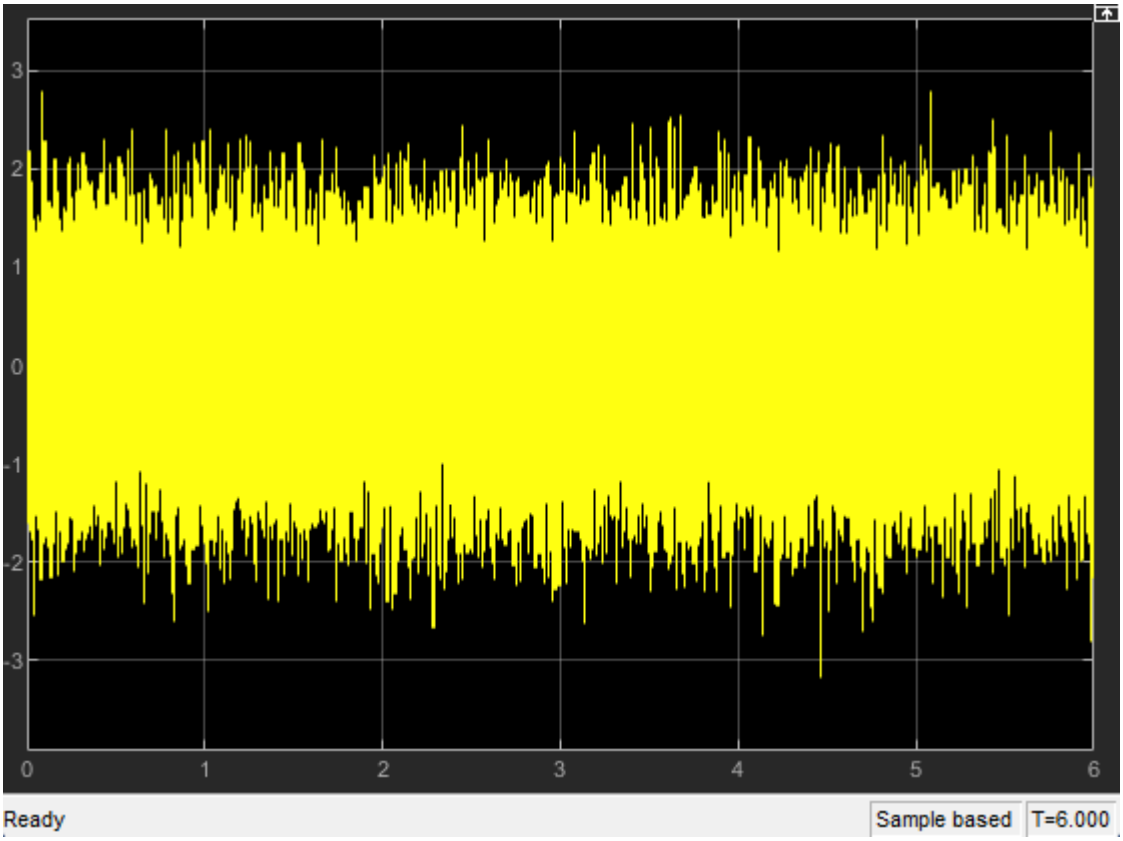

Figure 25: Added noise (white noise)

We incorporate this randomized noise signal into our model to see how the output varies in height and angles. We next try to manage the signal using our LQR and PID models to offset these additional changes. White noise is a kind of signal that is perfect for simulating different environmental conditions that our model might face. White noise offers a wide range of signals that can affect our system because its amplitude and frequency are completely random. This approach enables us test the robustness and effectiveness of our control strategies under varied scenarios.

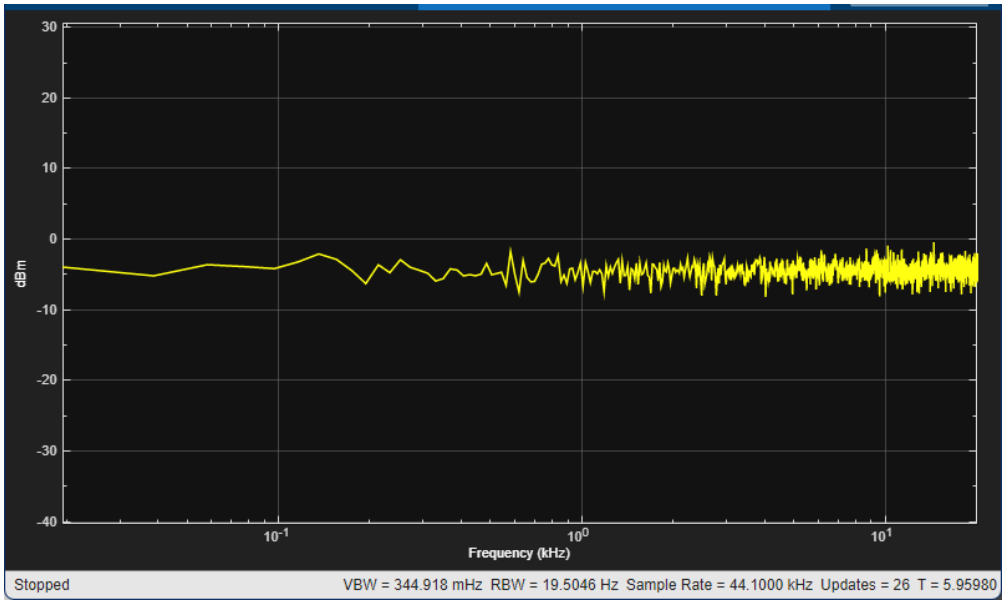

Figure 26: Noise distribution

It is evident that the noise is dispersed uniformly across the system, which leads to a wider range of oscillations that may impact our system. By subjecting the system to a variety of possible shocks, this distribution enables us to properly assess the resilience of our control systems.

7.1. PID response to noise

In this model we had to apply noise to each individual input to get the additon of noise into our system, this overall dissorientes our system into unstability and might make it difficult for PID controller to maintain our desired value:

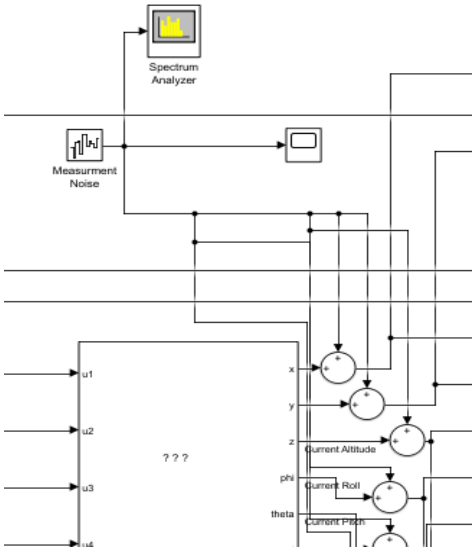

Figure 27: Added noise to PID model

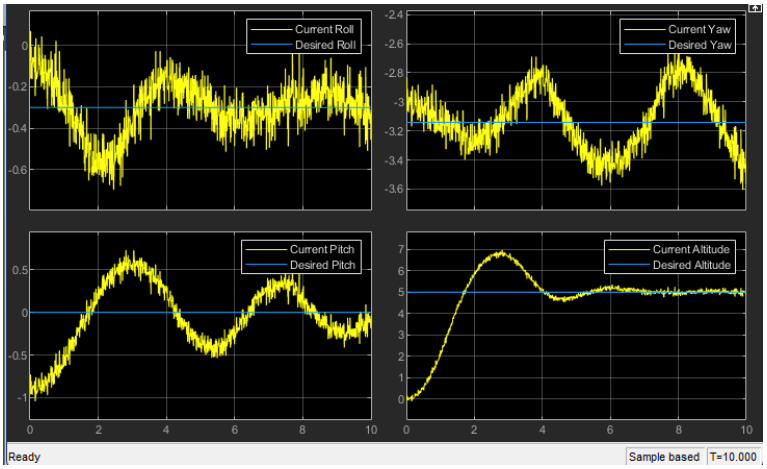

Figure 28: PID model outputs with added noise

7.2. LQR response to noise

We introduce a noise signal into the state space of our model, resulting in changes that cause the current signal to become disrupted and unstable. After that, the LQR control applies the LQR gain matrix to stabilize the system and make sure its target output values are maintained.

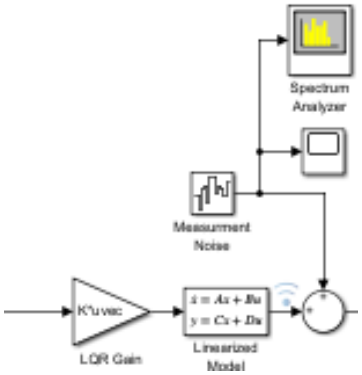

Figure 29: Added noise to LQR model

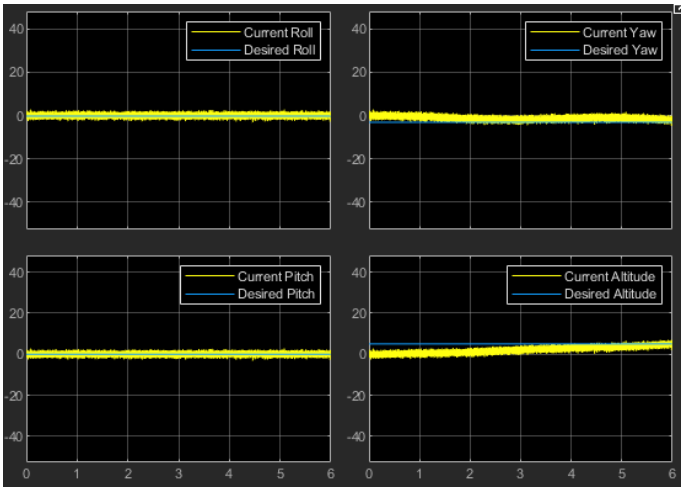

Figure 30: LQR model outputs with added noise

This graph shows that as compared to the PID controller, our LQR system performs noticeably better and is more stable. LQR controllers produce control techniques that are naturally resistant to disturbances like white noise by minimizing a cost function that takes the impacts of noise into account. Furthermore, full-state feedback is employed by LQR controllers, providing access to measurements of the complete system state. Because of this, LQR controllers are able to better attenuate white noise at particular frequencies than PID controllers and optimize control performance across a range of frequencies by shaping the closed-loop frequency response of the system.

8. Conclusion

In this research, we examined the mathematical modeling and control of quadcopters. The Newton-Euler and Euler-Lagrange equations were employed to derive the differential equations for quadcopter dynamics, which were validated through Matlab simulations. We initially used a PD controller to stabilize the quadcopter's attitude and developed a heuristic technique to manage its trajectory. While the PD controller effectively stabilized the quadcopter to the desired attitude and height, it did not account for the x and y positions, leading to deviations during stabilization due to roll and pitch angles.

To address this, we linearized the PID model to obtain the system matrices (A, B, C) for implementing an LQR controller. The LQR controller provided optimal control for the complex system, yielding better performance than the PD controller. Additionally, we introduced noise into the system to test robustness. The LQR controller demonstrated superior response to noise compared to the PD controller, maintaining stability and accurate trajectory tracking.

Overall, the heuristic approach produced good flight trajectories, and the integration of the LQR controller enhanced the quadcopter's stability and control, even in the presence of disturbances. The simulation results confirmed the effectiveness of the LQR controller in managing both the position and attitude of the quadcopter more efficiently than the PD controller alone.

9. References

[1] A. Espinoza, "Quadrotor model and control," GitHub repository:

https://github.com/AngeloEspinoza/quadrotor-model-and-control

[2] MathWorks, "LQR: Linear-Quadratic Regulator," MathWorks.:

https://www.mathworks.com/help/control/ref/lti.lqr.html

[3] MIT OpenCourseWare, "State-Space Representation," MIT OpenCourseWare:

https://ocw.mit.edu/courses/res-6-007-signals-and-systems-spring-2011/pages/lecture-notes/MITRES_6_007S11_lec19.pdf.

[4] J. O. Smith, "White noise," in Introduction to Digital Filters with Audio Applications, W3K Publishing:

https://www.dsprelated.com/freebooks/sasp/White_Noise.html

[5] MathWorks, "Spectrum Analyzer and Noise Measurement in Simulink," MathWorks:

https://www.mathworks.com/help/dsp/ref/spectrumanalyzer.html

[6] R. Abdolhoseini and M. Eghtesad, "Genetic Algorithm Based LQR Control for AGV Path Tracking Problem," ResearchGate:

https://www.researchgate.net/publication/352826613_Genetic_Algorithm_Based_LQR_Control_for_AGV_Path_Tracking_Problem

[7] J. R. Smith, "Lecture 15: Linear-Quadratic Regulator (LQR)," CSE 478: Capstone Software Design, University of Washington:

https://courses.cs.washington.edu/courses/cse478/20wi/site/resources/lec15_lqr.pdf