Autonomous Control of Multi-rotor Unmanned Aerial Vehicles

BY

Lachlan Drake

3 December, 2021

Supervisor:
Associate Professor José De Dona

A thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Electrical Engineering at The University of Newcastle, Australia

Abstract

This thesis outlines modelling, control and state estimation techniques for autonomous flight of multi-rotor unmanned aerial vehicles (UAV). First, an accurate mathematical model of the system was established and model parameters were measured or estimated. This model was implemented in MATLAB Simulink, such that various control techniques could be simulated. A stable and accurate control system was progressively developed by iterative testing and improvement. Initially PID control was implemented, followed by a backstepping controller and finally an integral backstepping controller. Actuator saturation was then accounted for with the use of a sigmoid function. This control system was tested and found to adequately stabilise the system and also allow positional commands to be followed. Next, state estimation techniques were explored with the use of various sensors in an extended Kalman filter (EKF) algorithm. In particular two algorithms were developed and tested, with one using Global Positioning System (GPS) technology and the other using an optical flow sensor (OFS) for use in GPS-limited environments. Finally, a UAV hardware platform was configured with open source flight control software (ArduPilot) and flown such that the EKF algorithm could be tested on real flight data gathered from the sensors. The simulation results demonstrated that the OFS-based algorithm was effective in estimating position with minimal drift when the vehicle is in a hover close to the ground. However, the GPS-based algorithm produced significantly more accurate position results, both in simulation and in hardware.

Acknowledgements

I would like to express my gratitude to all those who have supported me through my journey over the preceding year.

Firstly I would like to thank my supervisor José De Dona. Despite the additional difficulty of unpredictable lockdowns, José made every effort to be available both virtually and face-to-face (when possible) and provided invaluable guidance throughout the year. Without his insight and experience this project would not have been possible.

I would also like to extend my gratitude to all of the academic staff who have taught and guided me throughout my studies.

Lastly, I would like to thank all my family and friends for their support throughout this challenging year. I would especially like to thank my fiancée for her love and support throughout the year.

COVID Impact Statement

The Coronavirus (COVID-19) pandemic has harshly impacted the world in many ways for the last couple years. Although Australia has been extremely fortunate in avoiding the worst of the pandemic, there were a number of limitations imposed upon this project by the pandemic and related lockdowns. Due to being unable to access lab equipment for a significant period of time, this project was certainly made more difficult. The main impacts to the project are summarised below:

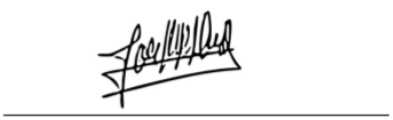
- Limited access to equipment for testing hardware, debugging, soldering and connecting sensors, etc. Resulting in larger amounts of time and effort being necessary to perform hardware-related tasks.
- An overall lack of space for storing and working on hardware.
- Inability to perform flight testing due to lockdowns. Unable to legally visit an open space which would be suitable for flight testing. Living in an apartment block with limited space meant that flight testing in any kind of unobstructed space was next to impossible during the lockdown period.
- Delays in component delivery and lack of availability. Due to the overstretched postal service, there were significant delays (4-5 weeks) in delivery of sensors and components.

Contributions

My main contributions to the project are summarised below:

- Performed a comprehensive literature review of current multi-rotor modelling methods, control techniques and sensor fusion algorithms.
- Derived a model for a hexacopter aircraft based upon Newton-Euler mechanics and implemented the model in Simulink.
- Developed a PID control system for a hexacopter and implemented the system in Simulink.
- Derived a robust control system based on the backstepping technique and implemented this system in Simulink.
- Applied the extended Kalman filter to develop a GPS-reliant sensor fusion algorithm for use both in simulation and with recorded data.
- Developed a second sensor fusion algorithm for use in situations where GPS is either unavailable or unreliable.
- Installed hardware and configured software on a custom hexacopter vehicle with a commercial flight controller.
- Performed test flights to record sensor data to verify the sensor fusion algorithms.

Lachlan Drake



A/Prof. José De Dona

Contents

Abstract i

Acknowledgements ii

COVID Impact Statement iii

Contributions iv

1 Introduction 1
- 1.1 Project Motivation . . . 1
- 1.2 Thesis Outline . . . 3

2 Background 4
- 2.1 Modelling Multi-Rotor Vehicles . . . 4
  - 2.1.1 Modelling Approaches . . . 4
  - 2.1.2 Newton-Euler Equations . . . 5
  - 2.1.3 Reference Frames and Co-ordinate Systems . . . 6
- 2.2 Control Techniques . . . 7
  - 2.2.1 Proportional Integral Derivative (PID) Control . . . 7
  - 2.2.2 Backstepping Control . . . 8
- 2.3 State Estimation . . . 10
  - 2.3.1 Extended Kalman Filter . . . 11
  - 2.3.2 Magnetometer . . . 11
  - 2.3.3 Barometer . . . 12
  - 2.3.4 GPS Denied Environments . . . 12
- 2.4 Chapter Summary . . . 13

3 Modelling of Multi-Rotor Aircraft 14
- 3.1 General Multi-Rotor Model . . . 14
  - 3.1.1 Reference Frames . . . 14
  - 3.1.2 Model Dynamics . . . 16
  - 3.1.3 Additional Translational Motion Equations . . . 18
- 3.2 Hexacopter Model . . . 19

3.3 Simulation . . . 21
3.4 Chapter Summary . . . 22

4 Control of Multi-Rotor Aircraft 23
4.1 Pseudo-inverse of Motor Speed Matrix . . . 23
4.2 PID Control . . . 24
4.2.1 Simplified Model . . . 24
4.2.2 Attitude Control . . . 25
4.2.3 Altitude Control . . . 25
4.2.4 Position Control . . . 26
4.2.5 Complete Control System . . . 26
4.2.6 Preliminary Results . . . 27
4.3 Backstepping Control . . . 29
4.3.1 Simplified State Space Model . . . 29
4.3.2 Attitude Control . . . 30
4.3.3 Position Control . . . 32
4.3.4 Preliminary Results . . . 33
4.4 Backstepping with Integral Action . . . 34
4.4.1 Position Control . . . 34
4.4.2 Preliminary Results . . . 36
4.5 Actuator Saturation . . . 37
4.6 Final Control System . . . 38
4.7 Chapter Summary . . . 40

5 State Estimation and Sensor Fusion 41
5.1 Filter Algorithm . . . 41
5.1.1 State Transition Functions . . . 41
5.2 State Estimation with GPS . . . 42
5.2.1 Sensor Models . . . 42
5.2.2 Preliminary Results . . . 43
5.3 State Estimation without GPS . . . 46
5.3.1 Sensor Models . . . 46
5.3.2 Preliminary Results . . . 46
5.4 Chapter Summary . . . 50

6 Implementation 51
6.1 Hardware . . . 51
6.1.1 The Cube Autopilot . . . 51
6.1.2 ArduPilot Software . . . 52
6.1.3 Companion Computer . . . 52
6.1.4 Sensors . . . 53
6.2 Flight Testing . . . 54

6.3 Discussion of Results 57
6.4 Chapter Summary 57

7 Conclusion and Extensions **58**
7.1 Conclusion 58
7.2 Future Extensions 60

Bibliography **61**

Appendices **A-1**

A Simulink Models **A-1**
A.1 Plant Model A-1
A.2 Control Inputs to Motor Speeds A-7
A.3 Control System A-8

Chapter 1

Introduction

This chapter introduces the topic of this thesis - unmanned aerial vehicles - and discusses their contemporary applications. In doing so, this chapter establishes the motivation for this thesis and potential applications of the results. An outline of the content in each chapter is also presented.

1.1 Project Motivation

Unmanned aerial vehicles (UAVs) allow tasks to be undertaken in difficult to access areas or risky environments without endangering human life. In many current applications, a human operator is required for remote control of the vehicle. However, with current technological advances in automation, it is becoming viable to use UAVs which operate completely autonomously i.e. without a human operator. This has the potential to improve the safety, efficiency, reliability, and cost of many processes. However, autonomous control of an aerial vehicle is a complex problem requiring the consideration of many areas of research, including but not limited to: modelling, control law design, sensor fusion, navigation, collision avoidance and fault tolerance.

UAVs were originally developed and used mainly by military organisations, beginning with the Sperry Aerial Torpedo developed during World War I for the US Navy, a radio-controlled biplane designed to be used as a flying bomb [1]. This was followed by decades of development leading to vehicles used for aerial target practice, reconnaissance, and remote bombing. However, in recent decades there has been an increased interest in the use of UAVs for both commercial and scientific applications. Commercial applications include surveying, aerial photography/videography, package delivery and pesticide spraying. UAVs are increasingly being utilised in many industries from mining to agriculture. Technological advances in computing and electronics have decreased the cost of UAVs and this has resulted in a large selection of vehicles being more widely available to

hobbyists, not just large companies. Scientific research has also benefited from the use of UAVs, with applications including monitoring bodies of water for algal blooms, identifying plant species, monitoring coastal erosion and tracking/counting animal populations. One recent study [2] uses UAVs to track aquatic vertebrates without using invasive tagging methods. Even more recently, the first unmanned flights on another planet were completed by NASA's Ingenuity helicopter on Mars [3], demonstrating that UAV technology could be a viable option for extra-terrestrial exploration in the near future. Further, NASA has a mission planned for launch in 2026 to send a multi-rotor aircraft, named Dragonfly, to explore Titan – the largest moon of Saturn [4]. In such applications communication with the vehicles is extremely limited due to the astronomical scale of the distance between the ground station and the vehicle, as well as the potential for obstructions and interference. Therefore the vehicle must be able to operate autonomously without human intervention. Also, since the conditions on other planets are not as well characterised as those on Earth, the control system must be robust to model uncertainties and disturbances. Another key difference is the availability of positioning data - i.e. there are no GPS satellite constellations orbiting Mars or Titan, so the UAV must have another method of position estimation. Likewise, many environments on Earth have limited or unreliable GPS data e.g. indoors, underground or hostile territory with GPS jamming devices. The versatility of UAVs results in an almost limitless number of applications for their use.

UAVs come in many different configurations with the main classifications being fixed-wing and rotary-wing, however there also exist some hybrid configurations. Rotorcraft have the advantage of being able to perform Vertical Take-Off and Landing (VTOL), i.e. they do not require a runway for take off and landing. Also, rotorcraft are able to hover in a fixed position, as opposed to fixed wing vehicles which need to continually move to generate lift. On the other hand, fixed-wing vehicles are generally able to travel at much higher speeds and can fly at higher altitudes. Hybrid configurations generally combine the benefits of the two types - resulting in a vehicle which is capable of VTOL and hovering, but also able to travel at high speeds and altitudes when necessary. A well-known example of a vehicle with hybrid capabilities is the V-22 Osprey. This thesis will explore rotorcraft configurations due to their versatility and widespread commercial availability.

The defining characteristic of rotorcraft configurations is the number of rotors present on the vehicle. There is an almost limitless number of configurations from single rotor vehicles (helicopters) up to octocopters (8 rotors) and beyond. The topic of this thesis is centred around unmanned multi-rotor vehicles and will begin with an overview of the basic dynamics of these systems. These ideas will then be extended to the specific configuration of a six-rotor vehicle - the hexacopter. Once an accurate model is developed, it will be used for simulation to enable the development and testing of a robust control system. A method for state estimation will then be explored with a focus on position estimation in GPS denied environments.

1.2 Thesis Outline

This thesis is divided into seven chapters. The contents of the chapters (including this one) are summarised below:

- Chapter One introduces the main concepts surrounding unmanned aerial vehicles and outlines the motivation for the project.
- Chapter Two investigates the technical background necessary for the concepts covered in this thesis. The key theory is summarised and useful formulae are presented and explained. The current literature around modelling, control and state estimation for multi-rotor vehicles is also evaluated.
- Chapter Three develops a comprehensive mathematical model to describe the dynamics of a general multi-rotor vehicle. This is then extended to a specific configuration with six propellers known as a hexacopter. This model is simulated using MATLAB Simulink to enable the development and verification of control systems in the following chapter.
- Chapter Four explores the control of the hexacopter through the development and testing of a number of progressively more complex control systems. The chapter begins with the development of a PID controller. Next, backstepping control theory is used to stabilise the system. Then integral action is added to the backstepping controller and finally the issue presented by actuator saturation is addressed.
- Chapter Five addresses the estimation of the systems states using extended Kalman filtering techniques. First, an algorithm which relies upon GPS data is developed. Next, an alternative algorithm which uses an optical flow sensor is developed and tested for use in situations where GPS signal is not available. The two algorithms are then compared in simulation.
- Chapter Six discusses a custom hexacopter hardware platform. Additional sensors were connected to the flight controller and a companion computer was added to enable wireless communication. Software was configured onboard the flight controller, the companion computer and on a ground system computer to allow autonomous flights to be planned and carried out. This hexacopter is then used for gathering flight data for processing with the state estimator algorithms.
- Chapter Seven, the final chapter, summarises the results of the project and discusses future extensions.

Chapter 2

Background

This chapter serves two main purposes. Firstly, it evaluates the available literature surrounding the effectiveness of techniques for modelling, control and state estimation in unmanned aerial vehicles and related platforms. Secondly, it introduces the key mathematical, physical and technical concepts necessary for the discussion in the following chapters.

2.1 Modelling Multi-Rotor Vehicles

2.1.1 Modelling Approaches

In order to control a system, it is first necessary to have an understanding of the dynamics of the system. There are multiple methods used for developing mathematical models in order to describe the dynamics of multi-rotor vehicles. The Newtonian approach uses forces and torques in order to describe the system, whereas the Lagrangian approach utilises energies [5].

Further, attitude of the vehicle can be described by using either Euler angles or quaternions. Euler angles express attitude with three angles, each representing the rotation in 3D space around each axis. Alternatively, quaternion algebra is used to represent attitude with four scalar variables representing a point on the unit sphere [6]. Euler representation is more intuitive, however quaternion representation has the advantage of avoiding a problem known as gimbal lock which arises when two of the Euler rotation axes align, resulting in a loss of a degree of freedom.

Newton-Euler formalism combines Newtonian mechanics and Euler angle representation in order to describe the rotational and translational dynamics of rigid bodies. This approach is commonly used to develop a mathematical model for multi-rotor vehicles. The main advantage of this method

is that it is simple to understand. Alternatively, using Euler-Lagrange formalism results in a more compact derivation [7].

Multiple studies demonstrate the effectiveness of multi-rotor models developed using Newton-Euler formalism [8], [9]. Therefore, this thesis will utilise this method. Due to the uncomplicated nature of this formulation, identifying issues in the model and debugging simulations will, in general, be easier as compared to similar models formed using quaternion algebra or Lagrangian techniques.

2.1.2 Newton-Euler Equations

The Basic Kinematic Equation (BKE) is used in Newton-Euler dynamics to describe the relative motion of two reference frames [10]. The BKE is given in Equation 2.1:

$$\frac{d\mathbf{Q}_a}{dt} = \frac{d\mathbf{Q}_b}{dt} + \omega_b \times \mathbf{Q}_b \tag{2.1}$$

where $\mathbf{Q}_a$ and $\mathbf{Q}_b$ represent a three dimensional quantity (e.g position, velocity) with respect to reference frames $\{\mathrm{A}\}$ and $\{\mathrm{B}\}$ respectively, and ω_b represents the angular velocity of $\{\mathrm{B}\}$ with respect to $\{\mathrm{A}\}$. Now, consider reference frame $\{\mathrm{B}\}$ to be fixed at the centre of mass of some rigid body. By considering the mass of the body and taking $\mathbf{Q}$ as the translational velocity of the body ($\mathbf{v}$), the BKE (Equation 2.1) can be written in terms of forces to give the first of the Newton-Euler equations.

$$\begin{aligned}
\frac{d\mathbf{v}_a}{dt} &= \frac{d\mathbf{v}_b}{dt} + \omega_b \times \mathbf{v}_b \\
m\dot{\mathbf{v}}_a &= m(\dot{\mathbf{v}}_b + \omega_b \times \mathbf{v}_b) \\
\mathbf{F}_a &= m(\dot{\mathbf{v}}_b + \omega_b \times \mathbf{v}_b)
\end{aligned}$$

The moments of forces (torques) about the centre of mass may be considered in a similar way in order to produce the second of the Newton-Euler equations [10]:

$$\mathbf{M}_b = \mathbf{I}_b\dot{\omega}_b + \omega_b \times \mathbf{I}_b\omega_b$$

where $\mathbf{M}_b$ represents the three dimensional moment about $\{\mathrm{B}\}$, and $\mathbf{I}_b$ is a 3×3 matrix representing the moment of inertia about the centre of mass.

2.1.3 Reference Frames and Co-ordinate Systems

Establishing a model of an aerial vehicle requires an understanding of coordinate systems or reference frames and how they relate to given quantities (displacement, velocity, acceleration, etc). There are two main types of reference frames: inertial frames and non-inertial reference frames. An inertial reference frame is one which is not accelerating and within which Newton's laws are valid [11]. On the other hand, a non-inertial reference frame is one which is experiencing acceleration.

There are many different reference frames which may be used to describe aircraft motion. For example, an Earth-fixed local frame is a reference frame with its origin fixed at an arbitrary point on the Earth. For the purposes of modelling an aerial vehicle, an Earth-fixed local frame can be considered as an inertial frame although it is experiencing acceleration due to the Earth's rotation. However, a reference frame which is fixed on the aircraft body is non-inertial as it experiences non-negligible acceleration due to the vehicle's movement. Rotation matrices enable the conversion of quantities between reference frames.

The most commonly used reference frame for Earth-based navigation identifies a position on the Earth in terms of latitudes and longitudes. In particular, GPS data is generally presented in this frame. Converting these global positions to displacements within a local reference frame can be done in a number of ways. The haversine formula is one such method, which considers two sets of coordinates and computes the displacement between them. The haversine formula arises from spherical trigonometry to compute the great-circle distance between two points on a sphere. The bearing (β) and distance (d) are given as:

$$\begin{aligned} d &= 2Rsin^{-1}\left(\sqrt{sin^2\left(\frac{\Delta\chi}{2}\right)+cos(\chi_1)cos(\chi_2)sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \\ \beta &= atan2\left[sin(\Delta\lambda)cos(\chi_2), cos(\chi_1)sin(\chi_2)-sin(\chi_1)cos(\chi_2)cos(\Delta\lambda)\right] \end{aligned} \tag{2.2}$$

where R is the Earth's radius, λ_1, χ_1 represent the initial latitude and longitude coordinates respectively and λ_2, χ_2 represent the final coordinates. $\Delta\lambda$ is equivalent to $(\lambda_2-\lambda_1)$ and $\Delta\chi$ is equivalent to $(\chi_2-\chi_1)$. The function $atan2(b,a)$ finds the angle between the positive x axis and the point defined by (a,b). It is worth noting that this formula considers the Earth as a perfect sphere, when in reality it is elliptical to some extent. However, over short distances the effect of the Earth's elliptical nature on the accuracy of the results is negligible.

2.2 Control Techniques

There are many control techniques both linear and nonlinear which have been employed to stabilise UAVs and allow position tracking. Two of the most common techniques - one linear technique and one nonlinear technique - will be explored in detail in this section. These are PID control (linear) and backstepping control (nonlinear). There are many other techniques which show promise for UAV control including (but not limited to) model predictive control (MPC), sliding-mode control, feedback linearisation, geometric techniques and learning-based control [12]. However, these will not be discussed in detail here.

2.2.1 Proportional Integral Derivative (PID) Control

A PID control law consists of three terms each associated with a gain:

$$u(t) = K_p e(t) + K_I \int_0^t e(t)\,dt + K_D \frac{de(t)}{dt}$$

Where e(t) represents the measured error between the desired setpoint and the measured value of the variable.

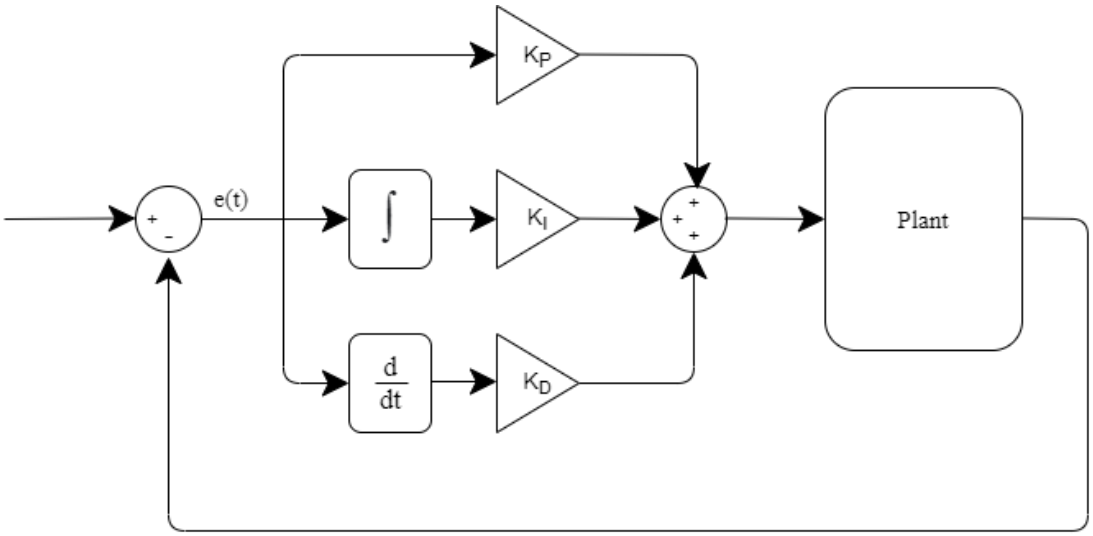


Figure 2.1: PID control block diagram.

PID control is comparatively simple compared to other techniques, but several studies suggest that it is generally effective for stabilising a UAV around a hovering setpoint, in the absence of large disturbances [8], [13], [14].

2.2.2 Backstepping Control

The plant dynamics of a multi-rotor vehicle are inherently nonlinear, therefore nonlinear control techniques may be necessary to stabilise the vehicle. Backstepping control is a nonlinear recursive technique which is based upon Lyapunov stability. The mathematical background on the backstepping method in this section is largely based upon the work presented by Kokotovic [15] and Krstić et al. [16].

Lyapunov Stability

Direct Lyapunov theorem, as stated in Appendix B of [17], considers a time invariant system of the form:

$$\dot{x} = f(x)$$

with $x \in \mathbb{R}^n$, $f(x)$ is smooth and $f(0) = 0$. If there exists a positive definite, proper and smooth function $V(x)$ which satisifies:

$$\frac{\partial V}{\partial x} f(x) < 0$$

for all $x \neq 0$, then the equilibrium of the system is globally asymptotically stable. This means that given any initial conditions, over time ($t \to \infty$) the system will approach its equilibrium point [18].

Now, consider a time invariant nonlinear system with states $\mathbf{x}$ and inputs $\mathbf{u}$:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2.3}$$

with equilibrium at $\mathbf{x} = \mathbf{0}$. Following from Direct Lyapunov theorem, to stabilise such a system, a control law $\mathbf{u} = \alpha(\mathbf{x})$ must be designed such that the equilibrium point of Eq. (2.3) is globally asymptotically stable. To do this, a function $V(\mathbf{x})$ may be chosen as a candidate Lyapunov function. The control input, $\alpha(\mathbf{x})$, must then be chosen such that $\dot{V}(\mathbf{x}) = \frac{\partial V}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, \alpha(\mathbf{x}))$ is negative definite for all $\mathbf{x}$.

In order to apply this method to the tracking problem, the states of the system may be redefined as $\mathbf{z} = \mathbf{x}_d - \mathbf{x}$, where $\mathbf{x}_d$ represents the desired value of the states.

Strict Feedback Form

Backstepping control is best demonstrated on a system in strict feedback form [15]. Consider the system:

$$\begin{aligned}\dot{x}_1 &= f_1(x_1, x_2)\\ \dot{x}_2 &= x_3 + f_2(x_1, x_2)\\ \dot{x}_3 &= u + f_3(x_1, x_2, x_3)\end{aligned}$$

Now consider that x_2 can be used as a "virtual control" to stabilise $\dot{x}_1$, by choosing an appropriate Lyapunov function $V_1(x_1)$. We will denote the virtual control (or desired value of the state) with a subscript 'd', e.g. $x_{2d}(x_1)$ denotes the desired value of x_2. It is worth noting that although this is a function of x_1, it will henceforth be written as x_{2d} for simplicity. Likewise for other virtual controls. Now consider the error between the desired and actual values: $z_2 = x_2 - x_{2d}$. As we are considering the stabilisation problem, $x_{1d} = 0$ and therefore $z_1 = x_1$. To stabilise z_1, the virtual control x_{2d} must be chosen such that the Lyapunov function $V_1(z_1)$ has a negative time derivative.

Likewise, x_3 may be used to stabilise z_2 with a virtual control ($x_{3d}(z_1, z_2)$). A third error term is defined: $z_3 = x_3 - x_{3d}$. Also, a second Lyapunov function is defined:

$$V_2(z_1, z_2) = V_1(z_1) + \frac{1}{2}z_2^2$$

The feedback law x_{3d} is chosen such that the time derivative of this Lyapunov function is negative.

The final iteration for this system is complete by choosing the actual control u to stabilise z_3. Again, this is done in the same way by defining a third Lyapunov function:

$$V_3(z_1, z_2, z_3) = V_2(z_1, z_2) + \frac{1}{2}z_3^2$$

The control input u is then chosen such that the time derivative is negative definite. This results in global asymptotic stability of the system at the equilibrium point. Thus the entire system is stabilised by the single control input (u) by the recursive backstepping of virtual control inputs (x_{2d}, x_{3d}). This concept is able to be applied to control the position of multi-rotor aircraft, by using the Euler angle values as virtual controls.

Implementation in UAV Control

A large number of studies have investigated various implementations of backstepping control strategies both in simulation and on physical UAV platforms. This control method is flexible and variations on the control law will arise depending on the design decision to cancel or retain nonlinearities. Also, the specific UAV platform and the modelling method used to describe the

system can result in differences between the derived control laws.

Backstepping control systems for multi-rotor UAVs are simulated with successful results in many studies. [19] demonstrates stabilisation of the rotational subsystem of a quadrotor using a backstepping control law, with the translational subsystem controlled using PID. [20] demonstrates satisfactory simulation results for trajectory tracking of a quadrotor in the presence of external disturbances. [21] presents similar results with a controller that allows a speed profile and yaw angle to be specified as a function of displacement along the path. A backstepping controller is found to perform better in the presence of wind as compared to sliding mode control in [14]. Other studies which show robust simulated results for backstepping controllers include [22], [23] and [24].

Further, multiple studies show good performance of backstepping controllers implemented on hardware platforms. [25] implements backstepping control on a UAV hardware platform on a test stand to demonstrate control of the yaw angle and z position only. [8] demonstrates the use of backstepping with integral action for controlling a quadrotor UAV's attitude and position autonomously during indoor flights. Further research is required to demonstrate robust autonomous control in the presence of physical (rather than simulated) wind and external disturbances.

2.3 State Estimation

To implement a stable control system, it is essential to have an accurate method of estimating the system states which the controller uses. There are many different types of sensors which can be used to gather data describing a vehicle's position, velocity, attitude and angular rates. A state estimation algorithm combines this data to reach a reasonable approximation of the system states. Common sensor fusion techniques include complementary filters, neural network-based algorithms and Kalman filter-based algorithms.

The Kalman filter (also known as linear quadratic estimator) is a recursive algorithm which uses measurement data whilst accounting for noise to estimate unknown variables. The standard Kalman filter is the optimal filter for linear systems that are well characterised by the established model. However, there are two main Kalman filter-based techniques which have been adapted for non-linear systems - the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). Both algorithms are similar however the UKF applies an unscented transformation to propagate the random variables throughout the algorithm [26]. However, [27] shows that the UKF is more computationally expensive than the EKF for inertial navigation and also that in GPS denied conditions the UKF offers no benefit over the EKF. Therefore, the EKF algorithm will be explored in more detail in this thesis.

2.3.1 Extended Kalman Filter

The extended Kalman filter applies the same concepts as the standard Kalman filter, with the addition of iteratively linearising the system around the current state estimates. The linearisation is performed using first order Taylor series. There are two main steps to the EKF algorithm: predict and update. The "predict" step first uses the state transition functions ($\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$) with the previous state estimate ($\hat{\boldsymbol{x}}_{k-1}$) and the current input ($\boldsymbol{u}$) values. Then the covariance matrix ($\boldsymbol{P}$) is predicted using the Jacobian of the state transition function (F).

$$\begin{aligned}\hat{\boldsymbol{x}}_k^- &= \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_k)\\ \boldsymbol{P}_k^- &= \boldsymbol{F}_k \boldsymbol{P}_{k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k\end{aligned}$$

where $\boldsymbol{Q}_k$ represents the process noise covariance matrix. $\boldsymbol{F}_k$ represents the Jacobian of the state transition function evaluated at ($\hat{\boldsymbol{x}}_k^-$, $\boldsymbol{u}_k$). That is $\boldsymbol{F}_k = \frac{\partial f}{\partial \boldsymbol{x}}|_{\hat{\boldsymbol{x}}_k^-,\, \boldsymbol{u}_k}$. The "-" superscript denotes the initial estimate from the predict step.

The "update" step uses these predictions and compares them with the sensor measurements to achieve more accurate estimates.

$$\begin{aligned}\boldsymbol{K}_k &= \boldsymbol{P}_k^- \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}\\ \hat{\boldsymbol{x}}_k &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \left[z_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^-)\right]\\ \boldsymbol{P}_k &= (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k^-\end{aligned}$$

where z_k represents the sensor measurements, $\boldsymbol{h}(\cdot)$ represents the measurement functions and $\boldsymbol{I}$ represents an identity matrix. $\boldsymbol{H}_k$ represents the Jacobian of the measurement function evaluated at $\hat{\boldsymbol{x}}_k^-$. That is $\boldsymbol{H}_k = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}|_{\hat{\boldsymbol{x}}_k^-}$.

2.3.2 Magnetometer

The Earth's magnetic field vector varies with latitude, longitude, altitude and time. In order to effectively utilise magnetometer data to determine true north it may be necessary to have knowledge of the magnetic field vector's expected value at a particular location. The International Geomagnetic Reference Field is a model of the Earth's magnetic field which is produced and updated by the International Association of Geomagnetism and Aeronomy (IAGA). This model allows a magnetic field vector to be estimated for any location on the globe and the magnetometer data will be compared to this reference to determine bearing in the sensor fusion algorithm. It is worth noting that many environments may have unpredictable magnetic fields caused by surrounding structures, machinery or other electrical equipment. For the purposes of state estimation it will be assumed

that the measured magnetic field is representative of the Earth's magnetic field.

2.3.3 Barometer

Air pressure varies with altitude, which allows barometric pressure sensors to be used in the estimation of relative height. The U.S. National Aeronautics and Space Administration (NASA) have developed a standard atmospheric model [28]. According to this model, pressure as a function of height above sea level can be found using Eq. (2.4), when temperature lapse rate is zero. Temperature lapse rate describes the rate at which temperature changes with altitude.

$$P = P_b exp\left[\frac{-gM(H-H_b)}{RT_b}\right] \tag{2.4}$$

where P_b, T_b and H_b are the reference values of pressure, temperature and height respectively. The subscript b is representative of one of seven layers of the atmosphere. H is the height at which pressure is calculated, g is gravitational acceleration, M is the molar mass of air and R is the universal gas constant.

Assuming the UAV operates at altitudes below 11 km above sea level, the subscript b is 0. Within this operating region the reference values are: $P_0 = 101325$ Pascals, $T_0 = 288.15$ Kelvin and $H_0 = 0$ metres.

2.3.4 GPS Denied Environments

Most state estimation algorithms currently in use for UAV systems are dependent on GPS data for position estimation. However in many situations, GPS signal may be unavailable, practically disabling the UAV. To expand the operating region of the vehicle, it is necessary to have some form of position tracking algorithm which is able to accurately monitor position without GPS data. There are a number of techniques for achieving this including visual odometry and simultaneous localisation and mapping (SLAM) [29]. Complex systems will use a number of cameras, however a simple approach to visual odometry can be achieved by utilising a single optical flow sensor (OFS). An OFS is a camera which can be attached to the underside of a vehicle to record visual data of the surface. As the vehicle moves, the OFS captures and compares sequences of images to determine the motion of the vehicle. Using this data in combination with IMU data has been shown to be effective for state estimation [30].

2.4 Chapter Summary

This chapter has provided the technical background necessary for the development in the following chapters. A comprehensive review of current literature has also been established in order to reveal the more effective methods and the gaps in current research.

Chapter 3

Modelling of Multi-Rotor Aircraft

In order to design a control system for an aircraft, a comprehensive mathematical model of the aircraft must first be defined. Within this chapter a complete state space representation is derived for a general multi-rotor vehicle. This is also extended to a specific vehicle configuration - the hexacopter. This final model is then simulated using Simulink to implement and validate the control methods in the following chapter.

3.1 General Multi-Rotor Model

The initial model will be derived for a general multi-rotor vehicle using Newton-Euler formalism. This is done by considering the torque around each of the three axes and the total force produced by the propellers as the inputs to the system. This model can then be extended to most standard multi-rotor configurations by considering how the propellers generate these forces and torques in the configuration being considered.

3.1.1 Reference Frames

We will consider a vehicle in three-dimensional space with six degrees of freedom (DoF)- three translational DoF (x, y and z), and three rotational DoF (roll ϕ, pitch θ and yaw ψ). For the derivation of this model, it is necessary to consider two reference frames: the Earth-fixed frame and the body-fixed frame. The Earth-fixed local frame is that which has its origin at an arbitrary, stationary point on the Earth's surface, while the body-fixed frame originates at the vehicle's centre of mass and travels with the vehicle [11]. The Earth-fixed frame will be considered an inertial frame for the purposes of this model. Note that in order to maintain right-hand orthogonality the inertial frame is traditionally defined with the positive x axis pointing North, the positive y axis pointing East and the positive z axis pointing down towards the Earth (known as North-East-Down

(NED) co-ordinates). The NED reference frame is represented in Fig. 3.1 a) with axes (x_n, y_n, z_n). For the purposes of modelling an aerial vehicle it is more intuitive to have the z axis positive in the upward direction (corresponding to vehicle altitude).Therefore, a second inertial frame will be defined with axes ($x = x_n$, $y = y_n$, $z = -z_n$) as shown in Fig. 3.1 b). This reference frame will be referred to as the Earth frame and will be the main coordinate system used throughout this thesis. The body frame is defined with its origin at the vehicle's centre of mass with the z axis opposite to the direction of propeller thrust. The input torques (τ_ϕ, τ_θ, τ_ψ) are defined around the axes of the body frame (x_b, y_b, z_b) as depicted in Fig. 3.1 c).

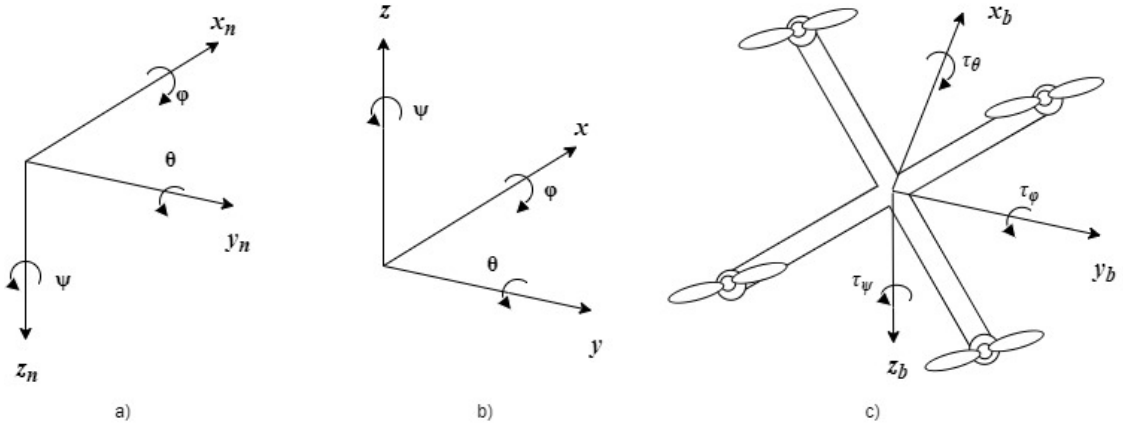


Figure 3.1: Reference frames and co-ordinate systems: a) NED Frame b) Earth Frame c) Body Frame

To transform linear quantities (displacement, velocity, acceleration) between the inertial NED coordinate system (x_n, y_n, z_n) and the body-fixed coordinate system (x_b, y_b, z_b) it is necessary to establish a rotation matrix (C_n^b) such that:

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = C_n^b \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix}$$

To establish this matrix it is necessary to consider the rotation of the body-fixed frame around each of its axes [31]. This is done by multiplying by three intermediate rotation matrices: $C_n^b(\phi)$, $C_n^b(\theta)$ and $C_n^b(\psi)$. Since matrix multiplication is not commutative, the order of these rotations is an important part of the model. The body frame is first rotated around the x axis (roll) by an angle ϕ to an intermediate reference frame:

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = C_n^b(\phi) \begin{bmatrix} x_{b_1} \\ y_{b_1} \\ z_{b_1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} x_{b_1} \\ y_{b_1} \\ z_{b_1} \end{bmatrix}$$

Rotating this set of coordinates around the y axis (pitch) will result in another intermediate reference frame:

$$\begin{bmatrix} x_{b_1} \\ y_{b_1} \\ z_{b_1} \end{bmatrix} = C_n^b(\theta) \begin{bmatrix} x_{b_2} \\ y_{b_2} \\ z_{b_2} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} x_{b_2} \\ y_{b_2} \\ z_{b_2} \end{bmatrix}$$

Finally, this frame is rotated around the z axis (yaw):

$$\begin{bmatrix} x_{b_2} \\ y_{b_2} \\ z_{b_2} \end{bmatrix} = C_n^b(\psi) \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix}$$

Therefore, via substitution, the multiplication of these three intermediate matrices gives the final rotation matrix- also commonly known as the Direction Cosine Matrix (DCM) [11]:

$$\begin{aligned} C_n^b &= C_n^b(\phi)C_n^b(\theta)C_n^b(\psi) \\ &= \begin{bmatrix} cos(\theta)cos(\psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi) & sin(\phi)sin(\theta)sin(\psi) + cos(\phi)cos(\psi) & sin(\phi)cos(\theta) \\ cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi) & cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi) & cos(\phi)cos(\theta) \end{bmatrix} \end{aligned}$$

An important characteristic of this rotation matrix is its orthogonality, which implies its inverse is equal to its transpose i.e. $C_b^n = (C_n^b)^{-1} = (C_n^b)^T$.

3.1.2 Model Dynamics

Standard configurations of multi-rotor vehicles consist of a number of propellers arranged symmetrically around the central body, all producing a force in the same direction. Therefore, regardless of the force produced by each individual propeller, the total thrust can only be produced in one direction which is aligned with the vehicle's vertical (z) axis. However, the differences in the forces produced by individual propellers will influence the rotational mechanics of the vehicle causing roll, pitch and yaw torques.

This system is an under-actuated system, since there are six degrees of freedom (x, y, z, ϕ, θ, ψ) but only four effective inputs- the torque around each axis and the total thrust ($\tau_\phi, \tau_\theta, \tau_\psi, F_T$). This means that it is not possible to directly affect two of the degrees of freedom. For example, assuming the body frame is initially aligned with the Earth frame, the two horizontal translational motions (x and y) can not be directly affected, as the thrust vector is aligned with the z axis. In order to cause translational motion in the x direction it is first necessary to change the pitch angle (θ), which causes the thrust vector to have an x component. Likewise, to move in the y direction the roll angle (ϕ) must first change.

To obtain the rates of change of the Euler angles ($\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$) from the angular velocities around the body-fixed axes (p, q, r), it is necessary to use the intermediate rotation matrices [32].

$$\begin{aligned}\begin{bmatrix} p \\ q \\ r \end{bmatrix} &= C_n^b(\phi)C_n^b(\theta)\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + C_n^b(\phi)\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & sin(\phi)cos(\theta) \\ 0 & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}\end{aligned}$$

Taking the inverse of this matrix gives a set of expression for the rates of change of the Euler angles.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix}\begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.1}$$

Next, the Newton-Euler equations of motion will be explored with respect to the body-fixed frame linear velocities (u, v, w) and angular velocities (p, q, r).

$$F = m\left(\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}\right) \tag{3.2}$$

For now it will be assumed the only forces acting on the vehicle are the vehicle's weight (in the positive direction of the NED frame z axis) and the total thrust produced by the propellers (in the negative direction of the body-fixed z axis). Substituting these forces into Equation Eq. (3.2) and rearranging gives:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw - gsin(\theta) \\ pw - ru + gsin(\phi)cos(\theta) \\ qu - pv + gcos(\phi)cos(\theta) - \frac{1}{m}F_T \end{bmatrix} \tag{3.3}$$

Next, the rotational motion is considered:

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = I_b\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I_b\begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Where I_b represents the vehicle's moment of inertia matrix. Assuming that the vehicle is symmet-

ric about its axes, I_b will be a diagonal matrix:

$$I_b = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

Rearranging for the derivatives of the angular velocities gives:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}} qr + \frac{1}{I_{xx}} \tau_\phi \\ \frac{I_{zz}-I_{xx}}{I_{yy}} pr + \frac{1}{I_{yy}} \tau_\theta \\ \frac{I_{xx}-I_{yy}}{I_{zz}} pq + \frac{1}{I_{zz}} \tau_\psi \end{bmatrix} \tag{3.4}$$

Finally, it will be necessary to obtain the position of the vehicle with respect to the Earth reference frame. Therefore, the derivatives of the NED frame x_n, y_n and z_n positions are obtained by multiplying the rotational matrix C_b^n by the body-fixed frame linear velocities (u, v and w). This gives Eq. (3.5), with c(•) and s(•) representing cos(•) and sin(•) respectively.

$$\begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.5}$$

These velocities can then be converted into the Earth frame with the z axis pointing up, as shown in Eq. (3.6).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ -\dot{z}_n \end{bmatrix} = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ s(\theta) & -s(\phi)c(\theta) & -c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.6}$$

Eq. (3.1), (3.3), (3.4) and (3.6) comprise a complete state space model describing a general multi-rotor vehicle with 12 states. This state space model will be used for simulation of the multi-rotor.

It is important to note that although this model does not make any approximations, it does not necessarily consider every force the vehicle may be subject to. For example, the effects of wind, air friction and gyroscopic effects have not been considered in this model for simplicity. Also the dynamics of the motors and propellers have not yet been considered, since the torques and total force were chosen as the inputs.

3.1.3 Additional Translational Motion Equations

Although the model defined in Section 3.1.2 is a complete state space model for the system, the translational motion is able to be represented in another form which will be useful for controllers

defined in the following chapter. The only forces acting on the vehicle are gravity - in the Earth frame negative z direction- and the total thrust - in the body frame negative z direction. This is expressed in Eq. (3.7)

$$\begin{aligned} F &= \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ s(\theta) & -s(\phi)c(\theta) & -c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -F_T \end{bmatrix} \\ &= \begin{bmatrix} (-c(\phi)s(\theta)c(\psi) - s(\phi)s(\psi))F_T \\ (-c(\phi)s(\theta)s(\psi) + s(\phi)c(\psi))F_T \\ c(\phi)c(\theta)F_T - mg \end{bmatrix} \end{aligned} \tag{3.7}$$

Using Newton's second law of motion ($F = ma$) in the Earth frame gives the second order differential equations in Eq. (3.8).

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} (-c(\phi)s(\theta)c(\psi) - s(\phi)s(\psi))\frac{F_T}{m} \\ (-c(\phi)s(\theta)s(\psi) + s(\phi)c(\psi))\frac{F_T}{m} \\ c(\phi)c(\theta)\frac{F_T}{m} - g \end{bmatrix} \tag{3.8}$$

3.2 Hexacopter Model

To extend the general multi-rotor model to a hexacopter, the six propeller forces need to be converted to the general model's inputs: three torques (around each axis) and one force (describing the total thrust). To achieve this, the configuration of the hexacopter must be defined. Fig. 3.2 shows the basic geometry of a hexacopter platform with $P_1, P_2, ..., P_6$ representing the six propellers.

Let $F_1, F_2, ..., F_6$ denote the force produced by each of the propellers as numbered in Fig. 3.2. The torques around each axis are proportional to the distance of each propeller from the axis. Using the geometry of the hexacopter, the torques around each axis can be derived.

$$\tau_\phi = d[(F_5 - F_2) + \cos(\alpha)(F_6 + F_4 - F_1 - F_3)]$$

$$\tau_\theta = d\sin(\alpha)(F_1 - F_3 - F_4 + F_6)$$

$$\tau_\psi = K_R(F_1 - F_2 + F_3 - F_4 + F_5 - F_6)$$

Where d represents the distance from each propeller to the hexacopter's centre of mass, α represents the angle between each of the arms and K_R represents a constant arising from the reaction

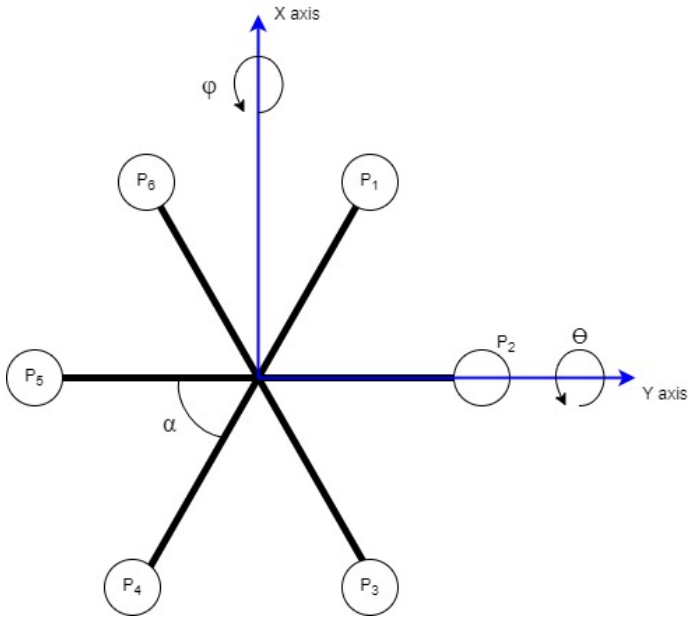


Figure 3.2: Basic configuration of the hexacopter with labelled axes.

force on the propellers. In this case, the angles between each of the arms will be considered to be equal, therefore α is 60°. This matches the configuration of the hardware platform discussed in Chapter 6. It follows that $\sin(\alpha) = \frac{\sqrt{3}}{2}$ and $\cos(\alpha) = \frac{1}{2}$

Next, it is necessary to express the torques in terms of the speed of the propellers. The force produced by the propellers is approximately proportional to the square of the speed of the propellers (Ω_i^2). For now it is sufficient to relate the quantities with a constant (C_T) which is known to be related to the physical properties of the propeller and the medium in which it is rotating. These quantities are related by a different constant when considering the yaw torque (τ_ψ) since this torque arises from reaction forces. The proportionality between the square of the propeller speeds and yaw torque will be represented by K_ψ. The total thrust produced by the propellers (F_T) is simply the sum of the forces produced by each propeller. The relationship between the propeller speeds and input torques and force can thus be expressed in matrix form as in Eq. (3.9).

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \\ F_T \end{bmatrix} = \begin{bmatrix} -\frac{C_T d}{2} & -C_T d & -\frac{C_T d}{2} & \frac{C_T d}{2} & C_T d & \frac{C_T d}{2} \\ \frac{C_T d\sqrt{3}}{2} & 0 & -\frac{C_T d\sqrt{3}}{2} & -\frac{C_T d\sqrt{3}}{2} & 0 & \frac{C_T d\sqrt{3}}{2} \\ K_\psi & -K_\psi & K_\psi & -K_\psi & K_\psi & -K_\psi \\ C_T & C_T & C_T & C_T & C_T & C_T \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \end{bmatrix} \tag{3.9}$$

It is also necessary to consider actuator saturation as part of this model. The motors which drive

the propellers are limited in that they can only spin in a single direction and have a maximum speed output. That is, motor speeds $(\Omega_1, \Omega_2, \Omega_3, \Omega_4, \Omega_5, \Omega_6)$ are restricted such that:

$$sat(\Omega_i) = \begin{cases} \Omega_{max}, & \Omega_i > \Omega_{max} \\ \Omega_i, & 0 \leq \Omega_i \leq \Omega_{max} \\ 0, & \Omega_i < 0 \end{cases} \tag{3.10}$$

The block diagram of the final hexacopter model which combines Eq. (3.1), (3.3), (3.4), (3.6), (3.9) and (3.10) is shown in Fig. 3.3.

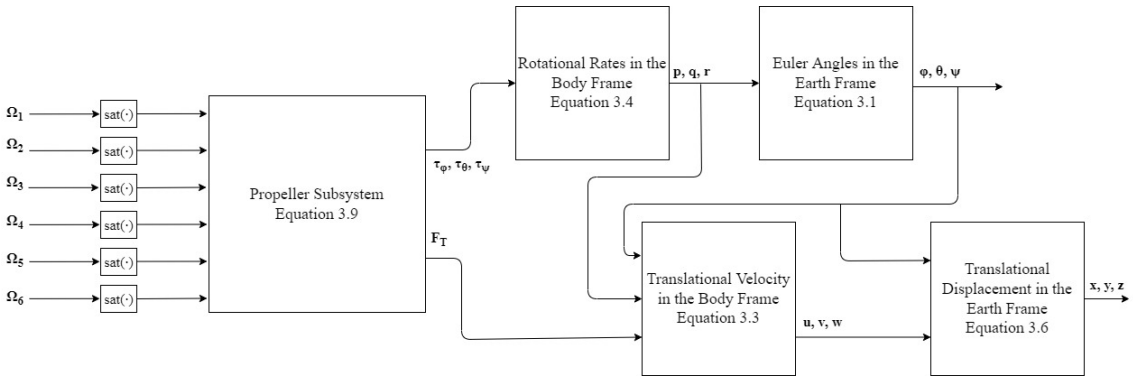


Figure 3.3: Hexacopter model block diagram.

3.3 Simulation

The model derived in the previous section is simulated using Simulink. The block diagrams created within Simulink are shown in Appendix A.1.

The intent is to simulate the physical vehicle described in Section 6.1. The vehicle's geometry and mass were measured, and are shown in Table 3.3 along with the other parameters used in the model. The maximum propeller speed was estimated based upon the voltage supplied by the battery and the manufacturer specifications. The propeller thrust coefficient was estimated, based upon the vehicles mass and the motors maximum speed. Since the vehicle is able to accelerate vertically, the thrust produced by the propellers must overcome the weight force, i.e. $C_T \Omega_{max}^2 > mg$. Since more accurate methods were not readily available, a reasonable value of C_T that satisfies this relationship was chosen. The yaw reaction torque coefficient was chosen to have the same order of magnitude as [8] and [33]. The moments of inertia around each axis are those measured for a similarly sized hexacopter using a pendulum by Capello et al. [34]. These parameters do not exactly characterise the physical platform that is to be used, however the problem of parameter uncertainty is addressed in Section 4.4.

Model Parameter	Symbol	Value	Units
Acceleration due to gravity	g	9.81	m s^{-2}
Vehicle mass	m	1.404	kg
Moment of inertia (x)	I_{xx}	0.0286	kg m^2
Moment of inertia (y)	I_{yy}	0.0254	kg m^2
Moment of inertia (z)	I_{zz}	0.0418	kg m^2
Angle between rotor arms	α	60	°
Maximum propeller speed	Ω_{max}	252	rev s^{-1}
Propeller thrust coefficient	C_T	4.5×10^{-5}	N s^2
Yaw reaction torque coefficient	K_ϕ	1.00×10^{-7}	N m s^2
Distance from centre of gravity to propeller	d	0.217	m

Table 3.1: Model simulation parameters

3.4 Chapter Summary

Within this chapter a complete state space representation has been derived to describe the dynamics of a general multi-rotor vehicle. This model has then been extended to a specific six rotor vehicle configuration. The simulation of the established model has been developed using Simulink. This simulated model will enable verification and testing of the control systems which are developed in the following chapter.

Chapter 4

Control of Multi-Rotor Aircraft

Now that a sufficiently accurate mathematical model of the hexacopter has been developed, the control problem is able to be approached. Within this chapter two control systems will be developed with the aim of controlling the aircraft position and attitude. Simulations will then be performed in Simulink to verify the effectiveness of each control system. For the purposes of these simulations it will be assumed that the vehicle's position and attitude (x, y, z, ϕ, θ, ψ) can be accurately measured at any given time.

4.1 Pseudo-inverse of Motor Speed Matrix

The control systems in this chapter will be based upon the effective inputs of the plant i.e. τ_ϕ, τ_θ, τ_ψ, F_T. Therefore, it will be necessary to implement an inverse of the matrix in Equation 3.9 in order to obtain the desired input motor speeds. This matrix is not square and is therefore not invertible. However, for a matrix A with linearly independent rows, the Penrose-Moore pseudo-inverse (A^+) is able to be computed such that:

$$A^+ = A^T(AA^T)^{-1}$$

In this case, this is a right inverse which will satisfy $AA^+ = I$ where I is an identity matrix. The pseudo-inverse matrix is shown in Equation 4.1 and will be used to reverse the relationship be-

tween the effective inputs and the propeller speeds.

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{6C_T d} & \frac{\sqrt{3}}{(6C_T d)} & \frac{1}{6K_{psi}} & \frac{1}{6C_T} \\ -\frac{1}{3C_T d} & 0 & -\frac{1}{6K_{psi}} & \frac{1}{6C_T} \\ -\frac{1}{6C_T d} & -\frac{\sqrt{3}}{(6C_T d)} & \frac{1}{6K_{psi}} & \frac{1}{6C_T} \\ \frac{1}{6C_T d} & -\frac{\sqrt{3}}{(6C_T d)} & -\frac{1}{6K_{psi}} & \frac{1}{6C_T} \\ \frac{1}{3C_T d} & 0 & \frac{1}{6K_{psi}} & \frac{1}{6C_T} \\ \frac{1}{6C_T d} & \frac{\sqrt{3}}{(6C_T d)} & -\frac{1}{6K_{psi}} & \frac{1}{6C_T} \end{bmatrix} \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \\ F_T \end{bmatrix} \tag{4.1}$$

4.2 PID Control

Initially, a control strategy for the hexacopter will be developed based upon Proportional Integral Derivative (PID) control. The basic concept of PID control is outlined in Section 2.2.1. This controller will be designed with the aim of stabilising the vehicle around a hovering point and this allows the model to be simplified.

4.2.1 Simplified Model

For the purposes of this controller, the vehicle will be assumed to be hovering, which implies the roll and pitch angles (ϕ θ) will only have small variations from the origin. These assumptions will be used in developing the controller, however the controller will be verified by simulation on the full nonlinear model. Likewise, we will assume the yaw angle (ψ) is held close to the origin. This allows the small angle approximation to be applied to the model defined in Section 3.1.2. Applying this assumption to Equation 3.1 gives Equation 4.2.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$
$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4.2}$$

Likewise, it will be assumed that the moments of inertia around each axis are approximately equal ($I_{xx} \approx I_{yy} \approx I_{zz}$). Using this assumption and the relationship in Equation 4.2, Equation 3.4 can be simplified to give Equation 4.3. This allows the rotational systems to be decoupled and therefore

more easily controlled using the PID technique.

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{xx}}\tau_{\phi} \\ \frac{1}{I_{yy}}\tau_{\theta} \\ \frac{1}{I_{zz}}\tau_{\psi} \end{bmatrix} \tag{4.3}$$

Further, by applying the small angle approximation to Eq. (3.8), the translational motion is able to be linearised. This is shown in Equation Eq. (4.4).

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\theta\frac{F_T}{m} \\ \phi\frac{F_T}{m} \\ \frac{F_T}{m} - g \end{bmatrix} \tag{4.4}$$

Under these assumptions, each Euler angle is directly controllable by the corresponding torque input, vertical translational motion is controllable by the total thrust and horizontal translational motion is controllable by the roll and pitch angles. Additionally, from Equations 4.3 and 4.4, it is apparent that there are double integrators in the plant dynamics[35]. Therefore, the integral term usually present in PID controllers will be unnecessary, i.e. the controllers designed in this section will use a PD structure.

4.2.2 Attitude Control

In order to control the orientation of the vehicle, a PD control system is used for each angle. The torque around each axis is used to control the corresponding angles. The PD control laws for each torque input are defined in Equation 4.5.

$$\begin{bmatrix} \tau_{\phi} \\ \tau_{\theta} \\ \tau_{\psi} \end{bmatrix} = \begin{bmatrix} K_{P_{\phi}}(\phi_d - \phi) + K_{D_{\phi}}(\dot{\phi}_d - \dot{\phi}) \\ K_{P_{\theta}}(\theta_d - \theta) + K_{D_{\theta}}(\dot{\theta}_d - \dot{\theta}) \\ K_{P_{\psi}}(\psi_d - \psi) + K_{D_{\psi}}(\dot{\psi}_d - \dot{\psi}) \end{bmatrix} \tag{4.5}$$

Where K coefficients represent non-negative constants and ϕ_d, θ_d and ψ_d represent the desired attitude of the vehicle.

4.2.3 Altitude Control

From Eq. (3.8), the relationship between the total thrust and acceleration in the z direction is given by: $F_T = \frac{m}{cos(\theta)cos(\phi)}(\ddot{z}+g)$. Using this, the control law for z can be defined as shown in Eq. (4.6).

$$F_T = \frac{m}{cos(\theta)cos(\phi)}[K_{P_z}(z_d - z) + K_{D_z}(\dot{z}_d - \dot{z}) + g] \tag{4.6}$$

Where K coefficients represent non-negative constants and z_d represents the desired altitude.

4.2.4 Position Control

Finally, the x and y position must be controlled to prevent the vehicle from drifting from its setpoint in the presence of disturbances, e.g wind gusts. This system should also allow the vehicle to travel to a specified coordinate within the Earth frame. Since the lateral and longitudinal motion of the vehicle cannot be directly controlled, the position is controlled by adjusting the desired values of the roll and pitch angles (ϕ_d, θ_d). The relationship between x,y, ϕ and θ is linearised in Equation 4.4. Further, since the vehicle is assumed to be hovering, the thrust force must counter the vehicle's weight i.e. $F_T \approx mg \Rightarrow \frac{F_T}{m} \approx g$.

$$\phi = \frac{1}{g}\ddot{y}$$

$$\theta = -\frac{1}{g}\ddot{x}$$

Using this linearised relationship, two PD controllers (scaled by $\pm\frac{1}{g}$) can be designed to take the position error as an input and will output the desired roll and pitch angles. These control laws are represented in Equation 4.7

$$\begin{bmatrix} \phi_d \\ \theta_d \end{bmatrix} = \begin{bmatrix} \frac{1}{g}[K_{P_y}(y_d - y) + K_{D_y}(\dot{y_d} - \dot{y})] \\ -\frac{1}{g}[K_{P_x}(x_d - x) + K_{D_x}(\dot{x_d} - \dot{x})] \end{bmatrix} \tag{4.7}$$

Where K coefficients represent non-negative constants, x_d and y_d represent the desired horizontal position in the Earth frame. This controller is added in an outer loop with its outputs being supplied to the attitude controller.

4.2.5 Complete Control System

By combining all of the previously described controllers, the vehicle is able to be controlled around the desired operating point. The overall block diagram is shown in Fig. 4.1.

The gains of each controller were tuned independently using the Simulink Design Optimization Package. The Response Optimizer app within this package uses gradient descent to tune the controller gains in order to achieve the specified performance of step response. The gains achieved using this method are given in Table 4.1.

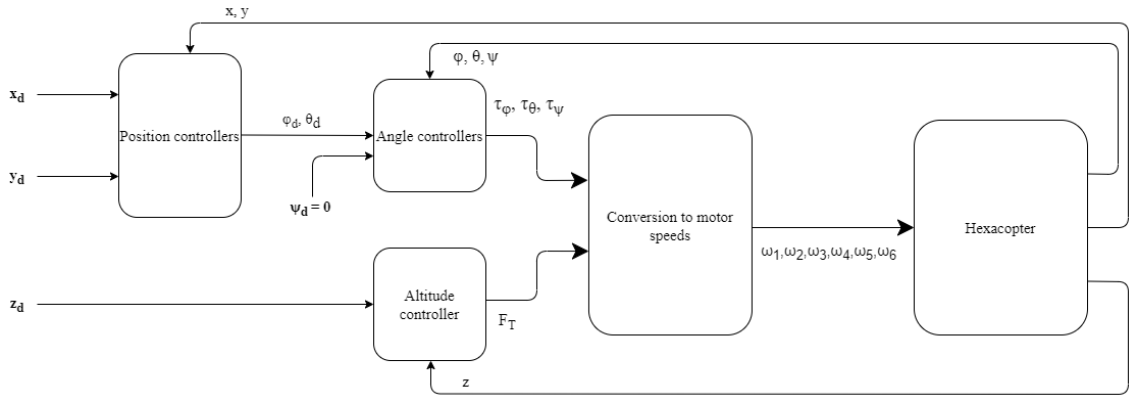


Figure 4.1: Complete PID control system block diagram

Controller	Gain	Value
Roll Angle	$K_{P,\phi}$	2.688
	$K_{D,\phi}$	2.385
Pitch Angle	$K_{P,\theta}$	2.688
	$K_{D,\theta}$	2.385
Yaw Angle	$K_{P,\psi}$	0.759
	$K_{D,\psi}$	0.664
x Position	$K_{P,x}$	3.322
	$K_{D,x}$	2.020
y Position	$K_{P,y}$	1.471
	$K_{D,y}$	1.681
z Position	$K_{P,z}$	218.4
	$K_{D,z}$	162.5

Table 4.1: PD controller gains

4.2.6 Preliminary Results

The PD control system was implemented in Simulink and its ability to stabilise the Euler angles from significant initial conditions was tested. The system simultaneously stabilises the position and counteracts any deviations from the origin, as seen in Fig. 4.2.

The system tracking capabilities were also tested with step reference inputs. The step responses for roll angle, x position and altitude can be seen in Fig. 4.3. The step response of the other angles (θ, ψ) are very similar to the roll angle response and the y position response is similar to that of the x position, therefore these responses are not shown for brevity. The control system was also shown to be reasonably effective in tracking a sinusoidal position command, as shown in Fig. 4.4.

The results show that this system is effective in stabilising the aircraft around a hovering point in the absence of major disturbances. It is also able to respond to relatively small positional step

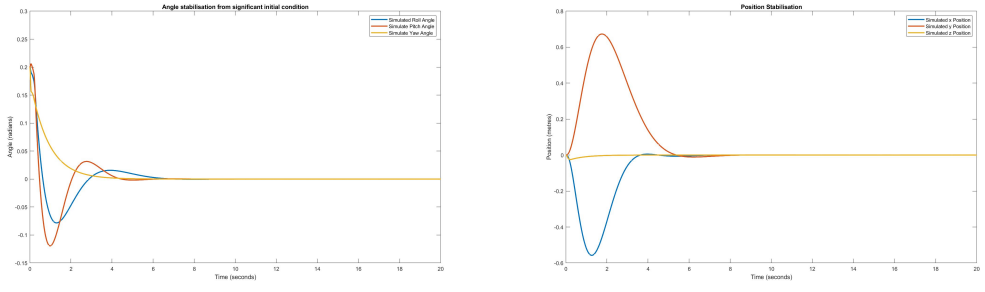


Figure 4.2: PD system stabilisation from significant initial angles

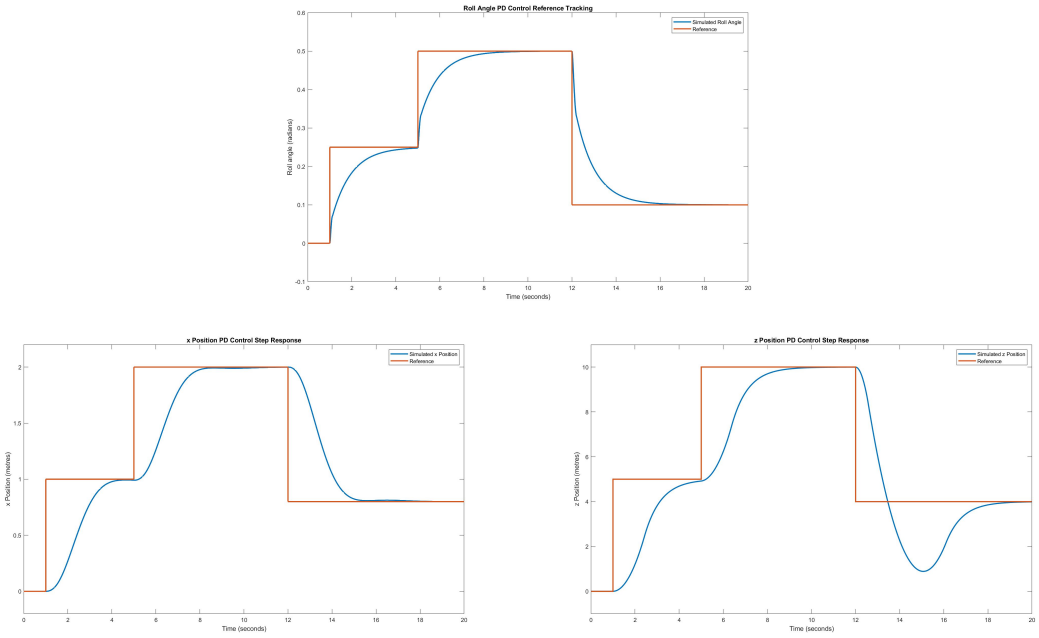


Figure 4.3: PD system step responses

commands. However, this control method is limited since it neglects the nonlinearities of the physical system. For example, it is ineffective with a non-zero yaw angle due to approximations made in the simplification of the model. The system is also prone to saturation when subject to large step commands.

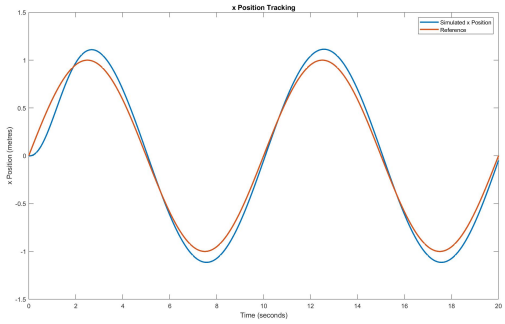


Figure 4.4: PID Position tracking

4.3 Backstepping Control

In order to improve upon the performance of the previous controller, the recursive Lyapunov-based control method known as backstepping will be applied to the multi-rotor system. The mathematical background of this method is discussed in Section 2.2.2.

4.3.1 Simplified State Space Model

The states used by the backstepping controller will be each of the positions in the Earth frame as well as their respective velocities. In order to simplify notation, the states will be referred to as x_i and are defined as:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \\ x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \end{bmatrix} \tag{4.8}$$

As seen in Section 4.2.1, the state space model which was presented in Section 3.1.2, is able to be greatly simplified by making a number of assumptions. For the purposes of this controller, the small angle approximation will be used in order to simplify the relationship between the torques and the Euler angles. That is, the relationship in Equation 4.2 will be assumed true. Applying this

to Equation 3.4 gives the following:

$$\begin{bmatrix} \dot{x}_2 \\ \dot{x}_4 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} a_1 x_4 x_6 + b_1 \tau_\phi \\ a_2 x_2 x_6 + b_2 \tau_\theta \\ a_3 x_2 x_4 + b_3 \tau_\psi \end{bmatrix} \tag{4.9}$$

with $a_1 = \frac{I_{yy} - I_{zz}}{I_{xx}}$, $a_2 = \frac{I_{zz} - I_{xx}}{I_{yy}}$, $a_3 = \frac{I_{xx} - I_{yy}}{I_{zz}}$, $b_1 = \frac{1}{I_{xx}}$, $b_2 = \frac{1}{I_{yy}}$, $b_3 = \frac{1}{I_{zz}}$. Now, the simplified state space model may be expressed:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} x_2 \\ a_1 x_4 x_6 + b_1 \tau_\phi \\ x_4 \\ a_2 x_2 x_4 + b_2 \tau_\theta \\ x_6 \\ a_3 x_2 x_4 + b_3 \tau_\psi \\ x_8 \\ [-cos(x_1)sin(x_3)cos(x_5) - sin(x_1)sin(x_5)]\frac{F_T}{m} \\ x_{10} \\ [-cos(x_1)sin(x_3)sin(x_5) + sin(x_1)cos(x_5)]\frac{F_T}{m} \\ x_{12} \\ [cos(x_1)cos(x_3)]\frac{F_T}{m} - g \end{bmatrix} \tag{4.10}$$

4.3.2 Attitude Control

Consider the tracking error of the roll angle, i.e. the first state:

$$z_1 = x_{1d} - x_1$$

Choose a candidate Lyapunov function such that it is positive definite:

$$V(z_1) = \frac{1}{2} z_1^2$$

Taking its derivative gives:

$$\dot{V}(z_1) = z_1 \dot{z}_1$$

A control Lyapunov function must have a negative definite derivative, therefore $\dot{z}_1$ is chosen to be

$-K_1 z_1$ with constant gain $K_1 > 0$. This gives the following result:

$$\dot{V}(z_1) = -K_1 z_1^2$$

Now, take the derivative of z_1 and substitute our chosen value.

$$\begin{aligned} \dot{z_1} &= \dot{x}_{1d} - \dot{x}_1 \\ -K_1 z_1 &= \dot{x}_{1d} - x_2 \end{aligned}$$

Now, choose a virtual control x_{2d} to satisfy this relationship:

$$x_{2d} = \dot{x}_{1d} + K_1 z_1$$

Now consider the tracking error of the second state:

$$\begin{aligned} z_2 &= x_{2d} - x_2 \\ &= \dot{x}_{1d} + K_1 z_1 - x_2 \end{aligned}$$

Now choose a second candidate Lyapunov function:

$$V(z_1, z_2) = \frac{1}{2}(z_1^2 + z_2^2)$$

Then take its derivative:

$$\begin{aligned} \dot{V}(z_1, z_2) &= \dot{V}(z_1) + z_2 \dot{z}_2 \\ &= -K_1 z_1^2 + z_2 \dot{z}_2 \end{aligned}$$

Now to ensure $\dot{V}(z_1, z_2)$ is negative definite, choose $\dot{z}_2 = -K_2 z_2$ with constant $K_2 > 0$. This gives the following result:

$$\begin{aligned} \dot{z}_2 &= \ddot{x}_{1d} + K_1 \dot{z}_1 - \dot{x}_2 \\ -K_2 z_2 &= \ddot{x}_{1d} + K_1(\dot{x}_{1d} - x_2) - a_1 x_4 x_6 - b_1 \tau_\phi \end{aligned}$$

Using this result, it is possible to choose a control value for the input τ_ϕ which will stabilise the roll angle:

$$\tau_\phi = \frac{1}{b_1}(\ddot{x}_{1d} + K_1(\dot{x}_{1d} - x_2) - a_1 x_4 x_6 + K_2 z_2) \tag{4.11}$$

The same steps can be followed to stabilise both the pitch and yaw angles with control inputs τ_θ and τ_ψ respectively. These control laws are given in Eq. (4.12).

$$\begin{aligned} \tau_\theta &= \frac{1}{b_2}(\ddot{x}_{3d} + K_3(\dot{x}_{3d} - x_4) - a_2 x_2 x_6 + K_4 z_4) \\ \tau_\psi &= \frac{1}{b_3}(\ddot{x}_{5d} + K_5(\dot{x}_{5d} - x_6) - a_3 x_2 x_4 + K_6 z_6) \end{aligned} \tag{4.12}$$

4.3.3 Position Control

The control laws developed in the previous section successfully allow stabilisation and tracking of the vehicle's attitude, however, the objective of this control system is to allow the vehicle to track a trajectory in 3D space. First consider the tracking error of the position in the Earth frame x direction:

$$z_7 = x_{7d} - x7$$

Following the same logic presented in the development of the attitude controllers, the candidate Lyapunov function is chosen as $V(z_7) = \frac{1}{2}z_7^2$. Thus, a virtual control for x_8 is chosen to satisfy $\dot{z}_7 = -K_7 z_7$:

$$x_{8d} = \dot{x}_{7d} + K_7 z_7$$

Now, consider the tracking error of the translational velocity, z_8, and its derivative:

$$\begin{aligned}
z_8 &= x_{8d} - x_8 \\
z_8 &= \dot{x}_{7d} + K_7 z_7 - x_8 \\
\dot{z}_8 &= \ddot{x}_{7d} + K_7 \dot{z}_7 - \dot{x}_8 \\
&= \ddot{x}_{7d} + K_7 \dot{z}_7 - [-cos(x_1)sin(x_3)cos(x_5) - sin(x_1)sin(x_5)]\frac{F_T}{m}
\end{aligned}$$

In order to ensure $\dot{z}_8 = -K_8 z_8$, a virtual control must be chosen. If it is assumed that the vehicle is in hovering flight and the Euler angles are stabilised around 0°, then the most logical choice of variable for a virtual control is the pitch angle (x_3). Thus, the virtual control x_{3d} in Equation 4.13 will be supplied as a desired value for the attitude controller developed in the previous section.

$$x_{3d} = sin^{-1}\left(-\frac{\frac{m}{F_T}[K_8 z_8 + \ddot{x}_{7d} + K_7 \dot{z}_7] + sin(x_1)sin(x_5)}{cos(x_1)cos(x_5)}\right) \tag{4.13}$$

The y position control law, shown in Eq. (4.14), is developed in the same way, with the roll angle (x_1) acting as the virtual control.

$$x_{1d} = sin^{-1}\left(\frac{\frac{m}{F_T}[K_{10} z_{10} + \ddot{x}_{9d} + K_9 \dot{z}_9] + cos(x_1)sin(x_3)sin(x_5)}{cos(x_5)}\right) \tag{4.14}$$

The z position control law in Eq. (4.15) is developed following the same method, with F_T as the control input.

$$F_T = \frac{m(\ddot{x}_{11d} + K_{11}\dot{z}_{11} + K_{12} z_{12} + g)}{cos(x_1)cos(x_3)} \tag{4.15}$$

4.3.4 Preliminary Results

The gains K_i were tuned using the Simulink Design Optimization Package. This controller was able to stabilise the angles and positions from significant initial conditions. It was also able to travel between waypoints in 3D space. Fig. 4.7 shows the hexacopter's path: taking off, travelling between waypoints and landing at a given location. However, this controller's response to large step inputs is unpredictable due to saturation effects. Also, note the slight steady state error in the z controller shown in Fig. 4.6.

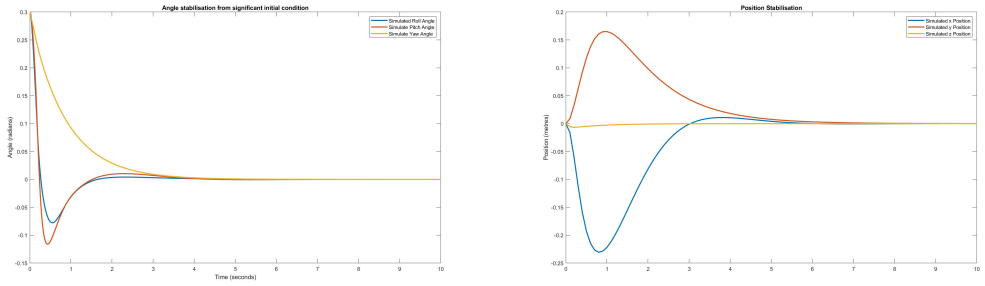


Figure 4.5: Backstepping system stabilisation from significant initial angles

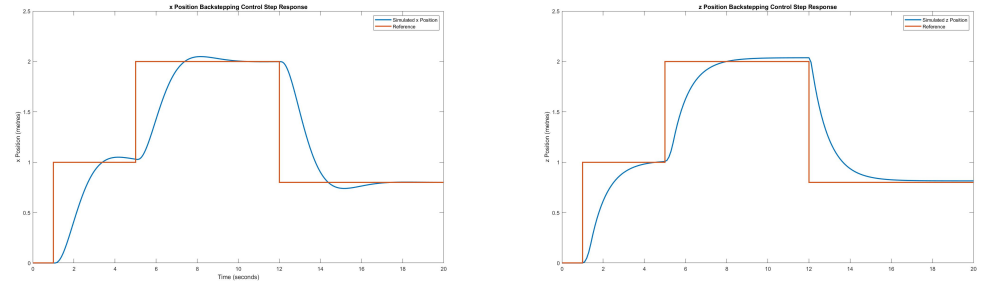


Figure 4.6: Backstepping position tracking

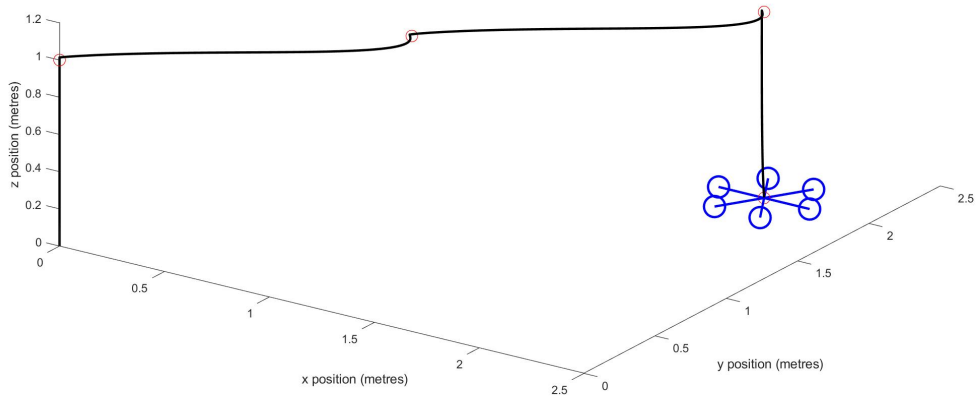


Figure 4.7: Backstepping 3D waypoint tracking

4.4 Backstepping with Integral Action

As an extension, to reduce the effects of uncertainties in model parameters it is desirable to include integral action within the controller. This method has been shown to be effective at reducing the effects of uncertainties and increasing the robustness of the controller in both simulation [36] and hardware [8] implementation.

4.4.1 Position Control

This controller will use the angle controllers developed in the previous section, only the position controllers will be adjusted to include integral action. This controller will also be derived by considering the position vector rather than individual components.

Consider the position and velocity error vectors:

$$\begin{aligned} \mathbf{e}_p &= \mathbf{p}_d - \mathbf{p} = \begin{bmatrix} x_d - x \\ y_d - y \\ z_d - z \end{bmatrix} \\ \mathbf{e}_v &= \mathbf{v}_d - \dot{\mathbf{p}} = \begin{bmatrix} v_{d,x} - \dot{x} \\ v_{d,y} - \dot{y} \\ v_{d,z} - \dot{z} \end{bmatrix} \end{aligned} \tag{4.16}$$

where $v_{d,x}$, $v_{d,x}$ and $v_{d,x}$ are virtual controls representing the desired translational velocities.

Choose a candidate Lyapunov function including an integral term:

$$V = \frac{1}{2}\left(e_p^T e_p + e_v^T e_v + k\Gamma^T\Gamma\right) \tag{4.17}$$

where $\Gamma = \int_0^t e_1(\gamma)d\gamma$ and k is a diagonal 3x3 matrix of non-negative constants. Taking the time derivative gives:

$$\dot{V} = e_p^T\dot{e}_p + e_v^T\dot{e}_v + k\Gamma^T\dot{\Gamma} \tag{4.18}$$

Now choose $\dot{V}$ such that it is negative definite:

$$\dot{V} = -ce_p^T e_p - \kappa e_v^T e_v \tag{4.19}$$

where c and κ represent diagonal 3x3 matrices of non-negative constants.

Now consider the derivative terms:

$$\begin{aligned} \dot{e}_p &= \dot{p}_d - \dot{p} \\ &= \dot{p}_d - v_d \\ \\ \dot{e}_v &= \dot{v}_d - \ddot{p} \\ \\ \dot{\Gamma} &= e_p \end{aligned} \tag{4.20}$$

The virtual control v_d is then chosen such that Eq. (4.18) satisfies Eq. (4.19):

$$v_d = \dot{p}_d + ce_p + k\Gamma$$

Substituting into the expression for e_v in Eq. (4.16) gives:

$$\begin{aligned} e_v &= \dot{\mathbf{p}}_d + ce_p + k\Gamma - \dot{\mathbf{p}} \\ e_v - ce_p - k\Gamma &= \dot{\mathbf{p}}_d - \dot{\mathbf{p}} \\ \dot{e_p} &= e_v - ce_p - k\Gamma \end{aligned} \tag{4.21}$$

Taking the time derivative of e_v gives:

$$\begin{aligned} \dot{e}_v &= \ddot{p}_d + c\dot{e}_p + ke_p - \ddot{\mathbf{p}} \\ \dot{e}_v &= \ddot{p}_d + c(e_v - ce_p - k\Gamma) + ke_p - \ddot{\mathbf{p}} \\ \dot{e}_v &= \ddot{p}_d + ce_v - c^2 e_p - ck\Gamma + ke_p - \ddot{\mathbf{p}} \end{aligned} \tag{4.22}$$

Note that an expression for the second derivative of translational position ($\ddot{\mathbf{p}}$) is given in Section 3.1.3, Eq. (3.8). Also note that in order to satisfy Eq. (4.19), $\dot{e}_v$ is chosen as:

$$\dot{e}_v = -\kappa e_v - e_p \tag{4.23}$$

Finally, equating Eq. (4.22) with Eq. (4.23) allows the control laws to be derived:

$$\begin{aligned} -\kappa e_v - e_p &= \ddot{p}_d + c e_v - c^2 e_p - ck\Gamma + k e_p - \ddot{\mathbf{p}} \\ \ddot{\mathbf{p}} &= \ddot{p}_d + c e_v - c^2 e_p - ck\Gamma + k e_p + \kappa e_v + e_p \\ \ddot{\mathbf{p}} &= \ddot{p}_d + (1 + k - c^2) e_p + (c + \kappa) e_v - ck\Gamma \end{aligned} \tag{4.24}$$

The control laws are then derived using θ_d, ϕ_d and F_T as the control inputs.

$$\begin{aligned} \theta_d &= sin^{-1}\left(\frac{-\frac{m}{F_T}[\ddot{x}_d + (1+k_1-c_1^2)e_x + (c_1+\kappa_1)e_{v,x} - c_1 k_1 \Gamma_1] - sin(\phi)sin(\psi)}{cos(\phi)cos(\psi)}\right) \\ \phi_d &= sin^{-1}\left(\frac{\frac{m}{F_T}[\ddot{y}_d + (1+k_2-c_2^2)e_y + (c_2+\kappa_2)e_{v,y} - c_2 k_2 \Gamma_2] + cos(\phi)sin(\theta)sin(\psi)}{cos(\psi)}\right) \\ F_T &= \frac{m(\ddot{z}_d(1+k_3-c_3^2)e_z + (c+\kappa_3)e_{v,z} - c_3 k_3 \Gamma_3 + g)}{cos(\phi)cos(\theta)} \end{aligned} \tag{4.25}$$

4.4.2 Preliminary Results

The position and altitude tracking results are shown in Fig. 4.8. The angle controllers are unchanged from the previous section, thus the angle stabilising result is omitted in this section.

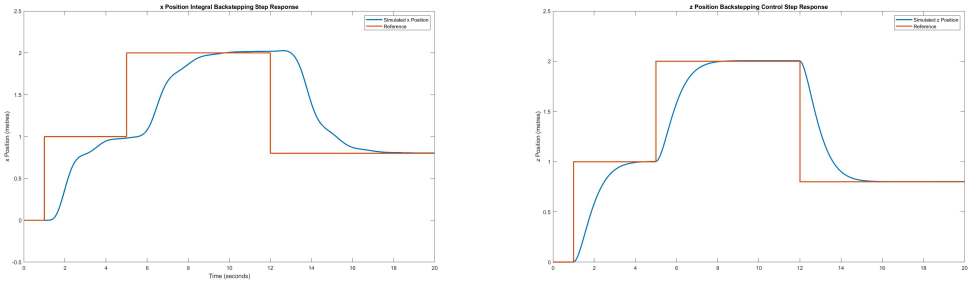


Figure 4.8: Integral Backstepping position tracking

A persisting limitation with this control method is its inability to stabilise when presented with a large step input. Fig. 4.9 shows the system response to a ten metre step command in the x direction.

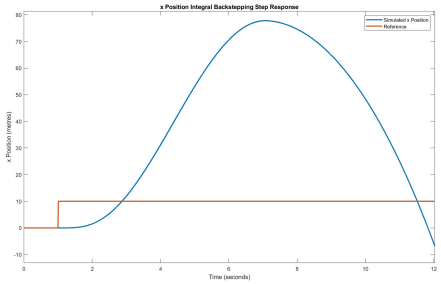


Figure 4.9: Integral backstepping response to a large step input

4.5 Actuator Saturation

To prevent actuator saturation when large step commands are given, a sigmoid function can be used to limit the effect of position error [37]. For this purpose, the sigmoid function is defined as:

$$\sigma(x) = p_{max} \frac{x}{1+|x|}$$

This sigmoid function is used to limit the output of the x and y position controllers prior to the arcsine function. The parameter p_{max} is chosen as 0.77 to limit the desired roll and pitch angles to $\pm 50^{\circ}$. This greatly improves the controllers response to large step inputs, as can be seen by comparing the ten metre step response in Fig. 4.10 with Fig. 4.9.

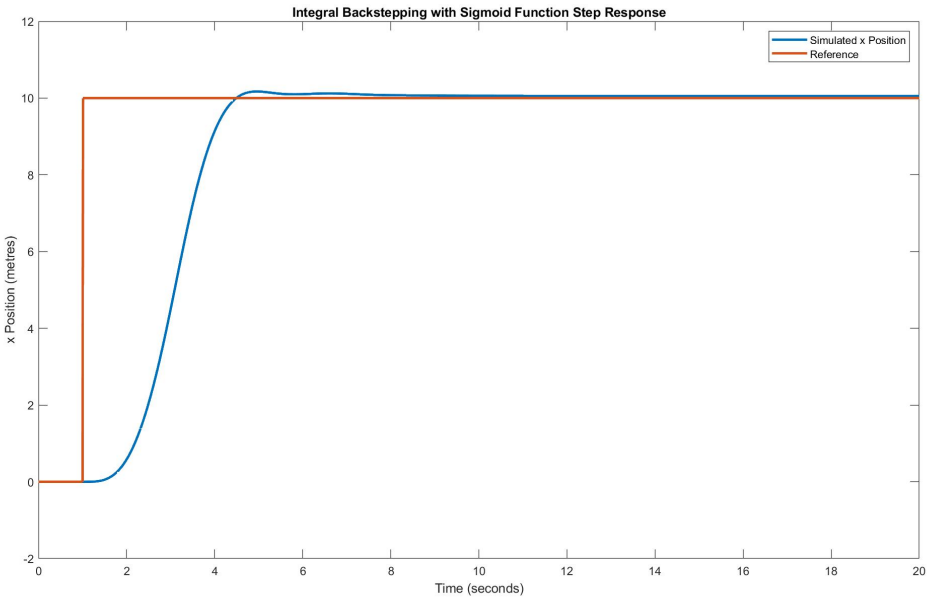


Figure 4.10: Integral Backstepping position tracking with sigmoid function

4.6 Final Control System

The final control system was developed progressively and documented throughout this chapter. The attitude control laws are based upon backstepping control and are those displayed in Eq. (4.11) and Eq. (4.12). The position controllers include the addition of integral action in the backstepping control law, as shown in Eq. (4.25). The Simulink model for the entire control system and each subsystem is shown in Appendix A.3.

Again the constant gain values were tuned using the Simulink Design Optimization Package. The final gains that were used for this system are given in Table 4.2.

Controller	Gain	Value
Roll Angle	K_1	1.65
	K_2	1.65
Pitch Angle	K_3	2.21
	K_4	2.21
Yaw Angle	K_5	3.11
	K_6	3.11
x Position	c_1	3.67
	k_1	0.01
	κ_1	1.57
y Position	c_2	3.92
	k_2	0.01
	κ_2	1.35
z Position	c_3	2.50
	k_3	0.01
	κ_3	1.00

Table 4.2: Final control system gains

This controller allows the vehicle to follow waypoint commands. The results in Fig. 4.11 show the path of the simulated vehicle as it travels between four waypoints - separated by 5 metres - before returning to its starting position and landing. The total simulated time for this flight was 40 seconds.

To investigate the robustness of this controller, process noise was added to the model to simulate effects which were not previously accounted for e.g. wind. Gaussian noise with zero mean and variance of 0.1 was added to the body frame translational motion (u, v) within the model. The same waypoints were given to the control system, and the resulting flight path is shown in Fig. 4.12. This demonstrates the ability of the control system to handle both external disturbances and model uncertainties.

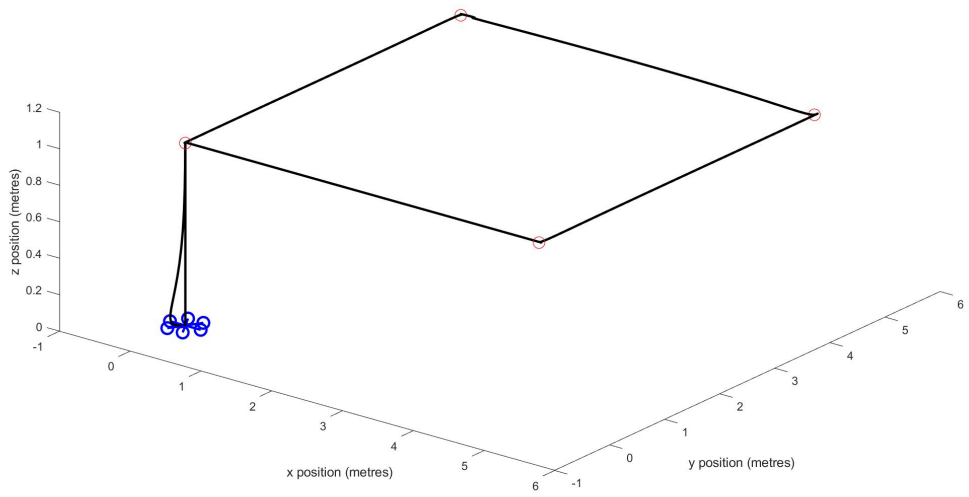


Figure 4.11: Final control system waypoint tracking.

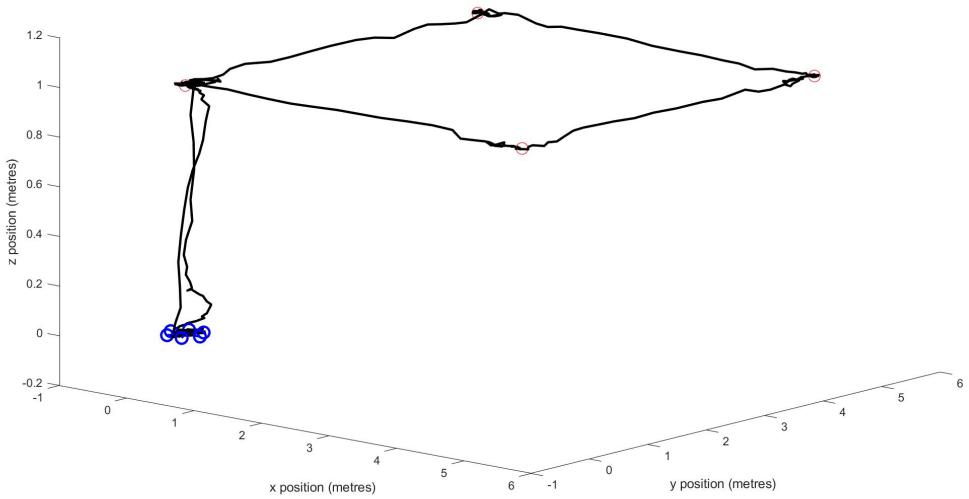


Figure 4.12: Final control system waypoint tracking with process noise.

4.7 Chapter Summary

In this chapter, three control systems were developed and tested in simulation. The PID control system was effective at stabilising the aircraft around a hovering point, but had limitations due to not considering the nonlinearities of the system. The backstepping control system improved upon this by accounting for the nonlinearities, however this controller still depended on accurate values of the model parameters. The final controller implemented backstepping control with integral action and was effective in adjusting to errors in the given model parameters, however it was prone to actuator saturation with large step commands. The final addition to the control system implemented a sigmoid function in the position control laws to mitigate the effects of actuator saturation. These control systems were all developed with the assumption that all of the system states are available. The next section will address how the states are estimated using sensor data.

Chapter 5

State Estimation and Sensor Fusion

This chapter describes the algorithms used for estimating the aircraft state from the available sensor readings. It is common for state estimation algorithms to rely on GPS systems for position data, however there are many situations where GPS may either be unavailable or unreliable. This chapter explores a state estimation technique which utilises an optical flow sensor for relative position data. This is then compared with a GPS-enabled algorithm.

5.1 Filter Algorithm

The algorithm chosen to fuse sensor data and estimate states is based upon an Extended Kalman Filter (EKF). The fundamentals of EKFs are discussed in Section 2.3.1. The measurement functions will be slightly different for each of the two following algorithms, as these functions are dependent on the sensors being used. However, the state transition functions are common to both algorithms.

5.1.1 State Transition Functions

The model developed in Chapter 3 could theoretically be used in defining the state transition functions of the system. However, this would require knowing the torque values around each axis and the total thrust produced by the propeller in real-time. In practice, accurate values for these inputs are not easily measured. Therefore, a separate model is used by considering body-fixed acceleration and angular rates as the inputs.

The states estimated by this EKF are those used by the backstepping controller, as expressed in Eq. (4.8). Namely, the positions, velocities, Euler angles and angular rates all expressed with respect to the Earth frame.In order to simplify notation, the states may be grouped into the fol-

lowing 4 vectors: $\boldsymbol{p}=\begin{bmatrix} x & y & z \end{bmatrix}^T$, $\boldsymbol{v}=\begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$, $\Phi=\begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ and $\omega=\begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$. Both algorithms will utilise accelerometer and gyroscope measurements and these measurements are defined as inputs in order to reduce the complexity of the state transition functions [30]. Thus the inputs are $\boldsymbol{u} = \begin{bmatrix} \tilde{a}_x & \tilde{a}_y & \tilde{a}_z & \tilde{\omega}_x & \tilde{\omega}_y & \tilde{\omega}_z \end{bmatrix}^T$. The state transition functions used for the prediction step of the EKF are defined in discrete time:

$$\begin{aligned}
\boldsymbol{p}_{k+1} &= \boldsymbol{p}_k + \Delta t \boldsymbol{v}_k \\
\boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \Delta t \left(C_b^n \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right) \\
\Phi_{k+1} &= \Phi_k + \Delta t \omega_k \\
\omega_{k+1} &= \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)sec(\theta) & cos(\phi)sec(\theta) \end{bmatrix} \begin{bmatrix} u_4 \\ u_5 \\ u_6 \end{bmatrix}
\end{aligned} \tag{5.1}$$

5.2 State Estimation with GPS

In industry and commercial applications UAV position is commonly estimated with the use of a GPS sensor, along with additional sensors. This method has the advantage of providing accurate position data that not many other sensors can offer.

5.2.1 Sensor Models

GPS data is recorded as latitude and longitude positions. These can be converted into the Earth frame by using the Haversine formula, as described in Section 2.1.3, with the relationship given in Eq. (2.2). The GPS data will be modelled by the following:

$$\begin{aligned}
\tilde{\lambda} &= \lambda + \mu_\lambda \\
\tilde{\chi} &= \chi + \mu_\chi
\end{aligned} \tag{5.2}$$

where λ and χ represent latitude and longitude respectively and the μ terms represent measurement noise

A standard Inertial Measurement Unit (IMU) contains both a 3-axis accelerometer and a 3-axis

gyroscope. These sensors are modelled by the following equations:

$$\begin{aligned}\begin{bmatrix}\tilde{a}_x\\ \tilde{a}_y\\ \tilde{a}_z\end{bmatrix} &= \begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} + C_n^b \begin{bmatrix}0\\ 0\\ g\end{bmatrix} + \begin{bmatrix}\mu_{a_x}\\ \mu_{a_y}\\ \mu_{a_z}\end{bmatrix}\\ \begin{bmatrix}\tilde{\omega}_x\\ \tilde{\omega}_y\\ \tilde{\omega}_z\end{bmatrix} &= \begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}\mu_{\omega_x}\\ \mu_{\omega_y}\\ \mu_{\omega_z}\end{bmatrix}\end{aligned} \tag{5.3}$$

A magnetometer is another commonly used sensor in inertial navigation systems. The magnetometer measures the Earth's magnetic field in order to estimate vehicle orientation. The measurements can be modelled by the following:

$$\begin{bmatrix}\tilde{m}_{x,b}\\ \tilde{m}_{y,b}\\ \tilde{m}_{z,b}\end{bmatrix} = C_n^b \begin{bmatrix}m_x\\ m_y\\ m_z\end{bmatrix} + \begin{bmatrix}\mu_{m_x}\\ \mu_{m_y}\\ \mu_{m_z}\end{bmatrix} \tag{5.4}$$

Where m_x, m_y and m_z represent the magnetic field vector components at the vehicle's location, and $\tilde{m}_{x,b}$, $\tilde{m}_{y,b}$ and $\tilde{m}_{z,b}$ are the magnetometer measurements in the vehicle body frame. To perform orientation determination, a known magnetic field vector at the location of the vehicle is required. For the purposes of this project, it will be assumed that the vehicle is flying outdoors without significant magnetic interference nearby, i.e. the Earth's magnetic field is the only magnetic field detected.

A barometer (or altimeter) is a sensor which measures air pressure and can be used to estimate relative altitude. The model for a barometric sensor is based upon the standard atmospheric model described in Section 2.3.3. This sensor model is given in Eq. (5.5).

$$\tilde{P} = P_0 exp\left[\frac{-gM(z+h_0)}{RT_0}\right] + \mu_P \tag{5.5}$$

where $\tilde{P}$ represents the measured pressure. In the model, z represents the height above the surface from which it launched, thus h_0 is added to account for the surface's altitude above sea level.

5.2.2 Preliminary Results

The EKF was tested in simulation with a number of steps in the x direction and a single step in the z direction. In these results, the control system developed in Chapter 4 is used. The controller is provided with the actual states rather than the estimated states. Additionally, the sample rate of all sensors is assumed to be 100 Hz.

The estimated states are compared with the actual states in Fig. 5.1. The angular rates and translational positions are estimated accurately, however there is some error in the translational velocities and Euler angle estimates. None of the measurement functions directly involve the velocity states, therefore they are essentially estimated based upon the state transition functions alone, which limits the accuracy of the estimates. The Euler angles are included in both the magnetometer and accelerometer functions in the form of the rotation matrix (C_n^b). However, the measurement functions do not directly measure the angles, i.e. there are terms such as $cos(\phi)sin(\phi)cos(\psi)$ within the function. This results in the estimated angles being correlated, i.e. a change in θ results in the estimates for ϕ and ψ also being affected.

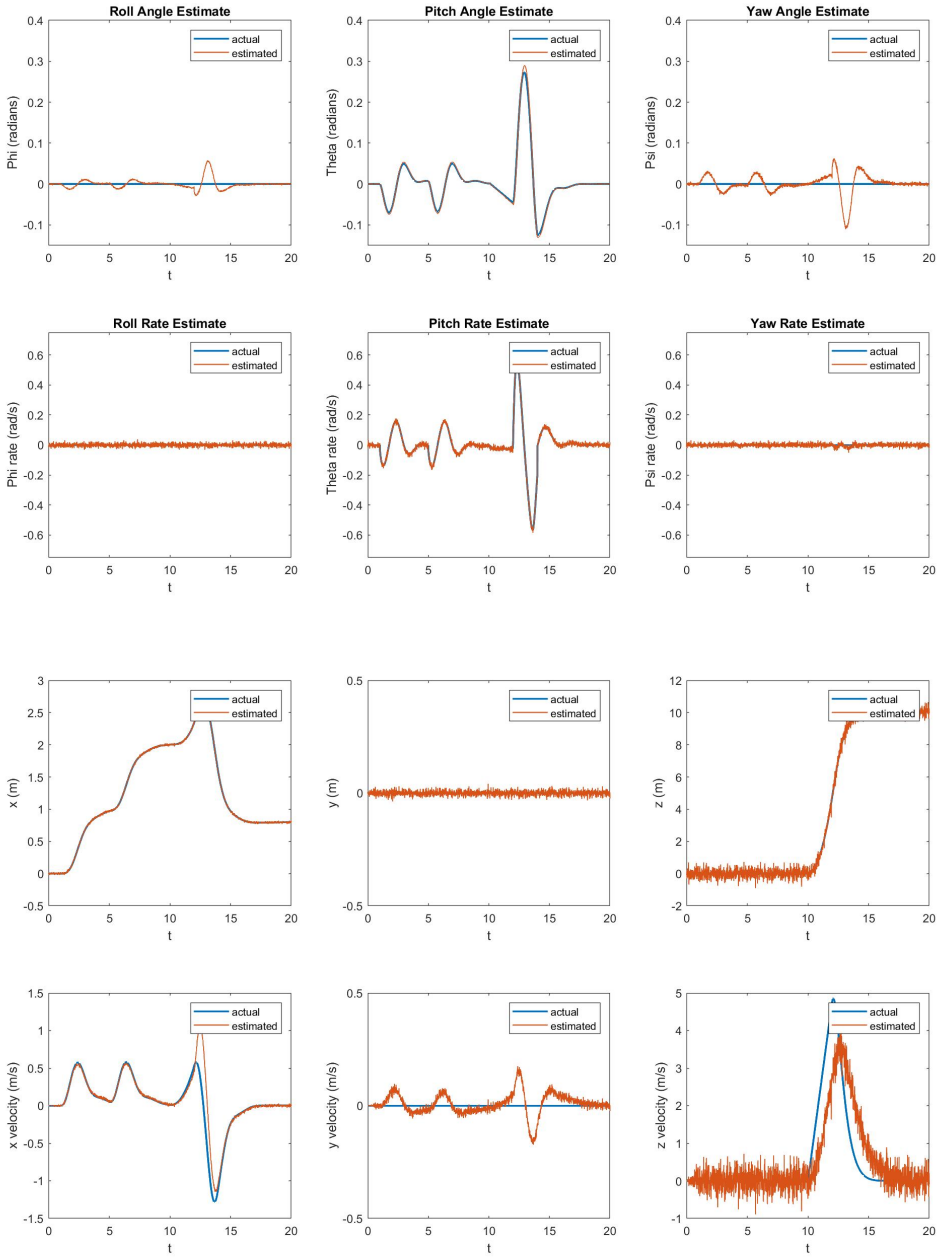


Figure 5.1: Estimated states using GPS extended Kalman filter.

5.3 State Estimation without GPS

There are many situations in which GPS signal may not be reliable. For example, GPS signal will generally be unreliable for indoor applications. Additionally, failure of the onboard GPS system could prove catastrophic if there is not another way of tracking position and velocity data.

5.3.1 Sensor Models

One type of sensor which is relatively inexpensive and can be utilised for position tracking in GPS-limited environments, is the optical flow sensor (OFS). An OFS is essentially a simple camera which compares consecutive frames to establish the motion that has occurred between frames. The OFS outputs data relating to the flow of pixels between frames. The velocity in m/s can be estimated by combining this with knowledge of the scene depth, i.e. the distance from the surface. In most cases, an OFS will be accompanied by either a lidar or sonar sensor in order to estimate scene depth. For the purposes of this model, the z position of the aircraft represents its height above the ground, which implies the terrain within the flight path is flat. The optical flow sensor and its accompanying lidar sensor are modelled by the following equations[30][38]:

$$\begin{aligned} \tilde{\rho}_x &= -\left(\frac{u}{h}+q\right)\Delta t_\rho f + \mu_{\rho_x} \\ \tilde{\rho}_y &= -\left(\frac{v}{h}+p\right)\Delta t_\rho f + \mu_{\rho_y} \\ \tilde{h} &= \frac{z}{cos(\phi)cos(\theta)} + \mu_h \end{aligned} \tag{5.6}$$

where Δt_ρ is the time between consecutive frames, f is the focal length in pixels, h is the scene depth and μ variables represent Gaussian noise with zero mean.
This algorithm also uses the IMU and magnetometer sensor models described in Section 5.3.1, but does not use the GPS or barometer.

5.3.2 Preliminary Results

This algorithm was tested in simulation using the same input reference as the previous algorithm. The results are shown in Fig. 5.2. It can be seen that the results are comparative to the previous GPS-enabled algorithm. The x and y velocity estimates are more accurate since these are measured by the optical flow sensor. However, the x and y position estimates are prone to drift, as the sensors do not directly measure these states. Also, the further the aircraft is above the surface, the larger the effect of noise on the velocity readings. This effect may make this method inappropriate for use at high altitudes.

Fig. 5.3 shows the estimated translational states while the vehicle hovers at a fixed point 10 meters above the origin. This demonstrates the drift of position estimates. However, over 1000 seconds the drift is limited to approximately ± 0.25 metres. If there is significant sensor bias, i.e. the noise has a non-zero mean, this drift could become more significant.

Fig. 5.4 visually compares the vehicle's simulated position with its estimated position in 3D space, whilst moving between a number of waypoints. It can be seen that, although there is no GPS data available a fairly accurate position estimate can be maintained even whilst the vehicle is tracking waypoints.

A more accurate and reliable EKF would utilise all of the previously mentioned sensors i.e. GPS, barometer, OFS and lidar.

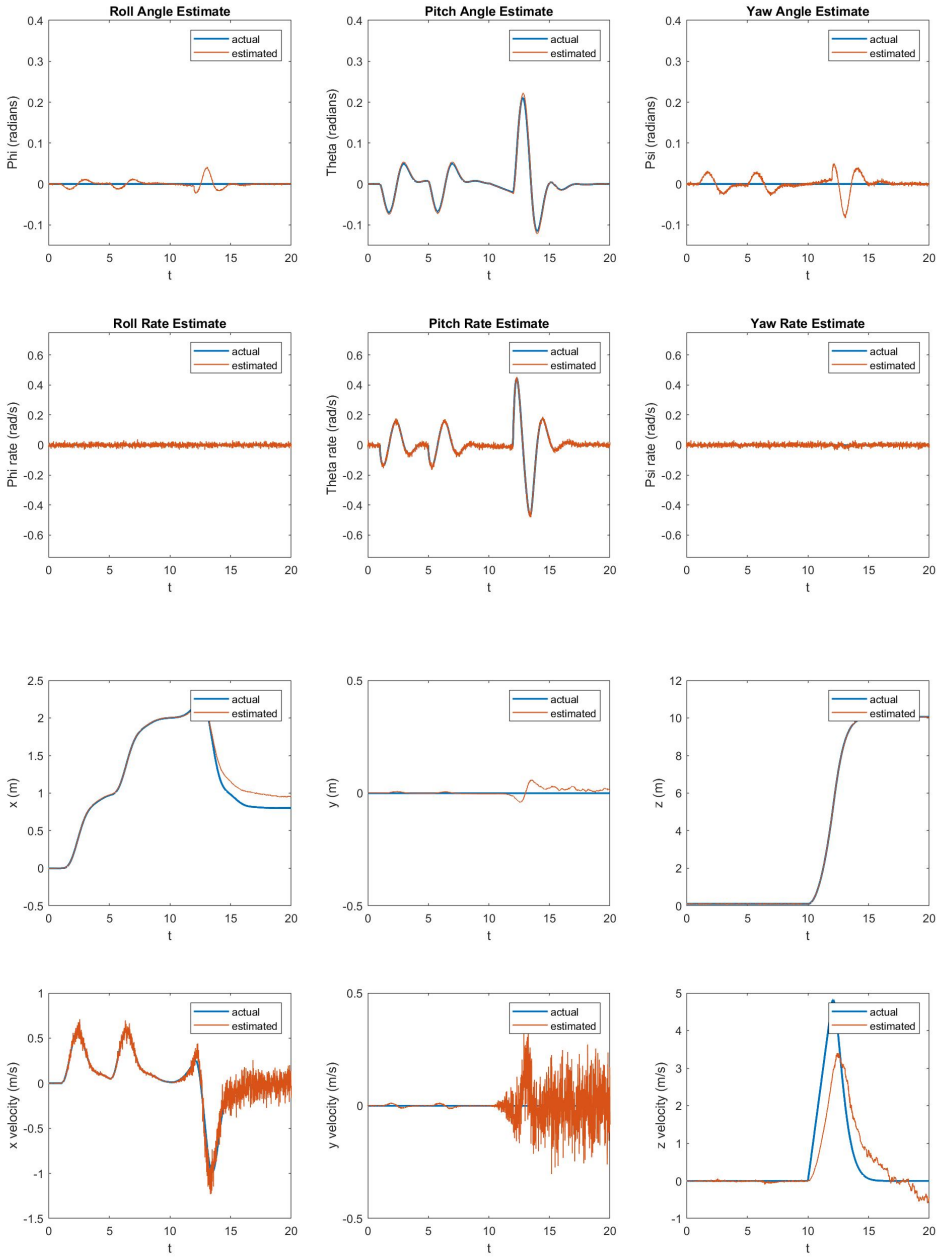


Figure 5.2: Estimated states using OFS extended Kalman filter.

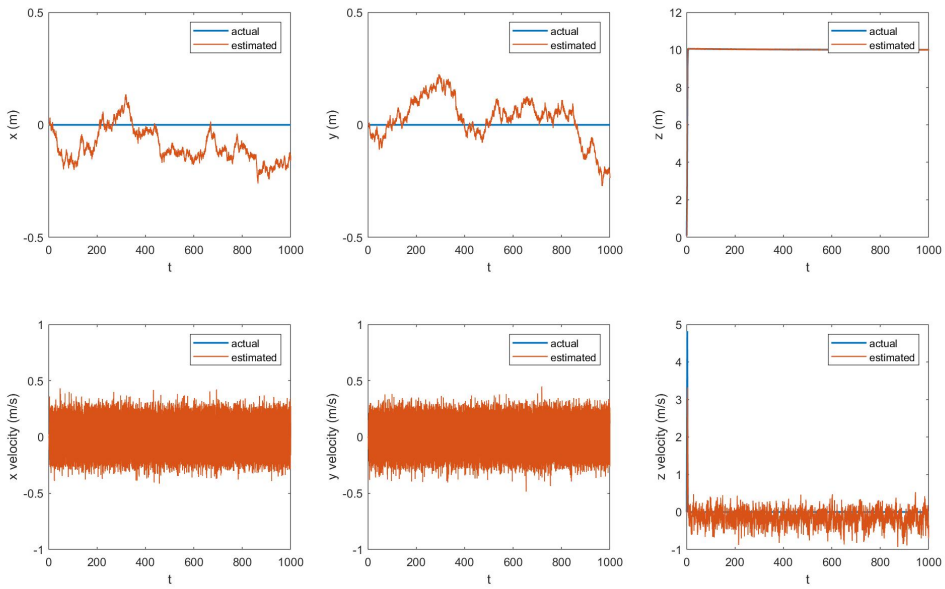


Figure 5.3: Estimated translational states over 1000 seconds.

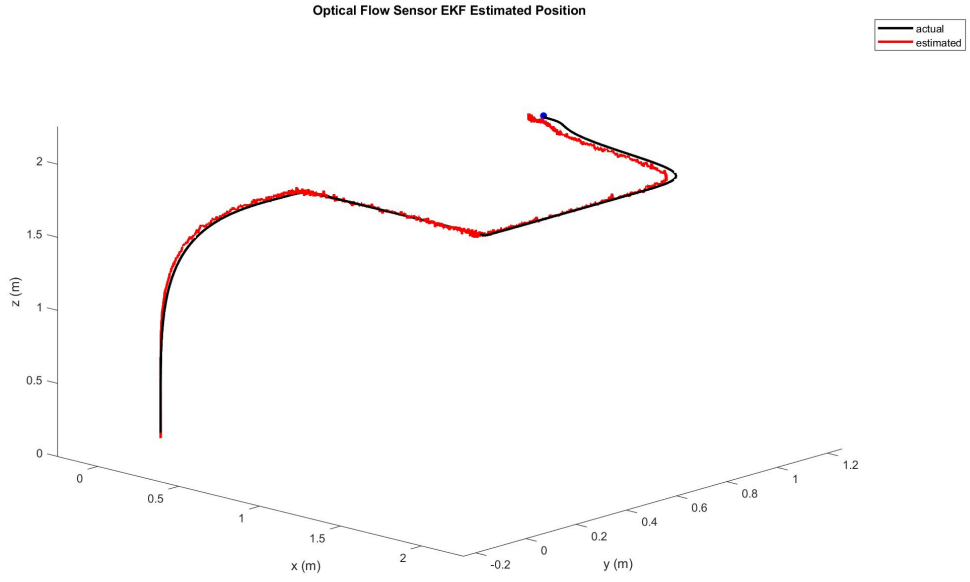


Figure 5.4: Estimated position of the vehicle in 3D space while moving between waypoints over a period of 20 seconds.

5.4 Chapter Summary

This chapter developed two sensor fusion algorithms based on the extended Kalman filter. The first algorithm utilises GPS data to estimate position. The second estimator utilises an optical flow sensor and does not rely on GPS. This has the advantage of allowing UAV operation in areas which have limited GPS signal, but is limited to operating with unobstructed line of sight to the ground. A more reliable sensor fusion algorithm would include the use of both groups of sensors. The next chapter will implement these algorithms with real sensor data gathered from test flights of a UAV platform.

Chapter 6

Implementation

This chapter outlines the specifications of the hardware platform to be used for flight testing. The state estimation algorithms developed in the previous chapter are now tested using sensor data from a number of flight tests.

6.1 Hardware

In addition to the simulation results, test flights were conducted with a custom hexacopter with the aim of collecting data to verify the EKF algorithm. The hexacopter frame is based upon a commercially available kit known as the Storm Drone 6, along with a custom PCB structure for power distribution. Power is supplied by a 5500 mAh 3S Lithium Polymer (LiPo) battery with a nominal voltage of 11.1 V. The battery voltage is regulated at 5 V to supply the flight controller. The propellers are driven by 1260 kv brushless DC motors. The speed of the motors is controlled by a PWM signal supplied to the associated electronic speed controllers (ESC).

6.1.1 The Cube Autopilot

The drone is controlled by a commercial flight controller known as The Cube Autopilot (formerly known as Pixhawk 2.1). The Cube runs a 32-bit STM32F427 ARM Cortex M4 core, has 256 kB of RAM and 2 MB of flash. It also has a co-processor (STM32F103), which handles failure conditions. It has a number of connectivity options for peripherals (e.g. UART, I2C, CAN buses), as well as 14 pulse width modulation (PWM) outputs for interfacing with actuators. This platform also has a microSD card slot for storage of flight data logs.

6.1.2 ArduPilot Software

The vehicle is controlled using the open source flight software Ardupilot, with the specific version being ArduCopter 4.1. The flight software is run on a Linux-based real-time operating system (RTOS) known as ChibiOS. The code base is mostly written in C++. State estimation is performed via an EKF-based algorithm. The control system is uses cascaded PID control similar to the controller developed in Section 4.2. There are also a number of flight modes which are able to be selected for various purposes, i.e. for autonomous flight or radio-controlled flight.

6.1.3 Companion Computer

Additionally, a computer-on-module known as the Intel Edison was added to the Cube flight controller in order to enhance the system capabilities and enable wireless communication. The Edison has a 500 MHz dual-core processor, 1 GB RAM, 4 GB storage and dual-band (2.4 and 5 GHz) WiFi connectivity. The Edison is connected via a designated 70-pin Hirose DF40 connector inside the case of the Cube. Communication with the ArduPilot software is via the MAVLink communication protocol.

APSync, an open source software package developed for use with ArduPilot, was installed on the Edison. This included a Linux operating system as well as a number of python and DroneKit packages. DroneKit is an open source API which allows python scripts to be run on flight control hardware. The relationship between the different systems is shown in Fig. 6.1.

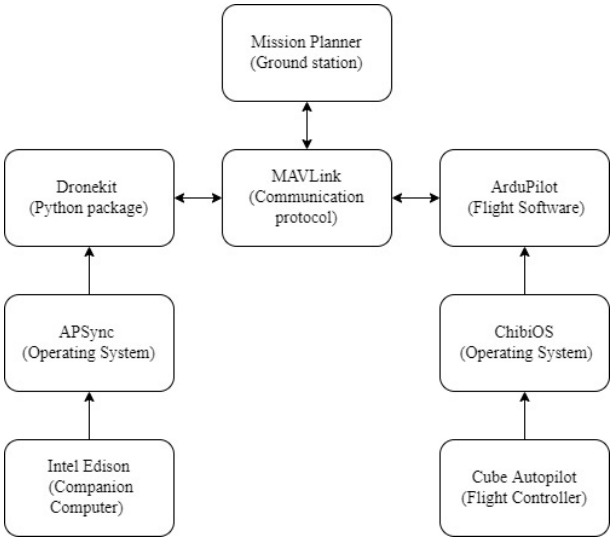


Figure 6.1: System hardware, software and firmware interconnections.

6.1.4 Sensors

The Cube flight controller contains three sets of IMUs (accelerometer, gyroscope and magnetometer), two of which are mechanically vibration-isolated. Two of the IMUs are the Invensense MPU9250 and the third is made up of the STM LSM303D (accelerometer and magnetometer) and L3GD20 (gyroscope). The Cube also contains two barometers, both of which are the MS5611. Multiple of each kind of sensor are provided for redundancy and the EKF in the ArduPilot firmware fuses the data from all sensors for additional accuracy.

Additional sensors were connected to the Cube Autopilot system to enable the previously developed EKFs to be tested.

A Ublox NEO-M8N GPS module was connected to provide positional data. This module also houses a magnetometer for additional external measurement of the Earth's magnetic field. The GPS module communicates with the flight controller via the I2C protocol.

An integrated optical flow and lidar sensor, Matek 3901-L0X, was also connected to the system. This module consists of a Pimoroni PMW3901 optical flow sensor, an Adafruit VL53L0X lidar sensor and a STM32L051 microcontroller, as well as additional power regulating circuitry. This sensor is able to communicate over UART using Multiwii Serial Protocol (MSP), which is supported by the ArduPilot firmware.

Figure 6.2: Hardware platform used for flight testing.

6.2 Flight Testing

Flight testing was conducted by programming the hexacopter with an autonomous mission, which involved taking off to a height of 1.5 metres, travelling approximately 11 metres to a second position and landing. The drone performed the mission successfully using the ArduPilot flight control software and the sensor data recorded during the flight was logged to the onboard SD card. This data was processed in MATLAB and used with the EKF algorithm described in Section 5.2. Fig. 6.3 shows the position and velocity results, and Fig. 6.4 show the angular results. The estimates produced by the previously developed algorithm are compared with the estimates of the ArduPilot (AP) software. Note that for the yaw angle results, an angle of 2π radians is equivalent to 0 radians, which explains the sudden decline in these results. The estimated yaw angle climbs above 2π radians before correcting and returning towards zero in the correct range.

A flight test was then repeated with the optical flow sensor installed, in order to evaluate the effectiveness of the EKF in the absence of GPS data. This flight involved taking off to a height off approximately 1.5 m, hovering at that point for approximately 10 seconds and then landing at the same point. The results of the EKF are shown in Fig. 6.5, compared with the AP estimation (which uses GPS data). The position and velocity estimates are not extremely accurate, however it can be seen that there is no significant drift in the position estimates over this time period.

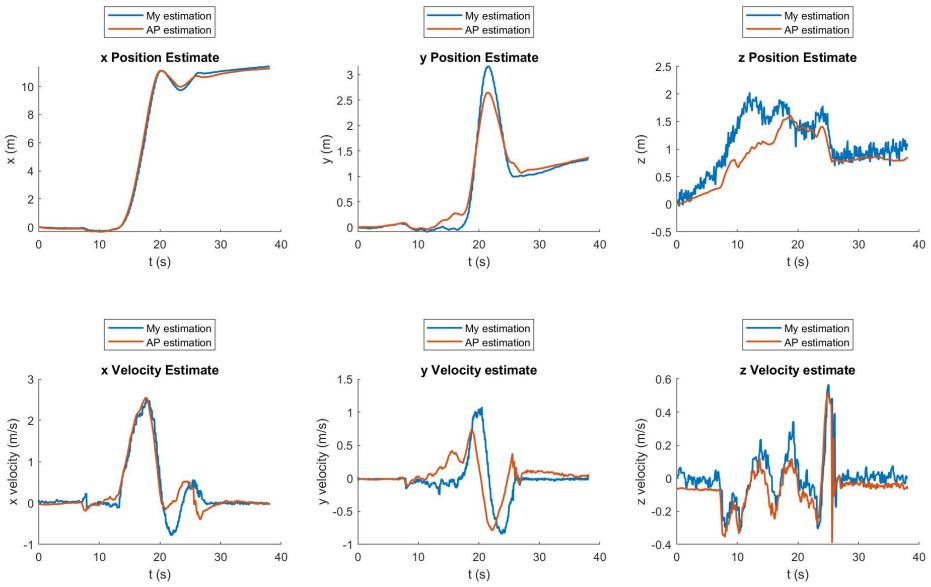


Figure 6.3: The EKF position and velocity estimates compared with the ArduPilot EKF estimates for flight test #1.

A third flight was undertaken in which the aircraft was programmed to take off to 1.5m, hover in place for 10 seconds, then travel some distance to land at another waypoint. This flight resulted in a collision with the ground (likely due to unreliable lidar data), however the first 25 seconds of data was able to be used for analysis. Using this data, it could be seen that using the OFS data alone was insufficient for tracking positional change.

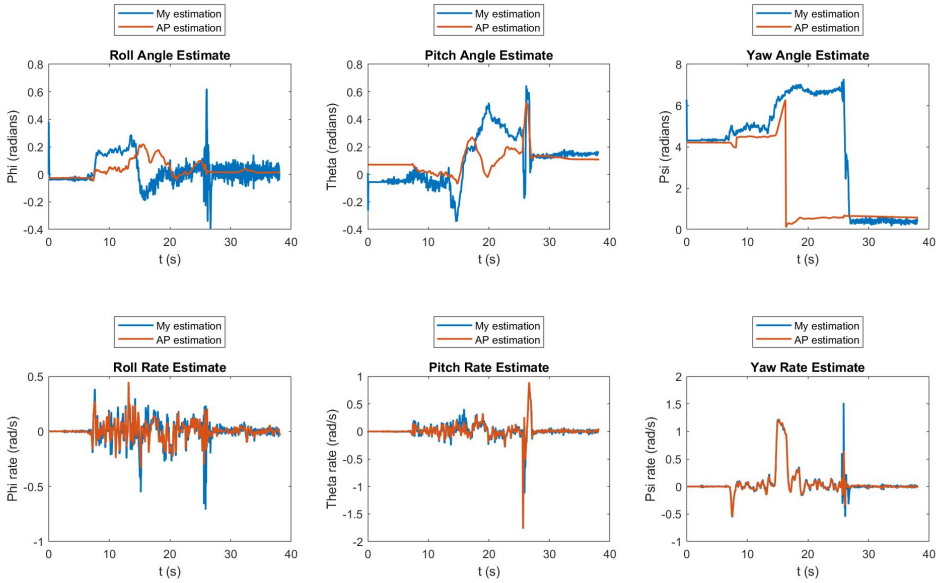


Figure 6.4: The EKF angular estimates compared with the ArduPilot EKF estimates for flight test #1.

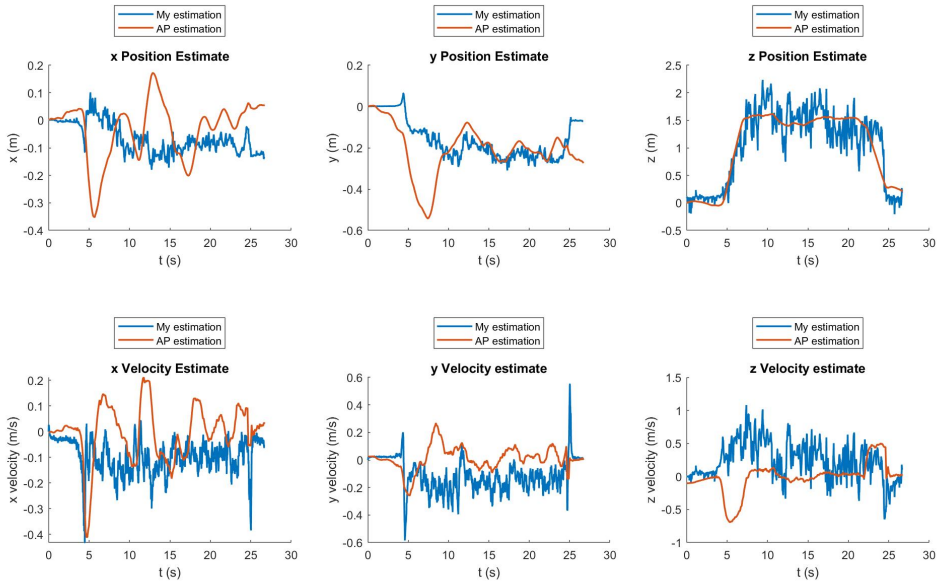


Figure 6.5: The OFS EKF position and velocity estimates compared with the ArduPilot EKF estimates for flight test #2.

6.3 Discussion of Results

To produce these results, the sensor data that was recorded on the SD card was processed post-flight in MATLAB. It is important to note that the data which is recorded on the SD card is not representative of all of the sensor data available to the flight controller. That is, the flight controller samples the sensors at a faster rate and not all data is logged. This severely limits the accuracy of the estimated states which are based upon this data. For example, the IMU is sampled at a rate of at least 1 kHz, however the data is logged at a rate of approximately 25 Hz. This issue was particularly prominent for the optical flow sensor data, which was logged at a rate of only 10 Hz but has a capability of 121 Hz. If the EKF was run directly on the hardware it may prove to be more accurate.

Another issue that arose was the unreliability of the lidar data. The datasheet states that the sensor only has an operating range of 2 m. Although, the flight test was conducted at 1.5 m above the ground, the angle of the vehicle during translational motion presumably caused the lidar line-of-sight distance to ground to exceed 2 m. This resulted in the lidar being unable to record data and therefore impacting the accuracy of the results - since the optical flow velocity estimates rely on knowledge of the scene depth. The operating range of this sensor makes it essentially useless for practical flights. However there are many other commercially available lidar sensors with larger operating ranges. Alternatively, many optical flow sensors are instead paired with sonar (ultrasonic) sensors for scene depth information.

It is also worth noting that sensors generally have significant biases and there are a number of models to describe the behaviour of sensor bias. A common model used in Kalman filter techniques is the random-walk model [39]. Errors were likely introduced into the data due to this effect and the developed algorithms do not account for this.

6.4 Chapter Summary

This chapter introduced the hexacopter platform's hardware and software architecture. A number of flight tests were carried out and the data was processed post-flight using the previously developed EKF algorithms. It is shown that with the particular optical flow sensor that was chosen, position data was not able to be accurately estimated during motion.

Chapter 7

Conclusion and Extensions

7.1 Conclusion

The aim of this thesis was to apply the concepts of modelling, control and state estimation to a specific configuration of rotary-wing aircraft to enable autonomous flight. In particular, Newton-Euler formalism was applied to the multi-rotor configuration to derive an accurate model for simulation. Both linear and nonlinear control techniques were developed and tested in simulation. State estimation algorithms based upon the extended Kalman filter were developed with a particular focus on examining the use of alternative sensors in GPS denied environments. The state estimation techniques were demonstrated in simulation and also tested on data gathered by sensors onboard a hexacopter platform during autonomous flights.

The initial model of the physical system was developed as a general multi-rotor model, i.e. this model could be applied to any number of configurations of multi-rotor vehicles. The total vertical thrust and each of the axial torques were taken as the four inputs to the system. However, the system has six degrees of freedom - three rotational dimensions and three translational dimensions - and hence is underactuated. This presents a challenging control problem as it is impossible to independently control horizontal motion in this case - rotational motion is first necessary to enact horizontal motion. To complete the model, the specific configuration of the hexacopter was explored. This allowed a more accurate model of the hardware platform to be developed and tested in simulation. Further, the parameters of the hardware platform were either measured or estimated based on external data such that the simulated model would adequately represent the dynamics of the physical system. The derivation of the model did not include any approximations or linearisations, however it did not necessarily account for all possible effects e.g wind, gyroscopic effects, air resistance. Previous studies demonstrate that these effects can be neglected without significant impacts to the accuracy of the model. Therefore, these effects were left out of the model to ensure the model was not unnecessarily complicated.

In the pursuit of autonomous flight, control theory is one of the most significant areas of research. A robust control system with a large operating region is vital to ensuring the success of an autonomous aircraft. The control problem was originally approached using PID control laws and these proved to be useful for hovering flight with minimal disturbances. However, since this control system relies on linearisation of the model about an operating point, the limitations become apparent if the vehicle is disturbed significantly from the assumed operating region. To improve upon this, backstepping - a nonlinear control technique - was employed in simulation. This resulted in improved operation, but nonzero steady state error was observed. The next progression was a set of backstepping control laws which also utilised integral action in the position controllers. This eliminated the previously observed steady state error whilst also increasing the robustness of the controller and limiting the effects of model uncertainties. The final problem was that of actuator saturation resulting in loss of control. This was countered by limiting the output of the position controllers with the use of a sigmoid function. The resulting controller was demonstrably robust in simulation in the presence of external disturbances. In summary, the resulting control system was able to respond to large positional commands and move to the given coordinates even in the presence of disturbances and model uncertainties.

An equally challenging problem for autonomous vehicles is that of state estimation. For an aircraft to accurately move to a given coordinate it must have a system to interpret sensor data to estimate its translational and rotaional states. Most modern systems rely on GPS data to estimate position - however GPS data is not always available and therefore an increasing amount of research is dedicated to state estimation in GPS denied environments. For this purpose two algorithms were proposed, with one utilising a GPS sensor and one instead making use of an optical flow sensor. Both algorithms were based upon the extended Kalman filter and used a number of common sensors including a barometer, IMU and compass. The algorithms were first tested in simulation which required models of each type of sensor to be developed with the inclusion of measurement noise. Both algorithms proved to be fairly accurate over short time periods, with the OFS-based algorithm experiencing only small amounts of drift in the positional estimates over extended time periods. However a more significant limitation of the OFS-based algorithm was the increased variability of velocity estimates when operating at larger distances above the surface - resulting in more significant positional drift. In summary, the GPS-based algorithm resulted in more accurate positional estimates, but the OFS-based algorithm was adequately accurate with little drift when positioned at a hover close to the surface.

Finally, a hexacopter platform was equipped with sensors and wireless capability to enable autonomous flights. The software onboard the hexacopter was ArduPilot - a versatile open source flight control architecture. Using this software, the aircraft performed multiple flight tests and the data gathered by the onboard sensors was stored on an SD card for post-flight processing. This

data was then processed using the state estimation algorithms in order to validate their effectiveness. However, the usefulness of the data was limited by the data logging rates being significantly slower than the sensor sampling rates. The state estimation techniques showed promise but would require further testing on hardware to ensure their accuracy.

Overall, the project was successful in deriving a mathematical model of the aircraft, validating a robust control system against that model and demonstrating accurate extended Kalman filter-based state estimation in simulation. A hardware platform was also configured with software as well as sensors, additional processing and wireless capability, resulting in successful autonomous flights.

7.2 Future Extensions

Automation of aircraft is a rapidly expanding area of research and there are many possible extensions to this project. A number of examples possible additions to this project are listed:

- Implement the backstepping control system on the hardware platform.
- Account for motor/propeller failure events during flight.
- Extend sensor fusion algorithm to handle sensor failure.
- Implement obstacle avoidance.
- Analyse the usefulness of additional sensors e.g differential pressure sensor (for airspeed).
- Consider the impacts of other effects e.g air resistance and gyroscopic effects.
- Consider slew rates of actuators, propeller inertia, etc.
- Develop a quaternion-based model to avoid singularities.
- Account for sensor biases in state estimation algorithms.
- Develop a path generation block as part of the control system.

Bibliography

[1] J. Stoff, *Historic aircraft and spacecraft in the Cradle of Aviation Museum*. Mineola, N.Y: Dover Publications, 2001.

[2] V. Raoult, L. Tosetto, and J. E. Williamson, "Drone-based high-resolution tracking of aquatic vertebrates," *Drones*, vol. 2, no. 4, 2018. [Online]. Available: https://www.mdpi.com/2504-446X/2/4/37

[3] A. Johnson, K. Fox, and D. Agle, "Nasa's ingenuity mars helicopter completes first one-way trip," *NASA Science Mars Exploration Program*, 2021. [Online]. Available: https://mars.nasa.gov/news/8942/nasas-ingenuity-mars-helicopter-completes-first-one-way-trip/

[4] G. Hautaluoma and A. Johnson, "Nasa's dragonfly will fly around titan looking for origins, signs of life," 2019.

[5] D. Raine, *Newtonian mechanics : a modelling approach*. Dulles, Virginia: Mercury Learning and Information, 2017.

[6] J. Voight, *Quaternion algebras*. Cham: Springer, 2021.

[7] X. Zhang, X. Li, K. Wang, and Y. Lu, "A survey of modelling and identification of quadrotor robot," *Abstract and Applied Analysis*, vol. 2014, pp. 1–16, 2014.

[8] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," 2006.

[9] R. Baránek and F. Šolc, "Modelling and control of a hexa-copter," in *Proceedings of the 13th International Carpathian Control Conference (ICCC)*, 2012, pp. 19–23.

[10] M. D. Ardema, *Newton-Euler Dynamics*. Springer-Verlag GmbH, Oct. 2006. [Online]. Available: https://www.ebook.de/de/product/11430198/mark_d_ardema_newton_euler_dynamics.html

[11] A. V. Nebylov, *Aerospace navigation systems*. Chicheter, West Sussex, United Kingdom: John Wiley & Sons, 2016.

[12] B. Rubí, R. Pérez, and B. Morcego, "A survey of path following control strategies for UAVs focused on quadrotors," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 241–265, sep 2019.

[13] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010, special Issue on Aerial Robotics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066110000456

[14] M. Moussid, A. Sayouti, and H. Medromi, "Dynamic modeling and control of a hexarotor using linear and nonlinear methods," *International Journal of Applied Information Systems*, vol. 9, pp. 9–17, 08 2015.

[15] P. Kokotovic, "The joy of feedback: nonlinear and adaptive," *IEEE Control Systems*, vol. 12, no. 3, pp. 7–17, jun 1992.

[16] K. Krstic, Kanellakopoul, *Nonlinear and Adaptive Control Design*. John Wiley and Sons, May 1995. [Online]. Available: https://www.ebook.de/de/product/4241932/krstic_kanellakopoul_kokotovic_nonlinear_control_design.html

[17] A. Isidori, *Nonlinear Control Systems*. Springer-Verlag GmbH, Aug. 1995. [Online]. Available: https://www.ebook.de/de/product/1345564/alberto_isidori_nonlinear_control_systems.html

[18] Chen, *Linear system theory and design*. New York: Oxford University Press, 1999.

[19] A. A. Mian and W. Daobo, "Modeling and backstepping-based nonlinear control strategy for a 6 DOF quadrotor helicopter," *Chinese Journal of Aeronautics*, vol. 21, no. 3, pp. 261–268, jun 2008.

[20] C. A. Arellano-Muro, L. F. Luque-Vega, B. Castillo-Toledo, and A. G. Loukianov, "Backstepping control with sliding mode estimation for a hexacopter," in *2013 10th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. IEEE, sep 2013.

[21] A. Roza and M. Maggiore, "Path following controller for a quadrotor helicopter," in *2012 American Control Conference (ACC)*. IEEE, jun 2012.

[22] T. Madani and A. Benallegue, "Backstepping control for a quadrotor helicopter," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2006.

[23] N. Xuan-Mung and S. K. Hong, "Robust backstepping trajectory tracking control of a quadrotor with input saturation via extended state observer," *Applied Sciences*, vol. 9, no. 23, p. 5184, nov 2019.

[24] X. Shao, J. Liu, and H. Wang, "Robust back-stepping output feedback trajectory tracking for quadrotors via extended state observer and sigmoid tracking differentiator," *Mechanical Systems and Signal Processing*, vol. 104, pp. 631–647, may 2018.

[25] T. Madani and A. Benallegue, "Control of a quadrotor mini-helicopter via full state backstepping technique," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006.

[26] E. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. IEEE.

[27] M. St-Pierre and D. Gingras, "Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system," in *IEEE Intelligent Vehicles Symposium*. IEEE, 2004.

[28] *U.S. Standard Atmosphere, 1976*, ser. NOAA - SIT 76-1562. National Aeronautics and Space Administration, National Oceanic and Atmospheric Administration, 1976. [Online]. Available: https://books.google.com.au/books?id=x488AAAAIAAJ

[29] G. Balamurugan, J. Valarmathi, and V. P. S. Naidu, "Survey on UAV navigation in GPS denied environments," in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. IEEE, oct 2016.

[30] S. Driessen, N. Janssen, L. Wang, J. Palmer, and H. Nijmeijer, "Experimentally validated extended kalman filter for uav state estimation using low-cost sensors," *IFAC-PapersOnLine*, vol. 51, no. 15, pp. 43–48, 2018, 18th IFAC Symposium on System Identification SYSID 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318317488

[31] M. V. Cook, *Flight dynamics principles : a linear systems approach to aircraft stability and control*. Waltham, MA: Butterworth-Heinemann, 2013.

[32] R. C. Nelson, *Flight Stability and Automatic Control*. MCGRAW HILL BOOK CO, Oct. 1997. [Online]. Available: https://www.ebook.de/de/product/3638064/robert_c_nelson_flight_stability_and_automatic_control.html

[33] A. Tayebi and S. McGilvray, "Attitude stabilization of a four-rotor aerial robot." IEEE, 2004.

[34] E. Capello, H. Park, B. Tavora, G. Guglieri, and M. Romano, "Modeling and experimental parameter identification of a multicopter via a compound pendulum test rig." IEEE, nov 2015.

[35] M. Herzog, "Design, implementation and analysis of a controller for a load suspended from an aerial vehicle," Master's thesis, KTH, School of Electrical Engineering (EES), 2016.

[36] W. Jasim and D. Gu, "Integral backstepping controller for quadrotor path tracking," in *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, jul 2015.

[37] D. Cabecinhas, R. Cunha, and C. Silvestre, "Rotorcraft path following control for extended flight envelope coverage." IEEE, dec 2009.

[38] W. Ding, J. Wang, and A. Almagbile, "Adaptive filter design for uav navigation with gps/ins/optic flow integration," in *2010 International Conference on Electrical and Control Engineering*, 2010, pp. 4623–4626.

[39] A. M. Sabatini, "Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation," *Sensors*, vol. 11, no. 10, pp. 9182–9206, sep 2011.

Appendix A

Simulink Models

A.1 Plant Model

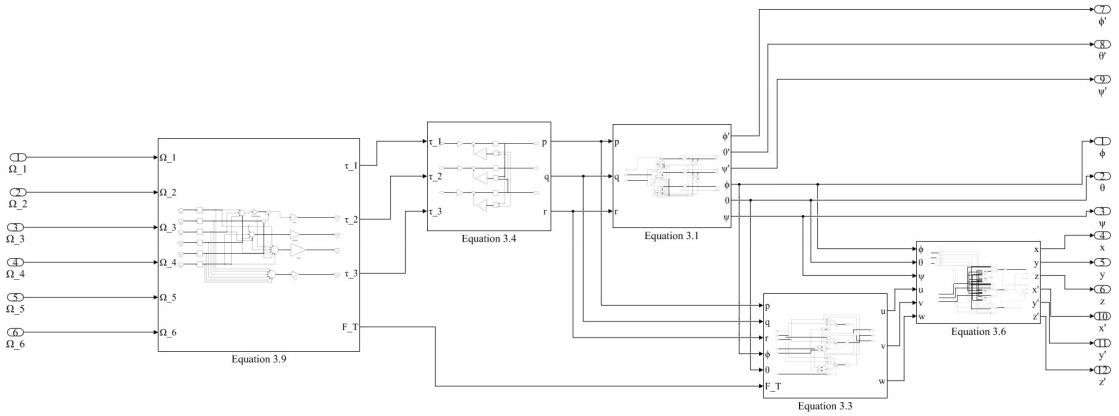


Figure A.1: Hexacopter plant Simulink model overview.

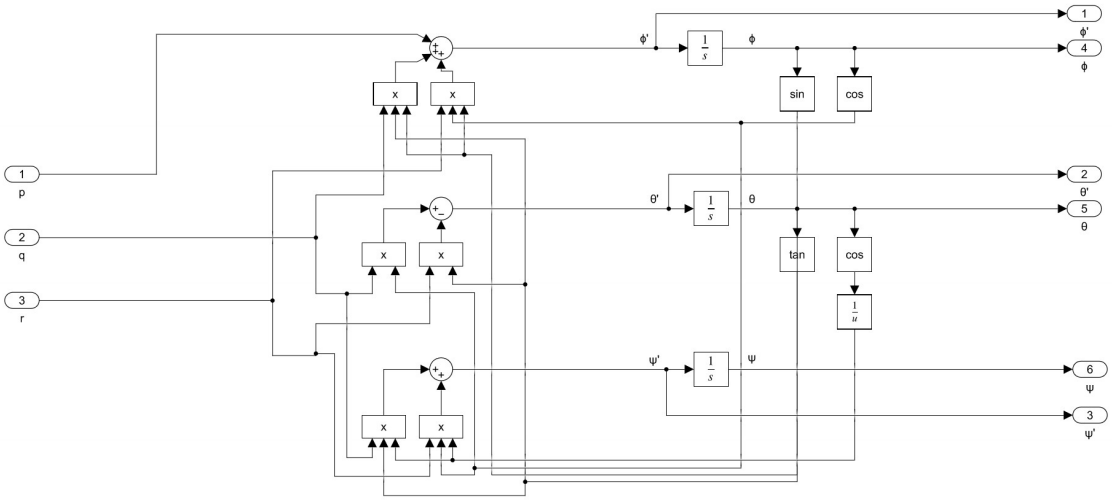


Figure A.2: Equation 3.1 Simulink subsystem.

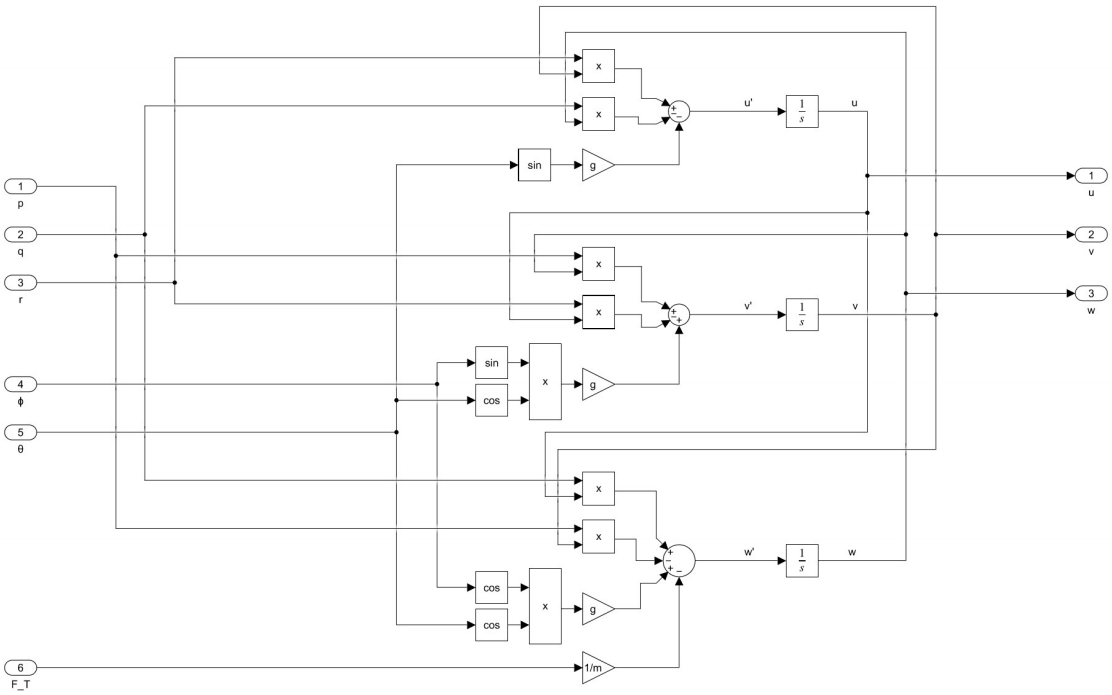


Figure A.3: Equation 3.3 Simulink subsystem.

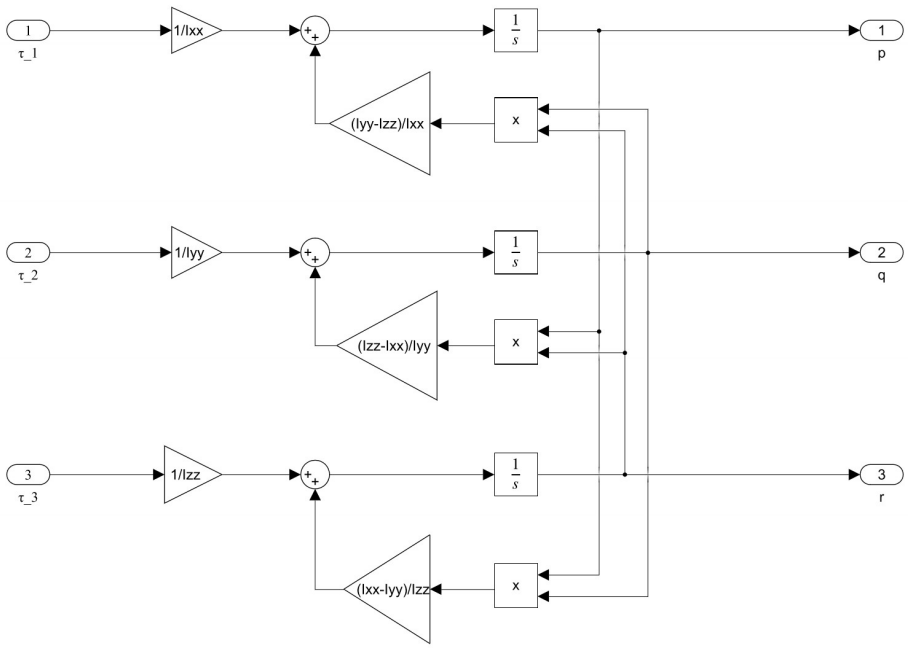


Figure A.4: Equation 3.4 Simulink subsystem.

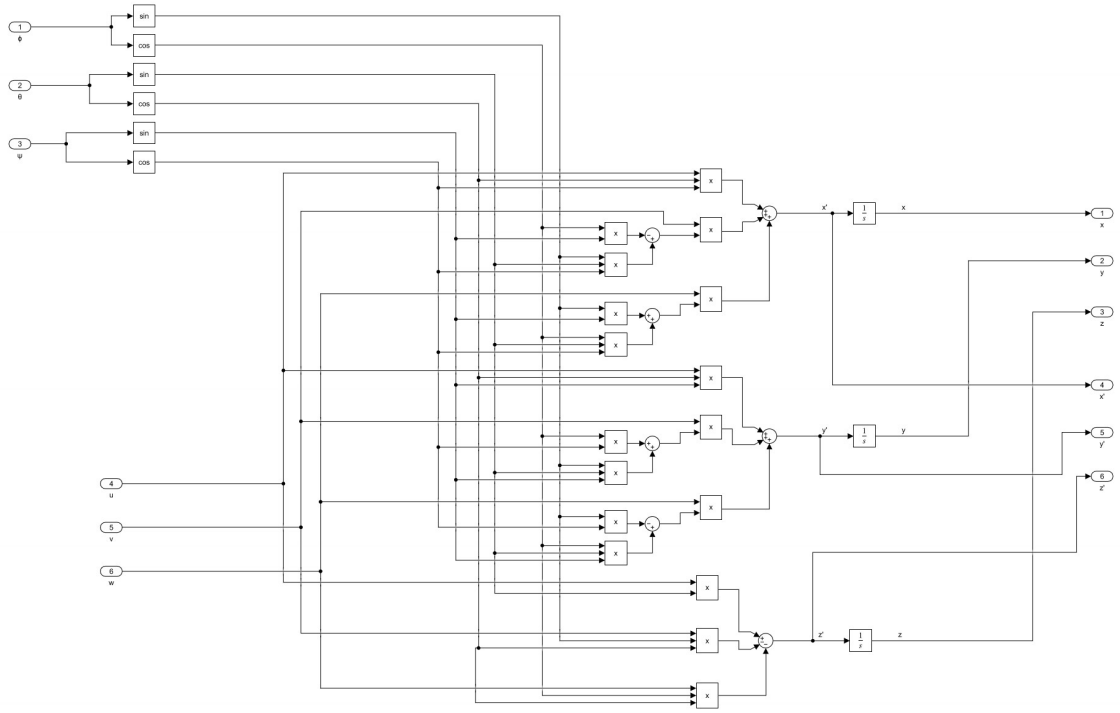


Figure A.5: Equation 3.6 Simulink subsystem.

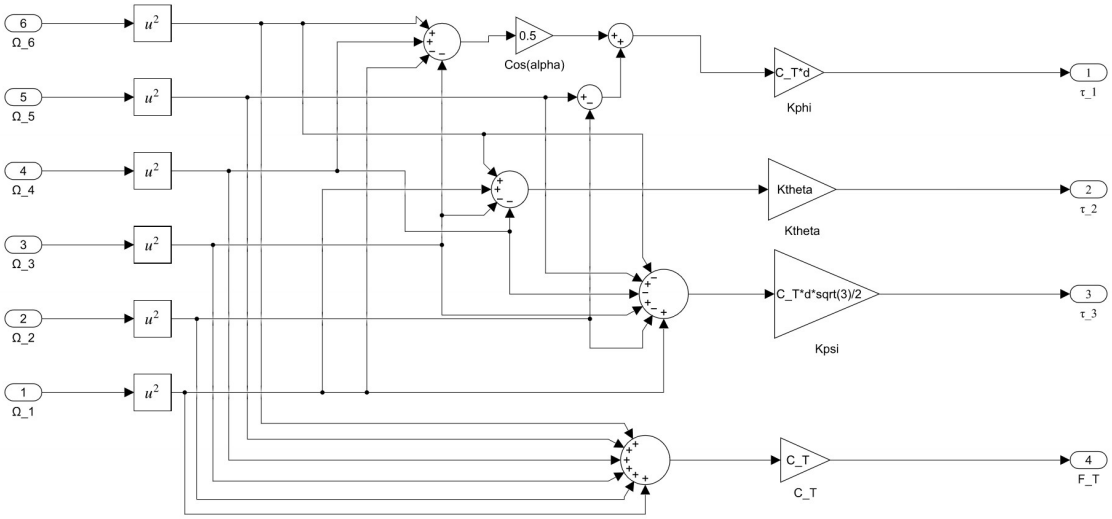


Figure A.6: Equation 3.9 Simulink subsystem.

A.2 Control Inputs to Motor Speeds

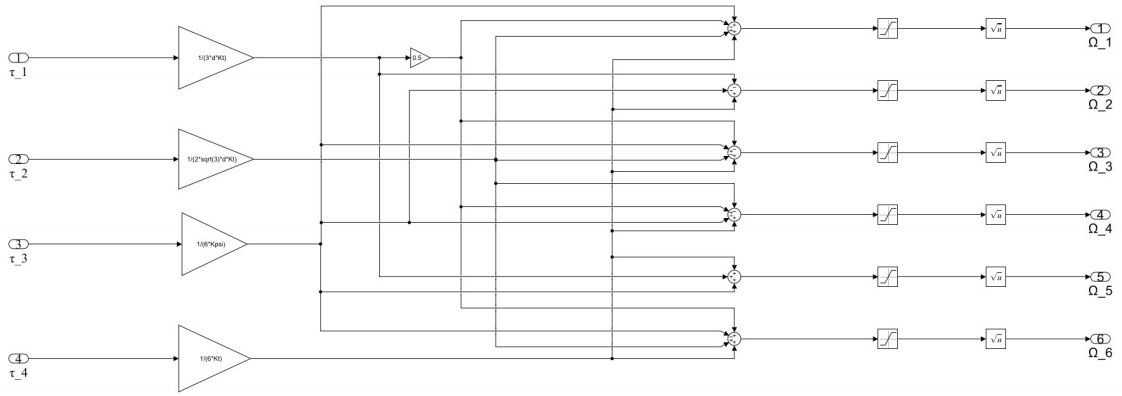


Figure A.7: Equation 3.9 Simulink subsystem.

A.3 Control System

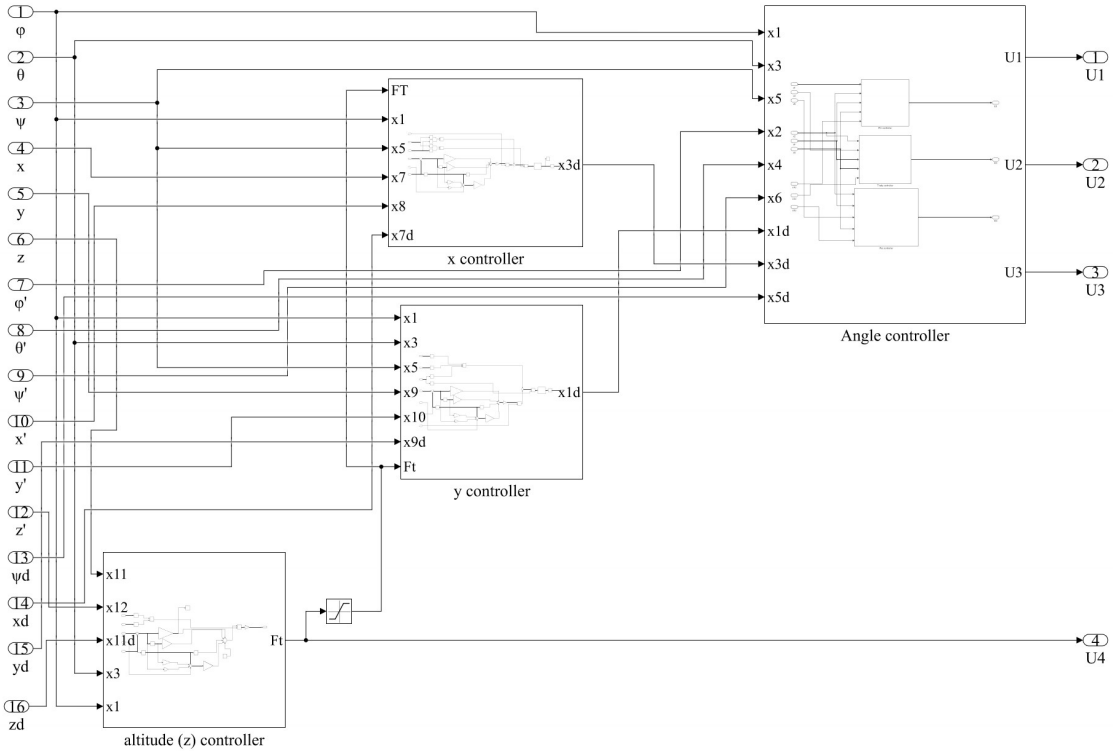


Figure A.8: Control system Simulink model overview.

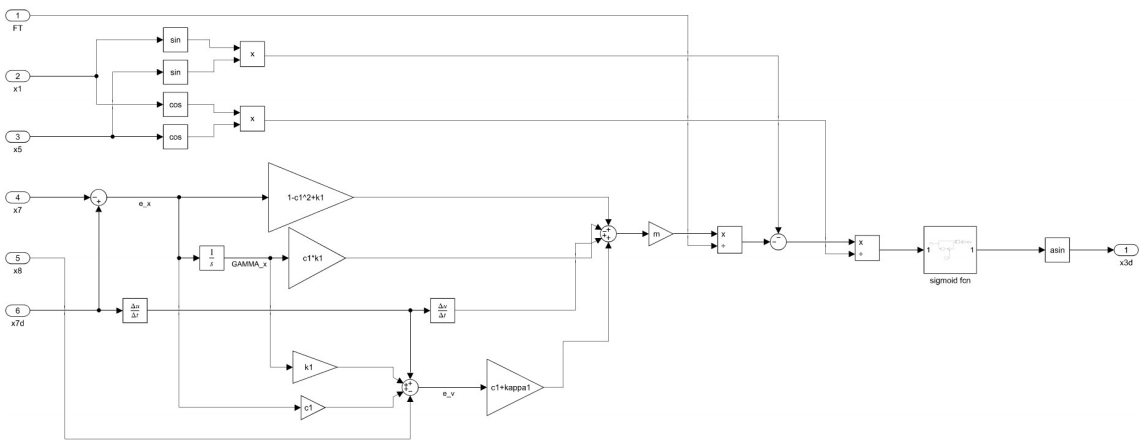


Figure A.9: X position controller Simulink subsystem.

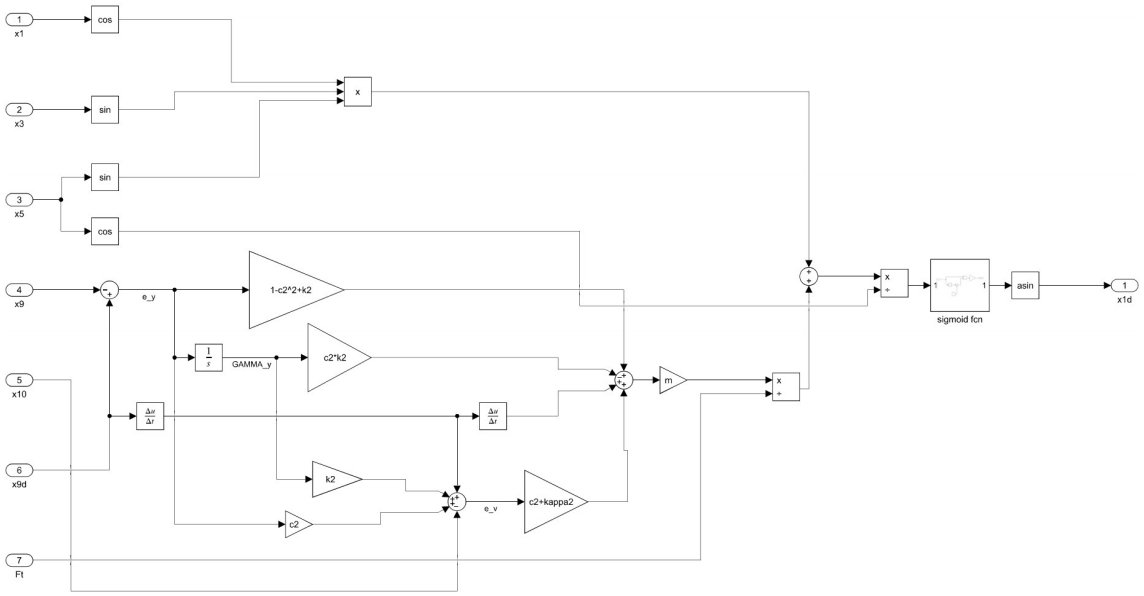


Figure A.10: Y position controller Simulink subsystem.

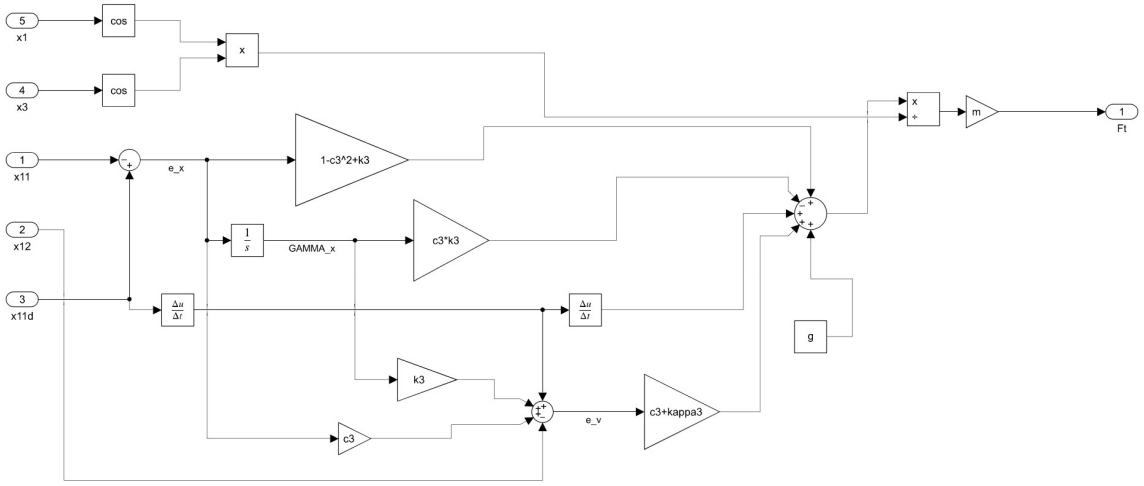


Figure A.11: Z position controller Simulink subsystem.

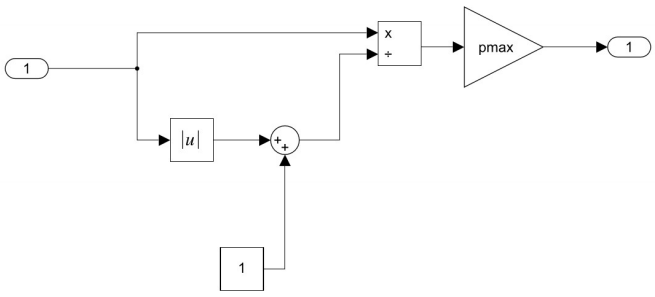


Figure A.12: Sigmoid function Simulink subsystem.

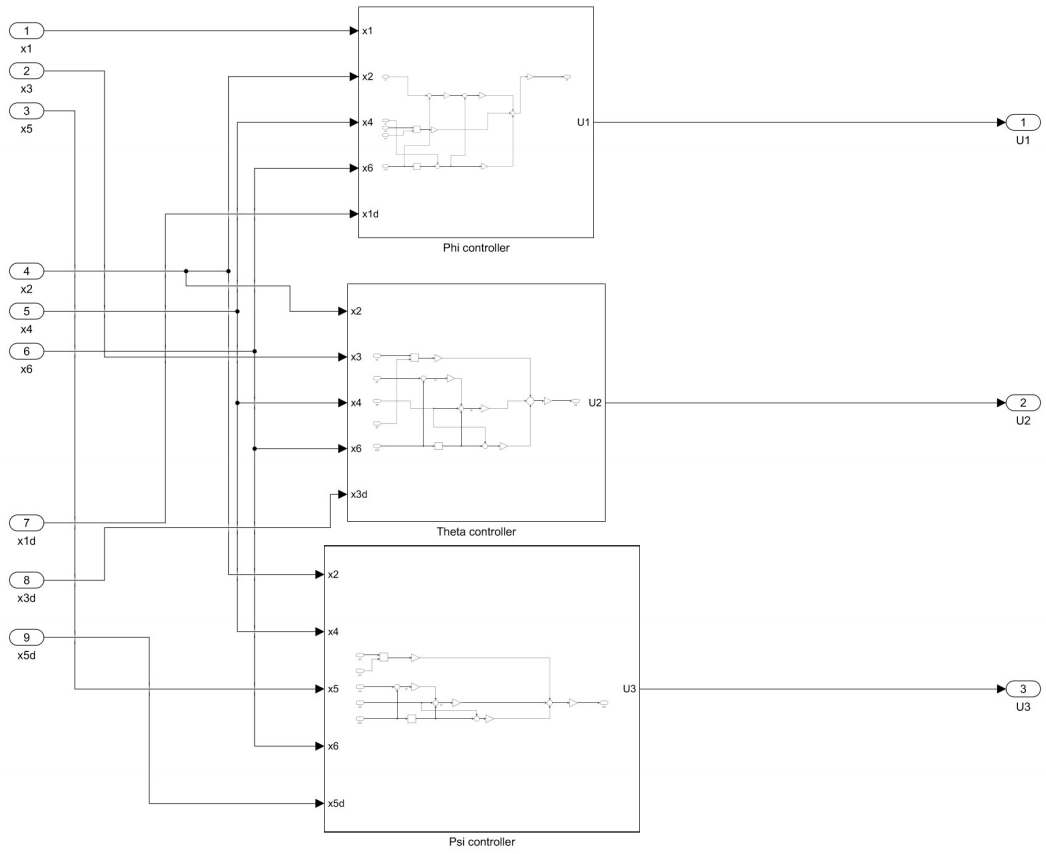


Figure A.13: Angle controller Simulink subsystem.

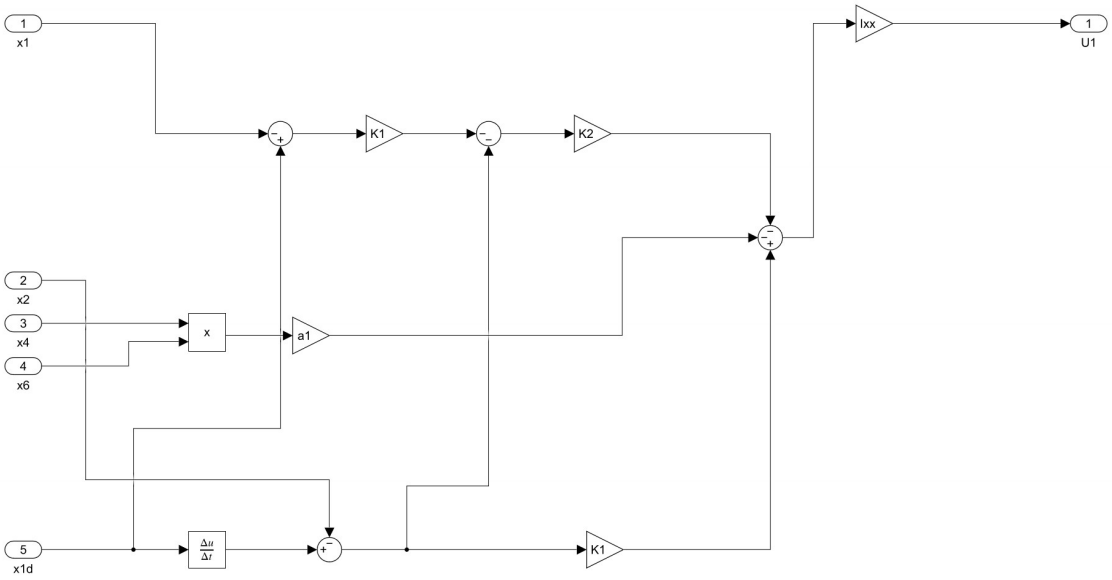


Figure A.14: Roll angle controller Simulink subsystem.

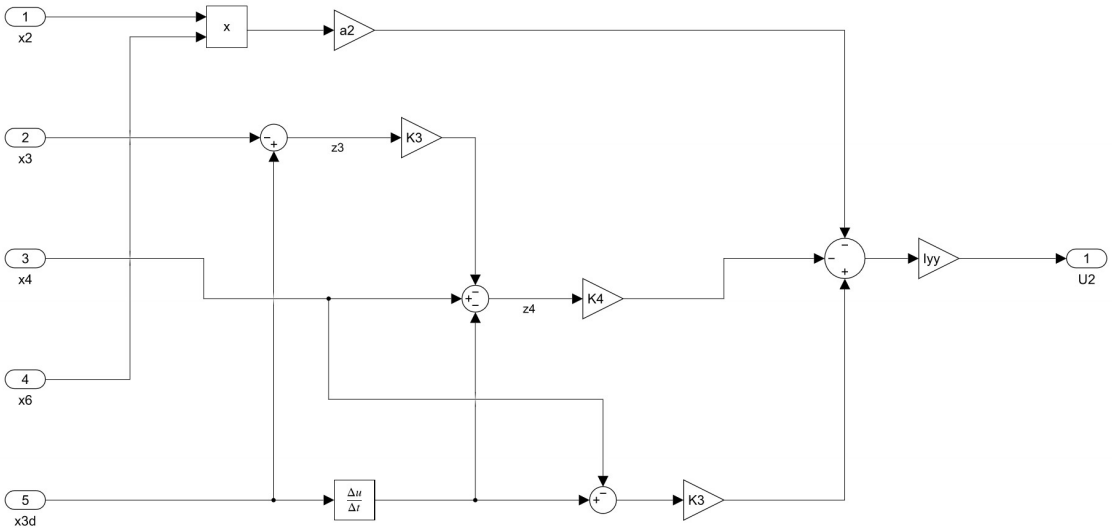


Figure A.15: Pitch angle controller Simulink subsystem.

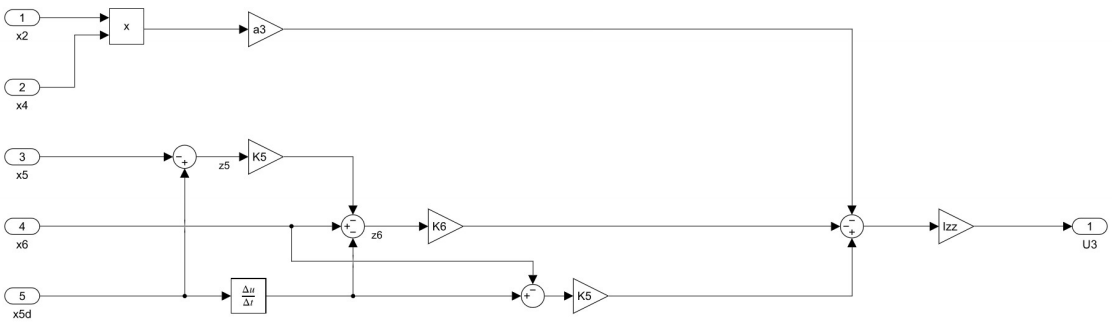


Figure A.16: Yaw angle controller Simulink subsystem.