Quadrotor Attitude and Position Stabilization

- Using PID and LQR -

Devrat Singh, Fikrican Özgür,
Codrin-Matei Căciuleanu

B.Sc. in Electronics and Computer Engineering
Group ED5-1-F19

Fifth Semester Project

This document was typeset using Overleaf, an online LaTeX editor. The control systems were analyzed using MATLAB R2019a. The simulations were performed in SIMULINK 9.3 and the graphs were made in MATLAB as well. The other figures were created in Inkscape and draw.io.

AALBORG UNIVERSITY
STUDENT REPORT

Faculty of Engineering and Science
Aalborg University
http://www.aau.dk

Title:
Quadrotor Attitude and Position Stabilization

Theme:
Automation

Project Period:
Autumn Semester 2019

Project Group:
ED5-1-F19

Participant(s):
Devrat Singh
Fikrican Özgür
Codrin-Matei Căciuleanu

Supervisor(s):
Petar Durdevic Løhndorf

Copies: 1

Page Count: 74

Date of Completion:
December 19, 2019

Abstract:

The present report explores the matter of regulating the attitude and position of a quadrotor employing fundamental techniques of classical and modern control theory. A suitable model retaining only the relevant system dynamics is constructed by adhering to the Newton-Euler technique, while the unknown parameters are identified empirically. Necessary signal processing approaches are taken to compensate for the sensor measurement shortcomings by means of a complemetary filter. On the control side, two methods are put forward: PID and LQR. With the aid of MATLAB and SIMULINK, their performance is assessed in continuous time within the context of stability and reference tracking. Additionally, the PID controller is successfully discretized, simulated and finally implemented on an MCU, allowing the design to be tested on the real setup that was constructed. The results evinced the success of utilizing the designed controller on an actual quadcopter.

Table of Contents

Preface **ix**

1 Introduction **1**

2 Literature Review and System Description **3**
- 2.1 State of the Art 3
- 2.2 System Overview 4

3 Modeling **7**
- 3.1 Preambular Notation 7
- 3.2 Euler Angles 8
- 3.3 Rate of Change of Euler Angles 10
- 3.4 Rotor Dynamics 11
- 3.5 The Newton-Euler Method 14
- 3.6 Aerodynamic Effects 15
- 3.7 Nonlinear Model 17
- 3.8 Linear Model 18
- 3.9 Thrust Mixer 20
- 3.10 Parameter Estimation 20
- 3.11 Open-Loop Response 24
- 3.12 Model Validation 26

4 Sensors and Signal Processing **27**
- 4.1 Attitude Estimation 27
- 4.2 Position Estimation 34

5 Controller Design **37**
- 5.1 Proportional Integral Derivative (PID) Control 37
  - 5.1.1 Altitude and Attitude Control 41
  - 5.1.2 Position Control 44
  - 5.1.3 Trajectory Tracking 48
- 5.2 Linear Quadratic Regulator (LQR) Control 49
  - 5.2.1 Altitude Controller 52
  - 5.2.2 Attitude Controller 54

5.2.3 xy-Position Controller 55
5.2.4 Trajectory Tracking 58

6 Testing 59

7 Discussion 62

8 Conclusion 64

Bibliography 65

A Vectors in Rotating Reference Frames 69

B Inertia Tensor 72
B.1 Moments of Inertia of a Solid Rectangular Cuboid Body 73

List of Figures

2.1 Simplified Block Diagram of the Overall Control System 5

3.1 *Left:* The three reference systems considered; *Right:* The quadrotor within the body reference frame 7
3.2 Yaw 9
3.3 Pitch 9
3.4 Roll 9
3.5 Simplified UAV Free Body Diagram 11
3.6 Illustration of the Relationship Between the Motor Angular Speeds, Thrust Forces and the Rotational Motion of the UAV. *Top to Bottom:* Roll Rotation, Pitch Rotation, Yaw Rotation (With Positive Angles) 13
3.7 Relationship Between Ω_i^2 and F_i 22
3.8 Relationship Between Ω_i^2 and M_i 22
3.9 Relationship Between Ω_i and the Pulse Width of the Signal Given to the ESC 23
3.10 Block Diagram of the Linear Decoupled Quadrotor Model 24
3.11 Open-Loop Response of the Linear and Nonlinear Systems 25

4.1 Complementary Filter Block Diagram 30
4.2 Accelerometer and Gyroscope Raw ϕ and θ Measurements - Offsets Are Removed 30
4.3 FFT of Raw Accelerometer Measurements - Quadrotor Is Stationary 31
4.4 Filtering Raw Accelerometer Measurements Using Different Continuous Time Filters 31
4.5 1st Order Direct-Form 1 IIR Filter Structure 32
4.6 Complementary Filter Magnitude Plot 33
4.7 Complementary Filter Input and Output Signals 33
4.8 Real-time MATLAB Animation of the Quadrotor Based on IMU Measurements 35
4.9 The Effect of Euler Angles on Distance Measurements 36
4.10 Comparison of Measured and Calculated y-Distances 36

5.1 PID Hierarchical Control Structure 38

5.2 *Left:* Importance of Knowing the Yaw Angle When Tracking an xy-Position Reference; *Right:* Necessity of Negative Gains for PID$_y$ 40
5.3 PID Altitude and Attitude Control Structure 41
5.4 Closed-Loop Magnitude Bode Plots for Continuous Altitude and Attitude Control 43
5.5 Closed-Loop Altitude Response of the Nonlinear Model 44
5.6 Closed-Loop Position Response of the Nonlinear Model *Top to Bottom:* ϕ, θ, ψ 45
5.7 Closed-Loop Magnitude Bode Plots for Continuous xy-Position Control 46
5.8 Closed-Loop Position Response of the Nonlinear Model *Top to Bottom:* x, y 47
5.9 Attitude Response of the Nonlinear Model During Position Control. *Top to Bottom:* ϕ, θ 47
5.10 3D Helix Trajectory Tracking Response of the Nonlinear Model . . 48
5.11 LQR Control Structure 51
5.12 Altitude Response for LQR Altitude Control 53
5.13 ϕ. θ, and ψ Responses for LQR Attitude Control 55
5.14 xy-Position Closed-Loop Response for LQR 57
5.15 Roll and Pitch Dynamics in the Case of Position Control for LQR . 57
5.16 3D Helix Trajectory Tracking Response of the Nonlinear Model for LQR 58

6.1 The Quadrotor Developed As Part of the Project 59
6.2 xyz-Position and Attitude Control Test Results 61

B.1 Rectangular Cuboid 74

List of Tables

3.1 Constant Parameters of the UAV 23

4.1 Acceleration and Position Errors Caused Due to Sensor Misalignment 35

5.1 Saturation of Hierachical PID Control Signals 41
5.2 PID: Attitude and Altitude Time Domain Specifications 44
5.3 xy-Position Time Domain Specifications 46
5.4 LQR Performance Requirements 52
5.5 Altitude Controller Parameters 53
5.6 Attitude Controller Parameters 55
5.7 xy-Position Controller Parameters 58

Preface

The project entitled "Quadrotor Attitude and Position Stabilization" was written by three students from the Electronics and Computer Engineering Bachelor's program at Aalborg University Esbjerg, for the P5 project in the fifth semester. Hereafter, every mention of "we" refers to the three co-authors listed below.

The completion of this project would not have been possible without the contribution of our supervisor, Petar Durdevic Løhndorf, who directed his efforts towards guiding us to the right path. We would like to express our gratitude for his support and relevant advice.

Aalborg University, December 19, 2019

[illegible]

Devrat Singh
<dsingh17@student.aau.dk>

Fikrican

Fikrican Özgür
<fyozgy17@student.aau.dk>

[illegible]

Codrin-Matei Căciuleanu
<ccaciu17@student.aau.dk>

Chapter 1

Introduction

The popularity and utilization of UAVs (Unmanned Aerial Vehicles) is at the highest it has ever been. This has all been possible due to increased accessibility to effective and economic sensors along with improved battery technology. Consequently, these resulted in the development of UAVs that are affordable, can fly for longer durations as well as carry larger weights as compared to their predecessors. These circumstances opened the gates for the exploration of drone applications in different sectors. As such, currently there are plethora of universities, as well as private firms, who focus their research on UAV technology and its applications. Although the research directions are numerous, one of the examples is in drone navigation, which treats problems like path finding or obstacle avoidance in unknown environments. Another area of research is the development of state-of-the-art controllers that can provide better stabilization in the presence of disturbances and uncertainties. With so much attention, there are bound to be areas of application in which UAVs come into prospect. These include safety (through monitoring of airspace), environmental protection (using pollution measurements), aerial inspection of infrastructure etc. [1].

UAVs are of many types but one of its subcategories, the quadrotor[1], is the subject of this project. Compared to other UAVs, the quadrotors exhibit a simple structure with capabilities for diverse flight characteristics, like hovering and the VTOL (Vertical Take Off and Landing), making it suitable for flying in almost every kind of environment such as indoors or places with tight spaces [2]. However, the quadrotor system is highly nonlinear, it exhibits 6 degrees of freedom (DoFs), but only 4 controllable inputs, making it an underactuated system [3]. Furthermore, in contrast to their terrestrial counterparts, drones have very little friction in air and need to generate their own damping forces in order to stop their movement and stabilize [4]. These factors, along with many others, make the stabilization of a quadrotor a fascinating control problem. Subsequently, this report deals with the modelling and control of a quadrotor in consideration to

[1]Henceforth, the terms "UAV", "quadrotor", "quadcopter" and "drone" will be utilized interchangeably.

hovering and position control.

To approach the problem systematically, the work is divided into five main sections. First, modelling, as the name suggests this section treats the objective of dynamical modelling of the quadrotor using the Newton-Euler method. The same section also gives an insight on the required mechanisms such as linearization and thrust mixing. Second, the sensors and signal processing section shifts its attention towards the sensor unit with details on the utilized sensory devices as well the necessary filters which are implemented in tandem with them. Third, the controller design and implantation, this section is aimed towards the development of the quadrotor controllers, which in our case constitute the PID and the LQR. For both the controllers, the results are shown from simulation for each of the controlled outputs. Lastly, the testing section discusses the test setup and presents the results of the implemented control strategy.

Chapter 2

Literature Review and System Description

2.1 State of the Art

In order to gain insight into the problem of stabilizing a quadcopter, carrying out a short review of the specialized literature was found appropriate. This section thus lists the prevailing approaches associated with the three following facets of drone control: modeling, control strategy and sensing.

Modeling

By far the most common way of describing the orientation of a quadcopter is through rotation matrices constructed based on one of the Euler angle conventions [1, 3, 5]. An alternative to this approach is represented by the less-intuitive quaternion formalism [6, 7, 8, 9]. When it comes to the dynamic equations, a very ubiquitous modeling method consists of the Newton-Euler technique [3, 10]. An equivalent way of arriving to the same relations is via the Euler-Lagrange method [11, 12]. Apart from these first principle procedures, system identification has been employed in works such as [13, 14, 15] and [16], the latter two choosing the utilization of artificial neural networks (ANNs) in this endeavour.

Control Strategy

Unsurprisingly, a very much preferred approach in the literature is to use the classical PID controller, which is extremely popular in the field of automatic industrial processes [14]. There is evidence of the use of a plethora of methods for identifying suitable PID quadrotor controller gains. Trial and error is utilized in [17, 18], Genetic Algorithm is implemented in [19, 20], while optimal PID gains are found in [14] via Particle Swarm Optimization (PSO).

Another popular choice is the well-known optimal control algorithm known as LQR. Many authors explore its performance, with some examples being [21, 22, 23]. A more complex time-varying version of LQR is mentioned in [10, 24].

In an attempt to circumvent the inherent disadvantages of linear control, nonlinear alternatives have been studied and implemented. In [25] Approximate Dynamic Programming (ADP) and Model-Predictive Control (MPC) are implemented on a real quadrotor. MPC is also used together with a nonlinear H_∞ controller in [26]. In addition, feedback linearization control is discussed in [27, 11], while the utilization of backstepping and sliding mode control is pursued in both [28] and [11].

Finally, as with modelling, neural networks became established tools within the control field and recently found their way into drone stabilization. The work conducted in [16, 29] is based on a direct inverse control using artificial neural network (DIC-ANN) scheme.

Sensors and State Estimation

The literature provides different approaches towards sensing and data acquisition. With GPS not being reliable indoors [30], for drone position and linear velocity estimation, motion capture (MOCAP) systems, (Vicon, in particular) are a favoured sensing solution, mainly for their exceptional accuracy [30, 31, 32]. Small cameras mounted on the quadrotor itself are used in [28] for position control and the author uses a sonar for altitude measuremnts. The authors of [33] suggest the use of RGBD cameras, Kinect or laser range finders for the task of measuring relative position in indoor environments.

Attitude and angular velocity information is usually extracted via an onboard Inertial Measurement Unit (IMU), encompassing a gyroscope, an accelerometer and possibly a magnetometer [28, 23, 32]. Additionally, state estimation through some form of complementary filter [24, 33, 34], Kalman Filter (KF) [35, 36] or Extended Kalman Filter (EKF) [8, 24] is an extensive and prevalent topic discussed in many papers and reports. An especially circumstantial description of measurement and state estimation in provided in [10], in which MOCAP, optical flow, IMU, laser ranging and ultra wide band (UWB) positioning are fused through an EKF, for all possible combinations.

2.2 System Overview

With a preliminary knowledge of the previous works documented in the quadrotor control literature, it was possible to define the methods to be utilized in the present project. As this represents the authors' first contact to the world of UAVs, the most common and well-documented modeling and control paths have been pursued. That is, rotation matrices and the Newton-Euler formalism were chosen as means of modeling, while PID and LQR were selected as the controllers of attitude, altitude and position. In the context of sensing, the simpler complementary filter option is picked for obtaining reliable attitude estimations from the IMU, while three-dimensional position sensory information is extracted via three Time-of-Flight (ToF) laser-ranging modules[1].

A block diagram illustrating the elements of the system and their corresponding positions within a very generic control loop is shown in Figure 2.1. The diagram applies almost directly to the structure used in PID control, while minor modifications can be made to accommodate for LQR-control representation.

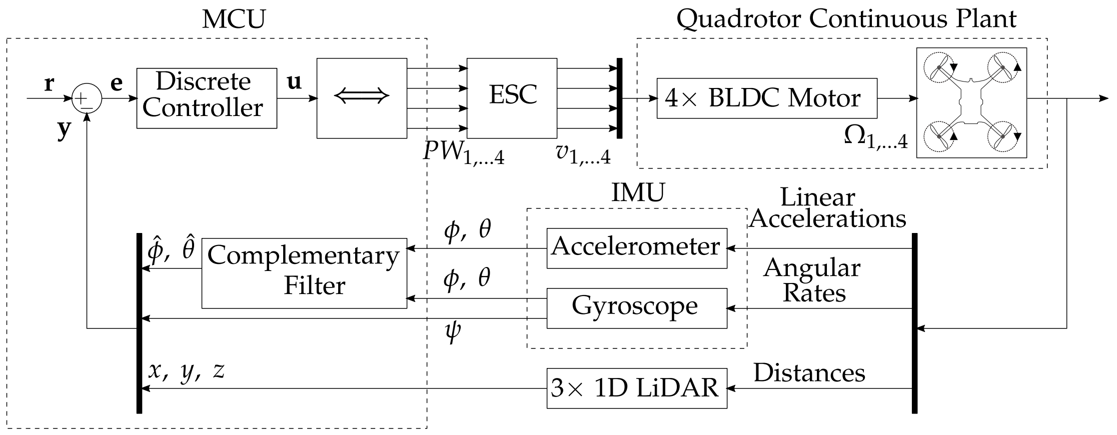


Figure 2.1: Simplified Block Diagram of the Overall Control System

Many more details are provided in the following chapters, though examining this superficial description of the system is useful for a general understanding of the procedure employed here for controlling a quadrotor, on both a hardware and a software level. A microcontroller (MCU) lies at the core of the control system. It is programmed to run the code of the discrete controller, who computes a certain control signal **u**, based on the error **e** between a user-selected reference **r** and the measured outputs of interest **y** of the physical system. To create the desired effect in the real world, four electrical signals of specific pulse widths (PW_i) must be sent to the Electronic Speed Control (ESC), which in turn excites the winding of each brushless DC (BLDC) motor with a voltage v_i. The electric motors convert the received electrical energy to mechanical energy, which manifests itself in the form of thrusts and torques acting on the UAV's airframe, thus causing changes in the system's state. Through sensory devices, several variables are measured and fed back to the MCU for a new computation of the control. As mentioned before, the sensors are three 1D LiDARs that acquire the realtive position (x, ,y, z) of the drone within a specific indoor testing environment, and an IMU supplying attitude measurements. This unit encompasses an accelerometer, able to perceive changes in the linear velocity of the quadcopter, and a gyroscope, which measures the angular speeds expressed in the body frame. Through calculations, it is possible to derive the actual inclination angles of the quadrotor, i.e.: roll (ϕ), pitch (θ) and yaw (ψ). Each measurement carries certain drawbacks, mitigated through the complementary filter (CF), which is

[1]alternatively referred to "1D LiDAR" in this report

able to produce reliable angular estimations. As it will be discussed in Chapter 4, we used the CF for getting roll and pitch, while yaw is simply computed based on gyroscopic data.

Materials

In terms of the specific parts utilized, we opted for Teensy 3.6 [37] as the controller implementation platform, and chose Aikon AK32 35A V2 4in1 BlHeli32 [38] as the ESC. The BLDC motors are EMAX RS2205 2600KV [39], the ToF modules are from STMicroelectronics, model VL53L0X [40], while the IMU number is GY-87 MPU6050 HMC5883L BMP180 [41], including on a single chip a 3-axis accelerometer, a 3-axis gyroscope, a 3-axis magnetometer and a barometric pressure sensor, thus measuring 10 DoFs. Apart from these, a battery was required to power the electronics, and the preferred choice was 3S Gens Ace EC5 Bashing Series, rated 5000mAh and 50C [42].

Chapter 3

Modeling

3.1 Preambular Notation

It was considered convenient to follow the lead of [10] and dedicate a short initial subsection to the notations that will be used throughout the modeling part. There are three discrete reference systems utilized for the localization of the quadrotor in space. The first one is the inertial global reference frame ($\mathcal{G}$), considered fiducial in time, i.e., attached to the earth (assumed stationary and flat in this report). This is followed by a second inertial frame ($\mathcal{I}$), whose origin coincides with the center of mass (COM) of the aircraft, and has its unit vectors parallel to those of the global frame. The third and last frame consists of the non-inertial body frame ($\mathcal{B}$), firmly attached to the drone and therefore rotating concomitantly with it. The latter coordinate system has its origin in the center of mass as well and will start by being superimposed to the inertial frame. This is illustrated in Figure 3.1.

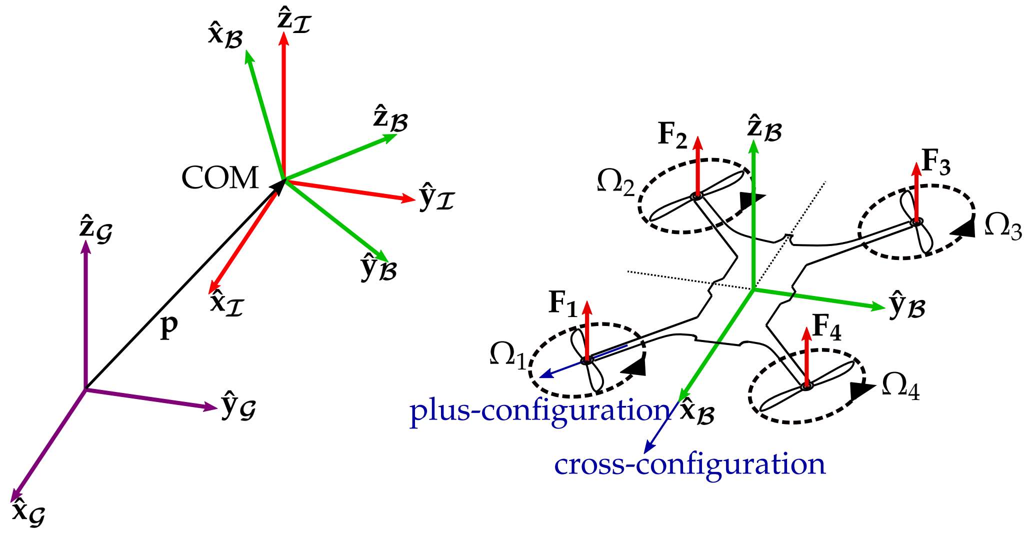

$\hat{\mathbf{x}}_{\mathcal{B}}$
$\hat{\mathbf{z}}_{\mathcal{I}}$
$\hat{\mathbf{z}}_{\mathcal{B}}$
COM
$\hat{\mathbf{y}}_{\mathcal{I}}$
$\hat{\mathbf{z}}_{\mathcal{G}}$
$\hat{\mathbf{y}}_{\mathcal{B}}$
$\hat{\mathbf{x}}_{\mathcal{I}}$
$\mathbf{p}$
$\hat{\mathbf{y}}_{\mathcal{G}}$
$\hat{\mathbf{x}}_{\mathcal{G}}$
$\mathbf{F_2}$
$\hat{\mathbf{z}}_{\mathcal{B}}$
$\mathbf{F_3}$
Ω_2
Ω_3
$\hat{\mathbf{y}}_{\mathcal{B}}$
$\mathbf{F_1}$
$\mathbf{F_4}$
Ω_1
Ω_4
plus-configuration
$\hat{\mathbf{x}}_{\mathcal{B}}$
cross-configuration

Figure 3.1: *Left:* The three reference systems considered; *Right:* The quadrotor within the body reference frame

Henceforth, we may refer to the aforementioned frames as "global frame" or "$\mathcal{G}$-frame", "inertial frame" or "$\mathcal{I}$-frame" and "body frame" or "$\mathcal{B}$-frame", respectively. Any vector related to the dynamic behaviour of the UAV can be expressed in each one of these frames of reference using the corresponding basis vectors. Which basis vectors are used will be indicated, when appropriate, by a right subscript associated with the correct frame, as it will be shortly shown.

Let the vector $\mathbf{p}$ mark the position of the UAV's center of mass with respect to the hypothetically-assigned origin of the $\mathcal{G}$-frame. Additionally, let $\mathbf{v}$ and $\boldsymbol{\omega}$ define the traslational and angular velocities of the body frame relative to the global (or inertial) frame. Finally, let $\boldsymbol{\eta}$ be the vector containing the three Euler angles, which are defined intrinsically. Equation 3.1 shows the notation for the three components of each of the preceding vectors, when each vector is expressed in the frame of reference found most convenient for writing the dynamic equations.

$$\mathbf{p}_{\mathcal{G}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \mathbf{v}_{\mathcal{B}} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \qquad \boldsymbol{\omega}_{\mathcal{B}} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad \boldsymbol{\eta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{3.1}$$

Figure 3.1 also shows two possible configurations for defining the body frame basis vectors and therefore the axes of rotations. The "cross configuration" defines the $x_{\mathcal{B}}$-axis in such a way that two motors (1 and 2 or 3 and 4) are employed in order to create a rotation around this axis. Another combination of motors is thus used to rotate around the $y_{\mathcal{B}}$-axis. In the alternative "plus-configuration", the $x_{\mathcal{B}}$-axis lies along a line linking the COM to one of the motors (1, in the figure). This topology allows only one motor to be used for rotating around the axis. The same can be stated about the $y_{\mathcal{B}}$-axis [10]. Moreover, further complications in calculating torques emerge if the positioning of the rotors does not describe the vertices of a square. For these reasons, the former configuration was selected.

3.2 Euler Angles

The convention adopted in this report is the one of the Tait-Bryan (alternatively named Cardano/Cardan [28, 43]) angles, which find a widespread use in aeronautical applications. It is particularly within this convention that the angles of roll, pitch and yaw are explicitly defined [44]. For convenience, the formulations "Euler angles" and "Tait-Bryan angles" will be utilized interchangeably throughout this report.

Citing from [44], "Euler proved that: Any two independent orthogonal coordinate frames with a common origin can be related by a sequence of three rotations about the local coordinate axes, where no two successive rotations may be about the same axis. In general, there are 12 different independent combinations of triple rotation about local axes". Equations 3.2-3.4 show the three chosen rotation matrices for describing the UAV attitude in terms of roll, pitch and yaw, while Figures 3.2-3.4 illustrate the rotations and the changes of bases that occur. A transformation resulting from multiplying a given vector with one of the matrices creates either a clockwise rotation of the vector about the respective axis, or a counterclockwise rotation of the coordinate system. The two interpretations are equivalent [45]. Adopting the following shorthand notations: $c_\alpha = \cos\alpha$ and $s_\alpha = \sin\alpha$, we have:

$$\mathbf{R}_z(\psi) = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \tag{3.3}$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \tag{3.4}$$

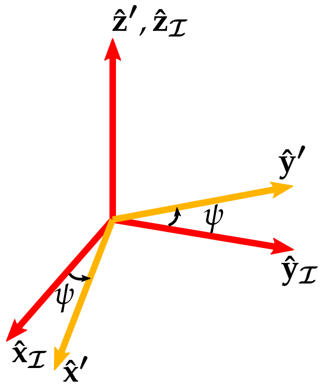

$\hat{\mathbf{z}}', \hat{\mathbf{z}}_\mathcal{I}$
$\hat{\mathbf{y}}'$
ψ
$\hat{\mathbf{y}}_\mathcal{I}$
ψ
$\hat{\mathbf{x}}_\mathcal{I}$
$\hat{\mathbf{x}}'$

Figure 3.2: Yaw

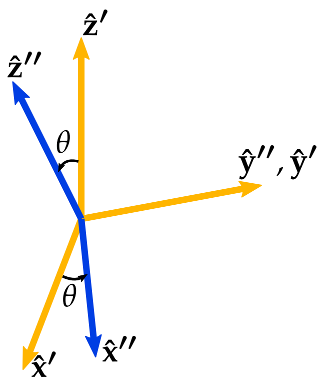


Figure 3.3: Pitch

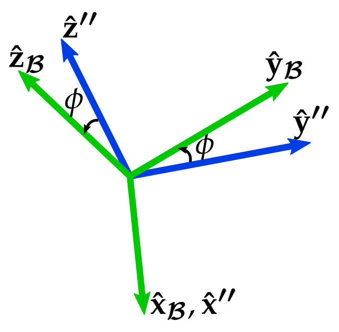

$\hat{\mathbf{z}}''$
$\hat{\mathbf{z}}_\mathcal{B}$
$\hat{\mathbf{y}}_\mathcal{B}$
ϕ
$\hat{\mathbf{y}}''$
ϕ
$\hat{\mathbf{x}}_\mathcal{B}, \hat{\mathbf{x}}''$

Figure 3.4: Roll

Any rotation in three dimensions from the inertial frame to the rotational one can be described by a composition of rotations around the three moving axes [44, 43]. Utilizing the intrinsic convention zyx, in which the rotations are performed with respect to the body-fixed coordinate system (as depicted in Figures 3.2-3.4), the overall rotation matrix converting between the two frames is computed as shown in Eqution 3.5. (We chose the same matrices and order of rotation as [10, 44, 46].)

$$ {}^{\mathcal{B}}_{\mathcal{I}}\mathbf{R} = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi) = \begin{bmatrix} c_\psi c_\theta & c_\theta s_\psi & -s_\theta \\ c_\psi s_\phi s_\theta - c_\phi s_\psi & c_\phi c_\psi + s_\phi s_\psi s_\theta & c_\theta s_\phi \\ s_\phi s_\psi + c_\phi c_\psi s_\theta & c_\phi s_\psi s_\theta - c_\psi s_\phi & c_\phi c_\theta \end{bmatrix} \tag{3.5}$$

Since rotation matrices are orthogonal [45, 44], the inverse mapping between reference frames is possible using the property in Equation 3.6.

$$ {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R} = {}^{\mathcal{B}}_{\mathcal{I}}\mathbf{R}^{-1} = {}^{\mathcal{B}}_{\mathcal{I}}\mathbf{R}^{T} \tag{3.6}$$

With the rotation matrix now defined, we can already write a relation leading to three equations that will be part of the nonlinear quadcopter model. Based on the definitions from Equation 3.1 and the notations in Appendix A, we have Equation 3.7, linking the linear speeds of the body frame with respect to the inertial frame, expressed in the basis of the inertial frame, and the same linear speeds, only this time written using the vector basis of the body frame:

$$\dot{\mathbf{p}}_{\mathcal{G}} = \mathbf{v}_{\mathcal{G}} = {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}\mathbf{v}_{\mathcal{B}} = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\phi s_\theta - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi s_\theta \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\psi s_\theta & c_\phi s_\psi s_\theta - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.7}$$

which, when performing the multiplications, becomes Equation 3.8:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} w(s_\phi s_\psi + c_\phi c_\psi s_\theta) - v(c_\phi s_\psi - c_\psi s_\phi s_\theta) + u c_\psi c_\theta \\ v(c_\phi c_\psi + s_\phi s_\psi s_\theta) - w(c_\psi s_\phi - c_\phi s_\psi s_\theta) + u c_\theta s_\psi \\ w c_\phi c_\theta - u s_\theta + v c_\theta s_\phi \end{bmatrix} \tag{3.8}$$

3.3 Rate of Change of Euler Angles

Due to the nature of the intrinsic Euler angles, intermediary Eulerian local reference frames are necessary for describing the sequence of rotations. When the derivatives of these angular displacements are computed, the result does not label the angular velocity of the body frame with respect to the inertial frame. However, this angular velocity can be expressed as a function of the rate of change of the Euler angles (also known as Euler frequencies) [44], as shown in Equation 3.9 [10, 46].

$$\boldsymbol{\omega}_{\mathcal{B}} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{I}_3 \begin{bmatrix} \frac{d\phi}{dt} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_x(\phi) \begin{bmatrix} 0 \\ \frac{d\theta}{dt} \\ 0 \end{bmatrix} + \mathbf{R}_x(\phi)\mathbf{R}_y(\theta) \begin{bmatrix} 0 \\ 0 \\ \frac{d\psi}{dt} \end{bmatrix} = \begin{bmatrix} 1 & 0 & s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.9}$$

In shorthand notation, we have the equivalent Equation 3.10:

$$\boldsymbol{\omega}_{\mathcal{B}} = \mathbf{W}(\boldsymbol{\eta})\dot{\boldsymbol{\eta}} \tag{3.10}$$

where $\mathbf{W}(\boldsymbol{\eta})$ denotes the matrix responsible for the mapping. The reciprocal of Equation 3.9 is valid as long as $\det(\mathbf{W}(\boldsymbol{\eta})) \neq 0$. It is clear that this condition is fulfilled when $\theta \notin \left\{(2k+1)\frac{\pi}{2}, k \in \mathbb{Z}\right\}$, as the matrix $\mathbf{W}(\boldsymbol{\eta})$ becomes singular at steep pitch angles. This phenomenon is known as gimbal lock and occurs in practice predominantly during aggressive flight [10]. If we assume $\mathbf{W}(\boldsymbol{\eta})$ to be non-singular, we can obtain the Euler frequencies in terms of $\boldsymbol{\omega}_{\mathcal{B}}$, as shown by Equation 3.11 (where $t_\alpha = \tan\alpha$):

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{W}^{-1}(\boldsymbol{\eta})\boldsymbol{\omega}_{\mathcal{B}} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.11}$$

The vector $\boldsymbol{\omega}_{\mathcal{B}}$ denotes the angular velocity of frame $\mathcal{B}$ relative to frame $\mathcal{I}$, as expressed using the body frame basis vectors. We may also choose to express it in the inertial frame by premultiplying with ${}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}$ (Equation 3.12). We also note that, since frames $\mathcal{G}$ and $\mathcal{I}$ have parallel basis vectors, the representation of any vector is the same in both coordinate systems.

$$\boldsymbol{\omega}_{\mathcal{G}} = \boldsymbol{\omega}_{\mathcal{I}} = {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}\boldsymbol{\omega}_{\mathcal{B}} \tag{3.12}$$

3.4 Rotor Dynamics

The most relevant dynamics affecting the flight of the quadrotor are the total thrust generated by the four motors and the torques about the three axis [33, 27]. The lift forces are depicted in Figure 3.1 and also in the right-hand side of Figure 3.5, from a different vantage point. The free body diagram in Figure 3.5 illustrates the reaction torques about the $\mathbf{z}_{\mathcal{B}}$-axis arising due to the rotation of the propellers (see the left half), while also elucidating the direction of the moments corresponding to the individual thrust forces causing the drone to either roll or pitch (see the right half). It should be noted that the aim of the present section is to provide insight into the crucial dynamics and hence the forces and torques having less significant effects will be neglected. The model can always be extended if necessary by fully or partially considering these effects.

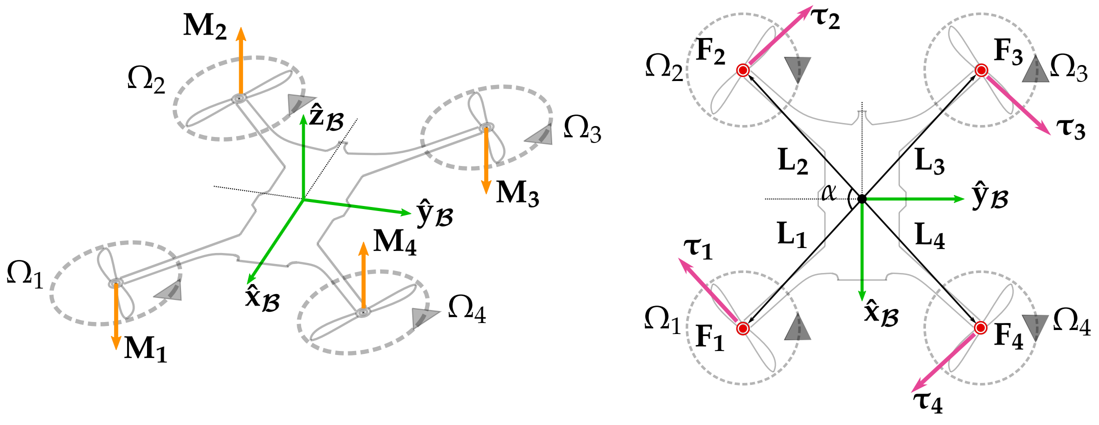


Figure 3.5: Simplified UAV Free Body Diagram

The amount of thrust generated by a single motor is given by the lumped model in Equation 3.13 [33]:

$$F_i = k_i {\Omega_i}^2 \tag{3.13}$$

where k_i is a constant encompassing the aerodynamics of the rotor. The total thrust $T_{\mathcal{B}}$ available for keeping the drone hovering (assuming identical motors for which $k_i = k_T,\ \forall i \in \{1, \dots 4\}$) is simply the sum of the individual contributions (Equation 3.14), all having the same orientation: parallel to $\hat{\mathbf{z}}_{\mathcal{B}}$.

$$T = \sum_{i=1}^{4} F_i = k_T \sum_{i=1}^{4} {\Omega_i}^2 \qquad \mathbf{T}_{\mathcal{B}} = T\hat{\mathbf{z}}_{\mathcal{B}} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{3.14}$$

As one propeller spins, it exerts a torque on the surrounding air. Based on Newton's third taw of motion, a reaction appears, creating a torque acting on the airframe in the opposite fashion. This torque tends to turn the body in the reverse direction as compared to the one of the rotor's tangential velocity [33]. If we follow the physical convention of [47, 48], all four moments will lie along $z_{\mathcal{B}}$, so the UAV will turn about this axis. Each torque can be modelled as in Equation 3.15:

$$M_i = b_i {\Omega_i}^2 + J_i \dot{\Omega}_i \approx b_i {\Omega_i}^2 \tag{3.15}$$

where b_i is another aerodynamic constant and J_i is the moment of inertia of a given rotor. Usually $\dot{\Omega}_i$ is small and thus the second term is omitted [3, 27]. The total torque τ_z about the $z_{\mathcal{B}}$-axis corresponds to the algebraic sum of all the individual motor torques. Again, assuming the rotors are identical, meaning $b_i = b_\tau,\ \forall i \in \{1, \dots 4\}$, we have the relation in Equation 3.16:

$$\tau_z = b_\tau(-{\Omega_1}^2 + {\Omega_2}^2 - {\Omega_3}^2 + {\Omega_4}^2) = b_\tau \sum_{i=1}^{4} (-1)^i {\Omega_i}^2 \tag{3.16}$$

The torques about the $y_{\mathcal{B}}$-axis and the $x_{\mathcal{B}}$-axis in the cross-configuration topology are obtained from the cross products of upward forces $\mathbf{F}_i$ - expressed in the body frame - and the corresponding distances $\mathbf{L}_i$ with respect to the COM [3, 10, 33]. Assuming $\|\mathbf{L_i}\| = L,\ \forall i \in \{1, \dots 4\}$, it becomes a matter of simple geometry to compute the required torques. (The directions of the torque vectors are again drawn in accordance to the classical convention in [47, 48].) The resulting formulae are the ones in Equations 3.17-3.18:

$$\tau_y = L \sin\frac{\alpha}{2}(-F_1 + F_2 + F_3 - F_4) = k_T L \sin\frac{\alpha}{2}(-{\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 - {\Omega_4}^2) \tag{3.17}$$

$$\tau_x = L \cos\frac{\alpha}{2}(-F_1 - F_2 + F_3 + F_4) = k_T L \cos\frac{\alpha}{2}(-{\Omega_1}^2 - {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2) \tag{3.18}$$

The total torque in the body frame $\boldsymbol{\tau}_{\mathcal{B}}$ gathers the preceding results in a vector, as indicated in Equation 3.19 [10]:

$$\boldsymbol{\tau}_{\mathcal{B}} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{3.19}$$

Figure 3.6 illustrates how rotational motion, as described using positive Euler angles, can be achieved by varying the motors' angular speeds in an adequate fashion. It is assumed in each case that the vertical component of the total thrust produced by the BLDC motors (in the $\mathcal{G}$-frame) is sufficient to balance Earth's gravitational pull. If motors 1 and 2 rotate at a slower rate as compared to the other two, the drone will roll to the left, as the sum of F_3 and F_4 is bigger than the one of F_1 and F_2. In the case when motors 2 and 3 are the ones rotating faster, the quadrotor will pitch forward, because this time the greater sum is formed by F_2 and F_3. Finally, whenever the speed of motors 2 and 4 is higher than the angular velocity of the 1-3 motor pair, the UAV yaws counterclockwise, since the torque formed by M_2 and M_4 overcomes the total moment acting in the opposite direction. In this case, $F_2 + F_4$ is obviously larger than $F_1 + F_3$.

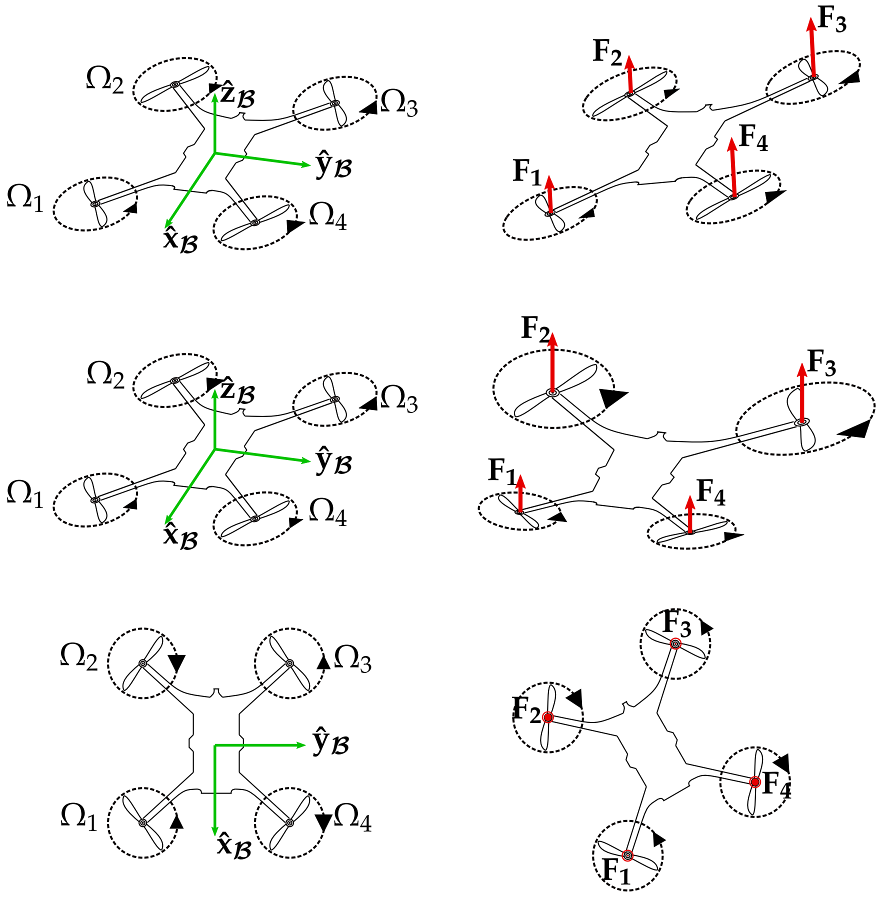

Ω_2
$\hat{\mathbf{z}}_{\mathcal{B}}$
Ω_3
$\hat{\mathbf{y}}_{\mathcal{B}}$
Ω_1
$\hat{\mathbf{x}}_{\mathcal{B}}$
Ω_4
$\mathbf{F_1}$
$\mathbf{F_2}$
$\mathbf{F_3}$
$\mathbf{F_4}$

Figure 3.6: Illustration of the Relationship Between the Motor Angular Speeds, Thrust Forces and the Rotational Motion of the UAV. *Top to Bottom:* Roll Rotation, Pitch Rotation, Yaw Rotation (With Positive Angles)

3.5 The Newton-Euler Method

In the specialized literature, one ubiquitous modelling approach to describe the movement of the quadrotor is represented by the rigid-body Newton-Euler equations from classical mechanics [3, 27]. In the body frame, these relations are the ones shown in Equations 3.20-3.21.

$$m\dot{\mathbf{v}}_{\mathcal{B}} + \boldsymbol{\omega}_{\mathcal{B}} \times m\mathbf{v}_{\mathcal{B}} = \mathbf{F}_{\mathcal{B}} \tag{3.20}$$

$$\mathbf{I}_{\mathcal{B}}\dot{\boldsymbol{\omega}}_{\mathcal{B}} + \boldsymbol{\omega}_{\mathcal{B}} \times (\mathbf{I}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}}) = \boldsymbol{\tau}_{\mathcal{B}} \tag{3.21}$$

where $\mathbf{F}_{\mathcal{B}}$ and $\boldsymbol{\tau}_{\mathcal{B}}$ are the net force and net torque, respectively, acting upon the airframe and expressed in terms of the body frame basis vectors. Additional details on vector differentiation in the context of rotating reference frames are provided by Appendix A.

Translation Equation

The acceleration of the quadrotor is a function of the total thrust $\mathbf{T}_{\mathcal{B}}$ produced by its four propellers and the gravitational force, both acting upon the UAV's COM. Thus, $\mathbf{F}_{\mathcal{B}}$ is equal to the sum of these two forces. Since the gravitational force impedes the upward vertical movement of the quadrotor by acting towards the center of the Earth, it is defined as a negative force parallel to $\hat{\mathbf{z}}_{\mathcal{G}}$ and denoted as $\mathbf{F}_g$, as shown in Equation 3.22:

$$\mathbf{F}_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{3.22}$$

In order to translate its effect to the body frame where we write Newton's second law of motion, ${}^{\mathcal{B}}_{\mathcal{I}}\mathbf{R}$ is premultiplied with it before the summation. The resulting translation equation is expressed in Equation 3.23.

$$\mathbf{F}_{\mathcal{B}} = \mathbf{T}_{\mathcal{B}} + {}^{\mathcal{B}}_{\mathcal{I}}\mathbf{R}\mathbf{F}_g = m\dot{\mathbf{v}}_{\mathcal{B}} + \boldsymbol{\omega}_{\mathcal{B}} \times m\mathbf{v}_{\mathcal{B}} \tag{3.23}$$

Performing the necessary substitutions using Equations 3.1, 3.5, 3.14, 3.22 and dividing by m on both sides yield Equation 3.24:

$$\begin{bmatrix} 0 \\ 0 \\ \frac{T}{m} \end{bmatrix} - \begin{bmatrix} c_\psi c_\theta & c_\theta s_\psi & -s_\theta \\ c_\psi s_\phi s_\theta - c_\phi s_\psi & c_\phi c_\psi + s_\phi s_\psi s_\theta & c_\theta s_\phi \\ s_\phi s_\psi + c_\phi c_\psi s_\theta & c_\phi s_\psi s_\theta - c_\psi s_\phi & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.24}$$

which can be further simplified by carrying out the matrix multiplications and additions to get Equation 3.25:

$$\begin{bmatrix} g s_\theta \\ -g c_\theta s_\phi \\ -g c_\phi c_\theta + \frac{T}{m} \end{bmatrix} = \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{3.25}$$

Rotation Equation

In the simplest possible model we aim for, the only torques acting on the airframe are the entries of the $\boldsymbol{\tau}_\mathcal{B}$ vector, produced by the motors. We can now evaluate Equation 3.21 as follows (see Equation 3.26) if we use Equation 3.19, the relevant parts of Equations 3.1, as well as Equation B.3 defining the inertia tensor $\mathbf{I}_\mathcal{B}$:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.26}$$

If we assume that the UAV is symmetric with respect to the axes belonging to the body frame, the inertia tensor will have the form given in Equation 3.27. Further information about this parameter is available in Appendix B.

$$\mathbf{I}_\mathcal{B} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.27}$$

We obtain Equation 3.28 by updating the inertia tensor in Equation 3.26.

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.28}$$

and performing the matrix multiplications and addition yields the simpler Equation 3.29.

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} - (I_{yy} - I_{zz})qr \\ I_{yy}\dot{q} - (I_{zz} - I_{xx})pr \\ I_{zz}\dot{r} - (I_{xx} - I_{yy})pq \end{bmatrix} \tag{3.29}$$

3.6 Aerodynamic Effects

While including the aerodynamic effects on the drone in the model will certainly improve the performance of the quadrotor, the simpler demands of this project, i.e.: stabilization of the quadrotor at hover condition, gives us the leeway of excluding non-consequential effects. Moreover, this section discusses those excluded aerodynamics as well as the reasoning for their exclusion.

Ground Effect

When the quadrotor operates near a level ground surface, about the height equal to half the diameter of the rotor, there appears a thrust augmentation which pushes the quadrotor away from the ground, this is related to the reduced airflow velocity [28, 49]. Although this could be beneficial in terms of fuel conservation as it leads to an increased rotor efficiency, it can also influence the quadrotor dynamics and make the control harder [49, 50]. One of the proposed mathematical descriptions of this effect is presented in Equation 3.30, such that T is the thrust produced by the propeller during ground effect and T_∞ is the thrust while being outside this effect's influence, at the same power. Also, R is the radius of the rotor and z is the altitude of the quadrotor.

$$\frac{T}{T_\infty} = \frac{1}{1 - \left(\frac{R}{4z}\right)^2} \tag{3.30}$$

Now, considering the situation where the drone is above the half diameter threshold, which leads to the ratio $\frac{z}{R} = 2$, then the predicted ratio between T and T_∞ is merely 1.016. Consequently, this gives basis for the negligence of ground effect above one rotor diameter [51]. On assessment of desired hovering altitude for our quadrotor (which is above one rotor diameter), there is no need for the inclusion of ground effect in the modelling procedure. However, it should be noted that the ground effect would still influence the take-off and landing, but its effects are believed to be compensated by the developed controller during those brief periods.

Gyroscopic Effect

Next, the considerations are in concern with the gyroscopic effect. According to the literature in [50] and [52], the gyroscopic effect could be neglected if the inertia of the propeller group is small as compared to the main quadrotor body (this could also be interpreted as mass of propellers being significantly less than the whole body). As such, the mass of the propeller group is $14g$, which contributes insignificantly to the weight of the whole body (777g), thus giving the grounds to ignore the said effect. Nonetheless, the unmodelled gyroscopic effect will be treated as a disturbance and thus will get rejected by the controller [50].

Complexities of Rotor Dynamics

The rotor dynamics play a crucial role in the modelling of the quadrotor but in consideration to the assumed fast transients for our motors, we utilized the static relation for the lift force F_i and the reaction moment M_i given in Equation 3.14 and 3.15.

Moreover, this simplistic assumption is effectual only in a limited prospect. In reality, the expressions for the lift force and the torque are complex functions of environmental conditions, such as air density, angle of attack (AOA) along with other rotor characteristics. During the translational movement, there is a thrust variation due to two related effects: translational lift and the change in angle of attack (AOA). This governs the dynamic relation for the total thrust, which could be derived using the combination of blade element theory and the momentum theory, as shown in [51]. But in the case of hovering (i.e.: static conditions), it is feasible to obtain the familiar $T \propto \Omega^2$ relation from the complex expression used for translational movement. Based on this understanding, it is fair to say that rotor dynamics are much more influential when translational motion and complex manoeuvres are at play. Although there will be some limited involvement of simplistic translational movement, any other complex manoeuvres are out of the scope of this project. On top of that, in the simple case of hovering (according to [50]), it is a reasonable assumption to ignore the aerodynamics of rotorcraft vehicles where the operation is at slow velocity. Therefore, in the end, the simplistic static model for lift force and reaction moment was utilized for modelling.

3.7 Nonlinear Model

The equations established, in the preceding sections, for describing the dynamics of the quadrotor can be transformed into the conventional state variable form and are illustrated below (Equation 3.31).

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{cases} \dot{p} = \frac{1}{I_{xx}}[\tau_x + (I_{yy} - I_{zz})qr] \\ \dot{q} = \frac{1}{I_{yy}}[\tau_y + (I_{zz} - I_{xx})pr] \\ \dot{r} = \frac{1}{I_{zz}}[\tau_z + (I_{xx} - I_{yy})pq] \\ \dot{u} = rv - qw + gs_\theta \\ \dot{v} = pw - ru - gc_\theta s_\phi \\ \dot{w} = qu - pv - gc_\phi c_\theta + \frac{T}{m} \\ \dot{\phi} = p + qs_\phi t_\theta + rc_\phi t_\theta \\ \dot{\theta} = qc_\phi - rs_\phi \\ \dot{\psi} = r\left(\frac{c_\phi}{c_\theta}\right) + q\left(\frac{s_\phi}{c_\theta}\right) \\ \dot{x} = w(s_\phi s_\psi + c_\phi c_\psi s_\theta) - v(c_\phi s_\psi - c_\psi s_\phi s_\theta) + uc_\psi c_\theta \\ \dot{y} = v(c_\phi c_\psi + s_\phi s_\psi s_\theta) - w(c_\psi s_\phi - c_\phi s_\psi s_\theta) + uc_\theta s_\psi \\ \dot{z} = wc_\phi c_\theta - us_\theta + vc_\theta s_\phi \end{cases} \tag{3.31}$$

where the state vector is given in Equation 3.32:

$$\mathbf{x} = \begin{bmatrix} x & y & z & u & v & w & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \in \mathbb{R}^{12} \tag{3.32}$$

and the input vector is the one in Equation 3.33:

$$\mathbf{u} = \begin{bmatrix} T & \tau_x & \tau_y & \tau_z \end{bmatrix}^T \in \mathbb{R}^{4} \tag{3.33}$$

3.8 Linear Model

The non-linearities present in the conglomeration of equations defining the model of the quadcopter present themselves as an impediment to the system analysis procedure. With a myriad of tools available for control system design of linear models, the natural succession to the process would be linearization. It begins with the selection of an equilibrium point around which a linear approximation of the system is identified. Since the quadrotor system is to be developed for the hovering condition, the operating point which constitutes the inputs and the states required to maintain the system in equilibrium serve as the point of linearization. The case of hovering is such that the quadrotor is oriented parallel to the ground with neither angular nor translational movement transpiring. Consequently, the angles, velocities and the input torque vector τ_B are all equal to zero. Equations 3.34 and 3.35 depict the input and state vector for the hovering condition, also dubbed the equilibrium point. In relation to the said equations, the force keeping the quadrotor aloft is the vertical thrust T, which has to match the gravitational force of the body, and the variables $\bar{x}$, $\bar{y}$, and $\bar{z}$ denote all the real translational positions possible for the quadrotor [27].

$$\bar{\mathbf{x}} = \begin{bmatrix} \bar{x} & \bar{y} & \bar{z} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \in \mathbb{R}^{12} \tag{3.34}$$

$$\bar{\mathbf{u}} = \begin{bmatrix} mg & 0 & 0 & 0 \end{bmatrix}^T \in \mathbb{R}^{4} \tag{3.35}$$

In addition, the linearization is pursued with regard to an assumption that the validity of the model is preserved for small deviations from the equilibrium point. This leads to a simplification of the non-linear model using small angle approximation (i.e.: $\sin\alpha = \alpha,\ \cos\alpha = 1$), with the result being represented by the simplified state of equations shown in Equation 3.36.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{cases} \dot{p} = \frac{1}{I_{xx}}[\tau_x + (I_{yy} - I_{zz})qr] \\ \dot{q} = \frac{1}{I_{yy}}[\tau_y + (I_{zz} - I_{xx})pr] \\ \dot{r} = \frac{1}{I_{zz}}[\tau_z + (I_{xx} - I_{yy})pq] \\ \dot{u} = rv - qw + g\theta \\ \dot{v} = pw - ru - g\phi \\ \dot{w} = qu - pv - g + \frac{T}{m} \\ \dot{\phi} = p + q\phi\theta + r\theta \\ \dot{\theta} = q - r\phi \\ \dot{\psi} = r + q\phi \\ \dot{x} = w(\phi\psi + \phi) - v(\psi - \phi\phi\theta) + u \\ \dot{y} = v(1 + \phi\psi\theta) - w(\phi - \psi\theta) + u\psi \\ \dot{z} = w - u\theta + v\phi \end{cases} \tag{3.36}$$

Finally performing the first order Taylor expansion on Equation 3.36 at the equilibrium point leads to the linear model of the quadrotor as presented in Equation 3.37, in which the Δ notation is dropped for the right-most side.

$$\dot{\mathbf{x}} \approx \mathbf{f}(\bar{\mathbf{x}},\bar{\mathbf{u}}) + \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}},\,\bar{\mathbf{u}}} \cdot \Delta\mathbf{x} + \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\bar{\mathbf{x}},\,\bar{\mathbf{u}}} \cdot \Delta\mathbf{u} \Rightarrow \dot{\mathbf{x}} = \begin{cases} \dot{p} = \frac{\tau_x}{I_{xx}} \\ \dot{q} = \frac{\tau_y}{I_{yy}} \\ \dot{r} = \frac{\tau_z}{I_{zz}} \\ \dot{u} = g\theta \\ \dot{v} = -g\phi \\ \dot{w} = \frac{T}{m} \\ \dot{\phi} = p \\ \dot{\theta} = q \\ \dot{\psi} = r \\ \dot{x} = u \\ \dot{y} = v \\ \dot{z} = w \end{cases} \tag{3.37}$$

An alternative to the above set of linear equations is illustrated in Equation 3.38, which will be utilized in the design of the state space controller LQR.

$$\dot{\mathbf{x}} = \begin{cases} \ddot{\phi} = \frac{\tau_x}{I_{xx}} \\ \ddot{\theta} = \frac{\tau_y}{I_{yy}} \\ \ddot{\psi} = \frac{\tau_z}{I_{zz}} \\ \ddot{x} = g\theta \\ \ddot{y} = -g\phi \\ \ddot{z} = \frac{T}{m} \end{cases} \tag{3.38}$$

3.9 Thrust Mixer

The system inputs are the thrust and the three torques corresponding to the rotations around the body-frame axes and they depend on the angular speeds of the four motors, as shown in Section 3.4. Equations3.39, where we denote the constant thrust mixing matrix by **P**, elucidates the underlying correlations:

$$\mathbf{u} = \begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \underbrace{\begin{bmatrix} k_T & k_T & k_T & k_T \\ -k_T L \cos\frac{\alpha}{2} & -k_T L \cos\frac{\alpha}{2} & k_T L \cos\frac{\alpha}{2} & k_T L \cos\frac{\alpha}{2} \\ -k_T L \sin\frac{\alpha}{2} & k_T L \sin\frac{\alpha}{2} & k_T L \sin\frac{\alpha}{2} & -k_T L \sin\frac{\alpha}{2} \\ -b_\tau & b_\tau & -b_\tau & b_\tau \end{bmatrix}}_{\mathbf{P}} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{3.39}$$

3.10 Parameter Estimation

Mass and Moment of Inertia

The Newton-Euler equations presented in Section 3.5 have been utilized to build a SIMULINK model of the quadcopter. In order to successfully monitor the behaviour of the open-loop system under different inputs and eventually devise adequate controllers for position and attitude, the mass m and inertia $\mathbf{I}_\mathcal{B}$ of the UAV had to be estimated. In the case of the former parameter, it was simply a digital scale that was required. For the elements of the inertia tensor however, calculations had to be performed assuming some prior approximations. Regarding the UAV as a cuboid-shaped solid body with a constant density function $\rho(x, y, z) = \rho$, and measurable dimensions x_b, y_b and z_b, the corresponding body-frame inertia tensor is, in view of the contents of Appendix B, the one in Equation 3.40.

$$\mathbf{I}_\mathcal{B} = \frac{m}{12} \begin{bmatrix} y_b^2 + z_b^2 & 0 & 0 \\ 0 & x_b^2 + z_b^2 & 0 \\ 0 & 0 & x_b^2 + y_b^2 \end{bmatrix} \tag{3.40}$$

Different and arguably more accurate geometric approximations are carried out in [53] and [54]. In the first case, the quadrotor is modelled as a sphere with a known radius and the motors are reduced to dimensionless mass points, while the second paper simplifies the airframe to two cylinders forming a cross structure, while also considering the masses and the moments of inertia of the motors and propellers. Even more exact methods for finding the values in the inertia tensor have been utilized in other works, such as direct extraction from a CAD model of the quadrotor [24, 34], or experimental estimation using the pendulum method [55]. However, building a detailed 3D model or performing complex experiments and geometric approximations to obtain very accurate estimations was deemed rather too time consuming. The approximation errors resulting from the cuboid method are expected to be compensated by the controller.

Thrust Mixer and Motor Control Constants

Additional constants were necessary to establish the entries of the thrust mixing matrix $\mathbf{P}$ appearing in Equation 3.39, as well as to link the ESC's signal pulse widths to the angular speeds. The parameters L and α, necessary for calculating the moment arms for τ_x and τ_y, were effortlessly measured with a scale and a protractor. On the other hand, k_T and b_τ, the constants describing the aerodynamic characteristics of the rotors, had to be identified experimentally. One motor, to which the corresponding propeller was attached, was spun by supplying a range of voltages in an attempt to cover as much as possible of its operating range.

The experiments were conducted utilizing the Series 1585 Dynamometer and Thrust Stand [56]. Under the reasonable assumption that all the BLDC motors are identical, only one of them was connected to the testing platform and was used for estimation purposes. The platform is equipped with sensors capable of measuring, among other parameters, the width of the applied PWM signal, the angular speed of the motor, the generated thrust and the reaction torque. The conducted experiment consisted of running a preexisting script - through the testbench's dedicated software package - that would steadily increase and then decrease the pulse width of the PWM signal sent to the ESC, between some user-selected boundaries. Employing the standard PWM protocol, the limits were manually set to 1000μs (the arming pulse width) and 1800μs. It is also important to specify that the battery was fully charged (a voltage of $12.6V$ was measured at its terminals) at the juncture when data was being collected.

The following three indispensable relation needed to be discovered via measurements: F_i as a function of Ω_i^2, τ_z as a function of Ω_i^2 and pulse width as a function of Ω_i, which was done by curve fitting. Upon a preliminary look at the data, for the ascending pulse widths, it was found that, if all motors were to produce the maximum thrust (corresponding to a pulse width of 1800μs), the drone would accelerate upwards at a rate of 13m/s^2, assuming zero Euler angles. This was found unnecessarily high, and for safety reasons it was decided to limit the maximum acceleration to the more manageable value of 8m/s^2. With this latest upper boundary, a new maximum thrust and thus a new maximum angular speed were soon identified. Moreover, lower limits were found as well, since the low-voltage dead bands of the BLDC motors had to be acknowledged and avoided during flight. In consequence, the data was truncated and the curve fitting was carried out only for the operating range $\Omega_i \in [\Omega_{i,min}, \Omega_{i,max}]$, where $\Omega_{i,min} = 311.5$rad/s and $\Omega_{i,max} =$ 1812.7rad/s. This saturation was also implemented in software, on the MCU.

The gathered data after truncation, is showcased in Figures 3.7, 3.8 and 3.9. In each of the three cases, a least squares straight line approximation (see [57]) has been performed on the data points, yielding a static relationship between the two parameters plotted against each other, as also done in [16].

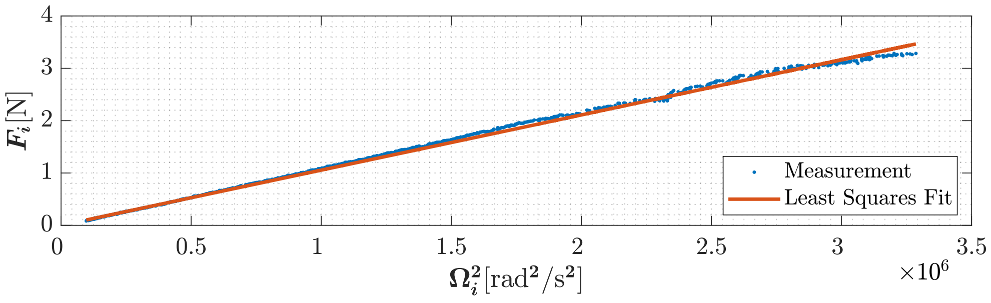


Figure 3.7: Relationship Between Ω_i^2 and F_i

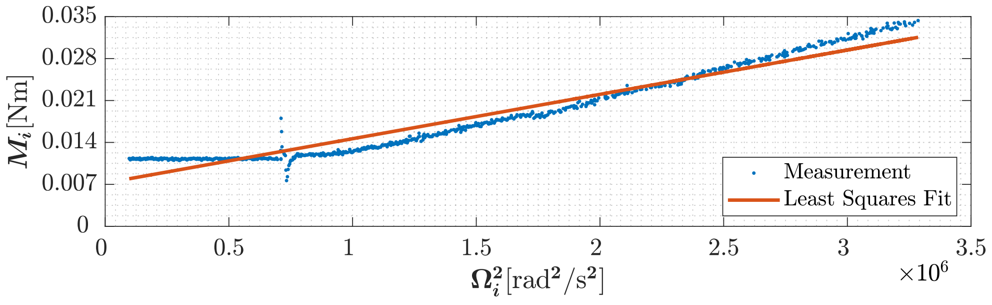


Figure 3.8: Relationship Between Ω_i^2 and M_i

The lines of best fit do not pass through the origin. In the first two scenarios, this means that a zero value of either thrust or torque would correspond to some non-zero angular speed, contrary to what Section 3.4 suggests. For F_i, a non-zero y-intercept would lead to the need of adding some offset to the right-hand side of Equation 3.39. In the case of M_i, due to the definition of τ_z, it does not matter if we have non-zero y-intercepts, as they would cancel each other out. This remarks conclude the method of finding k_T and b_τ. Once known, the numerical values, together with those of L and α can be substituted into the $\mathbf{P}$ matrix that dictates the thrust mixing. Notice that the entries in the vector of

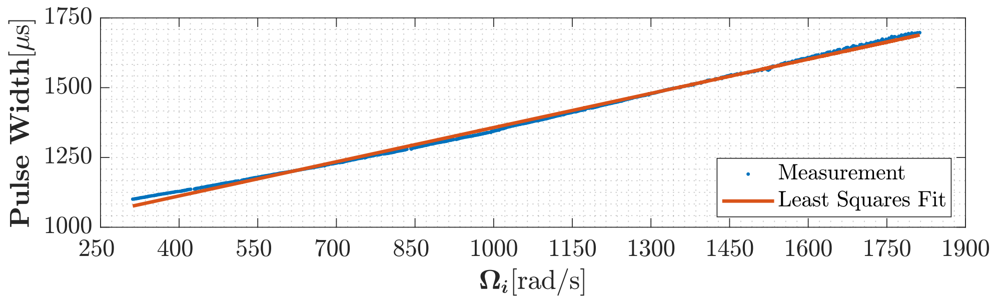


Figure 3.9: Relationship Between Ω_i and the Pulse Width of the Signal Given to the ESC

squared angular speeds are uniquely identified and can be easily calculated by multiplying a given set of values in $\mathbf{u}$ with the inverse of $\mathbf{P}$, i.e.: $\mathbf{P}^{-1}$, provided the matrix $\mathbf{P}$ is non-singular [58]. Checking the value of $\det(\mathbf{P})$ in MATLAB, it was found that the thrust mixing matrix is indeed invertible. Thus, for any $\mathbf{u}$ supplied by the controller, Equation 3.41 always holds:

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \mathbf{P}^{-1}\mathbf{u} \tag{3.41}$$

If any of the entries in the solution vector happens to be negative, the minimum speed $\Omega_{i,\,min}$ is utilized as a motor command instead. The relationship between the angular speeds and the pulse widths required to produce them (in the standard PWM protocol) is given in Equation 3.42. On the microcontroller, commands are sent using the Oneshot125 ESC protocol. The preceding formula is first used to convert desired angular speeds to pulse widths compatible with standard PWM, and a simple mapping is subsequently employed to obtain their correct Oneshot counterparts.

$$PW(\Omega_i) = 0.4\Omega_i + 948.9 \tag{3.42}$$

Table 3.1 lists all the constants that have been estimated.

Table 3.1: Constant Parameters of the UAV

Parameter	Estimated Value
m	0.777kg
I_{xx}	0.0067kgm^2
I_{yy}	0.0059kgm^2
I_{zz}	0.0116kgm^2
L	0.11m
α	1.439rad
k_T	$1.0547 \cdot 10^{-6}$Ns^2/rad^2
b_τ	$7.4038 \cdot 10^{-9}$Ns^2/rad^2

3.11 Open-Loop Response

After the unknown physical parameters have been identified, modeling the plant in SIMULINK was the next logical step. Using the sets of equations listed in Section 3.7, a linear, as well as a nonlinear model, have been constructed. The purpose of the present section is to briefly discuss the differences between the two.

The nonlinear model is too complicated to be made using discrete SIMULINK blocks. Therefore, it was built by implementing the nonlinear quadrotor equations given by Equation 3.31 through MATLAB functions within the SIMULINK environment. Since that is the case, there is little point in showing the utilized structure, for it is not intelligible without direct access to the *.slx* file. The linearized decoupled quadrotor equations (Equations 3.37 and 3.38) can be, however, easily represented in block diagram format, as shown in Figure 3.10.

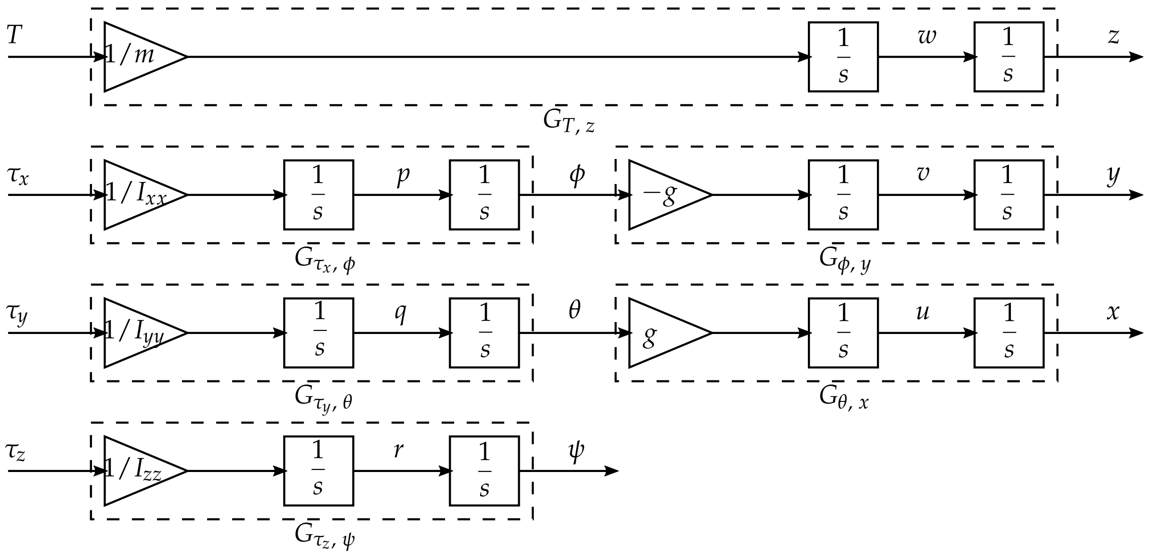
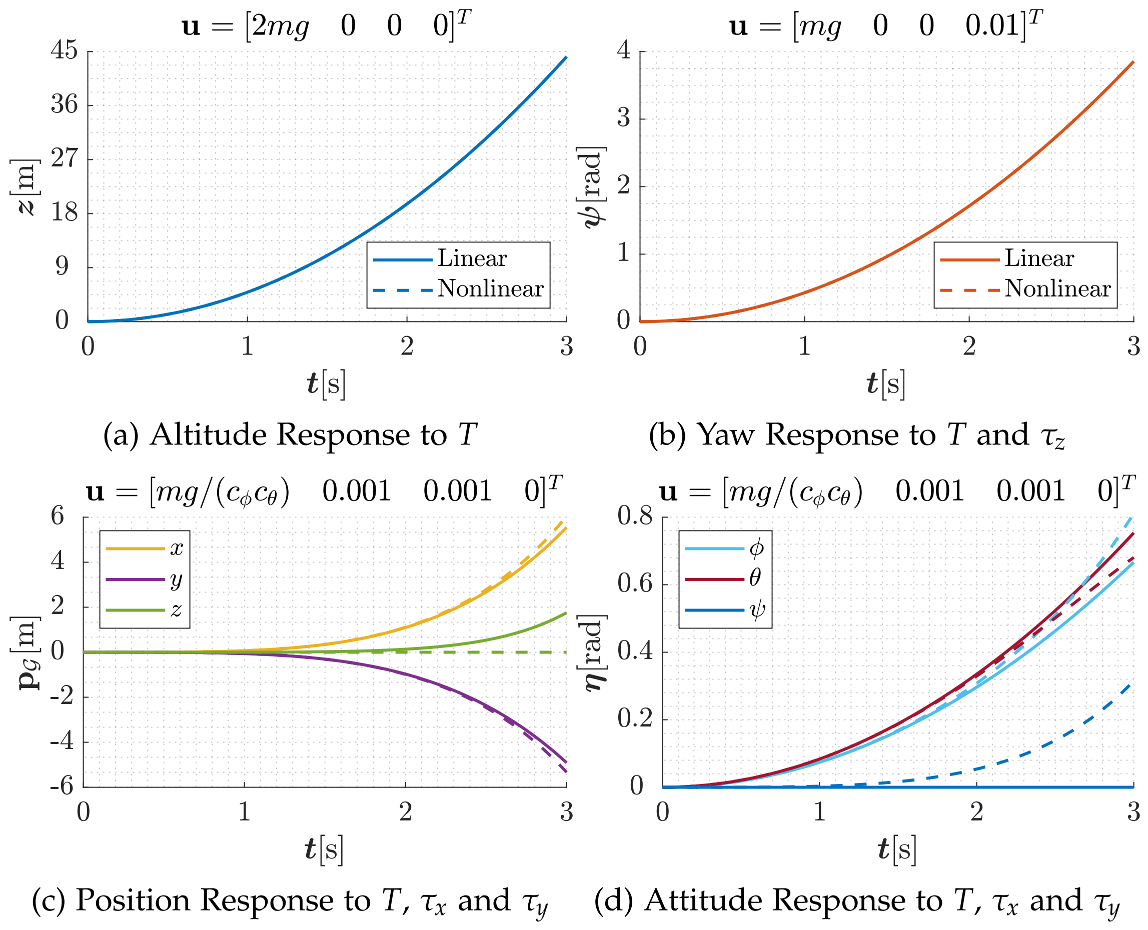


Figure 3.10: Block Diagram of the Linear Decoupled Quadrotor Model

It is important to mention that all the inputs and states in this block diagram denote the deviations from the hovering condition. This is especially relevant for T, which is actually the result of subtracting $\bar{T} = mg$ from the real thrust and should be accounted for when comparing the linear and nonlinear responses to inputs involving thrust. In order to assess the similarity between models, we supply different vectorial excitation inputs $\mathbf{u}$ close to $\bar{\mathbf{u}}$ and observe how some relevant states respond. Figure 3.11 illustrates precisely that. The exact input utilized in each studied case is placed as a sub-figure title.

It is quite apparent from Figure 3.11(a) that, when only an upward force higher than gravity, is inserted into the system, both models behave very similarly and the altitude increases exponentially. When an additional entry corresponding to the torque about the $z_{\mathcal{B}}$-axis is considered (Figure 3.11(b)), ψ



(a) Altitude Response to T

(b) Yaw Response to T and τ_z

(c) Position Response to T, τ_x and τ_y

(d) Attitude Response to T, τ_x and τ_y

Figure 3.11: Open-Loop Response of the Linear and Nonlinear Systems

responds in an exponential manner as well, and the curves are again superimposed. Nevertheless, differences start to appear once we remove τ_z, but add some small constant τ_x and τ_y instead. Thrust is also being manipulated in such a manner in order not to let the drone lose altitude due to the induced rolling and pitching. In Figure 3.11(c) and (d), the simulation is performed for small angular deviations. In this case, due to the approximations made, one can remark that the linear roll and pitch have a product that start equal to but slowly becomes smaller than that of their nonlinear counterparts. This means the dynamically calculated thrust will eventually be greater than the required one, which is exactly the observed behaviour in the linear case, while the nonlinear altitude stays at 0. It is also worth noting that the angular coupling created by the $\mathbf{W}(\boldsymbol{\eta})$ matrix forces the yaw to change even when $\tau_z = 0$ in the nonlinear model, while the decoupling prevents that from happening if we look at the linear scenario. Finally, we also observe from Figure 3.11(c) that the evolution of the x and y position is very similar in both situations.

Stability Analysis

Based on the linear block diagram in Figure 3.10, let us define the following continuous-time plant transfer functions in Equation 3.43. They can also be obtained by taking the Laplace transform of the system in Equation 3.38:

$$\begin{aligned} G_{\tau_x,\,\phi} &= \frac{1}{I_{xx}s^2} \qquad G_{\tau_y,\,\theta} = \frac{1}{I_{yy}s^2} \qquad G_{\tau_z,\,\psi} = \frac{1}{I_{zz}s^2} \\ G_{T,\,z} &= \frac{1}{ms^2} \qquad G_{\phi,\,y} = -\frac{g}{s^2} \qquad G_{\theta,\,x} = \frac{g}{s^2} \end{aligned} \tag{3.43}$$

All the preceding transfer functions include double integrators, which means that they have two poles at the origin of the s-plane. Therefore, according to [59], they are all unstable and controllers are needed to achieve stability and any desired behaviour.

3.12 Model Validation

The preceding section compared the linear and nonlinear open-loop quadrotor models and confirmed a behaviour resemblance around the operating point. Notwithstanding, it has not been so far demonstrated that any of the models accurately describe the dynamics of the real UAV system. Model validation is a procedure consisting of a comparison between the outputs of the model and actual system, when subjected to the same input. Differences can then be evaluated and, based on the purpose of the model, the mathematical representation can be modified accordingly [60].

Strictly validating the model of the quadcopter around the operating point in an open-loop configuration is rendered impossible by the inherent instability of the system. The absence of an already-existing controller that can ensure the attitude and altitude stabilization of the physical quadcopter system makes closed-loop model validation unachievable at this point as well. In view of these, a controller will be developed first and model validation can then be performed in consideration to the closed-loop response.

Chapter 4

Sensors and Signal Processing

4.1 Attitude Estimation

Inertial Measurement Unit (IMU)

The used IMU consists of an accelerometer, a gyroscope and a magnetometer. It is an essential element of the quadrotor as it gives information on linear acceleration, angular velocity and magnetism in three body frame axes, which one can use to estimate the orientation of the drone [61]. Below, each of these sensors are explained in enough detail to comprehend their advantage and disadvantage over each other in terms of attitude estimation.

Accelerometer

An accelerometer measures three-axis acceleration and its output follows the Equation 4.1 where a_m is the measured acceleration and m is the body mass [62]. The measured acceleration is with respect to the gravity and thus it is seen in the equation that gravitational force is subtracted from the sum of all forces on the body [63]. However, the utilised IMU considers positive $\hat{\mathbf{z}}_{\mathcal{G}}$ to point downwards which is in contrast to our convention where we chose it to point upwards. Therefore, we insert the negative of the already-defined gravitational force into the equation.

$$\mathbf{a}_m = \frac{1}{m}\left[\mathbf{F}_{\mathcal{B}} - {}^{\mathcal{B}}_{\mathcal{I}}\mathbf{R}\left(-\mathbf{F}_g\right)\right] \tag{4.1}$$

During the steady state of hovering, $\mathbf{F}_{\mathcal{B}}$ becomes equal to zero and the accelerometer measurements in the body frame takes the form in Equation 4.2 [64]:

$$\mathbf{a}_m = \frac{1}{m}{}^{\mathcal{B}}_{\mathcal{I}}\mathbf{R}\mathbf{F}_g = \begin{bmatrix} c_\psi c_\theta & c_\theta s_\psi & -s_\theta \\ c_\psi s_\phi s_\theta - c_\phi s_\psi & c_\phi c_\psi + s_\phi s_\psi s_\theta & c_\theta s_\phi \\ s_\phi s_\psi + c_\phi c_\psi s_\theta & c_\phi s_\psi s_\theta - c_\psi s_\phi & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = \begin{bmatrix} g s_\theta \\ -g c_\theta s_\phi \\ -g c_\phi c_\theta \end{bmatrix} \tag{4.2}$$

Extracting roll and pitch angles from three-axis acceleration measurements during steady state is now straightforward as shown in Equation 4.3 where the notation "^" signifies estimated angles.

$$\hat{\phi} = \arctan\left(\frac{a_{m,y}}{a_{m,z}}\right) \qquad \hat{\theta} = \arcsin\left(\frac{a_{m,x}}{g}\right) \tag{4.3}$$

Using only an accelerometer, roll and pitch angles can be estimated in this manner. However, the yaw angle cannot be determined in the same fashion because the accelerometer measurements do not depend on yaw, as seen in the result of Equation 4.2. It is important to note that we assumed $\mathbf{F}_{\mathcal{B}}$ to be zero, thus we did not take vibration and other forces into account. In reality, these forces will introduce high frequency noise and have direct influence on the estimated angles [65, 66]. It is also best practice to place the accelerometer close to the center of rotation as much as possible so that the measurements are least affected by rotations.

Gyroscope

A gyroscope measures three-axis angular velocity, $\boldsymbol{\omega}_{\mathcal{B}}$, and can be used to estimate all three Euler angles. Knowing that our quadrotor will need to accelerate to reach its reference position, one upside of using a gyroscope instead of an accelerometer is that it is not affected by external forces and, therefore, provides more reliable measurements under acceleration [67]. In order to calculate the orientation of the quadrotor using a gyroscope, we first need to obtain the rate of change of Euler angles from its angular velocity output due to apparent reasons explained in Section 3.3 [62]. Equation 3.11 gives us the Euler angle rates which we can integrate to estimate the Euler angles. Equation 4.4 is the mathematical description of this calculation where the sampling period is fixed at T_s and " $^+$ " indicates the new estimate.

$$\hat{\boldsymbol{\eta}}^+ = \hat{\boldsymbol{\eta}} + T_s \dot{\boldsymbol{\eta}} \tag{4.4}$$

This way of estimating Euler angles using a gyroscope is immune to propeller vibration and any sort of external forces. Nonetheless, the sensor is inherently noisy. As we iteratively sum the measurements to estimate the Euler angles, we are actually also introducing error to our estimation. Over time, the error will accumulate and the Euler angle estimates will drift away from the reality [65].

Magnetometer

A magnetometer measures the geomagnetic field $\mathbf{B}_m$ in three axes and is used to obtain a yaw estimate. If no magnetic distortion is present in the environment and the sensor is rotated around its axes in all directions, its measurements will lie on the surface of a sphere centered at the origin with a radius being equal to the strength of the geomagnetic field [67]. A magnetic disturbance, on the other hand, will distort these spherical measurements and, hence, result in less accurate yaw estimates [65]. Two main sources of distortion are hard-iron and soft-iron disturbances. A hard-iron source, such as an electric motor or a wire that has current flow through it, generates its own magnetic field in addition to the geomagnetic field and the sensor measures the sum of both. This disturbance due to the hard-iron source shifts the sphere away from the origin. On the other hand, a soft-iron source, such as a screw, is solely a magnetic material without the ability to generate its magnetic field. This type of sources bend the geomagnetic field and turn the spherical measurements into an odd-shaped spheroid. There is nothing one can do to eliminate randomly time-varying magnetic disturbances. However, if the disturbance sources are part of the system and rotate together with the rest of the body such as screws, nuts and motors, then their effect on the magnetometer measurements can be eliminated out with sensor calibration. As given in Equation 4.5, subtracting hard-iron offset vector $\mathbf{K}$ from $\mathbf{B}_m$ and multiplying the result with the inverse of the soft-iron distortion matrix $\mathbf{D}$ would correct the distorted measurements [62].

$$\mathbf{B}_{m,corrected} = (\mathbf{B}_m - \mathbf{K})\mathbf{D}^{-1} \tag{4.5}$$

Complementary Filter

Above we point out the strengths and the weaknesses of the three sensors and explain how their outputs can be used to estimate the Euler angles. In short, a gyroscope provides resistance to vibrations which is a feature that we want to acquire, but fails to maintain long-term stability due to drift in its measurements [65]. On the other hand, an accelerometer's attitude information comes with long-term stability. However, its measurements are noisier. A magnetometer provides an additional yaw estimate along with the gyroscopic one but it requires calibration for higher accuracy[1]. Due to these reasons, it is not possible to utilize a single sensor and accurately estimate the Euler angles. One way to do so is to combine their outputs and benefit from their separate advantages. Sensor fusion with a complementary filter will exhibit the best of all and produce accurate angle estimates that avoid long-term angular drift and are resistant to vibrations [68].

[1]Due to time constraints, the magnetometer has not been calibrated and used in this project. Instead, the gyroscope is utilized to obtain the yaw angle, which did not drift almost at all during the short hovering test sessions.

The basic building blocks of a complementary filter are a high-pass and a low-pass filter [62]. The filters are used to eliminate the sensor noise at different frequencies. Using this filter, one can complement reliable low frequency signal from the accelerometer and reliable high-frequency signal from the gyroscope for better estimation results. This idea is demonstrated in Figure 4.1 where x_a and x_b denote measurements from the gyroscope and the accelerometer, respectively.

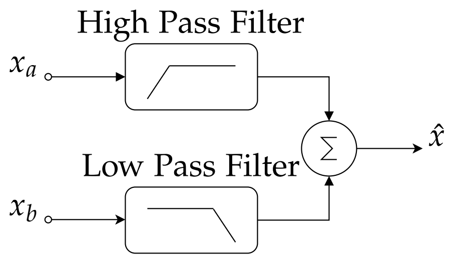


Figure 4.1: Complementary Filter Block Diagram

Raw IMU Measurements

Before the design procedure of the filters, it is important to examine the raw IMU measurements and confirm their behaviour. On the basis of several tests, it is observed that arbitrary offsets in IMU angle measurements that are large enough to make the quadrotor unstable are present. In order to calculate them dynamically, raw IMU measurements are summed for five seconds during the arming session of the motors and divided by the number of samples. The offsets are then subtracted from the sensory output. In Figure 4.2, one can see the raw accelerometer and gyroscope measurements for ϕ and θ, after offsets are removed. As discussed previously, accelerometer gives extremely noisy measurements while gyroscope's output drifts in the long run.

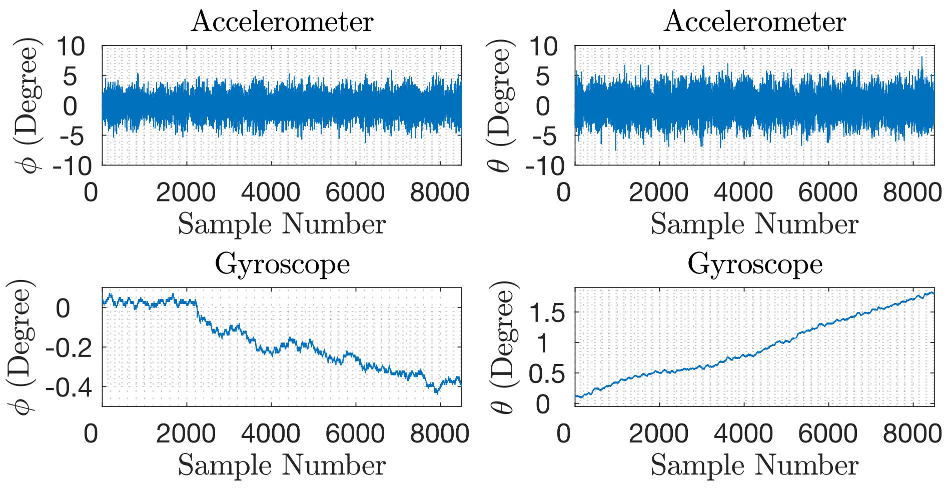


Figure 4.2: Accelerometer and Gyroscope Raw ϕ and θ Measurements - Offsets Are Removed

Low-Pass Filter Design

In order to determine the cut-off frequency, f_c, for the continuous-time low-pass filter, FFT of the raw accelerometer measurements are obtained using MATLAB, shown in Figure 4.3. It is seen that the noise can be found at frequencies as low as 1Hz and we anticipate that the cut-off frequency needs to be less than that.

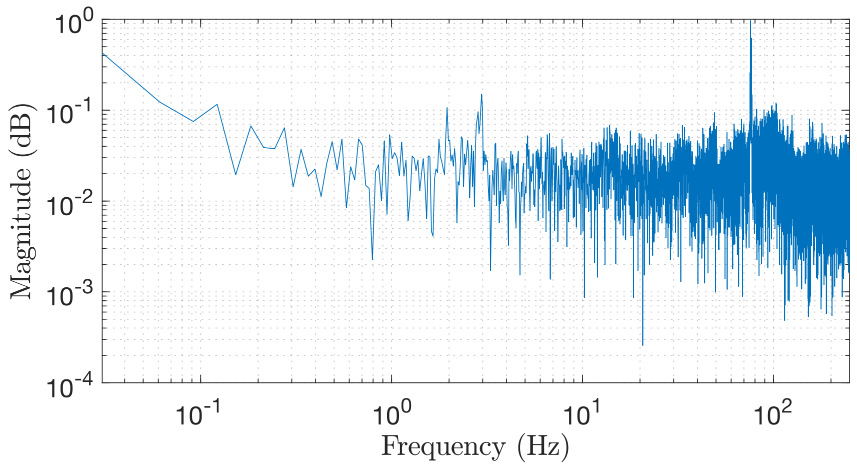


Figure 4.3: FFT of Raw Accelerometer Measurements - Quadrotor Is Stationary

In the process of determining the cut-off frequency, several filters that we designed using SPTool on MATLAB have been applied on the raw accelerometer measurements. The results are given in Figure 4.4. The best filtering is obtained when f_c is chosen to be 0.5Hz while significant level of noise could still be observed with the other filters.

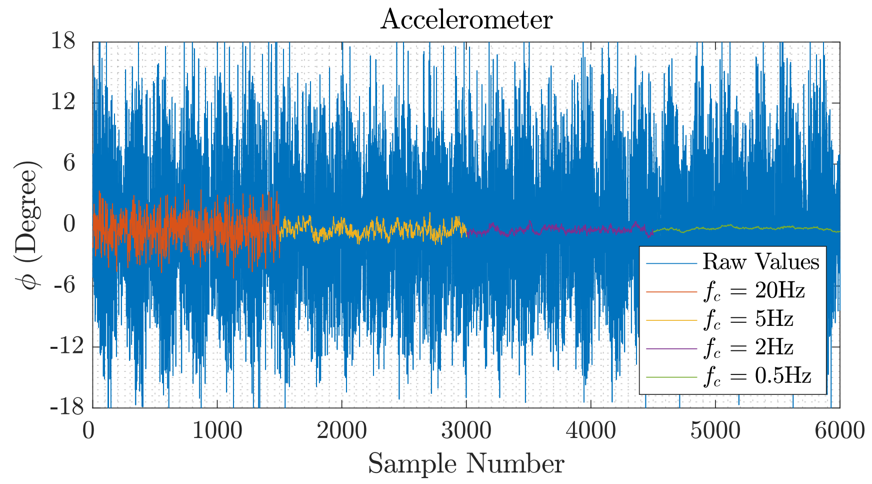


Figure 4.4: Filtering Raw Accelerometer Measurements Using Different Continuous Time Filters

Due to its ease of implementation on a microcontroller, we opted for implementing Direct-Form 1 IIR filter type [68]. After having finalized the filter design using SPTool using a sampling frequency of 500Hz, we saved it as a z-domain transfer function in the form demonstrated in Equation 4.6 where n and d stand for "numerator" and "denominator".

$$LPF(z) = \frac{n_1 z + n_2}{d_1 z + d_2} \tag{4.6}$$

The linear constant-coefficient difference equation (LCCDE) used to implement the filter on the microcontroller is found in Equation 4.7.

$$a_0 y[n] = \sum_{i=0}^{1} b_i x[n-i] - \sum_{j=1}^{1} a_j y[n-j] \tag{4.7}$$

where the coefficients are given in Equation 4.8:

$$\begin{cases} a_0 = 1 \\ a_1 = d_2/d_1 \\ b_0 = n_1/d_1 \\ b_1 = n_2/d_1 \end{cases} \tag{4.8}$$

Figure 4.5 illustrates the filter structure for the used 1^{st} order direct-form 1 IIR filter.

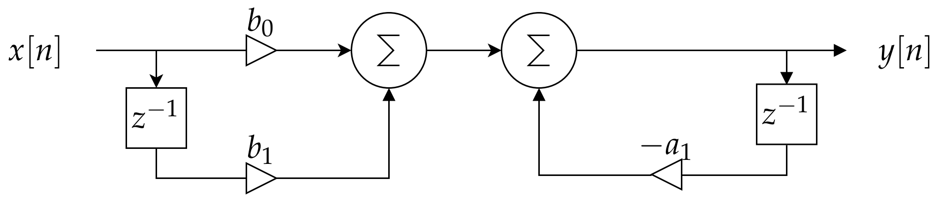


Figure 4.5: 1st Order Direct-Form 1 IIR Filter Structure

High-Pass Filter Design

A feature of complementary filters is the monotonic unity gain in the output such that the transfer function of the sum is an all-pass filter, meaning 0dB gain at all frequencies. In order to maintain this property, the transfer function for the high-pass filter for the gyroscope is calculated by subtracting the low-pass filter transfer function from unity as shown in Equation 4.9. Its discrete form is implemented on the microcontroller in the same manner that is done for the low-pass filter.

$$HPF(z) = 1 - LPF(z) \tag{4.9}$$

Complementary Filter Results

Figure 4.6 plots the magnitude response of the complementary filter across the frequency spectrum [65]. As it is expected, it does not attenuate or amplify the signal but rather passes it at its own magnitude while allowing us to eliminate signals of undesired frequencies from the sensor measurements.

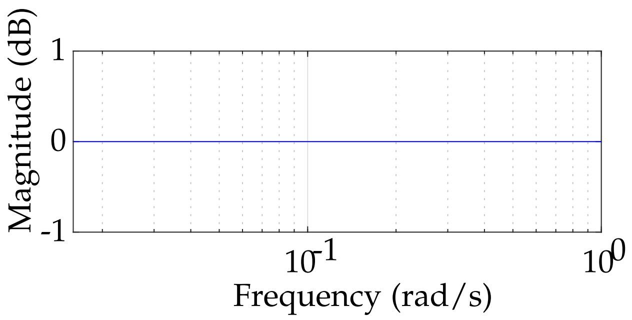


Figure 4.6: Complementary Filter Magnitude Plot

Figure 4.7 plots the raw measurements of both accelerometer and the gyroscope that are fed into the complementary filter. The output of the filtering is adequate because the high-frequency noise originating from the accelerometer and the drift in the gyroscope measurements are eliminated. The complementary filter outputs reliable Euler angle estimates and proves to be performing as intended.

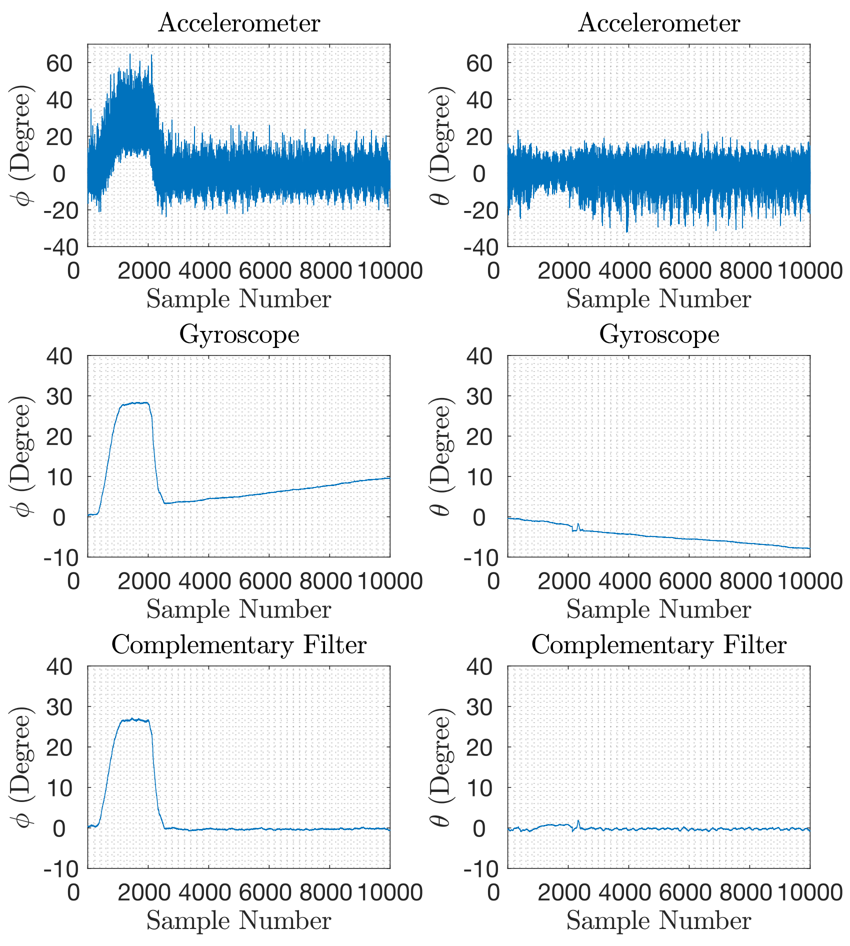


Figure 4.7: Complementary Filter Input and Output Signals

4.2 Position Estimation

Knowing the position of the quadrotor is essential to implementing trajectory control. Having already employed an accelerometer, we would like to question whether it is possible to utilize the measured linear accelerations to estimate the quadrotor position.

The mathematical equation of the accelerometer's output was given in Equation 4.1. Simplifying it further gives us Equation 4.10.

$$\mathbf{a}_m = \mathbf{a}_\mathcal{B} + \frac{1}{m}({}^\mathcal{B}_\mathcal{I}\mathbf{R}\mathbf{F}_g) \tag{4.10}$$

Solving it for $\mathbf{a}_\mathcal{B}$ and then rotating it into inertial frame are shown in steps captured in Equation 4.11

$$\mathbf{a}_\mathcal{B} = \mathbf{a}_m - \frac{1}{m}({}^\mathcal{B}_\mathcal{I}\mathbf{R}\mathbf{F}_g) \qquad \mathbf{a}_\mathcal{I} = {}^\mathcal{I}_\mathcal{B}\mathbf{R}\mathbf{a}_m - \frac{1}{m}\mathbf{F}_g \tag{4.11}$$

$\mathbf{a}_\mathcal{I}$ can be integrated twice to estimate the position in the inertial frame as stated in Equation 4.12 where the sampling period is fixed at T_s and $\mathbf{v}_\mathcal{I}$ and $\mathbf{r}_\mathcal{I}$ represent linear velocity and position in the inertial frame respectively.

$$\mathbf{v}_\mathcal{I}[n+1] = \mathbf{v}_\mathcal{I}[n] + T_s\mathbf{a}_\mathcal{I}[n] \qquad \mathbf{r}_\mathcal{I}[n+1] = \mathbf{r}_\mathcal{I}[n] + T_s\mathbf{v}_\mathcal{I}[n] \tag{4.12}$$

It seems that it is viable to estimate position using the accelerometer. However, the inaccuracies of the sensor are likely to introduce error. Even if we could tolerate those, there is another issue which is the orientation of the sensor. In an ideal scenario where the reference frame of the accelerometer and the quadrotor are perfectly aligned, measured accelerations can be assumed to reflect the ones of the quadrotor. However, if the sensor is misaligned even slightly, linear acceleration parallel to one of the body frame axes might result in sensor readings in two or three axes [66]. This small error due to the misalignment will accumulate in time and produce extremely faulty velocity and position estimates. Since we cannot know the orientation of the accelerometer with respect to the quadrotor body frame with high accuracy, implementing this approach to estimate the position was deemed not feasible. In order to illustrate the effect of sensor misalignment, we wrote a MATLAB script that displays quadrotor's position and orientation in real-time according to IMU measurements, based on the explained manner of position estimation. Figure 4.8 is the output of the script when the Euler angles and linear accelerations are set to zero.

As part of the experiment, we purposefully misaligned the accelerometer and accelerated the drone in $\hat{\mathbf{x}}_\mathcal{G}$ direction. By altering the degree of misalignment, we could see the effect of mismeasured acceleration over the position estimates. The estimated position after 10s via this experiment are given in Table 4.1. As it is seen, a slight misalignment as small as $\sim 1^\circ$ results in $\sim$9m drift over the course of 10s, which is something we cannot tolerate.

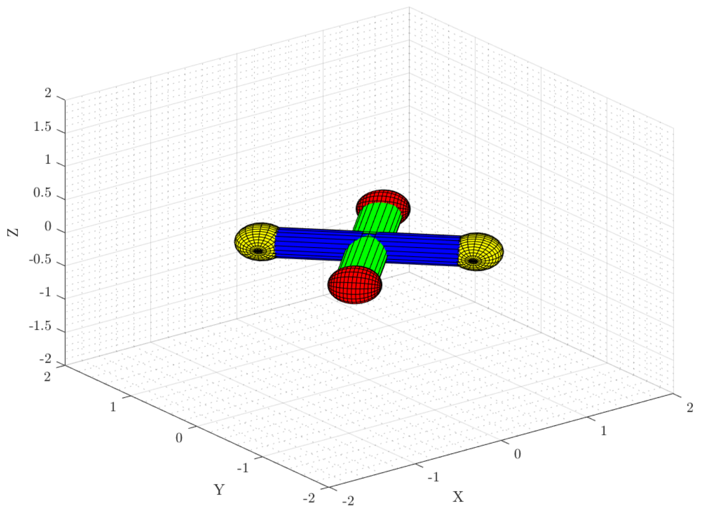


Figure 4.8: Real-time MATLAB Animation of the Quadrotor Based on IMU Measurements

Table 4.1: Acceleration and Position Errors Caused Due to Sensor Misalignment

Sensor Misalignment	Acceleration Error	Position Error after 10 seconds
∼1°	∼0.018 m/s^2	∼9 m
∼2°	∼0.344 m/s^2	∼17.2 m

Time-of-Flight (ToF) Laser-Ranging Sensor - 1D LiDAR

To estimate the position of the quadrotor reliably, we utilized three ToF ranging sensors. They are placed on the quadrotor looking in the positive direction of the body frame axis. The distance measurements done with ToFs are observed to be very less noisy and very accurate within their range. Therefore, we did not apply any filtering. However, one problem we faced was that when the quadrotor rolled, pitched or yawed the sensors also rotated and started measuring distances to other objects. Rotations around a certain axis altered the distance measurements around the other two axes while, in reality, these distance were kept the same. To resolve this problem, we took the Euler angles into consideration and calculated the actual distances from the measured ones. The theoretical effect of rotations around all three axes and the relationship between the actual and the measured distances are represented in Figure 4.9.

In order to see if this approach really works, we conducted a test where we manually changed the yaw angle of the quadrotor while keeping its position in 3D space unchanged. As it is seen in Figure 4.10, measured y-distance is subject to change when the yaw alters, giving a false distance information. On the other hand, the y-distance calculated based on the proposed manner remain around the same level, giving a better position estimation.

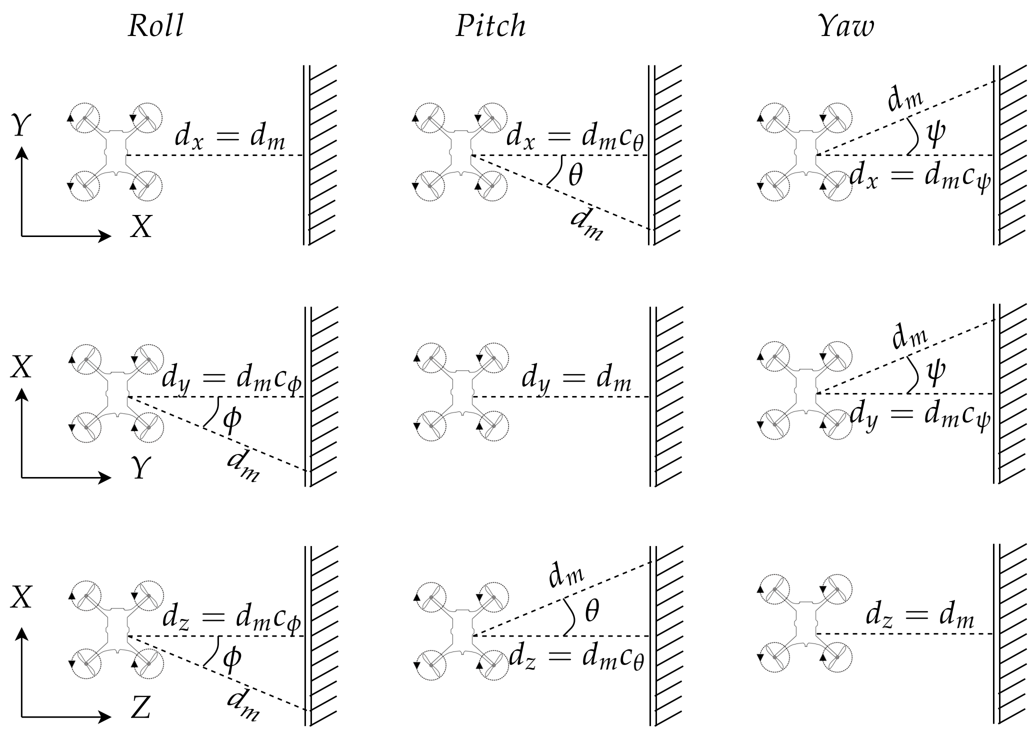


Figure 4.9: The Effect of Euler Angles on Distance Measurements

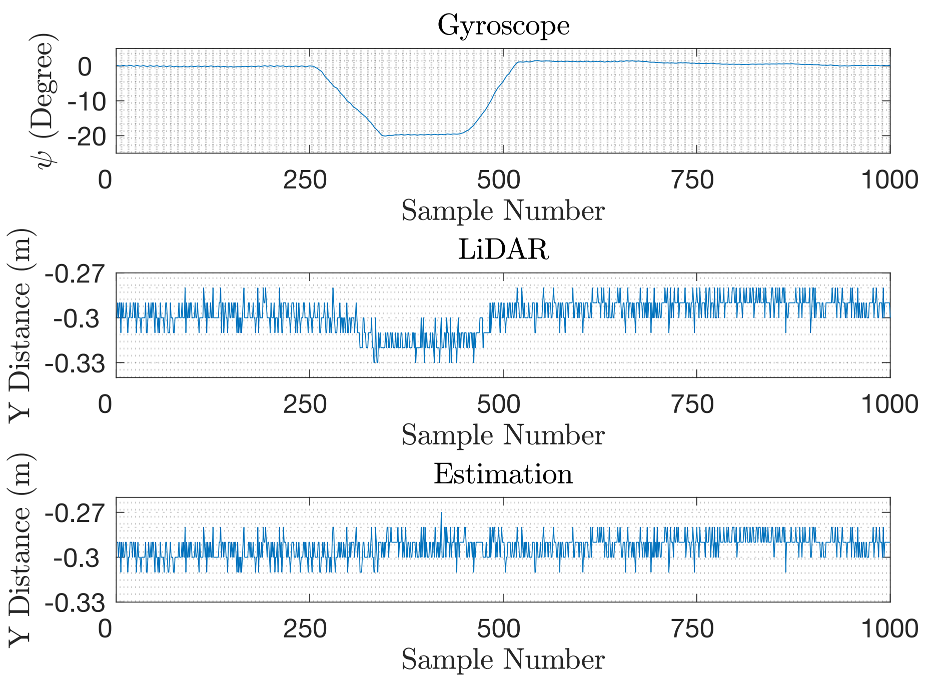


Figure 4.10: Comparison of Measured and Calculated y-Distances

Chapter 5

Controller Design

5.1 Proportional Integral Derivative (PID) Control

The initial control strategy adopted for regulating the quadcopter's altitude, *xy*-position and attitude was PID, chiefly for its simplicity, demonstrated efficacy and prevalence in the specialized literature. The well-known time-domain equation describing the standard PID controller is given in Equation 5.1 [59]:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt} \tag{5.1}$$

Since the controller was implemented on a discrete machine, it had to be discretized. The preceding continuous-time differential equation can be converted to a linear constant-coefficient difference equation, whose general form is the one in Equation 5.2 [69]:

$$\sum_{k=0}^{N} a_k u[n-k] = \sum_{m=0}^{M} b_m e[n-m] \tag{5.2}$$

The conversion starts by taking the Laplace transform of both sides of Equation 5.1. If we also add a low-pass filter for the derivative term, with cutoff at ω_d, we get the continuous transfer function $C(s)$ of the PID (as presented in SIMULINK) given in Equation 5.3:

$$C(s) = \frac{U(s)}{E(s)} = k_p + \frac{k_i}{s} + \frac{k_d \omega_d}{1 + \omega_d \frac{1}{s}} \tag{5.3}$$

Using the bilinear transform

$$s = \frac{T_s}{2} \frac{1 - z^{-1}}{1 + z^{-1}}$$

where T_s denotes the sampling time, to convert from the s-domain to the z-domain, leads to the discrete transfer function $C(z)$ in Equation 5.4:

$$C(z) = \frac{U(z)}{E(z)} = k_p + \frac{k_i T_s}{2}\frac{1+z^{-1}}{1-z^{-1}} + 2k_d\omega_d \frac{1-z^{-1}}{\omega_d T_s + 2 + (\omega_d T_s - 2)z^{-1}} \tag{5.4}$$

Bringing every term in the right-hand side of Equation 5.4 to a common denominator, rearranging and taking the inverse z-transform on both sides of the resulting relation, one obtains a linear constant-coefficient difference equation (Equation 5.5) for the PID controller:

$$u[n] = \frac{1}{a_0}(-a_1 u[n-1] - a_2 u[n-2] + b_0 e[n] + b_1 e[n-1] + b_2 e[n-2]) \tag{5.5}$$

where the coefficients are given in Equation 5.6:

$$\begin{cases} a_0 = 2T_s\omega_d + 4 \\ a_1 = -8 \\ a_2 = -2T_s\omega_d + 4 \\ b_0 = 4k_p + 2T_s k_i + 4k_d\omega_d + 2T_s k_p\omega_d + T_s^2 k_i\omega_d \\ b_1 = -8k_p - 8k_d\omega_d + 2T_s^2 k_i\omega_d \\ b_2 = 4k_p + 2T_s k_i + 4k_d\omega_d - 2T_s k_p\omega_d + T_s^2 k_i\omega_d \end{cases} \tag{5.6}$$

Complete Control Architecture

A ubiquitous, well-studied method of UAV PID control consists of a hierarchical structure comprising two loops, namely an inner loop for attitude stabilization, and an outer loop responsible for position tracking (i.e.: altitude and *xy*-position), yielding a total of 6 PID controllers. This nested architecture is presented in Figure 5.1.

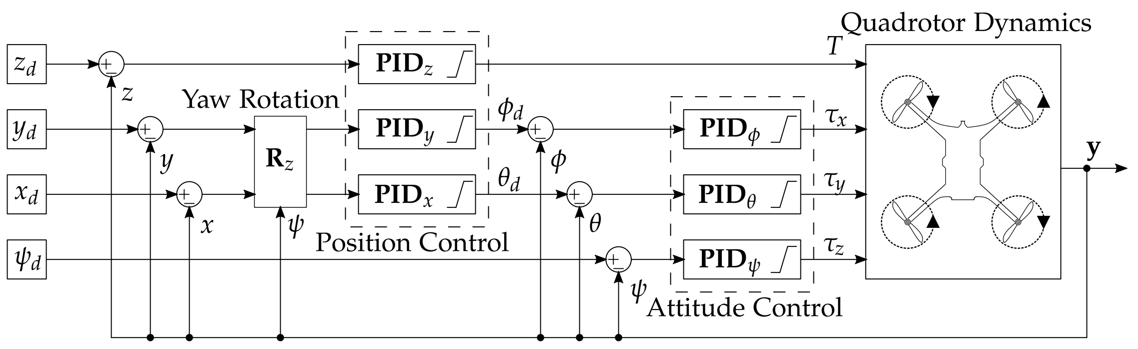


Figure 5.1: PID Hierarchical Control Structure

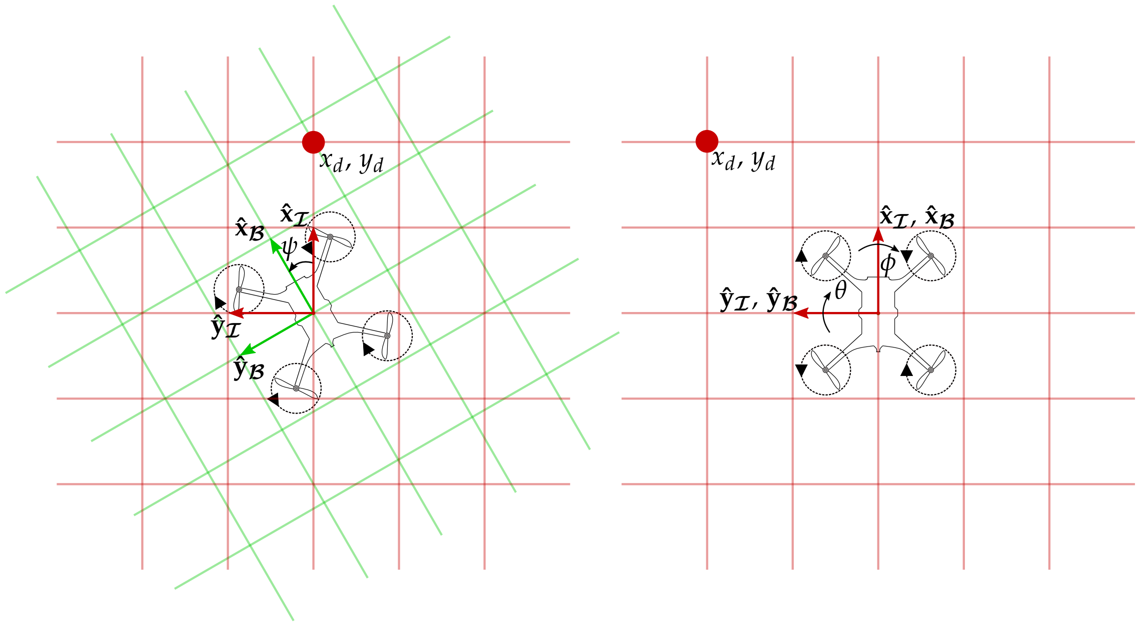
Placing the controllers in the fashion shown in the figure is not at all an aleatory choice. The equations describing the nonlinear system dynamics (Equation 3.31) prove that the translational motion is dependent on the angular one, as the Euler angles influence the position and speed of the quadrotor within the global frame, and therefore also in the body frame. In order to translate in the direction of $\hat{\mathbf{x}}_{\mathcal{B}}$, the UAV should pitch, i.e. rotate around the $\mathbf{y}_{\mathcal{B}}$-axis. Similarly, to move along a line parallel to $\hat{\mathbf{y}}_{\mathcal{B}}$, rolling is necessary. These simple ideas are already building some sort of intuition that the outer PIDs corresponding to the xy-positioning should produce the setpoints for the roll and pitch controllers in the nested attitude loop. As the drone gets closer to the desired location in the plane, the control signals produced by PID_x and PID_y will converge to 0, which is exactly what should also happen to the two aforementioned Euler angles to avoid further movement.

The entity that separates the xy-position errors and the corresponding PID_x and PID_y controllers is a calculation block indicating a matrix multiplication. The matrix in question is the yaw rotation matrix $\mathbf{R}_z(\psi)$ from Equation 3.2 that performs a change of basis from the inertial (or global) frame to the first intermediate Euler frame. The multiplication is a real necessity, as both the desired and the measured position values are expressed in the inertial frame, while the quadrotor may be tilted by some angle ψ with respect to what has been defined as the inertial coordinate system. If that is the case, the controller should be cognizant of the shift in order to produce the correct attitude setpoints.

A graphical representation of the importance of taking the yaw movement into account is shown in the left-hand side of Figure 5.2. Reaching the desired position marked by the red dot in the figure is only a matter of pitching if the UAV's body axes are aligned to the ones of the inertial frame. However, for a nonzero value of ψ, it can be observed that an additional need for a rolling motion emerges. Hence, one may argue that it is actually rather intuitive to include the aforementioned matrix operation in the proposed control structure. (Notice that for $\psi = 0$, the $\mathbf{R}_z(\psi)$ becomes the identity matrix $\mathbf{I}_3$ and thus no change of basis transpires.)

The right half of Figure 5.2 illustrates another subtlety which should be kept in mind when tuning the outer controller in simulation. We assume here, for simplicity, that the $\mathcal{I}$-frame and $\mathcal{B}$-frame are perfectly aligned. According to the convention utilized for defining the sign of the Tait-Bryan angles, a positive pitch tilts the drone forward, meaning that a PID_x controller with positive gains will increase θ_d, given a positive error in x. Notwithstanding, roll is defined as positive when the UAV lowers its right half. From the figure, it is obvious that the quadrotor would actually drift away from the positive global reference (again, the red dot), thus increasing the error, if PID_y had positive gains. Negative controller gains are thus required for position control in the y-direction. This was found more convenient than giving a negative reference y_d in simulation. Consult Figure 3.6 for doubts regarding rotations with positive Euler angles.

Regarding the four controllable inputs to the quadrotor, they are supplied through the other four controllers. Thrust is generated as a result of the effort of PID_z, whereas PID_ϕ, PID_θ and PID_ψ coalesce into the attitude compensator



x_d, y_d
$\hat{\mathbf{x}}_{\mathcal{I}}$
$\hat{\mathbf{x}}_{\mathcal{B}}$
ψ
$\hat{\mathbf{y}}_{\mathcal{I}}$
$\hat{\mathbf{y}}_{\mathcal{B}}$
x_d, y_d
$\hat{\mathbf{x}}_{\mathcal{I}}, \hat{\mathbf{x}}_{\mathcal{B}}$
ϕ
$\hat{\mathbf{y}}_{\mathcal{I}}, \hat{\mathbf{y}}_{\mathcal{B}}$
θ

Figure 5.2: *Left:* Importance of Knowing the Yaw Angle When Tracking an xy-Position Reference; *Right:* Necessity of Negative Gains for PID$_y$

which creates the required torque components of $\boldsymbol{\tau}_{\mathcal{B}}$.

Control Signal Saturation

Saturation plays a pivotal role in the controller's modus operandi, as it restricts the speed with which the controlled outputs can achieve the references, in an effort to take into account in simulation the limited real-world hardware-related system capabilities. The thrust was saturated between $[T_{min}, T_{max}]$, where the limits were taken as four times the upward force generated by one motor (recalculated using the linear least squares fit) for the chosen maximum speed $\Omega_{i,\,max}$ of 1812.7rad/s and minimum speed $\Omega_{i,\,max}$ of 311.5rad/s, respectively. In the case of τ_z, the limits $[-\tau_{z,\,max}, \tau_{z,\,max}]$ have been chosen such that $\tau_{z,\,max}$ is twice the maximum reaction torque at $\Omega_{i,\,max}$, because the motor pairs spin in opposite directions. Saturating boundaries for τ_x and τ_y have been set following the same argument. Another important utilization of saturation is concerned with the control signals ϕ_d and θ_d which act as desired roll and pitch values. Recall from Section 3.3 that the matrix $\mathbf{W}(\boldsymbol{\eta})$ is non-singular for non-steep pitch angels. Since aggressive flight does not constitute the focus of this report, the singularity was avoided by restricting both θ_d and ϕ_d to a relatively small range symmetric about 0 rad. Table 5.1 lists the calculated and chosen, when applicable, saturation boundaries for all the control signals present in the structure of the MIMO PID controller. Even though not explicitly indicated in Figure 5.1, the altitude output was saturated as well, between $[0, \infty]$, for there is no such thing as a (scalar) negative height in the real world.

Table 5.1: Saturation of Hierachical PID Control Signals

Control Signal	T	τ_z	τ_y	τ_x	ϕ_d	θ_d
Lower Saturation	0.41N	-0.049Nm	-0.5Nm	-0.57Nm	$-\frac{\pi}{20}$	$-\frac{\pi}{20}$
Upper Saturation	13.86N	0.049Nm	0.5Nm	0.57Nm	$\frac{\pi}{20}$	$\frac{\pi}{20}$

5.1.1 Altitude and Attitude Control

The tuning procedure for a hierarchical arrangement such as the one in Figure 5.1 supposes that the gains for the inner loop are chosen prior to the ones of the outer loop. In the further discussion of the tuning and simulation results, it is thus natural to look into the inner controllers first. Since PID$_z$ is not influencing the angle references, this is also included in this initial assessment. Omitting the xy-position controllers, the presented PID structure reduces to the one in Figure 5.3. This is the arrangement that was utilized to achieve altitude and attitude stabilization as the first phase of PID implementation.

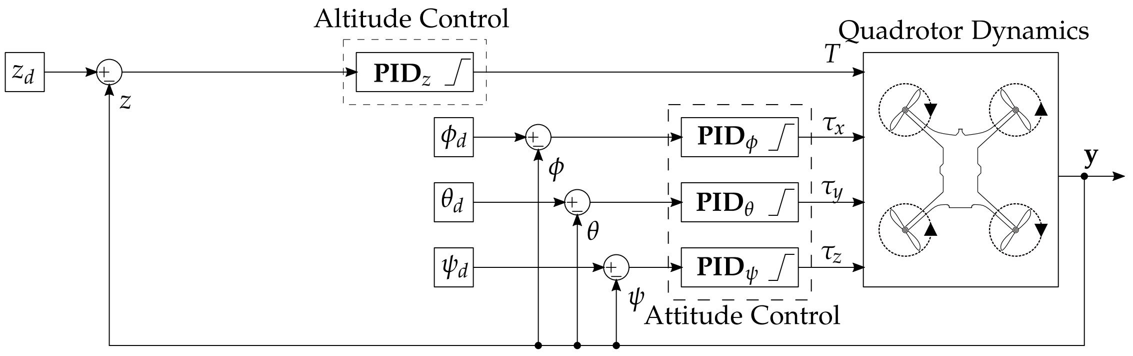


Figure 5.3: PID Altitude and Attitude Control Structure

Tuning a PID controller has been the subject of countless research articles. Apart from the complex and innovative approaches, classical control theory tools, such as root locus or frequency response are available to the designer. In spite of the wide documentation available on these latter tuning methods (e.g.: [59, 70]), employing them when trying to find suitable gains for the linearized version of a highly nonlinear system, only to change these gains later in simulation when control signal saturations and other necessary nonlinearities are considered, can be time consuming. Thus, in this project, the approach taken was a heuristic one, in which the tuning was performed in simulation, directly on the nonlinear model. Also, we first considered the continuous controller case. In order to expedite the manual tuning process, the closed-loop system was simulated employing the immediately-available SIMULINK PID blocks, with an (at first) ideal derivative and clamping chosen as the integrator anti-windup method.

The desired response characteristics were of course short rise and settling times, as well as low overshoot and steady-state error. Speed is especially important in the case of the attitude loop, which would be later placed inside its xy-position counterpart. Automated tuning was able to soon provide some initial gain values, which were further subjected to manual fine tuning. The final gains are given below:

$$\begin{aligned}
\text{PID}_z:&\ \ k_p = 15,\ \ k_i = 8\ \ k_d = 7\\
\text{PID}_\phi:&\ \ k_p = 3,\ \ k_i = 2\ \ k_d = 0.25\\
\text{PID}_\theta:&\ \ k_p = 3,\ \ k_i = 2\ \ k_d = 0.25\\
\text{PID}_\psi:&\ \ k_p = 2,\ \ k_i = 0.5\ \ k_d = 0.4
\end{aligned}$$

Regarding the fact that the parameters of the roll and pitch controllers turned out to be the same in magnitude did not come as a surprise. Table 3.1 demonstrates that the estimated moments of inertia about the airframe's x- and y-axis, respectively, (I_{xx} and I_{yy}) are almost equal, so a controller similarity was expected.

Sample Rate Selection

In the affair of implementing a controller on a discrete machine, sampling represents an important issue. The sampling times were selected based on the sensor capabilities. The IMU is able to produce new measurements quite fast (less than 2ms), while the one-dimensional ToF sensor measuring altitude outputs a value every 30ms. In view of these, we chose the following sampling periods:

$$T_{s,\eta} = 2\text{ms} \qquad\qquad T_{s,z} = 33\text{ms}$$

where $T_{s,\eta}$ corresponds to the attitude loop, whereas $T_{s,z}$ dictates the sampling for the altitude loop. In each case, on an MCU level, reading a new sensor value and performing the necessary PID calculations should be done within the correct available time frame.

Discretization has important effects on the controller performance and thus ensuring correct sampling represented a priority. In [71], the following relationship (Equation 5.7) between the closed-loop bandwidth f_b and the sampling rate f_s is suggested for practical use, to provide smooth time responses:

$$20 < \frac{f_s}{f_b} < 40 \tag{5.7}$$

even though the absolute minimum limit for f_s is $2f_b$, by Nyquist's sampling theorem [71].

In order to simulate, in continuous time, the effects of discretization, which adds delay to the system, a zero-order hold (ZOH) was placed between the continuous controller and the continuous plant model. From [71], we have that the transfer function that approximately models the delay introduced by the ZOH is given by Equation 5.8:

$$H_{d,ZOH} = \frac{2/T_s}{s + 2/T_s} \tag{5.8}$$

The closed-loop bandwidths of the system considered in this altitude and attitude arrangement that suffices hovering stabilization were calculated in MATLAB based on the linear model description in the s-domain (Equation 3.43). The Bode magnitude plots of the closed-loop transfer functions (including the continuous ZOH delay approximation) corresponding to each of the four controlled variables are shown in Figure 5.4. (The plots for the attitude loop are almost superimposed.)

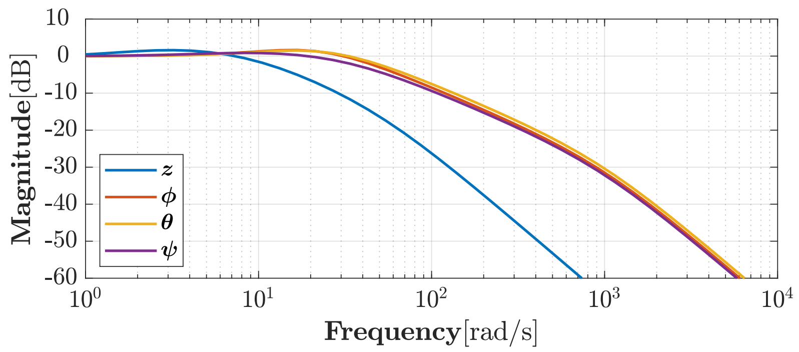


Figure 5.4: Closed-Loop Magnitude Bode Plots for Continuous Altitude and Attitude Control

while the bandwidths (converted to Hz) are listed below:

$$f_{b,z} = 2.01\text{Hz}, \quad f_{b,\phi} = 7.96\text{Hz}, \quad f_{b,\theta} = 8.81\text{Hz}, \quad f_{b,\psi} = 6.44\text{Hz}$$

It is now clear that the attitude loops are sampled sufficiently fast, given $f_{s,\eta} = 500\text{Hz}$. For altitude, we have $f_{s,z} = 30\text{Hz}$, which is 15 times greater than the bandwidth, an arguably satisfactory result.

With the bandwidths calculated, it was also possible to add low-pass filters for the individual PID derivative terms to limit sensor noise amplification [59], with each ω_d being taken as 10 times the corresponding bandwidth (in rad/s), to avoid filtering any useful signal.

Simulation

For hovering, seeking the accurate tracking of a height reference is evidently the end goal, while the desired behaviour for the roll and pitch Tait-Bryan angles is a decay to 0. However, setting a reference of 0 for the latter case does not prove that the attitude controller is able to eventually drive the outputs to this value[1]. Either initial non-zero values for the angles or non-zero angle references should be used for this purpose. We selected the set points $\phi_d = \theta_d = \psi_d = 1$.

[1]Doing this simply prohibits the angles in question from changing, which is essentially equivalent to using the more complex structure in Figure 5.1 with $x_d = y_d = 0$

The simulation results are shown in Figure 5.5, for altitude, and Figure 5.6, for attitude. In all pictures, the responses for both the continuous and discrete controller are illustrated, with the employed discretized PID being the one presented in Equation 5.4. We emphasize again that all responses correspond to the nonlinear model.

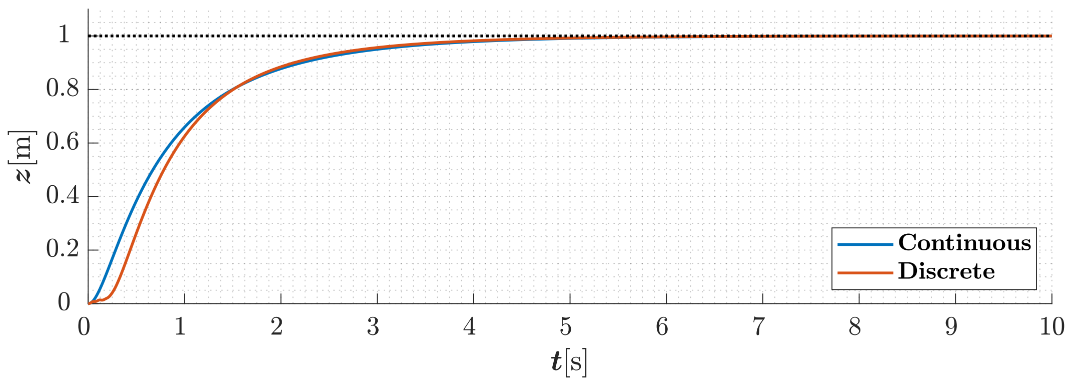


Figure 5.5: Closed-Loop Altitude Response of the Nonlinear Model

It is immediately noticeable that the discrete and continuous controller perform very similarly, especially when it comes to attitude stabilization. In altitude control, there is only a slight mismatch in the initial phase of the transient period. The approximate time-domain specifications for the responses generated by the discrete PID controller can be found in Table 5.2:

Table 5.2: PID: Attitude and Altitude Time Domain Specifications

	Rise Time [s]	Settling Time [s]	Overshoot [%]
z	3	4	0
ϕ	0.2	0.25	0.5
θ	0.2	0.25	0.5
ψ	0.75	1.3	15

5.1.2 Position Control

Moving back to the structure presented at the very beginning of the present section, the outer PIDs and the calculation block are now included in the hierarchical controller. With the inner loop now tuned to give a satisfactory performance, the process of selecting the parameters for xy-position control can begin. The gains of the controllers of global position in the horizontal plane were again selected following the same heuristic procedure, by constantly checking the response of the nonlinear continuous-time system. The final gain sets are given below:

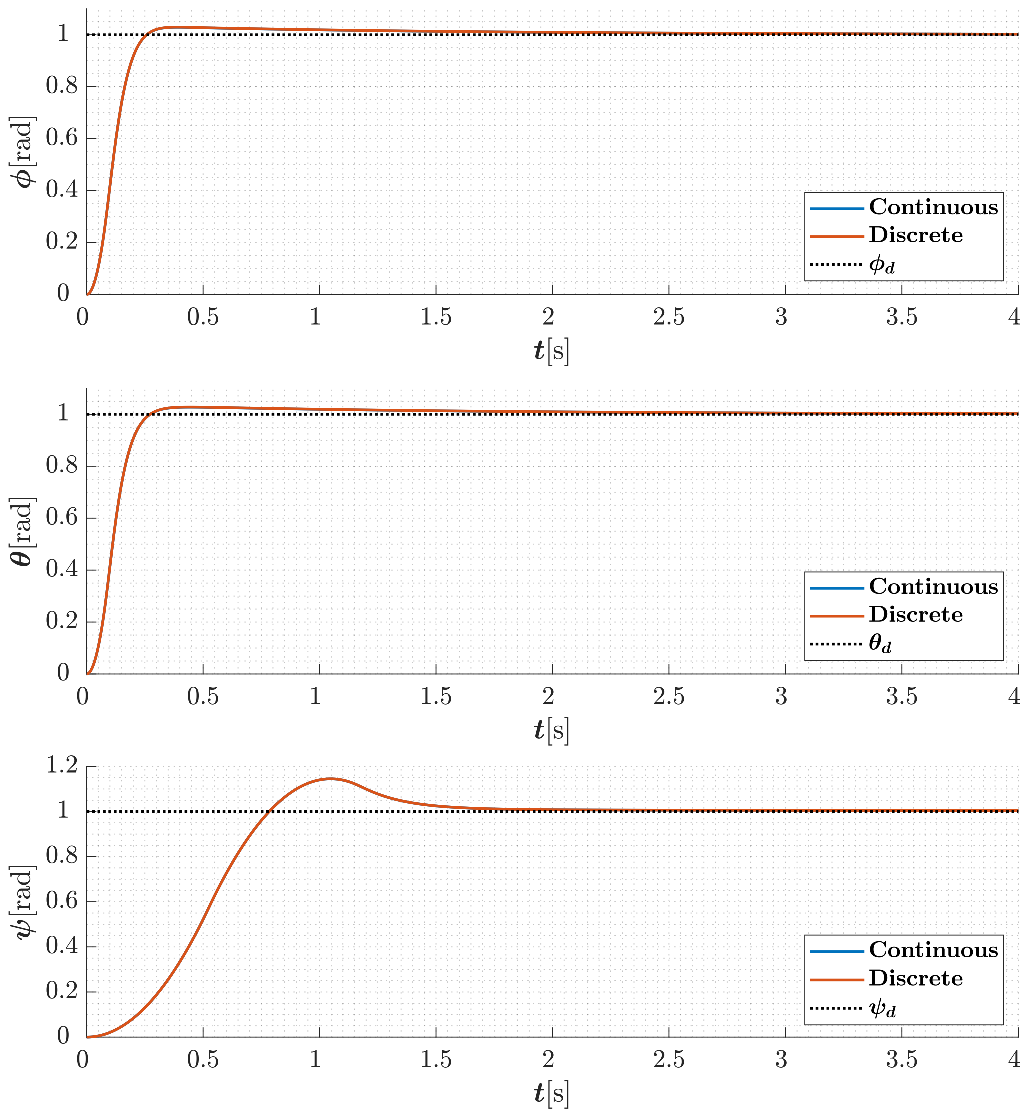


Figure 5.6: Closed-Loop Position Response of the Nonlinear Model *Top to Bottom: ϕ, θ, ψ*

$$\begin{aligned} \text{PID}_x: &\quad k_p = 0.6, \quad k_i = 0.004 \quad k_d = 0.4 \\ \text{PID}_y: &\quad k_p = -0.6, \quad k_i = -0.004 \quad k_d = -0.4 \end{aligned}$$

As expected, the PID_y gains are negative and have the same absolute value as the ones of PID_x.

Sampling Rate Selection

ToF sensors similar to the aforementioned one were used for obtaining information about the UAV's position within the global horizontal plane. Hence, the sampling time for this newly-added loop was chosen to be identical to T_z, namely 33ms. Let us relabel this time $T_{\mathbf{p}}$. We thus truly obtain a cascade control system, running at two distinct frequencies, $f_{\mathbf{p}}$ and f_{η}. As before, ZOH blocks were added after the PIDs. To check if the sampling satisifies Equation 5.7, the Bode magnitude plots of the closed-loop transfer functions (encompassing the previously-found closed-loop input-output relationship of the ϕ and θ loops) were consulted (see Figure 5.7):

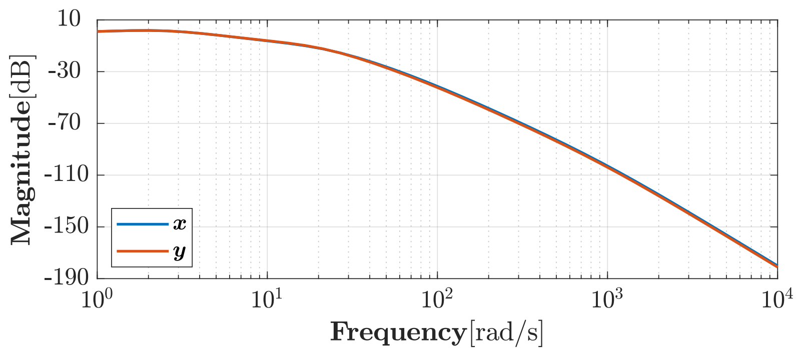


Figure 5.7: Closed-Loop Magnitude Bode Plots for Continuous xy-Position Control

The identified bandwidths were in this case:

$$f_{b,x} = 0.97\text{Hz}, \quad f_{b,y} = 0.98\text{Hz}$$

which are well within the desired range. Corner frequencies for the two derivative term low-pass filters of the PID controllers were chosen to be multiples of 10 of the bandwidths (converted to rad/s).

Simulation

The simulation results for a position reference $(x_d, y_d, z_d) = (1, 1, 1)$ and yaw reference $\psi_d = 0$ are displayed in Figure 5.8, showing the responses for xy-positioning, and Figure 5.9, illustrating how the inner loop tracks the reference of the outer loop until a correct decay to 0.

The fact that the system responses to the continuous and discrete controllers are almost indistinguishable substantiates again the good choice of sampling times. The time specifications for position control are listed in Table 5.3:

Table 5.3: xy-Position Time Domain Specifications

	Rise Time [s]	Settling Time [s]	Overshoot [%]
x	1.5	1.8	0.1
y	1.5	1.8	0.1

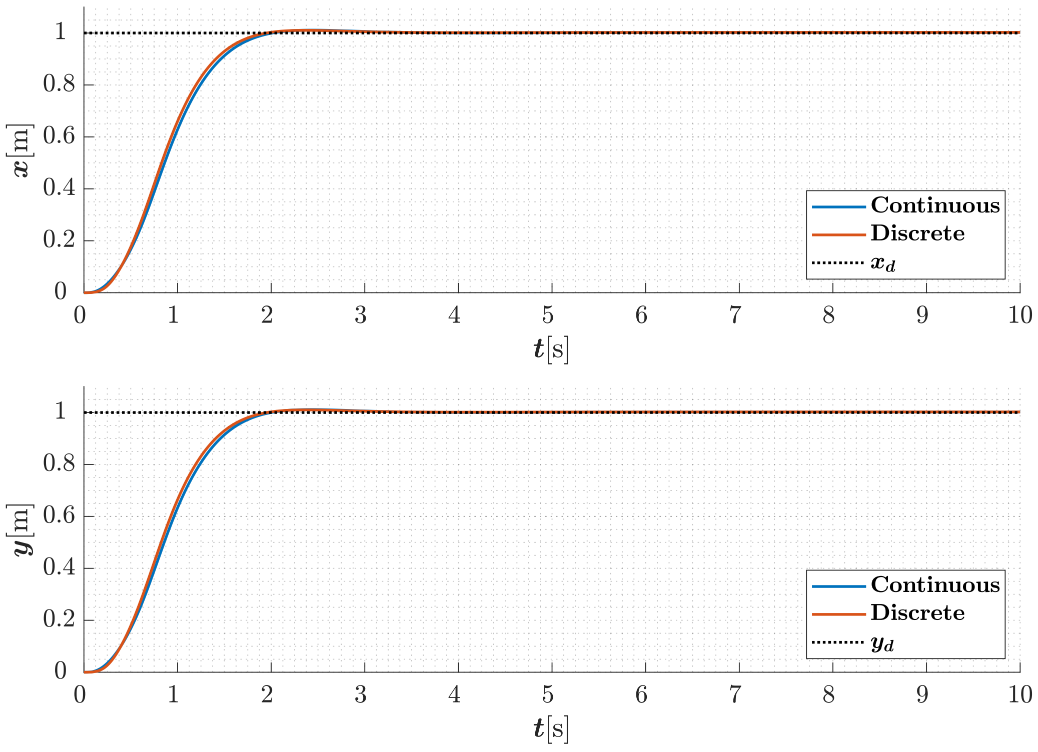


Figure 5.8: Closed-Loop Position Response of the Nonlinear Model *Top to Bottom: x, y*

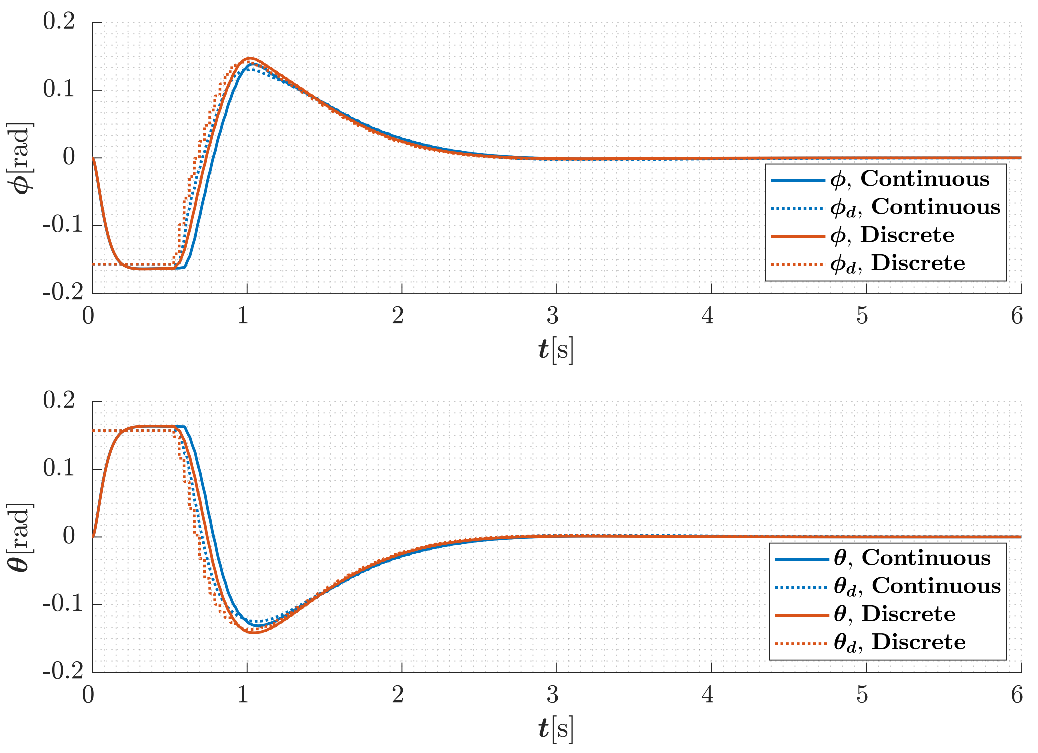


Figure 5.9: Attitude Response of the Nonlinear Model During Position Control. *Top to Bottom: ϕ, θ*

5.1.3 Trajectory Tracking

It was also investigated if the developed controllers could be utilized in conditions of changing references and trajectory tracking. A very popular choice utilized in the literature for such purposes is the rising 3D helix (or spiral) [72, 73, 74, 75]. One example of such a curve which was utilized as a reference trajectory in this project is given by the position vector in Equation 5.9:

$$
\begin{gathered}
r_{d,\mathcal{G}}(t) = \begin{bmatrix} x_d & y_d & z_d \end{bmatrix}^T \\
x_d(t) = \cos(0.4t), \quad y_d(t) = \sin(0.4t), \quad z_d(t) = 1 + 0.1t
\end{gathered}
\tag{5.9}
$$

which effectively describes a parametric curve whose projection on the xy-plane is a circle with a radius of 1m.

For simplicity, we let $\psi_d = 0$. The simulation is carried out for $t = 50$s and the results - for the two controller versions - when the system is excited with the preceding reference signal are pictured in Figure 5.10. It is clear that the drone would theoretically be able to closely follow the helical curve.

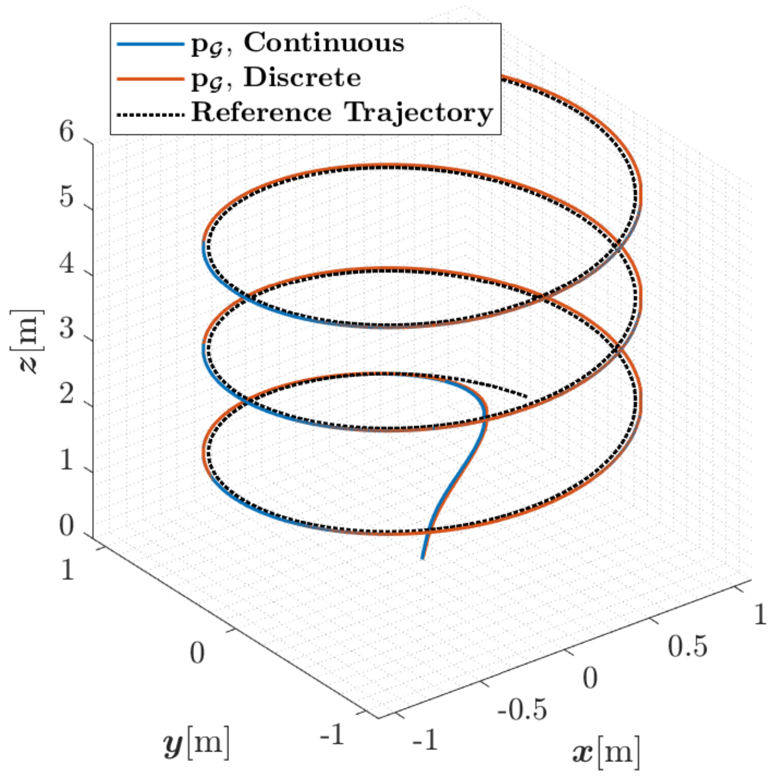


Figure 5.10: 3D Helix Trajectory Tracking Response of the Nonlinear Model

5.2 Linear Quadratic Regulator (LQR) Control

The following section shifts its attention towards the LQR controller and will deal with its control architecture as well as the tuning procedure with respect to the required performance parameters.

Introduction to LQR

Primarily, a state space model is created based on the set of linear equations in 3.38, presented in Section 3.8. This model needs to be checked for controllability which determines a system's ability to reach any linear combination of its states in a finite amount of time [70]. This is identified through the controllability matrix (shown in Equation 5.10), such that the said matrix should comprise of a full rank and the rank should be equal to the number of states, for the system to be controllable.

$$\mathcal{C} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \mathbf{A}^3\mathbf{B} & \dots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} \tag{5.10}$$

In the case of the quadrotor system, the matrix was found to be of full rank, equalling the number of states, i.e.: 12.

LQR is a modern control method which aims at optimal control through the use of a performance index or cost function, stated in Equation 5.11. To achieve the optimal control, this method seeks to find the appropriate control law which on application in a closed loop design will perform the optimal control actions resulting in a minimized cost function J.

$$J = \int_0^\infty (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})dt \tag{5.11}$$

In the cost function, the $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ are positive definite symmetric matrices, which implies that the cost is non negative and its minimum is zero [70]. $\mathbf{Q}$ is a diagonal matrix with each element of the diagonal corresponding to a particular state and if a certain element of the diagonal is large it means a higher cost for the associated state. Accordingly, to make the state converge faster we increase its corresponding diagonal element. Similarly, $\mathbf{R}$ is associated with the control input and the higher its values the more we penalize controller effort. Consecutively, the tuning procedure for LQR involves finding the appropriate $\mathbf{Q}$ and $\mathbf{R}$ matrices based on the performance requirements. These matrices are curial to finding the control law as illustrated in Equation 5.12. The optimal gain matrix $\mathbf{K}$ is identified using Equation 5.13, which employs the positive definite symmetric matrix $\mathbf{P}$ (not to be confused with the thrust mixing matrix denoted by the same letter). Now, $\mathbf{P}$ is determined through the algebraic Riccati equation (Equation 5.14).

$$\mathbf{u} = -\mathbf{K}\mathbf{x} \tag{5.12}$$

$$\mathbf{u} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{5.13}$$

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} = 0 \tag{5.14}$$

In this project's case, MATLAB command `K = lqr(A, B, Q, R)` was used to find the optimal gain matrix.

The tuning of **Q** and **R** matrices is mostly heuristic but as an aid, the Bryson's rule is employed which provides the initial values for these matrices. Correspondingly, according to the rule, the diagonal elements of the matrices are initiated to the inverse of the square of maximum acceptable value of the associated state or input, as illustrated in Equation 5.15.

$$Q_{i,i} = \frac{1}{x_{i,max}^2} \qquad R_{j,j} = \frac{1}{u_{j,max}^2} \tag{5.15}$$

The maximum values were applied based on Table 5.1 in Section 5.1. As for the position coordinates x, y, and z, the maximum threshold was obtained according to the range limit of the LiDAR sensors used for each coordinate.

Control Architecture

Similar to PID, the LQR controller design is done through an approach which divides the procedure into three parts, each in consideration to altitude, attitude and the xy-position control. As shown in the figure, the control structure is arranged in a hierarchical fashion with the outer layer comprising the translational position controller and the inner layer including the attitude controller. Additionally, the yaw rotation block is used in the xy-position control with the functionality of changing the basis of the x and y position error from inertial (or global) frame to the first intermediate Euler frame. This structure was designed with the assumption of full state feedback from the actual system. As compared to the method which embodies the system as just one single state-space, the hierarchical structure allows for a more intuitive and systematic approach as each controller could be tuned individually, without effecting the others. Furthermore, this enables the switching between different control structures based on the situation. For example, in the case of loss of x and y sensor output, the xy-controller could be removed and a switch to a fixed reference for the attitude controller can be made, in order to maintain the quadrotor in a hovering condition [21].

The controller is developed with the purpose of maintaining the system at a certain reference. However, in the case of the standard LQR architecture, the system won't reach the references resulting in a steady state error. Therefore, we add an ad hoc solution to the steady state error problem by using integral control. It is implemented by augmenting the integrator dynamics with the state

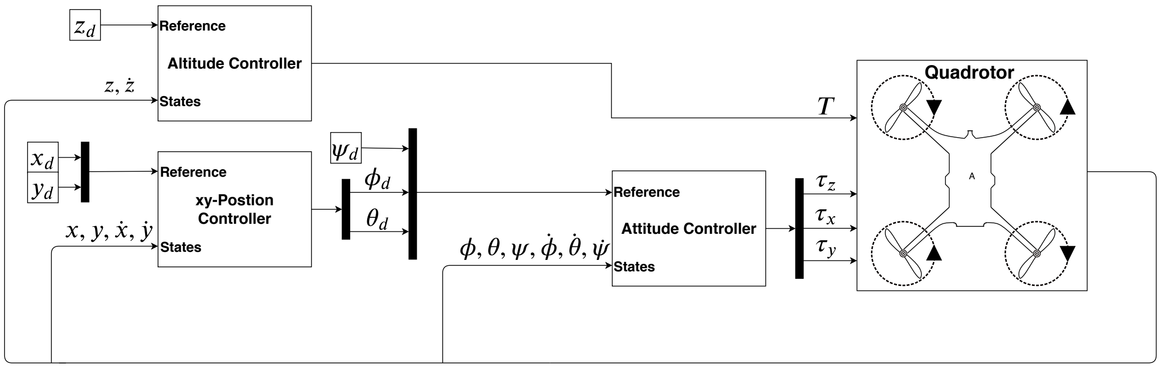


Figure 5.11: LQR Control Structure

vector such that the integral of the error is treated as a state and is fed-back. This results in a change in the state space model as well as the control law, shown in Equations 5.16 and 5.17 respectively.

$$\begin{bmatrix} \dot{x}_i \\ \dot{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{C} \\ 0 & \mathbf{A} \end{bmatrix} \begin{bmatrix} x_i \\ \mathbf{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{B} \end{bmatrix} \mathbf{u} + \begin{bmatrix} -1 \\ 0 \end{bmatrix} \mathbf{r}$$
$$\mathbf{y} = \begin{bmatrix} 0 & \mathbf{C} \end{bmatrix} \begin{bmatrix} x_i \\ \mathbf{x} \end{bmatrix} \tag{5.16}$$

$$\mathbf{u} = -\mathbf{K}_a \mathbf{x}_a, \text{ where } \mathbf{K}_a = \begin{bmatrix} \mathbf{K}_c & K_i \end{bmatrix} \tag{5.17}$$

Controller Design Requirements

In order to proceed further, some design requirements need to be specified for the LQR controllers. These requirements will act as guidelines while tuning the **Q** and **R** matrices. A general requirement, which is rather intuitive to understand is that the Tait-Bryan angles should stabilize faster to their steady states as compared to the translational positions. Subsequently, this is also evident from the cascaded nature of the attitude and xy-position controllers. Accordingly, as it can be seen in Figure 5.11, the roll and pitch controller make the inner loop of the cascaded structure, thus ϕ and θ angle should converge faster than the x and y positions. The roll and pitch angles are of primary importance when it comes to the quadrotor stabilization and as a result, the requirements for their controllers is the most aggressive. Also, due to the symmetric structure of the quadrotor, the rolling and pitching actions are very similar, resulting in the same magnitude for their controller gains. As for the outer loop, the xy-position controller requirements are based such that it is at least 4-5 times slower than the inner loop, in terms of rise time. Moreover, from the non-linear equations 3.31, it can be seen that the dynamics for the x and y positions are also influenced by the yaw angle. Therefore, the yaw control requirements are selected in such a way that the oscillations around yaw reference does not affect the controller effort for xy-stabilization. Lastly, the altitude controller desideratum is decided on the reasoning that an aggressive altitude controller can make the other states unstable. This is because the altitude controller puts an equal input to all the motors and this could result in a higher turning torque. Conclusively, each controllers' design needs are given in the table 5.4, in consideration to the rise time and overshoot. It should be noted that in the case of translational position, the specifications for overshoot was chosen based on the sensor limit. This is to prevent the loss of sensor info which might happen if the drone overshoots and flies out of the sensor range. In addition, the overshoot characteristics for Euler angles were based on the maximum tilt angles such that even in the case of overshoot, the drone doesn't exceed the tilt limit. Regarding settling time, no specific requirement is set, but as a guideline, the shortest possible time is preferred.

Table 5.4: LQR Performance Requirements

	Rise Time $[s]$	Overshoot $[\%]$
ϕ	≤ 0.5	≤ 10
θ	≤ 0.5	≤ 10
ψ	≤ 1	≤ 10
x	≤ 2	≤ 10
y	≤ 2	≤ 10
z	≤ 1.2	≤ 5

LQR Tuning

The tuning of the LQR controllers was performed in successive steps. First, the linear models for each of the altitude, attitude and xy-position were obtained by dividing the system into three parts based on the set of linear equations shown in 3.38. Next, the **Q** and **R** matrices for each controller were tuned based on these linear models. This provided a much simpler way to the tuning procedure as the linear equations are mostly decoupled, allowing the adjustment of the above-mentioned matrices without concerns of affecting the stability of other states, except in the cascaded scenario. Accordingly, once a satisfactory response was achieved, the performance of the controllers was corroborated on the nonlinear model of the quadrotor. In addition to the model being constructed based on the non-linear equations in 3.31, another aspect of non-linearity was added with the saturation limits for the inputs. During this process, the optimal gain matrix **K** is first identified in MATLAB for the linear model and then verified on the non-linear model in SIMULINK. In the case of lacking performance, the **Q** and **R** matrices were readjusted to meet the performance desideratum.

5.2.1 Altitude Controller

The state space for the altitude section of the system is governed by the system matrix $\mathbf{A}_{al}$ and the input matrix $\mathbf{B}_{al}$, given in Equation 5.18:

$$\mathbf{A}_{al} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{B}_{al} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \tag{5.18}$$

in consideration to the state vector for altitude in Equation 5.19:

$$\mathbf{x}_{al} = \begin{bmatrix} z & \dot{z} \end{bmatrix}^T \tag{5.19}$$

Furthermore, u_{al} and y_{al} are the input and the output for the said system, respectively, as shown in Equation 5.20, in which T refers to the thrust generated by the motors:

$$u_{al} = T \qquad y_{al} = z \tag{5.20}$$

Now, to implement integral action, an integral state is augmented and a new state space is obtained which follows the structure of Equation 5.16. For this system, the optimal gains $K_{i,al}$ and $\mathbf{K}_{c,al}$ are obtained corresponding to the tuned $\mathbf{Q}_{al} \in \mathbb{R}^{3\times3}$ and $R_{al} \in \mathbb{R}^{1\times1}$.

$$\mathbf{Q}_{al} = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.02 \end{bmatrix} \qquad R_{al} = 0.0003$$

$$K_{i,al} = 31.6228 \qquad \mathbf{K_{c,al}} = \begin{bmatrix} 28.8910 & 10.5624 \end{bmatrix}$$

The step response for the closed loop system is shown in Figure 5.12. Also, the properties of the response are listed in Table 5.5. According to these properties, it can be said that the controller fulfills the requirements.

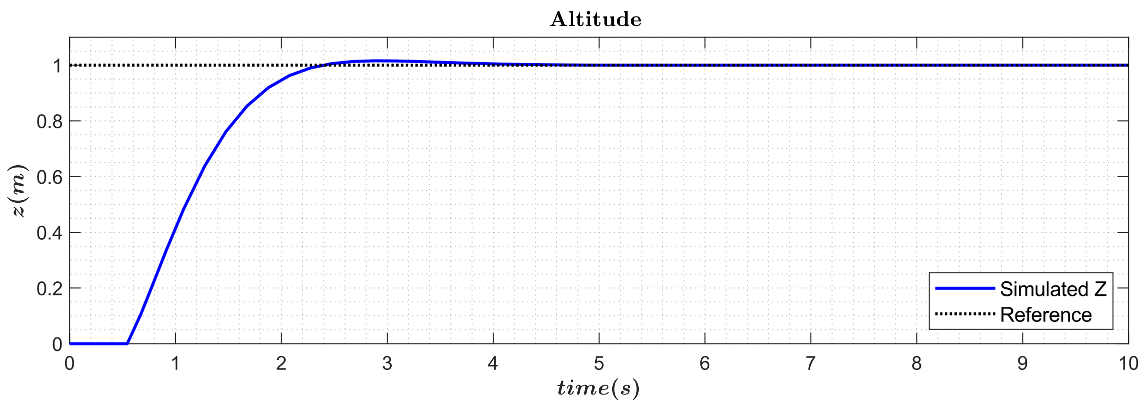


Figure 5.12: Altitude Response for LQR Altitude Control

Table 5.5: Altitude Controller Parameters

	Rise Time [s]	Settling Time [s]	Overshoot [%]
z	1.16	4	1.6

5.2.2 Attitude Controller

This controller aims at the stabilization of the Euler angles ϕ, θ, and ψ. Along with these angles, the angular velocities (equivalent to the Euler rates in the linearized model) are also treated as states of the system, consequently leading to the state vector $\mathbf{x}_{at}$ in Equation 5.21.

$$\mathbf{x}_{at} = \begin{bmatrix}\phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}^T \tag{5.21}$$

Moreover, the system matrix and the input matrix for attitude control is given by $\mathbf{A}_{at}$ and $\mathbf{B}_{at}$ respectively, both appearing in Equation 5.22:

$$\mathbf{A}_{at} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B}_{at} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \tag{5.22}$$

The system is controlled by the input torques τ_x, τ_y, and τ_z for rolling, pitching and yawing respectively. As for the output, it is comprised of the Euler angles, as shown in Equation 5.23.

$$\mathbf{u}_{at} = \begin{bmatrix}\tau_x & \tau_y & \tau_z\end{bmatrix}^T \qquad \mathbf{y}_{at} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T \tag{5.23}$$

Just like the altitude controller, integral control is applied to eliminate the steady state error. Once the new system is obtained, the optimal control gain matrices $\mathbf{K}_{i,at}$ and $\mathbf{K}_{c,at}$ are acquired through tuning $\mathbf{Q}_{at} \in \mathbb{R}^{9\times 9}$ and $\mathbf{R}_{at} \in \mathbb{R}^{3\times 3}$, shown below.

$$\mathbf{Q}_{at} = \text{diag}(\begin{bmatrix}30 & 30 & 15 & 0.01 & 0.01 & 0.5 & 0.01 & 0.01 & 1\end{bmatrix})\mathbf{R}_{at} = \text{diag}(\begin{bmatrix}0.1 & 0.1 & 1\end{bmatrix})$$

$$\mathbf{K}_{i,at} = \begin{bmatrix} 17.3205 & 0 & 0 \\ 0 & 17.3205 & 0 \\ 0 & 0 & 3.873 \end{bmatrix}$$

$$\mathbf{K}_{c,at} = \begin{bmatrix} 3.6739 & 0 & 0 & 0.3867 & 0 & 0 \\ 0 & 3.6357 & 0 & 0 & 0.3787 & 0 \\ 0 & 0 & 2.9164 & 0 & 0 & 1.0335 \end{bmatrix}$$

Similarly, the step response for the ϕ, θ, and ψ are shown in Figure 5.13 and performance characteristics can be seen in the Table 5.6.

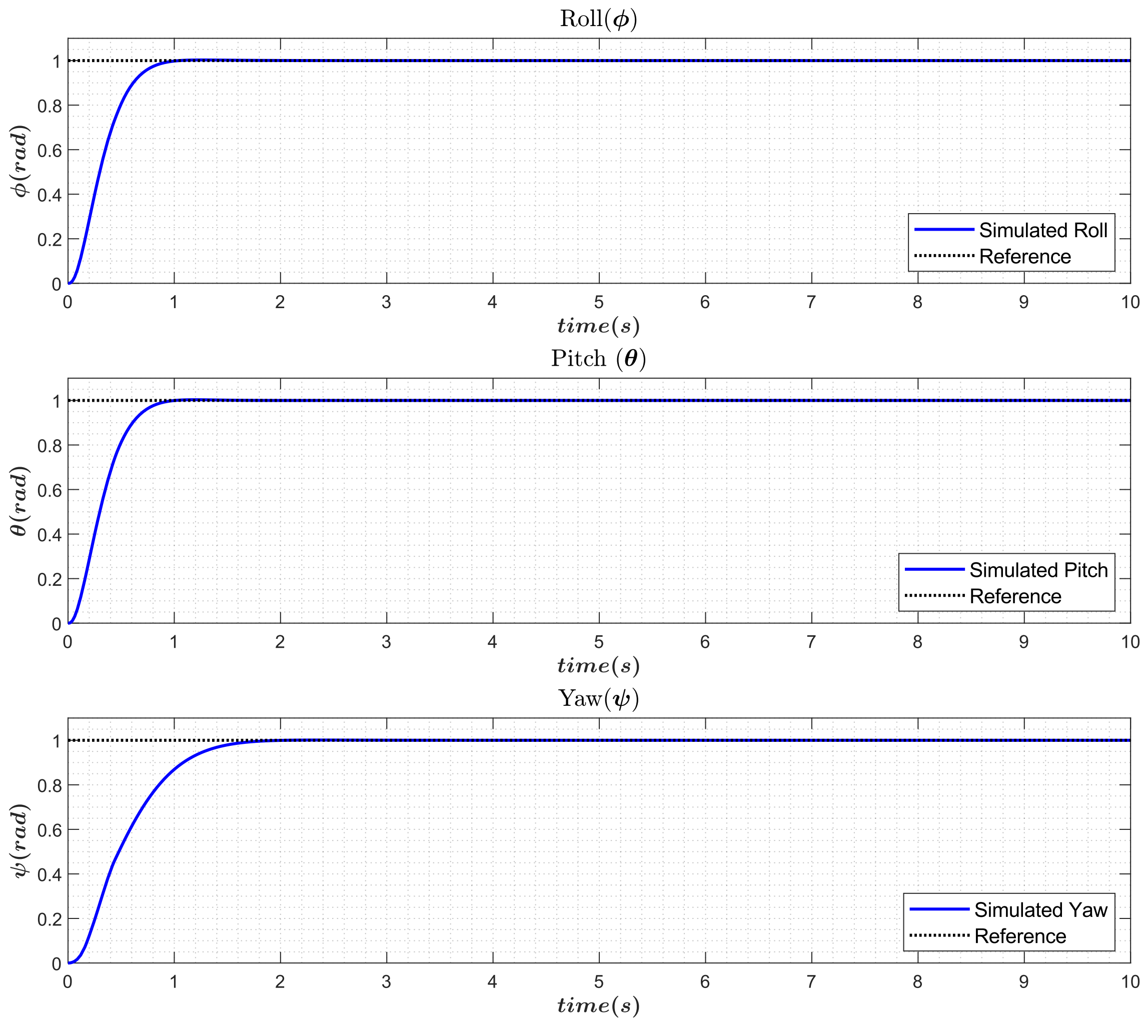


Figure 5.13: ϕ. θ, and ψ Responses for LQR Attitude Control

Table 5.6: Attitude Controller Parameters

	Rise Time [s]	Settling Time [s]	Overshoot [%]
ϕ	0.5	1	0.4
θ	0.5	1	0
ψ	0.9	2	0

5.2.3 xy-Position Controller

In the case of position control, the system results in the state space comprising of matrices $\mathbf{A}_{xy}$ and $\mathbf{B}_{xy}$ (Equation 5.24).

$$\mathbf{A}_{xy} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B}_{xy} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & g \\ -g & 0 \end{bmatrix} \tag{5.24}$$

The above state space uses the state vector $\mathbf{x_{xy}}$ which is equivalent to the combination of states shown in Equation 5.25.

$$\mathbf{x}_{xy} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T \tag{5.25}$$

Moreover, due to the cascaded structure of the attitude and xy-position controllers, the input vector $\mathbf{u}_{xy}$ for the outer loop is composed of the Euler angles relating to the roll and pitch motion of the quadrotor ($\mathbf{u}_{xy}$ shown in Equation 5.26). As a result, the references for the roll and pitch angles are no longer constant and change with time until converging to zero, which is the angular position that defines the hover condition. In addition, understandably so, the outputs of the system are the x and y positions, as illustrated in Equation 5.26.

$$\mathbf{u}_{xy} = \begin{bmatrix} \phi & \theta \end{bmatrix}^T \qquad \mathbf{y}_{xy} = \begin{bmatrix} x & y \end{bmatrix}^T \tag{5.26}$$

It is to be noted that during the tuning procedure for this controller, the initial control gains used for the inner attitude control are the same as the ones found in the last section. While using the same gains, the outer loop LQR is tuned to achieve satisfactory responses. However, to obtain a better performance the inner attitude controller is readjusted to be more aggressive resulting in a quicker response of the outer loop's x and y positions. Correspondingly, the $\mathbf{Q}_{xy} \in \mathbb{R}^{6\times 6}$ and $\mathbf{R}_{xy} \in \mathbb{R}^{2\times 2}$ are the driving factors for the found optimal control gains $\mathbf{K}_{i,xy}$ and $\mathbf{K}_{c,xy}$.

$$\mathbf{Q}_{xy} = \text{diag}(\begin{bmatrix} 5 & 5 & 0.8 & 0.8 & 0.1 & 0.1 \end{bmatrix}) \qquad \mathbf{R}_{xy} = \text{diag}(\begin{bmatrix} 100 & 100 \end{bmatrix})$$

$$\mathbf{K}_{i,xy} = \begin{bmatrix} 0 & -0.2236 \\ 0.2236 & 0 \end{bmatrix}$$

$$\mathbf{K}_{c,xy} = \begin{bmatrix} 0 & -0.3607 & 0 & 0.273 \\ 0.3607 & 0 & 0.273 & 0 \end{bmatrix}$$

Like the previous sections, the step responses for both outputs are plotted in Figure 5.14 along with the performance characteristics given in Table 5.7. Also, a comparison between the reference angles (supplied by the outer loop) and the ϕ and θ angle dynamics is depicted in the Figure 5.15.

In the end, based on the performance of each controller for all the controlled states, it is suffice to say that the tuned controllers were able to stabilize the system with respect to the required specifications.

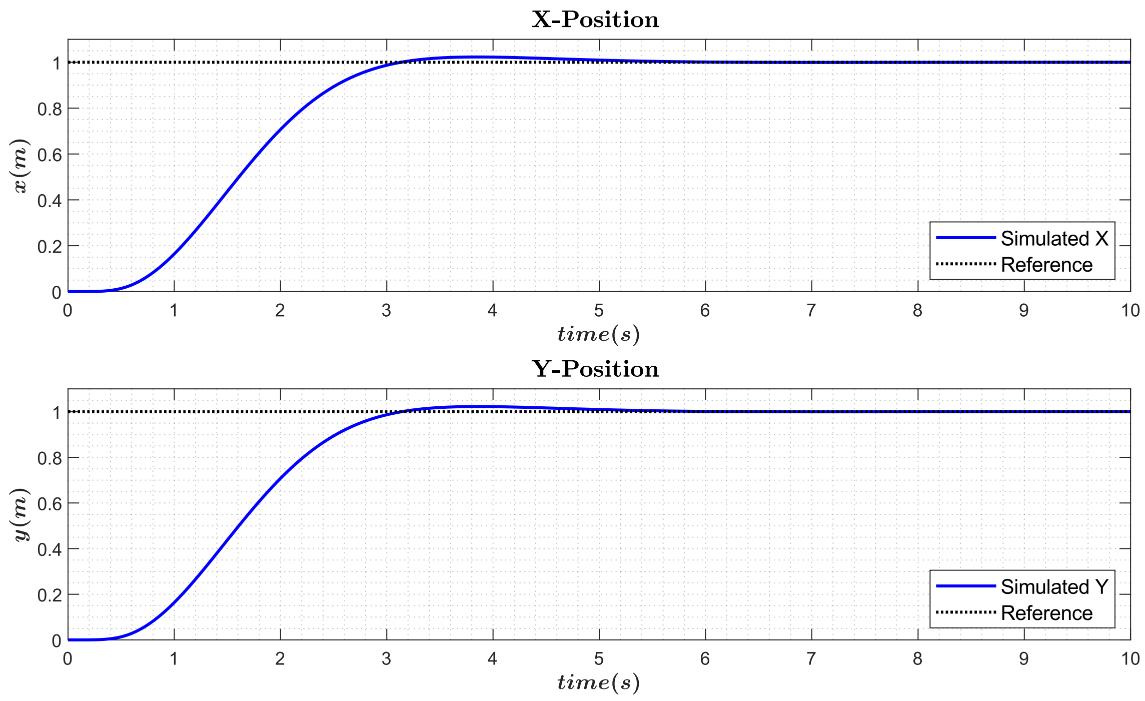


Figure 5.14: *xy*-Position Closed-Loop Response for LQR

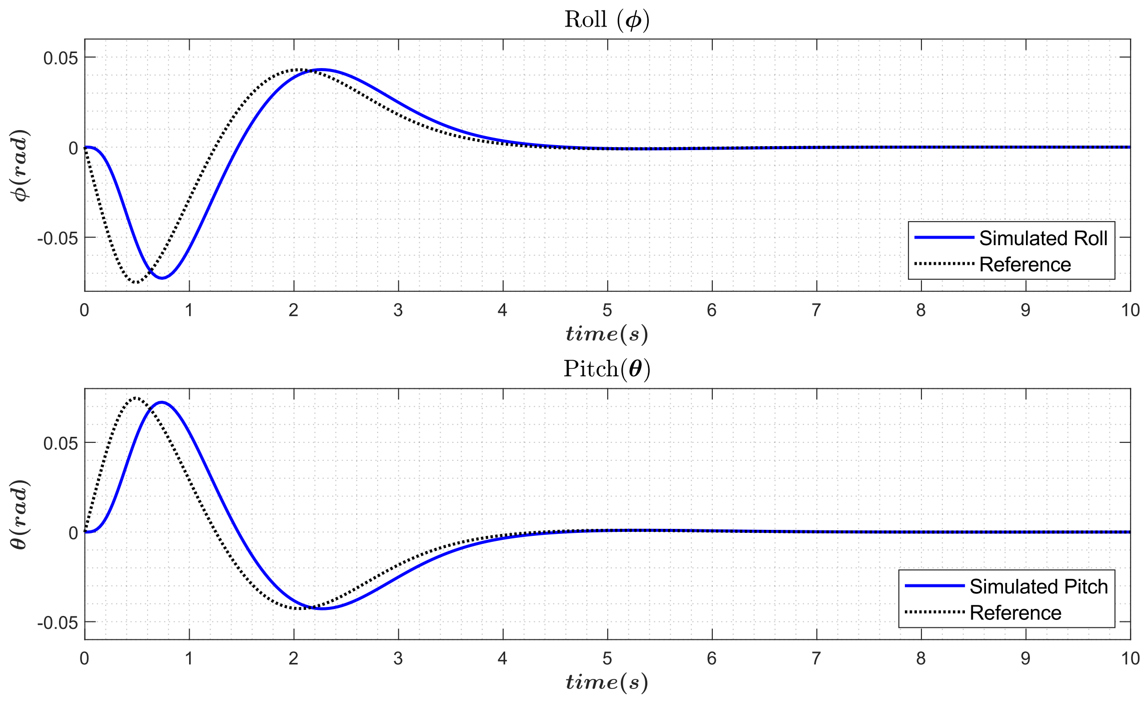


Figure 5.15: Roll and Pitch Dynamics in the Case of Position Control for LQR

Table 5.7: xy-Position Controller Parameters

	Rise Time [s]	Settling Time [s]	Overshoot [%]
x	1.7	5	2.3
y	1.7	5	2.2

5.2.4 Trajectory Tracking

Throughout the previous sections, the focus has been on the development of the suitable LQR controllers. As such, the obtained controllers can now be checked for their reference tracking capabilities. This can be done through the implementation of a helical reference trajectory, which has already been tested for the PID controller in Subsection 5.1.3. Therefore, to maintain consistency, the LQR controller will be verified for the same 3D curve. Accordingly, the position vector remains the same (see Equation 5.9). In addition, the ψ angle reference is fixed to 0 rad. Figure 5.16 illustrates the results to a helical reference for the quadrotor controller.

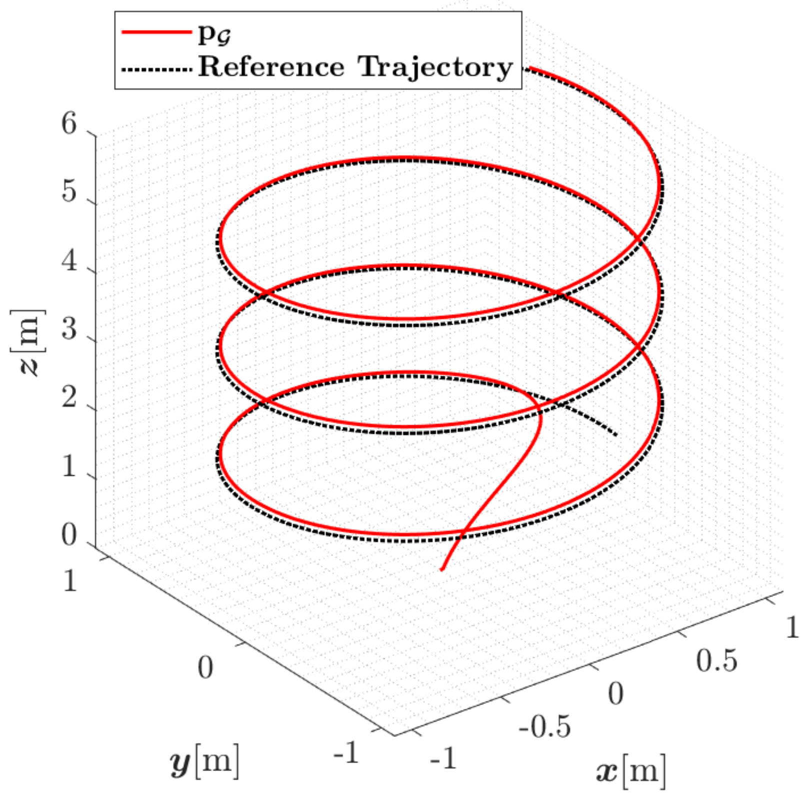


Figure 5.16: 3D Helix Trajectory Tracking Response of the Nonlinear Model for LQR

Moreover, from the figure, it is inherently clear that the developed controllers are adept in trajectory tracking as well. Hence, theoretically speaking, it will be safe to assume that when equipped with the LQR controller, the drone will have the capability to trace a curve that defines its translational position movement.

Chapter 6

Testing

A picture of the quadrotor which was built for testing purposes is shown in Figure 6.1. Due to its small size, finding the appropriate place for all electronic components that would let the motors spin freely has been challenging. All PID controllers for attitude and xyz-position were programmed in the MCU in accordance with their respective loop frequencies. Sensors are proved to be working as intended and it is made sure that they did not delay the stringent execution period of the controller loops. The program is written in a way that the quadrotor flies autonomously without any intervention. A timer that sets the altitude reference z_d to zero after a predefined period of time made autonomous landing possible. In case the attitude cannot be stabilized and the quadrotor poses danger by accelerating in undesired directions, a safety function that initiates a forced landing when the roll and pitch angles are above a certain threshold is implemented.

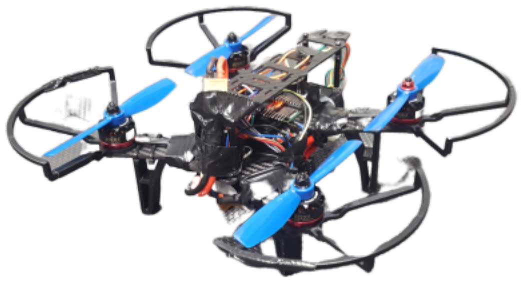


Figure 6.1: The Quadrotor Developed As Part of the Project

For the purpose of testing, $(x_d,\ y_d,\ z_d)$ are set to $(0.8, 0.8, 0.35\text{m})$ respectively. The yaw angle is also desired to converge to zero. Naturally, we initially utilized the gains found in simulation, but on inspection, they were found to be too aggressive and were consequently reduced. The graph presenting the result of this experiment is shown in Figure 6.2. Videos of two successive experiments (the first video corresponds to the data) with the mentioned references can be watched by accessing `https://bit.ly/2YZxmD7`.

Examining the Figure 6.2 and watching the test videos through the given link, it can be seen that the aimed performance of fast and error-free reference tracking as observed in the simulations was not achieved in real life. The data rather hints a lower controller performance with respect to reference tracking and time-domain specifications.

Altitude control seems to be the best among all. The quadrotor is able reach the reference in less than 2 s, indicating a better performance than the simulation. It maintains its height to a satisfactory level even when it is rolled and pitched. We believe the manipulation of the measured z-distances with respect to the Euler angles has a contribution in this achievement. Time to time, minor overshoots and undershoots in height is observed which are thought to originate due to partial loss of thrust in $\hat{\mathbf{z}}_G$ direction when the Euler angles are not zero.

The xy-position of the quadrotor reaches the reference rather fast and thus results in an overshoot of approximately 0.4m. Even though the controller is able to diminish the error in time, it is simply not fast enough. In order to obtain a better xy-position tracking performance, new controller coefficients were tested: integral gain was increased for faster error elimination and derivative gain was increased to decrease the amount of overshoot. However, none of the tests with the mentioned changes gave more promising results than what was already achieved. They either did not alter the performance noteworthily or led to the instability of the quadrotor.

Euler angles are also affected by the fact that there is an xy-position reference error. In such a scenario, non-zero roll and pitch angle references are calculated by the xy-position controllers. We notice that this leads ϕ and θ angles to rapidly change around the reference line that marks the optimum level. Despite how turbulent ϕ and θ seem, the fact that they are altering around desired level and xy-position error is decreasing over time proves that they do work. As in the case of xy-position control, no better performance was achieved by amending the PID coefficients. We also pondered upon the possibility that the observed controller performance may be due to the complementary filter and tried different cut-off frequencies with the hope of obtaining better results. However, the overall performance did not change positively.

ψ is controlled better compared to ϕ and θ, deviating only 3 degrees from the reference. This is due to the fact that ψ is not affected by the xy-position controllers. From the data, it is also seen that the controller performance is adequate.

The large overshoot in xy-control stands as a proof of the practical PID controller imperfection, although this unexpected behaviour may stem from unknown causes. This has unfavourably led to the preclusion of the closed-loop model validation process. Nonetheless, the overall testing results demonstrated successful controller implementation that needs further modification to approach the desired level of performance.

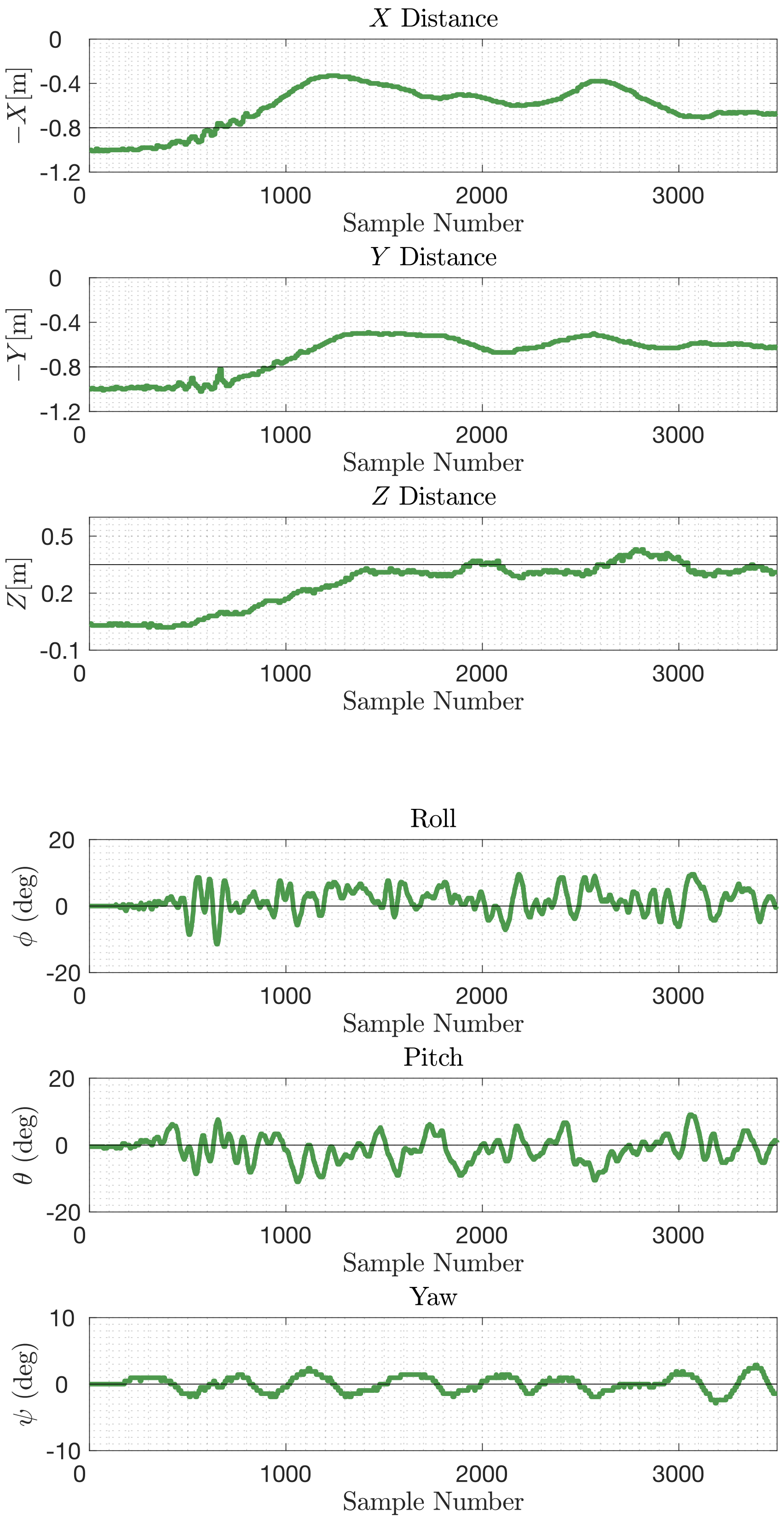


Figure 6.2: *xyz*-Position and Attitude Control Test Results

Chapter 7

Discussion

Modeling Techniques

As mentioned from the very beginning of the report, the dynamic equations of the quadrotor have been derived through the fairly straightforward Newton-Euler method. Although this technique sufficed for the necessary mathematical modeling, it would have been very beneficial, from a learning perspective, to develop the underlying relationships with the Euler-Lagrange methodology. Apart from this, literature seems to argue in favour of the quaternion formalism, which circumvents the singularity issue known as gimbal lock arising when adopting the rotation matrices. Thus, a study of quaternions is definitely worth pursuing in the future, but a stronger mathematical skill set needs to be developed beforehand.

In connection to the modeled forces and moments acting on the airframe, it was argued in Section 3.6 that the aerodynamic complexities are not very relevant in the context of this project, as compared to the thrust and body-frame reaction torques. Nevertheless, more effort could have been directed towards gaining a deeper understanding of the effects in question and identifying the dynamics that are worth including in the model, for the purpose of increasing the accuracy and the controller efficiency. Even more exactitude could have been achieved by including the mathematical representations of the other components of the propulsion model (encompassing models for ESC, motors and battery) [65].

State Estimation

Even though we have not noticed any significant drift in the gyroscope-based yaw angle estimation in our experiments, we anticipate that it would occur if we were to let the drone fly for a longer period. Therefore, under such tests, a more reliable yaw angle estimation could have been achieved if measurements from the gyroscope and the magnetometer were complemented.

$\mathbf{F}_{\mathcal{B}}$ is considered to be zero while deriving the formulae to estimate pitch and roll angles from the accelerometer in Section 4.1. By doing so, we excluded the effect of acceleration that we need in order to change quadrotor's position in 3D space. Predicting these accelerations from the actuator inputs by also taking the orientation of the drone into account and removing it from the measurements prior to using it at every sampling iteration would enhance the overall performance.

No mathematical approach has been taken to incorporate the sensor noise level and distribution into the sensor fusion algorithm due to the project timeframe and our lack of knowledge in statistics. We simply removed offsets from the sensor measurements and considered that the noise average is zero. Implementing a Kalman Filter, on the other hand, would bring a more mathematical approach to the issue.

Making use of the OptiTrack MOCAP measurements available in the Drone Lab of AAU by including them in the sensor fusion algorithm might be another endeavour to increase signal-to-noise ratio and to obtain better state estimation.

The utilized 1D-LiDARs are very accurate in their measurements, but have a working range of merely 1.2m. For this reason, we were compelled to place the quadrotor close to the corner of the testing arena in the Drone Lab, while conducting various experiments. In order to have a larger area in which trajectory controller can be tested, better distance-measuring sensors need to be installed.

LQR Controller Implementation

It should be clearly stated that we are very interested in the branch of modern control theory and had true intentions of testing the designed LQR controller on the real system. The C/C++ program was actually written, but the implementation stage was never reached due to time constraints. The chief impediment was represented by a 3 week delay in the ESC shipment, in mid-November. During that specific time period, no practical progress was possible.

Chapter 8

Conclusion

This report presents a bottom-up approach on quadrotor attitude and position stabilization. First, the conducted literature review allowed us to recognize the state of the art in modeling, control, sensors and state estimation. In light of prevailing techniques and technologies in these domains, and also by taking our level of knowledge into account, the most suitable approaches that fulfil the project's scope were identified and explored. The Newton-Euler equations were used to capture the UAV rigid-body dynamics, which led to the creation of a nonlinear quadcopter model. On top of that, a linear representation at the steady-state of hovering has also been deduced, with the end goal of its utilization for controller design (LQR, in particular). A simple thrust mixing algorithm was defined based on the relevant rotor dynamics, to link the computed controller outputs with the hardware inputs. In continuation, the parameter identification of unknown constants was a key stride towards the realization of SIMULINK models that were subsequently employed in controller design. PID and LQR compensators were tuned to obtain the best possible performance for attitude, altitude and xy-position control, while their trajectory tracking capabilities were additonally inspected with favourable results. From a data acquisition standpoint, reliable Euler angle estimation from IMU data was rendered possible with the aid of complementary filter design, a rudimentary and yet effective sensor fusion procedure. Thus, the disadvantages entailed by the individual use of accelerometer and gyroscope were nullified. Moving on to the implementation, in the case of PID specifically, discretization effects have been accounted for in simulation, in an attempt to represent the digital nature of the microcontroller. Emphasis has also been placed on the construction of a quadcopter which served its purpose as a plant for testing in an indoor environment. Through iterative readjustment of the controller gains while observing the response of the real system, an arguably satisfactory hovering stabilization was reached. Even though the controller performance was not at the desired level, and further refinement is undoubtedly required, the project is regarded as successful both in terms of implementation and learning objectives.

Bibliography

[1] A. L. Salih et al. "Modelling and PID controller design for a quadrotor unmanned air vehicle". In: *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. 2010.

[2] L. E. Romero, D. F. Pozo, and J. A. Rosales. *Quadcopter stabilization by using PID controllers*. 2014.

[3] T. Luukkonen. *Modelling and control of quadcopter*. Tech. rep. Aalto University, 2011.

[4] L.B. Lehn. "Model, Design and Control of a Quadcopter". MA thesis. Norwegian University of Science and Technology, 2015.

[5] E. Kuanatama et al. "PID and Fuzzy-PID Control Model for Quadcopter Attitude with Disturbance Parameter". In: *International Journal of Communication & Control* 12 (2017).

[6] D. Almeida. "Event-Triggered Attitude Stabilization of a Quadcopter". MA thesis. KTH Royal Institute of Technology, 2014.

[7] C. D. McKinnon and A. P. Schoelling. "Unscented External Force and Torque Estimation for Quadrotors". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016.

[8] D. Xia, L. Cheng, and Y. Yao. "A Robust Inner and Outer Loop Control Method for Trajectory Tracking of a Quadrotor". In: *Sensors (Basel)* 17 (2017).

[9] A. Tayebi and S. McGilvray. "Attitude stabilization of a four-rotor aerial robot". In: *43rd IEEE Conference on Decision and Control*. 2004, pp. 1216–1221.

[10] M. Greiff. "Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation". MA thesis. Lund University, 2017.

[11] I. C. Dikmen, A. A. Ansoy, and H. Tameltas. "Attitude Control of a Quadrotor". In: *2009 4th International Conference on Recent Advances in Space Technologies*. 2009.

[12] X. Liang, Y. Fang, and N. Sun. "Nonlinear Hierarchical Control for Unmanned Quadrotor Transportation Systems". In: *IEEE Transactions on Industrial Electronics* 66 (2018).

[13] R. M. M. de Oliveira. "Identification and Validation of a Quadrotor's Model Dynamics". 2014.

[14] T. T. Mac et al. "AR.Drone UAV control parameters tuning based on particle swarm optimization algorithm". In: *IEEE International Conference on Automation, Quality and Testing, Robotics - THETA 20th edition (AQTR)*. 2016.

[15] S. Bansal et al. "Learning Quadrotor Dynamics Using Neural Network for Flight Control". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. 2018.

[16] J. Muliadi and B. Kusumoputro. "Neural Network Control System of UAV Altitude Dynamics and Its Comparison with the PID Control System". In: *Hindawi Journal of Applied Transportation* 2018 (2018).

[17] M. Rich. "Model development, system identification, and control of a quadrotor helicopter". MA thesis. Iowa State University, 2012.

[18] P. Burman. "Quadcopter Stabilization with Neural Network". MA thesis. The University of Texas at Austin, 2016.

[19] H. Bolandi et al. “Attitude Control of a Quadrotor with Optimized PID Controller”. In: *Intelligent Control and Automation* 4 (2013).

[20] H. t. M. N. ElKholy. “Dynamic Modeling and Control of a Quadrotor Using Linear and Nonlinear Approaches”. MA thesis. The American University of Cairo, 2014.

[21] V. G. Adir and A. M. Stoica. “Integral LQR Control of a Star-Shaped Octorotor”. In: *INCAS BULLETIN* 4 (2012).

[22] L. M. Argentim et al. “PID, LQR and LQR-PID on a Quadcopter Platform”. In: *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*. 2013.

[23] R. Guardeno, M. J. Lopez, and V. M. Sanchez. “MIMO PID Controller Tuning Method for Quadrotor Based on LQR/LQG Theory”. In: *Robotics* 8 (2019).

[24] B. Landry. “Planning and Control for Quadrotor Flight through Cluttered Environments”. MA thesis. MIT, 2015.

[25] A. Romero. “Non-linear Control of Quad-copters via Approximate Dynamic Programming”. MA thesis. Swiss Federal Institute of Technology Zurich, 2018.

[26] G. V. Raffo, M. G. Ortega, and F. R. Rubio. “An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter”. In: *Automatica* 46 (2010).

[27] F. Sabatino. “Quadrotor control: modeling, nonlinear control design, and simulation”. MA thesis. KTH Royal Institute of Technology, 2015.

[28] S. Bouabdallah. “Design and Control of Quadrotors with Application to Autonomous Flying”. PhD thesis. EPFL, 2007.

[29] B. Kusumoputro, M. A. Heryanto, and B. Y. Suprapto. “Development of an Attitude Control System of a Heavy-lift Hexacopter using Elman Recurrent Neural Networks”. In: *2016 22nd International Conference on Automation and Computing (ICAC)*. 2016.

[30] C. Chamberlain. “System Identification, State Estimation, and Control of Unmanned Aerial Robots”. MA thesis. Brigham Young University, 2011.

[31] S. Lupashin et al. “A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips”. In: *2010 IEEE International Conference on Robotics and Automation*. 2010.

[32] D. W. Mellinger. “Trajectory Generation and Control for Quadrotors”. PhD thesis. University of Pennsylvania, 2012.

[33] R. Mahony, V. Kumar, and P. Corke. “Multirotor Aerial Vehicles”. In: *IEEE Robotics & Automation Magazine* 8 (2012).

[34] M. Elsamanty et al. “Methodology for Identifying Quadrotor Parameters, Attitude Estimation and Control”. In: *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2013, pp. 1343–1348.

[35] A. Z. Azfar and D. Hazry. “A Simple Approach on Implementing IMU SensorFusion in PID Controller for Stabilizing QuadrotorFlight Control”. In: *2011 IEEE 7th International Colloquium on Signal Processing and Its Applications*. 2011.

[36] X. Kong et al. “Robust Kalman Filtering for Attitude Estimation Using Low-cost MEMS-based Sensors”. In: *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*. 2014.

[37] PJRC. *Teensy USB Development Board*. 2019. URL: https://www.pjrc.com/store/teensy36.html (visited on 12/17/2019).

[38] RMRC. *Aikon AK32 V2 35A 4in1 6S ESC*. 2019. URL: https://www.readymaderc.com/products/details/aikon-32-4-in-1-esc-blheli32-dshot-35a (visited on 12/17/2019).

[39] Emax. *EMAX RS2205 RaceSpec Motor - Cooling Series*. 2019. URL: https://emaxmodel.com/emax-rs2205-racespec-motor.html (visited on 12/17/2019).

[40] STMicroelectronics. *World’s smallest Time-of-Flight ranging and gesture detection sensor*. 2018. URL: https://www.st.com/resource/en/datasheet/vl5310x.pdf (visited on 12/17/2019).

[41] Electropeak. *GY-87 10DOF IMU MPU6050 HMC5883L BMP180 Sensor Module*. 2019. URL: https://electropeak.com/gy-87-10dof-imu (visited on 12/17/2019).

[42] Gens Ace. *Gens ace 5000mAh 11.1V 3S1P 50C Lipo Battery Pack with EC5 Plug-Bashing Series*. 2019. URL: https://bit.ly/2YYR1Dc (visited on 12/17/2019).

[43] J Diebel. "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors". 2006.

[44] R. N. Jazar. *Theory of Applied Robotics: Kinematics, Dynamics, and Controls*. 2^{nd} ed. Springer, 2010.

[45] H. Goldstein, C. Poole, and J. Safko. *Classical Mechanics*. 3^{rd} ed. Addison Wesley, 2000.

[46] B. L. Stevens, Lewis F. L., and Johnson E. N. *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. 3^{rd} ed. John Wilwy & Sons, Inc., 2017.

[47] R. A. Serway and J. W. Jewett Jr. *Physics for Scientist and Engineers with Modern Physics*. 9^{th} ed. Cengage Learning, 2014.

[48] J. Walker. *Fundamentals of Physics: Halliday & Resnick*. 9^{th} ed. John Wiley & Sons, Inc., 2011.

[49] Federal Aviation Administration. *Helicopter Flying Handbook*. 2^{nd} ed. United States Department of Transportation, 2019.

[50] X. Zhang et al. "A Survey of Modelling and Identification of Quadrotor Robot". In: *Hindawi Journal of Abstract and Applied Analysis* 2014 (2014).

[51] C. Powers et al. *Influence of Aerodynamics and Proximity Effects in Quadrotor Flight*.

[52] P. Marques and A. D. Ranch. *Adavnced UAV Aerodynamics, Flight Stability and Control*. 2^{nd} ed. John Wiley and Sons Ltd, 2017.

[53] R. Beard. *Quadrotor Dynamics and Control Rev 0.1*. 2008.

[54] R. Syam. "Dynamics and Fuzzy Logic Method for Controlling Quadcopter". In: *Research Journal of Applied Sciences* 11 (2016).

[55] C. M. Harris and G. P. Allan. *Harris' Shock and Vibration Handbook*. 5^{th} ed. McGraw-Hill, 2002.

[56] Tyro Robotics Inc. *Series 1585 Dynamometer Datasheet*. 2019. URL: `https://bit.ly/38NQ0lY` (visited on 12/17/2019).

[57] J. D. Hoffman. *Numerical Methods for Engineers and Scientists*. 2^{nd} ed. Marcel Dekker, Inc., 2001.

[58] T. Bresciani. "Modelling, Identification and Control of a Quadrotor Helicopter". MA thesis. Lund University, 2008.

[59] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. 7^{th}*ed.*. Pearson, 2014.

[60] L. Ljung and T. Glad. *Modeling of Dynamic Systems*. 1^{st} ed. Prentice Hall, 1994.

[61] H. Chao et al. "A comparative evaluation of low-cost IMUs for unmanned autonomous systems". In: *2010 IEEE Conference on Multisensor Fusion and Integration*. 2010.

[62] K. P. Valavanis and G. J. Vachtsevanos. *Handbook of Unmanned Aerial Vehicles*. 1^{st} ed. Springer, 2015.

[63] J. K. Lee, E. J. Park, and Robinovitch S. N. "Estimation of Attitude and External Acceleration Using Inertial Sensor Measurement During Various Dynamic Conditions". In: *IEEE Transactions on Instrumentation and Measurement* 61 (2012).

[64] H. Hyyti and A. Visala. "A DCM Based Attitude Estimation Algorithm for Low-Cost MEMS IMUs". In: *International Journal of Navigation and Observation* 2015 (2015).

[65] Q. Quan. *Introduction to Multicopter Design and Control*. 1^{st} ed. Springer, 2017.

[66] V. Bistrov. "Performance Analysis of Alignment Process of MEMS IMU". In: *International Journal of Navigation and Observation* 2012 (2012).

[67] M. Kok, J. D. Hol, and T. B. Schön. *Using inertial sensors for position and orientation estimation*. Tech. rep. Linköpings Universitet, 2017.

[68] L. Benziane. "Attitude estimation & control of autonomous aerial vehicles". PhD thesis. Université de Versailles-Saint Quentin en Yvelines, 2015.

[69] A. V. Oppenheim and R. W. Schafer. *Discrte-Time Signal Processing*. 3^{rd}*ed.*. Pearson, 2010.

[70] R. T. Stefani et al. *Design of Feedback Control Systems*. 4^{th}*ed.*. Oxford University Press, 2002.

[71] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*. 3^{rd}*ed.*. Elssi-Kagle Press, 1998.

[72] L. Wang and H. Jia. “The Trajectory Tracking Problem Of Quadrotor UAV: Global Stability Analysis And Control Design Based On The Cascade Theory”. In: *Asian Journal of Control* 16 (2014).

[73] T. Congling et al. “Integral Backstepping Based Nonlinear Control for Quadrotor”. In: *35th Chinese Control Conference*. 2016.

[74] H. Lyu. “Multivariable Control of a Rolling Spider Drone”. MA thesis. University of Rhode Island, 2017.

[75] A. A. Najm and I. K. Ibraheem. “Nonlinear PID controller design for a 6-DOF UAV quadrotor system”. In: *Engineering Science and Technology, an International Journal* 22 (2019).

[76] *Topic 7: Rotating Co-ordinate Systems*. URL: `http://www.srl.caltech.edu/phys106/p106a00/topic7.pdf` (visited on 12/17/2019).

[77] *Vector Algebra and Calculus*. URL: `http://www.robots.ox.ac.uk/~sjrob/Teaching/Vectors/slides3.pdf` (visited on 12/17/2019).

[78] C. H. Edwards and D. E. Penney. *Calculus: Early Transcendentals*. 7^{th} ed. Pearson, 2007.

Appendix A

Vectors in Rotating Reference Frames

Consider the reference frames (and the associated three-dimensional coordinate systems) $\mathcal{I}$ (inertial frame) with basis vectors $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$, and $\mathcal{B}$ (non-inertial body frame), with basis vectors $\hat{\mathbf{i}}'$, $\hat{\mathbf{j}}'$ and $\hat{\mathbf{k}}'$. Let the two systems share the same origin and let $\mathcal{B}$ rotate with respect to $\mathcal{I}$ at an angular rate $\boldsymbol{\omega}$. A time-dependent vector r can be expressed using the basis vectors of either frame, as follows:

$$\mathbf{r}_{\mathcal{I}} = r_x\hat{\mathbf{i}} + r_y\hat{\mathbf{j}} + r_z\hat{\mathbf{k}} \qquad \mathbf{r}_{\mathcal{B}} = r_{x'}\hat{\mathbf{i}}' + r_{y'}\hat{\mathbf{j}}' + r_{z'}\hat{\mathbf{k}}' \tag{A.1}$$

There exists an orthogonal matrix $\mathbf{R}$ that essentially encodes the rotation of the coordinate system $\mathcal{B}$, and, when multiplied with $\mathbf{r}_{\mathcal{B}}$ gives $\mathbf{r}_{\mathcal{I}}$. Thus, we have:

$$\mathbf{r}_{\mathcal{I}} = \mathbf{R}\mathbf{r}_{\mathcal{B}} \tag{A.2}$$

Because of the orthogonality of $\mathbf{R}$, $\mathbf{R}^{-1} = \mathbf{R}^T$ and hence:

$$\mathbf{R}\mathbf{R}^T = \mathbf{I}_3 \Rightarrow \dot{\mathbf{R}}\mathbf{R}^T + \mathbf{R}\dot{\mathbf{R}}^T = 0 \Rightarrow \dot{\mathbf{R}}\mathbf{R}^T = -\mathbf{R}\dot{\mathbf{R}}^T = -(\dot{\mathbf{R}}\mathbf{R}^T)^T \tag{A.3}$$

The result in Equation A.3 proves that the matrix $\mathbf{R}\dot{\mathbf{R}}^T$ is skew-symmetric.

Now, for convenience, let the two reference frames be initially superimposed. If $\mathbf{r}_{\mathcal{B}}$ represents the position of a point P expressed in body coordinates, then the velocity of the body with respect to the inertial frame and expressed in body coordinates is $\mathbf{v}_{\mathcal{B}}$ and is obtained by differentiating $\mathbf{r}_{\mathcal{B}}$ with respect to the inertial basis. Remembering that the basis vector in $\mathcal{B}$ are not constant with respect to $\mathcal{I}$ carrying out the total differentiation yields:

$$\mathbf{v}_{\mathcal{B}} = \left.\frac{d\mathbf{r}_{\mathcal{B}}}{dt}\right|_{\mathcal{I}} = \frac{r_{x'}}{dt}\hat{\mathbf{i}}' + \frac{r_{y'}}{dt}\hat{\mathbf{j}}' + \frac{r_{z'}}{dt}\hat{\mathbf{k}}' + r_{x'}\frac{\hat{\mathbf{i}}'}{dt} + r_{y'}\frac{\hat{\mathbf{j}}'}{dt} + r_{z'}\frac{\hat{\mathbf{k}}'}{dt} \tag{A.4}$$

The first term in the left-hand side is the derivative of the vector within the body frame, while the second term describes the change of the body basis vectors relative to the inertial basis vectors. The nature of this term is the one of a tangential velocity emerging from the rotation and expressed as the vector product of $\mathbf{r}_{\mathcal{B}}$ and $\boldsymbol{\omega}_{\mathcal{B}}$.

$$\mathbf{v}_{\mathcal{B}} = \left.\frac{d\mathbf{r}_{\mathcal{B}}}{dt}\right|_{\mathcal{I}} = \left.\frac{d\mathbf{r}_{\mathcal{B}}}{dt}\right|_{\mathcal{B}} + \boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{r}_{\mathcal{B}} = \dot{\mathbf{r}}_{\mathcal{B}} + \boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{r}_{\mathcal{B}} \tag{A.5}$$

The preceding relation is the application of a general equation usually labeled "the Coriolis′ theorem" in the literature and holds for any vector. Therefore, the acceleration of the body relative to the inertial frame and expressed in body coordinates is:

$$\mathbf{a}_{\mathcal{B}} = \left.\frac{d\mathbf{v}_{\mathcal{B}}}{dt}\right|_{\mathcal{I}} = \left.\frac{d\mathbf{v}_{\mathcal{B}}}{dt}\right|_{\mathcal{B}} + \boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{v}_{\mathcal{B}} = \dot{\mathbf{v}}_{\mathcal{B}} + \boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{v}_{\mathcal{B}} \tag{A.6}$$

Now, consider the vector $\mathbf{r}_{\mathcal{B}}$ changes in such a manner that its coordinates in the body frame are constant in time, i.e., $\dot{\mathbf{r}}_{\mathcal{B}} = 0$. Then, from Equation A.2 and A.5, we have:

$$\mathbf{v}_{\mathcal{I}} = \mathbf{R}\mathbf{v}_{\mathcal{B}} = \left.\frac{d\mathbf{r}_{\mathcal{B}}}{dt}\right|_{\mathcal{I}} = \mathbf{R}(\boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{r}_{\mathcal{B}}) \tag{A.7}$$

Differentiating Equation A.2 in this case gives:

$$\dot{\mathbf{r}}_{\mathcal{I}} = \mathbf{v}_{\mathcal{I}} = \dot{\mathbf{R}}\mathbf{r}_{\mathcal{B}} \tag{A.8}$$

The two preceding equations are equivalent, so we can write:

$$\mathbf{R}(\boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{r}_{\mathcal{B}}) = \dot{\mathbf{R}}\mathbf{r}_{\mathcal{B}} \Rightarrow \boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{r}_{\mathcal{B}} = \mathbf{R}^T\dot{\mathbf{R}}\mathbf{r}_{\mathcal{B}} \tag{A.9}$$

We conclude that

$$[\boldsymbol{\omega}_{\mathcal{B}}]_\times = [\mathbf{R}^T\boldsymbol{\omega}_{\mathcal{I}}]_\times = \mathbf{R}^T[\boldsymbol{\omega}_{\mathcal{I}}]_\times\mathbf{R} = \mathbf{R}^T\dot{\mathbf{R}} \Rightarrow [\boldsymbol{\omega}_{\mathcal{I}}]_\times = \dot{\mathbf{R}}\mathbf{R}^T \tag{A.10}$$

and because we can write $[\boldsymbol{\omega}_{\mathcal{I}}]_\times$ as the following skew-symmetric matrix:

$$[\boldsymbol{\omega}_{\mathcal{I}}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{A.11}$$

it results that the matrix $\mathbf{R}\dot{\mathbf{R}}^T$ is also skew-symmetric, showing that Equations A.10 is indeed correct.

Notation for Derivatives

For complete clarity, the following notation has been adopted for this report:

$$\left.\frac{d\mathbf{r}_{\mathcal{I}}}{dt}\right|_{\mathcal{I}} = \dot{\mathbf{r}}_{\mathcal{I}} = \mathbf{v}_{\mathcal{I}} \tag{A.12}$$

is the speed of point P relative to the inertial frame and expressed using the basis of the inertial frame

$$\left.\frac{d\mathbf{r}_{\mathcal{B}}}{dt}\right|_{\mathcal{I}} = \mathbf{v}_{\mathcal{B}} = \mathbf{R}^T\mathbf{v}_{\mathcal{I}} \quad \text{is the speed of point P relative to the inertial frame and expressed using the basis of the body frame} \tag{A.13}$$

$$\left.\frac{d\mathbf{r}_{\mathcal{I}}}{dt}\right|_{\mathcal{B}} = \dot{\mathbf{r}}_{\mathcal{I}}^{\mathcal{B}} = \mathbf{v}_{\mathcal{I}}^{\mathcal{B}} \quad \text{is the speed of point P relative to the body frame and expressed using the basis of the inertial frame} \tag{A.14}$$

$$\left.\frac{d\mathbf{r}_{\mathcal{B}}}{dt}\right|_{\mathcal{B}} = \dot{\mathbf{r}}_{\mathcal{B}} = \mathbf{v}_{\mathcal{B}}^{\mathcal{B}} = \mathbf{R}^T\mathbf{v}_{\mathcal{I}}^{\mathcal{B}} \quad \text{is the speed of point P relative to the body frame and expressed using the basis of the body frame} \tag{A.15}$$

The Newton-Euler equations (Section 3.5) hold true in inertial reference frames, but can be written with respect to either basis vectors.

The preceding contents of Appendix A have been written based on [44, 76, 77].

Appendix B

Inertia Tensor

In general, we have that, for a rigid body rotating in three dimensions around a certain point, "the angular momentum is related to the angular velocity by a linear transformation" [45], as shown in Equation B.1:

$$\begin{cases} H_x = I_{xx}\omega_x + I_{xy}\omega_y + I_{xz}\omega_z \\ H_y = I_{yx}\omega_x + I_{yy}\omega_y + I_{yz}\omega_z \\ H_z = I_{zx}\omega_x + I_{zy}\omega_y + I_{zz}\omega_z \end{cases} \tag{B.1}$$

which can be rewritten succinctly as:

$$\mathbf{H} = \mathbf{I}\boldsymbol{\omega} \tag{B.2}$$

where **H** is the angular momentum about a certain point and $\boldsymbol{\omega}$ is the angular velocity. We can therefore define the transformation matrix:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{B.3}$$

where the diagonal elements are called the moment of inertia coefficients and the off-diagonal elements are referred to as the products of inertia. We can illustrate their general form by taking one example of each, as follows:

$$I_{xx} = \sum_{i=1}^{N} m_i(r_i^2 - x_i^2) \qquad\qquad I_{xy} = -\sum_{i=1}^{N} m_i x_i y_i \tag{B.4}$$

the relations being valid for an object comprised of N discrete particles, with r_i being the radius of the ith particle relative to the point in question. Notice that the moment of inertia coefficients cannot be negative.

Equation B.2 indicates that **I** is a member of a different class than the vector it acts upon (ω). It is actually defined as a more general quantity called a tensor of the second rank (a tensor of the first rank is a vector) and can be represented in matrix form. Thus, **I** is the inertia tensor.

It is easy to see that Equation B.1 is greatly simplified if the inertia tensor has a diagonal form, referred to as the principal moment of inertia tensor. Rotation matrices affect the entries of **I**, and it can be proven that it is always possible to bring the inertia tensor to a diagonal from by adequately choosing a rotation that converts to a set of orthogonal principal axes. It is also known that a rigid body has three principal axes and that an axis of symmetry is such a principal axis. Since that is the case, we can say that, in the drone's $\mathcal{B}$-frame, if symmetry with respect to the three body axes passing through the COM is assumed, the inertia tensor is diagonal:

$$\mathbf{I}_{\mathcal{B}} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{B.5}$$

We can transform between the diagonal inertia tensor $\mathbf{I}_{\mathcal{B}}$ and its non-diagonal counterpart $\mathbf{I}_{\mathcal{I}}$, i.e., the inertia tensor with respect to the axes of the inertial coordinate system, using the rotation matrix $\mathbf{R}$ (rotating from the body frame to the inertial frame):

$$\mathbf{H}_{\mathcal{I}} = \mathbf{I}_{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{I}} \Rightarrow \mathbf{R}\mathbf{H}_{\mathcal{B}} = \mathbf{I}_{\mathcal{I}}\mathbf{R}\boldsymbol{\omega}_{\mathcal{B}} \Rightarrow \mathbf{H}_{\mathcal{B}} = \mathbf{R}^T\mathbf{I}_{\mathcal{I}}\mathbf{R}\boldsymbol{\omega}_{\mathcal{B}} \tag{B.6}$$

leading to the similarity transformation:

$$\mathbf{I}_{\mathcal{B}} = \mathbf{R}^T\mathbf{I}_{\mathcal{I}}\mathbf{R} \tag{B.7}$$

The preceding contents of Appendix B have been written based on [45].

B.1 Moments of Inertia of a Solid Rectangular Cuboid Body

From [78], we know that, for a three-dimensional solid body T representing a bounded region is space, with a continuous density function $\rho(x, y, z)$, the three moments of inertia around the three coordinate axes can be calculated using the formula in Equation B.8:

$$\begin{cases} I_{xx} = \iiint_T \rho(x,y,z)(y^2+z^2)dV \\ I_{yy} = \iiint_T \rho(x,y,z)(x^2+z^2)dV \\ I_{zz} = \iiint_T \rho(x,y,z)(x^2+y^2)dV \end{cases} \tag{B.8}$$

For a rectangular cuboid body (Figure B.1) whose center of mass corresponds to the origin of the coordinate system, with dimensions L, W and H, mass m and constant density $\rho(x, y, z) = \frac{m}{LWH}$, the moments of inertia take, according to [58], the form in Equation B.9:

$$\begin{cases} I_{xx} = m\frac{W^2+H^2}{12} \\ I_{yy} = m\frac{L^2+H^2}{12} \\ I_{zz} = m\frac{L^2+W^2}{12} \end{cases} \tag{B.9}$$

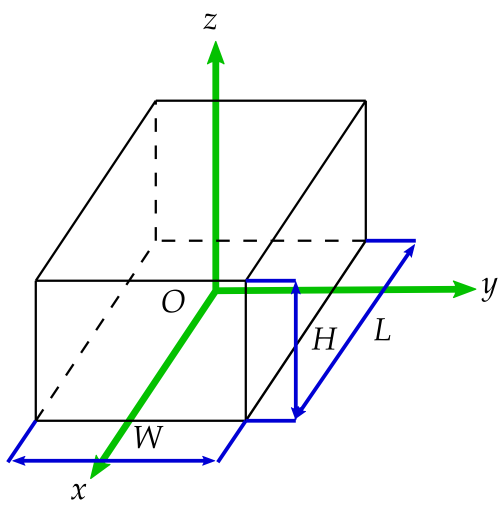


Figure B.1: Rectangular Cuboid