Addis Ababa University
Addis Ababa Institute of Technology
School of Electrical and Computer Engineering

MRAC design for a surveillance UAV for the detection of Water Hyacinth

A thesis submitted to School of Graduate Studies, Addis Ababa Institute of Technology, for the partial fulfilment of the requirement for the Degree of Master of Science in Electrical Engineering (Control Engineering)

By
Mihret Kochito

Advisor
Lebsework Negash (PhD)

November 18, 2021
Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

MRAC design for a surveillance UAV for the detection of Water Hyacinth

By: Mihret Kochito

APPROVED BY BOARD OF EXAMINERS

Name	**Signature**	**Date**
(Dean, School of Graduate Committee)		
(Advisor)		
(Internal Examiner)		
(External Examiner)		

Declaration

I, the undersigned, declare that this MSc thesis is my original work, has not been presented for fulfilment of any degree in this or any other University and all sources and materials used for the thesis are acknowledged.

Name **Signature**

_____ _____

Place:

Addis Ababa Institute of Technology, AAiT
Addis Ababa University, AAU
Addis Ababa,
Ethiopia.

Submitted in: November 18,2021

This thesis has been submitted for examination with my approval as a university advisor.

Advisor **Signature**

_____ _____

Acknowledgement

Throughout the work of this thesis, I have received a great deal of support and assistance. My special thanks goes to my advisor, Dr. Lebsework Negash, Assistant Professor of Electrical Engineering, whose expertise and guidance was invaluable throughout this work. The insightful feedbacks and dedicated support I have received have helped me to sharpen my thinking and brought my work to a higher level. I would also like to thank my family for their invaluable guidance throughout my studies. The continuous support and their sympathetic ear have provided me with the the tools that helped me to successfully complete my thesis. I would also like to thank my peers for the irreplaceable learning path we have had together with plenty of constructive discussions.

Abstract

Water hyacinth, locally named as 'Enboch', is an invasive aquatic weed posing a great threat to the worldwide aquatic ecosystem. Its existence has been reported to greatly diminish water surfaces' ecological value causing extensive nutrient reduction. An intuitive, but much feasible and inexpensive solution relies on the early detection of its presence followed by an action. This paper focuses on the design of a controller for a quadrotor able to perform area surveillance specifically suited for the detection of the hyacinth plant. The control design is done by taking the multivariate and non-linear nature of the problem into full consideration. The developed model reference adaptive controller (MRAC) comprising of both a standalone baseline controller and an adaptive augmentation is found to be able to stabilize the system in nominal scenarios and also restores nominal design performance in the presence of disturbances and parametric uncertainties. For the task of water hyacinth detection, the technique of transfer learning have been applied using the state-of-the-art VGG-16 model to perform feature extraction for a CNN architecture. The problem has been formulated as a multi-class classification problem considering three other aquatic plants identified as most probable on the habitats of water hyacinth. The trained model obtained an accuracy level of 93.34% through the training phase, 94.25% on a validation set, and 93% on a testing set.

Keywords: **Water hyacinth, Model reference adaptive controller, Adaptive augmentation, Minimum snap trajectory, Transfer learning, Fine tuning, Convolutional neural network**

Contents

List of Figures

List of Tables

List of Acronyms

UAV	Unmanned Aerial Vehicle
COG	Center of Gravity
DOF	Degree of Freedom
MRAC	Model Reference Adaptive Controller
LTV system	Linear Time Variant system
DMRAC	Direct Model Reference Adaptive Controller
InMRAC	Indirect Model Reference Adaptive Controller
PSO	Particle Swarm Optimization
ITAE	Integral Time Absolute Error
CNN	Convolutional Neural Networks
VGGNet	Visual Geometry Community Network
PID-gains	Proportion - Derivative - Integral gains

Chapter 1

Introduction

1.1 Background

Water Hyacinth, locally referred as 'Enboch', which is native to the Amazon basin, has currently become widespread throughout the world. This plant nowadays has been identified under seven species and its entire species comprise the genus Eichhornia, of which the Eichhornia crassipes species is common and widely distributed all over the world. The leaf stalks of young plants are swollen into spongy, bulbous structures and mature plants have elongated leaf stalks, which contributes to the plants free-floating capacity. Its flowers are blueish purple, large and extensively self-fertile. The plant has two reproductive schemes: through self-pollination and vegetative reproduction. It can reproduce through the production of horizontal stolon. On the other hand, its seeds are also produced in large numbers and are contained in capsules, each capsule containing up to 300 seeds. The seeds can remain viable for more than 28 years [1]. Once the flower blossoms the seeds capsule opens to release the seeds which will then sprout once conducive environment is achieved. Under perfect condition, colonies of water hyacinth can double their size every 5 to 15 days. The plant is considered as an invasive weed able to produce well over 650,000 offspring within eight months [2].

In Ethiopia, water hyacinth was observed and reported for the first time in 1965 in Lake Koka and River Awash [3]. Since 2011, water hyacinth has invaded Lake Tana, which is the largest lake in the highlands of Ethiopia. During the mentioned year, the Regional Environmental Bureau named water hyacinth as the most dangerous weed affecting Lake Tana. By then, about $20,000$ hectares of the north-eastern shore of the lake was already infested which made physical removal non-fruitful at the time. Researchers from Ethiopia found out that about one-third of the lake's shoreline, around 128km, was invaded by water hyacinth [4]. The invasion of the weed has been distributed to different water

bodies including Lake Tana, Lake Abaya, Lake Koka, Koka Dam . The presence of this aquatic weed has still been detected in colony after wide distribution was prevailed and water-bed of water hyacinth has been formed. The spread of the plant through floods and winds has made detection of the plant's early stage growth to not be guaranteed to be noticed by community dwellers. Due to the passage of a considerable time in control actions, solutions to eradicate infestations of water hyacinth have been done as an attempt to remove the already-made hyacinth colony. Biological control methods has been implemented against the infestation. Though being effective in eradicating water hyacinth, it seeks many years to achieve satisfactory control. Chemical control ways have also been tried to remove this invasive weed. But except for resulting chemical pollution of the water body,no considerable improvement have been achieved. Furthermore, spraying an entire heavy infestation can cause water hyacinth to sink and result in pollution from the rotting weed. Mechanical control is another attempt of physical automated removal that have been tried over the years. However, since water hyacinth seeds are extremely long-lived, new plants may spring up long after older plants get removed.

The highly invasive nature of the plant makes any other control action infeasible leaving early detection of its occurrence as the only solution to prevent infestation in new areas. It is helpful to destroy scattered early stage colony rather than delaying treatment until entire water bodies get choked. This helps to take advantage of mechanical control methods in removing the weed. It is also much easier and cheaper to prevent weed establishment when small weed infestations are treated quickly. Hence, early detection followed by rapid action of responsible authorities before the problem gets out of control is the appropriate solution towards the aim of removing the weed infestation.

Surveillance applications are greatly desirable in sectors where close control of areas is required. In recent years, unmanned aerial vehicles have been applied to various surveillance applications. Aerial image acquisition is one area where UAVs are mostly employed in prevailing much success. Their high maneuverability and low cost makes them suitable for the application of surveillance in the initiative of the early detection of water hyacinth through visual observations of susceptible areas.

1.2 Statement of Problem

Water hyacinth's extensive reproductive capability has enabled it cover large areas within just small-time span. It is already reported that one-third of the Lake Tana's shoreline is infested with the hyacinth plant. The rampant growth of water hyacinth and it's floating capacity can destroy native wetlands and waterways. Figure 1.1 presents a map of Lake Tana and it's major tributary rivers. The extreme seed viability and portability of the hyacinth plant endangers every tributaries of Lake Tana making them potentially susceptible for future large incursion. If not detected early, the extreme growth scheme presents a great threat even to the Great Ethiopian Renaissance Dam (GERD).

With the attempt to use UAV surveillance for the problem of the plant's recognition, it's important to note that that many of the quad-rotor's physical parameters are dependent on various physical operating conditions. For outdoor operations this effect is replicated due to temperature and many other unknown scenarios of the environment. In this cases, conventional feedback controllers fail to ensure stabilization. Hence, the problem of trajectory tracking and pose stabilization seeks a much robust control design.

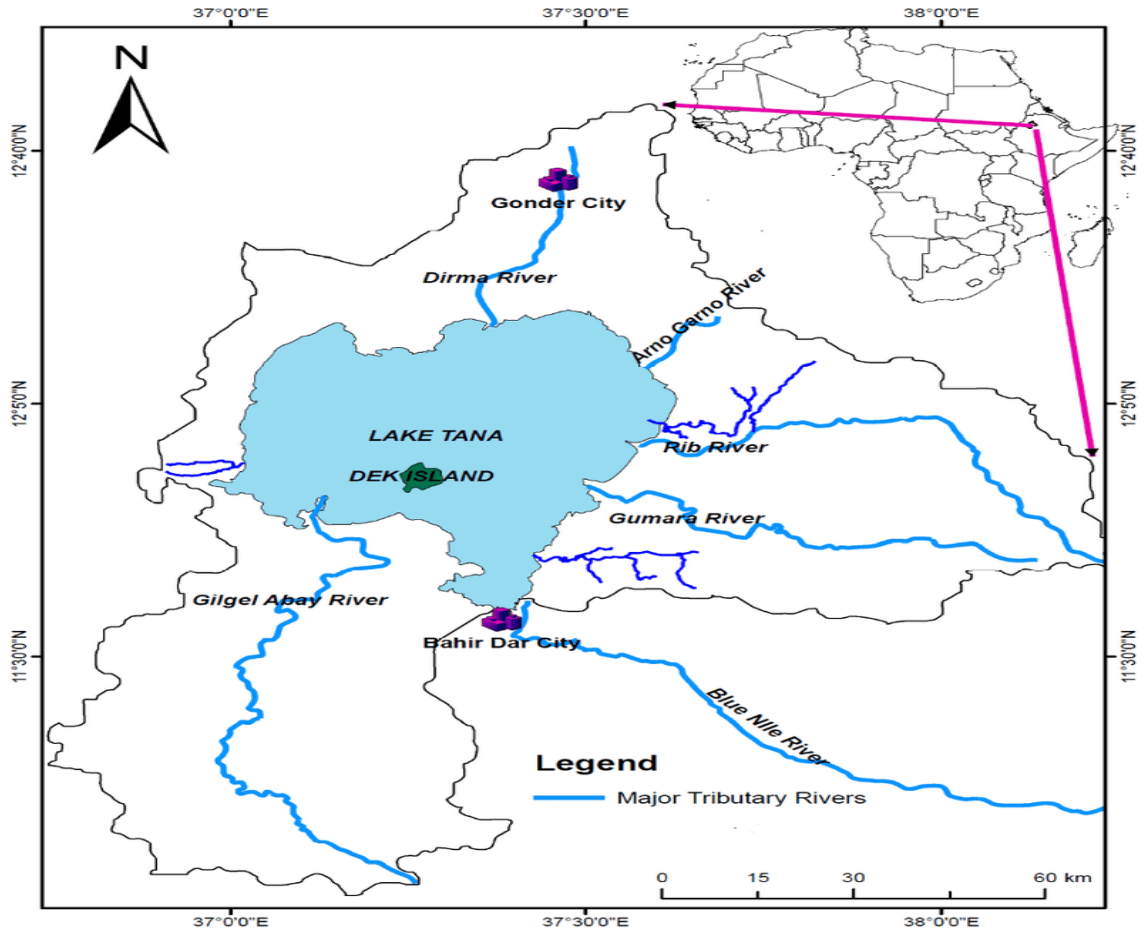


Figure 1.1: Lake Tana and its major tributaries

1.3 Objectives of the Study

1.3.1 General Objective

The main objective of this thesis is to design and simulate an adaptive control system for a quad-rotor suited for surveillance applications for use in the initiative of the detection of the water hyacinth aquatic weed.

1.3.2 Specific Objectives

The specific objectives of this thesis are:

- Modelling of a quad-rotor system by taking the multivariate and non-linear nature of the problem in to full consideration.
- Generation of a feasible trajectory that can be fed to the quad-rotor with out causing any abrupt change to the quad-rotor outputs and intermediary variables.
- Design of a baseline controller having the capability to stabilize both the rotational and translational dynamics under nominal conditions.
- Design of an adaptive controller which can be used on top of the baseline controller in order to maintain the stabilization of the quad-rotor when parametric uncertainties and disturbances are imposed in to the system.
- Simulation of the overall system using the MATLAB computing environment and the Simulink simulation platform.
- Developing, testing and analysing performance of an image recognition model suited for the detection of the water hyacinth aquatic weed.

1.4 Contribution of the Thesis

The application specific findings of this research will directly reflect to the benefits of societal groups whose daily lives greatly bound to the healthy retention of water bodies. It also has a long-term usage in the initiative of preserving water habitats and ecosystems. Implementation of this work will ensure the early detection of any occurrence of the water hyacinth aquatic weed so that physical removal would be a feasible task.

Moreover, the adaptive control design presented in this work can be replicated for quad-rotors suited to another tasks where operating conditions are not favourable. The control scheme provides position and attitude stabilization where the aerial vehicle's operation is expected to be threatened with disturbances and uncertainties. Moreover, due to the employed singularity free attitude representation, the control scheme can be used to provide attitude stabilization for quad-rotors needed to perform aggressive manuevers.

1.5 Scope

The work developed in this thesis focuses on design of a model based adaptive stabilizing controller for a quad-rotor vehicle which can be used for the detection and recognition of water hyacinth. The problem of trajectory generation and trajectory tracking has been approached using the presented schemes. However, the developed system does not present a fully autonomous design. A higher level path planner algorithm has to be implemented on the top of this work for the whole design to be autonomous.

The adaptive control scheme developed in this paper, angular rate and position error boundedness and convergence have been analysed analytically. However, the analysis of persistence of excitation in not done for the problem of parameter convergence for the considered uncertain parameters. Hence, a it is not guaranteed that estimated parameters converge to their true values using the adaptive scheme presented in this thesis.

Furthermore, even though there are currently about seven recognized species of the water hyacinth plant, in this thesis, focus is made only on the detection of the 'Eichhornia crassipes' species which is currently appearing as a problem in Ethiopia. Hence, the built deep learning recognizer model is able to identify only this species and the presented performance results are not guaranteed to be replicated for recognition problem of the other species of the aquatic weed.

1.6 Thesis Organization

The overall thesis is organized in to seven distinct chapters including this introductory chapter. Further details of the later chapters are herein presented.

Chapter 2 presents review of various technical and theoretical research papers which appear to be significant and act as ground stones for the work done in this paper.

Chapter 3 describes the modelling of the quad-rotor vehicle. The principles of operation of the quad-rotor is described along with ways of describing the quad-rotor position and attitude variables. Both the dynamical and kinematical equations are then formulated with consideration of all the significant physical effects on the quad-rotor.

Chapter 4 presents the overall control scheme design where the synthesis is decentralized in the baseline and adaptive controller. The details of attitude and position controller design is presented with their respective stability analysis and convergence properties.

Chapter 5 presents the preparation and generation of minimum snap trajectories that can be fed to the quad-rotor system as smooth and feasible reference signals . The analysis details and theoretical standpoints are presented in detail

Chapter 6 presents the details of the image recognition problem and how the problem is formulated along with that data preparation step. Furthermore the way of implementation are discussed.

Chapter 7 discusses the simulation scheme, trajectory tracking tests, the results obtained and their implications. The performance of the baseline controller and the adaptive augmentation is analysed in the presence of unfavourable scenarios simulated by imposed disturbances and parametric uncertainties. Numeric comparisons are then made to compare the performance of both. Furthermore the image recognition model is trained on the prepared dataset and analysis of the attained performance level are made afterwards.

Chapter 8 draws conclusion from the overall analysis performed and possible further research areas are implicated in detail.

Chapter 2

Literature Review

Over the years, several measures have been devised for the purpose of monitoring aquatic environment health. Technological measures have been taken with the aim of observing spatial changes that are thought to be indicators leading to undesired scenarios. Tackling this problem has been approached using various techniques. Proposed analytical solutions of earlier studies along with challenges presented are assessed as follows.

2.1 Proposed Solutions Against Hyacinth Infestation

Different studies succeeded in mapping out susceptible areas which provide conducive environment for the growth of water hyacinth. A study has been made presented in [3] to predict hotspot areas of water hyacinth over the surface of Lake Tana using the geographical information system GIS-based multi-criteria evaluation technique. Spatial analysis was used to prepare an overlay for the different parameters that explain the susceptibility of a given area. A final prediction map for water hyacinth infestation is then obtained outlining potential areas of Lake Tana for invasion by the water hyacinth with their respective obtained accuracy levels for consecutive three months. This study reported a much larger area to be potentially susceptible for future water hyacinth growth and expansion at the maximum historical lake level.

Remote sensing techniques have also been proposed by different studies to identify vegetation communities and weed availability. An application of remote sensing to monitor land in order to detect changes in terrestrial and aquatic landscapes has been employed in [5]. The data obtained was used to discover potentially predictive relationships between features and corresponding detected change in landscape. Another research presented Landsat remote sensing technique and tested it for identifying potential weed infestation

areas [6]. However, due to relatively low ground resolution compared to the patch size of weeds' invasion, this was found to serve only as a basis for locating areas for further refinement of weed distribution mapping which is expected to be done using a high-resolution remote sensing technique such as drone sensing or high resolution satellite missions.

Another study has been made in [7] to test two robust push-broom multi-spectral sensors named as Landsat-8 Operational Land Imager (OLI) and Sentinel-2 Multi Spectral Instrument (MSI) for the task of identifying, detecting, and mapping the spatial distribution and configuration of the water hyacinth weed in small river banks. However, the results of the study showed that water hyacinth in small reservoirs can be mapped with an overall accuracy of 68.44 % and 77.56% using Landsat-8 and Sentinel-2 data, respectively, which eventually represented low accuracy levels for the system to be implemented in large scale. A similar research in [8] proposed a method of estimating water hyacinth surface area based on a time series of a biophysical variable obtained from Sentinel-2 sensor images. After defining a reference period between two growing cycles, the method then uses fractional vegetation cover to estimate the water hyacinth water-bed formation in the study area. This method makes it possible to monitor water hyacinth development and estimates the total area it colonizes on water surfaces. Though some of the presented methods succeeded in mapping the area coverage of infestation, their solutions provided after-effect remedies which can be applied only after the hyacinth plant forms its water bed and control and eradication becomes harder.

2.2 UAVs and Aerial Surveillance

UAVs are nowadays being used in various sectors among which area surveillance is one of the main. Just as Satellites, UAVs have also been used to acquire imagery through remote sensing technique. UAVs as remote sensing platforms have used by research groups as presented in [9], with the aim of acquiring data at sufficiently low cost. With UAVs coming in many possible forms and sizes, they have adapted to remote sensing applications making them feasible to be applied in many sectors.

A comparison between UAVs and satellites in remote sensing has been made in another article [10] where UAVs were claimed to offer high versatility and flexibility, as compared to satellites. Their usage also presents the possibility of rapid operation without planned scheduling. Additionally, the possibility of flying at low altitudes and at various speeds, with the ability to acquire spatial and temporal high-resolution data were also reported as an important advantage against other conventional platforms that have been broadly used over the years.

Aquatic environment monitoring using a drone-based fluoro-sensor was performed in [11] for monitoring of Lasor inducted fluorescence from aquatic environments. Fixed-range remote sensing demonstration measurements were performed, and field recordings of natural river water fluorescence, oil- slicks as well as dye-marked natural water volumes were obtained from at drone flying at heights of about 10m. However, in order for the analysis to be done, a fluoro-sensor weighing about 1.5 kg should be carried by a drone for it to be capable of monitoring the aquatic area. This poses extra control cost in controlling the flight operation of the UAV.

Fixed networks of cameras or satellite imagery present more expensive solutions than UAV systems. Not only their low cost price, but also their maneuverability has made UAV systems to be widely applied to different sectors including disaster observation, aerial image acquisition and military surveillance.

Surveillance in UAV systems is employed by using images taken by an fixed on board cameras where the whole system is programmed to cover a specified area. The feasibility of applying UAVs for the problem of area surveillance was assessed in [12] where the goal was monitor and retrieve information about a certain target area. It was proposed that aerial surveillance can best be approached by using a group of UAVs consisting of several elements specified for the task of coverage of a target area with a pre-designed path. Thus, the problem of UAV surveillance is most convincingly reformed as an area coverage problem. In [13] mentioned that the sweep solution can provide a good coverage with with UAVs for the task surveillance of large areas. All in all, it was evidently shown that UAVs present a low cost, most reliable and flexible solution for the task of area surveillance.

Among the various kinds of UAVs, the quad-rotors are the ones which have got considerable attention over the years for area surveillance tasks. Its reduced mechanical complexity, payload augmentation, gyroscopic effect reduction are some of the advantages of quad-rotors over the other UAVs [14]. The quad-rotor is constructed with fixed pitch rotors and uses the variation in motor speed for navigation control enabling it to have a very simple design with basically no mechanical control linkages.

However, due of the quad-rotor's highly non-linear coupled dynamics and its under-actuated design configuration, the task of controlling a quad-rotor is a complex task making its flight control a challenging problem. Its modelling is also posed with several physical parameters which are uncertain during flight operations [15].

2.3 The Control Problem of UAVs

2.3.1 Conventional Control Schemes

Different control schemes [16], [17], [18] have relied on linearised models with the aim of simplifying the highly coupled nature of the quad-rotor vehicle. However, these schemes are mostly employed for sole altitude stabilization problems at the hovering position only. The application of these kinds of controllers for trajectory tracking problem is nearly impossible. The modelling schemes used for these linear controller suffer from inherent in-stabilization problem since they do not do not capture the entire attitudinal dynamical behaviour of the vehicle.

An attempt of avoiding this problem has been approached by utilizing non-linear controllers which enable trajectory tracking beyond the valid linearisation range for linear controllers. A methodology was presented in [19] which deals with the control of under-actuated systems treating various system constraints. A back-stepping controller was proposed in [20]. The downsides of these presented controller is that their attitudinal representations do rely on the Euler angles which suffer from inherent singularity problem named as gimbals [21]. Though attitude representation using angles is a very natural way of describing orientation, this representation has inherent singularities. Moreover, the presence of many non-linearities often represent complications in the design of control strategies. The of a quaternions instead of Euler angles to describe the rotational dynamics for quad-rotors provides a solution to the downsides of the euler angle representation.

2.3.2 Quaternion Based Controllers

A remarkable amount of researches are done with quaternion based attitude stabilization. An adaptive quaternion based non-linear controller architecture have also been proposed in [22] for both attitude and position control making use of the quaternion state feedback. Though the formulation utilized the singularity free attitude representations, designing the control scheme using quaternion state feedback has made the system to suffer form the so called unwinding phenomenon caused by inconsistent quaternion feedbacks.

Tackling the problem of the unwinding phenomenon, control schemes in [23], [24] makes use of the pure quaternion in order to synthesize a controller by utilizing Lyapunov Functions. An energy based quaternion controller have also been developed in [25] which proposed novel energy-based control laws described with unit quaternions providing stabilization of the quad-rotor in all its states.

2.3.3 Robust and Adaptive Controllers

The field of UAV surveillance poses an additional control threat since its operation seeks autonomous outdoor operation. During flight, the quad-rotor vehicle is always subjected to various disturbances and system uncertainties. This effects come in the the forms of parametric perturbations, noise and wind gusts. This makes the flight control problem challenging in order to escort the quad-rotor states from a large deviation and keep the system in the intended performance levels. Control schemes that have the capability of dealing with the posed uncertainties have to be implemented so as to ensure position and attitude stabilization during this scenarios. Numerous research studies have examined the attitude and position control design and implementation to retain satisfactory performance operation.

Many literatures have relied on adaptive and robust control schemes in order to meet error convergence while operating in adverse environments. An adaptive sliding mode controller in [26] and an adaptive back-stepping controller in have been proposed in [27]. however, both of the control schemes utilize a linearised attitude kinematics. The non-linear system in both cases is forced to follow a linear reference model and thus the performance is limited to the achievable by linear dynamics.

In contrast to the schemes developed using linear refernce models, a fully adaptive non-linear controller have been implemented in [28] which donot pose linearity restrictions on the refernce model. A non-linear composite adaptive control algorithm is developed in another research [29] for the problem of stabilization of a quad-rotor in scenarios where plant parameters such as mass, system inertia, thrust and drag factors are considered as fully unknown. Lyapunov based adaptive controllers are also employed in [30], [31]. However both are formulated as a fully adaptive schemes.

In [32] an adaptive augmentation of a non-linear baseline controller was proposed. The augmentation is used so as to restore a nominal, non-linear closed loop behavior. Another research [33], researched for ways of adaptive compensation for position tracking error of the quad-rotor. The developed adaptive compensation scheme is found to successfully reduce position tracking errors in the presence of constant disturbance. Unlike the schemes developed as fully adaptive, the approach of adaptive augmentation leads in general to more robust and highly performant controller. A recent research in [34] proposed a novel control scheme by constructing an indirect attitude adaptive augmentation. The control design offered a singularity free tracking of the attitude loop by enabling large aggressive manuevers. However, the control scheme developed is just to the attitude dynamics hence needed a higher level position controller to be fully performant.

From all the knowledge resulting from state-of-art implementations, it can be

concluded that UAVs present as the best candidate for area surveillance tasks. Though their design and mechanical simplicity come with a cost in their flight control, different control schemes have been developed over the years to tackle this problem. Adaptive and robust control schemes do provide with ways of stabilization of UAVs for operation in outdoor harsh environments. Moreover, control schemes relying on the the quaternion framework offer a singularity free attitude stabilization and do provide better computational efficiency.

In this thesis, a singularity free non-linear adaptive controller is going to applied for the problem of surveillance for water bodies for detection of water hyacinth aquatic weed. The control design developed in this paper can be taken as an extension of the basic work developed in [34] with the aim of offering both position and attitude adaptation to ensure error convergence in the presence of disturbances and parametric uncertainties.

Chapter 3

Modelling of the Quad-rotor

3.1 Principle of Operation

The quad-rotor's mechanical design comprises four identical and equally spaced rotors with propellers attached to shaft of the rotors in a direct coupling mechanism. Moreover, the propellers are arranged in counter-rotating pairs, with each opposite pair of rotor running in the same direction.

The motor-propeller system is responsible for thrust and torque generation. The thrust force produced by the propellers always act in the direction of the motors' axes directed from the motors to the propellers as seen in Figure 3.1. Another force which counteracts with the propulsion force is the gravitational force always pointing towards the center of gravity of the earth. It is to be noted that the generated thrust forces happen at the rotating tips of the quad-rotor. This scenario, in turn, produces turning moments named as the roll, pitch and yaw torques which happen about the center of gravity of the vehicle. Each of the rotating propellers, on the other hand, produce reaction torques about the axis of each rotating propeller directed opposite to the generated torques. A proper rearrangement and allocation of the propeller force and generated turning moments is mandatory to enable the quad-rotor's navigation.

Using the standard model-based design of systems, the analysis of the quad-rotor control system starts by adopting a higher fidelity non-linear model. The proposed modelling approach is shown with in-depth description of parameters used for modelling. Verification of the model using selected test cases is then performed so as to test the matching of the behaviour of the developed model and the underlying reality.

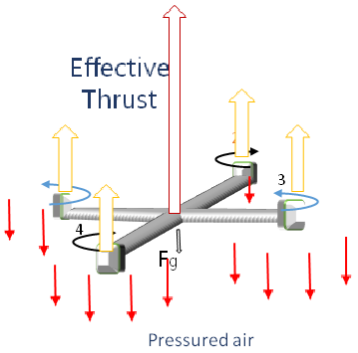


Figure 3.1: Thrust Generation

3.1.1 Configuration Schemes of the Quad-rotor

Quad-rotors do come with varying construction schemes offering a large variability in function. The quad-rotor's overall motion is primarily due to the action of the propellers. Depending on the vehicle's blade-orientation relative to its body-coordinate, the quad-rotor can be broadly classified into the plus-configuration, "+" and the cross-configuration, " × ". The illustration of each is shown in Figure 3.2.

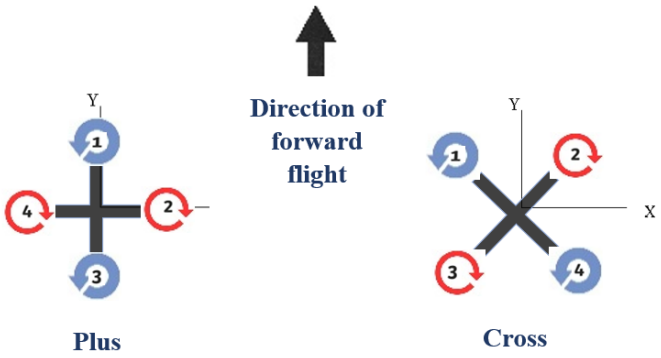


Figure 3.2: Configuration Schemes

The Plus configuration uses a pair of blades which spin in the same direction, with the pair of blades placed on X and Y coordinates of the body frame. This configuration provides easier control of the vehicle since both lateral and axial movements (either in the X or Y direction) require a controller to balance only the speed of two blades responsible for that desired direction.

In the cross configuration, all the four rotors are simultaneously engaged in motion generation. As a result, this produces high maneuverability and greater momentum [35]. Moreover, this configuration is considered to be more stable as compared to the plus configuration. Even though both of the configurations are widely practices amongst researchers and designers, the cross-configuration is chosen in this paper's design due to its stability to changing speed of each blade by a small amount during the creation of

lateral and axial motions.

3.1.2 Reference Frames Alignment

When talking about motion, a reference to which that motion is described should be allocated first. Only with understanding of frames can the degree of motion can be described fully. Quad-rotors are usually defined in spatial orientation using a 'two-reference-frames' system defined as inertial reference frame and the body reference frame. A sketch of the quad-rotor is shown in Figure 3.3, with the body reference frame attached to it, having the inertial frame as a reference to describe the quad-rotor's motion.

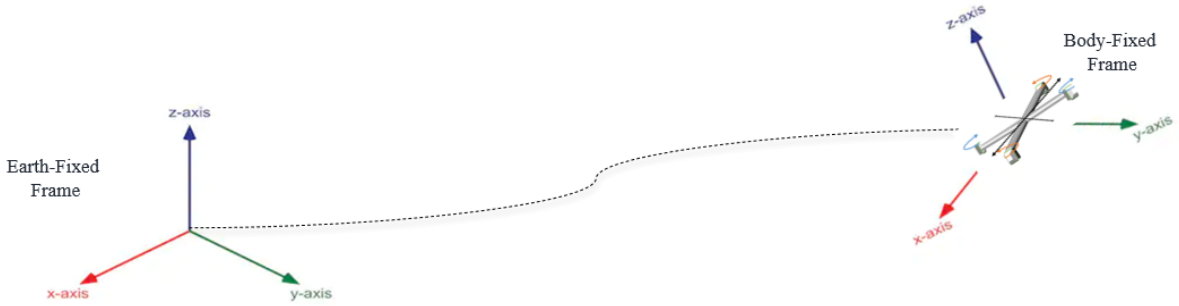


Figure 3.3: The Inertial and Body Coordinates of the Quad-rotor

- **The body reference frame:** is sometimes known as mobile reference frame since it is always attached to a moving body. The frame originates from the center of mass of the vehicle. The body reference frame vectors describing the vehicle's linear and angular positions are generally represented as translational velocities: $[\,u\ v\ w\,]$ and rotational velocities $[\,p\ q\ r\,]$.
- **The Inertial Reference frame** is earth centred. It's Z direction points toward the celestial pole with its X axis pointing towards the prime meridian and its Y axis chosen to be orthogonal to the other two. Within this frame, the vector $[\,x\ y\ z\,]$ describes the reference frame's linear positioning whereas attitude in this frame is decoded by the three euler angles $[\,\phi\ \theta\ \psi\,]$.

3.1.3 6-DOF Motion Generation

The six degrees of freedom of a quad-rotor and the idea behind motion generation is illustrated in Figure 3.4 with description presented afterwards.

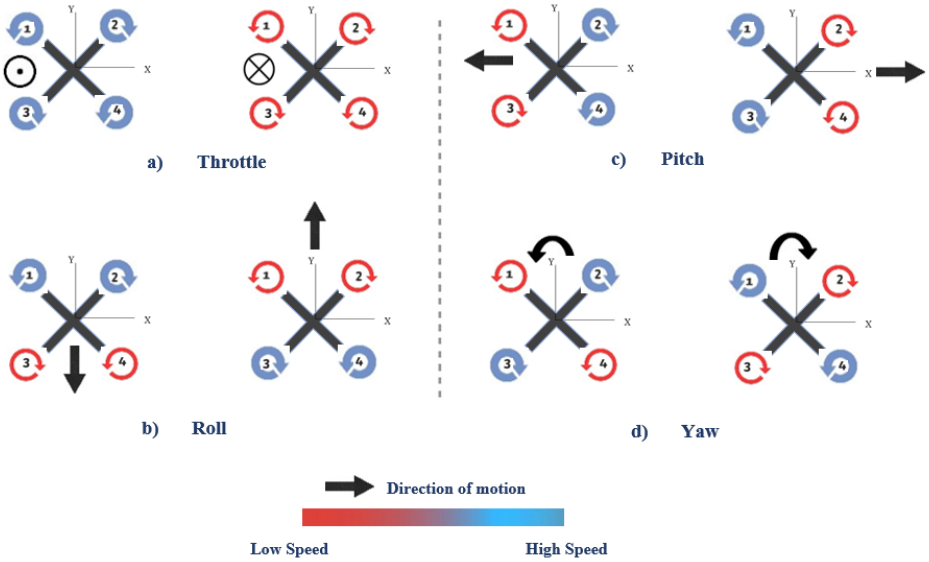


Figure 3.4: Basic Flight Movements of a Quad-rotor

Throttle : Vertical take-off and landing is possible by throttling which is achieved by augmenting (diminishing) all the values of propeller speeds simultaneously. This facilitates the generation of a vertical force with respect to the body fixed frame. If the vehicle is in its horizontal (non-tilted) position, then this command causes only vertical forces to be generated. Otherwise, the throttle command generates both vertical and horizontal motions as seen in the inertial frame. Figure 3.4 (a) shows an illustration of the quad-rotor driven by a throttle command.

Roll: The roll torque can be generated by decreasing (increasing) the speed of the two propellers found on left and by increasing (decreasing) the speed of the rest found on the right. This creates a torque to be generated about the X-axis of the vehicle. During a roll command the overall effective thrust has to be maintained so that only a roll acceleration is generated making the vehicle navigate in the Y-axis of the body reference frame as shown in Figure 3.4 (b).

Pitch: The generation of pitching movement is alike to that of the roll except for the axis of the moment generation . Here movement is achieved by decreasing (increasing) the speed of the two front propellers and by increasing (decreasing) the speed of the rear propellers. Hence torque will be generated across the Y-axis of the body reference frame causing the quad-rotor to translate in the X direction as shown in Figure 3.4 (c). Just as the roll command, only pitch acceleration is generated by maintaining the effective upward thrust to be constant.

Yaw: The yaw movement is achieved by increasing (decreasing) each of the opposite pairs of propellers rotating counter-clockwise (clockwise) directions. Hence, due to the unbalance of the overall torque, the vehicle turns itself around the Z-axis of the body reference frame as shown in Figure 3.4 (d).

3.1.4 Control Allocation

The quad-rotor is equipped with 6 degrees of freedom. However the vehicle has only four rotors. Hence control wise, it is not possible to reach to a desired set-point with explicit control force associated to each degree of freedoms. However, intermediary controllable variables associated to the quad-rotor DOF can be chosen which can then be decoupled to make the control design easier.

The four quad rotor control inputs are allied to the four basic movements allowing the vehicle to reach a certain height and pose. For the chosen cross quad-rotor configuration, the control allocation matrix can then be found as follows :

$$\begin{bmatrix} U \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} F_1 + F_2 + F_3 + F_4 \\ M_1 + M_2 - M_3 - M_4 \\ M_1 - M_2 - M_3 + M_4 \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix} \tag{3.1}$$

where F_i and M_i represent generated thrust and moment respectively. And,

U - Throttle command
τ_x - Roll command
τ_y - Pitch command
τ_z - Yaw command

The generated thrust and moment of the quad-rotor are dependent on many experimental parameters where their formulation can be described as in Eqn: 3.2.

$$F_i = k_f {w_i}^2 \qquad M_i = \frac{l}{\sqrt{2}} k_m {w_i}^2 \tag{3.2}$$

where $i = 1, 2, 3, 4$, k_f and k_m are thrust and moment coefficients respectively whose values are found empirically. Whereas, l is a physical parameter representing the arm length of the quad-rotor with $\frac{l}{\sqrt{2}}$ representing the perpendicular distance from the body-axis of the quad-rotor to the point where the propulsion forces occur.

3.2 Modelling Approach

The first step towards designing control schemes for any dynamical systems is the development of a mathematical model. The mathematical model is expected to mimic the behaviour and the operation of a certain system in a real world scenario. Modelling approaches of a quad-rotor do rely on the physics of the system. The analysis is done by partitioning of the whole system into smaller subsystems so as to ensure easier analysis and design [36]. In this section, a classical modelling approach is employed to arrive at the equations of motion that describe both the dynamics and the kinematics of the vehicle. Supporting concepts like attitude representation are also discussed in detail.

3.2.1 General Model Assumptions

Different assumptions are made across literatures for simplification of the quad-rotor modelling. These assumptions are made while still establishing a fairly accurate model that will not miss any significant non-linear behaviours of the vehicle. Universal modelling assumptions that are taken from different literatures [37], [38] and adopted in these paper are herein presented as follows:

- The Quad-rotor has symmetrical structure and the inertia matrix about this symmetry is diagonal.
- The propellers of the quad-rotor are assumed to be rigid.
- Centre of Gravity (CoG) of the vehicle coincide with the origin of the body reference frame.
- The physical structure of the quad-rotor is considered a rigid frame constructed with four rotors.
- The vehicle's mass is assumed to be time-invariant during motion.
- Aerodynamically effects such as blade-flapping and non-zero free stream velocity are ignored for the analysis to follow.

3.2.2 Attitude Representation

Relative translational position between two points in space can be described by three dimensional position vector obtained in relative to each other. Another problem toward modelling a navigating vehicle is the representation of relative orientation between two poses. There are different approaches with which pose and attitude of a certain rotating body can be described.

Many papers across the years have relied on the classical Euler angles to represent the attitude of the quad-rotor. This approach of attitude representation is assumed to be intuitive making it easier for implementation. Using Euler angles, any rotation can be casted in terms of the three successive rotations. The procedure follows by factoring a single rotation into the three successive rotations around the principal orthogonal axes (i, j and k). The rotation about each of the respective axes is then represented as the triple (ϕ, θ, ψ). An illustration of two successive rotations using the euler angles is shown in Figure 3.5.

However, a more complex design requirement cannot be done by an attitude description using the euler angles [21]. In flight control, the use of euler angles results in a singularity that results in a problem named as 'gimbals lock' which is illustrated in Figure 3.6. This singularity, clearly, is not a singularity in the rotation group since we can rotate a free body in space physically without bumping into any singularities. Rather, Euler's inherent singularity results from the loss of a degree of freedom in the representation itself called a coordinate singularity which happens when the pitch angle approaches $\frac{\pi}{2}$ [39]. This situation has forced any control design using the euler representation to seek pitch angle restriction and strict avoidance to never reach this unexplainable singularity point. The euler angle representation also appears to be computationally intensive due to the presence of sine and cosine terms, marking another downside of using them for attitude decoding [40].

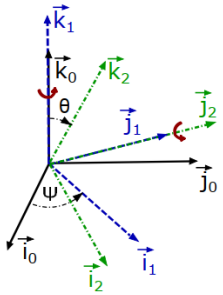


Figure 3.5: Two Consecutive Elementary Rotations: about k_o $--$ $>$ about j_1

Another way to come across the downsides imposed by the euler angle representation is using quaternions. This representation can be used to represent attitude of a certain rotating body using a singularity free description whose details are presented in the following section.

3.2.2.1 The Quaternion

Quaternions were first proposed by Hamilton as an extension of the two dimensional complex numbers in the three dimensional space [41]. In contrary to the euler angle

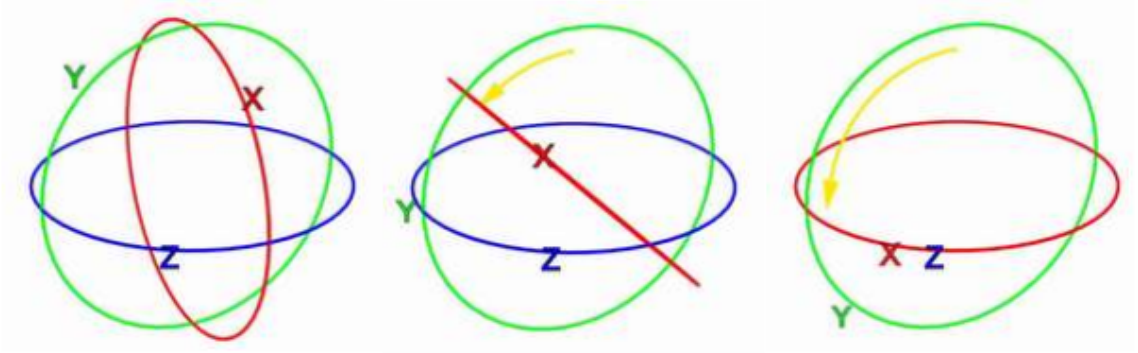


Figure 3.6: Occurrence of gimbal lock when the Y axis rotates 90^0 and the X and Z become aligned m aking the rotation scheme lose one degree of freedom

representation, quaternion attitude representation uses a single axis and an angle of rotation to describe attitude in the three dimensional space.

Though quaternions lack clear physical intuition which Euler angles have, their actions in avoiding singularity problems is a quite desirable property in some applications. They also offer a computationally efficient description of three-dimensional rotations.

A quaternion is seen as a hyper complex number obtained by extending the idea of complex numbers to the three dimensional space. A single quaternion can thus be represented as follows:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{3.3}$$

More compactly:$q = [q_0 \quad q_1 \quad q_2 \quad q_3]^T = (q_w, \overrightarrow{q_v})^T$ whose algebra is given by the following Hamilton's expression [41].

$$i^2 = j^2 = k^2 = ijk = -1$$

Further more, the following definition holds true in the quaternion space.

$$ij = k, \quad jk = i, \quad ki = j,$$
$$ji = -k, \quad kj = -i, \quad ik = -j$$

3.2.2.2 Quaternion Algebra

Algebraic operations of quaternions in the four dimensional space are listed as follows using p and q defined as $q = q_0 + q_1 i + q_2 j + q_3 k \quad and \quad p = p_0 + p_1 i + p_2 j + p_3 k$.

Property	Description
Quaternion Multiplication:	$$q \otimes p = \begin{bmatrix} q_w p_w - q_v p_v \\ q_v \times p_v + q_w p_v + p_w q_v \end{bmatrix} \quad (3.4)$$
Norm of a quaternion:	$$\|\|q\|\| = \sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2} \quad (3.5)$$
Complex conjugate:	$$q_* = (q_0, -q_1, -q_2, -q_3) \quad (3.6)$$
Inverse of a quaternion:	$$q^{-1} = \frac{q^T}{\|\|q\|\|} \quad (3.7)$$, if $\|\|q\|\| = 1$, then, $q^{-1} = q^T$
Multiplicative identity property:	$$q \otimes q^{-1} = q^{-1} \otimes q = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.8)$$
Pre multiplication by a quaternion:	$$q \otimes p = Q(q)p = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \quad (3.9)$$
Post multiplication by a quaternion:	$$q \otimes p = Q(p)q = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (3.10)$$

3.2.2.3 Quaternions as Rotary Operators in $3D-$ Space

A given quaternion corresponds to an orientation in 3D space. Changing a pair among the four values of a quaternion results in a body in 3D space getting reoriented. The idea of rotation using quaternions relies on the fact that a point or a coordinate frame can be

brought from an arbitrary initial orientation to an arbitrary final orientation by a single rigid rotation about a certain unit-length axis $u = u_x + u_y + u_z$ rotated by an angle of rotation θ. A vector v in three dimensional (3D) space is defined as a pure quaternion as follows:

$$v = 0 + v_1 i + v_2 j + v_3 k, \quad \text{with } v_0 = 0 \tag{3.11}$$

Then the vector's rotated version v' can be found by the following quaternion multiplication which comprises of the pre and post multiplication of quaternions whose illustration is shown in Figure 3.7.

$$\begin{aligned} v' &= qvq^{-1} = q^{-1}vq \\ v' &= qvq^{T} = q^{T}vq, \end{aligned} \tag{3.12}$$

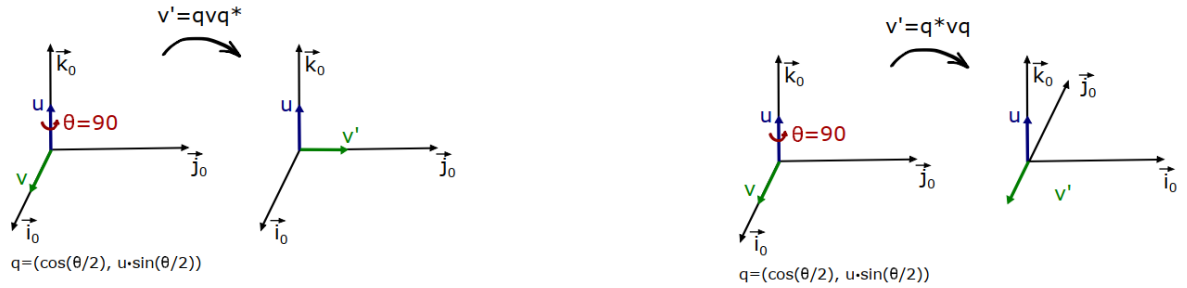


Figure 3.7: Quaternion as a Rotary Operator

Further more, using the matrix expansions of quaternion multiplication outlined in Eqn: 3.9 and Eqn: 3.10, the quaternion rotation operation between the two reference frames defined in subsection 3.1.2 can be put as a single rotation matrix as follows. The details of the rotation matrix representation can be seen in detail in Appendix B.1.

$$R_{\mathrm{E}}^{\mathrm{B}} = \begin{bmatrix} {q_0}^2 + {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix} \tag{3.13}$$

$$R_{\mathrm{B}}^{\mathrm{E}} = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix} \tag{3.14}$$

3.2.2.4 Quaternion's Double Coverage Scenario

Recalling the operation for rotating a vector as described in Eqn: 3.12, it can be seen that the operation involves both pre and post multiplication of the vector by a quaternion q so as to result in a single rotation. This is because multiplication of the vector with

a quaternion q corresponds to rotating the vector only by half of the angle and alters the vector in 3D-space in a certain way morphologically. Hence a further multiplication by the inverse quaternion q^{-1} (*for unit quaternions,* $q^{-1} = q^T$) is needed to continue rotating the vector up to the desired angle by correcting its altered morphology.

The double multiplication with a quaternion results in a scenario called double coverage for rotations in 3D-space as illustrated in Figure 3.8. The double coverage scenario results in the presence of two separate points in the 4D-space corresponding to a single orientation in the 3D-space. Noting that only unit quaternions can be used as

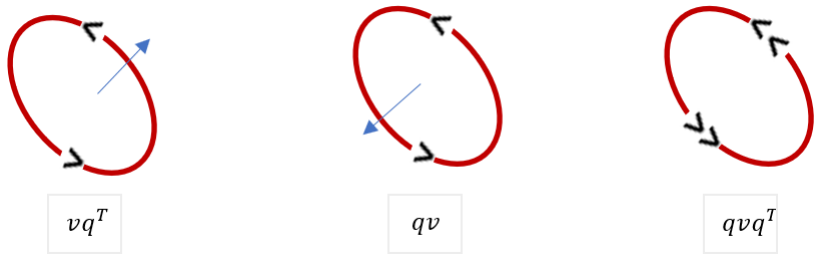


Figure 3.8: Double Coverage of SO(3)

rotation descriptors [42], the generalized Euler - Rodrigues identity shown in Eqn: 3.15 can be applied on them in the same way it is applied for two dimensional complex numbers. A more compact axis angle representation of the quaternion q can thus be obtained as follows:

$$e^{\theta u} = (\cos\theta + u\sin\theta) \tag{3.15}$$

$$q = e^{u\frac{\theta}{2}} = \cos\frac{\theta}{2} + u(\sin\frac{\theta}{2}) \tag{3.16}$$

where $u = u_x + u_y + u_z$ is an arbitrary, unit-length axis of rotation in three dimensional space and θ is the angle of rotation about the specified axis of rotation, u.

Quaternion for Shorter Rotation

The double scenario in-turn produces two possible options of reaching a certain attitudinal orientation in the 3D-space, one of which represents the longest rotation and the other representing the shortest rotation. Analysing the two analogous quaternions q and $-q$, it can be seen that the short and long way of rotation can be found depending on the sign of the scaler element of the quaternion q_w [34].

Hence the quaternion representing the short rotation with an angle less or equal to π, is shown as follows encoded by q^+:

$$q^+ = \begin{bmatrix} q & , q_o \geq 0, \\ -q & , q_o < 0, \end{bmatrix} \tag{3.17}$$

3.2.3 Newton Euler Formulation

The classical mechanics approaches, Newtonian and Lagrangian mechanics provide ways of obtaining mathematical formulations that would best describe the behaviour of dynamical systems. In the following sections, the general mathematical model describing the dynamics and kinematics of an aircraft navigating in a three-dimensional space is obtained by using the Newton-Euler approach as described in [43], [44], [45], [46]. This is done by representing the aircraft as a solid and rigid body subject to underlying forces and moments.

3.2.3.1 Dynamic Model of the Quad-rotor

Forces on a Flying Quad-rotor

Propulsion Force	Force of gravitation
The effective force generated by the propellers described in the body reference frame is as follows: $F^B = \begin{bmatrix} 0 \\ 0 \\ F_{thrust} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ (F_1 + F_2 + F_3 + F_4) \end{bmatrix}$ where F_i represents the thrust forces generated by the four propellers with F_thrust representing their whole summation. Making use the rotation matrix presented in Eqn: 3.14 , the propulsion force is given as follows in the inertial frame: $F^E = R_{\mathrm{B}}^{\mathrm{E}} F^B = \begin{bmatrix} 2(q_1 q_3 + q_0 q_2) F_{thrust} \\ 2(q_2 q_3 - q_0 q_1) F_{thrust} \\ ({q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2) F_{thrust} \end{bmatrix}$ $F^E = q \otimes F_{thrust} \otimes q^T$	A force which counteracts with the propulsion force is the force of gravitation presented as follows: $F_g{}^E = \begin{bmatrix} 0 \\ 0 \\ f_g \end{bmatrix}$ where f_g is the gravitational force which is most conveniently described in the inertial frame.

The net force on the quad-rotor as seen in the inertial frame is then as follows:

$$F_{net}{}^E = \begin{bmatrix} 2(q_1 q_3 + q_0 q_2) F_{thrust} \\ 2(q_2 q_3 - q_0 q_1) F_{thrust} \\ ({q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2) F_{thrust} - mg \end{bmatrix} \tag{3.18}$$

Moments of a Navigating Quadrotor

Analogous to the developed translational dynamics, the rotational dynamics can be formulated as follows by taking the developed net torque and reaction forces in to consideration. The analysis is done as follows:

Thrust Moments	Gyroscopic Moment
The thrust forces happen at the rotating tips of the quadrotor. This forces in turn produce torque about the centre of gravity of the body. Reaction torques are also produced about the centre of axis of each rotating propeller. The generated moments are formulated as follows: $\begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = \begin{bmatrix}RollMoment\\ PitchMoment\\ YawMoment\end{bmatrix}$ Due to the ease of mechanical structure of the quad-rotor, each of the Moments generated are associated to their respective attitudinal movements described by utilizing the control allocation matrix developed in subsection 3.1.4. $\begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = \begin{bmatrix}\frac{\sqrt{2}l}{2}({\omega_1}^2+{\omega_2}^2-{\omega_3}^2-{\omega_4}^2)\\ \frac{\sqrt{2}l}{2}({\omega_1}^2-{\omega_2}^2-{\omega_3}^2+{\omega_4}^2)\\ \frac{\sqrt{2}l}{2}(-{\omega_1}^2+{\omega_2}^2-{\omega_3}^2+{\omega_4}^2)\end{bmatrix}$	The gyroscopic moments occur due to the gyroscopic effect of the four rotating propellers. When there is a change in plane of rotation of each of the four propellers, a gyroscopic torque is induced about the axis perpendicular to the desired axis of rotation. The summation of this effect is put as the following: let, $f(u) = \omega_1 - \omega_2 + \omega_3 - \omega_4$ and, $M_{gx} = I_{prop}q(-\omega_1+\omega_2-\omega_3+\omega_4)$ $M_{gy} = I_{prop}p(\omega_1-\omega_2+\omega_3-\omega_4)$ $M_{gz} = 0$ where I_{prop} is the rotor inertia.

The net moment that acts on the quad-rotor about the in the body reference frame is:

$$ {M_{net}}^B = \begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = \begin{bmatrix}\frac{\sqrt{2}l}{2}({\omega_1}^2+{\omega_2}^2-{\omega_3}^2-{\omega_4}^2) - I_{prop}qf(u)\\ \frac{\sqrt{2}l}{2}({\omega_1}^2-{\omega_2}^2-{\omega_3}^2+{\omega_4}^2) + I_{prop}pf(u)\\ \frac{\sqrt{2}l}{2}(-{\omega_1}^2+{\omega_2}^2-{\omega_3}^2+{\omega_4}^2)\end{bmatrix} \tag{3.19} $$

3.2.3.2 Kinematic Model of the Quad-rotor

The kinematics of the quad-rotor depicts the relationship between a stationary reference frame and moving reference frames attached to the quad-rotor as shown in Figure 3.9. The propagation between the two reference frame comprises of a translational component and a rotational component both of which are described in detain in the following analysis.

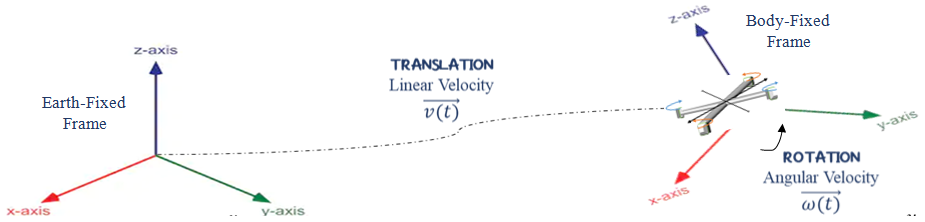


Figure 3.9: Kinematic Relationship

Position Propagation Equations	Attitude Propagation Equations
The relative position between a stationary reference frame and the body attached reference frame is described by the position propagation equations. The linear velocity, $[u, \quad v, \quad w]$, tracks the information of the timely change of the inertial position as seen in the body frame. The relationship between the linear velocity and its inertial parameters can be found by utilizing the rotation matrix developed as Eqn: 3.14. $$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_{\mathrm{B}}^{\mathrm{E}} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$ where $\begin{bmatrix} u \\ v \\ w \end{bmatrix}$ depicts the linear velocity and $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$ denotes its inertial counterpart.	To describe the attitude propagation, the quaternion differential equation has to be solved. The resulting equations relate angular velocity with the inertial attitude rates. With the algebraic quaternion $q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$ encoding orientation, its rate of change can be described as follows: $$\dot{q} = \frac{1}{2} q \otimes \begin{bmatrix} 0 \\ \omega \end{bmatrix} \tag{3.20}$$ where $\otimes$ represents the quaternionic multiplication. The detail of the derivation is presented in Appendix B.2. Making use of the quaternion matrix representation, described in Eqn: 3.9, the quaternion differential equation becomes as follows: $$q = \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ p \\ q \\ r \end{bmatrix}$$

3.2.3.3 Equations of Motion

The equations of motion are developed using the Newton's laws of motions along with Euler's laws formulated for the rotational dynamics. The analysis is as follows:

Conservation of linear momentum	Conservation of angular momentum
The net translational force is responsible for any translational motion the vehicle performs. The relationship can be derived by reformulating the Newton's second law as the rate of linear momentum.	Analogous to the Newton's second law of motion, rate of angular momentum is described as follows. The analysis results in equations of motion describing how the pose of the quad-rotor evolves.
$$F_{net} = ma = m\frac{dv}{dt} = \frac{dmv}{dt} = \frac{dP^E}{dt}$$	$$M_{net} = I\alpha = I\frac{d\omega}{dt} = \frac{dI\omega}{dt} = \frac{dH^E}{dt}$$
while performing the above analysis, the transport theorem has to be employed since two reference frames are being used [47]. The Coriolis terms embeds the information of relative velocity between the two reference frames.	Analogous to the translational subsystem, the transport theorem has to be applied so as to describe the rotational dynamics in the same reference frame. Hence the Coriolis terms appears as follows:
$$F_{net} = \frac{dP^B}{dt} + \omega^B \times P^B$$	$$M_{net} = \frac{dH^B}{dt} + \omega^B \times H^B$$
	Then the net moment is then reformulated as follows:
$$F_{net} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \begin{bmatrix} \dot{u} + qw - vr \\ \dot{v} + ur - pw \\ \dot{w} + pv - qu \end{bmatrix}$$	$$M_{net} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} - (I_{yy} - I_{zz})qr \\ I_{yy}\dot{q} - (I_{zz} - I_{xx})pq \\ I_{zz}\dot{r} - (I_{xx} - I_{yy})pr \end{bmatrix}$$

A more compact representation describing both the kinematics and dynamics of the quad-rotor with the neglection of the gyroscopic term presented in Eqn: 3.19, is shown in Eqn: 3.21. The equation describes how the quad-rotors' thirteen states evolve over time being constrained with various forces and turning effects. The control allocation along with the flight dynamics as built in Simulink is further illustrated in Figure 3.10.

$$\frac{d}{dt}\begin{bmatrix} p \\ \dot{p} \\ q \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{p} \\ q \otimes \frac{F_{thrust}}{m} \otimes q^T - \overrightarrow{g} \\ \frac{1}{2} q \otimes \omega \\ J^{-1}[\tau - \omega \times J\omega] \end{bmatrix} \tag{3.21}$$

where m is the vehicle mass, $p = [\text{x y z}]^T$ represents the vehicle's position in the inertial frame, the unit quaternion $q = (q_w, \overrightarrow{q_v})^T$ describes the vehicle's attitude, $\overrightarrow{g}$ denotes the

gravitational acceleration, J denotes the inertia tensor of the vehicle body and $\omega = [\,p\ q\ r\,]^T$ represents the angular velocity of the vehicle. Whereas, τ denotes the generated torque and F_{thrust} expresses the collective thrust force generated in body reference frame as a result of the forces generated by each of the propellers.

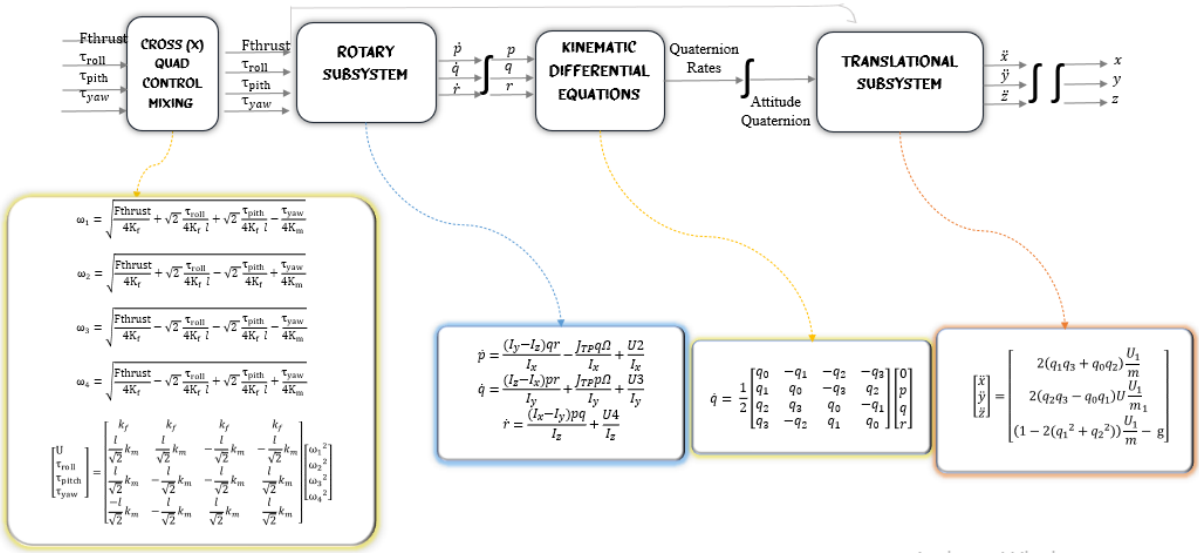


Figure 3.10: Overall Flight Dynamics

3.3 Actuator Dynamics

In the quad-rotor system, the motors not entirely capable of producing whatever thrust the controller requires. In reality the motors have limited capability. The thrust that the motors generate is limited by the peak torque the motors can generate. The peak torque poses limitations on the maximum acceleration that can be achieved. The maximum effective thrust is thus constrained by the maximum capability of the actuators.

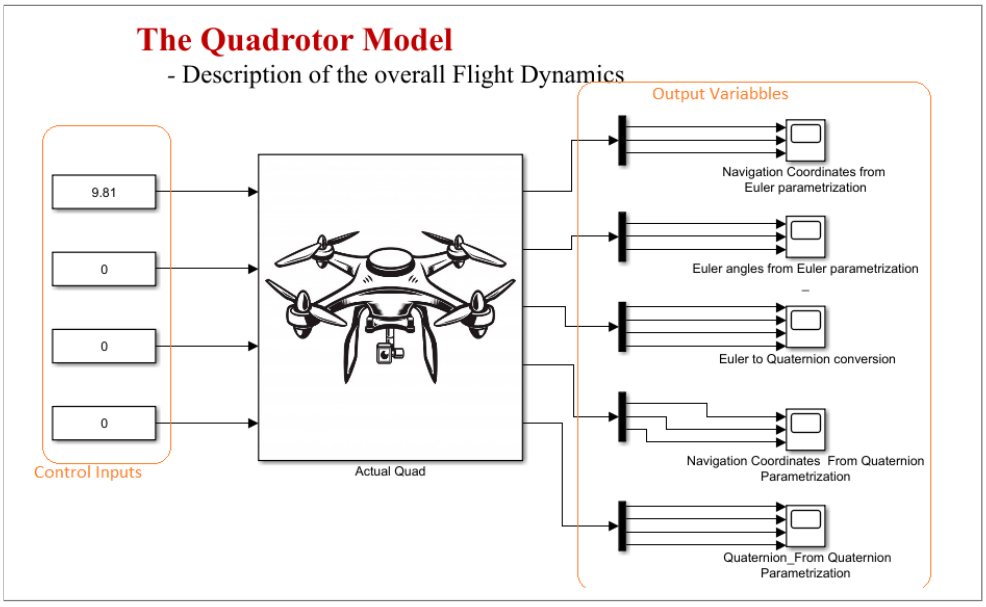
The commanded motor speeds ω_1, ω_2, ω_3 and ω_4 are calculated by the controller and are then commanded to the actuator. The response of the actuator can be modelled by a first order system with a constant gain k and a time constant τ as follows:

$$w = \frac{k \cdot w_c}{\tau s + 1} \tag{3.22}$$

where w corresponds to attained motor speed and w_c corresponds to commanded motor speed. However, in practice, it is fairly possible to assume that the motor dynamics is relatively fast compared to the rigid body dynamics and the aerodynamics [34]. Thus, in this paper's analysis, it is assumed that commanded speeds of the motors can be instantaneously achieved.

3.4 Model Verification

In this section, model verification is performed by testing the overall 6 DOF motion of the quad-rotor. Since the quaternion attitude representation lacks physical intuition, the Euler's parametrization of attitude is also used for the verification purpose only. Hence results from both parametrizations are compared so as to verify the faithful representation of the developed model. Figure 3.11 presents the open loop dynamics of the quad-rotor as built in Simulink. Attitudinal change along with navigation behaviour of the quad-rotor is tested with the application of the four control commands.



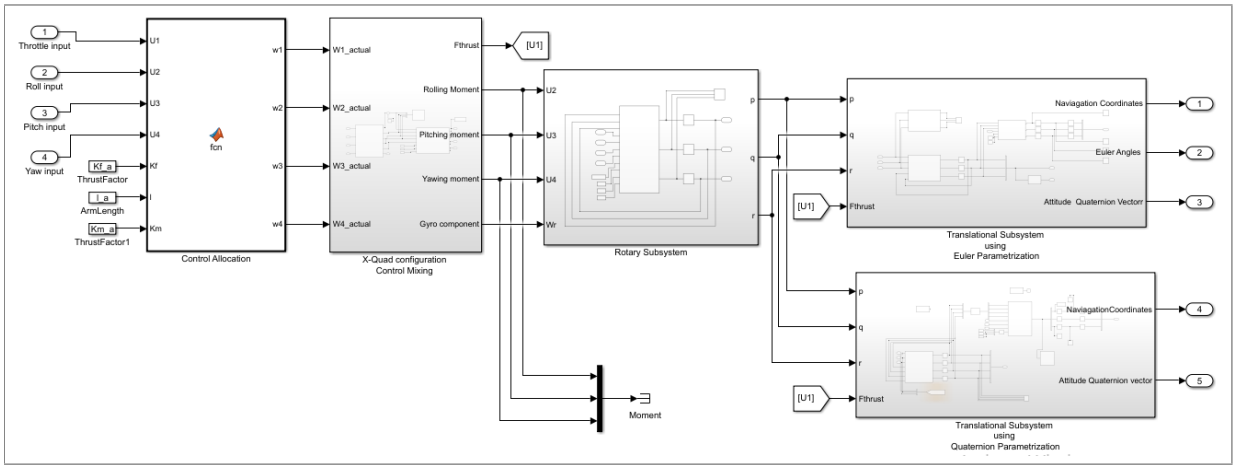
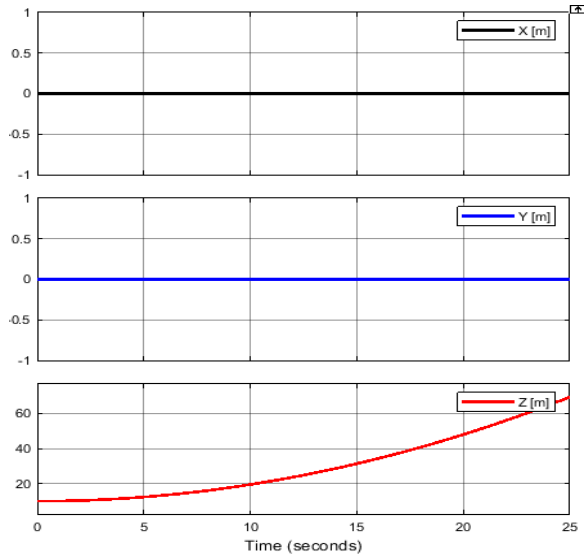
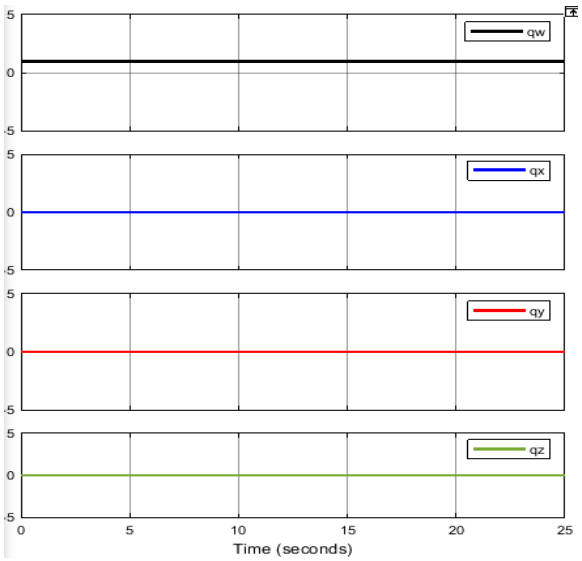


Figure 3.11: Flight Dynamic model

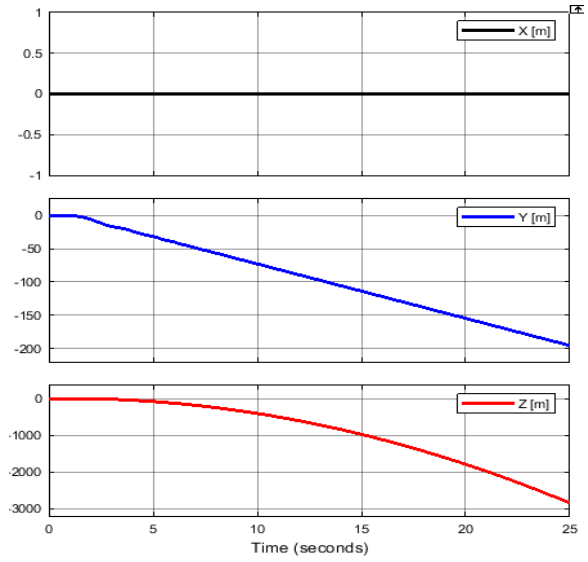
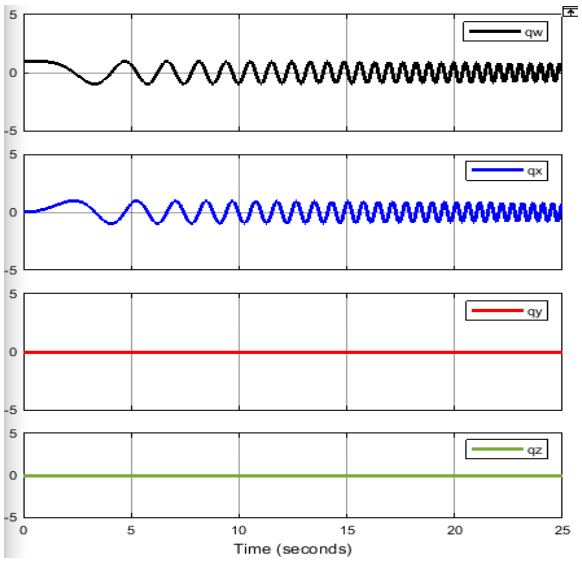
The throttle command:

The application of throttle command with positive U_1 which greater that the counteracting gravitational force mg results in the quad-rotor moving in the vertical Z direction with no attitudinal change. Thus, the attitude quaternion is at it's non-rotated position i.e: $q = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$.

Navigation Coordinates	Attitude Representation
	

The roll command:

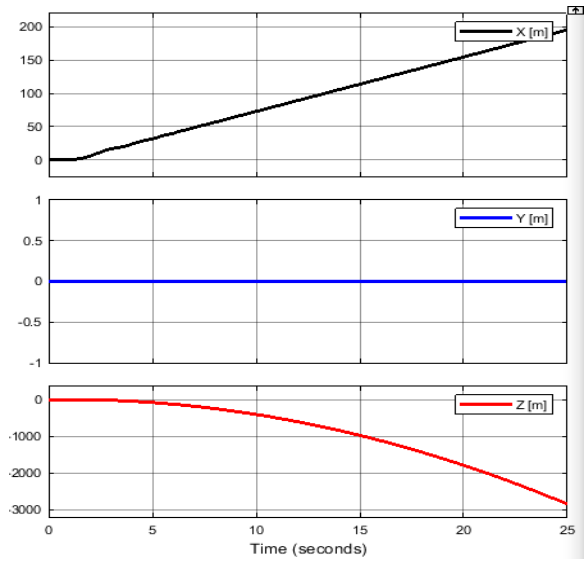
With a positive the roll command τ_x given, the quad-rotor is expected to tilt about X-axis making it navigate transnationally in the Y-direction. No attitudinal change is expected about the axes: $Y \quad and \quad Z$. Thus, the attitude quaternion will have the form: $q = \begin{bmatrix} q_\omega \\ q_x \\ 0 \\ 0 \end{bmatrix}$.

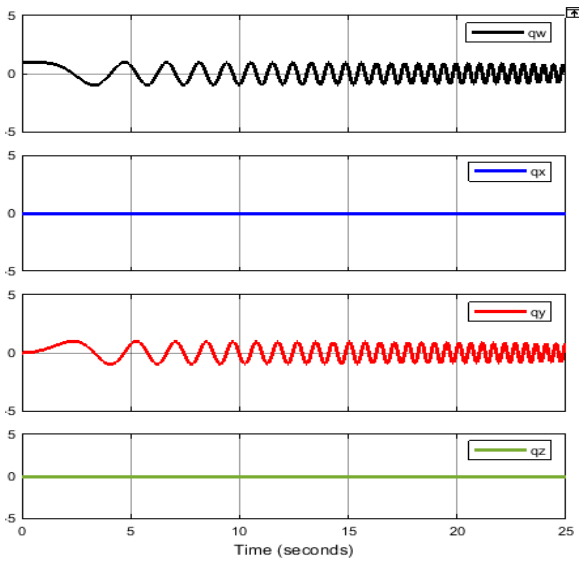
Navigation Coordinates	Attitude Representation
	

The pitch command:

With a positive the roll command τ_x applied to the quad-rotor, it is expected to tilt about Y-axis making it navigate transnationally in the X-direction. No attitudinal change is expected about the axes: $X \quad and \quad Z$. Thus, the attitude quaternion will have the form: $q = \begin{bmatrix} q_\omega \\ 0 \\ q_y \\ 0 \end{bmatrix}$.

Navigation Coordinates	Attitude Representation

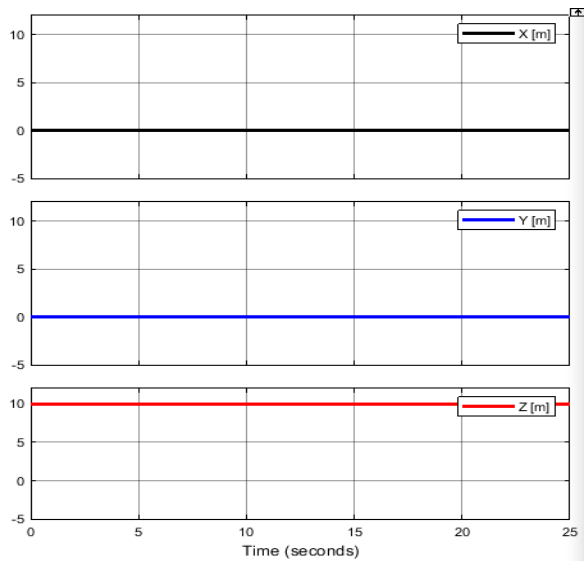


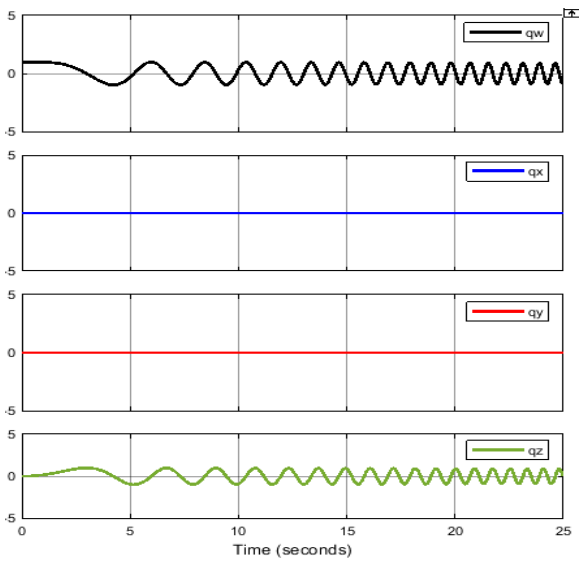


The yaw command:

A positive yaw command τ_z being given to the quad-rotor is expected to not alter the translational position of the quad-rotor with no attitudinal change expected about X and Y. Thus, the attitude quaternion will have the form: $q = \begin{bmatrix} q_\omega \\ 0 \\ 0 \\ q_z \end{bmatrix}$.

Navigation Coordinates	Attitude Representation





Chapter 4

Controller Design

4.1 Overview

In various cases, conventional feedback controllers make plants behave in undesired ways because of variations in process dynamics which is mainly due to imposed non-linear actuators and abrupt changes in environmental conditions. Modelling errors are also another source of model discrepancies which are imposed into systems as design-time errors. Robust control and adaptive control are two approaches in control systems that are employed with the aim of handling uncertainties and model discrepancies. Robust control systems are designed in a way that they allow fixed gains by considering worst case scenarios of system uncertainties, and hence systems are guaranteed to be stable with respect to such uncertainties [48]. Since robust control schemes are tuned for adverse case performance, there is no room for them to improve system response in time. So robust controls are often employed to make sure a certain closed loop system remains stable in the presence of adverse disturbance. On the other hand, adaptive control techniques are designed so that controller gains can be tuned on-line in response to plant variations so as to achieve better performance. Hence, they are made to learn system uncertainties and suppress their effects online which as a result makes system response to improve gradually through operation [49].

Adaptive control was first developed in the 1950's to automatically adapt to the changing parameters in aircraft [50]. Throughout the years, adaptive control have shown great success in ensuring acceptable performance of the problem of control of non-linear systems with various complexities operating in challenging environments.

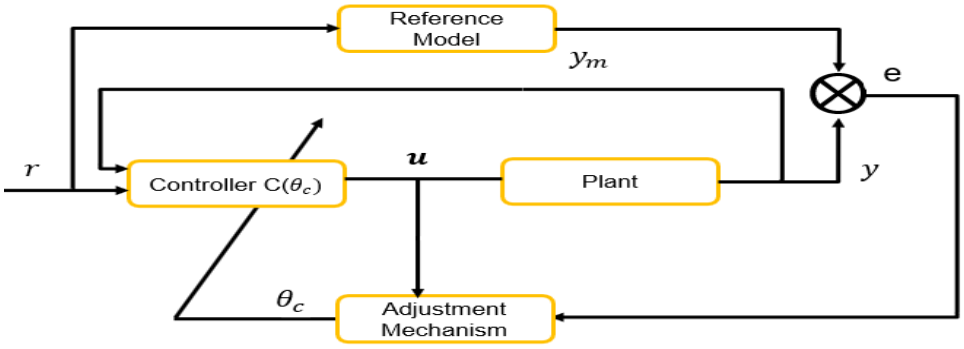
Aerial vehicles, like any other dynamical systems, suffer from parametric uncertainty. It's parameters maybe partially known or time varying. The capability of adaptive

controllers in ensuring reference tracking with the presence of system uncertainties makes them an ideal candidate to control systems like aerial vehicles. Outdoor performance of aerial vehicles is one area which poses different off-nominal scenarios to the overall stabilizing controller. In the presence of this scenarios, the main goal of adaptive control is to estimate controller parameters on-line and compute a control law in response to variation in the environment and internal parameters.

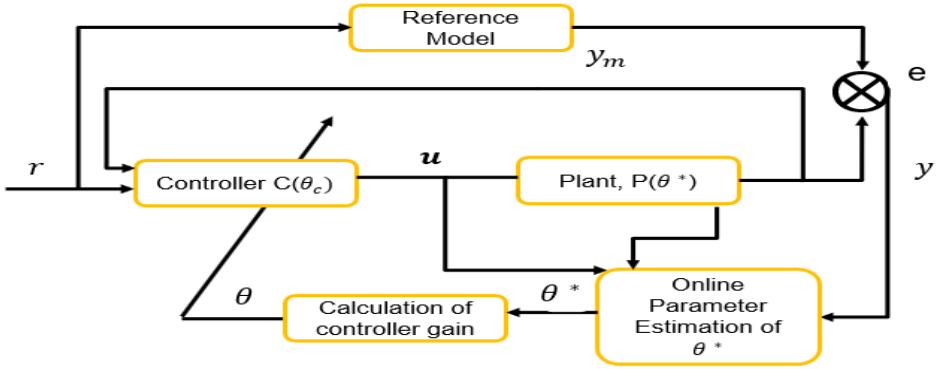
The model reference adaptive controller (MRAC) is a widely used scheme used in the arena of adaptive control. A certain model reference adaptive control scheme has the following important block of functions.

1. A reference model
    - The reference model is a certain structure which encodes the desired transient response of a certain plant. It offers a constraint for the true plant dynamics making it follow its encoded transient response.
2. A plant
    - Plant is a representation of the true plant dynamics with a known structure, but with fully or partially unknown parameters to be estimated using adaptation schemes.
3. An adaptation (update) law
    - Adaptation law provides parameter adaptation mechanism which ensure asymptotic convergence of model tracking error. The update law in conventional MRAC is driven solely by tracking error between the reference model and the measurement of the plant [49].
4. A control law
    - The control law which is driven from the adaptation law is designed in a way that it is linearly parametrized with respect to the unknown parameters. The control law is responsible to ensure tracking error convergence and stability of the whole system.

The main objective of model reference adaptive controller (MRAC) is to guarantee that a certain dynamical system follows the behaviour of a chosen reference model [50]. The reference model can be chosen in any way as long as it captures the desired performance and behaviour of the plant. Furthermore, it is intended that the chosen reference model is stable with an input which is bounded. MRAC design comes in two basic schemes named as the direct adaptive control and indirect adaptive control as illustrated in Figure 4.1.



(a) Direct MRAC



(b) Indirect MRAC

Figure 4.1: Model Reference Adaptive Control Schemes

- In direct adaptive control, there's is an on-line estimator which directly estimates the controller parameters which are then used to update the control law that goes in to the system.
- In the indirect scheme the adaptation of the controller parameters is done in two stages:
  i. on-line estimation of the plant parameters and
  ii. on-line computation of controller parameters depending on the estimated plant parameters.

4.2 Controller Architecture

The workspace of the quad-rotor is of six dimensions comprising of the three translational degree of freedoms and the three rotational degree of freedoms. However, the system allows only four control inputs. This under actuated nature of the

quad-rotor's makes the control design to be explicit. A hierarchical control approach is the common type control employed for quad-rotors where a lower level controller is designed for attitude tracking and a higher level control scheme is developed for position control.

The overall controller is developed by decoupling position and attitude control and designing an adaptation scheme for both. First a baseline controller is designed to guide the quad-rotor along a desired path when it is operating in nominal scenarios[1]. A model reference adaptive controller is then adopted to embed adaptation behaviour in to the controller for the problem of tracking control in off-nominal scenarios [2]. This is mostly important for outdoor applications of quad-rotors where the application area of this paper is planned on. The fact that the control architecture consists of two separate controllers gives ease to the overall controller by letting the adaptation process to emerge only for unpredictable scenarios which are inherent in harsh environments the quad-rotor may operate in. For nominal operating conditions, the baseline controller enables trajectory tracking and ensures system stability. Whereas the presence of off-nominal conditions seeks adaptive augmentation for overall system stability. The overall controller architecture developed in this paper is illustrated in the block diagram shown in Figure 4.2.

4.3 Baseline Controller Design

The baseline controller design comprises of two hierarchical controller: the position and attitude controllers which are desired to stabilize the system in nominal condition. A quaternion state feedback controller is employed for the outer loop position control. Whereas, an attitude controller which makes use of the concept of geometric tracking control is applied for the problem of attitude control. This section follows by developing the baseline quaternion state feedback position controller which is then used to identify the desired attitude reference to be fed to the quad-rotor system. Then the baseline attitude controller is formulated based on the obtained attitudinal reference.

4.3.1 Position Controller

In this section, a Lyapunov based quaternion state feedback controller is designed for position control of the quad-rotor. This controller is expected to provide position stabilization during nominal conditions the quad-rotor may operate in. The proposed

[1]Nominal Scenarios: operating conditions where no fault, model uncertainty and disturbance exist.
[2]Off-nominal Scenarios: situations where systems behave in a manner not intended at design time.

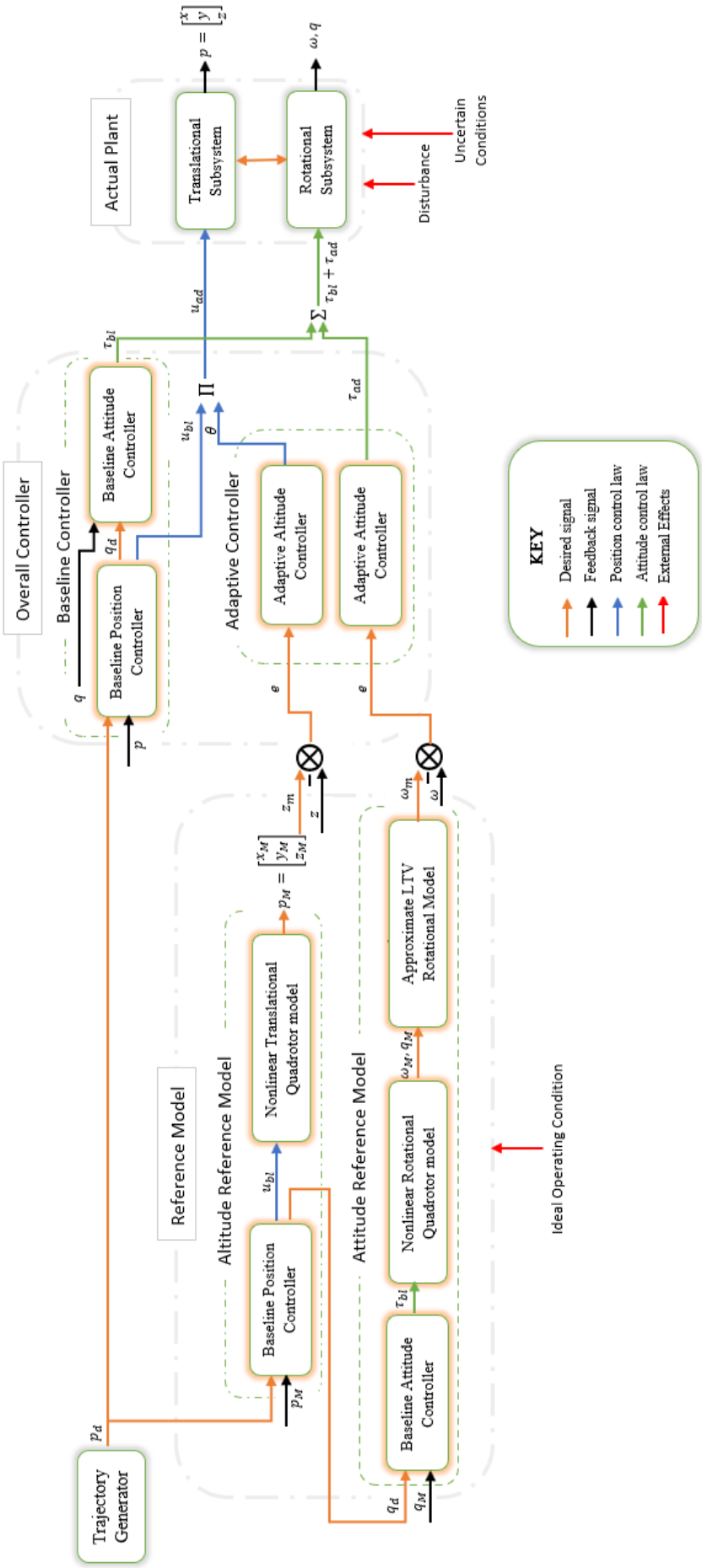


Figure 4.2: Overall Controller Architecture

quad-rotor model is used in the process of the controller design. Recalling the translational dynamical equations of the quad-rotor as recalled from Eqn: 3.21

$$\frac{d}{dt}\begin{bmatrix} p \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \dot{p} \\ q \otimes \dfrac{F_{thrust}}{m} \otimes q^T - \overrightarrow{g} \end{bmatrix}. \tag{4.1}$$

where m is the vehicle mass, $p = [\text{x y z}]'$ represents the vehicle's position in the inertial frame, the unit quaternion $q = (q_w, \overrightarrow{q_v})^T$ describes the vehicle's attitude and $\overrightarrow{g}$ denotes the gravitational acceleration. Whereas, F_{thrust} expresses the collective thrust force generated as a result of the forces generated by the propellers as seen in the body reference frame.

4.3.1.1 Closed Loop Error Dynamics

The error dynamics, which depicts the position tracking error, can be constructed for the translational subsystem of Eqn: 4.1. Given a trajectory as $p_d = [X_d\ Y_d\ Z_d]^T$, the position and the velocity errors are defined as follows:

$$p_e = p - p_d$$

$$\dot{p_e} = \dot{p} - \dot{p_d}$$

where p_d denotes the desired position vector and p denotes the attained position in a three dimensional space. Then, the translational error dynamics can be formulated as follows:

$$\frac{d}{dt}\begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} = \begin{bmatrix} \dot{p_e} \\ \dfrac{F_u}{m} - \overrightarrow{g} \end{bmatrix} \tag{4.2}$$

where $F_u = q \otimes F_{thrust} \otimes q^T$ represents the collective thrust force as seen in the inertial frame of reference. The translational error dynamics in Eqn: 4.2 can be rewritten as follows

$$\frac{d}{dt}\begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} = \begin{bmatrix} 0^{3x3} & I^{3x3} \\ 0^{3x3} & 0^{3x3} \end{bmatrix} \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} + \begin{bmatrix} 0^{3x3} \\ m^{-1} I^{3x3} \end{bmatrix} (F_u - m\overrightarrow{g})$$

From the error dynamics, it can be seen that the collective thrust force, F_u can be designed in the inertial frame so that the desired position and velocity converge to zero [51]. Hence, any desired position can be achieved with this set-up.

A Lyapunov based controller design is followed resulting in the following control force situated in the inertial frame.

$$F_u = -K_p(p - p_d) - K_v(\dot{p} - \dot{p_d}) + m\overrightarrow{g} \tag{4.3}$$

where K_p and K_v are positive definite 3×3 diagonal matrices denoting proportional and derivative gains of the controller respectively. With U_{bl} symbolizing the baseline position control law that stabilizes the system in nominal conditions, it can formulated as:

$$U_{bl} = F_u = -K_p(p - p_d) - K_v(\dot{p} - \dot{p_d}) + m\overrightarrow{g} \tag{4.4}$$

The lyapunov based design is presented in the following section where the control force is shown to guarantee the stabilization of the quad-rotor in any given desired position, p_d.

4.3.1.2 Error Convergence and Stability Analysis

By formulating a positive definite Lyapunov function as in Eqn: 4.5, it is shown that the chosen translational control force in Eqn: 4.3 ensures system stability and error convergence. The analysis is as follows:

$$V = \frac{1}{2}\begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} * \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} \tag{4.5}$$

In order to guarantee stability using Lyapunov analysis, the derivative of the chosen Lyapunov function has to be negative semi-definite [52]. The derivative of the Lyapunov function can then be analysed as follows:

$$\dot{V} = \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} * \frac{d}{dt}\begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix}$$

$$\dot{V} = \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} * \begin{bmatrix} \dot{p_e} \\ \dfrac{F_u}{m} - \overrightarrow{g} \end{bmatrix}$$

$$\dot{V} = \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} * \begin{bmatrix} 0^{3x3} & I^{3x3} \\ 0^{3x3} & 0^{3x3} \end{bmatrix} \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} + \begin{bmatrix} 0^{3x3} \\ m^{-1}I^{3x3} \end{bmatrix} (F_u - m\overrightarrow{g})$$

By the application of the control law defined in Eqn: 4.3, the derivative of the lyapunov function will be :

$$\dot{V} = \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} * \begin{bmatrix} \dot{p_e} \\ \dfrac{(-K_p p_e - K_v \dot{p_e}) + m\overrightarrow{g}}{m} - \overrightarrow{g}) \end{bmatrix} = \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} * \begin{bmatrix} \dot{p_e} \\ \dfrac{(-K_p p_e - K_v \dot{p_e})}{m} \end{bmatrix}$$

More compactly, the derivative of the lyapunov function, $\dot{V}$ can be rewritten as:

$$\dot{V} = \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} \left[\begin{bmatrix} 0_{3x3} & I_{3x3} \\ 0_{3x3} & 0_{3x3} \end{bmatrix} - \begin{bmatrix} 0_{3x3} \\ m^{-1} I_{3x3} \end{bmatrix} \begin{bmatrix} K_p & K_v \end{bmatrix} \right] \begin{bmatrix} p_e \\ \dot{p_e} \end{bmatrix} \tag{4.6}$$

In order for Eqn: 4.6 to be negative semi-definite, the following has to be true.

$$eig\left(\begin{bmatrix} 0_{3x3} & I_{3x3} \\ 0_{3x3} & 0_{3x3} \end{bmatrix} - \begin{bmatrix} 0_{3x3} \\ m^{-1} I_{3x3} \end{bmatrix} \begin{bmatrix} K_p & K_v \end{bmatrix} \right) \text{ has to have negative real parts.}$$

Therefore, with a selective choice of the proportional and derivative gains so that the above can be true, the control law formulated in Eqn: 4.3 is guaranteed to asymptotically stabilize the translational error dynamics presented in Eqn: 4.2. Hence the asymptotic stability of the translational subsystem as put in Eqn: 4.1 is ensured under the chosen control law.

4.3.1.3 The Quaternion Trajectory

Navigation systems for quad-rotors specify signal for the desired trajectory for each of the four flat outputs as described in Chapter - 5. Being provided with preprepared signals for these four flat outputs, the position controller is made to generate the desired attitude which is to be fed as a reference trajectory for the attitude controller.

The complete dynamics of the quad-rotor consisting of translational and rotational subsystems is highly coupled. Recalling the formulation in Eqn: 4.2, it can be seen that the desired effective thrust force, F_u can not be designed in an arbitrary direction since the effective thrust force that the propellers exert is greatly dependent on the attitude subsystem. For the stabilizing control law F_u such that $F_u = q \otimes F_{thrust} \otimes q^T$ to effectively control the translational dynamics, the quad-rotor's effective thrust F_{thrust} has to be oriented such that it aligns with its inertial representation F_u. Thus, the desired quaternion vector q has to formulated in a way that this alignment can be achieved.

The desired attitude trajectory formulation is based on the analysis of the shortest rotation between the two vectors: F_u and F_{thust}. It is to be noted that the designed control force F_u is situated in the inertial frame while the effective thrust force, F_{thrust} is generated in the body frame. Thus a certain quaternion vector is needed such that F_u becomes parallel to F_{thrust}.

By utilizing Euler-Rodrigues formula shown in Eqn: 3.15, the algebraic quaternion depicting the desired rotation, q_d can be reformulated as follows:

$$q_d = e^{\frac{1}{2}\theta_d \overrightarrow{u_d}} = \cos(\frac{\theta_d}{2}) + \overrightarrow{u_d}\sin(\frac{\theta_d}{2}) \tag{4.7}$$

where $\overrightarrow{u_d}$ denotes a unit vector to be used as axis of rotation and θ_d is the angle of the shortest rotation to the desired orientation.

Let the unit vectors $\overrightarrow{n_{th}}$ and $\overrightarrow{n_u}$ are normalized vectors of F_{thrust} and F_u, where $\overrightarrow{n_{th}} = {}^{[0\ 0\ 1]^T}$ is constant and always points in the Z-direction of the body axis. Then, the following definition holds true for the dot and cross product of these normalized vectors.

$$\overrightarrow{n_u} \cdot \overrightarrow{n_{th}} = \cos\theta_d \qquad \overrightarrow{n_u} \times \overrightarrow{n_{th}} = \overrightarrow{u_d}\sin\theta_d$$

By the consideration of the double coverage property developed in Section 3.2.2.4, the application of the half angle formula of trigonometric identities results in the following:

$$\cos(\frac{\theta_d}{2}) = \pm\sqrt{\frac{(1+\cos\theta_d)}{2}}, \quad \sin(\frac{\theta_d}{2}) = \pm\sqrt{\frac{(1-\cos\theta_d)}{2}}$$

The process of utilizing the half angle representations gives us the algebraic quaternion formula which rotates the quad-rotor such that F_{thrust} is parallel to F_u.

$$q_{xy} = \pm(\sqrt{\frac{(1+\overrightarrow{n_u}\cdot\overrightarrow{n_{th}}}{2}} + \frac{\overrightarrow{n_u}\times\overrightarrow{n_{th}}}{\|\overrightarrow{n_u}\times\overrightarrow{n_{th}}\|}\sqrt{\frac{(1-\overrightarrow{n_u}\cdot\overrightarrow{n_{th}}}{2}}) \tag{4.8}$$

$$\text{where,} \quad \theta_d = \frac{\overrightarrow{n_u}\times\overrightarrow{n_{th}}}{\|\overrightarrow{n_u}\times\overrightarrow{n_{th}}\|} \quad \text{and} \quad F_{thrust} \coloneqq \|(F_u\|$$

The normalized vector $\overrightarrow{n_{th}}$ is always aligned with the vertical axis of the quad-rotor's body frame. Hence the direction of rotation in Eqn: 4.8, described by the cross product of the two vectors depicts rotation only in the $X-Y$ plane with no information embedded regarding rotation about the vertical Z axis of the body frame. Thus another successive rotation has to be made in order to incorporate all the rotations in three dimensional space. This can be done by introducing another quaternion vector q_z which rotates the vehicle about the third axis and composing this rotation along with the previous rotation quaternion vector q_{xy} . And the fully describable desired quaternion, q_d, which comprises of all the three informations of rotation can be described by the following composite rotation.

$$q_d \coloneqq q_{xy} \otimes q_z \tag{4.9}$$

where q_z symbolizes the desired rotation over the z-axis.

Moreover, the desired angular velocity, ω_d, describing the rate at which the desired attitude has to change can be calculated from the relation in Eqn: 4.7 as follows:

$$q_d = e^{\frac{1}{2}\theta_d \overrightarrow{u_d}}$$

$$\theta_d \overrightarrow{u_d} = 2ln(q_d)$$

$$\omega_d = \frac{d}{dt}\theta_d = 2\frac{d}{dt}ln(q_d) \tag{4.10}$$

4.3.2 Attitude Controller

The navigation system along with the position controller provides the attitude controller with signal of desired orientation and its rate of change. Hence, given a desired trajectory, the attitude control loop has to make the vehicle track a desired attitude reference command. The attitudinal information is decoded by an algebraic quaternion quaternion, q_d, and the desired angular velocity, ω_d, as formulated in Eqn: 4.9 and Eqn: 4.10.

4.3.2.1 The Quaternion Error

The error in attitude tracking is depicted by the error quaternion q_e, defined as follows:

$$q_e \coloneqq q_d^{-1} \otimes q$$

where q_d encodes information of the desired orientation and q denotes the attained attitude of the vehicle.

Since the quaternions considered in the analysis are of unit length, the above equation can be rewritten as the following by making use of the property mentioned in Eqn: 3.7.

$$q_e \coloneqq q_d^{T} \otimes q \tag{4.11}$$

Attitude Controller Development

Using quaternions for attitude representation gives access to a description of attitude free from any singularities. However,a single full rotation decoded by a quaternion in the

four dimensional space is equivalent to a double coverage of the three dimensional space due to the double coverage scenario of quaternions described in Section 3.2.2.4. This non-unique behaviour of quaternions gives rise to regions of attraction and repulsion in the neighbourhood of $q \in \text{SO(3)}$ and $-q \in \text{SO(3)}$. This as a result causes the so called unwinding phenomenon[3] of attitude control [53].

The unwinding phenomenon is inherent in case of feedback control using quaternions. A feedback designed using unit quaternion do not appear to be consistent with a feedback designed on SO(3) since for some attitude q, a certain quaternion feedback controller may take one of the two possible values of quaternion feedback which can not be well defined in SO(3).

With the aim of avoiding the unwinding phenomenon, a novel Lyapunov based controller developed in [34] is adopted as a baseline controller to stabilize the attitude subsystem.

4.3.2.2 Closed Loop Error Dynamics

Given a two times differentiable desired attitude encoded by q_d and a desired angular velocity denoted by w_d, the closed loop error dynamics incorporating the attitude and angular velocity tracking error can be calculated as follows:

Rate of quaternion error:
The rate at which the quaternion error evolves is described by its closed loop dynamics as follows, by recalling the error quaternion developed in the previous section.

$$q_e \coloneqq q_d^T \otimes q$$

$$\dot{q_e} = \frac{d}{dt}(q_d^T \otimes q)$$

$$\dot{q_e} = \dot{q_d}^T \otimes q + {q_d}^T \otimes \dot{q}$$

$$\text{But,} \quad \dot{q_d}^T = -\dot{q_d} \qquad \text{, Thus} \quad \dot{q_e} = \frac{1}{2}[-\dot{q_d} \otimes q + {q_d}^T \otimes \dot{q}]$$

By the application of the attitude propagation equations of the quad-rotor as developed in Eqn: 3.20, the closed loop attitude error dynamics, $\dot{q_e}$, which holds the information of how the quaternion error evolves is formulated as follows:

[3]Unwinding Phenomenon : an inconsistent feedback scenario caused in feedback attitude control in the quaternion space

$$\dot{q} = \frac{1}{2} q \otimes \omega = \frac{1}{2} \omega \otimes q^T \qquad \text{and} \qquad q_e = q_d \otimes q$$

$$\dot{q_e} = \frac{1}{2}[-\omega_d \otimes {q_d}^T \otimes q + {q_d}^T \otimes q \otimes \omega]$$

$$\dot{q_e} = \frac{1}{2}[-\omega_d \otimes q_e + q_e \otimes \omega] \tag{4.12}$$

Rate of angular velocity tracking error:
The relative velocity between the desired and body reference frame can also be formulated as follows:

$$\omega_e = \omega - \omega_d$$

It has to be noted that the desired angular velocity is not described in the same reference frame to that of the measured angular velocity which is attached to the body reference frame. So a differentiation step done toward obtaining the rate at which the relative velocity evolves makes the 'Coriolis' term appear. The transport theorem [42] is utilized for the process, which connects the time derivative of a vector with respect to two different reference frames rotating with respect to the other. So the rate of the angular velocity error, which shows how the relative angular velocity evolves over time, can be formulated as follows with its components described in the body attached reference frame.

$$\dot{\omega_e} = \dot{\omega} - \dot{\omega_d} - \omega \times \omega_d$$

With the neglection of the gyroscopic term, the angular velocity dynamics can be recalled from Eqn: 3.21 as follows:

$$\dot{\omega} = J^{-1}[\tau - \omega \times J\omega] \tag{4.13}$$

By making use of Eqn: 4.13, the rate relative angular velocity can be reformulated in the body reference frame as follows:

$$\dot{\omega_e} = J^{-1}[\tau - \omega \times J\omega] - \dot{\omega_d} - \omega \times \omega_d$$

A further representation can be obtained by making use of the compact representation of cross products, $[\quad]_\times$, as follows:

$$\dot{\omega_e} = J^{-1}[\tau - \omega_\times J\omega] - \dot{\omega_d} - \omega_\times \omega_d \tag{4.14}$$

4.3.2.3 Error Convergence and Stability Analysis

Utilizing the attitude and angular velocity error dynamics from Eqn: 4.12 and 4.14, a positive definite lyapunov function can be constructed as follows:

$$V = \frac{1}{2} K_q \|q_e\|^2 + \frac{1}{2} {\omega_e}^T \omega_e \tag{4.15}$$

where $\|q_e\|$ represents the norm of the quaternion error.

In order to ensure stability, the gradient of the positive definite Lyapunov function should be negative semi-definite so as to ensure the convergence of the defined to zero. The gradient of V can thus be calculated as follows:

$$\dot{V} = \frac{1}{2} K_q \frac{d}{dt} \|q_e\|^2 + \omega_e \dot{\omega_e}$$

Substituting the expression for $\dot{\omega_e}$ from Eqn: 4.14, the derivative of the Lyapunov function becomes:

$$\dot{V} = \frac{1}{2} K_q \frac{d}{dt} \|q_e\|^2 + \omega_e [J^{-1}[\tau - \omega_\times J\omega] - \dot{\omega_d} - \omega_\times \omega_d] \tag{4.16}$$

A baseline control of the following form is chosen so as to stabilize the attitude dynamics and make rate of the Lyapunov function to be negative semi-definite.

$$\tau_{bl} = \omega_\times J\omega + J(\dot{\omega_d} + \omega_\times \omega_d - K_q \log_v {q_e}^+ - K_w \omega_e) \tag{4.17}$$

where K_q and K_w are positive controller gains. And q^+ denotes the shortest rotation between two points as described in Eqn: 3.17.

With the application of the control law in Eqn: 4.17, the gradient of the Lyapunov function becomes as follows:

$$\dot{V} = \frac{1}{2} K_q \frac{d}{dt} \|q_e\|^2 + \omega_e [J^{-1}\omega_\times J\omega + J^{-1}J(\dot{\omega_d} + \omega_\times \omega_d - K_q \log_v {q_e}^+ - K_w \omega_e) - J^{-1}\omega_\times J\omega] - \dot{\omega_d} - \omega_\times \omega_d]$$

$$\dot{V} = \frac{1}{2} K_q \frac{d}{dt} \|q_e\|^2 + \omega_e [\dot{\omega_d} + \omega_\times \omega_d - K_q \log_v {q_e}^+ - K_w \omega_e - \dot{\omega_d} - \omega_\times \omega_d]$$

$$\dot{V} = \frac{1}{2} K_q \frac{d}{dt} \|q_e\|^2 + \omega_e [-K_q \log_v {q_e}^+ - K_w \omega_e]$$

By using the axis angle representation of the quaternion, in [54], it is shown that the quadratic distance function between two quaternions can be represented using the following expression in the three dimensional space.

$$\frac{1}{2} \frac{d}{dt} ||q||^2 = \omega^T \log_v {q_e}^+ \tag{4.18}$$

with the angular velocity described in the body reference frame. Hence for the quaternion error vector q_e, the quadratic distance function can be refolmulated as :

$$\frac{1}{2} \frac{d}{dt} ||q_e||^2 = \omega_e \log_v {q_e}^+ \tag{4.19}$$

where the representation ${q_e}^+$ follows the definition in Eqn: 3.17.

Hence, the derivative of the Lyapunov function can thus be:

$$\dot{V} = K_q \omega_e \log_v {q_e}^+ + \omega_e[-K_q \log_v {q_e}^+ - K_w \omega_e]$$

$$\dot{V} = -K_w {\omega_e}^2 \tag{4.20}$$

Hence, with a choice of a positive gain, k_w, it is shown that the chosen stabilizing control law τ_{bl} formulated in Eqn: 4.17 makes the gradient of the Lyapunov function to be equal to $-K_w {\omega_e}^2$ which is always negative semi-definite. Hence the control law is guaranteed to ensure attitudinal error convergence making the system asymptotically stable.

4.4 Adaptive Controller Design

An adaptive controller, in a nutshell, is a controller with adjustable parameters and a mechanism for adjusting selected parameters [55]. In this paper, the adaptive scheme is developed as model reference adaptive controller which makes use of a reference model decoding desired transient response of the vehicle. When introducing an adaptation behaviour in to a certain control scheme, there are three basic considerations that need to be made.

- The overall architecture of the underlying controller.
- Proper selection of parameters that need to be adapted by the adaptation scheme.
- A certain parameter adjustment mechanism to be employed.

Proposed Reference Model:

Though a standard MRAC design seeks a linear reference model, the altitude adaptive scheme developed in this paper makes use of non-linear reference model so that important dynamical behaviours of the vehicle will not be missed. The reference model's transient behaviour is decoded by the closed-loop behaviour of the non-linear plant controlled by the baseline position controller developed in Section 4.3.

Same approach is followed to develop a reference model for the attitude adaptive loop where an additional step is performed to encode the closed-loop behaviour of the non-linear attitude dynamics controlled by the baseline attitude controller using a custom linear time variant (LTV) model. The details of the development is shown in the analysis to follow.

4.4.1 Adaptive Altitude Loop

The adaptation for the altitude is constructed as a direct adaptive problem where no intermediary plant parameter estimation mechanisms are needed. The adaptation scheme for the altitude is built by employing the MIT rule[4].

The design starts by defining the error e as follows:

$$e = z - z_m \tag{4.21}$$

where z is the altitude of the plant and z_m is the encoded output altitude of the reference model.

The controller of the closed loop system is adapted by an adjustment mechanism by taking the output of the reference model as an input to adjust the control force U defined in Eqn: 3.1. Scenarios which impose uncertainty in to the altitude loop are the physical parameter mass, m and any disturbance forces which act on to the z-axis of the quad-rotor's body frame, named as U_d. No deliberate mass change is required for the application area of the quad-rotor system used in this paper. Hence, the only effect posing difficulties for the baseline controller to guarantee closed loop stability at all times is the disturbance U_{dz}.

In order to adjust the baseline position controller in uncertain scenarios, a certain parameter θ is defined to be used as an adjustable gain for the developed baseline position controller.

$$U = \theta \cdot U_{bl} \tag{4.22}$$

where U_{bl} is the baseline control law developed as Eqn: 4.4.

The implementation of the Modified Model Reference Adaptive Controller(MMRAC) adds an additive PID controller to the adjustment mechanism in order to account for abrupt changes the system may encounter. The PID controller guides the baseline controller to drive the altitude error e defined in Eqn: 4.21 to zero. The control law Eqn: 4.22 is modified as follows using the additive augmentation.

$$\therefore \qquad U_{ad} = \theta \cdot U_{bl} + (K_p \cdot e + K_i \cdot \int e dt + K_d \cdot \frac{d}{dt} e) \tag{4.23}$$

where K_p, K_i and K_d represent the proportional, integral and the derivative gains of the PID controller and θ represents the MIT gain parameter which follows from the parameter adjustment mechanism.

[4]The MIT rule is a scalar parameter adjustment law developed in Massachusetts Institute of Technology in 1961.

4.4.1.1 Parameter Adjustment Mechanism

The MIT adjustment law is derived by approximating a gradient descent procedure for an integral error squared performance criterion. In order to achieve adjustment mechanism, the parameter θ, a cost function J is defined which encodes the error that is desired to be minimized.

$$J = \frac{1}{2}e^2 \tag{4.24}$$

where e denotes the error between the plant and the model output.

The idea of the MIT rule is to adjust the parameter θ so as to minimize the cost function J. And, this is done by making θ change in the negative direction of the gradient of the cost function J.

$$\text{i.e.} \quad \dot{\theta} = -\gamma \frac{\partial J}{\partial \theta}$$

where γ is the adaptation gain which decides the speed of adaptation. The parametrization can further be done by noting how the error, e, changes with respect to change in the parameter, θ, as presented below.

$$\dot{\theta} = -\gamma \cdot e \cdot \frac{\partial e}{\partial \theta} \tag{4.25}$$

Following the approach followed by [55], the rate at which the error changes in response to the parameter change, i.e. $\dfrac{\partial e}{\partial \theta}$ can be identified.

With the transfer function of the quad-rotor system defined as $G(s)$ in the frequency domain, an unknown influence on the system can be modelled by a system parameter $'k'$. $U(s)$ refers to the overall control input to the actual system and $U_{bl}(s)$, on the other hand, denotes the control input to the defined reference model. And by taking a known modelled influence which is inherent to the reference model modelled as $'k_o'$, the altitude error dynamics can be rewritten as follows in the frequency domain:

$$e(s) = kG(s)U(s) - k_o G(s)U_{bl}(s) \tag{4.26}$$

$$\frac{\partial e(s)}{\partial \theta} = kG(s)U(s) - k_o G(s)U_{bl}(s) = \frac{k}{k_o} k_o G(s)U_{bl}(s) = \frac{k}{k_o} z_m \tag{4.27}$$

where $G(s)U_{bl}(s)$ is the output z_m which is encoded by the reference model. By redefining the adaptation gain γ by lumping the constants together as a new constant $\gamma^{'} = \gamma \cdot \dfrac{k}{k_o}$, the final adaptation rule for the parameter θ will be as follows :

$$\dot{\theta} = -\gamma^{'} \cdot e \cdot z_m \tag{4.28}$$

4.4.1.2 Error Convergence and Stability Analysis

The presented adaptation rule, however, suffers from a zero model output i.e when $z_m = 0$. A zero model output results in an unchanging parameter θ even though the error between the model and the actual output has not converged to zero yet. This drives the system to instability, since encountered system parameter changes will not be compensated for, leaving the adaptation scheme of no use. Hence, another modification is proposed by [55] to make the adjustment mechanism to be dependent only on the actual output z and not to explicitly depend on the model output z_m.

$$\frac{\partial e(s)}{\partial \theta} = \frac{\theta}{\theta} kG(s)U_{1_{bl}}(s) = \frac{kG(s)U(s)}{\theta} = \frac{z}{\theta} \tag{4.29}$$

The modified adaptation law of the parameter θ will then be as follows.

$$\dot{\theta} = -\gamma \cdot e \cdot \frac{z}{\theta} \tag{4.30}$$

The parameter θ starts from a default value of 1 indicating that the output of the actual system has not deviated from the output of the encoded model. The presented adaptation law ensures that the parameter theta will never converge to zero, rather has values greater than it. A further arrangement, presented in [56], is performed to eradicate the problem of instability that may be caused with large inputs by using a normalization of the MIT rule as follows:

$$\dot{\theta} = -\frac{\gamma \cdot e \cdot \frac{\partial e}{\partial \theta}}{\alpha + (\frac{\partial e}{\partial \theta})^2} \tag{4.31}$$

$$\dot{\theta} = -\frac{\gamma \cdot e \cdot \frac{z}{\theta}}{\alpha + (\frac{z}{\theta})^2} \tag{4.32}$$

where α is a non-zero positive number that is introduced so as to guarantee that division by zero never happens when the value of $\frac{z}{\theta})^2$ is small.

4.4.2 Adaptive Attitude Loop

The baseline attitude controller developed in subsection 4.3.2 ensures attitude stabilization during nominal conditions where system parameters are certain and predictable. However, in the physical world, system parameters are uncertain causing unfavourable conditions. In this scenarios, the baseline controller fails to ensure convergence of attitude tracking error to zero. Hence an adaptive controller is

augmented to guide the baseline controller in these scenarios of operation. The adaptive attitude controller is formulated as an indirect model reference adaptive controller following the parametrization developed in [34].

Under the very basic assumption of the body reference frame aligning with the principal axis of the body and a neglection of the gyroscopic term for controller design purposes, the rotational subsystem can be recalled from Eqn: 3.21 as follows:

$$\dot{q} = \frac{1}{2} q \otimes \omega \tag{4.33}$$

$$\dot{\omega} = J^{-1}(\tau - \omega \times Jw) = J^{-1} \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} - \begin{bmatrix} \frac{I_z - I_y}{I_x} qr \\ \frac{I_x - I_z}{I_y} pr \\ \frac{I_y - I_x}{I_z} pq \end{bmatrix} \tag{4.34}$$

It is intuitive to note that there is a pure kinematic relationship in the equation describing the attitude propagation in the quad-rotor vehicle. Hence there is no uncertainty that appears in the quaternionic kinematic equations presented in Eqn: 4.33. On the contrary, the body rate dynamics presented in Eqn: 4.34. suffer from three main types of uncertainties listed below.

- The main physical parameter imposing uncertainty in to the rotational subsystem is the inertia tensor which can be described as a diagonal matrix comprising of Ix, Iy and Iz.
- The aerodynamic coefficients of thrust and moment as denoted by k_f and k_m are two other physical parameters which cause input uncertainty whose effect prevents the generated control torque from being as effective as it is intended to be.
- Disturbances are other unfavourable conditions that will have the vehicle accelerate in a certain undesired direction.

All the three main conditions have direct effect in making the rotary subsystem unstable posing threat to the overall navigation operation. The adaptive augmentation is expected to deal with these three types of uncertainties.

Parametrization of the three types of uncertainties inside the body rate dynamics results in the following reformulation.

$$\dot{\omega} = \Theta\Phi(\omega) + \Lambda\tau + \tau_d \tag{4.35}$$

$$\dot{\omega} = \begin{bmatrix} \theta_1 & 0 & 0 \\ 0 & \theta_2 & 0 \\ 0 & 0 & \theta_3 \end{bmatrix} \begin{bmatrix} -qr \\ -pr \\ -pq \end{bmatrix} + \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} + \begin{bmatrix} \tau_{dx} \\ \tau_{dy} \\ \tau_{dz} \end{bmatrix} \tag{4.36}$$

- $\Theta = \begin{bmatrix} \frac{I_z - I_y}{I_x} & 0 & 0 \\ 0 & \frac{I_x - I_z}{I_y} & 0 \\ 0 & 0 & \frac{I_y - I_x}{I_z} \end{bmatrix}$ denotes a lumped parametrization of the values of the inertia tensor.
- $\Phi(\omega) = \begin{bmatrix} -qr \\ -pr \\ -pq \end{bmatrix}$ is a regressor vector which represents known information about the body rate dynamics.
- $\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$ denotes the control effectiveness matrix of the vehicle which is parametrized by lumped values of the inertia tensor and the aerodynamic coefficients. Under nominal conditions, the design control effectiveness matrix is same as follows:

$$\text{That is : } \Lambda_D = {J_D}^{-1} = \begin{bmatrix} \frac{1}{I_x} & 0 & 0 \\ 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix}$$

where $[\quad]_D$ denotes the nominal, design-time parameter values.

- $\tau_d = \begin{bmatrix} \tau_{dx} \\ \tau_{dy} \\ \tau_{dz} \end{bmatrix}$ represents unknown disturbance torque applied on the vehicle which produces undesired effect on the angular acceleration.
- $\tau = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$ is the generated torque as a result of the action of the motors.

Matching uncertainties is also an important concept that should be noted in order to guaranteeing stability in MRAC. In Eqn: 4.35 , it can be noted that all mentioned uncertainties do lie in the image space of the control input, τ resulting a matched relationship.

4.4.2.1 The Reference Model

With the application of the baseline attitude control law, the body rate dynamics as presented in Eqn: 4.38 can be treated as a reference model that provides desired performance behaviours the actual plant has to follow.

$$\dot{\omega_M} = {J_D}^{-1}\tau_{bl} - {J_D}^{-1}\omega_{M\times}J\omega_M \tag{4.37}$$

Furthermore the parametrization presented in Eqn: 4.35 can be utilized to result in the following:

$$\dot{\omega_M} = \Theta_D\Phi(\omega) - \Lambda_D\tau_b \tag{4.38}$$

where $\dot{\omega_M}$ denotes the desired nonlinear angular rate dynamics of the vehicle.

However, the reference model described in 4.38 is inherently non-linear characterizing complex behaviours as compared to an analogous linearised model.

A standard Lyapunov based MRAC design, on the other hand, requires a linear reference model to decode the desired transient behaviour of a system. Therefore, following the parametrization in [34], a time dependent linear model is constructed which mimics the behaviour of the non-linear body rate dynamics of Eqn: 4.38 in a faithful way.

The approximate LTV model is developed having the architecture as in Eqn: 4.39 with its construction details further described in Appendix C.

$$\dot{\omega_m} = A_m(t)\omega_m + r \tag{4.39}$$

For simplicity, $A_m = A_m(t)$ for the analysis to follow.

The system matrix and the reference of 4.39 are found to be as follows.

$$A_m = -K_w I - \omega_{d\times}$$

$$r = -k_w\omega_d + \dot{\omega_d} - k_q \log {q_e}^+$$

The developed LTV model is further intended to be stable. Thus, the analysis is done using the theorem for linear systems' stability using Lyapunov equations [57], the proof of which is presented in Appendix D. The Lyapunov equation $A_m(t)P + PA_m(t) = -2K_wI$ is used where the matrix P is found to consistently positive definite having a value of an identity matrix, I, for all values of the time variant system matrix $A_m(t)$.

The construction of the custom reference model depicted by Eqn: 4.39 results in an open loop reference model. Closed loop reference models can be obtained by comprising information about tracking error, $e = \omega_m - \omega$. The closed loop construction is claimed to give improved transient dynamics in the design of adaptive controllers [58].

Incorporation of the tracking error in to the formulated reference model results in a closed loop reference model constructed as follows:

$$\dot{\omega_m} = A_m(t)\omega_m + r - K_e e \tag{4.40}$$

where k_e is a positive constant.

The custom reference model developed above is then used in the formulation of an indirect model reference control system for the attitude loop.

4.4.2.2 The Adaptation Law

The aim of the adaptive augmentation is to augment the baseline control law developed as Eqn: 4.17 in the presence of off-nominal conditions where the baseline control law is not guaranteed to provide stability. Hence the overall control law that is fed to the rotational subsystem to stabilize the attitude dynamics is as follows:

$$\tau = \tau_{bl} + \tau_{ad} \tag{4.41}$$

To design the adaptation controller, the body rate dynamics in Eqn: 4.34 is parameterized as follows incorporating the error between the linear reference model of Eqn: 4.39 and the non-linear reference model described in Eqn: 4.38 which is desired to be zero.

$$\dot{\omega} = \dot{\omega} + \dot{\omega}_m - \dot{\omega}_M \tag{4.42}$$

where,

- $\dot{\omega}$ is the rotational dynamics of the vehicle parametrized as in Eqn: 4.35
- $\dot{\omega}_m$ encodes the dynamics of the approximate LTV reference model of Eqn: 4.39
- $\dot{\omega}_M$ describes the dynamical behaviour of the original non-linear reference model presented in Eqn: 4.38.

With the application of the oveall control law, Eqn: 4.42 is thus reformulated as follows:

$$\dot{\omega} = \Theta\Phi + \Lambda(\tau_{bl} + \tau_{ad}) + \tau_d - [K_w I + \omega_{d\times}]\omega + r - \Theta_D\Phi - \Lambda_D\tau_{bl} \tag{4.43}$$

$$\dot{\omega} = (\Theta - \Theta_D)\Phi + \Lambda\tau_{bl} + \Lambda\tau_{ad} + \tau_d - [K_w I + \omega_{d\times}]\omega - \Lambda_D\tau_{bl} + r$$

$$\dot{\omega} = (\Theta - \Theta_D)\Phi + \Lambda\tau_{bl} + \Lambda[\tau_{ad} + (I - \Lambda^{-1}\Lambda_D)\tau_{bl}] + \tau_d + r - [K_w I + \omega_{d\times}]\omega$$

It is intended that the attitude dynamics is free from any uncertainties so that no system perturbation would occur with the uncertain behaviour of the parametrized physical parameters Θ, Λ and τ_d. Thus, it is desired that the adaptive law τ_{ad} removes the mentioned uncertainties from the body rate dynamics.

Hence, a choice of the control law as in Eqn: 4.44 ensures the removal of the uncertain parameters with the best estimates of Θ, Λ and τ_d.

$$\tau_{ad} = \widehat{\Lambda}^{-1}[-(\widehat{\Theta} - \Theta_D)\Phi - \widehat{\tau_d}] - (I - \widehat{\Lambda}^{-1}{J_D}^{-1})\tau_{bl} \tag{4.44}$$

where the parameters denoted as $\widehat{(.)}$ are the estimated values of each respective parameters, τ_{bl} is the baseline attitude controller developed as Eqn: 4.17, Θ_D and J_D denote the nominal design values.

The parameters presented as $\widehat{(.)}$ are updated using parameter update laws. In order to derive the adaptation scheme, the angular rate error dynamics has to be formulated in the presence of off-nominal conditions. The model tracking error is defined as:

$$e = \omega_m - \omega$$

$$\dot{e} = \dot{\omega}_m - \dot{\omega}$$

With the neglection of the closed loop reference augmentation developed as Eqn: 4.40, the rate of change of the tracking error is as follows :

$$\dot{e} = A_m\omega_m + r - \dot{\omega}$$

Making use the reformulated body rate dynamics developed as Eqn: 4.43, the angular rate dynamics can be found to be as follows:

$$\dot{e} = A_m\omega_m + r - \Theta\Phi - \Lambda(\tau_{bl} + \tau_{ad}) - \tau_d - A_m\omega + r + \Theta_D\Phi + \Lambda_D\tau_{bl}$$

$$\dot{e} = A_m(\omega_m - \omega) + (\Theta_D - \widehat{\Theta})\Phi + (\Lambda_D - \widehat{\Lambda})(\tau_{bl} + \tau_{ad}) - \widehat{\tau_d}$$

Then, the angular rate tracking dynamics is found to be:

$$\dot{e} = A_m(e) + \widetilde{\Theta}\Phi + \widetilde{\Lambda}(\tau) + \widetilde{\tau_d} \tag{4.45}$$

$$\text{where,} \qquad e = \omega_m - \omega$$

$$\widetilde{\Theta} \coloneqq (\Theta_D - \widehat{\Theta})$$

$$\widetilde{\Lambda} \coloneqq (\Lambda_D - \widehat{\Lambda}), \quad \text{and}$$

$$\widetilde{\tau_d} \coloneqq (\tau_{dD} - \widehat{\tau_d}), \text{ with } \tau_{dD} = 0$$

4.4.2.3 Parameter Adjustment Rule

The rate at which the parameters, defined in Eqn: 4.35, are updated is specified by the parameter adjustment laws. These laws are derived by utilizing Lyapunov's stability theorem [59]. By making use of the tracking and estimation errors defined in Eqn: 4.45, a positive definite Lyapunov function can be defined as follows:

$$V = e^T Pe + Tr(\widetilde{\Lambda}{\Gamma_\Lambda}^{-1}\widetilde{\Lambda}^T + \widetilde{\Theta}{\Gamma_\theta}^{-1}\widetilde{\Theta}^T + {\gamma_{\tau_d}}^{-1}\widetilde{\tau_d}\widetilde{\tau_d}^T) \tag{4.46}$$

where $P = I$ and Γ_Λ, Γ_Θ and γ_{τ_d} are positive adaptation gains which act as weights for the parameter adaptation rule.

The derivative of the selected Lyapunov function is then calculated as the following :

$$\dot{V} = 2e^T P\dot{e} + 2(\widetilde{\Lambda}{\Gamma_\Lambda}^{-1}\dot{\widetilde{\Lambda}}^T + \widetilde{\Theta}{\Gamma_\theta}^{-1}\dot{\widetilde{\Theta}}^T + {\gamma_{\tau_d}}^{-1}\widetilde{\tau}_d\dot{\widetilde{\tau}}_d^T) \tag{4.47}$$

Taking the expression for the angular velocity error from Eqn: 4.45, the derivative of Lyapunov function becomes:

$$\dot{V} = 2e^T P[A_m(e) + \widetilde{\Theta}\Phi + \widetilde{\Lambda}(\tau) + \widetilde{\tau_d}] + 2(\widetilde{\Lambda}{\Gamma_\Lambda}^{-1}\dot{\widetilde{\Lambda}}^T + \widetilde{\Theta}{\Gamma_\theta}^{-1}\dot{\widetilde{\Theta}}^T + {\gamma_{\tau_d}}^{-1}\widetilde{\tau}_d\dot{\widetilde{\tau}}_d^T)$$

$$\dot{V} = 2e^T PA_m(e) + 2e^T P\widetilde{\Theta}\Phi + 2e^T P\widetilde{\Lambda}(\tau) + 2e^T P\widetilde{\tau_d} + 2(\widetilde{\Lambda}{\Gamma_\Lambda}^{-1}\dot{\widetilde{\Lambda}}^T + \widetilde{\Theta}{\Gamma_\theta}^{-1}\dot{\widetilde{\Theta}}^T + {\gamma_{\tau_d}}^{-1}\widetilde{\tau}_d\dot{\widetilde{\tau}}_d^T)$$

But,

$$2e^T PA_m(e) = e^T PA_m(e) + e^T PA_m(e) = e^T PA_m(e) + e^T A_m^T P(e) = e^T(PA_m + A_m^T P)(e)$$

However, the following can be recalled from the stability analysis done for the formulated linear time variant angular rate dynamics.

$$PA_m + A_m^T P = -2K_w$$
$$2e^T PA_m(e) = -2e^T k_w e$$

Thus the expression for the derivative of the Lyapunov function becomes as follows:

$$\dot{V} = -2e^T k_w e + 2e^T P\widetilde{\Theta}\Phi + 2e^T P\widetilde{\Lambda}(\tau) + 2e^T P\widetilde{\tau_d} + 2(\widetilde{\Lambda}{\Gamma_\Lambda}^{-1}\dot{\widetilde{\Lambda}}^T + \widetilde{\Theta}{\Gamma_\theta}^{-1}\dot{\widetilde{\Theta}}^T + {\gamma_{\tau_d}}^{-1}\widetilde{\tau}_d\dot{\widetilde{\tau}}_d^T)$$

From the expression, it can be noted that the term $-2e^T k_w e$ is always negative. However, the sign of the other terms is uncertain. Hence, in order for the Lyapunov function to be negative semi-infinite, it is essential to eradicate these terms.

The standard MRAC parameter update laws can be adopted so as to ensure the defined Lyapunov function becomes negative definite using Lyapunov stability analysis. Thus, the parameter update laws are defined as the following:

$$\dot{\widehat{\Theta}}^T = -\Gamma_\theta \Phi e^T P$$
$$\dot{\widehat{\Lambda}}^T = -\Gamma_\Lambda \tau e^T P$$
$$\dot{\widehat{\tau}}_d^T = -\gamma_\tau e^T P$$

With the chosen parameter update laws, $\dot{V}$ is not found to be strictly negative since the result shows $\dot{V} \leq 0$. Hence, even though the Lyapunov theorem ensures that the error e is bounded, asymptotic stability cannot be ensured. Thus, by a further analysis using Barbalat's Lemma and LaSallie's Theorem as shown in [59] with the defined parameter adjustment laws, it is guaranteed that the update law defined in Eqn: 4.44 causes the angular velocity tracking error to asymptotically converge to zero.

Chapter 5

Optimal Trajectory Generation

In order to complete a certain flight mission, aerial vehicles have to be provided with certain way points through which they are desired to pass through. This process is done by a path planner algorithm which analyses the flight mission and prepares feasible way points that are needed for the vehicle's navigation plan.

Most path planning problems usually generate paths which have sharp turns and rough edges which require abrupt system changes for any dynamical system to follow. In the case of robotic vehicles like the quad-rotor, this abrupt change causes a lot of problems. Following a path with sharp turns require a certain quad-rotor to slow down at each way point before moving to the next path segment. That, in turn, produces undesired unnatural movements. In some scenarios, this on and off type of movement may not even be feasible kinematically.

In order to provide a solution for the problem a navigating quad-rotor may encounter, different solutions have been provided over the years. Cubic splines, bezier curves, B-splines and many more techniques have been employed to tackle the problem of sharp paths. This process of obtaining smooth trajectories from sharp edged paths between way-points is named as trajectory generation. To look at the type of trajectory generation which is best suited to the problem of quad-rotors, it is important to look at the order of the problem.

5.1 Differential Flatness of the Quad-rotor

A system is said to be differentially-flat, if there exists a set of outputs such that the system states and the inputs can be expressed in terms of selected flat outputs and a finite number of their derivatives. Mathematically decoding this relationship as in [60]: if

a certain system has states $x \epsilon R_N$, and inputs $u \epsilon R_M$, then the system is said to be flat if we can find outputs $y \quad \epsilon \quad R_M$ of the following form :

$$y = h(x, u, \dot{u}, ..., u^{(r)}) \tag{5.1}$$

Such that:

$$x = \phi(y, \dot{y}, ..., y^{(q)} \tag{5.2}$$

$$u = \alpha(y, \dot{y}, ..., y^{(q)}) \tag{5.3}$$

An optimization formulation for the presented problem is concerned with obtaining a flight trajectory that satisfies way-point constraints at specific scheduled flight timetable. Since the quad-rotor can perform motion with six degrees of freedom, the reference flight trajectory could contain six components. However, due to the under actuated property of the quad-rotor, only the longitudinal, lateral and altitude displacements represented by $[x, y, z]$ and the yaw angle, ψ can be directly controlled by the application of inputs. Whereas, the two remaining rotational degree of freedoms: the pitch and the roll movements, are calculated in a systemic manner from a cascaded higher loop position controller.

Taking this in to account, the standard common model of the quad-rotor is shown to be differentially flat, when the position in three dimensions,$[x, y, z]$ and the heading angle, ψ, serve as the flat outputs. In other words, both the states describing the quad-rotor and the inputs of it can be written as algebraic functions of four carefully selected flat outputs and their derivatives.

The differential flatness property of the quad-rotor is useful in scenarios where explicit trajectory generation is required. Due to the fact that all the behaviour of the differentially flat quad-rotor can be solely determined by the selected flat outputs, one can plan trajectories in the output space and then map these trajectories in to the appropriate inputs. This facilitates the generation of trajectories since any smooth trajectory (with bounded derivatives) in the space of flat outputs can be followed by the under actuated quad-rotor [61]. Being equipped with this property, a possible optimal trajectory generation process is explained in the next section.

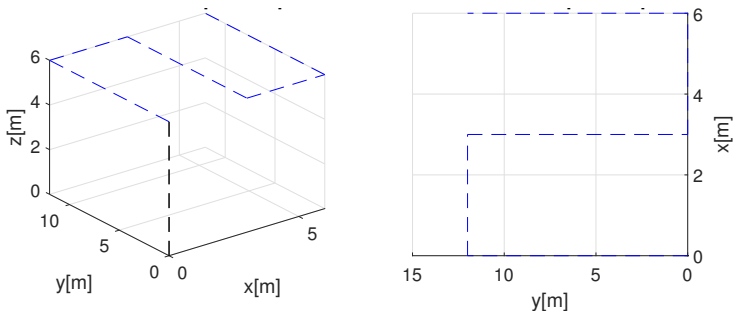
5.2 Minimum Snap Trajectory Generation

5.2.1 Problem Formulation

Given a three dimensional environment, we want to be able to specify start and end points and have the quad-rotor plan feasible trajectory in that environment. A more specified problem comprises of intermediate way-points that the quad-rotor must pass through.

The application area of this paper involves surveillance of water bodies. When talking about surveillance, the notion of area coverage is inevitable. This type of path planning problem is named as coverage path planning. A simple , but yet useful type of trajectory which enables area coverage is the sweep trajectory.

Given a set of way-points, a trivial solution to finding trajectory that satisfies the position constraints is one that directly interpolates between way-points using straight lines. A sample set of way-points and their trivial connection is depicted in the following figure. However this trajectory is inefficient because it has infinite curvature at the points which requires the quad-rotor to come to a stop at each point [61]. This type of path causes unnatural and infeasible trajectory for the quad-rotor to follow as any complex dynamics system. This underlines the need for a smooth trajectory generation.



In order to formulate the problem, it is important to note the following three points:

- The start and goal position along with specified way-point that pose constraint on the quad-rotors position.
- The smoothness criterion that the problem seeks: The quad-rotor system can not follow arbitrary trajectories. Hence it is mandatory that the trajectory that the quad-rotor is expected to follow have to be smooth.
- The order of the system is also another thing to note since the order of the system specifies how the inputs to the system are translated in to the respective outputs.

Recalling the quad-rotor dynamics, analysis can be made towards identifying the order of the overall system.

$$\frac{d}{dt}\begin{bmatrix} \mathrm{p} \\ \dot{p} \\ \mathrm{q} \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{p} \\ \mathrm{q}\otimes\dfrac{F_{thrust}}{m} \otimes q^T - \overrightarrow{g} \\ \dfrac{1}{2}\mathrm{q}\otimes\omega \\ J^{-1}[\overrightarrow{\tau} - \omega\times \mathrm{J}\omega] \end{bmatrix}. \tag{5.4}$$

The quad-rotor modelling as developed in Chapter 3 consists of two separate loops. First the outer position control loop that specifies U and the inner attitude control loop which determines the control torques $\tau_x, \quad \tau_y, and \quad \tau_z$. The inner loop is expected to ensure attitude stabilization in order for the position controller to guarantee stability. The linear accelerations depend on U in an explicit manner. Even though it is not explicit, the linear acceleration is also dependent on the control inputs $\tau_x, \quad \tau_y, and \quad \tau_z$. But this dependency is not direct, rather it is through the second derivative of the rotation matrix which aligns the body reference frame to the inertial reference frame. All in all , the consideration of this implicit dependency make the overall system of the quad-rotor a fourth order system.

An optimization problem can then be formulated by minimizing the snap[1] which essentially tries to minimize the fourth order of position integrated over the time history. Since the fourth order of position depends on all the control inputs ($U, \quad \tau_x, \quad \tau_y, and \quad \tau_z$), minimizing snap indirectly means minimizing the control inputs demanded to complete a certain trajectory. Thus to specify trajectories for the quad-rotor system , we need trajectories which are differentiable to atleast their third order. The overall problem is constructed as follows:

$$x_*(t) = argmin \int_0^T {x^{(iv)}}^2 dt \tag{5.5}$$

A manual solution to this optimization is solved in [62], whereas the optimization problem can be automatically solved using the quadratic programming function using MATLAB.

The problem is constructed as a multi-segment constrained optimization problem with constraints posed upto the sixth derivative of position between way-points. The constraints force the overall optimization problem to allow smooth transitions between successive path segments. Figure 5.1 shows a description of the constraints for a minimum snap constrained optimization problem by taking a sample of 4 way-points excluding an initial starting point.

[1]Snap - the name for the signal representing the forth order of position.

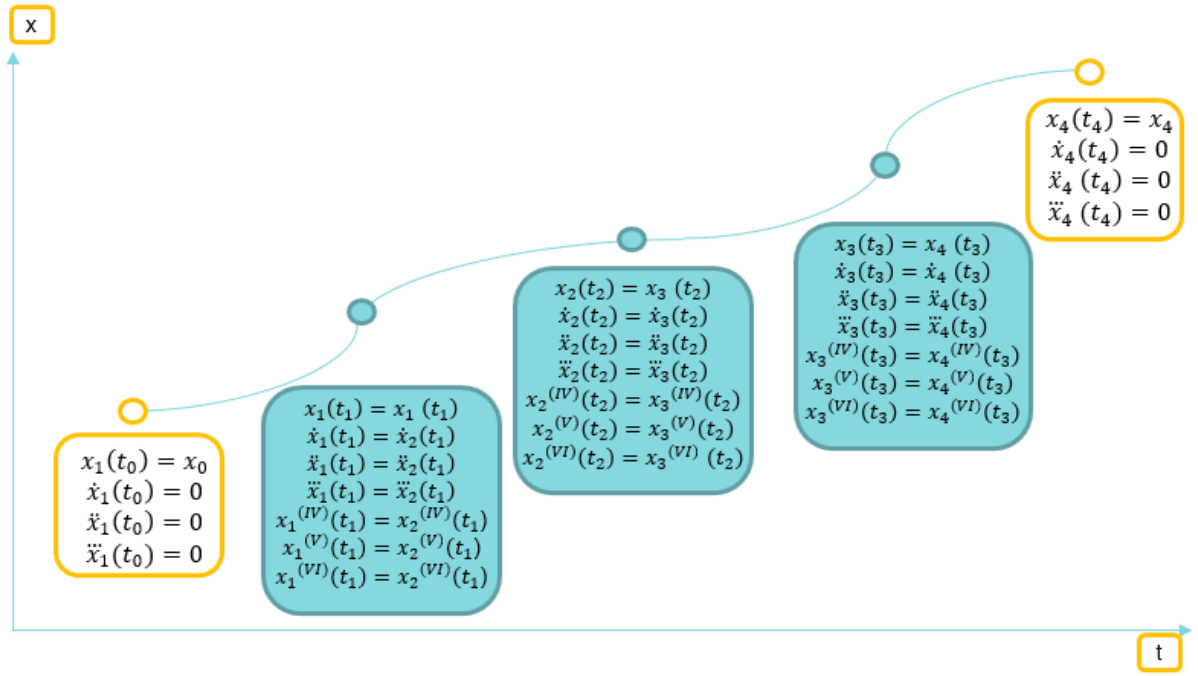


Figure 5.1: Constraints for the Minimum Snap Optimization Problem

A smooth sweep trajectory obtained as the solution of the minimum snap problem is used for further testing of the developed controller. Smooth trajectories generated for the four flat outputs and their derivatives upto the jerk signal are shown in each of testing scenarios presented in Chapter 7.

Chapter 6

Image Recognition

6.1 Overview

Image recognition is a computer vision technique that equips automated systems with the ability to interpret objects in an arbitrary three dimensional environment. Humans use their eyes and their brains to see and understand the three dimensional environment around them. The problem is often constructed as an image classification problem [63]. The science of computer vision aims to give a similar capability to computing machines. The process goes by developing methods which are capable to mimic the human visual system.

Over the last decade, computer vision along with machine learning has provided with various state of the art image based inventions. There has been a tremendous transformation of vision based technology from simple sensor relied devices to intelligent computing systems having the ability to percept their environment [64].

Previous attempts towards solving computer vision problems relied in traditional machine learning architectures [65]. With traditional approaches like the support vector machines (SVMs), feature extraction is a mandatory and most import step toward the problem. These features are hand-crafted features that are chosen by the machine learning designer. This factor made the success of traditional machine learning structures to be dependent on the engineered features that are fed to the network. However, recently deep learning techniques have offered a compelling alternative to that by offering the capability to automatically learn problem-specific features [63]. The problem nowadays has become to just understand the kind of the suitable deep network for a given problem and leave the feature extraction process to the deep network itself.

6.1.1 CNNs

One of the mostly employed deep networks in the arena of computer vision is the convolutional neural networks (CNNs). It is a class of artificial neural network which is mostly employed for visual recognition.

The concept of convolutional Neural Network (CNN) is not new in the field of artificial intelligence. Even though, CNNs are not new to the research community, it was nearly impossible to scale these networks to production level. This has been mainly due to memory and hardware constraints that were available at the time. Another bottleneck for CNNs to work well has also been it's data intensive property. Thus, due to the unavailability of large amounts of training data over the years, CNNs were nearly out of practice. However, with increase in computational power thanks to wide availability of graphical processing units, and the formulation of large scale datasets like the ImageNet dataset [66], it became possible to train larger, more complex models. The popular AlexNet model was the first model trained on a large scale data. This basically kick-started the usage of deep networks in the computer vision arena. Since then high fidelity models have been developed providing a nearly human like vision performance.

The main building block of the CNNs is the convolution operation. A convolution operation basically maps an input to an output using a specifically chosen filter and a sliding window. Simple vertical and horizontal edge detection steps of the convolution operation are the main building block of complex models able to extract high level and complex features.

6.2 Problem Formulation

The application of this thesis is concerned with surveillance of water bodies with the aim of the detection of the hyacinth aquatic weed. The image recognition problem thus have to be dealt with using deep leaning. A low-level binary image classification problem requires two separate dataset with class labels 1 and 0 with the former being the desired class and the later being the discriminated class. So analysis have to be made in order to formulate these classes during the data collection and preparation stage.

6.2.1 Dataset Collection

The data preparation stage of the problem requires the preparation of classes that have to be discriminated by the recognizer model to be built. According to [67], it is noted

that that the Water Primrose and the Pennywort aquatic plants are the most probable species of aquatic weeds that have great structural similarity with the water hyacinth aquatic plant with nearly similar habitats. Furthermore, the recognizer to be built has to discriminate a plain water surface with no aquatic weed even though, other activities are being done in the water surface. This facilitate the formulation of the problem as a multi-class image classification problem with four different classes being defined. The four classes are labelled as 'Water Hyacinth', 'No Invasive Aquatic Weed', 'Pennywort' and 'Water Primrose'.

Since there appears to be no preprepared dataset, the first step is to prepare a custom dataset comprising of the chosen image classes. Images were downloaded from search engines like Google, Yahoo and Bing. After downloading the images, the following processing stages were performed.

- Duplicated images are removed since duplicates are a major source of bias for any deep learning algorithm to learn from.
- In the downloaded images, images including watermarks and descriptors at any of their edges were plenty. This defects are corrected by cropping and resizing the image as long as the overall image quality and information is not destroyed. If not images are removed from the overall dataset.
- Images are resized so as to fit the input size of the network the images will be trained on.

Following the above three data preparation steps, the dataset was build comprising of 500 images for each of the four classes. A highlight of the class of images comprised in the dataset are illustrated in Figure 6.1.

6.2.2 Training a Deep Neural Network

Following the steps mentioned in subsection 6.2.1, the prepared dataset is made to comprise of 1000 images per class. This is a pretty small number with the consideration of the data intensive property of neural networks. The performance of a deep convolutional neural network is greatly dependent on large scale of data. Having these much small set of images, it is practically impossible to expect an acceptable accuracy level without over-fitting by training a convolutional neural network from scratch. A possible solution towards tackling the encountered data scarcity problem is the usage of pre-trained models using a concept called transfer learning.

(a) Water Hyacinth

(b) No Invasive Aquatic Weed

(c) Pennywort

(d) Water Primrose

Figure 6.1: Dataset of Images

6.2.2.1 Transfer Learning

While making decisions, it is intuitive for humans to use knowledge from their previous expertise in their current task of decision making. This makes the human knowledge clustered through years of experience of sharing. However, training a model from scratch leaves a certain machine learning model with no shared prior knowledge of the problem at hand.

Transfer Learning is inspired by the knowledge sharing capability of humans. It is a method of leveraging state of the art models which are trained on massive datasets for specific use-cases without the hassle of large dataset preparation and the need for large computing power for training [68]. The knowledge sharing paradigm of transfer learning is best illustrated in Figure 6.2

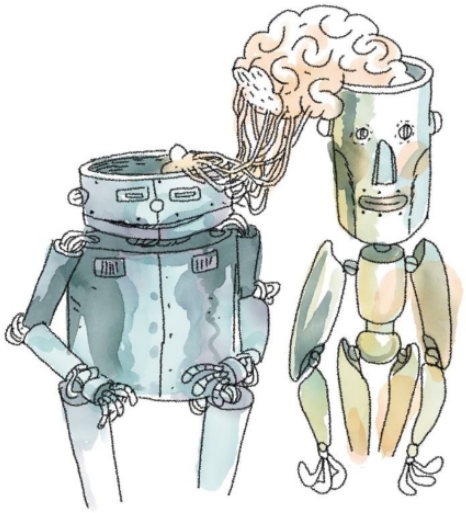


Figure 6.2: Illustration of Transfer Learning

6.2.2.2 Pre-trained Models

Along with the increase of processing power, a massive set of data has been collected over the years, making the computer vision problems yield accurate and production level models.

The ImageNet dataset has been collected in a global effort over the years with the aim of advancing computer vision and deep learning research. Nowadays he dataset contains more than 14 million images, more than 21 thousand classes (synsets) and more than 1 million images with bounding box annotations.

Utilizing this massively collected dataset, there are many state-of-the-art models whose performance has been satisfactory. The newest residual networks [69] trained on ImageNet

has achieved superhuman performance at recognising objects. Several other models have also achieved good accuracy level particularly in image classification task.

Transfer learning can be used to use the knowledge learnt by this state of the art models and adopt it to a custom use-case. There are various scenarios of transfer learning among which the fine tuning technique is implemented for this problem at hand [70]. From the state of the art models, the VGG-16 model is adopted for the recognition task in this paper.

VGG-16 : The VGG-16 [1] is one of the pre-trained models trained on the ImageNet dataset [69]. The overall VGG-6 architecture is shown in Figure 6.3. The number, 16, in VGG16 denotes that the architecture is comprised of 16 layers. The network as a whole approximately consists of about 14.7 M parameters.

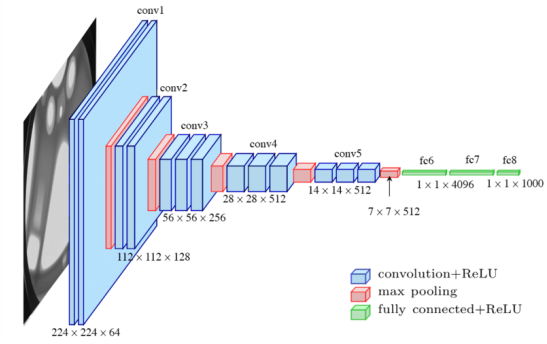


Figure 6.3: The VGG-16 Network Architecture

6.2.2.3 Feature Extraction Using VGG-16

There are different scenarios of transfer learning. One of them is to adopt the weights of the pre-trained model and have it do predictions on a custom dataset. In this scenario, there is no training step required. However,this method is suitable when the labels of the target dataset of the problem at hand are one among the 1000 classes the VGG-16 model was trained on. None of the four categories of the image dataset defined in this paper are part of the categories of the ImageNet dataset. So adopting the model without further training will result in an unacceptable accuracy level.

Another method on which the implementation of this paper is based on is feature extraction. In the method of feature extraction, earlier parameters of the pre-trained VGG-16 model are adopted to be used to extract features of images in the target dataset. Later on, final output layers are added on top of the model whose parameters are left to

[1]VGG-16 is a convolutional neural network architecture which was proposed by Oxford's renowned Visual Geometry Group for the large scale visual recognition challenge in 2014.

be learnt in the training process on the target dataset. In this paper, this technique have been implemented by freezing the weights of earlier layers and training parameters at the output layer. The trainable parameters of the model that were trained in this paper's implementation are 100,356 parameters among the 14,714,688 parameters of the original VGG-16 model.

6.2.2.4 Data Augmentation

When data scarcity problem persists, one of the counter ways in deep learning is using data augmentation. The technique of data augmentation acts in a way that tweaks the nature of original training images and present a whole new combination of input for the neural network to be faced with data having high variance. In the broadest sense, there are two types of how augmentation can be applied to datasets: feature-space augmentation and input-space space augmentation. As for the implementation in this paper, input space augmentation have been applied which embeds variations in the input images in contrast its counterpart which tries to alter extracted features of an input data.

There are various techniques that can be utilized when implementing data augmentation. Some of the simple transformations applied to the image are; geometric transformations such as flipping, rotation, translation, cropping, scaling, and color space transformations such as color casting, varying brightness, and noise injection.

The techniques, however, are much reliant on specific types of datasets and all cannot be applied to all types of dataset with out consideration. This is in a way that some of the techniques do make no significant changes and some do result in an infeasible type of data not compelling to the specific use-case. Taking the resulted changes in to consideration, only techniques that ought to bring significant variation in the dataset are selected and applied to the training process. Three types of transformations including random rotations, shearing effect and brightness variation have been applied during the training phase. In order to implement this, the image data generator method from the Keras[2] API is used, which basically renders transformed images of the original training dataset whilst the training process.

[2]Keras is a deep learning API written in Python developed with a focus on enabling fast experimentation in the arena of deep learning.

Chapter 7

Implementation and Simulation Results

7.1 Overall Set-up

The overall built Simulink model comprising of the controller design and the quadrotor modelling is shown in Figure 7.1. The plant parameters used for simulation are adopted from [71] and are listed in Appendix A.

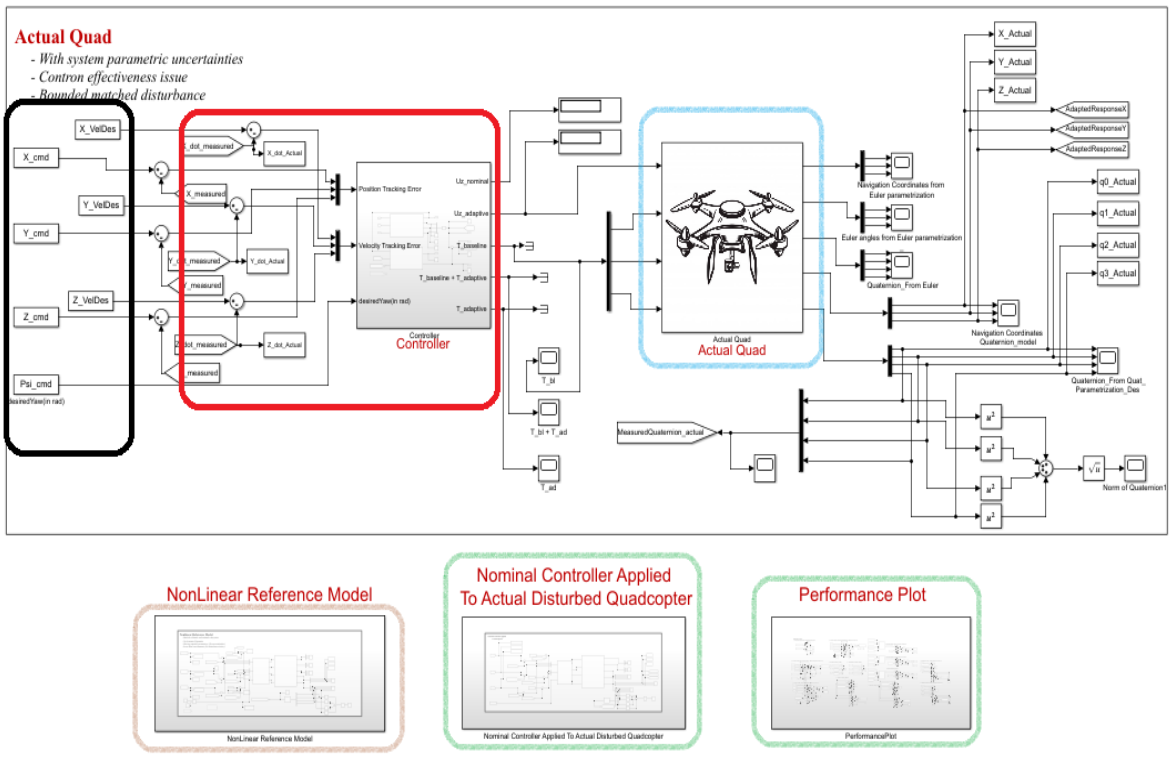


Figure 7.1: Overall Built Simulink Model

The overall design comprises of the following subsystems:

- The trajectory generator prepares feasible reference trajectories and then feeds it to the controller as the desired reference point.
- The reference model is a representation of a model that mimics the desired transient behaviour of the overall dynamic system. The reference model is controlled using the developed baseline controllers.
- Actual quad-rotor - This is the modelling of a quad-rotor possessing selected uncertain behaviours of a real world navigating quad-rotor. Furthermore, the quad-rotor is assumed to be in an environment where operating conditions are uncertain.
- Another block represents the behaviour of the baseline controller being applied to the uncertain quad-rotor. This block mimics how the system may have behaved if only the nominal controller has been applied to the actual quad-rotor operating in uncertain conditions.
- The last block in the Simulink model comprises of performance analysis plots that compare the performance of the developed nominal and adaptive controller.

For the purpose of controller design, gain acquisition has been done using the particle swarm optimization technique. Twelve controller gains were tuned for the baseline controller being elements of the four diagonal gain matrices presented in the control design. Setting up the PSO optimizer have been done by choosing the optimization function presented in Eqn: 7.1 The evaluation criteria proposed to monitor the performances of both the controllers is also the computation of integral time absolute error (ITAE) presented as follows.

$$J = \int_0^t t|e|dt \tag{7.1}$$

where $e = e_p + e_v + e_q$

e_p represents error in position tracking.

e_v represents error in linear velocity tracking.

e_q represents error in attitude tracking.

7.2 Trajectory Tracking

Prepared minimum snap trajectories are fed to the quad-rotor model for each scenarios where testing was done. Trajectories are prepared to be smooth up to their their third derivatives so as to ensure the quad-rotor's states will not abruptly change while in operation. The prepared trajectories are shown in each of the tracking tests performed.

Furthermore, to show the effectiveness of the controller complex manuevers are injected in to the quad-rotor system. Tested complex manuevers are the bow-tie and the helical trajectory which, in many literatures are presented as test signals to test the overall tracking performance of controllers. In all scenarios of testing, the quad-rotor's initial starting position is taken as the origin.

In the trajectory tracking tests, the main purpose is to show the effectiveness of both the baseline and adaptive controllers to take the quad-rotor to its target position via the specified trajectory. In the analysis, position and attitude tracking are examined. Furthermore, in each of the tests, the input commanded from the controller in order to perform tracking of the given reference signal is presented. While the controller is being tested with imposed disturbances, the time to which to which the whole system needs for recovery along with the control input casted out from the controller is analysed.

Furthermore, in the test plots presented, the signals labelled as measured are signals which are found from sensor measurements either in an explicit or implicit manner. Hence, if no implicit relationship is found, fed back signals are found by performing algebraic operations on explicitly measured signals which can be obtained directly from measuring devices.

7.2.1 Bow-tie Shaped Trajectory

Trajectory Preparation

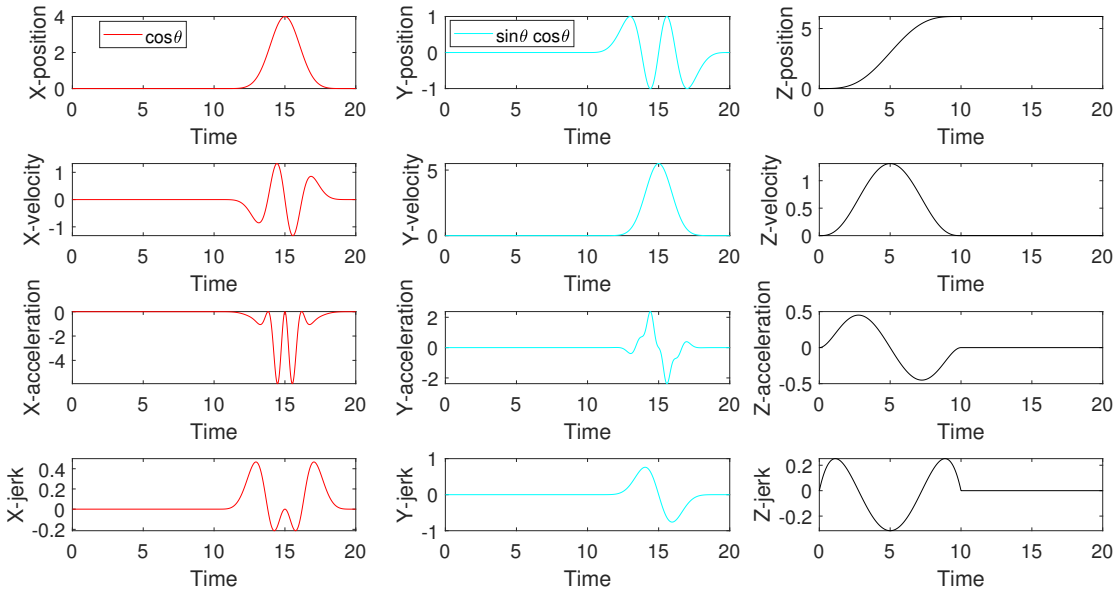


Figure 7.2: Minimum Snap Bow-tie Trajectory

Tracking Performance

The first scenario of trajectory tracking testing is the bow-tie trajectory where an initial starting position of the origin is taken. The operation is assumed to done in a nominal condition where no undesired environmental conditions occur. The structure of the injected desired position reference is given as follows

$$p_d = \begin{cases} \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix}, \forall t < 10, \\ \begin{bmatrix} r\cos\theta \\ r\sin\theta\cos\theta \\ 6 \end{bmatrix}, \forall t \geq 10, \end{cases}$$

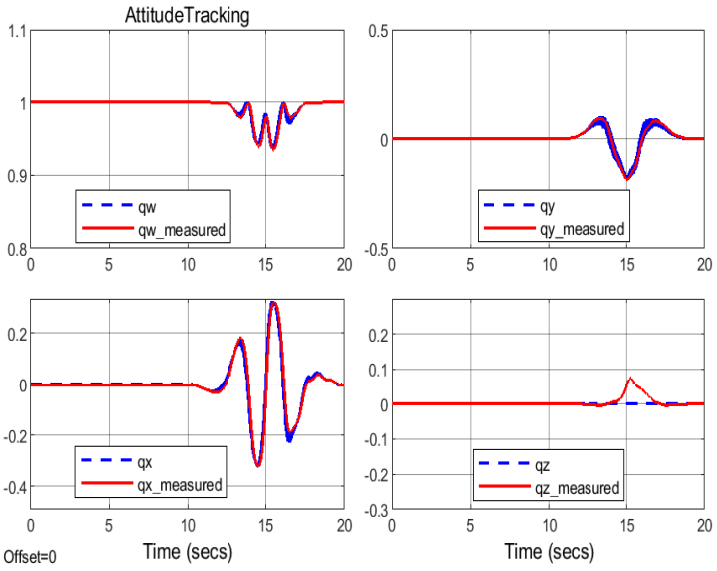
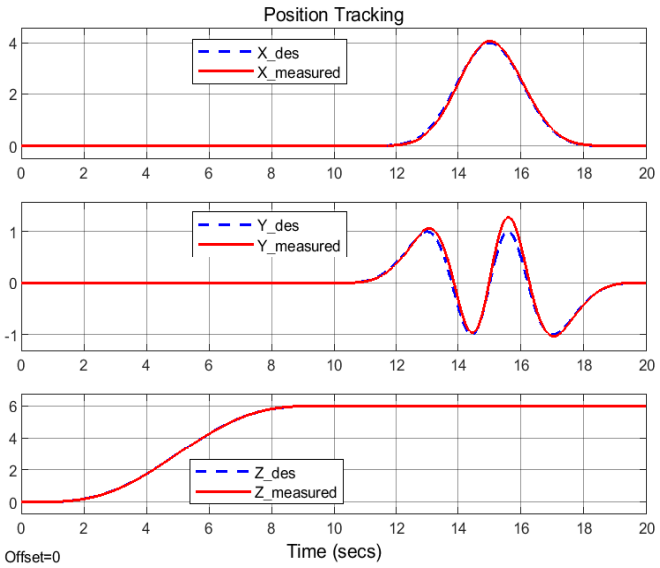


Figure 7.3: Attitude Tracking

Figure 7.4 shows that the quad-rotor's position very smoothly converges to its target configuration. With the commanded attitude quaternion computed to make the thrust



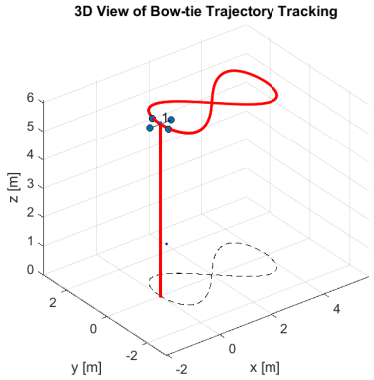


Figure 7.4: Position Tracking

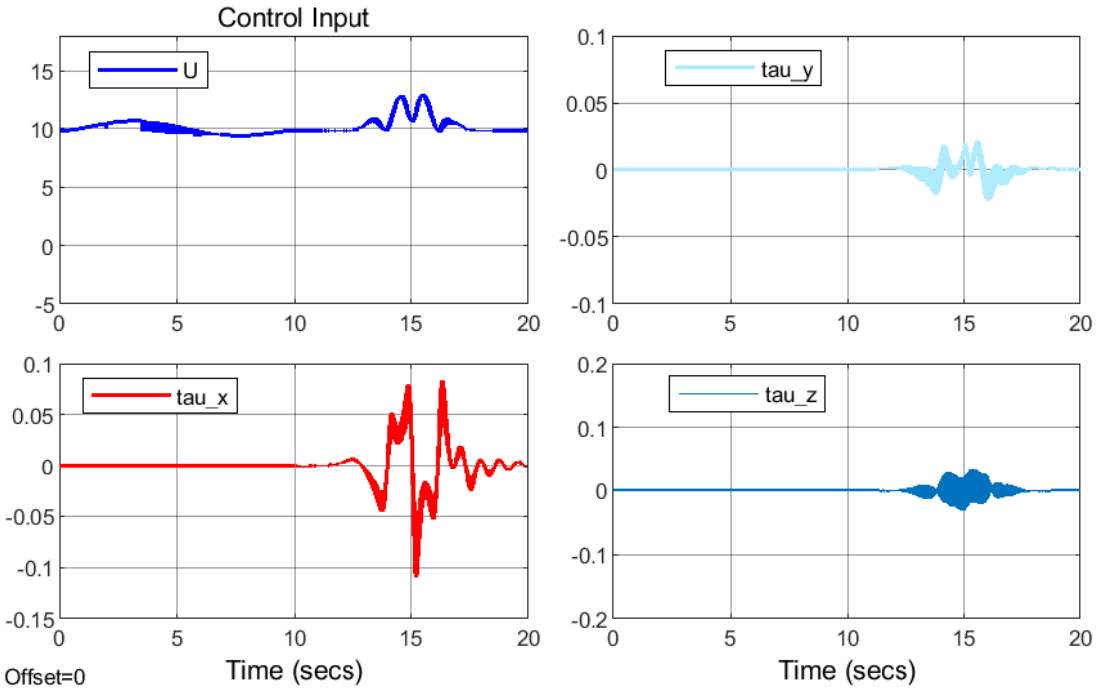


Figure 7.5: Control Inputs for Bow-tie Trajectory

force coincide with the desired force, Figure 7.3, shows how the quad-rotor vehicle tracks it. Eventhough, the quad-rotor exhibits considerable amount of attitudinal changes while performing this trajectory, the controller is able to ensure tracking. Furthermore, it can be seen that tracking is ensured utilising a fairly smooth the control energy as presented in Figure 7.5

7.2.2 Helical Trajectory

Trajectory Preparation

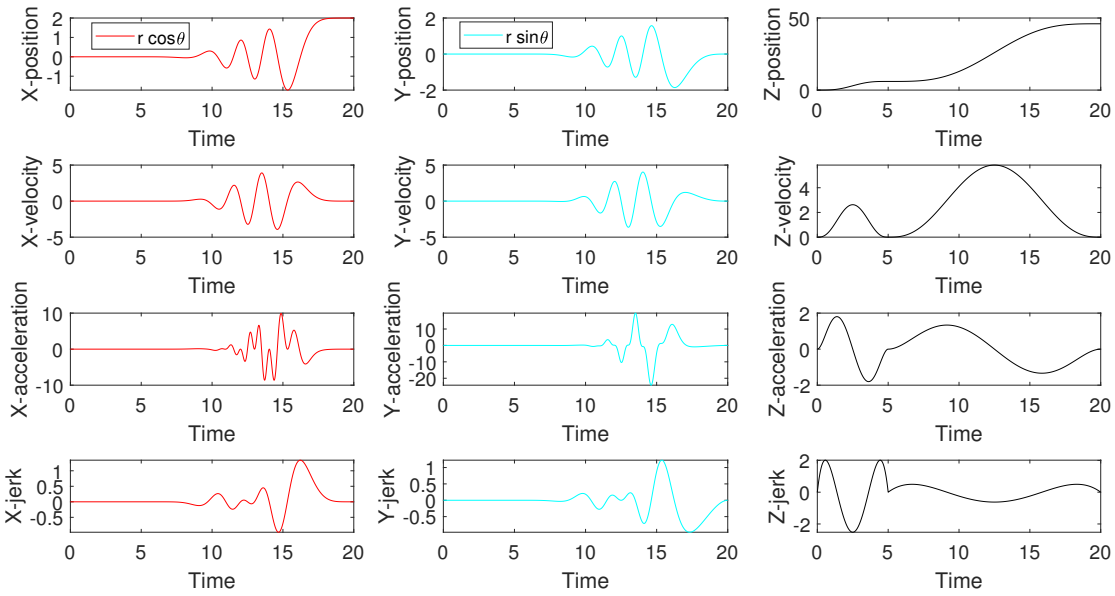


Figure 7.6: Minimum Snap Helical Trajectory

Tracking Performance

The helical trajectory is another test trajectory performed which demands a complex manuever from the quad-rotor. Tracking of he desired position reference signal of the following structure is analysed where nominal operating conditions are assumed.

$$p_d = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 6 \end{bmatrix}, \forall t < 5, \\ \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ 6 \end{bmatrix}, \forall t \geq 5, \end{bmatrix}$$

where $r = 0, \quad fort < 5$ and $r\epsilon[0,2], \quad fort \geq 5$

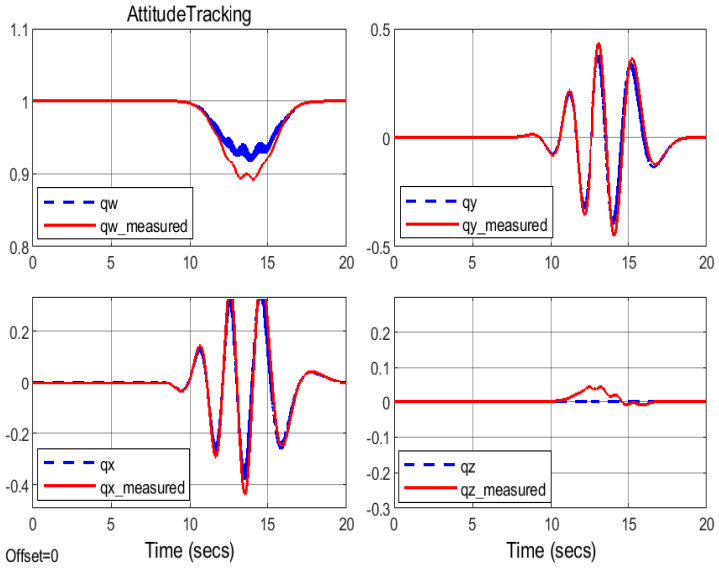


Figure 7.7: Attitude Tracking

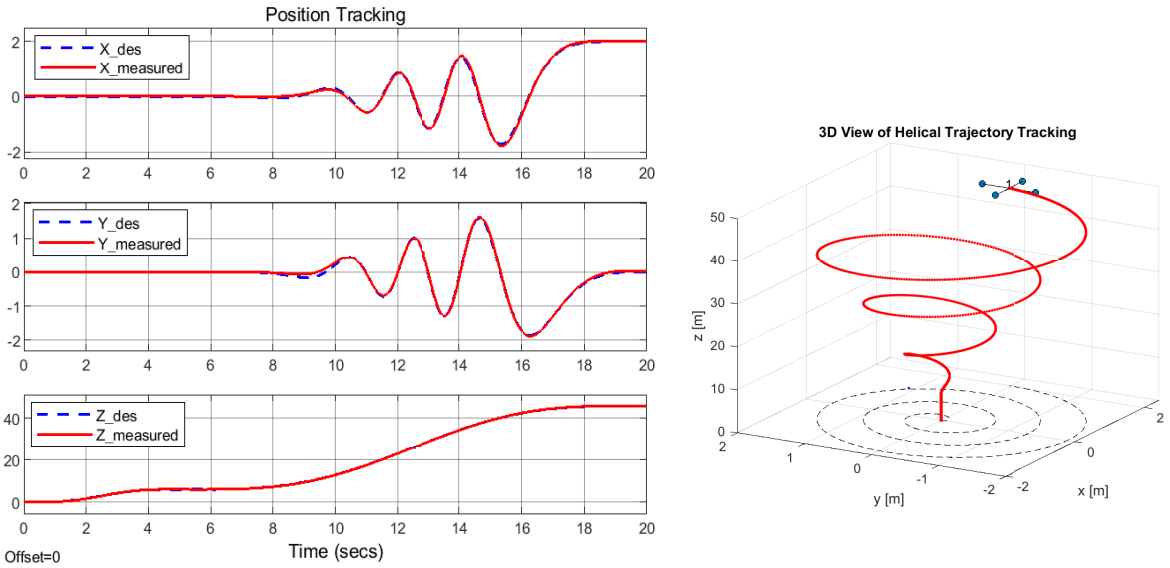


Figure 7.8: Position Tracking

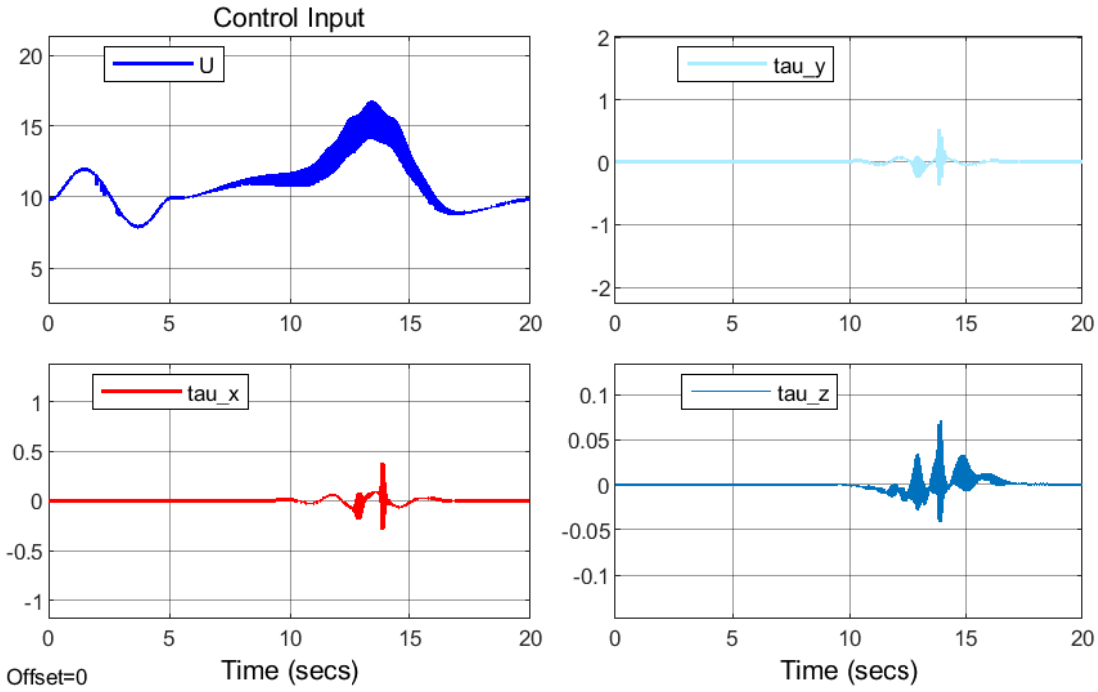


Figure 7.9: Control Inputs for Helical Trajectory

The navigation coordinates of the quad-rotor are seen in Figure 7.8 where they can be seen to track the given helical reference signal. The attained pose of the vehicle is also seen in Figure 7.7 along with the commanded attitude quaternion. Tracking is ensured given the generated control input presenteed in Figure 7.9.

7.2.3 Sweep Path

7.2.4 Area Coverage

The sweep path is a simple path that can be employed for ensuring area coverage. A higher level path planning algorithm is able to generate way-points. However, in this thesis, certain way points are defined manually. A navigation speed of $2m/s$ is taken to prepare the flight time table.

Table 7.1: Specified Way-points and Flight Time Table

x	Y	Z	Time (*sec*)
0	0	0	0
0	0	6	3
0	12	6	9
3	12	6	10.5
3	0	6	16.5
6	0	6	18
6	12	6	24
9	12	6	25.5
9	0	6	31.5
12	0	6	33

After the preparation of the flight time table, the problem of minimum snap trajectory generation is formulated as a constrained optimization problem to be solved using the Euler-Lagrange optimization technique.

Given the above specified way-points, both continuous and fixed constraints are imposed on the higher order terms of position between each segments of the path. In order to solve the constrained optimization problem, the trajectory generator is implemented in Matlab making use of the quadratic programming function. The solution of the optimization problem results in minimum snap trajectories decoded by a seventh order polynomial whose description do contain 8 coefficients. For the sweep trajectory, the smooth trajectories for the flat outputs x, y, z, ψ and their derivatives upto the third order are determined and are presented in Figure 7.10

Trajectory Preparation

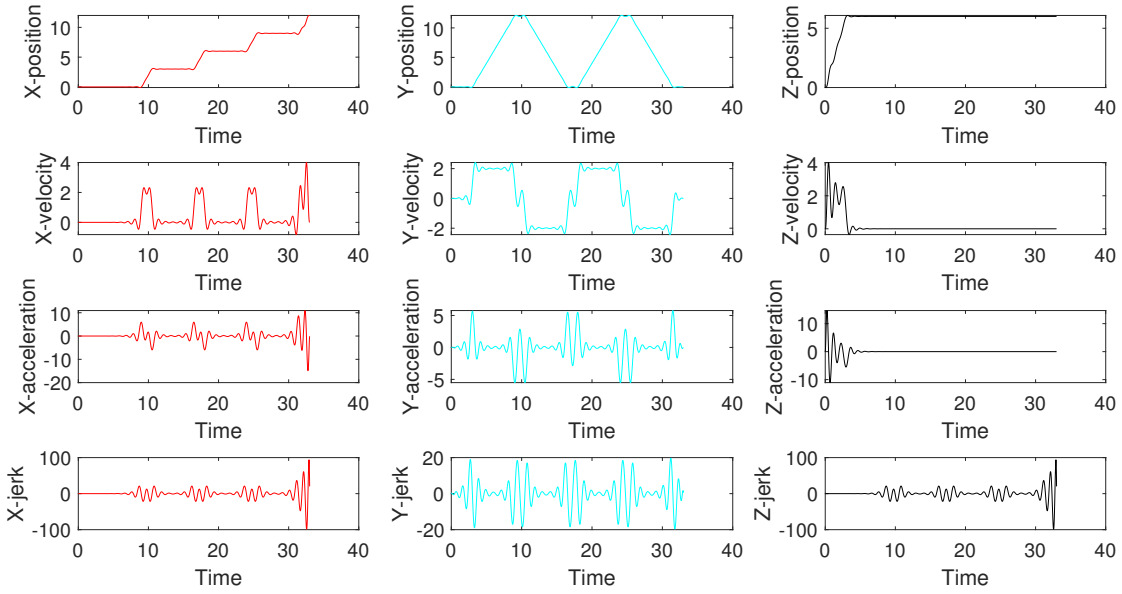


Figure 7.10: Minimum Snap Sweep Trajectory

Tracking Performance

Given the trajectory prepared, it is seen that the control scheme ensures that the quad-rotor maintains its attitude angles to allow it to perform tracking of the target configuration. Figure 7.11. shows the attitude quaternion vector corresponding to a nominal operating scenario.

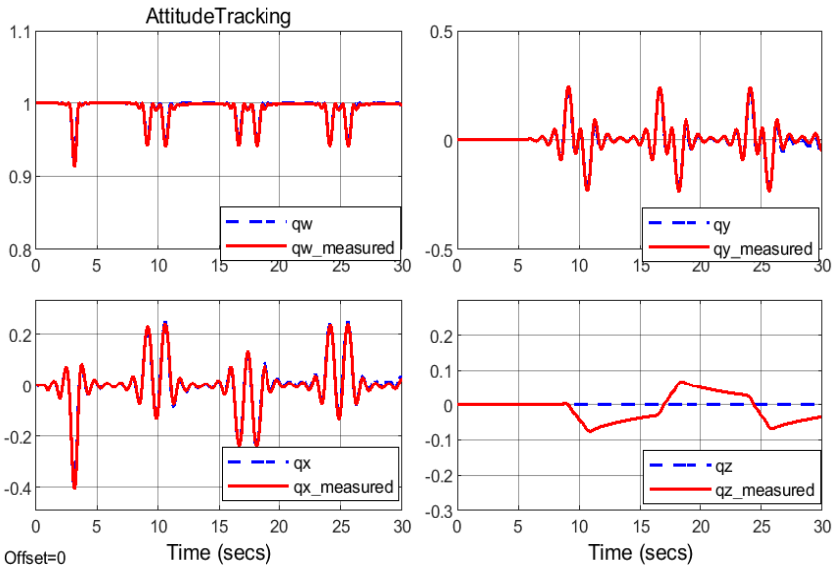


Figure 7.11: Attitude Tracking

The translational position of the quad-rotor is depicted in Figure 7.12 where the developed control scheme forces the quad-rotor to track the desired trajectory. The attained translational velocity of the quad-rotor is also seen to track the configuration of the velocity prepared from the reference position trajectory.

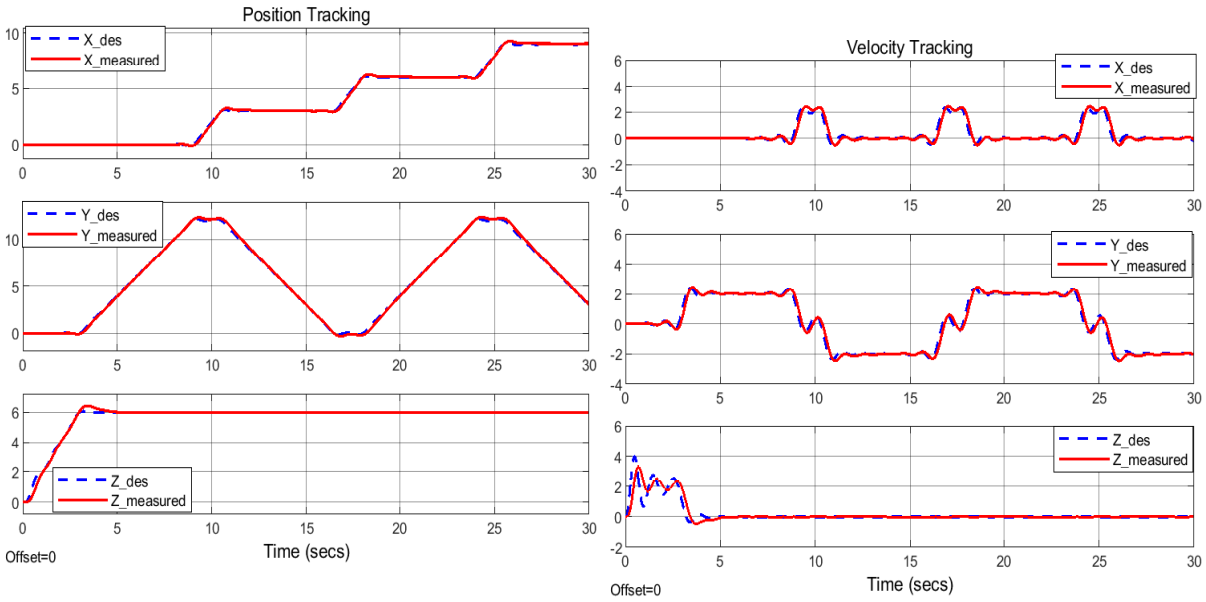


Figure 7.12: Position Tracking

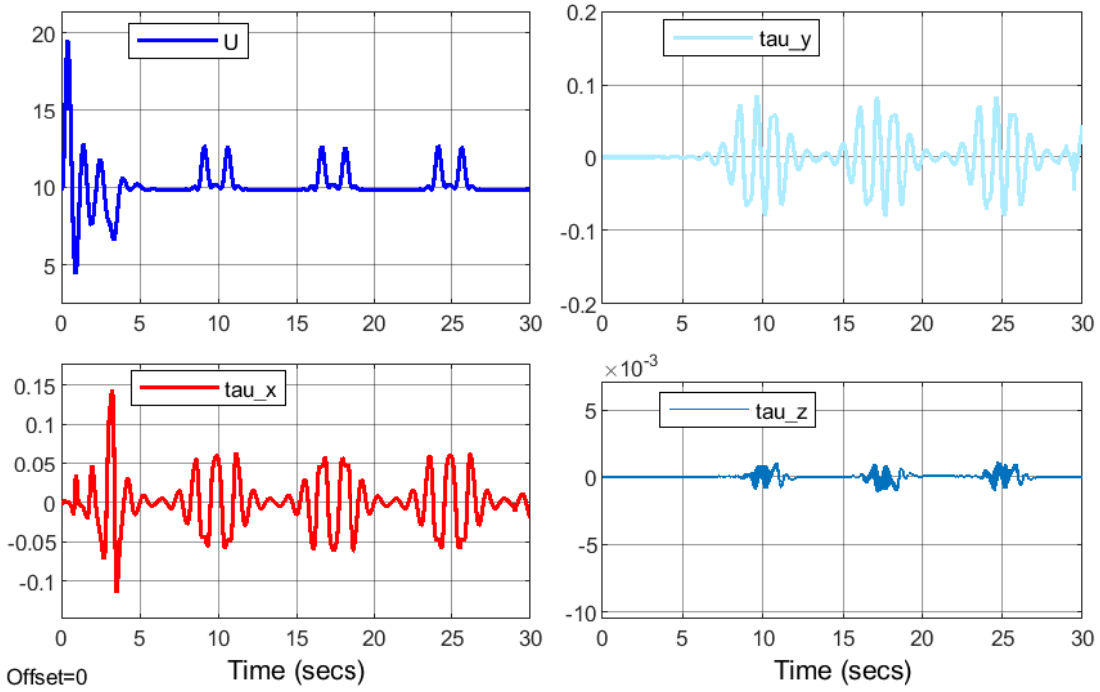


Figure 7.13: Control Inputs for Sweep Trajectory

The position error is seen to be smoothly converge to zero as seen in Figure 7.14 with the smooth control inputs shown in Figure 7.13. Furthermore in Figure 7.15, it is shown that the given sweep trajectory is tracked with smooth commanded speed to motors.

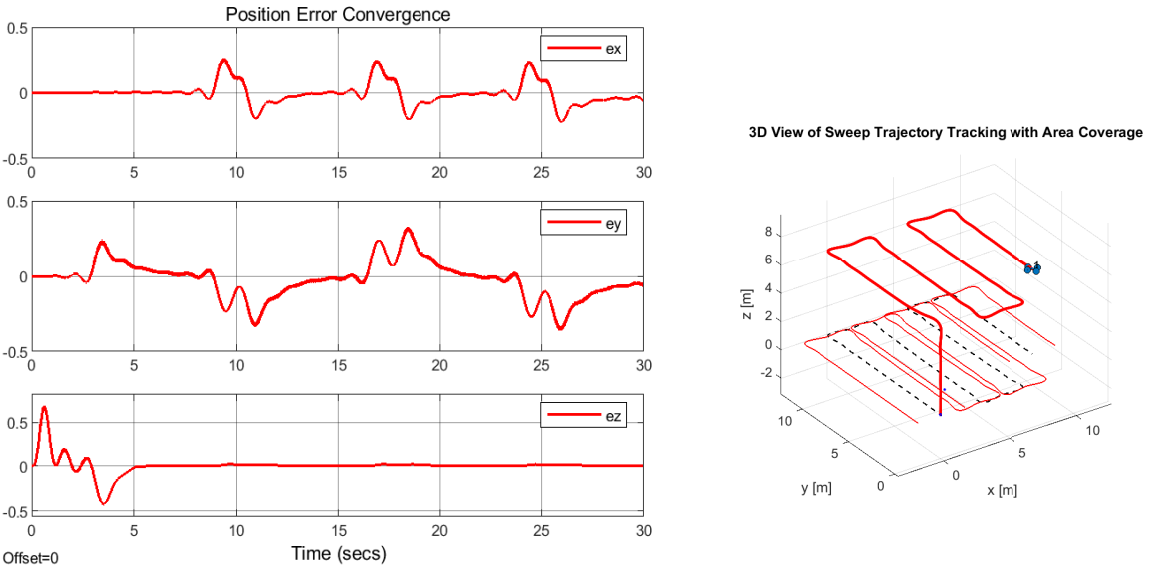


Figure 7.14: Position Error Convergence

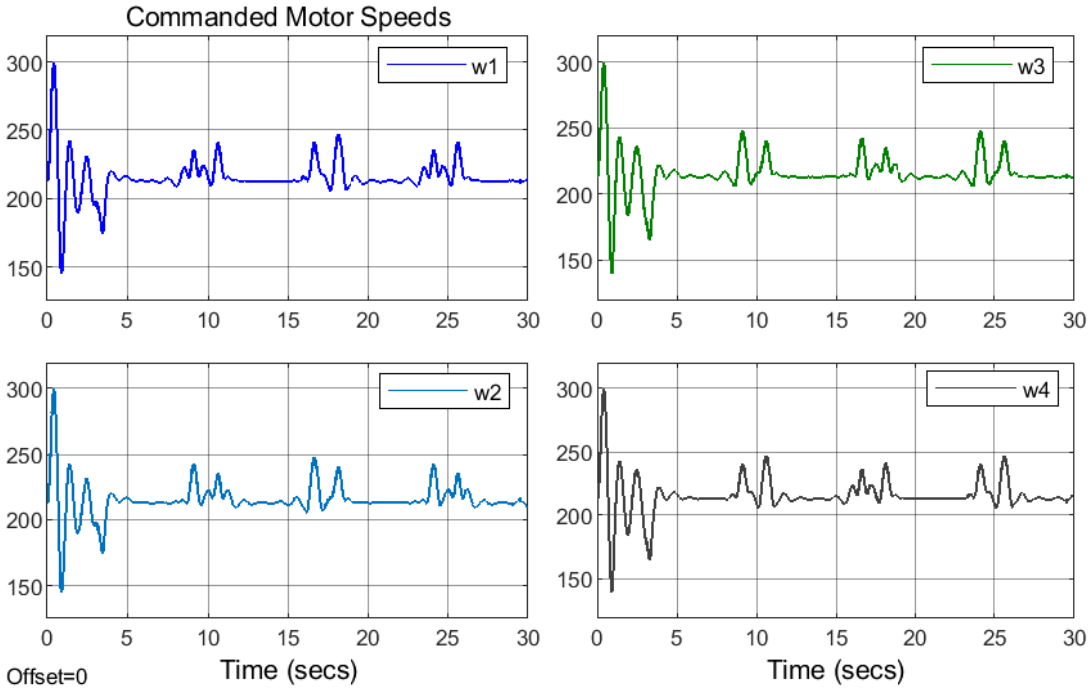


Figure 7.15: Speed Commanded to Motors

7.3 Disturbance Rejection and Uncertainty Tolerance

In this section the capability of adaptive augmentation to reject disturbances and overcome parametric uncertainties is assessed. The integral time absolute error (ITAE) is used as a performance index so as to compare the performance of the adaptive augmentation to that of the sole baseline controller. The application based sweep

trajectory is chosen to simulate this behaviour. Different scenarios are defined to test the developed adaptive augmentation for its performance in inherent uncertain conditions and largely disturbed environments.

7.3.1 Scenario-1: Operation in Disturbed Environment

Presence of external disturbance to the quad-rotor system is done by simulating a scenario of a windy environment whose illustration is presented in Figure 7.16.
A strong, simultaneous constant load disturbance momentum of $\tau_{dx} = \tau_{dy} = -0.3NM$ is assumed to act in the roll and pitch directions from time $t = 10sec$ to $t = 15sec$. Furthermore, it is assumed that, the quad-rotor is disturbed in its longitudinal dimension with a vertical drag force of $U_d = -1.5N$ during the specified time interval.
However, it can be seen from in Figure 7.17 that during the time the disturbance happens, the overall control command is adapted in a way that the overall systems' stability is not endangered.

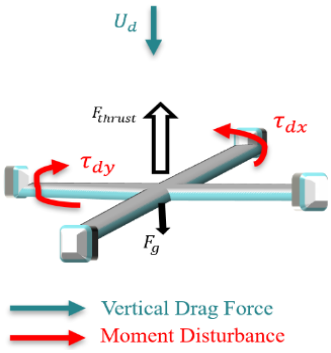


Figure 7.16: Windy Environment

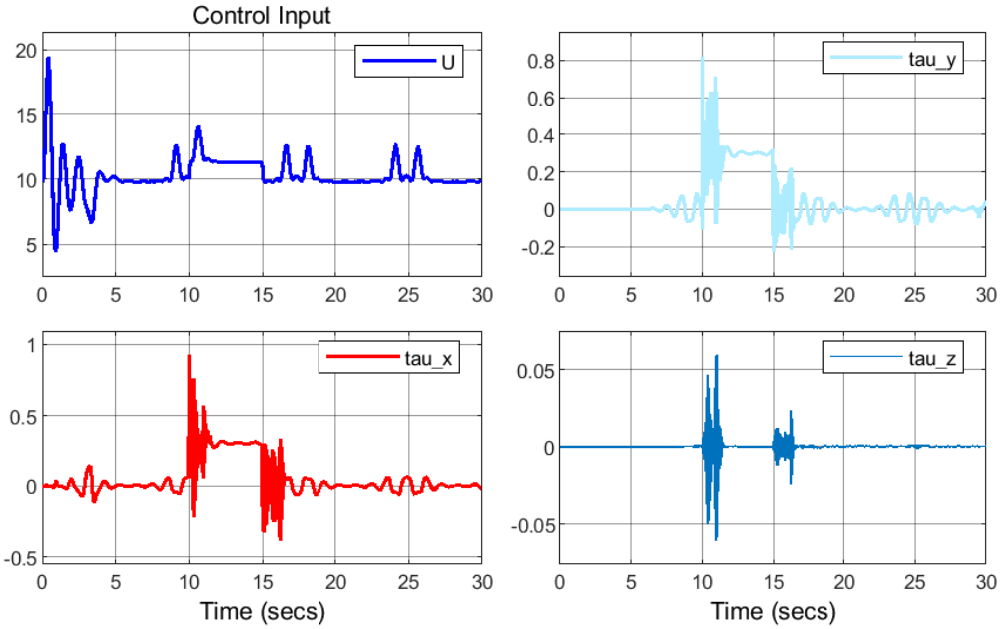


Figure 7.17: Control Input With Adaptive Augmentation - External Disturbance

Figure 7.18 shows the translational response of both the nominal (baseline) controller and the adaptive augmentation. Despite the large bounded disturbance imposed in to the quad-rotor system, it can be seen to compensated completely by the adaptive augmentation. Without the extension of adaptive augmentation, the baseline controller is unable to cope with the external disturbance leading to an unstable closed loop behaviour. However, the overall control scheme is seen to restore close to the

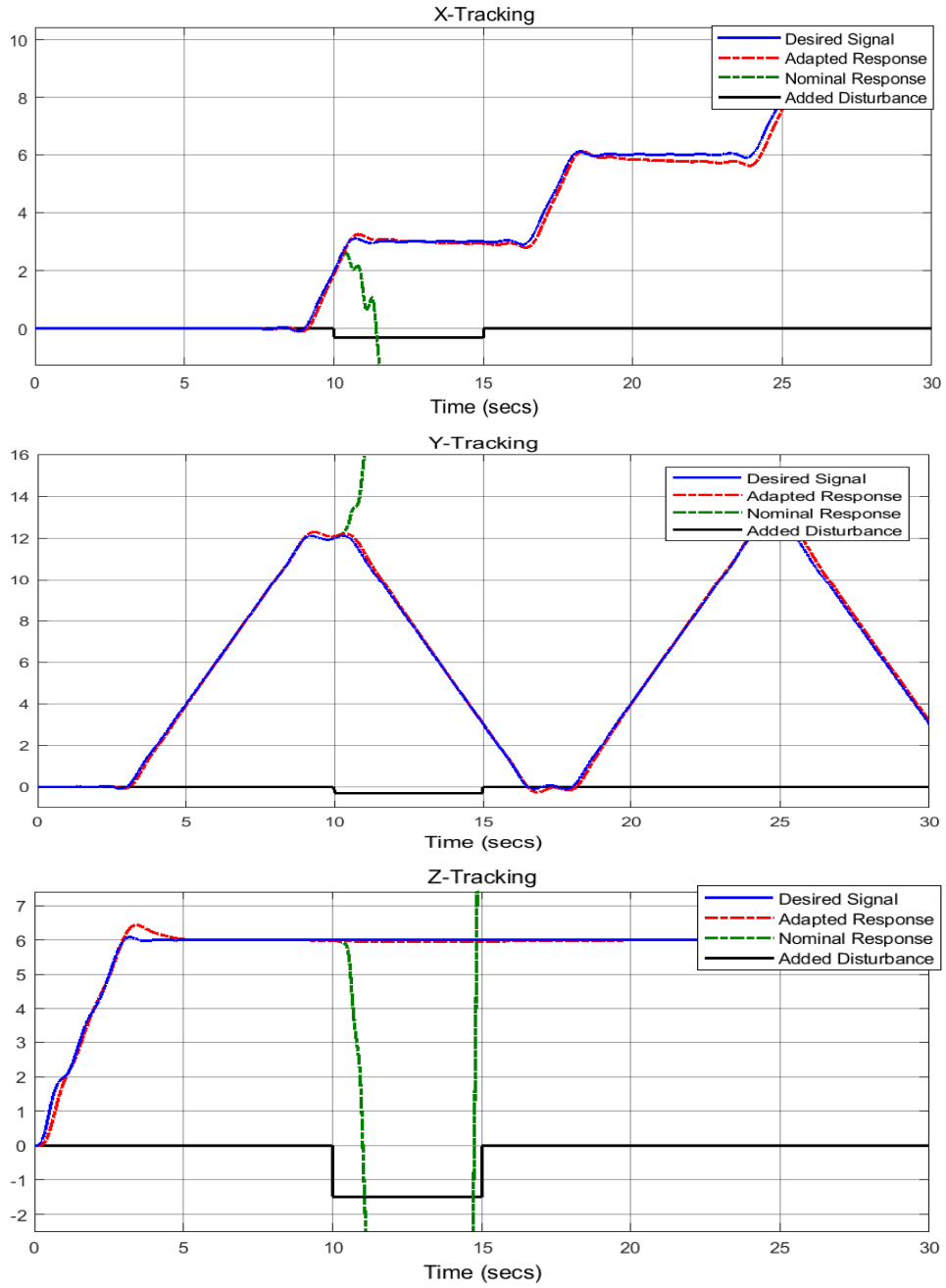


Figure 7.18: Adaptation in a Disturbed Environment

nominal design performance fastly as seen from the performance analysis summarized in Table: 7.3.

7.3.2 Scenario-2: Parametric Uncertainties

Uncertainty tolerance is performed for the problem of trajectory tracking of the sweep path while parametric uncertainties described in Table 7.2 are imposed into the overall system. Two types of parametric uncertainties are imposed. The two

aerodynamic coefficients needed for thrust and moment generation are assumed to be 5 times their nominal design values. Looking at the dynamics of the quad-rotor, it can be seen that uncertainty of this parameters causes control effectiveness issue. Further more the inertial tensor is assumed to be uncertain having 5 times the design time value. The capability of the adaptive augmentation so as to restore the nominal design performance is analysed. Furthermore comparison is made to compare the performance of the adaptive augmentation with the nominal design.

Figure 7.19 shows the control commands demanded from the controller so as to restore the nominal design performance while being possessed with this parametric uncertainties.

Table 7.2: Imposed Uncertainties

Imposed Uncertainty	Value
Parametric Uncertainty	$I_x = 5 * I_{xn}$, $I_y = 5 * I_{y_n}$, $I_x = 5 * I_{xn}$
Input uncertainty	$k_f = 5 * k_{f_n}$ $k_m = 5 * k_{mn}$,

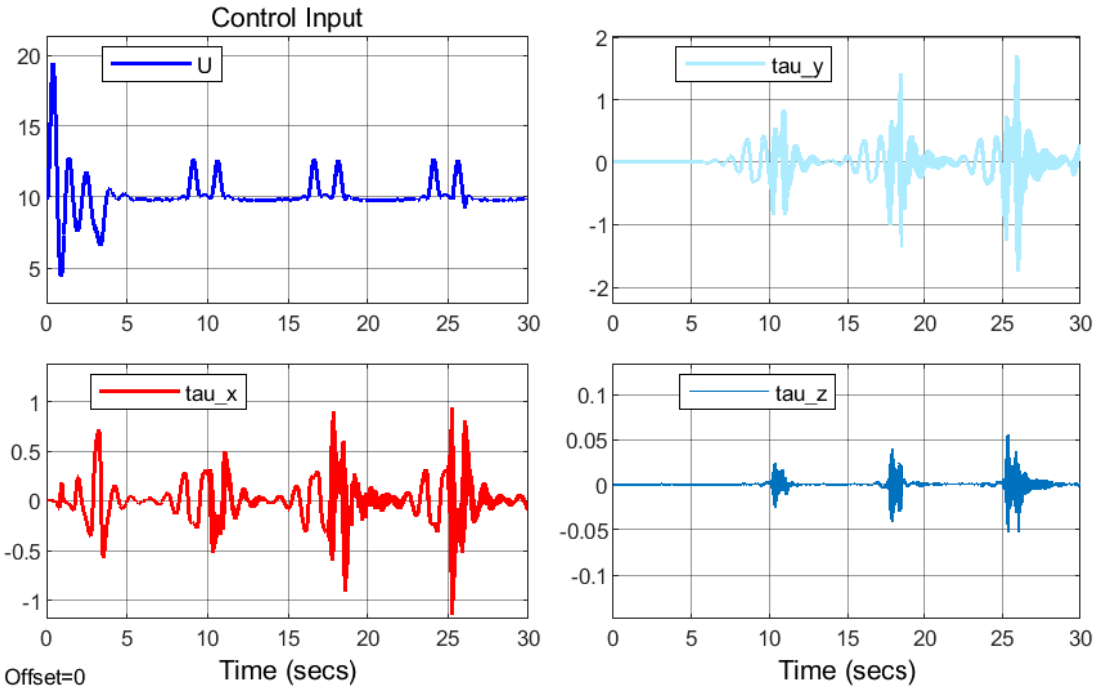


Figure 7.19: Adaptive Control Input - Inertial Uncertainty

Figure 7.20 shows the translational response of both the nominal controller and the adaptive augmentation while the system is posed with inertial and aerodynamics uncertainties. In the simulation presented, it can be noted that the adaptive controller maintains the stabilization of the quad-rotor. However, the fixed control parameters of the baseline controller couldn't cope with the varying system parameters. The closed loop system using the baseline controller gets inherently unstable while this undesired behaviour is completely suppressed by the augmented adaptive controller.

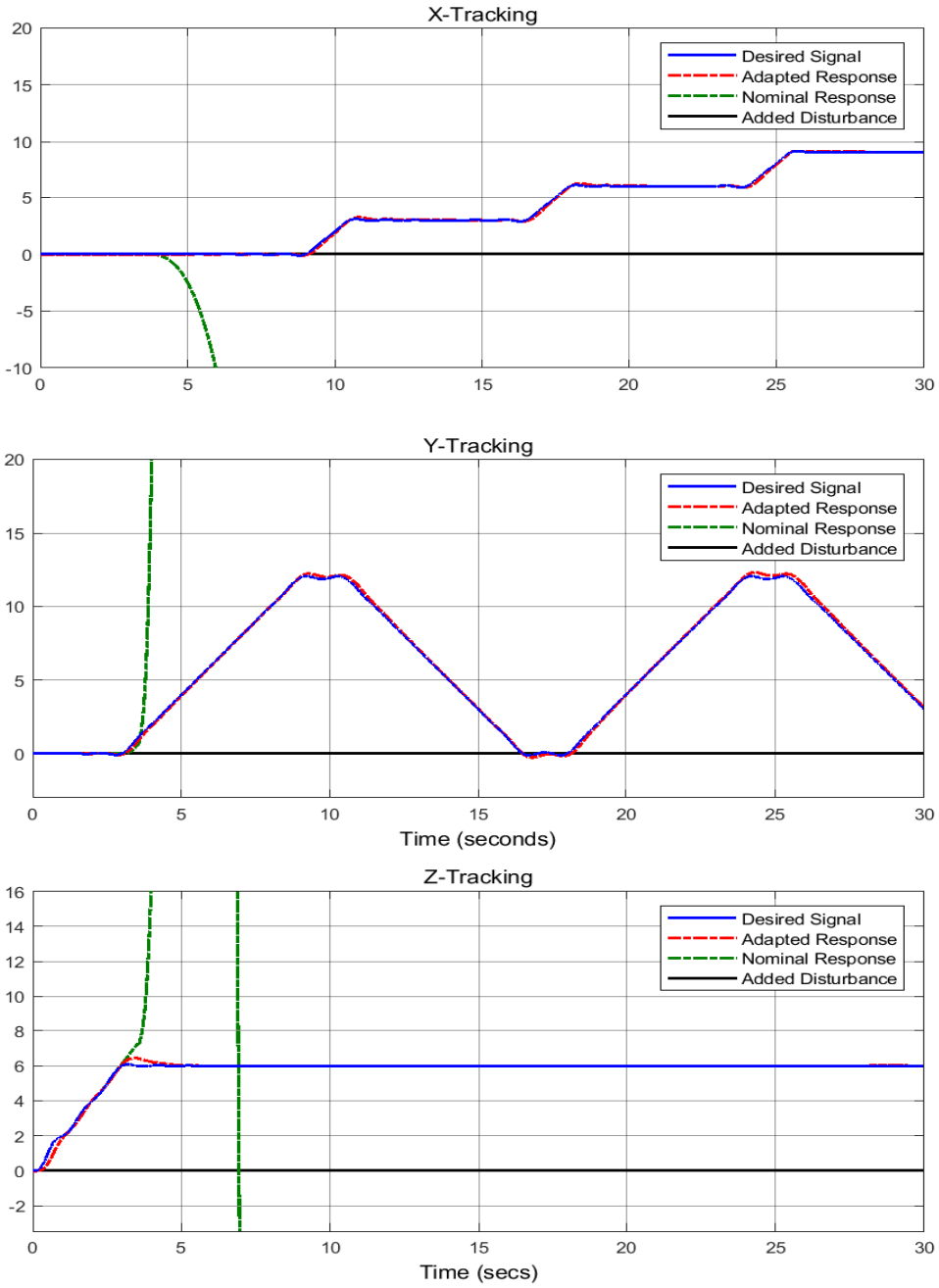


Figure 7.20: Adaptation of Navigation Coordinates - Imposed Inertial Uncertainty

Table: 7.3 shows the performance analysis of the quad-rotor's stabilization problem. The performance of both the nominal and adaptive augmentation is analysed in nominal, disturbed and uncertain operating conditions. With the adaptive augmentation, it is shown that nearly nominal design performance is restored even though the disturbed environment and uncertain conditions make the system controlled by the sole baseline controller inherently unstable.

Table 7.3: Performance Index

Operating Condition	Adaptive Controller response	Baseline Controller response
Nominal operating condition	$ITAE = 73.53$	$ITAE = 73.53$
Imposed parametric uncertainties	$ITAE = 76.53$	$ITAE = 1.742e + 06$
Operation in disturbed environment	$ITAE = 150.4$	$ITAE = 2.237e + 06$

7.4 Image Recognition

7.4.1 Data Preparation Phase

The image recognition task is done using 4000 images with each in one of the four different classes. The class labels of this multi class classification problem are 'Water Hyacinth','No invasive Aquatic Weed', 'Pennywort' and 'Water Primrose'. Each of the class categories have 1000 images under them. Then the train: validation : and test splitting was done.

Among the 1000 images in each category, only 800 images are used for training. 100 images are then utilized for the validation phase for determining the performance of the training model in each back-propagation steps.. While the remaining 100 images from each category are secured for testing purposes which will be utilized for the purpose of testing the model with unseen data.

The train : validation : test split then leaves the problem with the following set-up.

- 3200 Training Images equally balanced among the four classes of images
- 400 Validation Images and
- 400 Testing Images

7.4.2 Training Phase

The training is implemneted using the python programming language in the Jupytor-Notebook environment. As mentioned in Chapter-5, the VGG-16 architecture is adopted and a further rettuning is performed on top of that. The implementation overall is done as follows:

Importing the necessary libraries for the data preprocessing and training phase:

```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from glob import glob

import warnings
warnings.filterwarnings("ignore", category= FutureWarning)

from keras import backend as k
from keras.layers import Activation
from keras.metrics import categorical_crossentropy

from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools
%matplotlib inline

# ---------------
from PIL import Image
import os
from IPython.display import display
from IPython.display import Image as _Imgdis
from datetime import datetime
from keras.callbacks import ModelCheckpoint
from keras import callbacks
```

Specifying the directory where the training and validation dataset are stored:

```
train_path= '/JupyterNotebooks/ImageRecognizer/DataSet/Train'
valid_path= '/JupyterNotebooks/ImageRecognizer/DataSet/Validation'
test_path= '/JupyterNotebooks/ImageRecognizer/DataSet/Test'
```

The following is done to load the VGG-16 model from the keras applications module with the top fully connected layers not included. Then to make use the low-level features learned by the original $VGG-16$ model, the earlier layers are made to freeze and are directly adopted to the current problem.

```
# Loading the weights of pretrained model
vgg_model = VGG16(input_shape =IMAGE_SIZE +[3], weights='imagenet',
↪  include_top= False)

# Freezing the earlier layers of the loaded pre-trained VGG16- model
vgg_model.output
for layer in vgg_model.layers:
    layer.trainable =False
```

The output layer of the originally loaded VGG-16 model is flattened so that it's shape would fit to a column vector. Then an output layer with 4 neurons is added on top of the flattened last layer. The newly established model will then have an output layer with four neurons.

Since this is a multi-class classification problem, the activation function for each of the added neurons in the output layer is set to be the softmax activation function as shown in the following snippet.

```
# Making the only two changes that are required: the output layer and the
↪  input layer.
x= Flatten()(vgg_model.output)  # specifing a flattern layer, which ever
↪  output we get at the output layer, we can dense that in to one
↪  dimention using flatten layer

# There are thousand outputs coming from the vgg16 model output, so the
↪  flatten laayer constructs a layer with thousand neuron /node from
↪  that.
# Constructing output layer with only 4 neurons since the problem is a
↪  four class classification problem.
# The activation function is specified as softmax because it is multiclass
↪  classification problem.
prediction = Dense (len(FoldersInDir), activation='softmax')(x)
model =Model(inputs=vgg_model.input, outputs=prediction)
model.summary()
```

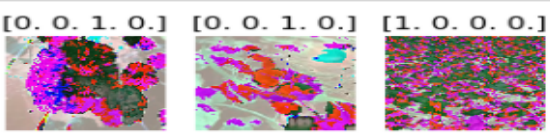
Then, in order for the training and validation images to fit that of the images that were originally used when the VGG-16 model was trained. Certain preprocessing has to be done.This image preprocessing can be by utilizing the pre-process input function.

In order to prevent the model from over-fitting to the training set, data variations have been embedded in to the training process through the technique of data augmentation. Random rotations, shearing effect and brightness variation are the techniques that have

been utilized.

```
# Applying data augumentation technique to the training set
train_datagen=ImageDataGenerator(
            preprocessing_function =preprocess_input,
            rotation_range =40,
            shear_range=0.2,
            brightness_range=[0.2,1.0],
            fill_mode='nearest',)
# Applying data augumentation technique to the testing set
test_datagen=ImageDataGenerator(
            preprocessing_function =preprocess_input,
            rotation_range =40,
            shear_range=0.2,
            brightness_range=[0.2,1.0],
            fill_mode='nearest',)
```

The following is a sample of the normalized preprocessed images that are ready to be fed to the training phase. The classes of the image is shown at the top of each image. In



order to do fine tuning on the top of the newly formed model, an optimizer function has to be used. Thus, the Adam optimizer is called as follows:

```
# Calling the adam optimizer as optimizer to be used in Backpropation step
↪  with a small learning rate of 0.0001
from tensorflow.keras import optimizers
adam= optimizers.Adam(learning_rate=.0001)
model.compile(optimizer= adam,
              loss ='categorical_crossentropy',
                     metrics=['accuracy'])
```

Training and validation data is generated as follows using the data generator object called above.

```
# TrainData generation
# batch size= 32 means the model will be trained on the batch of 32
train_set=train_datagen.flow_from_directory(train_path,
                                            target_size=(224,224),
```

```
                                                batch_size=32,
                                                ↪ classes=['Hyacinth',...
                                                ↪ 'NoAquaticWeed',...
                                                ↪ 'Pennywort','Primrose'])
# TestData generation
valid_set=test_datagen.flow_from_directory(valid_path,
                                                target_size=(224,224),
                                                batch_size=32,
                                                ↪ classes=['Hyacinth',...
                                                ↪ 'NoAquaticWeed',...
                                                ↪ 'Pennywort','Primrose'])
```

The training phase is started by specifying the following training parameters as follows along with the configuration for fitting the model.

- : Training batch size = 32
- : Validation batch size = 32
- : No of Epochs = 10
- : Steps per epoch = 100
- : Validation steps = 13
- : Metric= Validation Loss

```
# Configuring to save the best model using the validation loss as a metric
checkpoint= ModelCheckpoint(filepath=
↪ os.path.join(DATADIR,'AquaticWeedClassifier.h5'),
                            verbose= 2, save_best_only=True)
# Specifying an early stopping scenario
earlyStopping=callbacks.EarlyStopping(monitor = "val_loss",
                                      mode ="min",
                                      patience=6)
start = datetime.now()
# Fine tuning the model
model_history=model.fit(
                    train_set,
                    validation_data=valid_set,
                    epochs=10,
                    steps_per_epoch=100,
                    validation_steps=13,
                    callbacks=[checkpoint, earlyStopping],
                    verbose=2)
duration=datetime.now()-start
```

```
# The following is done to keep track of how much time the training
↪ process takes
print ("Training completed in time duration of: ", duration)
```

Over the training phase, the validation the training accuracy has evolved. The trainable parameters that were trained trough the training process are only 100,356 parameters among the 14,714,688 parameters of the original VGG-16 model.

7.4.3 Performance Analysis

7.4.3.1 Training and Validation Performance

Fifteen hours were spent for training to reach at an accuracy level of the following.

- : Training accuracy = 93.34 %
- : Validation accuracy = 94.25 %

A plot of the training history can be shown as follows. Each of the four parameters: the validation loss and accuracy and the training loss and accuracy are implicated over the training and validation phase. Any further training from the 6^{th} epoch onwards made the validation loss to increase resulting the model to over-fit to the training data. This is undesired in the sense that the model possibly will have poor performance on unseen data. Hence, the model trained up to the 6^{th} epoch with the mentioned training and validation accuracies is taken as the final feasible model.

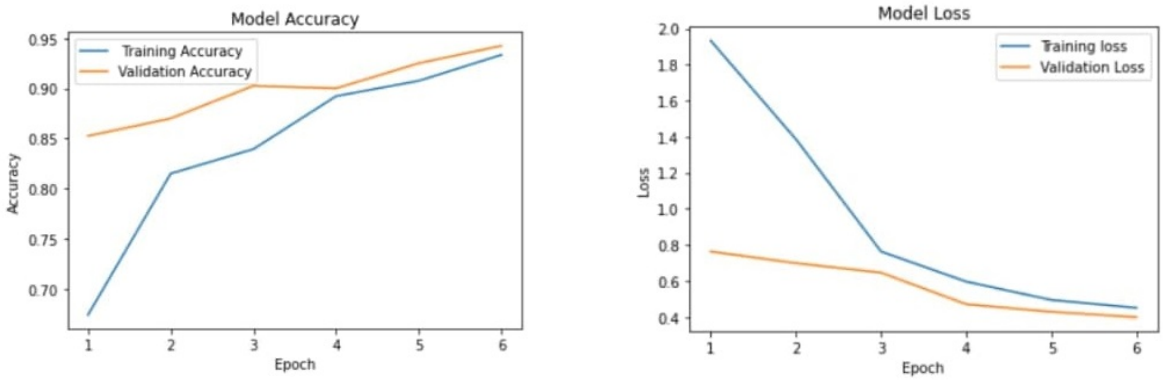


Figure 7.21: Model Accuracy and Model Loss

Good implication are obtained from the result since the validation accuracy is pretty close to the training accuracy with very slightly higher values. The accuracy level on unseen data being close to the training accuracy means that over-fitting has not been encountered during the training phase. Even though, the physical structures of the three

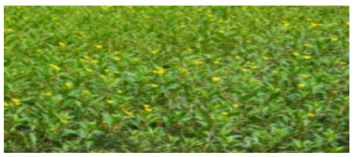
aquatic weeds: 'Water Hyacinth', 'Pennywort' and 'Water Primrose' is pretty similar, each was discriminated from the other in the validation phase.

7.4.3.2 Testing Images and Model Prediction

After the conclusion of the training and validation stage, the built recognition model was tested with the pre-split testing set. 400 images were used for testing among which 372 images were classified in correctly, the same way as their true labels. Over the whole testing set, the testing accuracy was found to be 93.0%.

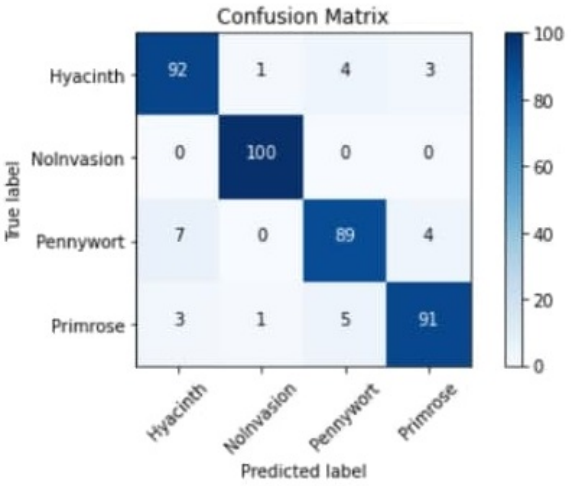
Some of testing samples are presented in Table 7.4 where the probability level acquired is presented depicting resemblance of the samples to each class labels. For reference purposes, the true labels of the selected tests samples is also presented alongside with their prediction.

Table 7.4: Testing Samples of Built Model

Testing Samples	Predicted Probability of resemblance to each class
	[3.9372324e-33 1.0000000e+00 0.0000000e+00 0.0000000e+00] True Label - 'No Aquatic Weed = [0 1 0 0]'
	[1.2261928e-06 1.3938758e-13 9.9999869e-01 1.1247215e-07] True Label - 'Penny Wort = [0 0 1 0]'
	[1.0000000e+00 1.9171901e-36 5.4850929e-30 8.0736936e-16] True Label - 'Water Hyacinth = [1 0 0 0]'
	[8.6953062e-01 2.3309435e-18 2.5889456e-05 1.3044347e-01] True Label - 'Water Hyacinth = [1 0 0 0]'
	[5.0385745e-14 4.0761062e-18 1.0000000e+00 1.7065624e-15] True Label - 'Penny Wort = [0 0 1 0]'
	[1.3778660e-04 1.0179858e-14 1.2283772e-07 9.9986207e-01] True Label - 'Primrose = [0 0 0 1]'

Confusion Matrix

A four by four confusion matrix is presented below for evaluating the performance of a classification model. The confusion matrix compares the actual class labels with those of the predicted labels by the built deep learning model. This gives us a holistic view of how well the classification model performed on unseen data from the testing set is performing and what kinds of errors are being committed by it. The Confusion matrix obtained while performing test is presented below in as follows. The information presented includes 'True positives', 'True negatives', 'False Positives' and 'False negatives' are depicted for each of the four classes. Further analysis is done on the obtained confusion matrix by computing other metrics whose indication of the performance of the trained model is presented in Figure 7.22.



	precision	recall	f1-score	support
Hyacinth	0.90	0.92	0.91	100
No Aquatic Weed	0.98	1.00	0.99	100
Pennywort	0.91	0.89	0.90	100
Primrose	0.93	0.91	0.92	100
accuracy			0.93	400
macro avg	0.93	0.93	0.93	400
weighted avg	0.93	0.93	0.93	400

Figure 7.22: Precision — Recall — F1-score — Support

- Precision: encodes the information of how many of the correctly predicted cases actually turned out to be positive. For each of the four classes, the precision is computed as the following :

$$Precition = \frac{TruePositive}{FalsePositive + TruePositive}$$

- Recall: presents how many of each classes' actual positive true cases are able to be predicted in a correct manner through the build recognizer model. This is computed for of each classes in the following manner.

$$Precition = \frac{TruePositive}{FalseNegative + TruePositive}$$

- F1-Score- The F1-score metric provides a way to capture the information how both the precision and recall change in a single value. Thus it presents a combined idea of both the two metrics. For each class predictions, it can be calculated as the following:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{7.2}$$

Chapter 8

Conclusion and Recommendation

8.1 Conclusion

The work in this thesis considers the modelling and stabilization problem of a quad-rotor. A non-linear model that best mimics the behaviour of the underlying real world quuad-rotor is developed by fair assumptions. Baseline controllers have been developed to for both the position and attitude subsystem. Then, a model reference adaptive controller is developed to be used as an augmentation to the baseline design when uncertain and disturbed operating conditions are considered.

The trajectory tracking performance of the developed baseline controller is tested by various flight manuevers when nominal flight conditions are considered. The developed baseline controller is able to perform trajectory tracking in operating conditions that do not deviate much from the design time considerations.

Furthermore, the performance of the adaptive scheme is tested by simulating a disturbed environment which consists of drag forces and opposing moments counteracting with the vehicle's motion. The adaptive scheme is found to be able to cope with the posed disturbances while the nominal controller results in an unstable closed loop behaviour. Furthermore parametric uncertainties are imposed in to the overall system with an assumption no knowledge of aerodynamic coefficients and the inertia parameters at design time. However, The adaptive augmentation is able to ensure stabilization of the quad-rotor and restores close to the nominal design performance despite the inherent instability observed in the baseline controller.

For the image recognition problem, 4000 images categorized in to 4 classes are

trained using a technique named fine tuning. Though data scarcity has been bottleneck for achieving a higher performance, the built recognition model has obtained 93.34 % of training accuracy, 94.25 % of validation accuracy and 93.0 % of testing accuracy. Moreover, the confusion matrix and other related metrics are computed to assess how well the developed model performs on unseen data.

8.2 Recommendation for Future Work

Possible future extensions of the work of this thesis are put in the following three main points:

- A possible higher level design is the implementation of an off-line or on-line coverage path planner algorithm. Consideration of area coverage of non-geometric target areas is also a possible future work. This as a result can make the overall system to be fully autonomous.
- Parameters estimation convergence to true values is another area where future analysis can be done. An implementation of this equips the designed system with the capability to be used as a parameter identification model.
- The image classification task implemented in this paper can further be reformulated as an object detection problem. This reformulation enables the system to be able to identify the presence of multiple classes of images in a single captured shot.

References

[1] T. Center, M. Hill, H. Cordo, and M. Julien, "Waterhyacinth," *Biological Control of Invasive Plants in the Eastern United States. Forest Health and Technology Enterprises Team, West Virginia*, pp. 41–64, 2002.

[2] W. T. Penfound and T. T. Earle, "The biology of the water hyacinth," *Ecological Monographs*, pp. 447–472, 1948.

[3] M. G. Dersseh, A. A. Kibret, S. A. Tilahun, A. W. Worqlul, M. A. Moges, D. C. Dagnew, W. B. Abebe, and A. M. Melesse, "Potential of water hyacinth infestation on lake tana, ethiopia: A prediction using a gis-based multi-criteria technique," *Water*, vol. 11, no. 9, p. 1921, 2019.

[4] N. KEDİRKAN, "Water surface changes of lakes in the central rift valley of ethiopia," *International Journal of Environment and Geoinformatics*, vol. 6, no. 3, pp. 264–267, 2019.

[5] K. H. Thamaga and T. Dube, "Remote sensing of invasive water hyacinth (eichhornia crassipes): A review on applications and challenges," *Remote Sensing Applications: Society and Environment*, vol. 10, pp. 36–46, 2018.

[6] J. Mukarugwiro, S. Newete, E. Adam, F. Nsanganwimana, K. Abutaleb, and M. Byrne, "Mapping distribution of water hyacinth (eichhornia crassipes) in rwanda using multispectral remote sensing imagery," *African Journal of Aquatic Science*, vol. 44, no. 4, pp. 339–348, 2019.

[7] G. Singh, C. Reynolds, M. Byrne, and B. Rosman, "A remote sensing method to monitor water, aquatic vegetation, and invasive water hyacinth at national extents," *Remote Sensing*, vol. 12, no. 24, p. 4021, 2020.

[8] Y. Ghoussein, H. Nicolas, J. Haury, A. Fadel, P. Pichelin, H. Abou Hamdan, and G. Faour, "Multitemporal remote sensing based on an fvc reference period using sentinel-2 for monitoring eichhornia crassipes on a mediterranean river," *Remote Sensing*, vol. 11, no. 16, p. 1856, 2019.

[9] J. Everaerts *et al.*, "The use of unmanned aerial vehicles (uavs) for remote sensing and mapping," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, no. 2008, pp. 1187–1192, 2008.

[10] L. Tang and G. Shao, "Drone remote sensing for forestry research and practices," *Journal of Forestry Research*, vol. 26, no. 4, pp. 791–797, 2015.

[11] Z. Duan, Y. Li, J. Wang, G. Zhao, and S. Svanberg, "Aquatic environment monitoring using a drone-based fluorosensor," *Applied Physics B*, vol. 125, no. 6, pp. 1–8, 2019.

[12] T. Zhang, "Deployment of real time uav aerial surveillance with coverage model," 2017.

[13] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[14] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012.

[15] A. Poultney, C. Kennedy, G. Clayton, and H. Ashrafiuon, "Robust tracking control of quadrotors based on differential flatness: Simulations and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 1126–1137, 2018.

[16] M. N. Duc, T. N. Trong, and Y. S. Xuan, "The quadrotor mav system using pid control," in *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2015, pp. 506–510.

[17] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2451–2456.

[18] Y. Sun, N. Xian, and H. Duan, "Linear-quadratic regulator controller design for quadrotor based on pigeon-inspired optimization," *Aircraft Engineering and Aerospace Technology*, 2016.

[19] H. Chen and N. Sun, "Nonlinear control of underactuated systems subject to both actuated and unactuated state constraints with experimental verification," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 9, pp. 7702–7714, 2019.

[20] A. Das, F. Lewis, and K. Subbarao, "Backstepping approach for controlling a quadrotor using lagrange form dynamics," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1, pp. 127–151, 2009.

[21] J. V. Harrison, J. L. Gallagher, and E. J. Grace, "An algorithm providing all-attitude capability for three-gimballed inertial systems," *IEEE Transactions on Aerospace and Electronic Systems*, no. 3, pp. 532–543, 1971.

[22] T. S. Andersen and R. Kristiansen, "Quaternion guidance and control of quadrotor," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 1567–1601.

[23] N. B. Knoebel and T. W. McLain, "Adaptive quaternion control of a miniature tailsitter uav," in *2008 American Control Conference*. IEEE, 2008, pp. 2340–2345.

[24] R. Kristiansen and P. J. Nicklasson, "Satellite attitude control by quaternion-based backstepping," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 907–912.

[25] E. Guerrero, H. Abaunza, P. Castillo, R. Lozano, and C. Garcia, "Quadrotor energy-based control laws: A unit-quaternion approach," *J. Intell. Robot. Syst.*, vol. 88, no. 2–4, pp. 347–377, 2017.

[26] D. Lee, H. J. Kim, and S. Sastry, "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter," *International Journal of control, Automation and systems*, vol. 7, no. 3, pp. 419–428, 2009.

[27] T. Madani and A. Benallegue, "Control of a quadrotor mini-helicopter via full state backstepping technique," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 1515–1520.

[28] T. Lee, "Robust adaptive attitude tracking on so3 with an application to a quadrotor uav," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1924–1930, 2012.

[29] B. J. Imran and A. Yesildirek, "Nonlinear composite adaptive control for quadrotor," 2013.

[30] T. Fernando, J. Chandiramani, T. Lee, and H. Gutierrez, "Robust adaptive geometric tracking controls on so (3) with an application to the attitude dynamics of a quadrotor uav," in *2011 50th IEEE conference on decision and control and European control conference*. IEEE, 2011, pp. 7380–7385.

[31] C. Diao, B. Xian, Q. Yin, W. Zeng, H. Li, and Y. Yang, "A nonlinear adaptive control approach for quadrotor uavs," in *2011 8th Asian Control Conference (ASCC)*. IEEE, 2011, pp. 223–228.

[32] V. S. Akkinapalli, G. P. Falconí, and F. Holzapfel, "Attitude control of a multicopter using l 1 augmented quaternion based backstepping," in *2014 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology*. IEEE, 2014, pp. 170–178.

[33] A. Razinkova, I. Gaponov, and H.-C. Cho, "Adaptive control over quadcopter uav under disturbances," in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*. IEEE, 2014, pp. 386–390.

[34] D. Mihailescu-Stoica, R. Acuna, and J. Adamy, "High performance adaptive attitude control of a quadrotor," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3462–3469.

[35] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor unmanned aerial vehicles," in *2012 Proceedings of IEEE Southeastcon*. IEEE, 2012, pp. 1–6.

[36] D. K. Chaturvedi, *Modeling and simulation of systems using MATLAB® and Simulink®*. CRC press, 2017.

[37] T. Bresciani, "Modelling, identification and control of a quadrotor helicopter," *MSc theses*, 2008.

[38] S. Musa, "Techniques for quadcopter modeling and design: A review," *Journal of unmanned system Technology*, vol. 5, no. 3, pp. 66–75, 2018.

[39] P. Donelan, *Kinematic singularities of robot manipulators*. INTECH Open Access Publisher, 2010.

[40] H. Goldstein, C. Poole, and J. Safko, "Classical mechanics," 2002.

[41] S. L. Altmann, "Hamilton, rodrigues, and the quaternion scandal," *Mathematics Magazine*, vol. 62, no. 5, pp. 291–308, 1989.

[42] A. Castillo, R. Sanz, P. Garcia, and P. Albertos, "A quaternion-based and active disturbance rejection attitude control for quadrotor," in *2016 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 2016, pp. 240–245.

[43] P. Castillo, R. Lozano, and A. Dzul, "Stabilization of a mini-rotorcraft having four rotors," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2693–2698.

[44] P. C. Garcia, P. Castillo, R. Lozano, and A. E. Dzul, *Modelling and control of mini-flying machines*. Springer Science & Business Media, 2005.

[45] L. D. Reid, *Dynamics of flight: stability and control*. Wiley, 1996.

[46] R. Lozano, *Unmanned aerial vehicles: Embedded control*. John Wiley & Sons, 2013.

[47] A. Kelly and N. Seegmiller, "A vector algebra formulation of mobile robot velocity kinematics," in *Field and Service Robotics*. Springer, 2014, pp. 613–627.

[48] Y. Wang, L. Xie, and C. E. De Souza, "Robust control of a class of uncertain nonlinear systems," *Systems & control letters*, vol. 19, no. 2, pp. 139–149, 1992.

[49] A. Astolfi, D. Karagiannis, and R. Ortega, *Nonlinear and adaptive control with applications*. Springer, 2008, vol. 187.

[50] M. Burri, T. Hinzmann, and M. Achtelik, "Adaptive rate control for multirotor uavs."

[51] H. A. Gonzalez, "Robust tracking of dynamic targets with aerial vehicles using quaternion-based techniques," Ph.D. dissertation, Université de Technologie de Compiègne, 2019.

[52] H. K. Khalil, "Lyapunov stability," *Control Systems, Robotics and AutomatioN–Volume XII: Nonlinear, Distributed, and Time Delay Systems-I*, p. 115, 2009.

[53] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel, "On quaternion-based attitude control and the unwinding phenomenon," in *Proceedings of the 2011 American Control Conference*. IEEE, 2011, pp. 299–304.

[54] H. Schaub and J. L. Junkins, *Analytical mechanics of space systems*. Aiaa, 2003.

[55] J. Rothe, J. Zevering, M. Strohmeier, and S. Montenegro, "A modified model reference adaptive controller (m-mrac) using an updated mit-rule for the altitude of a uav," *Electronics*, vol. 9, no. 7, p. 1104, 2020.

[56] P. Jain and M. Nigam, "Design of a model reference adaptive controller using modified mit rule for a second order system," *Advance in Electronic and Electric Engineering*, vol. 3, no. 4, pp. 477–484, 2013.

[57] W. J. Rugh, *Linear system theory*. Prentice-Hall, Inc., 1996.

[58] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "On adaptive control with closed-loop reference models: transients, oscillations, and peaking," *IEEE Access*, vol. 1, pp. 703–717, 2013.

[59] P. A. Ioannou and J. Sun, *Robust adaptive control*. Courier Corporation, 2012.

[60] P. Martin, R. M. Murray, and P. Rouchon, "Flat systems, equivalence and trajectory generation," 2003.

[61] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[62] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics research*. Springer, 2016, pp. 649–666.

[63] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A guide to convolutional neural networks for computer vision," *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1–207, 2018.

[64] C. Brown and C. Brown, *Advances in Computer Vision: Volume 1*. Psychology Press, 2014.

[65] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. Kruthiventi, and R. V. Babu, "An introduction to deep convolutional neural nets for computer vision," in *Deep Learning for Medical Image Analysis*. Elsevier, 2017, pp. 25–52.

[66] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[67] E. A. Bolch, M. J. Santos, C. Ade, S. Khanna, N. T. Basinger, M. O. Reader, and E. L. Hestir, "Remote detection of invasive alien species," in *Remote Sensing of Plant Biodiversity*. Springer, Cham, 2020, pp. 267–307.

[68] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[69] C. Szegedy, S. Ioffe, V. Vanhoucke *et al.*, "Inception-resnet and the impact of residual connections on learning [j]," *arXiv preprint arXiv:1602.07261*.

[70] D. Sarkar, R. Bali, and T. Ghosh, *Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*. Packt Publishing Ltd, 2018.

[71] N. Fethalla, "Modelling, identification, and control of a quadrotor helicopter," Ph.D. dissertation, École de technologie supérieure, 2019.

Appendix A

Design Time Plant Parameters

Plant Parameters	Symbol	Unit	Values
K_f	Thrust factor	Ns^2	$54.2 * 10^{-6}$
K_m	Drag factor	Nms^2	$1.1 * 10^{-6}$
g	Gravitational Acceleration	m/s^2	9.81
l	Arm length of the quad-rotor	m	0.24
m	Mass of quad-rotor	Kg	1
I_x	Thrust factor	Nms^2	$8.1 * 10^{-3}$
I_y	Thrust factor	Nms^2	$8.1 * 10^{-3}$
I_z	Thrust factor	Nms^2	$14.2 * 10^{-3}$

Appendix B

Quaternion Operations

B.1 The Quaternion Rotation Matrix

As described in subsection 3.2.2.3 rotating a vector $0 + v_x i + v_y j + v_z k$, defined in three dimensional space as described in a pure quaternion form, involves quaternion multiplication operations.

Rotated version of the vector, v' comprises of pre and post multiplication of the vector with a quaternion q. A rotation matrix which can be utilized for rotating a vector from its body reference frame to the inertial frame can be found as follows:

$$v' = qvq^{-1} \qquad \text{with} \quad q^{-1} = q^T$$

Utilization of expressions for pre and post multiplication of quaternions results in the following:

$$v' = Q(q)Q(q^{-1})v$$

$$\begin{bmatrix} 0 \\ v_x \\ v_y \\ v_z \end{bmatrix}' = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ v_x \\ v_y \\ v_z \end{bmatrix}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}' = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & q_1q_2 - q_0q_3 - q_0q_3 + q_1q_2 & q_1q_3 + q_0q_2 + q_1q_3 + q_0q_2 \\ q_1q_2 + q_0q_3 + q_0q_3 + q_1q_2 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & q_2q_3 + q_2q_3 - q_0q_1 - q_0q_1 \\ q_1q_3 - q_0q_2 + q_1q_3 - q_0q_2 & q_2q_3 + q_2q_3 + q_0q_1 + q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}' = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

B.2 Rate of Change of Quaternion

For a quaternions q, it's rate of change can be calculated using the axis angle representation of a quaternion presented in Eqn: 3.15. The analysis is as follows:

$$q = e^{\frac{\theta}{2}u} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}u$$

$$\dot{q} = e^{\frac{\theta}{2}u}.\frac{u}{2}\dot{\theta}$$

$$\dot{q} = \frac{1}{2}e^{\frac{\theta}{2}u}u\dot{\theta}u$$

But $e^{\frac{\theta}{2}u}$ is the axis angle representation of a quaternion and $\dot{\theta}$ represents the angular velocity.

$$\dot{q} = \frac{1}{2}q \otimes \omega$$

Appendix C

LTV Attitude Reference Model

With the application of the baseline attitude control law developed in Section 4.4, the nonlinear attitude reference model is as follows.

$$\dot{\omega}_m = {J_D}^{-1}\tau_{bl} - {J_D}^{-1}\omega_{mx}J\omega_m$$

$$\dot{\omega}_m = {J_D}^{-1}(\omega_x J\omega + J(\dot{\omega}_d + \omega_x\omega_d - k_q \log_v {q_e}^+ - k_w\omega_e)) - {J_D}^{-1}\omega_{mx}J\omega_m$$

A linear time variant (LTV) model that could mimic the behaviour of the non-linear attitude reference model in 4.38 can be constructed as follows.:

$$\dot{\omega}_m = {J_D}^{-1}\omega_x J_D\omega + {J_D}^{-1}J_D\dot{\omega}_d + {J_D}^{-1}J_D\omega_x\omega_d - {J_D}^{-1}J_D k_q \log_v {q_e}^+ - {J_D}^{-1}J_D k_w\omega_e - {J_D}^{-1}\omega_{mx}J_D\omega_m$$

$$\dot{\omega}_m = {J_D}^{-1}\omega_x J_D\omega + \dot{\omega}_d + \omega_x\omega_d - k_q \log_v {q_e}^+ - k_w(\omega - \omega_d) - {J_D}^{-1}\omega_{mx}J_D\omega_m$$

$$\dot{\omega}_m = \dot{\omega}_d + \omega_x\omega_d - k_q \log_v {q_e}^+ - k_w\omega + k_w\omega_d$$

But, $\omega_x\omega_d = -\omega_{dx}\omega$. Then,

$$\dot{\omega}_m = -\omega_{dx}\omega - k_w\omega + \dot{\omega}_d - k_q \log_v {q_e}^+ + k_w\omega_d$$

$$\dot{\omega}_m = -(K_w I + \omega_{dx})\omega + \dot{\omega}_d - k_q \log_v {q_e}^+ + k_w\omega_d$$

Then a linear time variant model can be constructed by formulating the time dependent system matrix, $A(t)$ and the reference, r as follows.

$$A_m(t) = -K_w I - \omega_{d\times} \tag{C.1}$$

$$r = -k_w\omega_d + \dot{\omega}_d - k_q \log {q_e}^+ \tag{C.2}$$

Appendix D

Stability Analysis of the LTV Model

The linear time variant model constructed to mimick the behaviour of the non-linear attitude subsystem of the quad-rotor is as follows:

$$\dot{\omega_B} = A_m(t)\omega_B + r$$

where the system matrix $A_m(t)$ is found to be as follows.

$$A_m(t) = -K_w I - \omega_{dx}$$

where,

$$\omega = \begin{bmatrix} p_d \\ q_d \\ r_d \end{bmatrix} \quad , \quad P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \quad \text{,and}$$

$$Q = \begin{bmatrix} q_{11} & 0 & 0 \\ 0 & q_{22} & 0 \\ 0 & p_0 & q_{33} \end{bmatrix}$$

The stability analysis of the LTV model can be analysed using the Lyapunov equation for linear systems as follows:

$$A^T P + PA = -Q \tag{D.1}$$

where in this section, $A_m(t)$ is written as A for simplicity purposes only. Then,

$$\begin{bmatrix} -k_w & -r_d & q_d \\ r_d & -k_w & -p_d \\ -q_d & p_d & -k_w \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} -k_w & r_d & -q_d \\ -r_d & -k_w & p_d \\ q_d & -p_d & -k_w \end{bmatrix} = - \begin{bmatrix} q_{11} & 0 & 0 \\ 0 & q_{22} & 0 \\ 0 & p_0 & q_{33} \end{bmatrix}$$

$$\begin{bmatrix} -k_w p_{11} - r_d p_{21} + q_d p_{31} - k_w p_{11} - r_d p_{12} + q_d p_{13} & -k_w p_{12} - r_d p_{22} + q_d p_{32} + r_d p_{11} - k_w p_{12} - p_d p_{13} & -k_w p_1 - r_d p_{23} + q_d p_{33} - q_d p_{11} + p_d p_{12} - k_w p_{13} \\ r_d p_{11} - k_w p_{21} - p_d p_{31} - k_w p_{21} - r_d p_{22} + q_d p_{23} & r_d p_{12} - k_w p_{22} - p_d p_{32} + r_d p_{21} - k_w p_{22} - p_d p_{23} & r_d p_1 - k_w p_{23} - p_d p_{33} - q_d p_{21} + p_d p_{22} - k_w p_{23} \\ -q_d p_{11} + p_d p_{21} - k_w p_{31} - k_w p_{31} - r_d p_{32} + q_d p_{33} & -k_w p_{12} + p_d p_{22} + -k_w p_{32} + r_d p_{31} - k_w p_{32} - p_d p_{33} & -k_w p_1 - +p_d p_{23} - k_w p_{33} - q_d p_{31} + p_d p_{32} - k_w p_{33} \end{bmatrix} = - \begin{bmatrix} q_{11} & 0 & 0 \\ 0 & q_{22} & 0 \\ 0 & p_0 & q_{33} \end{bmatrix}$$

The above matrix representation yields nine simultaneous equations to be solved to obtain the nine elements of the matrix P which yield the following

$$\begin{aligned} P_{12} = P_{13} = P_{21} = P_{23} = P_{31} = P_{32} = 0 \\ -2k_w P_{11} = q_{11} \\ -2k_w P_{22} = q_{22} \\ -2k_w P_{22} = q_{33} \end{aligned} \tag{D.2}$$

Choosing the matrix Q as follows, simplifies the solution resulting in a simpler expression for P.

$$Q = \begin{bmatrix} q_{11} & 0 & 0 \\ 0 & q_{22} & 0 \\ 0 & 0 & q_{33} \end{bmatrix} = 2 \begin{bmatrix} k_w & 0 & 0 \\ 0 & k_w & 0 \\ 0 & 0 & k_w \end{bmatrix}$$

Then,

$$\begin{aligned} -2k_w P_{11} = -2k_w \qquad & P_{11} = 1 \\ -2k_w P_{22} = -2k_w \qquad & P_{22} = 1 \\ -2k_w P_{33} = -2k_w \qquad & P_{33} = 1 \end{aligned}$$

It's is seen that the matrix P which is the solution to the Lyapunov equation of Eqn: D.1 is found to be an identity matrix for all values of the time variant system matrix $A(m)$. Hence, it's positive definiteness makes the constructed LTV model to be stable for any bounded input.