AN INTEGRATED SIMULATION PLATFORM FOR A QUADCOPTER SYSTEM WITH FUZZY-TUNED PID CONTROLLERS

MOAYAD HANI YACOUB ABU RMILAH

DESSERTATION SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING SCIENCE

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **Moayad Hani Yacoub Abu Rmilah**

Registration/Matric No: **KGA130023**

Name of Degree: **Master of Engineering Science**

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

An Integrated Simulation Platform for a Quadcopter System with Fuzzy-tuned PID Controllers

Field of Study: **Automation, Control and Robotics**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:

Designation:

Abstract

Quadcopters are powerful flying robots that are given great interest for their various emerging applications. A quadcopter is an under-actuated system that is naturally unstable and, hence, requires a robust controller to perform rotational and translational maneuvering with high stability. This thesis develops an integrated simulation platform with visualization capability for a quadrotor unmanned aerial vehicle (UAV). The nonlinear dynamic equations of motion for the six degrees of freedom of a quadcopter have been derived from the first principles. The quadcopter dynamical model has been stabilized by PID and fuzzy-tuned PID controllers in order to test and compare their performance. The simulation platform for the quadcopter was developed using MATLAB and Simulink and integrated with the FlightGear Flight simulator in order to visualize the quadcopter motion in real time. The simulation results have shown the effectiveness of the two proposed controllers during altitude, attitude and path tracking control. However, performance indicators of the time response curves for positions and attitude have shown that the fuzzy-tuned PID controller could stabilize the quadcopter more efficiently and provided a faster response and smaller overshoots. Fuzzy-tuned PID controllers have reduced the overshoot along the vertical axis from 10.6% to 4.25%. It enabled the quadcopter to strictly track the planned path with very small tracking errors and time delay. The response of this controller was leading the response of the conventional PID controller by 1.13s during path tracking. The developed simulation platform significantly reduces the time-consuming and intensive testing on expensive quadcopter prototypes and, hence, shortens the total development time and design cost. The simulation platform is a promising tool that is very useful for students and researchers who would like to study and analyze the performance of quadcopter systems.

Abstrak

Quadcopter merupakan robot penerbangan berpotensi tinggi yang diminati kerana kepelbagaiannya dalam aplikasi semasa. Secara semula jadinya quadcopter adalah sistem kurang-digerak yang tidak stabil dan memerlukan pengawal yang teguh untuk melaksanakan gerakan putaran dan translasi dengan kestabilan yang tinggi. Tesis ini akan mengembangkan platform simulasi bersepadu dengan keupayaan visualisasi untuk quatrator kenderaan udara tanpa pemandu (UAV). Persamaan dinamic tidak linear bagi kebebasan gerakan enam darjah quadcopter telah diperolehi daripada prinsip-prinsip pertama. Model dinamik quadcopter telah distabilkan dengan pengawal PID dan PID pelaran-sendiri kabur bagi menguji dan membandingkan prestasinya. Platform simulasi untuk quadcopter telah dibangunkan dengan menggunakan MATLAB dan Simulink yang telah diintegrasikan dengan perisian simulasi penerbangan FlightGear untuk menggambarkan pergerakan quadcopter dalam masa sebenar. Keputusan simulasi menunjukkan kedua-dua pengawal yang dicadangkan mempunyai keberkesanan dalam ketinggian, sikap dan pengesanan kawalan laluan. Walau bagaimanapun, penunjuk prestasi lengkungan tindak balas masa untuk kedudukan dan sikap telah menunjukkan bahawa pengawal PID pelarasan-sendiri kabur boleh menstabilkan quadcopter dengan lebih efisyen, memberi tindak balas yang lebih pantas dan kadar lajakan yang lebih kecil. Pengawal PID pelarasan-sendiri kabur telah mengurangkan kadar lajakan disepanjang paksi menegak daripada 10.6% kepada 4.25%. Ini membolehkan quadcopter untuk mengesan dengan ketat laluan yang dirancang dengan kesilapan pengesanan yang sangat kecil dan kelewatan masa. Riaksi dari pengawal ini semasa ujian pengesanan laluan telah mendahului pengawal PID konvensional dengan sebanyak 1.13s. Platform simulasi yang telah dibangunkan dapat mengurangkan masa dan kos bagi ujian intensif kepada prototaip quadcopter. Ini secara tidak langsung memendekkan masa pembangunan dan kos reka bentuk secara menyeluruh. Plaform simulasi adalah alat yang sangat berguna untuk pelajar dan penyelidik yang ingin belajar dan menganalisa prestasi sistem quadcopter.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

UAV: Unmanned Aerial Vehicle.

MAV: Manned Aerial Vehicle.

MEMS: Micro-Electromechanical Systems.

IMU: Inertial Measurement Unit.

PID Controller: Proportional-Integral-Derivative Controller.

PD: Proportional-Derivative Controller.

LQ: Linear-Quadratic Controller.

List of Symbols

m is the total mass of the quadcopter

g is the acceleration constant

w is the weight

φ is the roll angle

θ is the pitch angle

ψ is the yaw angle

R_{φ} is the roll rotation matrix

R_{θ} is the pitch rotation matrix

R_{ψ} is the yaw rotation matrix

τ_{φ} is the rolling torque

τ_{θ} is the pitching torque

τ_{ψ} is the yawing torque

f is motor thrust

ω is the rotor angular speed

k is the thrust factor

k_d is the friction constant.

$\{x_b, y_b, z_b\}$ is the local reference frame

$\{x_n, y_n, z_n\}$ is the inertial reference frame

$\dot{x}, \dot{y}$ and $\dot{z}$ are the linear speeds along x_n, y_n and z_n axes

$S_{\varphi} = \sin(\varphi)$, $S_{\theta} = \sin(\theta)$, $S_{\psi} = \sin(\psi)$, $C_{\varphi} = \cos(\varphi)$, $C_{\theta} = \cos(\theta)$ and $C_{\psi} = \cos(\psi)$

τ_M is the torque produced by each motor

c is the motor force-to-moment scaling factor

ρ is the mass density of air

A is the propeller cross-section area

c_d is the drag coefficient.

CHAPTER 1. INTRODUCTION

1.1 Introduction

Engineering modeling and simulation have acquired growing importance in solving real-world engineering problems and in bridging the gap between theory and engineering practice. In early days and prior to the emergence of powerful computer software, developing complex systems had been time-consuming and costly especially during the testing and validation stage. These days, the manufacturers of engineering products tend to minimize or eliminate the experimental data during system development using the simulation technology. Simulation has facilitated the design and development of a wide range of advanced products such as automotive systems, medical devices, electronics and telecommunications equipment, industrial machinery and aerospace flight control and avionics. Furthermore, simulation has a great potential to complement the educational process to improve its productivity and cope with the technological changes. This thesis develops an integrated platform for a quadcopter UAV system using the simulation technology.

1.2 Problem Statements

A quadcopter is an under-actuated, nonlinear, naturally unstable dynamical system that requires a high degree of coordination between its four high-speed rotors while hovering. It is difficult to achieve this coordination without designing a robust controller. Conventional PID controllers are widely used in the literature but these controllers are best suited for linear process. In this thesis, a proposed fuzzy-tuned PID controller is used to overcome and tackle these inherent problems.

Furthermore, basic cost, safety and space considerations would limit the availability of real quadcopters as test platforms for students and researchers to analyze and study

such systems. This project proposes a possible alternative to minimize this need by developing an integrated virtual lab for quadcopter systems.

1.3 Project Motivations

Controlling quadcopters is fundamentally difficult. A lot of research has been made to introduce more robust control techniques for such flying robots. This difficulty in controlling them is mainly caused by the high nonlinearity and natural instability of such vehicles. Although these vehicles are naturally unstable, they are still controllable. With the development of small-size microcontrollers, their control has become more feasible.

There are six degrees of freedom for a quadcopter to be controlled; three translational and three rotational. Since the number of actuators (rotors) is less than the number of degrees of freedom, the quadcopter is under-actuated, causing more difficulty in control.

Furthermore, helicopters have little friction to oppose their motion (unlike ground vehicles) and, hence, they have to produce their own damping to stop moving. To achieve this, a high degree of coordination among the four rotors should be accomplished by controlling their speeds properly.

Due to the highly nonlinear behavior of quadcopters, a fuzzy-tuned PID (**P**roportional-**I**ntegral-**D**erivative) controller is expected to be one of the best controllers that suit the proposed control system. Fuzzy logic controllers handle and tackle the parametric and modelling uncertainties of this highly nonlinear system. Fuzzy logic controller with the classical PID controller is expected to achieve a robust control of the vehicle and this work is going to study and test the performance issues of these

controllers on a simulated quadcopter.

Furthermore, hardware is usually available in the academic institutions for teaching purposes; however, basic cost, safety and space considerations would severely limit the availability of equipment (Sevgi, 2006). On the other hand, computer-aided virtual labs can be designed using the simulation technology to help overcome these limitations (Tamayo, Gage, & Walker, 2012). This thesis presents an integrated simulation tool for quadcopter systems, designed using MATLAB and Simulink.

Some of the features that distinguish quadcopters from their counterpart aircrafts are:

a) There are no mechanical linkages for changing the blade pitch to alter the translation and orientation of quadcopters.
b) Simplified design and maintenance and they can be easily assembled.
c) The propellers on the four motors are relatively small in diameter and, hence, less kinetic energy is needed to move them.
d) They are safer than others with less damage to themselves and the environment surrounding them.

1.4 Objectives

The main goal of this work is to fully simulate and control the quadcopter dynamics using fuzzy-tuned PID controllers. There are two objectives which lead to the main goal:

a) To derive the mathematical model of a quadcopter system and design its controller.
b) To develop an integrated simulation platform using MATLAB & Simulink that helps study the performance of the quadcopter and visualize its motion.

1.5 Thesis Outline

The remaining parts of this thesis are going to be organized as follows:

a) Chapter 2, the literature review: this chapter covers the previous works that were done on quadcopters and their control using PID and fuzzy-tuned PID controllers. It gives details about the definition and the concept behind these controllers.

b) Chapter 3, the methodology: Provides detailed discussion on the way how the model is derived. The desired dynamic equations were derived from the first principle. This chapter also provides the methodology of the control design and how the performance of the quadcopter is analyzed and compared.

c) Chapter 4, Simulation platform: This chapter is dedicated for discussion on the simulation and the user interface that were designed for the quadcopter system. It provides the reader with the software tools that helped in creating the system model and the way how these tools are integrated with each other to produce the final well-designed platform.

d) Chapter 5, Results and discussion: This chapter presents the performance data and plots that were obtained from the simulation platform. A comparison study, between the fuzzy-tuned PID controller and the traditional PID control technique, is conducted.

e) Chapter 6, Conclusion: Summarizes the work that has been done and proposes a future plan and some recommendations.

CHAPTER 2. LITERATURE REVIEW

2.1 Introduction

There are many different configurations and types of flying robots or unmanned Aerial Vehicles (UAVs) that move in 3-dimensional space. They have become so popular with a wide variety of shapes, sizes and applications. Figure 2.1 shows some different designs for flying robots. They are different from the ground robots in terms of modelling. They have 6 degrees of freedom (three translational and three rotational) and are actuated by forces. The motion model of ground robots are normally expressed in terms of velocities while flying robots model is expressed in terms of forces and torques (Corke, 2011).

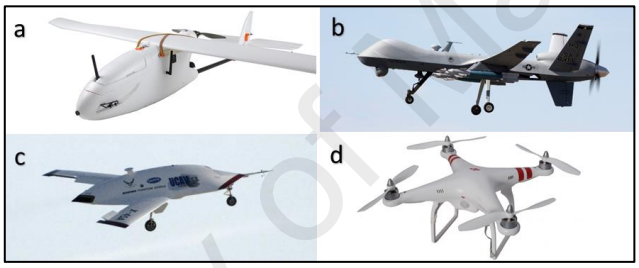

Figure 2.1: Flying robots. a. 3DR Aero skywalker (photo: courtesy of 3D Robotics Inc.) b. MQ-9 Reaper (photo: courtesy of General Atomics Aeronautical). c. Boeing X-45A UAV (photo: courtesy of Boeing). d. Phantom 1 Quadcopter (photo: courtesy of DJI).

Among the types of UAVs are the multi-rotor and fixed-wing UAVs. Multi-rotor UAVs come with different shapes and number of rotors. Within these UAVs are the usual helicopters having two rotors, the main and tail ones, quadrotors with four rotors, hexacopters with six rotors (HiSystems GmbH, n.d.) and octocopters with eight rotors (DJI, n.d.). Some manned aerial vehicles (MAVs) having the same concept of the multi-rotor UAVs emerged with 16 rotors (E-volo, n.d.).

2.2 Structure of a Quadcopter

A quadrotor consists of four equally-spaced, high-speed rotors to which propellers are mechanically attached and used to produce lift forces called thrust. The four rotors are

fixed to a cross frame which is symmetrical about the vertical axis. This symmetry of the geometric form simplifies the model where the important attitude equations are described by a single equation (Pounds, Mahony, & Corke, 2010).

In this thesis, the following assumptions are considered:

a) The quadcopter structure is rigid.

b) The quadcopter structure is symmetrical.

c) The propellers are rigid.

Figure 2.2 shows a typical quadcopter equipped with everything needed to make it able to take off and fly.

Figure 2.2: Typical quadcopter fully assembled (photo: courtesy of 3D Robotics Inc.)

For autonomous flight of a quadrotor, a circuit board equipped with several sensors is needed. These sensors are required to collect data about the quadcopter's internal states and about the environment around surrounding it. The rapid advances and development in the micro-electromechanical systems (MEMS) have made it possible to build light-weight quadcopters that are fully autonomous. The sensors required are:

a) 3D gyro to measure the quadrotor orientation.

b) Accelerometer for positional movement measurements.

c) Magnetometer for compensating orientation measurements.

d) Pressure sensor and sonar sensor for altitude positioning.

e) Camera for object identification, position estimation and surveillance.

The control board is also equipped with a microcontroller which processes the input measurements to produce proper commands to the four rotors. This board and the above sensors constitute the inertial measurement unit (IMU). To manually control the quadrotor, a compatible wireless receiver to receive commands from the base station is attached to it.

2.3 Quadrotors as Closed-loop Systems

Since a quadcopter is a system where sensors for state measurements are used, it is considered to be a feedback system. The mathematical model that is developed describes the behavior of the quadrotor system very well but the noise and disturbance make it hard to use this model as an open-loop system. Its behavior will easily deviate from the desired behavior without using sensors to get feedback data about the internal state of the quadcopter or about the surrounding environment. Sensor fusion is also required for the sake of drift compensation of sensors. For example, the data that is got from the gyro rate sensor tend to drift by time. This is compensated by fusing this data with the data obtained from the accelerometer and magnetometer and, therefore, drift correction is made possible.

Figure 2.3 shows a diagram for a typical feedback system. The goal of almost any feedback system is to maintain the output variable (y) close to the desired input (r). The system receives the reference signal or the desired value (r) and subtracts the measured value detected by the sensor (F) from it to produce the error signal (e). This error is processed by the controller (C) to produce an actuating signal which is fed into the plant

or the quadcopter (*P*), changing its output. The greater the error signal is, the greater the actuating signal (*u*) would be, and the greater the change is in the plant output. This process loops over to reach the goal which is to minimize the error signal and make it close to zero. For a zero error value, the value of the plant output y is very close to the desired value (*r*).

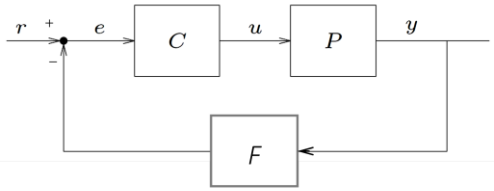

Figure 2.3: Block diagram of a basic feedback control system

2.4 Quadcopter Simulation

Due to the limitations concerning the availability of real quadcopters and their spare parts, researchers have used the simulation technology to investigate such systems. Some have introduced the conceptualization of simulation paradigms that assist in the study of such robotic vehicles as in (R. Veloso et al., 2014). However, the authors did not handle real analysis on the performance of the simulated quadcopter as it is more into conceptualization than implementation.

MATLAB has received the researchers' attention due to its powerful computational capabilities especially in the processing of matrix operations. It has many toolboxes that make the simulation of complex dynamic systems possible. (Ai-Omari, Jaradat, & Jarrah, 2013a) introduced a MATLAB based simulation platform; however, the paper just proves the applicability of the proposed PID controller on the quadcopter model. The 3D visualization and GUI was limited to a few control buttons without providing enough simulation tools that allow controlling the attributes of the quadcopter itself like its mass, radial length, inertial, etc.

What is noticed is that the simulations in literature are just proposed for proving the

effectiveness of a control algorithm and providing an analysis tool for the designers themselves. On the other hand, our proposed simulation platform has the potential to be used by any researcher who would like to study such platform with the help of the proposed flexible integrated simulation environment.

2.5 Quadcopter Control

Although quadcopters are naturally unstable, their control is still feasible thanks to small-size microcontrollers and new sensing and actuating technologies. This feature creates some difficulties in finding the best control algorithm that would control the quadcopter more robustly. The choice of the controller is crucial and is based on the controller performance.

What contributes more in the stability of a quadcopter is the software that runs it. Many researches were conducted to introduce more robust control for unmanned aerial vehicles (UAVs) including quadcopters. Researchers have designed many control techniques like PID controllers, feedback linearization controllers, LQ controllers, nonlinear PD controllers (Bouabdallah, Noth, & Siegwart, 2004), back-stepping controllers (Lee, Ha, & Zuo, 2013), adaptive nonlinear controllers, sliding-mode controllers and fuzzy logic controllers.

2.5.1 PID Controller

PID is an abbreviation for proportional integral derivative. A PID controller is a feedback control system which produces an actuating signal after processing the input error.

Figure 2.4 shows the block diagram of a PID control system. The PID loop offers positive corrections to remove the error from the controllable variable (the input) of the

process.

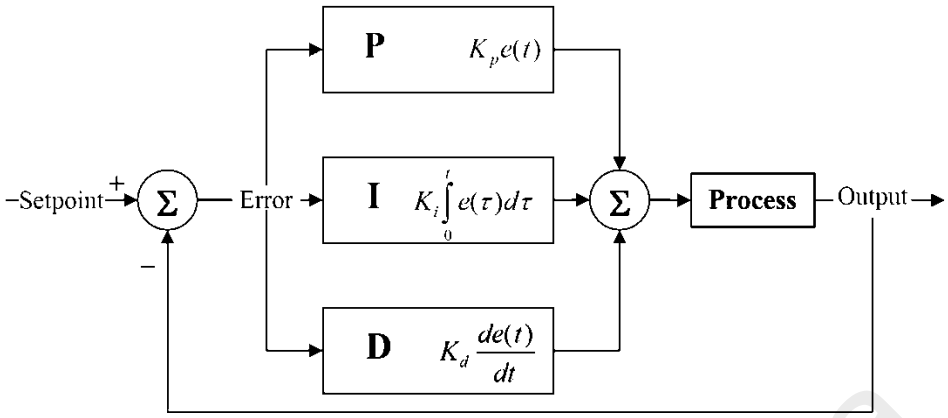

Figure 2.4: Block diagram of a general PID controller

The three control actions, namely *P*, *I* and *D*, cooperate to achieve the goal of reducing the error value in the shortest time possible. The final actuating output signal of the PID controller is produced by summing all of the three terms together as depicted in Figure 2.4. Equation (2-1) shows the mathematical representation of the PID output equation where u(t) is the actuating signal fed into the plant. It is worth mentioning that this signal does not have to be in the same domain as the input signal. For example, if the input error signal represents the error in the quadcopter vertical position, the output can possibly be a signal that represents the force required to minimize the error.

$$u(t) = P_{out} + I_{out} + D_{out} = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \qquad (2\text{-}1)$$

where K_p, K_i and K_d are the proportional, integral and derivative parameters respectively.

There has been a lot of research for decades on these controllers due to their implementation simplicity and control capability. PID controllers have been applied in various control systems including temperature, pressure, speed, water flow rate control. Applying them on a quadcopter system is also suitable and results have shown their applicability in stabilizing the simulated quadcopter.

(Salih, Moghavvemi, Mohamed, & Gaeid, 2010) proposed a PID controller to stabilize the quadcopter. A state space model was designed for two subsystems; one is fully actuated and another is under-actuated. The fully actuated subsystem provides the altitude and yaw dynamics and the under actuated subsystem provides the horizontal positions, pitch and yaw dynamics. Results show that the linear and angular positions have been stabilized. However, a huge overshoot of over 35% occurs in the vertical position. Chattering also appears before the rotation about the altitude is stabilized. These overshoots and chattering are undesired in control systems and must be reduced.

2.5.2 Fuzzy-Tuned PID Controllers

The fuzzy logic algorithm is a problem-solving control system that can process imprecise and noisy input to get a useful output. Fuzzy logic was founded by L. A. Zadeh. It is used to reach to definite conclusions in a simple way based upon noisy, ambiguous, vague, or imprecise input information.

In fuzzy logic, the mapping process from the system input to the output is called fuzzy inference. This process involves three things: membership functions (MF), IF-THEN rules and fuzzy operators. Figure 2.5 shows a block diagram of a fuzzy logic control system.

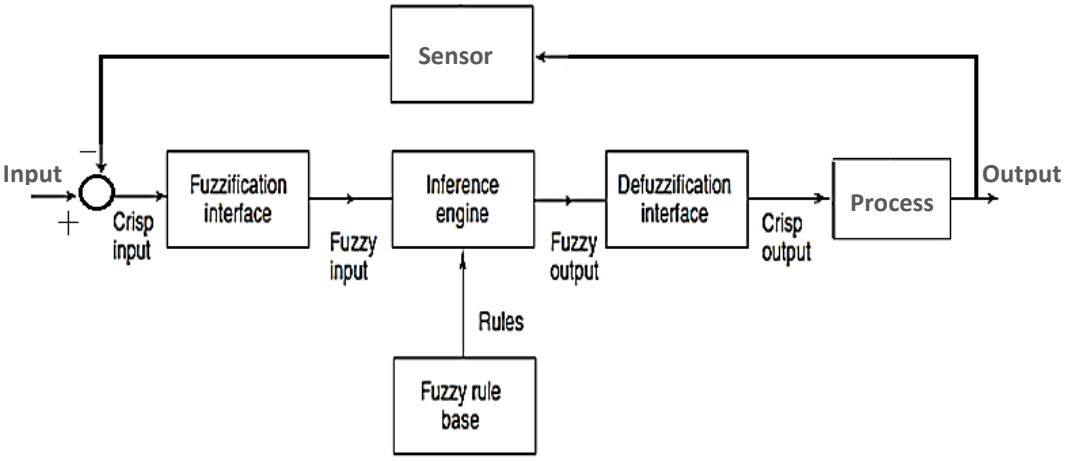

Figure 2.5: Block diagram for a fuzzy logic inference system

There are many types of membership functions used to fuzzify the input and output variables like Gaussian, trapezoidal, triangular, and linear functions. The Sugeno method (Ross, 2010) is used in the output variable to fuzzify it. The implementation of the AND/OR operators in the fuzzy rule evaluation is done using the Mamdani's method (Negnevitsky, 2005).

Since this project implements the Sugeno method to graphically represent the output variable, the weighted average is the technique used to calculate the crisp output. The weighted average is mathematically expressed by:

$$y_{final} = \frac{\sum_{i=1}^{n} w_i y_i}{\sum_{i=1}^{n} w_i}$$

Where *i* is the current rule number, *n* is the number of rules, *w* is the degree of truth and *y* is the crisp output value that changes the PID parameters.

A fuzzy logic controller has several features that make it a good controller for many applications. These features might be a strong reason behind the huge wave of interest among researchers in these controllers. These features include:

a) Precision of the sensors is insignificant.

b) Possible to be programed to fail safely if a part of the system fails.
c) The output is smooth and reacts with a wide range of input variations.
d) User-defined rules that govern the controlling criteria, and can be modified by changing the governing rules.
e) Not limited to a certain number of inputs or outputs.
f) Inexpensive sensors do not affect, so this decreases the overall cost.
g) Able to control nonlinear systems.

For the great advantages of fuzzy logic controllers over others and for their suitability on non-linear systems, fuzzy logic is tested on the quadcopter system in hand. Researchers have used it in many applications and some have tried to stabilize quadcopters with them. A paper by (Abeywardena, Amaratunga, Shakoor, & Munasinghe, 2009) proposed a simulation platform for a quadcopter whose roll and pitch angles, hence translation along x and y axes, were stabilized using fuzzy logic controllers. It assumes that the altitude and heading stabilization is manually handled by the external pilot. The inputs to the fuzzy controllers in this paper are the velocity and acceleration in the inertial frame and the output is a control signal that alters the roll and pitch angles accordingly. Using the velocity and acceleration as inputs to this fuzzy controller limits its functionality to velocity control only. In the proposed controller in this thesis, the error in translational and angular positions are the input to the fuzzy controller, giving the possibility for autonomous path tracking. The altitude and heading stabilization is also handled autonomously.

(Santos, López, & Morata, 2010) also presented a simulation platform for a quadcopter model controlled by an intelligent fuzzy controller. This controller determines the power used by each motor to alter the altitude, roll, pitch and yaw angles

based on desired commands.

(Seidabad, Vandaki, & Kamyad, 2014) proposed a PID controller whose coefficients are tuned by a fuzzy controller. Their paper presents quadcopter simulation based on MATLAB. Here, noisy sensor measurements under turbulence was simulated for the roll angle. PID controllers alone were not suitable and could not control the quadcopter under this turbulence. The application of fuzzy logic, however, did well in rejecting the noisy sensor measurements. The simulation proved the ability of fuzzy logic controllers to stabilize the quadcopter more efficiently. However, this study considered the control of the horizontal axes for lateral movements without controlling the altitude and yaw rotation.

(Sangyam, Laohapiengsak, Chongcharoen, & Nilkhamhang, 2010) studied the issue of tuning the PID controllers using the fuzzy logic controller. The authors chose the quadcopter system to test the proposed controller on it. The research focus was on analyzing the efficiency of the conventional and self-tuning PID controllers in tracking predefined paths. This study explains the limitation of a PID controller and their applicability on only certain systems operating under certain circumstances (e.g. no load and with load cases). These conclusions are convincing because this controller is attributed as linear. The authors only used a single fuzzy input without using the error rate as a second input. This might justify the deviation that appears in the self-tuning PID plots.

(Desa, Ahmed, & Azfar, 2013) also addressed the issue of controlling a quadcopter system by a fuzzy-tuned PID controller. The inputs to the fuzzy controllers are the rotor speed error and its derivative.

2.6 Quadcopter Applications

Quadcopters, like the one shown in Figure 2.1d, are UAVs that have been given a great interest among researchers and also in industry. They are widely available either as proprietary or as open-source projects. They are more flexible and adaptable platforms for aerial research.

One of the main reasons behind the huge wave of interest among researchers in quadrotor aircrafts is due to their recently emerging applications. Their advantageous features have made them remarkably suitable to a wide variety of useful applications. Possible applications include fire-fighting operations (Pandia, 2014), search and rescue operations, news and event photography, inspection of power lines and wind turbines and oil rigs, security and law enforcement surveillance and military observation to locate an enemy. Gadda and Patil introduced the application of border monitoring using a quadcopter that is controlled by an IR remote (Gadda & Patil, 2013). The mounted camera broadcasts real-time audio-visual to a PC in the base station. Some of these applications are really challenging, a feature that necessitates a more robust control to be developed. Some researchers have also used quadcopters to prove the applicability of other technologies on such powerful flying robots. The project proposed in (LaFleur et al., 2013) is one example. Here, a quadrotor was controlled in 3-dimensional space without any hand gadget to remotely control it. The human brain is used as the source of commands instead.

2.7 Study Plan

Although researchers have been working on developing robust control techniques for quadcopters, there is a clear research gap in providing an integrated simulation platform that supports motion visualization for a quadcopter robotic vehicle. Little research has

been made on the application of fuzzy-tuned PID controllers on quadcopter systems compared to the huge research that has been made on applying them to other dynamic control systems. According to the study made, a better performance in terms of fast response and noise rejection can be obtained if the controller is well used and applied properly to the model.

This work presents a new simulation platform with visualization and control GUI for a quadcopter system. The fuzzy-tuned PID controllers applied on a derived quadcopter model are further studied and analyzed as a serious attempt to produce even a better and robust control with high performance.

CHAPTER 3. METHODOLOGY

3.1 Introduction

The quadcopter control system is developed using the model-based design approach. In this approach, the development of the system is manifested in the following four steps:

a) Modelling the quadcopter dynamic or simply the plant.

b) Analyzing and synthesizing controllers for the modelled plant.

c) Simulating both the plant and its controller using powerful simulation tools.

d) Deploying the controller on the actual system.

However, based on the objectives of this thesis, the fourth phase of this approach is left for future work.

This chapter discusses the methodology for the proposed project. Discussion will be made on how the simulation is made and how the mathematical model and control is developed. Lastly, the control techniques and the way how to analyze their performance are introduced.

3.2 Model Development

Understanding the dynamics of the quadcopter and its working mechanism is the first step towards the development of the model. This understanding comes after deeply searching previous similar works in the literature. In this stage, the input and output variables and the relationship between them are clearly identified and assumptions are carefully put so that a simplified model is produced without losing generality.

In this modelling stage, investigation will be made on the relevant physical laws like Newton's motion laws, the aerodynamics of the air, the kinematics of the quadcopter and the limitations of the variables in order to reach approximated model equations that

well represent the quadcopter system. Different motion representations can be used to develop the final differential equations of the aircraft.

In this subsection, detailed explanation and thorough discussion about the working mechanism of a quadcopter and its mathematical model are made. Firstly, discussion on the adopted coordinate frames and the transformation between them is taken into account and, then, the mathematical model for the system dynamics is developed from the first principles.

3.2.1 Reference Coordinate Frames

Quadcopters receive commands remotely from the base station on ground. Any movement performed by this flying robot is made with reference to this base station. It is represented by a fixed inertial frame to which Newton's motion laws apply. In reality, this reference frame is not inertial but it is assumed to be so after neglecting its rotation with the earth (this is called the Coriolis Effect). This assumption is valid for the proposed project for the following reasons:

a) The earth completes one rotation per day which is a small speed relative to its mass and size.
b) The quadcopter does not fly large distances and for long periods of time.

Figure 3.1 shows a diagram of a quadcopter with the frames being well defined on it. Two different coordinate frames are used, namely the body-fixed coordinate frame or the local frame $\{x_b, y_b, z_b\}$ and the inertial reference frame or the global frame $\{x_n, y_n, z_n\}$. Both of these reference frames are orthogonal where each axis is perpendicular to the other. The body-fixed coordinate frame is fixed to the quadcopter, and moves and rotates with it, making it a non-inertial frame. This frame is assumed to coincide with

the center of mass of the quadcopter.

Commands are sent from the base station and the desired positions of the quadcopter are referenced to the inertial frame. On the other hand, the four actuators are attached to the body-fixed coordinate frames and, hence, the thrust f_i produced by motor *i* always points along the local vertical axis z_b. Based on the desired positions referenced to the inertial frame and the desired heading, the four motors, to which the body-fixed coordinate frame is attached, must receive proper actuating signals to change the orientation and/or translation of the quadcopter accordingly. This necessitates the use of a transformation technique between the two coordinate frames. Next section discusses this matter in depth.

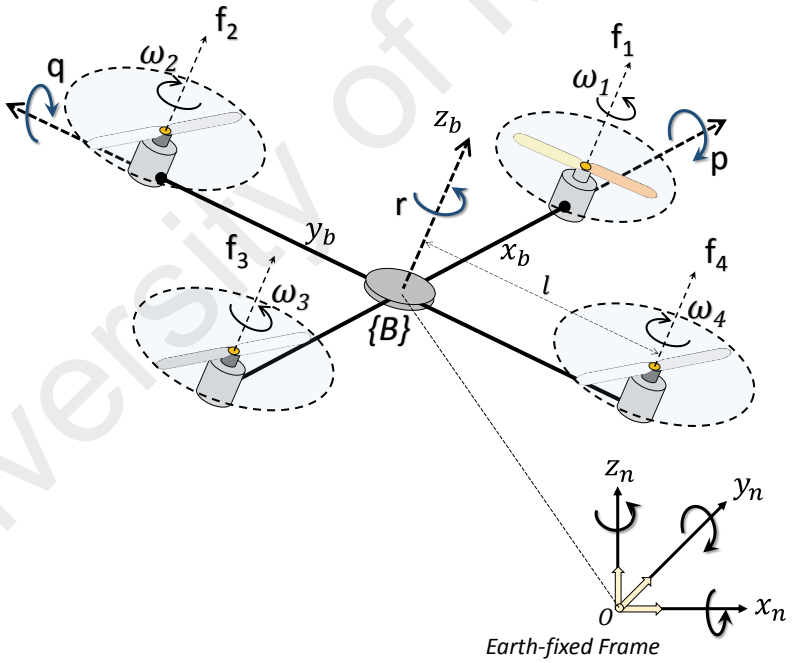

Figure 3.1: Quadcopter diagram showing global and local coordinate frames

3.2.2 Transformation between Frames

It is important to find a relation between the body-fixed coordinate frame and the inertial reference frame. Fortunately, it is easy to transform between these two frames with the help of rotation matrices and Euler angles.

- Why rotation matrices are needed

A rotation matrix is a convenient tool to make the frame transformation in the Euclidean space. Whatever orientation the quadcopter is in, the thrusts and torques produced by the motors must affect the quadcopter in a proper way.

A good and simple example to explain the need for the transformation is the force due to gravity or the weight. Weight always points downward along the vertical axis z of the inertial reference frame regardless of the orientation of the quadcopter. When the quadcopter is rotated about any of the horizontal axes x_n and y_n (aka non-zero roll and pitch angles), the weight has vector components in the local axes. Figure 3.2 illustrates this example for a case where the quadcopter has made a rotation of θ radians counterclockwise about the y axis. The quadcopter must always compensate for the gravity, otherwise it will not stay in air. The weight components w_x and w_z must be known to be able to find the required thrusts that the four motors have to produce so that the quadcopter fly while compensating for the gravity. Here, it is quite simple to calculate the components w_x and w_z as follows:

$$w_x = -w\sin\theta = mg\sin\theta\ ;\ w_z = w\cos\theta = -mg\cos\theta \qquad (3\text{-}1)$$

Where m is the total mass of the quadcopter and g ≈ 9.81m/s^2 is the acceleration constant due to gravity on earth's surface.

Equations (3-1) are valid for this special case but when the quadcopter performs complex movements and rotates about more than one axis, a more efficient transformation technique is needed. Euler angles and quaternions can be used to find the transformation matrices. Here, the choice was made on Euler angles for this purpose.

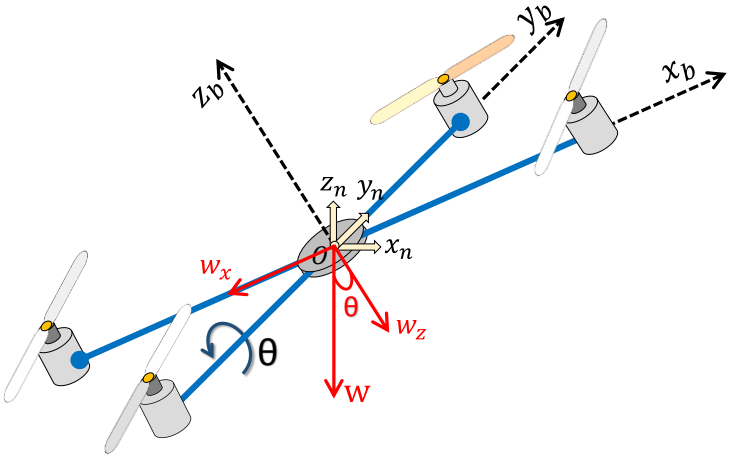

Figure 3.2: Quadcopter diagram with weight vector components

- Euler Angles and Transformation Matrix

Any orientation can be achieved by composing three elemental rotations, starting from a known orientation. Composing them would produce the transformation matrix that can transform vectors from the body-fixed coordinate frame to the inertial reference frame and vice versa. Euler angles can be used to create these elemental rotations. There are three Euler angles:

a) The roll angle about the x axis denoted by φ.

b) The pitch angle about the y axis denoted by θ.

c) The yaw angle (also called heading) about the z axis denoted by ψ.

Equations (3-2), (3-3) and (3-4) show the rotation matrices when for rolling, pitching and yawing rotations. Euler angles themselves are not orthogonal but the derived rotation matrices are orthogonal and, hence, they can be concatenated by multiplying them in order to get the transformation matrix for a composite orientation.

$$R_{\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\varphi & -sin\,\varphi \\ 0 & sin\,\varphi & cos\,\varphi \end{bmatrix}, \qquad (3\text{-}2)$$

$$R_\theta = \begin{bmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{bmatrix}, \quad (3\text{-}3)$$

and

$$R_\psi = \begin{bmatrix} cos\,\psi & -sin\,\psi & 0 \\ sin\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3\text{-}4)$$

There are twelve possible transformation matrices that can be adopted for the quadcopter application. Changing the sequence of rotation gives a different transformation matrix because matrix multiplication is not, in general, commutative. The sequence of rotation chosen is yaw, pitch and then roll or simply *ZYX*. Using this sequence, any target orientation can be reached starting from the known reference orientation $\{x_n, y_n, z_n\}$ as follows (see Figure 3.3):

a) The $\{x_b, y_b, z_b\}$ system rotates by ψ radians about the z_b axis (which coincides with the z_n axis). The y_b axis now lies on the line of nodes N_1. The range of rotation is: $-\pi < \psi < \pi$.

b) From the *x*-*y* plane, the $\{x_b, y_b, z_b\}$ system rotates about the now rotated y_b axis by θ radians. The x_b axis is now in its final orientation. The range of rotation is: $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$.

c) The $\{x_b, y_b, z_b\}$ system rotates a third time about the new x_b axis by φ radians. The range of rotation is: $-\frac{\pi}{2} < \varphi < \frac{\pi}{2}$.

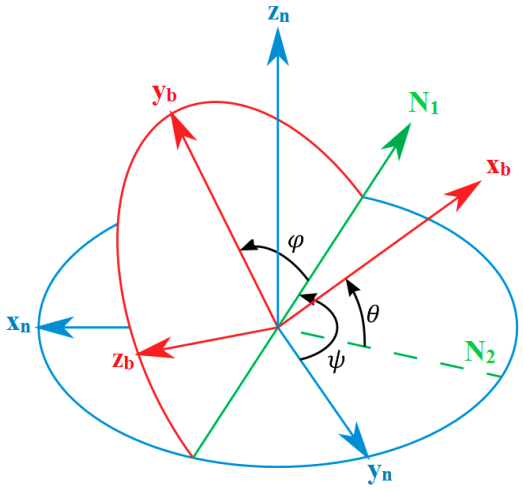

Figure 3.3: *ZYX* rotation convention of Euler angles

Post-multiplying the above three rotation matrices would give the following transformation matrix.

$$R_t(\varphi,\theta,\psi) = R_\psi \times R_\theta \times R_\varphi =$$

$$\begin{bmatrix} C_\theta C_\psi & C_\psi S_\theta S_\varphi - S_\psi C_\varphi & C_\psi S_\theta C_\varphi + S_\psi S_\varphi \\ S_\psi C_\theta & S_\psi S_\theta S_\varphi + C_\psi C_\varphi & S_\psi S_\theta C_\varphi - C_\psi S_\varphi \\ -S_\theta & C_\theta S_\varphi & C_\theta C_\varphi \end{bmatrix} \quad (3\text{-}5)$$

where $S_\varphi = \sin(\varphi)$, $S_\theta = \sin(\theta)$, $S_\psi = \sin(\psi)$, $C_\varphi = \cos(\varphi)$, $C_\theta = \cos(\theta)$ and $C_\psi = \cos(\psi)$.

Equation (3-5) is used to transform vectors from the body-fixed coordinate frame to the inertial reference frame. It is interesting to know that this matrix is orthogonal and, hence, the inverse matrix ${R_t}^{-1}(\varphi,\theta,\psi)$ is simply equal to the transpose of Equation (3-5). Its inverse matrix shown in Equation (3-6) is used to transform vectors from the inertial reference frame to the local body-fixed coordinate frame.

$${R_t}^{-1}(\varphi,\theta,\psi) = {R_t}^{T}(\varphi,\theta,\psi) =$$

$$\begin{bmatrix} C_\theta C_\psi & S_\psi C_\theta & -S_\theta \\ C_\psi S_\theta S_\varphi - S_\psi C_\varphi & S_\psi S_\theta S_\varphi + C_\psi C_\varphi & C_\theta S_\varphi \\ C_\psi S_\theta C_\varphi + S_\psi S_\varphi & S_\psi S_\theta C_\varphi - C_\psi S_\varphi & C_\theta C_\varphi \end{bmatrix} \quad (3\text{-}6)$$

For the weight example illustrated in Figure 3.2, Equations (3-1) can be found by multiplying the rotation matrix in Equation (3-6), with $\psi = 0$ and $\varphi = 0$, by the weight vector $\{0\ 0\ w\}^T$.

3.2.3 Forces and Torques

Depending on the mode of operation, proper control commands are sent to the four rotors so that the quadrotor rotates or translates as desired. It can be in hovering, ascending or descending, rolling, pitching or yawing mode or a mixture of these modes (Ai-Omari, Jaradat, & Jarrah, 2013b).

For the calculation of the quadcopter position and orientation in the inertial reference frame from the body-fixed coordinate frame, forces and torques affecting the system are studied, indicating that the method used for that calculation is not direct or inverse kinematics.

The thrust produced by a motor is directly proportional to the square of its angular speed. Equation (3-7) shows this relation.

$$f_i = k\omega_i{}^2, i = 1, 2, 3, 4 \qquad (3\text{-}7)$$

Where f_i is the thrust generated by the i^{th} rotor, ω_i is the angular speed, measured in radians, of the i^{th} rotor and k is some appropriately dimensioned constant called thrust factor and is ideally the same for all of the motors.

In the following subsections, discussion is made on how the quadcopter works in different translation and rotation modes. The related model equations like forces and torques are carefully derived.

- Hovering, Upward and Downward Modes

To move the quadrotor along the vertical axis, the speeds of all motors are either increased to raise the quadcopter up or decreased to lower it down. Hence, the overall thrust is no longer equal to the weight of the quadrotor.

Maintaining the height of the quadcopter and keeping it in the hovering state requires the fulfillment of following two conditions:

a) The total thrust produced by the propellers must compensate for the earth's gravity, hence, the net force on the quadcopter is zero.
b) The torques produced by the four motors must exactly sum to zero.

To achieve the second condition, opposite motors rotate in the same direction while the neighboring motors rotate in the opposite direction. The angular speeds of the four motors must be the same.

This opposite rotation is meant to balance the torques about the vertical axis and, hence, it cancels the counteracting aerodynamic torques (anti-torque effect). In traditional helicopters, the tail rotor compensates for the counteracting torque produced by the main rotor.

The argument above indicates that putting the quadcopter in any of these modes involves controlling the speeds of the four motors together. The change in the speeds of the four motors $\Delta\omega$ must be the same.

Figure 3.4 illustrates the aforementioned three modes (hovering, flying up and flying down). The arrows in the figure indicate the motor rotation direction and its thickness indicates the speed value.

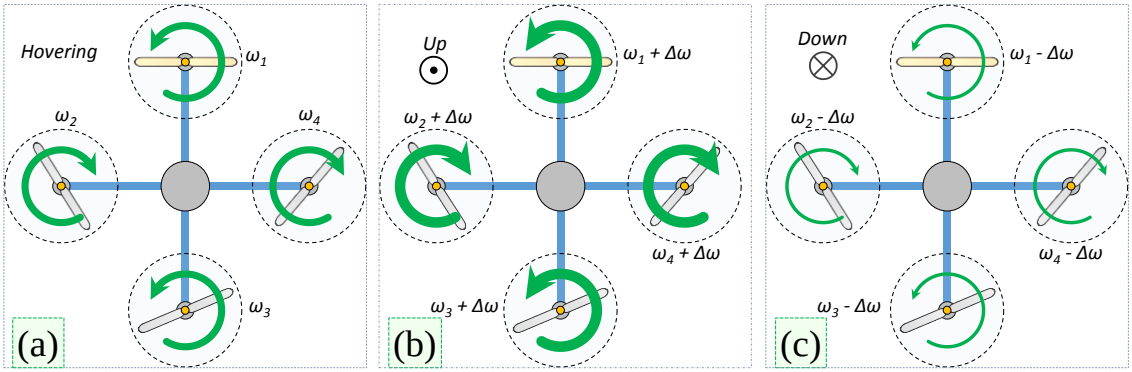

Figure 3.4: Quadcopter diagram for movements along the vertical axis. a. Hovering mode. b. Ascending mode. c. Descending down.

The total thrust u, measured in Newton, produced by the four propellers is given by Equation (3-8). This thrust always points towards the positive z_b axis which is to be transformed to the inertial reference coordinate frame as will be discussed later.

$$u = f_1 + f_2 + f_3 + f_4 \tag{3-8}$$

To express u in terms of the motor speeds, Equation (3-7) is substituted in Equation (3-8) to get:

$$u = k{\omega_1}^2 + k{\omega_2}^2 + k{\omega_3}^2 + k{\omega_4}^2 = k\sum_{i=1}^{4} {\omega_i}^2 \tag{3-9}$$

Equation (3-9) indicates that the greater the motors speeds the faster it goes up due to the higher force generated and the lower the motors speed, the faster it goes down.

- Roll Moment

The concept of making a roll rotation is greatly similar to that for pitch rotation. To achieve a roll rotation which is a rotation about the x axis, the control command is to be issued such that there is a speed difference between motor 2 and motor 4 while the speeds of motors 1 and 3 remain unchanged. To rotate about the positive x axis, the speed of the left motor (motor 2) must be higher than the speed of the right motor (motor 4). To achieve a negative roll angle, the opposite is true; the speed of the right motor must be higher than the speed of the left motor.

When the quadcopter rolls with a certain angle that is not zero, it translates laterally; either to the right or to the left. To translate to the right, the quadcopter must rotate with a certain positive roll angle (see Figure 3.5). To move to the left, it must rotate with a certain negative roll angle.

The torque difference between the left and right motors gives the torque required to achieve a rolling rotation. Equation (3-10) gives the relation between the torque τ_φ in the body frame, which is the torque required to roll, and the motor thrusts controlling the roll rotation.

$$\tau_\varphi = \tau_2 - \tau_4 = (f_2 - f_4)l \tag{3-10}$$

where τ_i is the torque about the quadcopter center of mass generated by the thrust of the i^{th} motor and l is the shortest distance in meters from the center of mass of the quadcopter to any rotor's shaft.

Writing Equation (3-10) in terms of motors speeds would be convenient and this done by substituting Equation (3-7) in it to obtain Equation (3-11).

$$\tau_\varphi = (k{\omega_2}^2 - k{\omega_4}^2)l = lk({\omega_2}^2 - {\omega_4}^2) \tag{3-11}$$

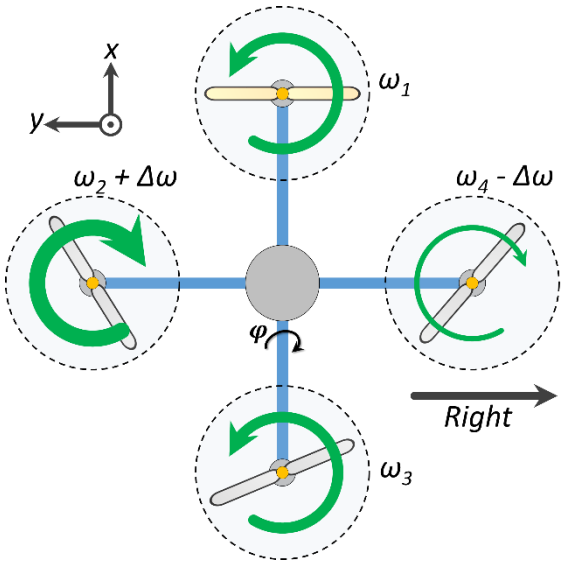

Figure 3.5: Quadcopter diagram for right translation (rolling)

- Pitch Moment

To achieve a pitch rotation which is a rotation about the local y axis, the control command is to be issued such that there is a speed difference between motor 1 and motor 3 while the speeds of motors 2 and 4 remain unchanged. To rotate about the positive y axis (clockwise rotation), the speed of motor 3 must be higher than the speed of motor 1. To achieve a negative pitch angle (counterclockwise rotation), the speed of motor 1 must be higher than the speed of motor 3.

When the quadcopter has a certain pitch angle that is not zero, it translates longitudinally; either forward or backward. To translate forward, the quadcopter must rotate with a certain positive pitch angle (see Figure 3.6). To move backward, it must rotate with a certain negative pitch angle.

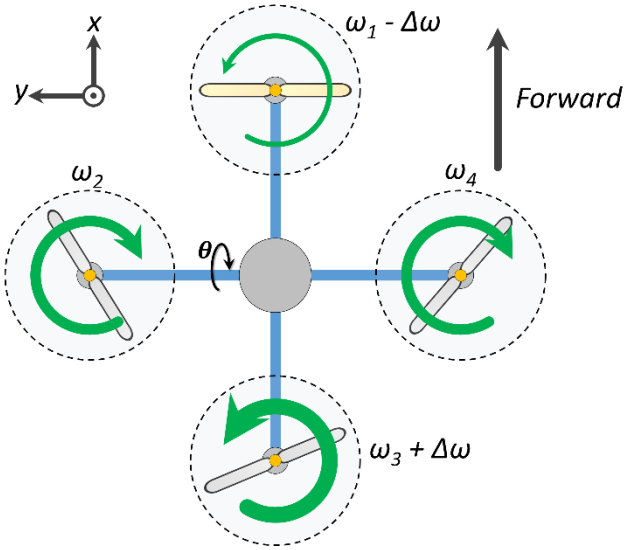

Figure 3.6: Quadcopter diagram for forward translation (pitching)

For a certain quadcopter setup, the following two things affect the speed by which the quadcopter translates in the horizontal plane:

a) The speeds of the motors.

b) The amount of the angle by which the quadcopter rotated.

The torque about the center of mass (the pivot) is simply the force multiplied by the distance from the motor shaft to the pivot. The torque difference between Motor 1 and 3 gives the pitch torque. Equation (3-12) gives the relation between the torque τ_θ about the y_b axis and the motor thrusts controlling the pitch rotation.

$$\tau_\theta = \tau_3 - \tau_1 = (f_3 - f_1)l \tag{3-12}$$

Equation (3-13) which is obtained by substituting Equation (3-7) in Equation (3-12) would give the torque about the local y axis in terms of motors speeds.

$$\tau_\theta = (k{\omega_3}^2 - k{\omega_1}^2)l = lk({\omega_3}^2 - {\omega_1}^2) \tag{3-13}$$

- Yaw Moment

To make a yaw rotation, the torque about the vertical axis of the body frame must be unbalanced. To achieve a yaw rotation about the vertical axis, the second condition that

the quadcopter has to fulfil while hovering must be invalidated. To turn the quadcopter left, about the positive z axis (counterclockwise rotation),

a) The speed of the right-rotating motors (motors 2 and 4) is increased by $\Delta\omega$.

b) The speed of the left-rotating motors (motor 1 and 3) is decreased by the same speed difference $\Delta\omega$.

To turn the quadrotor right about the negative *z* axis (clockwise rotation), the opposite is done, that is:

a) The speed of the left-rotating motors (motors 1 and 3) is increased by $\Delta\omega$.

b) The speed of the right-rotating motors (motor 2 and 4) is decreased by $\Delta\omega$.

It is clear that controlling the yaw angle involves the four motors to be properly commanded. Figure 3.7 illustrates this rotation when the quadcopter turns about the positive z axis. The speed difference $\Delta\omega$ must be equal to maintain the vertical position of the quadcopter while turning and, therefore, pure turning about the vertical is guaranteed.

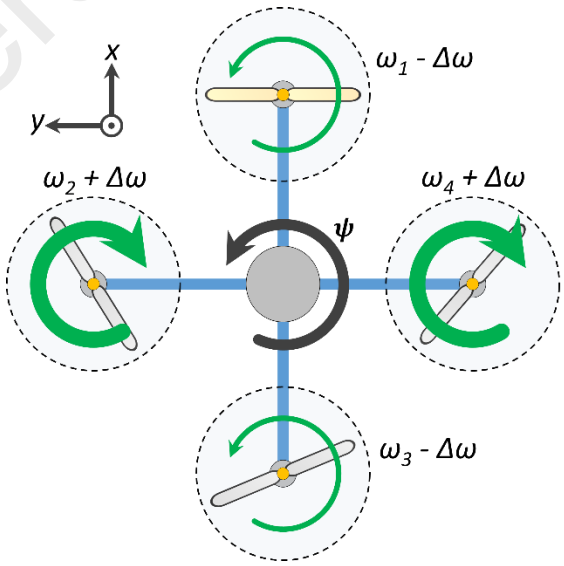

Figure 3.7: Quadcopter diagram for heading change

Equation (3-14) gives the relation between the yawing torque τ_ψ in the body frame and the four motors thrusts. The negative signs appear for f_2 and f_4 because the torque produced by them is about the negative z axis.

$$\tau_\psi = \sum_{i=1}^{4} \tau_{M_i} = c(f_2 - f_1 + f_4 - f_3) \tag{3-14}$$

where τ_{M_i} is the torque produced by the i^{th} motor about its shaft and c is a constant known as force-to-moment scaling factor.

The relationship between this torque and the motors speed is derived by substituting Equation (3-7) into Equation (3-14) and is given by Equation (3-15):

$$\tau_\psi = c(k{\omega_2}^2 - k{\omega_1}^2 + k{\omega_4}^2 - k{\omega_3}^2) = ck\sum_{i=1}^{4}(-1)^i{\omega_i}^2 \tag{3-15}$$

- Translations Associated with Pitch and Roll Rotations

When the quadcopter makes a roll or pitch rotation, the total thrust given by Equation (3-9) becomes having a vector component along the horizontal plane. This force component accelerates the quadcopter forward, backward, left or right along the *x*-*y* plane.

For example, the backward translation can be accomplished by making a negative pitch rotation and, hence, there is a vector component that translates the quadcopter along the *x* axis. Figure 3.8 illustrates this example where u_x is the vector component that moves the quadcopter backwards and u_z compensates for the weight of the quadcopter to maintain its vertical position.

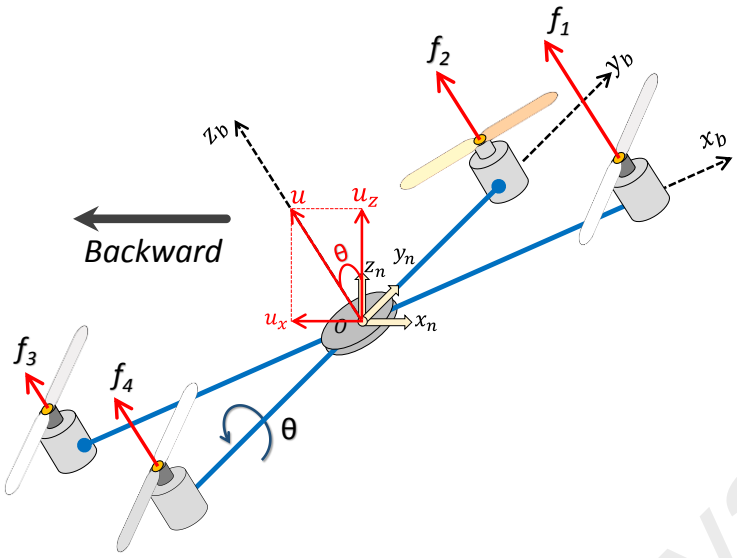

Figure 3.8: Vector components for the total thrusts generated (backward translation)

3.2.4 Control Distribution Matrix

For the sake of simplicity, Equations (3-8), (3-14), (3-12) and (3-10) can be augmented in a matrix form as shown in Equation (3-16).

$$\begin{bmatrix} u \\ \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & l & 0 & -l \\ -l & 0 & l & 0 \\ -c & c & -c & c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{3-16}$$

Equation (3-16) can be rewritten in terms of the angular speeds ω of the motors as shown in Equation (3-17). Substituting Equation (3-7) into Equation (3-16), the following is derived:

$$\begin{bmatrix} u \\ \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k & k & k & k \\ 0 & kl & 0 & -kl \\ -kl & 0 & kl & 0 \\ -kc & kc & -kc & kc \end{bmatrix} \begin{bmatrix} {\omega_1}^2 \\ {\omega_2}^2 \\ {\omega_3}^2 \\ {\omega_4}^2 \end{bmatrix} \tag{3-17}$$

This equation summarizes the forces and torques in terms of the motors angular speeds. With proper mapping between the voltages applied to the motors' terminals and the speeds, the controller would be able to produce a proper voltage level such that the motors rotate at the desired speeds according to the mode of operation.

Knowing the working principle of the quadcopter and deriving its force and torque equations are the keystone for the development of the quadcopter model and its dynamic equations. Next section discusses this in details.

3.2.5 Quadcopter Dynamic Equations

Having derived the force equation in the local frame, it becomes convenient to find the differential equations of motion like the linear acceleration considering the net forces and angular accelerations considering the acting net torques on the quadcopter.

- Linear Accelerations, Velocities and Positions

The forces that affect the linear motion of the quadcopter (e.g. the motion along the vertical axis and the horizontal plane) include the total force u given by equation (3-8) and the quadcopter weight. Rewriting u in a matrix form gives the vector $\vec{u}$:

$$\vec{u} = \begin{bmatrix} 0 \\ 0 \\ \mathrm{u} \end{bmatrix}$$

The *x* and *y* components of vector $\vec{u}$ are zero because it only points along the vertical *z* axis of the local frame. The linear accelerations of the quadcopter are found with respect to the inertial reference frame. Therefore, the body frame to inertial frame transformation with the help of Equation (3-5) is made on this vector $\vec{F}_R$ as follows:

$$\vec{F}_R = R_t(\varphi, \theta, \psi) \times \begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix} = \begin{bmatrix} sin\,\psi\, sin\,\varphi + cos\,\psi\, sin\,\theta\, cos\,\varphi \\ sin\,\psi\, sin\,\theta\, cos\,\varphi - cos\,\psi\, sin\,\varphi \\ cos\,\theta\, cos\,\varphi \end{bmatrix} u \qquad (3\text{-}18)$$

In addition to the thrust force, the friction will be modeled too. Friction is proportional to the square of the relative speed in every direction between the quadcopter itself and the air. In general, the drag force from the fluid dynamics is good enough to give the frictional force F_f in Equation (3-19):

$$F_f = \frac{1}{2}\rho v^2 c_d A \tag{3-19}$$

where ρ is the mass density of air, v is the speed of the quadcopter relative to air, A is the reference area (propeller cross-section) and c_d is the drag coefficient.

The speed v is assumed to be equal to the linear speed of the quadcopter because air speed is ignored. The air friction force in Equation (3-19) can be simplified for the proposed quadcopter model to:

$$\vec{F}_f = \begin{bmatrix} -k_d\dot{x}^2 \\ -k_d\dot{y}^2 \\ -k_d\dot{z}^2 \end{bmatrix} \tag{3-20}$$

Where k_d is the friction constant and $\dot{x}, \dot{y}\ and\ \dot{z}$ are the linear speeds along x_n, y_n and z_n-axes of the inertial coordinate frame.

Newton's laws apply on the inertial reference frame. The linear acceleration of the quadcopter is proportional to the net force acting on the quadcopter. From this Newton's law, the linear motion can be defined as:

$$m\vec{a} = m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \sum \vec{F} = \vec{F}_R + \vec{w} + \vec{F}_f \tag{3-21}$$

where $\ddot{x}, \ddot{y}\ and\ \ddot{z}$ are the second derivatives of the positions x, y and z and represent the linear accelerations along the x_n, y_n and z_n-axes and $\vec{w}$ is the weight vector.

Substituting Equation (3-18) and the weight vector in Equation (3-21) gives the desired linear accelerations in the inertial reference frame,

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} sin\,\psi\,sin\,\varphi + cos\,\psi\,sin\,\theta\,cos\,\varphi \\ sin\,\psi\,sin\,\theta\,cos\,\varphi - cos\,\psi\,sin\,\varphi \\ cos\,\theta\,cos\,\varphi \end{bmatrix}\frac{u}{m} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} - \begin{bmatrix} k_d\dot{x}^2 \\ k_d\dot{y}^2 \\ k_d\dot{z}^2 \end{bmatrix} \tag{3-22}$$

Equations (3-23), (3-24) and (3-25) give the accelerations relations individually.

$$\ddot{x} = \frac{\sin\psi \sin\varphi + \cos\psi \sin\theta \cos\varphi}{m} u - k_d \dot{x}^2 \tag{3-23}$$

$$\ddot{y} = \frac{\sin\psi \sin\theta \cos\varphi - \cos\psi \sin\varphi}{m} u - k_d \dot{y}^2 \tag{3-24}$$

$$\ddot{z} = \frac{\cos\theta \cos\varphi}{m} u - g - k_d \dot{z}^2 \tag{3-25}$$

Linear velocities $\dot{x}, \dot{y}$ and $\dot{z}$ are obtained by integrating the linear accelerations in Equation (3-22) and the linear positions are got by double-integrating the linear accelerations. This is done by the simulation software as will be discussed in the simulation chapter.

- Relation Between the Body Angular Rates and Euler Speeds

The previous subsection introduced the linear differential equations in which the total thrust was involved without considering the torques. In this subsection, the angular differential equations will be derived.

The gyroscope sensors report data with respect to IMU frame. This means that the reported angular rates are with respect to the body coordinate frame of the quadcopter because the IMU chip is attached to this frame. This suggests that the derivative of Euler angles are not the ones reported by the gyroscope sensor but they can be calculated from the reported sensor data after some conversion. Let *p*, *q* and *r* be the angular rates of the quadcopter reported by the rate gyro sensor. Therefore, the inequality is mathematically represented as follows:

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \neq \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3-26}$$

It was already mentioned that Euler angles are non-orthogonal because they are sequentially made with intermediate axes being involved in the rotations. Therefore, it is of interest to find a conversion matrix which can be used to find Euler angles. The torques developed in Section 3.2.3 will be involved to derive the angular differential equations.

To get the matrix that transforms the derivatives of Euler angles to the body angular rates, the last two elemental rotations, made in the rotation order adopted, are the ones used.

The components of Euler speeds on each body axis are found and then combined. Here, the inverse of the rotation matrices in Equations (3-2), (3-3) and (3-4) will be used because the roll, pitch and yaw rotations are reversed as seen from the body coordinate frame. Rolling appears to be the first rotation made starting from the reached orientation.

- Components of $\dot{\varphi}$ about the body axes: Intuitively, this Euler speed is made about the x axis of the body frame and has no components about the y and z-axes of the body frame:

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}_{\varphi} = \begin{bmatrix}\dot{\varphi}\\ 0\\ 0\end{bmatrix} \tag{3-27}$$

- Components of $\dot{\theta}$ about the body axes: Because the pitch rotation is just followed by the roll rotation, these components are obtained by using the rotation matrix in Equation (3-2). Therefore, going from the already made pitch rotation to the last elemental rotation would involve the roll rotation matrix:

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}_{\theta} = {R^{-1}}_{\varphi}\begin{bmatrix}0\\ \dot{\theta}\\ 0\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & cos\,\varphi & sin\,\varphi\\ 0 & -sin\,\varphi & cos\,\varphi\end{bmatrix}\begin{bmatrix}0\\ \dot{\theta}\\ 0\end{bmatrix} = \begin{bmatrix}0\\ \dot{\theta}\,cos\,\varphi\\ -\dot{\theta}\,sin\,\varphi\end{bmatrix} \tag{3-28}$$

- Components of $\dot{\psi}$ about the body axes: Because the yaw rotation is followed by the pitch and then roll rotation, its velocity components are obtained by using the matrices of the roll and pitch rotations in Equations (3-2) and (3-3). Their inverses are post-multiplied. Therefore, going from the already made yaw rotation to the last elemental rotation would involve the roll and pitch rotation matrices as follows:

$$R^{-1}{}_{\varphi} R^{-1}{}_{\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\varphi & sin\,\varphi \\ 0 & -sin\,\varphi & cos\,\varphi \end{bmatrix} \begin{bmatrix} cos\,\theta & 0 & -sin\,\theta \\ 0 & 1 & 0 \\ sin\,\theta & 0 & cos\,\theta \end{bmatrix}$$

$$= \begin{bmatrix} cos\,\theta & 0 & -sin\,\theta \\ sin\,\varphi\,sin\,\theta & cos\,\varphi & sin\,\varphi\,cos\,\theta \\ cos\,\varphi\,sin\,\theta & -sin\,\varphi & cos\,\varphi\,cos\,\theta \end{bmatrix}$$

Thus,

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_{\psi} = R^{-1}{}_{\varphi} R^{-1}{}_{\theta} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\dot{\psi}\,sin\,\theta \\ \dot{\psi}\,sin\,\varphi\,cos\,\theta \\ \dot{\psi}\,cos\,\varphi\,cos\,\theta \end{bmatrix} \tag{3-29}$$

- To get the final relation that gives the body rates with respect to Euler angular speeds, Equations (3-27), (3-28) and (3-29) are added.

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_{\varphi} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_{\theta} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_{\psi} = \begin{bmatrix} \dot{\varphi}\,cos\,\theta - \dot{\psi}\,sin\,\theta \\ \dot{\theta}\,cos\,\varphi + \dot{\psi}\,sin\,\varphi\,cos\,\theta \\ -\dot{\theta}\,sin\,\varphi + \dot{\psi}\,cos\,\varphi\,cos\,\theta \end{bmatrix}$$

Thus,

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sin\,\theta \\ 0 & cos\,\varphi & sin\,\varphi\,cos\,\theta \\ 0 & -sin\,\varphi & cos\,\varphi\,cos\,\theta \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3-30}$$

In the quadcopter application, what is more important is Euler rates with respect to the body rates that are got from the gyro sensor. This is simply found by taking the inverse of the 3x3 matrix in Equation (3-30). It is good to mention that this matrix is not orthogonal. It relates an orthogonal vector to Euler angular speeds which are non-

orthogonal. This means that the inverse of this matrix is not generally equal to its transpose.

$$\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & 0 & -sin\,\theta\\ 0 & cos\,\varphi & sin\,\varphi\,cos\,\theta\\ 0 & -sin\,\varphi & cos\,\varphi\,cos\,\theta\end{bmatrix}^{-1}\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$
$$= \begin{bmatrix}1 & sin\,\varphi\,tan\,\theta & cos\,\varphi\,tan\,\theta\\ 0 & cos\,\varphi & -sin\,\varphi\\ 0 & \frac{sin\,\varphi}{cos\,\theta} & \frac{cos\,\varphi}{cos\,\theta}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \quad (3\text{-}31)$$

- Angular Acceleration of the Quadcopter

In this subsection, the relation between the angular body rates and the torques will be derived. For this derivation, moment of inertia of the quadcopter will be used.

The torque is analogous to the force. The net force on the quadcopter is proportional to the linear accelerations $\ddot{x}, \ddot{y}$ and $\ddot{z}$ and torque is proportional to the angular acceleration $\dot{p}, \dot{q}$ and $\dot{r}$. The proportionality parameter for the force is the mass *m* (see Equation (3-21)) and for the torque is the moment of inertia about the axis of rotation.

The inertia matrix about the axes of the quadcopter is defined in the three axes of rotation as follows:

$$I = \begin{bmatrix}I_{xx} & I_{xy} & I_{xz}\\ I_{yx} & I_{yy} & I_{yz}\\ I_{zx} & I_{zy} & I_{zz}\end{bmatrix}$$

where I_{xx}, I_{yy}, and I_{zz} are the moments of inertia about the x_b, y_b, and z_b axes, respectively, and I_{xy}, I_{xz}, I_{yz}, I_{yx}, I_{zx}, and I_{zy} are the products of inertia.

Since the structure of the quadcopter is assumed to be perfectly symmetrical, the products of inertia become zero, and the mass moments of inertia are defined by

Equation (3-32).

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (3\text{-}32)$$

In general, torque is given by:

$$\tau = \alpha I$$

where α is the angular acceleration. From this equation, the mass moment of inertia for any random body can be calculated as follows:

1) Applying a certain amount of torque on the body about an axis.
2) Measure the angular acceleration which that torque produced using a high precision angular acceleration sensor.
3) Calculate the mass moment of inertia by dividing the applied torque by the measured angular acceleration.

This process is repeated about the three principal axes to find the inertia tensor. In reality, there are difficulties in dealing with the noise and signal lag in acceleration sensors as discussed in (Ovaska & Valiviita, 1998). Fortunately, the quadcopter in hand has simple geometry that makes the approximation of the mass moment of inertial more feasible.

Euler's equation of motion in Equation (3-33) describes the quadcopter's motion about its center of mass.

$$\tau = I\dot{\omega} + \omega \times (I\omega) \quad (3\text{-}33)$$

Solving for the body angular rates, the rotational equations of motion would be described as in Equation (3-34).

$$\vec{\omega} = \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{\tau_\varphi}{I_{xx}} \\ \frac{\tau_\theta}{I_{yy}} \\ \frac{\tau_\psi}{I_{zz}} \end{bmatrix} + \begin{bmatrix} \frac{I_{yy} - I_{zz}}{I_{xx}} qr \\ \frac{I_{zz} - I_{xx}}{I_{yy}} pr \\ \frac{I_{xx} - I_{yy}}{I_{zz}} pq \end{bmatrix} \quad (3\text{-}34)$$

3.3 Quadcopter Control

To fully control the six degrees of freedom of the quadcopter, the designed controller must control:

1. The altitude (*z* position).
2. The horizontal positions (*x* and *y* positions).
3. The yaw angle or heading.
4. The roll and pitch angles.

Controlling the horizontal positions implicitly means controlling the pitch and roll angles because translational maneuvering cannot be achieved without either pitching or rolling.

The ultimate outputs of the proposed controllers are the required thrust *u* and torques τ_φ, τ_θ and τ_ψ that must be produced by the four motors as given by Equation (3-17).

As already mentioned, a fuzzy-tuned PID controllers will be designed to control the model. With that being the expected best controller, other controllers will also be used for the sake of testing and comparing the performance of each control technique. The chosen controllers include PID and fuzzy-tuned PID controllers. The output of these controllers will be the required thrusts and torques that the four motors of the quadcopter have to produce to reach the desired positions and rotations. The inputs to

these controllers will be the state variables that are to be controlled.

3.3.1 PID Controllers for Quadcopters

Figure 3.9 and Figure 3.10 show the block diagram and Simulink model of the proposed PID controllers for the quadcopter system. These controllers determine the required total thrust and torques that the four motors must generate to reach the desired state based on the following received inputs:

1. The measured linear and angular positions.
2. The desired linear positions.
3. The desired heading angle.

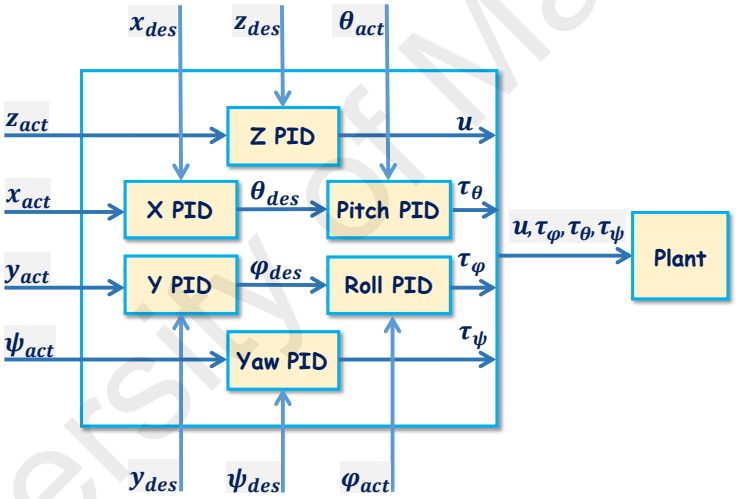

Figure 3.9 Structure of the designed PID controllers

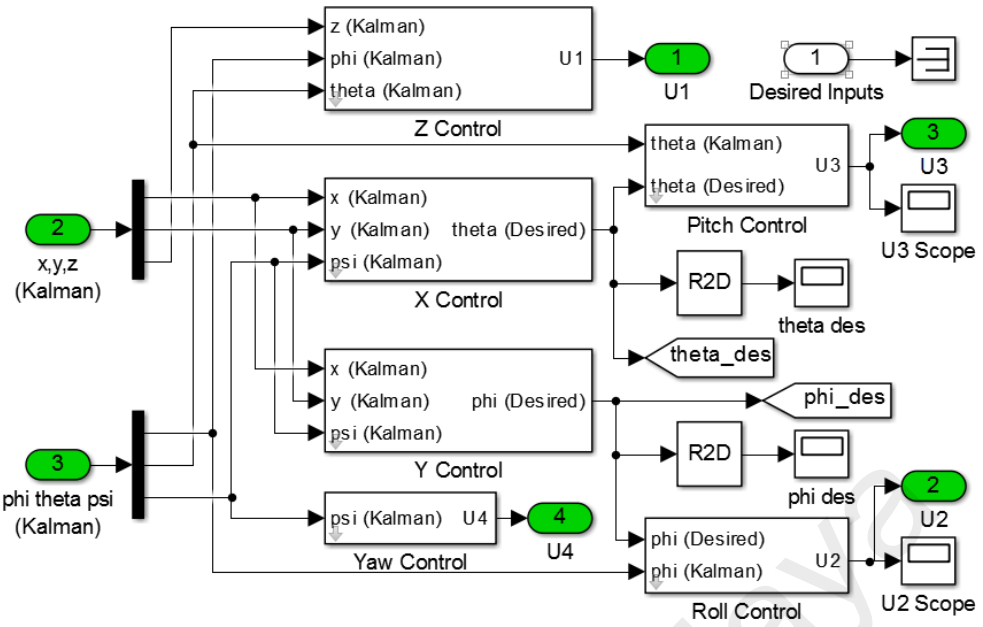

Figure 3.10 Simulink model for PID controllers

The positional PID output is defined by the following general equation, where $\tilde{x}_c = x_{cd} - x_c$ is the error of the control input signal x_c, T is the sampling time and K_p, K_i, K_d are the three PID gains:

$$u = K_p\tilde{x}_c(t) + K_iT\sum_{i=0}^{t-1}\tilde{x}_c(i) - K_d\dot{x}_c(t) \tag{3-35}$$

The controller has to maintain a constant altitude z_{des} during flight. We propose an altitude PID control with gravity compensation as:

$$u = \frac{1}{cos\,\varphi\,cos\,\theta}\left[K_{p_z}\tilde{z} + K_{i_z}T\sum_{i=1}^{t}\tilde{z}_i - K_{d_z}\dot{z} + mg\right] \tag{3-36}$$

where the denominator is used to produce a control output that determines the required thrust in the body coordinate frame.

The external pilot can hardly tune the pitch and roll angles during flight. Therefore, the X and Y PID controllers are designed to determine their desired values based on the commanded x and y positions. Before feeding the positional errors to these controllers in the inertial reference frame, any yaw rotation must be accounted for. It is easy to process the errors to get the actual control inputs as follows (see figure 3): $\tilde{x}_b =$

$\tilde{x}\cos\psi + \tilde{y}\sin\psi, \tilde{y}_b = -\tilde{x}\sin\psi + \tilde{y}\cos\psi.$

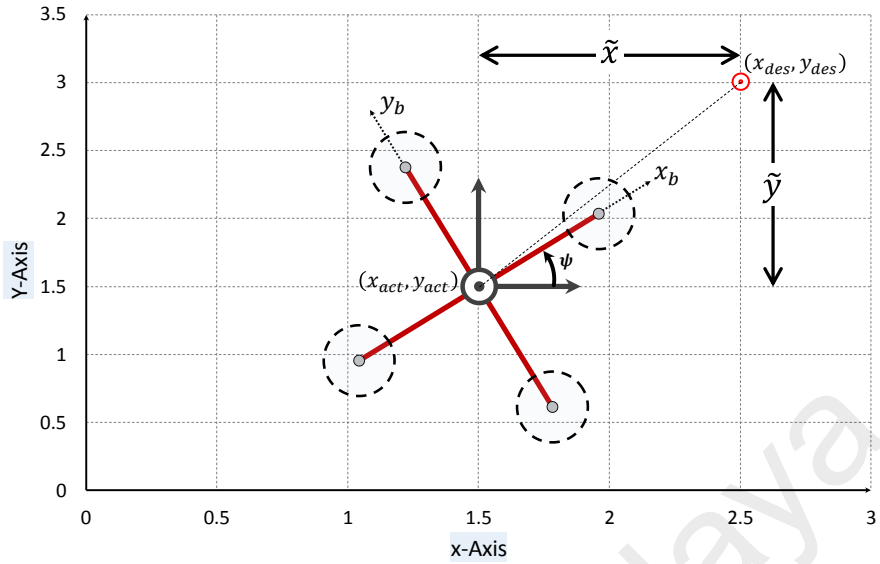

Fig. 3.11 Quadcopter moving to the desired destination with non-zero heading

3.3.2 Fuzzy-Tuned PID Controllers

The derived equations of motion in equations (3-22) and (3-34) clearly show that the quadcopter model is nonlinear. The classical PID controllers have certain limitations that cause response time delay and high overshoots and, therefore, they are commonly insufficient for nonlinear processes to achieve robust control and maintain high stability. While, in fact, no real system is perfectly linear, PID controllers are best for processes that are close to linearity throughout their dynamic range and operating points. Fuzzy logic controllers provide an alternative approach to solving control problems for nonlinear processes and, hence, they are used to dynamically tune the PID controllers to achieve a better control of such systems.

The designed fuzzy logic controller is a dual-input controller that receives the error and error rate of the linear and angular positions and outputs control signals that manipulate the PID gains. This fuzzy system maps the input space (the error and its rate) to the output space (the required PID gains) in a dynamic manner. As discussed earlier, six PID controllers were designed. Therefore, there are 18 parameters to be controlled: six

P parameters, six *I* parameters and six *D* parameters. It is truly cumbersome to design a new fuzzy logic controller to tune each parameter.

One way to simplify the control design is to calibrate or normalize each fuzzy output in the range [0,1] as proposed in (Sinthipsomboon, Hunsacharoonroj, Khedari, Pongaen, & Pratumsuwan, 2011) and (Zulfatman & Rahmat, 2009). The normalized crisp output is then weighted and scaled appropriately to produce the tuned PID gains (see the diagram in Figure 3.12). This process allows one fuzzy system to tune different PID controllers. This technique works well on the proposed quadcopter system because the logic of tuning the PID parameters is similar regardless of what axis the quadcopter is currently translating along or rotating about.

The PID gain $K \in [K_{min}, K_{max}]$ can be calibrated using equation (3-37) to get a control output $K' \in [0,1]$. $K^{'}$ is eventually weighted and scaled using equation (3-38) to finally get the desired PID gain for the current operating point. For example, let's assume that a certain PID gain *K* has values in the range [3, 20] throughout the system dynamic range, then the gain value for a current operating point can be defined by $K = 17K' + 3$. If the fuzzy controller outputs a value of $0.4 \in [0,1]$, the current PID gain that is fed into equation (3-35) will be $9.8 \in [3,20]$. Table 3.1 summarizes the formulas used to calculate the three parameters for the altitude PID controller described earlier.

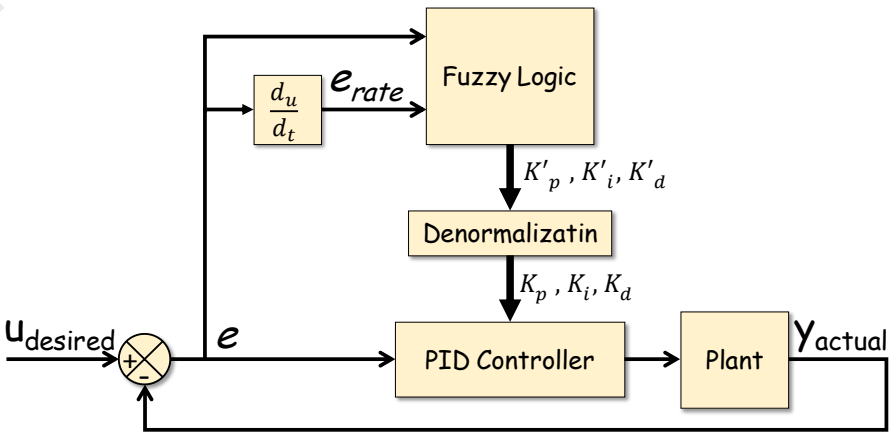

Figure 3.12 Structure of the fuzzy-tuned PID controller

$$K' = \frac{K - K_{min}}{K_{max} - K_{min}} \tag{3-37}$$

then,

$$K = (K_{max} - K_{min})K' + K_{min} \tag{3-38}$$

Table 3.1 Equations used to denormalize the output of the fuzzy controller to get K_p, K_i and K_d

Parameter	Min	Max	Equation
K_p	1	50	$K_p = 49\,K'_p + 1.0$
K_i	0.1	1	$K_i = 0.9\,K'_i + 0.1$
K_d	0.1	40	$K_d = 39.9\,K'_d + 0.1$

Table 3.2, Table 3.3 and Table 3.4 show the fuzzy sets used for the error input, error rate input and the parameters output. Figure 3.13 and Figure 3.14 show the graphical representation of the membership functions for the fuzzy input variables.

Table 3.2: Fuzzy sets and their range for the error input variable

Fuzzy set (Label)	Description	Range	Membership Function
Negative large (NL)	Error is negatively large	[-100 -100 -4 -1]	Trapezoidal
Negative small (NS)	Error is negatively small	[-2 -1 0]	Triangle
Ok	Error is OK	[-0.1 0 0.1]	Triangle
Positive Small (PS)	Error is positively small	[0 1 2]	Triangle
Positive large (PL)	Error is positively large	[1 4 100 100]	Trapezoidal

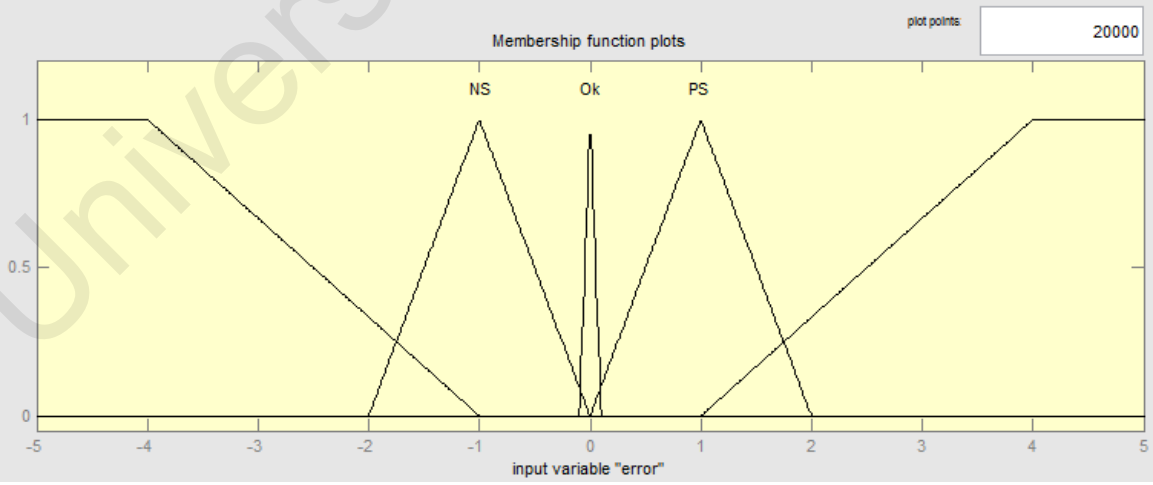

Figure 3.13: Graphical representation for the error membership functions

Table 3.3: Fuzzy sets and their range for the error rate input variable

Fuzzy set	Description	Range	Membership Function
NL	Error rate is negatively large	[-100 -100 -4 -1]	Trapezoidal
NS	Error rate is negatively small	[-4 -1 -0.25]	Triangle
Ok	Error rate is OK	[-0.5 0 0.5]	Triangle
PS	Error rate is positively small	[0.25 1 4]	Triangle
(PL	Error rate is positively large	[1 4 100 100]	Trapezoidal

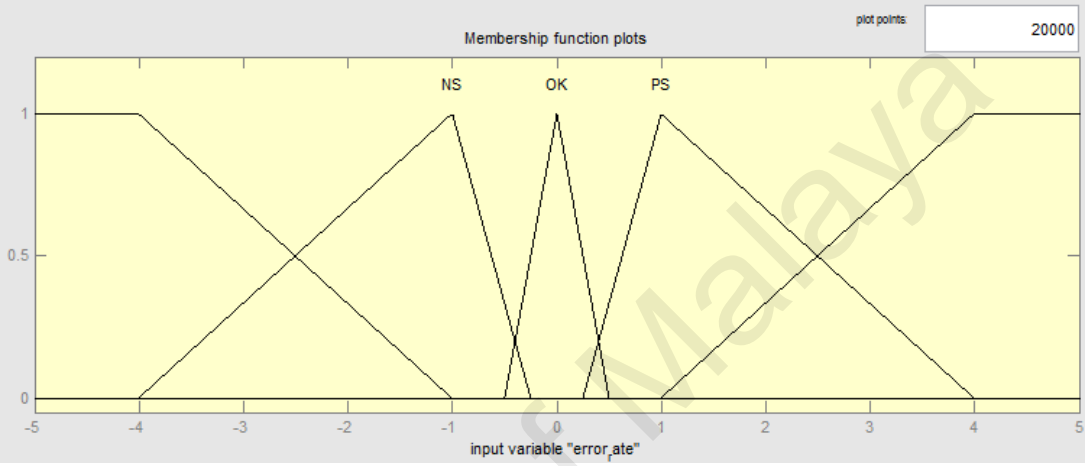

Figure 3.14: Graphical representation for the error rate membership functions

Table 3.4: Fuzzy sets for the output variable assigned to altitude control

Fuzzy set (Label)	Description	Value	Membership Function
Very low (VL)	PID gain is very low	0	Trapezoidal
Low (L)	PID gain is low	0.25	Triangle
M	PID gain is medium	0.5	Triangle
High (H)	PID gain is high	0.75	Triangle
Very high (VH)	PID gain is very high	1	Trapezoidal

Table 3.5 shows the fuzzy rules that govern the fuzzy controller and specifies its behavior based on the two input levels.

Table 3.5 Fuzzy rule table for K_p and K_d

e \ Δe	NL	NS	OK	PS	PL
NL	VH	VH	VH	VH	VH
NS	H	VH	H	VH	H
OK	VL	L	L	M	M
PS	H	H	H	H	VH
PL	VH	VH	VH	VH	VH

3.4 Performance Analysis

A step signal is used to analyze the performance of a transient system. At the instance the step signal is applied to the system, the frequency is ideally infinity because the rise time is ideally zero and when it reaches its maximum magnitude, it maintains its value all the way next with zero frequency. From this comes the importance of the step response testing where the response is tested with zero and infinite frequency. Figure 3.15 shows a typical step response.

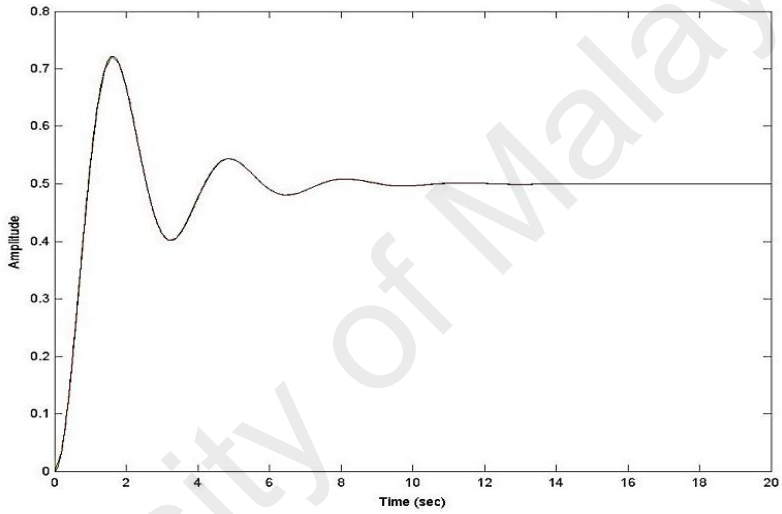

Figure 3.15: Typical step response

There are some key indicators used to describe the behavior and performance of a control system when using step input response. These indicators are:

a) Percentage overshoot, PO: Percentage of the overshoot value.
b) Peak time t_{max}: the time elapsed to reach the peak overshoot.
c) Delay time t_d: time elapsed to reach 50% of the steady state value.
d) Rise time t_r: time duration taken by the response curve from 10% to 90% of its steady state value.

e) Settling time t_s: time taken until the response curve enters an error band and remains in it. The error band might be 5% of the steady state value (the error percentage is defined by the analyst according to the requirements).

Another performance test was applied on the simulation model which is the path tracking. The quadcopter was commanded to follow a predefined path like a circle in midair. During this test, the behavior of the quadcopter under both PID and fuzzy-tuned PID controllers was closely monitored and compared.

CHAPTER 4. SIMULATION PLATFORM

4.1 Introduction

In this chapter, deep discussion will be made on how the mathematical model and control have been simulated. Discussion will also be made on how the system performance is monitored using proper simulation tools. Simulation also includes the virtual reality capability which efficiently helped visualize the system.

4.2 Simulation Software

Once the mathematical model is developed, simulation comes into place. MATLAB and Simulink will be used to create this simulation. MATLAB (an abbreviation for matrix laboratory) is a fourth-generation programming language and a numerical computing environment that is primarily designed to operate on matrices and arrays. MATLAB and Simulink, which is a part of MATLAB, are powerful software products that have many development tools and blocks which can be used to design and simulate the model. With simulation, the performance of the control system can be carefully analyzed and some parameters can be tuned with relative ease to monitor the effect of any changes made.

Figure 4.1 shows the general flow diagram for the simulation. It is a closed loop operation that keeps running for a specified duration set prior to running the simulation. At the beginning, all of the state variables have zero initial conditions. Setting the initial conditions to zero is logical because the quadcopter is assumed to always start from the base station with all states being zero.

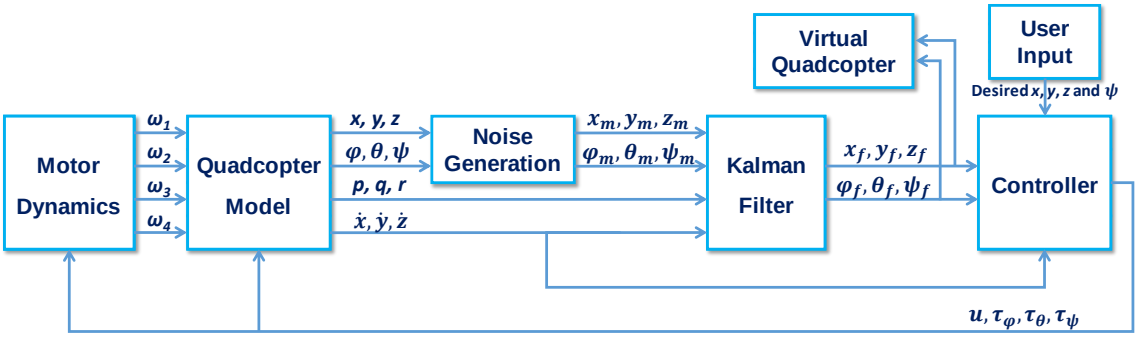

Figure 4.1: General flow diagram for the Simulation

Each block in this general flow diagram represents a subsystem in Simulink. Subsystems allow to create a hierarchical model comprising many layers. Every subsystem contains a set of Simulink blocks that may represent basic and complex mathematical operations, charts and plots, physical operations and conversions or graphical animations. In Simulink, the mathematical operators are represented graphically. The following are advantages of using subsystems:

a) They establish a hierarchical block diagram, where a subsystem block is on one layer and the blocks that make up the subsystem are on another.

b) They keep functionally-related blocks together.

c) They also help reduce the number of blocks displayed in the model window.

From the mathematical model of the quadcopter, there are so many variables and interrelated equations. The equations and variables must be well handled in simulation. The simulation flow must be neat and well organized to allow easy and effective monitoring and debugging.

4.3 Simulation Model on Simulink

Figure 4.2 shows the first layer or the general view of the whole Simulink model. One can match between simulation flowchart in Figure 4.1 and Simulink model. As seen in the figure, a Kalman filter is simulated to tackle the noisy measurements reported by the

sensors. Any sensor reading is actually a combination of the signal and measurement noise. This filter uses the equations of motion of the quadcopter to continuously correct the readings and enhance the signal to noise ratio while the simulation is running.

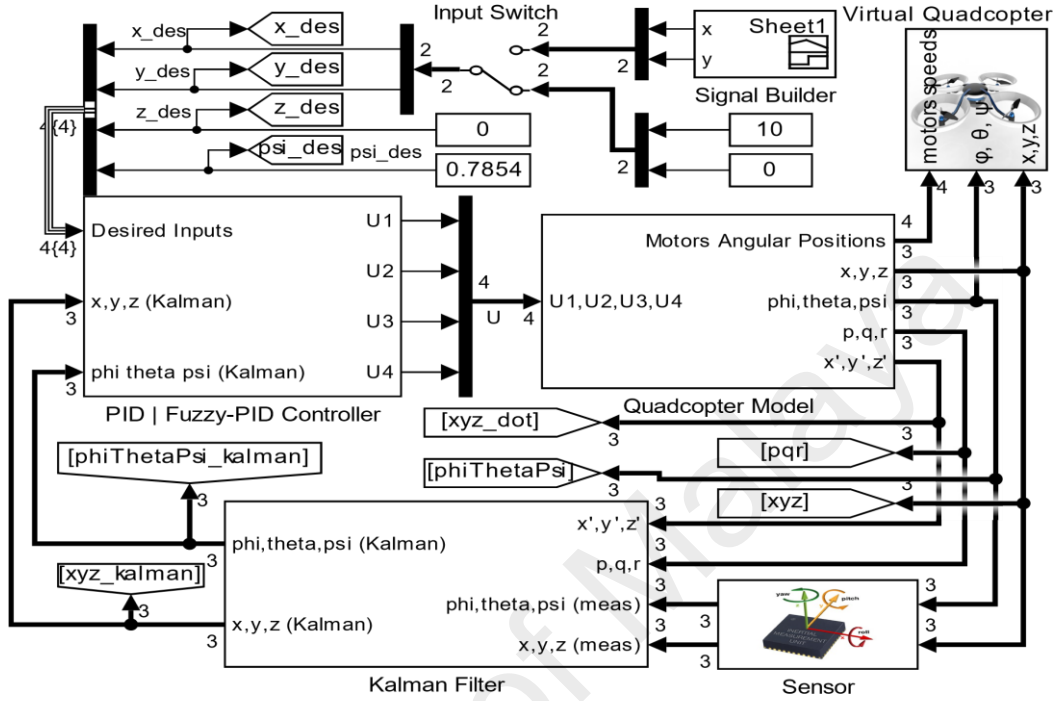

Figure 4.2: General view of the developed Simulink model

In the subsystem named "Quad Model", all of the dynamic differential equations were implemented. This subsystem contains equations derived in Subsection 3.2.5. The input to this system is the thrusts and motor speeds and it outputs the linear state variables. The figures below show the Simulink subsystem and blocks used to calculate Euler angles and their derivatives. These subsystems represent equation (3-31).

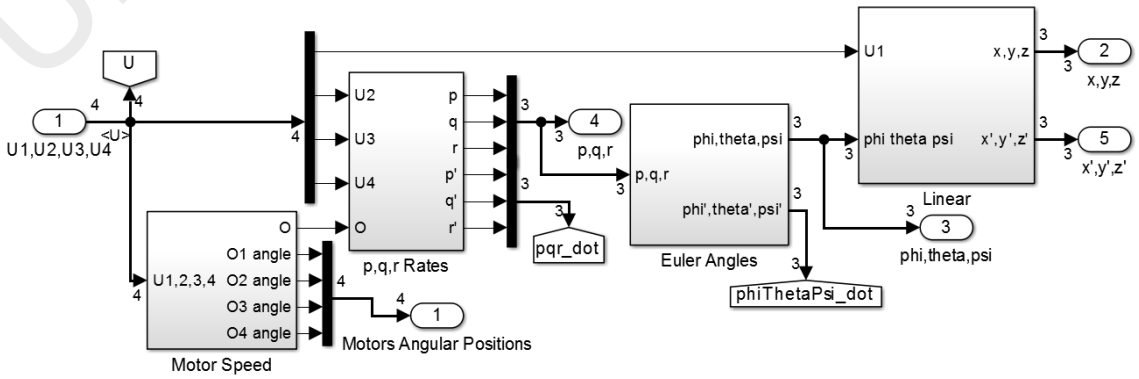

Figure 4.3: Quadcopter model subsystem.

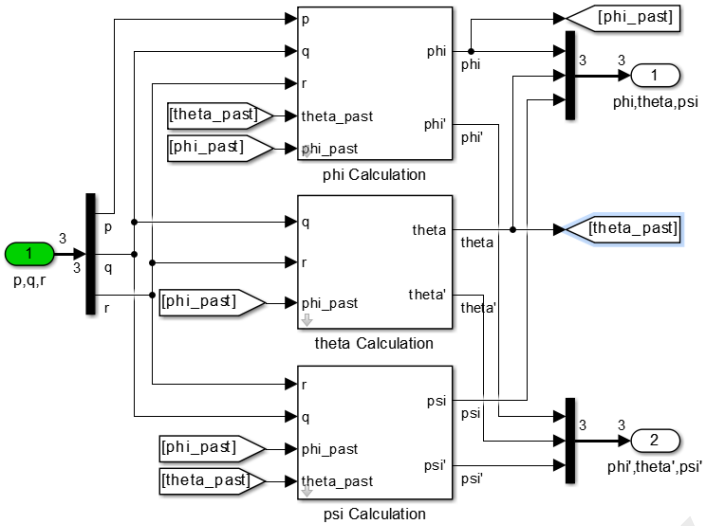

Figure 4.4: Euler angles subsystem.

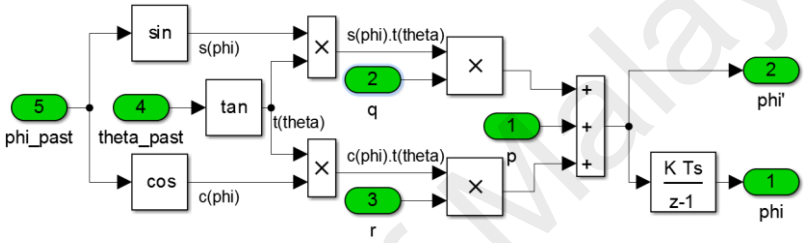

Figure 4.5: Subsystem to find the roll angle.

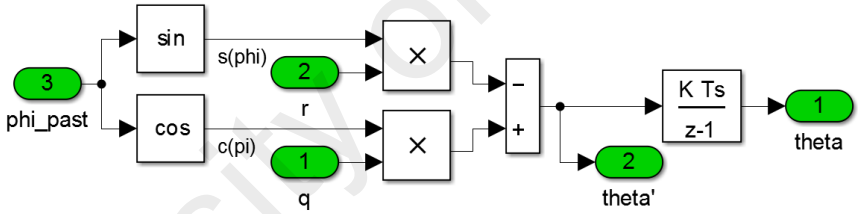

Figure 4.6: Subsystem to find the pitch angle

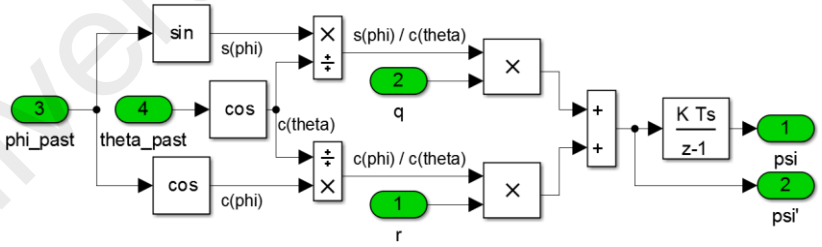

Figure 4.7: Subsystem to find the yaw angle.

The four motors speeds are also calculated as part of the model subsystem based on the torques and thrusts required to reach the desired state. Figure 4.8 shows how this subsystem block is implemented. This actually represents Equation (3-17).

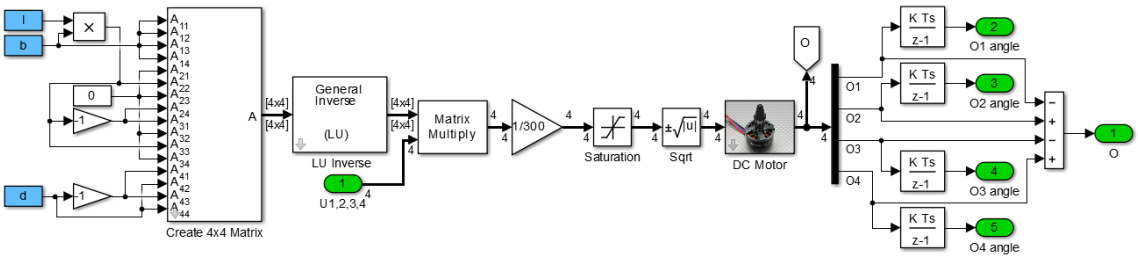

Figure 4.8: Subsystem to calculate motor speeds based on actuating thrusts and torques

The "linear" subsystem block in Figure 4.3 contains all of the blocks used in calculating the translational positions *x*, *y* and *z*. This subsystem represents the linear dynamic equations (3-23) to (3-25).

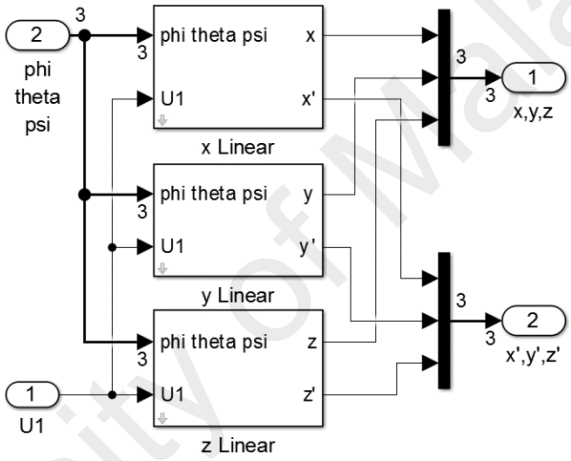

Figure 4.9: *x*, *y* and *z* linear positions subsystem

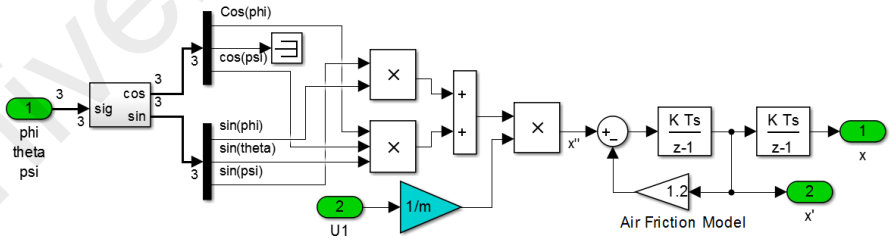

Figure 4.10: Subsystem to calculate *x* position and speed

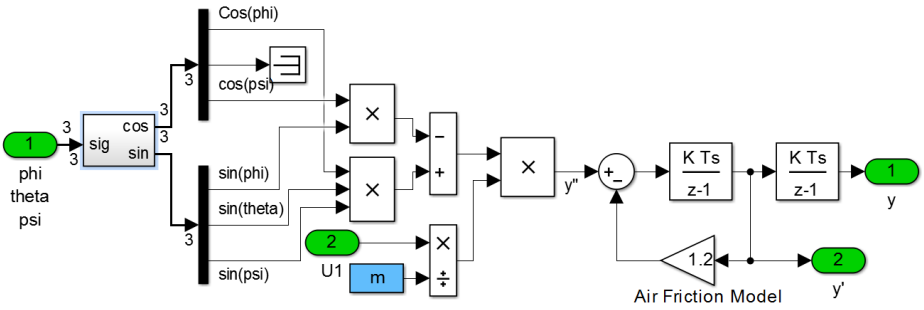

Figure 4.11: Subsystem to calculate y position and speed

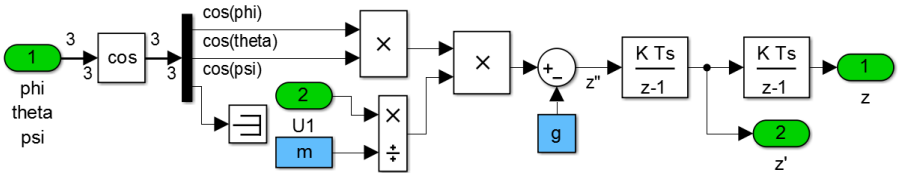

Figure 4.12: Subsystem to calculate *z* position and speed

4.4 System Graphical User Interface (GUI)

After developing the mathematical model that represents the key characteristics and behavior of the quadcopter system and designing its controller, the simulation platform was created to provide a comprehensive test environment for the quadcopter as a computational model. The main panel for the quadcopter system, shown in Figure 4.13, was built using GUIDE, a MATLAB tool helping developers to create GUIs and define view functionality. The full MATLAB source code that dictates the functionality of every view in the designed GUI is provided in APPENDIX A.

.

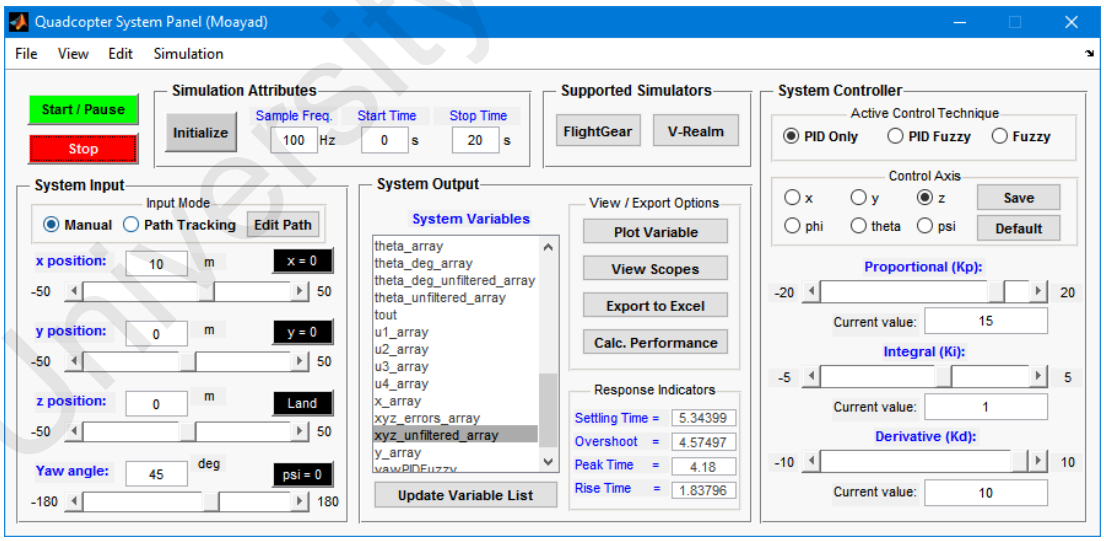

Figure 4.13 Central control panel for the quadcopter simulation platform

This main interface consists of visual elements that are grouped in a few sub-panels based on functionality. These are described as follows:

- The input sub-panel that enables the user to test the performance of the quadcopter

using an individual or a group of desired inputs. The input can be manually tuned, while the simulation is running, by specifying the desired values of the *x, y* and *z* positions and the yaw angle from their corresponding text boxes or slide bars. The input source can be a waveform like unit step, ramp, sinusoidal or saw tooth that the user can customize using the Simulink signal builder. This paper, for example, reports the test results for the quadcopter following a predefined circular path.

- The system control sub-panel where the user can select the active control technique (whether classical PID or fuzzy-tuned PID) and specify the PID control gains for each axis as discussed in section 3.3. The user also has the option to load the default PID gains or save the current gain values in the MATLAB workplace for later usage.
- The system output sub-panel which offers handy tools to monitor the quadcopter performance. The list box in this sub-panel lists all the output variables reported by the quadcopter Simulink model. It supports some control buttons to plot the responses and calculate response indicators, e.g. settling time, rise time, peak time and percentage overshoot. It is sometimes best to view the values of the selected variable in an excel file for better data analysis and recording; The "Export to Excel" button does this by creating an excel file with a custom name in the model directory.
- The user can also change the simulation start and stop time and sampling frequency, and can start, pause or stop the simulation after initializing it.

Almost all of these features are available in the designed menu appearing at the top left of the main interface. Figure 4.14 shows the menu lists which add more functionality such as opening and closing the model, saving changes, opening the virtual simulators, editing the fuzzy inference system and changing the quadcopter radial length, mass, and inertia. This interface is fully integrated with the quadcopter

Simulink models that are discussed in the previous subsection.

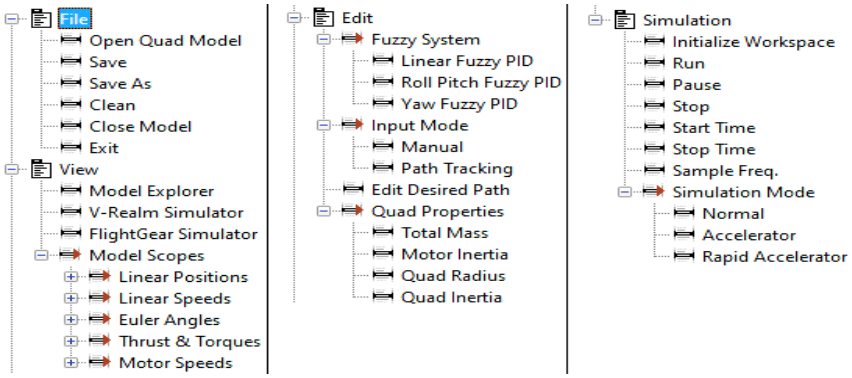

Figure 4.14 Menu contents for the main GUI window

To visualize the quadcopter while the simulation is running, the system was integrated with FlightGear, a flight simulator that is supported by Simulink. Snapshots of the quadcopter visualization in FlightGear is shown in Figure 4.15. The Aerospace Animation library in Simulink provides building blocks that assist in this integration process and allow sending and receiving commands. The quadcopter physical model was designed in a CAD software (e.g. SolidWorks and Blender) then integrated with FlightGear. The developed Simulink model sends the process outputs such as quadcopter positions, attitudes and motor speeds to the FlightGear simulator at 100Hz through TCP/IP network communication in order to visualize the quadcopter 3-dimensional motion in real time.

Figure 4.15 Visualization of the quadcopter with different orientation using FlightGear

4.5 Towards an Application in Engineering Education

The developed system presents an integrated functional tool of a real engineering

control problem that has attracted the attention of researchers due to their various applications. The simulation platform was carefully developed to well replicate the actual behavior of a quadcopter vehicle including the fundamental physical and dynamical laws that govern it. The platform includes all essential elements that allow for testing the quadcopter under different test scenarios. The GUI presents a central control panel that accepts user's desired inputs, provides tools to change the supported control parameters and outputs the response in a form of plots, data logs, response indicators and real-time 3D visualization. Advanced researchers who are interested in quadcopter UAVs might opt to go from the top end-user interface to the model building blocks that represent the governing mathematical equations. The model was segmented into hierarchical subsystems such that the complex model and control can be easily understood and independently tested by those researchers.

This platform has great potential to be a useful tool for engineering students and researchers to understand quadcopter systems and their working principles. It permits the engagement of learners to evaluate the performance of these systems in a realistic, interactive and meaningful way. For example, students can tune the mass of the quadcopter and their radial length to see how much more (or probably less) power is needed by the four motors to reach a desired position in 3D space. Another example is to instruct the students to use the platform to find the PID gains or to properly modify the corresponding fuzzy rules that would raise the quadcopter to the desired height in the shortest time and least overshoot. PID and fuzzy logic are one of the basic control techniques that are taught to engineering students in control engineering courses. The proposed platform is a good environment to better analyze such widely used controllers especially that the quadcopter can be tested on each axis individually while disabling the others.

In general, it is highly expensive to equip university laboratories with real flying

robots and their spare parts. However, the proposed platform was designed to possibly minimize this need. It can play a vital role as a risk-free learning tool installed on a lab PC. Furthermore, many universities around the world are now offering university-level online courses in a wide range of disciplines including aerial robotics and other engineering fields. The proposed platform can be a part of such online teaching process through engaging and evaluating the students using a series of performance tests during course development.

The following section provides some test scenarios that students can apply to the quadcopter using the developed simulation tool. The two implemented control techniques (traditional PID and fuzzy-tuned PID controller) are tested and compared.

CHAPTER 5. RESULTS AND DISCUSSION

5.1 Introduction

After simulating the quadcopter system on MATLAB and Simulink, testing and performance comparison are handled. In this chapter, a detailed discussion about the quadcopter performance will be made. This includes model validation, control analysis and visualization.

All of the constants described in the dynamic equations have been assigned suitable values and the quadcopter performance testing is based on these selected constants. Table 5.1 summarizes these model parameters and their values.

Table 5.1: Summary of the constant values used in the quadcopter dynamic equations.

Step Response Indicators	Value
Mass (m)	1 Kg
Inertia along x and y axes (I_{xx}, I_{yy})	0.0081 N.s^2/rad
Inertia along the z axis (I_{zz})	0.0142 N.s^2/rad
Quadcopter Radius (l)	0.24 m
Friction constant (K_d)	1.1*10^{-6} Nm/Kg
Thrust factor K	5.42*10^{-5} Ns^2/rad^2

The controller performance on the quadcopter was examined in terms of input tracking capabilities along the x_n, y_n and z_n axes. Altitude, horizontal translation and path tracking responses under the PID and fuzzy-tuned PID controllers are discussed in the following subsections. The sampling frequency was set to 100Hz which is reasonable for performance analysis on Simulink. The mass of the simulated quadcopter was set to 1.0kg for the next performance tests. All of the following performance tests were conducted with the presence of simulated noise imposed on the positional and angular process variables (see Figure 5.1).

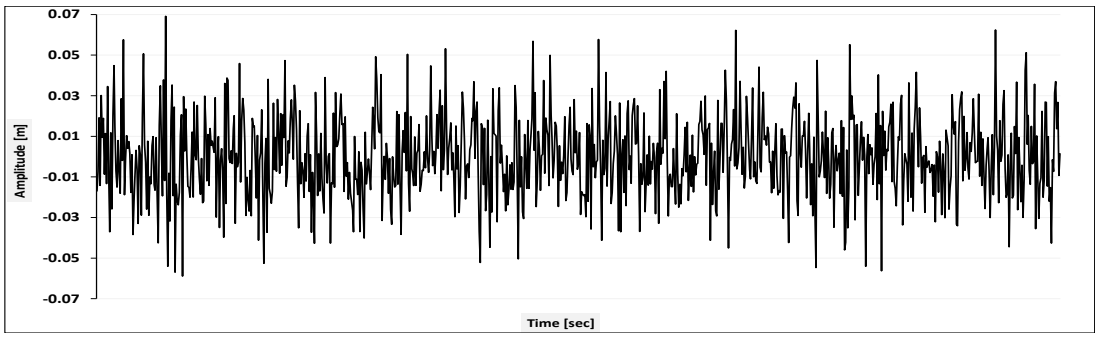

Figure 5.1 Simulated noise applied to the quadcopter positions and angles

5.2 Altitude Control

The quadcopter was commanded to translate 10 meters along the vertical axis in the inertial reference frame. Figure 5.2 shows the quadcopter altitude response and its error for the PID and fuzzy-tuned PID controllers.

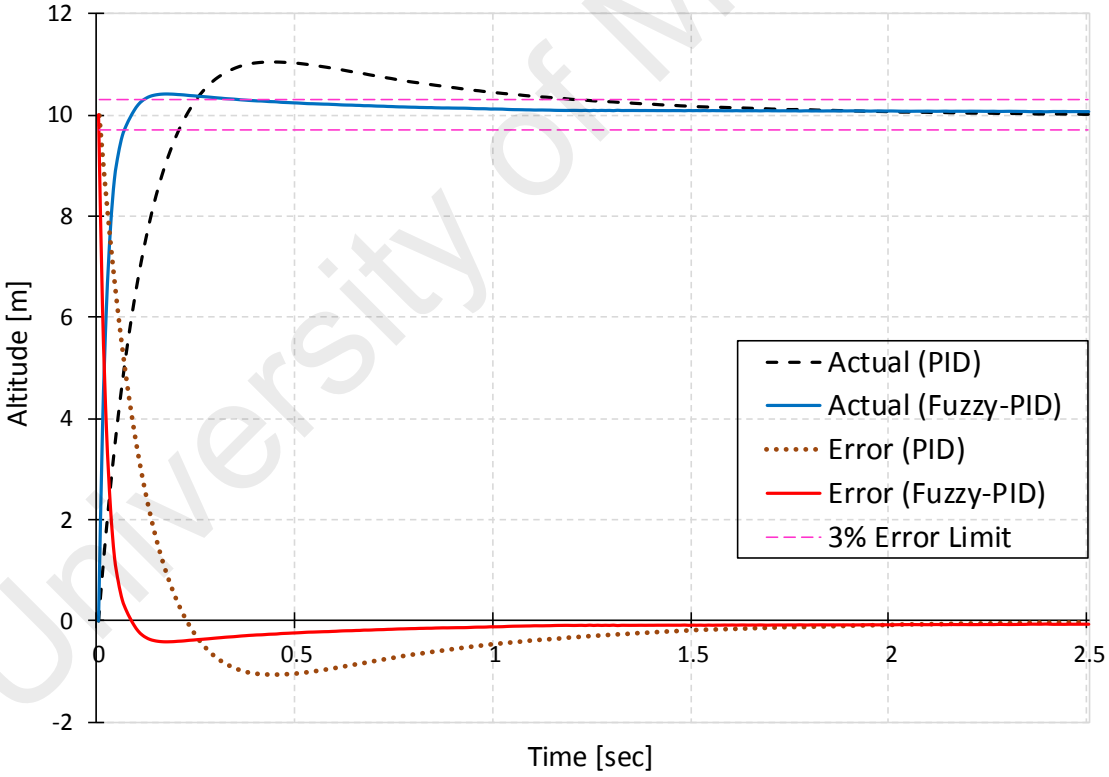

Figure 5.2: Quadcopter altitude and its error with PID and fuzzy-tuned PID controllers

In general, the two controllers could control and stabilize the quadcopter at the desired altitude (z = 10m), but with performance differences. The figure shows that the altitude error is maximum at time t = 0 then it quickly decreased with time and became

negative after the maximum overshoot had been reached. It is clear that the fuzzy-tuned PID controllers enabled the quadcopter to fly up to the desired z position faster than the PID controllers, and with 6.35% less overshoot. The fuzzy-tuned PID controller could reduce the rise time from 150ms to 45ms. This may be attributed to the mechanism of the fuzzy controller that offers fast clustering and selection at the operating point (B. A. Elsayed, Hassan, & Mekhilef, 2013; B. Elsayed, Hassan, & Mekhilef, 2015). The altitude response indicators (peak time, delay time, rise time and settling time) for both controllers are compared and summarized in Table 5.2.

Table 5.2: Performance indicators for altitude response

Step Response Indicators	Tuned PID	PID Only
Percentage Overshoot, *PO*	4.25%	10.6%
Peak Time, t_{max}	170ms	440ms
Delay Time, t_d	20ms	70ms
Rise Time, t_r	45ms	150ms
Settling time, t_s	380ms	1.25s

For example, the response indicators with the PID controller are calculated as follows (see section 3.4):

- Percentage overshoot, PO:

The maximum overshoot of the altitude, z_{peak} is 11.06m. The percentage overshoot is:

$$\mathrm{PO} = \frac{11.06 - 10}{10} \times 100\% = 10.6\%$$

- Peak time t_{max}:

The time taken to reach the maximum overshoot is: t_{max} = 440ms.

- Delay time t_d:

The time taken to reach 50% of the steady state value is: t_d = 70ms.

- Rise time t_r:

The time taken by the quadcopter to go up from 10% to 90% of the steady state altitude is: $t_r = t_{90\%} - t_{10\%} = 170\text{ms} - 20\text{ms} = 150\text{ms}$.

- Settling time t_s:

The time taken by the quadcopter to enter the 3% error band and remain in it is 1.25s.

As discussed before, the goal of the controller is to minimize the error until it almost becomes zero. Figure 5.2 also shows the error tracking of the quadcopter for altitude control. At time equal to zero, the error is at its maximum (the error is equal to the amplitude of the step signal, that is 10m). The error then reduces and starts to be negative at time 220ms when the quadcopter overshoots the desired altitude.

The enhancement introduced by the fuzzy-tuned PID controller over the conventional PID controller in terms of overshoot, peak time, delay time, rise time and settling time proves that the membership functions and the rules for the altitude control were wisely selected.

5.3 Horizontal Translation Control

5.3.1 Horizontal position

Figure 5.3 shows the translation response of the quadcopter for a 10m step input along the horizontal axes x_n and y_n. The responses of the quadcopter along these two axes are almost same because the structure of their controllers is identical. The overshoot with fuzzy-tuned PID controller is 0.89% lesser than the conventional PID controller. Referring to the performance indicators (peak time, delay time, rise time and settling time), the improvement in the horizontal translation control is more significant with the fuzzy-tuned PID controller. For example, the settling time with the fuzzy-tuned PID controller could be reduced to nearly 53% of that with the PID controller. Table 5.3

summarizes the translations response indicators for both controllers along x_n and y_n axes.

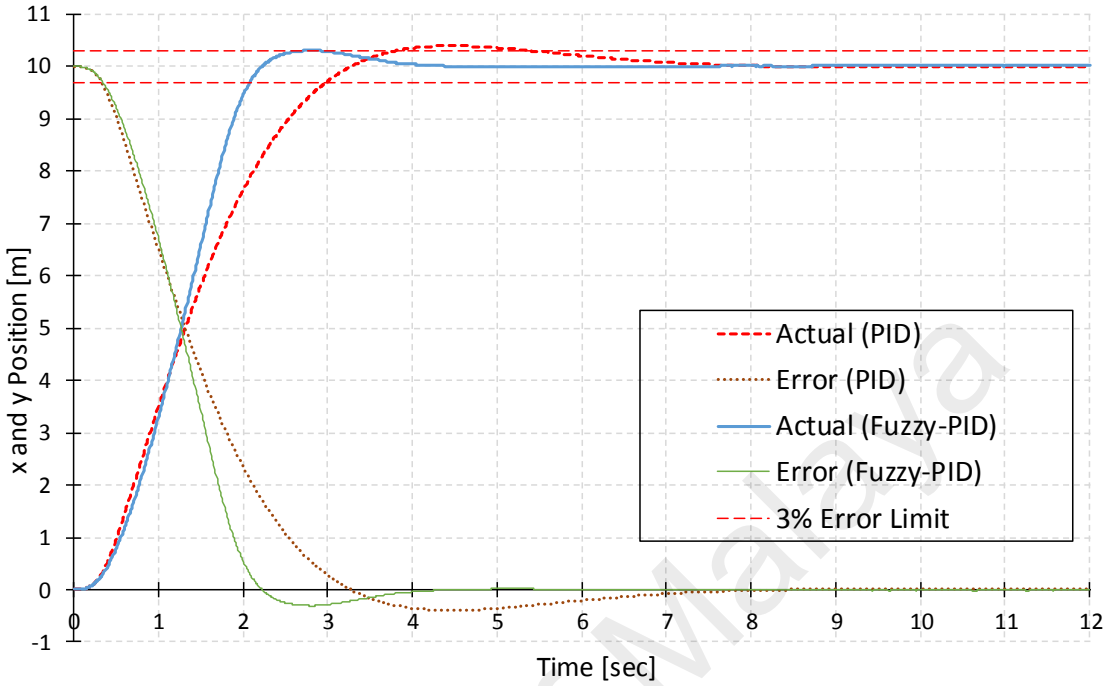

Figure 5.3: Quadcopter *x* and *y* position and their errors with PID and fuzzy-tuned PID controllers

Table 5.3: Performance indicators for translation along x_n or y_n

Response Indicators	Tuned PID	PID Only
Percentage Overshoot, *PO*	3.07%	3.96%
Peak Time, t_{max}	2.82s	4.44s
Delay Time, t_d	1.26s	1.31s
Rise Time, t_r	1.34s	2.03s
Settling time, t_s	2.91s	5.41s

The quadcopter was also commanded to translate along the horizontal plane with a step input with 0.5m to highlight the effectiveness of the Kalman filter in tracking the simulated noisy sensor measurements as seen in Figure 5.4.

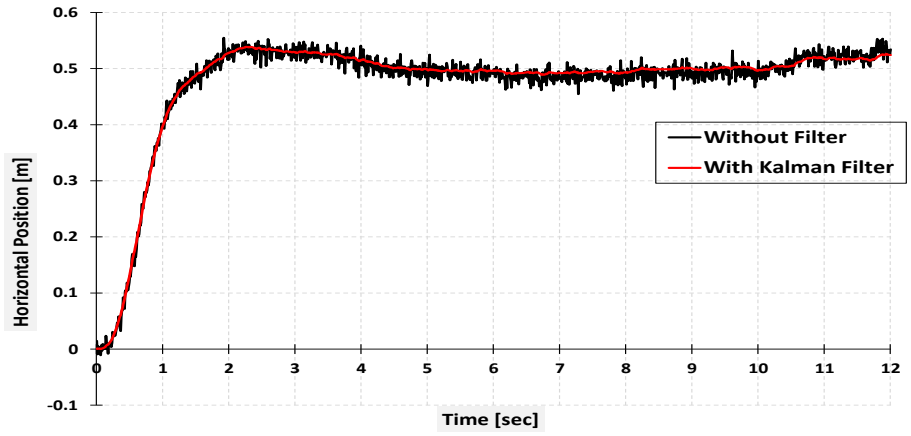

Figure 5.4. Horizontal translation with and without Kalman filter

5.3.2 Horizontal Speeds

The instantaneous horizontal speeds ($\dot{x}$ and $\dot{y}$) of the quadcopter under PID and fuzzy-tuned PID controllers are shown in Figure 5.5. During the 10-meter step input test, the speed increased rapidly to its maximum value under both controllers before it started to reduce. The speed reduction with the fuzzy-tuned PID controller started after the quadcopter had reached 69% of the desired position at t = 1.55s. With the classical PID controller, the speed reduction started after reaching 23% of the desired position at t = 0.76s. The maximum speed was also higher with the fuzzy-tuned PID controller ($\dot{x}_{max} = \dot{y}_{max} = 7\text{m/s}$). The speed of the quadcopter went down to negative after the actual position had reached the peak overshoot, allowing the quadcopter to come back from its overshoot.

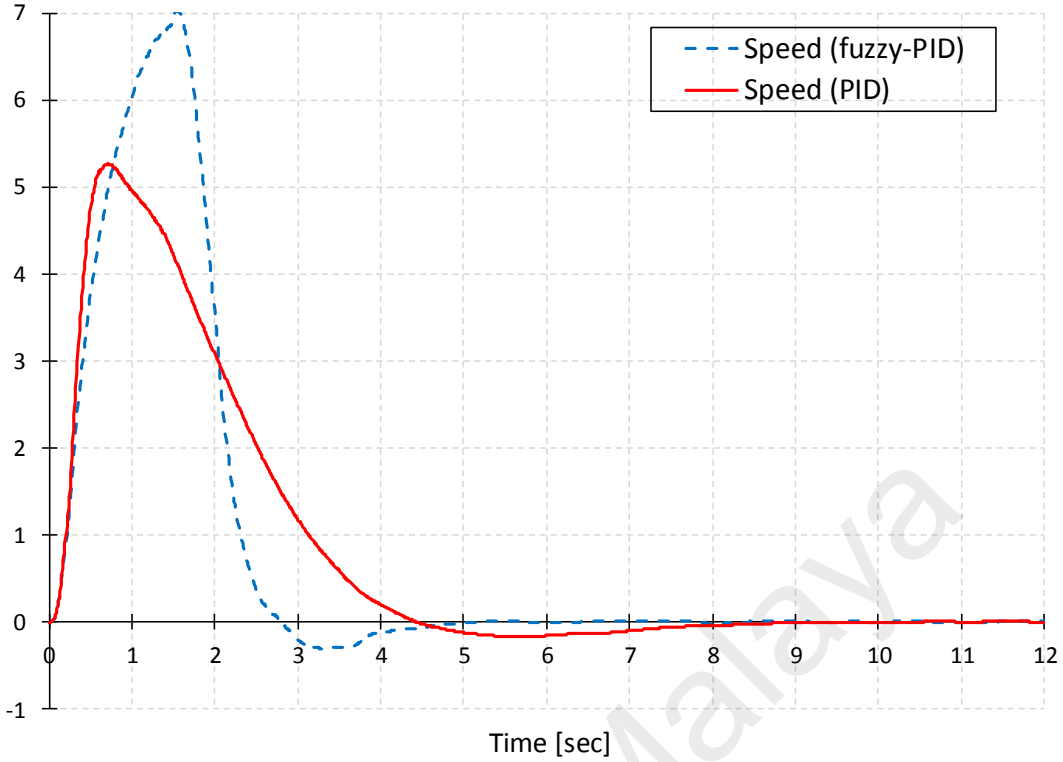

Figure 5.5: Quadcopter horizontal speeds along x_n and y_n direction under PID and fuzzy-tuned PID controllers

When it comes to pure translation along the x_n axis, the pitch angle is of greater interest than other Euler angles. Figure 5.6 shows the reported Euler angles during the 10-meter translation test along the x_n axis under the PID and fuzzy-tuned PID control. Both controllers caused the pitch angle to increase to the maximum right after applying the 10-meter step input. This maximized the thrust component u_x generated by the four propellers (see Figure 5.7) and, hence, the quadcopter accelerated along x_n axis. When the quadcopter approached the desired position (x = 10m), the pitch angle was nearly zero ($\theta \approx 0$). On the other hand, the fuzzy-tuned PID controller kept the pitch angle high for 1.5s and then made it go down to negative, causing the quadcopter to tilt oppositely. Due to the low air friction, this pitching behavior offered good damping that prepared the quadcopter to stop just a few milliseconds before it reached the desired position. Under PID control, the pitch angle started to gradually decrease from its maximum in order to reach the steady-state value $\theta = 0$.

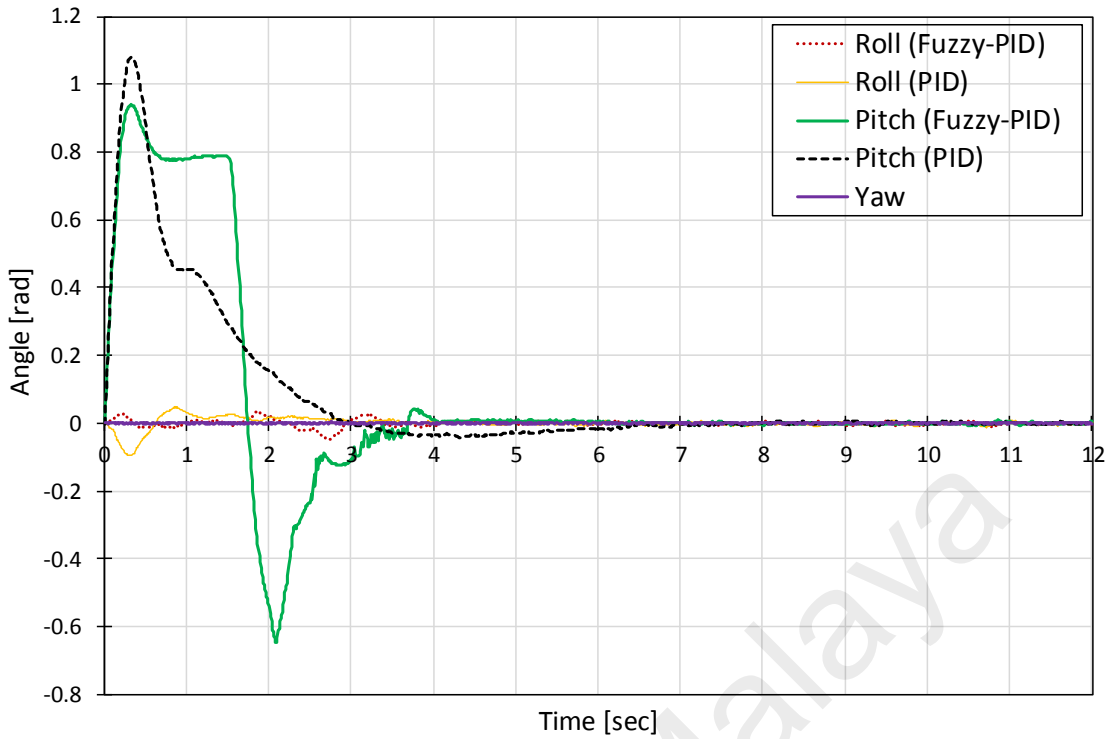

Figure 5.6: Euler angles for translation along the *x* direction

During this test, the roll and yaw angles remained at nearly zero with some vibration due to the noisy sensor readings. Figure 5.6 shows that the roll angle had maximum absolute values of 0.094rad ($\varphi \approx 5.39^{o}$) under PID control and 0.049rad ($\varphi \approx 2.81^{o}$) under self-tuning PID control. This is a good indicator that fuzzy logic has rejected the noise more efficiently.

Figure 5.7 shows the force components of thrust u generated by the four propellers during this test. Force u_z maintained an approximate value of 9.81N. This force is needed to compensate for the quadcopter weight, preventing the quadcopter from having any movement along the vertical axis while translating horizontally along the x_n axis.

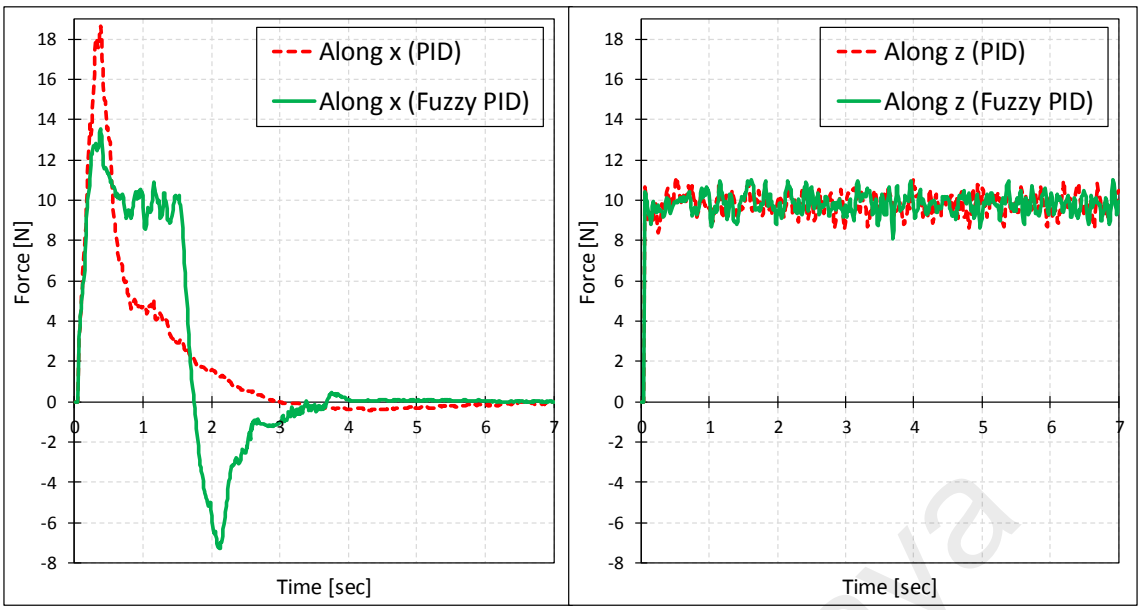

Figure 5.7. u_x and u_z forces when the quadcopter translates along x_n axis under PID and fuzzy-tuned PID controllers

Testing the quadcopter with a 10m step input along the y_n axis gave Euler angles in Figure 5.8. As discussed earlier, the roll angle is the one involved for translations along the y_n axis.

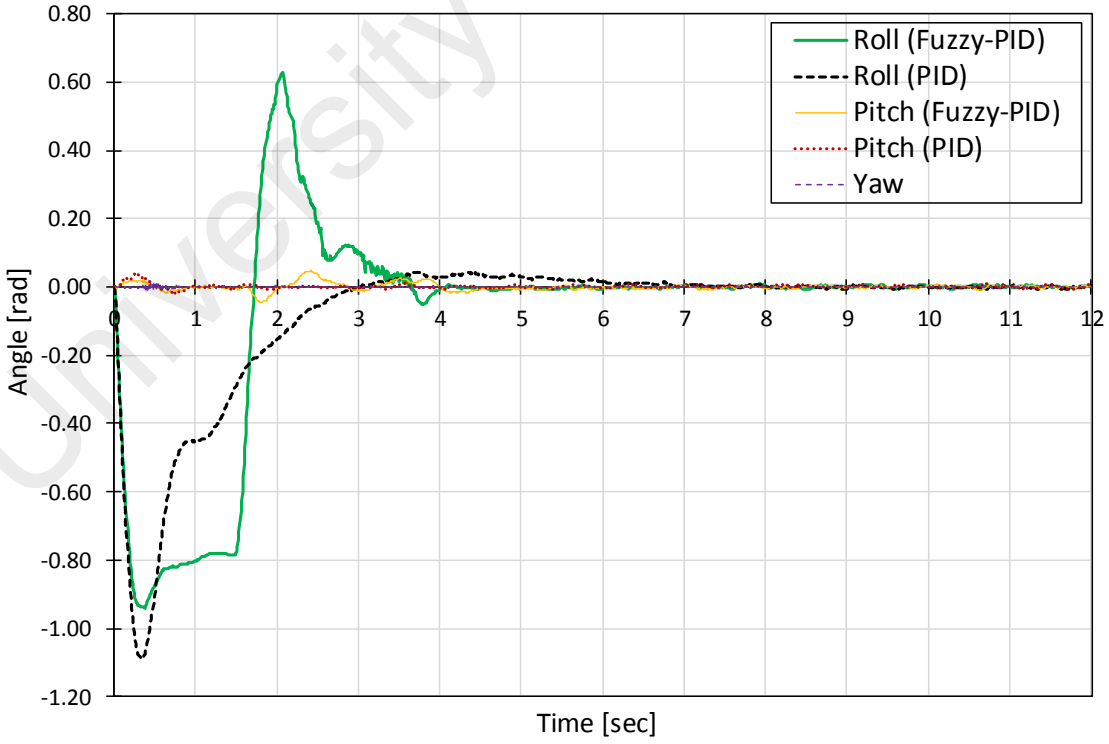

Figure 5.8: Euler angles for translation along the *y* direction.

5.4 Yawing Control Performance

The quadcopter was commanded to make a yaw rotation of 2.0944rad (120deg). This step input produced the response chart shown in Figure 5.9.

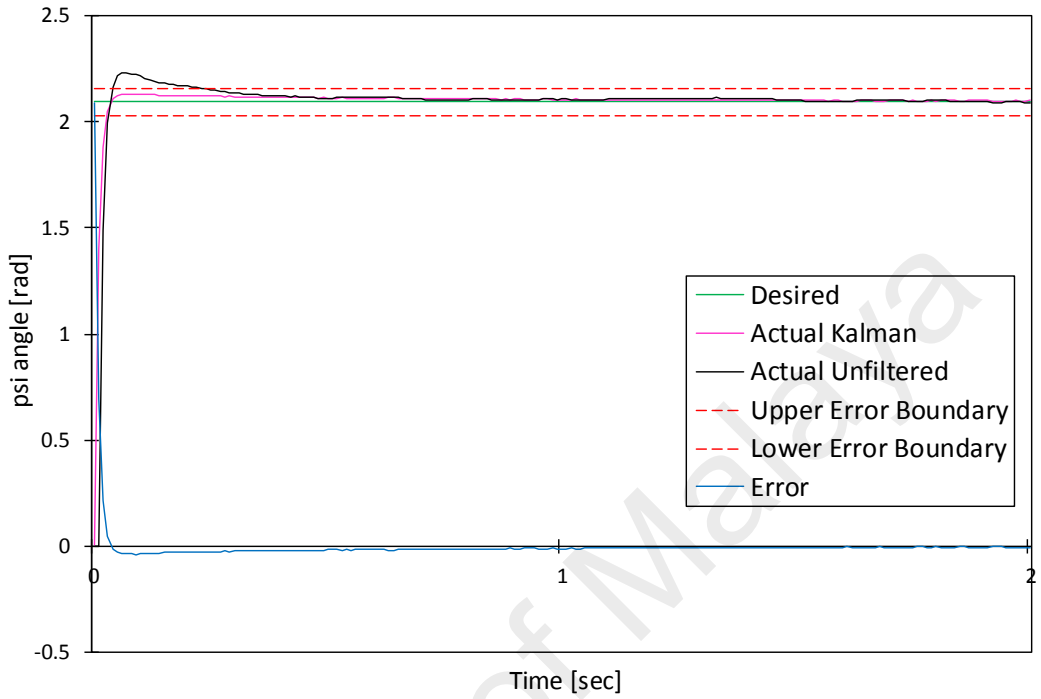

Figure 5.9: Yaw angle step response with PID controller

The performance indicators of this step response are calculated and summarized in Table 5.4.

Table 5.4: Step response indicators for yaw rotation with PID controller.

Step Response Indicators	Value
Percentage Overshoot, *PO*	1.87%
Peak Time, t_{max}	4.41s
Delay Time ,t_d	1.31s
Rise Time, t_r	2.03s
Settling time, t_s	5.43s

5.5 Path Tracking

The quadcopter was also commanded to follow a predefined trajectory fully autonomously. PID and fuzzy-tuned PID controllers were tested in order to compare their performance in stabilizing the quadcopter while tracking the path. The path to

follow was planned in the horizontal plane. Figure 5.10 shows the path that was planned for the quadcopter to follow autonomously. Based on this trajectory, the quadcopter is supposed to start from the origin to translate 10m along the x_n axis, then it continues to rotate counterclockwise in circle with a radius of 10m in the horizontal plane. The simulation stops when one cycle is completed reaching point (10, 0).

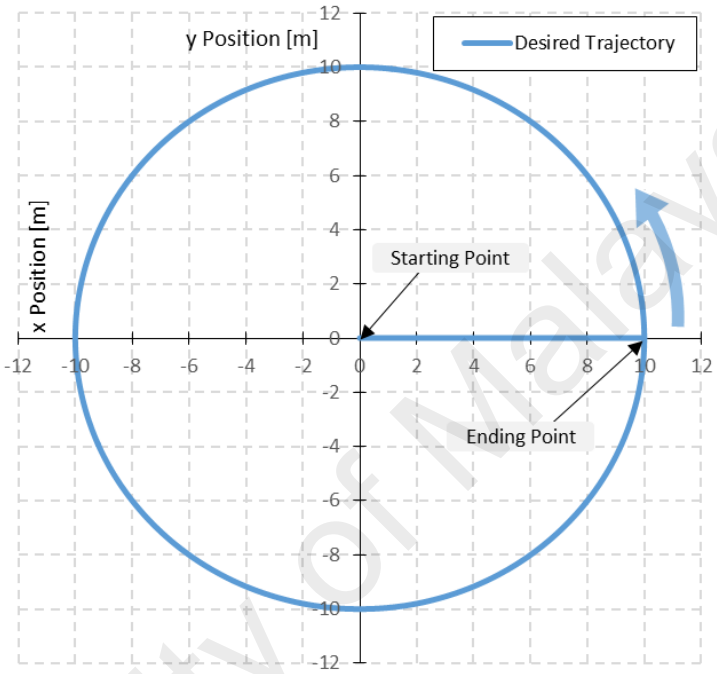

Figure 5.10: Desired trajectory along the horizontal plane

Figure 5.11 and Figure 5.12 show the actual reported track that the quadcopter followed when it is controlled by the PID and self-tuning PID controllers respectively. As can be seen, both of the control techniques are successful in forcing the quadcopter to autonomously follow the planned path with high stability. However, examining Figure 5.11 carefully would reveal a weakness of the implemented PID controller. It is the response time delay causing undesired response lag. The quadcopter is supposed to reach 10m after it initially translates along the x_n axis but, due to the slow controller response, it curved at point (7.30, 0) reaching point (9.80, 1.69) as a successful attempt by the controller to let the quadcopter follow up with the path. This response delay is also the reason why the quadcopter stopped at point (9.72, -2.17) without following the

path to the last planned point (10, 0).

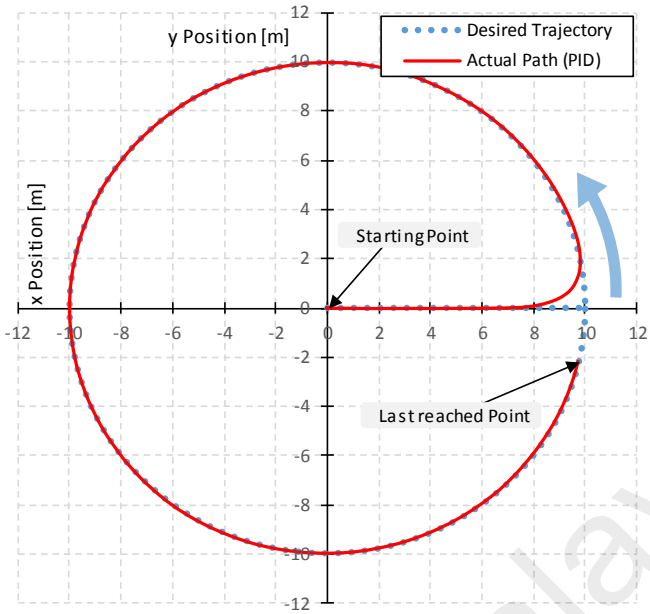

Figure 5.11: Quadcopter autonomous track with PID controllers

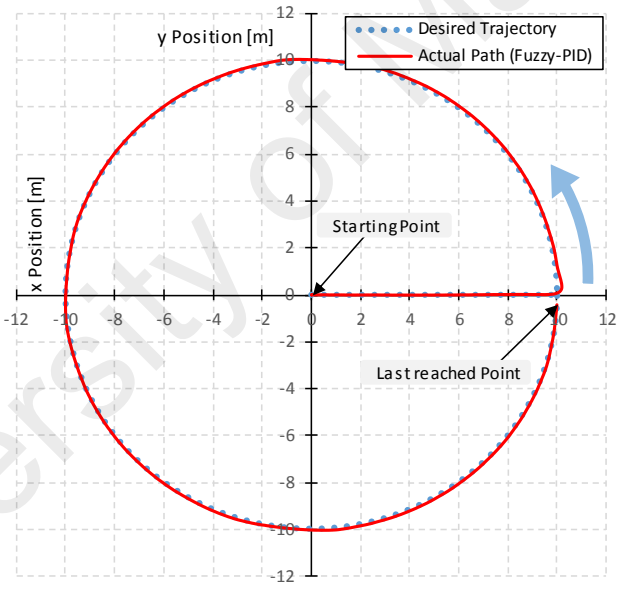

Figure 5.12: Quadcopter autonomous track with fuzzy-tuned PID controllers.

Figure 5.13 and Figure 5.14 compare the performance of the two control techniques for the trajectory tracking for *x* and *y* axes respectively. The time delay caused by the implemented PID controller is almost 1.13s larger than that caused by the fuzzy-tuned PID controller.

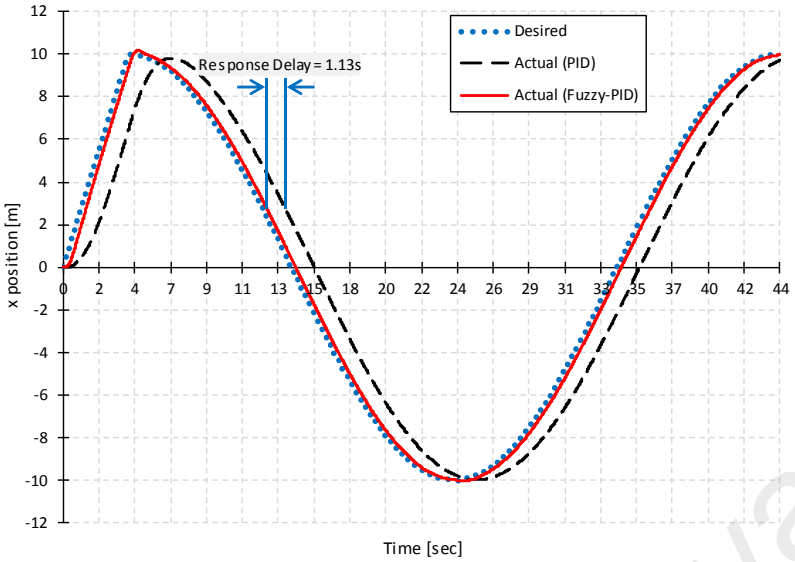

Figure 5.13: *x* axis tracking for PID and fuzzy-tuned PID controllers.

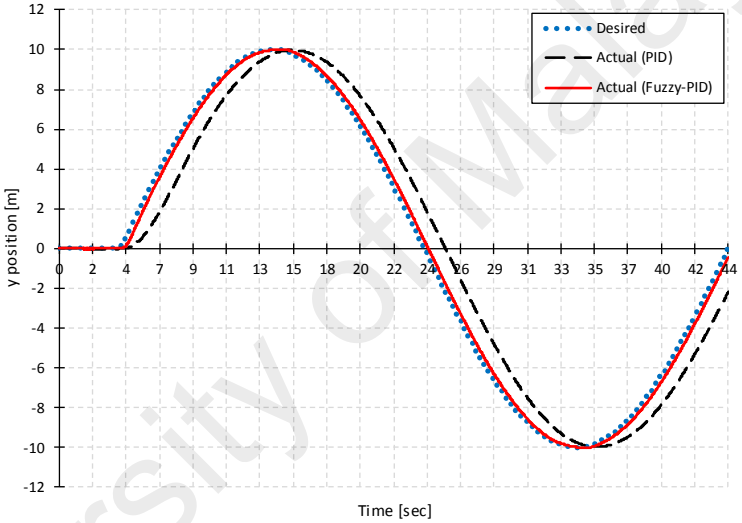

Figure 5.14: *y* axis tracking for PID and fuzzy-tuned PID controllers.

CHAPTER 6. CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

6.1 Conclusions

An integrated simulation platform for a quadcopter equipped with PID and fuzzy-tuned PID controllers has been developed. Quadcopter altitude, horizontal translation, and attitude have been modeled and simulated in Simulink. Quadcopter virtual model has been visualized with the FlightGear Simulator. The simulation results revealed that the two controllers could control and stabilize the quadcopter as a nonlinear system. The two controllers have autonomously controlled the quadcopter and could place it in the desired linear and angular positions. However, the fuzzy-tuned PID controller provided a faster response and smaller overshoots. It has reduced the overshoot along the z_n axis by 6.35% compared to that by the classical PID controller.

The simulation platform was expanded to support the path tracking in the horizontal plane. Fuzzy-tuned PID controllers could efficiently reduce the time delay and enable the quadcopter to strictly follow the planned path with very small tracking errors.

The proposed platform has great potential to complement the engineering educational process. It is a tool that can be included in university laboratories in a form of a virtual UAV lab. It can be a helping tool in eLearning that focuses on aerial vehicles.

The simulation platform could be further enhanced by integrating it with external controllers such as a joystick or smartphone which, in turn, sends commands to the developed simulation model to change the quadcopter motion in real time. The platform is also a promising tool for validating a quadcopter system prior to real implementation. This significantly reduces the time-consuming and intensive testing on expensive quadcopter prototypes; therefore, the total development time and design cost can be minimized.

6.2 Suggestions for Future Work

Simulation using MATLAB and Simulink is really powerful and would be a very helpful tool to use for testing prior to applying the algorithms developed on a real quadcopter. Having proved the effectiveness of the controllers would make it suitable to go ahead and test controllers on a real quadcopter. It is unlikely that the simulated controllers will work on the actual system as well as it did in simulation, so an iterative debugging process is carried out by analyzing results on the actual quadcopter and updating the controller model. This is set as a future plan. Another plan is to implement the sliding mode with PID controller for the sake of achieving even a more robust control and better performance.

References

Abeywardena, D. M. W., Amaratunga, L. A. K., Shakoor, S. A. A., & Munasinghe, S. R. (2009). A velocity feedback fuzzy logic controller for stable hovering of a quad rotor UAV. In *Industrial and Information Systems (ICIIS), 2009 International Conference on* (pp. 558–562). http://doi.org/10.1109/ICIINFS.2009.5429800

Ai-Omari, M. A. R., Jaradat, M. A., & Jarrah, M. (2013a). Integrated simulation platform for indoor quadrotor applications (pp. 1–6). Presented at the Mechatronics and its Applications (ISMA), 2013 9th International Symposium on. http://doi.org/10.1109/ISMA.2013.6547379

Ai-Omari, M. A. R., Jaradat, M. A., & Jarrah, M. (2013b). Integrated simulation platform for indoor quadrotor applications (pp. 1–6). Presented at the Mechatronics and its Applications (ISMA), 2013 9th International Symposium on. http://doi.org/10.1109/ISMA.2013.6547379

Bouabdallah, S., Noth, A., & Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor (Vol. 3, pp. 2451–2456). Presented at the Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, IEEE.

Corke, P. (2011). Mobile Robot Vehicles. In *Robotics, Vision and Control* (Vol. 73, pp. 65–86). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-20144-8_4 http://download.springer.com/static/pdf/344/chp%253A10.1007%252F978-3-642-20144-8_4.pdf?auth66=1402898121_9bbc7c5a952c78e3f57e17c304b1d7b5&ext=.pdf

Desa, H., Ahmed, S. F., & Azfar, A. Z. (2013). Adaptive Hybrid Control Algorithm Design for Attitude Stabilization of Quadrotor (UAV). *Archives Des Sciences*, *66*(2).

DJI. (n.d.). S1000 octocopter. Retrieved from http://www.dji.com/

Elsayed, B. A., Hassan, M. A., & Mekhilef, S. (2013). Decoupled third-order fuzzy sliding model control for cart-inverted pendulum system. *Appl. Math*, *7*(1), 193–201.

Elsayed, B., Hassan, M., & Mekhilef, S. (2015). Fuzzy swinging-up with sliding mode control for third order cart-inverted pendulum system. *International Journal of Control, Automation and Systems*, *13*(1), 238–248. http://doi.org/10.1007/s12555-014-0033-4

E-volo. (n.d.). VC200 Volocopter. Retrieved from http://www.e-volo.com/

Gadda, J. S., & Patil, R. D. (2013). Quadcopter (UAVs) for Border Security with GUI System. *IJRET: International Journal of Research in Engineering and Technology*, *2*(12).

HiSystems GmbH. (n.d.). MK Hexa XL MikroKopter. Retrieved from http://www.mikrokopter.de

LaFleur, K., Cassady, K., Doud, A., Shades, K., Rogin, E., & He, B. (2013). Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface. *Journal of Neural Engineering*, *10*(4), 46003.

Lee, D., Ha, C., & Zuo, Z. (2013). Backstepping Control of Quadrotor-Type UAVs and Its Application to Teleoperation over the Internet, 217–225.

Negnevitsky, M. (2005). *Artificial intelligence: a guide to intelligent systems*. Pearson Education.

Ovaska, S. J., & Valiviita, S. (1998). Angular acceleration measurement: a review. *Instrumentation and Measurement, IEEE Transactions on*, *47*(5), 1211–1217. http://doi.org/10.1109/19.746585

Pandia, A. (2014). Self Stabilized Fire Fighting Quadcopter: A Conceptual Prototype. *IJISET - International Journal of Innovative Science, Engineering & Technology*, *1*(4).

Pounds, P., Mahony, R., & Corke, P. (2010). Modelling and control of a large quadrotor robot. *Special Issue on Aerial Robotics*, *18*(7), 691–699. http://doi.org/10.1016/j.conengprac.2010.02.008

R. Veloso, G. Oliveira, L. S. Passos, Z. Kokkinogenis, R. J. F. Rossetti, & J. Gabriel. (2014). A symbiotic simulation platform for agent-based quadcopters. In *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1–6). http://doi.org/10.1109/CISTI.2014.6876955

Ross, T. J. (2010). Logic and Fuzzy Systems. In *Fuzzy Logic with Engineering Applications* (pp. 117–173). John Wiley & Sons, Ltd. Retrieved from http://dx.doi.org/10.1002/9781119994374.ch5

Salih, A. L., Moghavvemi, M., Mohamed, H. A. F., & Gaeid, K. S. (2010). Modelling and PID controller design for a quadrotor unmanned air vehicle. In *Automation Quality and Testing Robotics (AQTR), 2010 IEEE International Conference on* (Vol. 1, pp. 1–5). http://doi.org/10.1109/AQTR.2010.5520914

Sangyam, T., Laohapiengsak, P., Chongcharoen, W., & Nilkhamhang, I. (2010). Path tracking of UAV using self-tuning PID controller based on fuzzy logic (pp. 1265–1269). Presented at the SICE Annual Conference 2010, Proceedings of.

Santos, M., López, V., & Morata, F. (2010). Intelligent fuzzy controller of a quadrotor. In *Intelligent Systems and Knowledge Engineering (ISKE), 2010 International Conference on* (pp. 141–146). http://doi.org/10.1109/ISKE.2010.5680812

Seidabad, E. A., Vandaki, S., & Kamyad, A. V. (2014). Designing fuzzy PID controller for quadrotor. *International Journal of Advanced Research in Computer Science Technology*, *2*, 221–227.

Sevgi, L. (2006). Modeling and simulation concepts in engineering education: virtual tools. *Turkish Journal of Electrical Engineering & Computer Sciences*, *14*(1), 113–127.

Sinthipsomboon, K., Hunsacharoonroj, I., Khedari, J., Pongaen, W., & Pratumsuwan, P. (2011). A hybrid of fuzzy and fuzzy self-tuning PID controller for servo electro-hydraulic system. In *2011 6th IEEE Conference on Industrial Electronics and Applications* (pp. 220–225). IEEE.

Tamayo, S., Gage, S., & Walker, G. (2012). Integrated Project Management Tool for Modeling and Simulation of Complex Systems. In *AIAA Modeling and Simulation Technologies Conference* (pp. 13–16).

Zulfatman, & Rahmat, M. F. (2009). Application of self-tuning fuzzy PID controller on industrial hydraulic actuator using system identification approach. *International Journal on Smart Sensing and Intelligent Systems*, *2*(2), 246–261.