People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Power and Control

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of the

MASTER

In **Control Engineering**

Option: **Control Engineering**

Title:

Control Design for Trajectory Tracking of Quadrotor UAV

Presented by:

- **BOUCHERBA Djalal Eddine**

Supervisor:

Dr. AMMAR Abdelkarim

Registration Number:……/2022

Abstract

This work studies the modeling and control of the quadcopter. The Newton-Euler method is used to develop the quadcopter's nonlinear dynamic model, and design controllers for attitude and trajectory tracking. Two subsystems describe the motion of the quadcopter; a rotational subsystem for altitude and heading, and a translational subsystem for position.

Three controllers were proposed to achieve position tracking, the PID controller, the Fractional Order PID controller and the Sliding Mode controller. The controllers were implemented on the quadrotor model using Matlab/Simulink. Finally, the performance of the proposed controllers was demonstrated in a simulation study and analysis.

Dedication

To my beloved parents for their unrelenting support and sacrifices

To my dear brothers

And to my friends for their encouragement and motivation

Acknowledgement

First and foremost, Alhamdulillah for everything I have, Alhamdulillah for everything I had and Alhamdulillah for everything I will have.

I’d like to express my gratitude my supervisor Dr. AMMAR Abdelkarim, for all his help, guidance and patience throughout this project.

Table of Content

List of Figures

List of Tables

List of Abbreviations

UAV Unmanned Aerial Vehicles

DOF Degree Of Freedom

PID Proportional Integral Derivative

FOPID Fraction Order Proportional Integral Derivative

SMC Sliding Mode Controller

General Introduction

Originally developed for military applications, Unmanned Aerial Vehicles (UAVs) commonly known as drones sought to replace humans in tasks that were deemed too dull, dirty or dangerous [1]. Some of these tasks included reconnaissance, route and zone recognition, combat damage assessment and even target elimination

Some of the early developed UAVs during and after the World War II era were quite big in size matching that of normal manned aircrafts, they however continued to reduce in size as computers, sensors and actuators advanced in power and reduced in size.

This vast development of computer control systems coupled with the reduction of cost shifted the utility of UAVs to include non-military applications. Some of these applications for instance are monitoring large infrastructure such as power plants, damage assessment and disaster relief, crop spraying, surveillance, journalism and filmmaking, etc.

This has cemented the drone's spot as one of the most useful technological advancements of our time. And due to the different shapes/configurations a drone can take, in addition to its depth and complexity, it is important to design controllers that allow optimal performance. One such configuration is the quadrotor.

In recent years, the quadrotor has aroused many attentions of researchers since its unique structure, capability of hovering, vertical take-off and landing gives it good maneuverability and a relatively increased payload [2].

The objective of this project is to determine a model that describes the dynamics of a quadrotor, then design controllers that allow the vehicle to follow a given trajectory accurately.

The first chapter covers an introduction into unmanned aerial vehicles as well as a brief historical background, introduces the quadrotor configuration, its advantages as well as the challenges it poses. In chapter two, a quadrotor system model is established from the different dynamics that govern its motion. Chapter three concerns itself with controlling the established model, by choosing different control strategies and designing the necessary controllers accordingly. Chapter four is a showcase of the results of the tests performed on the model in SIMULINK, and a discussion into those results.

1. Chapter 1: Background

1.1. Introduction

UAVs have had a long and running history of development, being used in a multitude of applications throughout time. And due to their unmanned nature, the challenge of controlling these vehicles is at hand.

1.2. UAVs

Unmanned Aerial Vehicle or UAV is an aircraft that does not carry a human pilot and is thus controlled remotely, autonomously or semi-autonomously. Similar to many inventions, UAVs are the results of developments carried out during military conflicts and served a multitude of military purposes such as surveillance, reconnaissance and target elimination. Development leaps and cost reduction however, changed their target goal from a strictly military one to include civilian applications.

Small unmanned rotorcrafts are a type of UAVs that is characterized by its small size and capability to perform many functions. They can maneuver in three dimensions to collect information with onboard sensors and even physically interact with their environments using onboard grippers [3].

The small size of these vehicles enables them to operate indoors and in constrained spaces. This is particularly useful in dangerous situations such as searching for survivors in damaged buildings, entering and clearing areas with armed adversaries, and collecting information in contaminated locations. In these scenarios the ability to create situational awareness without ever having to put a human in harm's way is extremely valuable. They are also capable of completing an even wider range of applications such as security, surveillance, emergency response, infrastructure inspection, aerial photography, geographic mapping and express shipping and delivery.

One type of small unmanned rotorcrafts is the Quadrotor helicopters (or quadrotors). They have four fixed-pitch propellers attached to motors typically mounted in a cross configuration, and are available from several companies as research and commercial vehicles as well as toys. The long moment arms on which the propellers lie enables them to produce large control moments and perform aggressive maneuvers while retaining a mechanical simplicity to the quadrotor's design [4].

1.3. A Brief Historical Background

When we think of UAVs, hot-air balloons are typically not part of the discussion. From a technical standpoint however, these crafts were the first aircraft to not require a human pilot. Joseph-Michel and Jacques-Étienne Montgolfier hosted the first public demonstration of an unmanned aircraft, a hot-air balloon in Annonay, France back in 1783[5].

It wasn't however until 1935 that the first modern drone was developed. When the Royal Air Force's commenced in 1918, the UK needed effective methods for training pilots. While Target practice was typically accomplished by towing gliders behind crewed aircraft, that method failed to provide a realistic simulation for engaging enemy fighters in live combat. In

response, the De Havilland DH.82B Queen Bee aircraft was used a low-cost radio-controlled drone developed for aerial target practice [6].

Figure 1-1 Winston Churchill and the Secretary of State for War waiting to see the launch of a de Havilland Queen Bee radio-controlled target drone, 6 June 1941.

After many years of strict military use of drones, UAVs were permitted in US Civilian Airspace for the first time in 2005. Following the devastation caused by Hurricane Katrina, the FAA allowed UAVs to fly in civilian airspace for search, rescue and disaster relief operations [7]. Drones equipped with thermal cameras were able to detect the heat signatures of humans from up to 10,000 feet away.

Around this time, the consumer drone industry began to really take shape. While DJI had yet to become the marketplace giant it is today, companies like Parrot, 3DR, and many others were looking to take military UAV technology and repurpose it. The potential for industrial and consumer UAV markets was more than enough for many businesses to invest in the technology.

Around the year of 2013, Major Companies such as FedEx, UPS, Amazon and Google looked to Start drone delivery, and testing of various UAV concepts and work with regulatory agencies around the world began [8].

Since then, UAVs have continued to expand in capabilities and use cases, and as more industries explore how drones can make their work safer and more cost effective, growth is expected to rapidly surge in the coming years.

1.4. The Quadrotor and the Problem of Control

A quadrotor is a rotorcraft that uses two pairs of counter-rotating, fixed-spaced blades for lift. The use of fixed-pitch blades typically allows the vehicle's propellers to be directly attached to four separate engines without the need for complex linkages to control pitch. These motors are then connected in an "X" configuration. To drive and control the rotor, a battery and microcontroller are placed near the center of the vehicle so as to not cause imbalance. Changes in the aircraft's altitude and orientation are achieved by changing the speed of individual rotors.

Even with such a simple design, building a quadrotor poses some difficult challenges. In particular, quadrotors are hard to control and can tip over easily. Because its mass is concentrated in a small area, the rotors must react very quickly to counteract the tendency to tip over. Another challenge it presents is trajectory control, as quadrotors are controlled through altitude commands; angles that govern its rotations and thrust.

1.5. Conclusion

The mechanical simplicity of the quadrotor coupled with its great maneuverability makes it a very desirable choice for a UAV. And this thesis is an attempt to model and design a flight control system that is robust and efficient in order to accurately track a given trajectory for this aircraft.

2. Chapter 2: Quadrotor System Modeling

2.1. Introduction

The quadrotor is a complex system with many variables that control and describe its motion. This chapter covers the process with which a quadrotor model was derived from the different physical equations and dynamics that govern its motion. In order to achieve that a few concepts must be established.

2.2. Reference of frame

To understand the position and orientation of the quadrotor we must first define two references of frame:

- Inertial Frame {E}:Is reference fixed-frame to some specific location or place on the ground level represented by {x, y, z} where the x-axis of the inertial frame is pointed to the North, y-axis is pointed to the East and z-axis is pointed to the center of the Earth
- Body Frame {B}:Is body fixed-frame that has its origin located at the center of mass of the quadcopter represented by $\{xb, yb, zb\}$ where the xb axis is lies on propeller 1 while the yb axis lies on propeller 4 and the zb axis points upward, which means if the body rotates or moves the frame rotates and moves with it.

The Euler angles describe the altitude or position of the quadrotor and are defined by the orientation of B-frame with respect to E-frame. This orientation can be represented by a rotation matrix that is derived from a combination of a sequence of rotations where the quadrotor is first rotated along the z-axis (yaw motion) then by a rotation along the y-axis (pitch motion) followed by a rotation along the x-axis (roll motion)[9].

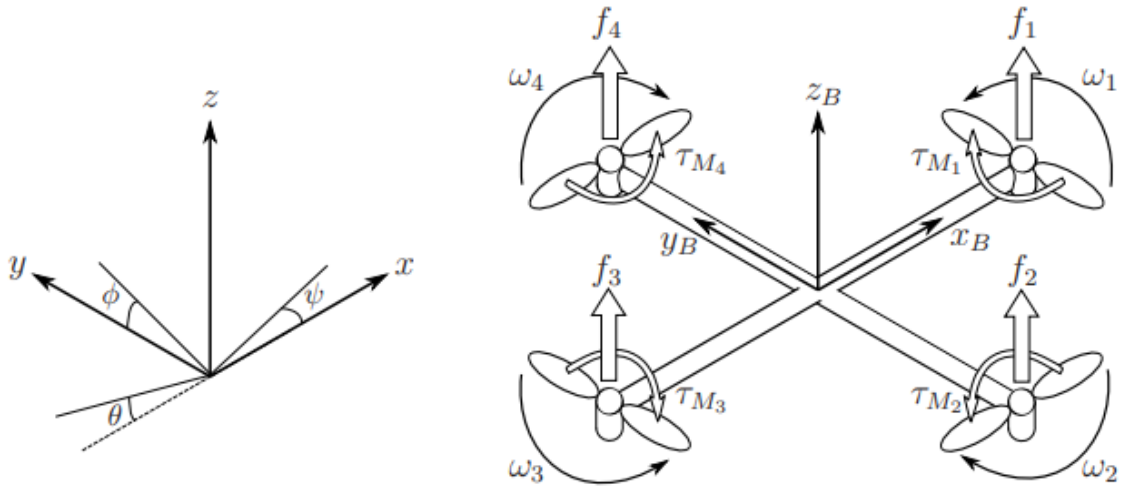

Figure 2-1 Inertial and Body frame of a quadrotor

Projecting on the y-axis and z-axis when rotating about x-axis, on the x-axis and z-axis when rotating about y-axis and on the x-axis and y-axis when rotating about x-axis yields to the following 3 matrices respectively [9]:

$$\begin{bmatrix} xb \\ yb \\ zb \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.1)$$

$$\begin{bmatrix} xb \\ yb \\ zb \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.2)$$

$$\begin{bmatrix} xb \\ yb \\ zb \end{bmatrix} = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.3)$$

The rotation matrix R is then derived based on the multiplication of the sequence of principle rotations roll (ϕ), pitch (θ) and yaw (ψ) angles about the x, y and z-axes respectively:

$$\begin{bmatrix} xb \\ yb \\ zb \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} cos(\varphi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.4)$$

After multiplying the three matrices it yields

$$\mathbf{R} = \begin{bmatrix} C(\theta)C(\psi) & C(\theta)S(\psi) & -S(\theta) \\ -C(\phi)S(\psi) + C(\psi)S(\theta)S(\phi) & C(\psi)C(\phi) + S(\theta)S(\phi)S(\psi) & C(\theta)S(\phi) \\ S(\psi)S(\phi) + C(\psi)C(\phi)S(\theta) & -S(\phi)S(\psi) + C(\phi)S(\theta)S(\psi) & C(\theta)C(\phi) \end{bmatrix} \quad (2.5)$$

where C is a cosine function and S is a sine function.

2.3. Six Degrees of Freedom (6-DOF)

Six degrees of freedom refers to the way a rigid body can move in three dimensional spaces. This movement the body can make can be a forward/backward along the x-axis, left/right along the y-axis and up/down along the z axis translation. It can also be combined with rotations along these axes, rotation about the x-axis is known as roll, rotation about the y-axis is known as pitch and rotation about the z-axis is yaw [10].

2.4. Flight Mechanics

2.4.1. Quadrotor Dynamics

The following assumptions were taken when deriving the quadrotor dynamics:

• The quadrotor's center of gravity coincides with the origin of the body frame.

• The quadrotor is a rigid body.

• The quadrotor is symmetrical with respect to the x and y axes.

• The quadrotor propellers are rigid.

• The thrust and drag exerted on the quadrotor are proportional to the square of the propellers' angular speed.

The quadrotor is modeled with four rotors in a cross (x) configuration. This structure is quite thin and light; however, it shows robustness by mechanically linking the motors which tend to be heavier than the structure. Each propeller is connected to the motor through the reduction

gears. All the propellers' axes of rotation are fixed and parallel. Furthermore, they have fixed-pitch blades and their airflows point downwards in order to achieve upwards lift.

These considerations point out that the structure is quite rigid and the only things that can vary are the propeller speeds. In our model however, neither the motors nor the reduction gears are fundamental because the movements are directly related just to the propeller's velocities. The others parts will be taken into account. Another neglected component is the electronic box. As in the previous case, the electronic box is not essential to understand how the quadrotor flies. It follows that the basic model to evaluate the quadrotor movements it is composed just of a thin cross structure with four propellers on its ends. The front and the rear propellers rotate counter-clockwise, while the left and the right ones turn clockwise.

Figure 2-2 shows the structure's model in hovering condition, where all the propellers have the same speed.

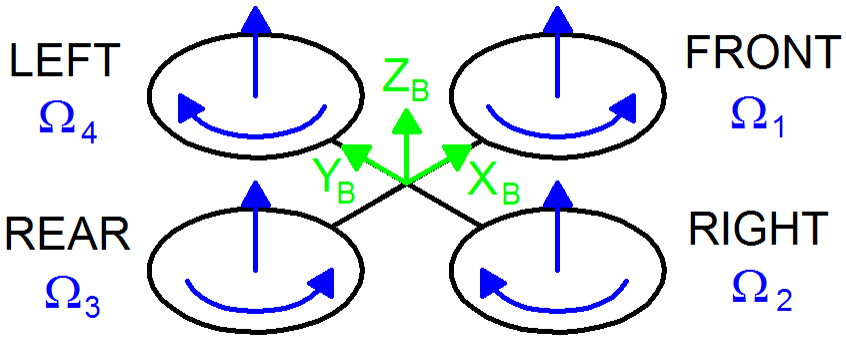

Figure 2-2 Simplified Quadrotor in Hovering Mode

The fixed-body B-frame is represented with green whereas the angular speed of the propellers is in blue. In addition to the name of the velocity variable, for each propeller, two arrows are drawn: the curved one represents the direction of rotation; the other one represents the velocity. In the model of Figure 2-2 all the propellers rotate at the same speed Ω_H to counterbalance the acceleration due to gravity. Thus, the quadrotor performs stationary flight, and no forces or torques move it from its position.

2.4.2. Forces, Moments and Torques

The quadrotor is subject to forces and moments, and the configuration mentioned above allows four basic movements. The movements, forces and moments are described as follows:

- Thrust Force (throttle) (U_1)

This command is provided by increasing or decreasing all propeller speeds by the same amount. It leads to a vertical force which raises or lowers the quadrotor. If the quadcopter is in horizontal position, the vertical direction of the inertial frame and that one of the body-fixed frame coincide. Otherwise, the provided thrust generates both vertical and horizontal accelerations in the inertial frame. Figure 2-3 shows the throttle command on a quadrotor sketch.

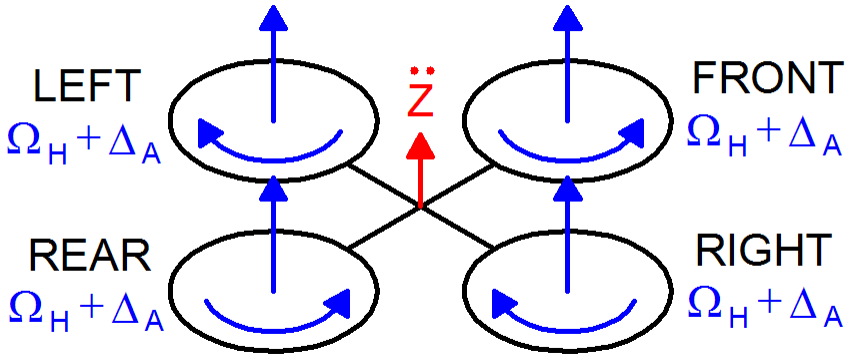

Figure 2-3 Throttle Movement

The speed of the propellers is specified in blue which in this case, is equal to $\Omega_H + \Delta_A$ for each one. Δ_A is a positive variable that can't be too large to avoid saturation.

Since all the motors are identical, the derivation is explained for a single one which is given by the momentum theory:

$$T_i = C_D \rho A r^2 \Omega_i^2 \tag{2.6}$$

where:

C_D is thrust coefficient of the motor.

ρ is the density of air.

A is the cross-sectional area of the propeller's rotation.

r is the radius of rotor.

Ω_i is the angular speed of the rotor.

For simple flight motion:

$$T_i = K\Omega_i^2 \tag{2.7}$$

Combining the thrust from all the 4 motor-propeller system, the net thrust in the body frame z direction is given by:

$$T = K(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{2.8}$$

K is the propeller thrust coefficient. Ω_1 is the front propeller speed, Ω_2 is the right propeller speed, Ω_3 is the rear propeller speed and Ω_4 is the left propeller speed

- Roll moment (U_2)

This command is provided by increasing (or decreasing) the left propeller speed and by decreasing (or increasing) the right one. It leads to a torque with respect to the x_B axis which

makes the quadrotor turn. The overall vertical thrust is the same as in hovering; hence this command leads only to roll angle acceleration. Figure 2-4 shows the roll command on a quadrotor sketch.

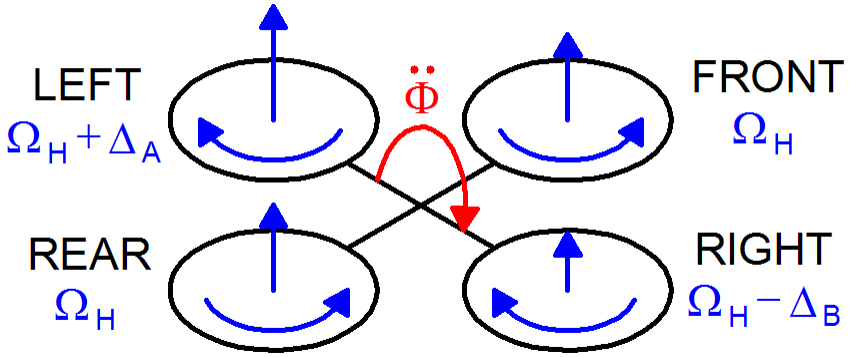

Figure 2-4 Roll Movement

The positive variables Δ_A and Δ_B should keep the vertical thrust unchanged and can't be too large to avoid saturation.

In our frame of reference, the roll movement is acquired by decreasing the 2nd rotor velocity and increasing the 4th rotor velocity and can then be expressed as:

$$T_\phi = lK(-\Omega_2^2 + \Omega_4^2) \tag{2.9}$$

Where l is the distance between the rotor and the center of mass.

- Pitch moment (U_3)

This command is very similar to the roll and is provided by increasing (or decreasing) the rear propeller speed and by decreasing (or increasing) the front one. It leads to a torque with respect to the y_B axis which makes the quadrotor turn. The overall vertical thrust is the same as in hovering; hence this command leads only to pitch angle acceleration.

As in the previous moment, the positive variables Δ_A and Δ_B are to maintain the vertical thrust unchanged and can't be too large.

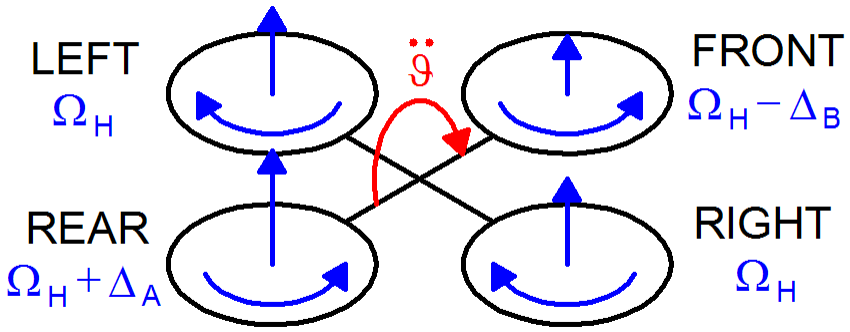

Figure 2-5 Pitch Movement

Similarly, the pitch movement is acquired by decreasing the 1st rotor velocity and increasing the 3rd rotor velocity. The pitch moment for the quadrotor can then be expressed as:

$$T_\theta = lK(-\Omega_1^2 + \Omega_3^2) \quad (2.10)$$

Where l is the distance between the rotor and the center of mass.

- Yaw moment (U_4)

This command is provided by increasing (or decreasing) the front-rear propellers' speed and by decreasing (or increasing) that of the left-right couple. It leads to a torque with respect to the z_B axis which makes the quadrotor turn. The yaw movement is generated due to the fact that the left-right propellers rotate clockwise while the front-rear ones rotate counterclockwise. Hence, when the overall torque is unbalanced, the quadcopter turns on itself around z_B. The total vertical thrust is the same as in hovering; hence this command leads only to yaw angle acceleration. Figure 2-6 shows the yaw command on a quadrotor sketch.

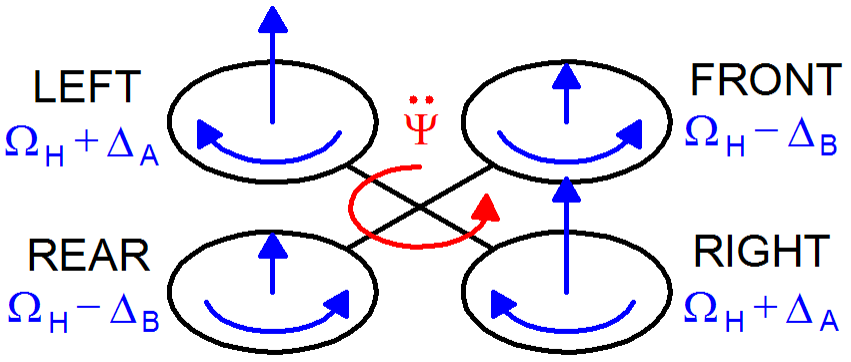

Figure 2-6 Yaw Movement

Once again, the positive variables Δ_A and Δ_B are to maintain the vertical thrust unchanged and they can't be too large. Furthermore, it maintains the equivalence $\Delta_B \approx \Delta_A$ for small values of Δ_A. Thus, in our frame of reference, the yaw movement is acquired by increasing the angular velocities of two opposite rotors and decreasing the velocities of the other two and it can be expressed as follows:

$$T_{\psi} = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \quad (2.11)$$

Where d is the drag coefficient.

All 3 moments can then be represented as:

$$\begin{bmatrix} T_{\phi} \\ T_{\theta} \\ T_{\psi} \end{bmatrix} = \begin{bmatrix} lK(-\Omega_2^2 + \Omega_4^2) \\ lK(-\Omega_1^2 + \Omega_3^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \quad (2.12)$$

- Gravitational Force: The gravitational force acting the quadcopter in the inertial frame can be expressed as:

$$G = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \quad (2.13)$$

Where m is the mass of the quadrotor and g is the gravitational acceleration

- Inertia Matrix

Moment of inertia gives the amount of moment needed to rotate a still object and moment needed to stop a rotating object. We need moments around all three axes of the quadcopter, so we will present it in matrix form; Moment of inertia is the square of distance from the center of mass of the body. For a symmetrical body such as the quadcopter the moment of inertia on opposite sides of the vehicle cancels each other.

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad (2.14)$$

It is crucial that we establish a mathematical model to describe our system dynamics before we attempt to design any controllers for it. And while there are many approaches to derive the equations governing 6-DOF rigid body movement such as (Newton-Euler, Euler Lagrange...), for the sake of simplicity we have chosen the Newton-Euler approach.

2.5. Quadrotor Equations of Motion

2.5.1. Translational Dynamics

From the Euler's first axioms of the Newton's second law[11]:

$$m\ddot{\varepsilon} = F^E \quad (2.15)$$

Using the above equation, the linear dynamic system for a quadrotor in the E-frame can be expressed as:

$$m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = RT + G \quad (2.16)$$

After substituting and some manipulation we get:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \frac{T}{m}(sin(\phi)sin(\psi) + cos(\phi)cos(\psi)sin(\theta) \\ \frac{T}{m}(cos(\phi)sin(\psi)sin(\theta) - cos(\psi)sin(\phi) \\ -g + \frac{T}{m}(cos(\phi)\, cos(\theta) \end{bmatrix} \tag{2.17}$$

Where $\ddot{x}, \ddot{y}\ and\ \ddot{z}$ are the linear accelerations in the x, y and z axis respectively.

2.5.2. Rotational Dynamics

From the Euler's second axioms of the Newton's second law:

$$I\dot{\omega} + \omega \times (I\omega) = \tau B \tag{2.18}$$

Substituting with equation (2.12) and equation (2.14) in equation (2.18) yields the angular dynamic system for a quadrotor in the B-frame that can be expressed as:

$$\begin{cases} \ddot{\phi} = \frac{1}{l_x}T_\phi + \frac{l_y}{l_x}\dot{\psi}\dot{\theta} - \frac{l_z}{l_x}\dot{\theta}\dot{\psi} \\ \ddot{\theta} = \frac{1}{l_y}T_\theta + \frac{l_z}{l_x}\dot{\psi}\dot{\phi} - \frac{l_x}{l_y}\dot{\theta}\dot{\psi} \\ \ddot{\psi} = \frac{1}{l_z}T_\psi + \frac{l_x}{l_z}\dot{\phi}\dot{\theta} - \frac{l_y}{l_z}\dot{\theta}\dot{\phi} \end{cases} \tag{2.19}$$

2.6. State Space Model

The acquired mathematical models will be formulated to a state space model to make the control problem easier. We can write the dynamic model equations as:

$$\dot{X} = f(X, U) \tag{2.20}$$

Where U is the input vector and $\dot{X}$ is the state vector.

2.6.1. State Vector X

$$X = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8 \quad x_9 \quad x_{10} \quad x_{11} \quad x_{12}]^T \tag{2.21}$$

This state vector is mapped to the degrees of freedom of the quadrotor in the following manner:

$$X = [\phi \quad \dot{\phi} \quad \theta \quad \dot{\theta} \quad \psi \quad \dot{\psi} \quad z \quad \dot{z} \quad x \quad \dot{x} \quad y \quad \dot{y}]^T \tag{2.22}$$

The state vector defines the position of the quadrotor in space and its angular and linear velocities.

2.6.2. Control Input U

A control input vector consisting of four inputs U_1 through U_4 is defined to be:

$$U = [U_1 \quad U_2 \quad U_3 \quad U_4]^T \tag{2.23}$$

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} K & K & K & K \\ 0 & -Kl & 0 & Kl \\ -Kl & 0 & Kl & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{2.24}$$

As mentioned in an earlier section, U_1 is the resulting upwards force of the four rotors which is responsible for the altitude of the quadrotor and its rate of change (z, $\dot{z}$). U_2 is the difference in thrust between the 2nd and the 4th rotor which is responsible for the roll rotation and its rate of change (ϕ,$\dot{\phi}$). On the other hand, U_3 represents the difference in thrust between the 1st and the 3rd rotor, thus generating the pitch rotation and its rate of change (θ,$\dot{\theta}$). Finally, U_4 is the difference in torque between the two clockwise turning rotors, and the two counterclockwise turning rotors generating the yaw rotation and ultimately its rate of change (ψ,$\dot{\psi}$).

U_1 will generate the desired altitude of the quadrotor, U_2 will generate the desired roll angle, and the desired pitch angle will be generated by U_3 whereas U_4 will generate the desired yaw angle.

As denoted before our subsystem is divided into two parts: The rotational and a transitional one which are represented in the state space.

2.6.3. The Translational Subsystem

Substituting T with U_1 and the corresponding states in equation (2.18) we get:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \frac{U_1}{m}(sin(x_1)\,sin(x_5) + cos(x_1)cos(x_5)sin(x_3)) \\ \frac{U_1}{m}(cos(x_1)sin(x_5)sin(x_3) - cos(x_5)sin(x_1)) \\ -g + \frac{U_1}{m}(cos(x_1)cos(x_3)) \end{bmatrix} \tag{2.25}$$

2.6.4. The Rotational Subsystem

Given that:

$$\begin{bmatrix} T_\phi \\ T_\theta \\ T_\psi \end{bmatrix} = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{2.26}$$

The rotational subsystem becomes:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} b_1 U_2 + a_1 x_4 x_6 \\ b_2 U_3 + a_2 x_2 x_6 \\ b_3 U_4 + a_3 x_2 x_4 \end{bmatrix} \tag{2.27}$$

Where: $a_1 = \frac{I_y - I_z}{I_x}$ $a_2 = \frac{I_z - I_x}{I_y}$ $a_3 = \frac{I_x - I_y}{I_z}$ $b_1 = \frac{l}{I_x} b_2 = \frac{l}{I_y} b_3 = \frac{l}{I_z}$

2.6.5. State Space Representation

Using equation (2.25) and equation (2.26), the complete mathematical model of the quadrotor can be written in a state space representation as follows:

$$\dot{x}_1 = \dot{\phi} = x_2$$

$$\dot{x}_2 = \ddot{\phi} = b_1 U_2 + a_1 x_4 x_6$$

$$\dot{x}_3 = \dot{\theta} = x_4$$

$$\dot{x}_4 = \ddot{\theta} = b_2 U_3 + a_2 x_2 x_6$$

$$\dot{x}_5 = \dot{\psi} = x_6$$

$$\dot{x}_6 = \ddot{\psi} = b_3 U_4 + a_3 x_2 x_4 \tag{2.28}$$

$$\dot{x}_7 = \dot{z} = x_8$$

$$\dot{x}_8 = \ddot{z} = -g + \frac{U_1}{m} (cos(x_1) cos(x_3))$$

$$\dot{x}_9 = \dot{x}$$

$$\dot{x}_{10} = \ddot{x} = \frac{U_1}{m} (sin(x_1)\, sin(x_5) + cos(x_1) cos(x_5) sin(x_3))$$

$$\dot{x}_{11} = \dot{y}$$

$$\dot{x}_{12} = \ddot{y} = \frac{U_1}{m} (cos(x_1) sin(x_5) sin(x_3) - cos(x_5) sin(x_1))$$

Then equation (2.20) can be written as:

$$\dot{X}= f(X,U) = \begin{bmatrix} x_2 \\ b_1U_2 + a_1x_4x_6 \\ x_4 \\ b_2U_3 + a_2x_2x_6 \\ x_6 \\ b_3U_4 + a_3x_2x_4 \\ x_8 \\ -g + \frac{U1}{m}(cos(x_1)cos(x_3)) \\ \dot{x} \\ \frac{U1}{m}(sin(x_1)\,sin(x_5) + cos(x_1)cos(x_5)sin(x_3)) \\ \dot{y} \\ \frac{U1}{m}(cos(x_1)sin(x_5)sin(x_3) - cos(x_5)sin(x_1)) \end{bmatrix} \tag{2.29}$$

2.7. Linearized Model

In order to linearize the system (2.29), we need to find an equilibrium point $\bar{X}$ which for fixed input $\bar{U}$ is the solution of the algebraic system (2.29), or the value of state's vector, which on fixed constant input is the solution of algebraic system:

$$f(\bar{X}, \bar{U}) = 0 \tag{2.30}$$

Since the function f is nonlinear, problems related to the existence of a unique solution of system (2.30) arise. In particular, for the system in hand, the solution is difficult to find because of trigonometric functions related each other in no-elementary way. For this reason, the linearization is performed on a simplified model which considers only small oscillations. This simplification is made by approximating the sine function with its argument and the cosine function with unity. The approximation is valid if the argument is small. The resulting system is described by the following equations [12]:

$$\hat{f}(X,U) = \begin{bmatrix} x_2 \\ b_1U_2 + a_1x_4x_6 \\ x_4 \\ b_2U_3 + a_2x_2x_6 \\ x_6 \\ b_3U_4 + a_3x_2x_4 \\ x_8 \\ -g + \frac{U1}{m} \\ \dot{x} \\ \frac{U1}{m}(x_1x_5 + x_3) \\ \dot{y} \\ \frac{U1}{m}(x_5x_3 - x_1) \end{bmatrix} \tag{2.31}$$

As illustrated above, an equilibrium point is needed in order to perform the linearization, and it was established from the previous equation as:

$$X = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad z \quad 0 \quad x \quad 0 \quad y \quad 0]^T \in \mathbb{R}^{12} \tag{2.32}$$

From equation (2.32), the equilibrium point is obtained by the constant input vector:

$$\bar{\mathrm{U}} = [\mathrm{mg} \quad 0 \quad 0 \quad 0]^{\mathrm{T}} \in \mathbb{R}^4 \tag{2.33}$$

It can be noted that input vector U_e represents the force necessary to offset the effect of gravity and hover the quadrotor. After finding the equilibrium point X_e and the corresponding nominal vector U_e, the linear model can be found in the form:

$$\dot{X} = AX + BU \tag{2.34}$$

$$\mathbf{A} = \frac{\partial f(X,U)}{\partial X}\Big|_{X=\bar{X}}^{U=\bar{U}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.35}$$

$$B = \frac{\partial f(X,U)}{\partial X}\Big|_{X=\bar{X}}^{U=\bar{U}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & b_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & b_2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b_3 \\ 0 & 0 & 0 & 0 \\ \frac{-1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.36}$$

2.8. Controllability

Controllability is a major concept in modern control system theory. Introduced by R. Kalman in 1960, it can be defined somehow as:

Controllability: In order to be able to do whatever we want with the given dynamic system under control input, the system must be controllable. Given the system (2.34) the pair (A, B) is said to be controllable if for any initial state X (0) = X_0 and any final state X_1, there exists an input that transfers $X_0$to X_1 in a finite time. Otherwise (A, B) is said to be uncontrollable [13]. A controllability matrix (W_C) must be established in order to check if our system is controllable or not. This can be done by checking if the controllability matrix is full rank:

$$W_C = [B \quad AB \quad A^2B \quad A^3B \quad A^4B \quad A^5B \quad A^6B \quad A^7B \quad A^8B \quad A^9B \quad A^{10}B \quad A^{11}B]$$

Using MATLAB, we were able to check the controllability of the system found it to be controllable.

2.9. Open Loop Simulation

An open loop simulation was executed in order to verify the derived mathematical model. The chosen input for this test is the step input.

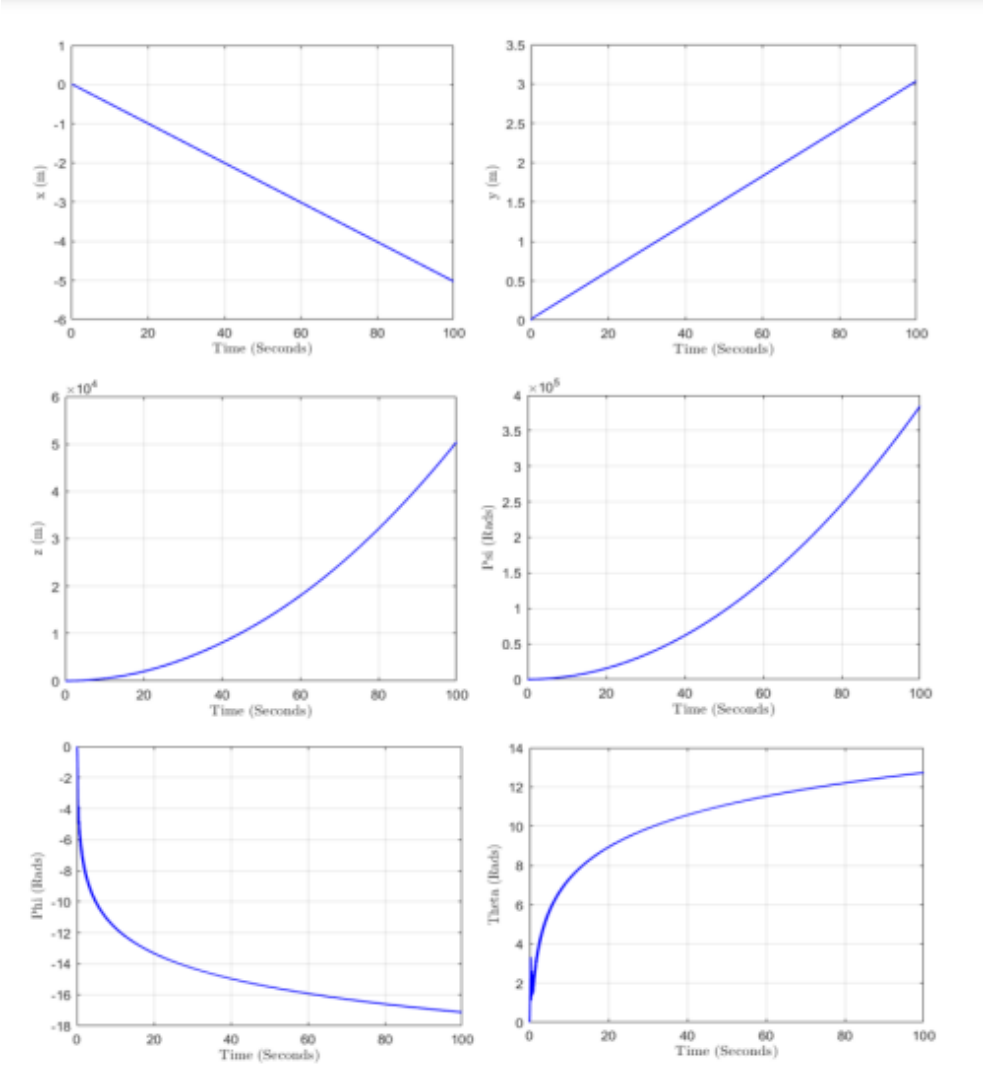

Figure 2-7 Open Loop Simulation Test Results

The open loop tests show that the system does not reach the desirable state it is given, but instead diverges towards infinity. This also shows that the system is unstable, and in order to stabilize its response controllers must be designed and applied to it.

2.10. Conclusion

In this chapter, we've established our mathematical model using the Newton-Euler approach after understanding the motion of the quadrotor. The open loop simulation results however show that the system is unstable and requires controllers to be applied to stabilize its response.

3. Chapter 3: Control Strategies and Implementation

3.1. Introduction

After establishing a quadrotor model, the different control strategies must now be chosen. This chapter covers these different strategies and the manner with which they are developed and applied to our system.

Due to the instability of our system, it is required to develop controllers to ensure it follows the commands it is given in a stable and accurate manner. There are numerous control methods that can be applied, and the ones selected in this project are: SMC, PID and Fractional PID.

3.2. Control Strategies

It is first important to establish the fact that the goal is to achieve trajectory control, this necessitates the need to develop two controllers for each chosen strategy; one for altitude control that follows desired altitude commands (theta, omega, psi and z), and another that follows the desired trajectory commands(x,y).

3.2.1. Trajectory Controller

The trajectory controller calculates the deviation from the desired path (in the body frame) and uses it to produce the desired values of pitch and roll. Calculating these values can be done from the translation equations (2.25) under the hovering condition where $T = mg$[14].

This allows equation (2.25) to be written as follows:

$$\begin{cases} \ddot{x} = g(\theta_d cos(\psi) + \phi_d sin(\psi)) \\ \ddot{y} = g(\theta_d \, sin(\psi) - \phi_d cos(\psi)) \end{cases} \tag{3.1}$$

Desired roll and pitch values can then be calculated as follows:

$$\begin{bmatrix} \phi_d \\ \theta_d \end{bmatrix} = \frac{1}{g} \begin{bmatrix} \sin(\psi) & -\cos(\psi) \\ \cos(\psi) & \sin(\psi) \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{3.2}$$

Expressions of $\ddot{x}\ and\ \ddot{y}$ change depending on the control method being applied.

It is important to note that giving x,y,z coordinates for trajectory is not enough as the yaw angle cannot be calculated and must therefore be supplied.

3.2.2. Altitude Controller

The altitude controller calculates the deviation in the roll, pitch and yaw angles as well as the deviation in the altitude z and aims to generate the control inputs U_1 to U_4. This too is done differently according to the applied control method.

The overall quadrotor control structure can be summarized in the following block diagram:

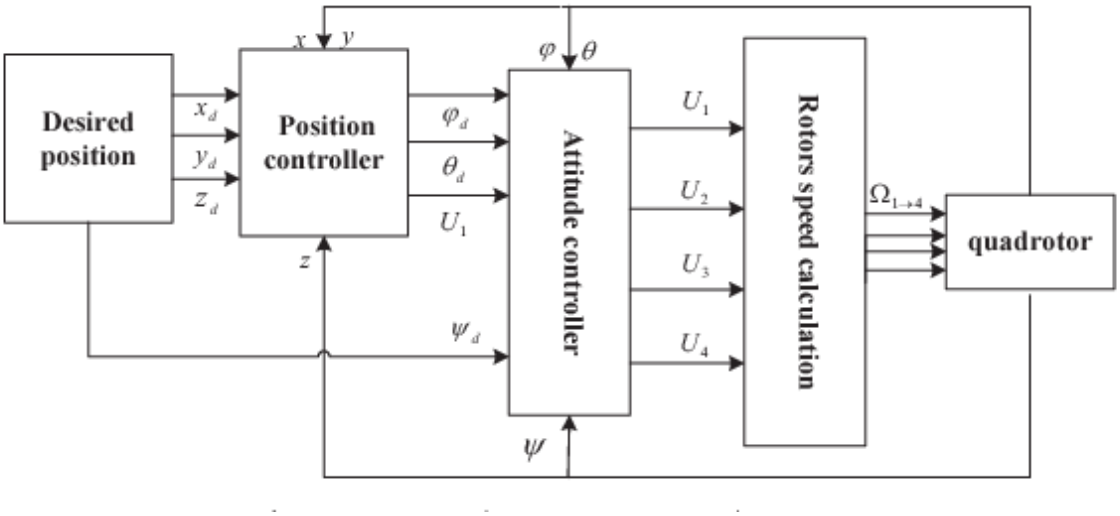

Figure 3-1 Block Diagram Describing the Quadrotor Model and Control

The altitude controller produces control signals U_1 to U_4, not to be confused with control inputs these signals are used to calculate the angular speed of the rotors Ω_1 to Ω_4 as follows:

$$\begin{cases} \Omega_1^2 = \frac{1}{4b}U_1 - \frac{1}{2bl}U_3 - \frac{1}{4d}U_4 \\ \Omega_2^2 = \frac{1}{4b}U_1 - \frac{1}{2bl}U_2 + \frac{1}{4d}U_4 \\ \Omega_3^2 = \frac{1}{4b}U_1 + \frac{1}{2bl}U_3 - \frac{1}{4d}U_4 \\ \Omega_4^2 = \frac{1}{4b}U_1 + \frac{1}{2bl}U_2 + \frac{1}{4d}U_4 \end{cases} \tag{3.3}$$

3.3. PID controller

In the industrial field the most used linear form of control is the PID controller. Even though lots of different algorithms provide better performance than the PID, its simple structure, good performance and ability to tune its operations without requiring a specific model of the controlled system makes it a solid desirable option [15].

3.3.1. PID Controller Design

The traditional PID structure is composed of the addition of three contributes, as shown in Figure 3-2 and equation (3.4) [15].

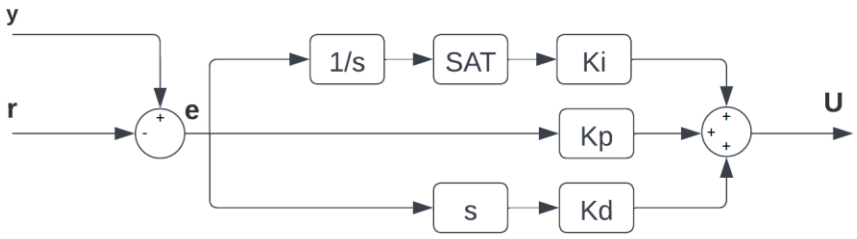

Figure 3-2 Traditional PID Structure

The blocks"$\frac{1}{s}$"and "s" represent respectively the integration and derivation operations.

$$U(t) = K_p e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \quad (3.4)$$

Where u is a generic control variable, e is the error between the task and the process output y, K_P is the proportional coefficient, K_I is the integral coefficient and K_D is the derivative coefficient.

The first contributor (P) is proportional to the error. The second contributor (I) varies according to the integral of the error. Even though this component increases the overshoot and the settling time, it has a unique propriety: it eliminates the steady state error. The third contribute (D) varies according to the derivate of the error. This component decreases the overshoot and the settling time.

The traditional PID structure presents two main drawbacks:

• The derivative action is calculated from the error. If the task adds a step in the reference, the output of the derivator would present an impulse. This sharp movement can saturate the actuators and push away the system from the linear zone. It is due to this that most PID architectures perform the derivative action on the process output only.

• The integral action combined with actuator saturation can provide a nonlinear effect which can decrease the performance of the control system. When the integral value is large and the error changes sign, it is necessary to wait a certain amount of time before the system restores its linear behavior. This phenomenon is called integral wind-up. To avoid it, a saturator is added after the integral to limit its maximum and minimum values. Figure 3-3 shows the enhanced PID structure.

Despite the fact that subsequent control equations showing the form of a tradition unenhanced PID, the enhanced structures were developed in SIMULINK.

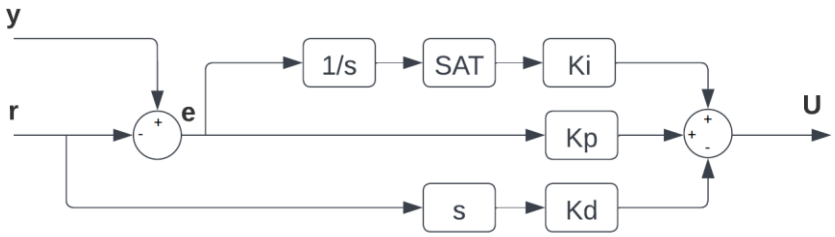

Figure 3-3 Enhanced PID Structure

3.3.1.1. Altitude Control

- Roll Controller

Another PID controller is developed to control the roll angle ϕ of the quadrotor. The derived control law generates the input U_2 that controls the roll angle as follows:

$$U_2 = K_{p\phi}(\phi_d - \phi) + K_{d\phi}(\dot{\phi_d} - \dot{\phi}) + K_{i\phi}\int(\phi_d - \phi)dt \tag{3.5}$$

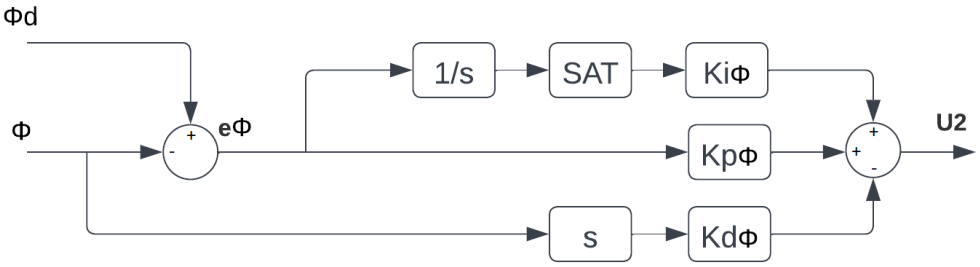

Figure 3-4 Roll PID Controller

Pitch and yaw control are both done in the same manner as roll control.

- Pitch Controller

$$U_3 = K_{p\theta}(\theta_d - \theta) + K_{d\theta}(\dot{\theta}_d - \dot{\theta}) + K_{i\theta}\int(\theta_d - \theta)dt \tag{3.6}$$

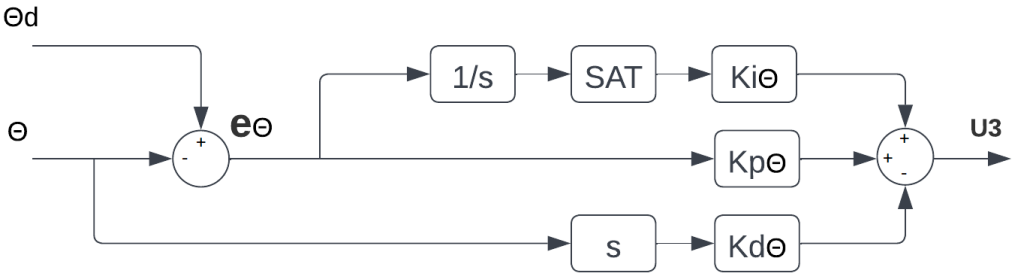

Figure 3-5 Pitch PID Controller

- Yaw Controller

$$U_4 = K_{p\psi}(\psi_d - \psi) + K_{d\psi}(\dot{\psi}_d - \dot{\psi}) + K_{i\psi}\int(\psi_d - \psi)dt \tag{3.7}$$

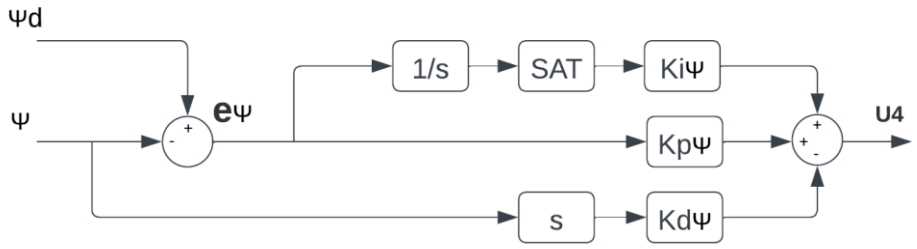

Figure 3-6 Yaw PID Controller

- Altitude/Height Controller

A PID controller is developed to control the altitude of the quadrotor. It generates the control input U_1 which is responsible for the altitude of the quadrotor as per equation (2.24). The derived control law is as follows:

$$U_1 = (K_{pz}(z_d - z) + K_{dz}(\dot{z_d} - \dot{z}) + K_{iz}\int(z_d - z)dt + g) \times \frac{m}{\cos(\theta)\cos(\phi)} \tag{3.8}$$

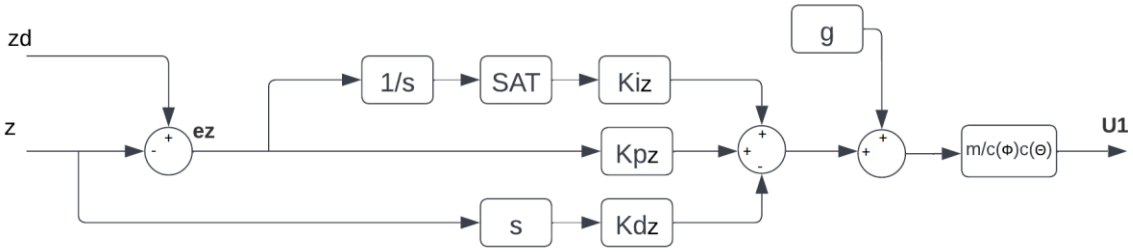

Figure 3-7 Height PID Controller

3.3.1.2. Trajectory Control

After establishing stable altitude controllers, we develop a PID controller for our trajectory. For x and y, we use two PID controllers to calculate the desired acceleration:

$$\begin{cases} \ddot{x}_d = K_{px}(x_d - x) + K_{dx}(\dot{x_d} - \dot{x}) + K_{ix}\int(x_d - x)dt \\ \ddot{y}_d = K_{py}(y_d - y) + K_{dy}(\dot{y_d} - \dot{y}) + K_{iy}\int(y_d - y)dt \end{cases} \tag{3.9}$$

Substituting equation (3.9) in (3.2) we get:

$$\begin{bmatrix} \phi_d \\ \\ \theta_d \end{bmatrix} = \frac{1}{g}\begin{bmatrix} sin(\psi) & -cos(\psi) \\ \\ cos(\psi) & sin(\psi) \end{bmatrix}\begin{bmatrix} K_{px}(x_d - x) + K_{dx}(\dot{x_d} - \dot{x}) + K_{ix}\int(x_d - x)dt \\ \\ K_{py}(y_d - y) + K_{dy}(\dot{y_d} - \dot{y}) + K_{iy}\int(y_d - y)dt \end{bmatrix} \tag{3.10}$$

3.4. Fractional PID Controller

Despite being developed over 300 years ago, Factional Order Calculus has only seen a rise in attention to its research recently, giving it a significant increase in practical applications. Fractional Calculus is a generalization of the traditional integral and differential operators from integer order to real number order; this attracted the attention of engineers that deal with dynamic systems since they can be better described by a non-integer order dynamic model. The depth and complexity of this branch gives it a larger feasible scope and greater flexibility in the system modeling and controller design methodology compared to the classical integer order one. And one of its many practical applications is the Fractional Order PID Controller [16].

This section will cover a brief overview of the FOPID and how to construct and implement it in our quadrotor system model.

3.4.1. Mathematical Background

The differintegral operator, abbreviated ${}_aD_t^q$, is a fractional calculus operator that combines differentiation and integration, by definitions it is introduced as follows [17]:

$$ {}_aD_t^q = \begin{cases} \frac{d^q}{dt^q} & if\ q > 0 \\ 1 & if\ q = 0 \\ \int_a^t (d\tau)^{-q} & if\ q < 0 \end{cases} \tag{3.11} $$

Where q is a real number representing the fractional order, a and t are the limits of the operation.

Over the years, many mathematicians have proposed various definitions for a non-integer order integral or derivative, each utilizing their own notation and technique. The Riemann Liouville definition is one form that has gained popularity in the realm of fractional calculus. Because most other fractional calculus definitions are basically modifications of the Riemann-Liuoville form, therefore, it will be the only definition discussed in our work. This definition is given by [18]:

$$ {}_aD_t^q f(t) = \frac{d^q f(t)}{d(t-a)^q} = \frac{1}{\Gamma(n-q)} \frac{d^n}{dt^n} \int_0^t \frac{f(\tau)}{(t-\tau)^{q-n+1}} d\tau \tag{3.12} $$

Where: n is the first integer which is not less than q, $n - 1 < q < n$. Γ() is the gamma function.

The Laplace transform of a q^{th} derivative with $q \in \Re$ of a signal x(t) (assuming zero initial conditions) is given by:

$$ L\{D^q x(t)\} = s^q X(s) \tag{3.13} $$

Thus, a fractional-order differential equation takes the following form:

$$ (a_n D^{q_n} + a_{n-1} D^{q_{n-1}} + \cdots + a_0 D^{q_0}) y(t) = \left(b_m D^{\beta_m} + b_{m-1} D^{\beta_{m-1}} + \cdots + b_0 D^{\beta_0}\right) u(t) \tag{3.14} $$

where a_n, $b_m \in R$ can be expressed as a fractional-order transfer function in form:

$$G(s) = \frac{b_m s^{\beta m} + b_{m-1} s^{\beta m-1} + \cdots + b_0 s^{\beta_0}}{a_n s^{q n} + a_{n-1} s^{q n-1} + \cdots + a_0 s^{q_0}} \tag{3.15}$$

The continuous-time transfer function can be expressed as a rational function H(λ) where $\lambda = s^p$:

$$H(\lambda) = \frac{\sum_{k=0}^{m} b_k \lambda^k}{\sum_{k=0}^{n} a_k \lambda^k} \tag{3.16}$$

Based on Equation (3.15), a fractional-order linear time-invariant system can also be represented by a state-space model:

$$\begin{cases} D^q x(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{3.17}$$

3.4.2. FOMCON Toolbox

"Fractional-order Modeling and Control (FOPID)" is a MATLAB toolbox dedicated to research and development of applications of fractional calculus to modeling and control of complex dynamic systems [19].

3.4.2.1. Toolbox Features

In FOMCON, the main object of analysis is the fractional-order transfer function given by the Equation (3.16). The toolbox focuses on the Single Input Single Output (SISO), Linear Time Invariant (LTI) systems [20].

The toolbox is comprised of the following modules:

• Main module (fractional system analysis).

• Identification module (system identification in time and frequency domains).

• Control module (FOPID controller design, tuning and optimization tools as well as some additional features).

All the modules are interconnected and can be accessed from the main module GUI; furthermore, a Simulink blockset is provided in the toolbox allowing more complex modeling tasks to be carried out. General approach to block construction was used where applicable.

The following blocks are currently realized:

• General fractional-order operator.

• Fractional integrator and differentiator.

• $PI^{\lambda}D^{\mu}$ controller.

• Fractional transfer function.

3.4.2.2. Toolbox Dependencies

The FOMCON toolbox relies on the following MATLAB set of tools:

• Control System toolbox, required for most features.

• Optimization toolbox, required for time-domain identification and integer-order PID tuning for common process model approximation.

3.4.3. $PI^{\lambda}D^{\mu}$ Controller Design

A $PI^{\lambda}D^{\mu}$ generated control signal takes the following form [21]:

$$U(t) = K_p e(t) + K_d D^{\mu} e(t) + k_i D^{\lambda} e(t) \tag{3.18}$$

Where U is a generic control variable, e is the error between the task and the process output y, K_P is the proportional coefficient, K_I is the integral coefficient, K_D is the derivative coefficient, D^{λ} fractional order integral and D^{μ} Fractional order derivative.

$\mu = 1$, $\lambda = -1$ for Conventional-order PID, the selections of $\lambda = -1$, $\mu = 0$, and $\lambda = 0$, $\mu = 1$ respectively corresponds conventional PI & PD controllers.

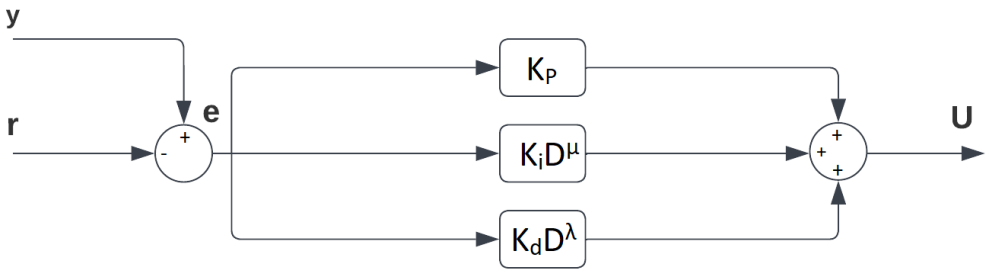

Figure 3-8 Fraction PID Controller Structure

Similar to the PID control, $PI^{\lambda}D^{\mu}$ controllers must be constructed for both altitude and position control.

3.4.3.1. Altitude Control

- Roll Controller

$$U_2(t) = K_{p\phi}(\phi_d - \phi) + K_{d\phi} D^{\mu}(\phi_d - \phi) + k_{i\phi} D^{\lambda}(\phi_d - \phi) \tag{3.19}$$

Where: $K_{P\Phi}$ is the proportional coefficient, $K_{I\Phi}$ is the integral coefficient and $K_{D\Phi}$ is the derivative coefficient. D^{λ} Fractional order Integral D^{μ} Fractional order derivative.

- Pitch Controller

$$U_3(t) = K_{p\theta}(\theta_d - \theta) + K_{d\theta} D^{\mu}(\theta_d - \theta) + k_{i\theta} D^{\lambda}(\theta_d - \theta) \tag{3.20}$$

Where: $K_{P\Theta}$ is the proportional coefficient, $K_{I\Theta}$ is the integral coefficient and $K_{D\Theta}$ is the derivative coefficient. D^{λ} Fractional order Integral D^{μ} Fractional order derivative.

- Yaw Controller

$$U_4(t) = K_{p\psi}(\psi_d - \psi) + K_{d\psi}D^{\mu}(\psi_d - \psi) + k_{i\psi}D^{\lambda}(\psi_d - \psi) \quad (3.21)$$

Where: $K_{P\Psi}$ is the proportional coefficient, $K_{I\Psi}$ is the integral coefficient and $K_{D\Psi}$ is the derivative coefficient. D^{λ} Fractional order Integral D^{μ} Fractional order derivative.

- Altitude/Height Controller

$$U_1 = (K_{pz}(z_d - z) + K_{dz}D^{\mu}(z_d - z) + K_{iz}D^{\lambda}(z_d - z) + g) \times \frac{m}{cos(\theta)cos(\phi)} \quad (3.22)$$

Where: K_{Pz} is the proportional coefficient, K_{Iz} is the integral coefficient and K_{Dz} is the derivative coefficient. D^{λ} Fractional order Integral D^{μ} Fractional order derivative.

3.4.3.2. Trajectory Control

Similar to the PID section, fractional order PID control is developed for our trajectory. For x and y, we use two FOPID controllers to calculate the desired acceleration:

$$\begin{cases} \ddot{x}_d = K_{px}(x_d - x) + K_{dx}D^{\mu}(x_d - x) + K_{ix}D^{\lambda}(x_d - x) \\ \ddot{y}_d = K_{py}(y_d - y) + K_{dy}D^{\mu}(y_d - y) + K_{iy}D^{\lambda}(y_d - y) \end{cases} \quad (3.23)$$

Substituting equations (3.22) in equation (3.2):

$$\begin{bmatrix} \phi_d \\ \\ \theta_d \end{bmatrix} = \frac{1}{g}\begin{bmatrix} sin(\psi) & -cos(\psi) \\ \\ cos(\psi) & sin(\psi) \end{bmatrix}\begin{bmatrix} K_{px}(x_d - x) + K_{dx}D^{\mu}(x_d - x) + K_{ix}D^{\lambda}(x_d - x) \\ \\ K_{py}(y_d - y) + K_{dy}D^{\mu}(y_d - y) + K_{iy}D^{\lambda}(y_d - y) \end{bmatrix} \quad (3.24)$$

3.5. Sliding Mode Control

Sliding Mode Control (SMC) is a form of nonlinear control known for its robustness, accuracy, ease of tuning and implementation, and because of these reasons it sees high use in nonlinear systems control [22].

The method applies a discontinuous control signal to drive the system to slide along a particular surface named the sliding surface. This make it very appealing as the objective in our case is to control the quadrotor such as it follows a given trajectory.

The SMC approach is considered a two-part control. The first part involves the design of a sliding surface so that the sliding motion satisfies design specifications. The second is concerned with the selection of a control law that will make the switching surface attractive to the system state.

3.5.1. Sliding Mode Controller Design

For the first part of choosing the sliding surface shape, the general form of equation (3.24) was proposed [23]:

$$S = (\lambda + \frac{d}{dt})^{n-1} e \tag{3.25}$$

Where $\lambda$is a tuning parameter is greater than zero, n is the order of the controlled system and e is the tracking error defined as:

$$e = r - r_d \tag{3.26}$$

Where r is the process and r_d is the desired value.

The quadrotor model of our case is of order n=2, equation (3.25) then gives the following sliding surface:

$$S = \lambda e + \dot{e} \tag{3.27}$$

For the second part concerning the control law, we define u(t) as follows:

$$u(t) = u_{eq}(t) + u_d(t) \tag{3.28}$$

Here we can see that $u(t)$ is divided into two parts:

A continuous part $u_{eq}(t)$ that determines the behavior of the system when an ideal sliding regiment is established, and is calculated given the following surface conditions:

$$\begin{cases} S = 0 \\ \dot{S} = 0 \end{cases} \tag{3.29}$$

A discontinuous $u_d(t)$ that compensates the uncertainties of the model, and is designed according to a Lyapunov candidate function V that must be positive definite and its derivative must be negative definite:

$$V = \frac{1}{2}S^2 > 0 \tag{3.30}$$

$$\dot{V} = S\dot{S} < 0 \tag{3.31}$$

The above conditions can be satisfied if:

$$u_d(t) = -\varepsilon\, sgn\big(S(t)\big) - KS(t) \tag{3.32}$$

Where:

$$sgn(t) = \begin{cases} -1 & if\ t < 0 \\ 0 & if\ t = 0 \\ 1 & if\ t > 0 \end{cases} \tag{3.33}$$

And $\varepsilon > 0$ and K > 0 are the sliding surface exponential approach coefficients.

3.5.1.1. Altitude Control

As mentioned earlier, the tracking error represented the difference between the current value and the desired value and is written as:

$$e = r - r_d \tag{3.34}$$

The sliding surface then becomes:

$$S_r = \lambda_r e_r + \dot{e_r} \tag{3.35}$$

And the exponential reaching law becomes:

$$\dot{S_r} = -\varepsilon_r\, \mathrm{sgn}(S_r) - k_r S_r = \lambda_r(\dot{r} - \dot{r_d}) + (\ddot{r} - \ddot{r_d}) \tag{3.36}$$

Solving for $\ddot{r}$:

$$\ddot{r} = -\varepsilon_r\, \mathrm{sgn}(S_r) - k_r S_r - \lambda_r(\dot{r} - \dot{r_d}) + \ddot{r_d} \tag{3.37}$$

According to equations (2.17) and (2.19), altitude controller commands can be written below.

- Roll Controller

$$U_2 = \left[-\lambda_\phi\big(\dot{\phi} - \dot{\phi_d}\big) - \dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) + \ddot{\phi_d} - \varepsilon_\phi\, sgn\big(S_\phi\big) - K_\phi S_\phi\right]\frac{I_x}{l} \tag{3.38}$$

- Pitch Controller

$$U_3 = \left[-\lambda_\theta\big(\dot{\theta} - \dot{\theta_d}\big) - \dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) + \ddot{\theta}_d - \varepsilon_\theta\, sgn(S_\theta) - K_\theta S_\theta\right]\frac{I_y}{l} \tag{3.39}$$

- Yaw Controller

$$U_4 = \left[-\lambda_\psi\big(\dot{\psi} - \dot{\psi_d}\big) - \dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{I_z}\right) + \ddot{\psi}_d - \varepsilon_\psi\, sgn\big(S_\psi\big) - K_\psi S_\psi\right]\frac{I_z}{l} \tag{3.40}$$

- Altitude/Height Controller

$$U_1 = [-\lambda_z(\dot{z} - \dot{z_d}) + g + \ddot{z_d} - \varepsilon_z\, sgn(S_z) - K_z S_z]\frac{m}{cos(\theta)cos(\phi)} \tag{3.41}$$

3.5.1.2. Position Control

Similar to previous positions controls, values of ϕ_d and θ_d must first be calculated, in this case this is done by applying sliding mode controllers to x and y to find desired values of acceleration, then converting them to desired roll and pitch values.

$$\begin{cases} \ddot{x} = -\lambda_x(\dot{x} - \dot{x_d}) + \ddot{x_d} - \varepsilon_x \, sgn(S_x) - K_x S_x \\ \ddot{y} = -\lambda_y(\dot{y} - \dot{y_d}) + \ddot{y_d} - \varepsilon_y \, sgn(S_y) - K_y S_y \end{cases} \tag{3.42}$$

Substituting in equation (3.2):

$$\begin{bmatrix} \phi_d \\ \\ \theta_d \end{bmatrix} = \frac{1}{g} \begin{bmatrix} sin(\psi) & -cos(\psi) \\ \\ cos(\psi) & sin(\psi) \end{bmatrix} \begin{bmatrix} -\lambda_x(x - \dot{x_d}) + \ddot{z_d} - \varepsilon_x \, sgn(S_x) - K_x S_x \\ \\ -\lambda_y(y - \dot{y_d}) + \ddot{y_d} - \varepsilon_y \, sgn(S_y) - K_y S_y) \end{bmatrix} \tag{3.43}$$

3.5.2. SMC Chattering Effect

The discontinuous nature of the sign function in the SMC creates an effect that is known as chattering. The chattering the effect is the phenomenon of finite- frequency, finite-amplitude oscillations appearing in many sliding mode implementations. This effect can be observed in Figure 3-9 and is caused by the high-frequency switching of the sliding mode controller.

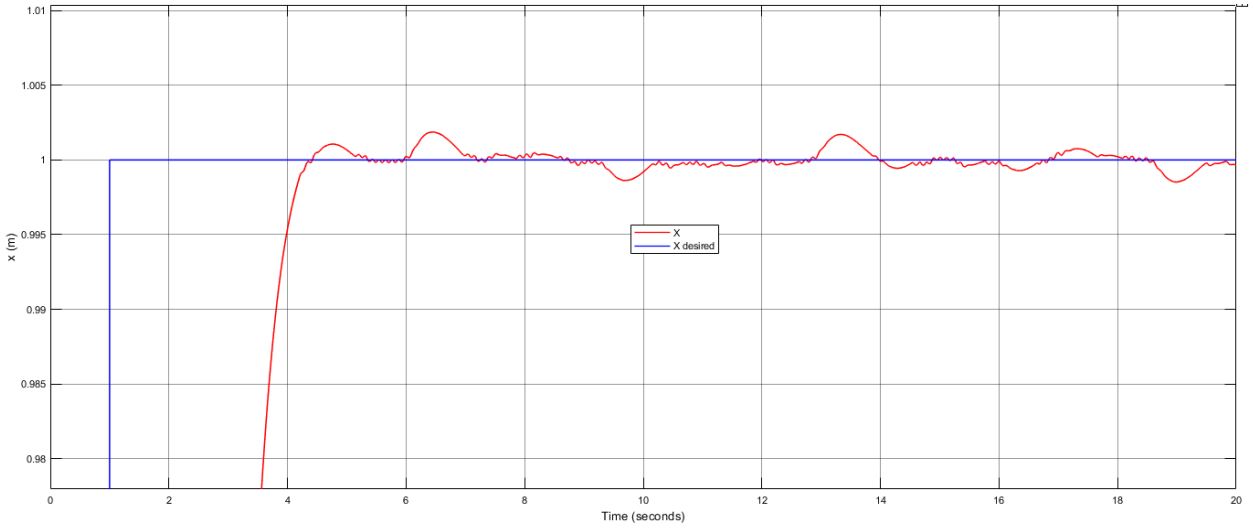

Figure 3-9 X step input response showing chattering

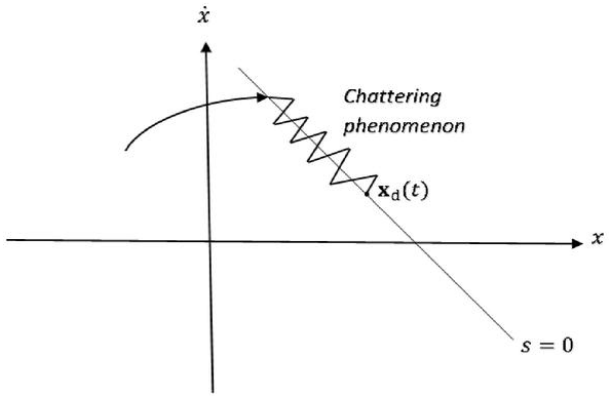

Figure 3-10 Chattering Phenomenon

This effect must be reduced as it causes wear of mechanical parts and possible actuator damage as well as power loss.

There are many approaches to reducing the effect; one such approach is replacing the discontinuous sign function with a continuous function. The chosen continuous function is the saturation function defined as:

$$sat(x) = \begin{cases} s \; if \; |s| \leq 1 \\ sgn(s) \; if \; |s| > 1 \end{cases} \qquad (3.44)$$

This was implemented in the designed SMC controller and we can observe change in the same signal of Figure 3-9 in Figure 3-11:

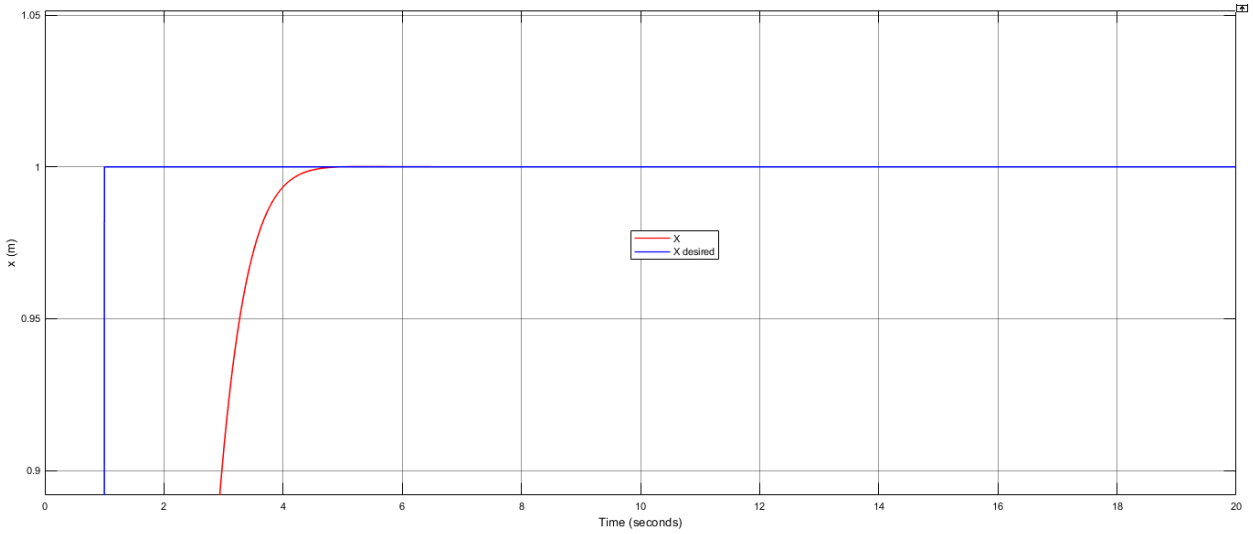

Figure 3-11 X step input response showing reduction of chattering

3.6. Conclusion

In this chapter we've tackled the controlling problem of our quadrotor model, where three controllers were designed and implemented and their results are shown and discussed in the next section.

4. Chapter 4: Results and Discussion

4.1. Introduction

This chapter is a summary of results for the different tests performed on the quadrotor model with the different controllers previously developed in SIMULINK and a brief discussion into them.

All the designed controllers were tested for their step input response and their trajectory response. The chosen trajectory for the test is:

$$\begin{cases} x = 0.5\sin(t) \\ y = 0.5\sin(0.5t) \\ z = 0.5t \\ Psi = 0.1 \end{cases}$$

The following code is run before simulations to establish the quadrotor parameters in the workspace:

```
clc;
clear all;
Ix = 7.5*10^(-3);  % Quadrotor moment of inertia around X axis
Iy = 7.5*10^(-3);  % Quadrotor moment of inertia around Y axis
Iz = 1.3*10^(-2);  % Quadrotor moment of inertia around Z axis
b = 3.13*10^(-5);  % Thrust factor
d = 7.5*10^(-7);  % Drag factor
l = 0.23;  % Distance to the center of the Quadrotor
m = 0.96;  % Mass of the Quadrotor in Kg
g = 9.81;   % Gravitational acceleration
```

4.2. PID Simulation and Results

The PID controllers are constructed and tuned using MATLAB's tuning tool. The Gains are summarized in table (4-1):

	Kp	Ki	Kd
X	50	0.5	15
Y	25	0.2	10
Z	4	0.2	4
Phi	9	4.5	0.5
Theta	9	7.5	1.5
Psi	4.5	3	1.5

Table 4-1 Chosen Gains for PID Control

The controller is then tested with the trajectory previously selected and the results are as follows:

Step input test results:

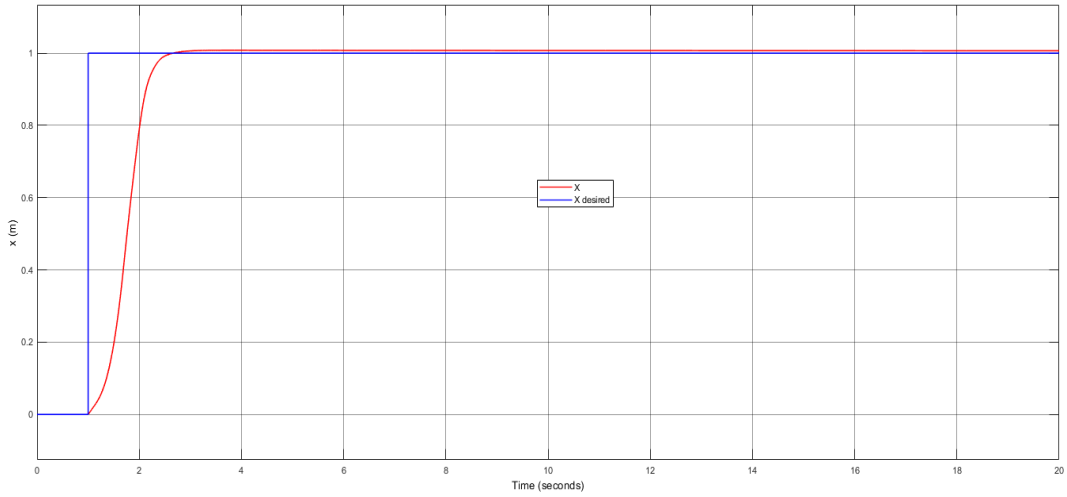

Figure 4-1 X step input response (PID)

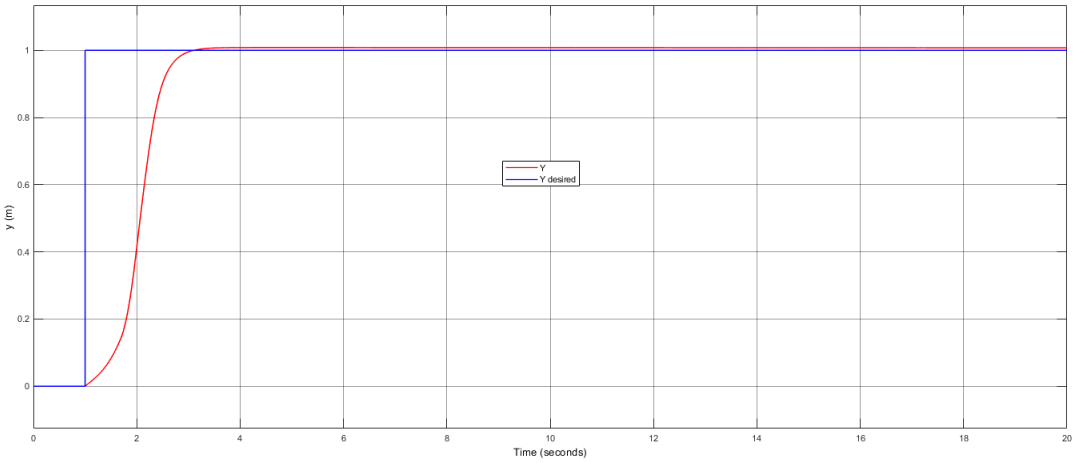

Figure 4-2 Y step input response (PID)

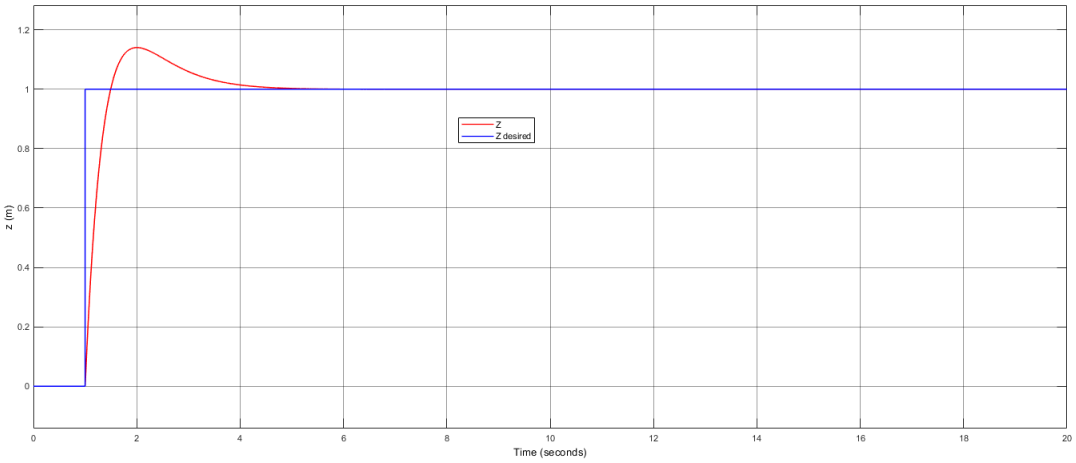

Figure 4-3 Z step input response (PID)

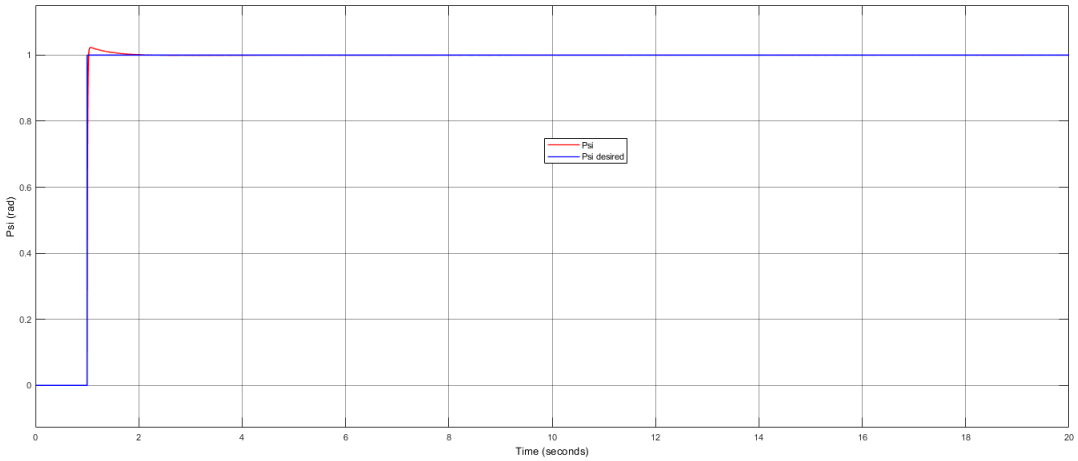

Figure 4-4 Psi step input response (PID)

	x(t)	y(t)	z(t)	Psi(t)
Rise Time (s)	0.785	0.9455	0.3603	0.0167
Overshoot	0.1397	0.0792	14.05	2.3339
Settling time (s)	2.438	2.8871	3.7876	1.1358

Table 4-2 Characteristic performances to a step input for PID Control

Trajectory test results:

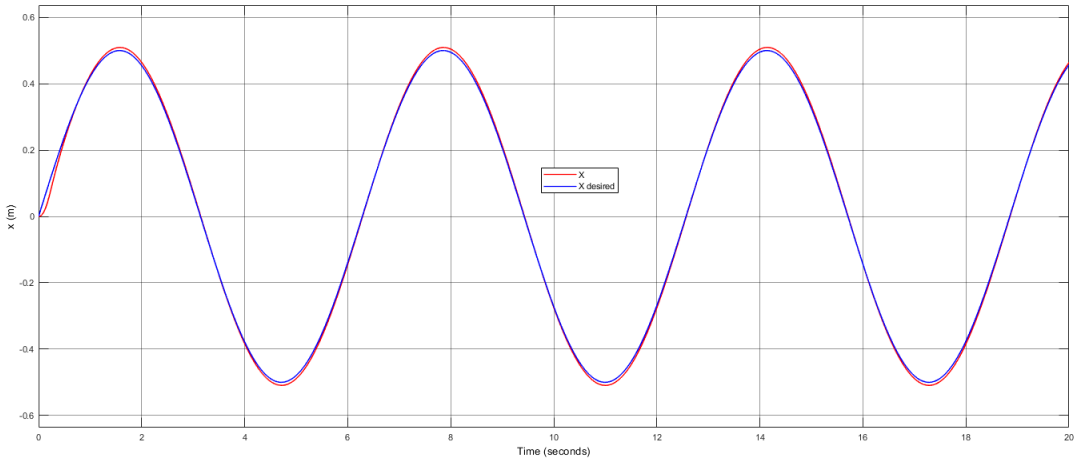

Figure 4-5 X Trajectory Response (PID)

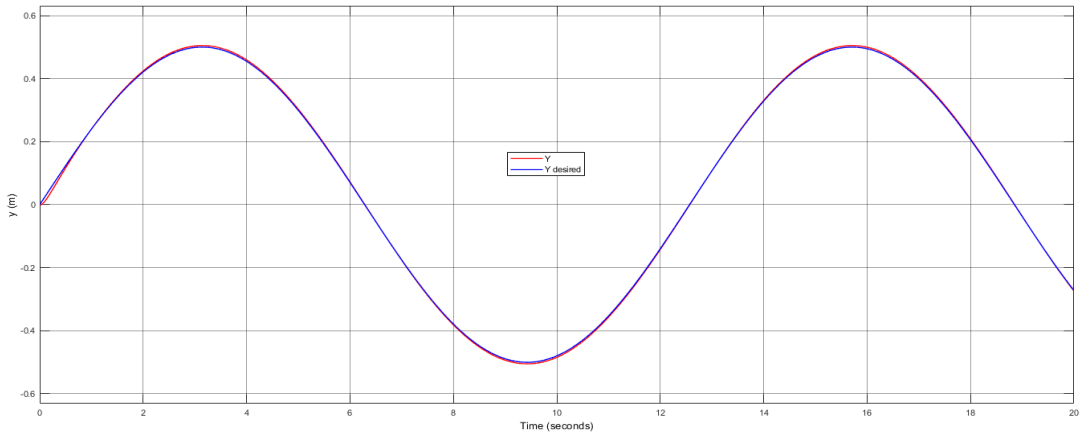

Figure 4-6 Y Trajectory Response (PID)

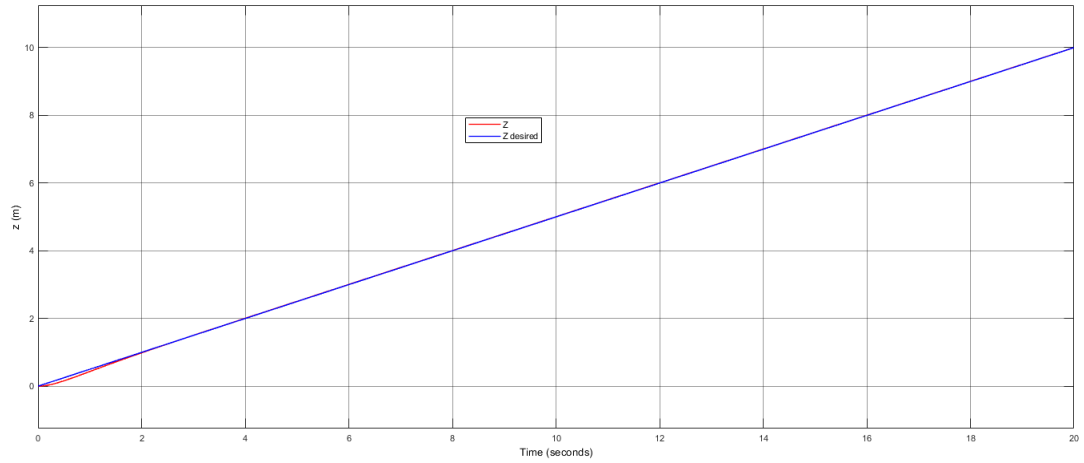

Figure 4-7 Z Trajectory Response (PID)

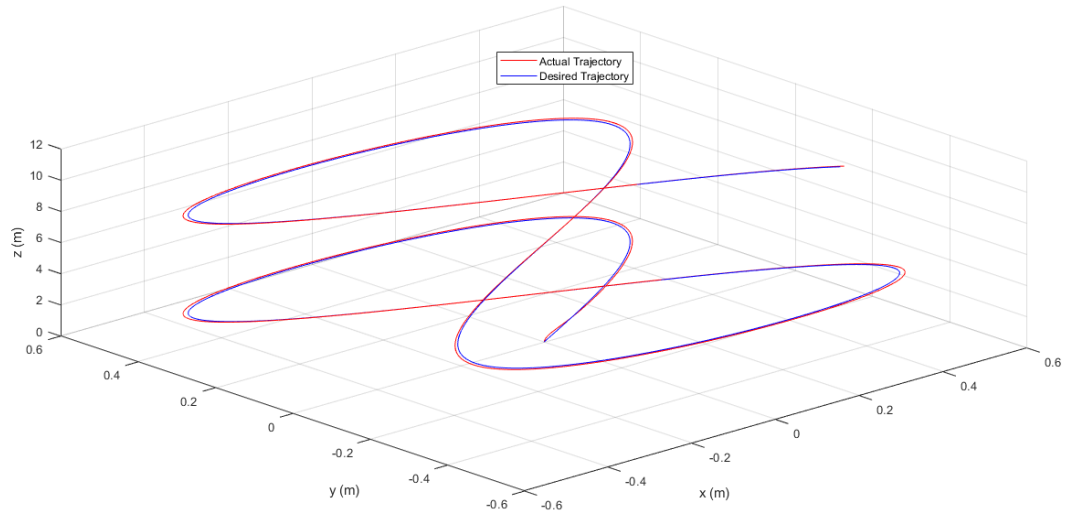

Figure 4-8 X-Y-Z Trajectory Response (PID)

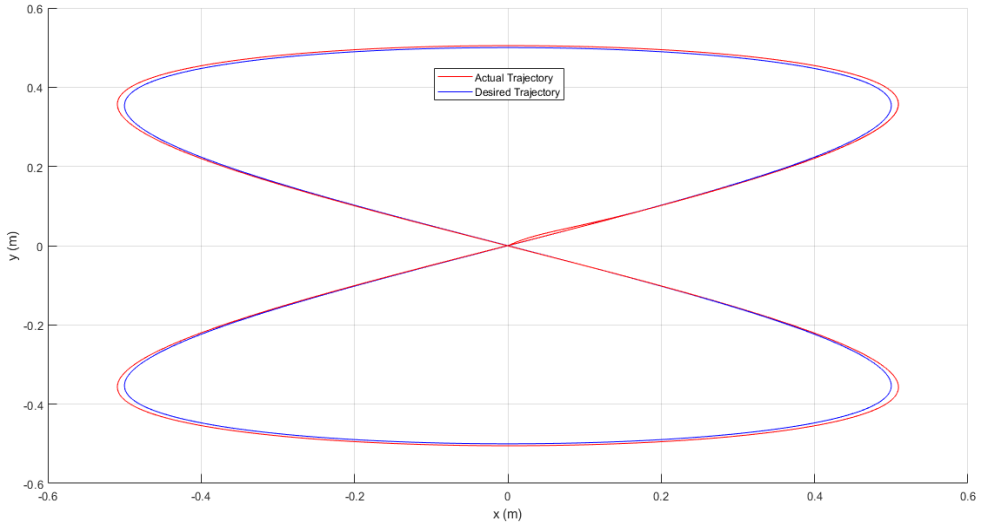

Figure 4-9 X-Y Trajectory Response (PID)

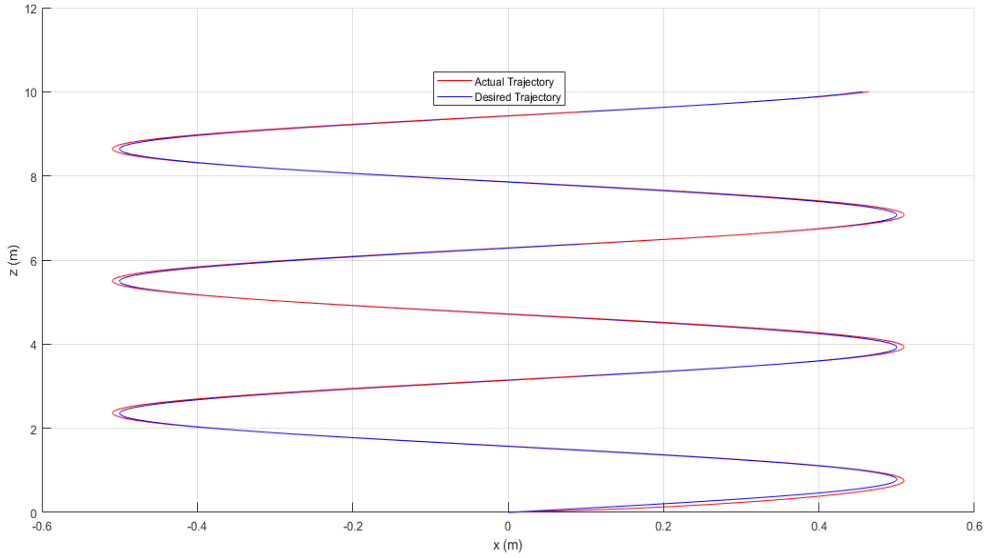

Figure 4-10 X-Z Trajectory Response (PID)

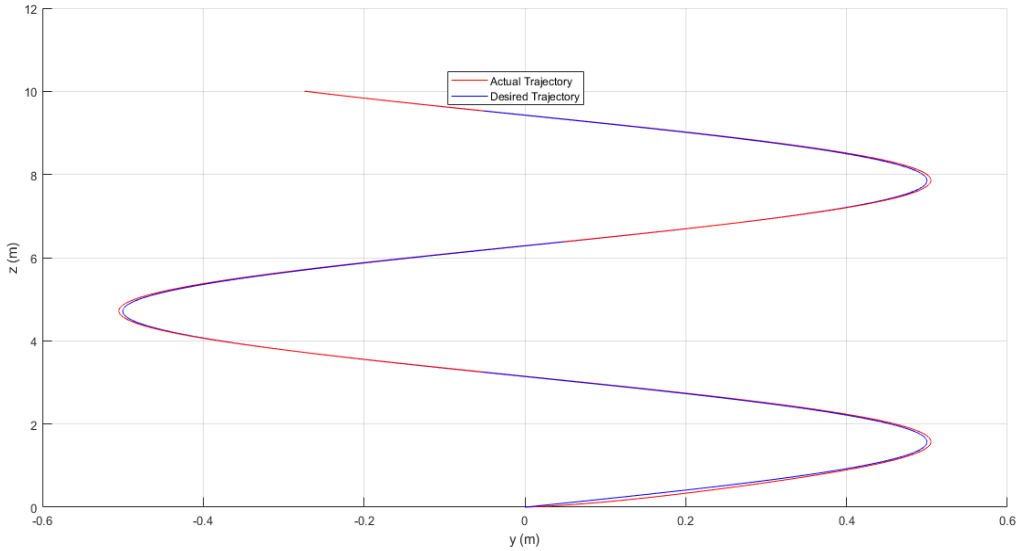

Figure 4-11 Y-Z Trajectory Response (PID)

4.3. FOPID Simulation and Results

The gains chosen for our control in this case are shown in table (4-2):

	Kp	Ki	Kd	Lambda	mu
X	100	1	30	-0.5	0.9
Y	70	1	30	-0.5	0.95
Z	4	0.2	4	-0.5	1
Phi	9	4	1	-1.2	0.8
Theta	9	7.5	0.7	-0.8	1.2
Psi	4.5	3	1.5	-1.5	1.2

Table 4-3 Chosen Gains for FOPID Control

The results for step input tests are as follows:

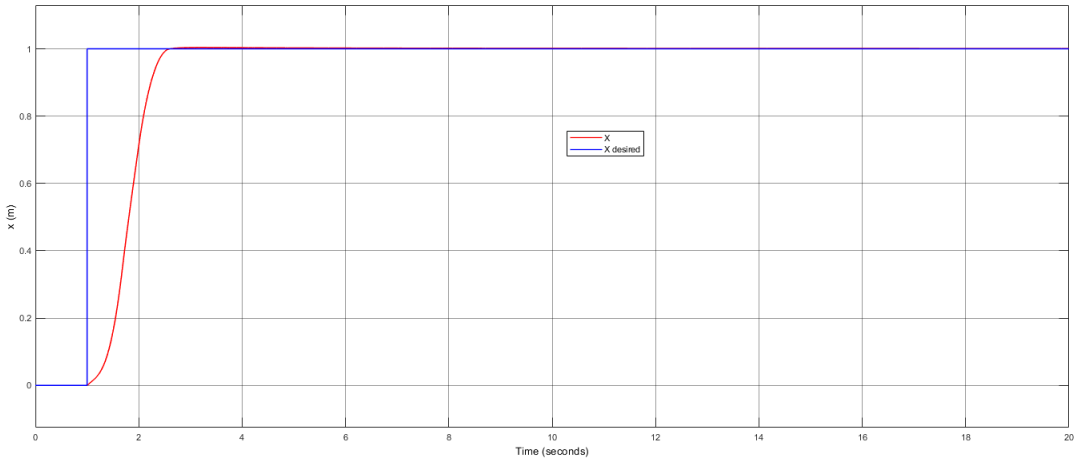

Figure 4-12 X step input response (FOPID)

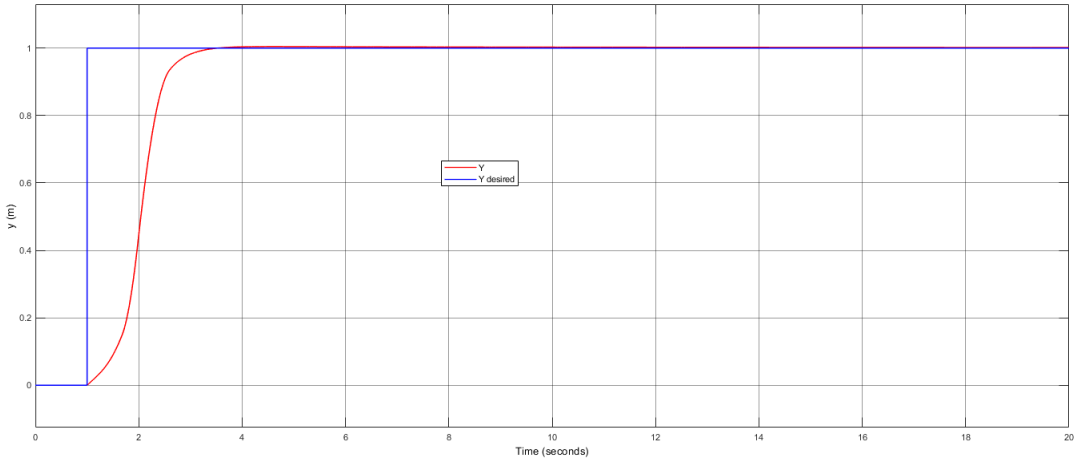

Figure 4-13 Y step input response (FOPID)

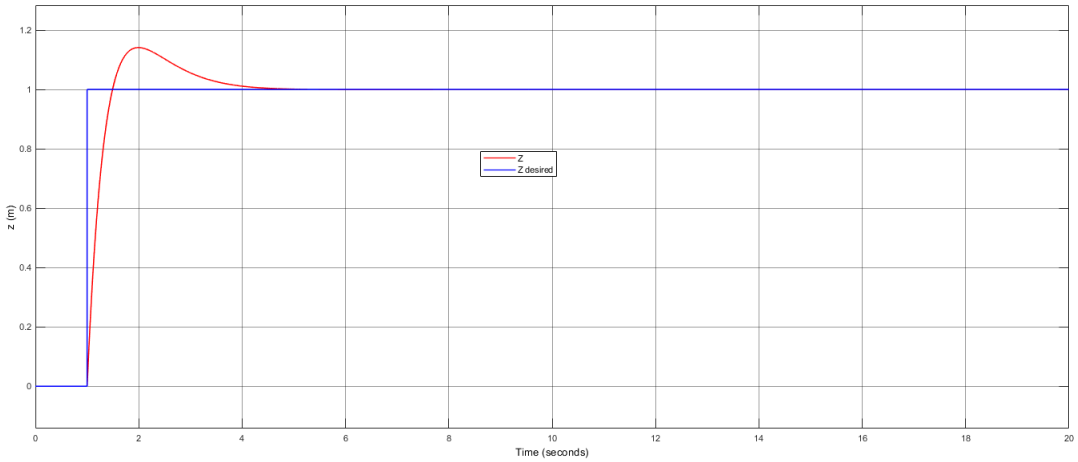

Figure 4-14 Z step input response (FOPID)

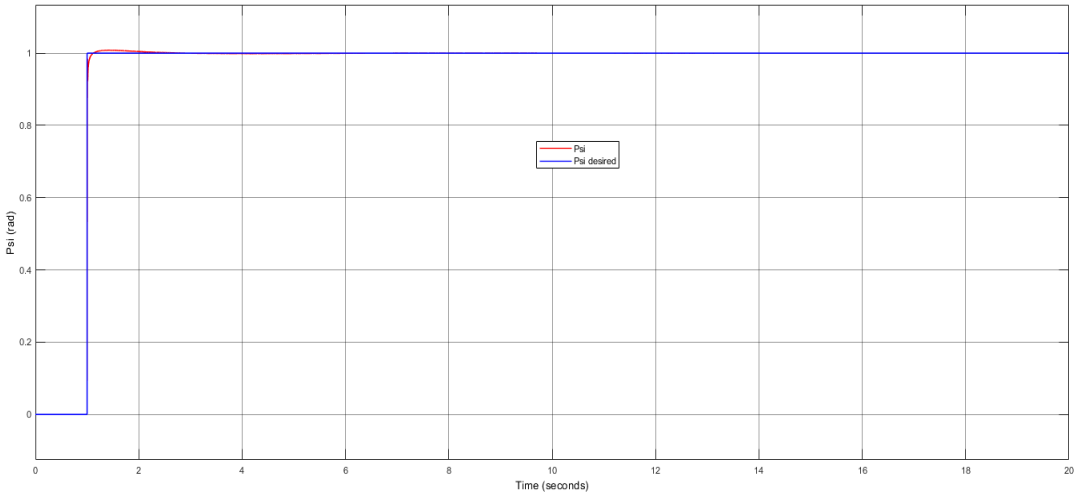

Figure 4-15 Psi step input response (FOPID)

	x(t)	y(t)	z(t)	Psi(t)
Rise Time (s)	0.7561	0.8871	0.3593	0.004
Overshoot	0.0199	0.2031	14.12	0.7943
Settling time (s)	2.39	2.5816	3.666	1.0399

Table 4-4 Characteristic performances to a step input for FOPID Control

Trajectory test results:

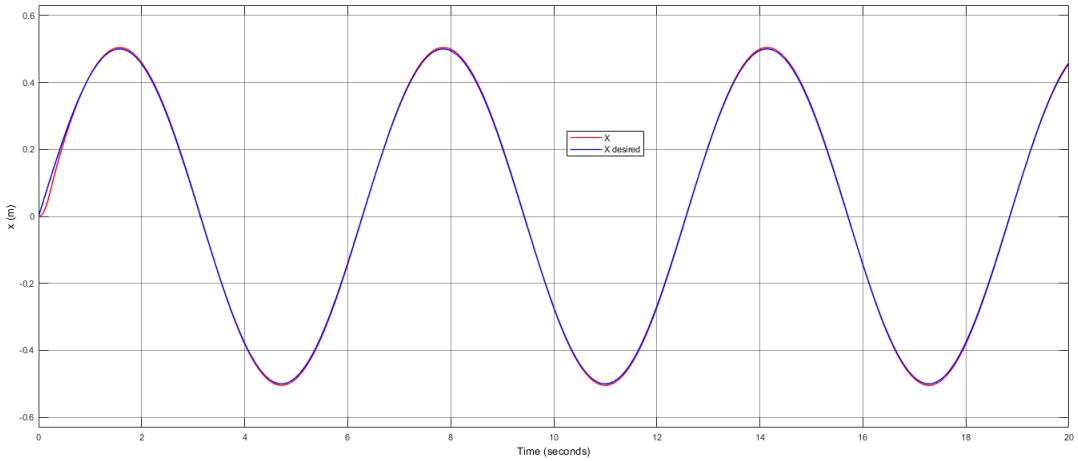

Figure 4-16 X Trajectory Response (FOPID)

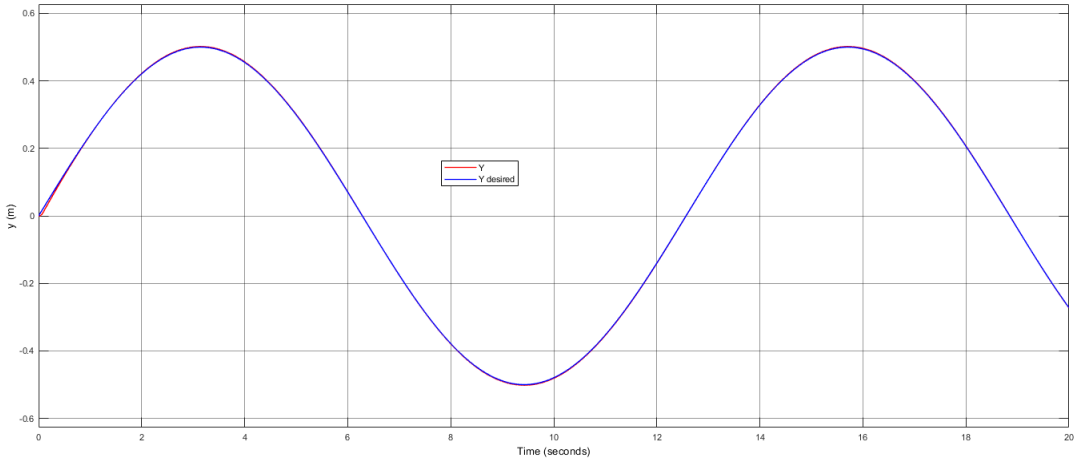

Figure 4-17 Y Trajectory Response (FOPID)

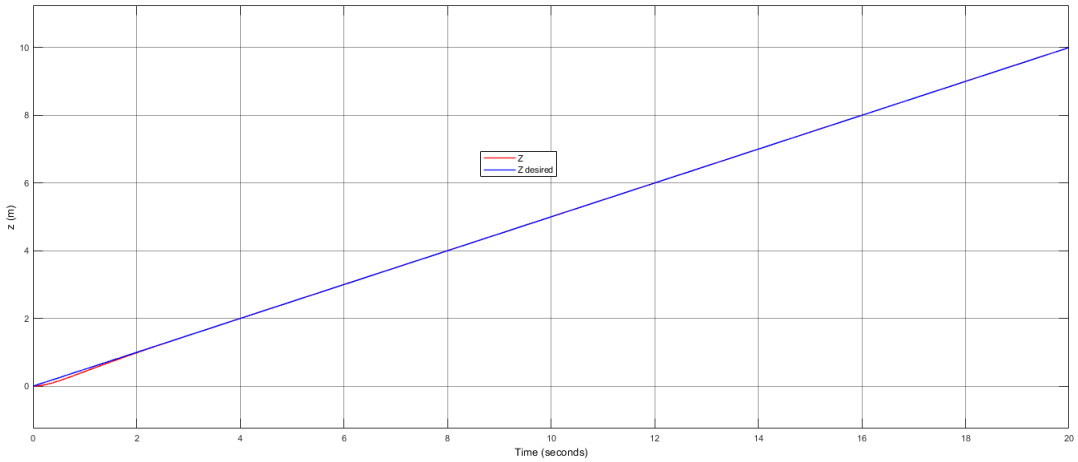

Figure 4-18 Z Trajectory Response (FOPID)

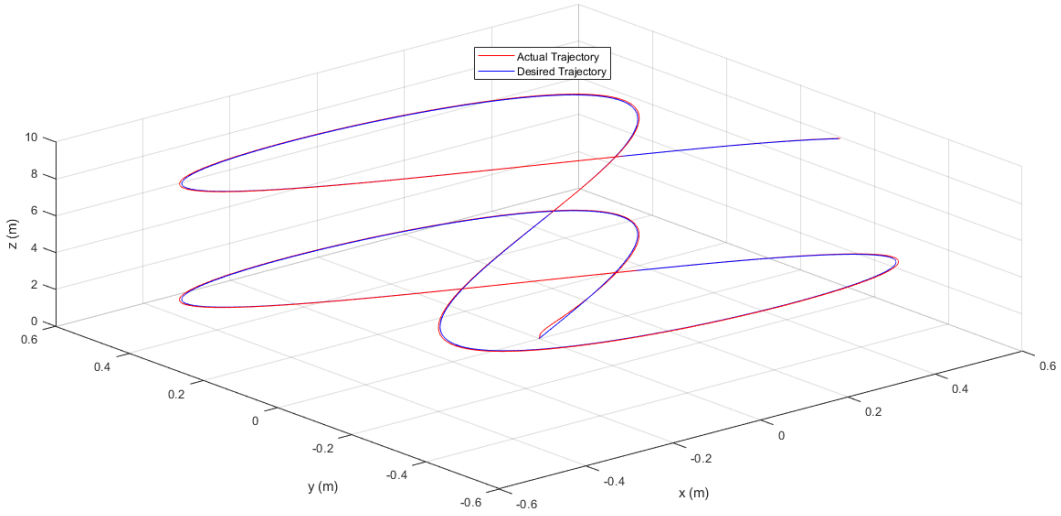

Figure 4-19 X-Y-Z Trajectory Response (FOPID)

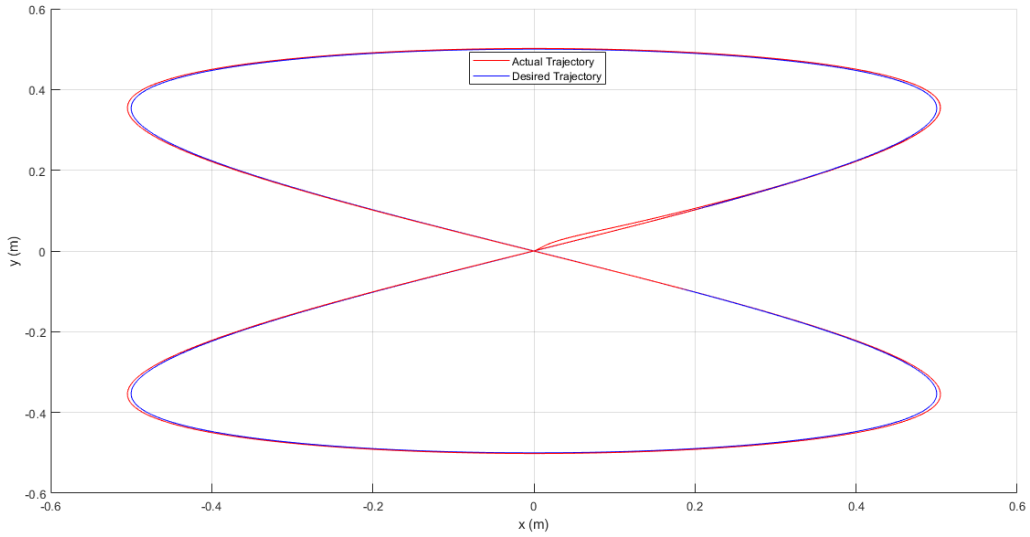

Figure 4-20 X-Y Trajectory Response (FOPID)

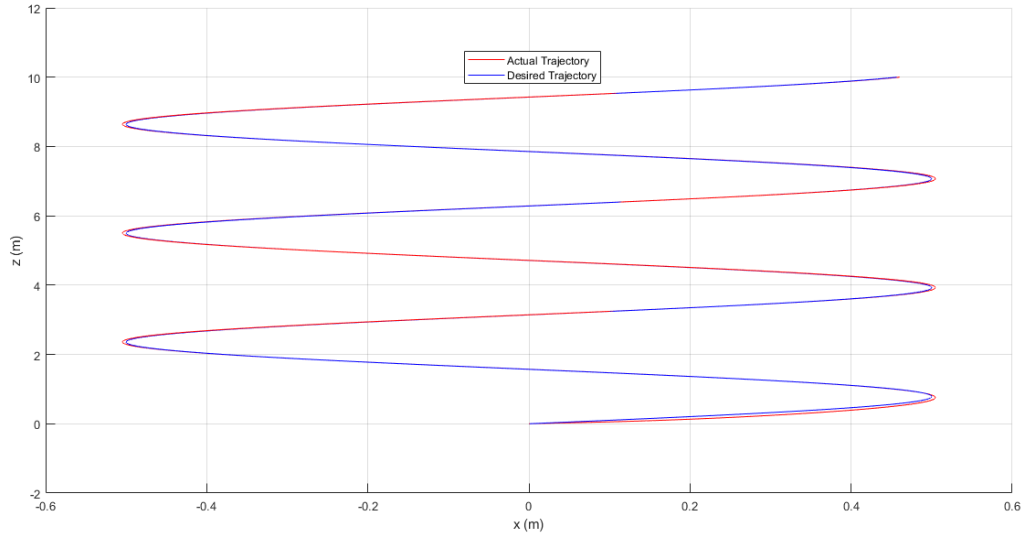

Figure 4-21 X-Z Trajectory Response (FOPID)

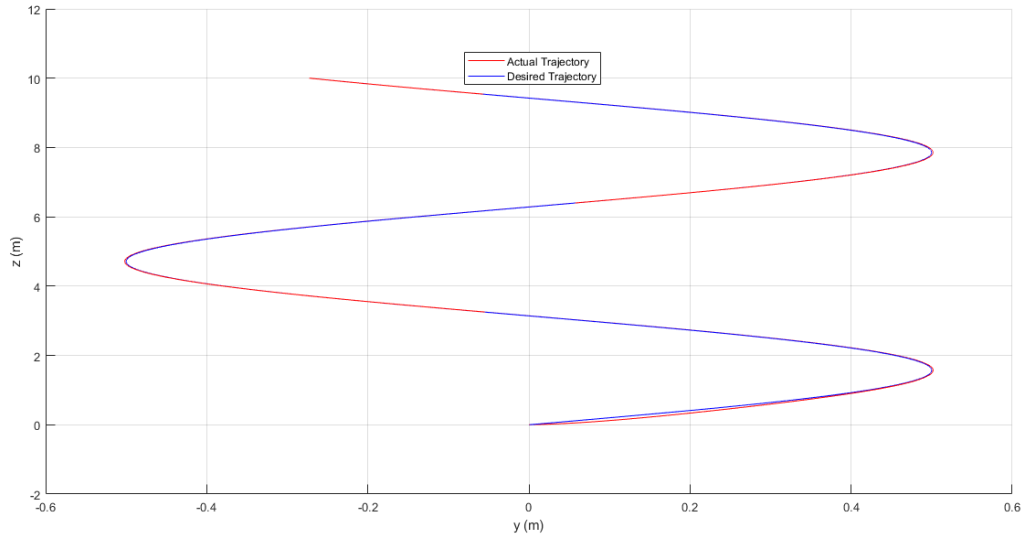

Figure 4-22 Y-Z Trajectory Response (FOPID)

4.4. SMC Simulation and Results

The gains selected for this controller are in table (4-3):

	Lambda	e	K
X	20	0.1	4
Y	20	0.1	5
Z	4	0.1	10
Phi	5	0.1	7.5
Theta	5	0.1	5
Psi	6	0.1	8

Table 4-5 Chosen gains for Sliding Mode Control

The results of the step input test are as follows:

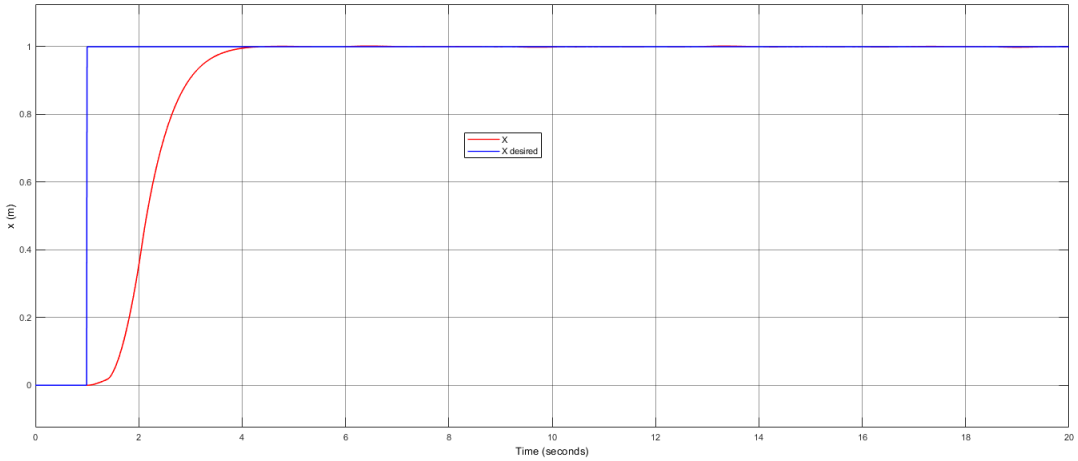

Figure 4-23 X step input response (SMC)

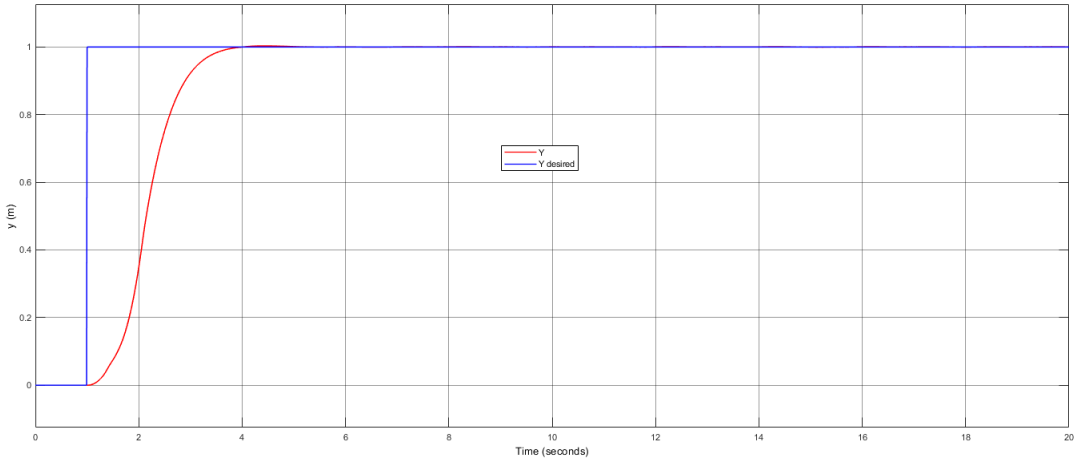

Figure 4-24 Y step input response (SMC)

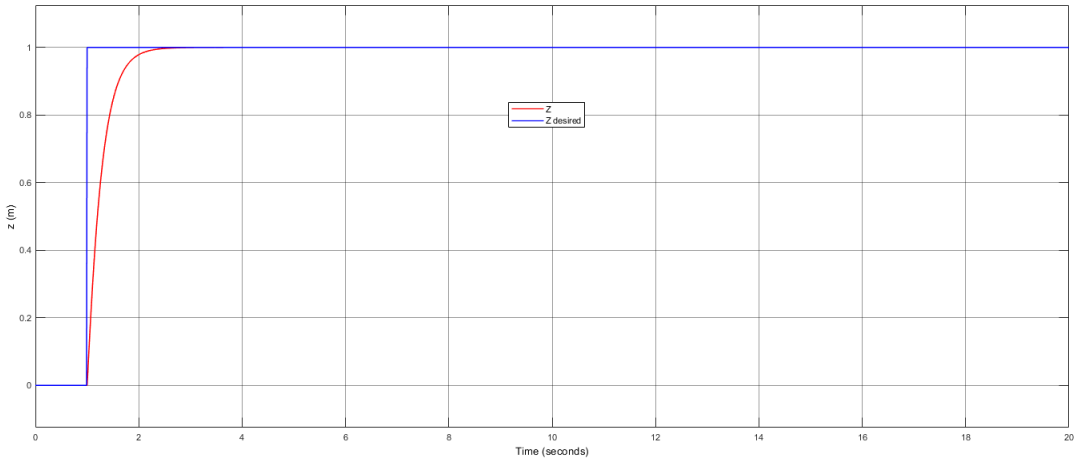

Figure 4-25 Z step input response (SMC)

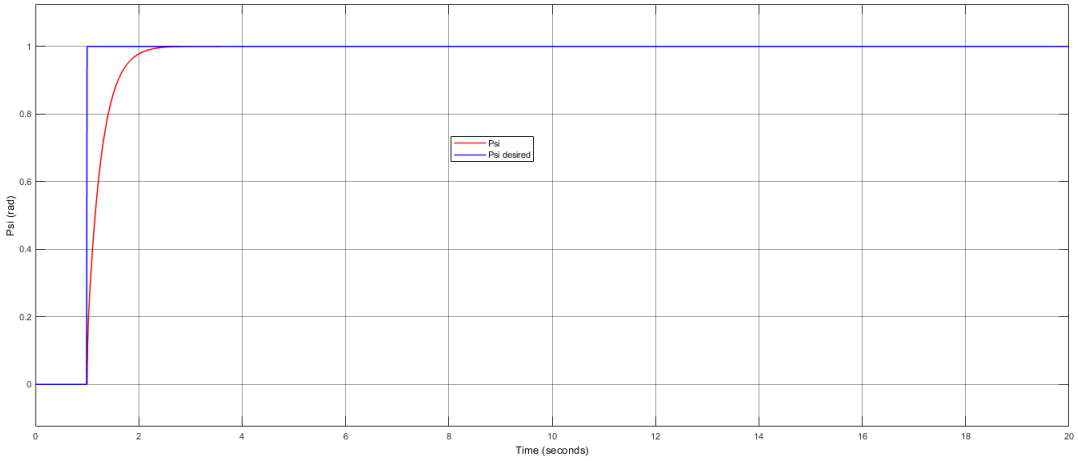

Figure 4-26 Psi step input response (SMC)

	x(t)	y(t)	z(t)	Psi(t)
Rise Time (s)	1.3121	1.3055	0.5778	0.5908
Overshoot	0.2204	0.3126	0.001	0.0009
Settling time (s)	3.59	3.4565	2.0181	2.0265

Table 4-6Characteristic performances to a step input for Sliding Mode Control

Trajectory test results are as follows:

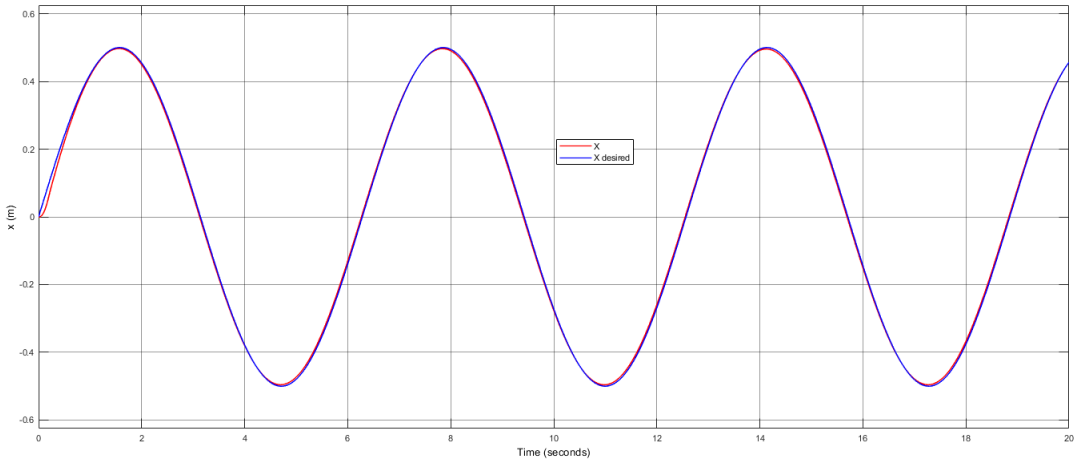

Figure 4-27 X Trajectory Response (SMC)

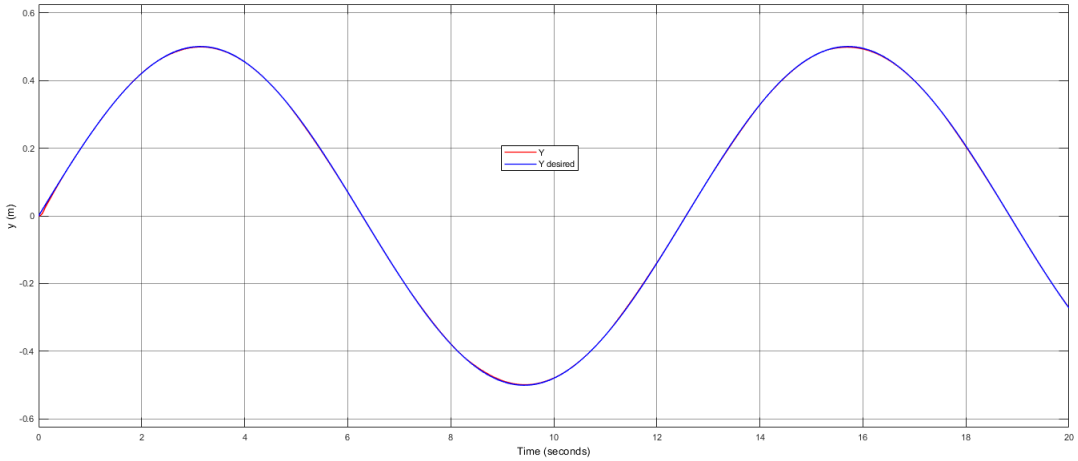

Figure 4-28 Y Trajectory Response (SMC)

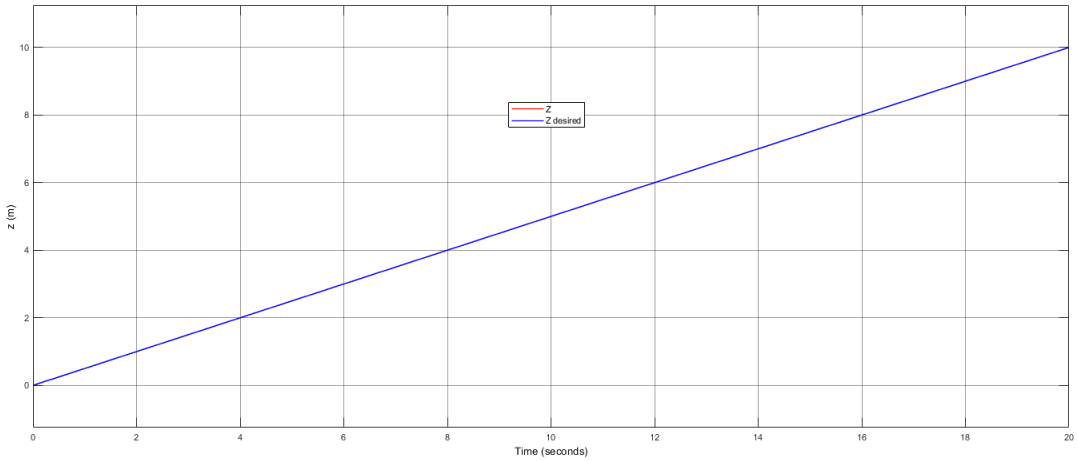

Figure 4-29 Z Trajectory Response (SMC)

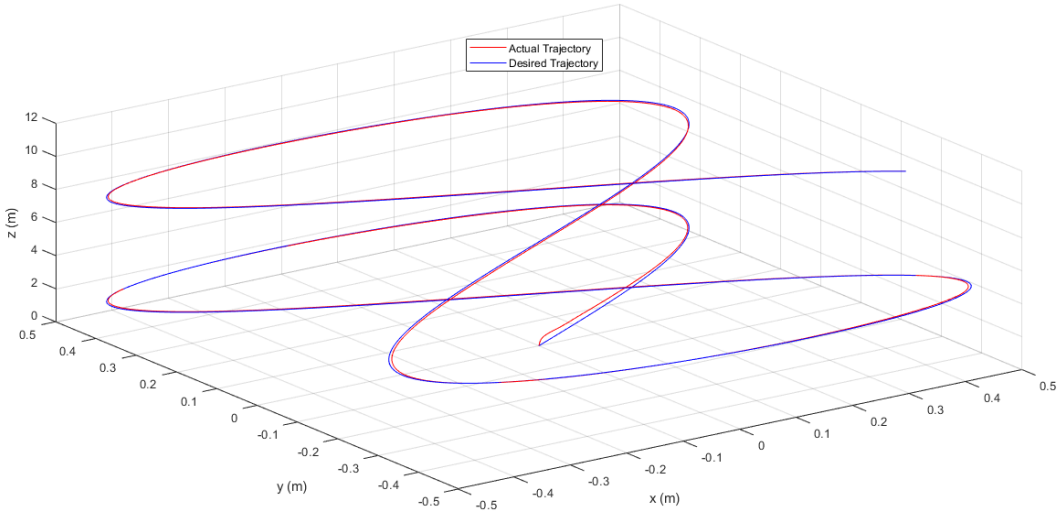

Figure 4-30 X-Y-Z Trajectory Response (SMC)

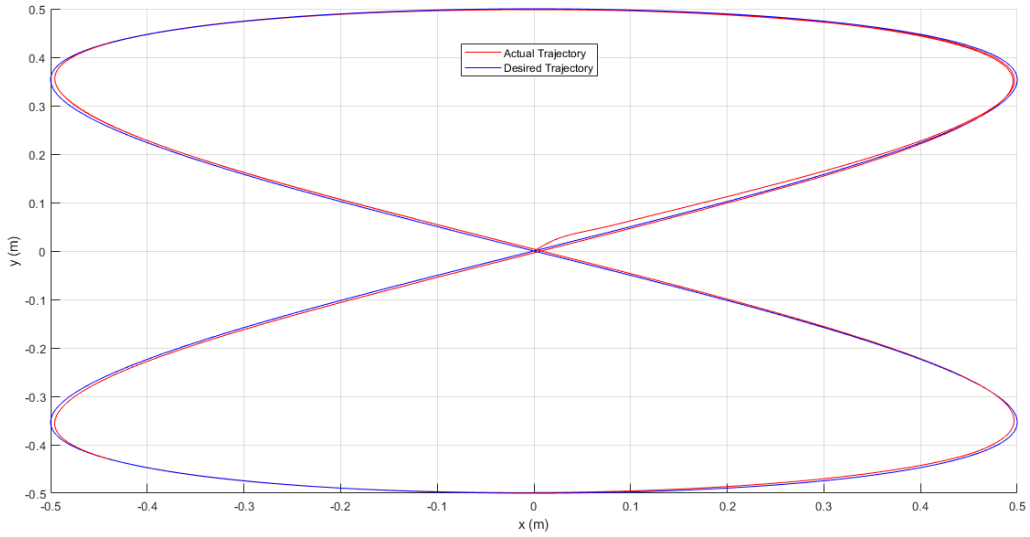

Figure 4-31 X-Y Trajectory Response (SMC)

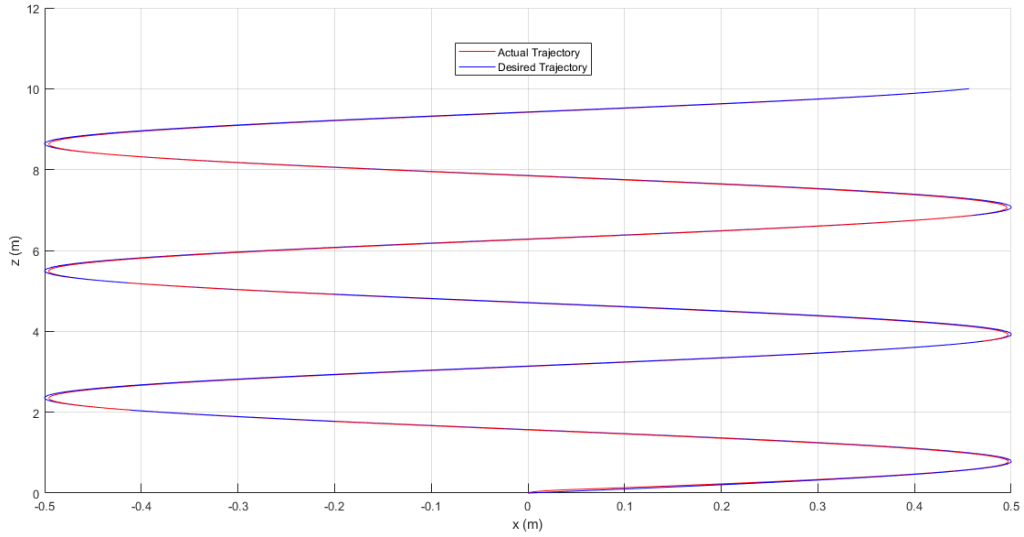

Figure 4-32 X-Z Trajectory Response (SMC)

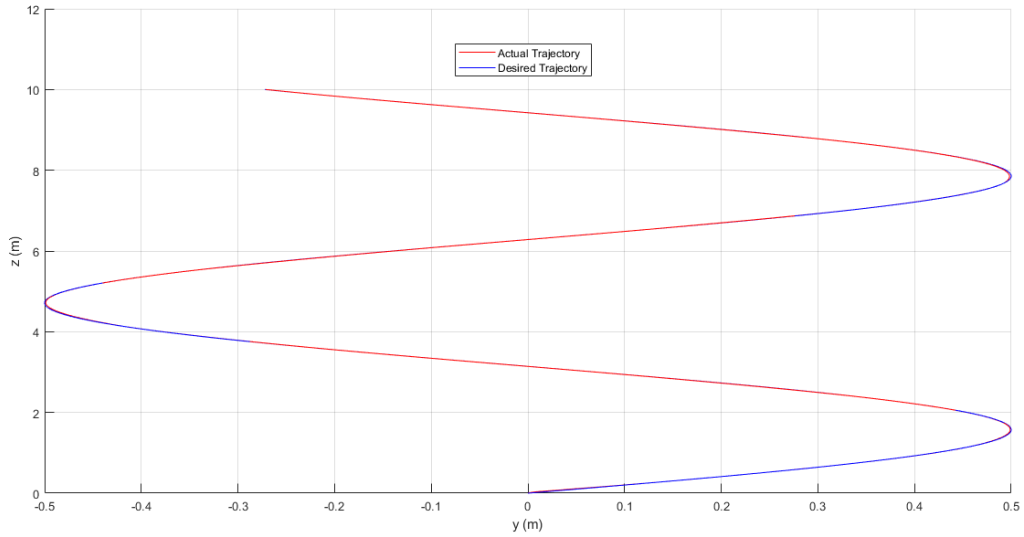

Figure 4-33 Y-Z Trajectory Response (SMC)

4.5.Modified SMC Simulation and Results

The gains chosen for the modified sliding mode controllers are the same as the unmodified one in table (4-3).

The results for the modified sliding mode controller are as follows:

For step inputs:

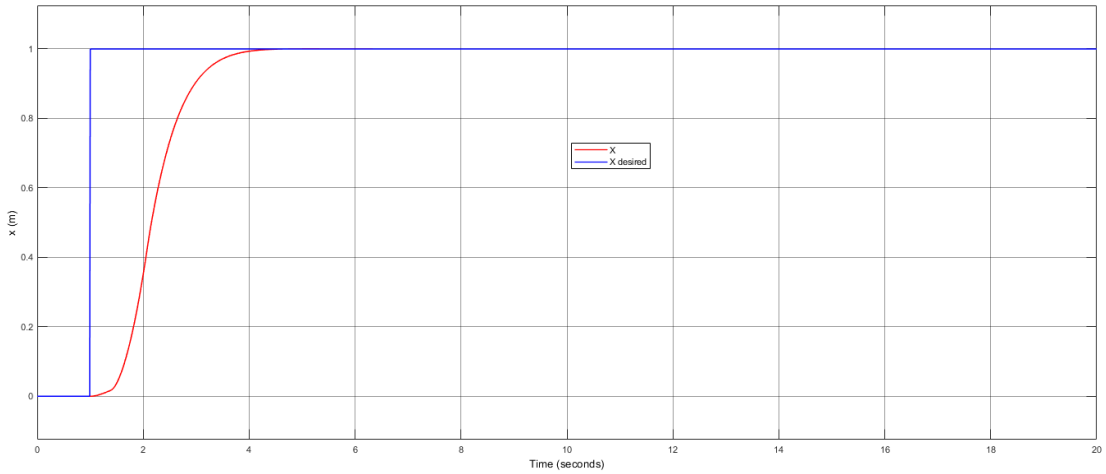

Figure 4-34 X step input response (Modified SMC)

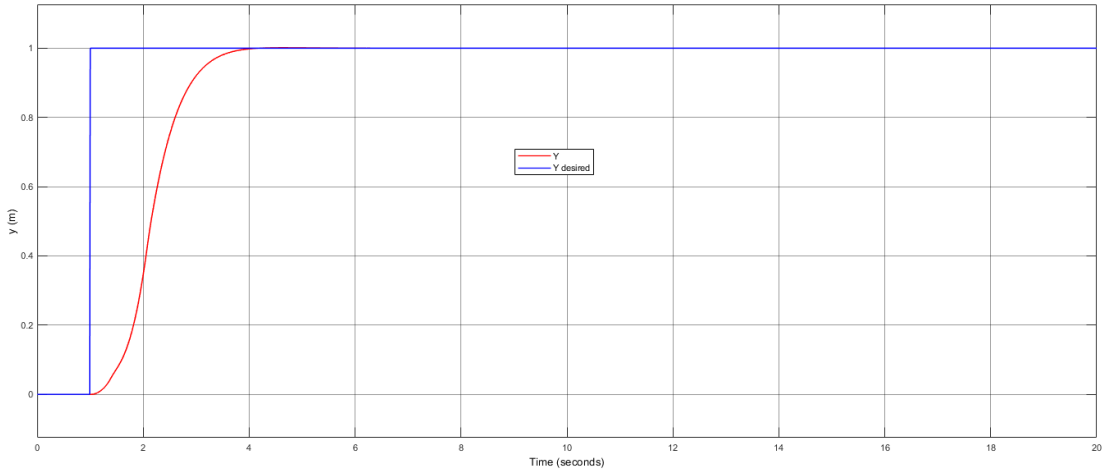

Figure 4-35 Y step input response (Modified SMC)

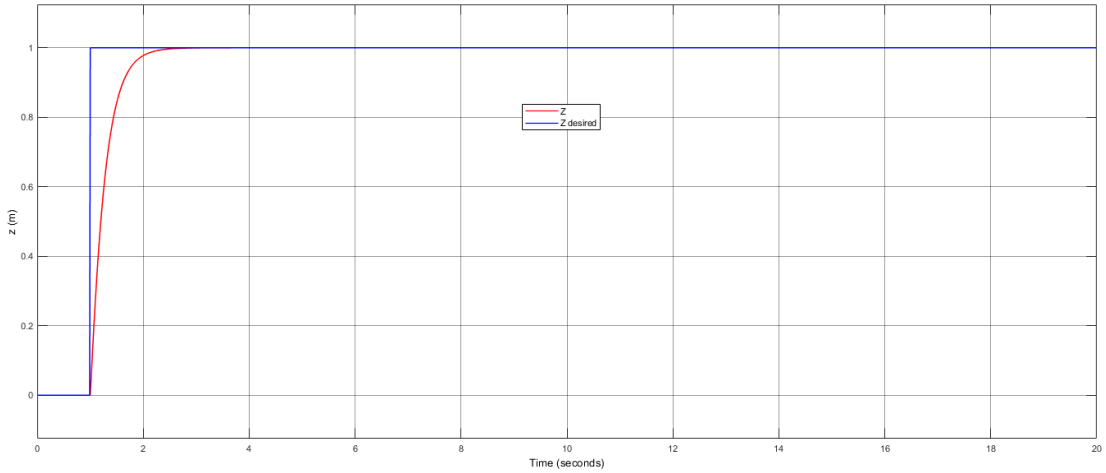

Figure 4-36 Z step input response (Modified SMC)

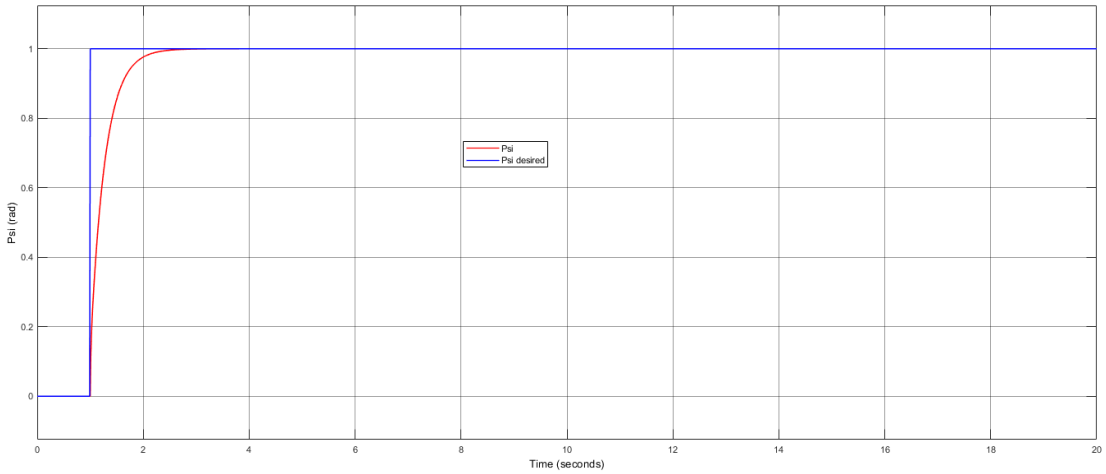

Figure 4-37 Psi step input response (Modified SMC)

	x(t)	y(t)	z(t)	Psi(t)
Rise Time (s)	1.3219	1.312	0.5825	0.5938
Overshoot	0.0415	0.1324	0	0
Settling time (s)	3.6318	3.4885	2.0287	2.049

Table 4-7Characteristic performances to a step input for Modified Sliding Mode Control

For trajectory:

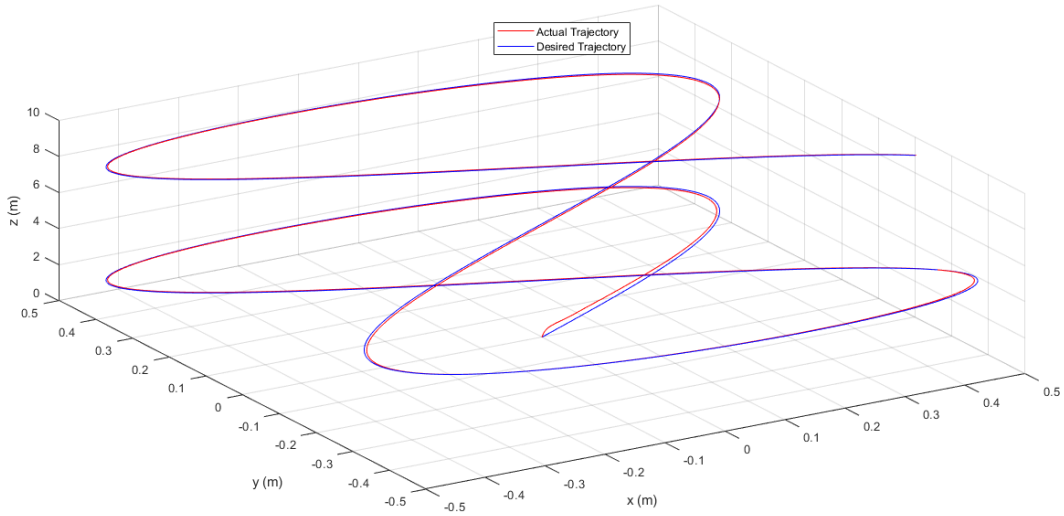

Figure 4-38 X-Y-Z Trajectory Response (Modified SMC)

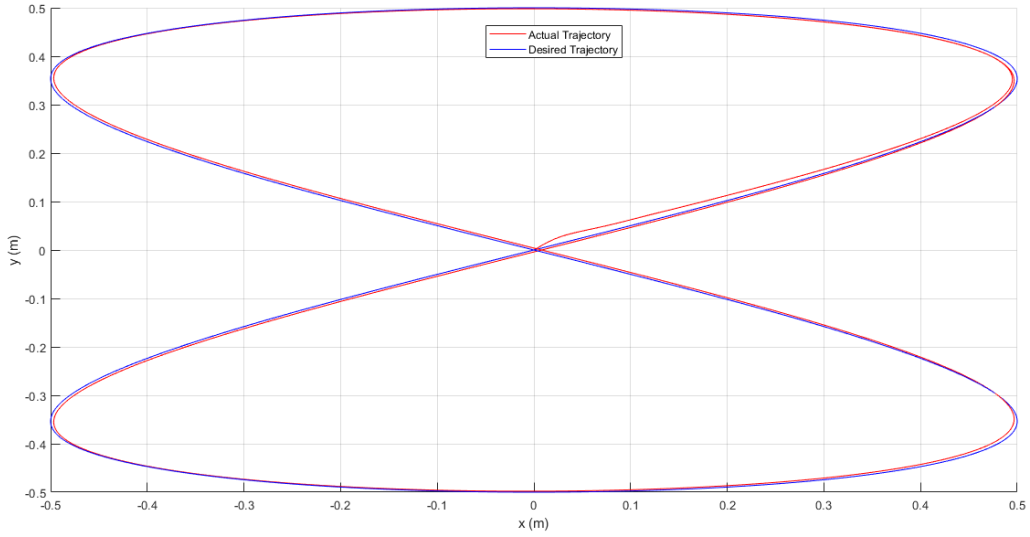

Figure 4-39 X-Y Trajectory Response (Modified SMC)

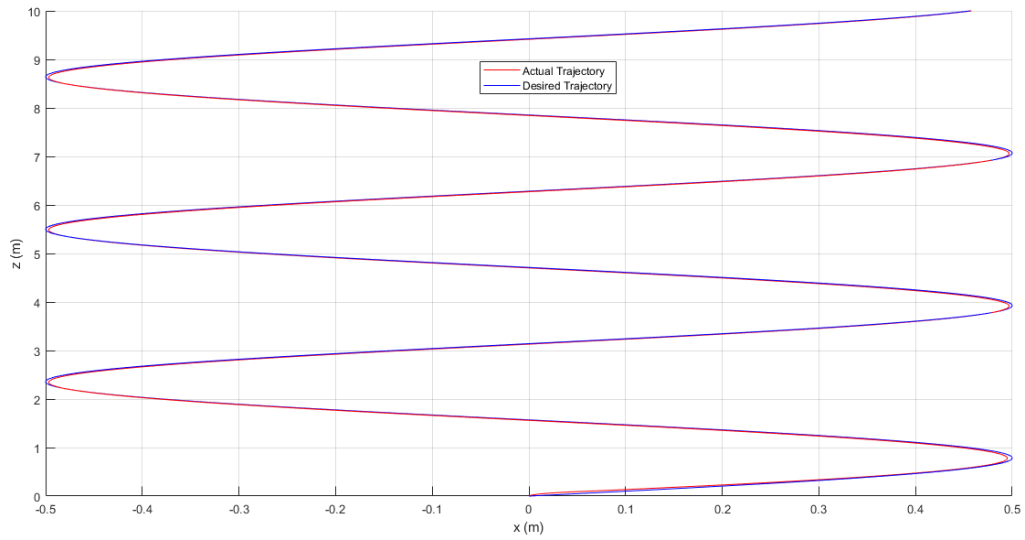

Figure 4-40 X-Z Trajectory Response (Modified SMC)

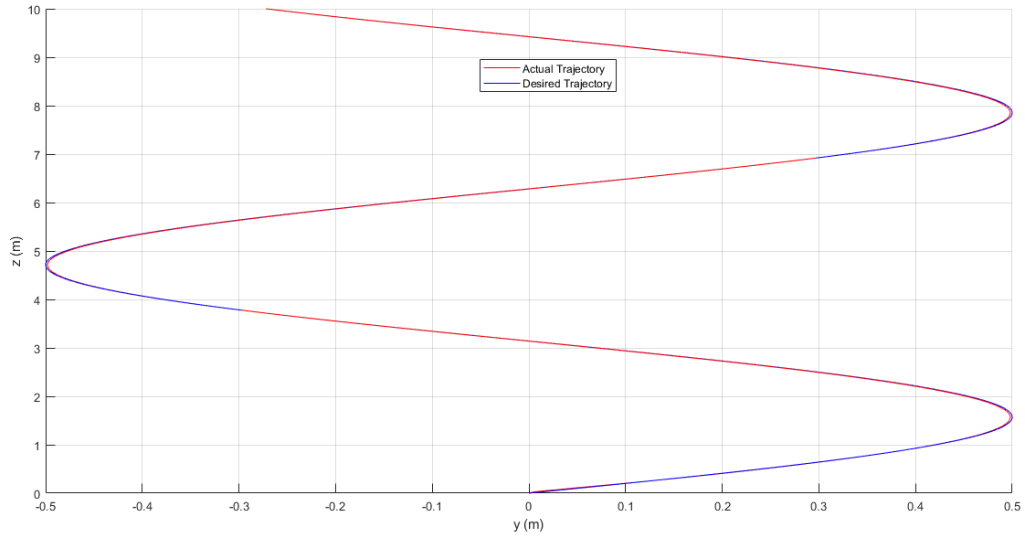

Figure 4-41 Y-Z Trajectory Response (Modified SMC)

4.6. Discussion

Throughout the thesis a total of 4 controllers were developed; PID, FOPID, SMC and a modified form of SMC.

As per simulations, the following conclusions can be drawn upon the controllers:

For step input tracking:

- The PID control proved to be fast in its response, albeit having some over shoot and steady state error.
- The FOPID control improved upon the response of the PID by retaining the response speed and adding more accuracy in both the transient and the steady state.
- The SMC was an alternative that doesn't suffer too much from the overshoot problem of the, but has a slower response time overall. It also encountered undesirable chattering phenomenon that could be potentially harmful of real dynamic systems.
- The chattering problem was reduced by modifying the SMC to have a continuous function rather than a discontinuous one. This proved to be successful and made the SMC much more desirable as control choice.

For trajectory tracking:

- The PID Control shows good performance despite having overshoot, and some tracking issues when direction change occurs.
- The FOPID Control has improved tracking in comparison to the PID; being able to perform direction changes in a more accurate manner without suffering from too much overshoot.
- The SMC provides better trajectory tracking compared to the previous two controllers. It however presents the problem of the chattering phenomenon.
- The modified SMC both reduces the undesirable chattering effect of the SMC and improved upon its performance by reducing the occurring overshoot.

4.7. Conclusion

This chapter covers the results of the different tests performed in MATLAB/SIMULINK on the different controllers designed and implemented as well as a discussion and analysis into these results.

Conclusion and Future Development

This project is an attempt to model and control the motion of the quadrotor, and to achieve that goal, a set of rules, methods and strategies were established and implemented.

The dynamic model of the quadrotor was derived using the Newton-Euler method and three control strategies were implemented. The controllers chosen, the PID, FOPID and SMC along with the quadrotor model were built and simulated in MATLAB/SIMULINK and had their performance analyzed. All controllers gave varying performances, establishing that the choice of controller depends on the performance parameters required.

These controllers have limitations that can be overcome; the PID and FOPID can have their performance improved upon by implementation of heuristic methods to tune their gains. The SMC has different algorithms to its application, and the use of other algorithms such as the "super twisting" algorithm may improve its performance.

The presented mathematical model does not consider external disturbances as they overlooked. In future work, the disturbances should be included along with the possible development of a controller capable of dealing with the failure of one or more rotors.

References

[1] R. Austin, "Unmanned Aircraft Systems: UAVS Design, Development and Deployment," 2010.

[2] 2013. [Online]. Available: https://www.forbes.com/sites/quora/2013/12/23/what-makes-the-quadcopter-design-so-great-for-small-drones/?sh=2c338c81654f.

[3] J. G. a. W. R. V. Ghadiok, "Autonomous indoor aerial gripping using a quadrotor," 2011.

[4] N. M. a. V. K. D. Mellinger, "Trajectory generation and control for precise aggressive maneuvers".

[5] "Joseph-Michel and Jacques-Étienne Montgolfier," [Online]. Available: https://www.britannica.com/biography/Montgolfier-brothers.

[6] "The Queen Bee," 2012. [Online]. Available: http://www.lightaircraftassociation.co.uk/2012/Magazine/June/QueenBee.pdf.

[7] "Small, Unmanned Aircraft Search for Survivors in Katrina Wreckage," 14 September 2005. [Online]. Available: https://www.nsf.gov/news/news_summ.jsp?cntn_id=104453.

[8] "Amazon drones: Obstacles to the Bezos dream," 2013. [Online]. Available: https://web.archive.org/web/20131207012354/http://www.politico.com/story/2013/12/obstacles-to-the-jeff-bezos-drone-dream-100536.html.

[9] G. G. Slabaugh, "Computing Euler angles from a rotation matrix," 1999.

[10] J. Batallé, "An Introduction to Positional Tracking and Degrees of Freedom (DOF)," 2013.

[11] A. M. a. M. Z. Sadeghzadeh, "Payload drop application of unmanned quadrotor helicopter using gain-scheduled pid and model predictive control techniques," 2012.

[12] S. Francesco, "Quadrotor control: modeling, nonlinear control design, and simulation," 2015.

[13] C.-T. Chen, "Linear System Theory and Design," 1999.

[14] D. W. Mellinger, "Trajectory Generation and Control for Quadrotors," 2012.

[15] G. C. Kiam Heong Ang, "PID Control System Analysis, Design, and Technology," 2005.

[16] J. L. E. P. S.A. David, "Fractional order calculus: historical apologia, basic concepts and some applications," 2012.

[17] K. B. O. a. J. Spanier, "Fractional calculus: Theory and applications of differentiation and integration to arbitrary order,academic press," 1974.

[18] S. Das, Functional fractional calculus for system identification and controls, 2008.

[19] A. Tepljakov, "FOMCON Fractional-order Modeling and Control," [Online]. Available: https://fomcon.net/

[20] "Overview of FOMCON toolbox for MATLAB," [Online]. Available: https://fomcon.net/fomcon-toolbox/overview/

[21] Y. C. Z. C. H. W. L. C. Tong Liu, "Adaptive fuzzy fractional order PID control for 6-DOF quadrotor," 2020.

[22] V. U. U. O. K.D. Young, "A control engineer's guide to sliding mode control," *IEEE Transactions on Control Systems Technology,* vol. 7, 1999.

[23] K. S. F. L. D. Abhijit, "Dynamic inversion with zero-dynamics stabilisation for quadrotor control," 2009.