Automatic
Control &
Systems
Engineering.

Modelling, Simulation, and Implementation of Linear Control for Asymmetric Multirotor Unmanned Aerial Vehicles

Akinola Alexander Dada

May 2020

Supervisor: Dr John Anthony Rossiter

A dissertation submitted in partial fulfilment of the requirements for the degree of MEng Mechatronic and Robotic Engineering

Abstract

Unmanned Aerial Vehicles (UAVs) are aerial systems not directly controlled by onboard personnel. There are many types of UAV platforms, chiefly defined by the characteristics of their mechanical constructions. One such class of UAVs are multirotors which possess multiple rotor wing actuators. These types of UAVs have a wide range of applications due to their ability to perform vertical/Short take-off and landing (VSTOL), stationary and low speed flight, coupled with their relatively simple mechanical designs when compared to single rotor constructions. The major challenge in dealing with these types of UAVs is their inherent instability in flight, save for the intervention of complex control systems. Therefore, in order to understand control systems capable of producing guaranteed performance, this project first seeks to investigate the physical characteristics and responses of UAVs through mathematical modelling, before proceeding to, design, develop, implement and compare the performance of 2 advanced linear control schemes on an embedded microprocessor, where this level of insight is requisite. These control schemes are, Linear Quadratic Gaussian (LQG) and Linear Quadratic Model Predictive Control (LQ-MPC).

Executive Summary

The first chapter provides a project introduction containing background information pertinent to the investigation, briefly discussing the nature and potential uses of UAVs. This section includes the systems engineering approaches taken, such as breaking down the project aims and objectives and setting them up as a set of tasks with milestones. This section provides information on project planning and management which includes the use of Gantt charts. This section will also detail all considerations and changes made to the project including those due to the SARSCOVID-2 pandemic.

The second chapter reviews and provides an analysis of the relevant supporting literature collected, which is referenced throughout the writing. The supporting research materials take the form of scientific journals articles, independent research publications, exerts from university lectures, and published books.

The third chapter briefly discusses the mechanical design and construction of the specific asymmetric hexarotor UAV being used as the case study. This Section goes through and details the processes and results of analysing the UAV and deriving a mathematical representation of the physical characteristic through a combination of first principles dynamic modelling and hardware-in-the-loop (HIL) testing. This section explores and expatiates on different modelling techniques while discussing the various assumptions made during the modelling process, a well as the reasoning behind the adopting of the techniques selected. This section then converts the derived model into various forms useful for control design, analysis and simulation.

The forth chapter presents an analyses of the system characteristics along with a dynamic simulation using industry standard applications. This section details the process behind the design and development of an LQG feedback controller, then discusses and analyses the simulation results. This section then discusses the limitations of LQG controllers prompting and justifying the design and development of an LQ-MPC feedback controller. This section presents and analysing the simulation results of the LQ-MPC feedback controller. This section then compares the performance of the LQG and LQ-MPC feedback controllers using appropriate metrics.

The fifth chapter discusses the characteristics of the embedded Linux single board computers (SBCs), using the BeagleBone Blue platform as a case study. This section discusses and details the processes and considerations behind implementing a UAV flight control systems on the platform. The section then presents a Unified Modelling Language (UML) Class diagram detailing an implementation proposal for an embedded LQG Controller.

The sixth and final chapter concludes and summarises the investigation results then presents possibilities for future work and development.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1. Background.
2. Aims and Objectives.
3. Project Management.

1.1 Background

UAVs are aerial systems which are not directly controlled by a human onboard the vehicle. There are many types of UAV platforms which are chiefly defined by the characteristics of their mechanical constructions [1],[2]. These defining criteria break down to include:

- The position, number and type of their actuators.
- Their mode/modes of flight.
- Their use cases specific features.

One such type of UAVs are multirotors, defined as such due to their multiple rotor wing actuators. UAVs of this description were developed in response to the mechanical complexity of single or dual rotor winged aircraft such as the various helicopter and auto-gyros variants. Despite both classes of UAVs possessing the ability to perform operations requiring VTOL, stationary and low speed flight, the greater simplicity of multirotor craft could potentially lead to reduced development, construction and maintenance costs as well as greater reliability in use [3].

These features enable them to be utilised in many wide ranging applications, where great dexterity in motion is required, and as such would not be possible using fixed winged platforms, such as operations required during; directed precision administration of substances to plants in precision agriculture, frequent and or detailed inspection of standing structures, precision wide area surveillance, amongst many others [4].

However, the major challenge in dealing with multirotor UAVs is their inherent instability in real world flight situations, where disturbances are present, making it impossible to operate these craft with direct open loop control of their actuators, as is possible and frequently done with fixed winged craft. Therefore, the need for the development of control systems is apparent.

In order for the developed control systems to behave in a guaranteed predictable manner and be able to meet defined performance specifications, it becomes necessary to understand the behaviour of the system to be controlled and thus derive a behavioural representation to be used in guiding and informing the design process, which is done through mathematical systems modelling. This level of insight is also required in the development of advanced control schemes.

1.2 Aims and Objectives

The framework of this project was set around the design and development of control software for VSTOL model aircraft. From this broad theme, the aims of modelling an asymmetric multirotor platform, developing multiple control schemes and then implementing these schemes on via an embedded SBC were derived.

These aims break down into multiple objectives and milestones which must each be achieved to fulfil the full scope of the aims outlined. These objectives can be broadly classified into 2 categories:

1.2.1 Basic Objectives

1. Develop a mathematical model representing the dynamics of the multirotor aircraft.
2. Develop a dynamic simulation of the crafts behaviour.
3. Develop LQG feedback control.

4. Investigate feedback control laws in simulation with the mathematical model to achieve behavioural targets.

5. Develop flight control software to interface with sensors and implement control laws.

6. Implement the flight control software on an embedded microprocessor unit(**).

7. Discuss the results of performance comparisons between the simulation and hardware implementations(**).

1.2.2 Advanced Objectives

1. Investigate the application of LQ-MPC schemes in simulation.

2. Incorporate and implement LQ-MPC schemes unto the flight control software(**).

3. Discuss the differences between the LQG and LQ-MPC implementations.

1.3 Project Management

In order to properly execute on the project Aims and Objectives, project activities were co-ordinated and managed by further breaking down and grouping tasks then laying them out in tabulations, Figures 1.1 and 1.2, which were used to generate the project Gantt charts showing required progress against time.

Due to the effects and responses to the global SARSCOVID-2 pandemic, The University of Sheffield was forced into closure/Cancellation of all on-site and in-person teaching, meetings and instruction from March of 2020. This coupled with the government mandated nationwide quarantine and lock down in the United Kingdom made access to the necessary facilities for the testing and evaluation of any developed UAV software impossible. As such, activities and work towards meeting the objectives succeeded by Double-Asterix in the aims and objectives subsection are not being undertaken. Project planning has been adjusted accordingly. This has also led to changes in term dates which resulted in adjustments being made to task timelines.

Regarding other aspects of the project, all material purchases were made on time, acquiring key hardware such as the main flight computer, the flight radio control system, which would

		Name	Duration	Start	Finish	Predecessors
1		**Development of V/STOL MultiRotor**	**189 days**	**21/10/19 08:00**	**27/04/20 08:00**	
2		S1 Week 4 Aims and Objectives	0 days	21/10/19 08:00	21/10/19 08:00	
3		S1 Week 6 Progress Review	0 days	04/11/19 09:00	04/11/19 09:00	
4		S1 Week 10 Second Reviewer	0 days	02/12/19 09:00	02/12/19 09:00	
5		S1 Week 11 Interim Report	0 days	09/12/19 09:00	09/12/19 09:00	
6		S1 Week 12 Second Reader Meeting	0 days	16/12/19 09:00	16/12/19 09:00	
7		S2 Week 1 Intereme Report Feedback	0 days	03/02/20 09:00	03/02/20 09:00	
8		S2 Week 6 Progress Review	0 days	02/03/20 09:00	02/03/20 09:00	
9		S2 Week 11 Final Report	0 days	20/04/20 08:00	20/04/20 08:00	
10		S2 Week 12 Presentation	0 days	27/04/20 08:00	27/04/20 08:00	
11		**Modelling**	**68 days**	**21/10/19 08:00**	**28/12/19 08:00**	
12		Obtain Parts	14 days	21/10/19 08:00	04/11/19 08:00	
13		Complete Multirotor Platform	14 days	04/11/19 08:00	18/11/19 08:00	12
14		Investigate Literature	14 days	21/10/19 08:00	04/11/19 08:00	
15		Develope Dynamical Model	40 days	04/11/19 08:00	14/12/19 08:00	14
16		Develop Model Simulation	14 days	14/12/19 08:00	28/12/19 08:00	15
17		**Control Systems Design**	**85 days**	**04/11/19 08:00**	**28/01/20 08:00**	
18		Investigate Literature	30 days	04/11/19 08:00	04/12/19 08:00	14
19		PID Control Design	14 days	28/12/19 08:00	11/01/20 08:00	16;18
20		LQG Control Design	21 days	28/12/19 08:00	18/01/20 08:00	16;18
21		Obtain results in Simulation	10 days	18/01/20 08:00	28/01/20 08:00	20
22		**Flight Software Development**	**150 days**	**21/10/19 08:00**	**19/03/20 08:00**	
23		Obtain key Libraries	7 days	21/10/19 08:00	28/10/19 08:00	
24		Discretize Control Laws	7 days	28/01/20 08:00	04/02/20 08:00	20;19;21
25		Modify/Develop Existing Software	30 days	04/02/20 08:00	05/03/20 08:00	23;24
26		Implement on Embedded Harware	14 days	05/03/20 08:00	19/03/20 08:00	25
27		**Implementation and Testing**	**21 days**	**19/03/20 08:00**	**09/04/20 08:00**	**13;15**
28		Implement/Test On platform	14 days	19/03/20 08:00	02/04/20 08:00	26
29		Obtain Results and Analyse Performance	7 days	02/04/20 08:00	09/04/20 08:00	28
30		**Advanced Objectives**	**176 days**	**21/10/19 08:00**	**14/04/20 08:00**	
31		Investigate Predictive control	120 days	21/10/19 08:00	18/02/20 08:00	
32		Implement in Simulation	14 days	18/02/20 08:00	03/03/20 08:00	16;31;21
33		Modify Flight Software	21 days	03/03/20 08:00	24/03/20 08:00	32
34		Implement/test on Hadware	5 days	09/04/20 08:00	14/04/20 08:00	33;29

Figure 1.1: Initial Timetable

		Name	Duration	Start	Finish	Predecessors
1		**Development of V/STOL MultiRotor**	**213 days**	**21/10/19 08:00**	**21/05/20 08:00**	
2		S1 Week 4 Aims and Objectives	0 days	21/10/19 08:00	21/10/19 08:00	
3		S1 Week 6 Progress Review	0 days	04/11/19 09:00	04/11/19 09:00	
4		S1 Week 10 Second Reviewer	0 days	02/12/19 09:00	02/12/19 09:00	
5		S1 Week 11 Interim Report	0 days	09/12/19 09:00	09/12/19 09:00	
6		S1 Week 12 Second Reader Meeting	0 days	16/12/19 09:00	16/12/19 09:00	
7		S2 Week 1 Intereme Report Feedback	0 days	03/02/20 09:00	03/02/20 09:00	
8		S2 Week 6 Progress Review	0 days	16/03/20 09:00	16/03/20 09:00	
9		S2 Week 11 Final Report	0 days	18/05/20 08:00	18/05/20 08:00	
10		S2 Week 12 Presentation	0 days	21/05/20 08:00	21/05/20 08:00	
11		**Modelling**	**68 days**	**21/10/19 08:00**	**28/12/19 08:00**	
12		Obtain Parts	14 days	21/10/19 08:00	04/11/19 08:00	
13		Complete Multirotor Platform	14 days	04/11/19 08:00	18/11/19 08:00	12
14		Investigate Literature	14 days	21/10/19 08:00	04/11/19 08:00	
15		Develop Dynamical Model	40 days	04/11/19 08:00	14/12/19 08:00	14
16		Develop Model Simulation	14 days	14/12/19 08:00	28/12/19 08:00	15
17		**Control Systems Design**	**137.375 days**	**04/11/19 08:00**	**20/03/20 17:00**	
18		Investigate Literature	30 days	04/11/19 08:00	04/12/19 08:00	14
19		LQG Control Design	73.375 days	28/12/19 08:00	10/03/20 17:00	16;18
20		Obtain results in Simulation	10 days	10/03/20 17:00	20/03/20 17:00	19
21		**Flight Software Development**	**7 days**	**21/10/19 08:00**	**28/10/19 08:00**	
22		Obtain key Libraries	7 days	21/10/19 08:00	28/10/19 08:00	
23		**Advanced Objectives**	**208 days**	**21/10/19 08:00**	**16/05/20 08:00**	
24		Investigate Predictive control	180 days	21/10/19 08:00	18/04/20 08:00	
25		Implement in Simulation	28 days	18/04/20 08:00	16/05/20 08:00	16;24;20

Figure 1.2: Final Timetable

have served as the main communication unit sending control commands to the multirotor, electronic speed controllers, and load cells which will be used to develop the test stand platform to identify the actuator dynamics. All Ad hoc minor changes to task completion periods are reflected in timeline changes from the initial proposal table, Figure 1.1, to the final proposal table, Figure 1.2.

The only major change to the project scope, not enforce by SARSCOVID-2, from its initial proposal, was the dropping of a PID control design section and all such tasks, which occurred before, and is reflected in, the S2 week project proposal review. This change was made to refocus the project objectives solely on model-based control design outcomes.

All changes to task completion dates for further objectives are displayed in the final proposal table, Figure 1.2, which was used to produce the final project Gantt chart available in Appendix A.

Chapter 2

Review Of Literature

2.1 Introduction

In preparing to execute the project tasks, various supporting and reference materials were acquired. The various reference literature take the form of articles published in scientific journals, research publications, exerts from university lectures, and published books, all of which together broadly cover and present 3 major themes that are relevant to the project's investigation:

1. Mathematical Modelling and Systems Identification.
2. Control Systems Design.
3. Simulation and Implementation.

2.2 Mathematical Modelling and System Identification

2.2.1 UAV

The exact details involved with the design, development and construction of UAVs is beyond the scope of the investigation but material, mechanical and electronic consideration when designing such craft are explored in [3].

2.2.2 Dynamics

In modelling systems, it is first necessary to state all assumptions made about its characteristics. These assumptions simplify the process of systems model abstraction. Following that, multi-

rotors can be defined as rigid-bodies free to move in three-dimensional (3-D) space, therefore, moving with 6 degrees of freedom (6-DOF). With this modelling assumption, UAV motion is subject to the laws of rigid body Kinematics and Kinetics. All the acquired material takes this stance.

Vehicle motion is defined in terms of 2 coordinate frames moving relative to one another where the physical quantities that change with time, called states, change with respect to one frame or the other. They are the Earth-Frame, fixed to the earth and the Body-Frame, attached to the vehicle body. Each frame consists of 3 orthogonal axes, whose nomenclature varies from source to source, about and along which rotational and translational motion can occur respectively [5],[6],[2],[7].

The sources obtained explore 2 ways for representing this rotational motion of the reference frames:

- Sources [5],[6],[1],[2],[8],[4],[9],[10] present their models using Euler angles which describe arbitrary orientation in the 3-dimensional Euclidean space using three parameters which represent a sequence of three elemental rotations about each axes of the coordinate system. Any orientation in 3D space can be achieved by composing these 3 elemental rotations.

- Sources [5],[11],[12] present representations through Quaternions which solve issues present with computing Euler angles such as computational expense and numerical singularities which occur when certain orientations are reached, as Quaternions do not require the calculation of sin and cos when certain angles go to 0 or 90 degrees and their multiples. Quaternions represent any arbitrary orientation by constructing rotations as that of a single angle about an imaginary axis.

The rigid body assumptions also allow for the utilisation of several Mechanical Modelling techniques and conventions which are used to derive the non-linear dynamics of the body frame mechanical systems considering both kinematics and Kinetics:

- The Newton-Euler convention which derives representations of systems dynamics through the application of first principles using newtons 2nd law of motion showing the effects of forces applied to the rigid body by its actuators and environment. Utilised in [6],[1],[8],[4],[9],[11],[10].

- The Euler-Lagrange convention which derives representations of systems dynamics through the study of energy differentials between kinetic and gravitational potential generated by motion. Utilised in [2].
- The Newton-Hamiltonian.
- Pure Hardware-in-the-Loop system identification, which decouples each axes of motion and determines approximate model frameworks for each then populates said framework with experimentally obtained parameters. Such was the process taken by [13]

This investigation utilises the Euler-angle representation and the Newton-Euler convention due to its intuitive familiarity and relative simplicity.
Due to the UAV's layout, it is asymmetric, this creates a system where its mass is not geometrically evenly distributed about the centre of mass leading to where inertia is not represented by a typical diagonal inertial tensor but instead one with non-trivial off diagonal elements and thus creating transposed reference frames. However, for the purpose of this investigation, it is assumed that the system is symmetrical as the off-diagonal terms in the inertia matrix are far lower in magnitude than the diagonal terms, thus can be reasonably neglected. Otherwise, the entire coordinates system would be adjusted and transformed to produce a true diagonal matrix as presented in [14].

In order to design the controllers, the non-linear model obtained via the Newton-Euler convention needs to be linearised. This is accomplished through the application of Jacobi's linearisation method as presented in [15]. This process produces a full linear state space representation of systems dynamics around stable operating points. This model is then discretized for analysis and implementation in digital embedded systems.

2.2.3 Actuator Dynamics

The UAV is actuated by 6 1000Kv brushless direct current (BLDC) motors [16], attached to 6 10x45inch propellers and is powered by a 4-cell lithium-polymer battery with an operational voltage range of 16.8 volts to 12.8 volts. To fully develop a system model, the characteristics of these parts must be modelled. This can either be done via first principles calculations as presented in [6],[11],[17] or via experimental systems identification as presented in [10],[3],[16],[7],[12].

The systems identification approach involves the derivation of a lumped parameter linear input-output model, encompassing the electronic speed control unit, motor and propeller dynamics, between input Pulse width modulated (PWM) signal pulse width and the output angular velocity, torque and thrust forces. This has the advantage of reducing complex dynamics enabling the utilisation of a minimum viable model and as such, is being implemented in this investigation.

2.3 Control Systems Design, Simulation and Implementation

2.3.1 LQG

The Linear Quadratic Gaussian makes use of the full state of a system, obtainable from a limited set of measurements through the application of a Gaussian estimator, in this case a Kalman filter [18],[19],[20],[21] to obtain the full state. It is a form of optimal control where its objective is to find a set of control actions which minimize some cost or performance function to bring the system's state to a desired set of values. Sources [22],[23],[24],[25] present derivations of state feedback control leading to different implementation of LQG control. Source [23] in particular details implementation of LQG control in embedded systems.

2.3.2 LQ-MPC

Like LQG, MPC is also a kind of optimal control, however, this technique's cost or performance function is limited to looking at future states over some fixed finite horizon and control signals can be determined so that they implicitly taking into account and determine control action so as to meet physical constraints on the system. MPC is also known as receding horizon control (RHC) as it only acts on the first step in the horizon of states before recalculating and performing this action over and over until the target is achieved.

MPC does not refer to any one algorithm or method but instead a methodological philosophy which implements certain key concepts and ideas in various ways as stated by [26]. Thus, this investigation makes a limiting delineation by specifying LQ-MPC which limits the scope to MPC consisting of linear systems and quadratic cost functions which specifically takes after the LQG.

Sources [25],[26],[27] flush out and gives step by step guides and best practices and consider-

ations to follow when designing Linear predictive control algorithms. Source [25] in particular discusses and reasons through the limitations of LQG and the derivation of LQ-MPC from the perspective of transitioning from an LQG formulation. one source, [13], also discusses various implementations methods including utilising machine leaning techniques to estimate certain optimisation parameters, as well as proposing and implementing such a controller.

The developed Control systems will be implemented on a Beagle Bone Blue Linux based single board computer (SBC) extending from the Robot Control Library (RCL) [28]. Sources [23],[29] presents consideration for the implementation of flight control systems and source [30] presents considerations for designing software in general through systems engineering approaches.

2.4 summary

This Chapter:

- Described the reference source material gathered.
- Highlighted the major take aways from each source while putting them in the context of the larger project objectives.
- Presented the reasoning behind the selective application of source material.

Chapter 3

Mathematical Modelling

3.1 Introduction

Mathematical models are used to describes the characteristics of physical systems. They are derived from assumptions made about a systems behaviour which are represented abstractly using mathematics. In modelling systems, it is first necessary to state all assumptions made about its characteristics. These assumptions simplify the process of model abstraction to ensure viability standards are met. This allows for a mapping of a multirotor movements and behaviour with the respect to it inputs and external influences, meaning mathematical models can be considered as a series of functions that map inputs onto outputs while determining all the important time dependant elements of the system. In order to successfully accomplish the development of an appropriate mathematical model for both simulation and control design, the following topics will be explored:

1. UAV
2. Kinematics
3. Kinetics
4. Actuator Dynamics
5. Linearisation

Figure 3.1: Y6 Hexarotor UAV

3.2 UAV

The UAV under consideration throughout this project is an asymmetrical Y6 hexarotor. This kind of configuration has actuators in 3 counter-rotating co-axial pairs and 3 motor groups. A coaxial Y6 hexarotor is mechanically simpler than a classical helicopter since it has propellers with constant pitch and without a swash plate [4]. The advantages of Y6 hexarotors with respect to other multirotor configurations and classical helicopters is that the coaxial configuration increases the thrust without increasing the frame size, while naturally eliminating the loss of efficiency due to direct torque compensation, another advantage of this configuration is better stability. However, the disadvantages are the higher energy consumption for six propulsion units and inefficiency due to the aerodynamic interaction present with co-axial actuator combinations [10]. The UAV is actuated by 6 1000Kv BLDC motors [16], attached to 6 10x45inch propellers and is powered by a 4-cell lithium-polymer battery. The exact details involved with the design, development and construction of UAVs is beyond the scope of the project investigation but material, mechanical and electronic consideration when designing such craft are explored in [3].

Following that, multirotors can be defined as; rigid-bodies free to move in 3-D space with 6-DOF, with all motion restricted to being either rotational or translational.

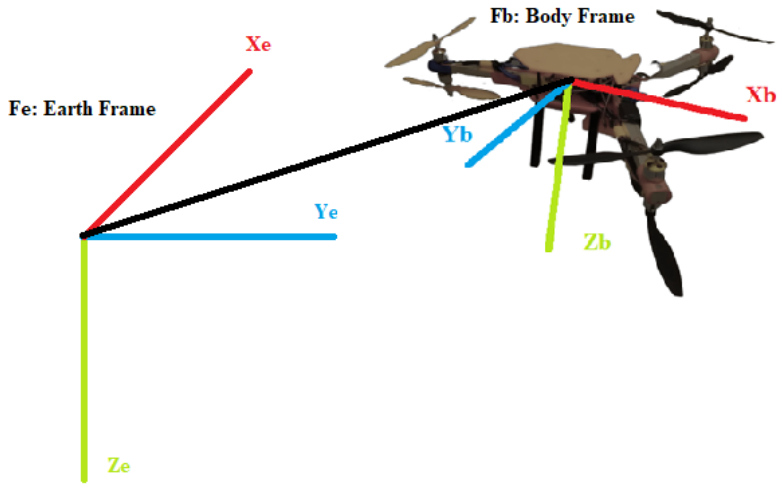


Figure 3.2: Coordinate Frames

More detailed mathematical models may describe certain multirotor characteristics more accurately, but they also require more computational resources which can lead to, longer simulation times, the inability to successfully complete simulations [8], and more complex than necessary control design processes and solutions.

This project will derive and present 2 models, a non-linear model utilised in simulation, and a Linear model utilised in analysis and control design.

3.3 Kinematics

Multirotor mathematical models have to describe attitude and position according to the geometry of the UAV [4]. one of the most important parts of multirotor modelling is understanding the geometric kinematic relationships between the reference frames [11]. Kinematics is the study of motion in terms of positions and velocities without regard to the forces causing the motion. Multirotor motion can be described by a number of variables called states, that are related and change with suitably chosen axes systems or reference frames [5]. These reference frames are:

- The Earth Frame: Fe
- The Body Frame: Fb

Each frame consists of 3 orthogonal axes, Xe, Ye, Ze and Xb, Yb, Zb respectively, about and along which rotational and translational motion respectively can occur. Where Fe is fixed to the earth near the multirotor, such that a flat earth assumption can be made and sustained [6], and Fb is attached to the vehicle centred at the UAV's centre of mass, which is free to move with respect to Fe. The usual convention for the axes representation is to have a fixed right-handed axes system with the positive Z axes pointing downwards, towards the earth when level or hovering, the positive X axis pointing forward and the positive y axis pointing towards the right or starboard side. This convention is referred to as the NORTH-EAST-DOWN Right-handed coordinate system [5],[6],[11]. Given the definition of the coordinate reference frames, system variables can now be defined in terms of these reference frames.

In Fe, we define vectors:

$$\vec{E}l = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.1}$$

which represent the linear translational positions of Fb's centre with respect to (w.r.t) Fe.

$$\vec{E}r = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{3.2}$$

which represent the angular rotation of Fb about all 3 axes respectively, w.r.t Fe. Which are also known as (roll, pitch, yaw) respectively.
In Fb, we define vectors:

$$\vec{B}l = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.3}$$

which represents the linear velocity in each Fb axis respectively.

$$\vec{B}r = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.4}$$

which represents the angular velocity in each Fb axis respectively.
From these 12 elemental variables and 4 elemental vectors all, subsequent states and motions can be derived and represented [5],[6],[2].

Using the ϕ, θ, ψ representation, the rotational motions of Fb w.r.t Fe can be derived by looking at rotations about each axis individually. Decomposing any arbitrary rotation and orientation in 3-D Euclidean space in this manner presents a sequence of three elemental rotations of each axes of Fb w.r.t its counterpart in Fe. These elemental rotation angles are known as *Euler angles* where rotations are represented as direction cosine matrices [5],[6]. These are:

$$R_\psi = \begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$R_\theta = \begin{bmatrix} C_\theta & 0 & -S_\theta \\ 0 & 1 & 0 \\ S_\theta & 0 & C_\theta \end{bmatrix} \tag{3.6}$$

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix} \tag{3.7}$$

These elemental rotations can by combined via the ZYX convention which multiplies them together in the specific sequence:

$$R^{-1} = R_\psi \cdot R_\theta \cdot R_\phi$$

to produce a matrix, the inverse of which, R, is able to map the translational motion of Fb unto Fe given any set of arbitrary ϕ, θ, ψ values. [5],[6],[1],[2],[8],[4],[9],[11],[10].

$$R = \begin{bmatrix} C_\theta C_\psi & -C_\theta S_\psi + S_\phi S_\theta C_\psi & -S_\phi S_\psi + C_\phi S_\theta C_\psi \\ C_\theta S_\psi & C_\phi C_\psi + S_\phi S_\theta S_\psi & -S_\phi C_\psi + C_\phi S_\theta S_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \tag{3.8}$$

Therefore:

$$\dot{\vec{E}}l = R \cdot \vec{B}l$$

Which maps the linear velocities and accelerations in Fb to linear velocities and accelerations with respect to Fe.
Similarly, to map the angular velocities and accelerations in Fb w.r.t Fe requires another transformation matrix, the inverse of which, T, is also derived from the manipulation of Euler angles [5],[2].

$$T = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & \frac{S_\phi}{C_\theta} & \frac{C_\phi}{C_\theta} \end{bmatrix} \tag{3.9}$$

Therefore:

$$\dot{\vec{E}}r = T \cdot \vec{B}r$$

Where S, C, T, represent Sin, Cosine and Tan respectively.

3.4 Kinetics

Kinetics is the study of motion considering the forces and torques which cause the motion. As stated in the previous UAV subsection, this investigation looks at a Y6 hexarotor UAV which possesses 6 BLDC motors with propellers in a coaxial motor layout. Each motor propeller unit produces a thrust force, the collective effect of which can be summed and lumped together as a single force F, which means:

$$\sum_{i=1}^{i=6} F_i$$

where i denotes the motor index.
Each motor propeller unit or propulsion unit, also produces a reaction torque. When a motor

turns, in overcoming air resistance, a reactive force acts on the propeller in the direction opposite to the motor's rotation which produce a torque acting on the UAV body. These torques, τ, can then also be summed together, meaning:

$$\sum_{i=1}^{i=6} \tau_i$$

The co-axial configuration also ensures single point torque balancing [4]. meaning, so long as all rotors pair each produce the same torque, the net reactive torque produced is zero. The translational and angular motion of the Y6 hexarotor is controlled by thrust forces and torques produced by each motor. The main thrust is the sum of all rotors thrust, and rotational movement is generated by differences in individual motor thrusts and torques [10].

3.4.1 Newton-Euler

The Multirotor 6-DOF rigid body kinetics takes into account the mass, and the inertia of the body. These are described by differential equations, which are derived through the utilization of the Newton-Euler modelling convention which derives representations of systems dynamics through the application of first principles via newtons 2nd law of motion aggregating all forces and torques applied to the rigid body by its actuators and environment while observing the resultant accelerations produced.

Resolving the Forces acting linearly on the UAV produces:

$$M\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = M\begin{bmatrix}0\\0\\g\end{bmatrix} - \begin{bmatrix}C_\theta C_\psi & -C_\theta S_\psi + S_\phi S_\theta C_\psi & -S_\phi S_\psi + C_\phi S_\theta C_\psi\\ C_\theta S_\psi & C_\phi C_\psi + S_\phi S_\theta S_\psi & -S_\phi C_\psi + C_\phi S_\theta S_\psi\\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta\end{bmatrix}\begin{bmatrix}\ddot{u}\\ \ddot{v}\\ \ddot{w}\end{bmatrix} - \begin{bmatrix}D_l xx & 0 & 0\\ 0 & D_l yy & 0\\ 0 & 0 & D_l zz\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} \tag{3.10}$$

$$M \cdot \ddot{\vec{El}} = R \cdot \ddot{\vec{Bl}} - M \cdot \vec{g} - diag(D_l) \cdot \dot{\vec{El}}$$

where M is mass, g is acceleration due to gravity acting in the vertical Ze axis plane, and D_l is a diagonal matrix of drag coefficients with aerodynamic drag acting directly proportional to velocity in the Fe frame.

Given that all thrust forces from the motor act along the same Zb axis plane, and there are no other force acting in the other axes therefore, the resultant equation can be rewritten as:

$$M\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = M\begin{bmatrix}0\\0\\g\end{bmatrix} - \begin{bmatrix}C_\theta C_\psi & -C_\theta S_\psi + S_\phi S_\theta C_\psi & -S_\phi S_\psi + C_\phi S_\theta C_\psi\\ C_\theta S_\psi & C_\phi C_\psi + S_\phi S_\theta S_\psi & -S_\phi C_\psi + C_\phi S_\theta S_\psi\\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta\end{bmatrix}\begin{bmatrix}0\\0\\\sum F_i\end{bmatrix} - \begin{bmatrix}D_l xx & 0 & 0\\ 0 & D_1 yy & 0\\ 0 & 0 & D_l zz\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} \tag{3.11}$$

Resolving the torques and forces acting angularly on the UAV in Fb produces:

$$\begin{bmatrix}Ixx & 0 & 0\\ 0 & Iyy & 0\\ 0 & 0 & Izz\end{bmatrix}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}A_p\\ A_q\\ A_r\end{bmatrix} - \begin{bmatrix}p\\ q\\ r\end{bmatrix} \times \begin{bmatrix}Ixx & 0 & 0\\ 0 & Iyy & 0\\ 0 & 0 & Iyy\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} - \begin{bmatrix}D_r xx & 0 & 0\\ 0 & D_r yy & 0\\ 0 & 0 & D_r zz\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{3.12}$$

$$I \cdot \dot{\vec{B}r} = A - C - diag(D_r) \cdot \vec{B}r$$

Where A is the vector of torques generated by the sum of motor forces and torques in each axis, C is the Corriolis matrix formed of the cross product between the angular velocity vector and the diagonal inertia Tensor I, and D_r is the diagonal matrix of drag coefficients acting against motion in each axis..

The A vector breaks down to:

$$A_p = L_1((F_1 + F_2) - (F_3 + F_4)) \tag{3.13}$$

$$A_q = L_2((F_1 + F_2 + F_3 + F_4) - L_3(F_5 + F_6)) \tag{3.14}$$

$$A_r = \sum_{i=1}^{i=6} \tau_i \tag{3.15}$$

Where L is the perpendicular distance from each motor group to the centre of mass.

3.4.2 Asymmetry

Due to the asymmetry of the Y6 hexarotor, mass is not even distributed across the UAV frame. This creates a particular imbalance of inertia about the centre of mass, which is the centre of all rotation in Fb. This leads to a case where I has off diagonal elements[14]. i.e:

$$\begin{bmatrix}Ixx & Ixy & Ixy\\ Iyx & Iyy & Iyz\\ Izx & Izy & Izz\end{bmatrix}$$

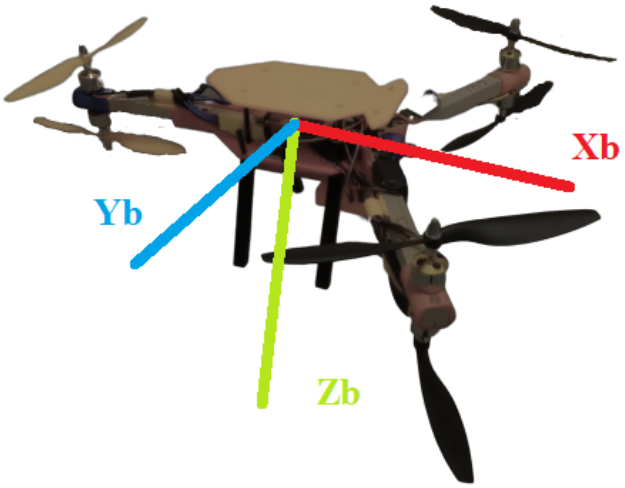


Figure 3.3: UAV with Body Frame

This creates a situation whereby Fb as shown in Figure 3.3, is inaccurate and instead, is transposed to some other orientation, meaning rotation about the centre mass will occur in an oblong manner.

To correct this, it is possible to adjust the inertia tensor by deriving some diagonalising operation such as performing an Eigenvector decomposition [14], this transformation will then need to be applied to all inertia dependant terms in the angular motion calculation.

However, for the purpose of this investigation, it is possible to operate under the assumption that the system is symmetrical, as presented by a purely diagonal inertial tensor. This is due to the off-diagonal terms in the inertia matrix being far lower in magnitude than the diagonal terms themselves, meaning that the transposed reference frame does not differ too significantly from assumed symmetrical axes, thus any deviations can be reasonably neglected [14].

3.5 Actuator Dynamics

The thrust forces and torques acting on the multirotor UAV are primarily generated by propulsion units consisting of a BLDC motor, shown in Figure 3.4, an ECS and propeller. In order to produce a useful and fully developed systems model, the characteristics of the propulsion units must be known. This can either be done via first principles analysis looking at the mechanical, electrical and aerodynamic properties of each individual component, before calculating and

Figure 3.4: A2212/13T 1000KV BLDC Motor

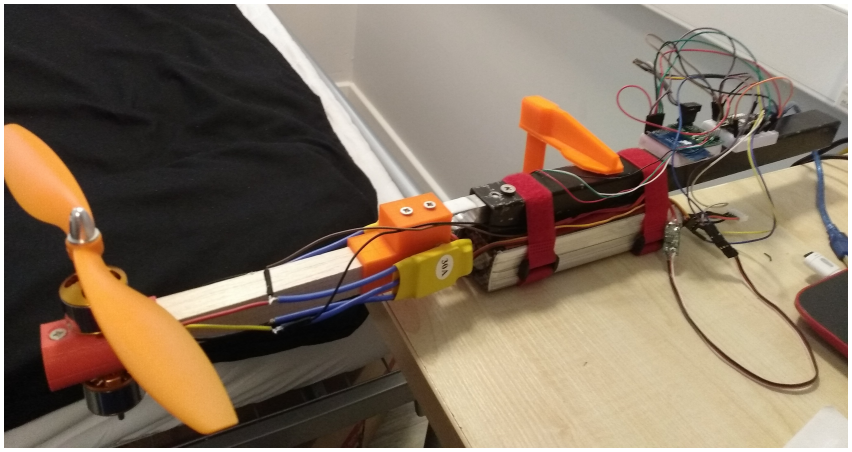


Figure 3.5: Motor Test Stand

then applying the results together in series to produce a high order, high fidelity model as is presented in [17].

However taking this approach in many cases is unnecessary. This is due to being able to derive a lumped parameter model, which views the sum total of all the individual components as a single system with singular dynamics. Therefore, using system response analysis and computational methods such as data fitting, it is possible via experimental systems identification to generate a lower order, minimum realisation model of sufficient fidelity [10],[3],[7],[12]. The systems identification approach involves the derivation of lumped parameter linear input-output models, between input PWM signal widths and the output angular velocity, torque and thrust forces respectively using a motor test stand, shown in Figure 3.5, which consists of a load cell,

for measuring forces and torques as well as an electronic revolutions per minute (RPM) sensor.

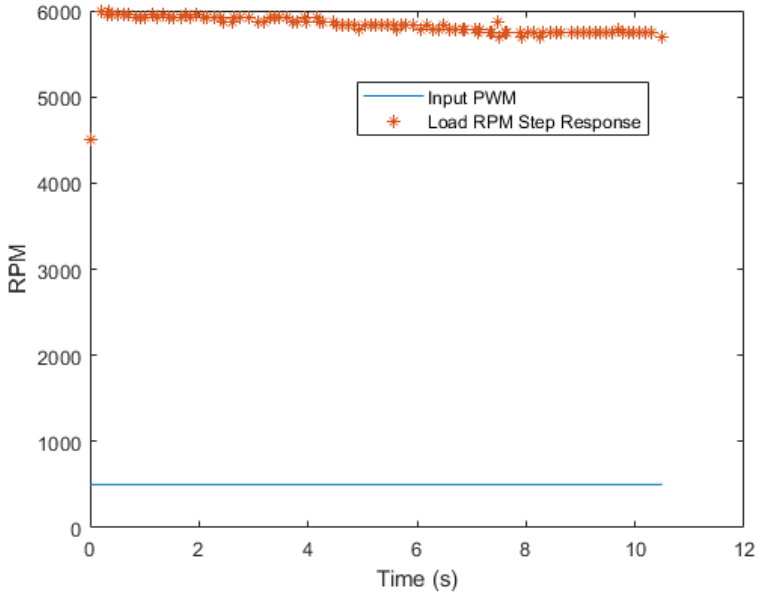


Figure 3.6: Motor Input Step Data

Using the Test stand, input-output PWM to RPM data is collected. Shown in Figure 3.6, is the response of a propulsion unit to a step input of $500\mu s$. This data was then passed through the MATLAB systems identification application where, in order to account for potential delays in the input response and produce a simple model, a first order transfer function model of the form:

$$\frac{K_u}{\tau_m s + 1}$$

was fit to the data, this transfer function can then be reformulated into a differential equation

$$\dot{\omega}_n = -\frac{1}{\tau_m}\omega_n + \frac{K_u}{\tau_m}\mu_n \tag{3.16}$$

Where, K_u, is motor gain, τ_m is the response time constant, μ_n is the input PWM signal width and ω_n is the angular velocity. Figure 3.7 shows the response of the identified model to the same step of $500\mu s$.

It is important to note that due to filtration present in the ESCs, the response time constant is sensitive to the frequency of the input PWM signal, therefore, throughout the investigation a frequency of 400 Hz is maintained. Similarly, using the test stand, as shown in Figures 3.8 and

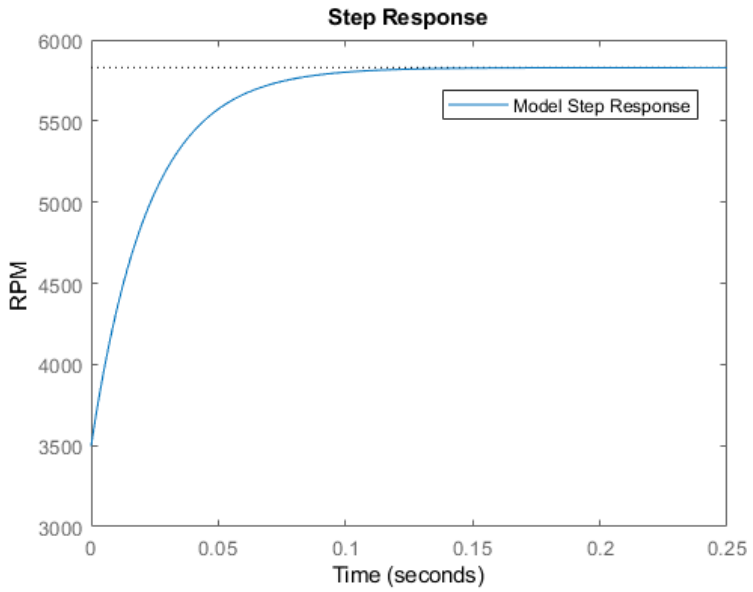


Figure 3.7: First Order Model Step Response

3.9, data displaying the relationship between the force and torque produced by the propulsion unit against generated angular velocity is shown. 2 quadratic models of the form:

$$F_n = K_a \omega_n^2 + K_b \omega_n \tag{3.17}$$

$$\tau_n = K_\tau \omega_n^2 \tag{3.18}$$

respectively are then fit to the data where ω is the angular velocity of the motor, K_a, K_b and K_τ are the force and torque coefficients respectively derived from the quadratic models. Due to the complex aerodynamic interactions from the overlapping air flow between the coaxial pairs, the actual thrust force produced is lower than would be expected from the sum of the two propulsion units [16]. This loss is represented by a gain K_w that is less than 1 and applied to all odd indexed propulsion units.

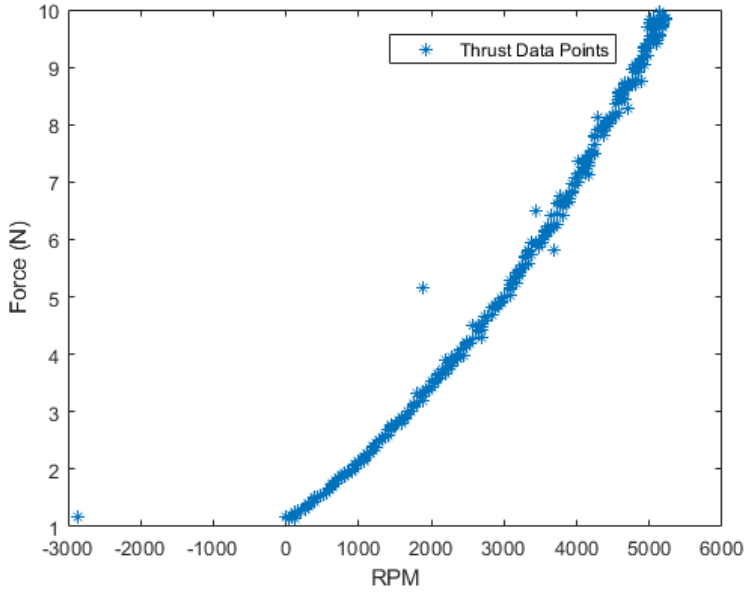


Figure 3.8: Thrust against RPM

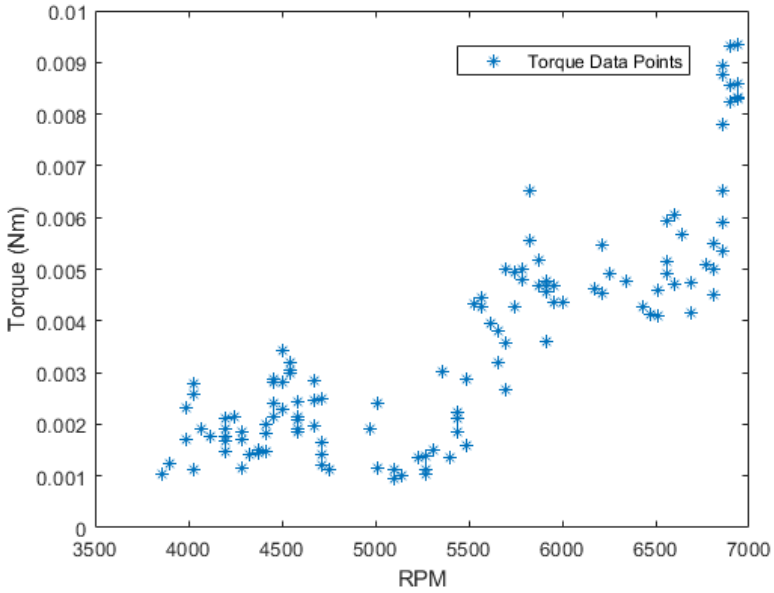


Figure 3.9: Torque against RPM

3.6 Non-Linear Model

Collating all the derived models process presents the following non-linear mathematical model describing the dynamics of the Y6 hexarotor UAV. which will be used to produce simulations of the system in MATLAB/Simulink.

Translational Motion:

$$M\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = M\begin{bmatrix}0\\0\\g\end{bmatrix} - \begin{bmatrix}C_\theta C_\psi & -C_\theta S_\psi + S_\phi S_\theta C_\psi & -S_\phi S_\psi + C_\phi S_\theta C_\psi\\ C_\theta S_\psi & C_\phi C_\psi + S_\phi S_\theta S_\psi & -S_\phi C_\psi + C_\phi S_\theta S_\psi\\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta\end{bmatrix}\begin{bmatrix}0\\0\\\sum F_n\end{bmatrix} - \begin{bmatrix}D_l xx & 0 & 0\\ 0 & D_1 yy & 0\\ 0 & 0 & D_l zz\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix}$$

Rotational Motion:

$$\begin{bmatrix}Ixx & 0 & 0\\ 0 & Iyy & 0\\ 0 & 0 & Izz\end{bmatrix}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}A_p\\ A_q\\ A_r\end{bmatrix} - \begin{bmatrix}p\\ q\\ r\end{bmatrix} \times \begin{bmatrix}Ixx & 0 & 0\\ 0 & Iyy & 0\\ 0 & 0 & Iyy\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} - \begin{bmatrix}D_r xx & 0 & 0\\ 0 & D_r yy & 0\\ 0 & 0 & D_r zz\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

Where:

$$A_p = L_1((F_1 + F_2) - (F_3 + F_4))$$

$$A_q = L_2((F_1 + F_2 + F_3 + F_4) - L3(F_5 + F_6))$$

$$A_r = \sum_{n=1}^{n=6} \tau_n$$

and motor propulsion unit dynamics are:

$$\dot{\omega_n} = -\frac{1}{\tau_m}\omega_n + \frac{K_u}{\tau_m}\mu_n$$

$$F_n = K_a\omega_n^2 + K_b\omega_n$$

$$\tau_n = K_\tau\omega_n^2$$

Then applying $\dot{\vec{E}}r = T \cdot \vec{B}r$ to convert angular velocity in Fb to angular velocity w.r.t Fe:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & S_\phi T_\theta & C_\phi T_\theta\\ 0 & C_\phi & -S_\phi\\ 0 & \frac{S_\phi}{C_\theta} & \frac{C_\phi}{C_\theta}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

3.7 Linearisation

As with the UAV dynamics presented in the previous subsection, the majority of real world systems are non-linear, in that the states of the systems evolve in some fashion that does not satisfy the criteria for linearity. However, the majority of techniques developed over the decades in systems and control theory have been developed for application to and analysis of linear systems. Therefore in order to utilise these techniques in the presence of Non-linearities it is pertinent to devise some means, a least in a limited sense, of operating with Non-linear systems as if they were linear [15]. one means of accomplishing this is through the application of the Jacobian process linearisation of non-linear systems about specific operating or equilibrium points [15],[22]. First however, several assumptions need to be made to reduce the model complexity making it more amicable to linearisation.

1. The UAV is operating in stationary hover conditions
2. The effects of drag are negligible
3. Motor dynamics are purely quadratic

Under these assumptions:

- $\ddot{\vec{E}}r \approx \dot{\vec{B}}r$
- C, D_r, and $D_l = 0$

This reduces the model down to:

$$M\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = M\begin{bmatrix}0\\0\\g\end{bmatrix} - \begin{bmatrix}C_\theta C_\psi & -C_\theta S_\psi + S_\phi S_\theta C_\psi & -S_\phi S_\psi + C_\phi S_\theta C_\psi\\ C_\theta S_\psi & C_\phi C_\psi + S_\phi S_\theta S_\psi & -S_\phi C_\psi + C_\phi S_\theta S_\psi\\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta\end{bmatrix}\begin{bmatrix}0\\0\\\sum F_n\end{bmatrix} \tag{3.19}$$

$$\begin{bmatrix}Ixx & 0 & 0\\ 0 & Iyy & 0\\ 0 & 0 & Izz\end{bmatrix}\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}L_1((F_1+F_2)-(F_3+F_4))\\ L_2((F_1+F_2+F_3+F_4)-L3(F_5+F_6))\\ \sum_{n=1}^{n=6}\tau_n\end{bmatrix} \tag{3.20}$$

$$\dot{\omega_n} = -\frac{1}{\tau_m}\omega_n + \frac{K_u}{\tau_m}\mu_n \tag{3.21}$$

$$F_n = K_a\omega_n^2 \tag{3.22}$$

$$\tau_n = K_\tau\omega_n^2 \tag{3.23}$$

Multiplying out the matrices and collecting terms produces 6 distinct 2nd order non-linear Differential, and 1 linear differential equation representing the dynamics of each motor:

$$\ddot{x} = \frac{\sum_{n=1}^{n=6} K_a \omega_n^2}{M} C\phi S\theta C\psi + \frac{\sum_{n=1}^{n=6} K_a \omega_n^2}{M} S\phi S\psi \tag{3.24}$$

$$\ddot{y} = \frac{\sum_{n=1}^{n=6} K_a \omega_n^2}{M} C\phi S\theta C\psi - \frac{\sum_{n=1}^{n=6} K_a \omega_n^2}{M} S\phi C\psi \tag{3.25}$$

$$\ddot{z} = g - \frac{\sum_{n=1}^{n=6} K_a \omega_n^2}{M} C\phi C\theta \tag{3.26}$$

$$\ddot{\phi} = \frac{L_1}{Ixx}((K_a\omega_1^2 + K_a\omega_2^2 +) - (K_a\omega_3^2 + K_a\omega_4^2)) \tag{3.27}$$

$$\ddot{\theta} = \frac{L_2}{Iyy}((K_a\omega_1^2 + K_a\omega_2^2 + K_a\omega_3^2 + K_a\omega_4^2) - \frac{L3}{Iyy}(K_a\omega_5^2 + K_a\omega_6^2)) \tag{3.28}$$

$$\ddot{\psi} = \frac{\sum_{n=1}^{n=6} K_\tau \omega_n^2}{Izz} \tag{3.29}$$

$$\dot{\omega_n} = -\frac{1}{\tau_m}\omega_n + \frac{K_u}{\tau_m}\mu_n \tag{3.30}$$

Here we can define the full state vector:

$$X = \left[x, \dot{x}, y, \dot{y}, z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6\right]$$

However for the purposes of this project the full state isn't utilised as feedback control authority in x and y are not required to maintain a stable hover. therefore, a reduced state can be defined:

$$X = \left[z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6\right]$$

3.7.1 Jacobian Method

In order to linearise the system equations, equilibrium points must be defined. These are the set of states and input values for which the system is stationary i.e:

$$\dot{X} = f(X_e, \mu_e) = 0$$

where f represents the systems dynamics,therefore, the following equilibrium input and state vectors can be defined:

$$X_e = \left[1.5, 0, 0, 0, 0, 0, 0, 0, 0, \omega_e 1, \omega_e 2, \omega_e 3, \omega_e 4, \omega_e 5, \omega_e 6\right]$$

$$\mu_e = \left[\mu_e 1, \mu_e 2, \mu_e 3, \mu_e 4, \mu_e 5, \mu_e 6\right]$$

Following this, the system states are redefined in terms of deviations from these equilibrium points:

$$\bar{X} = X - X_e$$

$$\bar{\mu} = \mu - \mu_e$$

These deviation variables are then substituted back into the dynamics equations and a first order Taylor's series expansion is performed. where:

$$\dot{\bar{X}} = f(\bar{X}, \bar{\mu}) + \frac{\partial f}{\partial X}\bigg|_{\mu=\mu_e}^{X=X_e} + \frac{\partial f}{\partial \mu}\bigg|_{\mu=\mu_e}^{X=X_e} \tag{3.31}$$

and:

$$f(\bar{X}, \bar{\mu}) = 0$$

These differential equations approximately govern the deviation variables, and as long as they remain small, this presents a linear, time-invariant, differential equation, as the derivatives of $\bar{X}$ are now linear combinations of the deviation states and the deviation inputs, $\bar{\mu}$ [15].
Finally, the Linear Continuous-Time State-Space model emerges following:

$$A = \frac{\partial f}{\partial X}\bigg|_{\mu=\mu_e}^{X=X_e}, \in R^{n\times n} \tag{3.32}$$

$$B = \frac{\partial f}{\partial \mu}\bigg|_{\mu=\mu_e}^{X=X_e}, \in R^{n\times m} \tag{3.33}$$

Where n is the number of states, m the number of inputs, A the n-by-n dynamics matrix and B the n-by-m input matrix.

Thus:

A =

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2K_a\omega_e1}{M} & \frac{2K_wK_a\omega_e2}{M} & \frac{2K_a\omega_e3}{M} & \frac{2K_wK_a\omega_e4}{M} & \frac{2K_wK_a\omega_e5}{M} & \frac{2K_wK_a\omega_e6}{M} \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2L_1K_a\omega_e1}{Ixx} & \frac{2L_1K_wK_a\omega_e2}{Ixx} & \frac{-2L_1K_a\omega_e3}{Ixx} & \frac{-2L_1K_wK_a\omega_e4}{Ixx} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2L_1K_a*\omega_e}{Iyy} & \frac{2L_2K_wK_a\omega_e2}{Iyy} & \frac{2L_2K_a\omega_e3}{Iyy} & \frac{2L_2K_wK_a\omega_4}{Iyy} & \frac{-2L_3K_a*\omega_e5}{Iyy} & \frac{-2L_3K_wK_a\omega_e6}{Iyy} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-2K_\tau\omega_e1}{Izz} & \frac{2K_\tau\omega_e2}{Izz} & \frac{2K_\tau\omega_e3}{Izz} & \frac{-2K_\tau\omega_e4}{Izz} & \frac{-2K_\tau\omega_e5}{Izz} & \frac{2K_\tau\omega_e6}{Izz} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau_m} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau_m} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau_m} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau_m} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau_m} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\tau_m}
\end{bmatrix} \tag{3.34}$$

B =

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\frac{K_u}{\tau_m} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{K_u}{\tau_m} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{K_u}{\tau_m} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{K_u}{\tau_m} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{K_u}{\tau_m} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{K_u}{\tau_m}
\end{bmatrix} \tag{3.35}$$

Where $n = 14$ and $m = 6$.

3.8 Summary

This Chapter:

- Introduced and defined the UAV system under consideration.
- Discussed and derived the mathematical equations describing the non-linear behaviour of the UAV, using both the kinematics and kinetics representations while implementing Euler angles, the Newton-Euler convention, and describing their relevance to modelling.
- Presented a means by which, through experimental identification, the key parameters that define the actuation forces may be derived, and discussed the relevance of mass distribution considerations.
- Presented a means of converting the derived non-linear dynamics into a form which could be utilised in the development of linear control systems, then developed and presented said linearised model.

Model Parameters		
Character	Name	Value
M	Mass	1.455882 Kg
Ixx	Inertia about Xb	0.014 N/m^2
Iyy	Inertia about Yb	0.028 N/m^2
Izz	Inertia about Zb	0.038 N/m^2
D_lxx	Translational Drag Coefficient in Xe	0.01212
D_lyy	Translational Drag Coefficient in Ye	0.01212
D_lzz	Translational Drag Coefficient in Ze	0.0648
D_rxx	Angular Drag Coefficient in Xb	0
D_ryy	Angular Drag Coefficient in Yb	0
D_rzz	Angular Drag Coefficient in Zb	0
L_1	Lateral Length From Motor 1 To Center Of Mass	0.19 m
L_2	Longitudinal Length From Motor 1 To Center Of Mass	0.18 m
L_3	Longitudinal Length From Motor 5 To Center Of Mass	0.30 m
K_u	Motor RPM Gain	$515.5 \times \tau_m$
τ_m	Motor RPM Time Constant	$\frac{1}{44.22}$
K_w	Quadratic RPM to Force Gain	0.85
K_a	Quadratic RPM to Force Gain	1.812e-07
K_b	Linear RPM to Force Gain	0.0007326
K_τ	Quadratic RPM to Torque Gain	7.708e-10
g	Acceleration Due To Gravity	9.81 m/s^2

Table 3.1: Table of Model Parameters

Chapter 4

Linear Quadratic Gaussian and Model Predictive Control

4.1 Introduction

The objective of this investigation is to devise control systems capable of taking the UAV to a state of stable hovering conditions. The purpose of designing and implementing control systems is to reject or minimise the effects of disturbance to the system, as well as change the characteristics of the system so as to meet performance requirements. Therefore in order to accomplish this with the UAV model derived in chapter 3, this investigation seeks to develop both LQG and an LQ-MPC control systems. These are both advanced linear model based controller that are extremely powerful and are able to nominally provide stability guarantees with wide margins. In order to develop and then implement these the following themes will be explored:

1. Discretization
2. Model Analysis
3. LQG
4. LQ-MPC

4.2 Discretization

The Linearised model obtained in chapter 3 is derived in the continuous-time domain. In order to design control systems which can be implemented unto a digital embedded microprocessor, the control laws must be in discrete-time domain.

This can be done either by designing the control systems in the continuous-time domain then converting the resulting continuous control laws into discrete ones, or by transforming the entire continuous-time model to the discrete-time domain then designing the control laws using the discrete-time system model [22]. This investigation utilises the latter method.

To convert a continuous-time state space model of form:

$$\dot{X(t)} = AX(t) + B\mu(t)$$

to the discrete-time domain Produces:

$$X(k+1) = A_{dt}X(k) + B_{dt}\mu(k) \tag{4.1}$$

where:

$$A_{dt} = e^{AT} \tag{4.2}$$

$$B_{dt} = A^{-1}(e^{AT} - I)B \tag{4.3}$$

and T is the discretization sample time, k is the time increment, and I an identity matrix of sufficient dimension.

4.3 Model Analysis

Model-Based feedback control systems design utilises insights gained from having the system model on hand to inform the design process and determine how best adjustments should be made. In order to accomplish this, the system is first thoroughly analysed and understood. Given the linear state state-space model developed at the end of chapter 3, the discretization performed in previous section, along with the values presented in the Table 3.1, it is possible to use linear system analysis techniques to arrive at key insight such as:

- Stability
- Reachability
- Observability

4.3.1 Stability

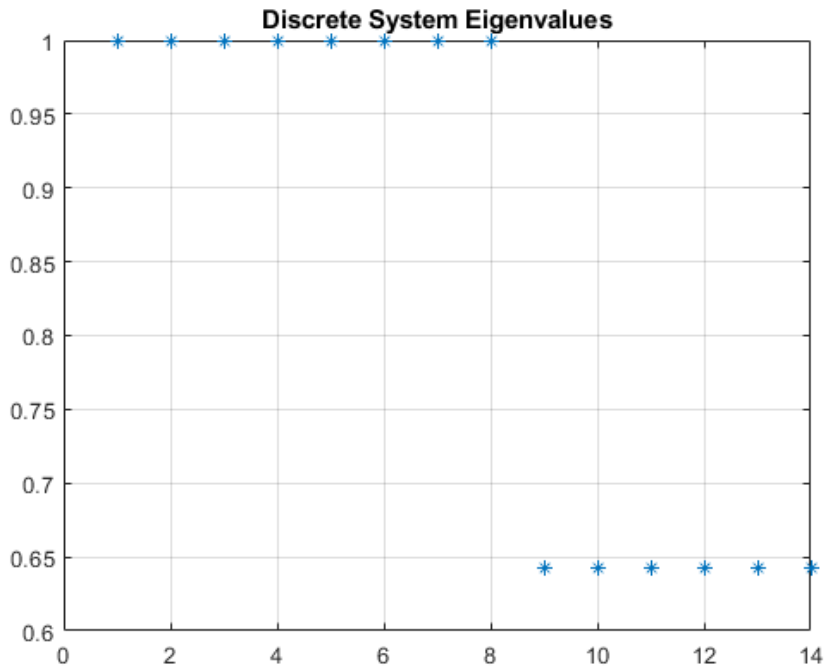


Figure 4.1: Open-Loop Eigenvalues

Stability theory plays a key role in the design of control. There are different kinds of stability problems that arise in the study of dynamical systems. The stability of systems about equilibrium operating points is usually characterised in the Lyapunov sense where an equilibrium point is said to be stable where all solutions starting nearby the equilibrium points stay nearby for all time, otherwise the system is said to be unstable [31],[22].

This definition of stability can be further specified to define stability in the asymptotic sense, where a system is said to be asymptotically stable if all solutions starting nearby the equilibrium points not only stay nearby, but also tend to the equilibrium point as time goes to infinity [31],[22].

For linear state-space systems, stability can be assertained from the locations of the eigenvalues, λ, of the A_{dt} matrix, This process is know as Lyapunov's indirect method [31] where:

$$det|A_{dt} - \lambda I| = 0 \tag{4.4}$$

Figure 4.1 reveals the presence 6 eigenvalues with a value less than 1, and 8 eigenvalues with a value of 1. For discrete-time systems, stability can be said to be asymptotic if all eigenvalues are strictly within the unit circle, Lyapunov if all eigenvalues are within or on the unit circle,

and unstable otherwise [31],[22]. Therefore the linear systems model can be said to be stable in the sense of Lyapunov. This implies also that the non-linear systems model is locally stable in the sense of Lyapunov about the equilibrium points X_e [31].

However due to real world disturbances such as model uncertainty, sensor noise and external wind disturbances, the system will not remain at the point of equilibrium save for the intervention of some control law to reject the effect of these disturbances. Hence, the system, although statically stable, is dynamically unstable.

4.3.2 Reachability

Reachability is a property which is determined by coupling the inputs into the system and the states, a system is said to reachable if for any initial set of state values X_0, and any final set of state value, X_f, there is a control signal $\mu(k)$ that can take the system from X_0 at k = 0, to X_f with some finite time [22],[25].

A system is said to be reachable if the reachability matrix:

$$W_r = \begin{bmatrix} B_{dt} & A_{dt}B_{dt} & \dots & A_{dt}^{n-1}B_{dt} \end{bmatrix} \tag{4.5}$$

is full rank equalling the number of state, n.

Thus, as the rank of W_r is equals to n, the system can be said to be reachable.

The less stringent property of stabilisability can also be defined where only the unstable states need be reachable.

4.3.3 Observability

Like reachability, Observability is a property which is determined by coupling, in this case, that between the system outputs and the system states. Therefore the system outputs, Y, must be defined where Y is a subset of the state X :

$$Y \in X$$

where:

$$Y = [z, \phi, \theta, \psi] \tag{4.6}$$

Therefore, a matrix to perform the state selection is defined as:

$$Y(k) = CX(k) \tag{4.7}$$

$$C \in R^{p \times n}$$

where p is the number of outputs.

Thus we define:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.8}$$

$$p = 4$$

where:

$$C_{dt} = C$$

A system is said to observable if for all time, it is possible to, from a limited set of state measurements $Y(k)$, and knowledge of the inputs $\mu(k)$, to *uniquely* obtain the full state, X(k) [22].

A system is said to be observable if the observability matrix:

$$W_o = \begin{bmatrix} C_{dt} \\ C_{dt}A_{dt} \\ \vdots \\ C_{dt}A_{dt}^{n-1} \end{bmatrix} \tag{4.9}$$

is full rank equalling n.

Again as with reachability, a less stringent property, detectability, can also be defined where only the unstable states need be observable.

Interestingly, for this combination of dynamics A_{dt} and C_{dt}, it is found that the W_0 is not full rank, and therefore the system in its current form is not observable. Upon further analysis, it is discovered that this is as a result of a phenomenon created due to the Coaxial propulsion unit configuration, as there are numerous combinations of thrust force values that each individual propulsion unit can generate which produce the same resultant torque about the centre of mass making it impossible to obtain a unique full state. One possible solution is to modify the UAV design to include RPM sensors to measure at least 2 non-coaxial motors. However, in the upcoming LQG section, a more elegant solution is proposed.

4.4 LQG

4.4.1 Full-State Feedback

Given a discrete-time linear state-space model of form:

$$X(k+1) = A_{dt}X(k) + B_{dt}\mu(k)$$

$$Y(k) = C_{dt}X(k) \tag{4.10}$$

Full-state feedback is a control technique which makes use of the full-state, $X(k)$ of a system, to modify the system's eigenvalue so as to change its characteristic responses[22]. This is done by determining some set of gains K which when applied to the system states, the resultant of which being fed back into the system as inputs, adjust the dynamics of the system, where:

$$\mu(k) = -KX(k) \tag{4.11}$$

This leads to a modified closed loop system:

$$X(k+1) = (A_{dt} - B_{dt}K)X(k) \tag{4.12}$$

Where the eigenvalue are now determined by:

$$det|(A_{dt} - B_{dt}K) - \lambda I| = 0 \tag{4.13}$$

4.4.2 LQR

LQR is a technique for finding the set of gains K, in a manner which is in some wise optimal [22],[23],[24],[25]. In order to accomplish this, LQR seeks to find a control sequence μ to minimise a *quadratic* cost function by formulating the control problem as optimisation problem $P_\infty(X(0))$, where :

$P_\infty(X(0)) =$

$$min_\mu V_\infty(X(0),\mu) = \sum_{k=0}^{k=\infty} X^T QX + \mu^T R\mu \tag{4.14}$$

s.t

$$X(k+1) = A_{dt}X(k) + B_{dt}\mu(k), k = 1,2,3\ldots$$

Where $Q^{n\times n}$, and $R^{m\times m}$, are 2-norm weighing matrices on the states and inputs respectively. These weight are selected via trial and error.

Such an optimisation over an infinite horizon, as $k \to \infty$, would be intractable, however through the application of the Dynamic Programming (DP) formulated from Bellman's principle of optimality, the solution emerges as a result of the derivation of a set of Discrete Algebraic Riccati Equation (DARE) [25] where:

$DARE =$

$$K = -(R + B_{dt}^T \Pi B_{dt})^{-1} B_{dt}^T \Pi A_{dt}$$

$$\Pi = Q + A_{dt}^T A_{dt} - A_{dt}^T \Pi B_{dt} (R + B_{dt}^T \Pi B_{dt})^{-1} B_{dt}^T \Pi A_{dt} \tag{4.15}$$

In order for the optimisation problem to be feasible, certain criteria must be satisfied [22],[25]:

- Q be at least Positive Semi-definite (P.S.D) and R be strictly Positive Definite (P.D).
- (A_{dt}, B_{dt}) must be at least stabilisable.
- (C_{dt}, A_{dt}) must be at least detectable.

4.4.3 Reference Tracking and Integral Action

The control problem as set up in the previous LQR subsection is one capable of realising a control system with the goal of driving all system states to the equilibrium points. However, in order for the control system to be usefully implemented on a UAV, it must be able to track and set the system outputs Y to, a desired reference target, r, preferably with no errors asymptotically.

Therefore, to accomplish this, the system's model must be augmented to include behaviour which seeks to minimise errors between a reference target and UAV output, this is done through the addition of integration states. hence:

$$X_{aug} = \begin{bmatrix} X \\ X_I \end{bmatrix} \tag{4.16}$$

Where $X_I = \mathrm{r} - Y$:

$$X_I = \begin{bmatrix} Z_I \\ \phi_I \\ \theta_I \\ \psi_I \end{bmatrix} \tag{4.17}$$

This Leads to augmented discrete state-space Matrices:

$$A_{aug} = \begin{bmatrix} A_{dt} & 0 \\ -C_{dt}A_{dt} & I \end{bmatrix}, B_{aug} = \begin{bmatrix} B_{dt} \\ -C_{dt}B_{dt} \end{bmatrix}, C_{aug} = \begin{bmatrix} C_{dt} & 0 \end{bmatrix} \tag{4.18}$$

Where 0 is a square zeros matrix of dimension P and I is an identity matrix of dimension p. Solving the LQR optimisation problem with, A_{aug},B_{aug},C_{aug} and $Q^{n+p\times n+p}$, given this augmented system, produces a control law [23]:

$$\mu(k) = -KX(k) + K_I X_I(k) \tag{4.19}$$

Where:

$$X_{aug}(k+1) = (A_{aug} - B_{aug}K_y)X_{aug}(k) + B_{aug}K_I\mathrm{r}(k) \tag{4.20}$$

$$K_y = \begin{bmatrix} K & -K_I \end{bmatrix}$$

This is implemented in the MATLAB-Simulink environment while taking into account the SBC dynamics such the effects of quantisation from digital to analog and visa-versa conversions which are always present when digital controllers that interface with real analogue continuous-time systems:

4.4.4 State Estimation

The control law devised in LQR subsection was produced under the assumption that the full state of the system, X, is available for full-state feedback, however, this is not the case as only a limited number of state measurements are available and these form the measured outputs, Y. Therefore, in order to utilise a full-state feedback control law, the full-state of the system must be estimated from these limited measurements present. This is performed through the

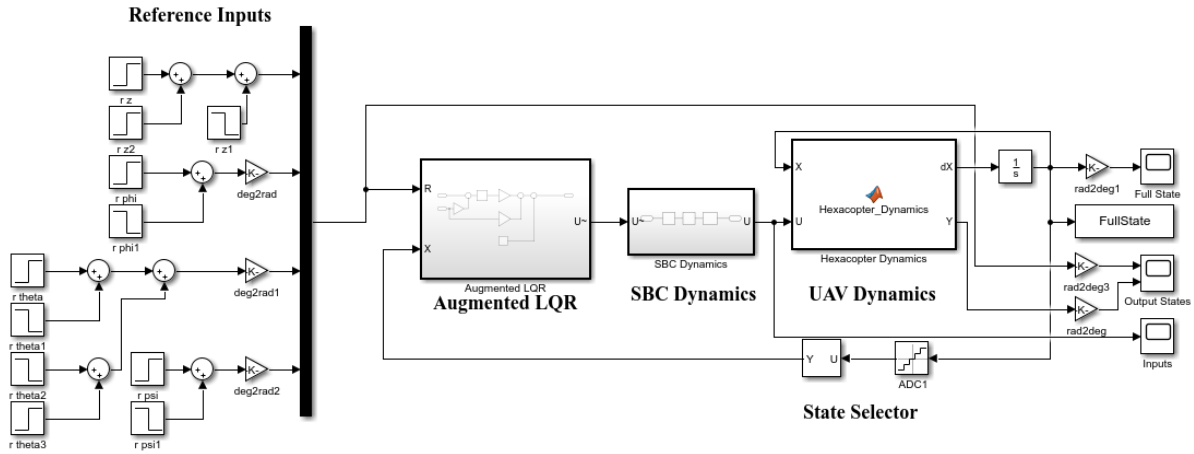


Figure 4.2: LQR Simulink Implimentation

implementation of a State Observer/Estimator. State estimators accomplish this by applying knowledge of the system dynamics through the utilisation of the system's model and its observability property [22],[18],[19],[20],[21].

The estimator mimics the system dynamics giving:

$$\hat{X}(k+1) = A_{dt}\hat{X} + B_{dt}\mu(k) + Le$$
$$\hat{Y}(k) = C_{dt}\hat{X} \tag{4.21}$$

where $\hat{X}$ and $\hat{Y}$ are the estimated system states and outputs, L is the estimator gain with e being the output error, which is the difference between the estimator model and the system outputs.

$$e = Y - \hat{Y}$$

As with the systems model, the behaviour of estimator model and thus the quality of the estimation is determined by the state estimation error dynamics [22], where:

$$\hat{X_e} = X - \hat{X}$$

The open-loop estimator dynamics are given by:

$$\hat{X_e}(k+1) = A_{dt}(X - \hat{X}) \tag{4.22}$$

with implementation of the observer gain L, the closed-loop estimator dynamics are give by:

$$\hat{X_e}(k+1) = (A_{dt} - LC_{dt})\hat{X_e} \tag{4.23}$$

If the eigenvalues of the estimator exist outside the unit circle, then the estimator is unstable and the state estimation is be divergent, and if the system is observable, then it is always possible to derive some gain L, to produce any set of desired eigenvalues [22]

4.4.5 Estimator Modification

In the earlier observability subsection, it was found that the system is not observable due to propulsion unit configuration, and the emergent thrust and torque combinations making the estimation of a unique full state impossible.

This can be rectified simply be making the system observable by modifying the hardware and measuring the value of some of the motor states via RPM sensors.

However, there is another way this can be solved without the need to obtain additional sensors, that is, by virtually augmenting the output, Y to include terms for at least 2 non coaxial motor pairs, then returning an error vector, e ,where the error value for those 2 motors is 0, therefore:

$$C_y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, Y_y = \begin{bmatrix} Z \\ \phi \\ \theta \\ \psi \\ \omega_1 \\ \omega_4 \end{bmatrix}, e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ 0 \\ 0 \end{bmatrix} \tag{4.24}$$

This augmentation has the effect of using the model to completely determine the value of the motor angular velocity by assuming the model is a perfect representation of the actual system. This augmentation is done solely for the purpose of state estimation so there are still only 4 reference input to the controller and only 4 measured outputs, but with this, the system is now fully observable.

The Kalman Filter

Kalman Filtering encompasses a series of techniques for obtaining an estimator gain L, which is in some wise optimal. This is accomplished by constructing an optimisation problem with

the objective of minimising a set of noise measurement covariances which arise from disturbances to the system such those due to model uncertainty and noise present in the output sensor readings, these sources of noise are modelled as Gaussian noise distributions. Thus, when used in combination with the LQR, the combination is called the Linear Quadratic Gaussian.
The Kalman Filter problem is a corollary problem to the LQR, where optimisation is also performed over infinite horizon, requiring the application of DP, and is thus derived resulting in another DARE [22],[18].
There are 2 major implementations of the Kalman filter which are popularly employed.

- The Linear Kalman Filter (LKF)

- The Extended Kalman Filter (EKF)

The process described above only fully pertains to the design of an LKF. The EKF is used in estimating the states of non-linear systems, utilising the non-linear system model directly but then implementing an online lineariser that linearises the system at each time interval, k, to produce a linear model which is then used to solve for the estimator gain L as stated above [18],[19],[243].
However, through experimentation with the estimator model, it was found that the systems states could be estimated by modifying the LKF to use the non linear model derived in chapter 3, similar to the EKF. This modified LKF would then use the linear model to derive the estimator gain L and the full state estimations would be derived from applying this gain to the discretised non-linear model.
This is possible due to the limited operational scope where the system is not expected to deviate far enough from the equilibrium operating points about which the non-linear model was linearised as presented in chapter 3. Therefore this method could be called a Limited EKF of sorts without the online linearisation.
This Limited EKF modification results in more accurate state and output predictions than the standard LKF implementation, as shown when the errors between the non-linear model outputs and estimator outputs are compared, as shown in Figures 4.3 and 4.4 during a 100 second simulation where the system is subjected to step inputs.

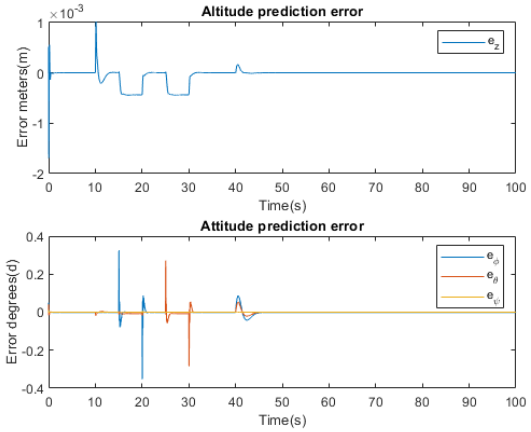


Figure 4.3: KF Error

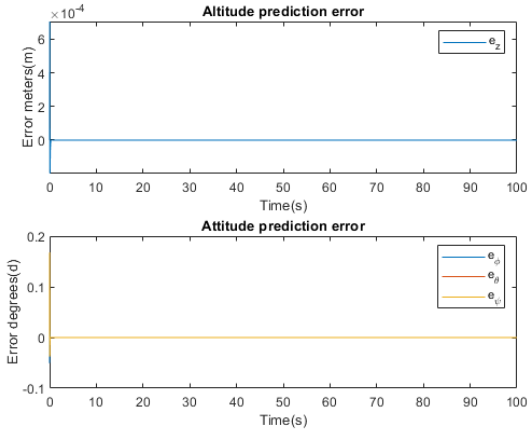


Figure 4.4: Limited EKF Error

4.4.6 Results

Figures 4.5 shows the closed-loop system eigenvalues, where after implementation of the designed LQG controller, all eiganvalues are now in within the unit circle, thus stabilising the system. Figures 4.6 to 4.8 show response plots of the system inputs, and both the estimated, with Limited EKF, and measured outputs respectively, as the simulation is run over an interval of 100 seconds. The system is subjected to reference step inputs where the results are displayed in Table 4.1:

Step Responses					
Output	Reference	Overshoot (%)	Settling Time (s)	Rise Time (s)	Steady State Error
Z	1 meter	< 5	3	< 2	0
ϕ	30 degrees	< 5	< 1	< 0.5	0
θ	30 degrees	< 5	< 1	< 0.5	0
ψ	90 degrees	0	< 5	< 3	0

Table 4.1: Table of LQG Step Response values

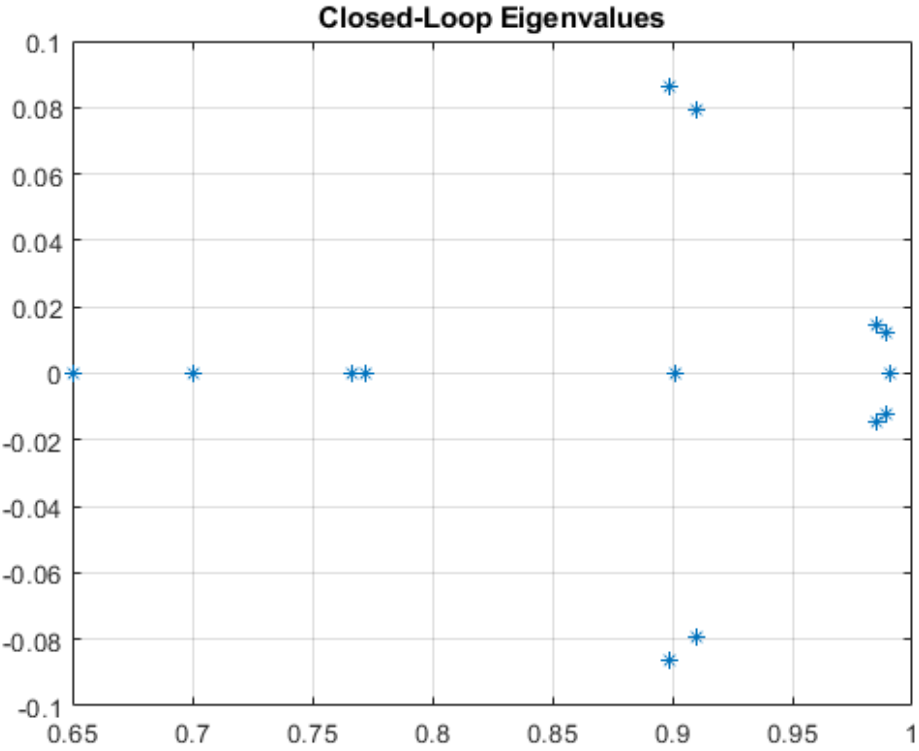


Figure 4.5: Closed-Loop Eigenvalues

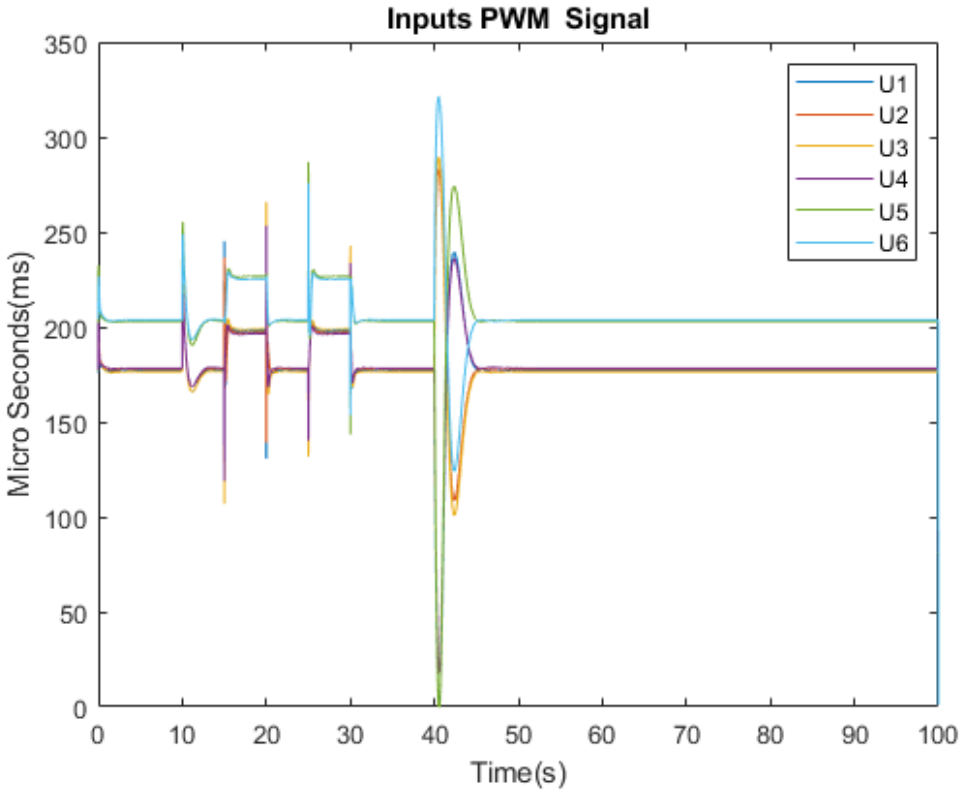


Figure 4.6: System Input PWM Signal

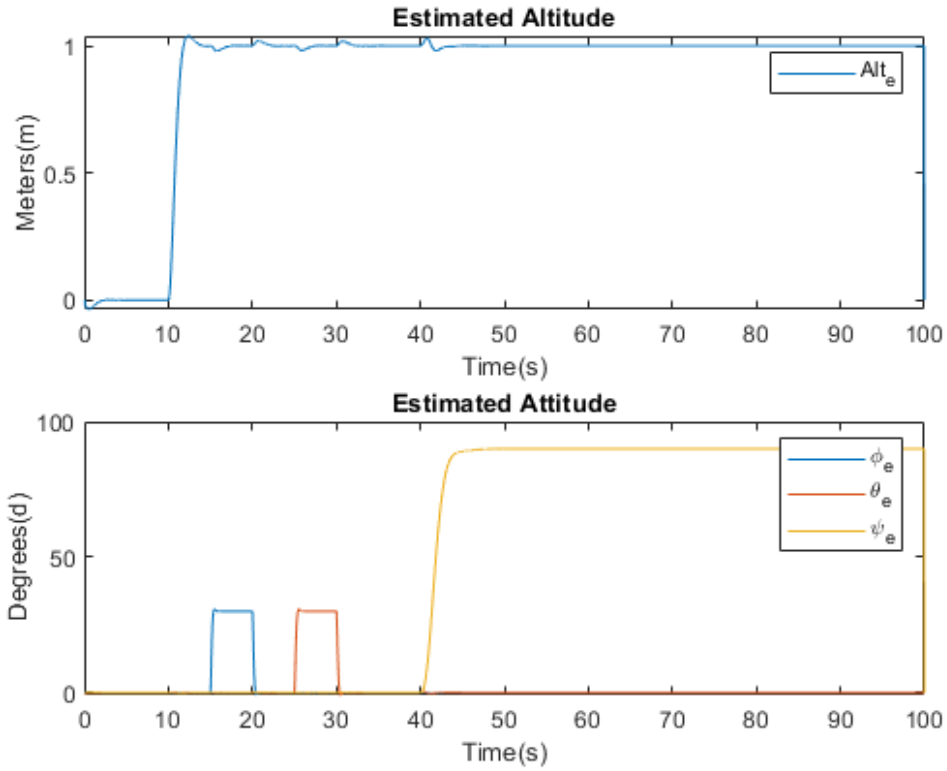


Figure 4.7: System Estimated Outputs

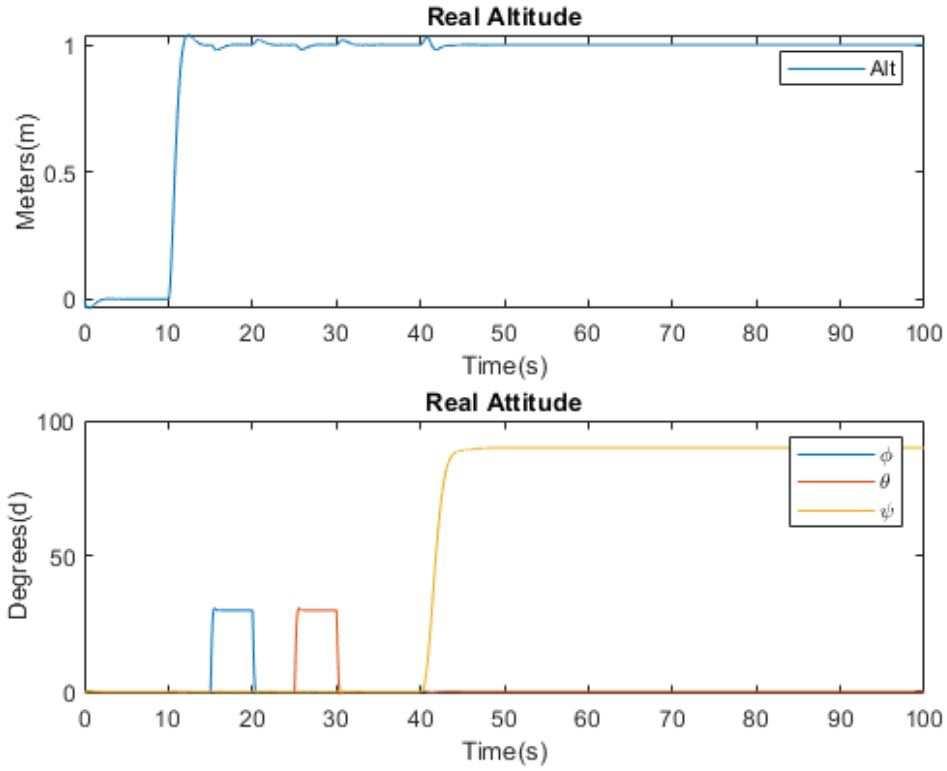


Figure 4.8: System Measured Outputs

4.5 LQ-MPC

4.5.1 Limitations of LQG control

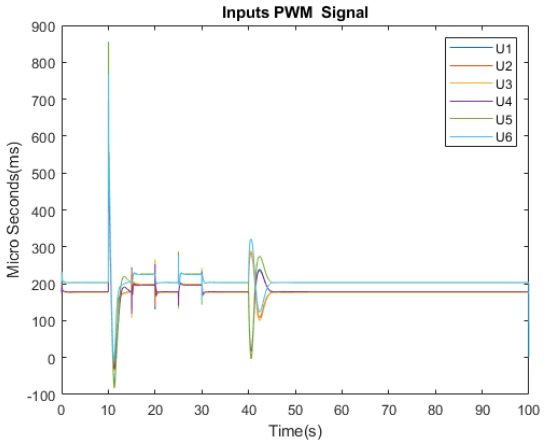


Figure 4.9: Input Constraint Violation

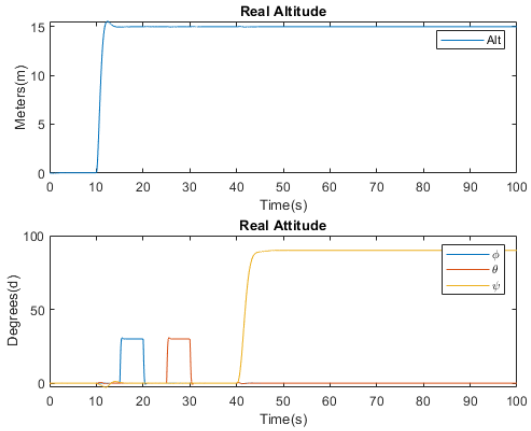


Figure 4.10: Output Constraint Violation

The LQG is a powerful control mechanism, however, there is one major drowback the using LQG control and that is the inability to taking into account both physical and performance constraint placed on the system states and inputs. This is shown in Figures 4. and 4. where the controller demands and inputs in excess of 800 μ_s and below 0 μ_s when a step change of 15 meters in z is demanded of the system, where 0 to 800 μ_s is the physical systems input operating range. This could potentially be hazardous if the application where safety critical and the controller were to violate a constraint.

Like LQG, MPC techniques are also part of the of optimal control family, however, with MPC implementations, the possibility exists where control signals can be determined such that they implicitly taking into account and produce control action so as to meet system constraints.

4.5.2 General Model Predictive Control (GPC)

MPC does not refer to any one algorithm or method but instead a methodological philosophy with which to approach control problems, containing certain key concepts and ideas implemented in various ways [26]. There are 3 main components in a GPC algorithm:

- Prediction
- Optimisation

- The Receding Horizon Principle

The prediction stage of a GPC algorithm uses the model to predict the future system states of the system, if provided with some sequence of control actions. This is dervived by recursively implementing the systems model unto itself to produce a higher order prediction model over a specified horizon length [25],[26],[27]. Given a system of the equation 4.10 form, a state prediction model for a horizon of length N, and evaluated at a time instance k, can be defined as:

$$\bar{X}(k) = FX(k) + G\bar{\mu}(k) \tag{4.25}$$

where:

$$\bar{X}(k) = \begin{bmatrix} X(k+1|k) \\ X(k+2|k) \\ \vdots \\ X(k+N|k) \end{bmatrix}, \bar{\mu}(k) = \begin{bmatrix} \mu(k|k) \\ \mu(k+1|k) \\ \vdots \\ \mu(k+N-1|k) \end{bmatrix}, F = \begin{bmatrix} A_{dt} \\ A_{dt}^2 \\ \vdots \\ A_{dt}^N \end{bmatrix}, G = \begin{bmatrix} B_{dt} & 0 & \dots & 0 \\ A_{dt}B_{dt} & B_{dt} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_{dt}^{N-1}B_{dt} & A_{dt}^{N-2}B_{dt} & \dots & B_{dt} \end{bmatrix} \tag{4.26}$$

The Optimisation stage seeks to formulate and solve an optimisation problem $P_N(X(0))$, to obtain the minimising control sequence over the whole horizon $\bar{\mu}$, evaluated at each time step k, where: $P_N(X(0)) =$

$$min_{\mu} V_N(X(0), \bar{\mu}) = \sum_{i=0}^{i=N-1} X^T(k+i|k)QX(k+i|k) + \mu^T(k+i|k)R\mu(k+i|k) \tag{4.27}$$

s.t

$$X(k|k) = X(k)$$

$$X(k+1+i|k) = A_{dt}X(k+i|k) + B_{dt}\mu(k+i|k), i = 1,2,3\dots,N-1$$

Where $Q^{n\times n}$, and $R^{m\times m}$, are 2-norm weighing matrices on the states and inputs respectively. If equation 4.25, is substituted into 4.27, this puts 4.27 into the standard form of a quadratic programming problems where:

$$V_N(X(0), \bar{\mu}) = min_{\mu} \frac{1}{2}\bar{\mu}^T(k)H\bar{\mu}(k) + C_q^T\bar{\mu}(k) + \alpha \tag{4.28}$$

and:

$$H = 2(G + \tilde{Q}G + \tilde{R})$$

$$C_q = LX(k), L = 2G^T\tilde{Q}F$$

$$\alpha = X(0)^T MX(0), M = F^T\tilde{Q}F + Q \tag{4.29}$$

With $\tilde{Q}$ and $\tilde{R}$ being $N-1 \times N-1$ dimensional matrix diagonal matrices, with the Q and R matrices as diagonal terms respectively.

Solving this Quadratic Program produces a 1 by *mxN* optimal control sequence of form:

$$\bar{\mu} = -H^{-1}LX(k) \tag{4.30}$$

where the optimal control gain, K_N, consists of the first $m \times n$ terms in the $-H^{-1}L$ due to the implementation of the receding horizon principle.

This is the reason MPC implementations are also known as receding horizon control (RHC) as they only perform the first set of control actions in the sequence $\bar{\mu}$, before recalculating and performing this action at every time interval, k, the target is reached. This continual updating provides feedback into the system [25],[26],[27].

4.5.3 LQ-MPC: Extending The LQR

The control law derived,K_N in the previous subsection as opposed to the LQR control law K, derived in the previous section, is not necessarily stabilising even under nominal conditions. This is due to the complex interaction between N,Q,R. [25],[26]. In order to mitigate this, the conditions at the terminal state, k+N, must be made to behave in a similar way as the infinite horizon LQR.

The is accomplish through the addition of an optimal terminal cost, P, to the cost function $P_N(X(0))$. This then becomes:

$$min_{\mu} V_N(X(0), \bar{\mu}) = \sum_{i=0}^{i=N-1} [X^T(k+i|k)QX(k+i|k) + \mu^T(k+i|k)R\mu(k+i|k)] + X^T(k+N|k)PX(k+N|k) \tag{4.31}$$

s.t

$$X(k|k) = X(k)$$

$$X(k+1+i|k) = A_{dt}X(k+i|k) + B_{dt}\mu(k+i|k), i = 1,2,3\ldots,N-1$$

This also changes$\tilde{Q}$ to include P as another diagonal term, making it $N \times N$ dimensional.

The value of P is derived through satisfying the application of the Lyapunov equation:

$$(A_{dt} + BK)^T P(A_{dt} + BK) - P = T \tag{4.32}$$

Where T is the infinite horizon cost and K is the LQR gain. Therefore, with the addition of P, if K is stabilising then, K_N is guaranteed asymptotically stabilising in the unconstrained case [26]. provided the system is

- Stabilisable
- Observable
- Q is at least P.S.D and R is P.D

This form of MPC is also know as dual mode-MPC as the controller is in effect switching between one set of control laws when far from the steady state, k < N, and another when near it, K $\geq$ N. [25],[26],[27].

4.5.4 Closed-Loop Predictions

The Closed-loop paradigm embeds the optimal stabilising controller gains, K, into the prediction model of the system, pre-stabilising and in doing to defines a new control variable, c, a perturbations about the embedded most desired performance where:

$$\mu(k+i-1) = -KX(k+i-1) + c(k+i-1), i = 1,2,\ldots,N \tag{4.33}$$

This paradigm is known as Optimal-MPC (OMPC) and produces a control implementation structure of the form shown in Figure 4.11, [26].

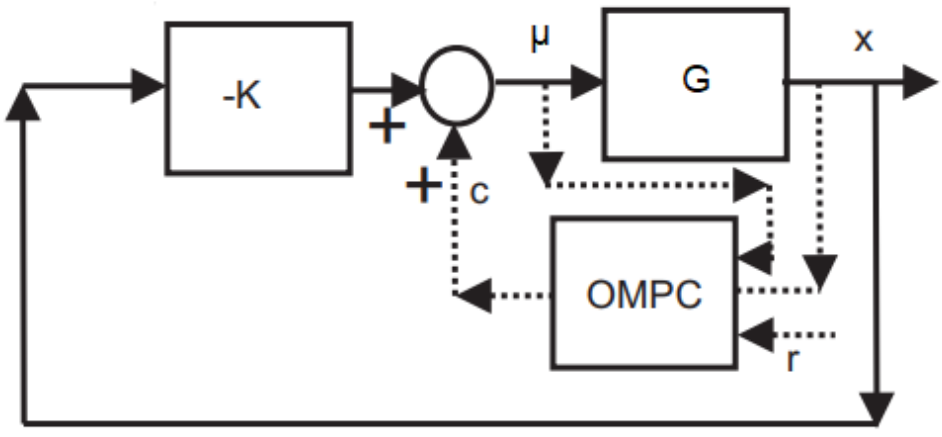


Figure 4.11: OMPC Implementation Structure

When equation 4.33 is substituted into equations 4.25 and 4.32, this produces the quadratic programming problem:

$$min_c V_N(X(0)) = c^T(k)S_c c(k) + 2c^T(k)S_{cx}X(k) \tag{4.34}$$

Which when solved produces:

$$\bar{c}(k) = -S_c^{-1}S_{cx}X(k) \tag{4.35}$$

This provide the closed-loop paradigm control law in the unconstrained case:

$$\mu(k) = -[K + S_c^{-1}S_{cx}]X(k) \tag{4.36}$$

The implementation of the Closed-loop paradigm is especially important when dealing with unstable systems,such as multirotors, as large prediction horizons N, the prediction matrices, F and G, may become numerically ill-conditioned leading to control implementation issue especially in embedded systems with limited computational power. However, the incorporation of the stabilising feedback K, into the prediction matrices changes their conditioning[26],[27].

4.5.5 Reference Tracking

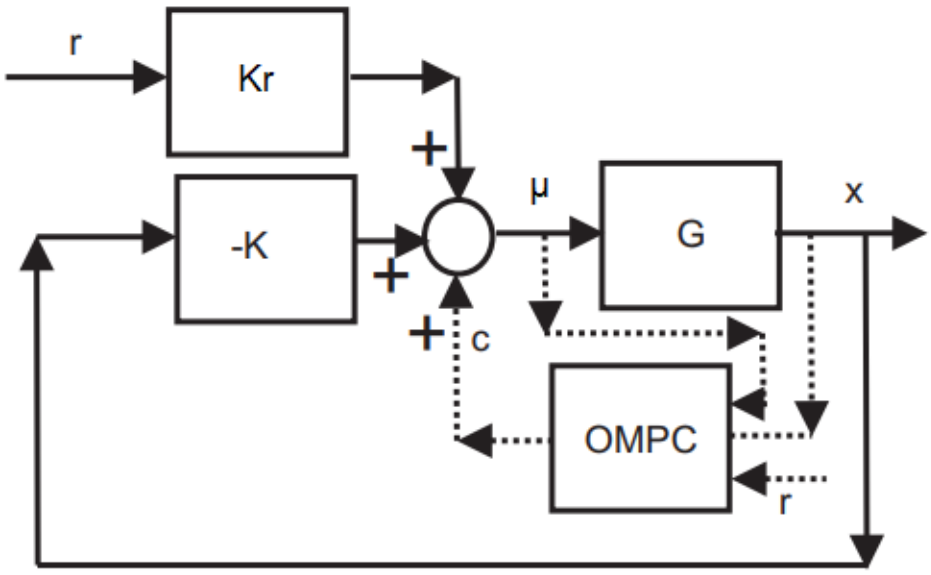


Figure 4.12: OMPC Implementation Structure

Reference tracking is achieved through the implementation of deviation variable. These are the state and input values at the desired target state where the equilibrium inputs μ_e, and the

reference signal r, make up these variable. where we define an $1 \times n$ vector Xr, which contains the output references and state equilibrium point:

$$\begin{bmatrix} r_1(k) & 0 & r_2(k) & 0 & r_3(k) & 0 & r_4(k) & 0 & \omega_e 1 & \omega_e 2 & \omega_e 3 & \omega_e 4 & \omega_e 5 & \omega_e 6 \end{bmatrix} \tag{4.37}$$

A control law of form:

$$\mu(k) = -[K + S_c^{-1} S_{cx}](X(k) - Xr) + \mu_e \tag{4.38}$$

In the unconstrained case, S_{cx} = 0, therefore the unconstrained Closed-Loop paradigm LQ-MPC, results in state feedback [26].

4.5.6 Results

Figures 4.13 to 4.16 show response plots of the system inputs, system perturbations, and both the estimated, with Limited EKF, and measured outputs respectively, as the simulation is run over an interval of 50 seconds. The system is subjected to reference step inputs where the results are displayed in Table 4.2:

Step Responses					
Output	Reference	Overshoot (%)	Settling Time (s)	Rise Time (s)	Steady State Error
Z	1 meter	< 5	3	< 2	0
ϕ	30 degrees	< 5	< 1	< 0.5	0
θ	30 degrees	< 5	< 1	< 0.5	0
ψ	45 degrees	0	< 5	< 3	0

Table 4.2: Table of LQ-MPC Step Response values

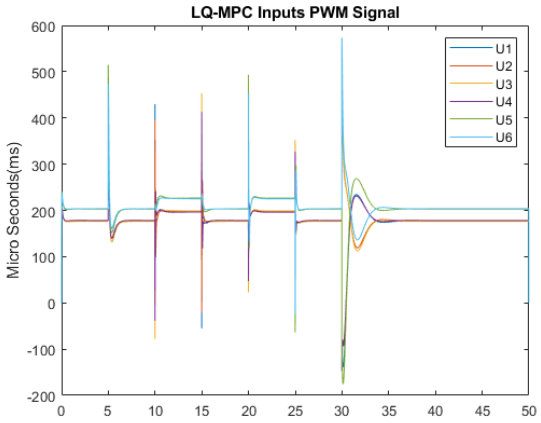


Figure 4.13: LQ-MPC Unconstrained Inputs

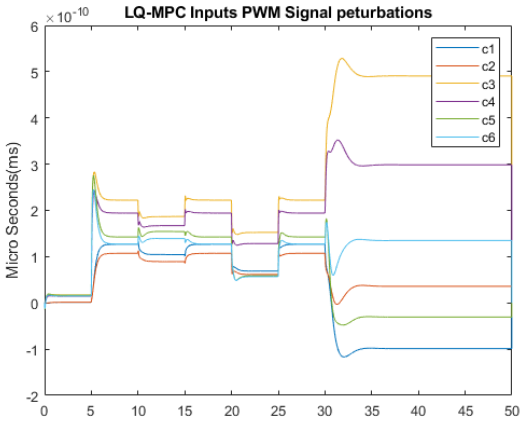


Figure 4.14: LQ-MPC Unconstrained Perturbations

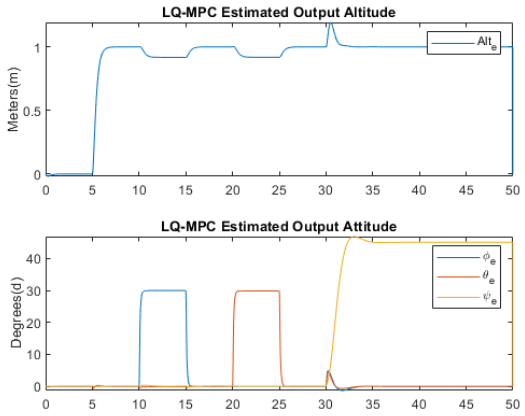


Figure 4.15: LQ-MPC Estimated Outputs

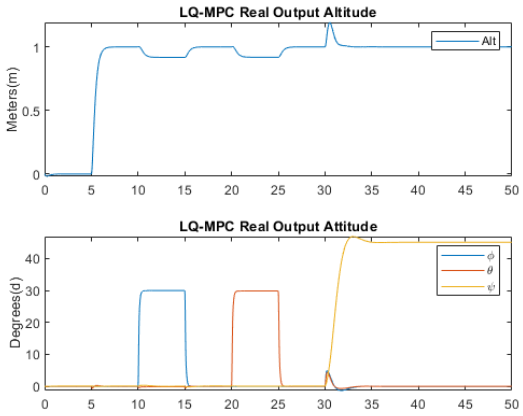


Figure 4.16: LQ-MPC Model Outputs

4.6 Summary

This Chapter:

- Converted the Linearised model into a for suitable for control analysis and design
- Analysed key system properties such as its stability in open loop, reachability and observability
- Discussed and derived full-state feedback control and the application of the LQR producing optimal feedback
- Discussed the necessity for state estimation and proposed a modified Kalman filter for state estimation thus producing an LQG controller whose performance was tested and

results presented

- Discussed limitations of the LQG and presented and derived an LQ-MPC controller as a solution, whose performance was tested and results presented.

Chapter 5

Embedded Control Systems Implementation

5.1 Introduction

Advanced control systems need to be implemented using digital computers. In an application such as with UAVs, this computation needs to occurs on a devices as physically small as possible, due to design consideration such as minimising total vehicle mass. One such class of applicably small computers are SBCs. These are usually fully capable computers in their own right running light weight operating systems, such as some version watered down Linux distributions. Many typically have smaller physical dimension comparable to that of debit cards. Such devices have become possible and previlent due to the advent of smartphone technology driving the trend towards cost effective minimisation of processing units.
This chapter will go through the considerations and processes involved in implementing control systems and propose an implementation, via a systems engineering framework, of a flight control system. These considerations are:

1. Hardware
2. Sensors
3. Software Libraries
4. UML Implementation

5.2 Hardware

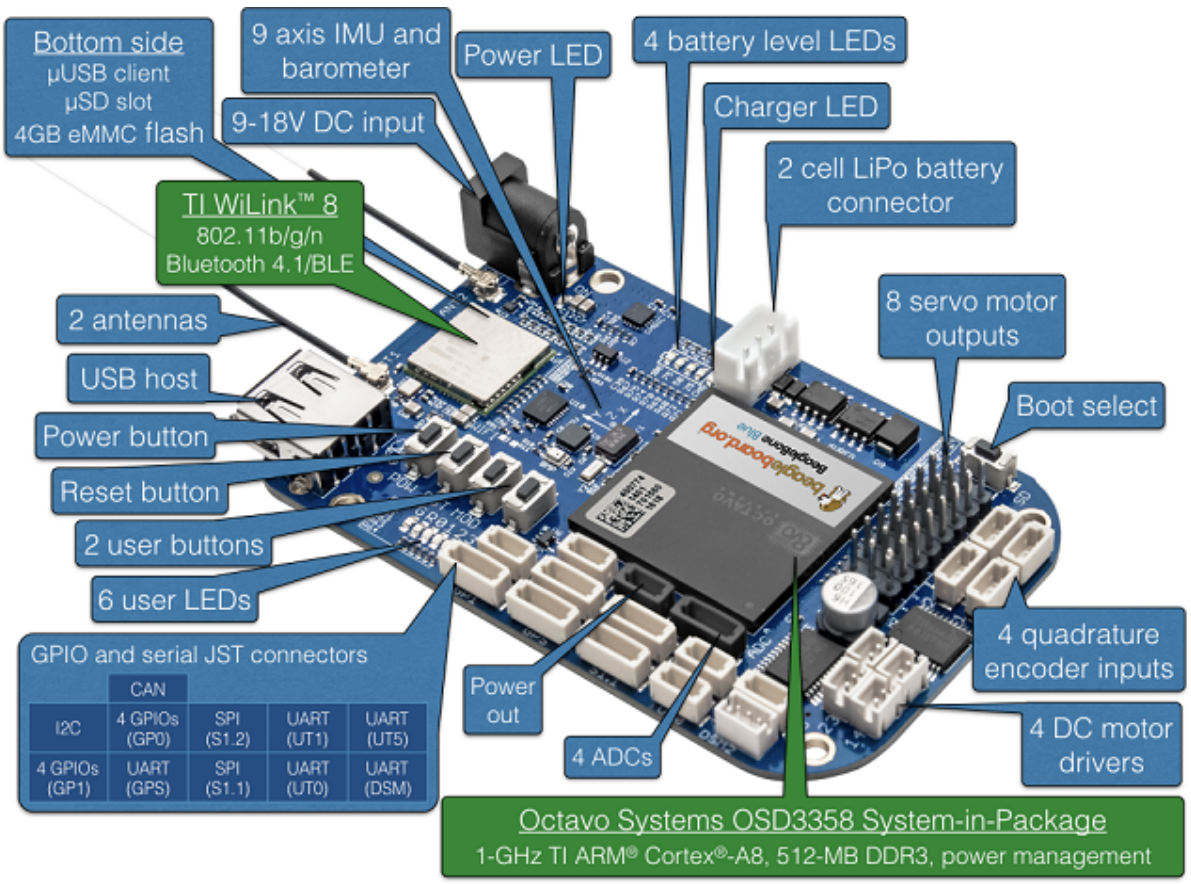


Figure 5.1: BeagleBone Blue SBC

The SBC under consideration for the project is the Beagle Bone Blue (BBB) from the BeagleBoard group, shown in Figure 5.1, which, in addition to the computation provided by all SBCs, has been developed with many additional features that make is suitable for implementation in robotics.

The BBB possesses a 1 Gigahertz ARM Cotrex-A8 mirco-processor unit, supported by 512 Megabytes of DDR3 random access memory (RAM). This SCB is also capable of wireless communication through various Bluetooth and Wi-Fi standards. However for the purposes of this investigation all communication to the SBC will occurs via a radio transmitter to a waiting receiver onboard the UAV. This transmitter will effectively be sending the reference commands, r, to the controller which will be read in as Pulse Position Modulated (PPM) values by the SBC. The BBB comes equipped with multiple, peripheral communication interfaces such as:

- Inter-Integrated Circuit (I^2C)
- Serial Periheral interface (SPI)
- Genral Purpose Input-Outputs (GPIOs) at 3.3v Logic Level (V_{op})
- Universal Asynchronous receiver-Transmitter (UART)

The BBB also comes equipped with multiple sensor, the addition of which, are the most pertinent hardware inclusions with regards to this investigation. These are:

- A Magnetic, Angular rate, Gravity Sensor (MARG)
- A Barometric Pressure Sensor (Barometer/Baro)

5.2.1 MARG and Barometer

A MARG sensor consists of a 3-axis gyroscope, which measures angular rate, a 3-axis accelerometer which measure linear acceleration, and a 3-axis magnetometer which measures the strength of the earth's magnetic field.
Barometers are sensors which measure atmospheric pressure. their use on UAVs is key as pressure on the surface of the earth is a function of the altitude, therefore the pressure readings from a barometer can be used to estimate the altitude above sea-level of a body [24],[29].
When combined, the sensors can be used to estimate the angular position and altitude of the body they are attached to. In the form factor able to be implemented on an SBC, MARGs come as mirco elctro-mechanical systems (MEMS) sensors which take the form of integrated circuits. MEMS devices contain moving masses and are able to convert the movement of these masses into readable data [24],[29]. The MARG implemented here is the MPU-9650 from innovosense and the Barometer is the BPM-280 from BOSH.
In order to be used these devices they must be calibrated to remove bias and offset errors due to the effect of external disturbances, especially on MEMS devices [29].

5.2.2 Actuator interfaces

Broken out of the board through its GPIO pin header, are 8 serve motor controller port, as show in figure 5.1, which are able to generate PWM signals and interface with ECSs. These pins will be used to interface with the propulsion units.

5.3 Software and UML

5.3.1 The Robot Control Library (RCL)

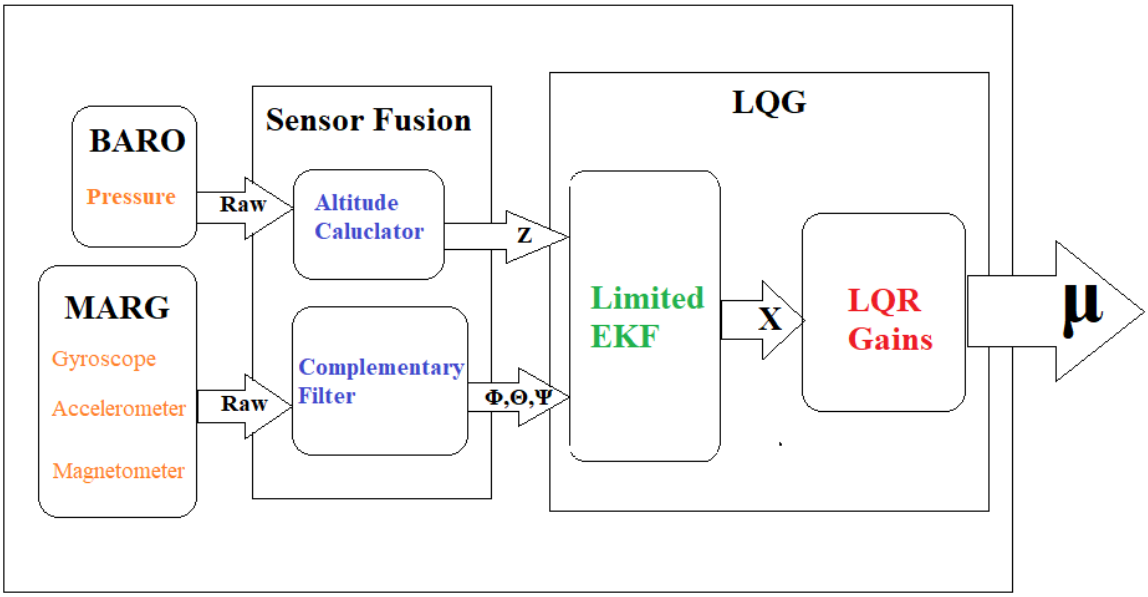


Figure 5.2: Embedded Component Structure

The BBB is an embedded Linux SBC running a customised Linux distribution which comes included with the RCL. The RCL is the primary software package for interfacing with and developing robotics software for the BBB and other BeagleBoard group SBCs. This package contains libraries written in the C programming language that provide an application programming interface (API) for accessing the sensors and communication interfaces for the Robot Control projects [28] The package also conveniently comes with examples and testing programs .

This package includes an extensive mathematics library for performing discrete-time feedback control such as including functions for:

- KF and EKF implementation.
- Digital Signal Processing and Filtering.
- Sensor Fusion Algorithms.

Besides these functions, the package also comes with various timing, multithreading and program flow functions, all of which is aimed at developing robotics software on embedded platforms.
Using the functions provided by the RCL, a basic implementation structure can be developed, as show in Figure 5.2, where:

1. Raw sensor values are read in by the SBC
2. These value are passed through a sensor fusion stage which calculates Z from raw pressure values and combines then converts the MARG value into $\phi, \theta\psi$ through a sensor fusion algorithm, the complementary filter [24], to make up the measured outputs Y.
3. Y is then sent to the Limited EKF developed in Chapter 4, the Limited EKF then returns the full state, X
4. X is then augmented with the integral term and multiplied with the LQR gains developed in chapter 4 to produce the input vector μ.

5.3.2 Universal Modelling Language (UML)

UML is an Industry standard set of tools which consists of a set of different diagram types, which is used to model software system [30].
The specific UML diagram implemented here is the UML class diagram. This diagram type is used to represent sections of software while showing the relationship between the different section. Class diagrams are typically utilised to represent system developed in an object-oriented manner with functionality separated into different classes, such as is possible with software developed in the C++ or the Java languages amongst many others [30].
However, class diagrams can be adapted to represent systems developed in a non-object oriented programming environments, such as the C language, if the functionality is sufficiently encapsulated and atomised. This is presented in Figure 5.3
The Presented UML diagram show key functionality grouped into blocks, as shown in Figure 5.2. Each block contaits Function extracted from the RCL Library which perform the blocks desired function [28].

- The Sensor.h block shows the functions for reading in data from the MARG and Barometer

- The SensorFusion.h block show functions from implementing a complementary and a built-in library script or calculating the altitude from pressure.

- The LimitedEKF.h shows functions for implementing an extended Kalman filter.

- The Receive.h block show functions which read in and parse through data sent by a radio transmitter using PPM

- The MotorControl.h block show functions for calibrating the propulsion unit ESCs and sending commands to them.

These blocks are then aggregated in the main.c block

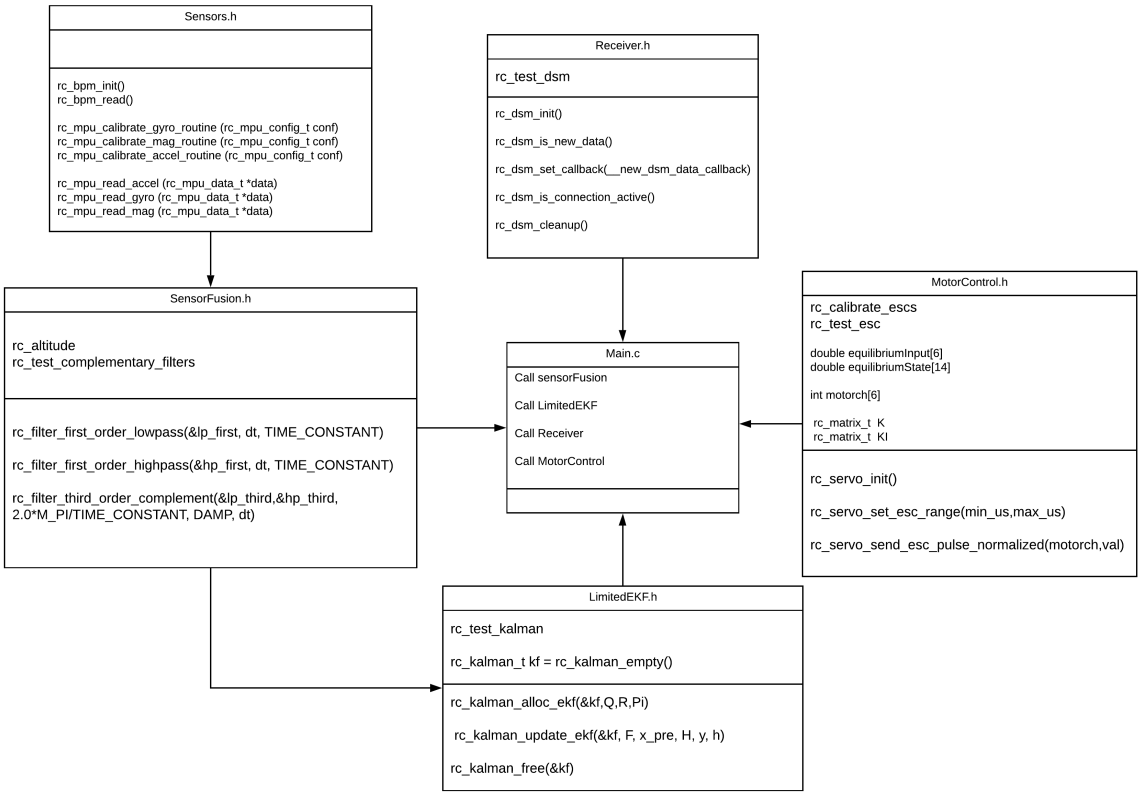


Figure 5.3: UML Class diagram of Functions

5.4 Summary

This Chapter:

- Discussed the utilisation of Single Board Computers
- Discussed key software and hardware elements available on the Beagle Bone Blue Single Board Computer
- Proposed and described an embedded flight control system implementation using the Unified Modelling Language's class diagram framework.

Control and Implementation Parameters		
Character	Definition	Value
T	Discrete Sample Time	0.01 s
DAC	Digital to Analog converter resolution	10 bits
ADC	Analog to Digital converter resolution	10 bits
V_{op}	SBC Operating Voltage	3.3 v

Table 5.1: Table of Control and Implementation Parameters

Chapter 6

Conclusion

In conclusion, this investigative project was abled to:

1. Develop a mathematical model representing the dynamics of the multirotor aircraft.
2. Convert the model into formats which that could be used for, analysis and control systems design using linear analysis and design techniques, and implementation unto digital embedded systems.
3. Investigate, develop and implement model-based LQG and LQ-MPC feedback control schemes in simulation.
4. Develop a proposal for implementing LQG based flight control software on an embedded microprocessor unit, conveyed using industry the standard UML software systems modelling tool.

Potential Future developments could include:

1. Implement the proposed flight control software onto an embedded microprocessor then developing and implementing the developed LQ-MPC control scheme.
2. Re-modelling the system using Quarternions to eliminate to possibility of singularity formation, thus opening up the full range of UAV motion
3. Investigate and develop non-linear control techniques capable of taking advantage of full motion range provide by a Quarternion based model
4. Implementing full control over 3-D euclidean space by incorporating feedback control authority over x and y. [5]

Bibliography

[1] M. K. Denis Kotarski, Petar Piljek, "Mathematical modelling of multirotor uav,"

[2] T. Luukkonen, "Modelling and control of quadcopter,"

[3] S. D. Prior, *Optimizing Small Multi-Rotor Unmanned Aircraft*.

[4] H. M. Mostafa Moussid, Adil Sayouti, "Dynamic modeling and control of a hexarotor using linear and nonlinear methods,"

[5] L. Whindborne, *Axes Systems, Orientation and Motion Notation*.

[6] *Fixed-Wing Dynamics and Control*.

[7] P. P. Denis Kotarski, Matija Krznar, "Experimental identification and characterization of multirotor uav propulsion,"

[8] D. K. Zoran Benic, Petar Piljek, "Mathematical modelling of unmanned aerial vehicles with four motors,"

[9] T. O. Oguz Kose, "Dynamic modeling and simulation of quadrotor for different flight conditions,"

[10] M. J. Roman Czyba, Grzegorz Szafransk, "Development of co-axial y6-rotor uav design, mathematical modeling, rapid prototyping and experimental validation,"

[11] "Appendix f: Vtol modeling.".

[12] A. Chevancova, "Mathematical modelling and parameter identification of quadrotor,"

[13] P. Bouffard, "On-board model predictive control of a quadrotor."

[14] *Lecture L26 - 3D Rigid Body Dynamics: The Inertia Tensor*.

[15] H. Packard, Poola, *Jacobian Linearizations, equilibrium points*.

[16] *A2212/13T Technical data*.

[17] H. Immaneni, "Mathematical modelling and position control of brushless dc (bldc) motor,"

[18] *Discrete-time Kalman filter*.

[19] *Robot Mapping: Extended Kalman Filter*.

[20] *Extended Kalman Filter Lecture re No*.

[21] *Understanding and Applying Kalman Filtering*.

[22] "Acs317: State-space design methods."

[23] V. M. S. Rafael Guardeno, Manuel J. Lopez, "Mimo pid controller tuning method for quadrotor based on lqr/lqg theory,"

[24] A. U. P.Tomas Larsson, "Scaling the speed of response using lqr design,"

[25] "Acs61161: Advanced control."

[26] J. A. Rossiter, *A First Course in Predictive Control*.

[27] J. A. Rossiter, *Model-Based Predictive Control: A Practical Approach*.

[28] "Robot control library documentation."

[29] T. S. Ng, *Flight Systems and Control*.

[30] T. Weilkiens, *Systems Engineering with SysML and UML*.

[31] H. K. Khalil, *Nonlinear Systems Third Edition*.

Appendix A: Gantt Chart

Figure 6.1: Gantt Table of Project Objectives and Dates

		Name	Duration	Start	Finish	Predecessors
1		**Development of V/STOL MultiRotor**	**213 days**	**21/10/19 08:00**	**21/05/20 08:00**	
2		S1 Week 4 Aims and Objectives	0 days	21/10/19 08:00	21/10/19 08:00	
3		S1 Week 6 Progress Review	0 days	04/11/19 09:00	04/11/19 09:00	
4		S1 Week 10 Second Reviewer	0 days	02/12/19 09:00	02/12/19 09:00	
5		S1 Week 11 Interim Report	0 days	09/12/19 09:00	09/12/19 09:00	
6		S1 Week 12 Second Reader Meeting	0 days	16/12/19 09:00	16/12/19 09:00	
7		S2 Week 1 Intereme Report Feedback	0 days	03/02/20 09:00	03/02/20 09:00	
8		S2 Week 6 Progress Review	0 days	16/03/20 09:00	16/03/20 09:00	
9		S2 Week 11 Final Report	0 days	18/05/20 08:00	18/05/20 08:00	
10		S2 Week 12 Presentation	0 days	21/05/20 08:00	21/05/20 08:00	
11		**Modelling**	**68 days**	**21/10/19 08:00**	**28/12/19 08:00**	
12		Obtain Parts	14 days	21/10/19 08:00	04/11/19 08:00	
13		Complete Multirotor Platform	14 days	04/11/19 08:00	18/11/19 08:00	12
14		Investigate Literature	14 days	21/10/19 08:00	04/11/19 08:00	
15		Develop Dynamical Model	40 days	04/11/19 08:00	14/12/19 08:00	14
16		Develop Model Simulation	14 days	14/12/19 08:00	28/12/19 08:00	15
17		**Control Systems Design**	**137.375 days**	**04/11/19 08:00**	**20/03/20 17:00**	
18		Investigate Literature	30 days	04/11/19 08:00	04/12/19 08:00	14
19		LQG Control Design	73.375 days	28/12/19 08:00	10/03/20 17:00	16;18
20		Obtain results in Simulation	10 days	10/03/20 17:00	20/03/20 17:00	19
21		**Flight Software Development**	**7 days**	**21/10/19 08:00**	**28/10/19 08:00**	
22		Obtain key Libraries	7 days	21/10/19 08:00	28/10/19 08:00	
23		**Advanced Objectives**	**208 days**	**21/10/19 08:00**	**16/05/20 08:00**	
24		Investigate Predictive control	180 days	21/10/19 08:00	18/04/20 08:00	
25		Implement in Simulation	28 days	18/04/20 08:00	16/05/20 08:00	16;24;20

ACS6420 VSTOL Project - page1

Figure 6.2: Gantt Chart Page 1

Figure 6.3: Gantt Chart Page 2

Figure 6.4: Gantt Chart Page 3

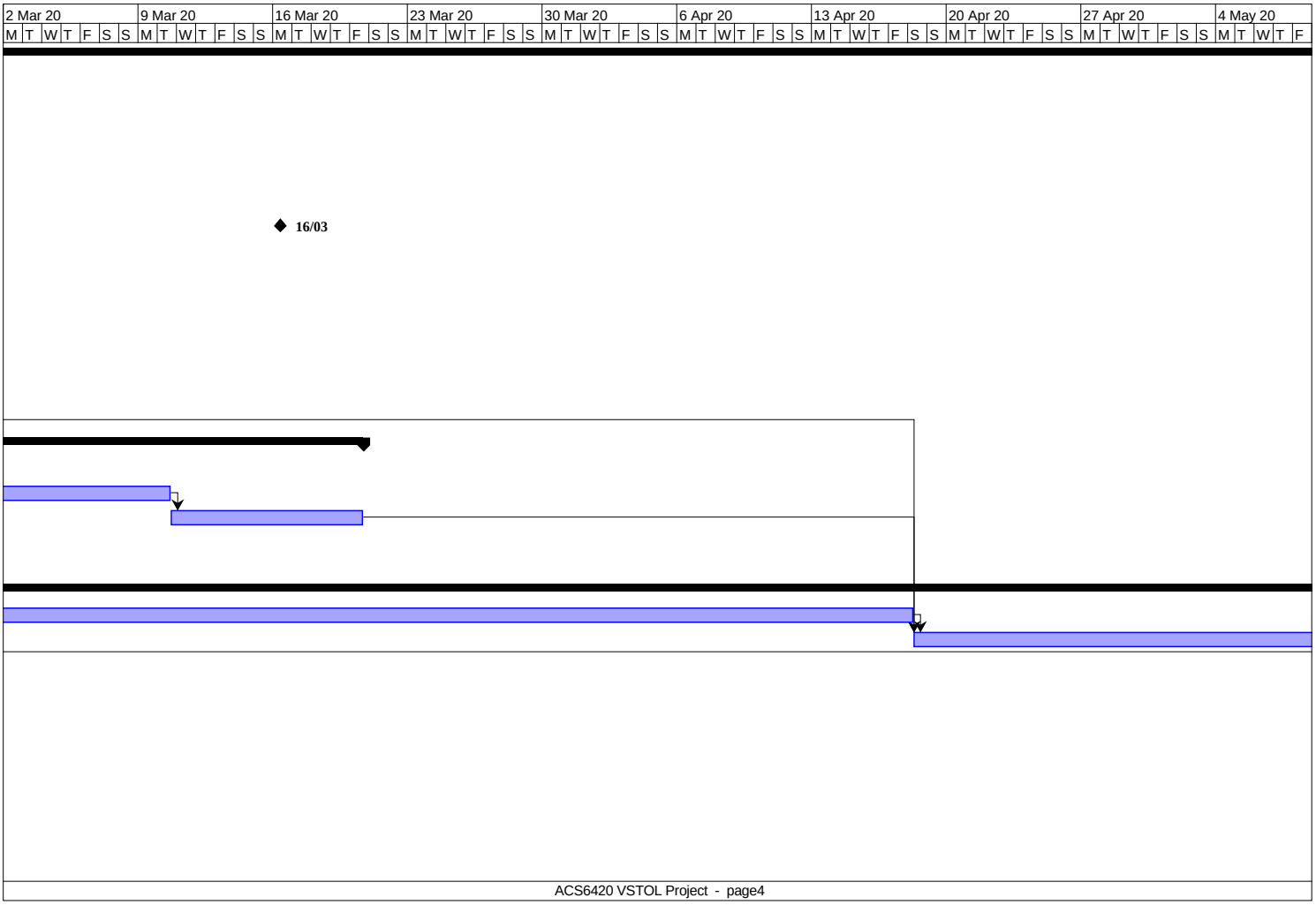
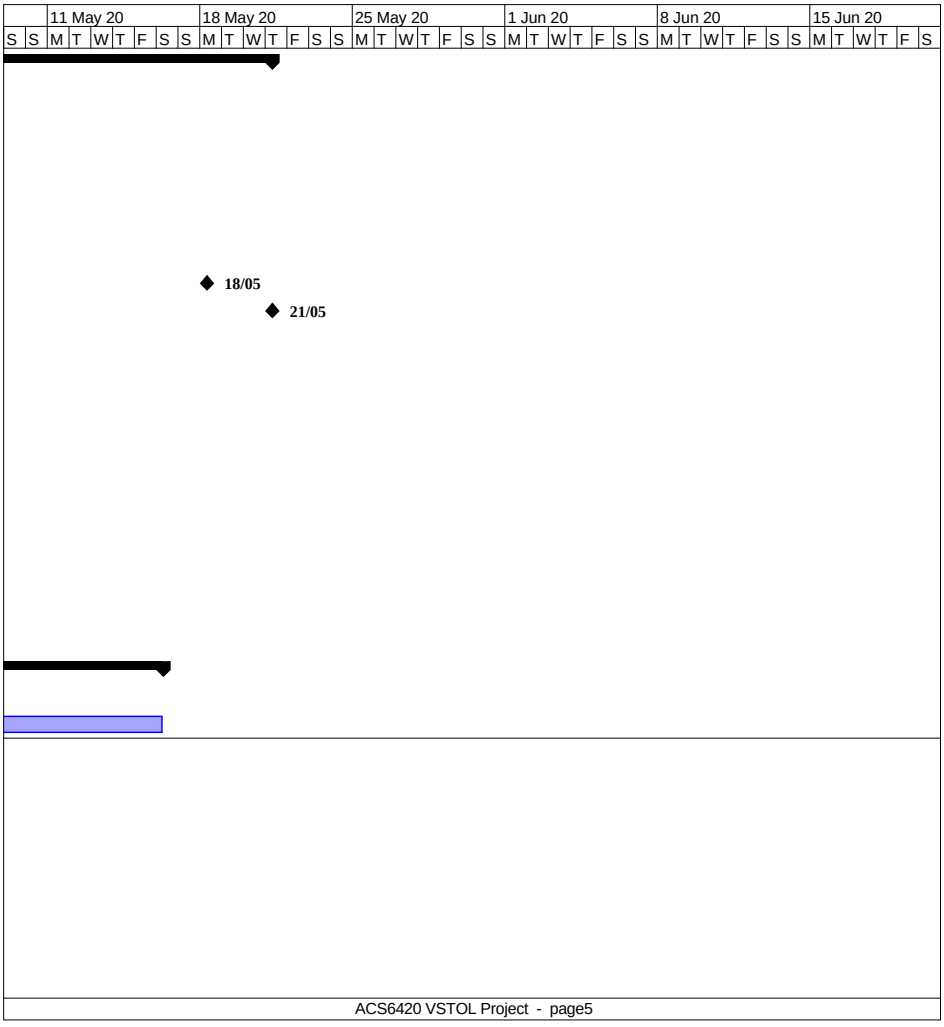


Figure 6.5: Gantt Chart Page 4



Appendix B: Control Design Parameters

$$Q_{LQG} = diag(200, 500, 1000, 10, 1000, 15, 1000, 10, 0, 0, 0, 0, 0, 0, 1, 35, 35, 0.1) \tag{6.1}$$

$$Q_{LQ-mpc} = diag(500, 0, 1000, 0, 1000, 0, 1000, 0, 0, 0, 0, 0, 0, 0) \tag{6.2}$$

$$R = diag(10^{-3}, 10^{-3}, 10^{-3}, 10^{-3}, 10^{-3}, 10^{-3}) \tag{6.3}$$

$$\mu_e = (176.1, 178.5, 177.2, 177.6, 202.2, 202.4) \tag{6.4}$$

$$\omega_e = \mu_e \times K_u \tag{6.5}$$