DYNAMICS AND CONTROL OF MINI AERIAL VEHICLE USING MODEL PREDICTIVE CONTROL

BY

MAIDUL ISLAM

INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

2018

DYNAMICS AND CONTROL OF MINI AERIAL VEHICLE USING MODEL PREDICTIVE CONTROL

BY

MAIDUL ISLAM

A thesis submitted in fulfilment of the requirement for the degree of Masters of Science in Engineering

Kulliyyah of Engineering
International Islamic University Malaysia

DECEMBER 2018

ABSTRACT

Nowadays Mini Aerial Vehicles (MAVs) are popular in many areas such as aerial photography, inspection, surveillance and search and rescue missions in complex and dangerous environments due to their low cost, small size, superior mobility, and hover capability. Multifarious applications of MAVs inspire researchers to concentrate on different types of controllers like linear, nonlinear or learning-based. The attention of this work is to design a robust controller and to develop an accurate mathematical model of Quadrotor, a type of MAV as it behaves roughly in uncertain environments. Quadrotor is an under-actuated and highly nonlinear system with six degrees of freedom (DOF). The mathematical model of quadrotor is derived based on Newton-Euler method that includes aerodynamic drag and moment that are sometimes overlooked in literatures. For higher precision modelling, model uncertainties are also included in the system. In addition, the kinematic model is derived utilizing Euler angles and Quaternion methods. Quaternion approach has the advantage of singularity free orientation while Euler angles are easy to visualize. This work investigates the performance of three different controllers which includes Proportional-Integral-Derivative (PID), Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) based on several performance evaluation factors. PID offers fast response to the system comparing to other controllers although choosing proper gain is challenging for PID. However, it cannot handle directly under-actuated system and due to the fact, some states are required to be decoupled. LQR ensures fast response and can deal with Multiple Input Multiple Output (MIMO) system at the same time. The main drawback of the LQR controller is its incapability of dealing with steady-state error. Conversely, MPC has the functionalities of dealing with MIMO system with constraints and uncertainties while other controllers fail. The performance of the controllers are presented based on tracking accuracy using Root Mean Square Error (RMSE) method and control stability using control input norm method. MATLAB and Simulink environment is considered to carry out the simulations. Based on simulated experiments, it is found that MPC could track the trajectories more accurately with stable control effort comparing to PID controllers and LQR.

خلاصة البحث

في الوقت الحاضر ، انتشرت المركبات الهوائية المصغرة (MAVs) في العديد من المناطق لأغراض التصوير الجوي والتفتيش والمراقبة ومهام البحث والإنقاذ في البيئات المعقدة والخطرة بسبب تكلفتها المنخفضة وصغر حجمها وحركتها الفائقة وقدرتها على التحليق . تلهم التطبيقات المتنوعة من MAVs الباحثين للتركيز على أنواع مختلفة من وحدات التحكم مثل الخطية أو غير الخطية أو القائمة على التعلم . هدف هذا العمل هو تصميم وحدة تحكم قوية وتطوير نموذج رياضي دقيق من Quadrotor ، وهو نوع من MAV تستخدم عادة في بيئات غير محددة. Quadrotor هو نظام غير سهل التحكم وغير خطي بامتياز مع ست درجات من الحرية (DOF) . تم اشتقاق النموذج الرياضي لرباعي الدوران على أساس طريقة نيوتن-أويلر التي تشمل السحب الهوائي و العزم الهوائي التي يتم تجاهلهما أحيانًا في الدراسات السابقة. من أجل تصميم نموذج عالي الدق ، يتم تضمين نماذج غير محدودة أيضًا في النظام . بالإضافة إلى ذلك ، يتم اشتقاق النموذج الحركي باستخدام زوايا أويلر وطرائق Quaternion. يتميز نهج Quaternion بميزة التوجه الحر للخلية بينما يسهل تصور زوايا أويلر. يدرس هذا العمل أداء ثلاث وحدات تحكم مختلفة تتضمن Proportional-Integral-Derivative (PID) ، و Linear Quadratic Regulator (LQR) و Model Predictive Control (MPC) على أساس العديد من عوامل تقييم الأداء. يقدم PID استجابة سريعة للنظام مقارنة مع وحدات التحكم الأخرى على الرغم من أن اختيار المعاملات الصحيحه يمثل تحديًا لـ PID. ومع ذلك ، فإنه لا يمكن التعامل مع نظام التشغيل غير المباشر مباشرة ، وبسبب حقيقة أن بعض الحالات لا بد من فصلها. تضمن LQR الاستجابة السريعة ويمكن أن تتعامل مع نظام الإدخال المتعدد للإخراج المتعدد (MIMO) في نفس الوقت . العيب الرئيسي لجهاز التحكم LQR هو عدم قدرته على التعامل مع الخطأ المستقر. على العكس ، لدى MPC وظائف التعامل مع نظام MIMO مع وجود قيود وشكوك بينما تفشل وحدات التحكم الأخرى. يتم تقديم أداء وحدات التحكم استنادًا إلى دقة التتبع باستخدام طريقة Root Mean Square Error (RMSE) و ثبات التحكم باستخدام طريقة معيار إدخال التحكم. يعتبر MATLAB و SIMULINK بيئة لتنفيذ عمليات المحاكاة. استنادًا إلى تجارب المحاكاة ، تبين أن MPC يمكنها تتبع المسارات بشكل أكثر دقة مع جهد تحكم مستقر مقارنة مع وحدات تحكم PID وLQR .

APPROVAL PAGE

I certify that I have supervised and read this study and that in my opinion, it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Science

…………………………………..
Mohamed El Sayed Aly Abd El Aziz Okasha
Supervisor

…………………………………..
Moumen Mohammed Idres
Co-Supervisor

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a thesis for the degree of Master of Science

…………………………………..
Waleed Fekry Faris
Internal Examiner

…………………………………..
Hishamuddin Jamaluddin
External Examiner

This thesis was submitted to the Department of Mechanical Engineering and is accepted as a fulfilment of the requirement for the degree of Master of Science

…………………………………..
AKM Mohiuddin
Head, Department of Mechanical Engineering

This thesis was submitted to the Kulliyyah of Engineering and is accepted as a fulfilment of the requirement for the degree of Master of Science

…………………………………..
Erry Yulian Triblas Adesta
Dean, Kulliyyah of Engineering

DECLARATION

I hereby declare that this dissertation is the result of my own investigations, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole for any other degrees at IIUM or other institutions.

Maidul Islam

Signature .... Date ....

ACKNOWLEDGEMENTS

Firstly, it is my utmost pleasure to dedicate this work to my dear parents and my family, who granted me the gift of their unwavering belief in my ability to accomplish this goal: thank you for your support and patience.

I wish to express my appreciation and thanks to those who provided their time, effort and support for this project. To the members of my dissertation committee, thank you for sticking with me.

Finally, a special thanks to Dr. Mohamed Okasha and Dr. Moumen for their support, encouragement and leadership, and for that, I will be forever grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

1.1 OVERVIEW

Unmanned Aerial Vehicles (UAVs) have achieved a great interest because of multifarious applications and the advancement of sensing and actuating technologies have expedited it more. Especially the interest is increasing among VTOLs (Vertical Take Off and Landing) that are mostly used for monitoring and exploration of any area.

A quadrotor is a vehicle of four rotors with a cross linked structure and considered as a VTOL UAV. The body of a quadrotor normally contains a power source, some sensors (i.e. GPS, altitude sensor etc.) and controlling equipment (i.e. Arduino, Raspberry-P, APM, Naza etc.). The rotors mainly produce thrust and by varying, it can perform any sorts of movement of quadrotor such as pitch, roll, yaw and upward-downward. Noted that all the rotors can be independently operated using a controller.

Quadrotor is a 6DOF and highly nonlinear system. Along with that, it also a coupled under-actuated system that has only four input to control six states at the same time. Moreover, in outdoor applications, it becomes more challenging in presence of uncertainty to the system. As a result, it is cardinal to develop a robust controller that can handle the uncertainty that influences the quadrotor performance.

This work introduces two different orientation system, Euler angle and Quaternion to describe quadrotor kinematic model. Remarkably, quaternion orientation system draws attention because it ensures singularity-free flight in all situations unlike Euler angle (Fresk & Nikolakopoulos, 2013). Noted that most of the applications of quadrotor can be performed using Euler angle orientation system and due to the reason, Euler angle orientation system is considered in this work as well.

In this thesis, three commonly used controllers, PID, LQR and MPC have been investigated on quadrotor platform based on two performance evaluation factors, tracking accuracy and control effort efficiency. PID and LQR ensures fast response to the system while the controllers cannot deal with uncertainties to the system and offer the feature of constraints. Conversely, MPC is capable to deal model uncertainties with its predicting behaviour and offer constraints at both inputs and outputs. MPC responds slower than others do because its predicting feature requires high computation. However, it can be overcome nowadays using high computational processors.

MATLAB & Simulink environment has been used to investigate the performances of the controllers considering two trajectories, circular and helical with different environments. The goal of this work is to develop a quaternion based mathematical model and to design a robust controller that may assist the quadrotor to track the trajectories comparatively more accurately under different conditions with smooth movement.

1.2 STATEMENT OF THE PROBLEM AND ITS SIGNIFICANCE

Quadrotor is one of the most suitable platforms for inspection, surveillance and rescue mission in complex environment. Therefore, it must be capable to maintain its flight with higher precision in such an environment that is not free from disturbance. For indoor application, quadrotor does not need to face any uncertainty while outdoor applications are more uncertain and challenging sometimes.

Moreover, stability and manoeuvrability is one of most complicated part for autonomous flight control because of its fast and agile movement with unknown environment. During its fast movement, it may face singularity problem, also known as Gimbal lock. Therefore, a suitable orientation system is required to be developed that may resolve the singularity problem. However, the best fitted control system for quadrotor is still under development because every controller has some advantages and disadvantages. Balancing between these advantages and disadvantages widens the opportunity to work on different controllers. Necessarily based on the gravity of applications, a suitable controller is ought to be designed for quadrotor. Primarily, some features such as fast response from the system, design simplicity, working with multiple constraints at control inputs, disturbance rejection along with higher precision in tracking are mostly expected for the controllers of quadrotor.

In this thesis, quaternion orientation system has been adopted in mathematical design to ensure singularity-free flight. Besides, three different controllers i.e. PID, LQR and MPC have been investigated because of their some special features as aforementioned. MATLAB/Simulink has been chosen in order to design and investigate the performance of the controllers. In order to evaluate the performance of the controllers, two evaluation parameters such as control effort efficiency and tracking accuracy have been considered during simulations.

1.3 RESEARCH OBJECTIVES

The study aimed to achieve the following objectives:

1- To develop mathematical model considering both Euler angle and quaternion orientation.

2- To design PID, LQR and MPC controllers for quadrotor based on mathematical model.

3- To investigate the performance of the controllers based on evaluation parameters.

4- To choose and finalize the most suitable controller among the three controllers for quadrotor based on investigation.

1.4 UNMANNED AERIAL VEHICLE

Unmanned Aerial Vehicle (UAV), also known as drone or remotely piloted vehicle (RPV), is an aircraft that flies without on-board pilot. It can either be remotely controlled from another location (i.e. ground, space or another aircraft) or pre-programmed with complete autonomy (ICAO, 2011). In another literature, from military aspect it is defined as a remotely operated vehicles or missiles that take flight for long duration at high altitudes or short duration at low altitudes and equipped with necessary sensors for surveillance (Amir & Weiss, 2003).

American Institute of Aeronautics and Astronautics (AIAA) defines that UAV is an aircraft that is independent of human pilot and operated by an onboard flight controller or a remote flight controller. So, it covers all sorts of aerial vehicle that is pre-programmed for flight and can be operated without any human interference (Rosenberg, 2009).

1.5 HISTORY OF UAV

Tracing back to the history, UAV was introduced by Austrian Armies for the first time to the world in Venice at August 22, 1849. They launched unmanned balloons of 23 feet diameter from Austrian ship named “Volcano” with explosives to attack Italy (Twain, 2016).

1.5.1 Historical development in Military operations

For the first time in the history, during World War I (WWI), pilotless aerial vehicle, as a counterpart was guided to the target, against naval torpedo although it crashed in United States after a while. After 2 years, in September 12, 1916, Hewitt-Sperry Automatic Airplane, also known as “Flying Bomb” is considered another one of the earliest UAV took some successful flights as a prototype (Ahmad et al., 2013). In 1918, around twenty pilot-independent aircrafts namely “Bugs” were made test flight successfully to validate the idea of Automatic Airplane that was developed by Sperry Gyroscope Company (Yanushevsky, 2007).

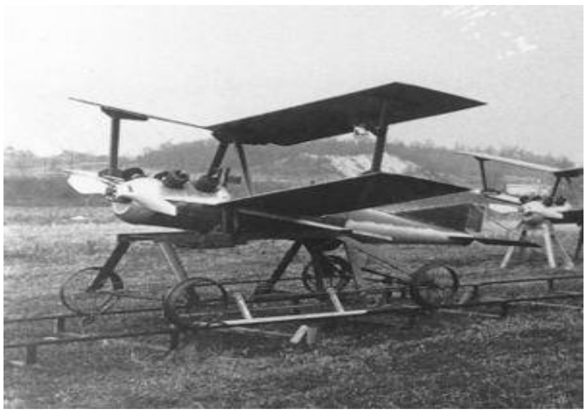


Figure 1.1 Kettering Bug (Schroer, 2003)

About a decade later of WWI, between 1930s and 1940s, Royal Air Force perfected a manned aircraft, Fairey Scout 111F and transformed it into remotely radio-controlled aircraft, “The Queen Bee”, that is considered as the first target drone in the history (Marshall et al., 2016). During World War II (WWII), at the end of October, 1944, U.S. demonstrated another remotely controlled aircraft, B-17 that was damaged the submarines of Germany devastatingly (Keane & Carr, 2013). On the same year, a couple of months later, U.S. Navy sent a troop of four TDR-1 drones which were loaded with 2000 lb bombs to attack on Japan (Lee, 2013).

After losing Vietnam War, U.S. fall in financial problems that made an impact on the research fund of UAV and it continued for almost a decade (Keane & Carr, 2013). However, after overcoming the financial crisis, U.S. started to work with Israel jointly on the development of small and cost-effective, motorcycle-powered engines UAVs that were equipped with video camera. Finally, Israel developed some new UAVs and used against Syria and Lebanon in 1982 (Karakoc et al., 2016; Rosenberg, 2009). Later on during 1990-1991, in Persian Gulf War, U.S. operated more than 300 flight operations of snowmobile powered engine drone with 17 feet wingspan, Pioneer, that were jointly developed by Israel and U.S. In Second Persian Gulf War, it was chosen as one of the primary weapon that took operational flight in Bosnia, Haiti and Somalia as well (Keane & Carr, 2013; Nonami, 2007)

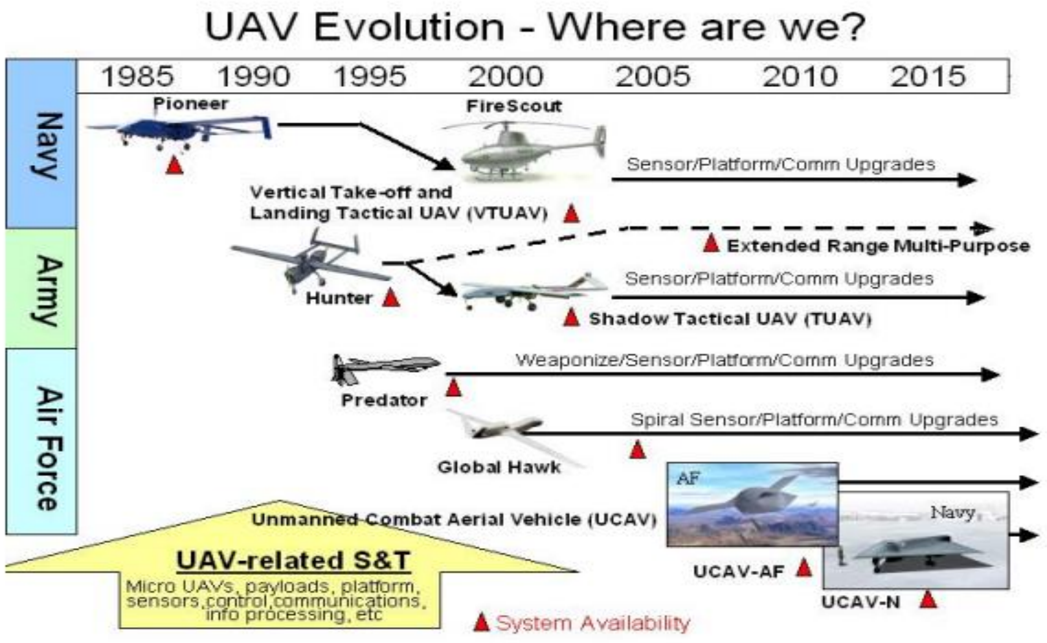

UAV Evolution - Where are we?
1985
1990
1995
2000
2005
2010
2015
Navy
Army
Air Force
Pioneer
FireScout
Sensor/Platform/Comm Upgrades
Vertical Take-off and Landing Tactical UAV (VTUAV)
Extended Range Multi-Purpose
Hunter
Sensor/Platform/Comm Upgrades
Shadow Tactical UAV (TUAV)
Weaponize/Sensor/Platform/Comm Upgrades
Predator
Spiral Sensor/Platform/Comm Upgrades
Global Hawk
AF
Navy
Unmanned Combat Aerial Vehicle (UCAV)
UAV-related S&T
Micro UAVs, payloads, platform, sensors, control, communications, info processing, etc
UCAV-AF
UCAV-N
System Availability

Figure 1.2 History of Military UAVs (Team, 2006)

1.5.2 Historical development in civil operations

Historically, the applications of UAVs were not restricted under only military operations rather civil operations though it was very well-known for military operations before. For civil operations, PA-30 Twin Comanche was debuted by NASA in 1967 that was controlled by ground station. During its test flight, a pilot was reserved on board to avoid any unexpected occurrence though the purpose was to fly it without pilot onboard (Koziol Jr, 1971). Later on, NASA initiated two different research programs such as Highly Maneuverable Aircraft Technology (HiMAT) and Drones for Aerodynamic and Structural Testing (DAST) program for civil operation (ElKholy, 2014; Murrow & Eckstrom, 1979). Another program, namely Environmental Research Aircraft and Sensor Technology (ERAST) was also initiated by NASA during 1990's. Interestingly, ERAST was developed to research, design and develop the low speed and inexpensive UAVs with the capability of long endurance at 60000 ft altitude. NASA claimed that it was a success of a long period effort for the development of aeronautical technologies and remotely controlled aircrafts of low cruise speed with long endurance. It was capable to analyze the environmental data for the assessment of climate changing and weather forecasting at the same time ("NASA Armstrong Fact Sheet: Altus II," 2014).

Altus II aircraft was developed by General Atomics Aeronautical Systems, Inc., under ERAST program as a variant of MQ-1 Predator that made the first flight in 1996. The flight was operated at 37000ft altitude with more than 26 hours by its single-stage turbocharger rear mounted engine. In the meanwhile, in 1990s, AeroVironment, Inc. was able to develop two solar powered UAVs, Helios and Pathfinder under their well-known EARST program. The main objective of the development of Helios and Pathfinders were to ensure the flight 100,000 ft with an endurance of 24 hours without any help of rudder (Gibbs, 2014a, 2014b). Surprisingly, the EARST program was terminated in 2003 although it could accomplish some successful projects (Wolfe, 2003).

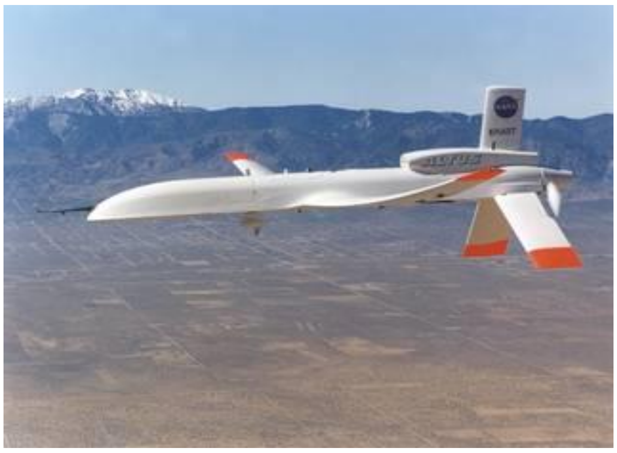


Figure 1.3 Altus II ("NASA Armstrong Fact Sheet: Altus II," 2014)

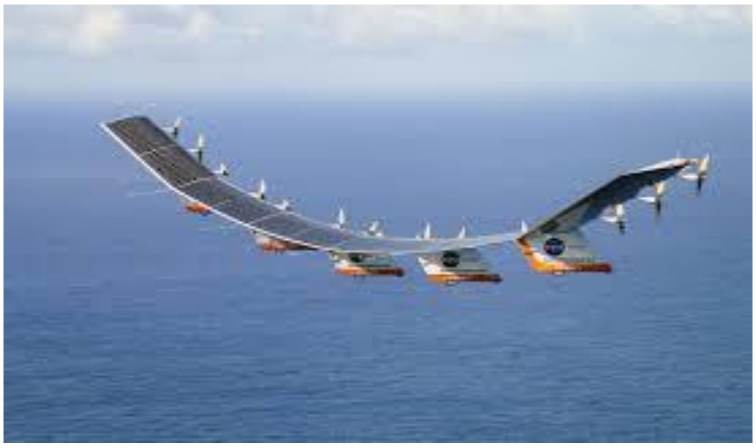


Figure 1.4 Helios (Conner, 2017)

1.6 APPLICATIONS OF UAV

Multifarious applications of UAVs are making itself more demanding and motivating researchers to find new application of UAVs. Mostly, UAVs are chosen as an alternative to perform some difficult, risky and dirty jobs instead. However, plethora of applications of UAVs have been introduced in defense and military purposes and a plenty of research funds are also being invested still for some more advanced applications of UAVs in these fields. In earlier history, military UAVs were mostly used for surveillance, reconnaissance and small strike while the recent UAVs have the capability to perform some more advanced and complex operation such as target detection and destruction, air combat, aerial transportation, anti-surface ship warfare, mine detection and defusing and so on. Apart from that, the applications of UAVs in civilian sector is getting wider and it is highly expected that in nearest future, UAVs will perform some sophisticated applications that were never expected before. Some common and potential civil and commercial applications of UAVs are:

1.6.1 Earth Science

UAVs can be used to observe any terrain or place from any side that helps to understand the condition precisely. Some similar missions are as follows (Team, 2006; Wegener et al., 2004):

i. Measurement of the deformation of earth's crust because of natural disasters like landslides, earthquake and volcanoes
ii. To have a study of transformations of gases and aerosols in cloud
iii. Observing the ozone chemistry in stratosphere
iv. Pollution of troposphere
v. Measurements of water vapor and total water
vi. Observations on coastal ocean
vii. Understanding about carbon cycle dynamics
viii. O_2, CO_2 and other gases measurements

ix. Studying on the breakup of glacier and ice sheet and measurement of ice sheet thickness.

1.6.2 Border patrol and security

Surveillance on border is a national security concern. Nowadays UAVs are used for patrolling and surveillances on border to identify and intercept any intruder or smuggler to trespass the border (Bolkcom, 2004; Girard et al., 2004; Haddal & Gertler, 2010; Sözen, 2014).

1.6.3 Search and rescue

UAVs equipped with camera and microphone, can give information about the survivors after natural disasters and any crashes (Waharte & Trigoni, 2010).

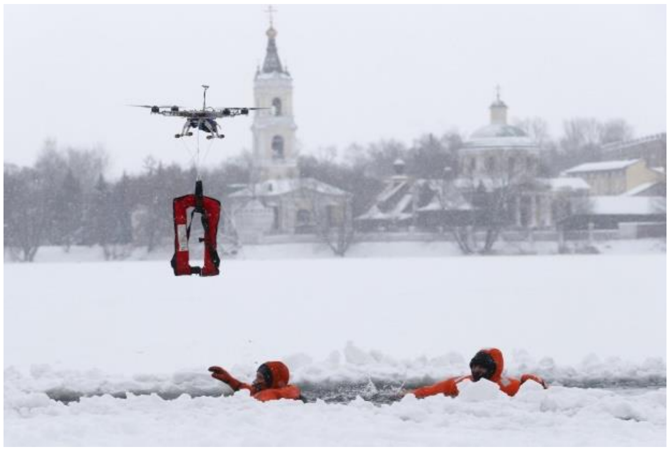


Figure 1.5 UAV in rescue mission (Cuthbertson, 2016)

1.6.4 Enforcement of law

UAVs are currently being used for some police works like chasing or traffic control in United States and Canada (Feng et al., 2013; Murphy & Cycon, 1999).

1.6.5 Industrial inspection and surveillance

Interestingly, in different industrial applications like gas and oil pipeline and nuclear reactor monitoring to ensure safety, security and maintenance, UAV is considered as a hassle free and more accurate alternative (Boudergui et al., 2011; Hausamann et al., 2005).

1.6.6 Research

UAVs play a very important role in research work and scientific projects as well. To observe any object from different angles without jerk, UAVs offer an amazing platform. Interestingly, some UAVs with noise suppression widens the horizon in research field when silent observation is very important. Some other versatile applications in research work are also performed by UAVs such as archaeological research, forestry, arctic research, marine research etc. (Casbeer et al., 2005; Hugenholtz et al., 2012; Runge et al., 2007; Saari et al., 2011; Tang & Shao, 2015; Themistocleous et al., 2014).

Figure 1.6 UAV in agricultural application (NASA, 2015)

1.6.7 Agricultural applications

Applications of UAVs in agriculture are quite vast nowadays. UAVs are used for detection of forest fire, monitoring harvesting sites, crop spraying, field mapping etc. (Gevaert et al., 2015; Grenzdörffer et al., 2008)

1.7 CLASSIFICATION OF UAV

A wide variety of metrics are used to classify UAVs that includes mass/weight, avionics complexity, speed, operational range, endurance, application, kinetic energy, operational area, operational failure consequences and other characteristics as well.

1.7.1 Classification based on Range and Endurance

A classification of UAVs in table 1.1 gives a comprehensive idea about different UAV system based on range and altitude.

Table 1.1: UAV classification on range and altitude (Van Blyenburgh, 1999; Weibel & Hansman, 2004)

Group	Category	Range (kilometer)	Altitude (meter)
Tactical UAVs	Micro	less than 10	250
	Mini	less than 10	350
	Close Range	10 to 30	3000
	Short Range	30 to 70	3000
	Medium Range	70 to 200	3000 to 5000
	Medium Range Endurance	more than 500	5000 to 8000
	Low Altitude Deep Penetration	more than 250	50 to 9000
	Low Attitude Endurance	more than 500	3000
	Medium Altitude Long Endurance	more than 500	5000 to 8000
Strategic UAVs	High Altitude Long Endurance	more than 1000	15000 to 20000
	Unmanned Combat Aerial Vehicle	close to 400	20000
Special Task UAVs	Lethal	300	3000 to 4000
	Decoys	Up to 500	50 to 5000

1.7.2 Classification based on configuration

Four different types of UAVs are available based on structural configuration such as fixed wing, rotary wing, flapping wings and blimps. Table 1.2 is offering the classification with their applications accordingly.

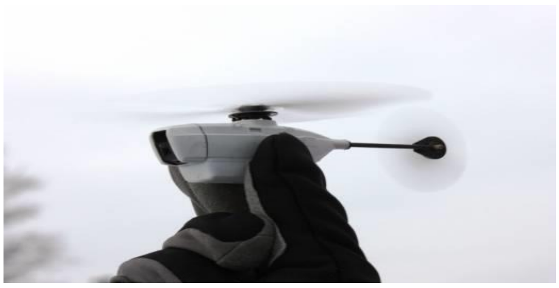


Figure 1.7 Micro UAV (Black Hornet Nano) (Yarrish, 2015)

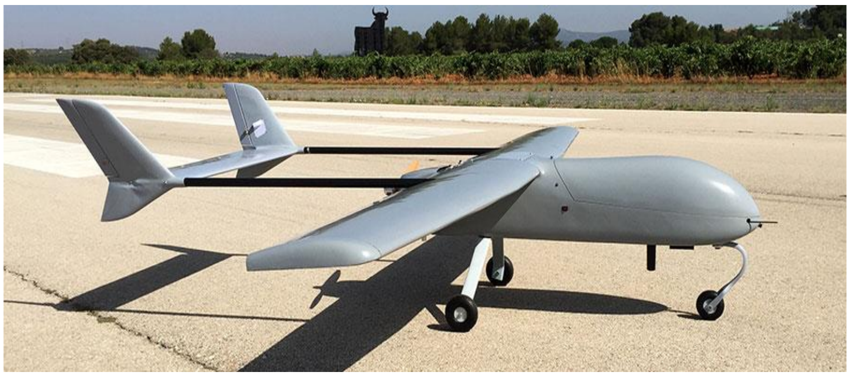


Figure 1.8 Fixed wing UAV (Embention, 2016)

Rotary wing UAVs are also classified by four different aerodynamic configurations (Bailey, 2012; ElKholy, 2014).

i. Single rotor UAVs: A main rotor is mounted at the top of these UAVs and a small rotor is placed at the rear to make them stable.

ii. Quadrotor UAVs: These types of UAV have four independent motors. They have two different configurations such as cross and plus configurations.

iii. Co-axial UAVs: Two rotors are mounted on a same shaft in opposite directions.

iv. Multi-rotor UAVs: Mostly, these UAVs have six or eight rotors.

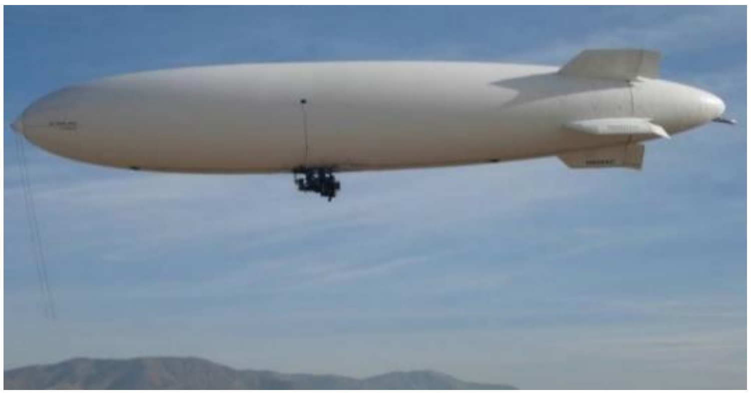


Figure 1.9 Blimps (Aria's Airship) (Staff, 2012)

Table 1.2: UAV classification based on aerodynamic configuration (Carrillo et al., 2012)

Category	Specifications	Applications
Fixed wing	long range, high altitude	Meteorological reconnaissance, environmental monitoring etc. (Carrillo et al., 2012)
Rotary wing	Vertical Take Off and Landing (VTOL), highly maneuverable	Search and rescue, monitoring, agricultural applications, inspection, law enforcement etc. (Chapman, 2017)
Flapping wing	VTOL, very low endurance, low power consumption, low payload	Surveying remote area, surveillance and safety of airport (Kamps, 2017; McDonald, 2016)
Blimps	Large in size, long endurance, low speed	Covering any event, advertising and transportation of heavy loads (Yoshimoto & Hori; Zhang & Kovacs, 2012)

1.7.3 Classification based on autonomy

UAVs are also can be classified according to the autonomy level. There are ten different autonomy levels are mentioned here in Table 1.3 (Clough, 2002; Cook & Das, 2004; Valavanis & Vachtsevanos, 2014).

Table 1.3 Classification on the basis of autonomy

Autonomous Control Level	Description
10	Completely autonomous
9	Battlespace swarm cognizance
8	Battlespace cognizance
7	Battlespace knowledge
6	Real-time multi-vehicle cooperation
5	Real-time multi-vehicle coordination
4	Fault/Event adaptive vehicle
3	Robust response to real-time faults/events
2	Changeable mission
1	Execute preplanned mission
0	Remotely piloted vehicle

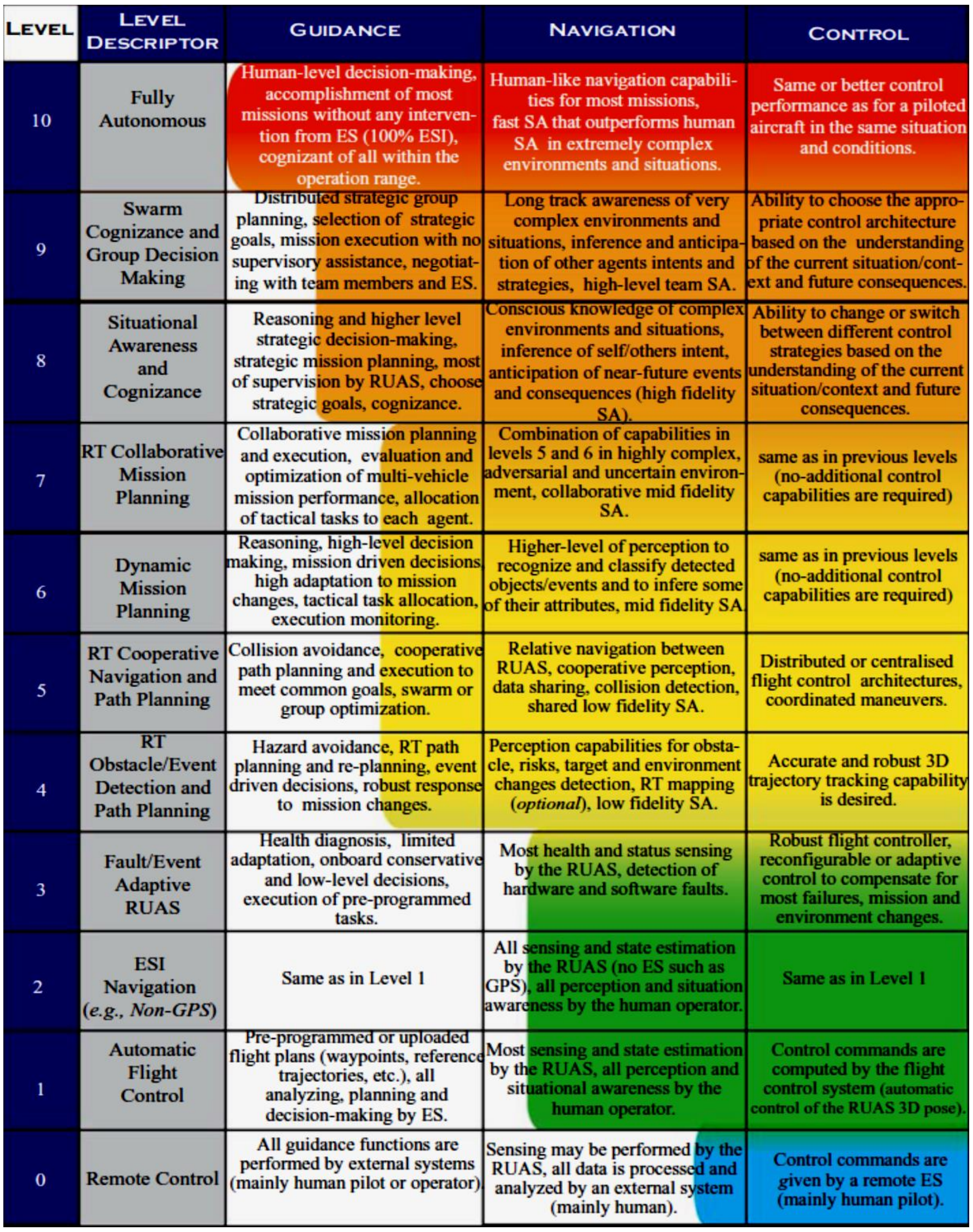
1.8 QUADROTOR

For the first time in history, an unmanned helicopter was designed and developed by a French Scientist Charles Richet but it did not take flight. Finally, Louis Breguet, a student of Charles Richet and his brother, Jacques could successfully develop a human carrying quadrotor for the first time in 1907 named Breguet -Richet Gyroplane No. 1 that took its flight successfully (Leishman, 2002).

Another French engineer Étienne Oehmichen took his first flight in 1924 and he crossed a distance of 360 m that is considered as the second flight of a quadcopter in the history (Esteves, 2014; Spooner, 1923).

1.8.1 Concept

The quadcopter is an aircraft of a rigid cross-linked structure that has four independent DC motors with propellers. In quadrotor, the directions of opposite rotors are always either clockwise or counter-clockwise. In Figure 1.13, propeller 1 and 3 rotate counter-clockwise direction while propeller 2 and 4 rotate in clockwise direction.



Level	Level Descriptor	Guidance	Navigation	Control
10	Fully Autonomous	Human-level decision-making, accomplishment of most missions without any intervention from ES (100% ESI), cognizant of all within the operation range.	Human-like navigation capabilities for most missions, fast SA that outperforms human SA in extremely complex environments and situations.	Same or better control performance as for a piloted aircraft in the same situation and conditions.
9	Swarm Cognizance and Group Decision Making	Distributed strategic group planning, selection of strategic goals, mission execution with no supervisory assistance, negotiating with team members and ES.	Long track awareness of very complex environments and situations, inference and anticipation of other agents intents and strategies, high-level team SA.	Ability to choose the appropriate control architecture based on the understanding of the current situation/context and future consequences.
8	Situational Awareness and Cognizance	Reasoning and higher level strategic decision-making, strategic mission planning, most of supervision by RUAS, choose strategic goals, cognizance.	Conscious knowledge of complex environments and situations, inference of self/others intent, anticipation of near-future events and consequences (high fidelity SA).	Ability to change or switch between different control strategies based on the understanding of the current situation/context and future consequences.
7	RT Collaborative Mission Planning	Collaborative mission planning and execution, evaluation and optimization of multi-vehicle mission performance, allocation of tactical tasks to each agent.	Combination of capabilities in levels 5 and 6 in highly complex, adversarial and uncertain environment, collaborative mid fidelity SA.	same as in previous levels (no-additional control capabilities are required)
6	Dynamic Mission Planning	Reasoning, high-level decision making, mission driven decisions, high adaptation to mission changes, tactical task allocation, execution monitoring.	Higher-level of perception to recognize and classify detected objects/events and to infere some of their attributes, mid fidelity SA.	same as in previous levels (no-additional control capabilities are required)
5	RT Cooperative Navigation and Path Planning	Collision avoidance, cooperative path planning and execution to meet common goals, swarm or group optimization.	Relative navigation between RUAS, cooperative perception, data sharing, collision detection, shared low fidelity SA.	Distributed or centralised flight control architectures, coordinated maneuvers.
4	RT Obstacle/Event Detection and Path Planning	Hazard avoidance, RT path planning and re-planning, event driven decisions, robust response to mission changes.	Perception capabilities for obstacle, risks, target and environment changes detection, RT mapping (*optional*), low fidelity SA.	Accurate and robust 3D trajectory tracking capability is desired.
3	Fault/Event Adaptive RUAS	Health diagnosis, limited adaptation, onboard conservative and low-level decisions, execution of pre-programmed tasks.	Most health and status sensing by the RUAS, detection of hardware and software faults.	Robust flight controller, reconfigurable or adaptive control to compensate for most failures, mission and environment changes.
2	ESI Navigation (*e.g., Non-GPS*)	Same as in Level 1	All sensing and state estimation by the RUAS (no ES such as GPS), all perception and situation awareness by the human operator.	Same as in Level 1
1	Automatic Flight Control	Pre-programmed or uploaded flight plans (waypoints, reference trajectories, etc.), all analyzing, planning and decision-making by ES.	Most sensing and state estimation by the RUAS, all perception and situational awareness by the human operator.	Control commands are computed by the flight control system (automatic control of the RUAS 3D pose).
0	Remote Control	All guidance functions are performed by external systems (mainly human pilot or operator).	Sensing may be performed by the RUAS, all data is processed and analyzed by an external system (mainly human).	Control commands are given by a remote ES (mainly human pilot).

Figure 1.10 ACL of UAVs (Kendoul, 2012)

As it is aforementioned that it has 6 DOF, quadrotor movements are described based on three axes as X, Y and Z. In Figure 1.13, x, y and z represents the movement along X, Y, Z-axis of quadrotor from Earth fixed frame and ϕ, θ, ψ denotes rotations around X, Y, and Z-axis.

Besides three cardinal movements such as roll, pitch and yaw are considered to describe the attitude of a quadcopter. Roll (ϕ) is the rotation along X-axis that is achieved by increasing or decreasing the speed of motor 2 and 4 while pitch (θ) is achieved by trade-off between motor 1 and 3. Along with that, lateral acceleration and longitudinal acceleration is obtained respectively by changing ϕ and θ angle. Yaw (ψ) is the rotation around Z axis and it is achieved by balancing the speed of the motor pair (1, 3) and (2, 4) simultaneously.

Figure 1.11 Breguet -Richet Gyroplane No. 1 (Leishman, 2002)

However, the mathematical model of quadrotor adopts both kinematics and dynamics model to explain the movement of quadrotor. Kinematics describes the motion of a body without considering any torque or forces on it while dynamics describes the motion considering torques and forces on the body. Moreover, the dynamics of quadrotor entails rotational and translational motion as Newton-Euler equations (Bresciani, 2008).

The rotational movements like roll, pitch and yaw are generally described in Euler angle representation system. Interestingly, the orientation of any object can easily be visualized in three-dimensions (3D) space though at some specific orientation it cannot be any more explainable that is widely known as “Gimbal lock”. It happens because of the singularity between two axes (Carino et al., 2015a; Swamp, 2016). In contrast, Quaternion, a hyper-complex numbering approach of four values, can overcome the problem though it is not so intuitive as Euler angler orientation is (Carino et al., 2015a; Fresk & Nikolakopoulos, 2013). However, in methodology chapter, both the Euler angle and quaternion approach will be discussed in detail.

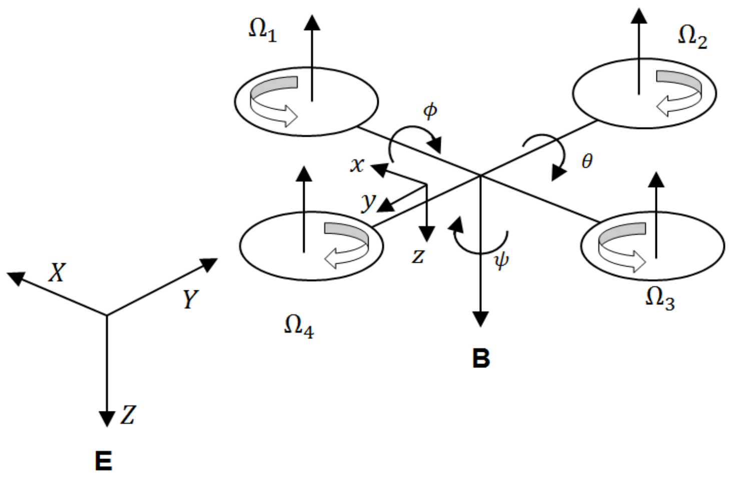


Figure 1.12 Configuration of quadrotor whereas B and E denotes Body fixed frame and Earth fixed frame respectively

1.8.2 Some features of quadrotor

Some special features of quadrotor are mentioned in the section that inspires to choose quadrotor platform for this thesis. The features are introduced as following (Bouabdallah & Siegwart, 2005; Hoffmann et al., 2007; Hou et al., 2010; Li & Li, 2011):

i. Rotor mechanics of quadrotor is simpler than helicopter because helicopter has variable pitch and quadrotor has fixed pitch where quadrotor approaches the functionalities of variable pitch by changing the speed of the rotors.

ii. As quadcopter has symmetrical configuration of rotor, it faces less gyroscopic effects comparing to helicopter.

iii. Quadrotor generates thrust by using four rotors with the propellers of small diameter where a helicopter produces the same thrust by using a long diameter propeller. As a result, a quadcopter needs a smaller area for its flight comparing to a helicopter.

iv. Since the wing of a quadrotor is normally enclosed within a frame, it has low risk to face any collision during its operation.

1.9 RESEARCH METHODOLOGY

This particular section carries out with an overview on the complete work with a view to achieving the aforementioned objectives. At first, a good review with proper analysis on previous works can give a clear idea about the difficulties, short-comings and vision for the works. Then some feasible solutions can be offered against some available problems in the literatures. As the research work particularly focuses on the controller and orientation system of Quadrotor, the review will take place only on these certain areas.

As it has already mentioned before about the limitations of Euler angle orientation system, Quaternion has been adopted in this work as a feasible solution for orientation system.

A suitable controller is another challenge among the control designers for quadrotor. Researchers investigated exceedingly on quadcopter control problem using various control techniques such as Proportional-Integral-Derivative (PID), Linear Quadratic Regulator (LQR) and H-infinity and these are considered as linear control technique. In the meanwhile, some well-known nonlinear control techniques such as

Backstepping, Feedback Linearization and Model Predictive Control (MPC) are being applied on quadcopter as well.

MPC achieves popularity greatly in industry because of its constraint handling capability as well as dealing with uncertainty to the system. Moreover, its predictive behavior, simplicity in tuning and dealing with multi-variable capability create some additional interest to the researchers.

In this work, PID, LQR as well as MPC approaches have been designed for quadrotor trajectory tracking under different environments. MATLAB and Simulink environment has been considered in order to evaluate the aforementioned capabilities of MPC approach considering different trajectories tracking considering two different orientation systems (i.e. Euler and Quaternion). Finally, a report will be submitted that necessarily may include system design, controllers design and evaluation of the designed controllers on the basis of some parameters in order to offer the best suited controller for quadrotor.

1.10 SCOPE AND LIMITATIONS

This study will help to choose a robust controller for quadrotor based on some performance indexes. In addition, two different orientation system have been applied to quadrotor platform in order to overcome some application limitations. For example, some applications require the quadrotor to move almost vertical to its body and that cause gimbal lock problem and hence, quaternion can be a solution to overcome such a problem. This thesis work has addressed some limitations of the PID and LQR controller as aforementioned that can be overcome by MPC controller mostly.

The validation of mathematical model design and controller design are performed based on simulation. However, it is recommended to validate the performance through experimental results. In addition, here, only position (x, y, z) tracking is considered while angular movements are considered for future work. Therefore, the tracking performance of the controllers has been evaluated based on only position. Furthermore, some assumptions has been appraised for quadrotor model design such as structural rigidity and symmetry, propellers’ rigidity and the coincidence of body fixed frame with centre of gravity of the quadrotor.

1.11 OUTLINE

This dissertation is organized in five chapters as follows.

An overview of the UAV, its classification and applications and quadrotor and its concept are illustrated in **Chapter one**. The problem statement with its significance, objectives, research methodology and scope of this dissertation are also discussed.

Chapter two describes a literature review on the commonly used control techniques for quadrotor and a comparative discussion among the controllers.

Chapter three presents the mathematical modeling and designing of quadrotor, rotor dynamics, state space model along with controller design.

Chapter four represents the discussion on the experimented results that take place at MATLAB and Simulink environment.

Chapter five will culminate at conclusion with some analogies based on findings, future works and suggestions.

CHAPTER TWO

LITERATURE REVIEW

2.1 QUADROTOR CONTROL

Technological advancement in Micro Electronics Mechanical System (MEMS) especially in sensors and microcontrollers motivates researchers greatly to work on quadrotor. Plenty of research works have taken places on quadrotor because of its versatile applications and some of its features such as simplicity to build, compactness in size and easier maneuverability. Among the works, some research works have been conducted on designing control techniques of quadrotor. Researchers from both robotics and control system get attracted to the control system because of ample of opportunities for developing new control algorithms. Different types of control techniques have been considered to achieve certain performances and complete missions. The existing control techniques can be categorized into three different control techniques such as linear, nonlinear and learning based control system (ElKholy, 2014; Junior et al., 2013; Kendoul, 2012).

2.1.1 Linear Control Techniques

LQR control, PID control, H_∞ algorithm and gain scheduling are the most commonly and conventional applied linear control techniques. In early 1970s, a full scale helicopter, CH-53A could achieve waypoints autonomously using a classical linear controller (Kendoul, 2012).

2.1.1.1 PID and LQ control

(Bouabdallah et al., 2004) compared the performance between PID and LQR control techniques on micro quadcopter and showed the system was stabilized around the hover position. However, in that study it is also found that at different operating points, PID offered poor performance and LQR showed steady-state error in an environment with disturbance. (Kodgirwar et al., 2014) used a complementary filter with PID controller that smoothed the feedback from gyroscope and accelerometer in order to achieve accurate roll and pitch angles. (Joyo et al., 2013) stabilized quadrotor around certain perturbed conditions using auto-tuned PID with extended Kalman Filter. In that study, extended Kalman Filter was applied to deal with model uncertainty to the system. (Argentim, 2013) compared the performances among a classical LQR, a PID tuned LQR and Absolute Error (ITAE) tuned PID. In the work, PID tuned LQR controller was found robust and simply applicable while the classical PID gave faster responses and insignificant robustness. (Cowling et al., 2007) applied a LQR controller on quadrotor platform to track quasi-optimal trajectories and finally validated the accuracy of the controller considering constraints and wind-gust at system inputs using optimal real time trajectories.

2.1.1.2 H_∞

H_∞ is a control approach that is normally applied to deal with imperfections of the system. The objective of the control approach is to attain a bounded ratio of two elements such as cost variable energy and disturbance signal energy (Raffo et al., 2011; Schaft & Arjan, 2000). (Araar & Aouf, 2014) designed LQR and H_∞ control approaches to track the trajectory under wind-gust condition. In that work, they demonstrated LQR successfully tracked the trajectories and H_∞ was able to deal with unmodelled nonlinearities like external disturbances (i.e. wind or gust) to the system. (Sorensen, 2010) presented a fully linear H_∞ control approach that achieved satisfactory performance in simulated results though it failed to stabilize the real-time hardware in presence of model uncertainty.

H_∞ control technique is advantageous when the system is multivariable with cross-coupled. Notwithstanding the controller requires a well-developed mathematical understanding and a well-designed dynamic model to achieve satisfactory performance (Cubillos et al., 2010).

2.1.1.3 Gain-scheduling

With a view to improving the capabilities of a linear model, a group of linear models are designed at some operating points. This approach is known as gain scheduling.

(Sadeghzadeh et al., 2012) compared the performance of PID and gain-scheduled PID controller at the time of dropping a payload of quadrotor where PID showed overshoot at that time while gain-scheduled PID showed promising performance. (Sawyer, 2015) demonstrated the performance of a gain-scheduled LQR control approach to track Lissajous and helix trajectories with changing yaw angle and experienced a satisfactory performance comparing to normal LQR approach.

2.1.2 Nonlinear Control Techniques

A linear model cannot represent a mathematically developed model accurately whereas it merely represents a certain nonlinear model around at a certain operating point. As a result, it only can perform well at those operating points at where it is linearized. It encourages researchers to think about a type of controller that will be able to deal with nonlinear model. Consequently, several control algorithms like feedback linearization, model predictive control, backstepping and sliding mode are most commonly used nonlinear control techniques.

2.1.2.1 Feedback linearization

Feedback linearization is also known as dynamic inversion approach. In feedback linearization, the states of a nonlinear model are transformed into a new type of coordinate system using nonlinear transformation approach where the model becomes linear. Subsequently, the linear model is again transformed back to the original coordinate system using linear tools via inverse transformation (Kendoul, 2012). (Bonna & Camino, 2015) used feedback linearization to track position and yaw where rotational and translational dynamics are linearized systematically. (Lanzon et al., 2014) designed a model for quadrotor in any rotor failure case that was controlled by feedback linearization approach. In that work, two different loops were used where a control loop was used for regulating trajectory and another was used for modifying desired trajectory that was shown successful in simulation environment.

2.1.2.2 Backstepping

Backstepping is known as recursive technique to control any under-actuated linear or nonlinear system. It disseminates controller into several steps and make the system stabilized progressively (Kendoul, 2012). (Madani & Benallegue, 2006) has applied the backstepping control approach based on Lyapunov theory to stabilize the quadcopter at desired position and attitude. In that work, an under-actuated subsystem was introduced to control horizontal position through roll and pitch angles. On the other hand, a fully-actuated subsystem was used to control vertical position through yaw and a propeller subsystem to control propeller forces. (Huo et al., 2014) applied an integral backstepping controller to stabilize quadrotor attitude. In that work, the controller could ensure promising performance of all the states of the system in presence of external disturbances to the system. (Fang & Gao, 2011) used adaptive integral backstepping control algorithm in order to ensure the robustness of the controller. This work considered online disturbances to the system and validated the robustness of the controller through proper trajectory following.

2.1.2.3 Sliding Mode

Sliding mode is a switching control technique. In this control technique, the system states are commanded towards a chosen desired surface known as sliding surface. The system states remain on surface with the help of a properly designed control law (Ben Ammar et al., 2016). (Xu & Ozguner, 2006) proposed a sliding mode control to stabilize under-actuated subsystem of the quadrotor with the help of a PID controller. They validated the robustness of the controller considering parametric uncertainties in the system. (Swamp, 2016) introduced a second order sliding mode control that was designed on the basis of Lyapunuv theory to stabilize the quadrotor. In the work, the second order sliding mode controller demonstrated promising results comparing to conventional sliding mode controller and ensures the robustness as well.

2.1.2.4 Model Predictive Control

Model Predictive Control (MPC) becomes one of the widespread controllers nowadays because of its functionalities like input and output constraints, dealing with disturbances, predictive behavior, simplicity in tuning and advance performance with multi-variables at the same time. MPC works on the base of optimization where cost function is minimized depending on the current control inputs and future time interval by handling the constraints of states and inputs (Kendoul, 2012; Bouffard, 2012). MPC controller is found as more effective and accurate than PID controller in industrial applications (Kozák, 2012). (Raffo et al., 2008) proposed a MPC to track the reference trajectory considering disturbances and nonlinear H-infinity to obtain the robustness of the system in quadrotor. (Alexis et al., 2010) applied MPC to track attitude reference under wind-gust condition of quadrotor and could achieve robust performance successfully. The work has successfully tracked the reference point by using a single MPC technique on the quadcopter platform that considers external disturbances in the system and constraints for the actuators saturation at control inputs. (Bouffard, 2012) used a new approach, Learning Based Model Predictive Control (LBMPC) in order to ensure robustness to the system. In that work, he demonstrated that the performance of the system can be improved by updating the model online which performs better than linear MPC.

2.1.3 Learning Based Control Techniques

Learning based controller is such a control technique that does not require accurate and precise dynamic model rather some trials and flight data for training the system to control a quadrotor (Kendoul, 2012). Some well-known control system like fuzzy logic, neural network and human based learning controller are considered under learning based controller. (Santos et al., 2010) developed an intelligent fuzzy controller that ensured promising performance in stability and precise movement of the system. The controller parameters tuning with the help of inter-dependent variables were the most successful part of the work. (Efe, 2011) could decrease the computational time and simplified the PID controller using Neural Network. (Pipatpaibul & Ouyang, 2013) compared PD Online Iterative Learning Control technique (ILC) with Switching PD ILC based on tracking performance in presence of model uncertainty. In that work, it is found that the tracking accuracy and disturbance rejection capability of Switching PD ILC is more satisfactory comparative to Online ILC which was evaluated by simulated results.

2.2 ADVANTAGES AND DISADVANTAGES OF DIFFERENT CONTROLLERS

In this section concentrates on an analysis of commonly used controllers in quadrotor like PID, LQR, SMC, Feedback Linearization, Backstepping and MPC based on their functional advantages and disadvantages. The classical PID controller is only applied with linear model. This controller gives the opportunity to design the controller according to the desired model performance. However, it becomes more challenging to design a well-performed PID controller when the model is nonlinear because the gain cannot be chosen in any more systematic way that classical PID controller requires. In addition, LQR also requires a linear model to get a proper controlled system and it can handle multiple input and output at the same time unlike PID controller. The main drawback comparing to PID is that sometimes it shows steady-state error because it does not offer any integral part (Argentim, 2013).

A systematic framework for modelling of a controller is the main advantage of feedback linearization. It is a well-performed controller when the difference between linear and nonlinear model is insignificant. However, it cannot guarantee the satisfactory response in presence of model uncertainties and offer the functionality of constraints handling as well. Hence, the robustness of this controller is not satisfactory always (Kurtz & Henson, 1998; Pop & Dulf, 2011; Zulu & John, 2016).

Backstepping is one of the mostly chosen nonlinear control techniques that requires a systematic procedure and follows recursive design methodology. It can cancel out the nonlinear terms in the system and as a result, it does not require precisely designed model unlike feedback linearization. It has the capability to overcome the mismatched perturbations and can attain the stability asymptotically. However, the main drawback of this controller is over-parameterization that implies it needs many parameters to give a satisfactory performance to the system that becomes sometimes very difficult to find out all the parameters accurately (Basri et al., 2014; Chung & Chang; Huo et al., 2014).

Sliding mode control (SMC) technique has achieved a great attention for designing robust controllers in high-order nonlinearity of any system under uncertainties. It is less sensitive in disturbances and parametric uncertainties that can ensure the robustness to the system. However, it offers chattering problem that happens because of continuous switching of controlled model. As a result, it may provoke energy loss, unmodeled dynamics and system instability that is hazardous for the system sometimes (Bendaas & Naceri, 2013; Levant, 2007; Runcharoon & Srichatrapimuk, 2013; Shtessel et al., 2014)

MPC has been used in different process of chemical industries and refineries for more than three decades. Currently researchers shows great interested to apply it in all type of complex controlling system because of its versatile capability as aforementioned (Bouffard, 2012).

2.3 CHAPTER SUMMARY

In this chapter, different types of control algorithm for quadrotor have been discussed. Different control techniques have their own specialties with their unique algorithms that depend on the applications of quadrotor. Finally, a comparative discussion of different controllers with their advantages and disadvantages have been carried out to attain a comprehensive and intuitive idea on controllers and their applications. From the review, it can be found out that the features of MPC like predicting behavior and multiple constraint handling are really attractive comparing to other controllers and can be considered as a suitable controller for quadrotor. Besides, it should be informed that tracking performance of MPC is slower because of high computation. However, with the blessing of modern technology, the drawback can be overcome using some advanced computational devices.

Table 2.1 A review of different controllers

Controllers	Advantages	Disadvantages
PID	Easy to choose gain; can overcome steady-state error.	Cannot handle constraints, noise and disturbance; can't deal with multiple inputs and outputs at the same time.
LQR	Can deal with multiple inputs and outputs	Sometimes fails to overcome steady-state error.
Feedback Linearization	Systematic model framework; well-performed when linear and nonlinear models are almost similar	Incapability of constraints handling and model uncertainties, poor robustness
Backstepping	Systematic and recursive designed; precisely designed model is not required; can handle nonlinearities to the system; can overcome mismatched perturbations and ensures stability.	Over-parameterization; difficult to choose proper parameters
SMC	Well-performed in high-nonlinearity; less sensitivity in disturbances and model uncertainties.	Chattering problem sometimes create system instability.
MPC	Predicts future behavior of the states; deal with multiple inputs and outputs at the same time; can handle constraints at inputs and outputs; can overcome noise and disturbances	Slow in tracking

CHAPTER THREE
METHODOLOGY

3.1 INTRODUCTION

This chapter describes mathematical modeling and controllers design of quadrotor. As orientation system, Euler angle and Quaternion representation system have been considered to derive the mathematical model. According to that, two different mathematical models has been developed here and then controllers have been designed based on the designed models with proper model verifications. The next chapter entails the performance of the controllers based on several factors to give a comprehensive idea about the designed controllers that will help to choose suitable controller further.

3.2 MATHEMATICAL MODEL

Mathematical model is required to develop the system and make the system compatible with controller. The mathematical model can be explained by kinematics and dynamics. Newton-Euler method has been considered to develop the mathematical model of quadrotor. In addition, aerodynamic drags and moments have been considered also in order to make the model more accurate that sometimes are neglected in other literatures.

3.2.1 Kinematic Model

Kinematic modelling is completely dependent on the coordinate system. In addition, it is required to explain the orientation of a rigid body with the help of a fixed coordinate system. In this section, two different orientation systems (i.e. Euler angle and

Quaternion) have been considered in order to describe the kinematic modelling respectively.

3.2.1.1 Euler Angle Representation

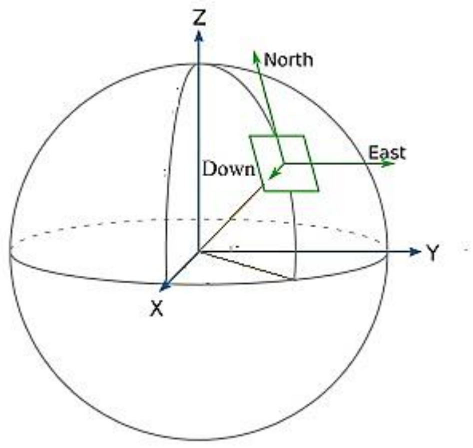


Figure 3.1 Coordinate system according to Euler angle

Figure 3.1 offers two different notation X-Y-Z and N-E-D. X-Y-Z illustrates the Earth fixed frame whereas N-E-D indicates North-East-Downward for the Body frame of quadrotor. In figure 1.15, Earth fixed frame, N-E-D is renamed as X-Y-Z in order to maintain the conventional naming. Here, it is considered that in body frame, center of gravity acts along Z axis direction and r is the distance between Earth fixed frame and body frame where $r = [x,\ y,\ z]^T$.

Euler Angle Rotation

Since the movement of quadrotor is expressed based on two different frame of references, a transformation matrix is necessarily developed to maintain a relation between these two frame of references. Here, R is considered as a transformation matrix that helps to determine the position and movement from Earth fixed frame to Body fixed frame.

Let, the quadrotor has changed its yaw of ψ angle, pitch of θ and roll of ϕ with respect to Earth fixed frame respectively. Hence, the following equations can be achieved.

$$R_\psi = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

$$R_\theta = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{3.2}$$

$$R_\phi = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \tag{3.3}$$

Therefore, the rotation matrix from Body frame to Earth fixed frame, R_{EB} can be achieved by the product of the aforementioned successive rotations as follows in equation (3.4) and (3.5). Noted that it is the most commonly used rotation matrix in different literatures (Bouabdallah, 2007; ElKholy, 2014; Lindblom & Lundmark, 2015).

$$R_{EB} = R_\phi R_\theta R_\psi \tag{3.4}$$

$$R_{EB} = \begin{pmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ c_\psi s_\phi s_\theta & s_\psi s\phi s_\theta + c_\theta c_\psi & s_\phi c_\theta \\ c\phi s_\theta c_\psi + s_\psi s\phi & c_\phi s_\theta s_\psi - s_\theta c_\psi & c_\phi c_\theta \end{pmatrix} \tag{3.5}$$

where, c, s and t denotes cos, sin and tan respectively.

Finally, the rotation matrix of quadrotor can be achieved that will describe the transformation from Earth fixed frame to Body frame as follows in equation (3.6) where subscript B and E denotes the position along axes in Body frame and Earth fixed frame respectively.

$$\begin{pmatrix} x_B \\ y_B \\ z_B \end{pmatrix} = R_{EB} \begin{pmatrix} x_E \\ y_E \\ z_E \end{pmatrix}$$

$$= \begin{pmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ c_\psi s_\phi s_\theta & s_\psi s\phi s_\theta + c_\theta c_\psi & s_\phi c_\theta \\ c\phi s_\theta c_\psi + s_\psi s\phi & c_\phi s_\theta s_\psi - s_\theta c_\psi & c_\phi c_\theta \end{pmatrix} \begin{pmatrix} x_E \\ y_E \\ z_E \end{pmatrix} \qquad (3.6)$$

Therefore, the inverse of R_{EB} will give the rotation matrix for a Body frame to Earth fixed frame. As R_{EB} is an orthogonal matrix, the inverse of this matrix and transpose of the matrix remain same.

$$R_{BE} = {R_{EB}}^{-1} = {R_{EB}}^{T}$$

$$= \begin{pmatrix} c_\theta c_\psi & c_\psi s_\phi s_\theta & c\phi s_\theta c_\psi + s_\psi s\phi \\ c_\theta s_\psi & s_\psi s\phi s_\theta + c_\theta c_\psi & c_\phi s_\theta s_\psi - s_\theta c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix} \qquad (3.7)$$

Euler Angle Dynamics

The transformation matrix is derived to formulate a relation between Earth fixed frame and Body frame. For instance, thrust forces are measured in Body frame while gravitational forces and the position of quadrotors are measured in Earth fixed frame. This transformation matrix helps to maintain relation between two frames of references along with to develop the dynamic model of the system also. Similarly, angular velocity of the quadrotor is measured on Body frame using on-board Inertial Measurement Unit (IMU). Hence, another transformation matrix R_r is required to make a relation between Euler rates, $\dot{\eta} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ and angular velocity of quadrotor, $\omega = [p,\ q,\ r]^T$ as follows (Islam et al., 2017; Sabatino, 2015).

$$\omega = R_r \dot{\eta} \qquad (3.8)$$

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi t\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \tag{3.9}$$

$$R_r = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -s\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{pmatrix} \tag{3.10}$$

3.2.1.2 **Quaternion**

Quaternion, a four-tuple orientation system is widely being used as an alternative of Euler angle orientation. In literatures, it is found that Euler angle shows singularities at some certain situations and it is failed to determine the accurate angle when any incremental changes take place over time in attitude (Kulumani & Lee, 2017). In contrary, Quaternion is more successful in both the situations comparing to Euler angle. Moreover, it is computationally cheaper, more stable and more efficient (Diebel, 2006; Fresk & Nikolakopoulos, 2013; Horn, 2001). A quaternion contains a scalar part or real number and a vector part or complex number part that is consists of three elements in the complex space as follows (Fresk & Nikolakopoulos, 2013)

$$q = q_0 + iq_1 + jq_3 + kq_4 \tag{3.11}$$

In different research works it is symbolizes as (Reyes-Valeria et al., 2013) follows.

$$q = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ \boldsymbol{q} \end{pmatrix} \tag{3.12}$$

Quaternion Rotation

According to Euler's rotation theorem, a rotation can be described by an angle α and a unit vector of three dimensions, e that can be expressed as $e = ie_1 + je_2 + ke_3$ where i, j and k symbolizes x, y and z axis (Carino et al., 2015b). Quaternion can be represent also as $q = cos\left(\frac{\alpha}{2}\right) + e\,sin\left(\frac{\alpha}{2}\right)$

The vector representation will be as follows:

$$q = \begin{pmatrix} cos\left(\frac{\alpha}{2}\right) \\ e\,sin\left(\frac{\alpha}{2}\right) \end{pmatrix} \tag{3.13}$$

Similar to Euler angles, Quaternion has also transformational matrix from Body frame to Earth fixed frame as follows (Kurtz & Henson, 1998).

$$Q_{BE} = \begin{pmatrix} 1 - 2({q_2}^2 + {q_3}^2) & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & 1 - 2({q_1}^2 + {q_3}^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & 1 - 2({q_1}^2 + {q_2}^2) \end{pmatrix} \tag{3.14}$$

Hence, the transformational matrix from Earth-fixed frame to Body frame as follows (Lindblom & Lundmark, 2015):

$$Q_{EB} = \begin{pmatrix} 1 - 2({q_2}^2 + {q_3}^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & 1 - 2({q_1}^2 + {q_3}^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & 1 - 2({q_1}^2 + {q_2}^2) \end{pmatrix} \tag{3.15}$$

Quaternion Dynamics

In order to define a complete orientation of quadrotor in space, Quaternion offers a rotational matrix that can represent the changes of orientation with respect to time alike Euler angle (Chung & Chang; Zulu & John, 2016).

$$\dot{q} = \frac{1}{2}\begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix}\begin{pmatrix} 0 \\ p \\ q \\ r \end{pmatrix} \tag{3.16}$$

3.2.2 Dynamic Model

The dynamic model of the quadrotor can be divided into translational motion and rotational motion. Translational motion is a consequence of some forces while rotational motion is a result of some torques which are generated by motor thrust. These two different types of motions are described as follows.

3.2.2.1 Translational Motion

The translational motion of a quadrotor is derived by Newton's second law and it is measured on Earth-fixed frame. This motion can be achieved by proper mathematical derivation among several forces like gravitational force (F_g), thrust forces (F_t), aerodynamic drag force (F_a) and disturbances (F_d) from surroundings as follows in equation (3.17). Noted that the generated force from a motor is considered as $F_i = k_f \Omega_i^2$ where Ω_i is denoted as angular velocity of i^{th} motor and k_f is symbolized as aerodynamic force constant (ElKholy, 2014).

$$m\ddot{r} = F_g + F_t + F_a + F_d \tag{3.17}$$

where,

$$F_t = \begin{pmatrix} 0 \\ 0 \\ -k_f(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{pmatrix}$$

$m =$ Quadrotor mass

$\ddot{r} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix}$ = Accelerations along axes

g = Gravitational acceleration = 9.81m/s^2

Equation (3.17) can be modified with help of equation (3.18) and becomes equation (3.19) as follows.

$$F_a = k_t \dot{r} \tag{3.18}$$

$$m \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} + R_{EB} \begin{pmatrix} 0 \\ 0 \\ -U_1 \end{pmatrix} - k_t \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} + F_d \tag{3.19}$$

where,

$F_t = R_{EB} \begin{pmatrix} 0 \\ 0 \\ -U_1 \end{pmatrix}$ = Thrust force produced by motors

$U_1 = k_f (\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$

$\dot{r} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}$ = Derivative of position vector of quadrotor with respect of time along axes

$k_t = \begin{pmatrix} k_{t_x} & 0 & 0 \\ 0 & k_{t_y} & 0 \\ 0 & 0 & k_{t_z} \end{pmatrix}$ = Aerodynamic drag force constant matrix

Similarly, in order to represent the equation (3.17) according to quaternion orientation system, it can be represented as follows (Lindblom & Lundmark, 2015).

$$m \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} + Q_{EB} \begin{pmatrix} 0 \\ 0 \\ -U_1 \end{pmatrix} - k_t \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} + F_d \tag{3.20}$$

3.2.2.2 Rotational Motion

Rotational equations of motion of a quadrotor are calculated using Newton-Euler method in Body frame. The generalized equation of rotational equations for quadrotor can be explained considering gyroscopic moment (M_g), moment on body frame (M_b), drag moment (M_a) and moment caused by disturbances (M_d).

$$I\dot{\omega} = -\omega \times I\omega - M_g + M_b + M_a + M_d \tag{3.21}$$

where,

$I\dot{\omega}$ and $\omega \times I\omega =$ the rate of change of angular momentum in the quadrotor body frame.

$\dot{\omega} = \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix}$ = Derivative of angular velocity of quadrotor with respect to time

$M_g = \omega \times \begin{pmatrix} 0 \\ 0 \\ I_r\omega_r \end{pmatrix}$ = Gyroscopic moment

$M_b =$ Moment acting on the body frame

$I_r =$ Rotors' inertia

$\omega_r =$ Rotors' relative speed

$= -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$

Matrix of Inertia

The inertia matrix is a square matrix 3 × 3 because of x, y and z-axes. As the quadrotor is considered symmetrical, the off diagonal elements are zero and it transforms into a 3 × 3 diagonal matrix as follows.

$$I_r = \begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix} \tag{3.22}$$

where, I_{xx}, I_{yy} and I_{zz} are the moment of inertia along x, y and z axes of body frame.

Gyroscopic Moment

Gyroscopic moment M_g is produced because of the rotation of motors that tries to spin the quadrotor along z-axis.

Moment Acting on the Body Frame

As the air density and maximum altitude of a quadrotor is limited, moment on individual motor is proportional to the square of motor. So the equation as follows:

$$M_i = k_M \Omega_i^2 \tag{3.23}$$

where, k_M = aerodynamic moment constant

Here, the moment of ith motor depends on the rotor speed of ith motor and its moment arm. Let, the required moment is now about x-axis. According to the right-hand rule, F_2 multiplied with the arm, l generates a negative moment while F_4 is multiplied with the arm l and generates positive moment. Therefore, the total moment can be defined as

$$\begin{aligned} M_x &= -F_2 l + F_4 l \\ &= -\, k_f \Omega_2^2 l + k_f \Omega_4^2 l \end{aligned} \tag{3.24}$$

If the required moment is considered about y-axis, according to the right-hand rule, F_1 creates a positive moment and F_3 creates a negative moment with the help of l similarly. Then the total moment about y-axis will be

$$\begin{aligned} M_y &= -F_3 l + F_1 l \\ &= -\, k_f \Omega_3^2 l + k_f \Omega_1^2 l \end{aligned} \tag{3.25}$$

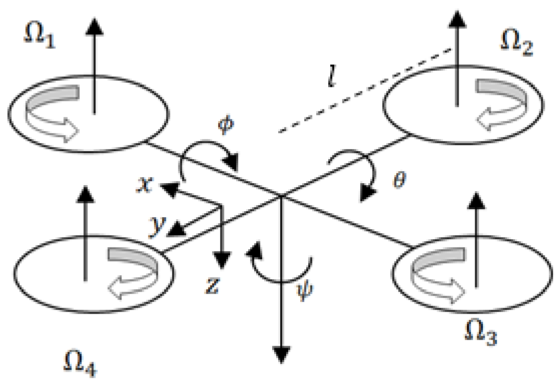


Figure 3.2 Forces and moments on Quadrotor

In case of z-axis, there is no moment that is generated by thrust force. However, according to the equation (3.23) and right hand rule, the total moment about z-axis will be expressed as:

$$\begin{aligned} M_z &= M_1 - M_2 + M_3 - M_4 \\ &= k_M \Omega_1^2 - k_M \Omega_2^2 + k_M \Omega_3^2 - k_M \Omega_1^2 \\ &= k_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_1^2) \end{aligned} \tag{3.26}$$

Therefore, from equation (3.24), (3.25) and (3.26), finally the moment, M_b can be represented in a matrix form as follows in equation (3.27)

$$M_b = \begin{pmatrix} k_f l(\Omega_4^2 - \Omega_2^2) \\ k_f l(-\Omega_3^2 + \Omega_1^2) \\ k_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_1^2) \end{pmatrix} \tag{3.27}$$

Drag Moment

Drag moment, M_g is produced because of air friction, as like drag force and it is proportional to angular speeds of the quadrotors as follows:

$$M_a = k_r \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{3.28}$$

where,

$$k_r = \begin{pmatrix} k_{r_x} & 0 & 0 \\ 0 & k_{r_y} & 0 \\ 0 & 0 & k_{r_z} \end{pmatrix} = \text{Aerodynamic drag coefficient matrix}$$

Hence, equation (3.21) can be represented as

$$I\dot{\omega} = -\omega \times I\omega - \omega \times \begin{pmatrix} 0 \\ 0 \\ I_r \omega_r \end{pmatrix} + \begin{pmatrix} lU_2 \\ lU_3 \\ U_4 \end{pmatrix} - k_r \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{3.29}$$

where,

$U_2 = k_f(\Omega_4^2 - \Omega_2^2)$

$U_3 = k_f(-\Omega_3^2 + \Omega_1^2)$

$U_4 = k_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_1^2)$

Finally the mathematical equation on Euler angle orientation can be developed by considering equation (3.9), (3.19) and (3.29) as follows:

$$\ddot{x} = \frac{-1}{m}[k_{t_x}\dot{x} + U_1(s\phi s\psi + c\phi c\psi s\theta)] \tag{3.30}$$

$$\ddot{y} = \frac{-1}{m}[k_{t_y}\dot{y} + U_1(s\phi c\psi - c\phi s\psi s\theta)] \tag{3.31}$$

$$\ddot{z} = \frac{-1}{m}[k_{t_z}\dot{z} - mg + U_1 c\phi c\theta] \tag{3.32}$$

$$\dot{\phi} = p + rc\phi t\theta + qs\phi t\theta \tag{3.33}$$

$$\dot{\theta} = qc\phi - rs\phi \tag{3.34}$$

$$\dot{\varphi} = r\frac{c\phi}{t\theta} + q\frac{s\phi}{c\theta} \tag{3.35}$$

$$\dot{p} = \frac{-1}{I_x}[k_{r_x}p - lU_2 - I_y qr + I_z qr + I_r q\omega_r] \tag{3.36}$$

$$\dot{q} = \frac{-1}{I_y}[-k_{r_y}q + lU_3 - I_x pr + I_z pr + I_r p\omega_r] \tag{3.37}$$

$$\dot{r} = \frac{-1}{I_z}[U_4 - k_{r_z}r + I_x pq - I_y pq] \tag{3.38}$$

Similarly, for quaternion, another mathematical model can be derived by adopting equation (3.16), (3.20) and (3.29) as follows:

$$\ddot{x} = \frac{-1}{m}[k_{t_x}\dot{x} + U_1(2q_0q_2 + 2q_1q_3)] \tag{3.39}$$

$$\ddot{y} = \frac{-1}{m}[k_{t_y}\dot{y} - U_1(2q_0q_2 - 2q_1q_3)] \tag{3.40}$$

$$\ddot{z} = \frac{-1}{m}[k_{t_z}\dot{z} - mg + U_1(2{q_0}^2 + 2{q_3}^2 - 1)] \tag{3.41}$$

$$\dot{q_0} = \frac{1}{2}[-pq_1 - qq_2 - rq_3] \tag{3.42}$$

$$\dot{q_1} = \frac{1}{2}[pq_0 - qq_3 + rq_2] \tag{3.43}$$

$$\dot{q_2} = \frac{1}{2}[pq_3 + qq_0 - rq_0] \tag{3.44}$$

$$\dot{q_3} = \frac{1}{2}[-pq_2 + qq_1 + rq_0] \tag{3.45}$$

$$\dot{p} = \frac{-1}{I_x}[k_{r_x}p - lU_2 - I_y qr + I_z qr + I_r q\omega_r] \tag{3.46}$$

$$\dot{q} = \frac{-1}{I_y}[-k_{r_y}q + lU_3 - I_x pr + I_z pr + I_r p\omega_r] \tag{3.47}$$

$$\dot{r} = \frac{-1}{I_z}[U_4 - k_{r_z}r + I_x pq - I_y pq] \tag{3.48}$$

3.3 ROTOR DYNAMICS

Brushless DC motors are very popular for quadrotors because of its low friction and high torque. In this work, it is considered that the motors are directly connected with propellers without any help of gear box. In general, the Brushless DC motor behaves like a conventional DC motor and due to that their dynamic is also same. The schematic diagram of a brushless DC motor has been demonstrated as follows.

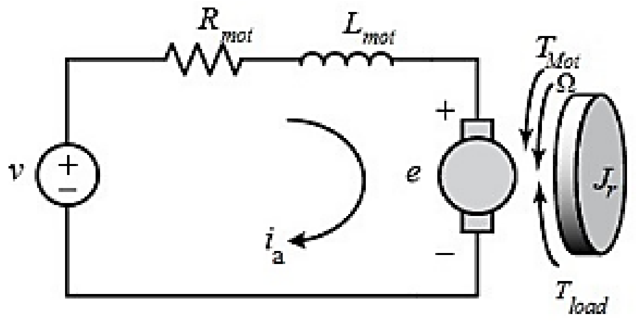


Figure 3.3 DC Motor Schematic Diagram (ElKholy, 2014)

The equation for motors can be achieved using Kirchhoff's law as follows.

$$V = R_{mot} i_a + L_{mot} \frac{di_a}{dt} + K_{mot} \Omega \tag{3.49}$$

where

R_{mot} = a motor's resistance

L_{mot} = a motor's inductance

i_a= flowing current around armature

V = motor input voltage

K_{mot}= motor torque constant

$K_{mot}\Omega$= Electromotive force

As the inductance of small motor is very small, the equation (3.49) can be represented as:

$$V = R_{mot} i_a + K_{mot} \Omega \tag{3.50}$$

From mechanical derivation of motor, another equation can be formulated as follows.

$$J_r \dot{\Omega} = T_{mot} - T_{load} \tag{3.51}$$

where,
T_{mot}= motor torque produced by electricity
T_{load}= load torque produced by propeller from equation (3.23)

Therefore the equation (3.52) can be represented by the following equation (3.55) where i_a has been derived from equation (3.51).

$$J_r \dot{\Omega} = K_{mot} i_a - k_M \Omega^2 \tag{3.52}$$

$$J_r \dot{\Omega} = K_{mot} \frac{V - K_{mot} \Omega_a}{R_{mot}} - k_M \Omega^2 \tag{3.53}$$

The voltage can be represented as follows in equation (3.54) from equation (3.53).

$$V = \frac{R_{mot}}{K_{mot}} J_r \dot{\Omega} + K_{mot}\Omega + k_M \Omega^2 R_{mot} \qquad (3.54)$$

A lag transfer function has been derived in order to achieve the rotor dynamics. The lag transfer function includes two variables like gain and time constant. These two variable are identified using MATLAB System Identification Toolbox (Mathworks, 2018). This lag transfer function offers a simple transfer function that is a ratio between actual propeller speed and desired propeller speed. Noted that both the speeds are directly proportional to the supplied voltage to the motors as found from equation (3.55). The transfer function that has been achieved for OS4 quadrotor as follow (Bouabdallah, 2007).

$$G(s) = \frac{\text{Actual speed of rotor}}{\text{Desired speed of the rotor}} = \frac{0.936}{0.178s + 1} \qquad (3.55)$$

3.4 STATE SPACE MODEL

A state space model is the representation of a dynamic model of the system. However, a state space model helps to estimate the behavior of any system and as a result, a suitable controller can be designed accordingly. Hence, a state space model has been developed in this section based on following steps.

State Vector

State Vector is necessarily mentioned in order to describe the complete dynamic model of a quadrotor. In this work, two different orientation systems have been described and hence, two different state vectors have been represented. Dynamic model for Euler angle orientation requires 12 states in equation (3.56) while quaternion requires 13 states in equation (3.57).

$$X_{s_e} = [x \quad y \quad z \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad \phi \quad \theta \quad \psi \quad p \quad q \quad r]^T \tag{3.56}$$

$$X_{s_q} = [x \quad y \quad z \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad q_0 \quad q_1 \quad q_2 \quad q_3 \quad p \quad q \quad r]^T \tag{3.57}$$

Control Input Vector *U*

The control input vector U consists of U_1, U_2, U_3 and U_4. The equations are as follows:

$$U_1 = k_f(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{3.58}$$

$$U_2 = k_f(\Omega_4^2 - \Omega_2^2) \tag{3.59}$$

$$U_3 = k_f(-\Omega_3^2 + \Omega_1^2) \tag{3.60}$$

$$U_4 = k_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_1^2) \tag{3.61}$$

Alternatively, equation (3.58) to (3.61), it can be represented in matrix form as follows in equation (3.62)

$$\begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} k_f & k_f & k_f & k_f \\ 0 & k_f & 0 & -k_f \\ -k_f & 0 & k_f & 0 \\ k_M & -k_M & k_M & -k_M \end{pmatrix} \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} \tag{3.62}$$

However, when the rotor velocities are required to be estimated from the control inputs, an inverse relationship between the control inputs and the rotors' velocities can be formulated that has been described as follows in equation (3.63).

$$\begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} = \begin{pmatrix} \frac{1}{4k_f} & 0 & \frac{1}{2k_f} & \frac{1}{4k_M} \\ \frac{1}{4k_f} & \frac{1}{2k_f} & 0 & -\frac{1}{4k_M} \\ \frac{1}{4k_f} & 0 & \frac{1}{2k_f} & \frac{1}{4k_M} \\ \frac{1}{4k_f} & -\frac{1}{2k_f} & 0 & -\frac{1}{4k_M} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} \tag{3.63}$$

State-space representation

Now considering equation from (3.30) to (3.38), a state-space representation can be described as follows according to Euler angle orientation system in equation (3.64). Here, the states space equations have been shown through the function of state vectors and control inputs. In Euler angle orientation, state vectors are denoted by X_{s_e}while control inputs are denoted by U.

$$f(X_{s_e}, U) = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \frac{-1}{m}[k_{t_x}\dot{x} + U_1(s\phi s\psi + c\phi c\psi s\theta)] \\ \frac{-1}{m}[k_{t_y}\dot{y} + U_1(s\phi c\psi - c\phi s\psi s\theta)] \\ \frac{-1}{m}[k_{t_z}\dot{z} - mg + U_1 c\phi c\theta] \\ p + rc\phi t\theta + qs\phi t\theta \\ qc\phi - rs\phi \\ r\frac{c\phi}{t\theta} + q\frac{s\phi}{c\theta} \\ \frac{-1}{I_x}[k_{r_x}p - lU_2 - I_y qr + I_z qr + I_r q\omega_r] \\ \frac{-1}{I_y}[-k_{r_y}q + lU_3 - I_x pr + I_z pr + I_r p\omega_r] \\ \frac{-1}{I_z}[U_4 - k_{r_z}r + I_x pq - I_y pq] \end{pmatrix} \tag{3.64}$$

Accordingly, the state space model for quaternion orientation system can be derived from equation (3.39-3.48). Similar to Euler angle orientation, state space equations for quaternion orientation have been illustrated through the function of state vectors, X_{s_q} and control inputs, U. Noted that disturbances have not been considered in both state space equations for both Euler angle and quaternion orientation.

$$f\left(X_{s_q}, U\right) = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \frac{-1}{m}[k_{t_x}\dot{x} + U_1(2q_0q_2 + 2q_1q_3)] \\ \frac{-1}{m}[k_{t_y}\dot{y} - U_1(2q_0q_2 - 2q_1q_3)] \\ \frac{-1}{m}[k_{t_z}\dot{z} - mg + U_1(2{q_0}^2 + 2{q_3}^2 - 1)] \\ \frac{1}{2}[-pq_1 - qq_2 - rq_3] \\ \frac{1}{2}[pq_0 - qq_3 + rq_2] \\ \frac{1}{2}[pq_3 + qq_0 - rq_0] \\ \frac{1}{2}[-pq_2 + qq_1 + rq_0] \\ \frac{-1}{I_x}[k_{r_x}p - lU_2 - I_yqr + I_zqr + I_rq\omega_r] \\ \frac{-1}{I_y}[-k_{r_y}q + lU_3 - I_xpr + I_zpr + I_rp\omega_r] \\ \frac{-1}{I_z}[U_4 - k_{r_z}r + I_xpq - I_ypq] \end{pmatrix} \tag{3.65}$$

3.5 LINEAR MODEL VERIFICATION

In classical controller design, derivation of a linear model is the most important concern before controller design. Besides, the model verification is another important step to the way of controller design as well. Significantly, the behavior of model is evaluated based on some parameters in order to make it worthy for proper controller design. Hence, the necessary steps have been demonstrated in this section in order to evaluate the behavior of the model.

3.5.1 Linearization

As both LQR and MPC approaches need a linear model, the nonlinear model is required to be linearized around an operating point, (X_{ss}, U_{ss}). Here, Jacobian method is applied to derive the linearized model from the dynamic model. Therefore, the state-space model for Euler angle orientation will be as follows:

$$A = \frac{\partial f(X_s\,, U)}{\partial X_s} \tag{3.66}$$

$$B \;= \frac{\partial f(X_s\,, U)}{\partial U} \tag{3.67}$$

Therefore, a 12×12 matrix, A and a 12×4 matrix, B (as mentioned in Appendix B) has been derived when the system is designed in Euler angle while A and B matrices are 13×13 and 13×4 in Quaternion based designed system (as mentioned in Appendix B). Hence, the generalized linear model of the quadrotor will be as follows:

$$\delta\dot{X}_s = \; A\delta X_s + B\delta U_s \tag{3.68}$$

$$\delta Y_s = C\delta X_s \tag{3.69}$$

where,

$$\delta X = X_s - X_{ss}$$

$$\delta U = U_s - U_{ss}$$

3.5.2 Controllability and Observability

Controllability and observability are the two most important concepts in the modern control theory for linearization. R. Kalman introduced these two concepts in 1960 to verify any model if it is solvable by linearization (Kalman, 1970).

Controllability

When it is possible to find out control inputs that can accept an initial state and lead it to the desired state, the system is called controllable.

Observability

When the states of a system can be measured by output values, the system is considered as observable.

A system can be considered as controllable and observable when it is full-ranked. The linear system can be considered as follows in order to find out the observability and controllability of the system (Sabatino, 2015).

$$\delta\dot{X}_s = A\delta X_s \text{ where, } X_{ss}(t_0) = X_0$$

$$\delta Y_s = C\delta X_s$$

According to the mathematical derivation, X_{ss} and Y_{ss} are 12×1 matrices. The observability matrix will be shown as follows.

$$O = [C \quad CA \quad CA^2 \quad \ldots \quad CA^{11}]^T$$

Here, O is 144×12 matrix. The linear system can be observable when the observability matrix will be full ranked.

$$C = [B \quad AB \quad A^2B \quad \ldots \quad A^{11}B]$$

Here, C is 12×48 matrix. The linear system can be controllable when the controllability matrix will be full ranked.

On the other hand, the system is also observable for quaternion orientation while X_s and Y_s are 13×1 matrices. Surprisingly, it is found that the system is not controllable for quaternion. Here, only 12 states are controllable and q_0 is not controllable. However, it is known from the definition of unit quaternion as follows (Reyes-Valeria et al., 2013):

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{3.70}$$

Therefore, q_0 can be solved from the equation 3.70 (Long et al., 2012). For quaternion, both X_s and Y_s are considered as 12×1 matrices. As a result, O will be

144×12 matrix and C is 12×48 matrix as well for quaternion orientation like Euler angle orientation.

MATLAB environment is used to check the observability and controllability of the system and finally it is found that the system is controllable and observable for both Euler angle and quaternion orientation after modification as aforementioned.

3.5.3 Open Loop Simulation

Open loop simulations of designed linear model and nonlinear model are proceeded in this section in order to ensure that both the linear model at a certain operating point and nonlinear model work with minimal error. MATLAB and Simulink environment has been adopted to proceed on the simulation process. Noted that the parameters of quadrotor have taken from the PhD thesis of Bouabdallah that was prepared based on project OS4 (Bouabdallah, 2007) (see in Appendix A).

Open simulations have taken places on both linear and nonlinear model in Simulink model and showed in figure 3.4 and 3.5.

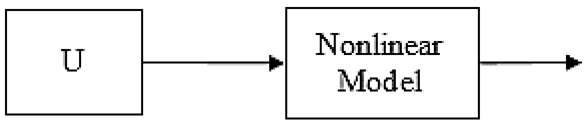


Figure 3.4 Open Loop Simulation for Nonlinear Model

According to the linear model definition, the difference between initial value, U_{ss} and desired value, U is considered as δU is sent to the model. As a result, δX will be achieved after providing the value of δU to the system. As a result, finally initial states, X_{ss} is added with δX to achieve desired states. Input and output can be defined by the equations as follows in equation (3.71) and (3.72):

$$U_s = U_{ss} + \delta U \tag{3.71}$$

$$X_s = X_{ss} + \delta X \tag{3.72}$$

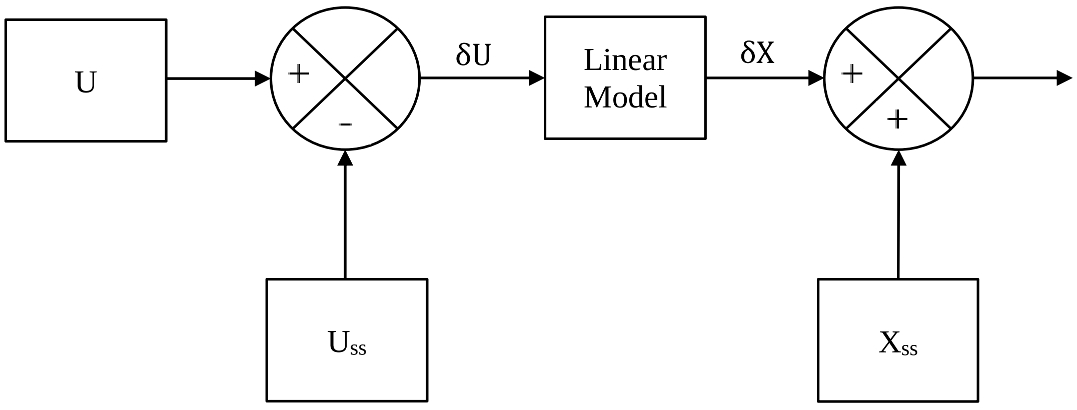


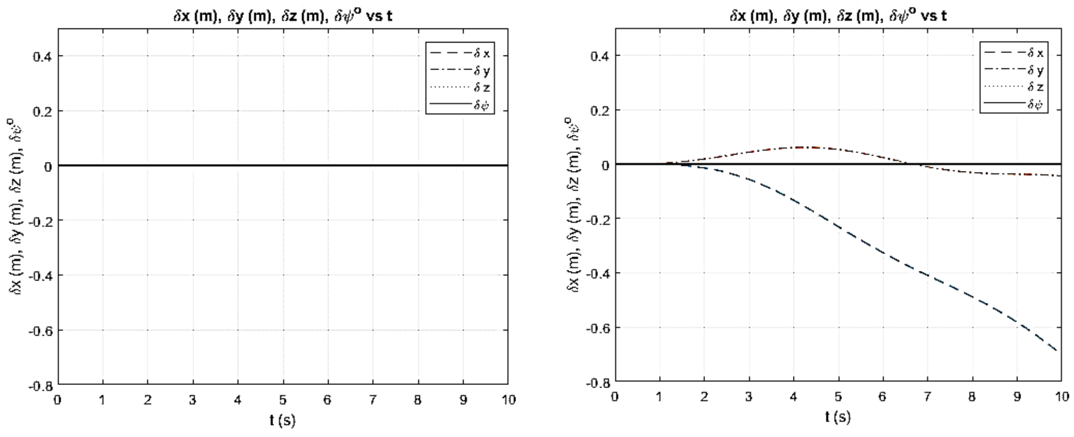
Figure 3.5 Open Loop Simulation in Linear Model

Open loop simulation on Euler angle representation

Accordingly, an operating point at any hover position is chosen to go proceed on the open loop simulation for any linear system. Therefore, a control input, U_{ss}= $[mg, 0, 0, 0]^T$ at any hover point has been chosen in figure 3.6 (a). It depicts that the deviation between linear and nonlinear model is zero at it was presumed. However, the input has been changed a minute than previous input as $U_s = [mg + 0.1, 0.001, 0, 0.1]^T$ and the consequence has been portrayed in figure 3.6 (b). It illustrates the behavior of linear and nonlinear models remain same until 2s and then the deviation starts. Hence, it is confirmed that the linear model at a certain operating can behave smoothly.

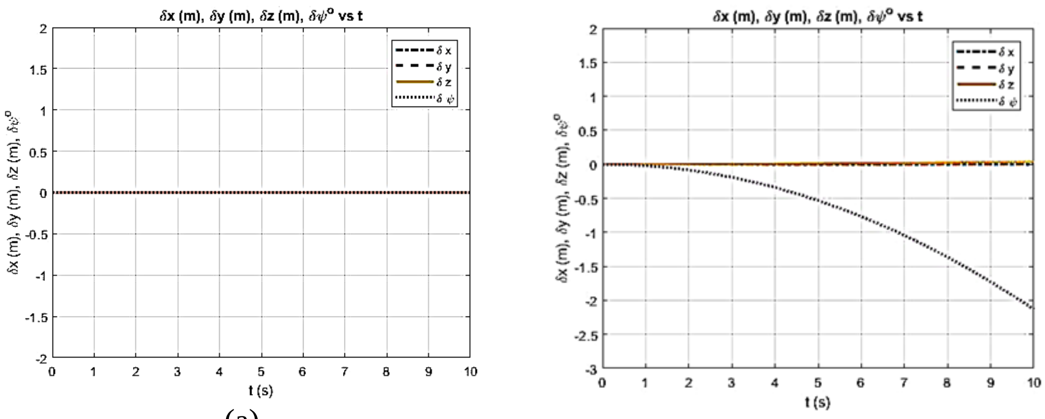
Open loop simulation on Quaternion

Similarly another two open simulations are accomplished in Quaternion designed model for both linear and nonlinear model of quadrotor where U_s were considered as $[mg, 0, 0, 0.0001]^T$and $[mg + 0.001, 0, 0, 0.0001]^T$ in order to check the compatibility of linear model with nonlinear model. Significantly, from figure 3.7 (b) it is found that the behavior of linear model started to change after 1.5s because the operating point is different. In contrast, in figure 3.7 (a), linear model behaves very smoothly and almost the responses are same to nonlinear model as their operating points are same.



(a) (b)

Figure 3.6 $\delta x, \delta y,\ \delta z$ and $\delta\psi$ when (a) U_S= $[mg, 0, 0, 0]^T$ (b) U_S= $[mg + 0.1, 0.001, 0, 0.1]^T$ in Euler angle



(a) (b)

Figure 3.7 $\delta x, \delta y,\ \delta z$ and $\delta\psi$ when (a) U= $[mg, 0, 0, 0]^T$ (b) U= $[mg + 0.001, 0, 0, 0.001]^T$ in Quaternion

3.6 CONTROLLER DESIGN

The three different control techniques, PID, LQR and MPC for both Euler angle and quaternion orientations are developed as follows.

3.6.1 PD Controller

PID control technique is a commonly used technique in both linear model and non-linear system because of its simplicity to design. Here, the dynamic model of a quadcopter does not show any steady-state error and due to the reason, integrator part of PID controller has not been used necessarily. As the model offers second order equations, the equations are required to be integrated twice that automatically remove the steady-state error from the system. However, the general form of a PD controller can be mentioned by following equations (3.73) and (3.74).

$$e(t) = X_d - X_a \tag{3.73}$$

$$U(t) = K_p e(t) + K_d \frac{\mathrm{d}}{\mathrm{dt}} e(t) \tag{3.74}$$

where, $e(t)$ symbolizes the error between desired states (X_d) and actual states (X_a), K_p denotes the proportional gain and K_d is the derivative gain. A block diagram of PD controller has been depicted in figure 3.8.

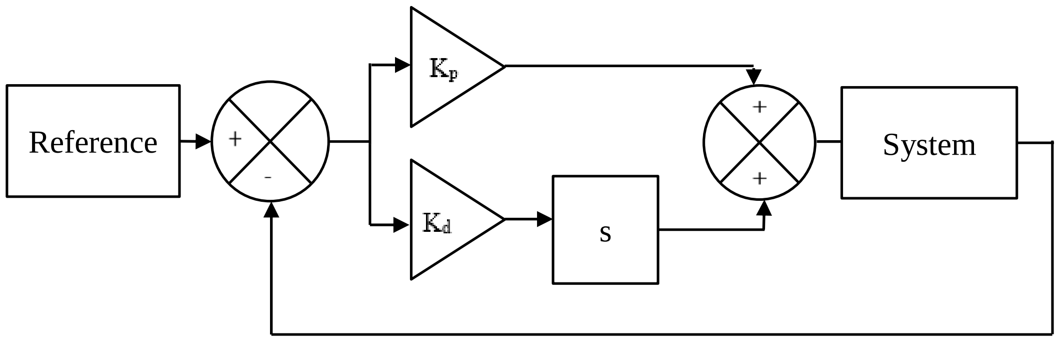


Figure 3.8 Block Diagram of PD controller

There are several tuning method for PID controllers like Ziegler-Nichols method, Tyreus-Luyben method, Damped oscillation method, Cohen and Coon method, Fertik method, Ciancone and Marline method, Internal Model Control (IMC) and Minimum Error Integral Criteria (IAE, ISE, ITAE) method are considered where some are applied in closed loop and some are in open loop control system (Shahrokhi &

Zomorrodi, 2013). Despite the PID tuning method, in most of the times, trial and error method are being used to deal with any nonlinear model (Padhee, 2014).

As quadrotor is under-actuated system, only four states can be regulated at the same time by using PD controller wherein either position and yaw angle or attitude and only altitude can be the options.

3.6.1.1 Attitude Mode in Euler angle

Attitude mode means the three angles (i.e. roll, pitch and yaw) and the altitude control mode. According to the equations (3.75), and (3.77), the PD controller can be designed individually for roll, pitch, yaw control and altitude control.

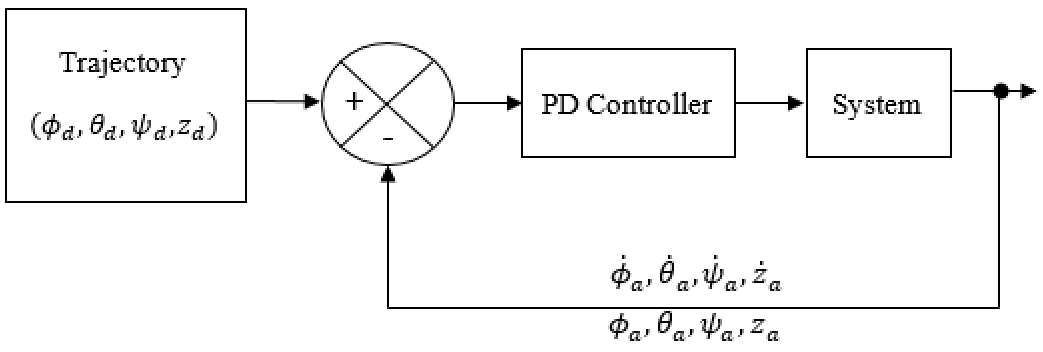

Trajectory
$(\phi_d, \theta_d, \psi_d, z_d)$
+
-
PD Controller
System
$\dot{\phi}_a, \dot{\theta}_a, \dot{\psi}_a, \dot{z}_a$
$\phi_a, \theta_a, \psi_a, z_a$

Figure 3.9 Block diagram of attitude mode for PD Controller in Euler Angle

Roll, pitch, yaw control

In order to control roll, pitch and yaw, three different PD controllers are required to be designed where U_2, U_3 and U_4 are the control inputs for roll, pitch and yaw respectively as follows.

$$U_2 = K_{p,\phi}(\phi_d - \phi_a) + K_{d,\phi}(\dot{\phi}_d - \dot{\phi}_a) \tag{3.75}$$

$$U_3 = K_{p,\theta}(\theta_d - \theta_a) + K_{d,\theta}(\dot{\theta}_d - \dot{\theta}_a) \tag{3.76}$$

$$U_4 = K_{p,\psi}(\psi_d - \psi_a) + K_{d,\psi}(\dot{\psi}_d - \dot{\psi}_a) \tag{3.77}$$

where,

K_p = Proportional gain

K_d = Derivative gain

ϕ_d, θ_d, ψ_d = Desired roll, pitch and yaw angle respectively

$\dot{\phi}_d, \dot{\theta}_d, \dot{\psi}_d$ = Desired roll, pitch and yaw angle rate of change respectively

ϕ_a, θ_a, ψ_a = Feedback roll, pitch and yaw angle respectively

$\dot{\phi}_a, \dot{\theta}_a, \dot{\psi}_a$ = Feedback roll, pitch and yaw angle rate of change respectively

Altitude control

Similarly, another PD controller was designed for altitude control that provides U_1 . It has been formulated based on equation (3.78) and (3.32).

$$\ddot{z}_d = K_p(z_d - z_a) + K_d(\dot{z}_d - \dot{z}_a) \tag{3.78}$$

$$U_1 = m\left[\frac{g - (\ddot{z}_d + \frac{k_{t_z}}{m}\dot{z}_a)}{\cos\phi\cos\theta}\right] \tag{3.79}$$

where,

z_d = Desired altitude

$\dot{z}_d$ = Desired altitude rate of change

z_a = Feedback altitude

$\dot{z}_a$ = Feedback altitude rate of change

$\ddot{z}_d$ = Desired acceleration along z-axis

3.6.1.2 Attitude Mode in quaternion

For attitude mode in quaternion, roll, pitch and yaw are considered in terms of quaternion. Here, the error between desired quaternion and actual quaternion can be formulated using quaternion multiplication. As quaternion multiplication and algebraic subtraction are two different methods, the design method of PD controller for quaternion is different from Euler angle orientation. A block diagram can depict it more comprehensively in figure 3.10.

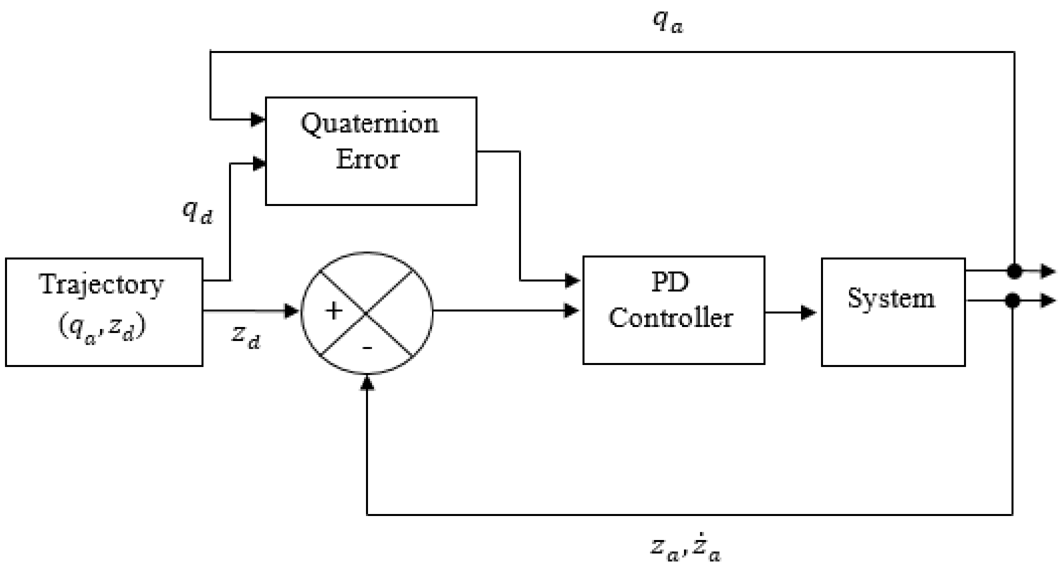


Figure 3.10 Block diagram of trajectory mode for PD Controller in Quaternion

Roll, pitch and yaw control

In quaternion, the control inputs are necessarily required to be decoupled into roll, pitch and yaw as like Euler angle orientation in order to give proper command to the motors. As a result, quaternion error is formulated by considering desired and current quaternion states as follows in equation (3.80) (Fresk & Nikolakopoulos, 2013).

$$q_e = q_d \otimes q_a^{-1} \tag{3.80}$$

where,

q_e = Quaternion Error

q_d = Desired Quaternion

q_a = Present Quaternion

Equation (3.80) also can be elaborated by equation (3.81) in order to make it more comprehensible (Kehlenbeck, 2014) as follows.

$$\begin{bmatrix} q_{0,e} \\ q_{1,e} \\ q_{2,e} \\ q_{3,e} \end{bmatrix} = \begin{bmatrix} q_{0,d} & q_{1,d} & q_{2,d} & q_{3,d} \\ -q_{1,d} & q_{0,d} & q_{3,d} & -q_{2,d} \\ -q_{2,d} & -q_{3,d} & -q_{0,d} & q_{1,d} \\ -q_{3,d} & q_{2,d} & -q_{1,d} & q_{0,d} \end{bmatrix} \begin{bmatrix} q_{0,a} \\ q_{1,a} \\ q_{2,a} \\ q_{3,a} \end{bmatrix} \tag{3.81}$$

Then the control inputs for roll, pitch and yaw can be achieved by the following equation (3.82).

$$\begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} = -K_{p,q} \begin{bmatrix} q_{1,e} \\ q_{2,e} \\ q_{3,e} \end{bmatrix} - K_{d,q} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.82}$$

where,

$K_{p,q}$ = Diagonal matrix for proportional gain

$K_{d,q}$ = Diagonal matrix for derivative gain.

The sign of $K_{p,q}$ determines whether the quadrotor moves to the desired rotation angle in shortest direction or longest direction. As all the unit quaternions are squared to explain three dimensions, the sign of $K_{p,q}$ is dependent on the sign of $q_{0,e}$ to confirm the rotation of quadcopter along with the possible shortest direction (Kehlenbeck, 2014; Wie, 2008).

Altitude Control

Altitude controller is designed by following equation (3.83) and (3.84) as like Euler angle orientation system.

3.6.1.3 Trajectory Mode in Euler angle

In trajectory mode, position along axes and yaw angle are taken into consideration for tracking. Here, roll and pitch angles are decoupled with position (x, y). Thus, four different PD controllers are applied to control the position of the quadrotor where two controllers are applied for position (x, y) and another two controllers are used for roll and pitch angle as shown in figure (3.11). Here, controller generates inputs to the system model or dynamic model and the model transfers the feedback to the desired trajectory zone.

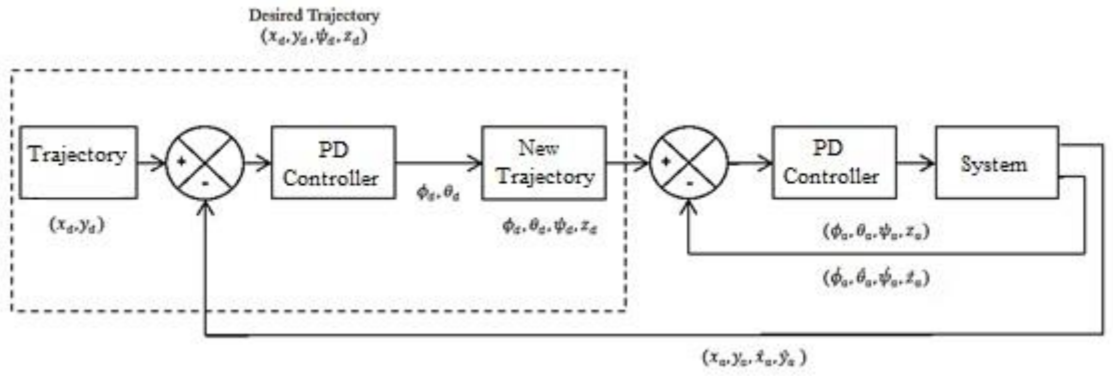

Desired Trajectory
(x_d, y_d, ψ_d, z_d)
Trajectory
(x_d, y_d)
+
-
PD Controller
ϕ_d, θ_d
New Trajectory
$\phi_d, \theta_d, \psi_d, z_d$
+
-
PD Controller
System
$(\phi_a, \theta_a, \psi_a, z_a)$
$(\dot{\phi}_a, \dot{\theta}_a, \dot{\psi}_a, \dot{z}_a)$
$(x_a, y_a, \dot{x}_a, \dot{y}_a)$

Figure 3.11 Block diagram of trajectory mode for PD Controller in Euler Angle

Position control (x, y)

For position control, two PD controllers are designed for desired accelerations along x and y axes as follows while these two values are used later.

$$\ddot{x}_d = K_p(x_d - x_a) + K_d(\dot{x}_d - \dot{x}_a) \tag{3.83}$$

$$\ddot{y}_d = K_p(y_d - y_a) + K_d(\dot{y}_d - \dot{y}_a) \tag{3.84}$$

Now in equation (3.30) and (3.31), the desired accelerations along x-axis, $\ddot{x}_d$ and y-axis, $\ddot{y}_d$ are substituted by the acquired values from equation (3.83) and (3.84) in order to achieve desired pitch (θ_d) and roll (ϕ_d) angle as follows.

$$\theta_d = \tan^{-1}\left[\frac{-\cos\psi\left(\ddot{x}_d + \frac{k_{t_x}}{m}\dot{x}_a\right) - \sin\psi\left(\ddot{y}_d + \frac{k_{t_y}}{m}\dot{y}_a\right)}{g - (\ddot{z}_d + \frac{k_{t_z}}{m}\dot{z}_a)}\right] \tag{3.85}$$

$$\phi_d = \tan^{-1}\left[\frac{\left(-\sin\psi\left(\ddot{x}_d + \frac{k_{t_x}}{m}\dot{x}_a\right) + \cos\psi\left(\ddot{y}_d + \frac{k_{t_y}}{m}\dot{y}_a\right)\right)\cos\theta_d}{g - (\ddot{z}_d + \frac{k_{t_z}}{m}\dot{z}_a)}\right] \tag{3.86}$$

Then by following equation (3.85) and (3.86), roll and pitch can be controlled by using PD controllers.

Altitude control

For altitude control in trajectory mode, equation (3.78) and (3.79) are used as it is controlled in attitude mode.

Yaw or heading control

In trajectory mode, equation (3.77) is used to control yaw or heading of the quadrotor as it is formulated in attitude mode.

3.6.1.4 Trajectory Mode in quaternion

For trajectory mode in quaternion system, almost similar approach to Euler angle system has been adopted except the quaternion error segment. In trajectory mode, desired position, (x, y, z) and a quaternion element, q_3 are considered as known values where q_3 is normally responsible for yaw movement. In trajectory mode, two different controllers are chosen where one is for position control and another for attitude control of quadrotor. The block diagram in figure 3.12 may illustrate the complete idea for trajectory mode.

In figure 3.12, initially desired position x_d, y_d, z_d and $q_{3,d}$ are known where $q_{0,d}$ and $q_{3,d}$ are responsible for the yaw movement. Then desired quaternion elements $q_{0,d}$, $q_{1,d}$, $q_{2,d}$ and U_1 can be solved from equation (3.39-3.41) as follows:.

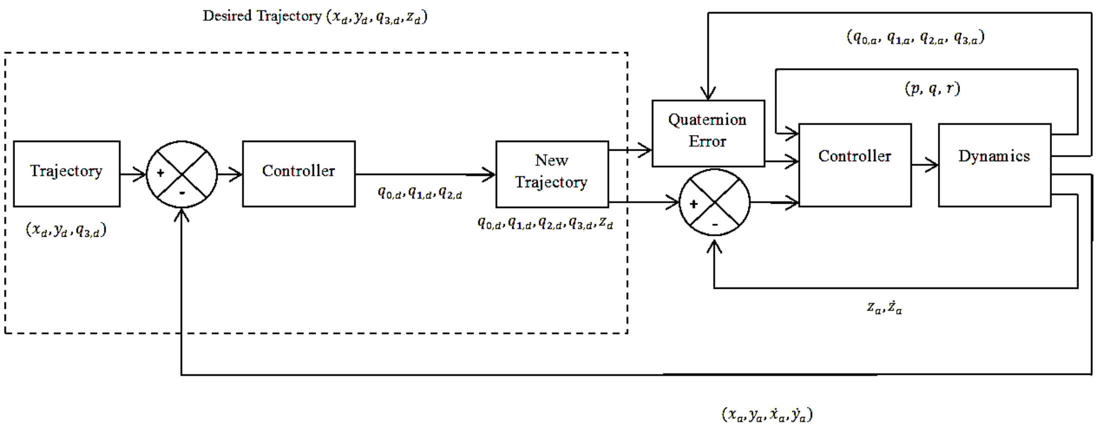

Desired Trajectory $(x_d, y_d, q_{3,d}, z_d)$
Trajectory
$(x_d, y_d, q_{3,d})$
+
-
Controller
$q_{0,d}, q_{1,d}, q_{2,d}$
New Trajectory
$q_{0,d}, q_{1,d}, q_{2,d}, q_{3,d}, z_d$
Quaternion Error
$(q_{0,a}, q_{1,a}, q_{2,a}, q_{3,a})$
(p, q, r)
Controller
Dynamics
$z_a, \dot{z}_a$
$(x_a, y_a, \dot{x}_a, \dot{y}_a)$

Figure 3.12 Block diagram of trajectory mode for PD Controller in Quaternion

Here, initially desired position (x_d, y_d, z_d) and quaternion element $q_{3,d}$ are known where $q_{0,d}$ and $q_{3,d}$ are responsible for the yaw movement. Then desired quaternion elements, $(q_{0,d}, q_{1,d}, q_{2,d})$ and control input U_1 can be solved from equation (3.39-3.41) as follows.

$$q_{0,d} = \sqrt{\frac{1}{2}\left(1 + \frac{g + \dot{z}}{\ddot{x}_d^2 + \ddot{y}_d^2 + (\ddot{z}_d + g)^2}\right) - {q_{3,d}}^2} \tag{3.87}$$

$$U_1 = \frac{m(g - \ddot{z}_d)}{2({q_{0,d}}^2 + {q_{3,d}}^2)^2 - 1} \tag{3.88}$$

$$q_{1,d} = \frac{1 - 2({q_{0,d}}^2 + {q_{3,d}}^2)}{2({q_{0,d}}^2 + {q_{3,d}}^2)(g - \dot{z}_d)}\left[q_{0,d}\left(\ddot{y}_d + \frac{k_{t_y}}{m}\dot{y}_a\right) - q_{3,d}\left(\ddot{x}_d + \frac{k_{t_x}}{m}\dot{x}_a\right)\right] \tag{3.89}$$

$$q_{2,d} = \frac{1 - 2({q_0}^2 + {q_3}^2)}{2({q_{0,d}}^2 + {q_{3,d}}^2)(g - \dot{z}_d)}\left[q_{0,d}\left(\ddot{x}_d + \frac{k_{t_x}}{m}\dot{x}_a\right) + q_{3,d}\left(\ddot{y}_d + \frac{k_{t_y}}{m}\dot{y}_a\right)\right] \tag{3.90}$$

From equation (3.87), (3.89) and (3.90), three quaternion elements $q_{0,d}, q_{1,d}$ and $q_{2,d}$ have been formulated. Hence, initial desired trajectory (x_d, y_d, z_d and $q_{3,d}$) transformed into new trajectory $(q_{0,d}, q_{1,d}, q_{2,d}, q_{3,d}, z_d)$ that is similar to attitude mode. This new trajectory will then follow the exactly same procedure as aforementioned in section 3.6.1.2.

3.6.2 LQR CONTROLLER

LQR is one of the most popular optimal control techniques for quadrotor. LQR is considered as linear controller and is developed based on linear model of the system. The controller follow Cost function minimizing approach also known as Optimal control method in order to compute the states of the system.

Similar to PD controller, LQR has been applied in both Euler and Quaternion orientation system that has been described as follows.

3.6.2.1 LQR in Euler angle system

According to the definition, LQR needs a linearized model that has been already derived before from equation (3.66-3.67). Since it is a generalized linear model, an operating point is required to be chosen to use the control technique while hovering with a heading angle is chosen as an operating point for the quadrotor.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{k_{t_x}}{m} & 0 & 0 & -gs\psi_T & -gc\psi_T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{k_{t_y}}{m} & 0 & gc\psi_T & gs\psi_T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{k_{t_z}}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -\frac{k_{t_x}}{I_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -\frac{k_{t_y}}{I_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -\frac{k_{t_z}}{I_z} \end{bmatrix} \quad (3.91)$$

Equation (3.91) and (3.92) offers matrices A and B which are derived based on a general hovering point and it can be applicable at any operating point in hovering condition.

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{l}{I_x} & 0 & 0 \\ 0 & 0 & \frac{l}{I_x} & 0 \\ 0 & 0 & 0 & \frac{l}{I_x} \end{bmatrix} \tag{3.92}$$

If the model is linearized at an nominal point (X_{ss}, U_{ss}) where $X_{ss} = [x_{ss}, y_{ss}, z_{ss,}\ \psi_{ss}\ ]^T$ and others states are considered as zero with $U_{ss} = [mg, 0, 0, 0]^T$ where,

$$\delta X = X_d - X_a \tag{3.93}$$

$$U = U_{ss} + \delta U \tag{3.94}$$

$$X = X_{ss} + \delta X \tag{3.95}$$

Now according to the control approach of LQR, a feedback control is required to be designed by following equation (3.96).

$$U = -K\delta X_s + U_{ss} \tag{3.96}$$

Where, K is the feedback gain matrix.

The gain matrix K has been calculated by minimizing the cost function as follows.

$$J = \int_0^\infty (\delta X_s Q \delta X_s + \delta U R \delta U) dt \tag{3.97}$$

where, Q is considered as a semi-positive definite matrix of $m \times m$ and R is a positive definite matrix of $m \times n$ wherein *m* symbolizes number of states and *n* is considered as the number of control input.

Therefore, the closed loop control system can appears as follows in equation (3.98)

$$\delta\dot{X}_s = (A - BK)\delta X_s \tag{3.98}$$

where, K has been calculated using Q and R matrices.

Figure 3.13 shows a Simulink model of quadrotor linear dynamic model with LQR control approach.

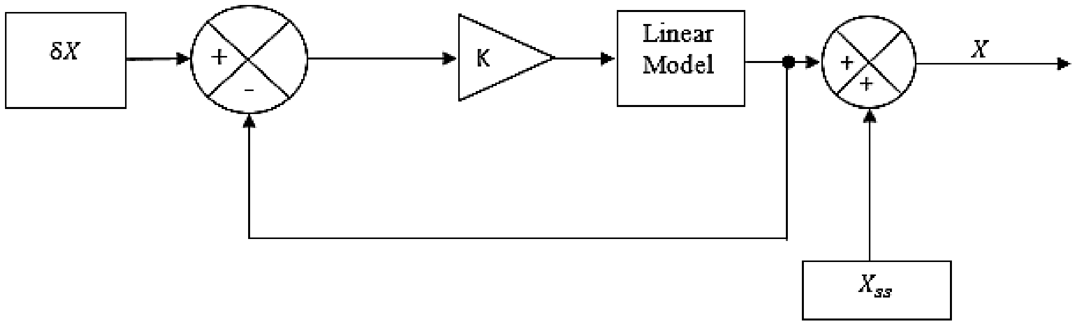


Figure 3.13 Block diagram for linear model with LQR

LQR also can be applied to nonlinear model but the approach should be a minute different from the approach to linear model. For nonlinear approach, equation (3.94) has been used for control input, U. In figure 3.14, a nonlinear model for quadrotor is shown in block diagram using LQR approach.

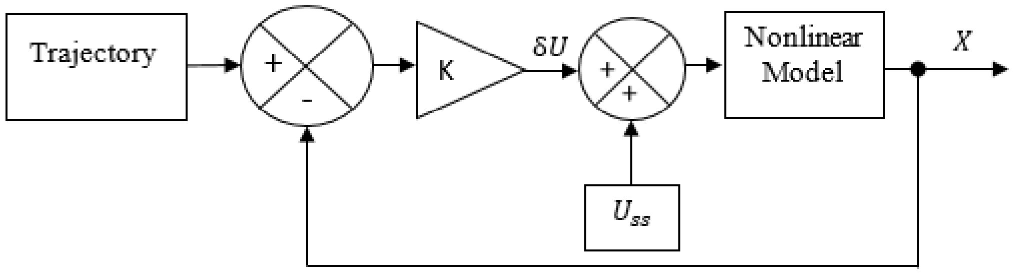


Figure 3.14 Block diagram for nonlinear model with LQR

3.6.2.2 LQR in Quaternion

To apply LQR control approach in quaternion orientation model, the nonlinear model should be linearized as like linearized model in Euler angle orientation. The linear model for quaternion system has been stated before in equation (3.66) and (3.67). Here equation (3.99) and (3.100) shows the A and B matrices for linear model in quaternion.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{k_{t_x}}{m} & 0 & 0 & 0 & -2gq_3 & -2gq_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{k_{t_y}}{m} & 0 & 0 & 2gq_0 & -2gq_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{k_{t_z}}{m} & -4gq_0 & 0 & 0 & -4gq_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-q_3}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{q_0}{2} & -\frac{q_3}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{q_3}{2} & \frac{q_0}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{q_0}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -\frac{k_{r_x}}{I_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -\frac{k_{r_y}}{I_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -\frac{k_{r_z}}{I_z} \end{bmatrix} \quad (3.99)$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{(2q_0^2 + 2q_3^2 - 1)}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{l}{I_x} & 0 & 0 \\ 0 & 0 & \frac{l}{I_x} & 0 \\ 0 & 0 & 0 & \frac{l}{I_x} \end{bmatrix} \quad (3.100)$$

The block diagram for linear model in quaternion using LQR has been illustrated figure 3.15.

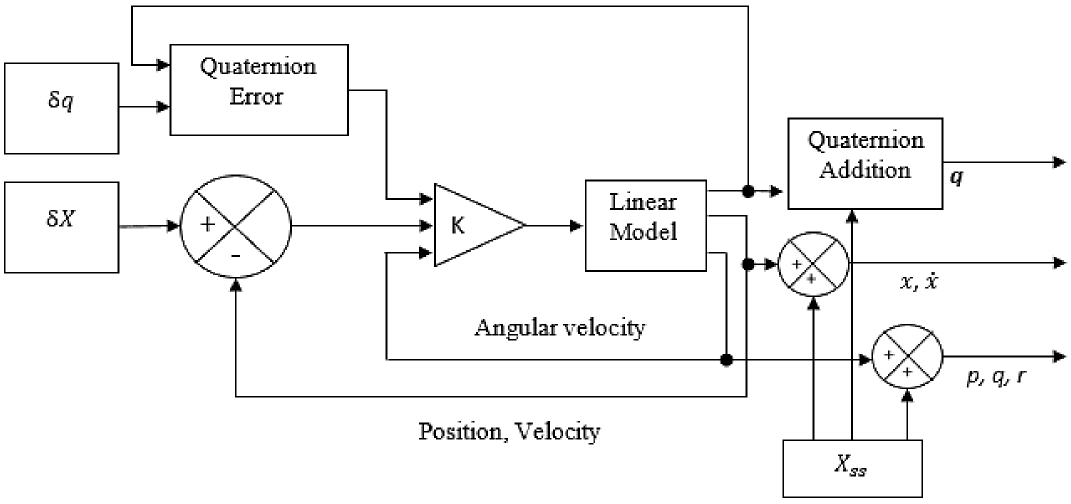


Figure 3.15 Block diagram for linear model with LQR in Quaternion

3.6.3 MPC CONTROLLER

As the control system of a quadrotor requires multiple input and multiple output (MIMO) and disturbances may influence system performances in case of outdoor applications, it needs a controller that can deal with both conditions. Moreover, it is quite impossible to employ constraints directly on the actual control signals to get optimized solutions because of coupling. Hence, Model Predictive Control (MPC) technique can be an option to overcome all of these problems.

MPC also known as receding horizon control (RHC) is a control approach that comprises a systematic algorithm. Here, the dynamic model of the system is formulated under a finite, moving horizon and closed loop control problem. It has the capability to use constraints in both control inputs and outputs on the system during the design process. According to its working principle, it predicts a number of outputs of the system so that it can generate an optimized control input for the system to reach the reference trajectory.

The optimization problem is calculated for each sampling time interval. The immediate optimized control signal is applied in the system until next sampling time interval. The process is repeated for each sampling time interval. In this section, the linear MPC control algorithm is described briefly.

Plant model and prediction horizon

A nonlinear system can be written in the form

$$\dot{x} = f(x(t), u(t)) \tag{3.101}$$

Where, $x(t) \epsilon R^n$ denotes the states of the system and $u(t) \epsilon R^m$ denotes system inputs.

So, the nonlinear system can be designed into a linear discrete-time system around a nominal point where as nominal states and nominal control inputs are x_T and u_T as follows where quadcopter dynamic model was linearized at hover condition.

$$\Delta x_{k+1+i} = A \Delta x_{k+i} + B \Delta u_{k+i} \tag{3.102}$$

$$\Delta y_{k+i} = C \Delta x_{k+i} + D \Delta u_{k+i} \tag{3.103}$$

Where, $i = 1, 2, 3, \ldots,$ N

$$\Delta x_k = x_k - x_T$$

$$\Delta u_k = u_k - u_T$$

k is sample time, $A \epsilon R^{n \times n}$ is the state matrix, $B \epsilon R^{n \times m}$ is input matrix, $y \epsilon R^p$ is system outputs, $C \epsilon R^{p \times n}$ is output matrix and $D \epsilon R^{p \times m}$ is feedforward matrix. For quadcopter we could know $n = 12$ and $m = 4$.

To reach the desired states, a prediction horizon, N is required to be determined so that controller can predict a number of future states. A state observer with an estimator is also required to be implemented in the controller to predict the future states while the estimator predicts the future behavior of the system. A Linear Quadratic Estimator is applied for the algorithm.

Now, by expanding equation (3.102-3.103), the future states and outputs can be achieved depending on initial states and future inputs as follows.

$$\begin{aligned}\Delta x_{k+1} &= A\Delta x_k + B\Delta u_k \\ &= A(A\Delta x_{k-1} + B\Delta u_{k-1}) + B\Delta u_{k+1} \\ &= A^2\Delta x_{k-1} + AB\Delta u_{k-1} + B\Delta u_{k+1} \\ &\vdots \\ \Delta x_{k+N} &= A^N\Delta x_k + A^{N-1}B\Delta u_k + A^{N-2}B\Delta u_{k+1} + \cdots + AB\Delta u_{k+N-2} \\ &\quad + B\Delta u_{k+N-1} \end{aligned} \tag{3.104}$$

Similarly, the system outputs can be derived as follows from equation (4.38).

$$\Delta y_{k+N} = CA^N\Delta x_k + C(A^{N-1}B\Delta u_k + A^{N-2}B\Delta u_{k+1} + \cdots + AB\Delta u_{k+N-2} + B\Delta u_{k+N-1}) \tag{3.105}$$

Therefore, the equations can be written in matrix form as follows.

$$\begin{pmatrix}\Delta x_k \\ \Delta x_{k+1} \\ \Delta x_{k+2} \\ \vdots \\ \Delta x_{k+N-1}\end{pmatrix} = \begin{pmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{N-1}\end{pmatrix}\Delta x_k + \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ B & 0 & \cdots & 0 & 0 \\ AB & B & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A^{N-2}B & A^{N-3}B & \cdots & B & 0\end{pmatrix}\begin{pmatrix}\Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \vdots \\ \Delta u_{k+N-1}\end{pmatrix} \tag{3.106}$$

$$\begin{pmatrix} \Delta y_k \\ \Delta y_{k+1} \\ \Delta y_{k+2} \\ \vdots \\ \Delta y_{k+N-1} \end{pmatrix} = \begin{pmatrix} C & 0 & 0 & \cdots & 0 \\ 0 & C & 0 & \cdots & 0 \\ 0 & 0 & C & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & C \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta x_{k+1} \\ \Delta x_{k+2} \\ \vdots \\ \Delta x_{k+N-1} \end{pmatrix} + \begin{pmatrix} D & 0 & 0 & \cdots & 0 \\ 0 & D & 0 & \cdots & 0 \\ 0 & 0 & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D \end{pmatrix} \begin{pmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \vdots \\ \Delta u_{k+N-1} \end{pmatrix} \quad (3.107)$$

In short form, we can write

$$\Delta X_k = A_m \Delta x_k + B_m \Delta u_k \quad (3.108)$$

$$\Delta Y_k = C_m \Delta x_k + D_m \Delta u_k \quad (3.109)$$

As D = 0 in most of the cases, the equation (3.109) can be written as

$$\Delta Y_k = C_m \Delta x_k \quad (3.110)$$

Control Design

The MPC technique must have a cost function in its control algorithm with a view to calculating the optimal solution at every sampling time interval. The cost function has to be designed in such a manner that the predicted outputs are directed towards the desired states. Here, the cost function is being minimized by the norm of the difference between the current outputs and desired trajectory and the norms of motor inputs as follows (Bemporad et al., 2017b). In addition, when it becomes an issue to find out the difference between quaternion outputs and desired trajectories are considered in order to get the cost function, quaternion error between quaternion outputs and desired trajectories are considered in lieu of normal algebraic subtraction as aforementioned in previous section.

$$J(\Delta x, \Delta u) = (\Delta u_k)^T \widehat{W}_u^{\,2} (\Delta u_k) + (\Delta Y_k - \Delta Y_k^r)^T \widehat{W}_y^{\,2} (\Delta Y_k - \Delta Y_k^r) \quad (3.111)$$

where,

$$\widehat{W}_u = \begin{bmatrix} W_{u|0,1} & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & W_{u|0,2} & \cdots & 0 & \ddots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W_{u|0,m} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & W_{u|N-1,1} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & W_{u|N-1,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & W_{u|N-1,m} \end{bmatrix}$$

$$\widehat{W}_y = \begin{bmatrix} W_{y|0,1} & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & W_{y|0,2} & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W_{y|0,m} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & W_{y|N-1,1} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & W_{y|N-1,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & W_{y|N-1,m} \end{bmatrix}$$

Quadratic Programming

As the cost function is in quadratic form, a quadratic programming can be applied to solve the optimization problem. The main purpose of the quadratic programming is to reduce the cost function $J(\Delta x, \Delta u)$ by finding out a feasible search direction Δu (Bemporad et al., 2017a).

Input and constraint handling

Constraint handling capability is an advantage of MPC formulation. It is necessary for quadrotor to handle the constraints in both the control efforts and magnitude of the angles to have proper stability of it.

Input constraints

During the designing of the quadrotor, it is important to apply constraint at the force of each motor so that it will behave like a realistic model. As a result, the presence of an upper, u_{ub} bound and lower bound, u_{lb} are very obvious at control inputs that can be expressed as

$$u_{lb} \leq u_{k+i} \leq u_{ub} \text{ for } i = 0, 1, 2, \ldots, N-1 \tag{3.112}$$

As dynamic model is linearized around a certain operating point, the MPC approach solves the perturbed control inputs for the linearized model as represented $u_{k+i} = \Delta u_T + \Delta u_{k+i}$. Therefore, equation 3.112 can be substituted as

$$u_{lb} \leq \Delta u_T + \Delta u_{k+i} \leq u_{ub}$$

$$u_{lb} - \Delta u_T \leq \Delta u_{k+i} \leq u_{ub} - \Delta u_T \tag{3.113}$$

However, equation 3.113 also can be expressed in matrix form as follows.

$$\begin{bmatrix} I_{m\times m} \\ -I_{m\times m} \end{bmatrix} \Delta u_{k+i} \leq \begin{bmatrix} u_{ub} - \Delta u_T \\ -(u_{lb} - \Delta u_T) \end{bmatrix} \tag{3.114}$$

where $I_{m\times m}$ is an identity matrix.

Besides, equation (3.115) is the representation of simple form of equation as follows.

$$I_u \Delta u_k \leq \Delta u_b \tag{3.115}$$

where,

$$I_u = \begin{bmatrix} \begin{bmatrix} I_{m\times m} \\ -I_{m\times m} \end{bmatrix} & 0 & \cdots & 0 \\ 0 & \begin{bmatrix} I_{m\times m} \\ -I_{m\times m} \end{bmatrix} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \begin{bmatrix} I_{m\times m} \\ -I_{m\times m} \end{bmatrix} \end{bmatrix}$$

$$\Delta u_b = \begin{bmatrix} \begin{bmatrix} u_{ub} - \Delta u_T \\ -(u_{lb} - \Delta u_T) \end{bmatrix} \\ \begin{bmatrix} u_{ub} - \Delta u_T \\ -(u_{lb} - \Delta u_T) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} u_{ub} - \Delta u_T \\ -(u_{lb} - \Delta u_T) \end{bmatrix} \end{bmatrix}$$

Output and states constraints

Sometimes it is also necessary to apply a limit on the any states in order to avoid any unwanted situation. Similar to input constraints, outputs can also be delimited by upper bound, z_{ub} and output bound, z_{lb} as follows.

$$z_{lb} \le C_z x_{k+i} \le z_{ub};\ i = 0, 1, 2, 3, \ldots., N-1 \tag{3.116}$$

As we get,

$$x_{k+i} = x_T + \Delta x_{k+i}$$

So,

$$z_{lb} \le C_z(x_T + \Delta x_{k+i}) \le z_{ub}$$

$$z_{lb} - C_z x_T \le C_z \Delta x_{k+i} \le z_{ub} - C_z x_T$$

It is also shown in matrix form as follows.

$$\begin{bmatrix} C_z \\ -C_z \end{bmatrix} \le \begin{bmatrix} z_{ub} - C_z x_T \\ -(z_{lb} - C_z x_T) \end{bmatrix}$$

So, from equation (3.108), the constraints can be described as follows where ΔX_k will be substituted.

$$\Gamma_z(A_m \Delta x_k + B_m \Delta u_k) \le \Delta z_b$$

Where,

$$\Gamma_z = \begin{bmatrix} \begin{bmatrix} C_z \\ -C_z \end{bmatrix} & 0 & \cdots & 0 \\ 0 & \begin{bmatrix} C_z \\ -C_z \end{bmatrix} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \begin{bmatrix} C_z \\ -C_z \end{bmatrix} \end{bmatrix}$$

$$\Delta z_b = \begin{bmatrix} \begin{bmatrix} z_{ub} - C_z x_T \\ -(z_{lb} - C_z x_T) \end{bmatrix} \\ \begin{bmatrix} z_{ub} - C_z x_T \\ -(z_{lb} - C_z x_T) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} z_{ub} - C_z x_T \\ -(z_{lb} - C_z x_T) \end{bmatrix} \end{bmatrix}$$

Combined Input and Output State Constraints

The control input and output constraints can be described by one single equation as follows.

$$M_u \Delta u_k \leq \Delta u_b$$

$$\Gamma_z (A_m \Delta x_k + B_m \Delta u_k) \leq \Delta z_b$$

$$\Gamma_z B_m \Delta u_k \leq \Delta z_b - \Gamma_z A_m \Delta x_k$$

So, the following equation can describe these two equations at the same time.

$$\Pi \Delta u_k \leq \Upsilon$$

Where,

$$\Pi = \begin{bmatrix} M_u \\ \Gamma_z B_m \end{bmatrix}$$

$$\Upsilon = \begin{bmatrix} \Delta u_b \\ \Delta z_b - \Gamma_z A_m \Delta x_k \end{bmatrix}$$

A flow chart can be shown that may carry out the complete concept of MPC (Tule, 2014).

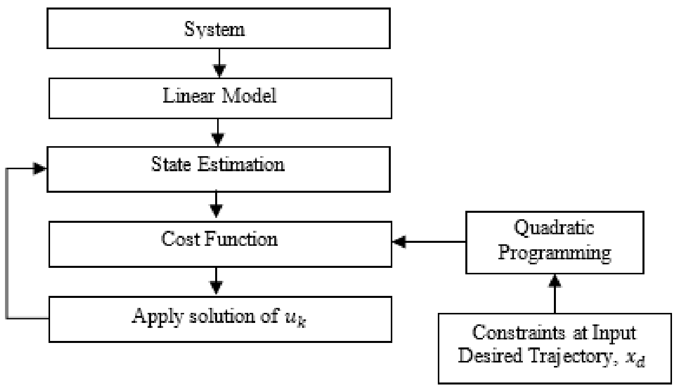


Figure 3.16 A Flowchart of MPC Process

3.7 CHAPTER SUMMARY

This chapter introduces with the dynamic model design of quadrotor, rotor dynamics and control algorithm development. Here, Euler angle and Quaternion orientation systems have been adopted in order to design the dynamic model of quadrotor. Rotor dynamics has been discussed also to comprehend the complete working procedure. Different control techniques have their own specialties with their unique algorithms and each control techniques is taken into consideration based on the type of applications of quadrotor. Significantly, MPC can be a suitable controller for outdoor applications where uncertainty may influence the model. Apart from that, it can ensure the motor safety by offering the feature of constraints at control inputs.

CHAPTER FOUR

RESULTS AND DISCUSSION

This chapter concentrates on the performance evaluation of the controller based on several situations like at certain operating point, helix and circular trajectories, and trajectories with disturbances. The performance evaluations have been performed based on MATLAB and Simulink environment. Here, several factors have been considered for performance evaluation that has been described elaborately in this chapter.

4.1 CONTROLLER PERFORMANCE AT CERTAIN POINT

This section focuses on the performances of the aforementioned controllers at a certain operating point that is considered as desired trajectory point. The performance of PD, LQR and MPC controllers in two different orientation systems as Euler angle representation and Quaternion have been demonstrated respectively.

4.1.1 PD Controller Simulation

4.1.1.1 Simulation for Euler angle orientation

The performance of PD controller depends on choosing proper gains. Therefore, it is one of the most difficult part to choose proper gains for PD controller. Here, Simulink optimization toolbox has been used to find out the proportional and derivative gains of PD controller. This optimization toolbox is dependent on gradient indent approach for optimization.

The control gains that were considered for altitude controller are K_p= 7.495 and K_d = 5. The optimization toolbox works on finding out the least possible error for altitude controller. By running the closed-loop simulation, settling time is found as 1.7681 sec and overshoot is found 0.0679% for the desired movement along z-axis.

Similarly, position and attitude controllers' gains are also generated using optimization toolbox. Table 4.1 and 4.2 illustrates the gains with settling time and overshoot along the three axes.

Table 4.1: PD controller performance for Euler orientation

	Desired Value	K_p	K_d	**Settling Time (sec)**	**Overshoot (%)**
x	3	2.1	2.8	4.0858	0
y	2	2.15	2.5	3.0730	0.0037
z	-5	7.495	5	1.7681	0.0679
ψ	15°	5	1.4	1.1497	2.9750

The gains for ϕ and θ are given in table 4.2.

Table 4.2: Gain for ϕ and θ

	K_p	K_d
θ	2.1	2.8
ϕ	2.15	2.5

Here, the maximum angular speed of a motor is considered 600 rad/s (Bouabdallah, 2007). Therefore, the ranges for different control inputs (i.e. U_1, U_2, U_3, U_4) can be defined based on the considered angular speed of each motor from equation (3.58-3.61) as follows:

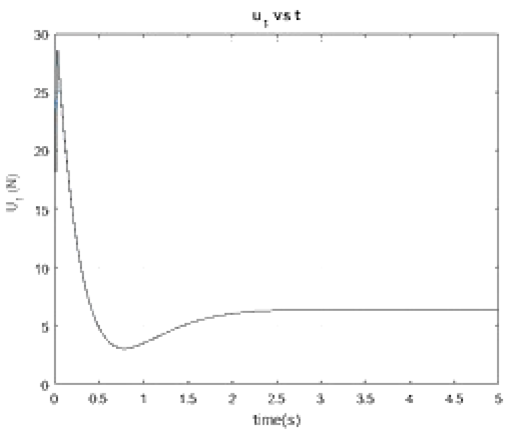
$$0 < U_1 < 45.0720$$

$$-11.2680 < U_2 < 11.2680$$

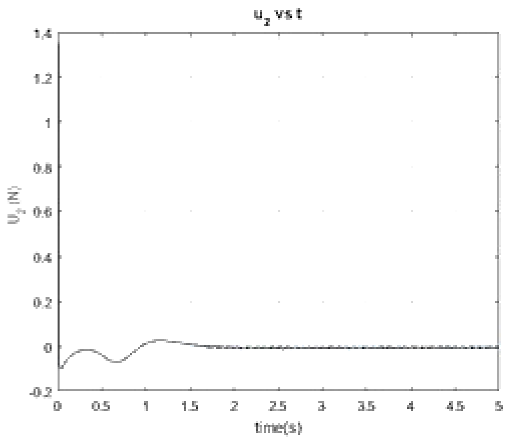
$$-11.2680 < U_3 < 11.2680$$

$$-0.54 < U_4 < 0.54$$

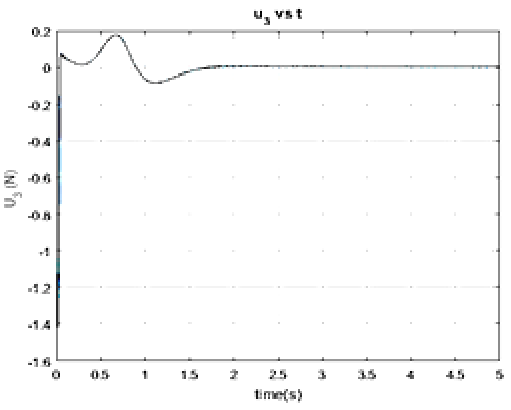
The control effort to the system and the output of the system has also been demonstrated in figures 4.1 and 4.2 respectively.



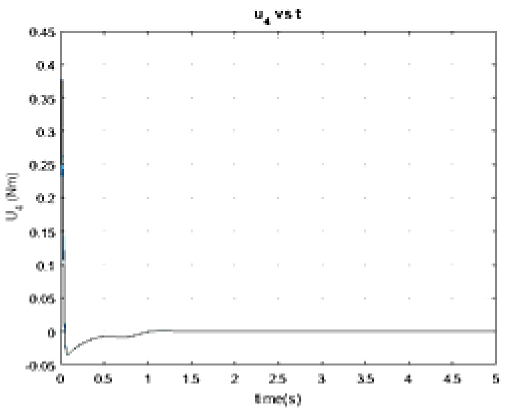
(a) Control Input, U_1



(b) Control Input, U_2



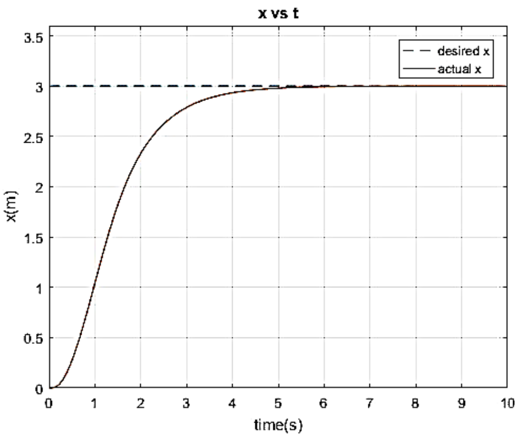
(c) Control Input, U_3



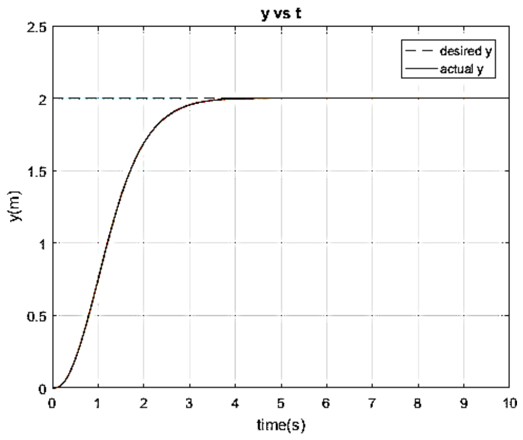
(d) Control Input, U_4

Figure 4.1 Control Inputs of PD controller in Euler angle orientation

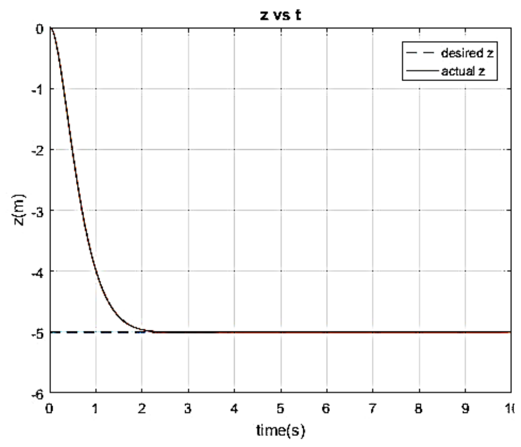
From figures 4.1 and 4.2, it is found that control inputs maintained the ranges and it could successfully achieve the trajectory. However, figure 4.2 shows the direction of z is negative because N-E-D coordinate system has been considered in this work as aforementioned.



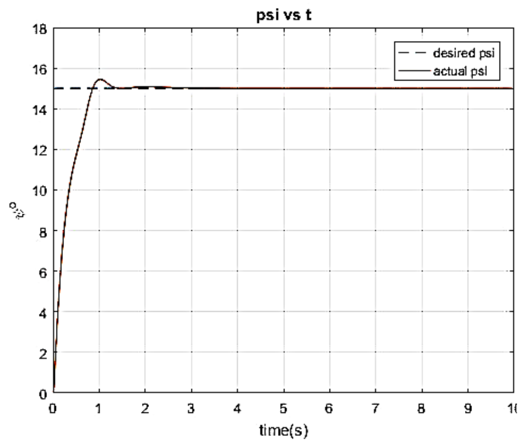
(a) x vs t



(b) y vs t



(c) z vs t



(d) ψ vs t

Figure 4.2 PD controller simulation response in Euler orientation

4.1.1.2 Simulation for Quaternion orientation

In quaternion orientation, similar procedures have been followed like Euler angle orientation. The gains K_p and K_d for position and attitude along with settling time and overshoot have been stated in tables 4.3 and 4.4.

Table 4.3 and 4.4 describes that PD controller could offer promising performance in settling time while in overshoot, significantly along y-axis, it failed to maintain the considerable performance.

Table 4.3: PD controller performance for Quaternion orientation

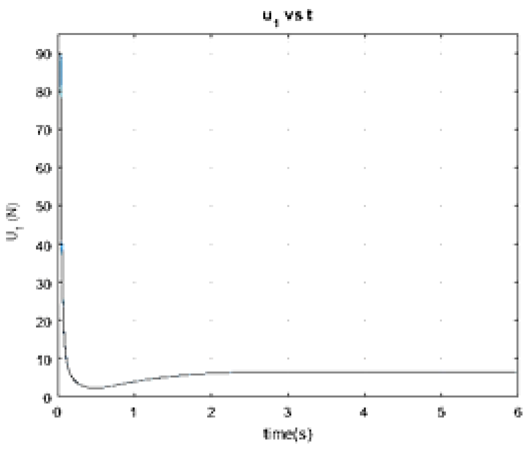
	Desired Value	K_p	K_d	**Settling Time (sec)**	**Overshoot (%)**
x	3	0.7	0.8	2.5943	1.7406
y	2	0.6	0.8	2.6587	13.1248
z	-5	550	35	2.3158	2.7858
q_3	0.132 ($\approx 15^o$)	10	2	1.4479	4.2522

The gains for $q_1$and q_2 are given in table 4.4.

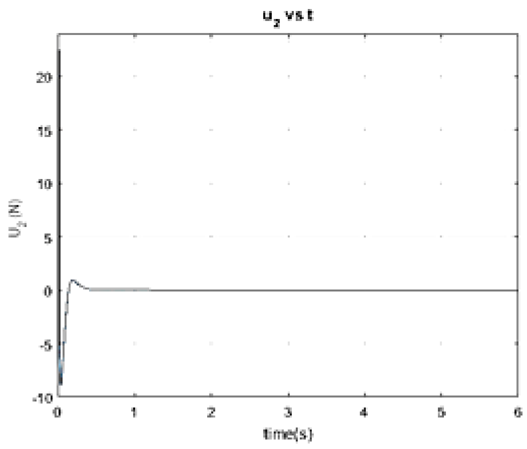
Table 4.4: Gains for q_1 and q_2

	K_p	K_d
q_1	10	2
q_2	10	1

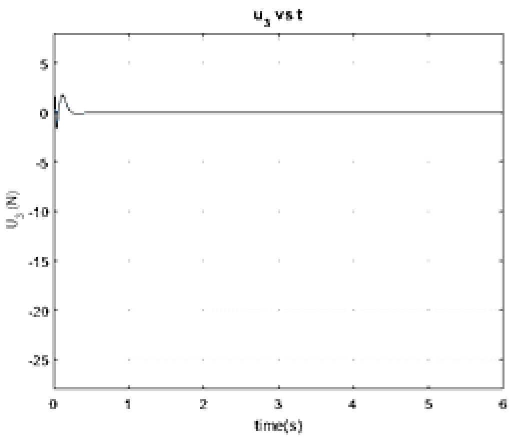
Figure 4.3 demonstrates the control inputs to the system. According to the requirement of this work, $0< U_1<45.0720$ is the range for control input U_1. Notably, figure 4.3 (a), shows that PD controller fails to maintain the range for the control input U_1. Apart from that, it offers some chattering in control input U_4 that is hazardous for motor. Therefore, PD is not able to satiate the requirements significantly at control inputs that have been considered in this work.



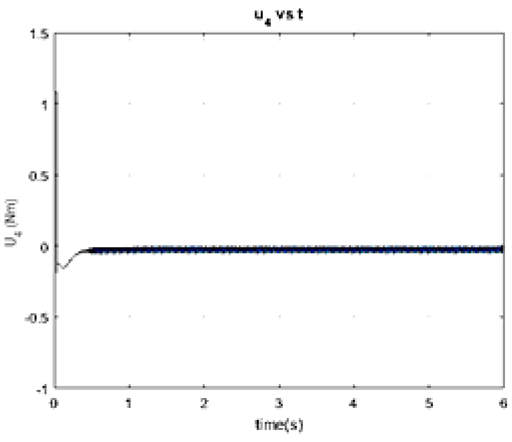
(a) Control Input, U_1



(b) Control Input, U_2



(c) Control Input, U_3



(d) Control Input, U_4

Figure 4.3 Control Inputs of PD controller in Quaternion orientation

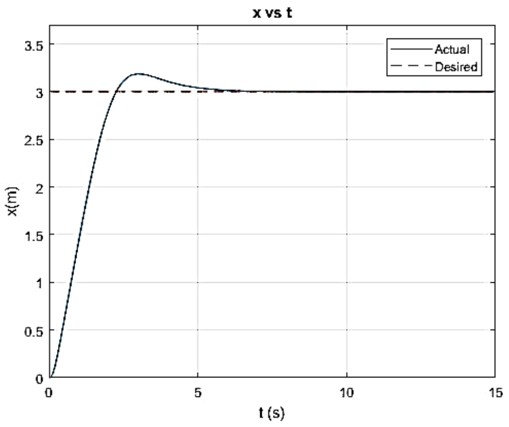
Figure 4.4 confirms that PD controller reach the desired point though the overshoot along y-axis is very high. Table 4.1 and 4.3 indicates that PD controller in quaternion orientation can offer improved performance especially for setting time. On the other hand, it cannot ensure better performance for overshoot when the system follows quaternion orientation.

4.1.2 LQR Controller Simulation

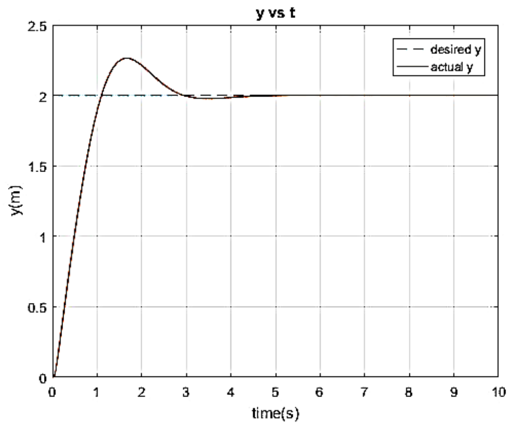
4.1.2.1 Simulation for Euler angle orientation

In LQR controller, the same desired values are considered as for PD controller. The values of Q and R matrices have been stated in table 4.5 and the performance of the system is demonstrated in table 4.6.

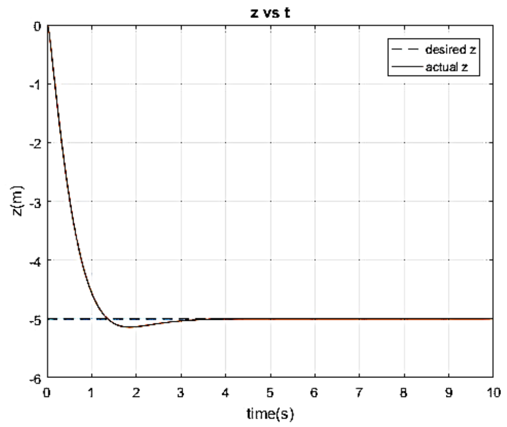
From table 4.6, it is notable that the overshoots along x, y and z-axes are zeros. In addition, settling time along the axes are also very small compared to PD controller. Noted that Q and R matrices are chosen to follow the control input constraints by proper trade-off.



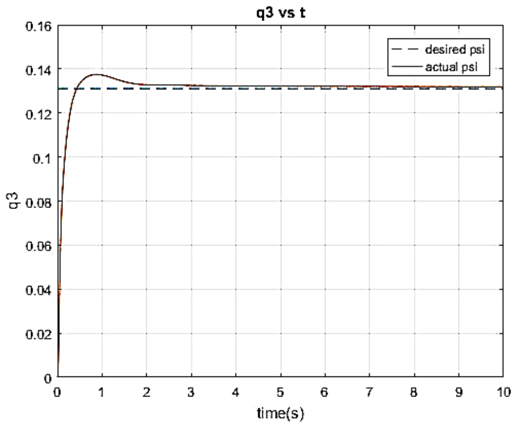
(a) x vs t



(b) y vs t



(c) z vs t



(d) q_3 vs t

Figure 4.4 PD controller simulation response in Quaternion orientation

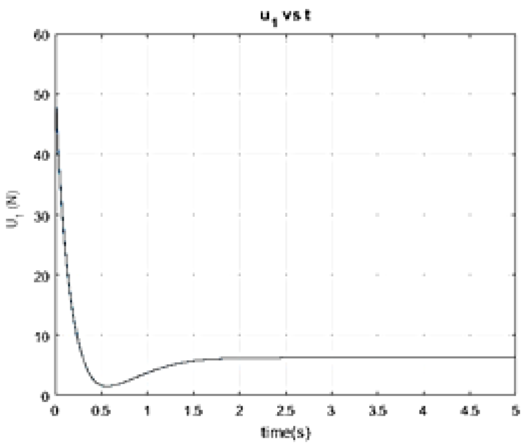
Table 4.5: Q and R matrix for LQR in Euler angle orientation

Q	R
$\begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 35 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 85 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$

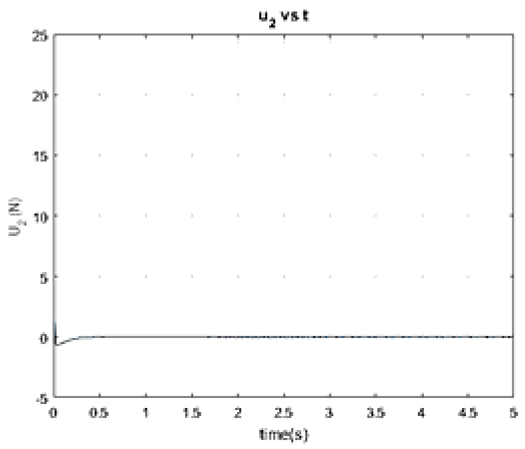
Table 4.6: LQR performance for Euler angle orientation

	Desired Value	Settling Time (sec)	Overshoot (%)
x	3	2.1634	0
y	2	1.8290	0
z	-5	1.8496	0
ψ	15^{o}	0.5992	3.0365

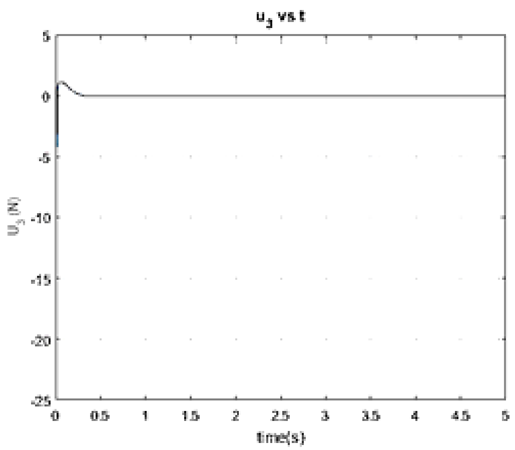
Figure 4.5 illustrates the control input for LQR controller. As Q and R matrices have been chosen considering constraints, the control inputs have maintained the limits. LQR needs very high control effort and it takes longer time to settle down compared to PD controller.



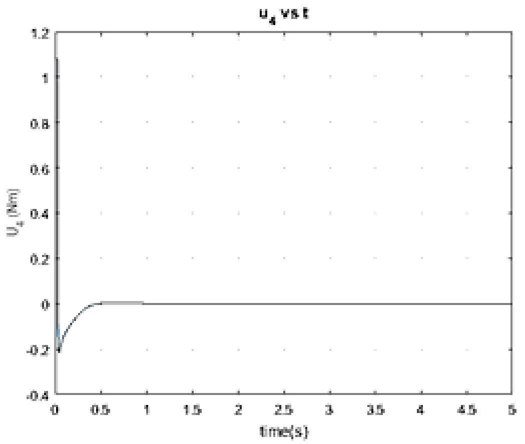
(a) Control Input, U_1



(b) Control Input, U_2

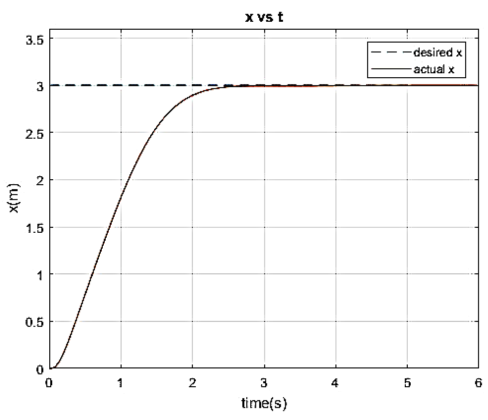


(c) Control Input, U_3

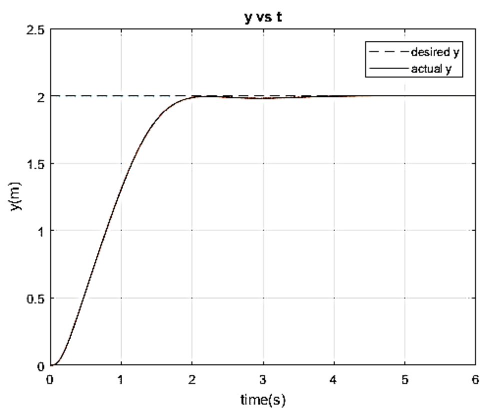


(d) Control Input, U_4

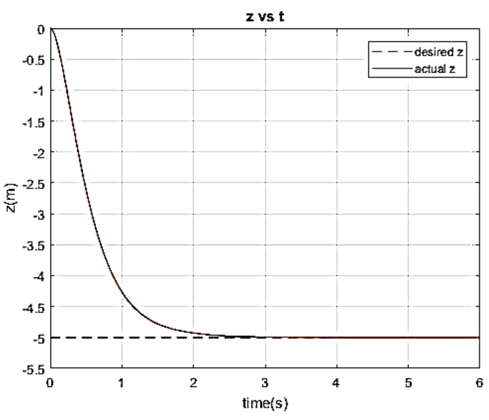
Figure 4.5 Control Inputs for LQR in Euler angle orientation



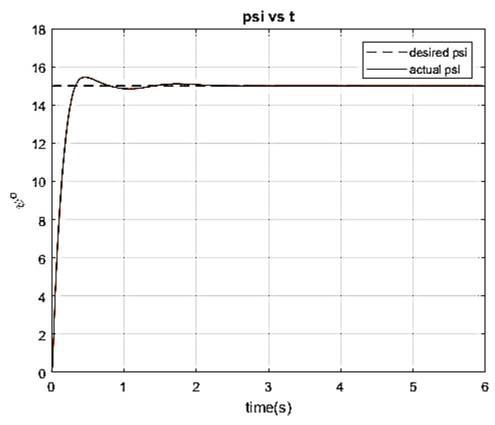
(a) x vs t



(b) y vs t



(c) z vs t



(d) ψ vs t

Figure 4.6 LQR simulation response in Euler angle orientation

Figure 4.6 shows that LQR performs well in case of settling time comparing to PD controller. In addition, it confirms zero overshoot along the axes also.

4.1.2.2 Simulation for Quaternion orientation

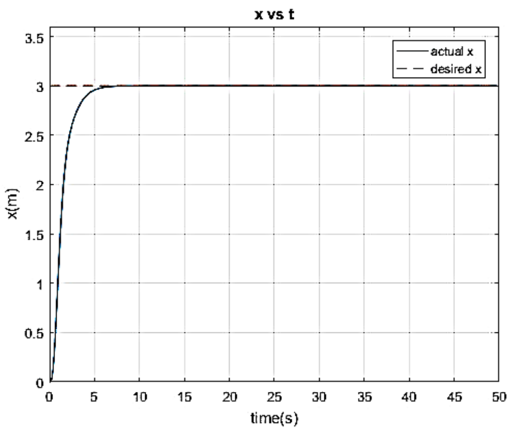
Here, LQR controller is designed for quaternion linear model. Q and R matrices are shown in table 4.7. Table 4.8 depicts the performance of the LQR at the same desired value.

Table 4.7: Q and R matrix for LQR in Quaternion orientation

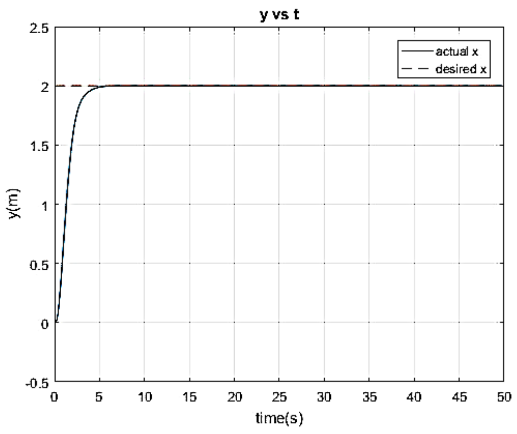
Q	R
$\boldsymbol{I_{12\times12}}$	$I_{4\times4}$

Table 4.8 LQR performance for Quaternion orientation

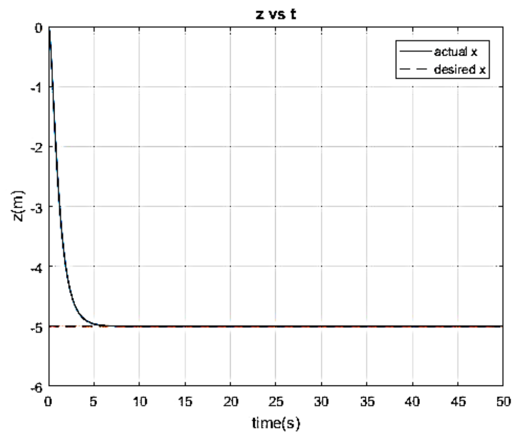
	Desired Value	Settling Time (sec)	Overshoot (%)
x	3	4.6274	0
y	2	3.992	0
z	-5	4.0933	0
q_3	0.131 (≈15°)	3.5380	0.006



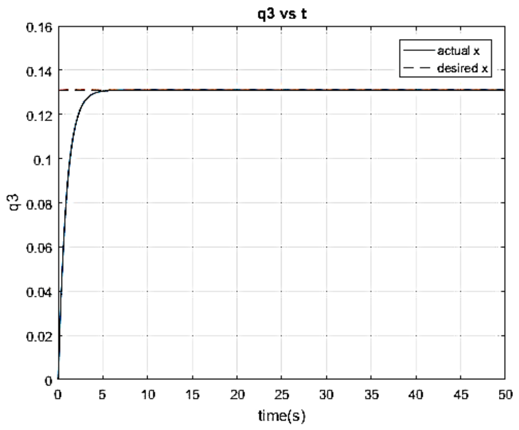
(a) x vs t



(b) y vs t



(c) z vs t

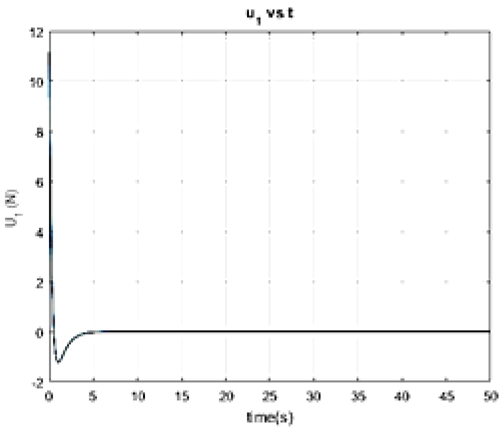


(d) ψ vs t

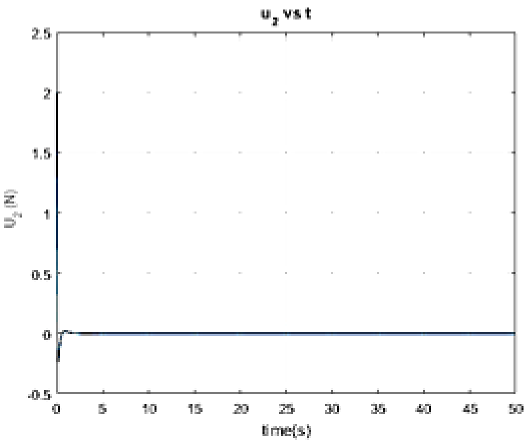
Figure 4.7 LQR simulation response in Quaternion orientation

From table 4.8 and figure 4.7, it is found that LQR ensures significant performance especially in overshoot while in settling time, it offers poor performance comparative to PD controller.

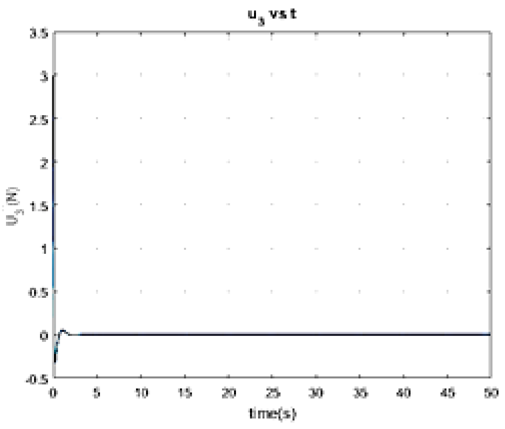
Figure 4.8 depicts that control input U_1 starts from 10N in LQR while for PD controller, it starts from 90N. It implies that PD controller effort requires nine fold of LQR. Therefore, LQR is more efficient in the extent of control input though the output performance of LQR is poorer than PD controller.



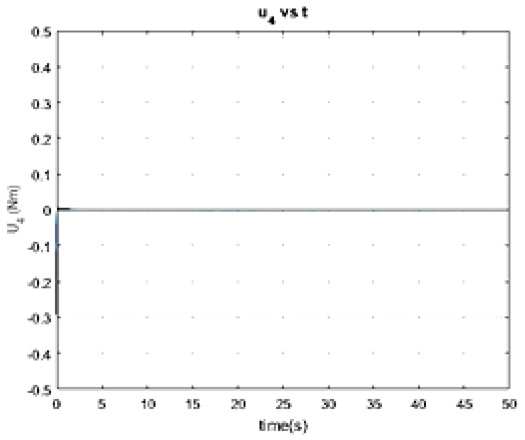
(a) Control Input, U_1



(b) Control Input, U_2



(c) Control Input, U_3



(d) Control Input, U_4

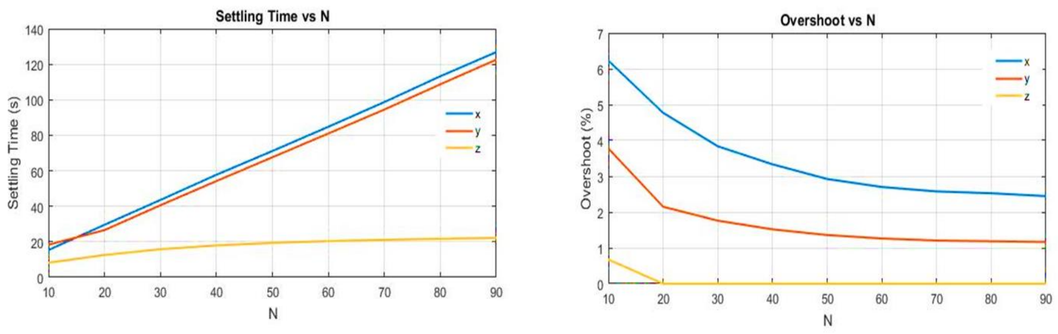
Figure 4.8 Control Inputs for LQR in Quaternion orientation

4.1.3 MPC Controller Simulation

4.1.3.1 Simulation for Euler angle orientation

As the dynamics of the quadrotor is nonlinear, equations are linearized around hovering condition. For this study, prediction horizon, N = 20, control horizon, M = 2, and sampling time, T_s= 0.25 are considered after several initial simulations based on two factors such overshoot and settling time. The effects of different N along x, y and z-axes on settling time and overshoot are shown in figures 4.9 and 4.10. As N increases, settling time increases and overshoot decreases. Figure 4.10 shows the impact of N based on each axis separately.

Based on the comparison among figure 4.5, 4.8 and 4.11, significantly it is found that MPC requires less control effort compared to other two controllers to achieve the same desired position.



(a) Settling time vs N

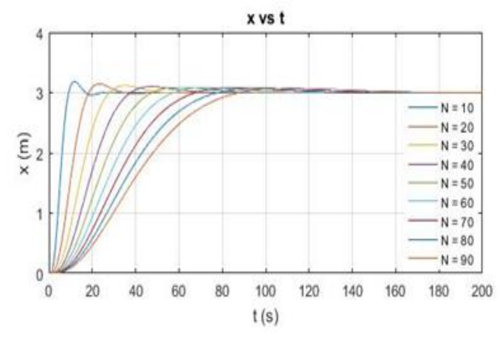
(b) Overshoot vs N

Figure 4.9 Effects of N on (a) settling time and (b) overshoot

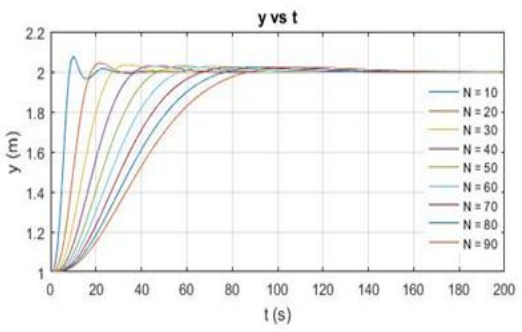
In the meanwhile, table 4.9 depicts MPC needs a few seconds more time to subdue the signal because of its continual optimization process although it is able to maintain acceptable overshoot for all four states. Moreover, it maintains the limits of control inputs accordingly.

Table 4.9: MPC performance in Euler angle orientation

	Desired Value	**Settling Time (sec)**	**Overshoot (%)**
x	3	6.9345	2.2619
y	2	8.4738	2.7672
z	-5	2.8177	0.0200
ψ	15^o	0.5589	1.2149



(a)



(b)

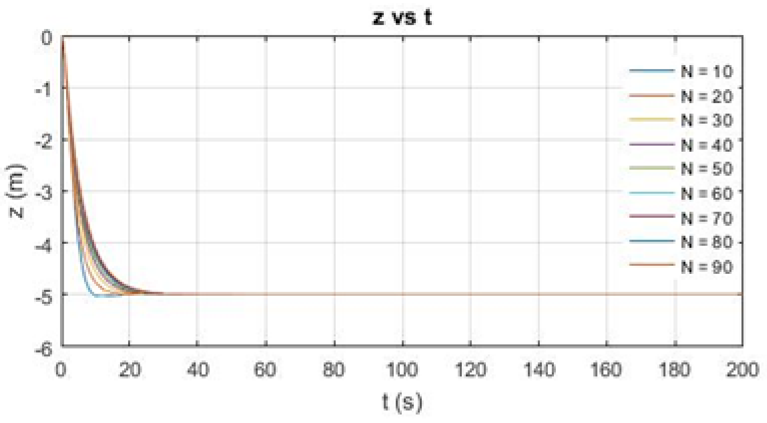
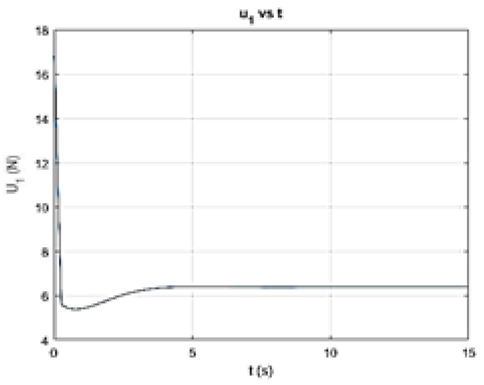
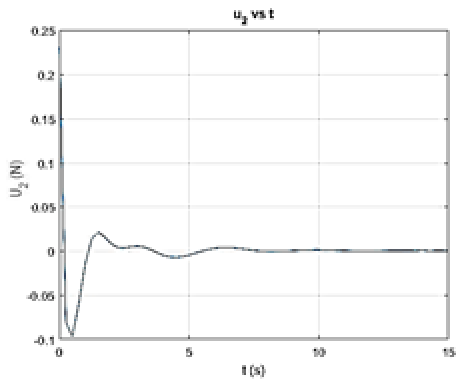


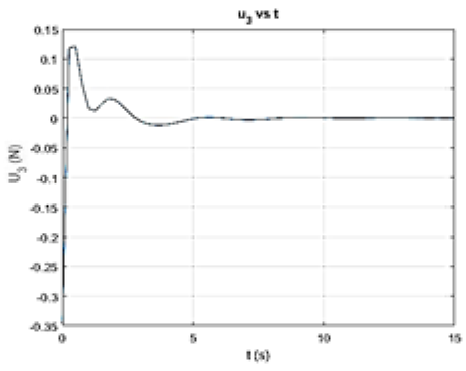
Figure 4.10 Effects of N on (a) x, (b) y and (c) z



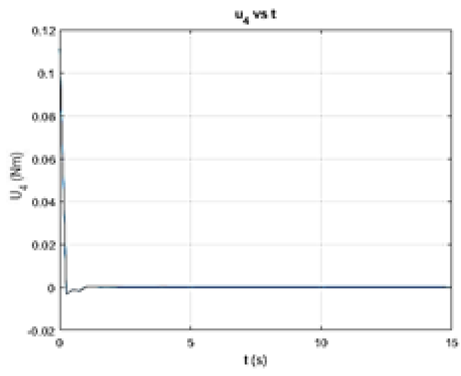
(a) Control Input, U_1



(b) Control Input, U_2



(c) Control Input, U_3



(d) Control Input, U_4

Figure 4.11 Control Inputs for MPC in Euler angle orientation

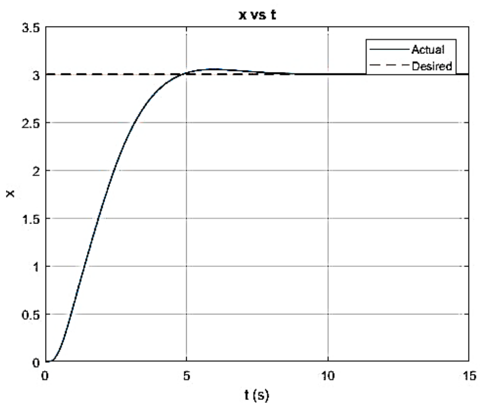
4.1.3.2 Simulation for Quaternion orientation

In quaternion, the nonlinear model is linearized around hover condition. In order to achieve the best performance, the prediction horizon, N=20, control horizon, M=2, sampling time, T_S= 0.25 and proper weight matrices (see in Appendix C) are used.

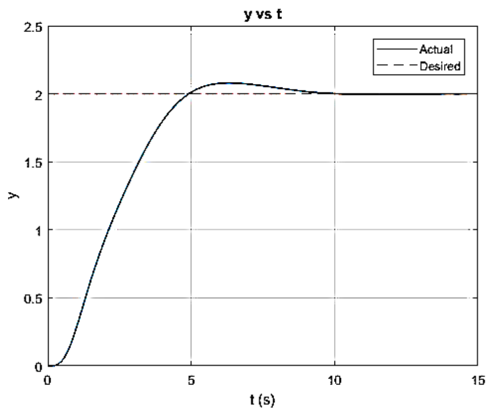
Table 4.10 MPC performance in Quaternion orientation

	Desired Value	**Settling time (sec)**	**Overshoot (%)**
x	3	4.4512	1.6351
y	2	8.1231	3.9754
z	-5	4.9083	2.2303
q_3	0.131 (≈15^o)	1.8986	0.2519

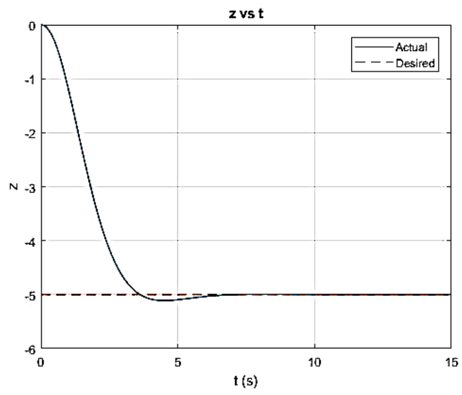
Table 4.10 and figure 4.13 describe the performance of the MPC controller in quaternion orientation. The settling time for MPC is the longest compared to other two controllers. On the other hand, the performance of MPC in overshoot for all the states are significantly better.



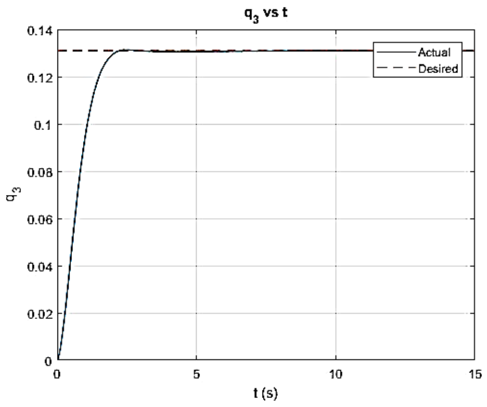
(a) x vs t



(b) y vs t



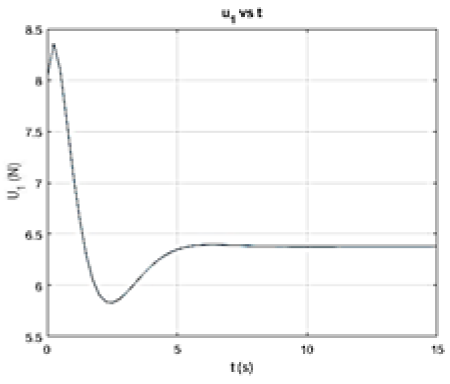
(c) z vs t



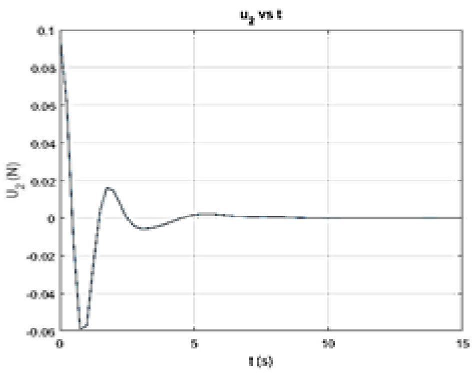
(d) q_3 vs t

Figure 4.12 MPC simulation response in Quaternion orientation

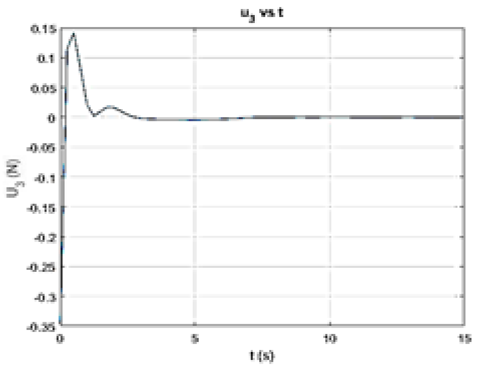
Figure 4.13 shows that the generated control efforts by MPC controllers are the least than the other two controllers, which are safe for the motors and it can make the system stable during flight.



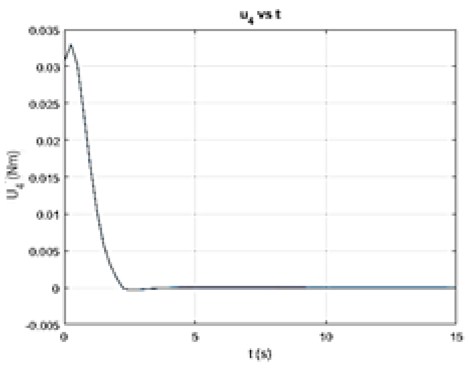
(a) Control Input, U_1



(a) Control Input, U_2



(c) Control Input, U_3

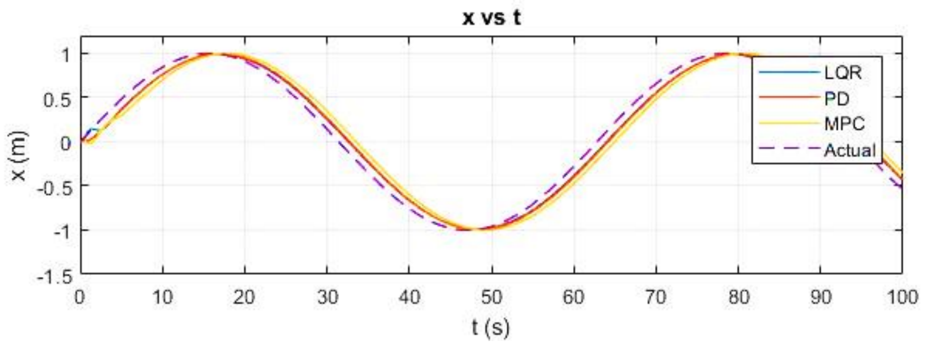


(d) Control Input, U_4

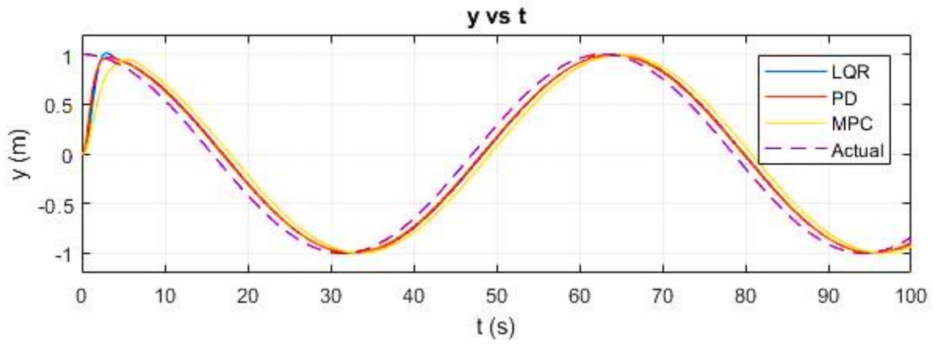
Figure 4.13 Control Inputs for MPC controller in Quaternion orientation

4.2 CONTROLLER PERFORMANCE IN TRAJECTORY VARIATION

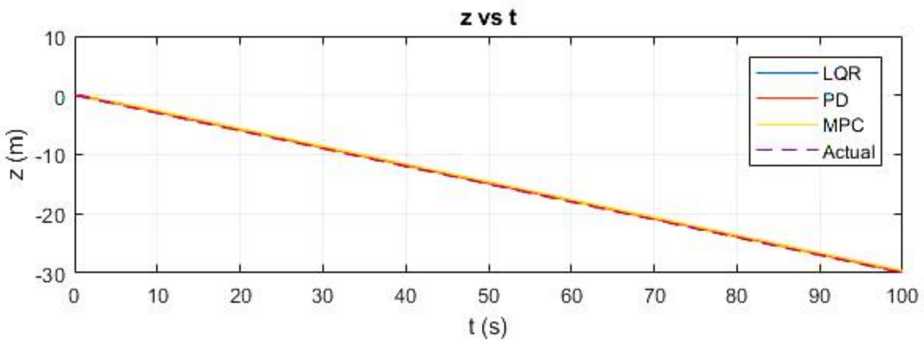
With a view to evaluating the performances of the controllers, system responses for the controllers are depicted in figure 4.14 and 4.15. It shows the response of the position of quadrotor following a helix trajectory for Euler angle orientation.



a) x vs t

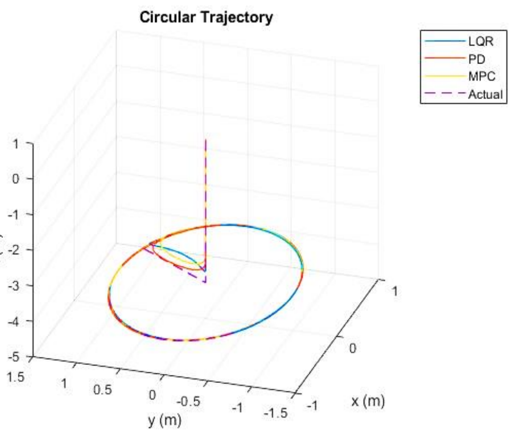


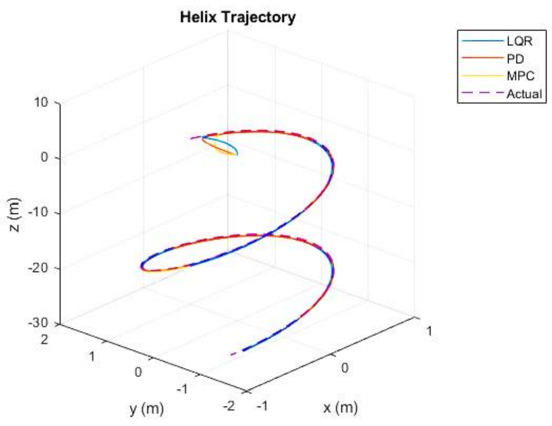
b) y vs t



c) z vs t

Figure 4.14 (a) x vs t, (b) y vs t, (c) z vs t in Euler angle orientation for helix trajectory

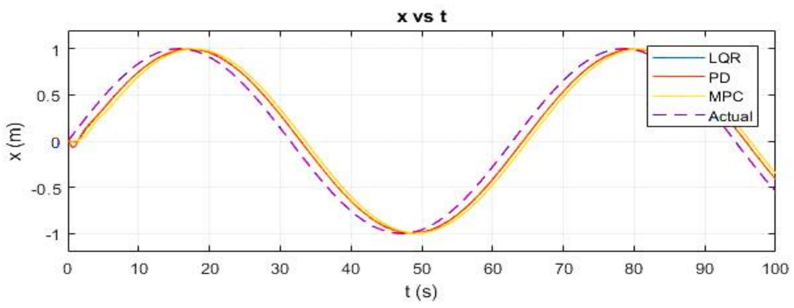




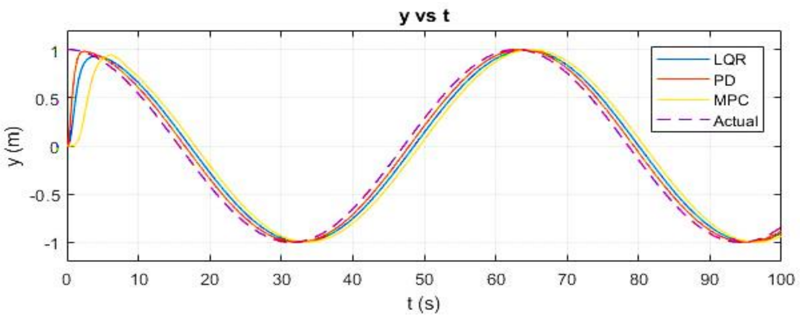
(a) (b)

Figure 4.15 (a) Circular Trajectory, (b) Helix Trajectory in Euler angle orientation

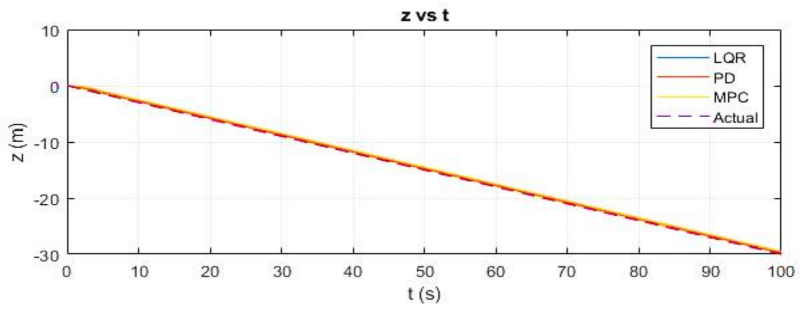
Figure 4.15 shows the performance of the controller in 3D position for both circular and helix trajectories.



a) x vs t



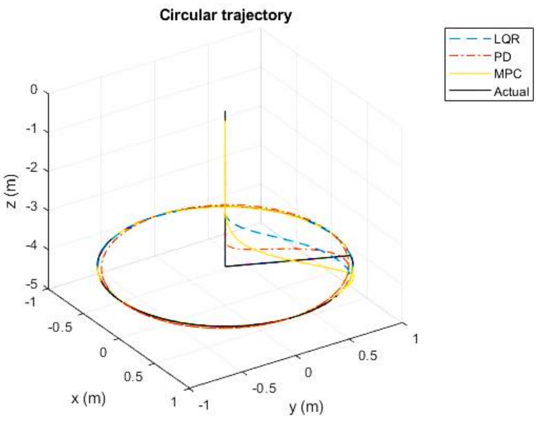
b) y vs t

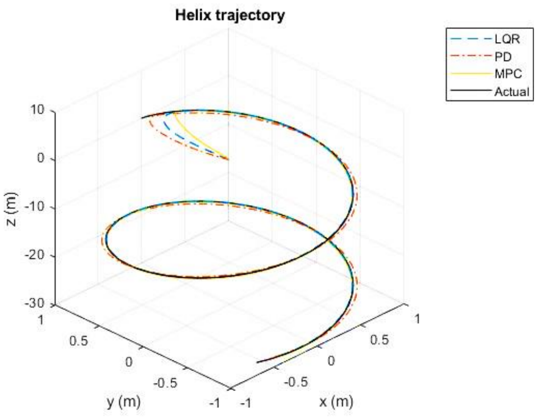


c) z vs t

Figure 4.16 (a) x vs t, (b) y vs t, (c) z vs t in Quaternion orientation

Figure 4.16 depicts the performance of the controller in helix trajectory and it run for 100s for quaternion orientation. Figure 4.18 demonstrates circular and helix trajectories for the controllers in 3D in quaternion orientation.





(a) (b)

Figure 4.17 (a) Circular Trajectory, (b) Helix trajectory in Quaternion orientation

Tables 4.11-4.14 describe the performances of the controllers along x, y and z-axis based on Root Means Square Error approach. Root-Mean-Square (RMS) is an approach to evaluate the accuracy of the data by comparison. The tracking performance of the controller is evaluated using RMS error (RMSE).

Table 4.11 RMSE along x, y and z-axes in Euler angle for helix trajectory

Controller	**RMSE along x-axis (%)**	**RMSE along y-axis (%)**	**RMSE along z-axis (%)**
PD	0.9333	0.5238	0.5024
LQR	0.9499	0.4868	0.4343
MPC	1.7356	1.9665	1.3508

Table 4.12 RMSE along x, y and z-axes in Quaternion for helix trajectory

Controller	**RMSE along x-axis (%)**	**RMSE along y-axis (%)**	**RMSE along z-axis (%)**
PD Quaternion	0.5393	0.8615	1.0474
LQR Quaternion	0.8388	2.5471	2.3180
MPC Quaternion	0.8445	3.9009	2.9437

Table 4.13 RMSE along x, y and z-axes in Euler angle for circular trajectory

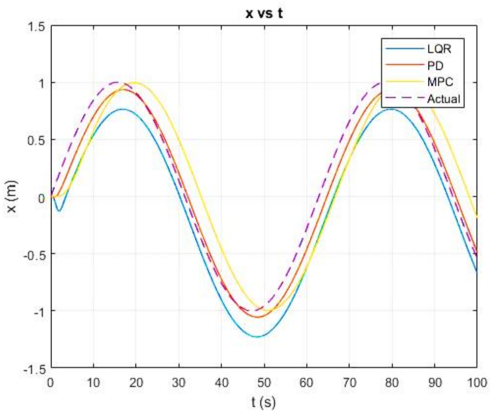
Controller	**RMSE along x-axis (%)**	**RMSE along y-axis (%)**	**RMSE along z-axis (%)**
PD	0.8163	1.4857	0.0965
LQR	1.7146	1.2035	0.3128
MPC	1.2787	6.2729	1.0022

Table 4.14 RMSE along x, y and z-axes in Quaternion for circular trajectory

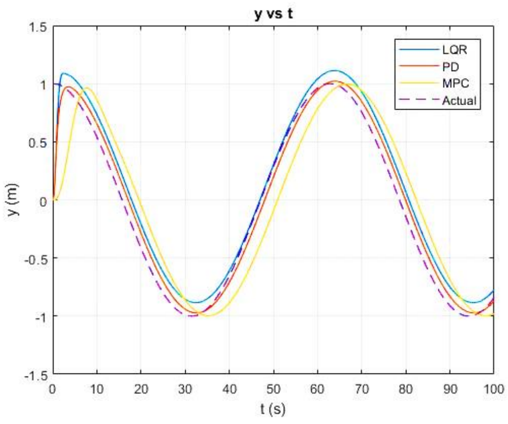
Controller	**RMSE along x-axis (%)**	**RMSE along y-axis (%)**	**RMSE along z-axis (%)**
PD Quaternion	0.6040	0.8812	0.3581
LQR Quaternion	0.9687	2.4359	1.0931
MPC Quaternion	2.8125	3.8419	2.3089

4.3 PERFORMANCE UNDER DISTURBANCES

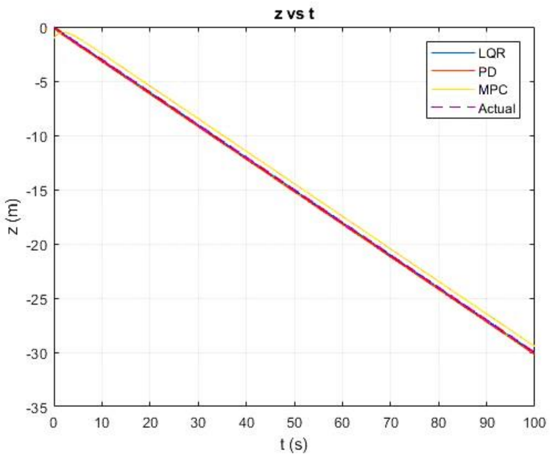
In order to evaluate the performances of the controllers, x, y and z are plotted with respect to time in presence of disturbances as in figure 4.18 and 4.19.



(a) x vs t



(b) y vs t



(c) z vs t

Figure 4.18 Position under disturbances for helix trajectory in Euler angle orientation

Figure 4.18 and tables 4.15-4.16 show that MPC maintains almost same RMSE along x, y and z-axes in both situations (i.e. with disturbances and without disturbances). On the other hand, it is noticeable that both PD and LQR show steady-state error along x and y-axes. Therefore, the disturbance creates an impact on both the controllers even though these controllers could offer promising performance without disturbance environment.

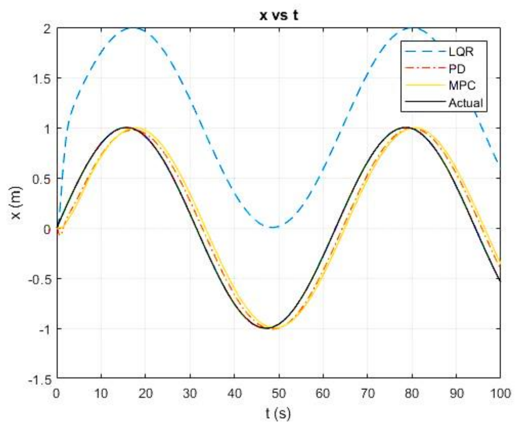
Table 4.15 RMSE along x, y and z-axes under disturbances in Euler angle for helix trajectory

Controller	RMSE along x-axis (%)	RMSE along y-axis (%)	RMSE along z-axis (%)
PD	0.9670	0.4754	0.5241
LQR	3.2462	1.7189	0.1261
MPC	1.7356	1.9665	1.3508

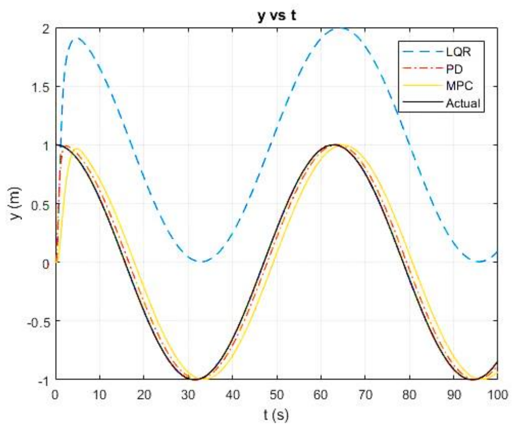
Table 4.16 RMSE along x, y and z-axes under disturbances in Euler angle for circular trajectory

Controller	RMSE along x-axis (%)	RMSE along y-axis (%)	RMSE along z-axis (%)
PD	2.9689	1.9976	9.0425
LQR	5.5368	0.8608	0.2266
MPC	1.8232	6.8588	2.0071

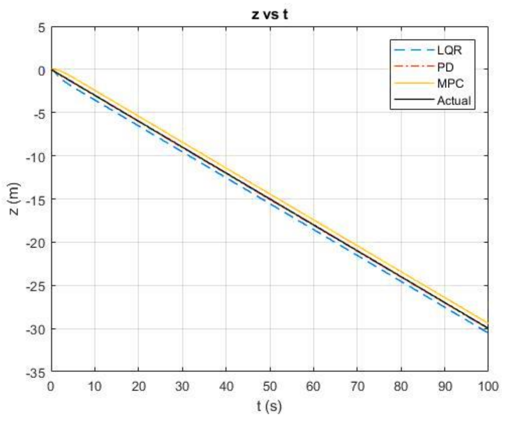
Figure 4.19 and table 4.17-4.18 show that both MPC and PD perform better compared to LQR when the system is influenced by disturbances. PD controller has very small steady-state error while LQR offers much. However, MPC offers promising performance as it does in the trajectories without disturbances and it has been validated by RMSE.



(a) x vs t



(b) y vs t



(c) z vs t

Figure 4.19 Position under disturbances for helix trajectory in quaternion orientation

Table 4.17 RMSE along x, y and z-axes under disturbances in quaternion for helix trajectory

Controller	**RMSE along x-axis (%)**	**RMSE along y-axis (%)**	**RMSE along z-axis (%)**
PD	1.1833	0.8135	0.2432
LQR	48.939	38.2880	2.6607
MPC	0.8524	3.9068	2.9437

Table 4.18 RMSE along x, y and z-axes under disturbances in Quaternion for circular trajectory

Controller	**RMSE along x-axis (%)**	**RMSE along y-axis (%)**	**RMSE along z-axis (%)**
PD	1.2041	0.9069	5.7295
LQR	48.7974	38.5159	19.4157
MPC	1.0322	2.9675	1.2987

From this section, it can be concluded that MPC controller can withstand disturbances and perform almost similar to an environment that is free from disturbances. Additionally, sometimes MPC shows larger tracking error compared to other controllers because it takes time to do optimization processes and tracking onwards.

4.4 CONTROL INPUT COMPARISON

Control effort comparison can be one of the comparison parameters in order to evaluate a controller performance.

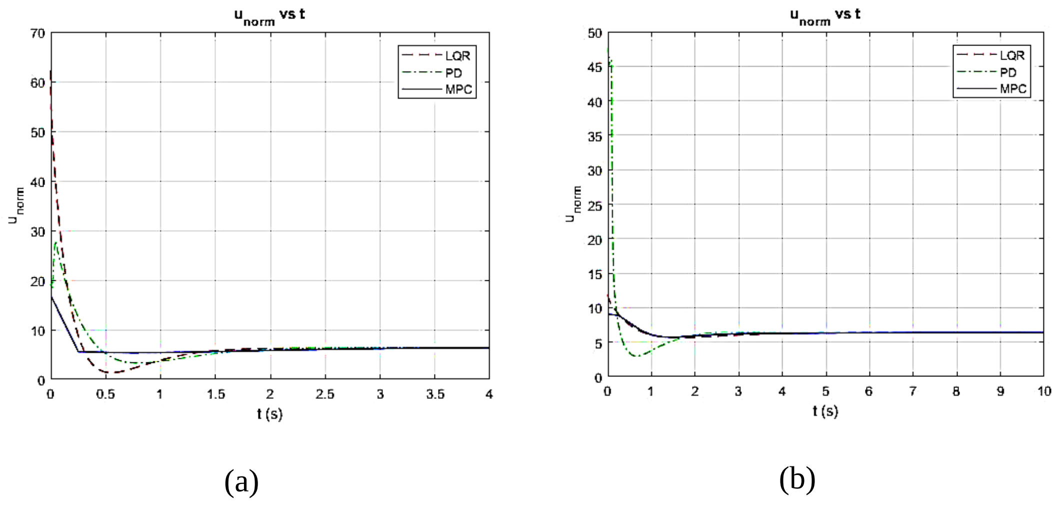


Figure 4.20 u_{norm} comparison in (a) Euler angle orientation (b) Quaternion orientation

Figure 4.20 demonstrates that MPC needs the lowest control effort compared to other controllers in both orientation systems. Moreover, MPC renders lower fluctuations at control inputs compared to other controllers that is safe for the system during flight condition. Therefore, MPC can be the best choice on the basis of control effort.

4.5 DISCUSSION

From the observation of the figure 4.1 to 4.20 and table 4.1 and 4.18, it is found that PD controller can give faster response maintaining reasonable overshoot. In addition, LQR is found more robust than PD controller. However, the effect of disturbances on RMSE is comparatively less in LQR than PD controller.

Besides, MPC offers promising performance based on tracking and control effort comparing to other two controllers that can withstand the uncertainty to the system and able to maintain almost same RMSE with less control effort. Along with that, it could ensure considerable overshoot. However, the settling time in MPC is higher than other two controllers because it estimates system and then moves forward.

In conclusion, a suitable controller for a quadrotor should have some capabilities like robustness with a less and stable control effort that can maintain stability to the system during flight. Therefore, this study can conclude that MPC can be one of the best-suited controllers in order to maintain all these criteria.

CHAPTER FIVE

CONCLUSION

The objectives of the work are to develop a mathematical model of a Mini Aerial Vehicle, quadrotor in two different orientation system such as Euler-angle and Quaternion systems. Moreover, the prime objective is to design a robust controller based on comparison among three different controllers (i.e. PD, LQR and MPC) considering some factors like constraints at inputs, handling model uncertainty, smooth control inputs and tracking accuracy.

Accordingly, two mathematical model have been developed considering two different orientation systems as like Euler angle and quaternion. In addition, aerodynamic drag and moment have been included in order to make model more accurate. Apart from that, rotor dynamics with proper angular velocity limit has been adopted in this work.

Three different control approaches such as PD, LQR and MPC have been applied to control quadrotor. As the main objective of this work is to choose a controller that can ensure the robustness to the system, some factors have been adopted to evaluate the performances of the controllers.

The performance evaluation has been proceeded using MATLAB and Simulink environment. However, the performance of the controllers have been investigated based on overshoot and settling time when the trajectory is considered at a fixed hovering point. As it is highly challenging to find out proper gains for PD controller, Matlab Optimization Toolbox has been considered. However, both PD and LQR perform well and almost similar in settling time. LQR shows comparatively better than others in overshoot and it is almost zeros along every axis. MPC takes much time to settle because of its high computation and predicting feature. However, the performance is still considerable for quadrotor.

Two different trajectories have been considered to evaluate the tracking performance of the controllers wherein RMSE method is used for the evaluation. Here, MPC performs similar to PD and LQR controller in terms of tracking performance. However, the presence of model uncertainty in the system makes MPC different from other two controllers. It maintains almost same RMSE while others have failed to do so. Both PD and LQR controllers start to create impact on the system responses whenever model uncertainty has been considered in the system.

Finally, control input performance has been taken into account to compare the performances among the controllers. The performance have been investigated through comparison among the control inputs individually of the controllers and using control input norm, u_{norm} approach. According to both approach, MPC offers smooth and the lowest control input to the system comparing to other controllers that helps to maintain the system stable in flight condition.

The presented work showed the use of Linear Model Predictive Control (LMPC) approach for different trajectories (i.e. circle and helix) under disturbances. The main advantage of MPC controller that makes it different from other controllers is its predicting behavior. Moreover, it can handle the constraints at control inputs and overcome the model-disturbances without affecting the system response as it is found from simulation results also.

The most crucial part of designing an MPC model includes choosing proper prediction horizon, control horizon and sample time because these all effect on the system stability. In addition, proper weight matrices for inputs and outputs also plays an important role in designing MPC that helps to achieve the desired trajectory. This study has successfully demonstrated that MPC is able to track properly with minimal RMSE along with minimal control efforts in presence of model uncertainty in the system.

In future, nonlinear Model Predictive Control (NMPC) approach will be designed that is expected to be more suitable for nonlinear quadcopter model. In this work, MPC is designed for linear model that motivates to move one-step further in future as designing MPC for nonlinear model that is expected to be more accurate for the system. In this work, it is considered that sensors work perfectly but in real situation, noise, and disturbances may exist. Therefore, in future work, the controllers should be applied in the real-time hardware to evaluate the performances of the controllers.

REFERENCES

Ahmad, A., Tahar, K. N., Udin, W. S., Hashim, K. A., Darwin, N., Hafis, M., Azmi, S. M. (2013). *Digital aerial imagery of unmanned aerial vehicle for various applications*. Paper presented at the Control System, Computing and Engineering (ICCSCE), 2013 IEEE International Conference on Nov 29.

Alexis, K., Nikolakopoulos, G., & Tzes, A. (2010). *Experimental model predictive attitude tracking control of a quadrotor helicopter subject to wind-gusts.* Paper presented at the Control & Automation (MED), 2010 18th Mediterranean Conference on June 23.

Amir, A. R., & Weiss, S. I. (2003). Aerospace Products, Manufacturers, And Markets. In T. E. o. E. Britannica (Ed.), *Aerospace industry*.

Araar, O., & Aouf, N. (2014). *Full linear control of a quadrotor UAV, LQ vs H∞.* Paper presented at the Control (CONTROL), 2014 UKACC International Conference on July 9.

Argentim, W. C. R., Paulo E. Santos, Renato A. Aguiar. (2013). PID, LQR and LQR-PID on a Quadcopter Platform. *IEEE*. doi:10.1109/ICIEV.2013.6572698

Bailey, M. W. (2012). *Unmanned aerial vehicle path planning and image processing for orthoimagery and digital surface model generation.* Vanderbilt University.

Basri, M., Ariffanan, M., Danapalasingam, K. A., & Husain, A. R. (2014). Design and optimization of backstepping controller for an underactuated autonomous quadrotor unmanned aerial vehicle. *Transactions of FAMENA, 38*(3), 27-44.

Bemporad, A., Morari, M., & Lawrence Ricker, N. (2017a). Model Predictive Control Toolbox™ User's Guide *QP Solver* (pp. 2-37): The MathWorks, Inc.

Bemporad, A., Morari, M., & Lawrence Ricker, N. (2017b). *Model Predictive Control Toolbox™ User's Guide* (Version 6.0 ed.): The MathWorks, Inc.

Ben Ammar, N., Bouallègue, S., & Haggège, J. (2016). *Modeling and sliding mode control of a quadrotor unmanned aerial vehicle.* Paper presented at the Proceedings of the 3th international conference on automation, control engineering and computer science (ACECS 2016), Hammamet, Tunisia.

Bendaas, I., & Naceri, F. (2013). A new method to minimize the chattering phenomenon in sliding mode control based on intelligent control for induction motor drives. *Serbian journal of electrical engineering, 10*(2), 231-246.

Bolkcom, C. (2004). *Homeland security: Unmanned aerial vehicles and border surveillance*. Retrieved from Washington DC:

Bonna, R., & Camino, J. (2015). *Trajectory Tracking Control of a Quadrotor Using Feedback Linearization.* Paper presented at the Proc. of the XVII International Symposium on Dynamic Problems of Mechanics DINAME-2015.–2015.

Bouabdallah, S. (2007). *Design and control of quadrotors with application to autonomous flying.* (PhD), Lausanne, EPFL.

Bouabdallah, S., Noth, A., & Siegwart, R. (2004). *PID vs LQ control techniques applied to an indoor micro quadrotor.* Paper presented at the Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference.

Bouabdallah, S., & Siegwart, R. (2005). *Backstepping and sliding-mode techniques applied to an indoor micro quadrotor.* Paper presented at the Robotics and

Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference.

Boudergui, K., Carrel, F., Domenech, T., Guenard, N., Poli, J.-P., & Ravet, A. (2011). *Development of a drone equipped with optimized sensors for nuclear and radiological risk characterization.* Paper presented at the Advancements in Nuclear Instrumentation Measurement Methods and their Applications (ANIMMA), 2011 2nd International Conference on June 6.

Bouffard, P. (2012). *On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments.* University of California, Berkeley.

Bresciani, T. (2008). *Modelling, identification and control of a quadrotor helicopter.* (Master MSc Theses), Lund University, Sweden.

Carino, J., Abaunza, H., & Castillo, P. (2015a). *Quadrotor quaternion control.* Paper presented at the Unmanned Aircraft Systems (ICUAS), 2015 International Conference on June 9.

Carino, J., Abaunza, H., & Castillo, P. (2015b). *Quadrotor quaternion control.* Paper presented at the Unmanned Aircraft Systems (ICUAS), 2015 International Conference on.

Carrillo, L. R. G., López, A. E. D., Lozano, R., & Pégard, C. (2012). *Quad rotorcraft control: vision-based hovering and navigation*: Springer Science & Business Media.

Casbeer, D. W., Beard, R. W., McLain, T. W., Li, S.-M., & Mehra, R. K. (2005). *Forest fire monitoring with multiple small UAVs.* Paper presented at the American Control Conference, 2005. Proceedings of the 2005.

Chapman, A. (2017). *Types of Drones: Multi-Rotor vs Fixed-Wing vs Single Rotor vs Hybrid VTOL.* Retrieved February 14, 2017 https://www.auav.com.au/articles/drone-types/

Chung, C.-W., & Chang, Y. Design of Adaptive Backstepping controller for Systems with Mismatched Perturbations to Achieve Asymptotical Stability.

Clough, B. (2002). *Metrics, Schmetrics! How do you Track A UAV's Autonomy?* Paper presented at the 1st UAV Conference.

Conner, M. (2017). Helios Prototype Flying Wing. Retrieved February 14, 2017 https://www.nasa.gov/centers/dryden/multimedia/imagegallery/Helios/ED03-0152-2.html

Cook, D., & Das, S. K. (2004). *Smart environments: Technology, protocols and applications* (Vol. 43): John Wiley & Sons.

Cowling, I. D., Yakimenko, O. A., Whidborne, J. F., & Cooke, A. K. (2007). *A prototype of an autonomous controller for a quadrotor UAV.* Paper presented at the Control Conference (ECC), 2007 European.

Cubillos, X., Celia, d. S., & Luiz Carlos, G. (2010). Using of H-infinity control method in attitude control system of rigid-flexible satellite. *Mathematical Problems in Engineering, 2009.*

Cuthbertson, A. (2016). Firefighting, Criminal Chasing, Rescue Drones Coming To Europe. *Newsweek.*

Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix, 58*(15-16), 1-35.

Efe, M. Ö. (2011). Neural Network Assisted Computationally Simple $PI^{\lambda}D^{\mu}$ Control of a Quadrotor UAV. *IEEE Transactions on Industrial Informatics, 7*(2), 354-361.

ElKholy, H. (2014). *Dynamic modeling and control of a quadrotor using linear and nonlinear approaches.* Master thesis, The American University in Cairo.

Embention. (2016). *NM& F300, New High Performance Fixed Wing UAV*. Retrieved from

Esteves, D. J. F. (2014). Development and Experimental Validation of an Indoor Low Cost Quadrotor: Hover Stabilization with Altitude Control.

Fang, Z., & Gao, W. (2011). *Adaptive integral backstepping control of a micro-quadrotor*. Paper presented at the Intelligent Control and Information Processing (ICICIP), 2011 2nd International Conference on.

Feng, L. X., Mei, G. L., Wei, G. Z., & Qing, S. Y. (2013). A UAV Allocation Method for Traffic Survillance in Sparse Road Network. *Journal of Highway and Transportation Research and Development, 7*(2), 81-87.

Fresk, E., & Nikolakopoulos, G. (2013). *Full quaternion based attitude control for a quadrotor*. Paper presented at the Control Conference (ECC), 2013 European on July 17.

Gevaert, C. M., Suomalainen, J., Tang, J., & Kooistra, L. (2015). Generation of spectral–temporal response surfaces by combining multispectral satellite and hyperspectral UAV imagery for precision agriculture applications. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 8*(6), 3140-3146.

Gibbs, Y. (2014a). NASA Armstrong Fact Sheet: Helios Prototype. Retrieved February 14, 2017 https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-068-DFRC.html

Gibbs, Y. (2014b). NASA Armstrong Fact Sheet: Pathfinder Solar-Powered Aircraft. https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-034-DFRC.html

Girard, A. R., Howell, A. S., & Hedrick, J. K. (2004). *Border patrol and surveillance missions using multiple unmanned air vehicles*. Paper presented at the Decision and Control, 2004. 43rd IEEE Conference on Dec 17.

Grenzdörffer, G., Engel, A., & Teichert, B. (2008). The photogrammetric potential of low-cost UAVs in forestry and agriculture. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 31*(B3), 1207-1214.

Haddal, C. C., & Gertler, J. (2010). *Homeland security: Unmanned aerial vehicles and border surveillance*. Retrieved from Library of Congree Washington DC: Retrieved February 14, 2017 http://www.dtic.mil/docs/citations/ADA524297

Hausamann, D., Zirnig, W., Schreier, G., & Strobl, P. (2005). Monitoring of gas pipelines–a civil UAV application. *Aircraft Engineering and Aerospace Technology, 77*(5), 352-360.

Hoffmann, G., Huang, H., Waslander, S., & Tomlin, C. (2007). *Quadrotor helicopter flight dynamics and control: Theory and experiment*. Paper presented at the AIAA Guidance, Navigation and Control Conference and Exhibit.

Horn, B. K. (2001). Some notes on unit quaternions and rotation. *Lecture handouts*.

Hou, H., Zhuang, J., Xia, H., Wang, G., & Yu, D. (2010). *A simple controller of minisize quad-rotor vehicle*. Paper presented at the Mechatronics and Automation (ICMA), 2010 International Conference on.

Hugenholtz, C. H., Moorman, B. J., Riddell, K., & Whitehead, K. (2012). Small unmanned aircraft systems for remote sensing and earth science research. *Eos, Transactions American Geophysical Union, 93*(25), 236-236.

Huo, X., Huo, M., & Karimi, H. R. (2014). Attitude stabilization control of a quadrotor UAV by using backstepping approach. *Mathematical Problems in Engineering, 2014*.

ICAO. (2011). *ICAO Circular 328-AN/190*. Canada: International Civil Aviation Organization.

Islam, M., Okasha, M., & Idres, M. (2017). *Dynamics and control of quadcopter using linear model predictive control approach.* Paper presented at the IOP Conference Series: Materials Science and Engineering.

Joyo, M. K., Ahmed, S. F., & Hazry, D. (2013). Position controller design for quad-rotor under perturbed condition. *Wulfenia Journal, 20*(7), 178-189.

Junior, J. C. V., De Paula, J. C., Leandro, G. V., & Bonfim, M. C. (2013). Stability control of a quad-rotor using a PID controller. *Brazilian Journal of Instrumentation and Control, 1*(1), 15-20.

Kalman, R. E. (1970). *Lectures on controllability and observability*. Retrieved from Retrieved February 14, 2017 http://www.dtic.mil/dtic/tr/fulltext/u2/704617.pdf

Kamps, H. J. (2017). *This flappy bird-drone keeps airports safe*. Retrieved February 14, 2017 https://techcrunch.com/2017/01/07/is-it-a-bird-is-it-a-plane-well-it-is-a-drone-actually/

Karakoc, T. H., Ozerdem, M. B., Sogut, M. Z., Colpan, C. O., Altuntas, O., & Açıkkalp, E. (2016). *Sustainable Aviation: Energy and Environmental Issues*: Springer.

Keane, J. F., & Carr, S. S. (2013). A brief history of early unmanned aircraft. *Johns Hopkins APL Technical Digest, 32*(3), 558-571.

Kehlenbeck, A. (2014). *Quaternion-based control for aggressive trajectory tracking with a micro-quadrotor UAV.* University of Maryland, College Park.

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics, 29*(2), 315-378.

Kodgirwar, Manish Vivek Kumar, & Sushant Sawant Shegokar. (2014). Design of control system for quadcopter using Complementary Filter and PID controller. *Internatioanl Journal of Engineering Research and Technology, 3*(4).

Kozák, Š. (2012). *Advanced control engineering methods in modern technological applications.* Paper presented at the 13th International Carpathian Control Conference (ICCC), 2012.

Koziol Jr, J. S. (1971). *Simulation model for the Piper PA-30 light maneuverable aircraft in the final approach*. Retrieved from

Kulumani, S., & Lee, T. (2017). Constrained Geometric Attitude Control on SO (3). *International Journal of Control, Automation, and Systems, 15*(6).

Kurtz, M. J., & Henson, M. A. (1998). Feedback linearizing control of discrete-time nonlinear systems with input constraints. *International Journal of Control, 70*(4), 603-616.

Lanzon, A., Freddi, A., & Longhi, S. (2014). Flight control of a quadrotor vehicle subsequent to a rotor failure. *Journal of Guidance, Control, and Dynamics, 37*(2), 580-591.

Lee, B. (2013). TDR-1. Retrieved February 14, 2017d from Retrieved February 14, 2017 http://www.nnapprentice.com/alumni/letter/TDR_1.pdf

Leishman, J. G. (2002). The breguet-richet quad-rotor helicopter of 1907. *Vertiflite, 47*(3), 58-60.

Levant, A. (2007). Principles of 2-sliding mode design. *automatica, 43*(4), 576-586.

Li, J., & Li, Y. (2011). *Dynamic analysis and PID control for a quadrotor.* Paper presented at the Mechatronics and Automation (ICMA), 2011 International Conference on.

Lindblom, S., & Lundmark, A. (2015). Modelling and control of a hexarotor UAV.

Long, Y., Lyttle, S., Pagano, N., & Cappelleri, D. J. (2012). *Design and quaternion-based attitude control of the omnicopter mav using feedback linearization.* Paper presented at the ASME International Design Engineering Technical Conference (IDETC).

Madani, T., & Benallegue, A. (2006). *Backstepping control for a quadrotor helicopter.* Paper presented at the Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.

Marshall, D. M., Barnhart, R. K., Hottman, S. B., Shappee, E., & Most, M. T. (2016). *Introduction to unmanned aircraft systems*: Crc Press.

Mathworks. (2018). System Identification Toolbox. Retrieved April 23, 2018. https://www.mathworks.com/products/sysid.html

McDonald, G. (2016). *New Wing Design Powers Crazy Bat Drones*. Retrieved February 14, 2016. https://www.seeker.com/new-wing-design-powers-crazy-bat-drones-1770896759.html

Murphy, D., & Cycon, J. (1999). Applications for mini VTOL UAV for law enforcement. *SPIE, 3577*. doi:doi:10.1117/12.336986

Murrow, H., & Eckstrom, C. (1979). Drones for Aerodynamic and Structural Testing (DAST)-A Status Report. *Journal of Aircraft*, *16*(8), 521-526.

NASA. (2015). Watch NASA Langley's "Safe2Ditch" UAV Crash Management Technology Webinar! Retrieved April 23, 2017d from https://www.nasa.gov/feature/register-for-nasa-s-safe2ditch-uav-technology-webinar/

NASA Armstrong Fact Sheet: Altus II. (2014). Retrieved April 24, 2017d from https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-058-DFRC.html

Nonami, K. (2007). Prospect and recent research & development for civil use autonomous unmanned aircraft as UAV and MAV. *Journal of system Design and Dynamics*, *1*(2), 120-128.

Padhee, S. (2014). Controller design for temperature control of heat exchanger system: Simulation Studies. *WSEAS Trans. Syst. Contol, 9*, 485-491.

Pipatpaibul, P.-i., & Ouyang, P. (2013). Application of online iterative learning tracking control for quadrotor UAVs. *ISRN robotics, 2013*.

Pop, C. I., & Dulf, E. H. (2011). Robust feedback linearization control for reference tracking and disturbance rejection in nonlinear systems *Recent Advances in Robust Control-Novel Approaches and Design Methods*: InTech.

Raffo, G. V., Ortega, M. G., & Rubio, F. R. (2008). MPC with Nonlinear $\mathscr{H}\infty$ Control for Path Tracking of a Quad-Rotor Helicopter. *IFAC Proceedings Volumes, 41*(2), 8564-8569.

Raffo, G. V., Ortega, M. G., & Rubio, F. R. (2011). Nonlinear H∞ controller for the quad-rotor helicopter with input coupling. *IFAC Proceedings Volumes, 44*(1), 13834-13839.

Reyes-Valeria, E., Enriquez-Caldera, R., Camacho-Lara, S., & Guichard, J. (2013). *LQR control for a quadrotor using unit quaternions: Modeling and simulation.* Paper presented at the Electronics, Communications and Computing (CONIELECOMP), 2013 International Conference on.

Rosenberg, A. S. (2009). *An evaluation of a UAV guidance system with consumer grade GPS receivers*: ProQuest.

Runcharoon, K., & Srichatrapimuk, V. (2013). *Sliding mode control of quadrotor.* Paper presented at the Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on May 9.

Runge, H., Rack, W., Ruiz-Leon, A., & Hepperle, M. (2007). *A solar powered hale-uav for arctic research.* Paper presented at the 1st CEAS European Air and Space Conference on Sept.

Saari, H., Pellikka, I., Pesonen, L., Tuominen, S., Heikkilä, J., Holmlund, C., . . . Antila, T. (2011). *Unmanned Aerial Vehicle (UAV) operated spectral camera system for forest and agriculture applications.* Paper presented at the SPIE Remote Sensing.

Sabatino, F. (2015). *Quadrotor control: modeling, nonlinearcontrol design, and simulation.* (Master), KTH Royal Institute of Technology, Stockholm, Sweden.

Sadeghzadeh, I., Abdolhosseini, M., & Zhang, Y. (2012). Payload drop application of unmanned quadrotor helicopter using gain-scheduled PID and model predictive control techniques. *Intelligent robotics and applications*, 386-395.

Santos, M., Lopez, V., & Morata, F. (2010). *Intelligent fuzzy controller of a quadrotor.* Paper presented at the Intelligent Systems and Knowledge Engineering (ISKE), 2010 International Conference on.

Sawyer, S. (2015). *Gain-Scheduled Control of a Quadcopter UAV.* University of Waterloo.

Schaft, & Arjan. (2000). *L2-gain and passivity techniques in nonlinear control*: Springer.

Schroer, R. (2003). UAVs: the future.[A century of powered flight: 1903-2003]. *IEEE Aerospace and Electronic Systems Magazine, 18*(7), 61-63.

Shahrokhi, M., & Zomorrodi, A. (2013). Comparison of PID controller tuning methods. *Department of Chemical & Petroleum Engineering Sharif University of Technology*.

Shtessel, Y., Edwards, C., Fridman, L., & Levant, A. (2014). Introduction: Intuitive theory of sliding mode control *Sliding Mode Control and Observation* (pp. 1-42): Springer.

Sorensen, A. (2010). Autonomous control of a miniature quadrotor following fast trajectories. *Aalborg University, Denmark, 12.*

Sözen, V. (2014). *Optimal deployment of unmanned aerial vehicles for border surveillance.* NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIFORNIA.

Spooner, S. (1923). A successful french helicopter. *Flight International, XVI,* 47.

Staff, D. I. D. (2012). *Thailand's Insurgency: The Blimp and I.* Retrieved February 14, 2017 http://www.defenseindustrydaily.com/Thailand-Contracts-Aria-for-Blimps-Communications-05401/

Swamp, A. (2016). *Second order sliding mode control for quadrotor.* Paper presented at the 2016 IEEE First International Conference on Control, Measurement and Instrumentation (CMI).

Tang, L., & Shao, G. (2015). Drone remote sensing for forestry research and practices. *Journal of Forestry Research, 26*(4), 791-797.

Team, C. U. A. (2006). *Earth Observations and the Role of UAVs*. Retrieved February 14,2017https://www.nasa.gov/centers/dryden/pdf/175939main_Earth_Obs_UAV_Vol_1_v1.1_Final.pdf

Themistocleous, K., Agapiou, A., King, H. M., King, N., & Hadjimitsis, D. G. (2014). *More Than a Flight: The Extensive Contributions of UAV Flights to Archaeological Research-The Case Study of Curium Site in Cyprus.* Paper presented at the EuroMed.

Tule, C. A. (2014). *Trajectory Generation And Constrained Control Of Quadrotors.* (Master MA Thesis), University of Texas. (USA)

Twain, M. (2016). 2 From battlefield to backyard. *Drones in Society: Exploring the strange new world of unmanned aircraft.*

Valavanis, K. P., & Vachtsevanos, G. J. (2014). *Handbook of unmanned aerial vehicles*: Springer Publishing Company, Incorporated.

Van Blyenburgh, P. (1999). UAVs: an overview. *Air & Space Europe, 1*(5-6), 43-47.

Waharte, S., & Trigoni, N. (2010). *Supporting search and rescue operations with UAVs.* Paper presented at the Emerging Security Technologies (EST), 2010 International Conference on Sept 6.

Wegener, S., Schoenung, S., Totah, J., Sullivan, D., Frank, J., Enomoto, F., Theodore, C. (2004). *UAV autonomous operations for airborne science missions.* Paper presented at the AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit.

Weibel, R. E., & Hansman, R. J. (2004). *Safety considerations for operation of unmanned aerial vehicles in the national airspace system*. Paper presented at the 3rd "Unmanned Unlimited" Technical Conference, Chicago, Illinois.

Wie, B. (2008). *Space vehicle dynamics and control*: Aiaa.

Wolfe, R. C. (2003). NASA ERAST non-cooperative DSA flight test. *Proceedings of AUVSI Unmanned Systems 2003*.

Xu, R., & Ozguner, U. (2006). *Sliding mode control of a quadrotor helicopter.* Paper presented at the Decision and Control, 2006 45th IEEE Conference on.

Yanushevsky, R. (2007). *Modern missile guidance*: CRC Press.

Yarrish, G. (2015). *British Soldiers in Afghanistan deploy Micro UAV Helis*. Retrieved February 14, 2017 http://www.modelairplanenews.com/british-soldiers-in-afghanistan-deploy-micro-uav-helis/

Yoshimoto, H., & Hori, K. Blimps as Performance Media.

Zhang, C., & Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: a review. *Precision agriculture, 13*(6), 693-712.

Zulu, A., & John, S. (2016). A review of control algorithms for autonomous quadrotors. *arXiv preprint arXiv:1602.02622*.

APPENDIX A

Table 1: Quadrotor Parameters and initial conditions for simulation.

Symbol	Description	Value	Unit
I	Moment of inertia	$\begin{pmatrix} 7.5e-3 & 0 & 0 \\ 0 & 7.5e-3 & 0 \\ 0 & 0 & 1.3e-2 \end{pmatrix}$	kg.m^2
l	Arm length	0.23	M
I_r	Inertia of motor	6e-5	kg.m^2
k_f	Thrust coefficient	3.13e-5	Ns^2
k_M	Moment coefficient	7.5e-7	Nms^2
m	Mass of quadrotor	0.65	Kg
g	Gravity	9.81	ms^2
k_t	Aerodynamic thrust drag coefficient	$\begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.15 \end{pmatrix}$	Ns/m
k_r	Aerodynamic moment drag coefficient	$\begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.15 \end{pmatrix}$	Nm.s

APPENDIX B

LINEARIZED MODEL

LINEARIZED MODEL OF EULER ANGLE ORIENTATION

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{k_{t_x}}{m} & 0 & 0 & -g\sin\psi_T & -g\cos\psi_T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{k_{t_y}}{m} & 0 & g\cos\psi_T & -g\sin\psi_T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{k_{t_z}}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{k_{r_x}}{I_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{k_{r_y}}{I_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{k_{r_z}}{I_z} \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{l}{I_x} & 0 & 0 \\ 0 & 0 & \frac{l}{I_x} & 0 \\ 0 & 0 & 0 & \frac{l}{I_x} \end{pmatrix}$$

LINEARIZED MODEL OF QUATERNION ORIENTATION

$$A = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{k_{t_x}}{m} & 0 & 0 & 0 & -2gq_3 & -2gq_0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{k_{t_y}}{m} & 0 & 0 & 2gq_0 & -2gq_3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{k_{t_z}}{m} & -4gq_0 & 0 & 0 & -4gq_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{q_3}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{q_0}{2} & -\frac{q_3}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{q_3}{2} & \frac{q_0}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{q_0}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{k_{r_x}}{I_x} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{k_{r_y}}{I_y} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{k_{r_z}}{I_z}
\end{bmatrix}$$

$$B = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
-\frac{2q_0^2 + 2q_3^2 - 1}{m} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & \frac{l}{I_x} & 0 & 0 \\
0 & 0 & \frac{l}{I_x} & 0 \\
0 & 0 & 0 & \frac{l}{I_x}
\end{pmatrix}$$

APPENDIX C

MATLAB CODE

NONLINEAR PLANT MODEL IN EULER ANGLE ORIENTATION

```
function [sys,x0,str,ts,simStateCompliance] = sfun_nonlinear_plant(t,s,u,flag, params)
switch flag,

 %%%%%%%%%%%%%%%%%%
 % Initialization %
 %%%%%%%%%%%%%%%%%%
 case 0,
  [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(params);

 %%%%%%%%%%%%%%%
 % Derivatives %
 %%%%%%%%%%%%%%%
 case 1,
  sys=mdlDerivatives(t,s,u, params);

 %%%%%%%%%%
 % Update %
 %%%%%%%%%%
 case 2,
  sys=mdlUpdate(t,s,u);

 %%%%%%%%%%%
 % Outputs %
 %%%%%%%%%%%
 case 3,
  sys=mdlOutputs(t,s,u);

 %%%%%%%%%%%%%%%%%%%%%%%
 % GetTimeOfNextVarHit %
 %%%%%%%%%%%%%%%%%%%%%%%
 case 4,
  sys=mdlGetTimeOfNextVarHit(t,s,u);

 %%%%%%%%%%%%%
 % Terminate %
 %%%%%%%%%%%%%
 case 9,
  sys=mdlTerminate(t,s,u);
  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end

function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(params)

sizes = simsizes;

sizes.NumContStates  = 12;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 12;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

x0  = params.x0;

str = [];

ts  = [0 0];

simStateCompliance = 'UnknownSimState';

function sys=mdlDerivatives(t,s,u, params)

m     = params.m;
g     = params.g;
Jx     = params.Jx;
Jy     = params.Jy;
Jz     = params.Jz;
Jr    = params.Jr;
l     = params.l;
ktx    = params.ktx;
kty    = params.kty;
ktz    = params.ktz;
krx    = params.krx;
kry    = params.kry;
krz    = params.krz;
kf    = params.kf;
km    = params.km;

xdot = s(4);
ydot = s(5);
```

```
zdot = s(6);

phi = s(7);
theta = s(8);
psi = s(9);

p = s(10);
q = s(11);
r = s(12);

u1 = u(1);
u2 = u(2);
u3 = u(3);
u4 = u(4);
wm_min = 0;

u1_max = kf*4*wm_max^2;
u2_max = kf*wm_max^2;
u3_max = kf*wm_max^2;
u4_max = km*2*wm_max^2;

if (u1 >= u1_max)
    u1 = u1_max;

else
    u1;

end

if (u2 >= u2_max)
    u2 = u2_max;

else
    u2;

end

if (u3 >= u3_max)
    u3 = u3_max;

else
    u3;

end

if (u4 >= u4_max)
    u4 = u4_max;

else
```

```
    u4;

end

wm1 = ((1/4*kf)*u1 + (1/2*kf)*u3 + (1/4*km)*u4);
wm2 = ((1/4*kf)*u1 - (1/2*kf)*u2 - (1/4*km)*u4);
wm3 = ((1/4*kf)*u1 - (1/2*kf)*u3 + (1/4*km)*u4);
wm4 = ((1/4*kf)*u1 + (1/2*kf)*u3 - (1/4*km)*u4);

wm_max = (90/100)*9000*(2*pi/60);
wm_min = 0;

if (wm1 <= wm_min)
    wm1 = wm_min;
else
    wm1;
end

if (wm2 <= wm_min)
    wm2 = wm_min;
else
    wm2;
end

if (wm3 <= wm_min)
    wm3 = wm_min;
else
    wm3;
end

if (wm4 <= wm_min)
    wm4 = wm_min;

else
    wm4;

end

if (wm1 >= wm_max)
    wm1 = wm_max;

else
    wm1;

end

if (wm2 >= wm_max)
    wm2 = wm_max;
else
```

```
    wm2;
end
if (wm3 >= wm_max)
    wm3 = wm_max;
else
    wm3;
end
if (wm4 >= wm_max)
    wm4 = wm_max;
else
    wm4;
end
wmr     = -sqrt(wm1)+sqrt(wm2)-sqrt(wm3)+sqrt(wm4);
sdot = [xdot
        ydot
        zdot
        -(ktx*xdot + u1*(sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta)))/m
        -(kty*ydot - u1*(cos(psi)*sin(phi) - cos(phi)*sin(psi)*sin(theta)))/m
        -(ktz*zdot - g*m + u1*cos(phi)*cos(theta))/m
        p + r*cos(phi)*tan(theta) + q*sin(phi)*tan(theta)
        q*cos(phi) - r*sin(phi)
        (r*cos(phi))/cos(theta) + (q*sin(phi))/cos(theta)
        -(krx*p - l*u2 - Jy*q*r + Jz*q*r + Jr*q*wmr)/Jx
         (l*u3 - kry*q - Jx*p*r + Jz*p*r + Jr*p*wmr)/Jy;
         (u4 - krz*r + Jx*p*q - Jy*p*q)/Jz]

 sys = sdot;
function sys=mdlUpdate(t,s,u)
sys = [];
function sys=mdlOutputs(t,s,u)
sys = s;
function sys=mdlGetTimeOfNextVarHit(t,s,u)
sampleTime = 1;
sys = t + sampleTime;
function sys=mdlTerminate(t,s,u)
sys = [];
```

LINEAR PLANT MODEL IN EULER ANGLE ORIENTATION

```
function [sys,dx,str,ts,simStateCompliance] = sfun_uav_linear(t,s,du,flag, params)
switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,dx,str,ts,simStateCompliance]=mdlInitializeSizes(params);

  %%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%
  case 1,
    sys=mdlDerivatives(t,s,du, params);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys=mdlUpdate(t,s,du);

  %%%%%%%%%%%
  % Outputs %
  %%%%%%%%%%%
  case 3,
    sys=mdlOutputs(t,s,du);

  %%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
  %%%%%%%%%%%%%%%%%%%%%%%
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,s,du);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case 9,
    sys=mdlTerminate(t,s,du);

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end
```

```
sizes = simsizes;

sizes.NumContStates  = 12;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 12;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

dx  = params.dx;

str = [];

ts  = [0 0];

simStateCompliance = 'UnknownSimState';



function sys=mdlDerivatives(t,s,du, params)

m     = params.m;
g     = params.g;

Jx     = params.Jx;
Jy     = params.Jy;
Jz     = params.Jz;

Jr    = params.Jr;
l     = params.l;

ktx    = params.ktx;
kty    = params.kty;
ktz    = params.ktz;



krx    = params.krx;
kry    = params.kry;
krz    = params.krz;
km     = params.km;
kf     = params.kf;

phi = 0;
theta = 0;
psi = 10*pi/180;

p = 0;
```

```
q = 0;
r = 0;

u1 = du(1);
u2 = du(2);
u3 = du(3);
u4 = du(4);

wm_max = (90/100)*9000*(2*pi/60);
wm_min = 0;

u1_max = kf*4*wm_max^2;
u2_max = kf*wm_max^2;
u3_max = kf*wm_max^2;
u4_max = km*2*wm_max^2;

if (u1 >= u1_max)
    u1 = u1_max;

else
    u1;

end

if (u2 >= u2_max)
    u2 = u2_max;

else
    u2;

end

if (u3 >= u3_max)
    u3 = u3_max;

else
    u3;

end

if (u4 >= u4_max)
    u4 = u4_max;

else
    u4;

end

wm1 = ((1/4*kf)*u1 + (1/2*kf)*u3 + (1/4*km)*u4);
```

```
wm2 = ((1/4*kf)*u1 - (1/2*kf)*u2 - (1/4*km)*u4);
wm3 = ((1/4*kf)*u1 - (1/2*kf)*u3 + (1/4*km)*u4);
wm4 = ((1/4*kf)*u1 + (1/2*kf)*u3 - (1/4*km)*u4);

wm_max = (90/100)*9000*(2*pi/60);
wm_min = 0;

if (wm1 <= wm_min)
   wm1 = wm_min;

else
   wm1;

end

if (wm2 <= wm_min)
   wm2 = wm_min;

else
   wm2;

end

if (wm3 <= wm_min)
   wm3 = wm_min;
else
   wm3;
end

if (wm4 <= wm_min)
   wm4 = wm_min;
else
   wm4;

end

if (wm1 >= wm_max)
   wm1 = wm_max;
else
   wm1;

end
if (wm2 >= wm_max)
   wm2 = wm_max;

else
   wm2;
end
```

```
if (wm3 >= wm_max)
   wm3 = wm_max;
else
   wm3;
end

if (wm4 >= wm_max)
   wm4 = wm_max;
else
   wm4;
end

wmr    = -wm1+wm2-wm3+wm4;

if (wmr <= minVal)
   wmr = minVal;
else
   wmr;
end



A = [ 0, 0, 0,      1,      0,      0,                                         0,                                                      0,
0,                    0,                    0,               0
     0, 0, 0,      0,      1,      0,                                         0,                                                      0,
0,                    0,                    0,               0
     0, 0, 0,      0,      0,      1,                                         0,                                                      0,
0,                    0,                    0,               0
     0, 0, 0, -ktx/m,      0,      0, -(u1*(cos(phi)*sin(psi) - cos(psi)*sin(phi)*sin(theta)))/m,                                   -
(u1*cos(phi)*cos(psi)*cos(theta))/m, -(u1*(cos(psi)*sin(phi) - cos(phi)*sin(psi)*sin(theta)))/m,                    0,
0,               0
     0, 0, 0,      0, -kty/m,      0,  (u1*(cos(phi)*cos(psi) + sin(phi)*sin(psi)*sin(theta)))/m,                                   -
(u1*cos(phi)*cos(theta)*sin(psi))/m, -(u1*(sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta)))/m,                    0,
0,               0
     0, 0, 0,      0,      0, -ktz/m,                        (u1*cos(theta)*sin(phi))/m,
(u1*cos(phi)*sin(theta))/m,                                         0,                    0,                    0,               0
     0, 0, 0,      0,      0,      0,            q*cos(phi)*tan(theta) - r*sin(phi)*tan(theta),            r*cos(phi)*(tan(theta)^2 + 1) +
q*sin(phi)*(tan(theta)^2 + 1),                                         0,                    1,       sin(phi)*tan(theta), cos(phi)*tan(theta)
     0, 0, 0,      0,      0,      0,                          - r*cos(phi) - q*sin(phi),                                                      0,
0,                    0,                cos(phi),          -sin(phi)
     0, 0, 0,      0,      0,      0,         (q*cos(phi))/cos(theta) - (r*sin(phi))/cos(theta), (r*cos(phi)*sin(theta))/cos(theta)^2 +
(q*sin(phi)*sin(theta))/cos(theta)^2,                                         0,                    0,       sin(phi)/cos(theta),
cos(phi)/cos(theta)
     0, 0, 0,      0,      0,      0,                                         0,                                                      0,
0,               -krx/Jx, -(Jz*r - Jy*r + Jr*wmr)/Jx,    (Jy*q - Jz*q)/Jx
     0, 0, 0,      0,      0,      0,                                         0,                                                      0,
0, (Jz*r - Jx*r + Jr*wmr)/Jy,                 -kry/Jy,   -(Jx*p - Jz*p)/Jy
     0, 0, 0,      0,      0,      0,                                         0,                                                      0,
0,        (Jx*q - Jy*q)/Jz,          (Jx*p - Jy*p)/Jz,          -krz/Jz];
```

```
B = [                                             0,    0,    0,    0
                                                0,    0,    0,    0
                                                0,    0,    0,    0
    -(sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(theta))/m,    0,    0,    0
     (cos(psi)*sin(phi) - cos(phi)*sin(psi)*sin(theta))/m,    0,    0,    0
                              -(cos(phi)*cos(theta))/m,    0,    0,    0
                                                0,    0,    0,    0
                                                0,    0,    0,    0
                                                0,    0,    0,    0
                                                0, l/Jx,    0,    0
                                                0,    0, l/Jy,    0
                                                0,    0,    0, 1/Jz];

sdot   = A*s+B*du;
sys = sdot;
function sys=mdlUpdate(t,s,du)
sys = [];
function sys=mdlOutputs(t,s,du)
sys = s;
function sys=mdlGetTimeOfNextVarHit(t,s,du)

sampleTime = 1;    %  Example, set the next hit to be one second later.
sys = t + sampleTime;
function sys=mdlTerminate(t,s,du)
sys = [];
```

NONLINEAR PLANT MODEL IN QUATERNION ORIENTATION

```
function [sys,x0,str,ts,simStateCompliance] = sfun_uav_nonlinear(t,s,u,flag, params)
switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(params);

  %%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%
  case 1,
    sys=mdlDerivatives(t,s,u, params);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys=mdlUpdate(t,s,u);

  %%%%%%%%%%%
  % Outputs %
  %%%%%%%%%%%
  case 3,
    sys=mdlOutputs(t,s,u);

  %%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
  %%%%%%%%%%%%%%%%%%%%%%%
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,s,u);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case 9,
    sys=mdlTerminate(t,s,u);

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end
```

```
% end sfuntmpl


function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(params)

sizes = simsizes;

sizes.NumContStates  = 13;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 13;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

x0  = params.x0;
str = [];

ts  = [0 0];

simStateCompliance = 'UnknownSimState';


function sys=mdlDerivatives(t,s,u, params)

m     = params.m;
g     = params.g;

Jx     = params.Jx;
Jy     = params.Jy;
Jz     = params.Jz;

Jr    = params.Jr;
l     = params.l;

ktx    = params.ktx;
kty    = params.kty;
ktz    = params.ktz;


krx    = params.krx;
kry    = params.kry;
krz    = params.krz;
kf     = params.kf;
km     = params.km;
```

```
xdot = s(4);
ydot = s(5);
zdot = s(6);

q0 = s(7);
q1 = s(8);
q2 = s(9);
q3 = s(10);

p = s(11);
q = s(12);
r = s(13);



u1 = u(1);
u2 = u(2);
u3 = u(3);
u4 = u(4);

wm1 = sqrt((1/4*kf)*u1 + (1/2*kf)*u3 + (1/4*km)*u4);
wm2 = sqrt((1/4*kf)*u1 - (1/2*kf)*u2 - (1/4*km)*u4);
wm3 = sqrt((1/4*kf)*u1 - (1/2*kf)*u3 + (1/4*km)*u4);
wm4 = sqrt((1/4*kf)*u1 + (1/2*kf)*u3 - (1/4*km)*u4);



minVal = 0;

if (wm1 <= minVal)
   wm1 = minVal;
else
   wm1;
end

if (wm2 <= minVal)
   wm2 = minVal;
else
   wm2;
end

if (wm3 <= minVal)
   wm3 = minVal;
else
   wm3;
end

if (wm4 <= minVal)
   wm4 = minVal;
else
   wm4;
```

```
end

wmr     = -wm1+wm2-wm3+wm4;

if (wmr <= minVal)
    wmr = minVal;
else
    wmr;
end



sdot = [                                                        xdot
                                                              ydot
                                                              zdot
                                    -(ktx*xdot + u1*(2*q0*q2 + 2*q1*q3))/m
                                    -(kty*ydot - u1*(2*q0*q1 - 2*q2*q3))/m
                                 -(ktz*zdot - g*m + u1*(2*q0^2 + 2*q3^2 - 1))/m
                                         - (p*q1)/2 - (q*q2)/2 - (q3*r)/2
                                          (p*q0)/2 - (q*q3)/2 + (q2*r)/2
                                          (p*q3)/2 + (q*q0)/2 - (q1*r)/2
                                          (q*q1)/2 - (p*q2)/2 + (q0*r)/2
                                 -(krx*p - l*u2 - Jy*q*r + Jz*q*r + Jr*q*wmr)/Jx
                                  (l*u3 - kry*q - Jx*p*r + Jz*p*r + Jr*p*wmr)/Jy
                                        (u4 - krz*r + Jx*p*q - Jy*p*q)/Jz];

sys = sdot;
function sys=mdlUpdate(t,s,u)
sys = [];
function sys=mdlOutputs(t,s,u)
sys = s;
function sys=mdlGetTimeOfNextVarHit(t,s,u)
sampleTime = 1;    %  Example, set the next hit to be one second later.
sys = t + sampleTime;
function sys=mdlTerminate(t,s,u)
sys = [];
```

LINEAR PLANT MODEL IN QUATERNION ORIENTATION

```
function [sys,x0,str,ts,simStateCompliance] = sfun_uav_nonlinear(t,s,u,flag, params)
switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(params);

  %%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%
  case 1,
    sys=mdlDerivatives(t,s,u, params);

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys=mdlUpdate(t,s,u);

  %%%%%%%%%%%
  % Outputs %
  %%%%%%%%%%%
  case 3,
    sys=mdlOutputs(t,s,u);

  %%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
  %%%%%%%%%%%%%%%%%%%%%%%
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,s,u);

  %%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%
  case 9,
    sys=mdlTerminate(t,s,u);

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end
```

```
function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(params)

sizes = simsizes;

sizes.NumContStates  = 13;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 13;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

x0  = params.x0;

str = [];

ts  = [0 0];

function sys=mdlDerivatives(t,s,u, params)
m     = params.m;
g     = params.g;

Jx     = params.Jx;
Jy     = params.Jy;
Jz     = params.Jz;

Jr    = params.Jr;
l     = params.l;

ktx    = params.ktx;
kty    = params.kty;
ktz    = params.ktz;



krx    = params.krx;
kry    = params.kry;
krz    = params.krz;
kf     = params.kf;
km     = params.km;

phi = 0;
theta = 10*pi/180;
psi = 10*pi/180;

q0 = cos(phi/2)*cos(theta/2)*cos(psi/2) + sin(phi/2)*sin(theta/2)*sin(psi/2);
q1 = sin(phi/2)*cos(theta/2)*cos(psi/2) - cos(phi/2)*sin(theta/2)*sin(psi/2);
```

```
q2 = cos(phi/2)*sin(theta/2)*cos(psi/2) + sin(phi/2)*cos(theta/2)*sin(psi/2);
q3 = cos(phi/2)*cos(theta/2)*sin(psi/2) - sin(phi/2)*sin(theta/2)*cos(psi/2);

p = 0;
q = 0;
r = 0;

u1 = du(1);
u2 = du(2);
u3 = du(3);
u4 = du(4);

wm1 = sqrt((1/4*kf)*u1 + (1/2*kf)*u3 + (1/4*km)*u4);
wm2 = sqrt((1/4*kf)*u1 - (1/2*kf)*u2 - (1/4*km)*u4);
wm3 = sqrt((1/4*kf)*u1 - (1/2*kf)*u3 + (1/4*km)*u4);
wm4 = sqrt((1/4*kf)*u1 + (1/2*kf)*u3 - (1/4*km)*u4);

minVal = 0;

if (wm1 <= minVal)
    wm1 = minVal;
else
    wm1;
end

if (wm2 <= minVal)
    wm2 = minVal;
else
    wm2;
end

if (wm3 <= minVal)
    wm3 = minVal;
else
    wm3;
end

if (wm4 <= minVal)
    wm4 = minVal;
else
    wm4;
end

wm1 = sqrt((1/4*kf)*u1 + (1/2*kf)*u3 + (1/4*km)*u4);
wm2 = sqrt((1/4*kf)*u1 - (1/2*kf)*u2 - (1/4*km)*u4);
wm3 = sqrt((1/4*kf)*u1 - (1/2*kf)*u3 + (1/4*km)*u4);
wm4 = sqrt((1/4*kf)*u1 + (1/2*kf)*u3 - (1/4*km)*u4);
wmr     = -wm1+wm2-wm3+wm4;
if (wmr <= minVal)
```

```
    wmr = minVal;
else
    wmr;
end

A =   [ 0, 0, 0,      1,      0,      0,           0,           0,           0,           0,                     0,                      0,              0
        0, 0, 0,      0,      1,      0,           0,           0,           0,           0,                     0,                      0,              0
        0, 0, 0,      0,      0,      1,           0,           0,           0,           0,                     0,                      0,              0
        0, 0, 0, -ktx/m,      0,      0, -(2*q2*u1)/m, -(2*q3*u1)/m, -(2*q0*u1)/m, -(2*q1*u1)/m,                     0,                      0,
0
        0, 0, 0,      0, -kty/m,      0,  (2*q1*u1)/m,  (2*q0*u1)/m, -(2*q3*u1)/m, -(2*q2*u1)/m,                     0,                      0,
0
        0, 0, 0,      0,      0, -ktz/m, -(4*q0*u1)/m,           0,           0, -(4*q3*u1)/m,                     0,                      0,              0
        0, 0, 0,      0,      0,      0,           0,        -p/2,        -q/2,        -r/2,                 -q1/2,                  -q2/2,          -q3/2
        0, 0, 0,      0,      0,      0,         p/2,           0,         r/2,        -q/2,                  q0/2,                  -q3/2,           q2/2
        0, 0, 0,      0,      0,      0,         q/2,        -r/2,           0,         p/2,                  q3/2,                   q0/2,          -q1/2
        0, 0, 0,      0,      0,      0,         r/2,         q/2,        -p/2,           0,                 -q2/2,                   q1/2,           q0/2
        0, 0, 0,      0,      0,      0,           0,           0,           0,           0,               -krx/Jx, -(Jz*r - Jy*r + Jr*wmr)/Jx,  (Jy*q -
Jz*q)/Jx
        0, 0, 0,      0,      0,      0,           0,           0,           0,           0, (Jz*r - Jx*r + Jr*wmr)/Jy,                -kry/Jy, -(Jx*p -
Jz*p)/Jy
        0, 0, 0,      0,      0,      0,           0,           0,           0,           0,        (Jx*q - Jy*q)/Jz,          (Jx*p - Jy*p)/Jz,          -krz/Jz];

B = [                        0,    0,    0,    0
                             0,    0,    0,    0
                             0,    0,    0,    0
       -(2*q0*q2 + 2*q1*q3)/m,    0,    0,    0
        (2*q0*q1 - 2*q2*q3)/m,    0,    0,    0
     -(2*q0^2 + 2*q3^2 - 1)/m,    0,    0,    0
                             0,    0,    0,    0
                             0,    0,    0,    0
                             0,    0,    0,    0
                             0,    0,    0,    0
                             0, l/Jx,    0,    0
                             0,    0, l/Jy,    0
                             0,    0,    0, 1/Jz];
sdot   = A*s+B*du;

sys = sdot;
function sys=mdlUpdate(t,s,u)
sys = [];

function sys=mdlOutputs(t,s,u)
sys = s;
function sys=mdlGetTimeOfNextVarHit(t,s,u)
sampleTime = 1;    %  Example, set the next hit to be one second later.
sys = t + sampleTime;
function sys=mdlTerminate(t,s,u)
sys = [];
```

GENERATED MPC CODE FOR EULER ANGLE

create MPC controller object with sample time

```
mpc1 = mpc(mpc1_plant_C, 0.25);
```

specify prediction horizon

```
mpc1.PredictionHorizon = 20;
```

specify control horizon

```
mpc1.ControlHorizon = 2;
```

specify nominal values for inputs and outputs

```
mpc1.Model.Nominal.U = [6.37650038006904;0;0;0;2;2;2;2];
mpc1.Model.Nominal.Y = [0;0;-1;0;0;0;0;-1.41586750769674e-15;0;0;0;0];
```

specify weights

```
mpc1.Weights.MV = [0 0 0 0];
mpc1.Weights.MVRate = [0.1 0.1 0.1 0.1];
mpc1.Weights.OV = [1 1 1 1 0 0 0 0 0 0 0 0];
mpc1.Weights.ECR = 100000;
```

specify simulation options

```
options = mpcsimopt();
options.MVSignal = mpc1_MVSignal;
options.RefLookAhead = 'off';
options.MDLookAhead = 'off';
options.Constraints = 'on';
options.OpenLoop = 'off';
```

MPC COST FUNCTION FOR EULER ANGLE

```
function [f,dfdy,dfdu,dfddu,dfdslack] = mpcCustomCostFcn(y,yref,u,uref,du,v,slack,varargin)

% Dimension
p = size(y,1);
nmv = size(u,2);
ny = size(y,2);
```

specify weights

```
beta = 1;
Wu = diag([0 0 0 0.5]*beta);
Wdu = diag([0.1 0.1 0.1 0.1]/beta);
Wy = diag([5 5 5 5 0 0 0 0 0 0 0 0]*beta);
Wecr = 100000;

% Cost Function
f = sum(sum(((y-yref)*Wy).^2))+sum(sum((du*Wdu).^2))+sum(sum((u*Wu).^2))+Wecr*slack^2;

% Gradients
dfdy = (y-yref)*(Wy.^2);
dfdu = zeros(p,nmv);
dfddu = du*(Wdu.^2);
dfdslack = Wecr*slack;
```

GENERATED MPC CODE FOR QUATERNION

create MPC controller object with sample time

```
mpc1 = mpc(mpc1_plant_C_1, 0.25);
```

specify prediction horizon

```
mpc1.PredictionHorizon = 20;
```

specify control horizon

```
mpc1.ControlHorizon = 2;
```

specify nominal values for inputs and outputs

```
mpc1.Model.Nominal.U = [6.37650038006904;0;0;0;1;1;1;1];
mpc1.Model.Nominal.Y = [0;0;0;0;0;0;1;0;0;0;0;0;0];
```

specify weights

```
mpc1.Weights.MV = [0 0 0 0];
mpc1.Weights.MVRate = [0.1 0.1 0.1 0.1];
mpc1.Weights.OV = [1 1 1 1 0 0 0 0 0 0 0 0 0];
mpc1.Weights.ECR = 100000;
```

specify simulation options

```
options = mpcsimopt();
options.MVSignal = mpc1_MVSignal_1;
options.RefLookAhead = 'off';
options.MDLookAhead = 'off';
options.Constraints = 'on';
options.OpenLoop = 'off';
```

MPC COST FUNCTION FOR QUATERNION

```
function [f,dfdy,dfdu,dfddu,dfdslack] = mpcCustomCostFcn(y,yref,u,uref,du,v,slack,varargin)

% Dimension
p = size(y,1);
nmv = size(u,2);
ny = size(y,2);

% Desired Quaternion
q1_d = yref(:,8);
q2_d = yref(:,9);
q3_d = yref(:,10);
q0_d = sqrt((1 - (q1_d.^2+q2_d.^2+q3_d.^2)).^2);

% Actual Quaternion
q1_a = y(:,8);
q2_a = y(:,9);
q3_a = y(:,10);
q0_a = sqrt((1 - (q1_a.^2+q2_a.^2+q3_a.^2)).^2);

% Looping according to Prediction Horizon
for i=1:p
quat_err(:,i) = [ q0_d(i)     q1_d(i)    q2_d(i)    q3_d(i);
                 -q1_d(i)     q0_d(i)    q3_d(i)   -q2_d(i);
                 -q2_d(i)    -q3_d(i)    q0_d(i)    q1_d(i);
                 -q3_d(i)     q2_d(i)   -q1_d(i)    q0_d(i)]*[q0_a(i);  q1_a(i);   q2_a(i);   q3_a(i)];
end

quat = quat_err.';
```

specify weights

```
beta = 1; % 0.36788;
Wu = diag([0 0 0 0.5]*beta);
Wdu = diag([0.1 0.1 0.1 0.01]/beta);



Wy = diag([0.1 0.005 0.05 0 0 0 0 0 0 0.03 0 0 0]*beta);



Wecr = 100000;



yerror = [y(:, 1:6)-yref(:, 1:6) quat(:, 1:4) y(:, 11:13)-yref(:, 11:13)];

f = sum(sum(((yerror)*Wy).^2))+sum(sum((du*Wdu).^2))+sum(sum((u*Wu).^2))+Wecr*slack^2;
% Gradients
```

```
dfdy = (y-yref)*(Wy.^2);
dfdu = zeros(p,nmv);
dfddu = du*(Wdu.^2);
dfdslack = Wecr*slack;
```