Analysis of Quadcopter Control & Trajectory Follower Main

A

REPORT ON

MINI SKILLED BASED PROJECT (CONTROL SYSTEM LAB)

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

ELECTRICAL ENGINEERING

By

SHASHANK CHANDRAVANSHI (0901EE211104)

Under the guidance of

Dr. Vikram Saini & Prof. Ashish Patra

Department of Electrical Engineering

DEPARTMENT OF ELECTRICAL ENGINEERING

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE,

GWALIOR (M.P.) – 474005

2024

CERTIFICATE

This is to certify that **SHASHANK CHANDRAVANSHI (0901EE21Q1104)** studying in **MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE,** BATCH-2021-2025) have completed their Mini Skilled Based Project entitled **"Analysis of Quadcopter Control & Trajectory Follower Main"** at **MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE** under my supervision.

It is further certified that they had attended required number of practical classes at **MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE**,GWALIOR for the completion of their Mini Skilled Based Project during 6th semester.

Dr. Vikram Saini & Prof. Ashish Patra

Project Supervisor

DECLARATION OF STUDENT

We hereby declare that the work presented in this SKILLED BASED PROJECT entitled "**Analysis of Quadcopter Control & Trajectory Follower Main"** which is being submitted in the partial fulfillment of the requirement for the award of degree of Bachelor of Engineering in Electrical Engineering is an authentic record of our own work carried out under the guidance of Dr. Vikram Saini & Prof. Ashish Patra, Electrical Engineering Department.

The matter presented in this project has not been submitted elsewhere by us for the award of any other degree/diploma.

SHASHANK CHANDRAVANSHI

(0901EE211104)

Date: 22/04/2024

Place: Gwalior

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief.

Guided by

Dr. Vikram Saini & Prof. Ashish Patra
Department of Electrical Engineering
MITS, Gwalior

ACKNOWLEDGEMENT

Engineers in all disciplines must acquire knowledge of project making. Student, in particular, will find 'project making' as an integral part of their studies that will infuse the spirit of doing practical work in them.

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible whose constant guidance crowned our efforts with success.

We sincerely express our deep gratitude to the management of our college for giving us liberty to choose and to work on the most relevant project i.e. **"Analysis of Quadcopter Control & Trajectory Follower Main".** We are thankful to Dr. Vikram Saini & Prof. Ashish Patra for ensuring that we have a smooth environment at the college and lab. At the very outset we would like to offer our never ending thanks to our project supervisor Dr. Vikram Saini & Prof. Ashish Patra who helped us with our project from the beginning till the end. His continuous surveillance over our work allowed us to work more efficiently.

SHASHANK CHANDRAVANSHI

(0901EE211104)

ABSTRACT

A Quadcopter is an Unmanned Aerial Vehicle (UAV) that is capable of vertical take-off and landing. This thesis presents the mathematical modeling and control of a nonlinear quadcopter system for stabilization and trajectory tracking. The mathematical model of the system dynamics of the quadcopter is derived using Newton and Euler equations with proper references to the appropriate frame or coordinate system. A PID control algorithm is developed for the nonlinear system for stabilization. Another nonlinear control technique called Feedback Linearization (FBL) using Nonlinear Dynamic Inversion (NDI) is discussed and implemented on the quadcopter system. The complete derivation of the full state FBL system using NDI is also shown in this paper. The proposed control algorithms are implemented on the quadcopter model using MATLAB and analyzed in terms of system stabilization and trajectory tracking. The PID controller produces satisfactory results for system stabilization but the FBL system performs better for trajectory tracking of the quadcopter system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

k_P , k_I , k_D	Proportional, Integral and Derivative gains
R	Rotation matrix
W_η^{-1}	Transformation matrix
ξ	Position variables in inertial frame
η	Angular variables in inertial frame, rad
v_I , ω_I	Linear and angular velocities in inertial frame
v_B , ω_B	Linear and angular velocities in body frame
$u \;\; v \;\; w$	Linear velocities in body frame, m/s
$p \;\; q \;\; r$	Angular velocities in body frame, rad/s
ϕ	Roll angle, rad
θ	Pitch angle, rad
ψ	Yaw angle, rad
J_r	Inertia moment of rotor, kg.m^2
ω_i	Angular velocity of the ith rotor, rad/s
Γ	Gyroscopic forces
I	Inertia matrix, kg.m^2
τ_B	External torque
τ_ϕ , τ_θ , τ_ψ	Roll, pitch and yaw torque components
k_t	Thrust coefficient
k_b	Drag coefficient, N.m.s

l	Distance between center of propeller and center of quadcopter, m
F_D	Drag force, N
T_i	Thrust of the i^{th} rotor, N
m	Quadcopter mass, kg
g	Acceleration due to gravity, m/s^2
$c(x)$	Cos x
$s(x)$	Sin x
$t(x)$	Tan x

LIST OF ABBREVIATIONS

VTOL	Vertical Take-Off and Landing
UAV	Unmanned Aerial Vehicle
GPS	Global Positioning System
PID	Proportional Integral Derivative
LQR	Linear Quadratic Regulator
SISO	Single Input Single Output
FPV	First Person View
HD	High Definition
ESC	Electronic Speed Controllers
IMU	Inertial Measurement Unit
MCU	Micro Controller Unit
MEMS	Micro Electromechanical Systems
FBL	Feedback Linearization
NDI	Nonlinear Dynamic Inversion
OS4	Omnidirectional Stationary Flying OUtstretched Robot

CHAPTER ONE

INTRODUCTION

1.1 Background of Study

A quadcopter, also known as a quadrotor, is a rotor-based aerial vehicle. It's a multi-rotor aircraft propelled by four rotors. To balance the torque, these rotors are built with two pairs of opposite rotors revolving clockwise and the other rotor pair moving anti- clockwise.

A quadcopter's dynamics are extremely nonlinear, it is an underactuated system with six degrees of freedom and four control inputs which are the rotor velocities. The quadcopter is controlled by adjusting the rotors' angular speeds, which alters the quadcopter's torque and thrust characteristics.

In the 1920s and 1930s, a few manned designs existed. These vehicles were among the first heavier-than-air vertical take-off and landing (VTOL) vehicles to be successfully tested [1]. However, due to weak stability augmentation and restricted control authority, early prototypes performed poorly, while later versions required too much pilot effort.

Quadcopter designs have recently been popular in unmanned aerial vehicle research (UAVs). To control and stabilize the aircraft, these UAVs use an electrical control system and sensors [1]. Small Unmanned Aerial Vehicles (sUAV) have become a reality thanks to recent advancements in microcomputer technology, sensor technology, control systems, and dynamics theory. Due to their compact size, low cost, and agility, these systems are being used in a range of applications.

1.2 Indoor and Outdoor Quadcopter

Indoor quadcopters are usually relatively small in size and cannot utilize GPS for absolute positioning. They thrive in the absence of strong winds and relatively stable light conditions and their missions are usually shorter than those of an outdoor quadcopter. [2][3].

Figure 1.1: A Ryze Tello Indoor Quadcopter [3]

Outdoor quadcopters are relatively bigger in size than the indoor quadcopter and can utilize GPS for absolute positioning. They are more durable and can fly for longer periods of time than indoor quadcopters. [2].

Figure 1.2: A DJI Phantom 4 Outdoor Quadcopter [4]

1.3 Advantages of Quadcopters

There are a few advantages of a quadcopter compared to other scaled helicopters. Firstly, the design and maintenance of the vehicle is simple due to the absence of additional mechanical linkages used in other helicopters to vary the pitch angle of the rotor blade [2]. Secondly, less damage is incurred during collision because they possess less kinetic energy during flight due to rotor size and design. Some small-scale UAVs have structures that enclose the rotor blades allowing flights through more difficult terrains with a lesser chance of causing damage to the vehicle or its surroundings [2].

1.4 Uses of a Quadcopter

Quadcopters have applications in several fields. Some of them are:

i. Aerial photography and videography.

ii. Aerial surveillance for security purposes.

iii. Reconnaissance in military operations to gather intelligence and scout enemy territory.

iv. Used in modelling of huge structures and terrain.

v. Research platforms – used by researchers to test and develop new ideas in various fields such as navigation, robotics, flight control theory etc.

vi. Agriculture - Crop spraying and monitoring.

Other fields are mining, thermal inspection, product delivery, mapping and surveying, meteorology etc.

1.5 Significance of Study

As the range and complexity of applications for quadcopters expands on a daily basis, the control techniques utilized on the system must also improve.

Historically, basic linear control techniques were used to ensure easy computation and flight stability. However, due to improved modelling techniques and faster computational capabilities, it is now possible to run comprehensive nonlinear techniques on real-time [5]. Linear control techniques can be implemented on quadcopter systems by linearizing the system about an equilibrium point but such approximation may not preserve the dynamics of the system at every point and are not usually very effective in practical scenarios. Due to this conviction, this thesis would be studying the modelling of the quadcopter system and nonlinear control methods that can be implemented on the system for stability and trajectory tracking.

1.6 Aim and Objectives

The project aims to develop a mathematical model of a quadcopter system and implement nonlinear control techniques on the derived model for stabilization and trajectory tracking of a quadcopter.

The objectives of this project are to:

i. Develop the mathematical model of a quadcopter system dynamics.
ii. Develop a PID control algorithm for the derived nonlinear quadcopter system dynamics.
iii. Derive the full state Feedback Linearized system of the derived nonlinear quadcopter system dynamics.
iv. Simulate and perform a comparative analysis of the implemented control techniques on the quadcopter system for stabilization and trajectory tracking.

1.7 Methodology

The following methodology will be employed in achieving the above stated objectives:

The system dynamics of the quadcopter will be developed by deriving the mathematical model of the system based on certain assumptions using the Newton and Euler equations and the state space model would be obtained.

A PD controller will be developed for the derived nonlinear quadcopter system dynamics. The full state Feedback Linearized system of the developed nonlinear quadcopter system dynamics will also be derived using the principle of Nonlinear Dynamic Inversion (NDI) without making any small angle approximation.

The developed control algorithms will be simulated on the quadcopter model with established parameter values using MATLAB and their performances would be analyzed in terms of stabilization and trajectory tracking of the quadcopter system.

1.8 Thesis Organization

The present chapter provides an overview of the project, background information, motivation and importance of the project, the aim and objectives of the project, as well as a short explanation of the methodology to be used throughout the project.

This thesis is broken down into the following chapters:

Chapter Two: This chapter provides a survey of the literature pertaining to the project's subject area. It also includes theoretical context and history, as well as additional concepts necessary for comprehension of the project.

Chapter Three: This chapter details the project's methodology, including the derivation of the system dynamics and state-space equations, as well as the discussion, development, and derivation of the proposed control techniques.

Chapter Four: This chapter illustrates and discusses the results obtained through the application of control techniques. The quadcopter's performance under the proposed control algorithms is also demonstrated using graphical illustrations on MATLAB.

Chapter Five: This chapter highlights the project's accomplishments and contributions. Conclusions, recommendations, and potential future developments of the work are presented.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Quadcopters are used in various fields such as surveillance, reconnaissance, rescue (and so on), because of its ability to engage in maneuvers and navigate complex trajectories. In order to achieve this, stabilization and trajectory tracking of quadcopters are important and it requires comprehension of the system dynamics and implementation of a suitable controller. This chapter will provide insight into certain topics needed to properly understand the scope of the thesis. It will also conduct a review of academic literature on previous works based the system dynamics and control of a quadcopter.

2.2 Components of a Quadcopter

The major components of a quadcopter are [6]:

i. Frame
ii. Propeller
iii. Motors
iv. Speed Controller
v. Flight Controller
vi. Battery
vii. Transmitter and Receiver

Other parts include FPV Camera, Video Transmitter, Antenna, GPS, HD Camera etc.

Figure 2.1: Components of a Quadcopter [7]

2.2.1 Frame

The quadcopter's frame is the principal framework or skeleton on which the other components are connected. After deciding what type of activities the quadcopter would be performing, the size should be determined based on the specific needs. The frame size will define the size of the propellers to be used or vice versa. The size of the propellers will also determine the size of the engines, which indicate the speed controller’s current rating.

Figure 2.2: Frame

2.2.2 Propellers

Propellers are chosen based on their thrust generating ability according to motor RPM to achieve the required vehicle performance and altitude. To allow hovering, at least each propeller should produce a force that is equivalent to a fourth of the weight of the vehicle when spinning within the RPM range of the motors. Two propellers rotate in the clockwise direction and the remaining two rotate anticlockwise in order to balance the torque generated by the propeller rotation.

Figure 2.3: Propellers

2.2.3 Motors

The motors are the primary battery drain in the quadcopter thus it is extremely essential to have an effective combination of propeller and motor. The motor speed rating is in kV, usually a lower kV motor produces greater torque and a higher kV spins faster, although this is without the propeller connected.

Figure 2.4: Motor

2.2.4 Electronic Speed Controllers (ESCs)

ESC is a device that converts signal from the flight controller into phased electric pulses to determine and control the speed of a brushless motor. The current rating of the ESC must be greater than the current drawn by both the propellers and motors combined.

Figure 2.5: Electronic Speed Controller

2.2.5 Flight Controller

The flight controller is a major part of the quadcopter and is regarded as the vehicle's brain. A flight controller obtains input from sensors and makes use of algorithms to control the behavior of the vehicle while in motion. The flight controller consists of the Inertial Measurement Unit (IMU) and the Microcontroller Unit (MCU). The IMU is an electronic device that measures the body's orientation by monitoring external forces acting on the certain parts of the body. A gyroscope, accelerometer and a magnetometer are used in most of the IMUs (MEMS) to determine the position, acceleration, and angular rates. The MCU contains the processors and memory and oversees different operations of the controller.

Figure 2.6: Flight Controller

2.2.6 Battery

Batteries affect a quadcopter's flight in terms of speed, flight time and thrust. Stronger batteries will definitely enhance these characteristics. However, recommended batteries for the chosen motors should always be used since more powerful batteries may overheat

and ultimately burn motors. Batteries to be used for a quadcopter should be selected based on the maximum current that may be drawn by the quadcopter.

Figure 2.7: Battery

2.2.7 Transmitter and Receiver

A transmitter enables pilot and drone communication. The transmitter translates human instructions for various flying operations into signals for the vehicle. The number of channels is the most essential factor to consider when choosing a transmitter. A receiver accepts the radio signal from a transmitter and transforms it to the microcontroller's electrical signal. The number and type of channels are key elements in selecting a receiver.

Figure 2.8: Transmitter and Receiver

2.3 Mathematical Modelling

Mathematical modelling is the initial and most essential step towards understanding the dynamics of a quadcopter system. The state equations governing the system dynamics of a quadcopter are derived using the Newton-Euler equations and Euler-Lagrange equations as shown in [8]. This paper gives a simplified but also a more accurate model because the drag force caused by air resistance is also considered. It neglects other complex dynamics such as blade flapping and certain aerodynamic effects due to difficulties in modelling. A matrix approach and the use of bond graphs to describe the dynamic model of a quadrotor is elaborated in [9]. A more detailed derivation of the system dynamics is given in [10]. This paper also elaborates the derivation of the external forces on the quadcopter considering other aerodynamic effects that were neglected in the models established in

[8] and [9]. Momentum theory is used to describe the steady state rotor thrust and torque for a hovering rotor. The dynamic model of the system taking into account the rotor flapping and stiffness for induced drag is also shown here. A complex study of the dynamic model of the quadcopter is also seen in [1]. This paper considers exogenous effects on the quadcopter including aerodynamic effects such as propeller rotation and blades flapping, inertial counter torques, friction, gravity and gyroscopic effects. The aerodynamic forces and moments here are also derived using a combination of blade element and momentum theory.

2.4 PID Controller

A PID controller is a feedback loop mechanism that continuously calculates an error value which is the difference between a desired value and a measured variable and tries to correct such errors using proportional, integral and derivative terms.

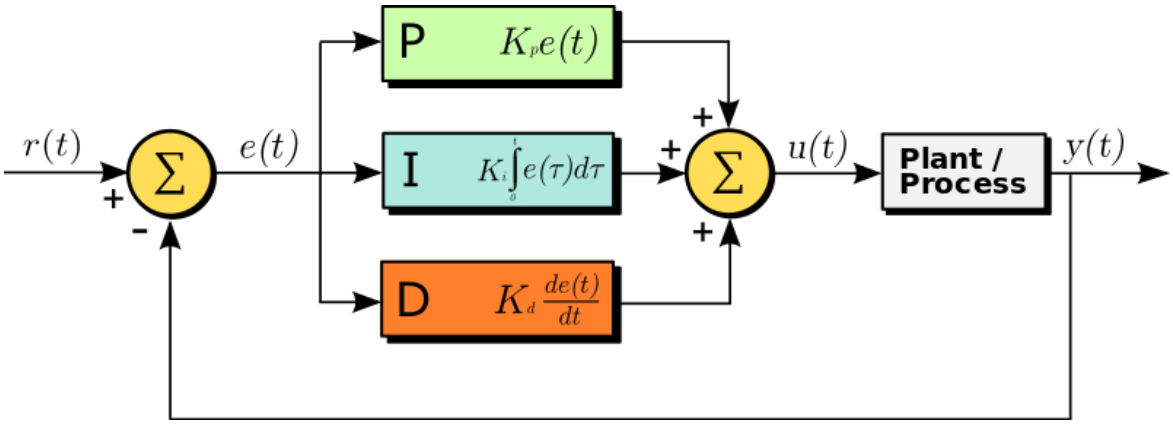

Figure 2.9: Block diagram of a PID controller in a feedback loop.

The PID controller is the simplest control algorithm to implement. From Figure 2.9, where r(t) is the desired value and y(t) is the measured variable, the general form of the PID control algorithm is given by:

$$e(t) = r(t) - y(t) \tag{2.1}$$

$$u(t) = K_P e(t) + K_I \int e(\tau)d\tau + K_D \frac{d}{dt} e(t) \tag{2.2}$$

where the u(t) represents the control input and e(t) is the error between the desired value and the measured variable. K_P, K_I and K_D are the parameters for the proportional, integral and derivative elements of the PID controller.

The PID controller may utilize the three controller terms of proportional, integral and derivative to implement precise and optimum control output [11]. The controller attempts to minimize the error over time by adjusting the control variable u(t). Proportional produces an output that is proportional to the error value e(t). This output is obtained by multiplying the error value by a proportional constant. Integral considers the past error values and seeks to reduce the residual error by adding a control effect obtained from the

integration of the cumulative previous errors. Derivative anticipates or predicts the future

behavior of the error and exerts a control effect produced by the rate of change of the error with respect to time [12].

Although the PID control strategy has three control terms, some situations might only need the use of one or two terms to achieve suitable control. This is done by assigning a zero value to the unused term or parameter and it becomes a P, I, PI or PD controller if the other control terms are not present. The PID control algorithm is widely used because it relies on the measured variable's response rather than standard information of a basic process or model [11].

2.4.1 PID Control Effect on Closed-Loop Systems

A proportional controller will decrease rise time and reduce the steady-state error, but never eliminate it. While an integral controller eliminates steady-state error, it may deteriorate the transient response. A derivative controller improves system stability, decreases overshoot, and also improves transient response. Table 2.1 shows the effect of PID parameter gains on system characteristics [2].

Table 2.1: Response of the different parameters of the PID gain

Controller	**Rise time**	**Overshoot**	**Settling time**	**Steady-state error**	**Stability**
Proportional (Kp)	Decreases	Increases	Little change	Decreases	Degrade
Integral (Ki)	Decreases	Increases	Increases	Eliminate	Degrade
Derivative (Kd)	Little change	Decreases	Decreases	No effect	Increase if Kd is small

Note that these relationships may not be precisely the same since they are mutually dependent. Changing one of these controller parameters may alter the effects of the other two.

2.4.2 PID Control Effect on a Quadcopter

The modification of any of the PID parameters affects system stability. There are usually three PID loops with their own coefficients, so that for each axis (pitch, roll and yaw), you have to specify P, I and D values [2].

i. ***Proportional Gain Coefficient***: This is the most important parameter as the quadcopter cannot attain stable flight without it. This coefficient prioritizes a signal between human control or gyroscope measured values. The greater the value of the gain, the more sensitive the quadcopter becomes to angular change. The Quadcopter will be more difficult to maintain stability if the value is too low and if the value is too high, it will oscillate with a high frequency.

ii. ***Integral Gain Coefficient***: This parameter may improve the precision of the angular orientation. For instance, if the Quadcopter is disrupted and its position changes, this parameter accounts for the difference and attempts to rectify it. A high value of this parameter could cause the quadcopter to respond slowly and oscillate as well but with a lesser frequency.

iii. ***Derivative Gain Coefficient***: This enables the Quadcopter to attain its desired state faster. It also reduces control action quickly if error is rapidly decreasing. It increases the response speed of the quadcopter and an increase the Proportional parameter might be observed.

2.5 Linear Quadratic Regulator

The Linear Quadratic Regulator (LQR) is a popular linear control algorithm that provides optimally controlled feedback gains to enable stability and high-performance design of closed-loop systems. The LQR controller is implemented on a linearized system and makes use of a mathematical algorithm to minimize a cost function with when the weighting factors are provided. The cost function is often described as the sum of deviations of certain key measurements from their desired values [13].

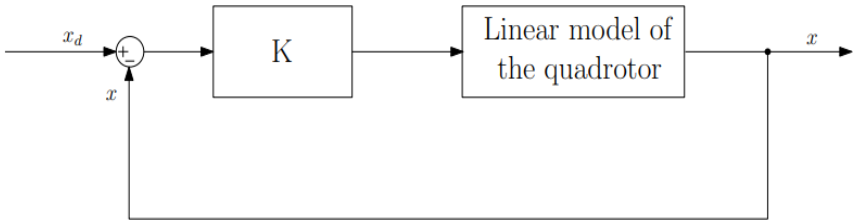

Figure 2.10: LQR Control Diagram

Considering a dynamic system of the form:

$$\dot{x} = Ax + Bu \tag{2.3}$$

$$y = Cx \tag{2.4}$$

The cost function is given by:

$$J = \int_{t0}^{\infty} \{u(t)^T \,.R.\, u(t) + [x(t) - x_d(t)]^T \,.Q.\, [x(t) - x_d(t)]\}\, dt \tag{2.5}$$

Where R and Q are the costs of the actuators and state respectively.

The control input is given by:

$$u = -k \, . \, [x(t) - x_d(t)] \tag{2.6}$$

where:

$$k = R^{-1}B^T P \tag{2.7}$$

P is a symmetric matrix whose eigenvalues are positive and is the solution to the algebraic Riccati equation:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{2.8}$$

The value of k is determined by solving the Riccati algebraic equation. This is done by MATLAB using the LQR function:

$$k = LQR(A, B, Q, R)$$

2.6 Nonlinear Dynamic Inversion

Aircrafts don't act as linear systems all the time. They exhibit nonlinear characteristics in certain flight scenarios and therefore need a nonlinear controller to control them in such cases. These control methods have to be robust as possible since the model is never a completely accurate representation of the actual system. Nonlinear dynamic inversion has proven to be a simple method of controlling nonlinear systems.

If we assume the aircraft model is a single input single output (SISO) model of the form:

$$\dot{x} = f(x) + g(x)u \tag{2.9}$$

where x represents the state vector and $f(x)$ can be a nonlinear function. Equation 2.9 above can be rewritten as:

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} x_2 \\ \vdots \\ x_n \\ b(x) \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a(x) \end{bmatrix} u \tag{2.10}$$

All the nonlinear terms and the input only affects x_n. The control input can be defined as:

$$v = b(x) + a(x)u \tag{2.11}$$

$$u = a^{-1}(x)(v - b(x)) \tag{2.12}$$

where v represents the virtual control input which can be utilized to control the whole system in a linear manner. This approach is called nonlinear dynamic inversion which is used to control nonlinear systems in a linear way.

State feedback is used to set the virtual input, therefore:

$$v = -k_0 x - k_1 \frac{dx}{dt} - k_2 \frac{d^2x}{dt^2} \dots - k_{n-1} \frac{d^{n-1}x}{dt^{n-1}} \tag{2.13}$$

Since $\frac{d^n x}{dt^n} = v$, the entire system can be turned into a linear closed loop system of the form:

$$\frac{d^n x}{dt^n} + k_{n-1} \frac{d^{n-1}x}{dt^{n-1}} + \dots + k_1 \frac{dx}{dt} + k_0 = 0 \tag{2.14}$$

The characteristics of the closed-loop system can be set by choosing the appropriate value of k_i. Two loops are considered in NDI, the outer loop which involves finding v and the inner loop which involves finding the corresponding value of u and applying it to the system.

Certain conditions must be fulfilled in order to implement NDI; the full state of the system and the entire system model must be known. If the full state is not known, an observer or estimator can be used to approximate it to an extent.

2.7 Control Architecture

Control strategies or algorithms are the types of logic used in a controller to evaluate the error between a measured value and a desired value, process it and engage in error correction processes to enable proper control of the system [14].

Several control algorithms have been researched for quadcopters for attitude stabilization and trajectory tracking. The objective is to implement a control algorithm that permits the quadcopter's states to converge to an unpredictable set of time-varying reference states. It is possible to control the quadcopter by linearizing the system dynamics around a stable point and applying linear control techniques as shown in [15]. Nonlinear control algorithms such as backstepping, sliding mode [16], [17] and feedback linearization [16],
[18] are effective for quadcopter control as better performances have been observed from their implementation.

Numerous control algorithms for attitude control have been studied. These include PID controllers [5], [8], [10], inverse dynamic control and sliding mode controller [19], linear quadratic regulator [5], and feedback linearization control [19], [18], [20]. A hierarchical control approach is used in [10], where different levels (controlling rotor rotational speed, vehicle attitude and trajectory position) form feedback loops as presented in Figure 2.11.

The controller function is categorized into trajectory controller and attitude stabilization controller.

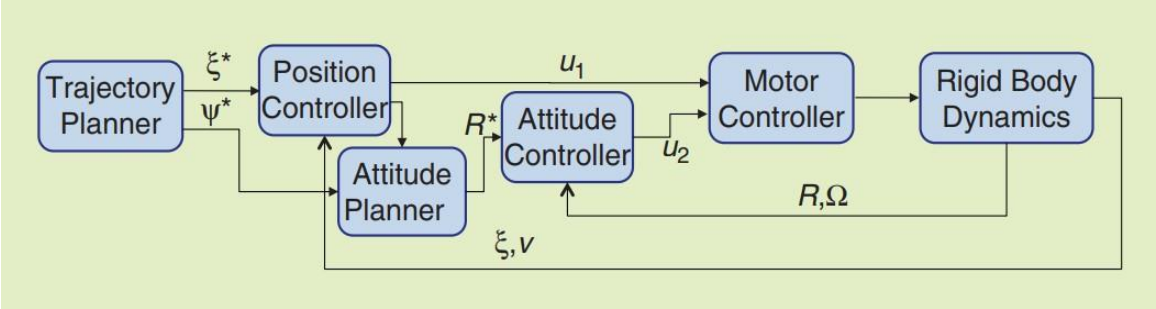

Figure 2.11: Control Model [10]

A rapidly converging attitude controller is used because of errors due to rotations. A skew-symmetric matrix is obtained to arrive at the desired attitude vector from the actual attitude vector. The error matrix is linearized for small deviations from the hover position and a PD controller gives adequate performance. A PD controller with fewer complexities is implemented in [8]. The torque and thrust components here are gotten from the error measured in attitude values and the various rotor angular speeds are processed from thrust and torque controls. A PID controller with provision for angular acceleration feedback is utilized in [21] for attitude stabilization. This causes significant gain increase resulting in higher bandwidth.

Feedback linearization is a commonly used method in the control of nonlinear systems. The method involves transforming the nonlinear system into a linear system by changing the variables and providing an appropriate control input. There are two methods of feedback linearization control presented in [16]. One of them is dynamic feedback which entails the use of dynamic feedback control law. The position variables are considered as the output function and the second derivative of the thrust input is used. Once the system

satisfies the condition necessary for feedback linearization, it can be converted into a system that is completely linear and controllable using dynamic feedback. The second method is dynamic inversion which is done with small angle approximation and makes use of attitude variables as the output variables. The two methods are discussed while comparing feedback linearization and adaptive sliding mode control for a quadrotor in [16]. This paper also implements Linear Quadratic Regulator on a quadcopter where the system is linearized using small angle approximation.

2.8 Comparative Study of Control Algorithms on A Quadcopter System

A comparative analysis between different control algorithms for attitude stabilization is presented in [19]. Several control algorithms are implemented on the system and researched to determine the optimal quadcopter controller. The control algorithms applied

to the quadcopter system here are PID control, backstepping, inverse and sliding mode control.

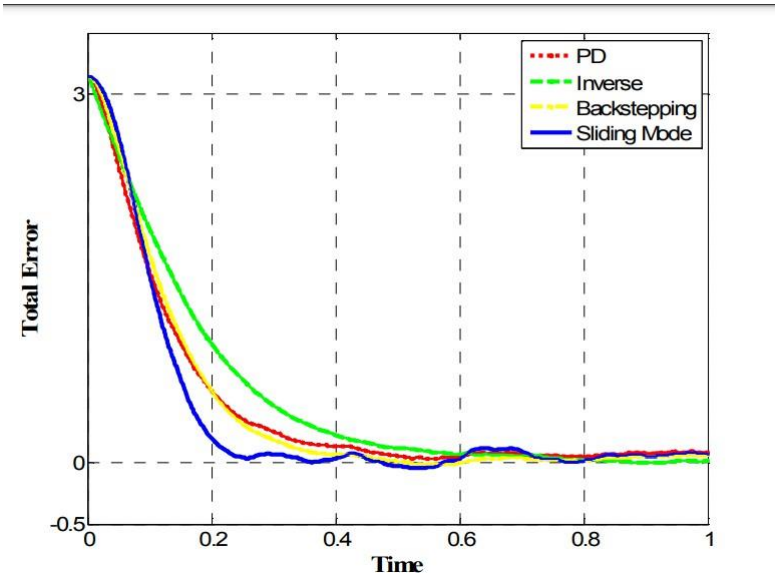

Figure 2.12: Comparison of control algorithm based on total error evaluation [19]

Figure 2.12 shows that the sliding mode control has a better performance than the other control algorithms especially on high initial conditions. The PD controller also presents adequate performance and it is much easier to implement. A better settling time is observed in the inverse control algorithm although the response is slower compared to the PID and sliding mode control algorithms.

A comparison between Linear Quadratic Regulator and feedback linearization control algorithm is presented in [5]. The feedback linearization schemes are the dynamic inversion with zero dynamics stabilization based on Static Feedback Linearization and the exact linearization and non-interacting control via dynamic feedback based on Dynamic Feedback Linearization. These nonlinear control algorithms are compared with LQR control in terms of performance. From the analysis, the Linear Quadratic Regulator is observed to be slower with a low value of overshoot. A higher value of overshoot is seen

using the dynamic inversion with zero-dynamics stabilization control and this control algorithm is faster than the Linear Quadratic Regulator and the exact linearization and non-interacting control via dynamic feedback.

2.9 Review of Related Works

A comprehensive mathematical modeling of the kinematics and dynamics of a quadcopter system is described in [22]. The kinematic aspect of the modeling depicts the quadcopter motion without taking into account the forces acting on it whereas the dynamics explains the forces causing the motion. This gives an overview of the response of quadcopters when under the influence of extraneous forces. Nonlinear state equations for the system were derived and the manual method to obtain minimum steady state error for PID tuning is presented. This research resulted in a quadcopter with a composite frame, assembled and implemented in real time for pitch, roll and stabilization with PID control. The results yield adequate performance when the hardware configuration is at the hover point.

The mathematical modelling and control of a quadcopter is presented in [8]. This paper shows the basics of quadcopter modelling and control to serve as a stepping stone for future research advancement. A detailed study of the mathematical model of the quadcopter dynamics is given and the differential state equations of the system dynamics is derived from both the Newton-Euler and Euler-Lagrange equations. A simple mathematical model is presented in this paper neglecting several aerodynamic effects and the modelling of the electric motor spinning the rotors. The developed model is simulated to analyze the behavior of the system and control algorithms are implemented for stability

and trajectory control of the system. The PD controller is used to achieve this and a

heuristic method is developed for trajectory control of the quadcopter flight. An integration of the PD controller and the heuristic method is used to minimize the disturbance on the quadcopter system caused by external forces. The PD control algorithm was integrated into the heuristic method because the heuristic method developed did not account for unmodelled disturbances (wind). A decline in performance of the PD controller was observed if the parameter values are extremely small or high. The necessity for an actual experimental prototype was identified, so that realistic and accurate findings could be acquired and comparison between the simulated results and real-life measurements would be possible.

A review of control algorithms for autonomous quadrotors is presented in [20]. Several control algorithms are analyzed on the system to propose hybrid systems which would combine the advantages from more than one control algorithm. The focus of this paper was to implement the optimal control algorithm of the quadrotor manned aerial vehicle for game counting in the protected game reserves in Africa. The control algorithms analyzed in this paper are: PID, Linear Quadratic Regulator (LQR), Sliding mode control, Backstepping control, Adaptive control and Artificial neural networks. From the review, no particular control algorithm gives the best performance in all required features in terms of fast response, disturbance rejection, stability, robustness, adaptability, optimality and tracking ability. A hybrid control system could have the best combination of some of these features but doesn't guarantee an overall optimum performance. Certain features are more important depending on the application of the quadrotors hence compromises would have to be made to implement control algorithms that excel in such areas of application. A quadrotor to be used for game counting is designed prioritizing certain characteristics such

as high endurance, high agility, high cruising, low noise and vertical take-off and landing ability. This paper concluded that the designed quadrotor would be suitable for assisting nature conservationist in game counting and obtaining accurate animal statistics.

A thesis on quadcopter flight mechanics model and control algorithms is shown in [2]. The focus of this thesis was to develop a nonlinear model of a quadcopter flight mechanics with suitable control algorithms for stabilization and implement it in MALTLAB/Simulink. The dynamics of the quadcopter were derived from the equation of motion and forces and a nonlinear model was obtained based on these dynamics. Aerodynamic effects such as non-zero free steam and blade-flapping are ignored in the derivation of the system equations but other forces such as air friction and drag forces are considered. For analysis of the system, the model is linearized at a stable hover point and both the linear and nonlinear models are analyzed. The Gradient Descent Method is used for controller tuning to obtain optimum control parameters and the system is simulated for a given period of time.

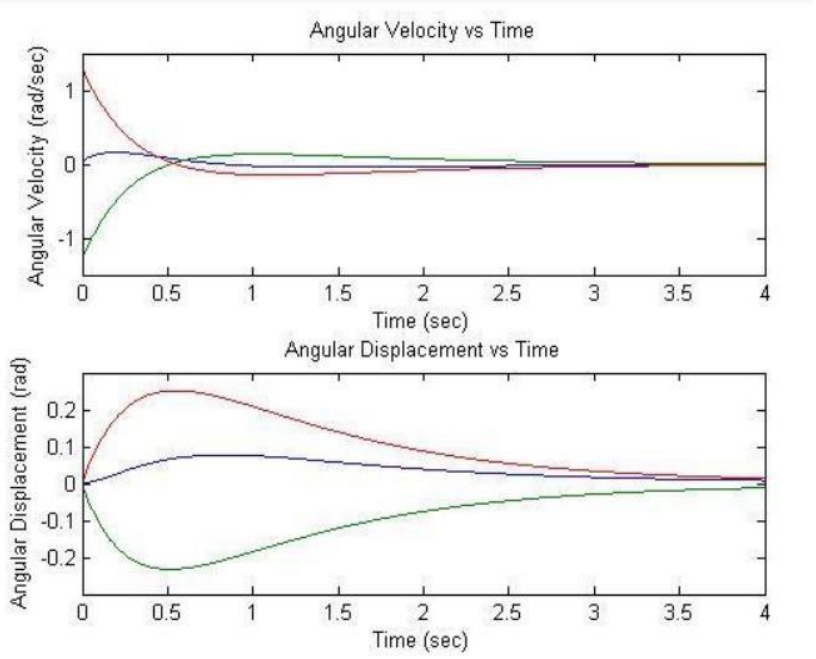

Figure 2.13: Response of the Quadcopter due to PID controller

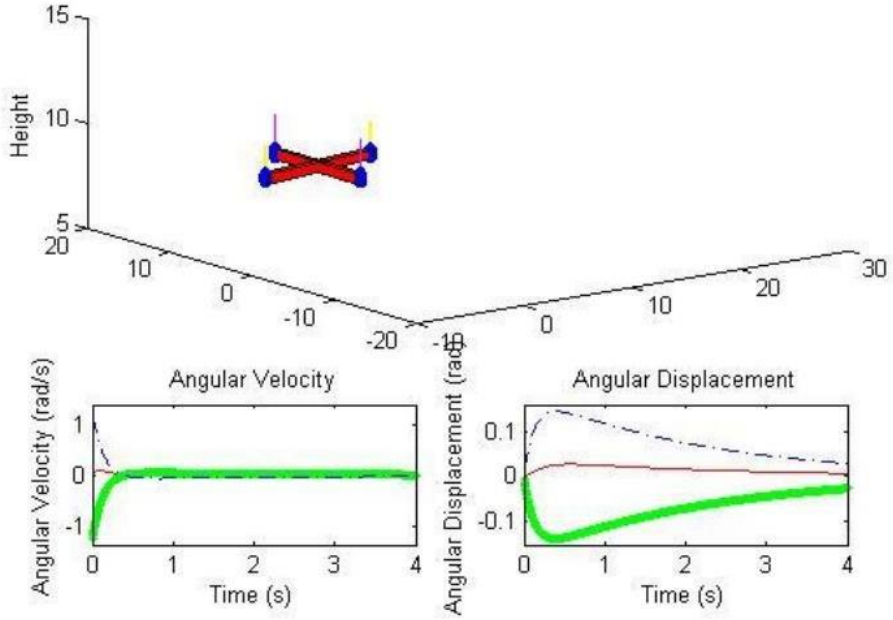

Figure 2.14: Visualization of the Quadcopter for the manually tuned PID controller [2]

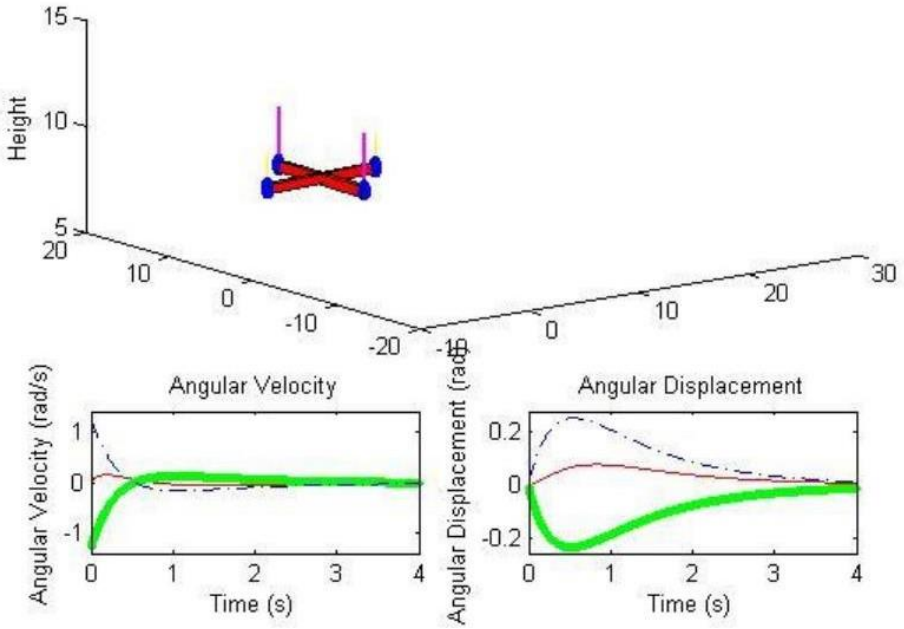

Figure 2.15: Quadcopter Visualization for the automatically tuned PID controller [2]

Figure 2.13, 2.14 and 2.15 show results from the simulations, a better performance was observed from the automatically tuned PID controller (using the Gradient Descent method) than the manually tuned PID controller.

The design and control of quadrotors in relation to autonomous flying [1] is a thesis that entails the modelling, design and control of small flying drones with a focus on Vertical Take-Off and Landing (VTOL) systems. In this thesis, an autonomous quadrotor called OS4 is modelled and simulated with various controllers. A more complex mathematical model is presented here as the system dynamics accounts for more realistic aerodynamic coefficients along with sensor and actuator models. The OS4 quadcopter model is simulated with five different control algorithms. The first one was based on Lyapunov theory and was implemented for attitude stabilization but after simulation on OS4, it was observed that it was not stable enough to allow hover flight. The second control algorithm

implemented was the PID controller and it was suitable for attitude stabilization of the quadcopter system when flying near hover but this was only possible in the absence of huge disturbances. The third control algorithm was an LQR controller which presented adequate stabilization but this control algorithm was observed to be less adaptive than the PID. The fourth control algorithm was the Backstepping controller which achieved excellent stabilization and control in the presence of relatively large disturbances. The fifth control algorithm implemented was the Sliding-mode technique but this control algorithm did not give desirable results because the switching characteristics of the controller was not compatible with the dynamics of the quadcopter system. After the implementation and analysis of these five control algorithms, a suitable control algorithm was developed by the integration of the PID and the Backstepping algorithm called Integral Backstepping. This was used for altitude, attitude, position and trajectory control and the implementation of this control algorithm on the OS4 enabled the system to take- off, hover, land and avoid collisions. The thesis noted that the OS4 was the first collision avoidance quadcopter system.

2.10 Conclusion

In this chapter, previous works on modelling and control of a quadcopter were discussed and reviewed. A precise overview of those topics was analyzed and examined to provide a more rounded understanding of the quadcopter system components, modelling, and control. From the papers reviewed above, the importance of implementing a good control for a quadcopter is emphasized and different linear and nonlinear control algorithms are studied. In this thesis, the full state FBL system control based on dynamic inversion for

the nonlinear system will be derived and computed without the use of small angle approximation which was used in the reviewed papers above. This paper will focus on presenting an adequate mathematical representation of the system dynamics and effective control techniques for a quadcopter.

CHAPTER THREE

SYSTEM MODELLING AND CONTROL

3.1 Introduction and Basic Concepts

A quadcopter is an aerial vehicle with four rotors that enable motion in different directions. Proper explanation of the system dynamics would require comprehension of the 6-degrees of freedom concept which presents the position and orientation of the quadcopter in three dimensions (3-D).

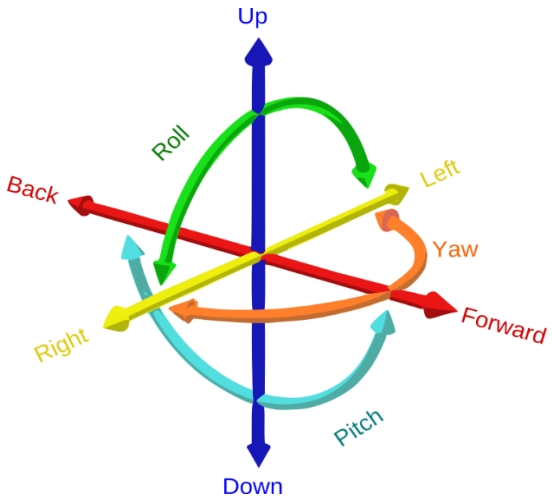

Figure 3.1: Six Degrees of Freedom [taken from Wikipedia]

The 6-degrees of freedom describes the position of a body with six coordinates categorized in two reference frames. The first reference frame is the fixed coordinate system known as the inertial or earth frame which is depicted by the x, y, and z coordinates in the cardinal points of North, East and Down. The second reference frame is a mobile coordinate system known as the body frame which is depicted by the ϕ, θ, and ψ angles with respect to the body center of gravity.

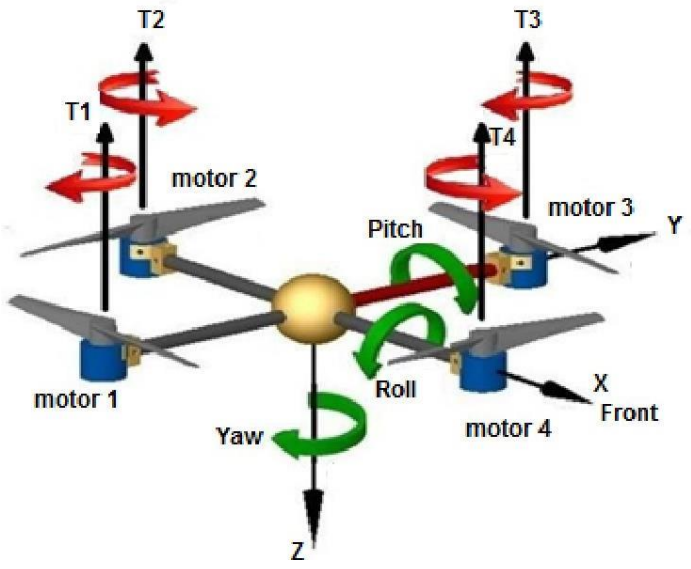

Figure 3.2: Quadcopter Dynamics [22]

The quadcopter is designed with four rotors in cross-configuration as seen in Figure 3.2. Two opposite rotors rotate in the same direction, the altitude and position of the quadcopter can be controlled by varying the rotor's angular speed. The quadcopter can be kept in a balanced position without spinning if the generated torque of the motors T1, T2, T3, and T4 are the same. Thrust controls the altitude of the quadcopter for ascending and descending and is achieved by increasing or reducing the rotational speed of motors 1, 2, 3 and 4 simultaneously. Roll is movement of the quadcopter by tilting either left or right to allow side movements. Pitch is movement of the quadcopter by tilting either front or back to allow forward or backward movement. Yaw is movement in a clockwise or anticlockwise manner while staying level to the ground to change the direction of the quadcopter. These flight motions can be achieved by controlling the rotational speeds of the four motors. Figure 3.3 illustrates different flight movements of a quadcopter due to variations in rotor speed.

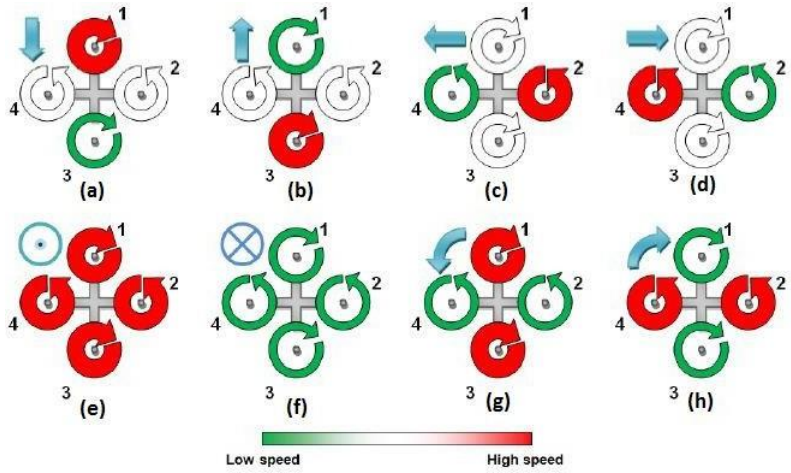

Figure 3.3: Flight movements of a Quadcopter

3.2 Quadcopter System Modelling

The quadcopter is an underactuated nonlinear system because it has four inputs and six outputs. The system is complex and in order to control it, the quadcopter is modelled on the following assumptions[1]:

i. The structure is rigid
ii. The structure is axis symmetrical
iii. The Centre of Gravity and the body fixed frame origin coincide
iv. The propellers are rigid
v. Thrust and drag are proportional to the square of the propeller's speed

3.2.1 Euler Angles

The Euler angles are three angles introduced by Leonhard Euler to describe the orientation of a rigid body in a coordinate system. They are also used to describe the relationship between two different reference frames and to convert the coordinates of a point in one

reference frame to coordinates of the same point in another reference frame. Euler angles are denoted as $\phi, \theta \; and \; \psi$ for the roll, pitch and yaw angles respectively and represent the rotations of a body about the axes of a coordinate system.

Any orientation of a rigid body can be achieved by combination of the three basic Euler angles. The rotation matrices are given by [5]:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} \tag{3.1}$$

$$R_y(\theta) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \tag{3.2}$$

$$R_z(\psi) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where $c(\phi) = cos(\phi)$, $s(\phi) = sin(\phi)$, $c(\theta) = cos(\theta)$, $s(\theta) = sin(\theta)$, $c(\psi) = cos(\psi)$, $s(\psi) = sin(\psi)$. The rotation matrix depicting the relationship between the inertial frame and the body frame is given as:

$$R = R_z(\psi) \times R_y(\theta) \times R_x(\phi) \tag{3.4}$$

$$R = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \tag{3.5}$$

R is a rotational matrix and is orthogonal such that $R^{-1} = R^T$.

3.2.2 Reference Frame Transformation

Assume $[x\ \ y\ \ z\ \ \phi\ \ \theta\ \ \psi]^T$ to be a vector of linear and angular positions in the inertial frame and let $[u\ \ v\ \ w\ \ p\ \ q\ \ r]^T$ be a vector of linear and angular velocities in the body frame.

Typically, the derivative of angular positions should give angular velocities but the angular positions and velocities above are in a different frame so we require some transformation matrix to convert from one reference frame to another.

$$Let\ \xi = [x\ \ y\ \ z]^T\ and\ \eta = [\phi\ \ \theta\ \ \psi]^T$$

$$Let\ v\ = [\dot{x}\ \ \dot{y}\ \ \dot{z}]^T\ and\ \omega_I = [\dot{\phi}\ \ \dot{\theta}\ \ \dot{\psi}]^T$$

$$Let\ v_B = [u\ \ v\ \ w]^T\ and\ \omega_B = [p\ \ q\ \ r]^T$$

From this statement, $v_I \neq v_B\ and\ \omega_I \neq \omega_B$, instead:

$$v_I = R \cdot v_B \tag{3.6}$$

$$\omega_I = W_\eta^{-1} \cdot \omega_B \tag{3.7}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.8}$$

Conversely,

$$\omega_B = W_\eta \cdot \omega_I \tag{3.9}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.10}$$

3.2.3 Rotational Motion

Assuming the quadcopter is a rigid body and using Euler's equations for rigid bodies, the dynamics equation in the body frame is given as:

$$I\dot{\omega}_B + [\omega_B \times (I\omega_B)] + \Gamma = \tau_B \tag{3.11}$$

We also assume that the quadcopter has a symmetric structure with the four arms aligned with the body x and y-axes. Therefore, the inertia matrix I is a diagonal matrix in which $I_{xx} = I_{yy}$.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.12}$$

The gyroscopic forces Γ are caused by the combined rotation of the four rotors and the quadcopter body such that:

$$\Gamma = \sum_{i=1}^{4} J_r(\omega_B \wedge \hat{e_3})(-1)^{i+1}\omega_r \quad {}_i \tag{3.13}$$

In matrix form,

$$\Gamma = J_r\omega_B\omega_r \qquad where\ \omega_r = -\omega_1 + \omega_2 - \omega_3 + \omega_4 \tag{3.14}$$

The external torque,

$$\tau_B = [\tau_\phi\ \tau_\theta\ \tau_\psi]^T \tag{3.15}$$

The roll torque component τ_ϕ, and the pitch torque component τ_θ, are obtained from standard mechanics where $i = 1$ and $i = 3$ motors are arbitrarily chosen to be on the roll- axis while $i = 2$ and $i = 4$ are arbitrarily chosen to be on the pitch-axis.

$$\tau_\phi = \Sigma r \times T = l(-k_t\omega_2^2 + k_t\omega_4^2) = lk_t(-\omega_2^2 + \omega_4^2) \quad (3.16)$$

$$\tau_\theta = \Sigma r \times T = l(-k_t\omega_1^2 + k_t\omega_3^2) = lk_t(-\omega_1^2 + \omega_3^2) \quad (3.17)$$

For the yaw-axis, the rotor axis is pointing in the z-direction in the body frame, the torque created around the rotor axis is given by:

$$\tau_\psi = (-1)^{i+1}k_b\omega^2 + I_m\dot{\omega}_i \quad (3.18)$$

where $(-1)^{i+1}$ is positive for the i^{th} propeller if the propeller is spinning clockwise and negative if it is spinning counter-clockwise. The term $I_m\dot{\omega}_i$ can be ignored because in steady state, $\dot{\omega}_i \approx 0$.

Therefore, the total torque about the z-axis is given by the sum of all the torques from each propeller:

$$\tau_\psi = k_b(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \quad (3.19)$$

Therefore, the torque matrix can be written as:

$$\tau_B = [\tau\theta] = \begin{bmatrix} \tau_\phi \\ \\ \tau_\psi \end{bmatrix} \begin{matrix} lk_t(-\omega_2^2 + \omega_4^2) \\ lk_t(-\omega_1^2 + \omega_3^2) \\ k_b(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{matrix} \quad (3.20)$$

where k_t is the thrust coefficient, k_b is the drag coefficient and l is the distance between rotor and the center of mass of the quadcopter.

3.2.4 Translational Motion

Using Newtonian equation to model the linear dynamics, the extraneous forces acting on the quadcopter is given as:

$$m\dot{v}_1 = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + RT_B + F_D \tag{3.21}$$

Drag force F_D, is the force acting on the system as a result of fluid friction (air resistance). A simplified equation form is used where friction is modelled as being proportional to the linear velocity in all directions.

$$F_D = -k_d v_I \tag{3.22}$$

The angular velocity of the i^{th} rotor creates a force F_i in the direction of the rotor axis (z-direction). The combined forces create thrust T in the direction of the body z-axis such that:

$$T = \sum_{i=1}^{4} F_i = k_t \sum_{i=1}^{4} \omega_i^2 \tag{3.23}$$

Since it acts in the z-axis:

$$T_B = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} = k_t \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} \omega_i^2 \end{bmatrix} \tag{3.24}$$

3.2.5 State Space Model

For the rotational dynamics discussed in section 3.4, equation 3.11 can be rewritten as:

$$\dot{\omega}_B = I^{-1}(-\omega_B \times (I\omega_B) - \Gamma + \tau_B) \tag{3.25}$$

$$\dot{\omega}_B = I^{-1}\left(\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_{xx}p \\ I_{yy}q \\ I_{zz}r \end{bmatrix} - J_r \begin{bmatrix} p \\ q \\ r \end{bmatrix} \omega_r + \tau_B\right) \tag{3.26}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{(Iyy - Izz)qr}{Ixx} \\ \frac{(Izz - Ixx)pr}{Iyy} \\ \frac{(Ixx - Iyy)pq}{Izz} \end{bmatrix} - J_r \begin{bmatrix} q \\ -p \\ 0 \end{bmatrix} \omega + \begin{bmatrix} \frac{\tau\phi}{Ix} \\ \frac{\tau\theta}{Iy} \\ \frac{\tau\psi}{Izz} \end{bmatrix} \quad (3.27)$$

For the linear dynamics discussed in section 3.5, equation 3.21 can be written out as:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \\ s(\psi)s(\theta)c(\phi) + c(\psi)s(\phi) \\ c(\theta)c(\phi) \end{bmatrix} - \frac{1}{m} \begin{bmatrix} k_{dx} & 0 & 0 \\ 0 & k_{dy} & 0 \\ 0 & 0 & k_{dz} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \quad (3.28)$$

3.3 Quadcopter System Control

A quadcopter has only four control inputs and primarily six outputs of interest $[x\ y\ z\ \phi\ \theta\ \psi]$ which makes it an underactuated system. This is resolved by separating it into two different control loops, with one working with attitude states and the other with position states. The quadcopter's angular motion is independent of the linear components, but the linear motion is determined by the angle variables. As a result, the goal is to control the attitude variable, which is independent of the linear motion, and then control the linear motion. It's possible to integrate the attitude control with the trajectory controller once it's designed and optimized. The control architecture to be implemented on the quadcopter system is shown in Figure 3.4.

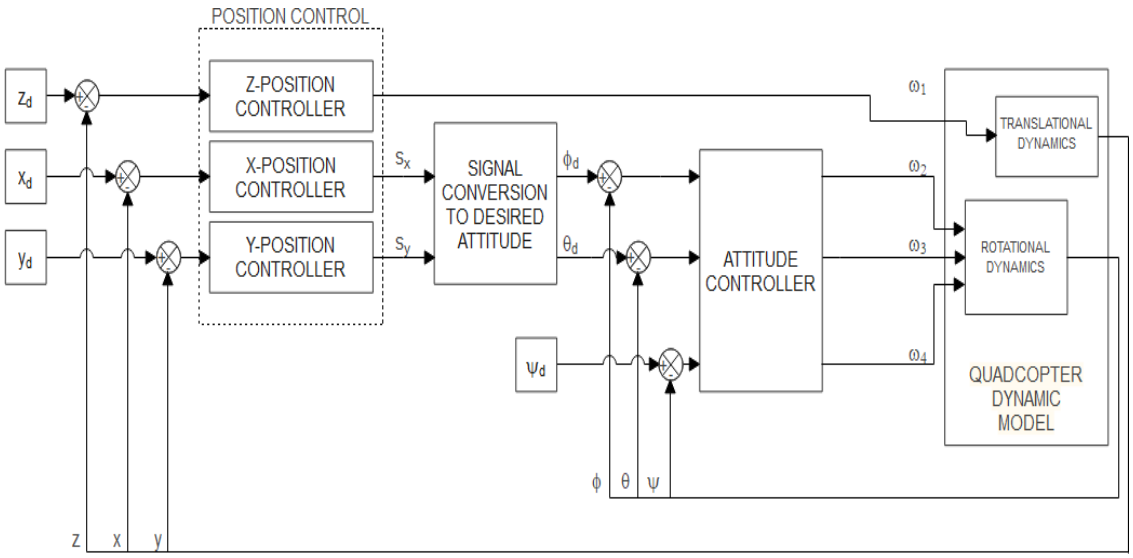

Figure 3.4: Control Architecture

3.4 PID Control

The general form of a PID controller is given as:

$$e(t) = r(t) - y(t) \tag{3.29}$$

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{d}{dt} e(t) \tag{3.30}$$

where u(t) represents the control input, r(t) represents the desired state and y(t) is the current or actual state. K_P, K_I and K_D are the gain parameters for the proportional, integral and derivative terms of the PID controller.

The proportional and derivative terms would be used for the quadcopter control. The torque generated is proportional to the angular velocities therefore we set the torques to be proportional to the controller output such that:

$$\tau = I \times u(t) \tag{3.31}$$

$$\begin{matrix} \phi \\ [\tau\theta \\ \tau_\psi \end{matrix} = \begin{vmatrix} \tau \\ ] \\ \end{vmatrix} \begin{matrix} \left[I_{xx}\left(k_{P\phi}(\phi_{des} - \phi) + k_{D\phi}(\dot{\phi}_{des} - \dot{\phi})\right)\right] \\ I_{yy}\left(k_{P\theta}(\theta_{des} - \theta) + k_{D\theta}(\dot{\theta}_{des} - \dot{\theta})\right) \\ \left[I_{zz}\left(k_{P\psi}(\psi_{des} - \psi) + k_{D\psi}(\dot{\psi}_{des} - \dot{\psi})\right)\right] \end{matrix} \tag{3.32}$$

The input to the quadcopter system is the angular velocity of the rotors. Recall equation 3.20 which relates the torque to the square of the angular velocity of the rotors, there are three equations but four unknowns. To allow simplification, the total thrust T which affects the acceleration in the z-direction is set to be equal to mg. This constraint is enforced to keep the quadcopter flying. Converting this thrust equation to the appropriate reference frame and utilizing a PD to minimize the error in the z-axis:

$$T = \left(g + k_{Pz}(z_{des} - z) + k_{Dz}(\dot{z}_{des} - \dot{z})\right) \frac{m}{c(\phi)c(\theta)} \tag{3.33}$$

Solving for the angular velocities of the rotor $w_1^2, w_2^2, w_3^2, and\ w_4^2$ by computing equation 3.32 and equating it to equation 3.20:

$$\begin{matrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ [w_4^2] \end{matrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ 0 & -lk_t & 0 & lk_t \\ -lk_t & 0 & lk_t & 0 \\ -k_b & k_b & -k_b & k_b \end{bmatrix}^{-1} \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{3.34}$$

Simplifying equation 3.34 would give the equations:

$$w_1^2 = \frac{T}{4k_t} - \frac{\tau_\theta}{2lk_t} - \frac{\tau_\psi}{4k_b} \tag{3.35}$$

$$w_2^2 = \frac{T}{4k_t} - \frac{\tau_\phi}{2lk_t} + \frac{\tau_\psi}{4k_b} \tag{3.36}$$

$$w_3^2 = \frac{T}{4k_t} + \frac{\tau_\theta}{2lk_t} - \frac{\tau_\psi}{4k_b} \tag{3.37}$$

$$w_4^2 = \frac{T}{4k_t} + \frac{\tau_\phi}{2lk_t} + \frac{\tau_\psi}{4k_b} \tag{3.38}$$

3.5 Feedback Linearization

Feedback linearization is a nonlinear control technique that has sparked a lot of interest in recently. The main concept is to mathematically convert nonlinear system dynamics into (fully or partially) linear ones, allowing linear control methods to be used. The goal is to design a controller such that it exactly cancels out the system dynamics.

From equation 3.28, we can replace $\ddot{x} = k_{ax}(\dot{x} - \dot{x}_d)$, $\dot{x} = k_{dx}k_x(x - x_d)$, where k_{ax}, k_{dx}, and k_x are gain values and replace $\ddot{y}, \dot{y}, \ddot{z}, and\ \dot{z}$ accordingly.

$$\frac{T}{m}\begin{bmatrix} c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \\ s(\psi)s(\theta)c(\phi) + c(\psi)s(\phi) \\ c(\theta)c(\phi) \end{bmatrix} = \begin{bmatrix} k_{ax}(\dot{x} - \dot{x}_d) \\ [k_{ay}(\dot{y} - \dot{y}_d)] \\ k_{az}(\dot{z} - \dot{z}_d) \end{bmatrix} + g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{1}{m}\begin{bmatrix} k_{dx}k_x(x - x_d) \\ [k_{dy}k_y(y - y_d)] \\ k_{dz}k_z(z - z_d) \end{bmatrix} \tag{3.39}$$

Expanding equation 3.39 above would give:

$$k_{ax}(\dot{x} - \dot{x}_d) + \frac{k_{dx}k_x}{m}(x - x_d) = \frac{T}{m}c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \tag{3.40}$$

$$k_{ay}(\dot{y} - \dot{y}_d) + \frac{k_{dy}k_y}{m}(y - y_d) = \frac{T}{m}s(\psi)s(\theta)c(\phi) + c(\psi)s(\phi) \tag{3.41}$$

$$k_{az}(\dot{z} - \dot{z}_d) + g + \frac{k_{dz}k_z}{m}(z - z_d) = \frac{T}{m}s(\psi)s(\theta)c(\phi) + c(\theta)c(\phi) \tag{3.42}$$

Taking the square of both sides and adding equation 3.40, 3.41 and 3.42 together to get:

$$\frac{T^2}{m^2} = k_{ax}{}^2(\dot{x} - \dot{x}_d)^2 + \frac{k_{dx}{}^2 k_x{}^2}{m}(x - x_d)^2 + \frac{2k_{ax}k_{dx}k_x}{m}(x - x_d)(\dot{x} - \dot{x}_d)$$

$$+k_{ay}{}^2(\dot{y} - \dot{y}_d)^2 + \frac{k_{dy}{}^2 k_y{}^2}{m}(y - y_d)^2 + \frac{2k_{ay}k_{dy}k_y}{m}(y - y_d)(\dot{y} - \dot{y}_d)$$

$$+k_{az}{}^2(\dot{z} - \dot{z}_d)^2 + \frac{k_{dz}{}^2 k_z{}^2}{m}(z - z_d) + g^2 + 2gk_{az}(\dot{z} - \dot{z}_d)$$

$$+\frac{2gk_{dz}k_z}{m}(z - z_d) + \frac{2k_{az}k_{dz}k_z}{m}(z - z_d)(\dot{z} - \dot{z}_d) \quad (3.43)$$

To obtain ϕ_d, from the x and y axes portion of equation 3.39, let:

$$F_x = c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \quad (3.44)$$

$$F_y = s(\psi)s(\theta)c(\phi) - c(\psi)s(\phi) \quad (3.45)$$

From equation 3.45:

$$s(\theta)c(\phi) = \frac{F_y + c(\psi)s(\phi)}{s(\psi)} \quad (3.46)$$

Substitute equation 3.46 into equation 3.44:

$$F_x = \frac{c(\psi)}{s(\psi)}(F_y + c(\psi)s(\phi)) + s(\psi)s(\phi) \quad (3.47)$$

$$s(\phi) = \frac{F_x - \frac{F_y}{t(\psi)}}{\frac{c(\psi) + s(\psi)t(\psi)}{t(\psi)}} \quad (3.48)$$

$$s(\phi) = \frac{F_x t(\psi) - F_y}{c(\psi) + s(\psi)t(\psi)} \quad (3.49)$$

$$\phi_d = sin^{-1}\left(\frac{F_x tan(\psi) - F_y}{cos(\psi) + sin(\psi) tan(\psi)}\right) \tag{3.50}$$

To obtain θ_d, from the x and z axes portion of equation 3.39, let:

$$F_z = c(\theta)c(\phi) \tag{3.51}$$

$$c(\phi) = \frac{F_z}{c(\theta)} \tag{3.52}$$

Substituting equation 3.49 and 3.52 into equation 3.44:

$$F_x = \frac{F_z c(\psi)s(\theta)}{c(\theta)} + \frac{F_x t(\psi)s(\psi) - F_y s(\psi)}{c(\psi) + s(\psi)t(\psi)} \tag{3.53}$$

$$F_z c(\psi)t(\theta) = F_x - \frac{F_x t(\psi)s(\psi) - F_y s(\psi)}{c(\psi) + s(\psi)t(\psi)} \tag{3.54}$$

$$t(\theta) = \frac{F_x}{F_z c(\psi)} - \frac{F_x t(\psi)s(\psi) - F_y s(\psi)}{F_z c^2(\psi) + F_z s^2(\psi)} \tag{3.55}$$

Recall:

$$cos^2 x + sin^2 x = 1 \tag{3.56}$$

$$t(\theta) = \frac{F_x}{F_z c(\psi)} - \frac{F_x t(\psi)s(\psi) - F_y s(\psi)}{F_z} \tag{3.57}$$

$$t(\theta) = \frac{F_x - F_x s^2(\psi) - F_y s(\psi)c(\psi)}{F_z\, c(\psi)} \tag{3.58}$$

$$t(\theta) = \frac{F_x c^2(\psi) - F_y s(\psi)c(\psi)}{F_z\, c(\psi)} \tag{3.59}$$

$$\theta_d = tan^{-1}\left(\frac{F_x cos^2(\psi) - F_y sin(\psi)cos(\psi)}{F_z cos(\psi)}\right) \tag{3.60}$$

For equation 3.50 and equation 3.60, F_x, F_y and F_z are given as:

$$F_x = \frac{(mk_{ax}(\dot{x} - \dot{x_d}) + k_{dx}k_x(x - x_d))}{T} \tag{3.61}$$

$$F_y = \frac{(mk_{ay}(\dot{y} - \dot{y_d}) + k_{dy}k_y(y - y_d))}{T} \tag{3.62}$$

$$F_z = \frac{(mk_{az}(\dot{z} - \dot{z_d}) + mg + k_{dz}k_z(z - z_d))}{T} \tag{3.63}$$

From equation 3.8 and 3.32 in terms of desired angles, we can replace $\phi_d = k_\phi(\phi - \phi_d)$, where k_ϕ is the gain values and replace θ_d and ψ_d accordingly:

$$\begin{matrix} p_d \\ [q_d] \\ r_d \end{matrix} = \begin{bmatrix} 1 & s(\phi_d)t(\theta_d) & c(\phi_d)t(\theta_d) \\ 0 & c(\phi_d) & -s(\phi_d) \\ 0 & \frac{s(\phi_d)}{c(\theta_d)} & \frac{c(\phi_d)}{c(\theta_d)} \end{bmatrix}^{-1} \begin{matrix} k_\phi(\phi - \phi_d) \\ [\, k_\theta(\theta - \theta_d)\,] \\ k_\psi(\dot{\psi} - \dot{\psi}_d) \end{matrix} \tag{3.64}$$

$$\begin{matrix} \tau_\phi \\ [\tau\theta\,] \\ \tau_\psi \end{matrix} = \begin{matrix} I_{xx}k_{P\phi}(p - p_d) \\ [I_{yy}k_{P\theta}(q - q_d)] \\ I_{zz}k_{P\psi}(r - r_d) \end{matrix} - \begin{matrix} (I_{yy} - I_{zz})q_d r_d + J_r q_d \omega_r \\ [(I_{zz} - I_{xx})p_d r_d + J_r p_d \omega_r\,] \\ (I_{xx} - I_{yy})p_d q_d \end{matrix} \tag{3.65}$$

The angular velocities w_1, w_2, w_3 and w_4 can be obtained using equation 3.34.

3.6 Trajectory Tracking

The goal of trajectory control is to take the quadcopter system from its current position to the desired position by regulating the quadcopter's rotor angular velocities. Due to its complex dynamics, finding the best quadcopter trajectory is a huge challenge. To relate the desired position x and y which are not controllable to the desired angles of ϕ and θ that are controllable, we make two assumptions:

1) Small angle approximation such that $\sin(x) \approx x$ and $\cos(x) \approx 1$.
2) The desired ψ angle is zero.

From the above assumptions, equation 3.28 can be simplified as:

$$\ddot{x}_{des} = \frac{T}{m}\theta_{des} - \frac{k_{dx}}{m}\dot{x}_{des} \tag{3.66}$$

$$\ddot{y}_{des} = \frac{T}{m}\phi_{des} - \frac{k_{dy}}{m}\dot{y}_{des} \tag{3.67}$$

$$\ddot{z}_{des} = -g + \frac{T}{m} - \frac{k_{dz}}{m}\dot{z}_{des} \tag{3.68}$$

Equations 3.39, 3.40 and 3.41 can be rewritten as:

$$\theta_{des} = \frac{1}{T}[m\ddot{x}_{des} + k_{dx}\dot{x}_{des} + k_{Pxy}(x_{des} - x)] \tag{3.69}$$

$$\phi_{des} = -\frac{1}{T}[m\ddot{y}_{des} + k_{dy}\dot{y}_{des} + k_{Pxy}(y_{des} - y)] \tag{3.70}$$

$$T_{des} = m(\ddot{z}_{des} + g) + k_{dz}\dot{z}_{des} \tag{3.71}$$

θ_{des}, ϕ_{des} and T_{des} represent the desired values of pitch angle, roll angle and thrust.

3.7 Conclusion

In this chapter, the orientation of the quadcopter in terms of two coordinate systems was discussed. The quadcopter system was also modelled based on a few established assumptions using the Euler and Newton's equations while considering the extraneous forces influencing quadcopter system. The control actions for the PD controller were derived and the complete equations for the full state FBL using dynamic inversion were derived and computed without small angle approximation. Trajectory tracking was also discussed and equations for mapping the x and y position coordinates to the desired angle values were derived.

CHAPTER FOUR

SIMULATION AND RESULTS

4.1 Introduction

In this chapter, the derived model for the quadcopter system will be simulated and the control techniques discussed in the previous chapter would be implemented on the model and simulated. The results would be analyzed and compared to determine the most effective control technique for stabilization and trajectory tracking.

4.2 Simulation

The dynamic model and controllers are implemented in MATLAB 2021 for simulation with MATLAB programming language. The complete MATLAB code for the model, attitude stabilization and trajectory tracking are shown in Appendix A.

Table 4.1: Quadcopter parameter values for simulation

Parameter	Value	Unit
g	9.81	m/s^2
m	0.468	kg
l	0.225	m
J_r	3.357×10^{-5}	$kg\ m^2$
k_t	2.980×10^{-6}	
k_b	1.140×10^{-7}	
I_{xx}	4.856×10^{-3}	$kg\ m^2$

I_{yy}	4.856×10^{-3}	kg m^2
I_{zz}	8.801×10^{-3}	kg m^2
k_{dx}	0.25	
k_{dy}	0.25	
k_{dz}	0.25	

The quadcopter model is simulated with parameter values as shown in Table 4.1 [8]. The initial conditions assigned to the system for simulation is given in Table 4.2.

Table 4.2: Initial conditions for simulation

State	**Value**	**State**	**Value**
x	-1	ϕ	10
y	2	θ	-10
z	1	ψ	5

The simulation progresses at 0.001 second intervals to a total time of 15 seconds. The control inputs (angular velocities of the rotors) are shown in Figure 4.1, the positions x, y, z and the angles ϕ, θ *and* ψ in Figure 4.2.

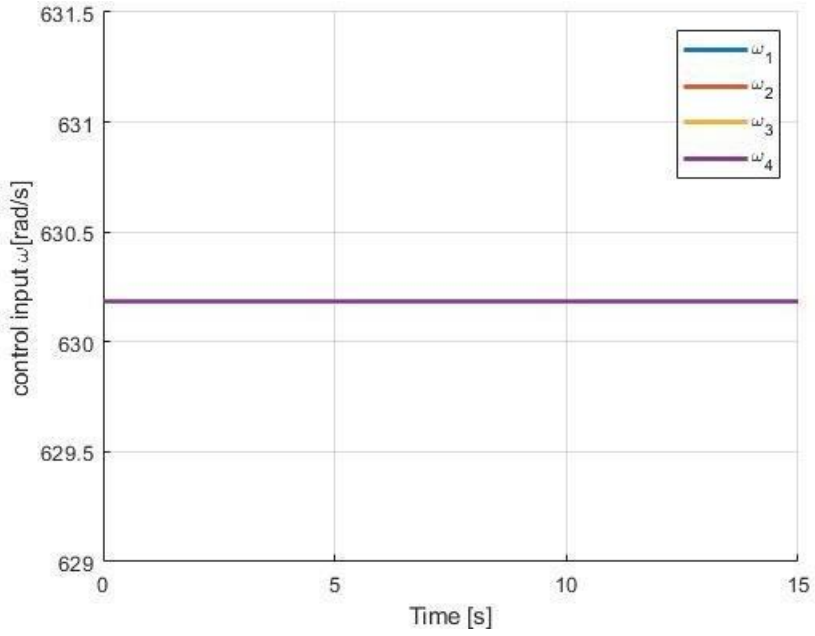

Figure 4.1: Control input

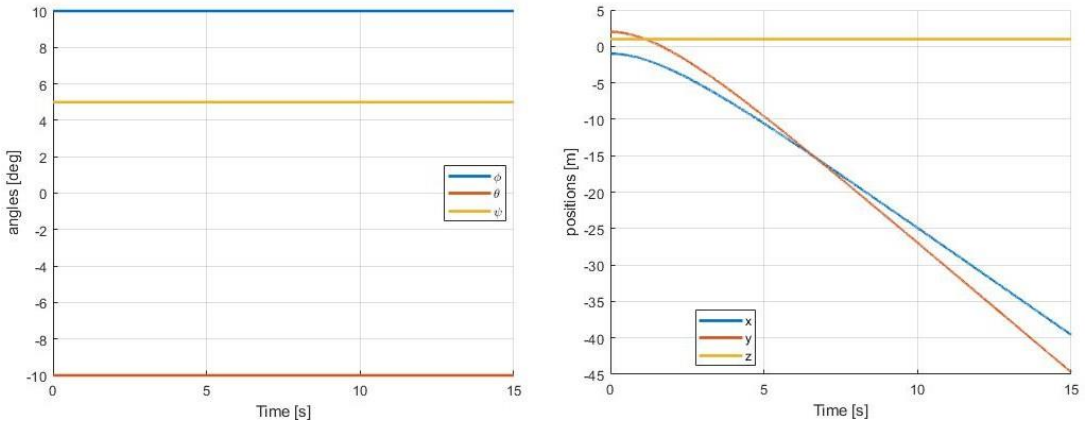

Figure 4.2: Position and angle variables against time

The control input remains constant and hence the angles remain the same but the positions become unstable.

4.3 Attitude Stabilization

A PD controller is implemented on the nonlinear system to stabilize and drive the system states to zero. The initial conditions assigned for the system remain the same as shown in Table 4.2. The control gain values for the PD controller are determined by manual tuning and are shown in Table 4.3.

Table 4.3: Gain values for PD controller

Parameter	Value	Parameter	Value
$k_{P\phi}$	1.5	$k_{D\phi}$	2.6
$k_{P\theta}$	1.5	$k_{D\theta}$	2.6
$k_{P\psi}$	1.5	$k_{D\psi}$	2.6
k_{Pz}	6	k_{Dz}	1.5

The simulation progresses at 0.001 second intervals to a total time of 15 seconds. The angles ϕ, $\theta\ and\ \psi$ are shown in Figure 4.3, the angular velocities of the rotors are shown in Figure 4.4, and the positions $x, y, and\ z$ in Figure 4.5. The angles are stabilized to zero after 10 seconds and the position z is stabilized to zero after 5 seconds. The x and y positions do not stabilize to zero because the states are not observable.

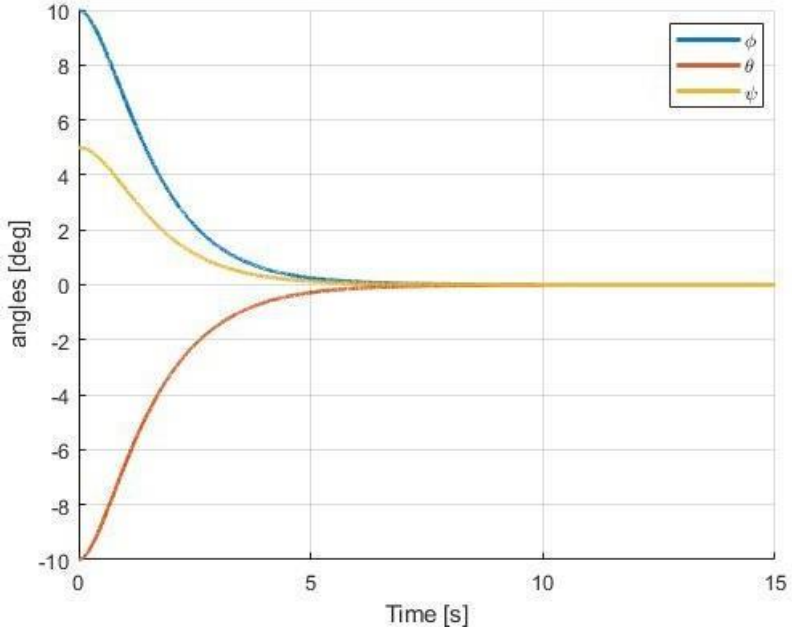

Figure 4.3: Angles ϕ, θ and ψ

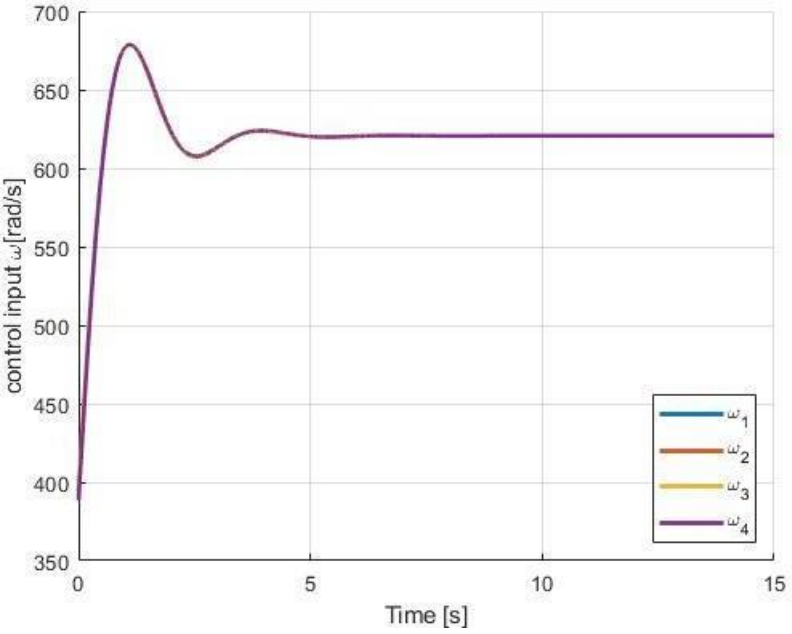

Figure 4.4: Control input

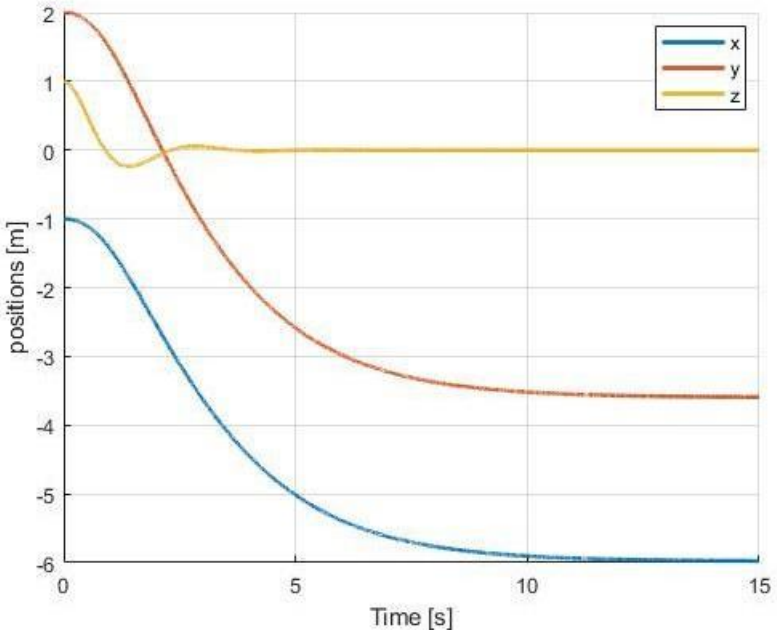

Figure 4.5: Positions x, y and z

Using the concept of trajectory tracking, an attempt can be made to drive the x and y positions to zero. This can be done by mapping the zero desired states of x and y to the desired ϕ and θ states which are controllable as shown in equation 3.42, 3.43 and 3.44. An additional proportional controller k_{Pxy}, with a control gain of 0.04 is implemented to drive the system states to zero and the simulation advances at 0.001 second intervals to a total time of 45 seconds. The angular velocities of the four rotors and the angles $\phi, \theta \; and \; \psi$ are shown in Figure 4.6, the modified positions x, y and z are shown in Figure 4.7.

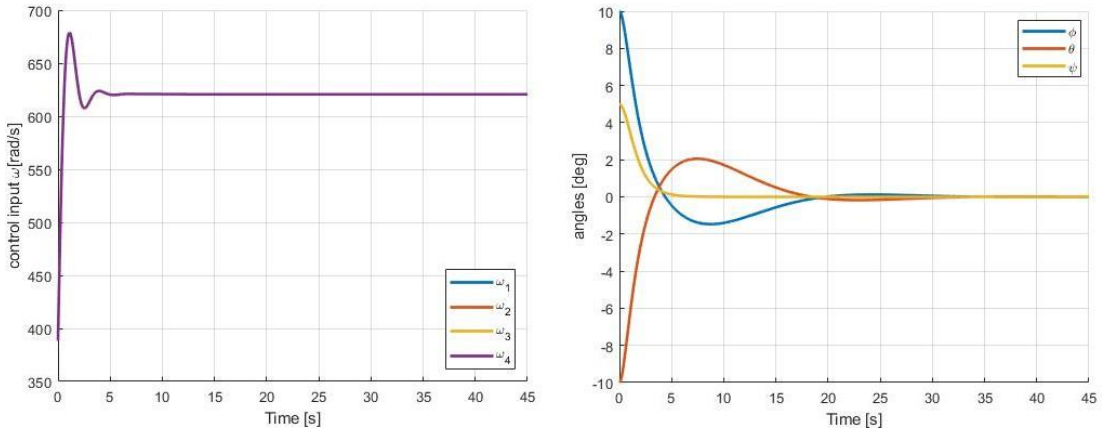

Figure 4.6: Control input and angles ϕ, θ and ψ

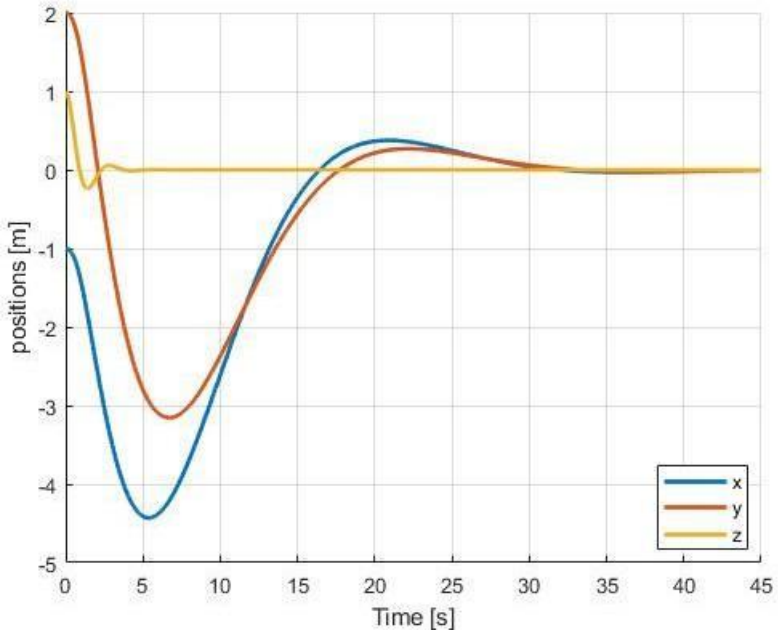

Figure 4.7: Positions x, y and z

The angles ϕ, θ and ψ are stabilized to zero after 35 seconds and the positions x, y and z are stabilized to zero after 45 seconds.

4.4 Trajectory Tracking

Two controls for trajectory tracking are considered and simulated for two different trajectories. The first control strategy is implemented with a PD controller and the second strategy is a PD controller integrated with the feedback linearized system. The total time assigned for the simulation in both cases is 45 seconds.

4.4.1 Trajectory Tracking with PD Controller

The comparison between the desired and actual trajectory for the first path is shown in Figure 4.8. Figure 4.9 shows the graph of the position variables against time.

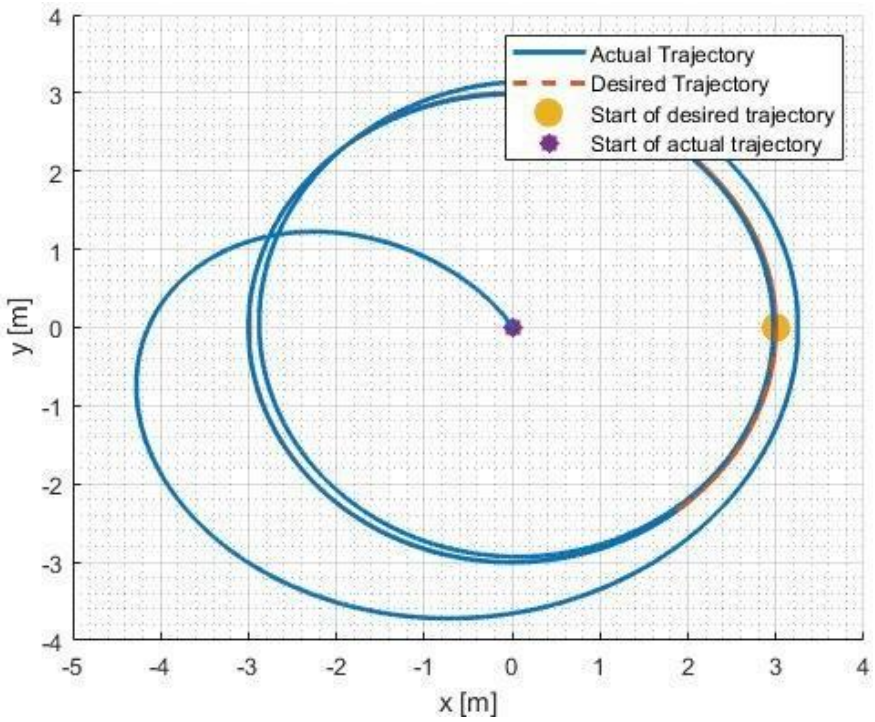

Figure 4.8: Desired and actual trajectory

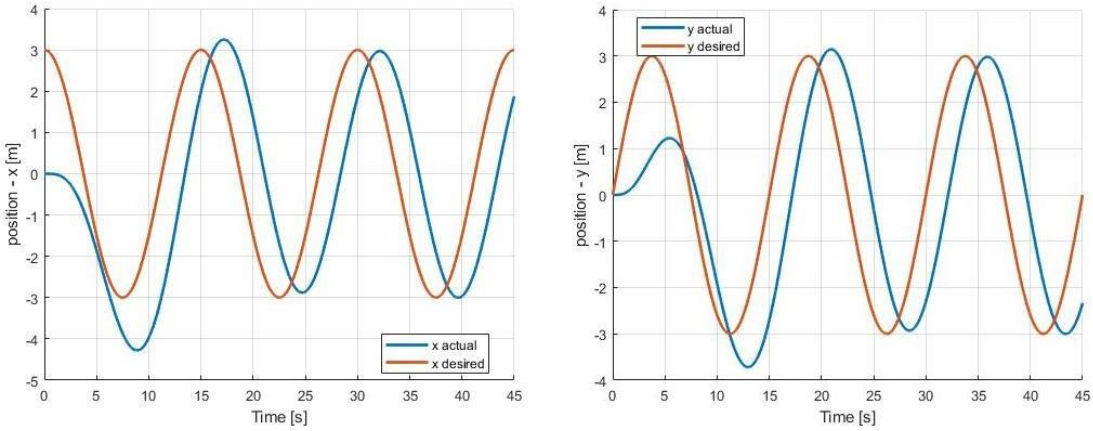

Figure 4.9: Coordinate wise comparison between desired and actual trajectory

Another trajectory path is simulated and the result is shown in Figure 4.10. The graph of the position variables against time for this path is shown in Figure 4.11.

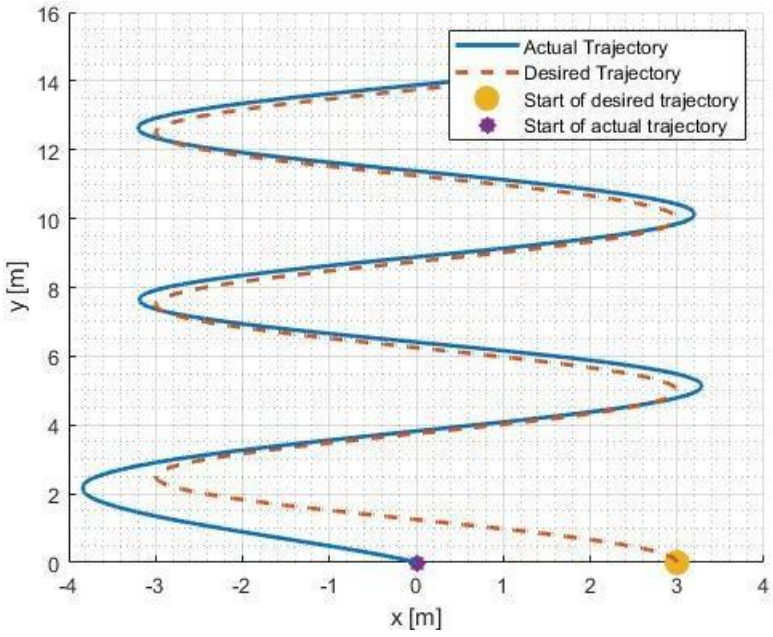

Figure 4.10: Desired and actual trajectory

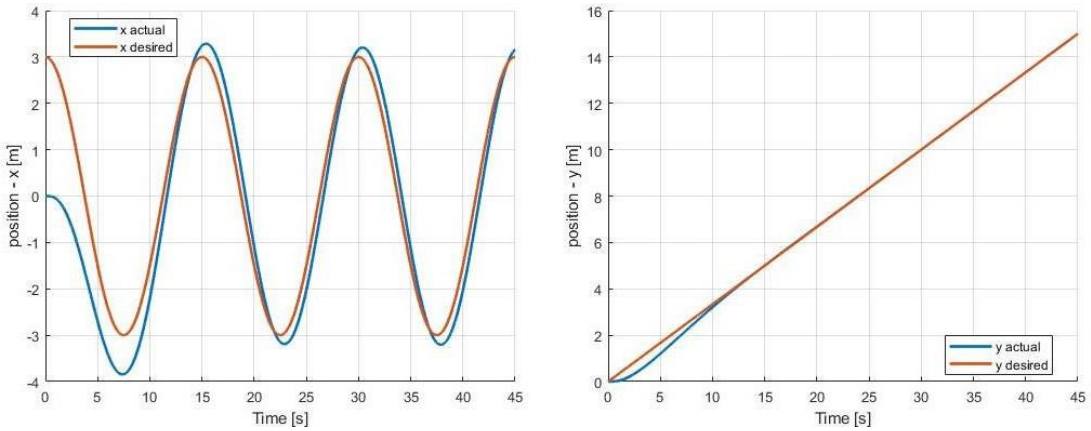

Figure 4.11: Coordinate wise comparison between desired and actual trajectory

A similar result as seen in Figure 4.9 is also obtained; the quadcopter is unable to follow the desired trajectory with precision.

4.4.2 Trajectory Tracking with Feedback Linearization and PD Controller

Feedback linearization technique is implemented on the quadcopter model and simulated with a PD controller. The desired trajectory and the actual trajectory for the first path is shown in Figure 4.12. Figure 4.13 shows the graph of the position variables against time.

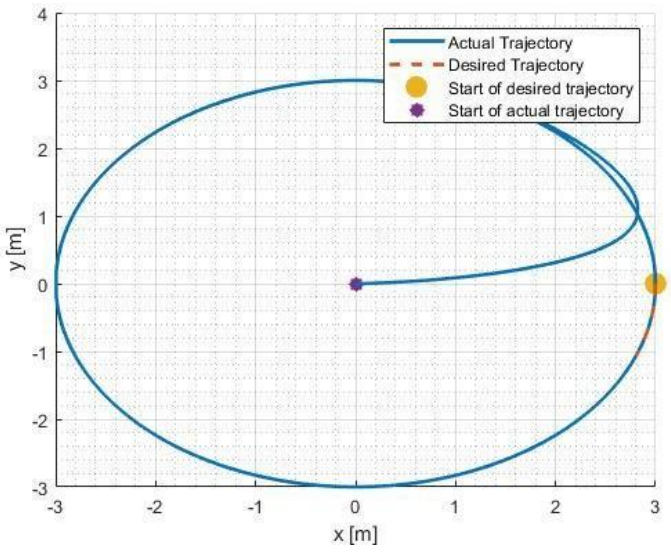

Figure 4.12: Desired and actual trajectory

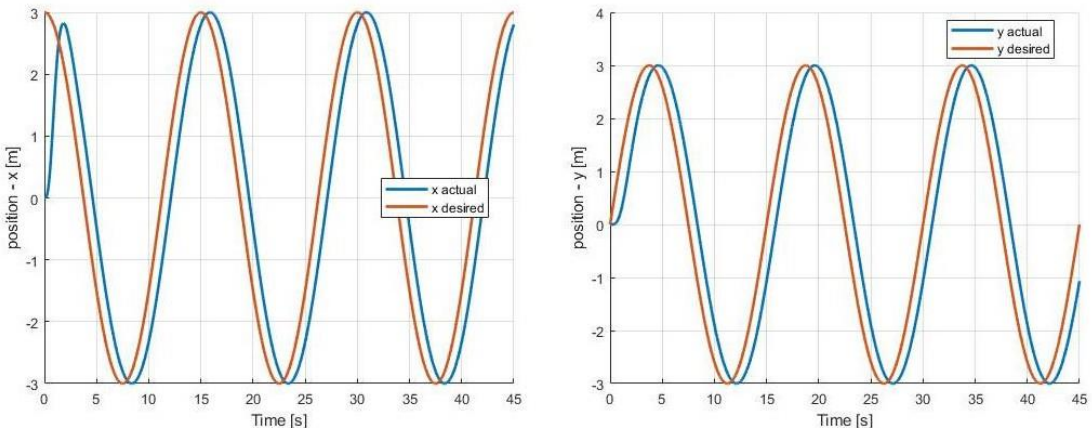

Figure 4.13: Coordinate wise comparison between desired and actual trajectory

Another trajectory path is simulated and the result is shown in Figure 4.14. A similar result as seen in Figure 4.12 is also obtained; the quadcopter is able to lock on to the desired trajectory with more accuracy with this control.

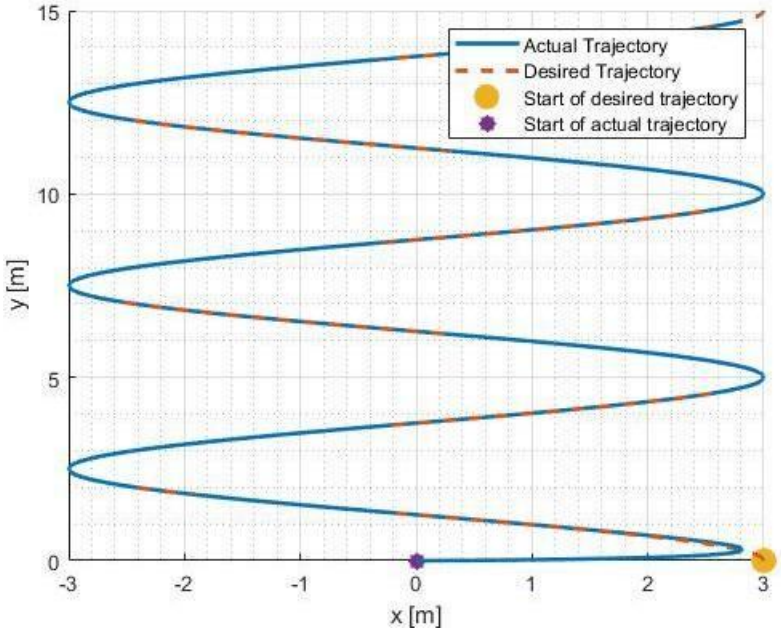

Figure 4.14: Desired and actual trajectory

4.5 Result Analysis

A detailed analysis on the performance of the PD controller for stabilization and its integration with the FBL system is required to determine the most suitable control strategy.

4.5.1 Attitude Stabilization

As discussed in Section 4.1, the PD controller with control gains shown in Table 4.3 was able to stabilize the angles to the desired zero state after 10 seconds and also the drive the z position variable to zero as well in 5 seconds. The x and y positions are eventually stabilized but they do not stabilize to the desired zero state. By mapping the desired states of x and y to the desired ϕ and θ states which are controllable, an additional proportional controller is used to drive all the system states to zero. However, this approach is not efficient because it takes too long for the angles and position to stabilize. The angles ϕ, θ and ψ stabilize to zero after 35 seconds and the positions x, y and z stabilize to zero after 45 seconds. The ϕ, θ, x and y states experience overshoots before stabilizing to zero. A side-by-side comparison of the angle and position variables of the two approaches are shown in Figure 4.15 and 4.16.

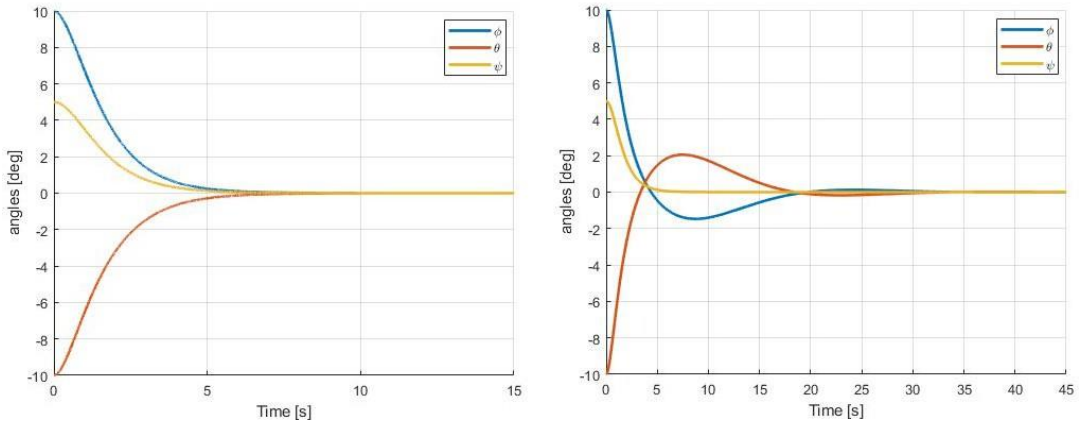

Figure 4.15: Angles ϕ, θ and ψ

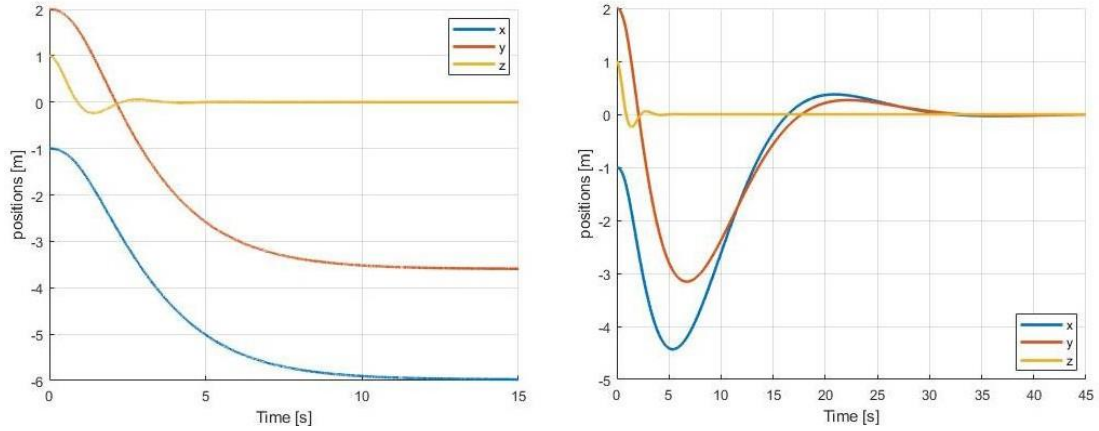

Figure 4.16: Positions x, y and z

4.5.2 Trajectory Tracking

The results of the two control strategies implemented for trajectory tracking were discussed in Section 4.2. A better performance is observed in the feedback linearized system with PD control. The quadcopter is able to lock on to the desired trajectory and follow the path with accuracy in a lesser time period. Figure 4.17, 4.18 and 4.19 show the plot of the desired and actual trajectory after 15, 30 and 45 seconds respectively for both the PD control and the FBL with PD control.

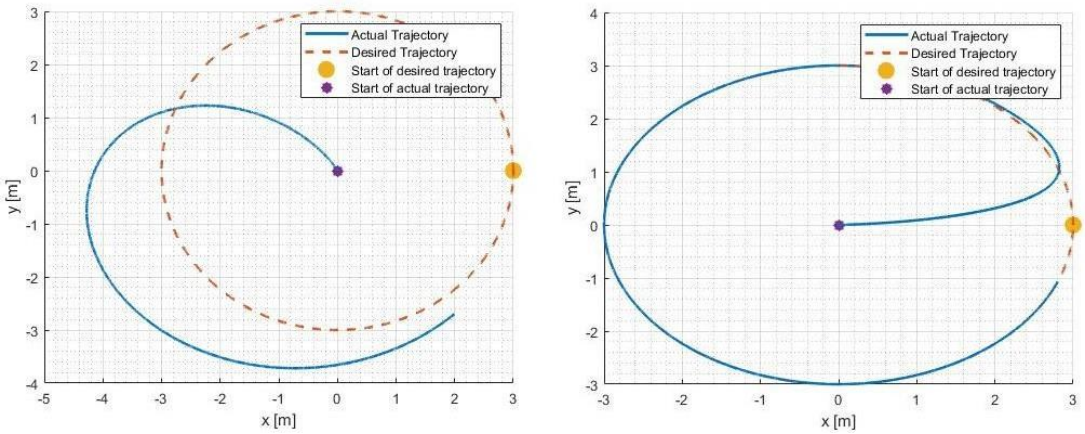

Figure 4.17: Trajectory for PD (left) and FBL with PD control (right) after 15 seconds

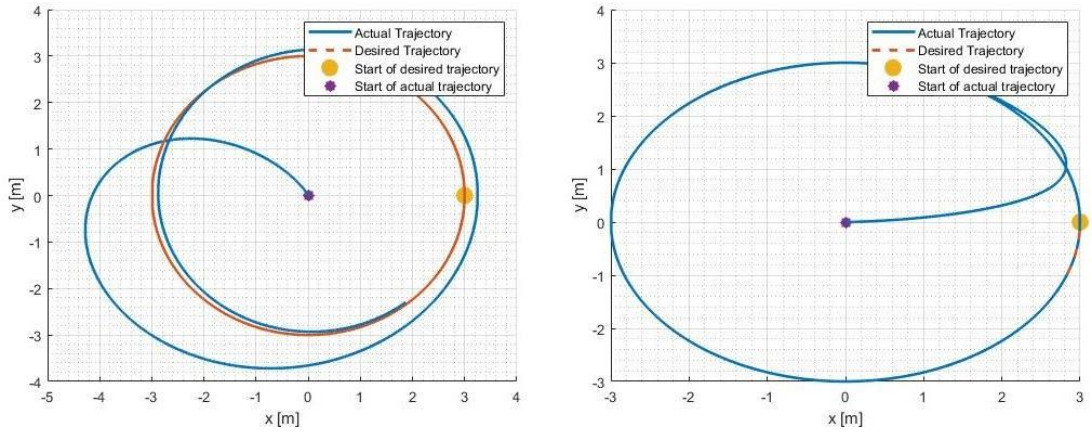

Figure 4.18: Trajectory for PD (left) and FBL with PD control (right) after 30 seconds

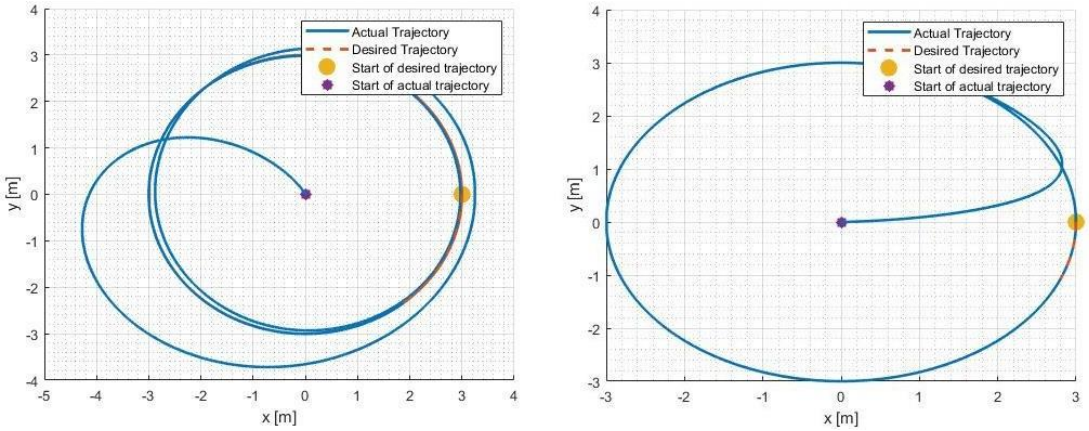

Figure 4.19: Trajectory for PD (left) and FBL with PD control (right) after 45 seconds

The quadcopter system simulated with the FBL and PD control is able to lock on to the desired trajectory after the first 15 seconds while the system with only the PD control is unable to do that properly even after 30 seconds. This shows that the feedback linearization technique with the PD control is more effective in trajectory tracking than the regular PD control.

CHAPTER FIVE

CONCLUSION

This chapter gives an overview of all the work done in previous chapters of this thesis, a review of contributions and achievements as well as the challenges encountered in the project. A few recommendations are also proposed for future development of this project towards optimal control of quadcopter systems.

The number and complexity of applications for quadcopter systems continues to grow on a daily basis, the control techniques that are employed must also improve in order to enhance performance and adaptability to new situations. Based on this conviction, this thesis studied modelling and nonlinear control of a quadcopter for stability and trajectory control. Understanding the dynamics of the quadcopter was a continuous challenge and so system modelling, simulation and analysis were explored. This provides proper understanding not only for the complex design of the system, but also for the selection of quadcopter control methods.

APPENDICES

APPENDIX A

Quadcopter Control

```
%% Notations
% xi = [x, y, z]
% nu_I = [x_dot, y_dot, z_dot] %inertial frame
% nu_B = [u, v, w] %body frame

% eta = [phi, theta, psi]
% omega_i = [phi_dot, theta_dot, psi_dot] % inertia frame
```

```
% omega_b = [p, q, r] % body frame
%% Prepare workspace clear;
close all; clc;
% Select Control Type
% 1 - stabilization, 2 - stabilization with modifications for x and y  implement_control = 1;
% Select 0 to simulate without control and 1 to simulate with control type 1
% recall x and y are not observable so further modifications have to be made
% 3 - trajectory tracking
% 4 - Feedback Linearization.
param.control_type = 4;

% select trajectory type
% 1 - Circular 2- Spiral
param.traj = 1;

% Time change
% 1 - 1 * T, 2 - 2 * T, 3 - 3 * T
param.time_change = 3;
```

```
%% Quadcopter Variables
param.g = 9.81;
param.m = 0.468;
param.l = 0.225; param.Jr
= 3.357e-5; param.Kt =
2.98e-6; param.Kb =
1.140e-7;
Kdx = 0.25; Kdy = 0.25; Kdz = 0.25;
param.Kd = diag([Kdx; Kdy; Kdz]);
Ixx = 4.856e-3; Iyy = 4.856e-3; Izz = 8.801e-3;
param.I = diag([Ixx; Iyy; Izz]);

%% Control Variable
% Nonlinear PD Gains
% Kd_phi = Kd_theta = Kd_psi = Kd
% Kp_phi = Kp_theta = Kp_psi = Kp if
param.control_type == 1
    if implement_control == 0 PD_var.Kd
        = 0; PD_var.Kp = 0;
        PD_var.Kd_z = 0; PD_var.Kp_z = 0; else
        PD_var.Kd = 2.6; PD_var.Kp = 1.5;
        PD_var.Kd_z = 1.5; PD_var.Kp_z = 6;
        PD_var.Kp_xy = 0.04;
    end
else
    PD_var.Kd = 2.6; PD_var.Kp = 1.5;
    PD_var.Kd_z = 1.5; PD_var.Kp_z = 6;
    PD_var.Kp_xy = 0.04; end
% For Nonlinear PD with helix trajectory
```

```
if param.control_type == 3 && param.traj == 2
    PD_var.Kd = 5; PD_var.Kp = 15;
    PD_var.Kd_z = 1.5; PD_var.Kp_z = 1;
    PD_var.Kp_xy = 0.05; end
% Feedback controller gains
PD_var.FB = [-5.5 * ones(3,1);  -2 * ones(3,1)
            -5.2 * ones(3,1); -10 * ones(3,1)];
%% Initialize Simulation
start_time = 0;
T = 15; % in seconds
if param.control_type == 1
    end_time = T;
else
    end_time = param.time_change * T; end
dt = 0.001;
sim_time = (start_time:dt:end_time)';
n_sim = length(sim_time);

xi =  zeros(n_sim, 3); xi_desired =
zeros(n_sim, 3); nu_I =
zeros(n_sim, 3);
eta = zeros(n_sim, 3);
omega_I = zeros(n_sim, 3);
control_values = zeros(n_sim, 4);

% intitial states
if param.control_type == 1 || param.control_type == 2
    % since it is a stabilization control, we give it an initial condition xi(1, :) =
    [-1, 2, 1]; % position [m]
```

```
    eta(1, :) = deg2rad([10, -10, 5]); % angle [deg] end
%% Simulation
for idx_t = 1:n_sim
    % extract current time step data
    xi_current = xi(idx_t, :)'; nu_I_current
    = nu_I(idx_t, :)'; eta_current =
    eta(idx_t, :)';
    omega_I_current = omega_I(idx_t, :)';

    % Obtain control input from controller
    if param.control_type == 1 || param.control_type == 2
        % desired position = 0
        xi_des = zeros(1, 3);
        xi_des_d = zeros(1, 3);
        xi_des_dd = zeros(1, 3);
    elseif param.control_type == 3 || param.control_type == 4
        % desired position is a circular or spiral trajectory if
        param.traj == 1
            [xi_des, xi_des_d, xi_des_dd] = circular_traj(sim_time(idx_t), T); elseif
        param.traj == 2
            [xi_des, xi_des_d, xi_des_dd] = spiral_traj(sim_time(idx_t), T); end
    end

    % convert from inertia frame to body frame
    omega_B_current = inertia2body(omega_I_current, eta_current);

    % Select control algorithm if
    param.control_type == 4
```

```
        control_input = Feedback_Lin(omega_B_current, eta_current, xi_current, nu_I_current,
param, PD_var, xi_des);
    else
        control_input = PD(omega_I_current, eta_current, xi_current, nu_I_current, param,
PD_var, xi_des, xi_des_d, xi_des_dd); % omega square end


    nu_I_dot = linear_acceleration(control_input, eta_current, nu_I_current, param); omega_B_dot =
    rotational_acceleration(control_input, omega_B_current, param);

    % linear motion evolution using discretized Euler
    nu_I_next = nu_I_current + dt * nu_I_dot; xi_next =
    xi_current + dt * nu_I_next;

    % rotational motion evolution using discretized Euler
    omega_B_next = omega_B_current + dt * omega_B_dot;
    % convert from bodyframe to inertia frame
    omega_I_next = body2inertia(omega_B_next, eta_current);
    eta_next = eta_current + dt * omega_I_next;

    % fill in next time step data xi(idx_t +
    1, :) = xi_next'; nu_I(idx_t + 1, :) =
    nu_I_next'; eta(idx_t + 1, :) =
    eta_next';
    omega_I(idx_t + 1, :) = omega_I_next';
    control_values(idx_t, :) = sqrt(control_input)';

    if param.control_type == 3 || param.control_type == 4 xi_desired(idx_t, :)
        = xi_des';
    end
```

```
end

%% Plots
figure
hold on
plot(sim_time, control_values(:, 1), 'LineWidth', 2)
plot(sim_time, control_values(:, 2), 'LineWidth', 2)
plot(sim_time, control_values(:, 3), 'LineWidth', 2)
plot(sim_time, control_values(:, 4), 'LineWidth', 2) hold
off
xlabel('Time [s]')
ylabel('control input \omega[rad/s]')
legend('\omega_1', '\omega_2', '\omega_3', '\omega_4', 'Location', 'Best') grid on

if param.control_type == 1 || param.control_type == 2 figure
    hold on
    plot(sim_time, xi(1:end-1, 1), 'LineWidth', 2)
    plot(sim_time, xi(1:end-1, 2), 'LineWidth', 2)
    plot(sim_time, xi(1:end-1, 3), 'LineWidth', 2) hold off
    xlabel('Time [s]')
    ylabel('positions [m]')
    legend('x', 'y', 'z', 'Location', 'Best')
    grid on

    figure
    hold on
    plot(sim_time, rad2deg(eta(1:end-1, 1)), 'LineWidth', 2)
    plot(sim_time, rad2deg(eta(1:end-1, 2)), 'LineWidth', 2)
```

```
    plot(sim_time, rad2deg(eta(1:end-1, 3)), 'LineWidth', 2) hold off
    xlabel('Time [s]')
    ylabel('angles [deg]')
    legend('\phi', '\theta', '\psi', 'Location', 'Best') grid on

elseif param.control_type == 3 || param.control_type == 4 figure
    hold on
    plot(sim_time, xi(1:end-1, 1), 'LineWidth', 2)
    plot(sim_time, xi_desired(:, 1), 'LineWidth', 2) hold
    off
    xlabel('Time [s]')
    ylabel('position - x [m]')
    legend('x actual', 'x desired', 'Location', 'Best') grid on

    figure
    hold on
    plot(sim_time, xi(1:end-1, 2), 'LineWidth', 2)
    plot(sim_time, xi_desired(:, 2), 'LineWidth', 2) hold
    off
    xlabel('Time [s]')
    ylabel('position - y [m]')
    legend('y actual', 'y desired', 'Location', 'Best') grid on


    figure
    hold on
```

```
    plot3(xi(1:end-1, 1), xi(1:end-1, 2), xi(1:end-1, 3), 'LineWidth', 2)
    plot3(xi_desired(1:end-1, 1), xi_desired(1:end-1, 2), xi_desired(1:end-1, 3),'--',
'LineWidth', 2)
    plot3(xi_desired(1,1), xi_desired(1,2), xi_desired(1,3),'o', 'LineWidth', 6,'MarkerSize',
6)
    plot3(xi(1,1), xi(1,2), xi(1,3),'*', 'LineWidth', 6,'MarkerSize', 6)

    xlabel('x [m]')
    ylabel('y [m]')
    zlabel('z [m]')
    grid on
    grid minor
    legend('Actual Trajectory','Desired Trajectory','Start of desired trajectory','Start of actual
trajectory')
end

%% Functions
function nu_I_dot = linear_acceleration(control_input, eta, nu_I, param)
    % Compute translational/linear motion using Newtonian method
    % gravity
    nu_I_dot_1 = -param.g * [0, 0, 1]'; R
    = rotation(eta);
    % total thrust
    T_B = [0, 0, param.Kt * sum(control_input)]';
    nu_I_dot_2 =  (1 / param.m) * R * T_B;
    % drag force
    nu_I_dot_3 = -(1 / param.m) * param.Kd * nu_I;

    nu_I_dot = nu_I_dot_1 + nu_I_dot_2 + nu_I_dot_3; end
```

```
function omega_B_dot = rotational_acceleration(control_input, omega_B, param)
    % Compute rotational motion using Euler method tau =
    torques(control_input, param);
    gamma = computeGamma(control_input, omega_B, param);
    % using direct representation for ease of computation
    omega_B_dot = inv(param.I) * (tau - gamma + cross(omega_B, param.I * omega_B));
end

function gamma = computeGamma(control_input, omega_B, param)
    % compute the gyroscopic forces causes by combined rotation of motors omega
    = sqrt(control_input);
    omega_gamma = -omega(1) + omega(2) - omega(3) + omega(4); gamma =
    param.Jr * cross(omega_B, [0; 0; 1]) * omega_gamma;
end

% Compute torques, given current inputs, length, drag coefficient, and thrust coefficient. function
tau = torques(control_input, param)
    % Inputs are values for ${\omega_i}^2$ tau
    = [
        param.l * param.Kt * (-control_input(2) + control_input(4)) param.l
        * param.Kt * (-control_input(1) + control_input(3))
        param.Kb * (-control_input(1) + control_input(2) - control_input(3) + control_input(4))
    ];
 end
function R = rotation(eta)
    % Compute rotational matrix
    phi = eta(1); theta = eta(2); psi = eta(3);

    R11 = cos(psi) * cos(theta);
```

```
    R12 = cos(psi) * sin(theta) * sin(phi) - sin(psi) * cos(phi); R13 =
    cos(psi) * sin(theta) * cos(phi) + sin(psi) * sin(phi);

    R21 = sin(psi) * cos(theta);
    R22 = sin(psi) * sin(theta) * sin(phi) + cos(psi) * cos(phi); R23 =
    sin(psi) * sin(theta) * cos(phi) - cos(psi) * sin(phi);

    R31 = -sin(theta);
    R32 = cos(theta) * sin(phi);
    R33 = cos(theta) * cos(phi);

    R = [R11 R12 R13;...
        R21 R22 R23;...
        R31 R32 R33];
end

function I2B = inertia2body(omega_I, eta) phi
    = eta(1); theta = eta(2); psi = eta(3); W = [1
    0       -sin(theta);...
        0  cos(phi) cos(theta) * sin(phi);...  0  -
        sin(phi)  cos(theta) * cos(phi)];
    I2B  =  W  *  omega_I;
end

function B2I = body2inertia(omega_B, eta) phi
    = eta(1); theta = eta(2); psi = eta(3);
    W_inv = [1    sin(phi) * tan(theta)         cos(phi) * tan(theta);...
            0    cos(phi)                  -sin(phi);...
            0   sin(phi) / cos(theta)        cos(phi) / cos(theta)];
    B2I = W_inv * omega_B;
end
```

```
%% Controls
function ctrl = PD(omega_I, eta, xi, nu_I, param, PD_var, xi_des, xi_des_d, xi_des_dd) ctrl =
    zeros(4, 1);
    % Unload parameters
    phi = eta(1); theta = eta(2); psi = eta(3);
    phi_d = omega_I(1); theta_d = omega_I(2); psi_d = omega_I(3); z =
    xi(3); z_d = nu_I(3);

    % desired position computation - x and y desired translated to phi and theta z_des =
    xi_des(3); z_des_d = xi_des_d(3);
    % desired angle
    if param.control_type == 2 || param.control_type == 3
        eta_des = pos2ang(xi, xi_des, xi_des_d, xi_des_dd, nu_I, param, PD_var); else
        eta_des = zeros(size(eta));
    end
    % desired angle derivative eta_des_d
    = zeros(size(eta));
    % unload values
    phi_des = eta_des(1); phi_des_d = eta_des_d(1);
    theta_des = eta_des(2); theta_des_d = eta_des_d(2);
    psi_des = eta_des(3); psi_des_d = eta_des_d(3);

    T = (param.g + PD_var.Kd_z * (z_des_d - z_d) + PD_var.Kp_z * (z_des - z)) * ( param.m /
(cos(phi) * cos(theta)) );
    tau_phi = (PD_var.Kd * (phi_des_d - phi_d) + PD_var.Kp * (phi_des - phi)) * param.I(1,1);
    tau_theta = (PD_var.Kd * (theta_des_d - theta_d) + PD_var.Kp * (theta_des - theta))
* param.I(2,2);
    tau_psi = (PD_var.Kd * (psi_des_d - psi_d) + PD_var.Kp * (psi_des - psi)) * param.I(3,3);
```

```
    ctrl(1) = ( T / (4 * param.Kt) ) - ( tau_theta / (2 * param.Kt * param.l) ) - ( tau_psi / (4
* param.Kb) );
    ctrl(2) = ( T / (4 * param.Kt) ) - ( tau_phi / (2 * param.Kt * param.l) ) + ( tau_psi / (4
* param.Kb) );
    ctrl(3) = ( T / (4 * param.Kt) ) + ( tau_theta / (2 * param.Kt * param.l) ) - ( tau_psi / (4 *
param.Kb) );
    ctrl(4) = ( T / (4 * param.Kt) ) + ( tau_phi / (2 * param.Kt * param.l) ) + ( tau_psi / (4
* param.Kb) );
end

%% Desired Trajectory
% position trajectory
function [xi_des, xi_des_d, xi_des_dd]  = circular_traj(t, T) radius =
    3;
    ang = 2 * pi * t/T;

    xi_des = zeros(3, 1);
    xi_des_d = zeros(3, 1);
    xi_des_dd = zeros(3, 1);
    % define circular trajectory for the x and y direction
    xi_des(1) = radius * cos(ang);
    xi_des(2) = radius * sin(ang);
    xi_des(3) = t/T; %  equation of line
    % define derivative
    xi_des_d(1) = -2 * pi * radius * sin(ang)/T;
    xi_des_d(2) = 2 * pi * radius * cos(ang)/T;
    xi_des_d(3) = 1/T;
    % define double derivative
    xi_des_dd(1) = -4 * radius * pi * pi * cos(ang) / (T * T);
    xi_des_dd(2) = -4 * radius * pi * pi * sin(ang) / (T * T);
```

```
    xi_des_dd(3) = 0;
end

function [xi_des, xi_des_d, xi_des_dd]  = spiral_traj(t, T) radius
    = 3;
    ang = 2* pi * t/T; h
    = 5;

    xi_des = zeros(3, 1);
    xi_des_d = zeros(3, 1);
    xi_des_dd = zeros(3, 1);
    % define circular trajectory for the x and y direction
    xi_des(1) = radius * cos(ang);
    xi_des(2) = h * t/T;
    xi_des(3) = t/T; %  equation of line
    % define derivative
    xi_des_d(1) = -2 * pi * radius * sin(ang)/T;
    xi_des_d(2) = h / T;
    xi_des_d(3) = 1/T;
    % define double derivative
    xi_des_dd(1) = -4 * radius * pi * pi * cos(ang) / (T * T);
    xi_des_dd(2) = 0;
    xi_des_dd(3) = 0;
end

% convert position trajectory to angle
function eta_des = pos2ang(xi, xi_des, xi_des_d, xi_des_dd, nu_I, param, PD_var) x_des
    = xi_des(1); x_des_d = xi_des_d(1); x_des_dd = xi_des_dd(1);
    y_des = xi_des(2); y_des_d = xi_des_d(2); y_des_dd = xi_des_dd(2); z_des =
    xi_des(3); z_des_d = xi_des_d(3); z_des_dd = xi_des_dd(3);
```

```
    x = xi(1); x_d = nu_I(1);
    y = xi(2); y_d = nu_I(2);
    z = xi(3); z_d = nu_I(3);

    % Psi_des will be fixed to zero
    eta_des = zeros(3, 1);

    T_des = param.m * ( z_des_dd + param.g) + param.Kd(3, 3) * z_des_d;
    eta_des(1) = -(param.m * y_des_dd + param.Kd(2, 2) * y_des_d + PD_var.Kp_xy * (y_des - y)) /
T_des;
    eta_des(2) = (param.m * x_des_dd + param.Kd(1, 1) *  x_des_d  + PD_var.Kp_xy * (x_des - x)) /
T_des;
end

function ctrl = Feedback_Lin(omega_B, eta, xi, nu_I, param, PD_var, xi_des)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The inputs are:
% Omega_B is the angular velocity in the body frame [p, q, r]
% eta is the euler angles [phi, theta, psi]
% xi is the current position in the inertia frame [x, y, z]
% nu_I is the velocity in the inertia frame [x_dot, y_dot, z_dot]
% param is a structure containing the vehicle parameters such as I,g,kt etc
% PD_var is a vector containing the gains used in the feedback linearization
% xi_des is a 3x1 vector of the desired position. Desired yaw is assumed to
% be 0
% The current version does not use nu_I as it is assumed to  be a functions
% of the desired postion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Unpack control gains
```

```
k_x_hat      =      PD_var.FB(1);
k_y_hat      =      PD_var.FB(2);
k_z_hat      =      PD_var.FB(3);
k_ax_hat     =      PD_var.FB(4);
k_ay_hat     =      PD_var.FB(5);
k_az_hat     =      PD_var.FB(6);
k_phi_hat    =      PD_var.FB(7);
k_theta_hat  =      PD_var.FB(8);
k_psi_hat    =      PD_var.FB(9);
k_p_hat      =     PD_var.FB(10);
k_q_hat      =     PD_var.FB(11);
k_r_hat   =   PD_var.FB(12);
%Unpack variables (Parameters) g
= param.g;
m   = param.m; l
= param.l;
Jr  = param.Jr;
Kt  = param.Kt;
Kb  = param.Kb;
Kdx = param.Kd(1,1);
Kdy = param.Kd(2,2);
Kdz = param.Kd(3,3);
Ixx = param.I(1,1); Iyy
= param.I(2,2); Izz =
param.I(3,3);

% Assume gyrosopic effect gamma = 0 since omega_gamma approx 0
% Unpack variables (current states) x =
xi(1); x_d = nu_I(1);
y = xi(2); y_d = nu_I(2);
z = xi(3); z_d = nu_I(3);
```

```
    phi = eta(1); theta = eta(2); psi = eta(3);
    p = omega_B(1); q = omega_B(2); r = omega_B(3);

    % Unpack variables (desired states)
    x_des = xi_des(1); y_des = xi_des(2); z_des = xi_des(3);

    %Compute the desired inner loop (rotation) from desired outer loop (translation) Fx =
    k_ax_hat * x_d + Kdx * k_x_hat * (x - x_des) / m;
    Fy = k_ay_hat * y_d + Kdy * k_y_hat * (y - y_des) / m;  Fz =
    k_az_hat * z_d + Kdz * k_z_hat * (z - z_des) / m + g;

    T = m * sqrt(Fx^2 + Fy^2 + Fz^2);

    % desired angle computation - x and y desired translated to phi and theta
    psi_des = 0; % Set the desired psi to 0
    phi_des = asin( (m * tan(psi_des) * Fx / T - Fy * m / T) / ... (tan(psi_des) *
                sin(psi_des) + cos(psi_des)) );
    theta_des = atan( (m * Fx * cos(psi_des) * cos(psi_des) / T - m * cos(psi_des) * sin (psi_des) *
Fy / T) / ...
                (cos(psi_des) * Fz * m / T) );

    %Compute the desired controls from desired inner loop
    omega_B_des = [1, sin(phi_des) * tan(theta_des), cos(phi_des) * tan(theta_des); 0,
                                cos(phi_des),                -sin(phi_des);
            0, sin(phi_des) / cos(theta_des), cos(phi_des) / cos(theta_des)] \ ...
            [k_phi_hat * (phi - phi_des);
             k_theta_hat * (theta - theta_des);
             k_psi_hat * (psi - psi_des)];
    p_des = omega_B_des(1); q_des = omega_B_des(2); r_des = omega_B_des(3); tau_des =
    diag([Ixx * k_p_hat, Iyy * k_q_hat, Izz * k_r_hat]) * (omega_B -
omega_B_des) - ...
```

```
                [(Iyy - Izz) * q_des * r_des; (Izz -
                 Ixx) * p_des * r_des; (Ixx - Iyy)
                 * p_des * q_des];

    %Compute propeller omegas from control allocation
    inputs = [T; tau_des];
    ctrl = [  Kt,      Kt,   Kt,   Kt;
              0, -l*Kt,      0, l*Kt;
           -l*Kt,       0, l*Kt,      0;
             -Kb,      Kb,  -Kb,    Kb] \ inputs;
end
```