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

MPC Based Attitude Control of Quadcopter

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

By
Banchiywab Aschalew

Advisor
Dr. Lebsewerk Negash

September 24, 2021
Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

MPC Based Attitude Control of Quadcopter

By: Banchiywab Aschalew

APPROVED BY BOARD OF EXAMINERS

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

Declaration

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

Name	**Signature**
_____	_____

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

Submitted in: September 24, 2021

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

Advisor	**Signature**
_____	_____

Acknowledgment

Glory be to the Father and to the Son and to the Holy Spirit. My Heavenly Father, I want to thank you so much for all that you have done for me. Especially for having given us Virgin Mary most holy as our Mother, Teacher and Queen. I know that all that I have comes from Him and pray that all that I do is for His glory.

I also want to thank my advisor, Dr. Lebsework Negash. Thank you for assisting me in broadening my views and taught me to always take a step back and investigate the problem thoroughly.

Finally, I would like to express my gratitude to my family and friends for their unwavering support, prayers, and encouragement.

Abstract

Quadcopter is a type of UAV having two pairs of counter rotating rotors. Its movement is controlled by adjusting the relative speed or relative trust and torque of each rotor which are spun by electric motors. However, quadcopter is nonlinear, underactuated MIMO system and also its inputs and outputs are constrained, which provide a suitable platform for control algorithm development and investigation of both stabilization and trajectory tracking control.

Because quadcopter is severely under-actuated and coupled system, a cascade control approach is proposed and applied for trajectory tracking control. Three different MPC methods, namely Linear MPC, Feedback linearization based MPC and Nonlinear MPC are designed for attitude control. Moreover, the control of x, y, and z position is also addressed, by utilizing a conventional PID controller, to test the performance of the attitude controllers during trajectory tracking control.

The nonlinear mathematical model of a quadcopter's dynamics, solved from Newton's and Euler's laws, is investigated by realizing the designed controllers using MATLAB/Simulink. The introduced attitude controllers are compared based on three performance evaluation factors, tracking accuracy, control effort efficiency and output disturbance rejection capacity. The performance of the proposed control schemes are verified by comparative simulation results and Root Mean Square Error (RMSE) tracking accuracy performance measure. The simulation results shows Linear MPC offer poor performance even if it is simple to design and offer fast response compared with the other. Feedback linearization based MPC ensure the utilization of linear MPC for nonlinear plant, in this case for nonlinear quadcopter model. In addition, Feedback linearization based MPC and Linear MPC strategies couldn't withstand output disturbances. Moreover, Feedback linearization based MPC strategy is incapable to deal with input constraints. Conversely, nonlinear MPC offers a good performance with constraints and output uncertainty. Even if it is computationally intensive.

Contents

List of Figures

List of Tables

List of Abbreviations

3D	three dimensional
DC	direct current
FBL	feedback linearization
IMU	inertial measurement unit
LMPC	linear model predictive control
LQR	linear quadratic regulator
MIMO	multi input multi output
MPC	model predictive control
MRAC	model reference adaptive control
NMPC	nonlinear model predictive control
PI	proportional integral
PID	proportional integral derivative
PD	proportional derivative
rpm	revolution per minute
UAV	unmanned aerial vehicle
VTOL	vertical take off and landing

Chapter 1

Introduction

1.1 Background

UAVs, unmanned aerial vehicles, are self-powered flying objects with no human pilot. In recent years they are so familiar because of their application in many sectors including military, security, agriculture, photography and videography with considerable significance. Generally, these vehicles are categorized into two major categories, Remotely Piloted UAVs and Autonomous UAVs. The common thing in all categories is that there is no person specifically involved or present in the flight environment. In the first one, human beings monitor and direct the UAV by some form of wireless communication. In the second category the vehicle is rendered enough intelligent to fly and perform the designated mission by itself. However, both cases demonstrate the capability of UAVs for use in applications where the environment is not friendly for human life. The other fundamental distinction in the classification of UAVs is based on their body type. According to the body type there are four physical types of UAVs, such as multi rotor UAVs, fixed wing UAVs, helicopters and VTOL hybrid fixed wing UAVs. The most common type among both specialists and general users is multi rotor UAVs.

Quadcopter is a multi-rotor UAV with four rotors. It is so common because of its many applications in various fields and its simple mechanical design. It is arousing the attention of several researchers in a range of disciplines resulted from the need for aircraft with greater maneuverability, hovering ability, and to make autonomous flight possible.

Quadcopter system has several interesting features like, instability, nonlinearity, underactuation and strong coupling, for developing control laws. But, the best fitted control system for quadcopter is still under development because every controller has its own some advantage and disadvantage. Balancing between this advantage and disadvantage widen the

opportunity to work on different controllers. Necessarily based on the gravity of applications, a suitable controller must be designed for quadcopter. Primarily, some features like fast response from the system, design simplicity, working with constraints and disturbance rejection during tracking are mostly expected for the controller of the quadcopters.

Quadcopter has four cross configured propellers. The two pairs of propellers rotate in opposite directions. The movement of the vehicle in space results from changes in the speed of the rotors. Although, the vehicle has four independent rotors, it has six degree of freedom. Hence, the objective would be to command the vehicle to follow a given trajectory or to hover. From simple intuition, it is impossible to fly even in the absence of disturbances like wind gust, meaning the problem is open loop unstable.

One can adjust the lift force and produce motion by changing the rotor rpm. Therefore, raising or lowering the speeds of the four propellers together creates vertical movement. Conversely, changing the speed of the two propellers creates roll or pitch rotation combined with lateral motion. The rotation of Yaw comes from the difference between each pair of propellers in the counter torque. Therefore, developing control laws is cardinal. In order to fly the quadcopter safely the developed control law must be robust to overcome the disturbance that affect or influence the performance of the quadcopter.

In this thesis, three different MPC methods, linear MPC, nonlinear MPC and Feedback linearization based MPC, have been investigated on quadcopter as attitude controller and conventional PID controller as position controller. This work give emphasis for attitude of quadcopter due to the fact that if the angle subsystem handled well, the position of the quadcopter is simpler to manage [5]. The introduced controllers are compared based on three performance evaluation factors, control effort efficiency, tracking accuracy and output disturbance rejection capacity. Linear MPC and Feedback linearization based MPC ensures fast response. While the linear MPC cannot deal with disturbances and Feedback linearization based MPC cannot offer input constraints and disturbance rejection. Conversely, nonlinear MPC is capable to deal with disturbances and offer constraints in both inputs and outputs. Nonlinear MPC responds slower than other do because it utilizes the actual nonlinear model for predicting the future require high computation. However, it can be overcome nowadays using high computational processors.

MATLAB/Simulink environment has been used to investigate the performance of the controllers considering three trajectories, circular, helical and infinity shape with different envi-

ronments.

1.2 Problem Description

The quadcopter has four rotors that are controlled independently. The movement of the quadcopter results from changes in the speed of these four rotors. The main challenges in controlling a quadcopter are that the quadcopter has six degrees of freedom but there are only four control inputs also the dynamic model of a quadcopter exhibits nonlinearity, underactuation and strong coupling. Moreover, practical problems need optimal solution which consider control and state constraints.

The mathematical model of the quadcopter system as obtained from Newton's and Euler's Equations of Motion yields a twelve state variable system. This mathematical model which has coupling and nonlinearity terms is further simplified by a set of reasonable assumptions.

For effective trajectory tracking control of the quadcopter system, a cascade control structure is utilized. Because quadcopter is severely under-actuated system, the six degrees of freedom in space are controlled only with four independent inputs, the control must be defined for a subset of four of its degree of freedom. In addition, the control of translational movements depend on the orientation of the quadcopter. Due to this facts, a cascade control approach is proposed. Then, the control of this system is defined for two distinct controller which are called inner loop controller and outer loop controller. The outer loop X-Y controller is designed to generate the reference pitch and roll angle based on the reference trajectory X and Y position coordinate respectively and also in the outer loop there is an altitude (Z position) controller which is responsible to offer the trust force. The generated reference roll and pitch angles with the command yaw angle are used as an input for the inner loop controller. The inner control loop design is based on different MPC methods. The complicated dynamics that must be handled are those of the inner loop. In outer loops, nonlinearities are simpler to manage if the inner loop is well behaved. Therefore, Classical PID controller is used for the outer control loop.

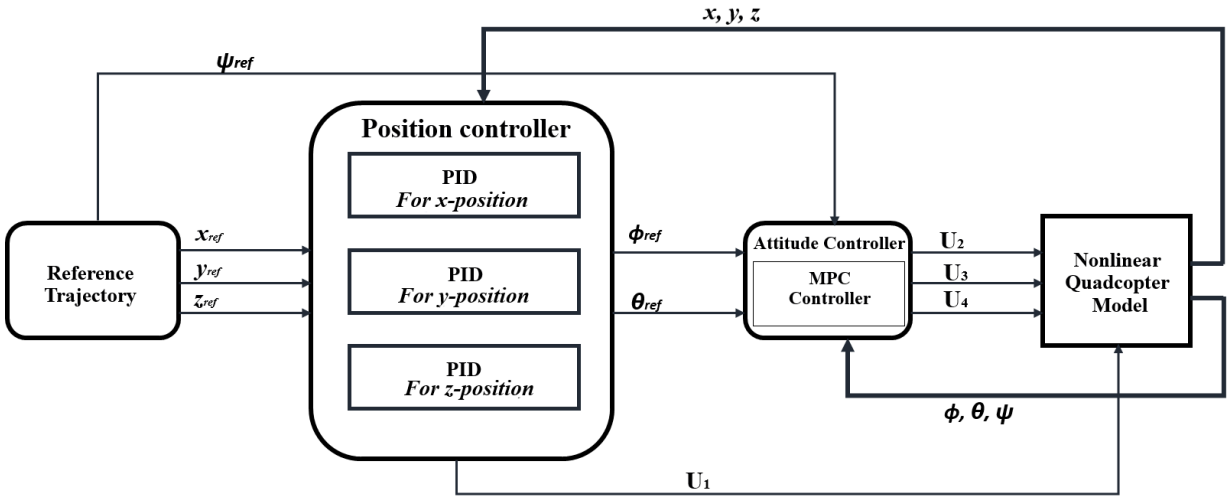


Figure 1.1: Overall Control Architecture

1.3 Objectives of The Thesis

1.3.1 General Objective

The main objective of this thesis is, to design robust controller that may assist the quadcopter to track the trajectory comparatively more accurately under different condition with smooth movement.

1.3.2 Specific Objectives

- To study the mathematical model of quadcopter system for control design.
- To select the appropriate parameters for the control of the quadcopter system.
- To design inner loop controller using different MPC approaches.
- To design outer loop controller using conventional PID technique.
- To choose and finalize the most suitable controller for attitude of quadcopter among the three controllers based on the investigation .

1.4 Scope of The Thesis

In this thesis the dynamics of quadcopter is studied and a mathematical model using the Newton-Euler formulation is developed first and different Model Predictive Control approaches with conventional PID controller are designed and compared in order to find the best tracking controller. The outcome of this research is based on the simulation of the proposed techniques using a MATLAB/ Simulink.

1.5 Thesis Outline

This thesis is organized into 7 chapters including appendix. The rest part of this thesis discuss, review of related and relevant previous research works in chapter 2. Modeling for kinematics and dynamics of the quadcopter in Chapter 3. Variant of MPC controllers design for attitude subsystem and PID controller design for position control in chapter 4. Performance test of the controllers using simulation in chapter 5. Conclusion and recommendation will be presented in chapter 6. Finally codes and models used in simulation are stated on appendix.

Chapter 2

Literature Review

Because autonomous control of quadcopter is a difficult and interesting challenge, there are many papers in the literature on the subject. The majority of the research focuses on quadcopter attitude control because it is the foundation for comprehensive control. Trajectory/path tracking control of the quadcopter, on the other hand, necessitates simultaneous control of both attitude and position, making it more difficult. And it is also the subject of several studies. The literature review will be centered on the content of this thesis, which is quadcopter attitude control. However, there will be a brief discussion of studies related to trajectory tracking.

Various control approaches have been developed in the literature to achieve quadcopter attitude control. Linear control techniques, nonlinear control techniques, and intelligent/adaptive control techniques are the three categories in which these approaches can be classified.

2.1 Linear Control Techniques

PID and LQR control are the most used linear control approaches. It is known that linear control techniques are designed for linearized system model. In [6], [7] and [8] PID control is proposed. Because the quadcopter's attitude subsystem is a highly nonlinear, coupled, and multi-input multi-output system, traditional PID control techniques failed to provide satisfactory results. However, in most cases of flight control, the conventional classical control approach PID continues to play a significant role. In quadcopter flight control, a double loop PID have been used to control the attitude and position of the quadcopter, resulting in quadcopter flight stability over a small range [9]. The quadcopter, on the other hand, is a complex device with a nonlinear multi-input multi-output and strong coupling. Decoupling using the classical PID control method is difficult. Moreover, this approach couldn't handle

the implementation of restrictions and uncertainty.

LQR, which is employed in [10], [11], [12], and [13], is the other most popular linear control strategy used by researchers. LQR is a method of optimal control. A quadratic cost function (performance index) is considered to be reduced in order to provide optimal control efforts. A linear dynamic model of the system is required to solve the LQR method. As a result, the system's nonlinear dynamic model must be linearized around a trim condition, such as hovering. Some design requirements may benefit from LQR control because these requirements could be described and specified in a cost function. Furthermore, this approach is capable of dealing with MIMO systems and provides excellent results in terms of closed loop control optimization. But the assumptions do not match the real conditions because of the quadcopter dynamical model has severe non-linearities. Besides, the controller is unable to deal with uncertainty [14].

2.2 Nonlinear Control Techniques

Because linear control methods have lost their dynamic performance, nonlinear control methods have received a lot of interest in the research community. For nonlinear systems, sliding-mode control is one of the most used approaches. However, in real-world applications, the high frequency of switching mode control signals could trigger the unmodeled dynamics in the real system, resulting in undesirable oscillations known as chattering. To prevent this, system modeling should take into account sensor and actuator dynamics, which are considerably faster than system dynamics [15]. Even though chattering appears in the control signal, stabilization at the hovering condition is maintained using this approach where the roll, pitch, and yaw angles converged to the desired value [16]. In contrast, a difficult initial yaw angle causes the system to have a large negative overshoot in the pitch angle. Backstepping control, which is developed for nonlinear dynamics systems, is another viable alternative. Despite the fact that the rigorous initial conditions have been set, it is capable of bringing the roll, pitch, and yaw angles to the equilibrium point that is zero. However, in that instance, there is some angle drift on the yaw angle control signal [16].

The most prominent strategy among nonlinear controllers is feedback linearization control, which offers two ways. Exact linearization and non-interacting control via dynamic feedback and Dynamic inversion with zero-dynamics stabilization which are described in [17]. The nonlinear system cannot be addressed using static feedback control, hence the former approach requires the employment of a dynamic feedback control law. The thrust input is

delayed until its second derivative, and the system is extended to include the thrust input and its first derivative as the system states, with the position variables chosen as the output function. The extended system satisfies the feedback linearization criterion and can be turned into a linear and controllable system via dynamic feedback. The latter method employs attitude variables as output variables and performs dynamic inversion with small angle approximation. The attitude variables are differentiated until the input terms appear, and the system is not really extended. The linearization is conducted using small angle approximation. The study also shows how the Linear Quadratic Regulator (LQR) is implemented on the quadcopter, with the linearized model and the LQR function in Matlab used to solve the algebraic equation using Riccati's approach and produce the control gains.

In [17] and [18], the attitude variables are viewed as outputs of interest, and feedback linearization by dynamic inversion is described. For structured tracking, a two-layer design is used, with a dynamic inversion inner loop and an internal dynamics stabilization outer loop. Small angle approximations for the Euler angles are used in dynamic inversion, resulting in a reduced expression for the inverted matrix. In [16], the linear controller for the linearized dynamics and residual dynamics is designed using a back-stepping technique. A full stability analysis for the two controllers is also presented in the paper. However, in [18] a PID is employed to perform trajectory following and internal dynamics stabilization, while a standard PD controller is used to provide linear control inputs in the inner loop.

[5] addresses a comparison of PID, sliding mode, backstepping, and feedback linearization controllers for attitude stabilization and control. The controllers are applied to the system and studied separately to determine the best controller for the quadcopter. The sliding mode technique outperforms the others. However, the PID controller performs reasonably well when compared to the sliding mode technique based on ease of implementation because its implementation is considerably easier. Because of this most of the literature includes a PID controller in the model. Inverse control based on feedback linearization is likewise simple to implement, with a faster settling time than the other controllers despite a slower response. As shown in this research, feedback linearization and sliding mode controllers have better performance than PID and backsteppping controllers.

Different control methods can be used, but the most convenient control can only be achieved by achieving the best performance in real-world settings. As a result, disturbance effects and physical limits must be considered for a robust and trustworthy control. In the model predictive control optimization technique, these disturbance and constraint limitations can

be properly computed in each iteration step ahead [19]. The Model Predictive Controller (MPC) could include constraints in the optimization process directly, which is useful for real-world systems with physical restrictions. To achieve stability and autonomous navigation for quadcopters, Bemporad et al. [20] used a hierarchical MPC. A linear MPC is used to achieve stable movement, while a hybrid MPC technique is used to generate a collision-free trajectory that the linear MPC tracks. However, the performance of the controller has not been examined in the presence of disturbances. Alexis et al. [21] provide a Robust MPC technique for quadcopter stabilization and demonstrate the efficiency of the proposed controller under wind disturbances with slung load to add robustness. Another researches [22] demonstrate a hierarchical MPC for stabilization and collision avoidance and real application performance under the strong disturbance conditions have not verified yet.

In this thesis, three MPC algorithms (linear MPC, feedback linearization based MPC, and NMPC) are designed and compared in simulation for quadcopter attitude control. Furthermore, environmental disturbances are treated as output disturbances in order to compute a robust control law while taking into consideration the quadcopter's physical and mechanical limits.

2.3 Adaptive Control Techniques

All control techniques, whether linear or nonlinear, require complete knowledge of the system model and model parameters, yet inaccuracies in the identified values of the parameters can cause the controller's performance to deteriorate significantly. Furthermore, during flight, unmodeled changes in system characteristics (such as mass or inertia) can generate considerable stabilizing problems. Adaptive approaches can be used to avoid the necessity for an exact nonlinear model of quadcopter dynamics. These methods can detect and fix inaccuracies in model parameter values, update parameter estimates whenever they change, and compensate for external perturbation.

Model Reference Adaptive Control (MRAC) and other linear adaptive approaches have been proposed [23]. However, like with most linear algorithms, the quadcopter's possible trajectory is limited by the assumption of linearization. Huang et al. [19] proposed an adaptive backstepping method, while Zeng et al. [24] extended this methodology to include inertia parameters in the adaptation law. Indirect adaptive approaches, such as least-squares method (for mass), have been used in the work of Kumar et al. [25] in autonomous grasping and construction employing quadcopters. All indirect techniques, on the other hand, correct

parameter errors by comparing expected and actual plant outputs, but they do not explicitly correct model parameters (as done by direct adaptive methods). Craig et al. [26] were the first to propose direct adaption methods for mechanical manipulators.

2.4 Trajectory Tracking Control

Quadcopters' trajectory tracking is a well-studied problem. Nonlinear approaches such as input-output linearization using differential flatness theory and back-stepping control, which allows for acrobatic moves, have received a lot of attention in the literature. Such methods aren't required for typical trajectory tracking. In [27] trajectory control is achieved by generating a symmetric function for jounce(the forth time derivative of position vector) to control the acceleration and, in turn, determining the control inputs to produce the desired trajectory using a heuristic technique. The heuristic method includes a PD controller to improve respond to disturbances and stabilize the quadcopter during trajectory tracking. The dynamics in [28] are linearized around the intended trajectory. To reduce the error in the trajectory, an acceleration vector command is calculated. The roll and pitch commands are then calculated and supplied to the attitude controller based on the acceleration vector and desired yaw angle. A position controller is also discussed that does not use linearization and projects the position error along the yaw axis. [28] discusses a trajectory planning technique that simplifies the problem by assuming differential flat theory for quadcopters that respect the dynamics of the underactuated system. A cost function built from jounce and yaw accelerations is used to generate the trajectory. As a result, real-time planning is used to produce the best path.

[29] shows a different trajectory planning algorithm. A set of desired waypoints is entered, and a dynamically viable trajectory is constructed that crosses the waypoints in the shortest time possible while adhering to acceleration and velocity limits. The controller is made up of a piecewise PI control in the along direction and a PID control in the cross track direction, both of which supply the necessary control inputs for the desired path to be followed. The controllers indicated above demand derivatives of the desired path as input, which limits the quadcopter's aggressive maneuvers. This necessitates the development of a trajectory controller that is independent of higher order derivatives and capable of enabling the quadcopter's various motions.

Chapter 3

System Modeling

This chapter examines the quadcopter's kinematics and dynamics, which aids in studying and understanding the mechanics and behavior of the quadcopter. The aerodynamic impacts on the quadcopter body, as well as the rotor dynamics of the quadcopter's actuators, will be addressed after the quadcopter's kinematics and dynamics models have been determined. The chapter will end up with the formulation of a state space model for the quadcopter system, which will be examined in the following chapters.
The kinematics and dynamics models are obtained using a Newton-Euler formulation, by assuming the following:

- The structure is rigid and symmetrical
- The center of gravity of the quadcopter coincides with the body fixed frame origin.
- The propellers are rigid.
- Thrust and drag are proportional to the square of propeller's speed.

3.1 Kinematic Model

In order to go in to quadcopter modeling, first it is necessary to establish the coordinate frames that will be utilized. Figure 3.1 shows the Earth reference frame with X_E, Y_E and Z_E axes and the body frame with X_b, Y_b and Z_b axes. The Earth frame is an inertial frame fixed to the ground in a specified given location as its name implies, it uses the X_E - Y_E - Z_E notation where the axes are pointing in the direction of the North, East and downwards respectively. The body frame, on the other hand, is located at the center of the quadcopter's body., with its x-axis towards the direction of propeller 1, y-axis towards the direction of propeller 2 and the z-axis is towards the direction of the ground.

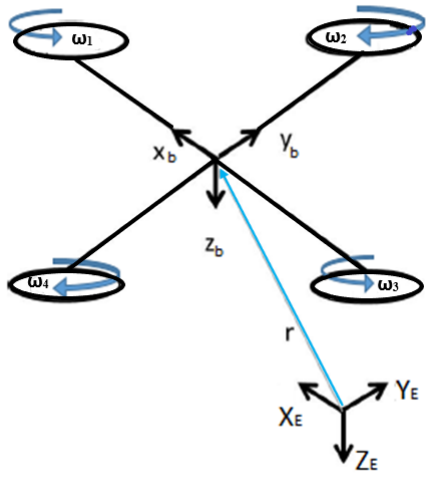


Figure 3.1: Quadcopter Reference Frame

The distance between the Earth frame and the body frame determines the quadcopter's absolute location, $r = [x\ y\ z]^T$. Roll, pitch and yaw angles (ϕ, θ and ψ), which represent rotation about the X, Y and Z-axes respectively, are used to define the quadcopter's orientation. The rotation matrix, R, maps the coordinates of the inertial into the body fixed frame frame, assuming the order of rotation is roll (ϕ), pitch (θ) then yaw (ψ)

$$R = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\theta c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{3.1}$$

This notations have been used in the previous equation and the following: $c_k = cos(k), s_k = sin(k), t_k = tan(k)$.

The derivation of the rotation matrix, R, is shown in details in Appendix. R will be used to formulate the quadcopter's dynamics model, because a transition from one frame to the other is required to establish a relationship between the two reference frame; the fact that some states are measured in the body frame (e.g. propeller thrust forces) while others are measured in the inertial frame (e.g. gravitational forces and quadcopter's position) gives it value.

To obtain information about the angular velocity of the quadrotor, an on-board Inertial Measurement Unit (IMU) is utilized commonly, which then gives the velocity in the body

coordinate frame. In order to link the Euler rates, $\dot{\eta} = \left[\dot{\phi}\,\dot{\theta}\,\dot{\psi}\right]^T$, that are measured in the inertial frame and angular body rates $\omega = [p\,q\,r]^T$, a transformation is needed as follows:

$$\omega = T\dot{\eta} \tag{3.2}$$

where,

$$T = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix}$$

The derivation for the transformation matrix T is shown in Appendix.

3.2 Dynamics Modeling

The quadcopter's motion is composed of two different motions: rotational and translational motion. The rotational subsystem is completely actuated, whereas the translational motion is underactuated and depends on the orientation of the quadcopter, which is on the rotational motion of the quadcopter[30]. The rotational motion and translational motion equations of the quadcopter system is dirived in the following subsections.

3.2.1 Rotational Equations of Motion

The Newton-Euler method is used to derive the rotational equations of motion in the body frame, with the following general formalism:

$$M_B = I\dot{\omega} + \omega \times I\omega + M_G \tag{3.3}$$

Where

M_B – *Moments acting on the quadcopter in the body frame*

I – *Quadcopter's inertia Matrix*

ω – *Angular body rates*

M_G – *Gyroscopic moments due to rotors' inertia*

The first two terms of Equation 3.3, $I\dot{\omega}$ and $\omega \times I\omega$ represent the angular acceleration of the inertia and the centripetal forces in the body frame. While M_G represent the gyroscopic moments due to the rotors' inertia I_r. The Gyroscopic moments are defined to be

$$M_G = \omega \times \begin{bmatrix} 0 & 0 & I_r\omega_r \end{bmatrix}^T$$

thus the rotational equation of the quadcopter's motion can be written as,

$$M_B = I\dot{\omega} + \omega \times I\omega + \omega \times \begin{bmatrix} 0 & 0 & I_r\omega_r \end{bmatrix}^T \tag{3.4}$$

Where

$$\begin{aligned} I_r &- \textit{Rotors' inertia Matrix} \\ \omega_r &- \textit{Rotors' relativespeed} \\ \omega_r &= -\omega_1 + \omega_2 - \omega_3 + \omega_4 \end{aligned}$$

The reasons behind deriving the rotational equations of motion in the body frame and not in the inertial frame are

- The inertia matrix is time-invariant in body frame and
- Advantage of body symmetry can be taken to simplify the equations.

Inertial Matrix

The moment of inertia describes the dynamic behavior of a body in rotation around a defined axis. The inertia matrix for the quadcopter is a diagonal matrix, the off diagonal elements, which are the product of inertia, are zero due to the symmetry of the quadcopter.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.5}$$

Where I_{xx}, I_{yy} and I_{zz} are the moments of inertia about the principle axes in the body frame.

Gyroscopic Moment

The rotor's gyroscopic moment is a physical effect in which gyroscopic torques or moments try to align the spin axis of the rotor along the inertial z-axis[31]. The cross coupling of angular speeds induces centrifugal and coriolis forces, which create this effect.

Moments Acting on the Quadcopter (M_B)

Moments Acting on the Quadcopter, M_B, is a physical effect produced by rotors rotation. This generated moment is called the aerodynamic moment. There is also a physical effect which is produced by rotors rotation which is called aerodynamic force or the lift force. Equations (3.6) and (3.7) show the aerodynamic moment M_i and aerodynamic force F_i produced by the i^{th}rotor [30].

$$M_i = \frac{1}{2}\rho A C_D r^2 \omega_i^2 \tag{3.6}$$

$$F_i = \frac{1}{2}\rho A C_T r^2 \omega_i^2 \tag{3.7}$$

Where

$$\begin{aligned} \rho &- \textit{Air density} \\ A &- \textit{Blade area} \\ C_D,\ C_T &- \textit{Aerodynamic coefficients} \\ r &- \textit{radius of blade} \\ \omega_i &- \textit{Angular velocity of rotor} \end{aligned}$$

Clearly, the aerodynamic moment and force are influenced by the propeller's geometry as well as the air density. Since the maximum height of quadcopters is normally limited, the air density can be assumed to be constant. Equation (3.6) and (3.7) can be reduced to,

$$M_i = K_m \omega_i^2 \tag{3.8}$$

$$F_i = K_f \omega_i^2 \tag{3.9}$$

where K_M and K_f are aerodynamic moment and force constant respectively. The aerodynamic moment constant can be determined experimentally for each propeller type.
By defining the forces and moments generated by the propellers, the moments M_B acting on the quadcopter can be studied. The forces and moments acting on the quadcopter are

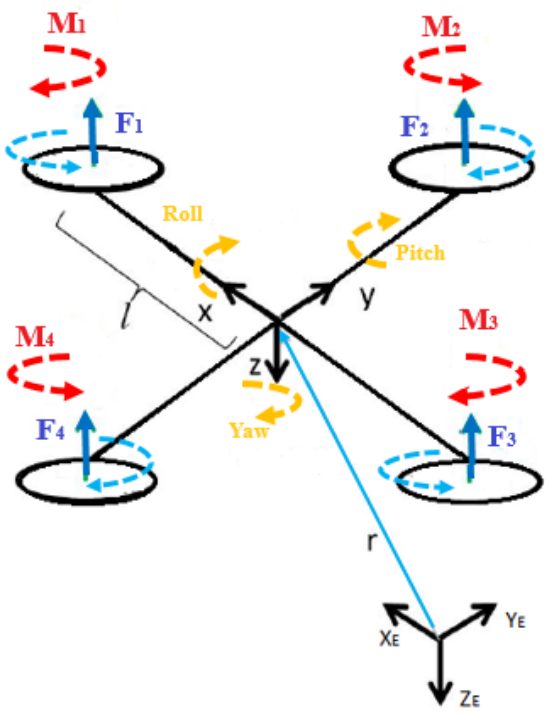


Figure 3.2: Forces and Moments acting on Quadcopter

shown in Figure 3.2. Each rotor produces an upward thrust force F_i and a moment M_i in the opposite direction of rotation as the corresponding rotor i rotates.
Starting from the moments about the x-axis of the body frame, according to right-hand law in conjunction with the axes of the body frame, F_2 multiplied by the moment arm l produces a negative moment about the y-axis, while F_4 generates a positive moment. As a result, the total moment along the x-axis can be written as;

$$\begin{aligned} M_x &= -F_2 l + F_4 l \\ &= -(K_f \omega_2^2) l + (K_f \omega_4^2) l \\ &= l K_f (-\omega_2^2 + \omega_4^2) \end{aligned} \tag{3.10}$$

The moment generated about y-axis of the body frame, also using right hand rule, the thrust of rotor 1 generates a positive moment, while the thrust of rotor 3 generates a negative

moment about the y-axis. The total moment can be written as follows:

$$\begin{aligned} M_y &= F_1 l - F_3 l \\ &= (K_f \omega_1^2) l - (K_f \omega_3^2) l \\ &= l K_f (\omega_1^2 - \omega_3^2) \end{aligned} \tag{3.11}$$

The thrust of the rotors does not produce a moment about the body frame's z-axis. On the other hand, the rotors' rotation causes a moment as seen in Equation 3.6. The moment around the z-axis of the body frame can be formulated as follows:

$$\begin{aligned} M_z &= M_1 - M_2 + M_3 - M_4 \\ &= (K_m \omega_1^2) - (K_m \omega_2^2) + (K_m \omega_3^2) - (K_m \omega_4^2) \\ &= K_m (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{aligned} \tag{3.12}$$

By combining Equation (3.10), (3.11) and (3.12) M_B expressed as follows

$$M_B = \begin{bmatrix} l K_f (-\omega_2^2 + \omega_4^2) \\ l K_f (\omega_1^2 - \omega_3^2) \\ K_m (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \tag{3.13}$$

The rotational Equation of motion is formulated by substituting Equation (3.13) in to Equation(3.4)

$$M_B = I\dot{\omega} + \omega \times I\omega + \omega \times \begin{bmatrix} 0 & 0 & I_r \omega_r \end{bmatrix}^T$$

$$\begin{bmatrix} l K_f (-\omega_2^2 + \omega_4^2) \\ l K_f (\omega_1^2 - \omega_3^2) \\ K_m (\omega_1^2 - \omega_2^2 + \Omega_3^2 - \omega_4^2) \end{bmatrix} = \begin{bmatrix} Ixx & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} Ixx & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} +$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ I_r\omega_r \end{bmatrix}$$

$$\begin{bmatrix} lK_f(-\omega_2^2+\omega_4^2) \\ lK_f(\omega_1^2-\omega_3^2) \\ K_m(\omega_1^2-\omega_2^2+\omega_3^2-\omega_4^2) \end{bmatrix} = \begin{bmatrix} I_{xx}\ddot{\phi} \\ I_{yy}\ddot{\theta} \\ I_{zz}\ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\theta}I_{zz}\dot{\psi} - \dot{\psi}I_{yy}\dot{\theta} \\ \dot{\psi}I_{xx}\dot{\phi} - \dot{\phi}I_{zz}\dot{\psi} \\ \dot{\phi}I_{yy}\dot{\theta} - \dot{\theta}I_{xx}\dot{\phi} \end{bmatrix} + \begin{bmatrix} \dot{\theta}I_r\omega_r \\ -\dot{\phi}I_r\omega_r \\ 0 \end{bmatrix}$$

$$\begin{aligned} \ddot{\phi} &= \frac{l}{I_{xx}}K_f(-\omega_2^2+\omega_4^2) - \frac{I_r}{I_{xx}}\dot{\theta}\omega_r + \frac{I_{yy}}{I_{xx}}\dot{\psi}\dot{\theta} - \frac{I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} \\ \ddot{\theta} &= \frac{l}{I_{yy}}K_f(\omega_1^2-\omega_3^2) + \frac{I_r}{I_{yy}}\dot{\phi}\omega_r + \frac{I_{zz}}{I_{yy}}\dot{\phi}\dot{\psi} - \frac{I_{xx}}{I_{yy}}\dot{\psi}\dot{\phi} \\ \ddot{\psi} &= \frac{1}{I_{zz}}K_m(\omega_1^2-\omega_2^2+\omega_3^2-\omega_4^2) + \frac{I_{xx}}{I_{zz}}\dot{\theta}\dot{\phi} - \frac{I_{yy}}{I_{zz}}\dot{\phi}\dot{\theta} \end{aligned} \tag{3.14}$$

The rotational Equation of motion (3.14) is formulated by substituting Equation (3.13) in to Equation(3.4) as seen above.

3.2.2 Translational Equations of Motion

The quadcopter's translational equation of motion is derived in the Earth inertial frame and based on Newton's second law [30].

$$ma = \sum F$$

Where

$$\begin{aligned} m &- mass \\ a &- accleration \\ F &- Forces\ on\ the\ system \end{aligned}$$

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + RF_B \tag{3.15}$$

Where

$$r = \begin{bmatrix} x & y & z \end{bmatrix}^T - \textit{Quadcopter's distance from the inertial frame}$$

$$m - \textit{Quadcopter's mass}$$

$$g - \textit{gravitational acceleration } (g = 9.81 m/s^2)$$

$$F_B - \textit{nongravitational force acting on the quadcopter on the body frame}$$

$$R - \textit{Rotational matrix}$$

Nongravitational Forces Acting on the Quadrotor

When the quadcopter is in a horizontal orientation (i.e., not rolling or pitching), the only nongravitational forces acting on it is the thrust provided by the propellers' rotation, which is equal to the square of the propeller's angular velocity, as seen in Equation (3.9). Therefore, the nongravitational forces acting on the quadcopter, F_B, can be written as,

$$F_B = \begin{bmatrix} 0 \\ 0 \\ -K_f(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{bmatrix} \tag{3.16}$$

Since there are no forces in the X and Y directions, the first two rows of the force vector are zeros; the last row is a sum of the thrust forces provided by each propellers. Since the thrust is upwards and the positive z-axis in the body framed is pointed downwards, the symbol is negative.

The thrust force of the rotors are transformed from the body frame to the inertial frame by multiplying F_B by the rotation matrix R, allowing the equation to be extended in any orientation of the quadcopter.

By expanding the terms of translational equation of motion (3.15) gives

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix}=\begin{bmatrix}0\\ 0\\ mg\end{bmatrix}+\begin{bmatrix}c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi\\ c\theta s\psi & s\phi s\theta s\psi + c\theta c\psi & c\phi s\theta s\psi - s\phi c\psi\\ -s\theta & s\phi c\theta & c\phi c\theta\end{bmatrix}\begin{bmatrix}0\\ 0\\ -K_f(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2)\end{bmatrix}$$

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix}=\begin{bmatrix}(c\phi s\theta c\psi + s\phi s\psi)(-K_f(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2))\\ (c\phi s\theta s\psi - s\phi c\psi)(-K_f(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2))\\ (c\phi c\theta)(-K_f(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2))+mg\end{bmatrix} \tag{3.17}$$

The accelerations, in terms of the other variables, can be gotten by rewriting equation (3.17).

$$\begin{aligned}\ddot{x} &= \frac{-K_f(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2)}{m}(c\phi s\theta c\psi + s\phi s\psi)\\ \ddot{y} &= \frac{-K_f(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2)}{m}(c\phi s\theta s\psi - s\phi c\psi)\\ \ddot{z} &= g - \frac{K_f(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2)}{m}(c\phi c\theta)\end{aligned} \tag{3.18}$$

As seen on the above equation, the translational subsystem is underactuated and it is dependent on both translational and rotational variables.

3.3 Aerodynamic Effects

The aerodynamic effects acting on the quadcopter body were ignored in the previous dynamics formulation. Aerodynamic impacts, on the other hand, should be considered in order to provide a reliable and practical model that can be used in simulations. Drag forces and drag moments are the two kinds of aerodynamic effects [32].

3.3.1 Drag Forces

Because of the friction between the moving quadcopter body and the air, a drag force acts on the quadcopter's body for resisting motion. The drag forces on the quadcopter increase

as the quadcopter's velocity increases. F_d, the drag force, can be approximated by,

$$F_d = K_t \dot{r} \tag{3.19}$$

Where

$$\begin{aligned} K_t &- \textit{Aerodynamic translation coeffcient matrix (it is constant matrix)} \\ \dot{r} &- \textit{Time derivative of the position vector } r \end{aligned}$$

The translational equation of motion, Equation (3.15), should be rewritten, because there is an extra force acting on the quadcopter body.

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + RF_B - F_d \tag{3.20}$$

3.3.2 Drag Moments

There is also a drag moment, M_d, acting on the quadcopter body due to air friction, which can be approximated by,

$$M_d = K_r \dot{\eta} \tag{3.21}$$

Where

$$\begin{aligned} K_r &- \textit{Aerodynamic rotational coeffcient matrix (it is constant matrix)} \\ \dot{\eta} &- \textit{Euler rate} \end{aligned}$$

Accordingly, the rotational equation of motion expressed by Equation (3.4) can be rewritten as,

$$M_B - M_d = I\dot{\omega} + \omega \times I\omega + \omega \times \begin{bmatrix} 0 & 0 & I_r\omega_r \end{bmatrix}^T \tag{3.22}$$

3.4 Rotor Dynamics

The motors typically used in quadcopters are brushless DC motors that provide high torque and little friction. In the following derivation, it is assumed that the rotors are nongeared with rigid mechanical coupling between the motors and the propellers. The dynamics of

a brushless DC motor at steady state is the same as a conventional DC motor [33]. The schematic for a brushless DC motor at steady state is shown in Figure 3.3. Using Kirchhof's

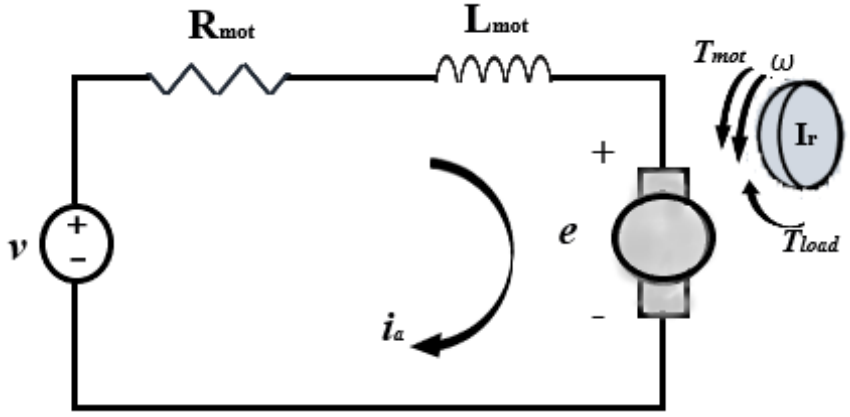


Figure 3.3: DC Motor schematic diagram

voltage law, the following equation can be derived

$$v = R_{mot} i_a + L_{mot} \frac{di_a}{dt} + K_{mot} \omega \tag{3.23}$$

Where;

$$\begin{aligned} R_{mot} &- i_{th}\ motor's\ resistance \\ L_{mot} &- i_{th}\ motor's\ inductance \\ i_a &- Armature\ current \\ v &- the\ Input\ voltage \\ K_{mot} &- Motor\ torque\ constant \\ \omega &- i_{th}\ motor's\ speed \end{aligned}$$

The term $K_{mot}\omega$ represents the generated emf, e. Since the quadcopter relies on small motors, their inductance is very small and thus can be neglected leading to

$$v = R_{mot} i_a + K_{mot} \omega \tag{3.24}$$

or

$$i_a = \frac{v - K_{mot}\omega}{R_{mot}} \tag{3.25}$$

Moving to the mechanical derivation

$$I_r\dot{\omega} = T_{mot} - T_{load} \tag{3.26}$$

T_{mot} is the torque produced by the motor.

$$T_{mot} = K_e i_a$$

T_{load} is the load torque which is the torque generated from the propeller system. Where K_e is motor's electric constant, for small motors it is approximately equal to K_{mot}.
Substituting the equation of T_{mot} together with the current equation from Equation (3.22) yields,

$$I_r\dot{\omega} = K_{mot}\frac{v - K_{mot}\omega}{R_{mot}} - T_{load} \tag{3.27}$$

After simplification the voltage can be written as a function of the rotor's velocity as follows

$$v = \frac{R_{mot}}{K_{mot}} I_r\dot{\omega} + K_{mot}\omega + \frac{R_{mot}}{K_{mot}} T_{load} \tag{3.28}$$

3.5 State Space Model

The control problem would be easier to solve if the quadcopter's mathematical model is converted into a state space model.

State Vector x

Expressing the state vector of the quadcopter to be,

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{bmatrix}^T \tag{3.29}$$

which is mapped to the quadcopter's degrees of freedom in the following way:

$$\mathbf{x} = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & x & \dot{x} & y & \dot{y} & z & \dot{z} \end{bmatrix}^T \tag{3.30}$$

The quadcopter's orentation and position in space, as well as its angular and linear velocities, are defined by the state vector.

Control Input Vector U

The control input variable, **U**, is a vector which consists four inputs. The four inputs, U_1, U_2, U_3 and U_4, are defined as

$$\mathbf{U} = \begin{bmatrix} U_1 & U_2 & U_3 & U_4 \end{bmatrix} \tag{3.31}$$

Where

$$\begin{aligned} U_1 &= K_f(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_2 &= lK_f(-\omega_2^2 + \omega_4^2) \\ U_3 &= lK_f(\omega_1^2 - \omega_3^2) \\ U_4 &= K_m(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{aligned} \tag{3.32}$$

Equations (3.33) can be rearranged in a matrix form to result in,

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ 0 & -lK_f & 0 & lK_f \\ lK_f & 0 & -lK_f & 0 \\ K_m & -K_m & K_m & -K_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3.33}$$

U_1 is the resultant upward force of the four rotors, which affects the quadcopter's altitude and altitude rate of change$(z, \dot{z})$. U_2 is the difference in thrust of rotors 2 and rotor 4 which is responsible for the roll rotation and its rate of change$(\phi, \dot{\phi})$. U_3 on the other hand represents the difference of rotors 1 thrust and rotor 3 thrust which affect pitch rotation and its rate of change$(\theta, \dot{\theta})$. Finally, U_4 is the torque difference between the two clockwise and counterclockwise rotating rotors that generate the yaw rotation, as well as its rate of change $(\psi, \dot{\psi})$.

From the control inputs rotors velocities can be derived. The relationship between the control

inputs and the rotors' velocities is derived using Equation (3.35) as follows

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4K_f} & 0 & \frac{1}{2lK_f} & \frac{1}{4K_m} \\ \frac{1}{4K_f} & -\frac{1}{2lK_f} & 0 & -\frac{1}{4K_m} \\ \frac{1}{4K_f} & 0 & -\frac{1}{2lK_f} & \frac{1}{4K_m} \\ \frac{1}{4K_f} & \frac{1}{2lK_f} & 0 & -\frac{1}{4K_m} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{3.34}$$

The rotors' velocities can be determined using the square root of the above equation as follows:

$$\begin{aligned} \omega_1 &= \sqrt{\frac{1}{4K_f}U_1 + \frac{1}{2lK_f}U_3 + \frac{1}{4K_m}U_4} \\ \omega_2 &= \sqrt{\frac{1}{4K_f}U_1 - \frac{1}{2lK_f}U_2 - \frac{1}{4K_m}U_4} \\ \omega_3 &= \sqrt{\frac{1}{4K_f}U_1 - \frac{1}{2lK_f}U_3 + \frac{1}{4K_m}U_4} \\ \omega_4 &= \sqrt{\frac{1}{4K_f}U_1 + \frac{1}{2lK_f}U_2 - \frac{1}{4K_m}U_4} \end{aligned} \tag{3.35}$$

Rotational Equation Of Motion

Substituting equations (3.33) in equation (3.13), the equation of the total moments acting on the quadcopter becomes

$$M_B = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{3.36}$$

Rewrite Equation (3.14)

$$\begin{aligned}
\ddot{\phi} &= \frac{l}{I_{xx}}U_2 - \frac{I_r}{I_{xx}}\dot{\theta}\omega_r + \frac{I_{yy}}{I_{xx}}\dot{\psi}\dot{\theta} - \frac{I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} \\
\ddot{\theta} &= \frac{l}{I_{yy}}U_3 - \frac{I_r}{I_{yy}}\dot{\phi}\omega_r + \frac{I_{zz}}{I_{yy}}\dot{\phi}\dot{\psi} - \frac{I_{xx}}{I_{yy}}\dot{\psi}\dot{\phi} \\
\ddot{\psi} &= \frac{1}{I_{zz}}U_4 + \frac{I_{xx}}{I_{zz}}\dot{\theta}\dot{\phi} - \frac{I_{yy}}{I_{zz}}\dot{\phi}\dot{\theta}
\end{aligned} \tag{3.37}$$

For simplification, define,

$$\begin{aligned}
a_1 &= \frac{I_{yy} - I_{zz}}{I_{xx}} & a_4 &= \frac{I_r}{I_{yy}} & b_1 &= \frac{l}{I_{xx}} \\
a_2 &= \frac{I_r}{I_{xx}} & a_5 &= \frac{I_{xx} - I_{yy}}{I_{zz}} & b_2 &= \frac{l}{I_{yy}} \\
a_3 &= \frac{I_{zz} - I_{xx}}{I_{yy}} & & & b_3 &= \frac{l}{I_{zz}}
\end{aligned}$$

Using the definition of constant coefficients, $a_1,\ a_2,\ ....\ a_5$ *and* $b_1,\ b_2,\ b_3$, Equation (3.38) can then be rewritten in a simpler form,

$$\begin{aligned}
\ddot{\phi} &= b_1U_2 - a_2\dot{\theta}\omega_r + a_1\dot{\psi}\dot{\theta} \\
\ddot{\theta} &= b_2U_3 - a_4\dot{\phi}\omega_r + a_3\dot{\phi}\dot{\psi} \\
\ddot{\psi} &= b_3U_4 + a_5\dot{\theta}\dot{\phi}
\end{aligned}$$

Therefore, the rotational subsystem state space representation becomes

$$
\begin{aligned}
\dot{x_1} &= \dot{\phi} = x_2 \\
\dot{x_2} &= \ddot{\phi} = x_4 x_6 a_1 - x_4 \omega_r a_2 + b_1 U_2 \\
\dot{x_3} &= \dot{\theta} = x_4 \\
\dot{x_4} &= \ddot{\theta} = x_2 x_6 a_3 + x_2 \omega_r a_4 + b_2 U_3 \\
\dot{x_5} &= \dot{\psi} = x_6 \\
\dot{x_6} &= \ddot{\psi} = x_2 x_4 a_5 + b_3 U_4
\end{aligned}
$$

$$
f_1(\mathbf{x}, \mathbf{U}) = \begin{bmatrix} x_2 \\ x_4 x_6 a_1 - x_4 \omega_r a_2 + b_1 U_2 \\ x_4 \\ x_2 x_6 a_3 + x_2 \omega_r a_4 + b_2 U_3 \\ x_6 \\ x_2 x_4 a_5 + b_3 U_4 \end{bmatrix} \tag{3.38}
$$

Translational Equation Of Motion

Substituting equations (3.33) in equation (3.16), the equation of the nongravitational force acting on the quadcopter becomes

$$
F_B = \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} \tag{3.39}
$$

Rewrite Equation (3.18),

$$\begin{aligned} \ddot{x} &= \frac{-U_1}{m}(c\phi s\theta c\psi + s\phi s\psi) \\ \ddot{y} &= \frac{-U_1}{m}(c\phi s\theta s\psi - s\phi c\psi) \\ \ddot{z} &= g - \frac{U_1}{m}(c\phi c\theta) \end{aligned} \tag{3.40}$$

The translational subsystem state space representation becomes,

$$\begin{aligned} \dot{x_7} &= \dot{x} = x_{10} \\ \dot{x_8} &= \ddot{x} = \frac{-U_1}{m}(cosx_1 sinx_3 cosx_5 + sinx_1 sinx_5) \\ \dot{x_9} &= \dot{y} = x_{12} \\ \dot{x_{10}} &= \ddot{y} = \frac{-U_1}{m}(cosx_1 sinx_3 sinx_5 - sinx_1 cosx_5) \\ \dot{x_{11}} &= \dot{z} = x_8 \\ \dot{x_{12}} &= \ddot{z} = g - \frac{U_1}{m}(cosx_1 cosx_3) \end{aligned}$$

$$f_2(\mathbf{x}, \mathbf{U}) = \begin{bmatrix} x_8 \\ \frac{-U_1}{m}(cosx_1 sinx_3 cosx_5 + sinx_1 sinx_5) \\ x_{10} \\ \frac{-U_1}{m}(cosx_1 sinx_3 sinx_5 - sinx_1 cosx_5) \\ x_{12} \\ g - \frac{U_1}{m}(cosx_1 cosx_3) \end{bmatrix} \tag{3.41}$$

From Equation (3.39) and (3.42), it can be seen that the rotational subsystem is fully actuated and it is only dependent on the rotational state variables $x_1 to x_6$ that correspond to $\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}$ respectively, and the translational subsystem is underactuated as it dependent

on both the translational state variables and the rotational ones.

State Space Representation

Using the equations of the rotational angular acceleration, Equation (3.39), and those of translation, Equation (3.42), the complete mathematical model of the quadcopter can be written in a state space representation as follows,

$$f(\mathbf{x},\mathbf{U}) = \begin{bmatrix} x_2 \\ x_4x_6a_1 \ - \ x_4\omega_r a_2 \ + \ b_1U_2 \\ x_4 \\ x_2x_6a_3 \ + \ x_2\omega_r a_4 \ + \ b_2U_3 \\ x_6 \\ x_2x_4a_5 \ + \ b_3U_4 \\ x_8 \\ \frac{-U_1}{m}(cosx_1sinx_3cosx_5 + sinx_1sinx_5) \\ x_{10} \\ \frac{-U_1}{m}(cosx_1sinx_3sinx_5 - sinx_1cosx_5) \\ x_{12} \\ g - \frac{U_1}{m}(cosx_1cosx_3) \end{bmatrix} \tag{3.42}$$

Chapter 4

System Control Design

In this chapter the control strategies that has been used in this work are explained. Quadcopter system can be seen as nonlinear and unstable system, for that reason it is important to design efficient and reliable control system. There are many techniques that have been developed for controlling quadcopter. The most popular technique used by researchers is a cascade control approach. In a cascade control setup, there are two controllers which are called inner loop controller and outer loop controller. Outer loop controller's output generate the set point of the inner loop controller. In this technique the complex control system is divided in to two connected subsystems. As a result, one divided subsystem can be used for attitude control, while the other can be used for position control.

The designed controllers for attitude subsystem, inner loop control, are three different Model predictive control based algorithms. The three different MPC algorithm are Linear MPC, MPC with feedback linearization and nonlinear MPC. For the lateral position, the outer loop, PID controller is designed.

The command is made for the three position coordinates and orientation of the yaw i.e. the controllers inputs are the desired reference trajectory with the desired heading (yaw angle). The conventional PID controllers receive the reference trajectory command and generate orientation instructions, which are roll and pitch angle reference, and Trust force. For attitude control, MPC scheme is then applied. The MPC reference inputs are the instructions for roll, pitch, and yaw command. The roll and pitch commands have been generated from the conventional PID controllers, whereas the yaw command is user defined. Each control systems are formulated and modeled in MATLAB/Simulink. The proposed control structure is shown in Figure 4.1. The physical features of the quadcopter were necessary in the design of the proposed controllers. In Table 4.1, a summary of the physical

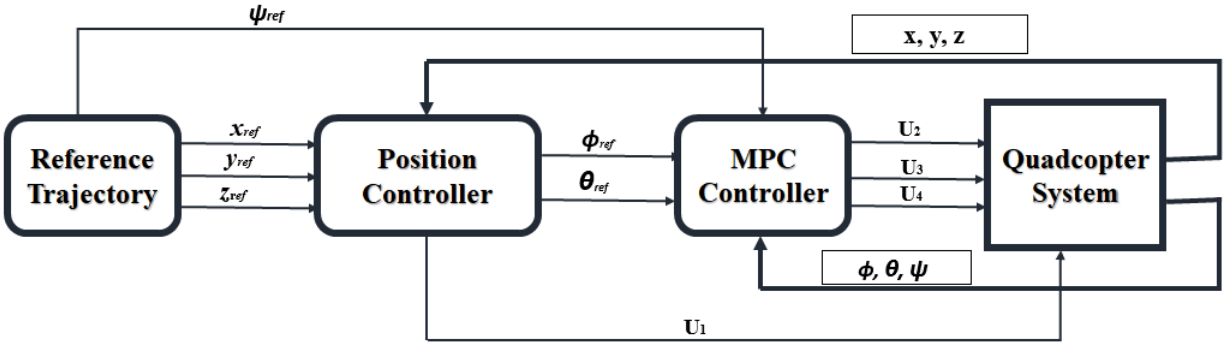


Figure 4.1: System Control Structure

properties that are adopter form [1] is shown.

Parameters	Values	unit
Moment of inertia about X-axis, I_{xx}	2.217×10^{-2}	$kg.m^2$
Moment of inertia about Y-axis, I_{yy}	2.217×10^{-2}	$kg.m^2$
Moment of inertia about Z-axis, I_{zz}	2.82×10^{-2}	$kg.m^2$
Moment arm, l	0.243	m
Rotor inertia, I_r	~ 0	$kg.m^2$
Mass, m	1.587	kg
Aerodynamics thrust constant, K_f	4.0687×10^{-7}	N/rpm^2
Aerodynamics moment constant, K_m	8.4367×10^{-9}	Nm/rpm^2

Table 4.1: Quadcopter parameters and constants[1]

4.1 Overview of Model Predictive Control

Model Predictive Control (MPC) is an optimization-based control methodology that solves a finite-horizon optimal control problem while subjecting the system to specified input and state constraints at each sampling period. In the receding or moving horizon technique, future outputs are estimated at each sampling time t for a specific prediction horizon using the system/process model. Solving an optimization problem for a control horizon with a specified optimality criteria yields the control sequence, or future inputs.

The general goal of MPC problem is to find a control input $\mathbf{u}$ to a system,

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

That minimizes a cost function J which is a function state x and input u.
A mathematical model of the process to be controlled is utilized in model predictive control to predict the model output of the process across a time horizon known as the prediction horizon. The selected sample time is for each prediction step, with the number of steps equaling the prediction horizon's size. Over a control horizon, a series of control inputs is generated by minimizing a cost function of the error between predicted outputs and reference or set point values, as well as a weighted control input. The size of the control horizon is equal to the number of control inputs in this sequence. The control actions required to drive the model and process to the reference values are represented by this series of control inputs. Because the process measurement and reference values are continually updated in successive sampling instants, just the first element of the control input sequence is applied in both the model and the process. This minimization and implementation approach is repeated at subsequent sampling instants, with the assumption that the reference value remains constant throughout the prediction horizon at each instant. Figure 4.2 shows the MPC principle in a block diagram, while Figure 4.3 shows the prediction horizon in MPC with a sampling time of one. The present measured states or the outputs of the process are utilized in

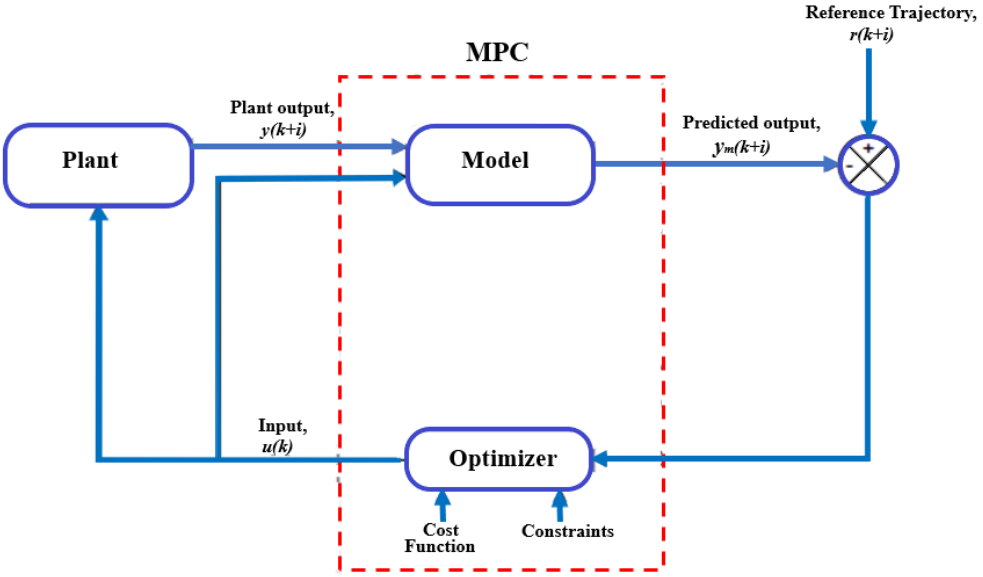


Figure 4.2: Principle of Model Predictive Control (Adapted from [1])

updating the future predictions to guarantee that a more accurate model of the process is in use. Because the prediction horizon is always going away at future sample instants, the

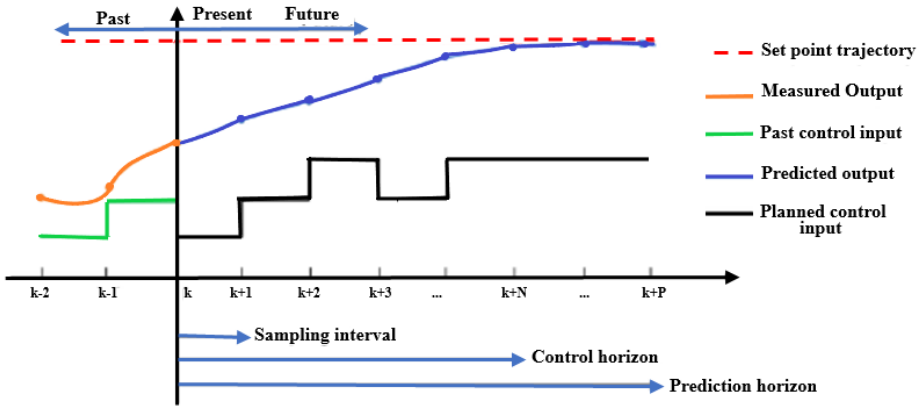


Figure 4.3: Prediction Horizon (Adapted from [2])

phrase receding horizon is frequently used to describe predictive control. Points that were previously beyond the prediction horizon are taken into account at the current sampling time [34].

The following table illustrate the receding horizon principle.

Sampling instant	Horizon window
	0 1 2 3 4 5 6 7 8
0	→
1	→
2	→
3	→
⋮	⋮

Table 4.2: Illustration of receding horizon ([2])

MPC's capacity to systematically meet physical restrictions on control inputs and outputs is one of its most appealing features. The optimization process handles these restrictions directly. Figure 4.2 depicts the optimization process in the optimizer block, which receives constraints and a cost function as inputs. MPC includes two primary components, as shown in Figure 4.2 a plant model and an optimizer. The following are the explanations for both:

Plant model

A model of the process/plant might be linear or nonlinear in an MPC controller. For system performance prediction, linear model predictive control (LMPC) use a linear plant/process model. Nonlinear model predictive control predicts system performance using a nonlinear plant/process model. MPC's algorithm is commonly used in digital devices such as computers, microcontrollers, and other types of microprocessors. As a result, the model of the process (that is being controlled) should be discrete. The goal of discretization is to make it possible to execute the model digitally at a specific sample rate. In discrete time, the continues time model can be represented by a difference equation of the following form:

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k))$$

Where k is a sampling instant.
The model is then augmented using the method that presented in Liuping Wang (2009) [35], which will be employed in the design of predictive control.
The big advantage of augmented plant model is to achieve off-set free tracking, integral action needs to be embedded by modifying the plant model [1]. The following steps show how the plant model is augmented.
The linear discrete state space model used in formulating a linear MPC controller state space equations could be stated as follows.

$$x_m(k+1) = A_m x_m(k) + B_m u(k) \tag{4.1}$$

$$y_m(k) = C_m x(k) \tag{4.2}$$

Then,

$$x_m(k+1) - x_m(k) = A_m(x_m(k) - x_m(k-1)) + B_m(u(k) - u(k-1)) \tag{4.3}$$

$$y_m(k+1) = C_m(x_m(k+1) - x_m(k)) + y(k) \tag{4.4}$$

Let,

$$\Delta x_m(k+1) = x_m(k+1) - x_m(k)$$
$$\Delta x_m(k) = x_m(k) - x_m(k-1)$$
$$\Delta u(k) = u(k) - u(k-1)$$

Equation (4.3) and (4.4) became,

$$\Delta x_m(k+1) = A_m \Delta x_m + B_m \Delta u(k) \tag{4.5}$$

$$y_m(k+1) = C_m(A_m\Delta x_m + B_m\Delta u(k)) + y(k)$$

$$y_m(k+1) = C_mA_m\Delta x_m + C_mB_m\Delta u(k) + y(k) \tag{4.6}$$

Now the input to the state space model is $\Delta u(k)$. The vector that relates Δx_m and $y(k)$ is,

$$x(k) = \left[\Delta x_m(k)^T y(k)\right]^T \tag{4.7}$$

Where T stands for the matrix transpose.
The augmented state space model became,

$$\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} A_m & O_{q\times n}^T \\ C_mA_m & I_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_m \\ C_mB_m \end{bmatrix} \Delta u(k) \tag{4.8}$$

$$y(k) = \begin{bmatrix} O_{q\times n} & I_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{4.9}$$

Where,

$I_{q\times q}-$ is the identity matrix with dimensions $q \times q$, where q is the number of outputs.

$O_{q\times n}-$ is zero matrix with dimensions $q \times n$, where n is the number of states of the system or the state space dimensions.

Generally the augmented model can be written as follows:

$$x_a(k+1) = A_a x_a(k) + B_a\Delta u(k) \tag{4.10}$$

$$y_a(k) = C_a x_a(k) \tag{4.11}$$

Where, A_a, B_a and C_a corresponding to the augmented state matrices.

Optimizer

Model predictive control aims to bring predicted outputs closer to the targeted reference values. This is accomplished by minimizing a cost function. This optimization yields a set of inputs, ΔU, which are utilized to drive the predicted outputs to the reference. The general optimization problem is formulated as follows,

$$\min J(\mathbf{x}, \mathbf{u})$$

Subjected to:
$\mathbf{x}(0) = \mathbf{x}_0$;
$\mathbf{x}_{min} \leq \mathbf{x}(k) \leq \mathbf{x}_{max}$
$\mathbf{u}_{min} \leq \mathbf{u}(k) \leq \mathbf{u}_{max}$
Where $\mathbf{x}$ is state vector, $\mathbf{u}$ is input vector and $\mathbf{x}_0$ is the initial condition of state.
At each control interval, model predictive control solves an optimization issue. Until the next control interval, the solution decides the control inputs to be employed in the plant. The key items of this optimization issue are presented below;

- The objective, or "cost", function; it is a nonnegative, scalar measure of controller performance that should be minimized.
- Constraints; Physical restrictions on control inputs and plant output variables and they are illustrations of conditions that the solution must meet.
- Decision; The adjustments to the control inputs that minimize the cost function while meeting the restrictions.

The Model Predictive Control Toolbox^{TM} package in MATLAB comes with two built-in optimization algorithms/solvers in order to solve the optimization problem. Interior Point solver and Active Set solver are the two solvers. They are commonly employed to solve quadratic programming issues. These solvers use different approaches to dealing with constraints.

An active-set solver can offer rapid and reliable solutions for small and medium scale optimization problems. For large-scale optimization problems, such as MPC applications that apply constraints over vast prediction and control horizons, an interior-point solver can give improved performance. The performance of the built-in solvers are affected by the size and configuration of the MPC problem. Multiple controller simulation scenarios employing both solvers are required to determine which solver is better for a certain application.

4.2 Attitude Control

The control algorithms for the quadcopter's attitude subsystem will be described in this section. Linear Model Predictive Control, Feedback Linearization with Model Predictive Control, and nonlinear Model Predictive Control are the three designed control approaches.

4.2.1 Linear Model Predictive Control

In MPC control a plant model might be linear or nonlinear. Linear model predictive control (LMPC) use a linear plant model for prediction and linear constraints in its optimization problem.

Prediction Model

LMPC algorithm require linear dynamic model of the plant. Therefore, the nonlinear attitude subsystem state space model have to be linearized at trim point. The nonlinear attitude subsystem state space equation was stated in equation (3.38),

$$f_1(\mathbf{x},\mathbf{U}) = \begin{bmatrix} x_2 \\ x_4x_6a_1 \ - \ x_4\omega_r a_2 \ + \ b_1U_2 \\ x_4 \\ x_2x_6a_3 \ + \ x_2\omega_r a_4 \ + \ b_2U_3 \\ x_6 \\ x_2x_4a_5 \ + \ b_3U_4 \end{bmatrix}$$

The nonlinear attitude dynamics is linearized around equilibrium points. The equilibrium points are defined below [36].

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The obtained linear attitude dynamics forms a continuous time state space model which is expressed as follows;

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{U} \\ \mathbf{y} &= C\mathbf{x} \end{aligned} \tag{4.12}$$

In the above equation A, B and C are state, input and output matrices respectively of the linearized state space model. $\mathbf{x}, \mathbf{U}$ and $\mathbf{y}$ are state, input and output vectors respectively.

The linearization is done using Model Predictive Control Toolbox^{TM} software in MATLAB, due to this it is not explicitly expressed here. The state space form of the linear dynamics is a continuous time, however, LMPC algorithm is solved in discrete time. Then, continuous time equation obtained in equation (4.12) is discretized by choosing 0.1sec sample time. It is recommended that the sample time(T_s) should be between

$$\frac{T_r}{20} \leq T_s \leq \frac{T_r}{10}$$

Where T_r is the rise time, which takes for the response to rise from 10% to 90% of the steady state response, of open loop system step response.
The sampling time is then chosen by trial-and-error, using simulations to judge the effectiveness.
Discrete time state space equation obtained by using MATLAB can be defined as follows:

$$\begin{aligned} \mathbf{x}(k+1) &= A_d\mathbf{x}(k) + B_d\mathbf{U}(k) \\ \mathbf{y}(k) &= C_d\mathbf{x}(k) \end{aligned} \tag{4.13}$$

In equation (4.13), A_d, B_d, and C_d are discrete time state, input and output matrices respectively. "k" represents discrete time step. The model is then augmented.

Predicted states and output variables

Future state variables are necessary to determine the future plant output. Because the future controller states can't be measured, they have to be estimated. In order to predict future controller states, the controller employs a steady-state Kalman filter [37] and the following prerequisites must be met:

- The prediction model has to be time invariant.

- The prediction model has to be controllable and observable.

The variables for the future state are defined as follows:

$$x(k+1), \quad x(k+2), \quad x(k+3), \quad \dots, \quad x(k+p) \tag{4.14}$$

Where, k is the current sample time and p is the prediction horizon.
The controller calculates the future plant output using the future control signal as the adjustable variable. Within a preset prediction horizon, this prediction is made. The following equation depicts the future control trajectory:

$$\Delta\mathbf{U} = \left[\Delta\mathbf{U}(k), \quad \Delta\mathbf{U}(k+1), \quad \Delta\mathbf{U}(k+2), \quad \dots, \quad \Delta\mathbf{U}(k+n-1) \right]^T \tag{4.15}$$

Where, k is the current sample time and n is control horizon.
The prediction horizon must be chosen to be longer than the control horizon. Using the discretized and augmented model in equations, the future state and output predictions are generated repeatedly as follows:
at $k+1$,

$$\mathbf{x}(k+1) \; = \; A\mathbf{x}(k) \; + \; B\Delta\mathbf{U}(k) \tag{4.16}$$

$$\mathbf{y}(k+1) \; = \; C\mathbf{x}(k+1) \tag{4.17}$$

at $k+2$,

$$\mathbf{x}(k+2) \; = \; A\mathbf{x}(k+1) \; + \; B\Delta\mathbf{U}(k+1) \tag{4.18}$$

$$\mathbf{y}(k+2) \; = \; C\mathbf{x}(k+2) \tag{4.19}$$

By putting equation 4.16 and 4.17 in to equation 4.18 and 4.19 in order to eliminate $\mathbf{x}(k+1)$, results in

$$\mathbf{x}(k+2) \; = \; A^2\mathbf{x}(k) \; + \; AB\Delta\mathbf{U}(k) \; + \; B\Delta\mathbf{U}(k+1) \tag{4.20}$$

$$\mathbf{y}(k+2) \; = \; C(A^2\mathbf{x}(k) \; + \; AB\Delta\mathbf{U}(k) \; + \; B\Delta\mathbf{U}(k+1)) \tag{4.21}$$

at $k+3$,

$$x(k+3) \; = \; A\mathbf{x}(k+2) \; + \; B\Delta\mathbf{U}(k+2) \tag{4.22}$$

$$\mathbf{y}(k+3) \;=\; C\mathbf{x}(k+3) \tag{4.23}$$

Putting equation 4.20 and 4.21 in to 4.22 and 4.23

$$\mathbf{x}(k+3) \;=\; A^2\mathbf{x}(k+1) \;+\; AB\Delta\mathbf{U}(k+1) \;+\; B\Delta\mathbf{U}(k+2) \tag{4.24}$$

$$\mathbf{y}(k+3) \;=\; C(A^2\mathbf{x}(k+1) \;+\; AB\Delta\mathbf{U}(k+1) \;+\; B\Delta\mathbf{U}(k+2)) \tag{4.25}$$

Similarly, in order to eliminate $\mathbf{x}(k+1)$, equations 4.16 and 4.17 are inserted into equations 4.24 and 4.25

$$\mathbf{x}(k+3) \;=\; A^3\mathbf{x}(k+1) \;+\; A^2B\Delta\mathbf{U}(k) \;+\; AB\Delta\mathbf{U}(k+1) \;+\; B\Delta\mathbf{U}(k+2) \tag{4.26}$$

$$\mathbf{y}(k+3) \;=\; C(A^3\mathbf{x}(k+1) \;+\; A^2B\Delta\mathbf{U}(k) \;+\; AB\Delta\mathbf{U}(k+1) \;+\; B\Delta\mathbf{U}(k+2)) \tag{4.27}$$

In order to generate p-step forward predictions, this procedure is repeated over and over.

Constraints

Constraints are limitations that must be considered when developing a real-world system. The input and angle restrictions of the quadcopter, which is the controlled system in this work, were taken into account in the MPC formulation.

Input constraints

The quadcopter system is constrained by the maximum speeds of the motors. The four brushless motors have a maximum speed of 4720 revolutions per minute. In order to calculate input constraints, the total thrust produced by the motors had to be determined. The quadcopter is 1.587 kg in weight. Using a $9.81 m/s^2$ gravitational acceleration,

$$weight = 1.587 \times 9.81 = 15.57N$$

The thrust force is calculated as follows;

$$T = K_f \sum_{i=1}^{4} \omega_i^2 \tag{4.28}$$

Where

$$K_f \ - \ thrust\ coefficient,\ N/rpm^2$$
$$\omega_i \ - \ speed\ of\ each\ motor,\ rpm$$

From Table 4.1 the value of thrust coefficient is $4.0687 \times 10^{-7} N/rpm^2$. The calculation determine the smallest speed of each quadcopter motor which is necessary to overcome gravitational force:

$$15.57 = K_f \sum_{i=1}^{4} \omega_i^2$$

$$\omega_i = \sqrt{\frac{15.57}{4 \times 4.0687 \times 10^{-7}}}$$
$$\approx 3093 rpm$$

As described in the previous chapters , torques in the roll, pitch, and yaw axes control the quadcopter's rotational motion. The torque equations are as follows:

$$\begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} lK_f(\omega_4^2 - \omega_2^2) \\ lK_f(\omega_1^2 - \omega_3^2) \\ K_m(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{bmatrix}$$

The input constraints are the maximum and lowest torques, which are determined below. The maximum input torques are:

$$U_2^{max} = lK_f(\omega_{4max}^2 - \omega_{2min}^2) = 1.257 \tag{4.29}$$

$$U_3^{max} = lK_f(\omega_{1max}^2 - \omega_{3min}^2) = 1.257 \tag{4.30}$$

$$U_4^{max} = K_m(\omega_{1max}^2 + \omega_{3max}^2 - \omega_{2min}^2 - \omega_{4min}^2) = 0.2145 \tag{4.31}$$

A vector representation of these terms is as follows:

$$U^{max} = \begin{bmatrix} 1.257 & 1.257 & 0.2145 \end{bmatrix} \tag{4.32}$$

The minimum input torques were obtained by switching the direction of adjacent motors.

$$U^{min} = \begin{bmatrix} -1.257 & -1.257 & -0.2145 \end{bmatrix} \tag{4.33}$$

Output constraints

It's also critical to keep the angles limited so that no kinematic singularities emerge as a result of the model's limitation. As a result, the angles must fall within the boundaries of the following equation.

- Roll angle, ϕ, must be constrained between $-\pi/2$ and $\pi/2$

$$-\pi/2 < \phi < \pi/2 \tag{4.34}$$

- Pitch angle, θ, must be constrained between $-\pi/2$ and $\pi/2$

$$-\pi/2 < \theta < \pi/2 \tag{4.35}$$

- Yaw angle, ψ, must be constrained between $-\pi$ and π

$$-\pi < \psi < \pi \tag{4.36}$$

Optimization

At each control interval, the controller solves an optimization problem, specifically a quadratic program (QP), to select the control inputs to use in the plant until the next control interval. The amount of variables and constraints in this thesis is considered small due to the focus is on attitude control. To tackle the linear MPC quadratic programming issue, the active set algorithm was chosen. The main advantage of the active set algorithm is that it can be warm started. Warm starting means that, due to the nature of the MPC problem, it is possible to find the solution much faster at time step (k +1) if the algorithm can use knowledge gained from the solution evaluated at previous time (k) [38]. The detail of quadratic programming is found on appendix.
The objective or cost function is formulated as follows:

$$J(z_k) = \sum_{i=0}^{p-1} [e_y^T(k+i)Qe_y(k+i)] + [\Delta u^T(k+i)R\Delta u(k+i)] \tag{4.37}$$

Where,

$$
\begin{aligned}
Q &- \textit{Positive semi} - \textit{definite matrix it size is } n_u \times n_u \textit{ where } n_u \textit{ is the length of output } y \\
R &- \textit{Positive semi} - \textit{definite matrix it size is } n_y \times n_y \textit{ where } n_y \textit{ is the length of input } u \\
e_y(k+i) &- \textit{Error between the reference values and plant outputs at the } i^{th} \textit{ prediction horizon step} \\
\Delta u(k+i) &- \textit{Diffrence of two consequent manipulated variables or input values} \\
Z_k &- \textit{QP decision, given by :}
\end{aligned}
$$

$$
Z_k^T = \begin{bmatrix} u(k/k)^T & u(k+1/k)^T & ... & u(k+p-1/k)^T \end{bmatrix}
$$

The cost function's first term minimizes the errors between the predicted output and the reference value, while the second term minimizes the input U. Instead of just considering the cost of the current step, a predictive controller needs to calculate the cost of future inputs, or in other words the predicted cost. A predicted cost at any time(k) is a sum of individual cost contributions from the time (k) up to the end of the prediction horizon, that is $(k+p)$ [39, 40].

4.2.2 Linear model predictive control with feedback linearization (FBL)

The main idea behind feedback linearization based control is to linearize a nonlinear system by canceling its nonlinear dynamics with appropriate input signal selection. As a result, a linear system could be established that exactly maps the nonlinear dynamics, and linear control techniques could be used. This is accomplished by using nonlinear feedback with the new input for the linear model and diffeomorphism to apply the new state vector [41].
FBL strategy is illustrated in schematic way in Figure 4.4. In the figure $\varphi(x)$ denotes the nonlinear feedback and $h(z, v)$ denotes the diffeomorphism. Due to the fact that the non-

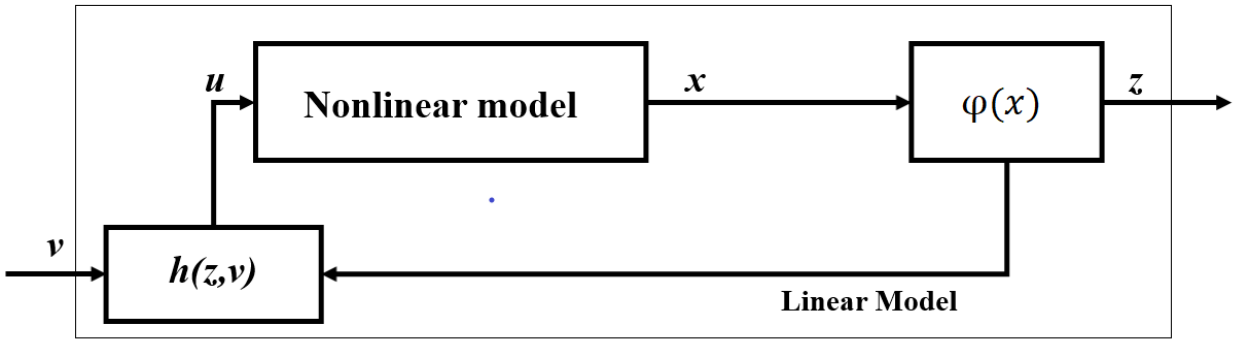


Figure 4.4: Feedback Linearization scheme [3]

linear system description could be restated as a linear system motivates the use of feedback linearization in model predictive control. Despite the linear input constraints becoming nonlinear, the computational load is significantly decreased. The fundamental drawback of feedback linearization is the lack of robustness that might result from model mismatch.
Key assumption for feedback linearization is that the quadcopter dynamic model is known, and all states necessary for negative feedback computation are observable at all times.

Feedback linearization of the Plant model

The dynamics of nonlinear systems could be described as follows:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{4.38}$$

In designing FBL controller the first thing is checking whether the equation is in the form that can guarantee existence of feedback linearizing controller. The system must have the representation which is described in equation (4.39), in order to guarantee state linearizablity.

$$\dot{x} = Ax + B\gamma(x)(u - \alpha(x)) \tag{4.39}$$

Where, A,B is controllable.
Otherwise, if the nonlinear dynamic equation is not in the form stated in equation (4.39) determine a diffeomorphism function to transfer the state representation of the system to the form that can be used for state linearization.
However, if the nonlinear dynamic equation is in the form stated in equation (4.39), then the control signal for linearizing the nonlinear dynamics will be

$$u = \alpha(x) + \beta(x)v \tag{4.40}$$

Where,

$\beta(x) = \gamma^{-1}(x)$
v - Vertual input of the linearized system

However, input state linearization may result in nonlinear output. To check for input to output linearizability, differentiate the output, y, until the derivative of y is dependent of control signal, u.

$$\dot{y} = \frac{\partial(h)}{\partial(x)}[f(x) + g(x)u] = L_f h(x) + L_g h(x)u \tag{4.41}$$

Where,
$L_f h(x)$ is the Lie derivative of $h(x)$ with respect to$f(x)$

$$L_f h(x) = \frac{\partial h}{\partial x} f(x) \tag{4.42}$$

$L_g h(x)$ is the Lie derivative of$h(x)$ with respect to $g(x)$

$$L_g h(x) = \frac{\partial h}{\partial x} g(x) \tag{4.43}$$

If the second term of equation (4.41),$L_g h(x)$, is zero then the derivative of y is independent of control signal. Continue the differentiation until thei^{th} derivative is a function of u.

$$y^{(p)} = L_f^i h(x) + L_g^{i-1} h(x)u \tag{4.44}$$

Then the control signal, u become

$$u = \frac{1}{L_f L_g^{i-1} h(x)} [L_f^i h(x) + v] \tag{4.45}$$

$$y^{(i)} = v \tag{4.46}$$

By following the above procedures the transformed linear dynamic equation for quadcopter attitude subsystem has been determined. And it is stated in Equation(4.47).

- The equivalent linear system became

$$\begin{aligned} \ddot{\phi} &= v_1 \\ \ddot{\theta} &= v_2 \\ \ddot{\psi} &= v_3 \end{aligned}$$

$$\begin{aligned} Z_1 &= \phi \\ Z_2 &= \theta \\ Z_3 &= \psi \\ Z_4 &= \dot{\phi} \\ Z_5 &= \dot{\theta} \\ Z_6 &= \dot{\psi} \end{aligned}$$

$$\begin{bmatrix} \dot{Z}_1 \\ \dot{Z}_2 \\ \dot{Z}_3 \\ \dot{Z}_4 \\ \dot{Z}_5 \\ \dot{Z}_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{4.47}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \end{bmatrix}$$

$$v = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$$

Where Z and v are the linearized system state and input vectors respectively.

Linear Model predictive control for feedback linearized system

The linear control law,v, is designed by using LMPC technique. In order to design LMPC controller first the feedback linearized system, which stated in equation(4.47), must be discretized , because MPC is discrete time controller, and augmented. Then, model predictive control is applied. The overall control structure is shown in the following figure.

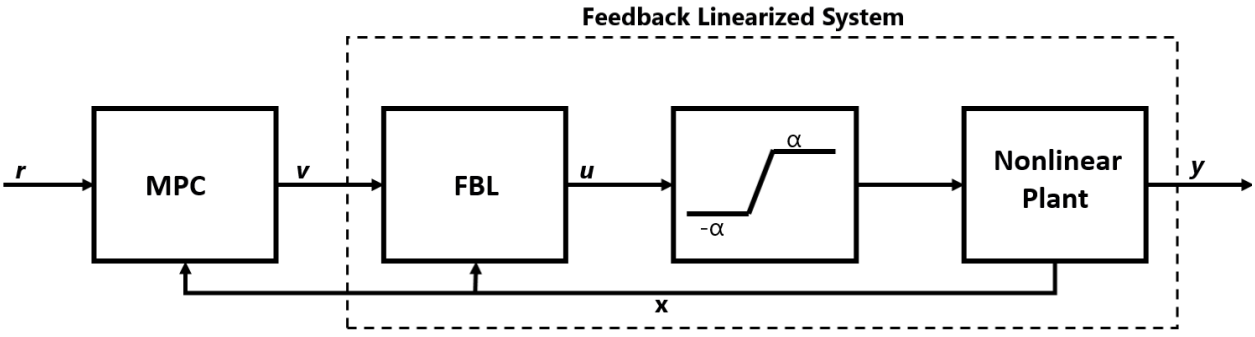


Figure 4.5: LMPC with FBL control structure [4]

Constraints

In such approaches, dealing with input constraints is so tough. Because of the nonlinear transformation, feedback linearization, straightforward bound restrictions on the input

become state dependent. As a result, this mapping converts the process's original input constraints into nonlinear and state-dependent constraints that cannot be solved using quadratic programming.
Therefore, the optimization process is not directly influenced by input restrictions on u. As a result, optimal v is employed in each step, but if it violates restrictions on u, u is chopped off.

Optimization

The control strategy is based on minimization of the performance index, objective function. That means model predictive control is applied through the minimization of the objective function.

$$J_z(h_k) = \sum_{i=0}^{p-1} [e_z^T(k+i)Qe_z(k+i)] + [\Delta v^T(k+i)R\Delta v(k+i)] \tag{4.48}$$

Where,

$$\begin{aligned}
Q &- \textit{Positive semidefinite } n_z \times n_z \textit{ matrix where } n_z \textit{ is the length of linearized system output} \\
R &- \textit{Positive semidefinite } n_v \times n_v \textit{ matrix where } n_v \textit{ is the length of the vertual input } v \\
e_z(k+i) &- \textit{error between reference values and plant outputs at the } i^{th} \textit{ prediction horizon step} \\
\Delta v(k+i) &- \textit{diffrence of two consequent manipulated variables or vertualinput values} \\
h_k &- \textit{QP decision, given by :}
\end{aligned}$$

$$h_k^T = \begin{bmatrix} v(k/k)^T & v(k+1/k)^T & ... & v(k+p-1/k)^T \end{bmatrix}$$

Here the utilized weighting matrices obtained by linear approximation of objective function J and J_z. The method for calculating and deriving FBL weights via linear approximation are detailed in [3] and [42].

Input transformation

The virtual control inputs, v, were designed by LMPC technique then this virtual inputs must be converted to the actual input in order to feed to the quadcopter system. Feedback

linearizing control signal, to cancel the nonlinear dynamics, are stated as follows,

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} \frac{v_1 - a_1 x_4 x_6}{b_1} \\ \frac{v_2 - a_3 x_2 x_4}{b_2} \\ \frac{v_3 - a_5 x_2 x_6}{b_3} \end{bmatrix} \tag{4.49}$$

4.2.3 Nonlinear Model predictive control

Nonlinear Model Predictive Control, or NMPC, is a type of model predictive control (MPC) that uses nonlinear system models to anticipate outcomes. Despite the fact that the idea of nonlinear MPC (NMPC) is the same as the linear MPC case, it is widely known that NMPC design is significantly more difficult, due to the nonlinear and non-convex optimization that must be addressed in each sample period in a very short time [43].
Nonlinear MPC, like standard linear MPC, uses a combination of model-based prediction and constrained optimization to determine control actions at each control interval. The following are the main distinctions between LMPC and NMPC:

- The prediction model can be nonlinear and include time-varying parameters.
- The equality and inequality constraints can be nonlinear.
- The cost function to be minimized can be a nonquadratic (linear or nonlinear) function of the decision variables.

Plant Model

The prediction model in nonlinear model predictive control is the real nonlinear system model. The attitude subsystem's nonlinear equation and state space representation were given in equations (3.14) and (3.39).

$$\begin{aligned} \ddot{\phi} &= b_1 U_2 - a_2 \dot{\theta} \omega_r + a_1 \dot{\psi} \dot{\theta} \\ \ddot{\theta} &= b_2 U_3 - a_4 \dot{\phi} \omega_r + a_3 \dot{\phi} \dot{\psi} \\ \ddot{\psi} &= b_3 U_4 + a_5 \dot{\theta} \dot{\phi} \end{aligned}$$

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{bmatrix} = \begin{bmatrix} x_2 \\ x_4 x_6 a_1 \ - \ x_4 \omega_r a_2 \ + \ b_1 U_2 \\ x_4 \\ x_2 x_6 a_3 \ + \ x_2 \omega_r a_4 \ + \ b_2 U_3 \\ x_6 \\ x_2 x_4 a_5 \ + \ b_3 U_4 \end{bmatrix}$$

The state space model for a nonlinear MPC controller must be discretized because it is a discrete-time controller. The implicit trapezoidal rule is used to discretize the model. This approach can handle fairly stiff models, and the accuracy of its predictions is determined by the controller sample time; a long sample time can lead to erroneous predictions [37].
The following is the general form of a discrete-time prediction model:

$$\begin{aligned} x(k+1) \ &= \ f(x(k), u(k)) \\ y(k) \ &= \ g(x(k)) \end{aligned} \tag{4.50}$$

The state function($f(x(k), u(k))$ and the output function $g(x(k))$ are two functions in the discrete-time prediction model. The prediction model's output function connects states and inputs to the outputs at the current control interval.

Cost Function

The cost function used is a standard quadratic cost function that can be used for reference tracking and disturbance rejection. The standard cost function is made up of two terms, each of which focuses on a different element of controller performance:

$$J(z_k) = J_y(z_k) + J_{\Delta u}(z_k) \tag{4.51}$$

Where,

$$\begin{aligned} k &- \textit{Current control interval} \\ z_k &- \textit{is the QP decision given by;} \\ z_k^T &= \begin{bmatrix} u(k/k)^T & u(k+1/k)^T & ... & u(k+p-1/k)^T \end{bmatrix} \\ J_y(z_k) &- \textit{stands for output reference tracking.} \\ J_{\Delta u}(z_k) &- \textit{stands for manipulated variable move suppression} \end{aligned}$$

Each term has weights that assist to balance competing objectives, as detailed below.
For output reference tracking, the controller utilizes the scalar performance measure, $J_y(z_k)$, which is described below:

$$J_y(z_k) = \sum_{j=1}^{n_y} \sum_{i=1}^{p} \{ \frac{w_{i,j}^y}{s_j^y} [r_j(k+i/k) - y_j(k+i/k)] \}^2 \tag{4.52}$$

Where,

$$\begin{aligned} p &= \textit{Prediction horizon} \\ n_y &= \textit{Number of plant output variables} \\ y_j(k+i|k) &- \textit{Predicted value of } j^{th} \textit{ plant output at } i^{th} \textit{ prediction horizon step} \\ r_j(k+i|k) &- \textit{Reference value for } j^{th} \textit{ plant output at } i^{th} \textit{ prediction horizon step} \\ s_j^y &- \textit{Scale factor for } j^{th} \textit{ plant output} \\ w_{i,j}^y &- \textit{Tuning weight for } j^{th} \textit{ plant output at } i^{th} \textit{ prediction horizon step} \end{aligned}$$

For control input move suppression, the cost function utilizes the scalar performance measure, $J_{\delta u}(z_k)$.

$$J_{\Delta u}(z_k) = \sum_{j=1}^{n_u} \sum_{i=0}^{p-1} \{ \frac{w_{i,j}^{\Delta u}}{s_j^u} [u_j(k+i/k) - u_j(k+i-1/k)] \}^2 \tag{4.53}$$

Where, n_u is control horizon.

Constraints

Input and output constraints were computed, when designing the linear MPC.

Input Constraints

$$U^{max} = \begin{bmatrix} 1.257 & 1.257 & 0.2145 \end{bmatrix}$$

$$U^{min} = \begin{bmatrix} -1.257 & -1.257 & -0.2145 \end{bmatrix}$$

Output Constraints

$$-\pi/2 < \phi < \pi/2$$

$$-\pi/2 < \theta < \pi/2$$

$$-\pi < \psi < \pi$$

4.3 Trajectory Tracking Control

Trajectory tracking control of the quadcopter needs simultaneous control of both attitude and position. Attitude control has been covered in the previous section. In this section position control is then discussed.

4.3.1 Position control

Commanding accelerations in the x-y plane is related to commanding pitch and roll angle, since tilting the quadcopter in any direction causes a component of the thrust vector to point in that direction. The $x-y$ position controller commands the inner loop, attitude controller, by generating reference pitch and roll angle for the quadcopter to track the $x-y$ position command. The altitude (z position) controller generate lifting force or the thrust force. $x-y$ and z position control were implemented using conventional proportional–integral–derivative (PID) controllers. The position control structure is shown in the following figure.

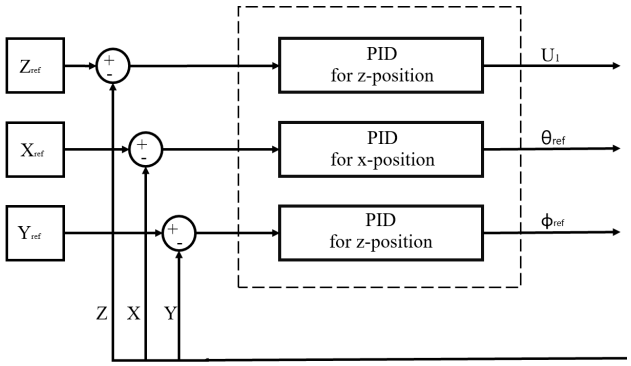


Figure 4.6: Position control scheme.

Altitude control (z-position control)

A nested loop of two discrete PID controllers is utilized to control the z position or altitude. The altitude of the quadcopter is controlled by a discrete P controller, and the rate of change of altitude or the velocity of the quadrotor in the z-axis is also controlled by a discrete P controller. The z velocity P controller uses the control signal from the z position P controller as a reference signal. The feed forward term $(m*g)$ is calculated to generate the force required to perfectly cancel the quadcopter's weight. In order to vary the altitude, the PD control only needs to shift the trust little up and down.

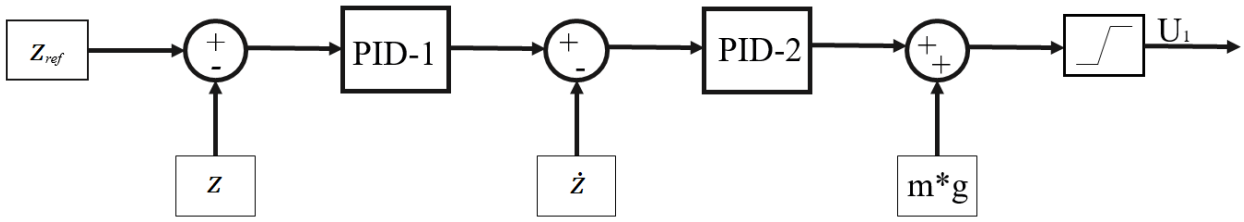


Figure 4.7: Altitude Control Structure

x-y Position Control

Using a nested loop of two distinct PID controllers, x and y position control is achieved. There is a conversion formula between inertial reference frame and body reference frame. The reason is that the x-y position error is relative to the ground or the inertial reference frame. Where as roll and pitch are relative to the body of the quadcopter. Therefore pitch does not always move the quadcopter in the x-direction and roll does not always move the quadcopter in the y-direction, it depends on how the quadcopter is rotated or its yaw angle. If there is a need to move the quadcopter to a very specific x-y direction then need to know yaw angle in order to determine whether roll, pitch or some combination of the two is needed to achieve that. So the x-y position controller uses yaw angle to convert between the inertial x-y reference frame and the body x-y reference frame [44].

$$\begin{bmatrix} x^B \\ y^B \end{bmatrix} = \begin{bmatrix} sin(\psi) & cos(\psi) \\ cos(\psi) & -sin(\psi) \end{bmatrix} \tag{4.54}$$

y-Position Controller

The control signal of y position PID controller becomes the reference signal for roll angle PID controller. There is a gain block of -1 value, due to opposite relation between y distance and roll angle i.e. if roll angle is positive, y distance increase in negative. Otherwise, if roll angle is negative, y distance increase in positive.

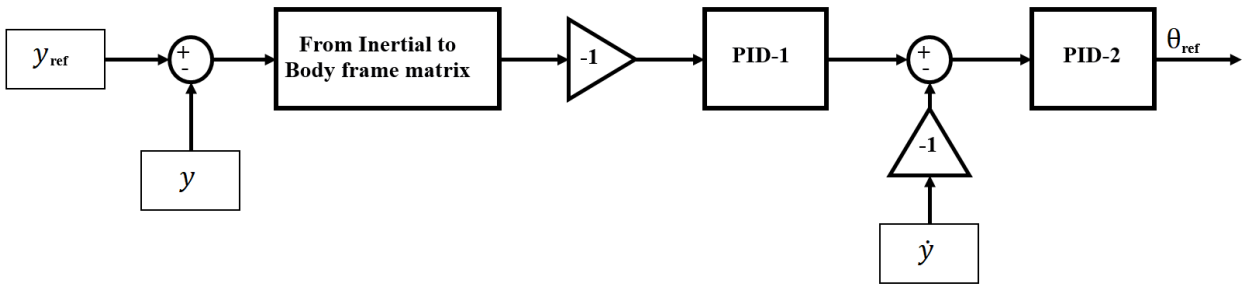


Figure 4.8: y-Position Controller

x-Position Controller

The control signal of x distance PID controller becomes the reference signal for pitch angle PID controller.

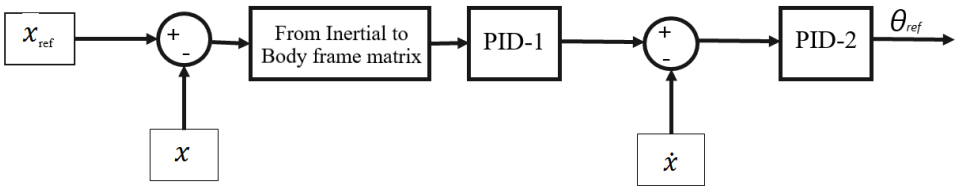


Figure 4.9: x-Position Controller

Summery

The system dynamic model has been divided in to two subsystems, the position and angular subsystem. The position subsystem is controlled by a conventional PID approach, which calculates the control input U_1, desired roll angle (ϕ_{ref}), and desired pitch angle (θ_{ref}) required for the quadcopter to attain its goal position. Desired roll (ϕ_{ref}) and pitch (θ_{ref}) angles , as well as the desired yaw angle (ψ_{ref}), are then provided as reference values to the Inner loop controller. Three variant MPC strategies were designed for attitude control. These tree strategies are:

1. Linear MPC (LMPC)
2. LMPC in combination with feedback linearization (FBL)
3. Nonlinear MPC (NMPC)

LMPC utilizes the linear plant model, which is linearized at equilibrium points, as a prediction model, however it is applied on nonlinear plant model. In the later approaches, LMPC+FBL and NMPC, the nonlinear predictive control approached in indirect and direct way respectively. The indirect way was based on Feedback Linearization (FBL) as the method which gives exact linear model by employing virtual state and input signal. The obtained linear model was used as a prediction model and then LMPC applied. The direct way employs exact NMPC which utilizes the nonlinear prediction model. NMPC generally result in non-convex nonlinear programs, even if the cost function is quadratic and constraint are linear sets.

Chapter 5

Simulation Results and Discussion

This chapter presents simulation results of the proposed control methods. It is divided in to two section where the first section illustrate the simulation result of each controller designed for attitude subsystem and the second section discusses the simulation result of the full dynamic system, by combining the position and attitude subsystem along with the proposed controllers.
The modeling of the quadcopter and the controllers design have been done in the MATLAB/Simulink environment. Separate simulations were carried out for attitude control and trajectory tracking performance comparison. The complete Simulink model including the path commands, position controller, attitude controller and the quadcopter model is shown in Appendix.

5.1 Attitude Control

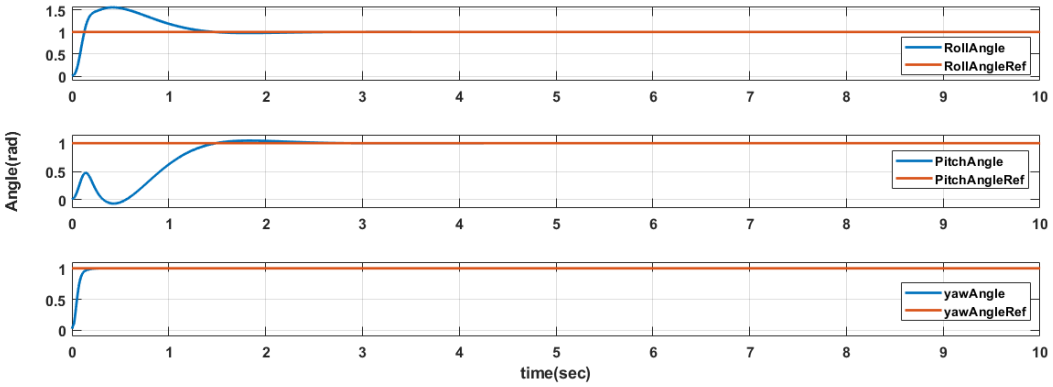
In order to control the orientation of the quadcopter, three distinct attitude controllers have been implemented and the results were analyzed. Generally each controller has been designed based on the best possible trade-off among the prefer performance indices. Sinusoidal signals and unit step signals were used as the references to be tracked by the controllers. Their performance were observed by plotting the model output (response) together with the reference signals. Moreover, the controllers performance were tested under wind disturbance.

5.1.1 Controllers Performance Under Normal Condition

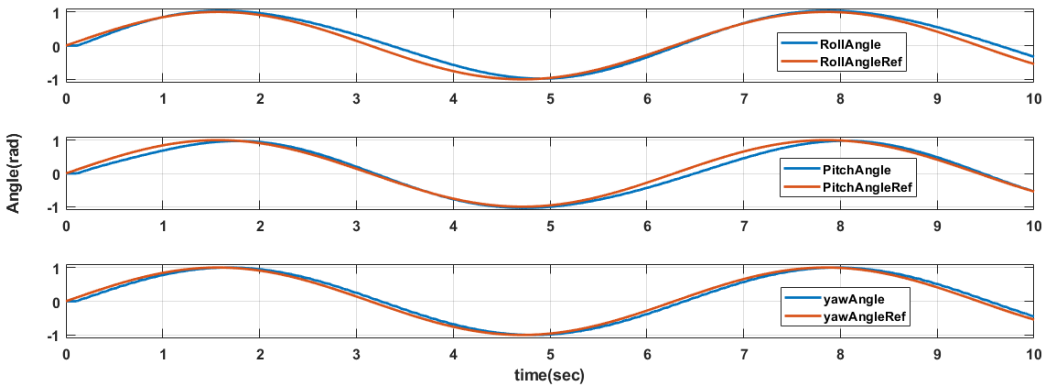
A) Linear Model Predictive Controller

The LMPC attitude controller designed in the previous chapter was simulated in MATLAB to test its performance under normal condition i.e. without disturbance. In order to achieve

the best performance, the prediction horizon, N=20, control horizon, M=3, and sampling time, Ts= 0.1 along with the weight matrices that have been chosen based on trial and error have been utilized. The weights for output roll angle, pitch angle, and yaw angle were 1; 1; 1, weights on increments of the control inputs were 0.1; 0.1; 0.1.The following figure, Figure 5.1, shows the responses for unit step signal and sine wave signal.



(a) Unit step response



(b) Sine wave signal response

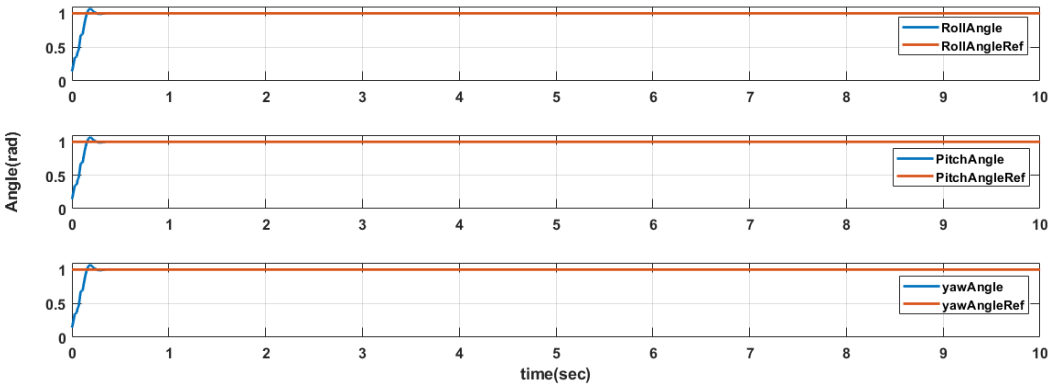
Figure 5.1: Attitude subsystem with LMPC controller

Performance measure	Roll Angle	Pitch Angle	Yaw Angle
Overshoot (%	54.9	11.7	0
Settling time(sec)	1.87	2.33	0.2

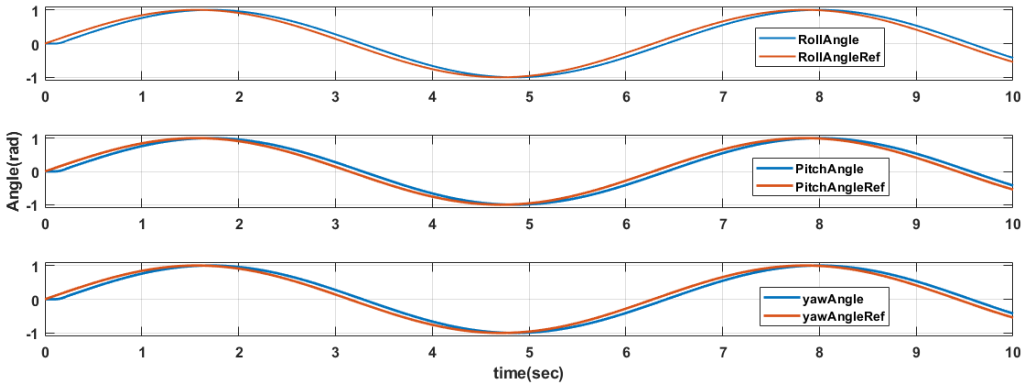
Table 5.1: Transient performance measure of LMPC

B) Linear Model Predictive Controller with Feedback Linearization Controller

Feedback linearization with Model predictive control has been implemented for attitude control. The simulation block layout is shown in Appendix. The following parameters of the LMPC controller were used during the simulations: sampling period 0.1s, prediction horizon 20 , control horizon 3, weights for output roll angle, pitch angle, and yaw angle 1;1;1, weights on increments of the control inputs are 0.1;0.1;0.1. The responses for unit step signal and sine wave signal are shown in Figure 5.2.



(a) Unit step response



(b) Sine wave signal response

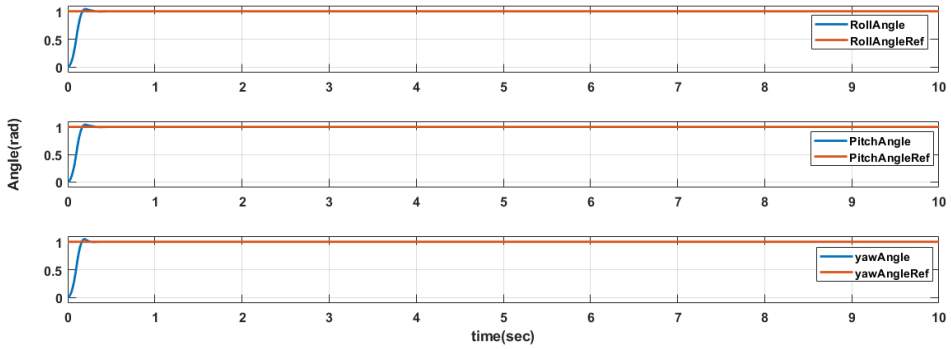
Figure 5.2: Attitude subsystem with FBL+LMPC controller

Performance measure	Roll Angle	Pitch Angle	Yaw Angle
Overshoot (%	7.29	7.29	7.29
Settling time(sec)	0.24	0.24	0.24

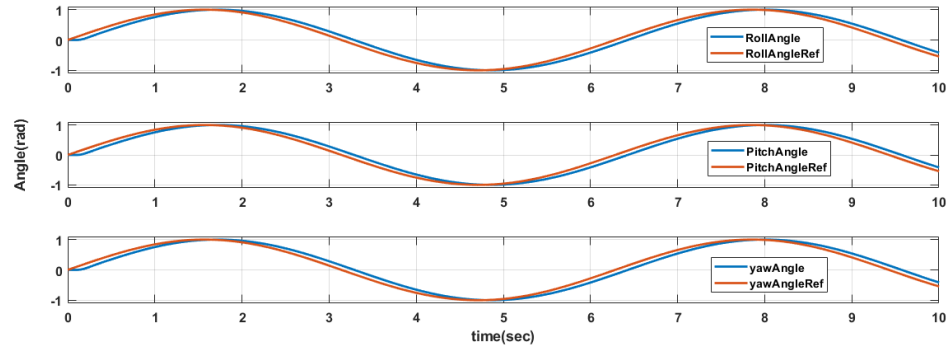
Table 5.2: Transient performance measure of FBL+LMPC

C) Nonlinear Model Predictive Controller

The designed nonlinear model predictive controller (NMPC) has been tested by applying unit step and sine wave signals as a reference, in order to visualize and compare its performance with the other two controllers. The following figure,Figure 5.3, depicts the simulation results. The utilized controller parameters are discussed as follows; sampling period 0.1s, prediction horizon 20 , control horizon 3, weights for output roll angle, pitch angle, and yaw angle 1;1;1, weights on increments of the control inputs are 0.1;0.1;0.1.



(a) Unit step response



(b) Sine wave signal response

Figure 5.3: Attitude subsystem with NMPC controller

Performance measure	Roll Angle	Pitch Angle	Yaw Angle
Overshoot (%	4	4	5
Settling time(sec)	0.26	0.27	0.24

Table 5.3: Transient performance measure of NMPC

5.1.2 Controllers Performance Under Wind Disturbance

Real world quadcopter flights are subjected to external disturbances like wind. The disturbance model which used in this simulation is a Gaussian noise with 0 mean and unity covariance. The disturbance signal is shown in Figure 5.4.

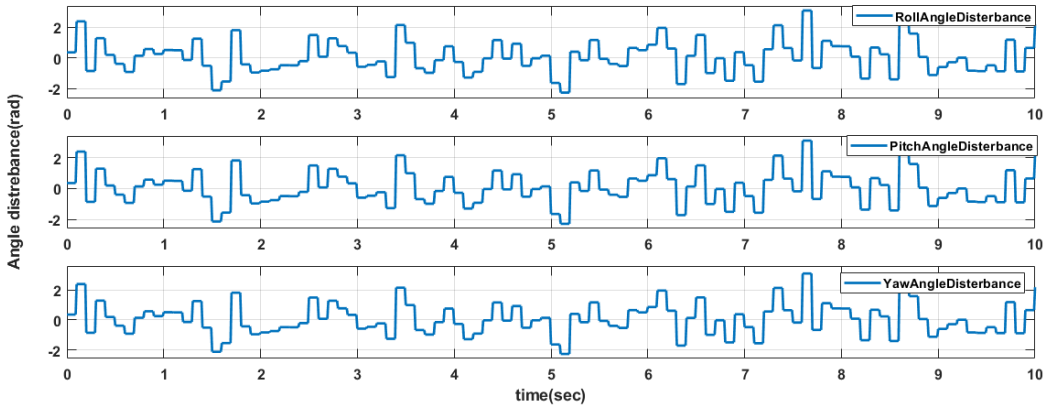


Figure 5.4: Wind Disturbance

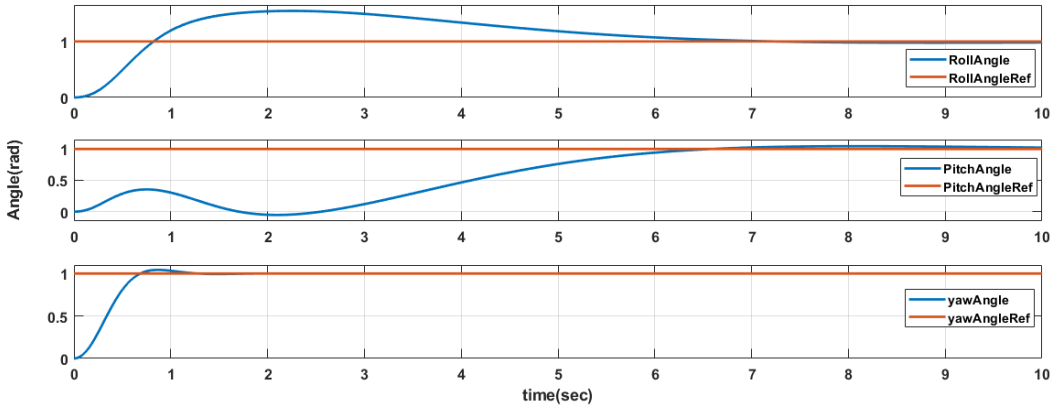


Figure 5.5: LMPC under wind disturbance

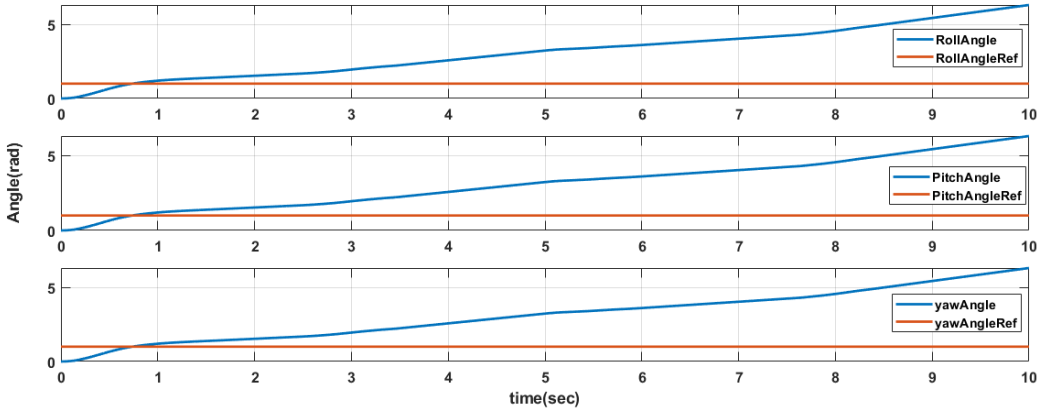


Figure 5.6: FBL+LMPC under wind disturbance

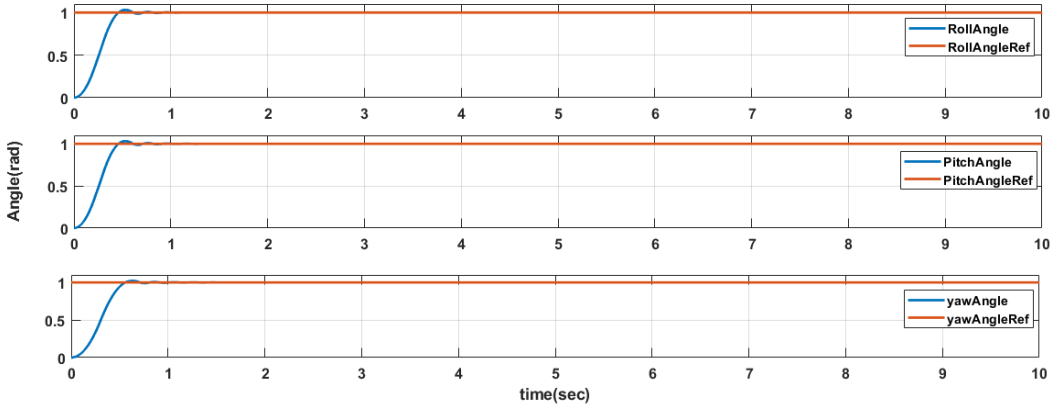


Figure 5.7: NMPC under wind disturbance

5.1.3 Control Effort Comparison

Control effort efficiency is an important issue when implementing the algorithms in real-time application. The power source that drives all the systems on the quadcopter and allows it to fly is the quadcopter battery. Once the battery run out of power, the quadcopter can't going anywhere. Therefore the control effort exerted should be efficient. The Control effort efficiency for each control algorithm is determined by calculating the area under the control input Vs time curve i.e $\int_0^t U_i dt$. The results shows that NMPC requires the the least amount of control effort compared to other controllers. Furthermore, NMPC renders lowest fluctuations at control inputs compared to other controllers which is safe for the system during flight condition.

A) Control Effort when unit step signal is a reference

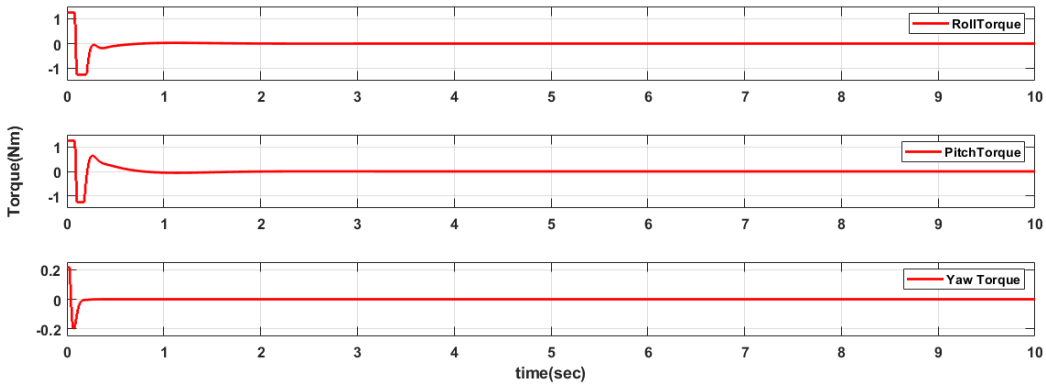


Figure 5.8: LMPC Output Command

Figure 5.9: FBL+LMPC Output Command

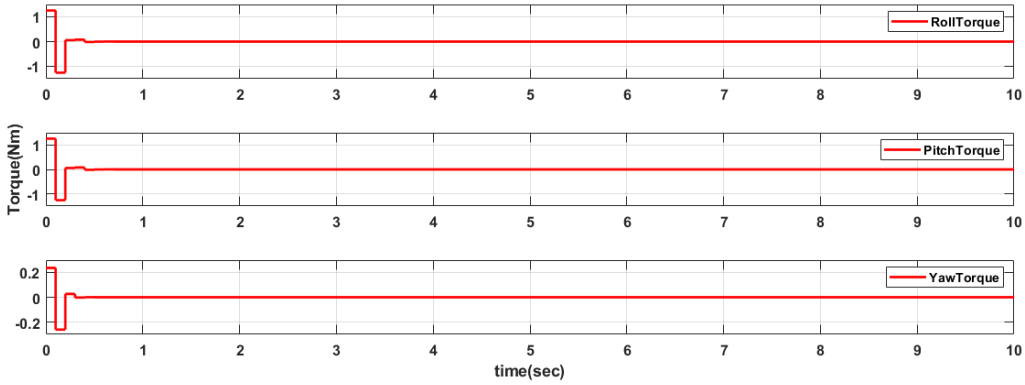


Figure 5.10: NMPC Output Command

	LMPC	LMPC with FBL	NMPC
$\lvert \int U_2 dt \rvert$	0.3071	0.4572	1.827×10^{-13}
$\lvert \int U_3 dt \rvert$	0.3339	0.4572	1.827×10^{-13}
$\lvert \int U_4 dt \rvert$	5.503×10^{-16}	1.73×10^{-14}	1.5×10^{-15}

Table 5.4: Control effort efficiency when unit step signal is reference

B) Control Effort when sine wave signal is a reference

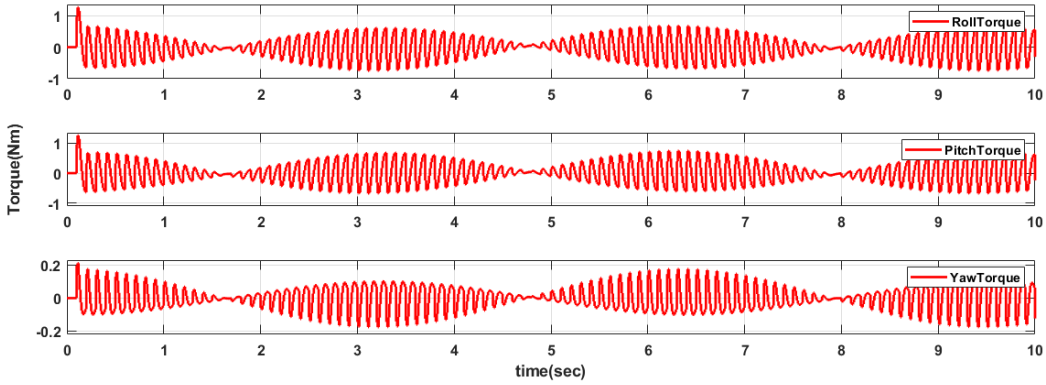


Figure 5.11: LMPC Output Command

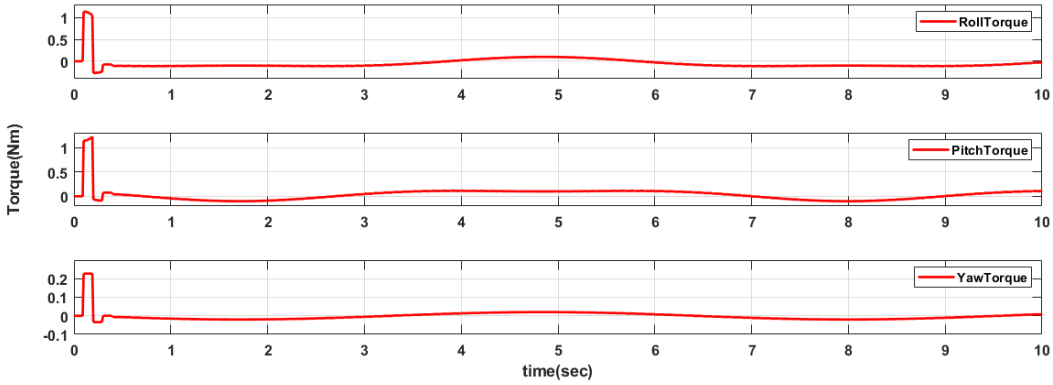


Figure 5.12: FBL+LMPC Output Command

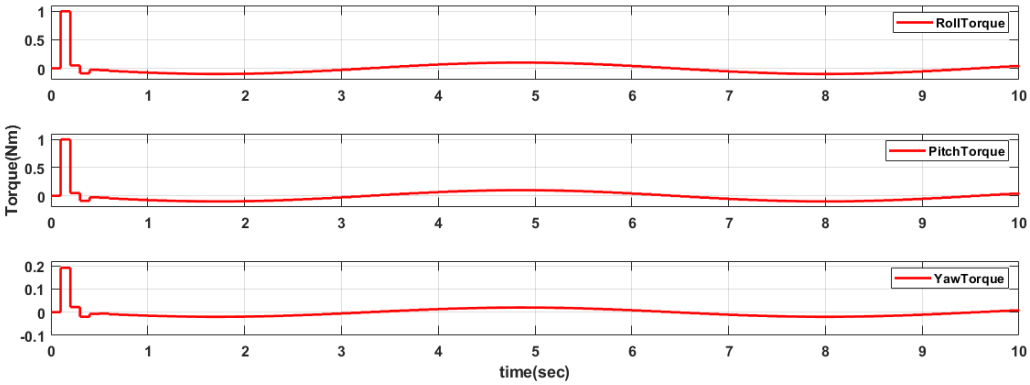


Figure 5.13: NMPC Output Command

	LMPC	LMPC with FBL	NMPC
$\lvert \int U_2 dt \rvert$	1.391	1.304	0.01314
$\lvert \int U_3 dt \rvert$	1.418	1.329	0.01314
$\lvert \int U_4 dt \rvert$	0.001086	0.002483	0.002676

Table 5.5: Control effort efficiency when sine wave signal is reference

Summary

Three variant of MPC controllers were designed for quadcopter attitude subsystem and their performance was tested in simulation. The controllers steady sate and transient state performances were tested and also the controllers tracking performance were tested with and without disturbance. From the simulation that can be concluded that nonlinear model predictive control (NMPC) is the natural choice since the quadcopter system exhibits considerable nonlinearity.

At some extents like offering low design costs, ease of implementation and high tracking efficiency particularly when system is free from disturbance LMPC+FBL performs better than NMPC. However, in presence of disturbance in the system, the performance of the LMPC+FBL is the worst and NMPC is able to show higher tracking efficiency comparing the other two controllers. Therefore, it is quite uncertain to ensure satisfactory tracking performance in all sorts of environments and trajectories from LMPC and LMPC+FBL while NMPC is reliable.

Moreover, based on another performance index, control effort efficiency, significantly NMPC is more efficient than other two controllers as both LMPC and LMPC+FBL requires comparatively more control effort. Thus NMPC was found as suitable method when control requirements are strict and the quadcopter platform is for outdoor application, where disturbance is very high. But with higher implementation and design costs.

5.2 Trajectory Tracking Control

The main goal of this work is to control the attitude subsystem by three different methods based on the concept of MPC, however for completeness and in the real world quadcopters are usually executing missions, trajectory tracking control has been carried out. The purpose of trajectory control is to move the quadcopter from the original location to the desired location. In trajectory tracking control the position and attitude controllers have to be implemented together. Attitude controller simulations were illustrated in the previous section. In order to control the position of the quadcopter, a position controller has been implemented. The position controller consists of three decoupled controllers, x-position controller, y-position controller and z- position controller. Each of them were described in section 4.3. To choose the parameters of the three controllers, they were simulated together with the each attitude controllers in separate simulation using different values of the controller parameters.
The performance of the controllers are evaluated using RMS error (RMSE). Root Mean Square (RMS) is an approach to evaluate the accuracy of the data by comparison.

5.2.1 Linear MPC for attitude control and PID controller for position control

Controller gains

For the x, y and z position PID controller, the values of controller gains are determined by using PID tuner till the desired performance is achieved.

Controllers	PID-1 Gain Values			PID-2 Gain Values		
PID Position Controllers	K_P	K_I	K_D	K_P	K_I	K_D
x-Controller	1	0	0.1	0.2	0	0.1
y-Controller	1	0	0.3	0.1	0	0.2
z-Controller	2.46	0	0.2	16.4	0	0
Attitude Controller	LMPC Parameters					
LMPC	Sampling Time	Prediction Horizon	Control Horizon			
	30	6	0.01			

Table 5.6: Controllers Parameters when PID is position controller and LMPC is attitude controller

Path commands

The path.m file, which performs the trajectory planning, is first run to obtain the path command data. While the desired x, y and $z$commands are obtained as a time series data, the ψ values are given as a constant function. The chosen reference trajectories in order to show the proposed controller tracking performance were circular, helix and infinity shaped trajectories.

a) Circular Trajectory

For circular trajectory, $x = 2sin(0.3t), y = 2cos(0.3t), z = 3m$.

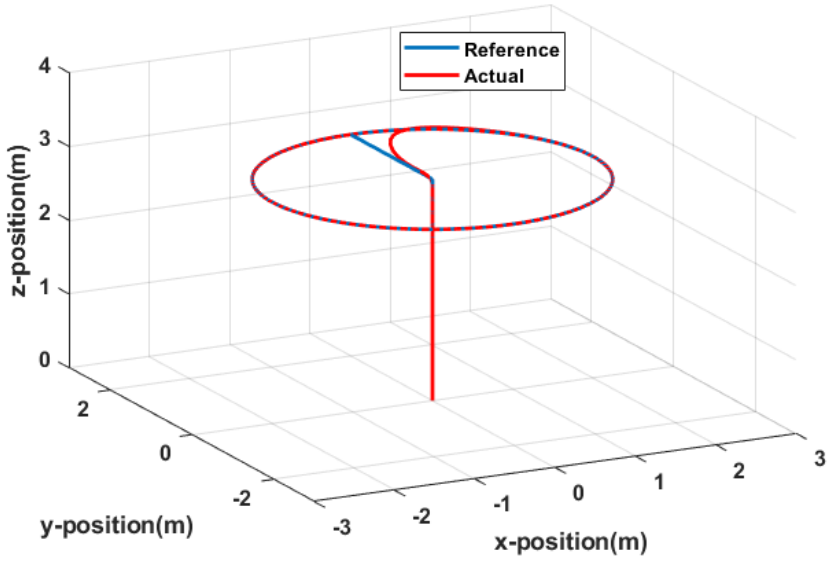


Figure 5.14: 3D plot for circular trajectory tracking

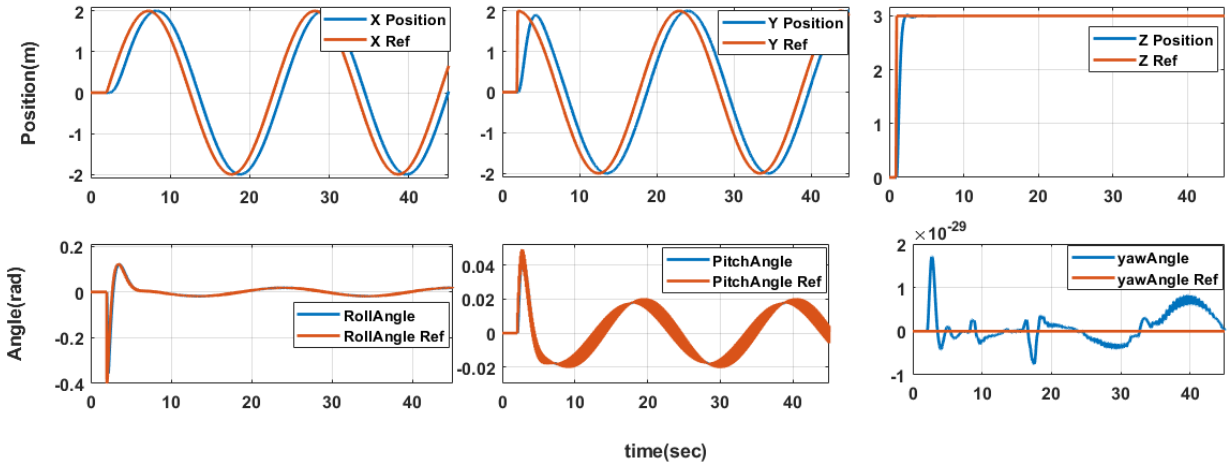


Figure 5.15: Circular Trajectory tracking with PID and LMPC controllers

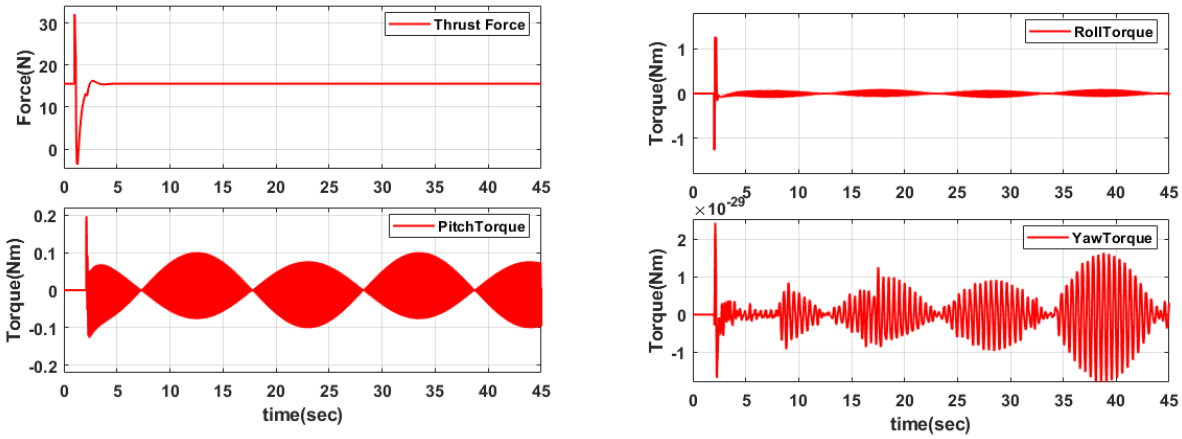


Figure 5.16: Control Inputs

b) Helix Trajectory

For helix trajectory, $X = 2cos(t), Y = 2sin(t), Z = 0.3t$

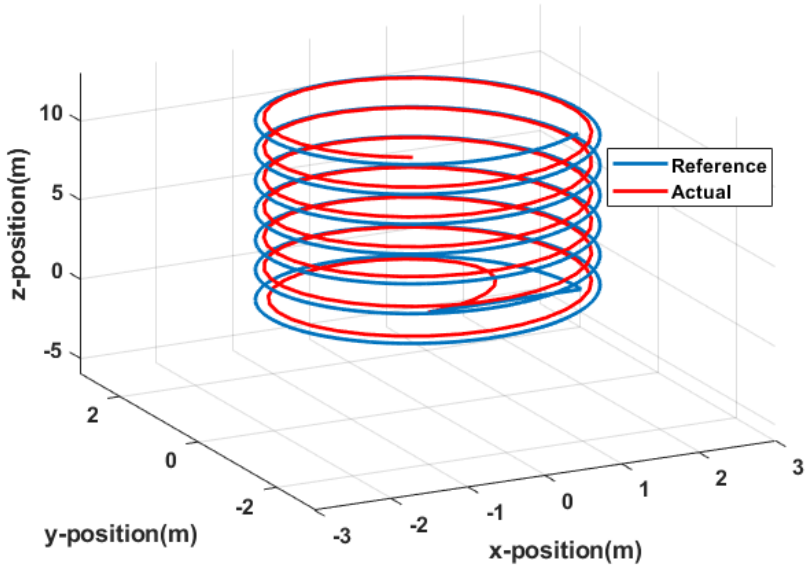


Figure 5.17: 3D plot for helix trajectory tracking

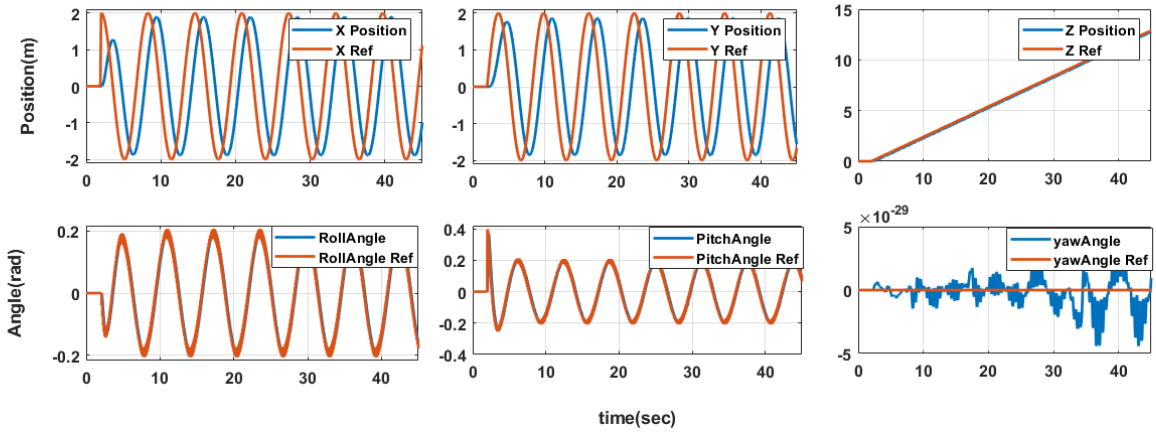


Figure 5.18: Helix Trajectory tracking with PID and LMPC controllers

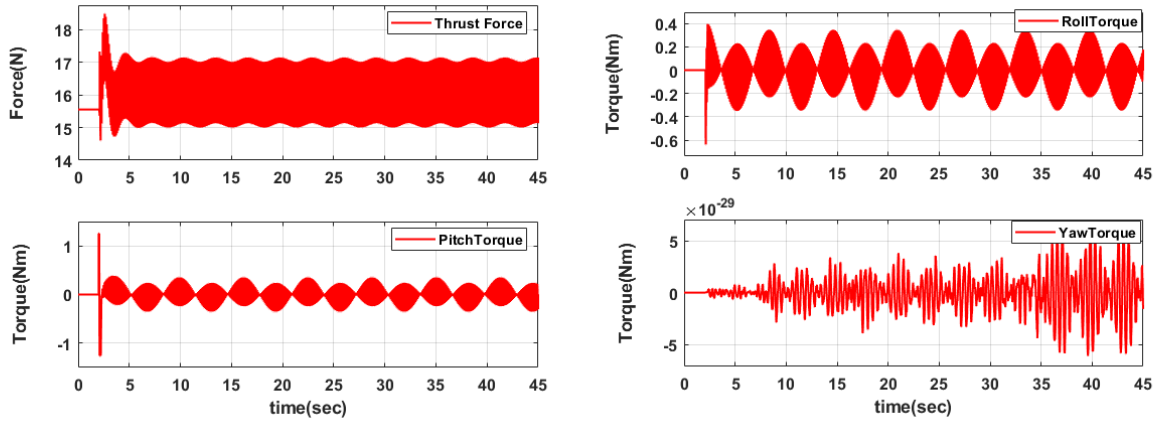


Figure 5.19: Control Inputs

c) Infinity shape Trajectory

For infinity shape trajectory, $X = cos(t), Y = \frac{sin(2t)}{2}, Z = 1m$

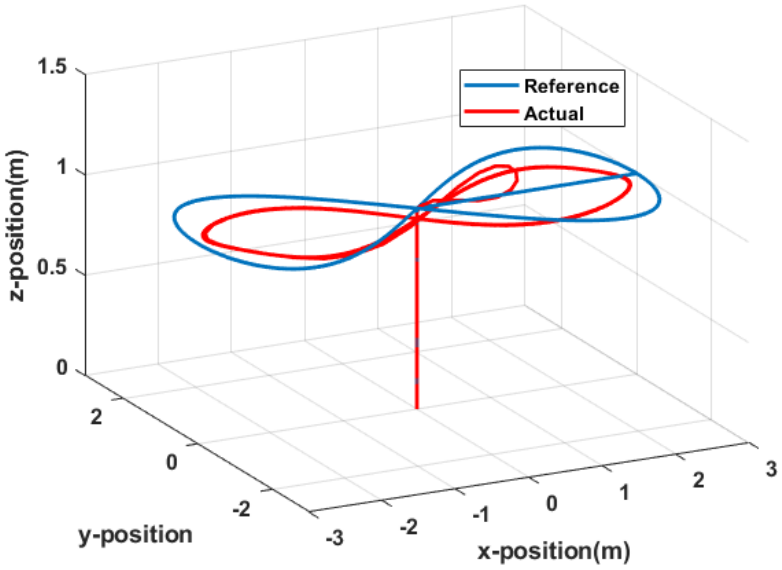


Figure 5.20: 3D plot for infinity shape trajectory tracking

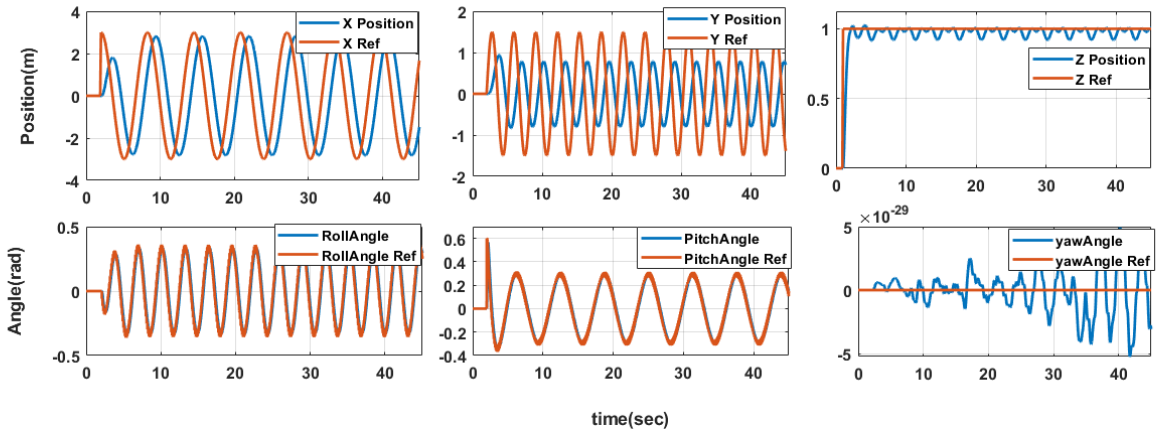


Figure 5.21: Infinity shape Trajectory tracking with PID and LMPC controllers

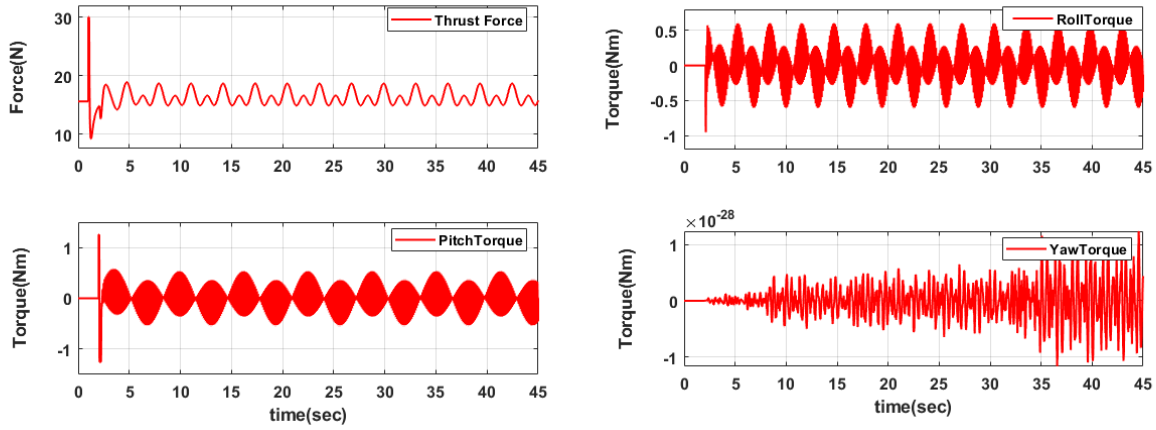


Figure 5.22: Control Inputs

Tracking error RMSE values

Trajectory	RMSE along x-axis (%)	RMSE along y-axis (%)	RMSE along z-axis (%)
Circular Trajectory	0.4164	0.4855	0.2317
Helix Trajectory	1.3963	1.3978	0.1278
Infinity shape Trajectory	2.0951	1.4084	0.0854

Table 5.7: Tracking Error RMSE value when PID is position controller and LMPC is attitude controller

Trajectory	IAE along x-axis (%)	IAE along y-axis (%)	IAE along z-axis (%)
Circular Trajectory	1.6458×10^3	1.8017×10^3	124.1
Helix Trajectory	5.5287×10^3	5.5210×10^3	559.5
Infinity shape Trajectory	8.2972×10^3	5.5628×10^3	155.4

Table 5.8: Tracking Error IAE value when PID is position controller and LMPC is attitude controller

5.2.2 FBL with Linear MPC for attitude control and PID controller for position control

Controllers gain values

Controllers	PID-1 Gain Values			PID-2 Gain Values		
PID Position Controllers	K_P	K_I	K_D	K_P	K_I	K_D
x-Controller	1.8	0	0.1	0.3	0	0.2
y-Controller	1.9	0	0.1	0.4	0	0.1
z-Controller	4.86	0	0	19.6	0	0
Attitude Controller	LMPC Parameters					
FBL+LMPC	Sampling Time	Prediction Horizon	Control Horizon			
	30	6	0.01			

Table 5.9: Controllers Parameters when PID is position controller and FBL+LMPC is attitude controller

Path Commands

a) Circular Trajectory

For circular trajectory, $x = 2sin(0.3t), y = 2cos(0.3t), z = 3m$.

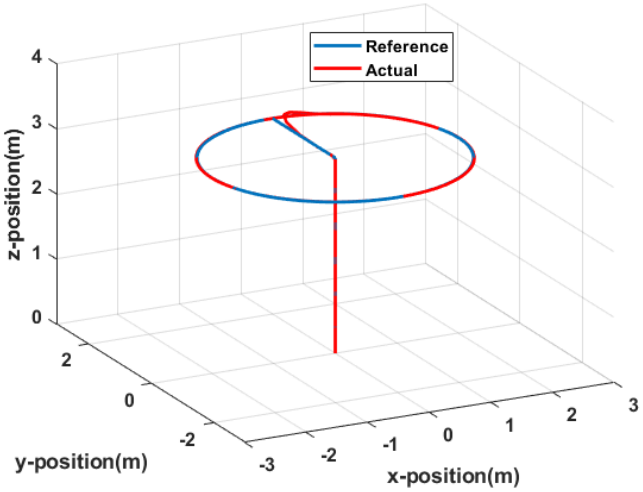


Figure 5.23: 3D plot for circular trajectory tracking

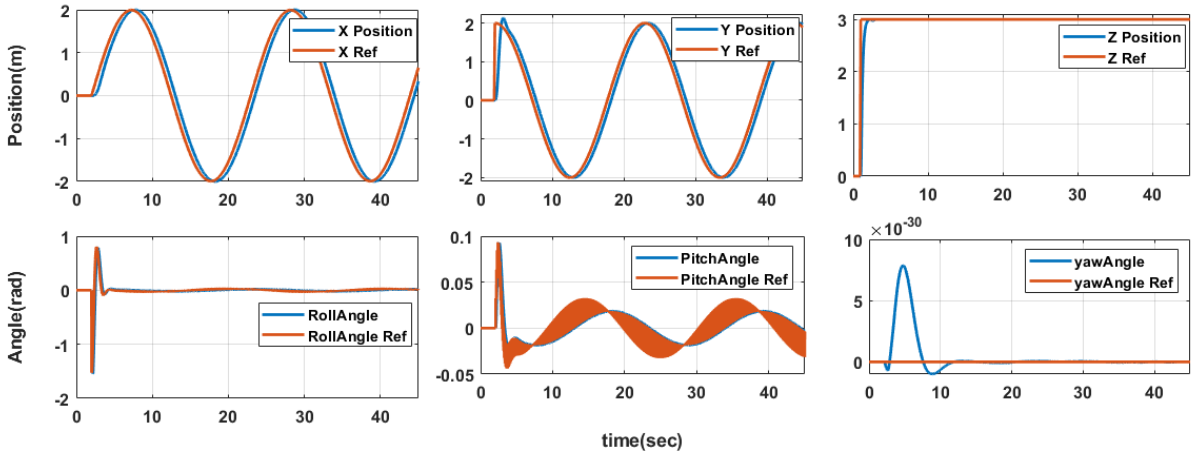


Figure 5.24: Circular Trajectory tracking with PID and FBL+LMPC controllers

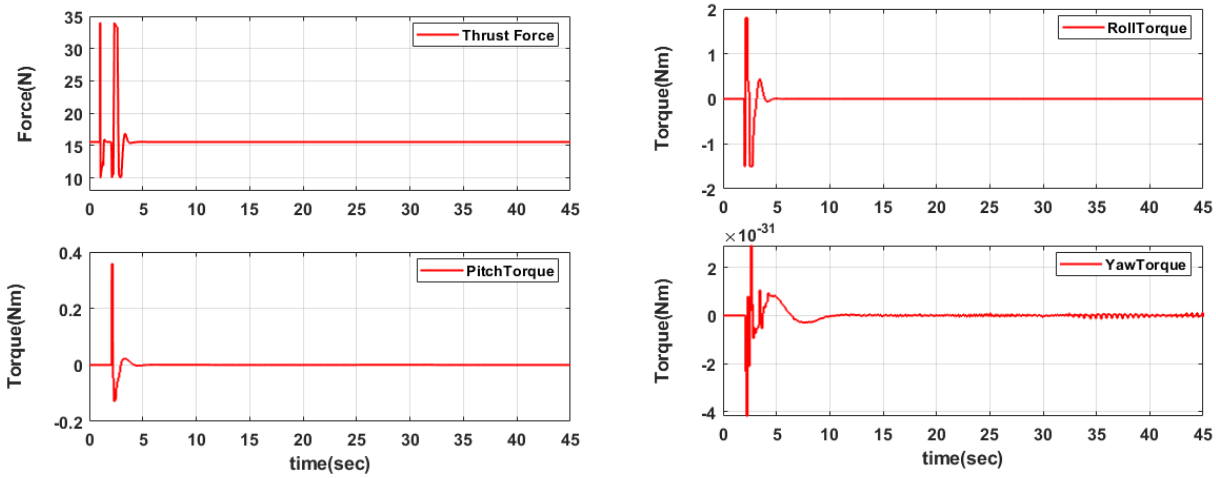


Figure 5.25: Control Inputs

b) Helix Trajectory

For helix trajectory, $x = 2cos(t), y = 2sin(t), z = 0.3t$

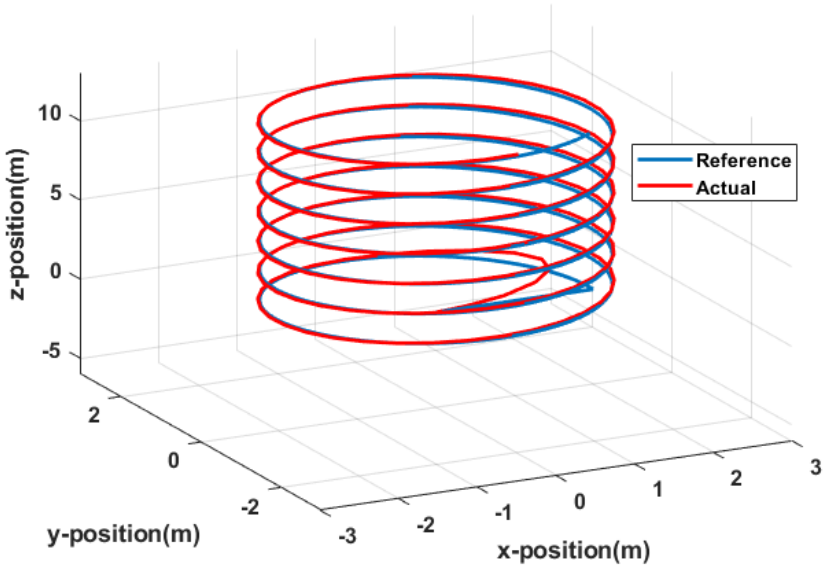


Figure 5.26: 3D plot for helix trajectory tracking

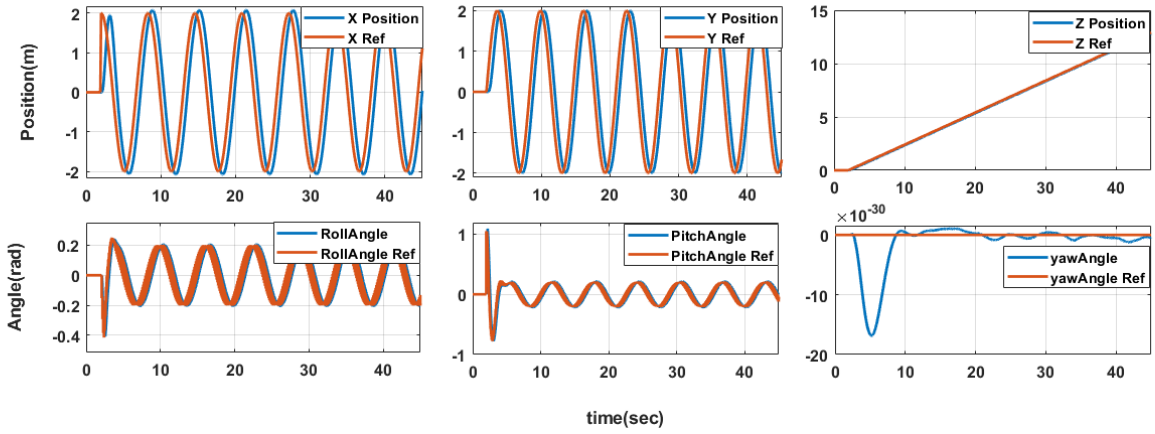


Figure 5.27: Helix Trajectory tracking with PID and FBL+LMPC controllers

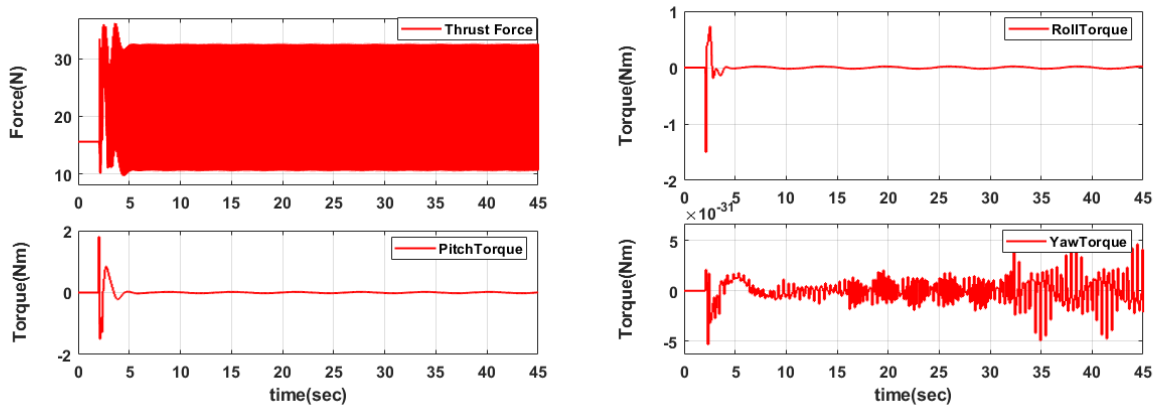


Figure 5.28: Control Inputs

c) Infinity shape Trajectory

For infinity shape trajectory, $x = cos(t), y = \frac{sin(2t)}{2}, z = 1m$

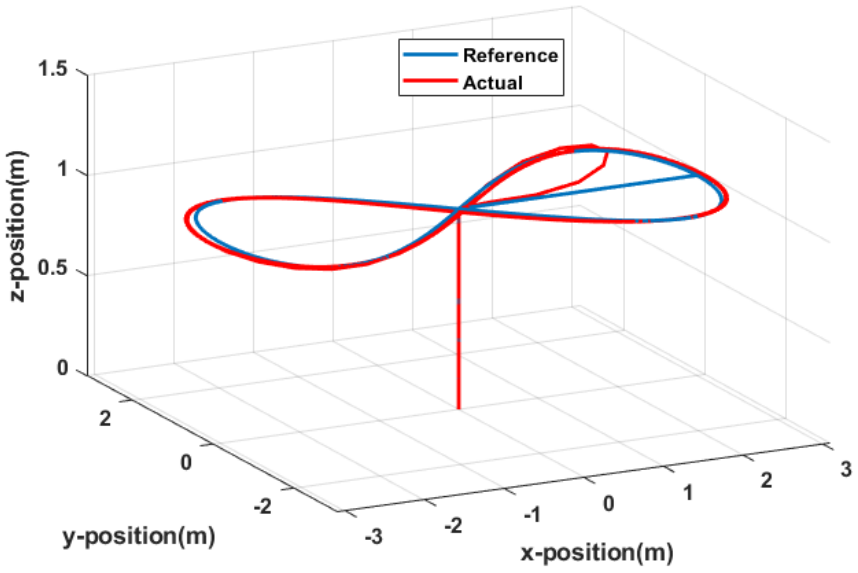


Figure 5.29: 3D plot for infinity shape trajectory tracking

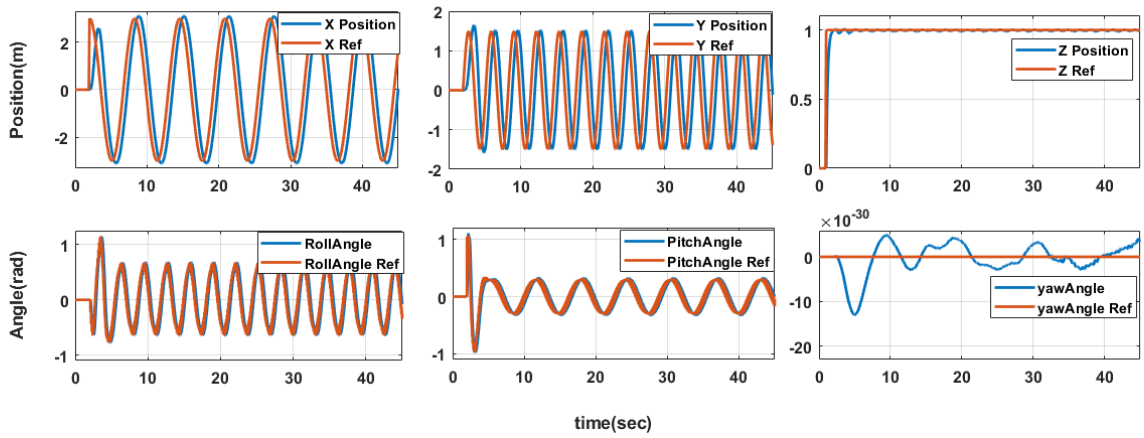


Figure 5.30: Infinity shape Trajectory tracking with PID and FBL+LMPC controllers

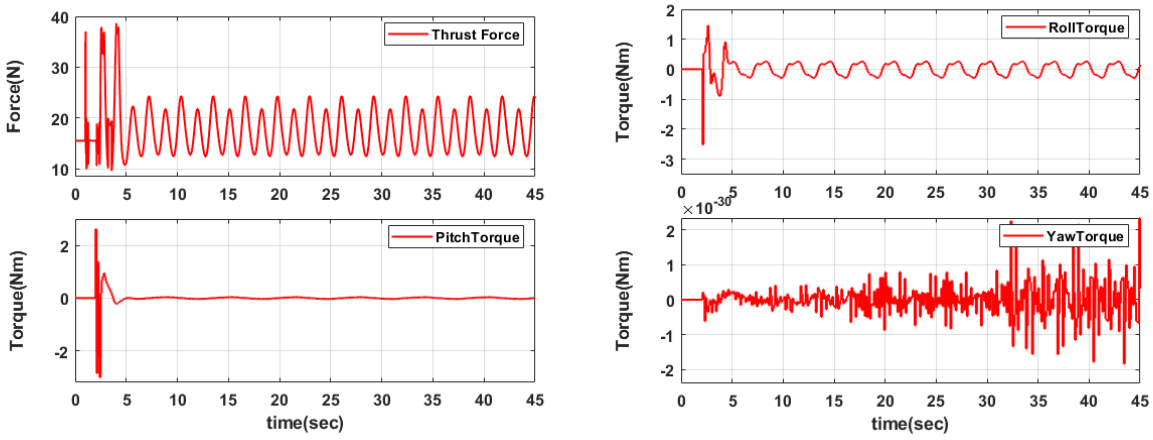


Figure 5.31: Control Inputs

Trajectory tracking error RMSE values

Trajectory	RMSE along x-axis (%)	RMSE along y-axis (%)	RMSE along z-axis (%)
Circular Trajectory	0.2145	0.2755	0.1723
Helix Trajectory	0.7664	0.6605	0.0613
Infinity shape Trajectory	0.3843	0.3245	0.0516

Table 5.10: Tracking Error RMSE value when PID is position controller and FBL+LMPC is attitude controller

Trajectory	IAE along x-axis (%)	IAE along y-axis (%)	IAE along z-axis (%)
Circular Trajectory	847.1750	879.6616	75.0850
Helix Trajectory	3.0068×10^3	2.6038×10^3	266.3
Infinity shape Trajectory	1.5097×10^3	1.2822×10^3	23.5

Table 5.11: Tracking Error IAE value when PID is position controller and FBL+LMPC is attitude controller

5.2.3 NMPC for attitude control and PID controller for position control

Controllers gain values

Controllers	PID-1 Gain Values			PID-2 Gain Values		
PID Position Controllers	K_P	K_I	K_D	K_P	K_I	K_D
x-Controller	2.16	0	0.1	0.505	0	0.1
y-Controller	2.07	0	0.2	0.493	0	0.2
z-Controller	1.6	0.1	0	22	0	0
Attitude Controller	NMPC Parameters					
NMPC	Sampling Time	Prediction Horizon	Control Horizon			
	20	6	0.01			

Table 5.12: Controllers Parameters when PID is position controller and NMPC is attitude controller

Path Commands

a) Circular Trajectory

For circular trajectory, $x = 2sin(0.3t), y = 2cos(0.3t), z = 3m$.

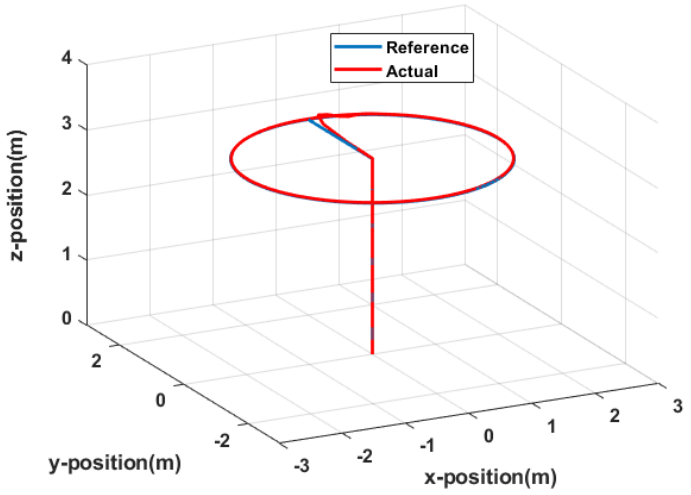


Figure 5.32: 3D plot for circular trajectory tracking

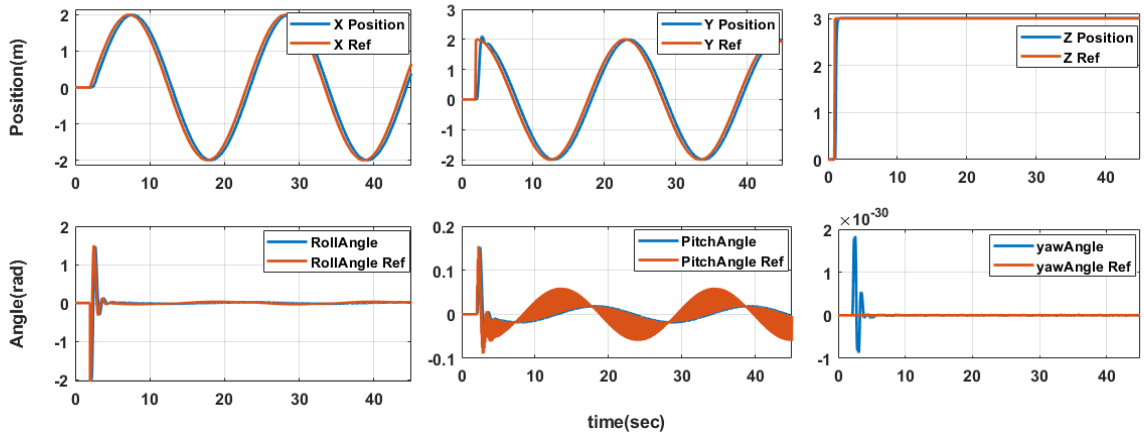


Figure 5.33: Circular Trajectory tracking with PID and NMPC controllers

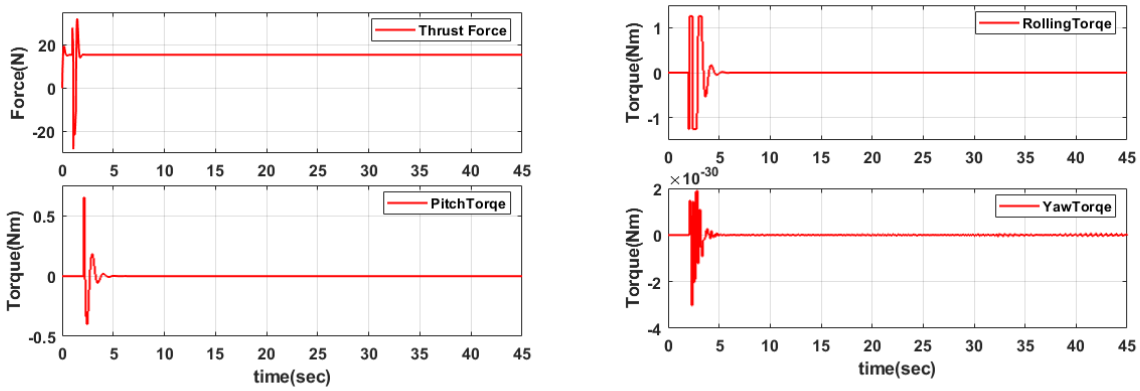


Figure 5.34: Control Inputs

b) Helix Trajectory

For helix trajectory, $x = 2cos(t), y = 2sin(t), z = 0.3t$

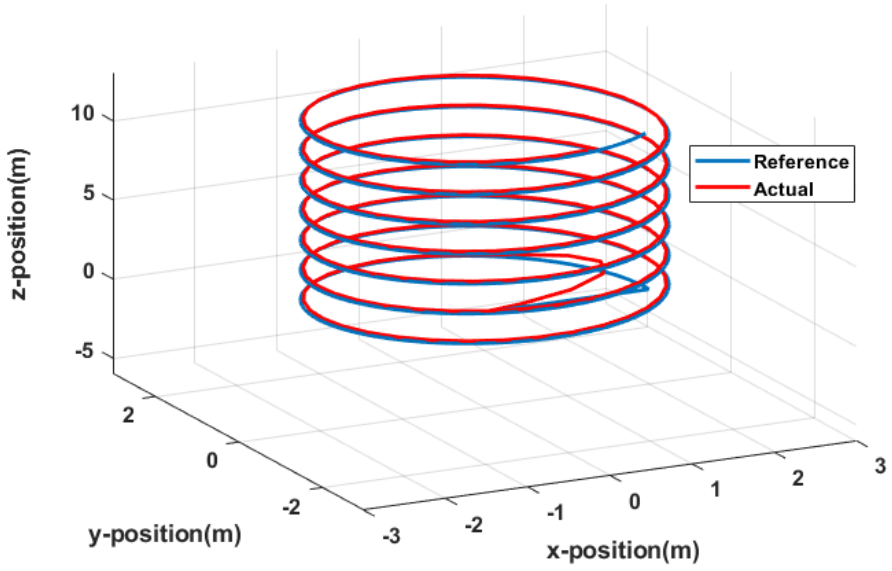


Figure 5.35: 3D plot for helix trajectory tracking

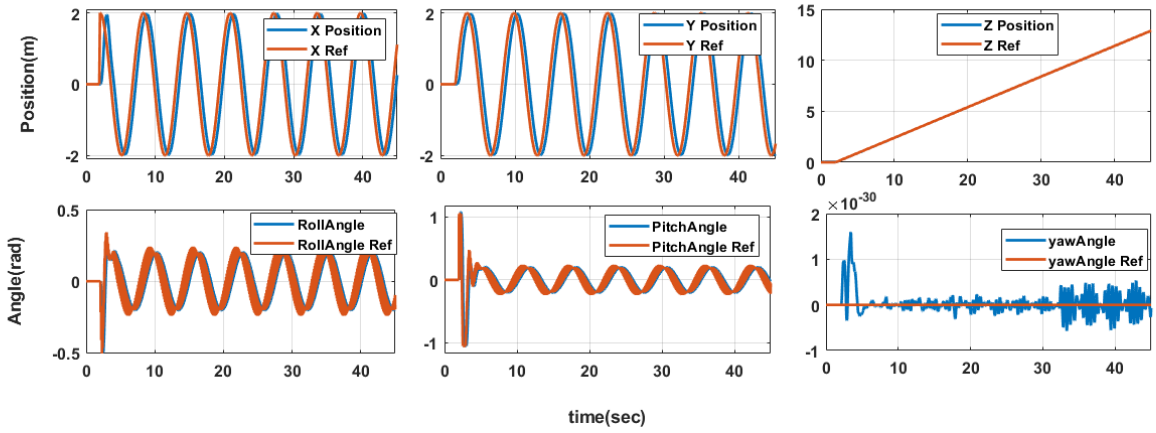


Figure 5.36: Helix Trajectory tracking with PID and NMPC controllers

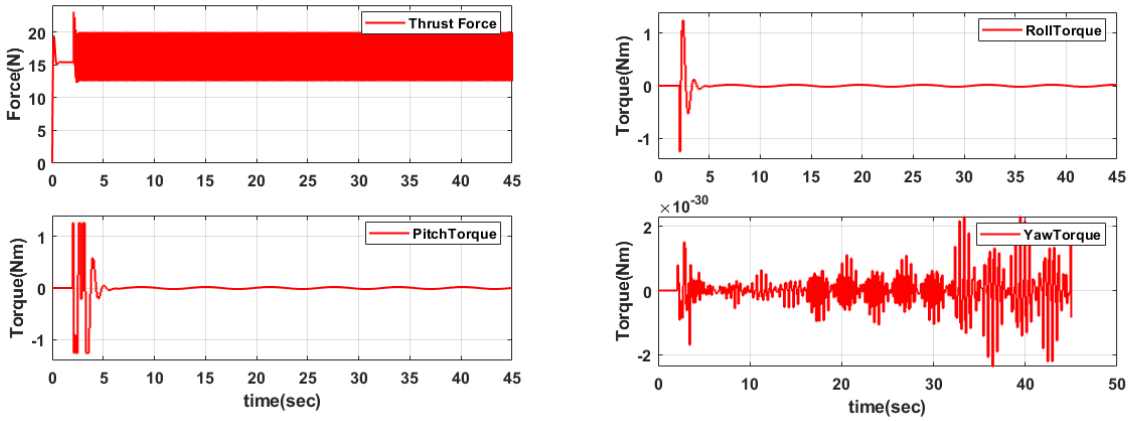


Figure 5.37: Control Inputs

c) Infinity shape Trajectory

For infinity shape trajectory, $x = cos(t), y = \frac{sin(2t)}{2}, z = 1m$

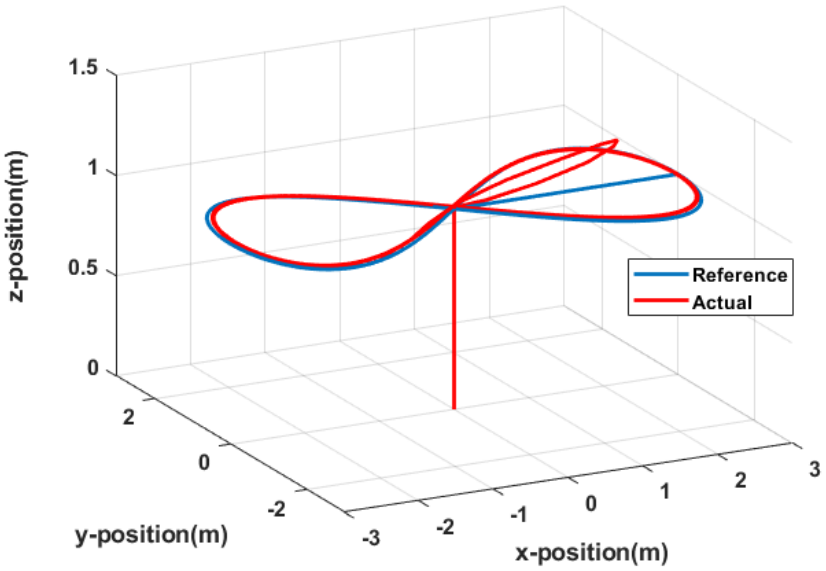


Figure 5.38: 3D plot for infinity shape trajectory tracking

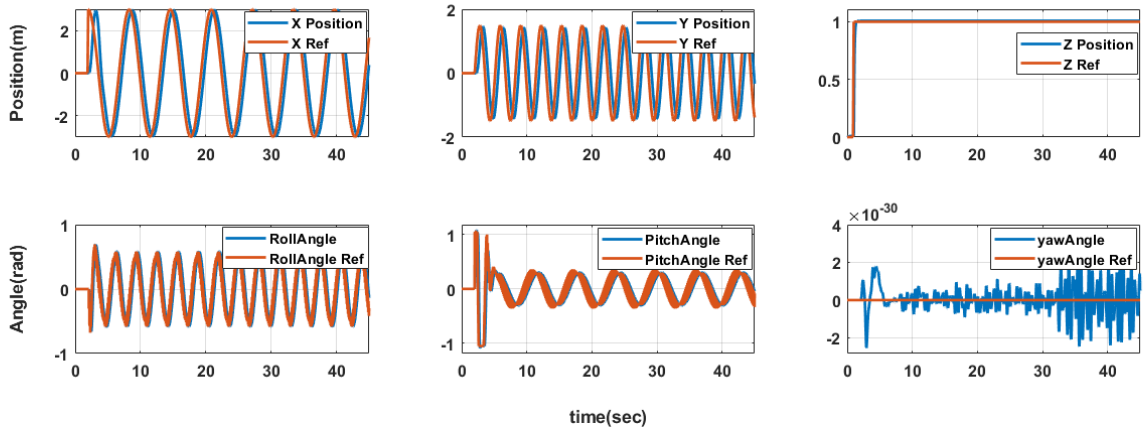


Figure 5.39: Infinity shape Trajectory tracking with PID and NMPC controllers

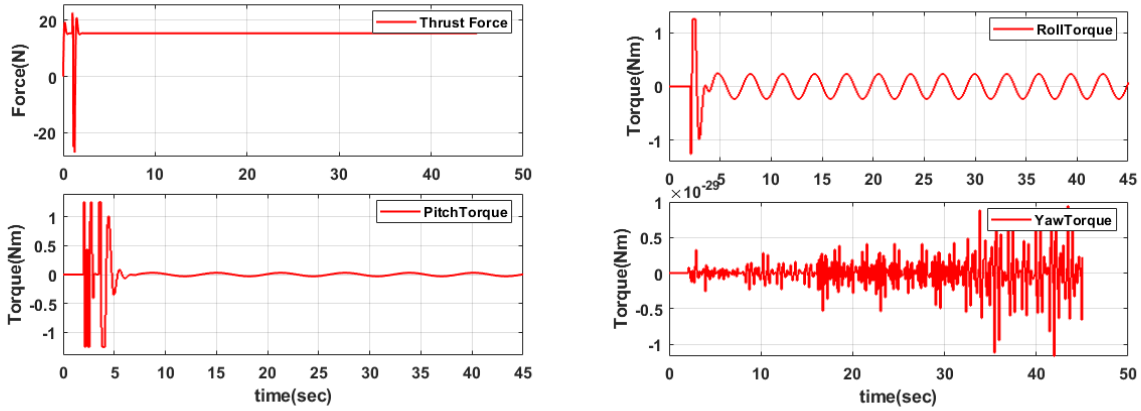


Figure 5.40: Control Inputs

Trajectory tracking error RMSE values

Trajectory	RMSE along x-axis (%)	RMSE along y-axis (%)	RMSE along z-axis (%)
Circular Trajectory	0.1911	0.2705	0.1549
Helix Trajectory	0.5990	0.5556	0.0138
Infinity shape Trajectory	0.2957	0.2793	0.0418

Table 5.13: Tracking Error RMSE value when PID is position controller and NMPC is attitude controller

Trajectory	IAE along x-axis (%)	IAE along y-axis (%)	IAE along z-axis (%)
Circular Trajectory	686.5882	787.3296	87.2579
Helix Trajectory	2.3290×10^3	2.1877×10^3	48.3
Infinity shape Trajectory	1.1537×10^3	1.1032×10^3	50.7

Table 5.14: Tracking Error IAE value when PID is position controller and NMPC is attitude controller

Summary

The goal of this section was to simulate position controller to allow the quadcopter to perform missions. These missions, trajectories, are defined by a sequence of coordinates that the quadcopter will attempt to reach and trajectories to follow when moving to the destinations. The implementation of the trajectory controller was successful, with the quadcopter being capable of maintaining the reference trajectories and travelling to the defined coordinates. Three different attitude control algorithms were implemented with the same position controller, PID controller. Their trajectory tracking performance were presented by using RMSE measure. The comparison based on the RMSE measure is stated in the following tables. PID controller with NMPC shows the best performance over LMPC and LMPC+FBL control schemes.

Controllers	RMSE along x-axis (%)	RMSE along y-axis (%)	RMSE along z-axis (%)
PID and LMPC	0.4164	0.4855	0.2317
PID and FBL+LMPC	0.2145	0.2755	0.1723
PID and NMPC	0.1911	0.2705	0.1549

Table 5.15: Tracking Error RMSE value comparison for circular path

Controllers	RMSE along x-axis (%)	RMSE along y-axis (%)	RMSE along z-axis (%)
PID and LMPC	1.3963	1.3978	0.1278
PID and FBL+LMPC	0.7664	0.6608	0.0613
PID and NMPC	0.5990	0.556	0.0138

Table 5.16: Tracking Error RMSE value comparison for helix trajectory

Controllers	RMSE along x-axis (%)	RMSE along y-axis (%)	RMSE along z-axis (%)
PID and LMPC	2.0951	1.4084	0.0854
PID and FBL+LMPC	0.3843	0.3245	0.0516
PID and NMPC	0.2957	0.2793	0.0418

Table 5.17: Tracking Error RMSE value comparison for infinity shape trajectory

Chapter 6

Conclusion and Future work

The major goal of this thesis was to demonstrate the benefits and drawbacks of several Model predictive control (MPC) algorithms on a quadcopter attitude subsystem. To accomplish so, the quadcopter's control coordinate frames were established, and a nonlinear mathematical model of the quadcopter was derived. The controller was designed once the quadcopter's mathematical model was completed. Two sub-controllers were developed. The position variables x, y, and z are all controlled by outer loop controller. The angles are controlled by attitude controller, inner loop controller, which uses three different MPC techniques.

The first derived attitude control method was a linear MPC, which uses a linearized version of the plant model. The Quadcopter nonlinear model is linearized around the equilibrium points, and the LMPC uses this linearized model as a reference or prediction model. The main reasons behind this technique are first to reduce computation burden, this is due to the fact that LMPC with linear model and linear constraint produce a convex optimization problem which is relativly easy to solve so the efficincy in time is much better and , on the other hand, the technique produces good results when the plant is functioning around the equilibrium point. However, when the system operates far from the equilibrium points and faces disturbance, the performance of the controller is relatively poor because the control actions are applied to the existing nonlinear plant.

A feedback linearization technique with LMPC is designed to overcome the effect of linearization, which make the system only works well around equilibrium point and to reduce the computational burden, which occured by utilizing nonlinear MPC (NMPC) directly. The NMPC technique uses a nonlinear model for prediction which leads to a non-convex nonlinear program, even though the cost function and constraint sets are convex. Thus, finding a global optimum is a challenging and computationally intensive task. Generally, this con-

trol technique, FBL with MPC, allows the utilization of linear MPC (LMPC) for nonlinear systems without taking into account a single operating point, while reducing computational burden. Despite these facts, when the system is subjected to disturbances, this technique performs poorly. Moreover, the technique couldn't consider constraints, because the linear input constraints becoming nonlinear and can't be included in the optimization process.

NMPC was the third attitude control method. Which predicts plant output and generates control signals using the original nonlinear plant model. In order to handle tighter performance specifications and constraints, due to environmental and safety considerations, the control system should consider system nonlinearities and constraints explicitly. In this regard nonlinear model predictive control (NMPC) is the best controller to meet these requirements. Therefore, this strategy provides the best result in both constraint handling and disturbance rejection.

An outer loop controller has been implemented as a track generator to tackle the path tracking problem. The main reason of designing this outer loop controller is to test and verify the capability of the inner loop (Attitude) controllers in path tracking problem. The outer loop controller generates the necessary thrust force and reference angles using a previously determined path reference and a standard PID controller. The reference trajectory tracking was then done by merging the outer loop controller with the inner loop controller, i.e. as a cascaded controller. The performance of each control strategy was then compared based on the tracking of three different reference trajectories, circular trajectory, helix trajectory and infinity shape trajectory. The simulation findings were thoroughly described.

The combination of PID and LMPC controllers shows poor tracking performance than the other. This is due to the fact that the inner loop control is not well performing because the nonlinear plant was approximated by its linear prediction model. Therefore, the system performed poorly ,when the operating range has been far from the leanerization point. The other two trajectory tracking controller was a fusion of PID and FBL with LMPC and PID and NMPC. It is shown that both set of controllers outperform the PID LMPC combination controller in tracking performance. Furthermore, due to considerably lower computational load, the combination of PID with FBL+LMPC may be preferable for indoor application, which is under normal condition, because it isn't requires a powerful hardware when implementing in real time application. On the other hand, the PID NMPC controller is very successful in disturbance rejection and constraint handling even if implementing it in real time application using conventional industrial hardware is very hard.

Future work could be designing Adaptive Feedback linearization in combination with LMPC in order to overcame the model and parameter uncertainty. Moreover, try to find ways to handle a constraints and applying the three strategies for real quadcopter in order to obtain the best performance.

References

[1] C. A. Amadi *et al.*, "Design and implementation of a model predictive control on a pixhawk flight controller." Ph.D. dissertation, Stellenbosch: Stellenbosch University, 2018.

[2] M. Moradzadeh, R. Boel, and L. Vandevelde, "Anticipating and coordinating voltage control for interconnected power systems," *Energies*, vol. 7, no. 2, pp. 1027–1047, 2014.

[3] J. Zietkiewicz, A. Owczarkowski, and D. Horla, "Performance of feedback linearization based control of bicycle robot in consideration of model inaccuracy," in *International Conference on Automation*. Springer, 2016, pp. 399–410.

[4] J. A. Primbs and V. Nevistic, "Mpc extensions to feedback linearizable systems," in *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, vol. 3. IEEE, 1997, pp. 2073–2077.

[5] İ. C. Dikmen, A. Arisoy, and H. Temeltas, "Attitude control of a quadrotor," in *2009 4th International Conference on Recent Advances in Space Technologies*. IEEE, 2009, pp. 722–727.

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

[7] A. L. Salih, M. Moghavvemi, H. A. Mohamed, and K. S. Gaeid, "Flight pid controller design for a uav quadrotor," *Scientific research and essays*, vol. 5, no. 23, pp. 3660–3667, 2010.

[8] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.

[9] E. Johnson and S. Mishra, "Flight simulation for the development of an experimental uav," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2002, p. 4975.

[10] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Epfl, Tech. Rep., 2007.

[11] Y. Li and S. Song, "A survey of control algorithms for quadrotor unmanned helicopter," in *2012 IEEE fifth international conference on advanced computational intelligence (ICACI)*. IEEE, 2012, pp. 365–369.

[12] Y. M. Al-Younes, M. A. Al-Jarrah, and A. A. Jhemi, "Linear vs. nonlinear control techniques for a quadrotor vehicle," in *7th International Symposium on Mechatronics and its Applications*. IEEE, 2010, pp. 1–10.

[13] B. Panomrattanarug, K. Higuchi, and F. Mora-Camino, "Attitude control of a quadrotor aircraft using lqr state feedback controller with full order state observer," in *The SICE Annual Conference 2013*. IEEE, 2013, pp. 2041–2046.

[14] M. Islam and M. Okasha, "A comparative study of pd, lqr and mpc on quadrotor using quaternion approach," in *2019 7th International Conference on Mechatronics Engineering (ICOM)*. IEEE, 2019, pp. 1–6.

[15] J. Guldner and V. Utkin, "The chattering problem in sliding mode systems," in *Fourteenth intenational symposium of mathematical theory of networks and systems, MTNS2000*, 2000.

[16] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 2247–2252.

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

[18] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199, no. 1.

[19] M. Huang, B. Xian, C. Diao, K. Yang, and Y. Feng, "Adaptive tracking control of underactuated quadrotor unmanned aerial vehicles via backstepping," in *Proceedings of the 2010 American Control Conference*. IEEE, 2010, pp. 2076–2081.

[20] A. Bemporad, C. A. Pascucci, and C. Rocchi, "Hierarchical and hybrid model predictive control of quadcopter air vehicles," *IFAC Proceedings Volumes*, vol. 42, no. 17, pp. 14–19, 2009.

[21] K. Alexis, C. Papachristos, R. Siegwart, and A. Tzes, "Robust model predictive flight control of unmanned rotorcrafts," *Journal of Intelligent & Robotic Systems*, vol. 81, no. 3-4, pp. 443–469, 2016.

[22] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[23] B. Whitehead and S. Bieniawski, "Model reference adaptive control of a quadrotor uav," in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 8148.

[24] W. Zeng, B. Xian, C. Diao, Q. Yin, H. Li, and Y. Yang, "Nonlinear adaptive regulation control of a quadrotor unmanned aerial vehicle," in *2011 IEEE International Conference on Control Applications (CCA)*. IEEE, 2011, pp. 133–138.

[25] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2668–2673.

[26] J. J. Craig, P. Hsu, and S. S. Sastry, "Adaptive control of mechanical manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 16–28, 1987.

[27] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, p. 22, 2011.

[28] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics and Automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[29] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, 2008, p. 7410.

[30] A. Nagaty, S. Saeedi, C. Thibault, M. Seto, and H. Li, "Control and navigation framework for quadrotor helicopters," *Journal of intelligent & robotic systems*, vol. 70, no. 1, pp. 1–12, 2013.

[31] L. Derafa, T. Madani, and A. Benallegue, "Dynamic modelling and experimental identification of four rotors helicopter parameters," in *2006 IEEE international conference on industrial technology*. IEEE, 2006, pp. 1834–1839.

[32] G. Regula, "Formation control of autonomous aerial vehicles," 2013.

[33] H. M. ElKholy, "Dynamic modeling and control of a quadrotor using linear and nonlinear approaches," *American University in Cairo*, 2014.

[34] J. Rossiter, "Model based predictive control: A practical approach, control," 2003.

[35] L. Wang, *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[36] F. A. T. Al-Saedi and R. A. Sabar, "Design and implementation of autopilot system for quadcopter," *IJCSET*, vol. 5, no. 6, pp. 190–199, 2015.

[37] A. Bemporad, M. Morari, and N. L. Ricker, "Model predictive control toolbox™ user's guide," *http://www. mathworks. com/access/helpdesk/help/toolbox/mpc/*, 2004.

[38] G. Takács and B. Rohal'-Ilkiv, *Model predictive vibration control: efficient constrained MPC vibration control for lightly damped mechanical structures*. Springer Science & Business Media, 2012.

[39] P. S. Agachi, Z. K. Nagy, M. V. Cristea, and Á. Imre-Lucaci, *Model based control: case studies in process engineering*. John Wiley & Sons, 2007.

[40] L. G. Chambers, "Practical methods of optimization (2nd edn), by r. fletcher. pp. 436.£ 34.95. 2000. isbn 0 471 49463 1 (wiley)." *The Mathematical Gazette*, vol. 85, no. 504, pp. 562–563, 2001.

[41] J. Slotin and W. Li, "Applied nonlinear control, printice-hall of india pvt. limited," *New Delhi*, 1991.

[42] J. Zietkiewicz, "Linear quadratic control with feedback linearized models," *Studies in Automation and Information Technology*, vol. 40, pp. 37–49, 2015.

[43] J. Zietkiewicz and A. Owczarkowski, "Direct nonlinear model predictive control and predictive control with feedback linearization. a comparison of the approaches," in *2017 18th International Carpathian Control Conference (ICCC)*. IEEE, 2017, pp. 451–455.

[44] Z. He, Y. Chen, Z. Shen, E. Huang, S. Li, Z. Shao, and Q. Wang, "Ard-mu-copter: A simple open source quadcopter platform," in *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. IEEE, 2015, pp. 158–164.

[45] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[46] M. Kvasnica, *Real-time model predictive control via multi-parametric programming: theory and tools*. VDM-Verlag, 2009.

[47] T. Backx, O. Bosgra, and W. Marquardt, "Integration of model predictive control and optimization of processes: enabling technology for market driven process operation," *IFAC Proceedings Volumes*, vol. 33, no. 10, pp. 249–260, 2000.

[48] M. Cannon, "Model predictive control, lecture notes. michaelmas term 2005 (4 lectures), course code 4me44. university of oxford," 2005.

[49] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 18, no. 8, pp. 816–830, 2008.

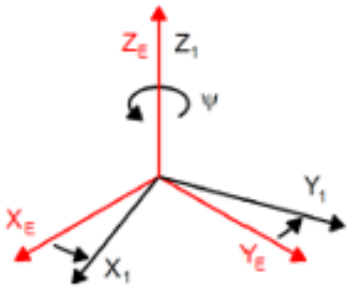
Appendix

Rotation and Transfer matrices

A) Rotational matrix

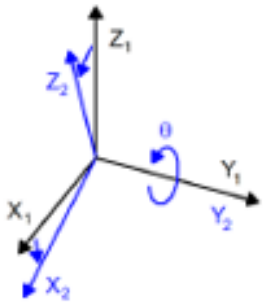
Rotational matrix, R, is a 3x3 matrix which is used to take the E-frame translational measurements into the B-frame. Post-multiplying the three fundamental rotation matrices in the following order yields the rotation matrix R:

$\rightarrow$ Rotation about Z-axis of the inertial frame, Z_E, of the yaw angle, ψ, through $R(\psi, Z)$



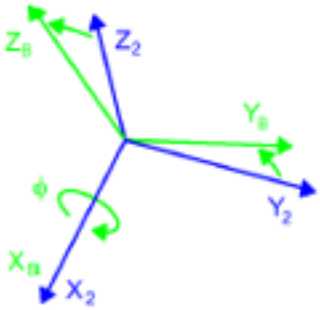
$$R(\psi, Z) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

→ **Rotation about Y-axis of the inertial frame, Y_E, of the pitch angle, θ, through** $R(\theta, Y)$



$$R(\theta, Y) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{2}$$

→ **Rotation about X-axis of the inertial frame, X_E, of the roll angle, ϕ, through** $R(\phi, X)$



$$R(\phi, X) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{3}$$

Then the rotational matrix, R, became:

$$R = R(\psi, Z)R(\theta, Y)R(\phi, X)$$

$$R = \begin{bmatrix} cos\theta cos\psi & sin\phi sin\theta cos\psi - cos\phi sin\psi & cos\phi sin\theta cos\psi + sin\phi sin\psi \\ cos\theta sin\psi & sin\phi sin\theta sin\psi + cos\theta cos\psi & cos\phi sin\theta sin\psi - sin\phi cos\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \tag{4}$$

B) Transfer matrix

Transfer matrix, T, is a 3x3 matrix which is used to take the E-frame rotationa measurements into the B-frame. T can be resolved as follows:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} R(\phi, X)^{-1} + \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\phi, X)^{-1}R(\theta, Y)^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = T^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{5}$$

$$T^{-1} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & cos(\theta)sin(\phi) \\ 0 & -sin(\phi) & cos(\theta)cos(\phi) \end{bmatrix} \tag{6}$$

$$T = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & \frac{sin(\phi)}{cos(\theta)} & \frac{cos(\phi)}{cos(\theta)} \end{bmatrix} \tag{7}$$

Quadratic Programming

A quadratic programming optimization problem minimizes a multivariable quadratic function, which is subject to linear constraints on the variables. Let us assume u is in general a vector containing the optimization variables, while **H** is a symmetric matrix and **G** is a

vector. A quadratic programming problem is then defined by [45, 46]:

$$\begin{aligned} &minimize\ f(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{H}\mathbf{u} + \mathbf{G}^T\mathbf{u} \\ &\qquad\qquad \mathbf{A}_c\mathbf{u} \leq \mathbf{b}_0 \\ &subjected\ to \\ &\qquad\qquad \mathbf{A}_e = \mathbf{b}_e \end{aligned} \tag{8}$$

where the first constraint is the inequality constraint and the second is an equality constraint. If $\mathbf{H}$ is a positive semidefinite matrix, then the function $f(u)$ is convex and it has a global minimizer, if there exists a feasible vector u. Feasibility means that the variable u satisfies all constraints. Given a positive definite $\mathbf{H}$ and a feasible u the global minimizer of the QP is unique [38].

Active set algorithm

One of the best-known quadratic programming algorithms is referred to as the active set (AS) algorithm [**?**]. In essence, the active set algorithm finds the optimal solution of the constrained QP optimization by evaluating problems involving only equality constraints. The active set QP algorithm is used with medium-scale optimization problems, which are typical for model predictive control.

Let us have a look at the general inequality constraint formulation in (8). Each row of the matrix inequality formulation is an individual linear constraint, which we can denote with a simple sequential number i. Therefore, we may separate the individual constraints if we denote the rows of $\mathbf{A}_c$ with A_c^i and the rows of $\mathbf{b}_0$ with b_0^i to get $A_c^i\mathbf{u} \leq b_c^i$ for each constraint. A certain constraint is said to be inactive if the term $A_c^i\mathbf{u}^* < b_c^i$ holds. However, in case the term $A_c^i\mathbf{u} = b_c^i$ holds, then the i^{th} constraint is said to be active. It is also possible for the active set to be empty if none of the inequality constraints are in effect at the solution [47]. The optimal solution of the problem is the solution of an equality constrained problem where only the active equality constraints are considered while the rest is discarded. We can denote this by:

$$\begin{aligned} &minimize\ f(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{H}\mathbf{u} + \mathbf{G}^T\mathbf{u} \\ &subjected\ to\ A_c^i\mathbf{u} = b_0^i \end{aligned} \tag{9}$$

where $a^* = i : A_c^i\mathbf{u}^* = b_0^i$ is the configuration of the active constraints at the solution of the quadratic programming problem [48].

The rough outline of an active set quadratic programming algorithm may be given by [45, 48, 49, 47]. Generally in order to solve a quadratic programming problem (given a feasible initial solution u_0 and an active set a_0), at each iteration p perform the following procedure:

- solve the equality problem corresponding to the active set $\mathbf{a}_p$ given by (9)
- test the optimality of the solution
  - if the solution is optimal ($\mathbf{u}_p = \mathbf{u}^*$), terminate
  - if the solution is not optimal, continue
- find which constraint i violated feasibility and add it to the active set to create a new active set $\mathbf{a}_p + 1$. Repeat the procedure to find an improved solution $\mathbf{u}_p + 1$.

For more detailed explanation you can refer [38].

MATLAB Script to determine design parameters

```
% Setup Quadcopter
m = 0.2;
Ix = 0.1;
Iy = 0.1;
Iz = 0.02;
a1 = (Iy-Iz)/Ix;
a2 = (Iz-Ix)/Iy;
a3 = (Ix-Iy)/Iz;
b1 = 1/m;
b2 = 1/Ix;
b3 = 1/Iy;
b4 = 1/Iz;
% sample time
ts = 0.01;
% Environment
g = -9.8;
k =1;
```

MATLAB Script for Path Generation

```
% This script generate (x,y,z) data points by writing the path equations.
Tfinal = 43;
tss = 0.1;
t = 0:tss:Tfinal;
% Write the equations of the desired path
x = 2*sin(0.3*t);
y = 2 - 2*cos(0.3*t);
z = 3 + 1*sin(0.6*t);

A = zeros(length(t),3);
 for i = 1:length(x)
     A(i,1) = x(i);
     A(i,2) = y(i);
     A(i,3) = z(i);
 end

 clear x;
 clear y;
 clear tss;
 clear t;


PathPts = A;

% Generate Reference signals
TakeOffTime = 1;
XYflyTime = 2;
tss = 0.1;
Tfinal = XYflyTime + tss*(length(PathPts)-1);
t = 0:tss:Tfinal;
sigZ = [t;0*t];
for i = 1:length(sigZ)
    takeoffindex = (TakeOffTime/tss)+1;
    startindex = (XYflyTime/tss)+1;
```

```
    if i < takeoffindex
        sigZ(2,i) = 0;
    elseif i < startindex
        sigZ(2,i) = PathPts(1,3);
    else
        sigZ(2,i) = PathPts(i-startindex+1,3);
    end
end

sigX = [t;0*t];
sigY = [t;0*t];
for i = 1:length(sigX)
    startindex = (XYflyTime/tss)+1;
    if i < startindex
        sigX(2,i) = 0;
        sigY(2,i) = 0;
    else
        sigX(2,i) = PathPts(i-startindex+1,1);
        sigY(2,i) = PathPts(i-startindex+1,2);
    end
end

Zcmd = timeseries(sigZ(2:end,:),sigZ(1,:));
Ycmd = timeseries(sigY(2:end,:),sigY(1,:));
Xcmd = timeseries(sigX(2:end,:),sigX(1,:));
```

MATLAB Script for 3D Animation

```
% Simulink 3D Animation
figure;
xlimit = [-10 10];
ylimit = [-10 10];
zlimit = [-6 6];
width = 700;
height = 600;
```

```
N_Figure(xlimit,ylimit,zlimit,-27,27,width,height,Xref,Yref,Zref);
hold on;
pause(1)
AnimTar(out.time,out.XYZ,out.EulerAngles,out.VXYZ)
% Plot XYZ vs Time
figure;
subplot(2,3,1);
plot(out.time,out.XYZ(:,1));
xlabel('time');
ylabel('X Pos');
grid on;
hold on;
plot(Xcmd);
legend('X_Position','X_Ref');

subplot(2,3,2);
plot(out.time,out.XYZ(:,2));
xlabel('time');
ylabel('Y Pos');
grid on;
hold on;
plot(Ycmd);
legend('Y_Position','Y_Ref');


subplot(2,3,3);
plot(out.time,out.XYZ(:,3));
xlabel('time');
ylabel('Z Pos');
grid on;
hold on;
plot(Zcmd);
legend('Z_Position','Z_Ref');

subplot(2,3,4);
plot(out.time,out.EulerAngles(:,1));
```

```
xlabel('time');
ylabel('Roll angle');
grid on;
hold on;
plot(out.time,out.EulerAngls_Ref(:,1));
legend('RollAngle','RollAngle_Ref');

subplot(2,3,5);
plot(out.time,out.EulerAngles(:,2));
xlabel('time');
ylabel('Pitch angle');
grid on;
hold on;
plot(out.time,out.EulerAngls_Ref(:,2));
legend('PitchAngle','PitchAngle_Ref');

subplot(2,3,6);
plot(out.time,out.EulerAngles(:,3));
xlabel('time');
ylabel('Yaw angle');
grid on;
hold on;
plot(out.time,out.EulerAngls_Ref(:,3));
legend('yawAngle','yawAngle_Ref');
pause(1)

%% Local Functions

function N_Figure(xlim,ylim,zlim,viewx,viewy,w,h,Xref,Yref,Zref)
    set(gca, 'XLim', xlim,'YLim',ylim,'ZLim',zlim);
    view(viewx,viewy)
    x0=10;
    y0=10;
    set(gcf,'position',[x0,y0,w,h])
    hold on;
    grid on;
```

```
    plot3(Xref,Yref,Zref)
end

function AnimTar(t_plot,XYZs,EulerAngles,VXYZs)
    t_section = 0;
    curve = animatedline('LineWidth',0.5);
    %curveTR = animatedline('LineWidth',1,'LineStyle',':');
    for i = 1:length(t_plot)
        if abs( t_plot(i) - t_section) < 0.0001
            % Do Animation
            Euler = EulerAngles(i,:);
            XYZ = XYZs(i,:);
            VXYZ = VXYZs(i,:);

            O = eye(3);
            T_BtoI = matrixB2I(Euler(1),Euler(2),Euler(3));
            O_I = T_BtoI*O;

            addpoints(curve, XYZ(1), XYZ(2),XYZ(3))


            line1 = drawline(XYZ,O_I(:,1),'b--',1.5);
            line2 = drawline(XYZ,O_I(:,2),'b--',1.5);
            line3 = drawline(XYZ,O_I(:,3),'b--',1.5);
            line5 = extendline(XYZ,O_I(:,1),'g:');

            frame1 = drawline(XYZ,0.5*O_I(:,1)+0.5*O_I(:,2),'black',2.5);
            frame2 = drawline(XYZ,0.5*O_I(:,1)-0.5*O_I(:,2),'red',2.5);
            frame3 = drawline(XYZ,-0.5*O_I(:,1)+0.5*O_I(:,2),'red',2.5);
            frame4 = drawline(XYZ,-0.5*O_I(:,1)-0.5*O_I(:,2),'black',2.5);

            drawnow
            pause(0.05)

            % logs
            xlabel(string( num2str(t_plot(i),'%.1f') )+' sec   ');
```

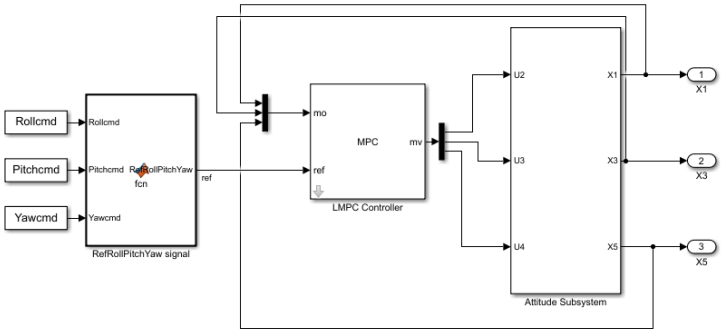
```
            dispstr7 = string( num2str( VXYZ(1),'%.1f' ) );
            dispstr8 = string( num2str( VXYZ(2),'%.1f' ) );
            dispstr9 = string( num2str( VXYZ(3),'%.1f' ) );
            vstr = 'Velocity ['+dispstr7+ ' , '+dispstr8 + ' , ' + dispstr9 + ']';

            dispstr1 = string( num2str( rad2deg(Euler(1)),'%.1f' ) );
            dispstr2 = string( num2str( rad2deg(Euler(2)),'%.1f' ) );
            dispstr3 = string( num2str( rad2deg(Euler(3)),'%.1f' ) );
            dispstr4 = string( num2str( XYZ(1),'%.1f' ) );
            dispstr5 = string( num2str( XYZ(2),'%.1f' ) );
            dispstr6 = string( num2str( XYZ(3),'%.1f' ) );
            title('EulerAngle ['+dispstr1+' , '+dispstr2+' , '+dispstr3+']  XYZ [' + dis
                + ' , ' +dispstr5+ ' , '+dispstr6 +']   ' + vstr);

            t_section = t_section + 0.2;

            delete(line1)
            delete(line2)
            delete(line3)
            delete(line5)

            delete(frame1)
            delete(frame2)
            delete(frame3)
            delete(frame4)

        end
    end
end


function m = matrixB2I(phi,theta,psi)
    T_BtoV2 = [[1 0 0];[0 cos(-phi) sin(-phi)];[0 -sin(-phi) cos(-phi)]];
    T_V2toV1 = [[cos(-theta) 0 -sin(-theta)];[0 1 0];[sin(-theta) 0 cos(-theta)]];
    T_V1toI = [[cos(-psi) sin(-psi) 0];[-sin(-psi) cos(-psi) 0];[0 0 1]];
```

```
    m = T_V1toI*T_V2toV1*T_BtoV2;
end

function line = drawline(p1,p2,color,width)
% MYMEAN Local function that calculates mean of array.
    pt1 = p1;
    pt2 = pt1 + transpose(p2);
    pts = [pt1;pt2];
    line = plot3(pts(:,1), pts(:,2), pts(:,3),color,'LineWidth',width);
end

function line = extendline(p1,p2,color)
% MYMEAN Local function that calculates mean of array.
    pt1 = p1;
    pt2 = pt1 + 20*transpose(p2);
    pts = [pt1;pt2];
    line = plot3(pts(:,1), pts(:,2), pts(:,3),color,'LineWidth',1.5);
end
```

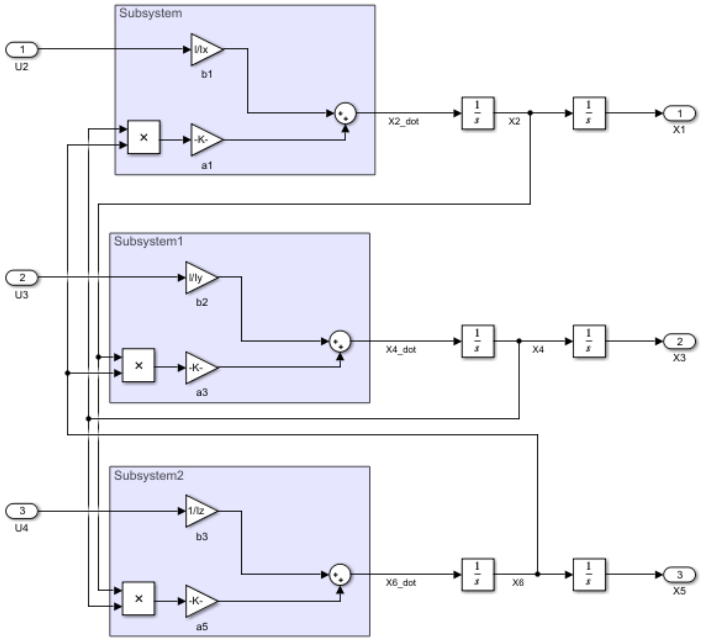
MATLAB/Simulink Block diagrams

Attitude control

A) Attitude control by LMPC Simulink block diagram



(a) Simulink block diagram of Attitude subsystem with LMPC controller



(b) Simulink block diagram of Attitude subsystem

Figure 1: Attitude subsystem with LMPC controller

B) Attitude control by FBL+LMPC Simulink block diagram

C) Attitude control by NMPC Simulink block diagram

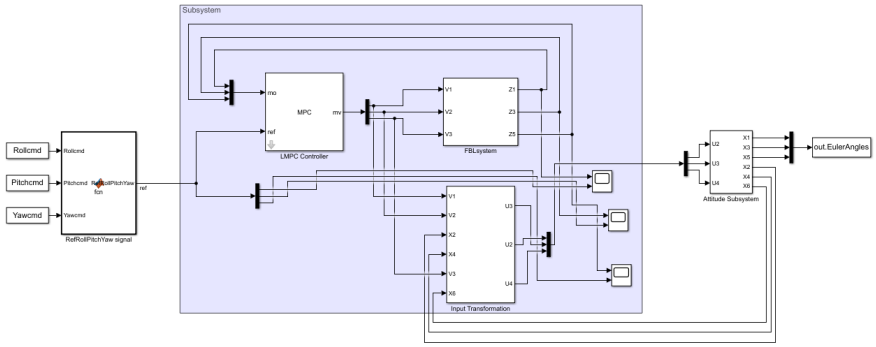


Figure 2: Simulink block diagram of Attitude subsystem with FBL+LMPC controller

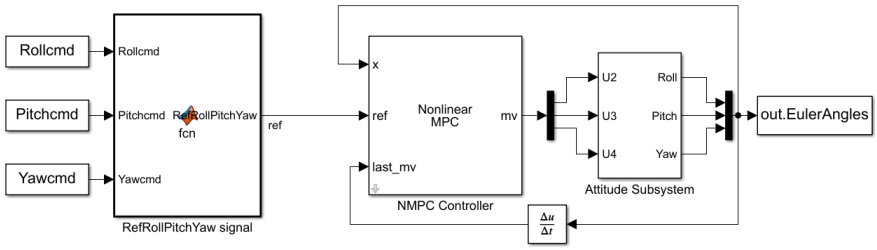
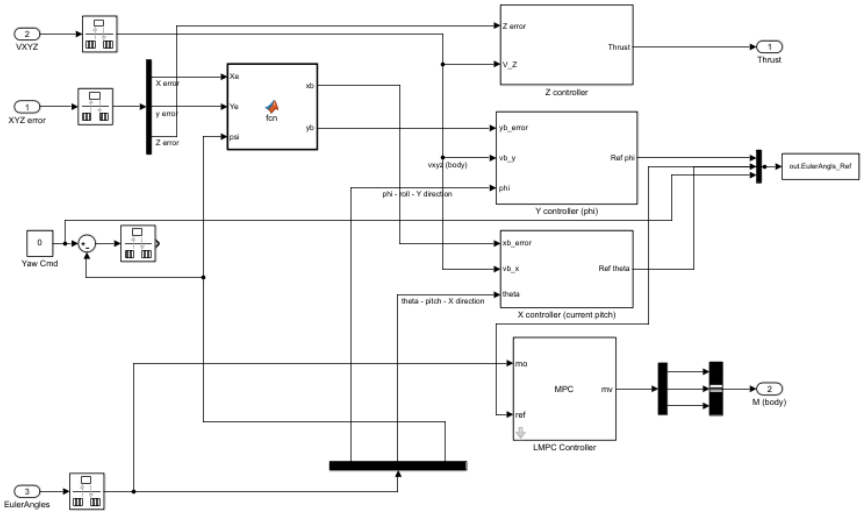


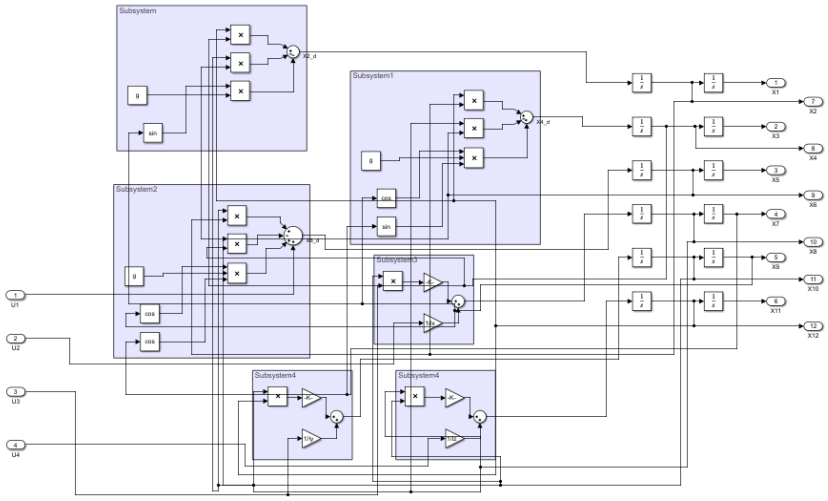
Figure 3: Simulink block diagram of Attitude subsystem with NMPC controller

Trajectory tracking control Simulink block diagram

A) Trajectory tracking control by LMPC and PID controllers Simulink block diagram



(a) Simulink block diagram of trajectory tracking control by LMPC and PID controllers



(b) Simulink block diagram of Quadcopter system

Figure 4: Quadcopter trajectory tracking control by LMPC and PID controllers

B) Trajectory tracking control by FBL+LMPC and PID controllers Simulink block diagram

C) Trajectory tracking control by NMPC and PID controllers Simulink block diagram

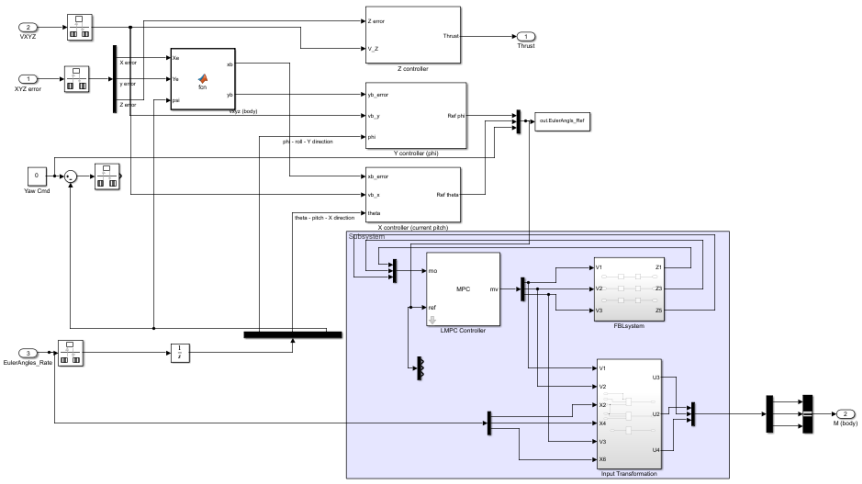


Figure 5: Quadcopter trajectory tracking control by FBL+LMPC and PID controllers

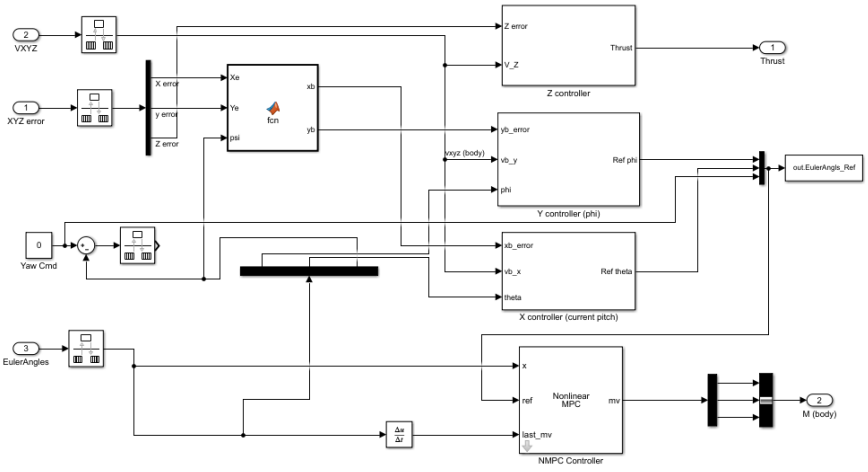


Figure 6: Quadcopter trajectory tracking control by NMPC and PID controllers