ISLAMIC AZAD UNIVERSITY

SCIENCE AND RESEARCH BRANCH

Faculty of Mechanics, Electrical and Computer -Department of Electronic Engineering

Thesis for receiving «BSc» degree in Electrical Engineering

Subject:

Design and simulation of MPC controller for quadcopter in MATLAB

Thesis Advisor:

B. Mahboobi Ph.D.

By:

Alireza Kokabi Dezfuli

Winter 2023

Abstract:

These days, quadcopters play an important role in the progress of modern societies and their use in various fields is increasing day by day. Meanwhile, controlling quadcopters and designing an optimal controller for them is one of the main challenges that has attracted the attention of many engineers in recent years. In this project, the design of the predictive controller for the quadcopter is discussed, and this controller has one of the most optimal control outputs for multivariable systems. Also, in this project, the kinematic and dynamic structure of the quadcopter is analyzed, which helps to better understand the mechanical system of the quadcopter.

Key words:

Quadcopter-rotary motion system-transitive motion system-model predictive controller-MATLAB-Simulink

Contents

Chapter 1

Introduction

1.1 Motivation

In this project, one of the main goals is to get to know the dynamic and kinematic system of the quadcopter so that the desired controller can be designed using science and knowledge about the system. Quadcopters are multi-variable and inherently unstable systems, and this feature causes them to face many problems in designing a suitable controller for them. for this reason, choosing an optimal and accurate controller is very important. The most important control part of a quadcopter system is related to its rotary movement system, the output of this system affects the location and states of the quadcopter, so we use a predictive controller to control this system.

1.2 Historical Background

These days, quadcopters have attracted the attention of a wide range of society and are used in many fields, including military, relief, and urban services. In the initial design and construction of the quadcopter, one of the main challenges is the design of a controller. The reason is that the quadcopter has six degrees of freedom and six different movements, but it only has four inputs, which causes problems. During the last two decades, many researchers and engineers have tried to design a controller for the quadcopter system according to the defined problem using different control methods. Among the performed methods, we can mention the design of PID, fuzzy, LQR, etc. controllers. However in the research conducted, the predictive control has the best and most practical results, and today this controller is also widely used in the robotics industry.

1.3 Problem Description

In this project, the main goal is to design a predictive controller for the rotary motion system of the quadcopter that follows the given reference path for rotary motions and obtains the most optimal final solution for the system.

1.4 Research Definition

In this project, the research and description of the dynamic system and the governing equations of the quadcopter system were first discussed and then the predictive controller and its design steps were explained both in theory and code.

1.5 Research Purposes

The research in this project aims to learn about the dynamic and kinematic system of the quadcopter, and after knowing the desired system, the design of the predictive controller for the rotational motion system of the quadcopter is discussed.

1.6 Research Methode

In this research, the results and explanations of numerous articles and theses were used, which were written concerning the design of the predictive controller for the quadcopter. Also, in this project, MATLAB and Simulink software have been used to design and simulate the controller.

1.7 Thesis structure

In this thesis, the explanation of the dynamic system and the description of the governing equations of the system are discussed first. In the following, the theory and design steps of the predictive controller are explained. After describing how to design the controller, we implement the explained steps in MATLAB software.

Chapter 2

System Modeling

This chapter examines the kinematics and dynamics of the quadcopter, which helps to study and understand the mechanics and behavior of the quadcopter. The aerodynamic effects on the quadcopter body will also be investigated after defining the kinematic and dynamic models of the quadcopter. The chapter ends with the formulation of a state space model for the quadcopter system, which will be explored in subsequent chapters for controller design. Kinematic and dynamic models are obtained using Newton-Euler's formula assuming the following:

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

2.1 Kinematic Model

To enter the quadcopter modeling, it is first necessary to determine the coordinate frame that will be used. Figure 2-1 shows the Earth reference frame with axes and the body frame with axes. An Earth frame is an inertial frame that, as the name suggests, is fixed at a fixed location on the Earth, using a notation where the axes are in the north, east, and down directions, respectively. On the other hand, the body frame is located in the center of the quadcopter body and its x-axis is towards propeller 1, y-axis is towards propeller 2 and z-axis is towards the direction of the ground.

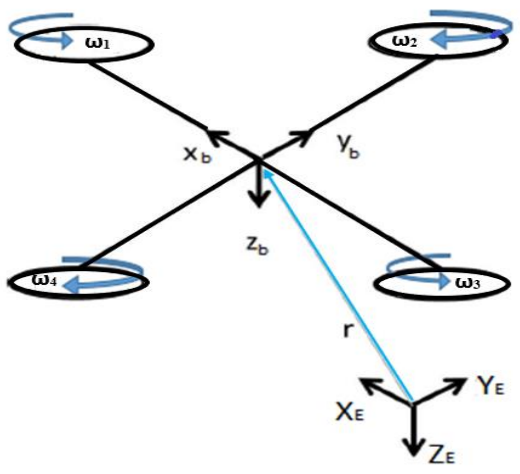


Figure 2.1: Quadcopter reference frame

The distance between the Earth frame and the body frame determines the absolute location of the quadcopter, which is shown below.

$$\mathbf{r} = [x, y, z]^T$$

The roll, yaw, and yaw angles (ϕ, θ, ψ), which represent the rotation around the X, Y, and Z axes, respectively, are used to determine the orientation of the quadcopter. The transformation matrix R transforms the inertial coordinates into the fixed frame of the body, assuming that the order of rotation is roll (ϕ), pitch (θ), and then yaw (ψ). The R matrix is as follows.

$$\mathbf{R} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) - \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) - \cos(\phi)\cos(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{2.1}$$

The transformation matrix R itself is obtained by multiplying three matrices, which represent the angular and spatial changes during rotation around the x, y, and z axes. The transformation or adaptation in the coordinate system is through three successive rotations around three axes. The order and order of rotation around three axes is important, and therefore, this order must be respected in the development of equations. If we want to match the reference device of the body to the Earth through the Euler angles, the transformation of the following rotations should be obtained in order and then multiplied together in turn. It is also worth noting that these transformations are performed assuming a constant time and a constant coordinate vector, and the diffusion matrix should be used in case of investigation over a while.

Rotation around the z-axis in the inertial frame ($\mathbf{z_E}$) with the angle of deviation ψ

$$\mathbf{R}(\psi, Z) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

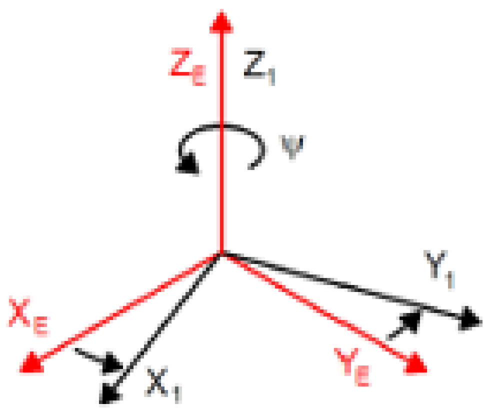


Figure 2.2: Rotation around the z-axis by the yaw angle

According to the above matrix, the changes around the z axis are the same and therefore the third column of the matrix does not include any angular changes. But the first and second columns of the matrix, which are related to the x and y axes, respectively, indicate the amount of angular changes of the quadcopter's new position relative to the original reference.

Rotation around the y-axis in the inertial frame (y_E) with the pitch angle θ

$$\mathbf{R}(\theta, Y) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.3}$$

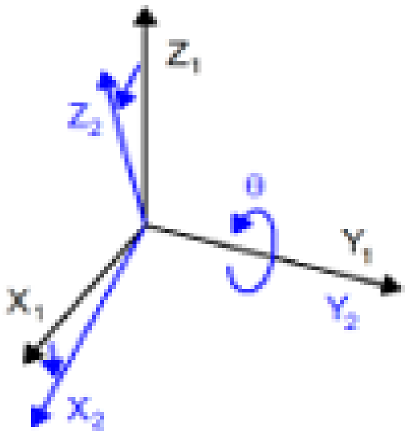


Figure 2.3: Rotation around the y-axis with a pitch angle

Similar to the matrix related to rotation around the z-axis, the second column, which shows the changes in the y-axis, has no angular change, and the coordinates of the quadcopter remain constant after the rotation in the y-axis. But as it is clear in Figure 2-3 and according to the first and third column of the matrix which is related to the x and z axes, angular and coordinate changes have been obtained in these two axes, each of the changes is proportional to each axis.

Rotation around the x-axis in the inertial frame (x_E) with the roll angle ϕ

$$\mathbf{R}(\phi, X) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.4}$$

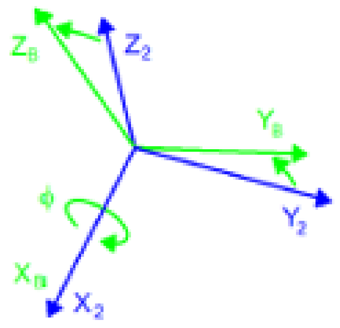


Figure 2.4: Rotation about the x-axis with the roll angle

According to the previous two matrices and the matrix related to the rotation around the x-axis, we can conclude that the changes around the x-axis are zero, and for this reason, we put a value of one in the first column of the matrix corresponding to the x-axis. But in the rest of the columns, which are respectively for the y and z axes, we write the number of angular changes according to their changes relative to the other two axes.

Now by multiplying the previous three matrices, we have

$$\mathbf{R} = \mathbf{R}(\psi, Z) \times \mathbf{R}(\theta, Y) \times \mathbf{R}(\phi, X)$$

$$\mathbf{R} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.5}$$

$$= \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) - \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) - \cos(\phi)\cos(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ 0 & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

The R matrix is used to formulate the quadcopter's dynamic model because a transition from one frame to another is required to establish the relationship between two reference frames. The fact that some modes are measured in the body frame (eg propeller thrust forces) while others are measured in the inertial frame (eg gravitational forces and quadcopter position) illustrates the utility of the R matrix.
To obtain information about the quadcopter's angular velocity, an onboard IMU is typically used, which then gives the velocity in the body's coordinate frame. To relate the Euler angular rates, $\dot{\eta} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$, measured in the inertial frame, and the angular body rates $\omega = \begin{bmatrix} p & q & r \end{bmatrix}^T$ a transformation is required as follows:

$$\omega = T\,\dot{\eta} \rightarrow \begin{bmatrix} p & q & r \end{bmatrix} = T \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix} \tag{2.6}$$

the matrix T in the above relation is equal to

$$\mathbf{T} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \dfrac{\sin(\phi)}{\cos(\phi)} & \dfrac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \tag{2.7}$$

The matrix T itself is obtained from the following equation

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

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.8}$$

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

In obtaining the transformation matrix T, we have used the relationship between angular rates and Euler's angles, and using the transformation matrix R, we write the relationship between them at the beginning. Then, by integrating the matrices corresponding to each transformation, we finally calculate transformation matrix T.

2.2 Dynamic Model

Quadcopter motion consists of two different motions, rotational motion and translational motion. Rotational motion is fully activated, while translational motion is dependent and depends on the direction of the quadcopter, which in turn depends on the rotational motion of the quadcopter. The equations of rotational motion and translational motion of the quadcopter system are defined in the following subsections.

2.2.1 Rotational Equations of Motion

Newton-Euler's method is used to derive the rotational equations of motion in the body in the following general form.

$$\mathbf{M_B} = I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} + \mathbf{M_G} \tag{2.9}$$

In the above equation, M_B represents the moment on the body, the I matrix represents the moment of inertia matrix of the quadcopter, ω represents the angular rate of the body, and represents the moment due to the inertia of the rotors. The first two terms of equation 2.9, $I\dot{\omega}$ and $\omega \times I\omega$, represent the inertial angular acceleration and centripetal forces in the body frame. while it shows the moments caused by the inertia of the rotors. The moments caused by the inertia of the rotors are defined as follows.

$$\mathbf{M_G} = \boldsymbol{\omega} \times \begin{bmatrix} 0 & 0 & I_r \boldsymbol{\omega_r} \end{bmatrix}^T \tag{2.10}$$

As a result, by substituting equations 2.10 in 2.9, we have a quadcopter rotation equation.

$$\mathbf{M_B} = I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} + \boldsymbol{\omega} \times \begin{bmatrix} \mathbf{0} & \mathbf{0} & I_r \boldsymbol{\omega_r} \end{bmatrix}^{\mathrm{T}} \tag{2.11}$$

In the above equation, it shows the inertia matrix of the rotors and the speed associated with the rotors, which is equal to

$$\boldsymbol{\omega_r} = \boldsymbol{\omega_1} + \boldsymbol{\omega_2} + \boldsymbol{\omega_3} + \boldsymbol{\omega_4} \tag{2.12}$$

Inertia Matrix (I)

Moment of inertia values describes the dynamic behavior of an object in rotation around a certain axis. The inertia matrix for the quadcopter is diagonal, and the upper and lower elements of the matrix diameter, which are the product of the inertia values, are zero due to the symmetry of the quadcopter. The inertia matrix is as follows.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.13}$$

In the above equation, I_{xx}, I_{yy}, I_{zz} are the moments of inertia along the main axes in the body frame.

Moments Acting on the Quadcopter ($\mathbf{M_B}$)

The total torque applied to the quadcopter is a physical effect caused by the rotation of the rotors. This created moment is called an aerodynamic moment. There is another physical effect produced by the rotation of the rotors, which is called aerodynamic force or lift force. Equations 2.14 and 2.15 show the aerodynamic torque and aerodynamic force generated by each rotor.

$$\mathbf{M_i} = \frac{1}{2}\rho A C_D r^2 \boldsymbol{\omega}_\mathbf{i}^\mathbf{2} \tag{2.14}$$

$$\mathbf{F_i} = \frac{1}{2}\rho A C_T r^2 \boldsymbol{\omega}_\mathbf{i}^\mathbf{2} \tag{2.15}$$

In the above equations ρ is the air density, A represents the area of the quadcopter fin, C_D, C_T are constant coefficients of moment and aerodynamic force, r is the radius of the fin, and ω is the angular velocity of each rotor.

Aerodynamic force and moment are influenced by the geometry of the propeller as well as air density. Since the maximum altitude of quadcopters is usually limited, the air density can be assumed to be constant. As a result, equations 2.14 and 2.15 can be simplified as follows, with the remaining values constant.

$$K_m = \frac{1}{2}\rho A C_T r^2 \to \mathbf{M_i} = K_m \boldsymbol{\omega}_\mathbf{i}^\mathbf{2} \tag{2.16}$$

$$K_f = \frac{1}{2}\rho A C_D r^2 \to \mathbf{F_i} = K_f \boldsymbol{\omega}_\mathbf{i}^\mathbf{2} \tag{2.17}$$

In equations 2.16 and 2.17, the values K_m and K_f are the constant coefficients of aerodynamic moment and aerodynamic force, respectively. The aerodynamic moment constant can be determined experimentally by performing the required tests for each type of propeller. By defining the forces and moments created by the propellers, the torque M_B on the quadcopter can be checked. The forces and torques on the quadcopter are shown in Figure 5-2. Each rotor generates an upward thrust force F_i and a moment M_i in the opposite direction of rotation with the rotation of the corresponding rotor i. The reason for the cross-shaped design of the quadcopter is due to symmetry in the shape. Also, in a quadcopter, the direction of torque or moment in each rotor is opposite to the direction of rotation of the rotor due to the orientation of the rotors. Each rotor is mounted at a 90-degree angle from the next rotor, with two of the rotors rotating clockwise and the other two rotating counterclockwise. When the rotors rotate clockwise, they produce torque in one direction. When the rotors rotate counterclockwise, they produce a torque in the opposite direction. This balances the torques, allowing the quadcopter to maintain stability and control. In addition, the opposing torques produced by the rotors also help to dampen any vibrations that may occur, which improves overall stability and smoothness of flight. To understand why the direction of torque in any rotor is opposite to the direction of rotation of the rotor, we need to consider the basic principles of Newton's third law of motion. This law states that for every action, there is an equal and opposite reaction. When a rotor spins, it creates a downward force, pushing the air toward the ground. According to Newton's third law, this downward force creates an equal and opposite upward force on the rotor itself. This upward force is commonly known as lift or thrust. Now, if all the rotors of a quadcopter were rotating in the same direction, the torques produced by each rotor would add up and create a net torque in one direction. This causes the quadcopter to spin uncontrollably in that direction, making it difficult to maintain stability and control. To deal with this rotational movement, quadcopters with two rotors in opposite directions have been designed. For example, front-left and rear-right rotors rotate clockwise, while front-right and rear-left rotors rotate counterclockwise. This configuration creates a balancing effect where the torques produced by the clockwise rotors cancel out the torques produced by the counterclockwise rotors. By having opposite torques, the quadcopter can maintain a stable and smooth flight. When the pilot wants to change the direction of the quadcopter or perform maneuvers, he adjusts the speed of the individual rotors and changes the torque distribution, causing the quadcopter to tilt, turn, or move in the desired direction. This configuration not only provides stability but also helps reduce vibration. Since the torques produced by the clockwise and counterclockwise rotors are opposite, they tend to cancel each other out, minimizing vibrations that can affect flight performance and the overall experience. In summary, the opposite direction of torque on each rotor in a quadcopter is a fundamental design choice that allows for stable flight, precise control, and reduced vibration by balancing the rotational forces created by the rotating rotors.

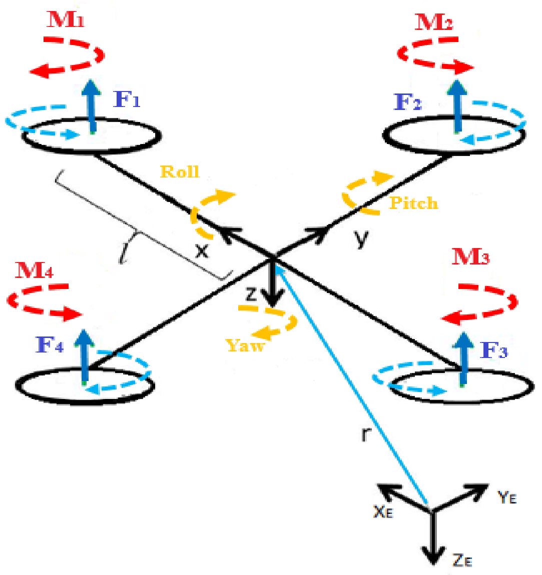


Figure 2.5: Moments and forces on the quadcopter

Starting with the moments about the x-axis of the body frame, according to the right-hand rule about the body axes, F2 multiplied by the length of the L-arm produces a negative moment along the y-axis, while F4 produces a positive moment. As a result, the total moment along the x-axis can be written as

$$\begin{aligned}\mathbf{M}_{\mathbf{x}} &= -\mathbf{F}_2 L + \mathbf{F}_4 L \\ &= -(K_f\,\boldsymbol{\omega}_2^2)L + (K_f\,\boldsymbol{\omega}_4^2)L \\ &= LK_f\,(\text{-}\boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_4^2)\end{aligned} \tag{2.18}$$

The torque generated about the y-axis of the fuselage frame, also using the right-hand rule, the thrust of rotor 1 produces a positive torque, while the thrust of rotor 3 produces a negative torque along the y-axis. The total moment can be written as follows.

$$\begin{aligned}\mathbf{M}_{\mathbf{y}} &= \mathbf{F}_1 L - \mathbf{F}_3 L \\ &= (K_f\,\boldsymbol{\omega}_1^2)L - (K_f\,\boldsymbol{\omega}_3^2)L \\ &= LK_f\,(\boldsymbol{\omega}_1^2 \text{ - } \boldsymbol{\omega}_3^2)\end{aligned} \tag{2.19}$$

The thrust of the rotors does not create along the z-axis of the body. On the other hand, the rotation of the rotors creates the torque we saw in equation 2.14. The moment about the z-axis of the body frame can be written as follows.

$$\begin{aligned}&\mathbf{M_z} = \mathbf{M_1} - \mathbf{M_2} + \mathbf{M_3} - \mathbf{M_4}\\ &= (K_m\boldsymbol{\omega}_1^2) - (K_m\boldsymbol{\omega}_2^2) + (K_m\boldsymbol{\omega}_3^2) - (K_m\boldsymbol{\omega}_4^2)\\ &= K_m(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_3^2 - \boldsymbol{\omega}_4^2)\end{aligned} \tag{2.20}$$

By combining equations 2.18, 2.19, and 2.20, we have the total moment on the body.

$$\mathbf{M_B} = \begin{bmatrix} LK_f(-\boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_4^2) \\ LK_f(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_3^2) \\ K_m(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_3^2 - \boldsymbol{\omega}_4^2) \end{bmatrix} \tag{2.21}$$

The rotational equation of motion becomes as follows by substituting equation 2.21 in equation 2.11.

$$\mathbf{M_B} = I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} + \boldsymbol{\omega} \times \begin{bmatrix} \mathbf{0} & \mathbf{0} & I_r\boldsymbol{\omega}_\mathbf{r} \end{bmatrix}^\mathbf{T}$$

$$\begin{bmatrix} LK_f(-\boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_4^2) \\ LK_f(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_3^2) \\ K_m(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_3^2 - \boldsymbol{\omega}_4^2) \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ I_r\boldsymbol{\omega}_\mathbf{r} \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} LK_f(-\boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_4^2) \\ LK_f(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_3^2) \\ K_m(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_3^2 - \boldsymbol{\omega}_4^2) \end{bmatrix} = \begin{bmatrix} I_{xx}\ddot{\phi} \\ I_{yy}\ddot{\theta} \\ I_{zz}\ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\theta}I_{zz}\dot{\psi} - \dot{\psi}I_{yy}\dot{\theta} \\ \dot{\psi}I_{xx}\dot{\phi} - \dot{\phi}I_{zz}\dot{\psi} \\ \dot{\phi}I_{yy}\dot{\theta} - \dot{\theta}I_{xx}\dot{\phi} \end{bmatrix} + \begin{bmatrix} \dot{\theta}I_r\boldsymbol{\omega}_\mathbf{r} \\ -\dot{\phi}I_r\boldsymbol{\omega}_\mathbf{r} \\ 0 \end{bmatrix}$$

$$\rightarrow \begin{cases} \ddot{\phi} = \dfrac{L}{I_{xx}}K_f(-\boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_4^2) - \dfrac{I_r}{I_{xx}}\dot{\theta}\boldsymbol{\omega}_\mathbf{r} + \dfrac{I_{yy}}{I_{xx}}\dot{\theta}\dot{\psi} - \dfrac{I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} \\ \ddot{\theta} = \dfrac{L}{I_{yy}}K_f(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_3^2) - \dfrac{I_r}{I_{yy}}\dot{\phi}\boldsymbol{\omega}_\mathbf{r} + \dfrac{I_{zz}}{I_{yy}}\dot{\phi}\dot{\psi} - \dfrac{I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} \\ \ddot{\psi} = \dfrac{1}{I_{zz}}K_m(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_3^2 - \boldsymbol{\omega}_4^2) + \dfrac{I_{xx}}{I_{zz}}\dot{\theta}\dot{\phi} - \dfrac{I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} \end{cases} \tag{2.22}$$

2.2.2 Translational Equations of Motion

The transfer equation of quadcopter movement is obtained in the framework of the earth's inertia and is based on Newton's second law. According to Newton's second law, we have

$$
\begin{aligned}
&m\mathbf{a} = \sum_{i=1}^{2} \mathbf{F}_i \\
&\mathbf{F}_1 = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \\
&\mathbf{F_2} = R\mathbf{F_B}
\end{aligned}
\tag{2.23}
$$

In the above equation, m is the mass, a is the acceleration, and F is any force acting on the quadcopter during the transfer movement. By inserting the relations governing the quadcopter in equation 2.23, we have

$$
m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R\mathbf{F_B} \tag{2.24}
$$

In the above equation $r = [x, y, z]$, g is the gravitational acceleration, F_B is the forces acting on the quadcopter and R is the transformation matrix.

Forces acting on the Quadcopter

When the quadcopter is in a horizontal position (i.e. no roll or pitch movement), the only non-gravitational force acting on it is the thrust force created by the rotation of the propellers, which is equal to the square of the propeller's angular velocity. Therefore, the non-gravitational forces acting on the quadcopter can be written as follows.

$$
\mathbf{F_B} = \begin{bmatrix} 0 \\ 0 \\ -K_f\,(\boldsymbol{\omega}_1^2 + \boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_3^2 + \boldsymbol{\omega}_4^2) \end{bmatrix} \tag{2.25}
$$

Since there are no forces in the X and Y directions, the first two rows of the force vector are zero. The last row is the sum of the thrust forces provided by each propeller. Since the thrust is upward and the positive z-axis is downward in the body frame, the symbol is negative. The thrust of the rotors is converted from the body frame to the inertial frame by multiplying F_B by the transformation matrix R, allowing the equation to be extended in any direction of the quadcopter. By expanding the translational motion equation, we have 2.24.

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

(2.26)

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

Accelerations in terms of other variables are obtained by rewriting equation 2.25 as follows.

$$\begin{cases}\ddot{x}=\dfrac{-K_f\,(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2)}{m}(\cos(\phi)\sin(\theta)\cos(\psi)+\sin(\phi)\sin(\psi))\\ \ddot{y}=\dfrac{-K_f\,(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2)}{m}(\cos(\phi)\sin(\theta)\sin(\psi)-\sin(\phi)\cos(\psi))\\ \ddot{z}=g-\dfrac{K_f\,(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2)}{m}(\cos(\phi)\cos(\theta))\end{cases}\qquad(2.27)$$

2.3 State Space Model

By converting the mathematical model of the quadcopter into a state space model, the controller design problem becomes easier for the system. So, first, we convert the model into the form of the state space.

State space vector x

To consider the possible states of the system state space, 12 items should be considered, which include Cartesian coordinates, linear velocity, Euler angles, and angular velocities. As a result, the state vector of the system will be as follows.

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{bmatrix}^T \tag{2.28}$$

By placing the parameters related to the quadcopter in the state vector, we have

$$\mathbf{x} = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & x & \dot{x} & y & \dot{y} & z & \dot{z} \end{bmatrix} \tag{2.29}$$

It can also be seen from the state space that the orientation and position of the quadcopter 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 consisting of four inputs. The four inputs U1, U2, U3, and U4 are defined as follows.

$$\mathbf{U} = \begin{bmatrix} U_1 & U_2 & U_3 & U_4 \end{bmatrix} \tag{2.30}$$

In the above vector, the U's are defined as follows.

$$\begin{cases} U_1 = K_f\,(\boldsymbol{\omega}_1^2 + \boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_3^2 + \boldsymbol{\omega}_4^2) \\ U_2 = LK_f\,(-\boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_4^2) \\ U_3 = LK_f\,(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_3^2) \\ U_4 = K_m\,(\boldsymbol{\omega}_1^2 - \boldsymbol{\omega}_2^2 + \boldsymbol{\omega}_3^2 - \boldsymbol{\omega}_4^2) \end{cases} \tag{2.31}$$

Equation 2.31 can be converted into a matrix. By doing this we have

$$\begin{bmatrix}\mathbf{U_1}\\ \mathbf{U_2}\\ \mathbf{U_3}\\ \mathbf{U_4}\end{bmatrix}=\begin{bmatrix}K_f & K_f & K_f & K_f\\ 0 & -LK_f & 0 & LK_f\\ LK_f & 0 & -LK_f & 0\\ K_m & -K_m & K_m & -K_m\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega}_1^2\\ \boldsymbol{\omega}_2^2\\ \boldsymbol{\omega}_3^2\\ \boldsymbol{\omega}_4^2\end{bmatrix} \tag{2.32}$$

Input U1 is the upward force from the four rotors, which affects the quadcopter's altitude and rate of change of altitude ($z,\dot{z}$). Input U2 is the difference between the thrust force of rotor 2 and rotor 4, which causes the roll to rotate and its speed of change ($\phi,\dot{\phi}$). The U3 input, on the other hand, represents the difference between the thrust of rotor 1 and the thrust of rotor 3, which affects the rotation of the screw and its rate of change ($\theta,\dot{\theta}$). Finally, the U4 input is the torque difference between the two clockwise and counterclockwise rotating rotors, which creates the deviating rotation and its rate of change ($\psi,\dot{\psi}$). The rotor speed can be obtained from the control inputs. The equation between the control inputs and the speed of the rotors is obtained as follows using equation 2.33.

$$\begin{bmatrix}\boldsymbol{\omega}_1^2\\ \boldsymbol{\omega}_2^2\\ \boldsymbol{\omega}_3^2\\ \boldsymbol{\omega}_4^2\end{bmatrix}=\begin{bmatrix}\frac{1}{4K_f} & 0 & \frac{1}{2LK_f} & \frac{1}{4K_m}\\ \frac{1}{4K_f} & -\frac{1}{2LK_f} & 0 & -\frac{1}{4K_m}\\ \frac{1}{4K_f} & 0 & -\frac{1}{2LK_f} & \frac{1}{4K_m}\\ \frac{1}{4K_f} & \frac{1}{2LK_f} & 0 & -\frac{1}{4K_m}\end{bmatrix}\begin{bmatrix}\mathbf{U_1}\\ \mathbf{U_2}\\ \mathbf{U_3}\\ \mathbf{U_4}\end{bmatrix} \tag{2.33}$$

The speed of the rotors can be determined using the square root of the above equation as follows.

$$\begin{cases} \boldsymbol{\omega}_1 = \sqrt{\dfrac{1}{4K_f}\mathbf{U}_1 + \dfrac{1}{2LK_f}\mathbf{U}_3 + \dfrac{1}{4K_m}\mathbf{U}_4} \\ \boldsymbol{\omega}_2 = \sqrt{\dfrac{1}{4K_f}\mathbf{U}_1 - \dfrac{1}{2LK_f}\mathbf{U}_2 - \dfrac{1}{4K_m}\mathbf{U}_4} \\ \boldsymbol{\omega}_3 = \sqrt{\dfrac{1}{4K_f}\mathbf{U}_1 - \dfrac{1}{2LK_f}\mathbf{U}_3 + \dfrac{1}{4K_m}\mathbf{U}_4} \\ \boldsymbol{\omega}_4 = \sqrt{\dfrac{1}{4K_f}\mathbf{U}_1 + \dfrac{1}{2LK_f}\mathbf{U}_2 + \dfrac{1}{4K_m}\mathbf{U}_4} \end{cases} \tag{2.34}$$

The reason for squaring the angular velocities in equation 2.32 is related to the physics of the lift force or thrust created by the rotors. This equation comes from the aerodynamic characteristics of the rotor blades. When the rotor spins faster, the airflow over the blades increases, resulting in more lift or thrust. However, this equation is not linear and is closer to a quadratic equation. In other words, lift or thrust is roughly proportional to the square of the angular velocity. By squaring the angular velocities of individual rotors and summing them in the equation, the total thrust or control signal required by the quadcopter is obtained. The k coefficients are also proportional constants that relate the squared angular velocity to the desired input or control signal. Also, the general relation of angular speed in brushless motors used in quadcopters is as follows.

$$\boldsymbol{\omega} = \frac{2\pi n}{60} \tag{2.35}$$

n represents the speed of each rotor.

2.3.1 Equation of Rotary Motion

By inserting equation 2.33 into equation 2.22, the equation of the total moments applied to the quadcopter becomes.

$$\mathbf{M_B} = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{2.36}$$

It is noteworthy that in the rotary motion equation, because only the inputs u2, u3, and u4 affect the rotary motion as mentioned before, only the mentioned inputs are placed.
Now we rewrite equation 2.22 as below.

$$\begin{cases} \ddot{\phi} = \frac{L}{I_{xx}}U_2 - \frac{I_r}{I_{xx}}\dot{\theta}\omega_r + \frac{I_{yy}}{I_{xx}}\dot{\psi}\dot{\theta} - \frac{I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} \\ \ddot{\theta} = \frac{L}{I_{yy}}U_3 - \frac{I_r}{I_{yy}}\dot{\phi}\omega_r + \frac{I_{zz}}{I_{yy}}\dot{\psi}\dot{\phi} - \frac{I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} \\ \ddot{\psi} = \frac{1}{I_{zz}}U_4 + \frac{I_{xx}}{I_{zz}}\dot{\phi}\dot{\theta} - \frac{I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} \end{cases} \tag{2.37}$$

Therefore, the state space representation of the rotational motion subsystem becomes as follows.

$$\begin{cases} \dot{x}_1 = \dot{\phi} = x_2 \\ \dot{x}_2 = \ddot{\phi} = \frac{I_{yy} - I_{zz}}{I_{xx}} x_4 x_6 - \frac{I_r}{I_{xx}} x_4 \omega_r + \frac{L}{I_{xx}} U_2 \\ \dot{x}_3 = \dot{\theta} = x_4 \\ \dot{x}_4 = \ddot{\theta} = \frac{I_{zz} - I_{xx}}{I_{yy}} x_2 x_6 - \frac{I_r}{I_{yy}} x_2 \omega_r + \frac{L}{I_{yy}} U_3 \\ \dot{x}_5 = \dot{\psi} = x_6 \\ \dot{x}_6 = \ddot{\psi} = \frac{I_{xx} - I_{yy}}{I_{zz}} x_2 x_4 + \frac{1}{I_{zz}} U_4 \end{cases} \tag{2.38}$$

$$\mathbf{f}_1(x, U) = \begin{bmatrix} x_2 \\ \frac{I_{yy} - I_{zz}}{I_{xx}} x_4 x_6 - \frac{I_r}{I_{xx}} x_4 \omega_r + \frac{L}{I_{xx}} U_2 \\ x_4 \\ \frac{I_{zz} - I_{xx}}{I_{yy}} x_2 x_6 - \frac{I_r}{I_{yy}} x_2 \omega_r + \frac{L}{I_{yy}} U_3 \\ x_6 \\ \frac{I_{xx} - I_{yy}}{I_{zz}} x_2 x_4 + \frac{1}{I_{zz}} U_4 \end{bmatrix} \tag{2.39}$$

2.3.2 Equation of Transitive Motion

By replacing equation 2.33 in equation 2.26, the equation of the non-gravitational force acting on the quadcopter becomes as follows.

$$\mathbf{F_B} = \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} \tag{2.40}$$

By rewriting equation 2.37

$$\begin{cases} \ddot{x} = \dfrac{-U_1}{m}(c\phi s\theta c\psi + s\phi s\psi) \\ \ddot{y} = \dfrac{-U_1}{m}(c\phi s\theta s\psi - s\phi c\psi) \\ \ddot{z} = g - \dfrac{U_1}{m}(c\phi c\theta) \end{cases} \tag{2.41}$$

Now we write the representation of the state space of the translational motion subsystem as follows.

$$\begin{cases} \dot{x}_7 = \dot{x} = x_8 \\ \dot{x}_8 = \ddot{x} = \dfrac{-U_1}{m}(\cos(x_1) sis(x_3)\cos(x_5) + \sin(x_1)\sin(x_5)) \\ \dot{x}_9 = \dot{y} = x_{10} \\ \dot{x}_{10} = \ddot{y} = \dfrac{-U_1}{m}(\cos(x_1)\sin(x_3)\sin(x_5) - \sin(x_1)\cos(x_5)) \\ \dot{x}_{11} = \dot{z} = x_{12} \\ \dot{x}_{12} = \ddot{z} = g - \dfrac{U_1}{m}(\cos(x_1)\cos(x_3)) \end{cases} \tag{2.42}$$

$$\mathbf{f_2}(x, U) = \begin{bmatrix} x_8 \\ \dfrac{-U_1}{m}(\cos(x_1)\sin(x_3)\cos(x_5) + \sin(x_1)\sin(x_5)) \\ x_{10} \\ \dfrac{-U_1}{m}(\cos(x_1)\sin(x_3)\sin(x_5) - \sin(x_1)\cos(x_5)) \\ x_{12} \\ g - \dfrac{U_1}{m}(\cos(x_1)\cos(x_3)) \end{bmatrix}$$

From equations 2.38 and 2.42, it can be seen that the rotary motion system is fully activated and depends only on the rotary state variables x1 to x6, which correspond to respectively. Also, the transmission system is

dependent and cannot start independently. to work and depends on both transition state variables and rotational variables.
Using the equations of rotational motion 2.38 and the equations of translational motion 2.42, the complete mathematical model of the quadcopter can be written in a state space representation as follows.

$$\mathbf{f}(x,U)=\begin{bmatrix} x_2 \\ \frac{I_{yy}-I_{zz}}{I_{xx}}x_4x_6-\frac{I_r}{I_{xx}}x_4\omega_r+\frac{L}{I_{xx}}U_2 \\ x_4 \\ \frac{I_{zz}-I_{xx}}{I_{yy}}x_2x_6-\frac{I_r}{I_{yy}}x_2\omega_r+\frac{L}{I_{yy}}U_3 \\ x_6 \\ \frac{I_{xx}-I_{yy}}{I_{zz}}x_2x_4+\frac{1}{I_{zz}}U_4 \\ x_8 \\ \frac{-U_1}{m}(\cos(x_1)\sin(x_3)\cos(x_5)+\sin(x_1)\sin(x_5)) \\ x_{10} \\ \frac{-U_1}{m}(\cos(x_1)\sin(x_3)\sin(x_5)-\sin(x_1)\cos(x_5)) \\ x_{12} \\ g-\frac{U_1}{m}(\cos(x_1)\cos(x_3)) \end{bmatrix} \qquad (2.43)$$

Chapter 3

System Control Design

After modeling the dynamic system mathematically, the next step is to design the controller for the rotary motion system and the translational motion system of the quadcopter. A predictive controller has been used to design the controller for the rotary movement system of the quadcopter. In the second part of the report, the explanation of the predictive controller will be explained first, and then after the explanation of the control method, the design of the predictive controller for the rotary movement system of the quadcopter will be explained. At first, the design is done mathematically, and then its code in MATLAB will be described.

3.1 Principle of Model Predictive Control

In predictive control, a mathematical model of the process to be controlled is used to predict the output of the model during a horizon called the prediction horizon. is the chosen sample time i for each prediction step, where the number of steps is equal to the size of the prediction horizon. A sequence of control inputs by minimizing a cost function of the error between predicted outputs and reference values or set points r(k + i), and a given weighted control input expression u(k + i) in A control horizon is obtained. The number of control entries in this sequence is equal to the size of the control horizon. This sequence of control inputs is the control actions required to drive the model and process to the reference values. Only the first element of the control input sequence is implemented in both the model and the process, as the process measurement and reference values are constantly updated at subsequent sampling moments. This minimization and implementation method is repeated at successive sampling moments and each moment, it is assumed that the reference value remains constant in the forecast horizon. The block diagram in Figure 3.1 provides a visual representation of the principles of the MPC controller, and

Figure 3.2 shows the concept of the prediction horizon in MPC with a sampling time value of 1.

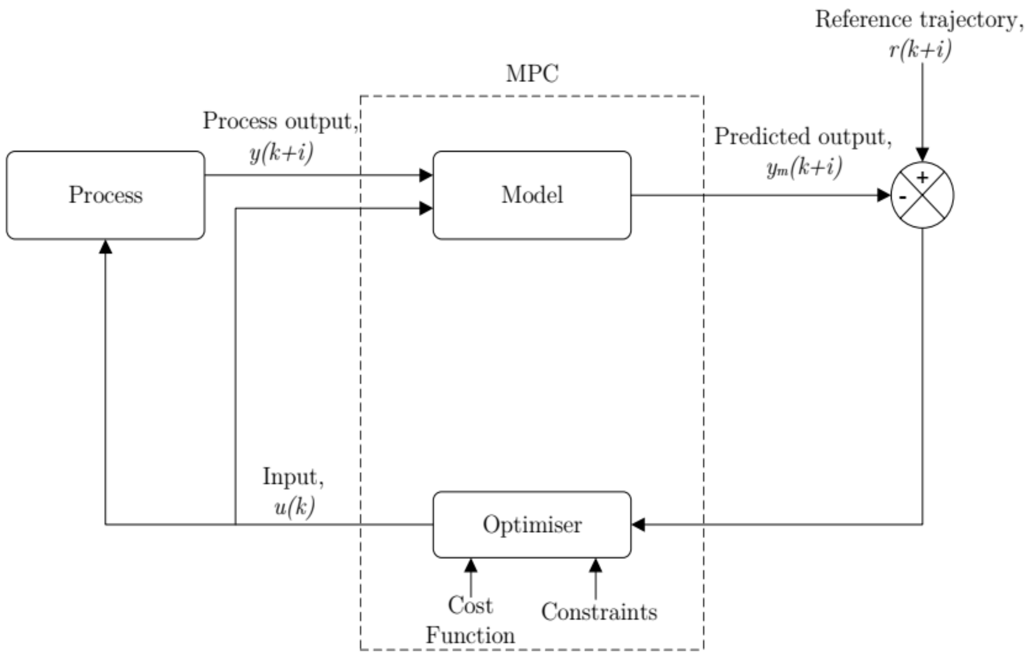


Figure 3.1: Model Predictive Control

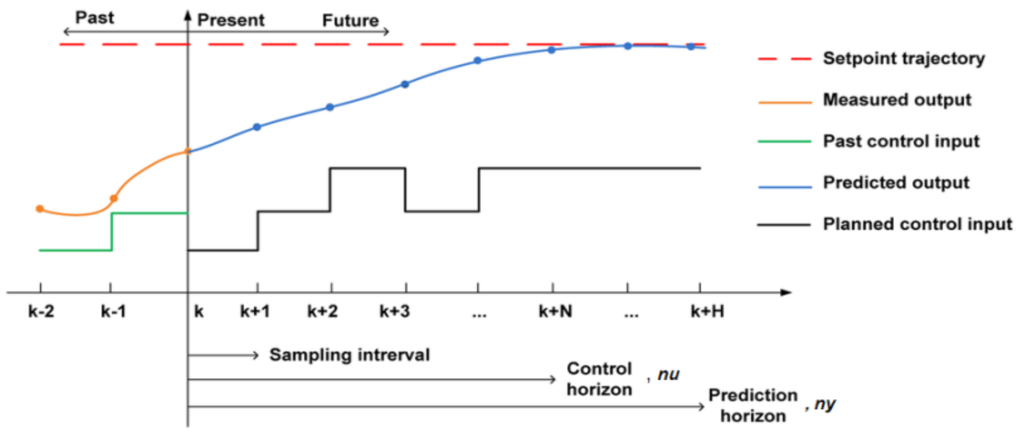


Figure 3.2: Prediction horizon

Current measured states or process outputs are used in updating future predictions to ensure that a more accurate model of the process is being used. The term receding horizon is often used for predictor control because the prediction horizon is constantly moved away at subsequent sampling moments. At the current sample time, points that were previously beyond the forecast horizon are considered, and Figure 3-3 illustrates the concept of the receding horizon.

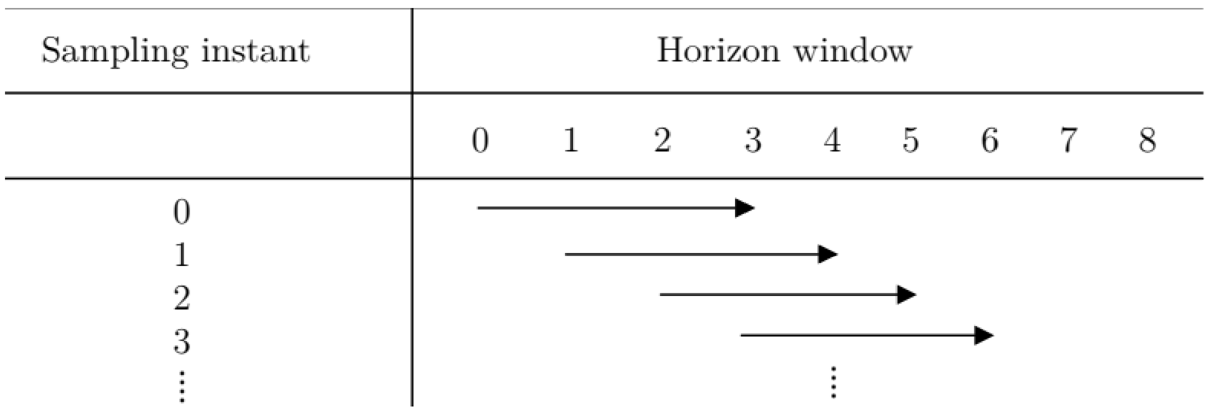


Figure 3.3: receding horizon

One of the main strengths of MPC is its ability to systematically satisfy physical constraints on control inputs, input change rates, and/or outputs. These constraints are handled directly in the optimization process. The optimization process is shown in Figure 3.1 in the optimizer block, which takes constraints and a cost function as its inputs.

3.2 MPC and PID Controller Comparison

As we know, different control methods can be implemented on the quadcopter. MPC and PID are considered the best control methods to control the rotary motion system of the quadcopter. The previous section showed that MPC uses a model to predict the system response over a chosen receding horizon and that minimizing the error between the output and reference predictions creates a necessary control path to guide the current states of the model to the reference. However, only the first element of the path is applied to the system. System constraints are considered in the controller formulation to ensure that control actions do not violate these constraints. PID runs the risk of driver saturation because the physical constraints of the system are not explicitly considered in determining the control actions. Integral desaturation is one of the techniques used to deal with these limitations. By anticipating and systematically managing the constraint, MPC is better equipped to deal with stimulus saturation. Current system states are usually used as initial states in MPC. Predictions are made using these states and this provides feedback to the MPC controller while simultaneously compensating for modeling assumptions, uncertainties, and disturbances. Model predictive control is also suitable for multiple input multiple (MIMO) systems, of which the quadcopter is one of the clear examples of this type of system. This is another advantage because MPC can systematically combine a model, constraints, and cost function with relative ease. Despite the advantages of MPC over PID, MPC is computationally more intensive because it involves many matrix and vector operations. PID does not require a model of the system and there are no matrix or vector operations to perform.

3.3 Augmented State Space Matrices

The following linear discrete state space model is used in the formulation of a linear MPC controller.

$$\begin{cases} x_m(k+1) = A_m x_m(k) + B_m u(k) \\ y(k) = C_m x_m(k) \end{cases} \tag{3.1}$$

m is the symbol of the model.

To perform trajectory tracking, the integral action must be embedded by modifying the quadcopter model in Equation 3.1. The model is strengthened according to the following process. It starts by doing a differential process in equation 3.1.

$$\begin{cases} x_m(k+1) - x_m(k) = A_m(x_m(k) - x_m(k-1)) + B_m(u(k) - u(k-1)) \\ y(k+1) - y(k) = C_m(x_m(k+1) - x_m(k)) \end{cases} \tag{3.2}$$

Now we assume the differences as follows.

$$\begin{cases}\Delta x_m(k+1) = x_m(k+1) - x_m(k) \\ \Delta u(k) = u(k) - u(k-1) \\ \Delta x_m(k) = x_m(k) - x_m(k-1)\end{cases} \tag{3.3}$$

By substituting equations 3.3 in 3.2, we have

$$\begin{cases}\Delta x_m(k+1) = A_m \Delta x_m + B_m \Delta u(k) \\ y(k+1) = C_m(A_m \Delta x_m + B_m \Delta u(k)) + y(k)\end{cases}$$
$$\rightarrow \begin{cases}\Delta x_m(k+1) = A_m \Delta x_m + B_m \Delta u(k) \\ y(k+1) = C_m A_m \Delta x_m + C_m B_m \Delta u(k) + y(k)\end{cases} \tag{3.4}$$

Now we can consider $\Delta u(k)$ as input and define the new state values as follows.

$$\mathbf{x(k)} = [\Delta x_m(k)^T, y(k)]^T \tag{3.5}$$

The augmented state space model after putting equations 3.4 and 3.5 together will be as follows.

$$\begin{cases}\begin{bmatrix}\Delta x_m(k+1) \\ y(k+1)\end{bmatrix} = \begin{bmatrix}A_m & O_{q\times n}^T \\ C_m A_m & I_{q\times q}\end{bmatrix}\begin{bmatrix}\Delta x_m(k) \\ y(k)\end{bmatrix} + \begin{bmatrix}B_m \\ C_m B_m\end{bmatrix}\Delta u(k) \\ y(k) = \begin{bmatrix}O_{q\times n} & I_{q\times q}\end{bmatrix}\begin{bmatrix}\Delta x_m(k) \\ y(k)\end{bmatrix}\end{cases} \tag{3.6}$$

In 3.6, $I_{q\times q}$ is the unit matrix with $q \times q$ dimensions where q is the number of outputs. It is a zero matrix with dimensions q × n, where n represents the number of states of the system or the dimensions of the state space. Equation 3.6 can be simplified to the following form.

$$\begin{cases}x(k+1) = Ax(k) + B\Delta u(k) \\ y(k) = Cx(k)\end{cases} \tag{3.7}$$

3.4 Controllability

By changing the initial discrete state space model to an augmented model, it is necessary to check the controllability of the augmented state space model. A system is controllable if there is a control input that brings any state of the system to zero in a finite time. An LTI system is controllable if and only if its controllability matrix has full rank such that rank(CO) = n. CO controllability matrix is defined as follows.

$$\mathbf{CO} = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{3.8}$$

3.5 Output Prediction

In the previous section, the controllability of the enhanced quadcopter model was explained. The next step in predictive controller design is to calculate the predicted output with the next control signal as the adjustable variable(s). This prediction is described in a pre-selected prediction horizon. Taking k as the current sample time, the future control path is shown as follows.

$$\mathbf{\Delta U} = \begin{bmatrix} \Delta u(k) & \Delta u(k+1) & \Delta u(k+2) & \dots & \Delta u(k+n_u-1) \end{bmatrix}^T \tag{3.9}$$

The control horizon is chosen to be less than or equal to the prediction horizon. The future state variables are required to calculate the predicted output and the future state variables are shown below.

$$x(k+1), x(k+2), x(k+3), \cdots, x(k+n_y) \tag{3.10}$$

The process used in calculating the future state and output prediction is as follows. Using the enhanced quadcopter model in equation 3.7, the prediction of the future state and output is recursively calculated as follows.

K+1:

$$\begin{cases} x(k+1) = Ax(k) + B\Delta u(k) \\ y(k+1) = Cx(k+1) \end{cases} \tag{3.11}$$

K+2:

$$\begin{cases} x(k+2) = Ax(k+1) + B\Delta u(k+1) \\ y(k+2) = Cx(k+2) \end{cases} \tag{3.12}$$

By substituting equation 3.11 in equation 3.12, we have

$$\begin{cases} x(k+2) = A^2x(k) + AB\Delta u(k) + B\Delta u(k+1) \\ y(k+2) = C(A^2x(k) + AB\Delta u(k) + B\Delta u(k+1)) \end{cases} \tag{3.13}$$

K+3:

$$\begin{cases} x(k+3) = Ax(k+2) + B\Delta u(k+2) \\ y(k+3) = Cx(k+3) \end{cases} \tag{3.14}$$

By substituting equation 3.13 in 3.14, we have

$$\begin{cases} x(k+3) = A^2x(k+1) + AB\Delta u(k+1) + B\Delta u(k+2) \\ y(k+3) = C(A^2x(k+1) + AB\Delta u(k+1) + B\Delta u(k+2)) \end{cases} \tag{3.15}$$

Now we substitute equation 3.11 in equation 3.15 to remove the term x(k+1).

$$\begin{cases} x(k+3) = A^2x(k+1) + AB\Delta u(k+1) + B\Delta u(k+2) \\ y(k+3) = C(A^2x(k+1) + AB\Delta u(k+1) + B\Delta u(k+2)) \end{cases} \tag{3.16}$$

This process continues recursively until the predicted value of the n_y step. Therefore, the output for n_y step can be expressed as:

$$\mathbf{y(k+n_y)} = C\left[A^{n_y}x(k) + A^{n_y-1}B\Delta u(k+1) + A^{n_y-2}B\Delta u(k+2) + \cdots + B\Delta u(k+n_y-1)\right] \tag{3.17}$$

Future output projections are shown as vectors below.

$$\mathbf{Y} = \left[y(k+1) \quad y(k+2) \quad y(k+3) \quad \cdots \quad y(k+n_y)\right]^T \tag{3.18}$$

The output prediction vector is compactly expressed as follows.

$$\begin{bmatrix} \mathbf{y(k+1)} \\ \mathbf{y(k+1)} \\ \mathbf{y(k+1)} \\ \vdots \\ \mathbf{y(k+1)} \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{n_y} \end{bmatrix} x(k) + \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{n_y-1}B & CA^{n_y-2}B & CA^{n_y-3}B & \cdots & CA^{n_y-n_u}B \end{bmatrix} \Delta U \quad (3.19)$$

This expression can be simplified as follows.

$$Y = Px(k) + H\Delta U \quad (3.20)$$

In above equation

$$\mathbf{P} = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{n_y} \end{bmatrix}; \mathbf{H} = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{n_y-1}B & CA^{n_y-2}B & CA^{n_y-3}B & \cdots & CA^{n_y-n_u}B \end{bmatrix}$$

3.6 Constraints

The last step before setting the cost function is to define the system constraints. In practice, all systems are subject to operational constraints such as limited dimensions and limited control capacity. In many situations, these limits are imposed intentionally and are as limited as possible to reduce power consumption, minimize resource usage, or simply reduce the size of a particular device. Constraints can be imposed on the control variable (also known as the input), u(k), the rate-of-change variable of the input, Δu(k), as well as the output, y(k), or the state variables, x(k). According to the quadcopter, which is controlled by changing the angular speed of each motor, the speeds vary from a minimum value of zero to a maximum value that depends on the motor specifications. This limitation is shown below.

$$0 < \omega_i < \omega_{\max}$$

where i=1,...4 represents each rotor.

In MPC design, only the constraints related to the control variables, u(k), and the rate of change of u(k) Δ are considered. The formulation of the constraints used in the design of the predictive controller is as follows.

$$u_{\min} \le u(k) \le u_{\max} \tag{3.21}$$

Since there are three control variables necessary to control the rotational motion of the quadcopter, each control variable is subject to the limitations shown below.

$$\begin{cases} u_1^{\min} \le u_1(k) \le u_1^{\max} \\ u_2^{\min} \le u_2(k) \le u_2^{\max} \\ u_3^{\min} \le u_3(k) \le u_3^{\max} \end{cases} \tag{3.22}$$

This method is also applicable to input change rates. The limits of the rate of change are as follows.

$$\begin{cases} \Delta u_1^{\min} \le \Delta u_1(k) \le \Delta u_1^{\max} \\ \Delta u_2^{\min} \le \Delta u_2(k) \le \Delta u_2^{\max} \\ \Delta u_3^{\min} \le \Delta u_3(k) \le \Delta u_3^{\max} \end{cases} \tag{3.23}$$

Constraints on input change rates can be grouped into a variable ΔU to account for subsequent sample times. So we have k sample times.

$$\Delta U(k)^{\min} \le \Delta U(k) \le \Delta U(k)^{\max} \tag{3.24}$$

Each one is defined as follows.

$$\mathbf{\Delta U(k)^{min}} = \begin{bmatrix} \Delta u_1^{\min} \\ \Delta u_2^{\min} \\ \Delta u_3^{\min} \end{bmatrix}, \mathbf{\Delta U(k)} = \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \end{bmatrix}, \mathbf{\Delta U(k)^{max}} = \begin{bmatrix} \Delta u_1^{\max} \\ \Delta u_2^{\max} \\ \Delta u_3^{\max} \end{bmatrix} \tag{3.25}$$

Moreover, these constraints can be divided into separate inequalities. Considering the sample times, the following expression is used to represent the maximum rate of change of the input.

$$\begin{bmatrix} \Delta U(k) \le \Delta U^{\max} \\ \Delta U(k+1) \le \Delta U^{\max} \\ \vdots \\ \Delta U(k+n_{u-1}) \le \Delta U^{\max} \end{bmatrix} \tag{3.26}$$

We also have an input for the minimum rate of change.

$$\begin{bmatrix} -\Delta U(k) \le -\Delta U^{\min} \\ -\Delta U(k+1) \le -\Delta U^{\min} \\ \vdots \\ -\Delta U(k+n_{u-1}) \le -\Delta U^{\min} \end{bmatrix} \tag{3.27}$$

These constraints are generalized in the equation below.

$$\begin{bmatrix} -\Delta U \le -\Delta U^{\min} \\ \Delta U \le \Delta U^{\max} \end{bmatrix} \tag{3.28}$$

$\Delta U, \Delta U^{\min}, \Delta U^{\max}$ are vectors with the number of elements corresponding to the size of the control horizon, in the number of rate of change constraints of the input. The above expression in matrix form is as follows.

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U \le \begin{bmatrix} -\Delta U^{\min} \\ \Delta U^{\max} \end{bmatrix} \tag{3.29}$$

Input constraints can be written in a form that accommodates input change rates. For this, we act as follows.

$$\begin{bmatrix} \mathbf{u(k)} \\ \mathbf{u(k+1)} \\ \vdots \\ \mathbf{u(k+n_u-1)} \end{bmatrix} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} u(k-1) + \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ I & I & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I & I & \cdots & I & I \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+n_u-1) \end{bmatrix} \tag{3.30}$$

The above equation can be shown in a compact form as follows.

$$\begin{cases} -(C_1u(k-1)+C_2\Delta U) \leq -U^{\min} \\ (C_1u(k-1)+C_2\Delta U) \leq -U^{\max} \end{cases} \tag{3.31}$$

$$\mathbf{C_1} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}, \mathbf{C_2} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ I & I & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I & I & \cdots & I & I \end{bmatrix} \tag{3.32}$$

Control input and input change rates can be grouped as follows.

$$\begin{bmatrix} C_u \\ C_{\Delta u} \end{bmatrix} \Delta U \leq \begin{bmatrix} d_u \\ d_{\Delta u} \end{bmatrix} \tag{3.33}$$

$$\mathbf{C_u} = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix}; \mathbf{d_u} = \begin{bmatrix} -U^{\min} + C_1u(k-1) \\ U^{\max} - C_1u(k-1) \end{bmatrix}; \mathbf{C_{\Delta u}} = \begin{bmatrix} -I \\ I \end{bmatrix}; \mathbf{d_{\Delta u}} = \begin{bmatrix} -\Delta U^{\min} \\ \Delta U^{\max} \end{bmatrix} \tag{3.34}$$

For simplification, equation 3.33 can be written as follows.

$$CC\Delta U \leq d \tag{3.35}$$

$$\mathbf{CC} = \begin{bmatrix} C_{\Delta u} \\ C_u \end{bmatrix}, \mathbf{d} = \begin{bmatrix} d_{\Delta u} \\ d_u \end{bmatrix} \tag{3.36}$$

3.7 Optimisation

As mentioned before, the purpose of predictive control is to direct the predicted outputs to the desired reference values. This objective is achieved by minimizing a cost function that includes the error between the predicted outputs and the reference values, subject to system constraints. The result of this optimization is an input series, ΔU, which is used to guide the predicted outputs to the reference. At sample time, k, output predictions are generated according to the size of the prediction horizon. For example, for a system with a prediction horizon value of 8, a series of 8 output predictions is generated. Given that at each sample time, there is only one set of reference values, r(k), and the cost function of errors relates between the reference values and the predicted outputs, it is assumed that for each optimization interval, Reference values are fixed. This is shown in Equation 3.37 below, which is a unitary matrix with dimensions corresponding to the number of outputs. The number of unit matrices is equivalent to the size of the prediction horizon. In this regard, the same matrices convert the desired reference path into the R matrix according to the prediction horizon of the control system. The matrix R, which is the matrix of our reference values in our desired prediction horizon, is brought in the relation of the cost function as a reference path to calculate the difference of its values with the predicted outputs of the system error and then optimize it. The next input values of the system will make the output of the system closer to the desired values. As a result, it can be said that the matrix R contains the values we want in the prediction horizon and the new entries are calculated according to them and the error obtained.

$$\mathbf{R}_s = \begin{bmatrix} \mathbf{I}_{no \times no} \\ \mathbf{I}_{no \times no} \\ \vdots \\ \mathbf{I}_{no \times no} \end{bmatrix}_{n_u \times 1} \mathbf{r(k)} \tag{3.37}$$

$$J = (\mathbf{R}_s - \mathbf{Y})^T (\mathbf{R}_s - \mathbf{Y}) + \boldsymbol{\Delta}\mathbf{U}^T \mathbf{W} \boldsymbol{\Delta}\mathbf{U} \tag{3.38}$$

The first term in the cost function minimizes the errors between the predicted output and the reference value while the second term is associated with minimizing the size of ΔU. W is the diagonal weight matrix for control inputs. A weight matrix with low values indicates that the focus of minimizing the cost function above is on the error between the reference values and the predicted output. This results in large ΔU values and this is equivalent to fast control as the ΔU values decrease rapidly. However a weight matrix with high diagonal values places importance on minimizing ΔU values, and this results in slower control because ΔU values decrease more slowly.

By substituting the equation 3.20 in 3.38

$$J = (\mathbf{R}_s - \mathbf{P}x(k))^T (\mathbf{R}_s - \mathbf{P}x(k)) - 2\boldsymbol{\Delta}\mathbf{U}^{\mathbf{T}}\mathbf{H}^{\mathbf{T}}(\mathbf{R}_s - \mathbf{P}x(k)) + \boldsymbol{\Delta}\mathbf{U}^{\mathbf{T}}(\mathbf{H}^{\mathbf{T}}\mathbf{H} + \mathbf{W})\boldsymbol{\Delta}\mathbf{U} \tag{3.39}$$

3.8 Quadratic Programming Formulation

Taking the first derivative of the cost function in equation 3.39, the following equation is obtained.

$$J = \frac{1}{2}\mathbf{\Delta U^T E \Delta U} + \mathbf{\Delta U^T F} \tag{3.40}$$

With constraints $\mathbf{C C \Delta U} \le \mathbf{d}$

$$\mathbf{E} = 2(\mathbf{H^T H} + \mathbf{W}), \mathbf{F} = -2\mathbf{H^T}(\mathbf{R_s} - \mathbf{P}x(k))$$

Considering the number of matrix and vector operations that must be performed in MPC, it is important to choose an efficient method for solving the quadratic programming problem in Equation 3.40. Linear MPC generally leads to the solution of structured convex quadratic programs. The convexity of the quadratic program ensures that a general solution to the problem is achievable. Two methods are usually used in solving quadratic problems, which are the interior point method (IPM) and the active set method (ASM). These methods differ in their approach to handling linear inequalities, where our inequalities are system constraints. Before describing these methods, the necessary conditions that must be met in the minimization of the objective functions with inequality constraints will be stated. These conditions are known as KKT conditions. For simplicity of notation, the following objective function will be used.

$$J = \frac{1}{2}x^T \mathbf{E}x + x^T \mathbf{F} \tag{3.41}$$

When $Mx \le \gamma$. Also, the conditions of KKT are as follows.

$$\begin{aligned} &\mathbf{E}x + \mathbf{F} + M^T \lambda = 0 \\ &Mx - \gamma \le 0 \\ &\lambda^T (Mx - \gamma) = 0 \\ &\lambda \ge 0 \end{aligned} \tag{3.42}$$

where λ is the vector of Lagrange multipliers. Considering to represent an indicative set of active constraints, the condition is now expressed as

$$\begin{aligned}
&\mathbf{E}x + \mathbf{F} + \sum_{i \in S_{act}} \lambda_i M_i^T = 0 \\
&M_i x - \gamma_i = 0, i \in S_{act} \\
&M_i x - \gamma_i < 0, i \notin S_{act} \\
&\lambda_i \geq 0, i \in S_{act} \\
&\lambda_i = 0, i \notin S_{act}
\end{aligned} \tag{3.43}$$

Where M_i is the i-th row is the matrix M. An equality constraint indicates that the constraint is active. However, the constraint indicates that the constraint is satisfied. If a Lagrange coefficient is zero, the constraint is inactive, while a non-negative Lagrange coefficient indicates that the constraint is active. Interior point methods approach the KKT condition for the quadratic program inequality problem using successive descent steps. Each descent step is obtained by Newton's method for optimization, which forms a linear system to be factored and solved. The active set method is more suitable for solving the quadratic problem of the preinterlinear controller. The active set methods define a set of constraints called the working set, which is treated as the active set. A working set is a subset of the constraints that are active at the current point. The current point is an interior solution to the original problem if all Lagrange coefficients, λi ≥ 0. Otherwise, if there is λi < 0, the corresponding constraint is removed from the constraint equation. Active set methods belong to a group of methods known as primitive methods. Since active set methods require the identification of active constraints, this can create a large computational burden if the constraints are large. To deal with this problem, a dual method can be used to systematically remove the constraints. The derivation of the dual problem of the original problem is given below.

$$\min_{\lambda \geq 0} \min_{x} \left[\frac{1}{2} x^T \mathbf{E} x + x^T \mathbf{F} + \lambda^T (Mx - \gamma) \right] \tag{3.44}$$

By minimizing x

$$x = -\mathbf{E}^{-1}(\mathbf{F} + M^T \lambda) \tag{3.45}$$

By substituting in equation 3.44

$$\min_{\lambda \geq 0} \left(-\frac{1}{2} \lambda^T \mathbf{T} \lambda - \lambda^T \mathbf{K} - \frac{1}{2} \gamma^T \mathbf{E}^{-1} \gamma\right) \tag{3.46}$$

In the above equation, the T and K matrices are

$$\begin{aligned}\mathbf{T} &= M\mathbf{E}^{-1}M^{T} \\ \mathbf{K} &= \gamma + M\mathbf{E}^{-1}\mathbf{F}\end{aligned} \qquad (3.47)$$

Maximizing an objective function is equivalent to minimizing the negative of that objective function. Therefore, equation 3.46 becomes as follows.

$$\min_{\lambda\geq 0}\left(\frac{1}{2}\lambda^{T}\mathbf{T}\lambda + \lambda^{T}\mathbf{K} + \frac{1}{2}\gamma^{T}\mathbf{E}^{-1}\gamma\right) \qquad (3.48)$$

To solve equation 3.48, there is a quadratic algorithm called Hildreth, which is explained below in the MATLAB programming section of the predictive controller.

3.9 MPC Design in MATLAB

Now, after reviewing the principles and steps of designing the predictive controller, the explained steps in MATLAB in the form of commands, and using a series of defined functions and the functions that we will define, the predictive controller is designed for the rotary motion system of the quadcopter. do. For this purpose, we first need to have the system state space. According to the equations of the rotational motion system of the bird, the linear state space will be as follows. The state space matrices are as follows.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.49)$$

The states and inputs of the system will be as follows.

$$\mathbf{x} = \begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} LK_f(\omega_4^2 - \omega_2^2) \\ LK_f(\omega_1^2 - \omega_3^2) \\ K_m(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{bmatrix} \tag{3.50}$$

We also need the values and parameters of the dynamic system to design the controller. For this purpose, according to one of the references, the values of the dynamic system of the quadcopter are as follows.

parameters	values
Mass(m)	1.587 Kg
Length(L)	0.243 m
Thrust coefficient	$4.0687\times10^{-7} N\ / rpm^2$
Drag coefficient	$8.4367\times10^{-9} Nm\ / rpm^2$
Moment of inertia about x-axis(I_{xx})	$0.0213 kmg^2$
Moment of inertia about y-axis(I_{yy})	$0.02217 kmg^2$
Moment of inertia about y-axis(I_{zz})	$0.0282 kmg^2$

Table 3.1: Dynamic system parameters

Now, according to the obtained state space and system values, we design our controller.

3.9.1 State Matrices and Vectors

The MPC controller uses a linear discrete state space model. MATLAB function cd2m was used to convert the continuous state space model to a discrete model. This method is shown below.

```
[A_m, B_m, C_m, D_m] = c2dm(A, B, C, D, ts);
```

The function takes continuous state and sample time matrices as arguments and returns discrete state space matrices. Matrices 3.51 show the state space model of the linear discrete model and matrices 3.52 show the matrices of the augmented state space model. To obtain these equations, a sample time of 0.2 seconds was used, and the values of moments of inertia from Table 3.1 have been replaced.

$$\mathbf{A_m} = \begin{bmatrix} 1 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B_m} = \begin{bmatrix} 0.939 & 0 & 0 \\ 9.3897 & 0 & 0 \\ 0 & 0.9021 & 0 \\ 0 & 9.0212 & 0 \\ 0 & 0 & 0.7092 \\ 0 & 0 & 7.0922 \end{bmatrix} \tag{3.51}$$

$$\mathbf{C_m} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The augmented mode space matrices of the quadcopter are as follows.

$$\mathbf{A} = \begin{bmatrix} 1 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0.939 & 0 & 0 \\ 9.3897 & 0 & 0 \\ 0 & 0.9021 & 0 \\ 0 & 9.0212 & 0 \\ 0 & 0 & 0.7092 \\ 0 & 0 & 7.0922 \\ 9.3897 & 0 & 0 \\ 0 & 9.0212 & 0 \\ 0 & 0 & 7.0922 \end{bmatrix} \tag{3.52}$$

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

The state and input/control vectors will be as follows.

$$\mathbf{x} = \begin{bmatrix} \Delta x_m \\ y \end{bmatrix}, \mathbf{\Delta u} = \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \end{bmatrix} \tag{3.53}$$

$$\mathbf{\Delta x_m} = \begin{bmatrix} \Delta\phi & \Delta\dot{\phi} & \Delta\theta & \Delta\dot{\theta} & \Delta\psi & \Delta\dot{\psi} \end{bmatrix}^T$$

3.9.2 Constraints

The limitations of the quadcopter system are the maximum speed of the motors. We assume that the quadcopter under review has four EMAX MT3506 650 kv brushless motors, the maximum speed of the motors is 4720 rpm. The constraints used in the MPC formula were input constraints and input change rates. To determine these limits, the total thrust produced by the engines must be calculated. According to Table 3.1, the mass of the quadcopter is equal to 1.587 kg. Assuming that gravity is 9.81, we have

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

The thrust is obtained from the following equation.

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

According to Table 3.1, the speed value of each quadcopter rotor is obtained from the following equation.

$$15.57 = K_f \sum_{i=1}^{4} \omega_i^2 \rightarrow \omega_i = \sqrt{\frac{15.57}{4 \times 4.0687 \times 10^{-7}}} \approx 3093 rpm \tag{3.54}$$

As discussed earlier, the input for the rotary motion of the quadcopter is the torque in the roll, pitch, and yaw axes. These torque equations are given below.

$$\begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \mathbf{U}_3 \end{bmatrix} = \begin{bmatrix} K_f L(\omega_4^2 - \omega_2^2) \\ K_f L(\omega_1^2 - \omega_3^2) \\ K_m(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{bmatrix} \tag{3.55}$$

where U1, U2 and U3 are $\tau_\phi, \tau_\theta, \tau_\psi$ torque respectively.

The input limits are the maximum and minimum torque and are calculated below. For example, in calculating the maximum torque of the roll movement, $\tau_{\phi\max}$, the maximum speed of the motor depends on ω_4 and the minimum speed of the motor depends on ω_2. The maximum and minimum engine speed is 4720 rpm and 3093 rpm, respectively. The maximum torques are calculated below.

$$\begin{cases} \boldsymbol{\tau}_{\phi\mathbf{max}} = 0.243\times 4.0687\times 10^{-7}(4720^2 - 3093^2) = 1.257 \\ \boldsymbol{\tau}_{\boldsymbol{\theta}\mathbf{max}} = 0.243\times 4.0687\times 10^{-7}(4720^2 - 3093^2) = 1.257 \\ \boldsymbol{\tau}_{\psi\mathbf{max}} = 8.4367\times 10^{-9}(4720^2 + 4720^2 - 3093^2 - 3093^2) = 0.2145 \end{cases} \tag{3.56}$$

Now we write the values of the maximum torques in the following vector.

$$\mathbf{\Delta U^{max}} = \begin{bmatrix} 1.257 & 1.257 & 0.2145 \end{bmatrix}^T \tag{3.57}$$

The minimum torque values are the negative of the maximum values. So the vector of minimum moments is equal to.

$$\mathbf{\Delta U^{min}} = \begin{bmatrix} -1.257 & -1.257 & -0.2145 \end{bmatrix}^T \tag{3.58}$$

The input change rate is considered to be 60% of the input, which is to ensure the stable operation of the system.

$$\begin{aligned} \mathbf{\Delta U^{max}} &= \begin{bmatrix} 0.7542 & 0.7542 & 0.1287 \end{bmatrix}^T \\ \mathbf{\Delta U^{min}} &= \begin{bmatrix} -0.7542 & -0.7542 & -0.1287 \end{bmatrix}^T \end{aligned} \tag{3.59}$$

At a particular sample instant, k, the current rate of change of the input, Δu(ki) Each input variable is bounded by the maximum and minimum rates of change of the input, as shown in the following equations.

$$\begin{cases} \Delta u_1^{\min} \le \Delta u_1 \le \Delta u_1^{\max} \\ \Delta u_2^{\min} \le \Delta u_2 \le \Delta u_2^{\max} \\ \Delta u_3^{\min} \le \Delta u_3 \le \Delta u_3^{\max} \end{cases} \tag{3.60}$$

The above equation is compressed in the following form.

$$\Delta u_i^{\min} \le \Delta u(k) \le \Delta u_i^{\max} \tag{3.61}$$

where i represents inputs.

For example, choosing a control horizon with a value of 3, the future control path at the sample instant, k, is shown as follows.

$$\left[\Delta U(k) \quad \Delta U(k+1) \quad \Delta U(k+2)\right]^T \tag{3.62}$$

Each element of this control path consists of three variable rates of input change, each of which is bounded in a similar manner to Equations 3.60. Therefore, restrictions are imposed on the control path as follows.

$$\begin{cases} \Delta u_i^{\min} \le \Delta u_i(k) \le \Delta u_i^{\max} \\ \Delta u_i^{\min} \le \Delta u_i(k+1) \le \Delta u_i^{\max} \\ \Delta u_i^{\min} \le \Delta u_i(k+2) \le \Delta u_i^{\max} \end{cases} \tag{3.63}$$

The above relation can be compressed as follows.

$$\Delta U^{\min} \le \Delta U \le \Delta U^{\max} \tag{3.64}$$

According to the explanations given in the section, we anticipate the limitations of the controller.

$$\begin{gathered} C_{\Delta u}\Delta U \le d_{\Delta u} \\ \mathbf{C}_{\mathbf{\Delta u}} = \begin{bmatrix} -I \\ I \end{bmatrix}, \mathbf{d}_{\mathbf{\Delta u}} = \begin{bmatrix} -\Delta U^{\min} \\ \Delta U^{\max} \end{bmatrix} \end{gathered} \tag{3.65}$$

Using control horizon 2, in the MPC model, the unit matrix dimension is six. With the constraints defined earlier in this section, Equation 3.65 expands as follows.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \Delta u_1(k) \\ \Delta u_2(k) \\ \Delta u_3(k) \\ \Delta u_1(k+1) \\ \Delta u_2(k+1) \\ \Delta u_3(k+1) \end{bmatrix} \le \begin{bmatrix} 0.7542 \\ 0.7542 \\ 0.1287 \\ 0.7542 \\ 0.7542 \\ 0.1287 \\ -0.7542 \\ -0.7542 \\ -0.1287 \\ -0.7542 \\ -0.7542 \\ -0.1287 \end{bmatrix} \tag{3.66}$$

Also, the input restrictions explained in the restrictions section are defined as follows.

$$C_u \Delta U \le d_u$$
$$\mathbf{C_u} = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix}, \mathbf{d_u} = \begin{bmatrix} -U^{\min} + C_1 u(k-1) \\ U^{\max} - C_1 u(k-1) \end{bmatrix} \tag{3.67}$$
$$\mathbf{C_1} = \begin{bmatrix} I \\ I \end{bmatrix}, \mathbf{C_2} = \begin{bmatrix} I & 0 \\ I & I \end{bmatrix}$$

Entry restrictions will be as follows.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \Delta u_1(k) \\ \Delta u_2(k) \\ \Delta u_3(k) \\ \Delta u_1(k+1) \\ \Delta u_1(k+1) \\ \Delta u_1(k+1) \end{bmatrix} \le \begin{bmatrix} 1.257 - u_1(k-1) \\ 1.257 - u_3(k-1) \\ 0.2145 - u_3(k-1) \\ 1.257 - u_1(k-1) \\ 1.257 - u_2(k-1) \\ 0.2145 - u_3(k-1) \\ -1.257 + u_1(k-1) \\ -1.257 + u_2(k-1) \\ -0.2145 + u_3(k-1) \\ -1.257 + u_1(k-1) \\ 1.257 + u_2(k-1) \\ -0.2145 + u_3(k-1) \end{bmatrix} \tag{3.68}$$

Now we put the input limits and the input rate in a equation, which will be as follows.

$$\mathbf{CC\Delta U} \le \mathbf{d}$$

$$\mathbf{CC} = \begin{bmatrix} C_{\Delta u} \\ C_u \end{bmatrix}, \mathbf{d} = \begin{bmatrix} d_{\Delta u} \\ d_u \end{bmatrix}$$

Chapter 4

MATLAB Implementation

MATLAB simulation starts with the initialization of the required parameters in the continuous state space model and the controller. These include moments of inertia for each axis, quadcopter thrust Kf and drag coefficient Kd, maximum and minimum engine speed, control horizon nu, prediction horizon ny, number of inputs, number of outputs, number of states, initial state x, initial output y, referential Traces are the adjusted reference matrix Rs, the perturbation and the number of simulations Nsim. The following steps show the design path of the predictive controller.

4.1 Dynamic System and MPC Parameters

First, we must write the initial values and important and basic parameters of the dynamic system of the quadcopter and the predictive controller in the program. First, we will write the dynamic parameters, which include the values of inertia, mass, lift and drag coefficients, and the minimum and maximum speed of the engines. According to Table 3.1, we have the values of the dynamic system.

```
% quadcopter dynamic parameters

m=1.587; %mass
g=9.81; %gravity
Ixx=0.0213; %x-axis moment of inertia
Iyy=0.02217; %y-axis moment of inertia
Izz=0.0282; %z-axis moment of inertia
b=4.0687e-7; %thrust coefficient
k=8.4367e-9; %drag coefficient
L=0.243; %distance between the rotor and the center of mass
max_speed=4720; %max motor speed
min_speed=3093; %min take off speed
```

Then we write the initial values of our controller, which include the number of states, the number of inputs, the number of outputs, the value of the prediction horizon, the value of the control horizon, and the initial values of the states, outputs, and inputs. The number of modes according to the controller is 6, and the number of inputs and outputs of the controller is 3. To design the controller, we take the control horizon equal to the number of outputs, i.e. 3, and the value of the prediction horizon equal to the number of nodes, i.e. 6. Of course, these values can be changed and by changing these parameters, we can optimize the output of our controller. We also consider the initial values of states, inputs, and outputs to be zero.

```
% MPC controller parameters

n_states=6; %number of states
n_inputs=3; %number of inputs
n_outputs=3; %number of outputs
nu=3; %control horizon
ny=6; %prediction horizon
n_sim=20; %number of simulation
x=zeros(6,1); %initial state values
y=zeros(3,1); %initial output values
u=[0;0;0]; %initial control values
t=0:0.2:n_sim; %step for sinusoidal reference
```

4.2 Define the Reference Path

Now, in the next step, we will define our reference path. The reference paths for quadcopter rotational movements include a sine path, a path with sine and cosine subordination, and a helix path, which creates 100 path samples in a time interval. Also, according to equation 3.37 and the explanation of that section, the reference matrix R will be a single matrix with the dimensions of the output value, which has the same matrix as the number of prediction horizons. This matrix is defined as follows. A perturbation is also defined that is applied to the output.

```
%reference to be tracked
% sin and cos reference
r=[sin(t)+cos(3*t)/2;sin(t)+cos(3*t)/2;sin(t)+cos(2*t)/2+sin(3*t)/3];
% sin path reference
% r=[sin(t);sin(t);sin(t)];
% helix path reference
% r=[2*cos(t);2*sin(t);t/(2*pi)];
Rs=[eye(n_outputs);eye(n_outputs);eye(n_outputs);...
    eye(n_outputs);eye(n_outputs);eye(n_outputs)]; %reference adjusting matrix
dist=(rand(3,101)*2-1)*0.5; %disturbance
```

4.3 Augmented State Space Matrices

In the next step, we must define the system state space. According to equation 3.49, we define the state space of the system. Then we convert our state space to a discrete state using the c2dm command with a sampling rate of 0.2 seconds. After we have obtained the discrete state of our system, we need to obtain the augmented matrix state of the state space, which is done using the function defined augment_mimo. This function takes the discrete state space matrices as input and creates our augmented state space matrices in the form described in Section 3.3 and Equation 3.6.

```
%state space matrices
A=[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];

B=[0 0 0; 1/Ixx 0 0; 0 0 0;0 1/Iyy 0;0 0 0;0 0 1/Izz];
C=[0 1 0 0 0 0; ...
    0 0 0 1 0 0; ...
    0 0 0 0 0 1];

D=zeros(3,3);

%continuous to discrete time convertion
ts=0.2; %sampling time
[Ad,Bd,Cd,Dd]=c2dm(A,B,C,D,ts);

%augmented model
[Aa,Ba,Ca]=augment_mimo(Ad,Bd,Cd,n_states,n_inputs,n_outputs);
```

The augment_mimo function is defined as follows. As we can see, according to relation 3.6, our augmented matrix A consists of discrete matrix A and a zero matrix in the first row and the product of discrete matrices A and C and a unit matrix in the next row. The augmented matrix B is also composed of the discrete matrix B in the first row and the multiplication of the discrete matrices B and C in the second row. Also, the augmented matrix C is composed of a zero matrix and a unit matrix.

```
function [Aa,Ba,Ca]= augment_mimo(Ad,Bd,Cd,n_states,n_inputs,n_outputs)

Aa=[Ad zeros(n_states,n_inputs); Cd*Ad eye(n_outputs)];
Ba=[Bd;Cd*Bd];
Ca=[zeros(n_inputs,n_states) eye(n_inputs)];
```

4.4 Controllability

According to the explanations given in the next step, the controllability of the system should be checked after discretization and converting it to extension mode. According to the CO matrix given in relation 3.8, we have to check the controllability of the system. If the rank of the CO matrix was equal to the number of our inputs, then the system is controllable.

```
%controllability
CO=[Ba,Aa*Ba,Aa^2*Ba,Aa^3*Ba,Aa^4*Ba,Aa^5*Ba];
controllability=rank(CO);
if controllability==n_states
    disp("system is controllable")
end
```

4.5 Prediction Matrices

According to section 3.5, to obtain the predicted outputs, we need two matrices P and H. According to relation 3.20, these two matrices will be as follows.

$$\mathbf{P}=\begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \\ CA^6 \end{bmatrix}, \mathbf{H}=\begin{bmatrix} CB & 0_{3\times3} & 0_{3\times3} \\ CAB & CB & 0_{3\times3} \\ CA^2B & CAB & CB \\ CA^3B & CA^2B & CAB \\ CA^4B & CA^3B & CA^2B \\ CA^5B & CA^4B & CA^3B \end{bmatrix}$$

```
%prediction matrices

%output predictions

P=[Ca*Aa;Ca*Aa^2;Ca*Aa^3;Ca*Aa^4;Ca*Aa^5;Ca*Aa^6];

H=[Ca*Ba,zeros(nu),zeros(nu);...
    Ca*Aa*Ba,Ca*Ba,zeros(nu);...
    Ca*Aa^2*Ba,Ca*Aa*Ba,Ca*Ba;...
    Ca*Aa^3*Ba,Ca*Aa^2*Ba,Ca*Aa*Ba;...
    Ca*Aa^4*Ba,Ca*Aa^3*Ba,Ca*Aa^2*Ba;...
    Ca*Aa^5*Ba,Ca*Aa^4*Ba,Ca*Aa^3*Ba];
```

4.6 Constraints

In the next step, we must define the system's limitations. First, we need to get the input values of u1,u2,u3. The relations of system inputs are given in relation 3.55 and we obtain these values according to the minimum and maximum speed of our engines. After obtaining the inputs, we determine the maximum and minimum values of the inputs and then we also obtain the rate of change of the maximum input values, which is 60% of the maximum values. After obtaining the initial values of system constraints, according to relation 3.35, we obtain the matrices corresponding to this relation. These matrices are obtained using the defined function constraints_mimo, which will be explained below.

```
% contraints
u1=L*b*(max_speed^2-min_speed^2);
u2=u1;
u3=k*(max_speed^2+max_speed^2-min_speed^2-min_speed^2);
u_max=[u1;u2;u3];
u_min=-u_max;
Dumax=0.6*u_max; %rate of input changes

%constraint matrices
[CC,dd,dupast]=constraints_mimo(Dumax,u_max,u_min,n_inputs,nu);
```

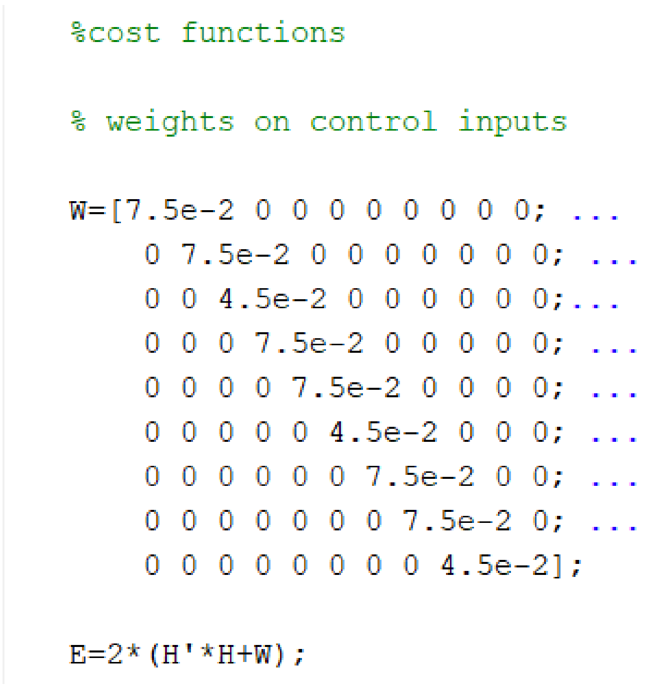
In the constraints_mimo function, the maximum and minimum values of the inputs, the values of the rate of change of the inputs, the number of inputs, and the value of the control horizon are taken as inputs, and according to the equation 3.35, two important matrices CC and d are obtained. According to the given dimensions and definitions for both matrixes, we have system constraints.

```
function [CC,dd,dupast] = constraints_mimo(Dumax,u_max,u_min,n_inputs,nu)
CC=[];
dim = nu*n_inputs; %matrices dimensions
CC(1:dim,1:dim)=eye(dim);
CC(dim+1:2*dim,1:dim) = -eye(dim);
s=0;
E=zeros(dim,dim);
for j=1:nu
    E(1+s:n_inputs*j,1:n_inputs) = eye(n_inputs);
    s=s+n_inputs;
end
p=n_inputs+1;
for counter = 1:nu-1
    i=counter;
    for k = (n_inputs*counter)+1:n_inputs:dim-n_inputs+1
        E(k:n_inputs*(i+1),p:n_inputs*(counter+1))= eye(n_inputs);
        i=i+1;
    end
    p=p+n_inputs;
end

Cu=[eye(dim);-eye(dim)];
CuE=Cu*E;
CC(2*dim+1:4*dim,1:dim)=CuE;
ddeltaU=zeros(2*dim,1);
t=1;
for i=1:n_inputs:size(ddeltaU,1)
    ddeltaU(i:t*n_inputs,1) = Dumax;
    t=t+1;
end
du=zeros(2*dim,1);
```

The above codes are for obtaining and calculating the CC matrix, which according to the dimensions of the system, the one and zero matrices are defined in the main CC matrix. In the following, the codes are related to the definition of matrix d, which are again defined according to the required dimensions of unit matrices and added to the main matrix d.
All relations and values are explained in section 3.9.2.

4.7 Input Weights and Simulation Loop Parameters

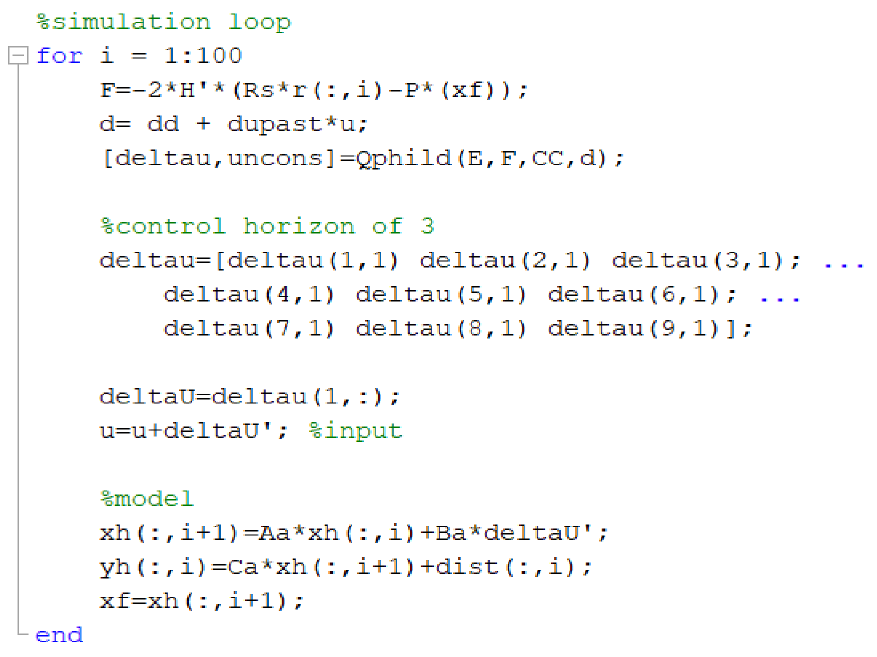
As mentioned before, we should minimize the relation 3.40. In this regard, the weight matrix W is a square diagonal matrix whose dimensions correspond to the size of the control horizon. For example, with three inputs and a control horizon of two, W would be a 6 x 6 matrix where the last three diagonal elements are repetitions of the first three elements. The weight matrix selected for the controller design is selected according to one of the references, which is by the characteristics of the system.
E and F are the final parameters required to perform the minimization of Equation 3.40. E is defined before the minimization loop while the F matrix is obtained in the loop as it is updated at the start of each iteration of the loop. The augmented state vector Xf, also defined before the minimization loop, is initialized as a zero vector with the number of rows equal to the number of rows of each augmented state matrix Aa or Ba.



```
%cost functions

% weights on control inputs

W=[7.5e-2 0 0 0 0 0 0 0 0; ...
    0 7.5e-2 0 0 0 0 0 0 0; ...
    0 0 4.5e-2 0 0 0 0 0 0;...
    0 0 0 7.5e-2 0 0 0 0 0; ...
    0 0 0 0 7.5e-2 0 0 0 0; ...
    0 0 0 0 0 4.5e-2 0 0 0; ...
    0 0 0 0 0 0 7.5e-2 0 0; ...
    0 0 0 0 0 0 0 7.5e-2 0; ...
    0 0 0 0 0 0 0 0 4.5e-2];

E=2*(H'*H+W);
```

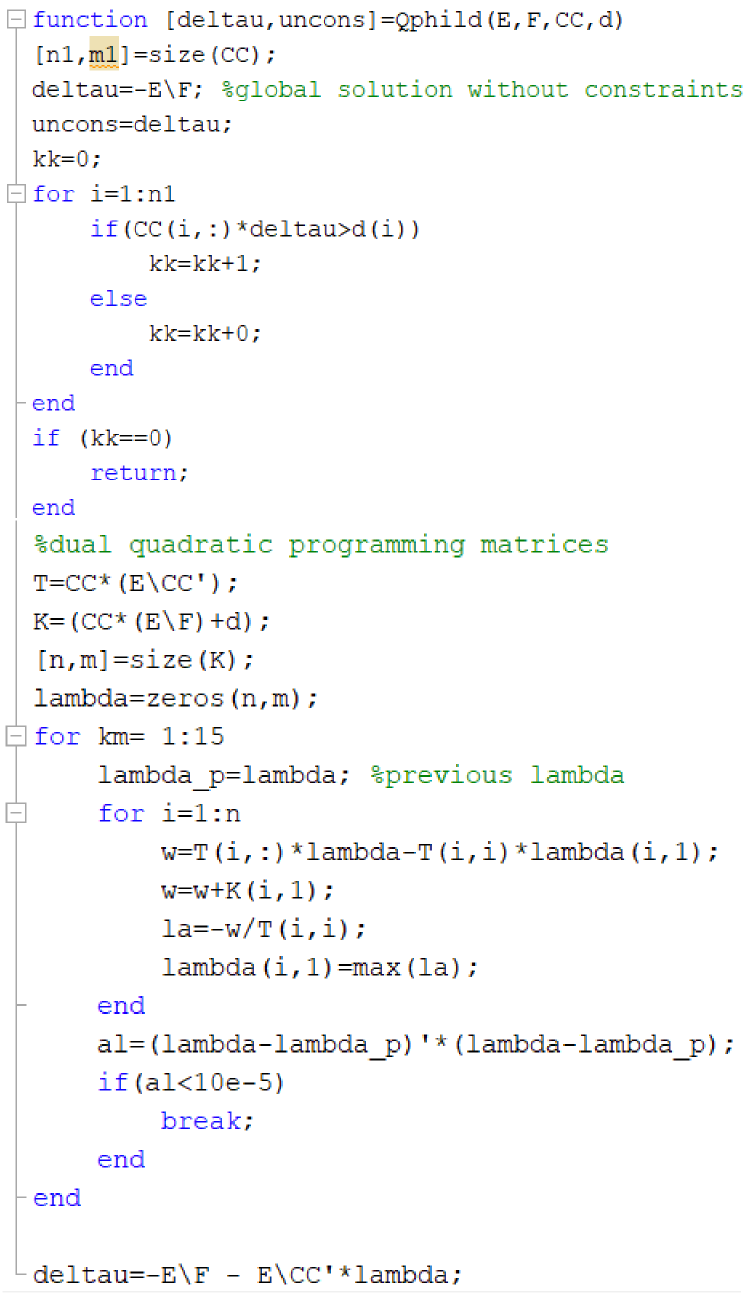
4.8 Minimisation Loop

After defining the weight matrix and obtaining the value of the E matrix, the F matrix and the system inputs should be obtained. To do this, we create a loop where the for loop variable, i, starts with an initialization of 1. After that, the variable F and the constraint vector d are updated before being sent to the defined Hildreth function QPhild(). In the code below, DeltaU is a solution to problem 3.40 and Unconst is also an unconstrained solution to the problem. DeltaU is the control path of the input actions, where the number of columns of this matrix is equal to the size of the control horizon. Only the first column of input values is implemented on the system. After extracting the first column from DeltaU, the next step is to update the model for the next loop iteration. where deltau is the first column of input values and xh is a state vector with the same dimensions as Xf and dist is the disturbance that is added to the output yh. To get the current input vector to send to the quadcopter, delta is added to the previous input vector as shown below.



```
%simulation loop
for i = 1:100
    F=-2*H'*(Rs*r(:,i)-P*(xf));
    d= dd + dupast*u;
    [deltau,uncons]=Qphild(E,F,CC,d);

    %control horizon of 3
    deltau=[deltau(1,1) deltau(2,1) deltau(3,1); ...
        deltau(4,1) deltau(5,1) deltau(6,1); ...
        deltau(7,1) deltau(8,1) deltau(9,1)];

    deltaU=deltau(1,:);
    u=u+deltaU'; %input

    %model
    xh(:,i+1)=Aa*xh(:,i)+Ba*deltaU';
    yh(:,i)=Ca*xh(:,i+1)+dist(:,i);
    xf=xh(:,i+1);
end
```

The Hildreth function Qphild whose application was explained in Section 3.8 is defined as follows. This function takes the values of system constraints and matrices E and F as inputs and then delivers the system inputs as the final response of the optimization cost function in the output.



```
function [deltau,uncons]=Qphild(E,F,CC,d)
[n1,m1]=size(CC);
deltau=-E\F; %global solution without constraints
uncons=deltau;
kk=0;
for i=1:n1
    if(CC(i,:)*deltau>d(i))
        kk=kk+1;
    else
        kk=kk+0;
    end
end
if (kk==0)
    return;
end
%dual quadratic programming matrices
T=CC*(E\CC');
K=(CC*(E\F)+d);
[n,m]=size(K);
lambda=zeros(n,m);
for km= 1:15
    lambda_p=lambda; %previous lambda
    for i=1:n
        w=T(i,:)*lambda-T(i,i)*lambda(i,1);
        w=w+K(i,1);
        la=-w/T(i,i);
        lambda(i,1)=max(la);
    end
    al=(lambda-lambda_p)'*(lambda-lambda_p);
    if(al<10e-5)
        break;
    end
end

deltau=-E\F - E\CC'*lambda;
```

4.9 Output Simulation

Now, after designing the controller and obtaining the output values, we must define the reference path and our outputs. Using the plot command in MATLAB, we plot our values in three images, each image corresponding to one of the modes of the quadcopter rotation system.

```
% plots

 i=1:100;

% output1
subplot(3,1,1)
plot(i,r(1,i),'r',i,yh(1,i),'b')
legend('reference','output');
xlabel('time');
ylabel('phi')

% output2
subplot(3,1,2)
plot(i,r(2,i),'r',i,yh(2,i),'b')
legend('reference','output');
xlabel('time');
ylabel('theta')

% output3
subplot(3,1,3)
plot(i,r(3,i),'r',i,yh(3,i),'b')
legend('reference','output');
xlabel('time');
ylabel('psi')
```

The outputs of the system will be as follows.

```
r=[sin(t)+cos(3*t)/2;sin(t)+cos(3*t)/2;sin(t)+cos(
2*t)/2+sin(3*t)/3]
```

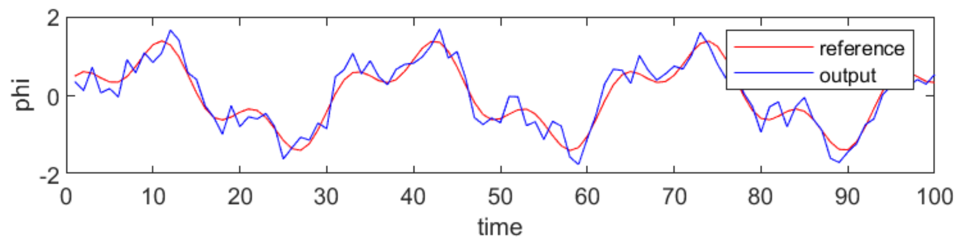


Figure 4.1: Roll motion output

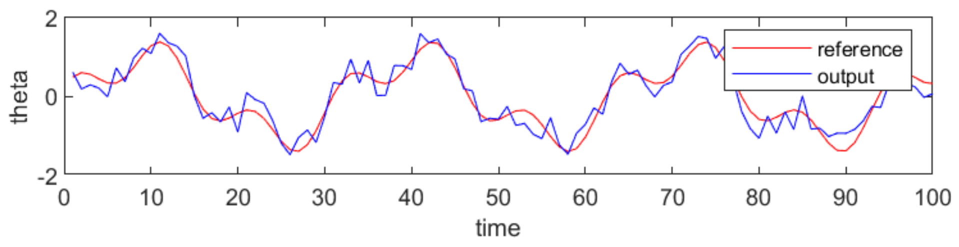


Figure 4.2: Pitch motion output

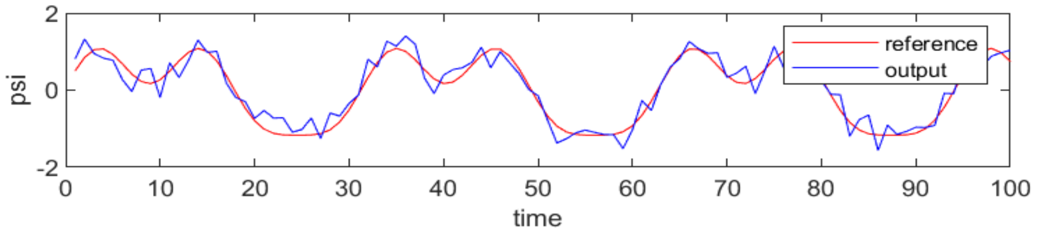


Figure 4.3: Yaw motion output

```
r=[sin(t);sin(t);sin(t)]
```

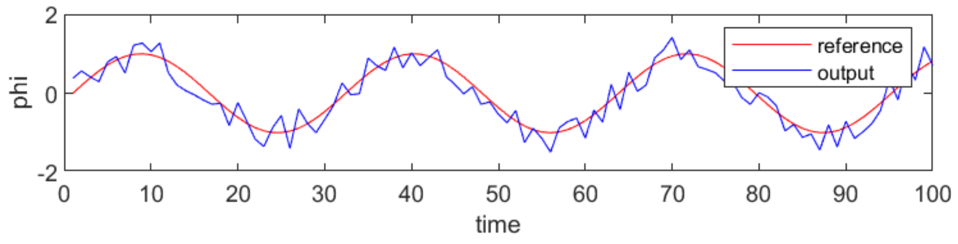


Figure 4.4: Roll motion output

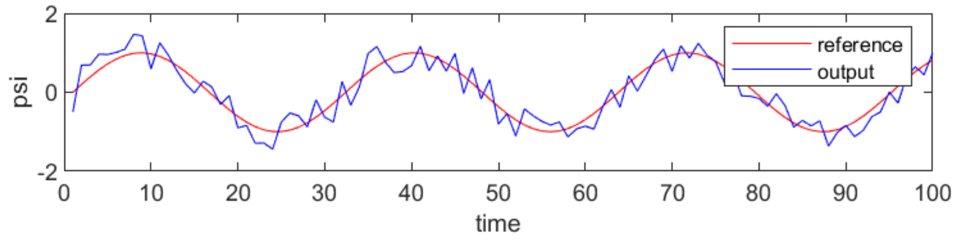


Figure 4.5: Pitch motion output

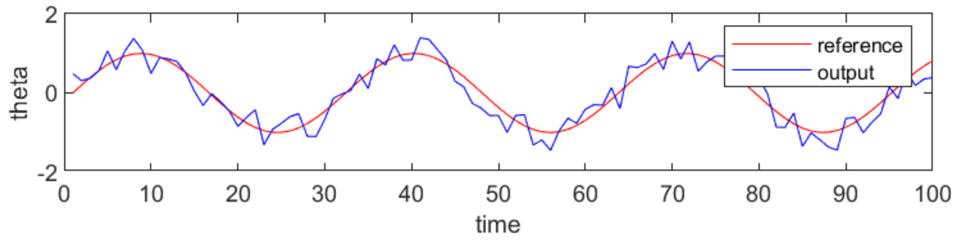


Figure 4.6: Yaw motion output

```
r=[2*cos(t);2*sin(t);t/(2*pi)];
```

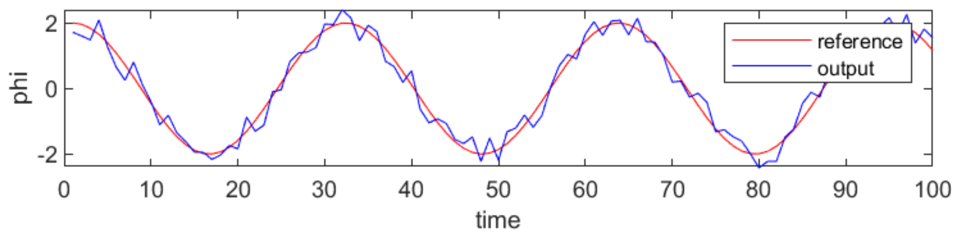


Figure 4.7: Roll motion output

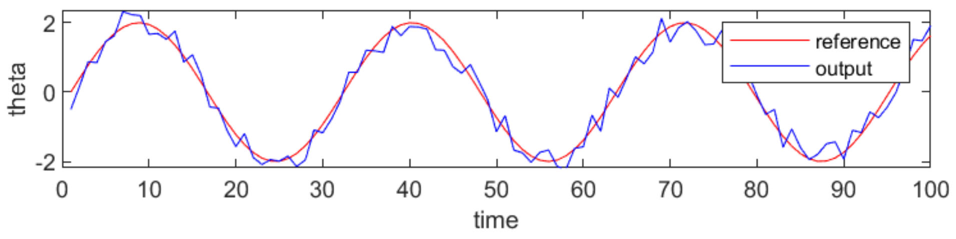


Figure 4.8: Pitch motion output

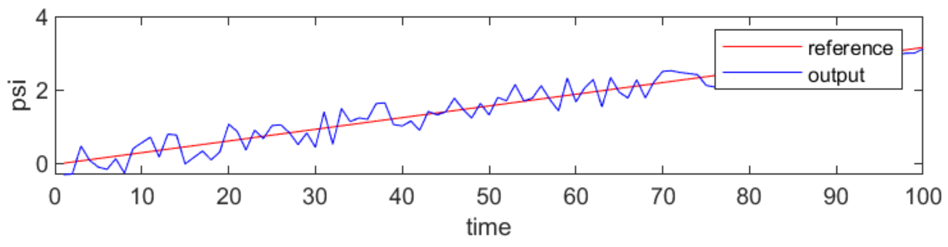


Figure 4.9: Yaw motion output

Chapter 5

Conclusion and Future

5.1 Conclusion

According to the outputs obtained in the simulation part of the controller, it can be concluded that the quadcopter performs its rotational movements stably and follows the reference path and the desired sinusoidal values of its rotational movements. As mentioned, the movement of quadcopters and other types of flying devices depends on their rotational movement, and this issue can also be seen according to the equations governing the flight system. So it can be concluded that according to the obtained outputs and regardless of the vertical flight movement, the overall movement of the quadcopter is desirable. The disturbances observed in the output are normal and are caused by the disturbances introduced into the system, which are added due to considering the environmental and internal noises of the system.

5.2 Future

Image processing and machine learning or deep learning can be used to continue this project. The reason for suggesting the use of these features is the visual tracking of the route and people. It can also be used by adding image processing to your quadcopter system. For example, after adding a camera, image processing, and machine learning algorithms to your quadcopter, it can be used in agriculture, relief, military affairs, surveillance, and security.

References

[1] MPC-Based Attitude Control of Quadcopter

[2] Design and Implementation of Model Predictive Control on Pixhawk Flight Controller

[3] Jan Maciejowski - Predictive Control with Constraints-Prentice Hall (2000)

[4] Fundamentals-of-aerodynamics-6-Edition

[5] Model Predictive Control System Design and Implementation Using MATLAB®

[6] J . L . Meriam, L.G. Kraige - Engineering Mechanics_ Dynamics. 8th Edition-Wiley (2015)

Appendices

A. MATLAB Code

MPC_controller

```
% quadcopter dynamic parameters

m=1.587; %mass
g=9.81; %gravity
Ixx=0.0213; %x-axis moment of inertia
Iyy=0.02217; %y-axis moment of inertia
Izz=0.0282; %z-axis moment of inertia
b=4.0687e-7; %thrust coefficient
k=8.4367e-9; %drag coefficient
L=0.243; %distance between the rotor and the
center of mass
max_speed=4720; %max motor speed
min_speed=3093; %min take off speed

% MPC controller parameters

n_states=6; %number of states
n_inputs=3; %number of inputs
n_outputs=3; %number of outputs
nu=3; %control horizon
ny=6; %prediction horizon
n_sim=20; %number of simulation
x=zeros(6,1); %initial state values
y=zeros(3,1); %initial output values
u=[0;0;0]; %initial control values
t=0:0.2:n_sim; %step for sinusoidal reference

%reference to be tracked
% sin and cos reference
r=[sin(t)+cos(3*t)/2;sin(t)+cos(3*t)/2;sin(t)+cos(
2*t)/2+sin(3*t)/3];
% sin path reference
% r=[sin(t);sin(t);sin(t)];
% helix path reference
% r=[2*cos(t);2*sin(t);t/(2*pi)];
Rs=[eye(n_outputs);eye(n_outputs);eye(n_outputs);.
..
```

```
    eye(n_outputs);eye(n_outputs);eye(n_outputs)];
%reference adjusting matrix
dist=(rand(3,101)*2-1)*0.5; %disturbance

%state space matrices
A=[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];

B=[0 0 0; 1/Ixx 0 0; 0 0 0;0 1/Iyy 0;0 0 0;0 0
1/Izz];
C=[0 1 0 0 0 0; ...
    0 0 0 1 0 0; ...
    0 0 0 0 0 1];

D=zeros(3,3);

%continuous to discrete time convertion
ts=0.2; %sampling time
[Ad,Bd,Cd,Dd]=c2dm(A,B,C,D,ts);

%augmented model
[Aa,Ba,Ca]=augment_mimo(Ad,Bd,Cd,n_states,n_inputs
,n_outputs);

%controllability
CO=[Ba,Aa*Ba,Aa^2*Ba,Aa^3*Ba,Aa^4*Ba,Aa^5*Ba];
controllability=rank(CO);
if controllability==n_states
    disp("system is controllable")
end

%prediction matrices

%output predictions

P=[Ca*Aa;Ca*Aa^2;Ca*Aa^3;Ca*Aa^4;Ca*Aa^5;Ca*Aa^6];

H=[Ca*Ba,zeros(nu),zeros(nu);...
    Ca*Aa*Ba,Ca*Ba,zeros(nu);...
    Ca*Aa^2*Ba,Ca*Aa*Ba,Ca*Ba;...
    Ca*Aa^3*Ba,Ca*Aa^2*Ba,Ca*Aa*Ba;...
    Ca*Aa^4*Ba,Ca*Aa^3*Ba,Ca*Aa^2*Ba;...
    Ca*Aa^5*Ba,Ca*Aa^4*Ba,Ca*Aa^3*Ba];

% contraints
u1=L*b*(max_speed^2-min_speed^2);
u2=u1;
u3=k*(max_speed^2+max_speed^2-min_speed^2-
min_speed^2);
u_max=[u1;u2;u3];
u_min=-u_max;
Dumax=0.6*u_max; %rate of input changes

%constraint matrices
[CC,dd,dupast]=constraints_mimo(Dumax,u_max,u_min,
n_inputs,nu);

%cost functions

% weights on control inputs

W=[7.5e-2 0 0 0 0 0 0 0 0; ...
    0 7.5e-2 0 0 0 0 0 0 0; ...
    0 0 4.5e-2 0 0 0 0 0 0;...
    0 0 0 7.5e-2 0 0 0 0 0; ...
    0 0 0 0 7.5e-2 0 0 0 0; ...
    0 0 0 0 0 4.5e-2 0 0 0; ...
    0 0 0 0 0 0 7.5e-2 0 0; ...
    0 0 0 0 0 0 0 7.5e-2 0; ...
    0 0 0 0 0 0 0 0 4.5e-2];

E=2*(H'*H+W);

% simulation loop parameters
xf=zeros(size(Ba,1),1);
xh=zeros(size(Ba,1),1);
yh=y;

%simulation loop
for i = 1:100
    F=-2*H'*(Rs*r(:,i)-P*(xf));
    d= dd + dupast*u;
    [deltau,uncons]=Qphild(E,F,CC,d);

    %control horizon of 3
    deltau=[deltau(1,1) deltau(2,1) deltau(3,1);
...
        deltau(4,1) deltau(5,1) deltau(6,1); ...
        deltau(7,1) deltau(8,1) deltau(9,1)];

    deltaU=deltau(1,:);
    u=u+deltaU'; %input

    %model
    xh(:,i+1)=Aa*xh(:,i)+Ba*deltaU';
    yh(:,i)=Ca*xh(:,i+1)+dist(:,i);
    xf=xh(:,i+1);

end

% plots

 i=1:100;

% output1
subplot(3,1,1)
plot(i,r(1,i),'r',i,yh(1,i),'b')
legend('reference','output');
xlabel('time');
ylabel('phi')

% output2
subplot(3,1,2)
plot(i,r(2,i),'r',i,yh(2,i),'b')
legend('reference','output');
xlabel('time');
ylabel('theta')

% output3
subplot(3,1,3)
plot(i,r(3,i),'r',i,yh(3,i),'b')
legend('reference','output');
xlabel('time');
ylabel('psi')
```

augment_mimo function

```
function [Aa,Ba,Ca]=
augment_mimo(Ad,Bd,Cd,n_states,n_inputs,n_outputs)

Aa=[Ad zeros(n_states,n_inputs); Cd*Ad
eye(n_outputs)];
Ba=[Bd;Cd*Bd];
Ca=[zeros(n_inputs,n_states) eye(n_inputs)];
```

Constraints_mimo function

```
function [CC,dd,dupast] =
constraints_mimo(Dumax,u_max,u_min,n_inputs,nu)
CC=[];
dim = nu*n_inputs; %matrices dimensions
CC(1:dim,1:dim)=eye(dim);
CC(dim+1:2*dim,1:dim) = -eye(dim);
s=0;
E=zeros(dim,dim);
for j=1:nu
    E(1+s:n_inputs*j,1:n_inputs) = eye(n_inputs);
    s=s+n_inputs;
end
p=n_inputs+1;
for counter = 1:nu-1
    i=counter;
    for k = (n_inputs*counter)+1:n_inputs:dim-
n_inputs+1

E(k:n_inputs*(i+1),p:n_inputs*(counter+1))=
eye(n_inputs);
        i=i+1;
    end
    p=p+n_inputs;
end

Cu=[eye(dim);-eye(dim)];
CuE=Cu*E;
CC(2*dim+1:4*dim,1:dim)=CuE;
ddeltaU=zeros(2*dim,1);
t=1;
for i=1:n_inputs:size(ddeltaU,1)
    ddeltaU(i:t*n_inputs,1) = Dumax;
    t=t+1;
end
du=zeros(2*dim,1);
%upper limits
t=1;
for i=1:n_inputs:size(du,1)/2
    du(i:t*n_inputs,1)=u_max;
    t=t+1;
end
% lower limits
for i=1+size(du,1)/2 : n_inputs: size(du,1)
    du(i:t*n_inputs,1)=-u_min;
    t=t+1;
end

dd=[ddeltaU;du];
% dupast past inputs

L=zeros();
t=1;
for i=1:n_inputs:(n_inputs*nu)
    L(i:n_inputs*t,1:n_inputs)=eye(n_inputs);
    t=t+1;
end

CuL=Cu*L;
dupast=[zeros(size(ddeltaU,1),n_inputs);-CuL];
```

Qphild function

```
function [deltau,uncons]=Qphild(E,F,CC,d)
[n1,m1]=size(CC);
deltau=-E\F; %global solution without constraints
uncons=deltau;
kk=0;
for i=1:n1
    if(CC(i,:)*deltau>d(i))
        kk=kk+1;
    else
        kk=kk+0;
    end
end
if (kk==0)
    return;
end
%dual quadratic programming matrices
T=CC*(E\CC');
K=(CC*(E\F)+d);
[n,m]=size(K);
lambda=zeros(n,m);
for km= 1:15
    lambda_p=lambda; %previous lambda
    for i=1:n
        w=T(i,:)*lambda-T(i,i)*lambda(i,1);
        w=w+K(i,1);
        la=-w/T(i,i);
        lambda(i,1)=max(la);
    end
    al=(lambda-lambda_p)'*(lambda-lambda_p);
    if(al<10e-5)
        break;
    end
end
```