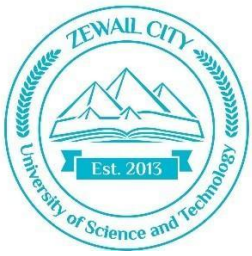


Zewail City of Science, Technology, and Innovation
University of Science and Technology
Aerospace Engineering

"Design and Control of a VTOL Aircraft: Fixed-Wing and Quadrotor Integration for Autonomous Flight"

A Graduation Project
Submitted in Fulfillment of
B.Sc. Degree Requirements in
Aerospace Engineering

Prepared By

Muhammad Ayman	201801356
Ahmed Medhat	201801207
Dena Qatry	201800642
Areeg Ayman	201801902
Mariam Wagdy	201801585

Supervised By
Dr. Mustafa Abdallah

2022/2023

Acknowledgment

We would like to express our sincere gratitude to everyone who has contributed to the completion of this thesis. It has been an incredible journey, and we are grateful for the opportunity to work on this project together.

Firstly, we would like to acknowledge Eng. Ahmed El Gohary for his guidance, support, and valuable feedback throughout the entire research process. His expertise and insights have been instrumental in shaping our research and helping us achieve our goals.

We would also like to thank Dr. Mostafa Abdallah and Dr. Ahmed El Taweel for their support and encouragement throughout the journey of 5 years at the University of Science and Technology in Zewail City. Their guidance and knowledge have been invaluable in shaping our understanding of the subject matter and helping us navigate the challenges we faced along the way.

We would also like to express our gratitude to our families and friends for their unwavering support and encouragement throughout this journey. Their love and encouragement have been a source of motivation for us, and we are grateful for their constant presence in our lives.

Finally, we would like to thank all the researchers and authors whose works have been cited in this thesis. Their research and insights have been instrumental in shaping our understanding of the subject matter and have guided our research in many ways.

Declaration

We declare that this thesis has been composed by the students whose names and ids are shown on the title page.

We also declare that this thesis represents our own work as the team responsible for the software part of VTOL aircraft design, modeling, and control.

Finally, we declare that this work has not been submitted for any other degree or professional qualification except as specified.

Abstract

This project focuses on the modeling and control of an unmanned aerial vehicle (UAV) of the VTOL (Vertical Takeoff and Landing) type for the purpose of payload delivery. Our objective is to develop an efficient control system that enables the UAV to accurately maintain an altitude of 120 m and deliver a package within a specified distance from the origin. The VTOL combines the advantages of both fixed-wing and quadrotor of having long-range and high maneuverability. This work involves the design, implementation, and testing of different controllers to identify the most effective controller for both the fixed-wing and quadrotor components individually. For the quadrotor, we test and compare the modified PID controller with Genetic Algorithm and Nonlinear Model Predictive Controller (NLMPC). As for the fixed-wing aircraft, Linear Quadratic Controller (LQR), Total Energy Control System (TECS), Fuzzy Logic Controller(FLC), and MPC are picked for analysis and comparison. The controllers are analyzed in terms of transient response, disturbance rejection, and noise resilience. For the fixed wing, MPC has proven to be the better controller for the fixed UAV, and the NMPC is chosen for the Quadrotor, They should be combined in the VTOL design to achieve a high level of performance and accuracy in payload delivery.

Table of Contents

List of Abbreviations

VTOL	Vertical Take Off and Landing
MPC	Model Predictive Control
SISO	Single Input Single Output
MIMO	Multi Input Multi Output
FLC	Fuzzy Logic Controller
LQR	Linear Quadratic Control
PID	Proportional Integral Derivative controller
NMPC	Non Linear Model Predictive Control

Chapter 1: Introduction

Unmanned Aerial Vehicles (UAVs), also known as drones, have transformed the way we approach a variety of jobs, from aerial photography to scientific study and even package delivery. These autonomous or remotely piloted aircraft have pushed the boundaries of aviation, with enormous promise for both commercial and military uses. Among the various forms of UAVs, Vertical Takeoff and Landing (VTOL) aircraft stand out as a particularly intriguing category. Unlike standard fixed-wing aircraft that require runways for takeoff and landing, VTOLs have the unique ability to climb and descend vertically, giving them exceptional maneuverability and access to hard-to-reach locations. This feature, along with developments in control systems and technical innovation, has driven VTOL aircraft to the forefront of R&D efforts. In this work, we explore the modeling and control of VTOL UAVs, delving into the complexities of their design and the difficulties encountered in realizing their full potential. Through a thorough examination of these cutting-edge aircraft, we want to uncover the boundless potential they represent in different industries ranging from transportation and surveillance to disaster response and beyond.

1.1. Motivation

Modeling and control of VTOL (Vertical Takeoff and Landing) aircraft is an intriguing study field, notably focused on both fixed-wing and quadrotor designs. Advancements made in control systems have expanded the capabilities of unmanned aerial vehicles (UAVs) by providing more mobility, adaptability, and operating capacities. UAVs may achieve accurate control of attitude, position, and trajectory by efficiently combining modern control algorithms and sensor technologies, enabling for efficient payload delivery, surveillance missions, and even autonomous operations.

Combining both quadrotors and fixed-wing aircrafts can result in quite advantageous characteristics. Namely, fixed-wing aircrafts are well-known for their efficiency and long-range capabilities. On the other hand, quadrotors excel in hovering, maneuvering in tight areas, and

performing vertical takeoffs and landings. The combined design can result in higher longevity, increased cargo capacity, and the ability to function in a variety of conditions.

The complicated dynamics, aerodynamics, and interactions between many actuators and control surfaces necessitate the development of control algorithms and techniques in order to maximize the performance of VTOL aircraft. Furthermore, research into modeling and control can improve safety, dependability, and operational efficiency, boosting the integration of UAVs into a variety of sectors and applications.

1.2. Literature review

In a recent paper, Chipade et al. consider the development of a biplane UAV that combines the desirable characteristics of both quadrotor and fixed-wing aircrafts. The authors aim to design such UAV to be used in payload delivery missions. The UAV is designed to carry 6 kg payload and transport it to a destination at 16 km from origin. As for the control part, the authors develop a PID controller along with implementation on open source PixHawk autopilot. The results demonstrate that the proposed VTOL was able to achieve robust performance in payload delivery missions. The VTOL demonstrates efficient maneuverability due to its variable pitch prop rotors (Chipade et al., 2018). In addition, It had a long range owing to the addition of the biplane wings configuration in its structure. In this work, we examine and analyze the performance of several controllers on both quadrotor and fixed-wing aircrafts. We expanded the analysis to include controllers like modified PID, Model Predictive Controller (MPC) for the quadrotor. In addition, we tested the Linear–quadratic regulator (LQR), Total Energy Control System (TECS), MPC, and Fuzzy Logic Controller for the fixed-wing part. Our analysis is focused on examining different controllers for the VTOL in order to achieve optimal performance, robustness, and stability.

1.2.1 Fixed Wing

The Total Energy Control System (TECS) is a control law structure created in the 1980s by Boeing and NASA, which relies on principles of energy management to track altitude and speed during longitudinal flight. It controls the total energy, which is the combination of kinetic and potential energy, and their distribution between the two forms. Compared to conventional control systems, TECS has improved model independence and considers the mechanical coupling between altitude and speed dynamics. This allows it to overcome some limitations commonly found in proportional-integral controllers. TECS has been used successfully in

various optimization problems in several fields such as engineering, finance, and computer science. Additionally, many variations of the basic TECS algorithm exist to improve its performance in specific types of problems. Overall, TECS is a well-known control law structure that utilizes energy management principles to track both altitude and speed in longitudinal flight, offering notable advantages over conventional control systems.

Model Predictive Control (MPC) and Fuzzy Logic Control (FLC) are two widely used control techniques in various industrial applications. MPC is a control strategy that involves predicting the system's future behavior based on a mathematical model of the system and optimizing the control inputs over a finite horizon. MPC is a powerful tool for controlling complex systems with multiple inputs and outputs, and it has been applied in various fields such as chemical processes, power systems, and aerospace applications. On the other hand, FLC is a control approach that utilizes fuzzy logic to map inputs to outputs. The FLC approach is based on the concept of linguistic variables, which allows the controller to handle imprecise and uncertain information. FLC has been applied in various fields such as automotive control, industrial processes, and robotics, due to its ability to handle nonlinear and uncertain systems.

Despite the differences in their approaches, both MPC and FLC have demonstrated their effectiveness in controlling complex systems. MPC has been shown to provide superior performance in systems with long-time delays or complex dynamics, while FLC is more suitable for systems where the mathematical model is difficult to obtain or the system is subject to significant uncertainties or nonlinearities. Researchers have also explored the combination of MPC and FLC, where FLC is used to tune the MPC controller, resulting in improved performance and robustness (Mughees et al., 2022).

1.2.2 Quadrotor

PID controllers are widely used due to their simplicity and efficiency. Contemporary research in PID controllers is concerned with improving the performance by applying modifications to the classical PID controller. In one paper, Khuwaja et al. examined a modified PID that includes the classical PID in addition to gain optimization using Genetic Algorithm (GA). The authors used GA to compute the optimum PID gains that could minimize the cost function. The cost function used was a weighted sum of ITAE (Integral Time Absolute Error), ISE (Integral square error), and IAE (Integral absolute error). The results showed improvement in the stability of the quadrotor in transient response and disturbance rejection. In addition, that was reflected on the

performance indices as well; rise time, settling time, and overshoot (Khuwaja et al., 2018). In this work, the PID controller is expanded to examine if the same methodology could be optimized further to achieve acceptable performance in trajectory tracking as well.

In another paper, the authors examined using different optimization algorithms to tune the PID controller. Methods like Genetic Algorithms (GAs), the Crow Search Algorithm (CSA), and Particle Swarm Optimization (PSO) were used to tune the controller and their results were compared to evaluate which algorithm optimized the response best. The results show small variations across the used methods on the performance index (Sheta et al., 2021). In that sense, other factors, such as cost function selection, optimization settings, search space, generations, and population number, contribute more to the convergence of the algorithm towards the optimal or near-optimal solution.

1.3. Mission Requirements and Parameters

The UAVs (unmanned aerial vehicles) market is spreading as they have shown in many applications to reduce cost, decrease safety accidents and enhance project management. To illustrate, UAVs now are being used for aerial photography, forest monitoring, construction monitoring, and in delivery systems. Hybrid VTOL (Vertical landing and take-off) drones, which implement the benefits of fixed-wing with quadcopter design, are more cost-effective for undergoing long-range, heavy payload, 24 kg in our case, missions with over 8 hours of flight time. Our project focuses on planning a delivery system using drones. First, we are designing an autopilot to control the UAV and have autonomous modes without having a human continuously guide the UAV. The controller will hold position, and attitude, in addition to following guided paths to return to a specific predefined point in space. In order to accomplish this, both controllers for fixed-wing surfaces and quadcopter motors are required to be designed. Parallel to this, we are working on the drone design to be able to withstand stresses, carry heavy weight and produce the least drag. Finally, testing the drone will then be done to ensure all the subsystems are working properly.

Chapter 2: Aerodynamic Stability Derivatives Calculations using Tornado (Vortex Lattice Method)

The aerodynamic stability derivatives are obtained using Tornado (vortex lattice method).

2.1 Introduction

Tornado is a MATLAB-based 3D-vortex lattice program with a configurable wake. It is utilized for many different jobs requiring aerodynamic coefficients in the wind and body axes, 3D forces acting on each panel, and stability derivatives with respect to angle of attack, angle of sideslip, control surfaces deflections, including rudder deflection, and angular rates. According to GNU, Tornado is an open-source application that requires MATLAB versions 5.3 and above. Its foundation is potential flow theory's conventional vortex lattice theory.

The form of the wake emanating from each lifting surface's trailing edge varies depending on the flying conditions. Tornado is intended for usage during the conceptual design phase. It is compatible with a wide range of designs, including multi-wing designs with tapered, twisted, swept, cambered, and cranked wings that may or may not include dihedral angles. It also enables designs for flying wings. Furthermore, any quantity of wings or control surfaces may be used. Additionally, canards, flaps, ailerons, elevators, and rudders may be used. Incorporate fences, winglets, and engine mounts into the design as well. The utilization of tiny angles of attack is one of the main presumptions of the vortex lattice approach, which should be considered.

Therefore, it is advised to be cautious while employing comparatively greater angles of attack and rotating speeds. Additionally, when computing, the impacts of the fuselage, frictional drag, compressibility, and thickness of the lifting surfaces are not considered.

Tornado and its user manual can be downloaded from its website, included in the references. This section will provide a thorough explanation of the analysis of the UAV, from its geometry to using the program's output data. In Appendix A, there are more specific instructions on how to use Tornado.

2.2 Required Inputs

The required data used by the program is as follows:

1. The location of the aircraft's center of gravity.
2. The detailed geometry of the aircraft, i.e., airfoils of each wing, wing chord and span, dimensions and positions of the horizontal and vertical tails, and positions of the control surfaces, etc.
3. The state in which the aircraft is flying.

2.3 Coordinate System

The coordinate system of Tornado differs from the traditional coordinate system used in Flight Dynamics References as the positive Z direction is taken upward as shown in Figure 2.3.1. The center of the first wing's leading edge is the origin of the coordinate system. The forces operating on the airplane are depicted in the preceding image as X, Y, and Z, respectively, with their positive values pointing in the same direction along each axis. The letters P, Q, and R stand for the aircraft's respective angular rates about the x, y, and z axes. Whereas l, m, and n depict the moments affecting the airplane about the indicated axes through the reference point.

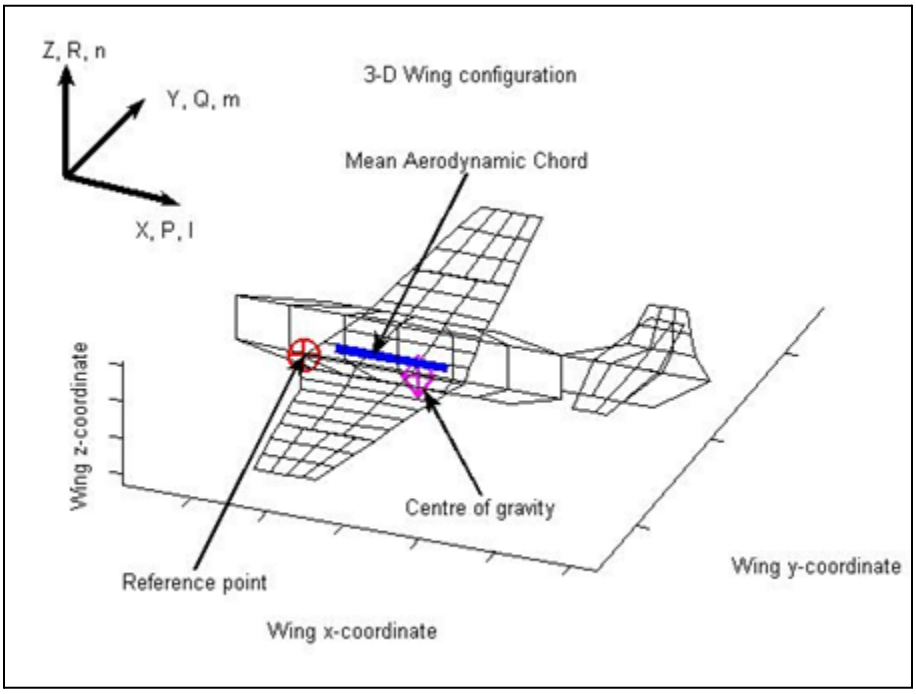


Figure 2.3.1. Definition of Tornado Coordinate System

2.4 Tornado Derivatives Definitions

The aerodynamic derivatives the program calculates can be obtained from the help center of Tornado. Figure 2.4.1 displays these derivatives.

$CL_\alpha = \partial CL / \partial \alpha$
$CL_\beta = \partial CL / \partial \beta$
$CL_\delta = \partial CL / \partial \delta$
$CL_P = \partial CL\ 2V / (\partial Pb)$
$CL_Q = \partial CL\ 2V / (\partial Q C_{mac})$
$CL_R = \partial CL\ 2V / (\partial R b)$

$Cl_\alpha = \partial Cl / \partial \alpha$
$Cl_\beta = \partial Cl / \partial \beta$
$Cl_\delta = \partial Cl / \partial \delta$
$Cl_P = \partial Cl\ 2V / (\partial Pb)$
$Cl_Q = \partial Cl\ 2V / (\partial QC_{mac})$
$Cl_R = \partial Cl\ 2V / (\partial Rb)$

$CD_\alpha = \partial CD / \partial \alpha$
$CD_\beta = \partial CD / \partial \beta$
$CD_\delta = \partial CD / \partial \delta$
$CD_P = \partial CD\ 2V / (\partial P b)$
$CD_Q = \partial CD\ 2V / (\partial Q C_{mac})$
$CD_R = \partial CD\ 2V / (\partial R b)$

$Cm_\alpha = \partial Cm / \partial \alpha$
$Cm_\beta = \partial Cm / \partial \beta$
$Cm_\delta = \partial Cm / \partial \delta$
$Cm_P = \partial Cm\ 2V / (\partial P b)$
$Cm_Q = \partial Cm\ 2V / (\partial Q C_{mac})$
$Cm_R = \partial Cm\ 2V / (\partial Rb)$

$CY_\alpha = \partial CY / \partial \alpha$
$CY_\beta = \partial CY / \partial \beta$
$CY_\delta = \partial CY / \partial \delta$
$CY_P = \partial CY\ 2V / (\partial P b)$
$CY_Q = \partial CY\ 2V / (\partial Q C_{mac})$
$CY_R = \partial CY\ 2V / (\partial R b)$

$Cn_\alpha = \partial Cn / \partial \alpha$
$Cn_\beta = \partial Cn / \partial \beta$
$Cn_\delta = \partial Cn / \partial \delta$
$Cn_P = \partial Cn\ 2V / (\partial Pb)$
$Cn_Q = \partial Cn\ 2V / (\partial Q C_{mac})$
$Cn_R = \partial Cn\ 2V / (\partial R b)$

Figure 2.4.1. Tornado Aerodynamic Derivatives

2.5 Geometry

All the different geometries presented were obtained from Gohary et.al.

2.5.1 Location of the Center of Gravity

Parameter	Value [m]
X_{CG}	0.16
Y_{CG}	0
Z_{CG}	0.02

Table 2.5.1. Location of Center of Gravity of Mian Wing

2.5.2 Main Wing Parameters

Parameter	Value
Wingspan (b_w)	1.7 [m]
Wing Chord (c_w)	0.3 [m]
Quarter Chord Line Sweep Angle (Λ_w)	3º
Dihedral Angle (Γ_w)	2º
Incidence Angle (i_w)	0º
Twist Angle	0º
Taper Ratio (λ)	0.86
Airfoil	NACA 0016

Table 2.5.2. Specifications of the Main Wing

2.5.3 Horizontal Tail Parameters

Parameter	Value
Wingspan (b_{HT})	0.65 [m]
Wing Chord (c_{HT})	0.2 [m]

Quarter Chord Line Sweep Angle (Λ_{HT})	0.22689º
Dihedral Angle (Γ_{HT})	0º
Incidence Angle (i_{HT})	0º
Twist Angle	0º
Taper Ratio (λ)	0.6
Airfoil	Flat Plate

Table 2.5.3. Specifications of the Horizontal Tail

2.5.4 Vertical Tail Parameters

Parameter	**Value**
Wingspan (b_{VT})	0.195 [m]
Wing Chord (c_{VT})	0.235 [m]
Quarter Chord Line Sweep Angle (Λ_{VT})	0.69813º
Dihedral Angle (Γ_{VT})	0º
Incidence Angle (i_{VT})	0º
Twist Angle	0º
Taper Ratio (λ)	0.266
Airfoil	Flat Plate

Table 2.5.4 Specifications of the Vertical Tail

Given these data, a solution can be obtained.

2.6 Tornado geometry plots

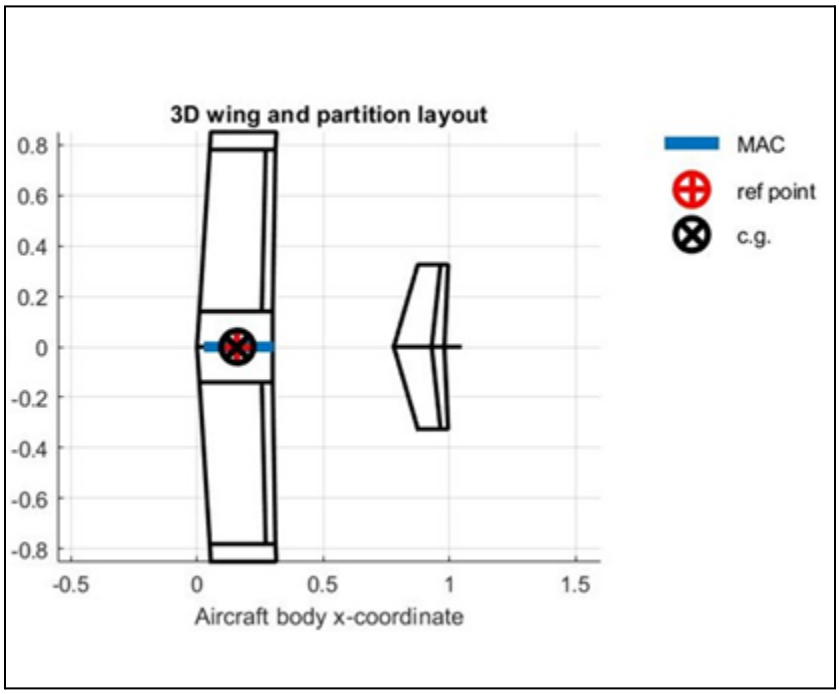


Figure 2.6.1. Tornado Geometry Plots

Figure 2.6.1 shows the aircraft's geometry as viewed by Tornado. The main wing is cut into three partitions with the middle one flapped. While each of the horizontal and vertical tails is one whole flapped partition; the flaps represent the elevator and rudder, respectively. All dimensions are verified to align with the input data.

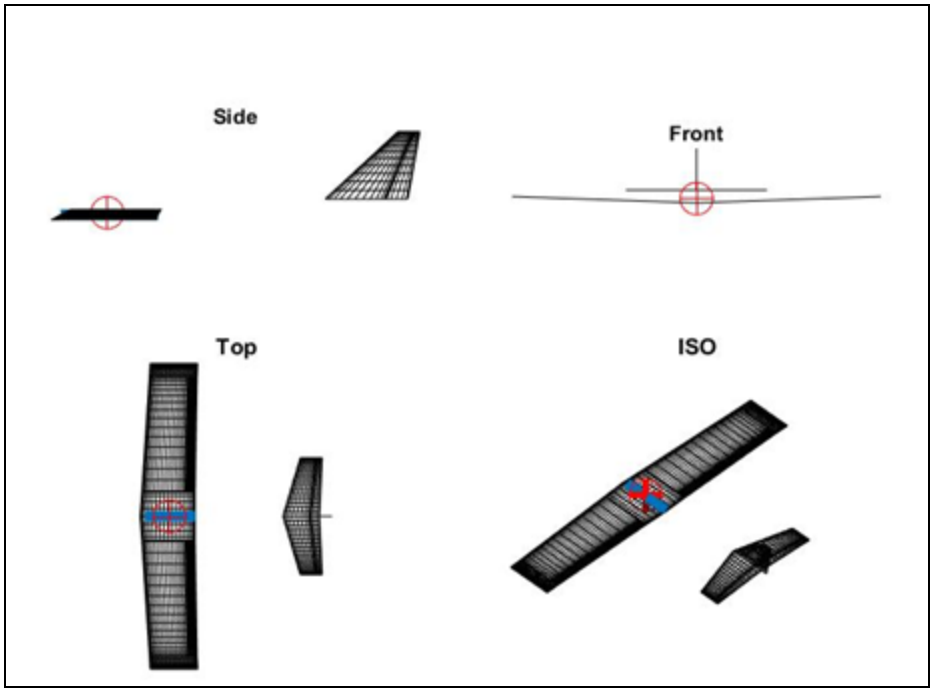


Figure 2.6.2. Panels Distribution in Tornado

Figure 2.6.2 shows the distribution of the panels over the entire aircraft.

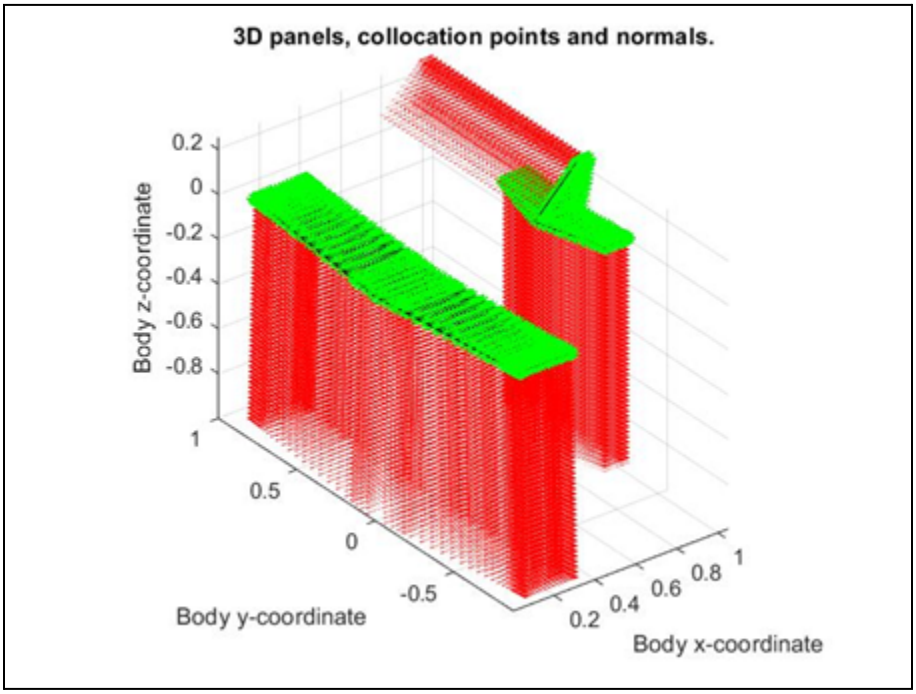


Figure 2.6.3. 3D Panels, Collocation Points, and Normal

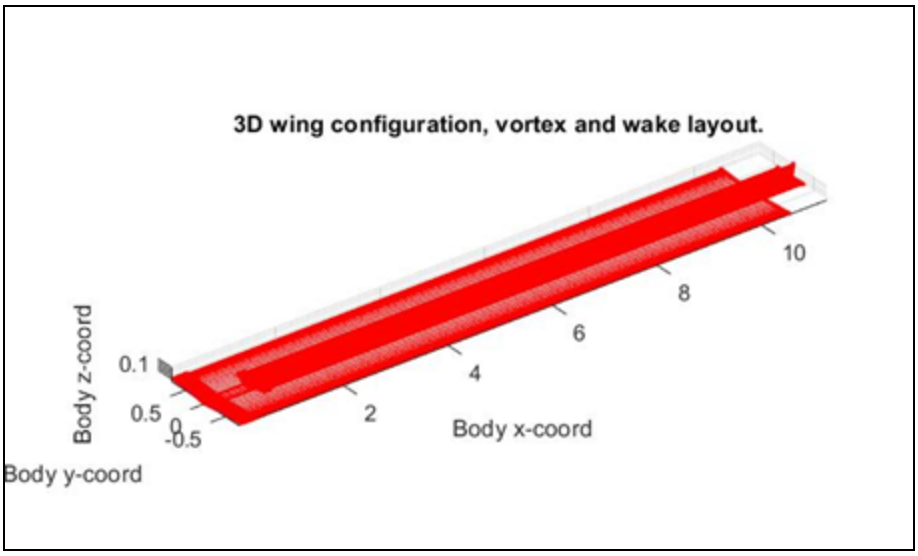


Figure 2.6.4. 3D Wing Configuration, Vortex, and Wake Layout

2.7 Control Surfaces Derivatives

TORNADO CALCULATION RESULTS, Central difference, RUDDER DERIVs

JID:	Trim				
Reference area:	0.47101	α:	5.1	P:	0
Reference chord:	0.27756	β:	0	Q:	0
Reference span:	1.7	Airspeed:	18	R:	0

CL_δ	-9.9114e-06 0.55068 -3.057e-07	CD_δ	2.0296e-05 0.032266 2.3574e-06	CY_δ	0.025136 -7.5198e-13 -0.10456
Cl_δ	0.33201 1.0967e-11 0.0047589	Cm_δ	3.5149e-06 -1.3722 1.4466e-06	Cn_δ	0.039231 3.521e-13 -0.049223

Figure 2.7.1. Control Surfaces' Aerodynamic Derivatives

All the control surfaces used by a tornado are referred to as rudders, however each control surface is dependent on the wing to which it is attached. The control surfaces are the aileron, elevator, and rudder in that sequence, following the input of the main wing, horizontal tail, and vertical tail. To clarify, the CL δ data in Figure 2.7.1 can be presented in table 2.7.1 as:

Table 2.7.1. Lift Coefficients Due to Control Surfaces

Parameter	**Value**
$CL_{\delta\ aileron}$	-9.9114e-6
$CL_{\delta\ elevator}$	0.55068
$CL_{\delta\ rudder}$	-3.057e-7

2.8 Aerodynamic Derivatives

It must be noted that the change in positive Z-axis direction has an effect of the sign of some stability derivatives presented in Figure 2.8.1. This calls for a correction that will be done in the upcoming sections.

TORNADO CALCULATION RESULTS, Derivatives

JID:	Trim				
Reference area:	0.47101	α [deg]:	5.1	P [rad/s]:	0
Reference chord:	0.27756	β [deg]:	0	Q[rad/s]:	0
Reference span:	1.7	Airspeed:	18	R[rad/s]:	0

CL derivatives :		CD derivatives :		CY derivatives :	
CL_α	4.8179	CD_α	0.2153	CY_α	0
CL_β	0	CD_β	0	CY_β	-0.1813
CL_P	0	CD_P	0	CY_P	-0.006
CL_Q	7.2787	CD_Q	0.3864	CY_Q	0
CL_R	0	CD_R	0	CY_R	-0.1794

Roll derivatives :		Pitch derivatives :		Yaw derivatives :	
Cl_α	0	Cm_α	-0.1209	Cn_α	0
Cl_β	0.0309	Cm_β	0	Cn_β	-0.0786
Cl_P	-0.4532	Cm_P	0	Cn_P	-0.0356
Cl_Q	0	Cm_Q	-11.5874	Cn_Q	0
Cl_R	0.0087	Cm_R	0	Cn_R	-0.0801

Figure 2.8.1. Tornado's Aerodynamic Derivatives

2.9 Aerodynamic Derivatives at Trim Conditions

To calculate the trimming angle of attack, the lifting force was set to counter the weight and the total moment was assumed to be "zero". Therefore, by simply equating these conditions,

$W = L, \quad M = 0 \qquad mg = 0.5 * \rho V^2 S C_L$

Where:

$$m = 4\,[Kg], \quad \rho = 1.211\,[Kg/m^3]$$

$$S = 0.471\,[m^2], \quad V = 18\,[m/s]$$

$$C_L = mg/(0.5\rho V^2 S) = 0.424667$$

$$C_L = C_{L_\alpha}\,\alpha + C_{L_{\delta_e}}\,\delta_e$$

$$0.424667 = 4.8375[rad]\,\alpha + 0.56118[rad]\,\delta_e \quad C_m = C_{m_\alpha}\,\alpha + C_{m_{\delta_e}}\,\delta_e$$

$$0 = (-0.1651)[rad]\,\alpha + (-1.3977)[rad]\,\delta_e$$

$$\delta_e = -0.1651/1.3977\ \alpha$$

By substituting in C_L equation,

$$\alpha_{trim} = 0.089006\,[rad] = 5.0996\,[deg] \approx 5.1\,[deg]$$

$$\delta_{-e_{trim}} = -0.01051\,[rad] \approx -0.6319\,[deg]$$

Tornado Computation Results, version 136

JID:	Trim				
Run end time:	13-Jun-2023 02:18:20				
Geometry:	last				
Project:	.			Downwash matrix condition:	1626.5902
Reference area:	0.47101	Reference	0.16		0.16
Reference chord:	0.27756	point pos:	0	Center of gravity:	0
Reference span:	1.7		0.02		0.02

STATE:					
α [deg]:	5.1	Airspeed:	18	P [rad/s]:	0
β [deg]:	0	Altitude:	120	Q [rad/s]:	0
α_{dot} [rad/s]:	0	Density:	1.211	R [rad/s]:	0
β_{dot} [rad/s]:	0	Mach:	0.052968	PG Corr.:	0

Net Wind Forces: (N)		Net Body Forces: (N)		Net Body Moments: (Nm)	
Drag:	0.87054	X:	-2.6158	Roll:	0
Side:	6.5477e-16	Y:	6.5477e-16	Pitch:	0.1171
Lift:	39.1798	Z:	39.1021	Yaw:	0

CL	0.424	CZ	0.4232	Cm	0.0046
CD	0.0094	CX	-0.0283	Cn	0
CY	0	CC	0	Cl	0
$CD_{trefftz}$	N/A				

	0.3879		0.0075		0
CL per wing	0.0361	CD per wing	0.0019	CY per wing	0
	0		0		0

	-0.6319		
Rudder setting [deg]:	0	Allmoving wing setting [deg]:	
	0		

Figure 2.9.1. Aerodynamic Forces at Trim Conditions

The α_{trim} and $\delta_{e\ trim}$ were then entered in Tornado to calculate the aerodynamic derivatives at the specified trim conditions. These derivatives are presented in Figure 2.9.1. Further detailed steps on the tornado procedure are found in Appendix A

The value of C_m is almost "zero" because these calculations are at the trim state. Likewise, the net moments acting on the aircraft in addition to C_n *and* C_l can be considered "zero".

To calculate the C_{L0} and C_{m0}, the angle of attack must be set to "zero" at first: yielding a similar figure but for a "zero" angle of attack and the C_{L0} and C_{m0}.

2.10 Results

The detailed procedures performed to obtain each of the following results are included in Appendix A.

2.10.1 Static margin and the stall angle of attack

Parameter	**Value**
Static Margin	0.038523
Aerodynamic center (x,y,z)	(0.17069, 0, 0.02)
Stall Angle of Attack α_{stall}	10.6008^{o}

Table 2.10.1 Lift Coefficients Due to Control Surfaces

2.10.2 Bending Moment Diagram over the Main Wing

Figure 2.10.1 shows the bending moment acting on the main wing in the SI units.

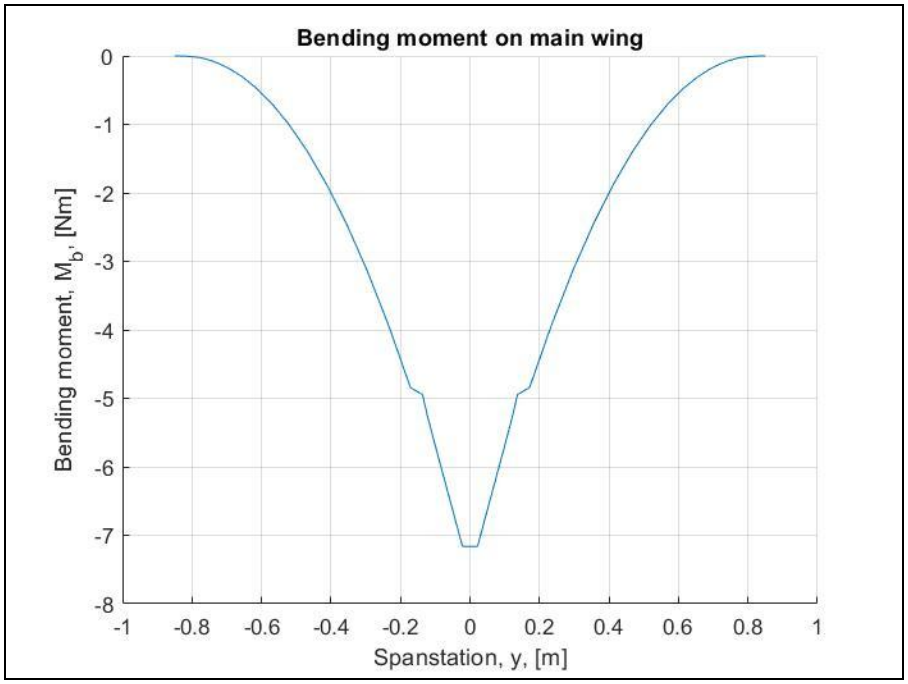


Figure 2.10.1 bending moment on the main wing

2.10.3 Shear Force Diagram over the Main Wing

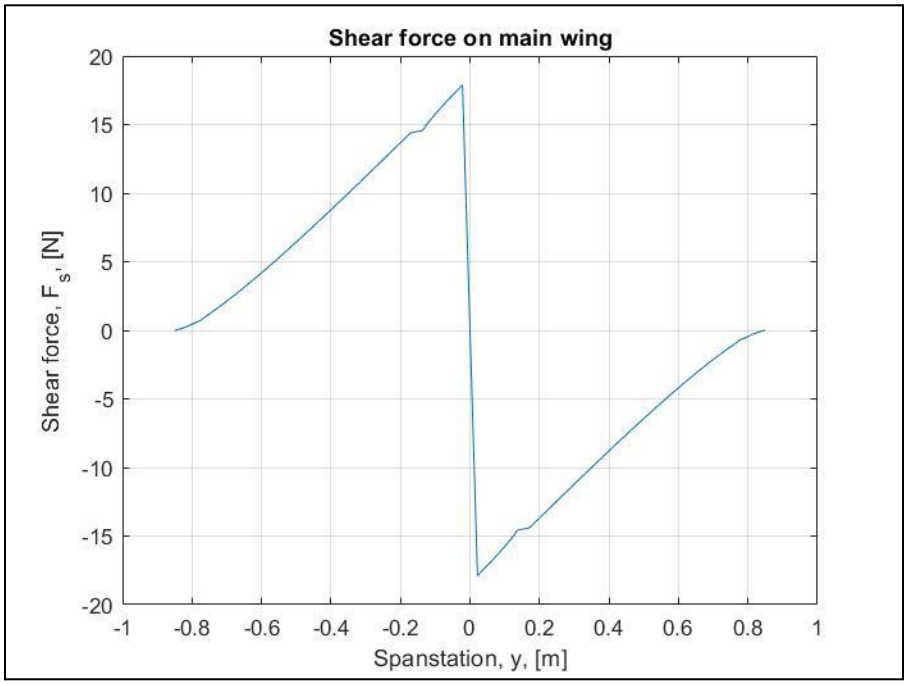


Figure 2.10.2. Shear Force Diagram over the Main Wing

2.10.4 The Local Lift Coefficient on the Main Wing

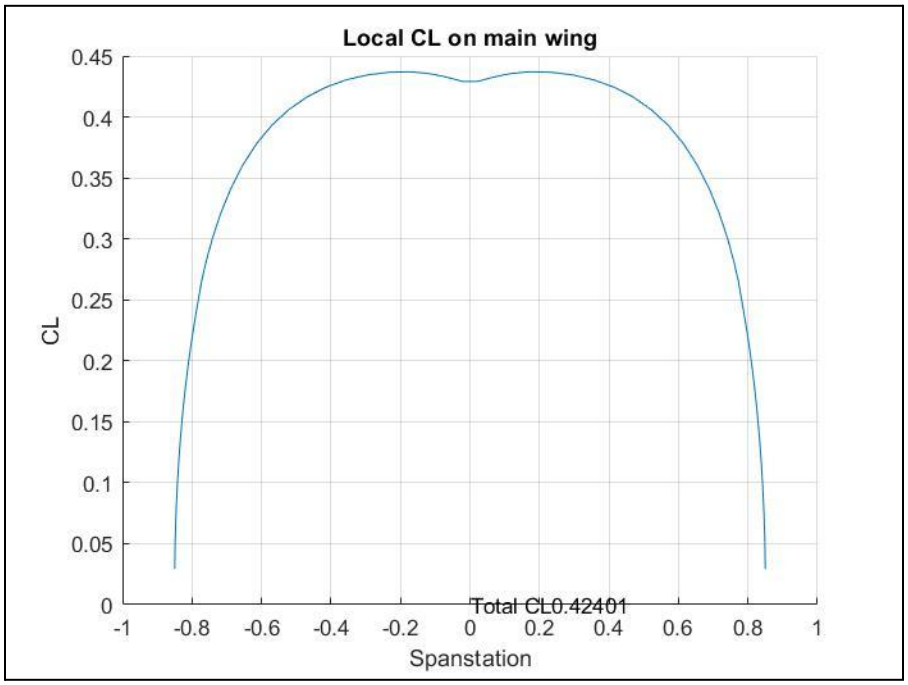


Figure 2.10.3. Local Lift Coefficient on the Main Wing

2.10.5 Normalized Local Lift Coefficient on the Main Wing

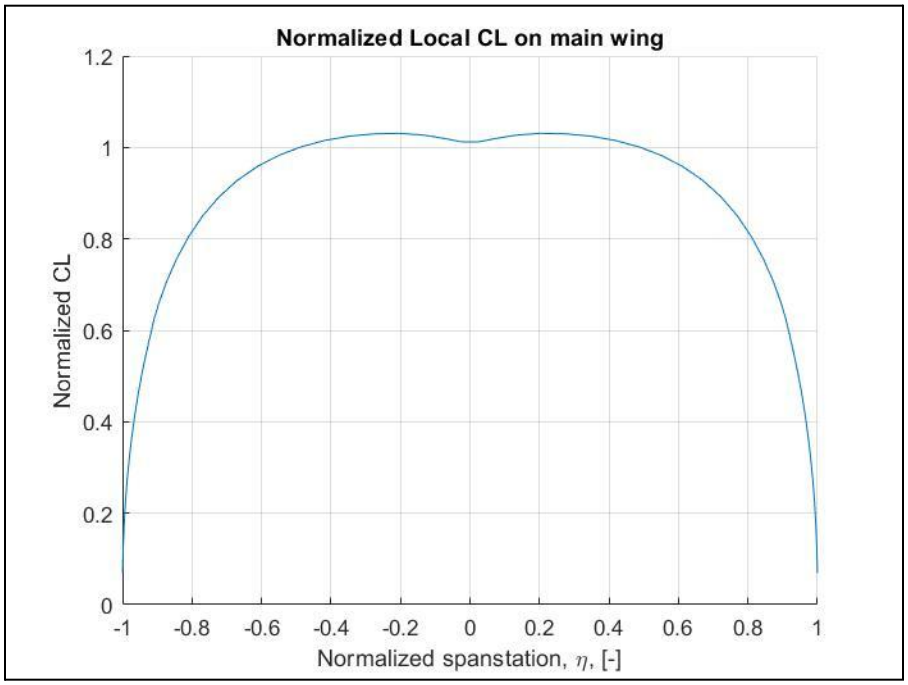


Figure 2.10.4. Normalized Local Lift Coefficient on the Main Wing

2.10.6 The Span Load over the Main Wing

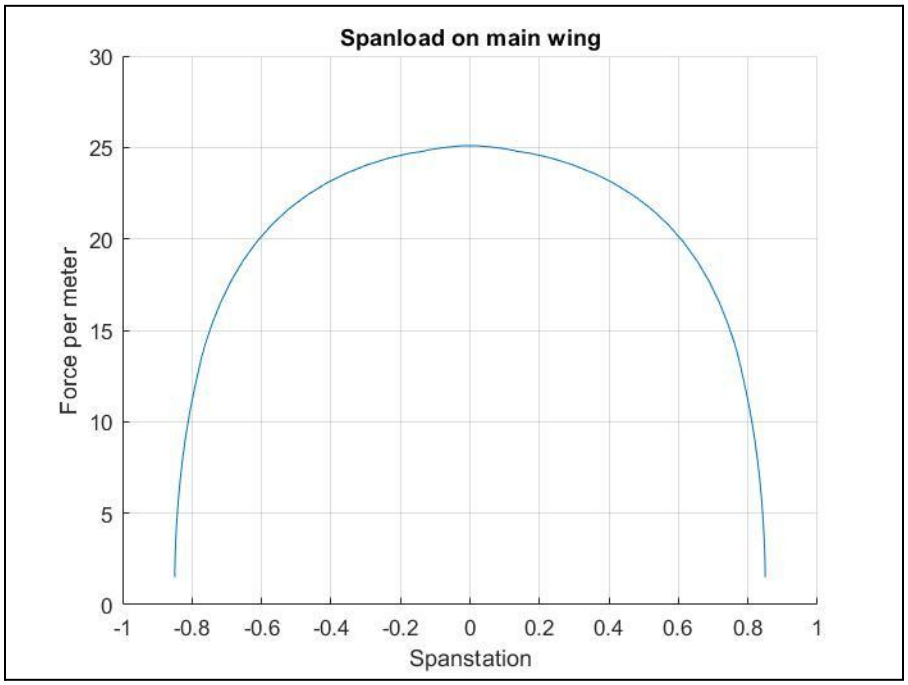


Figure 2.10.5. The Span Load over the Main Wing

2.10.7 The Induced Drag Polar

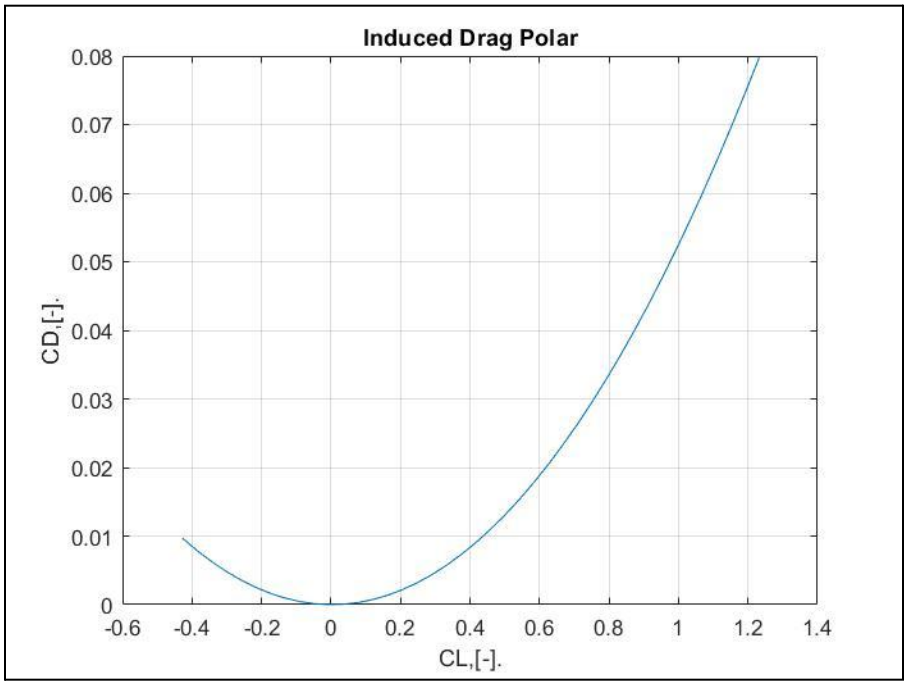


Figure 2.10.6. The Induced Drag Polar

2.10.8 Polar and Lift Curve

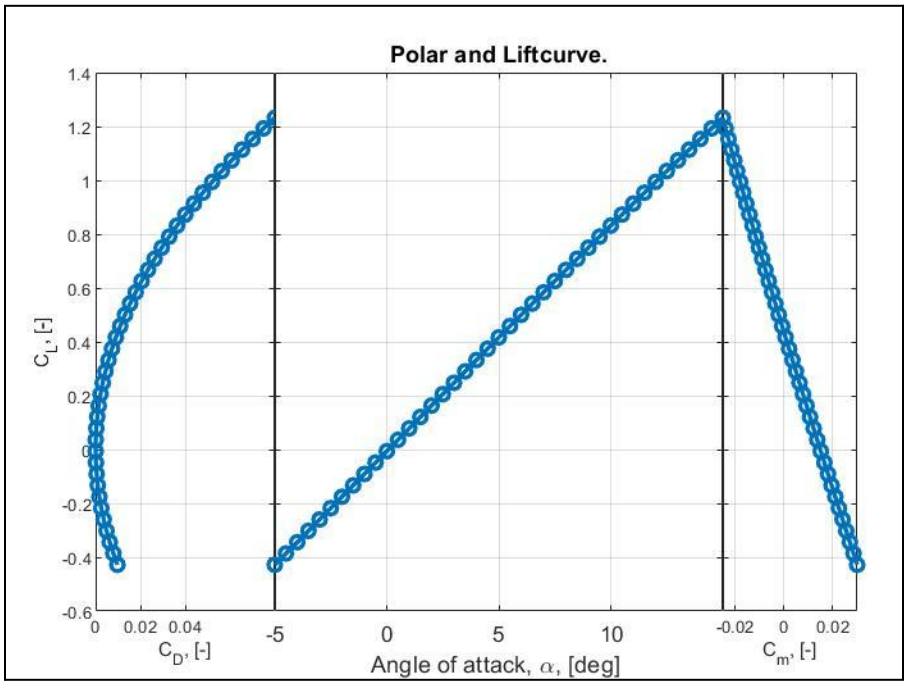


Figure 2.10.7. Polar and Lift Curve

Again, the drag calculated by Tornado, or any vortex lattice program, is only the aerodynamic drag. Hence, the fuselage contribution and the viscosity effects are not considered.

2.10.9 The Distribution of the "Difference" in Pressure Coefficient

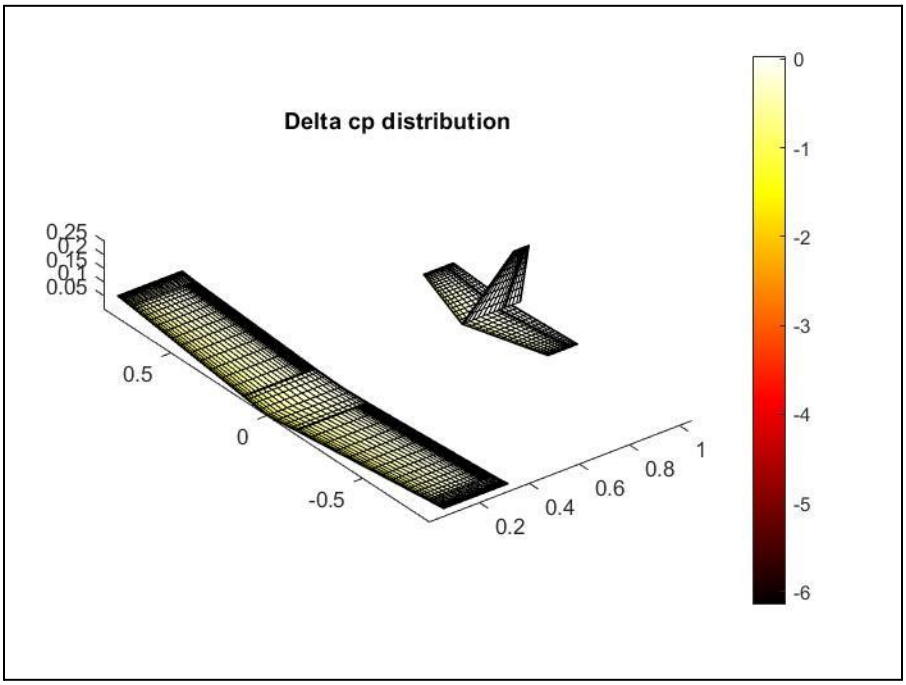


Figure 2.10.8. The Distribution of the "Difference" in Pressure Coefficient

2.10.10 The Aerodynamic Coefficients Dependency on the Angle of Attack (α)

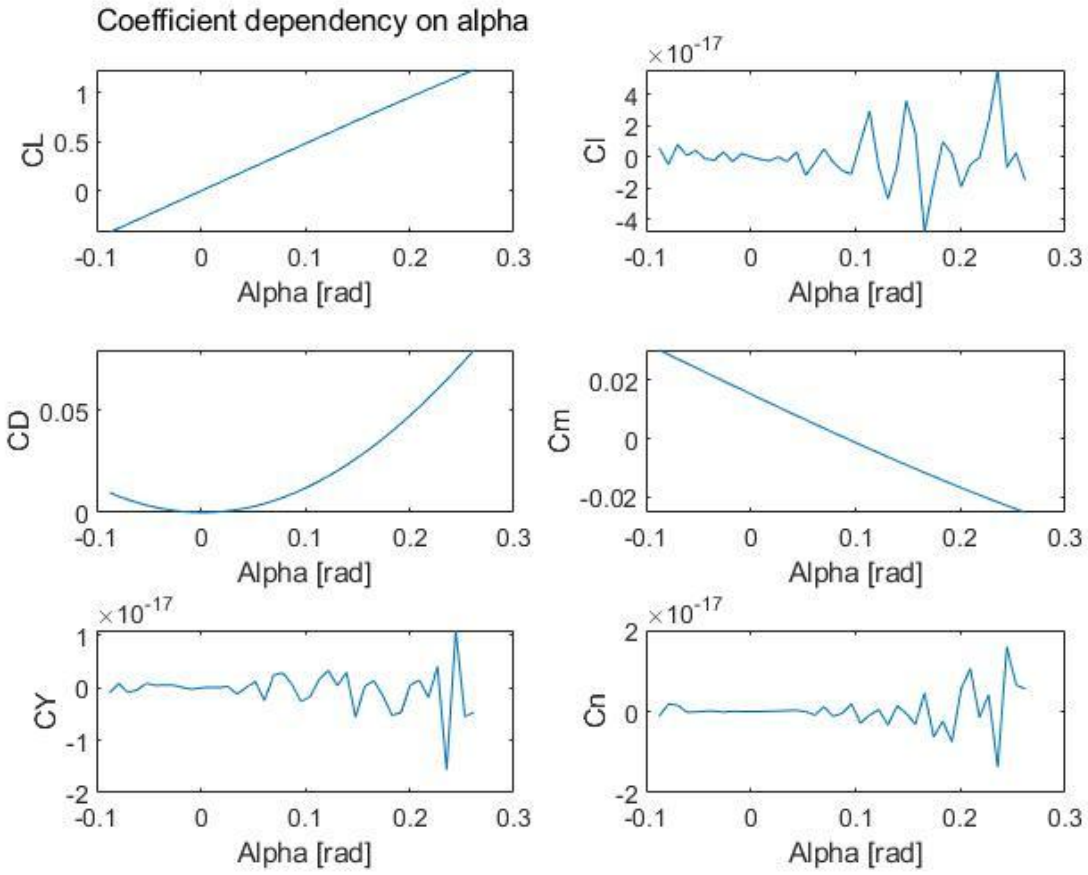


Figure 2.10.9. The Aerodynamic Coefficients Dependency on the Angle of Attack (α)

To get the precise trim conditions, it can be seen from C_m vs α curve that at Cm ≈ 0, α = 0.089 [*rad*] that is $\alpha \approx 5.099o \approx 5.1o$. Some of the parameters shown in Figure 2.10.9 such as C_Y vs α, C_l vs α, *and* C_n vs α curves display meaningless fluctuations; because they do not depend on the changes in the 20 angle of attack α. Additionally, these fluctuations are in the order of 10^{-17}, considered "zero".

2.10.11 The Aerodynamic Coefficients Dependency on the Side Slip Angle (β)

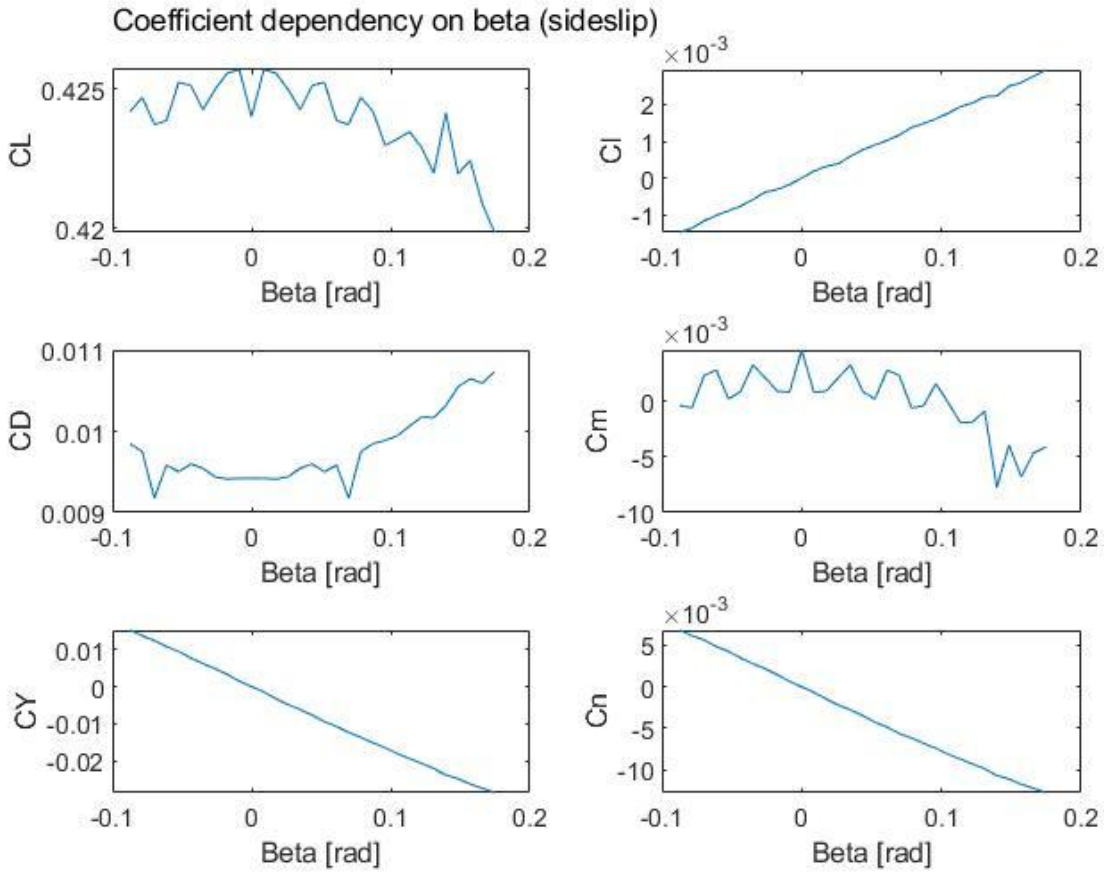


Figure 2.10.10.The Aerodynamic Coefficients Dependency on the Side Slip Angle (β)

2.10.12 The Aerodynamic Coefficients Dependency on the Roll Rate (*P*)

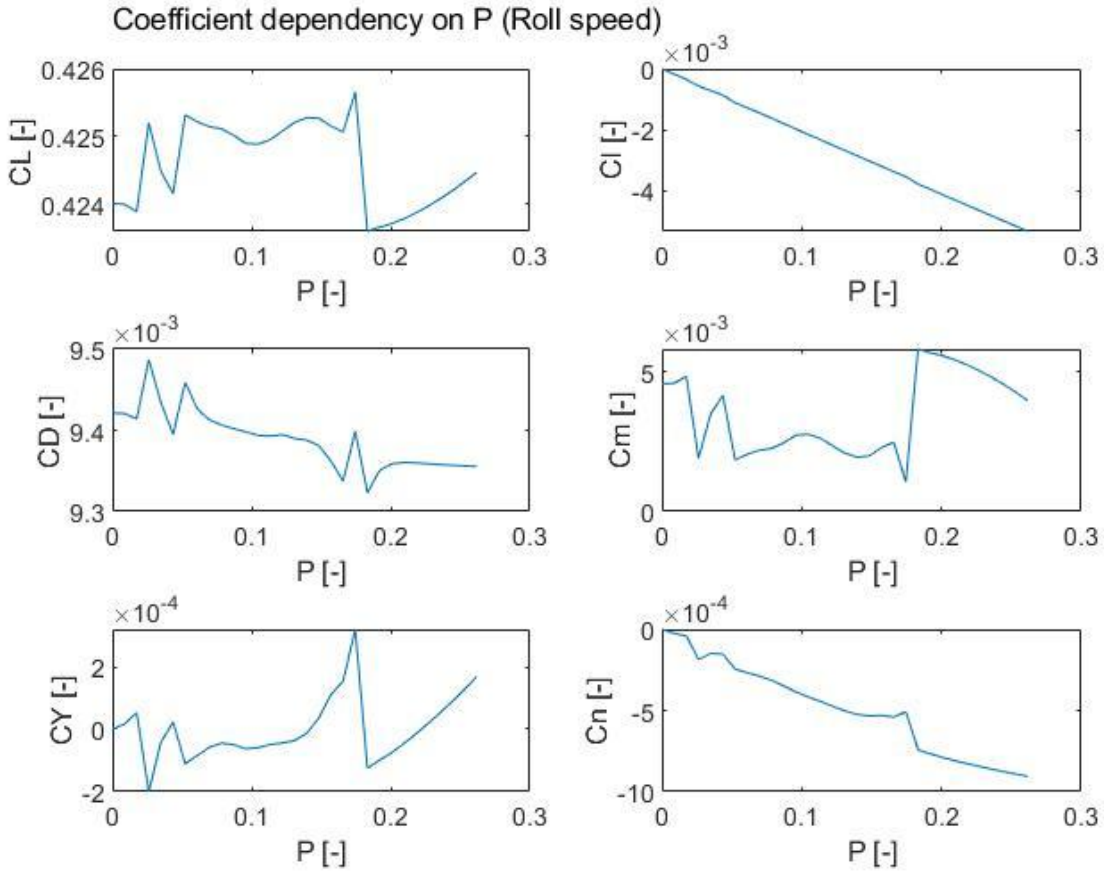


*Figure 2.10.11.The Aerodynamic Coefficients Dependency on the Roll Rate (**P**)*

2.10.13 The Aerodynamic Coefficients Dependency on the Pitch Rate (*Q*)

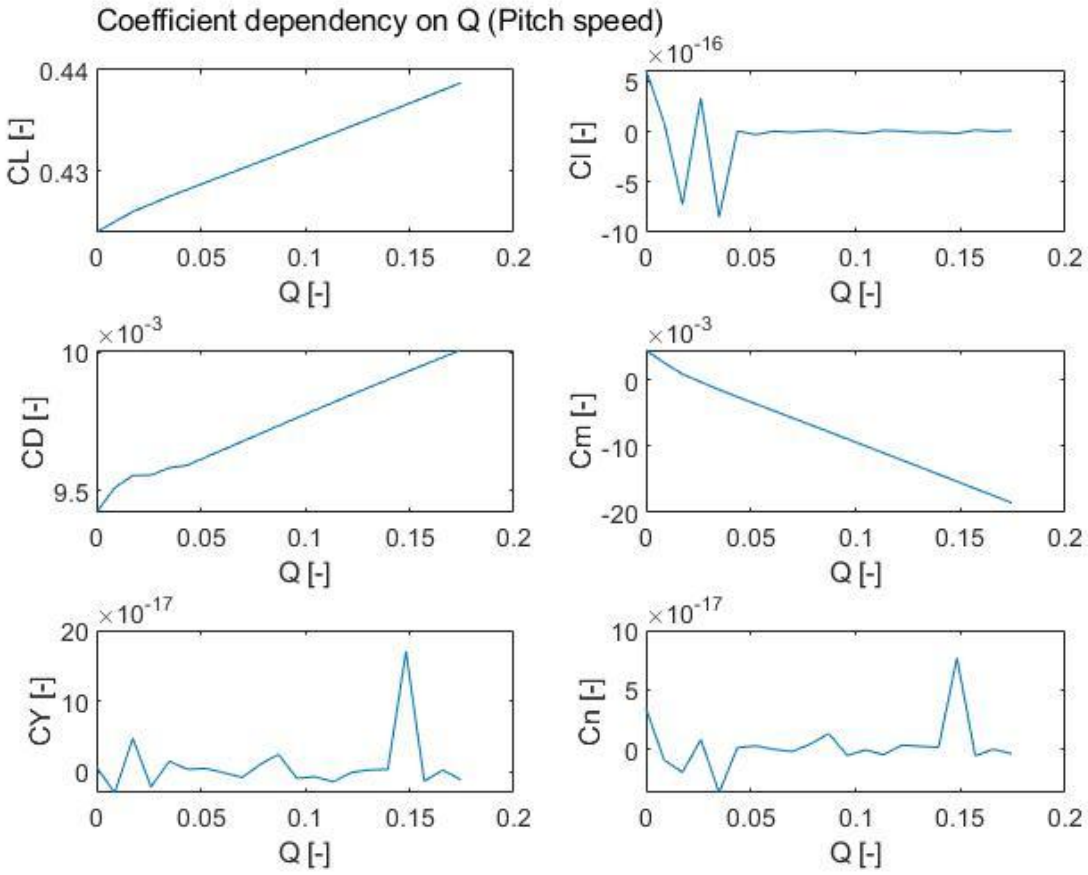


*Figure 2.10.12. The Aerodynamic Coefficients Dependency on the Pitch Rate (**Q**)*

2.10.14 The Aerodynamic Coefficients Dependency on the Deflection of the Aileron δ_a

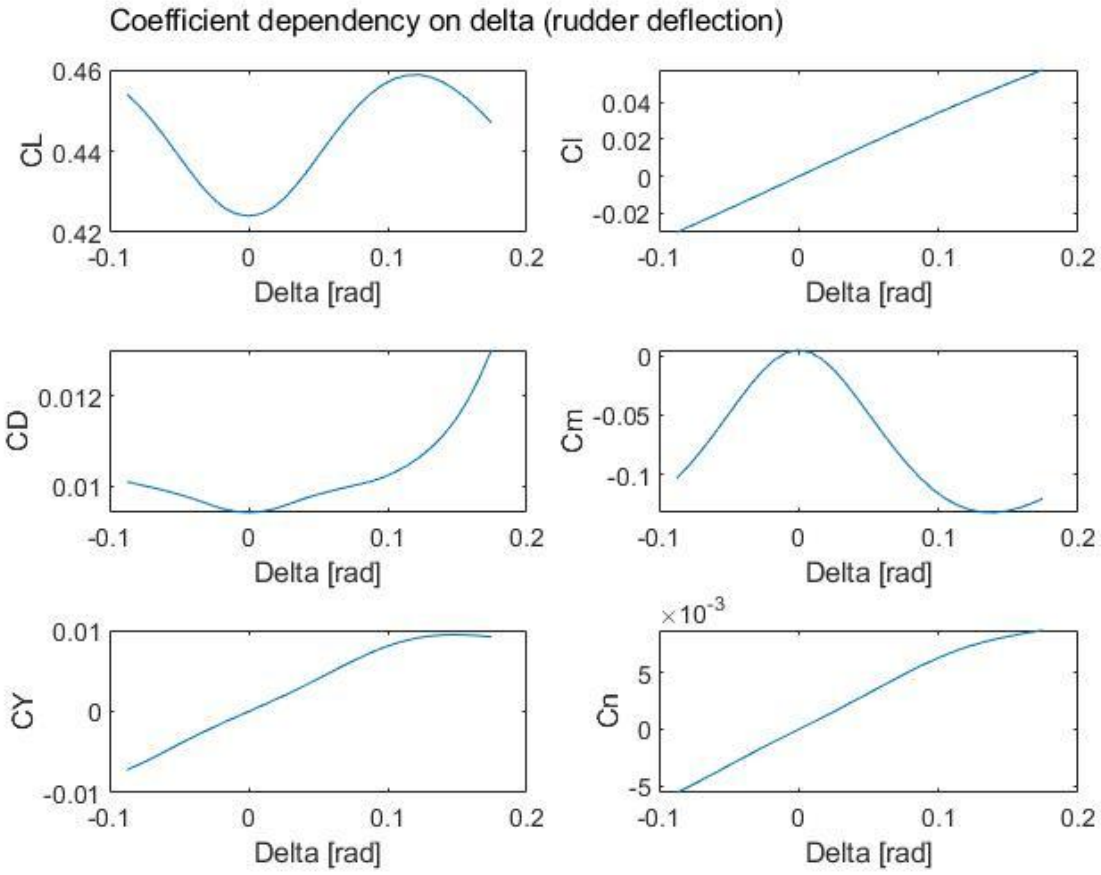


Figure 2.10.13. The Aerodynamic Coefficients Dependency on the Deflection of the Aileron ***δa***

2.10.15 The Aerodynamic Coefficients Dependency on the Deflection of the Elevator δ_e

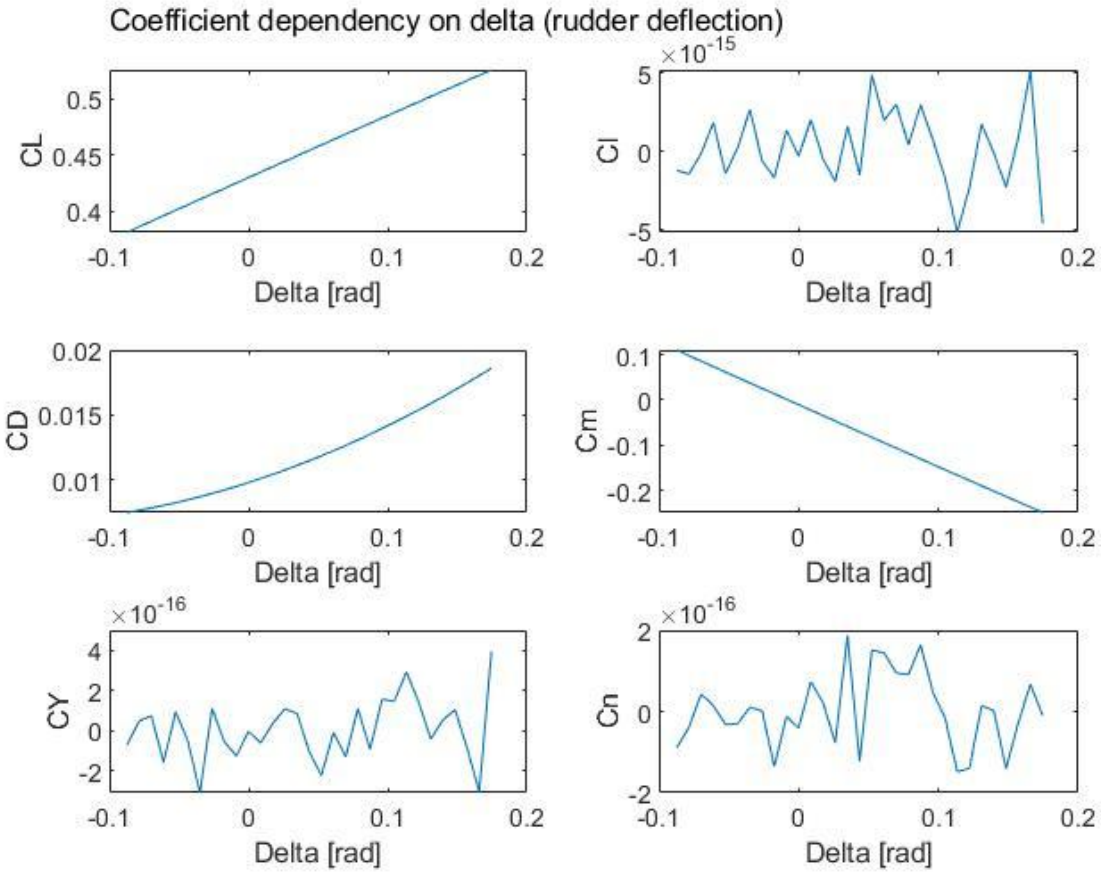


Figure 2.10.14. The Aerodynamic Coefficients Dependency on the Deflection of the Elevator ***δe***

2.10.16 The Aerodynamic Coefficients Dependency on the Deflection of the Rudder δ_r

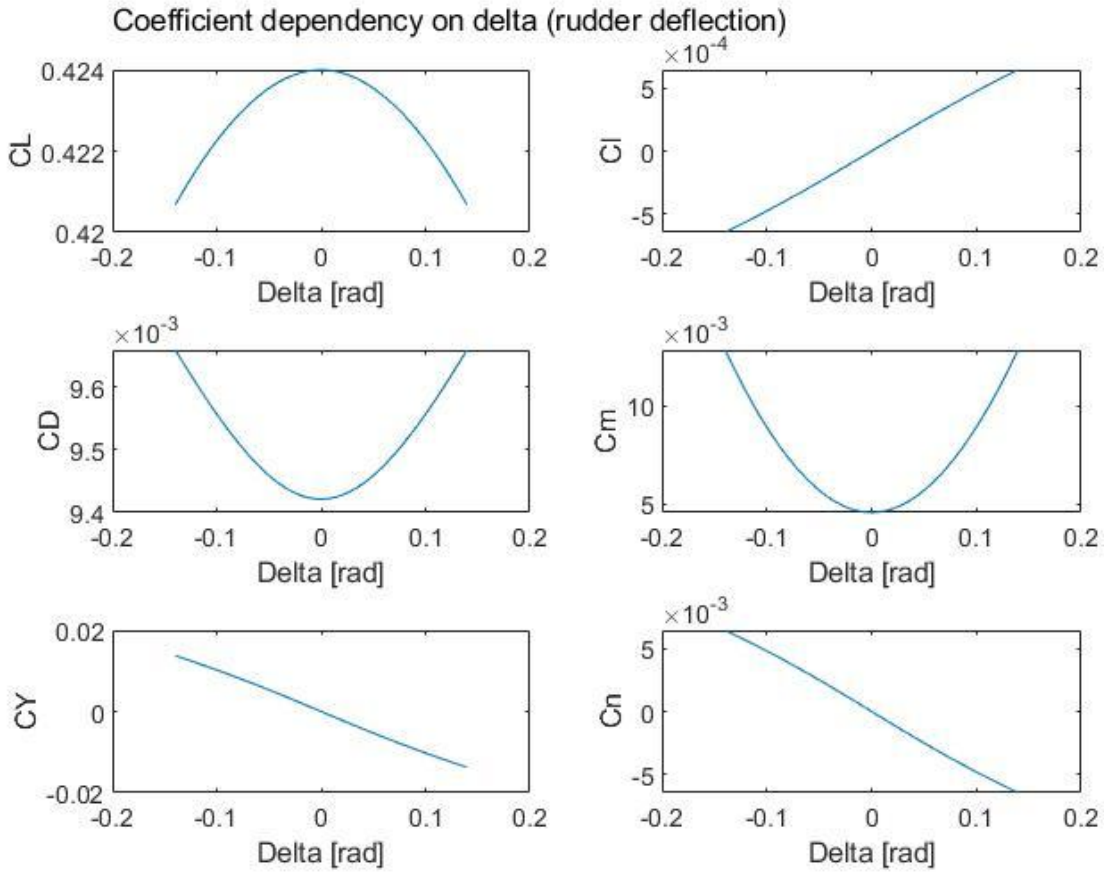


Figure 2.10.15. The Aerodynamic Coefficients Dependency on the Deflection of the Rudder δr

Chapter 3: VTOL Modeling

3.1. Fixed Wing Modeling

The modeling stage is the first step in identifying a dynamic system and controlling it. This is done by describing the governing differential equations to comprehend the temporal behavior of the system's states. In order to acquire these constants, a parameter estimation process is required. Once the parameters are approximated and the model is finished, the system is identified.

3.1.1 Introduction

The "Flight Stability and Automatic Control" reference (Nelson, 2010) is the main reference for the equations and axis systems defined in this chapter. Before determining the governing equations of motion, the axis system of the aircraft must be determined. Figure 3.1.1 shows the inertial frame system fixed to the Earth and the body axis system fixed to the airplane. Additionally, the aircraft's mass is believed to be fixed and it is presumed to be a rigid body.

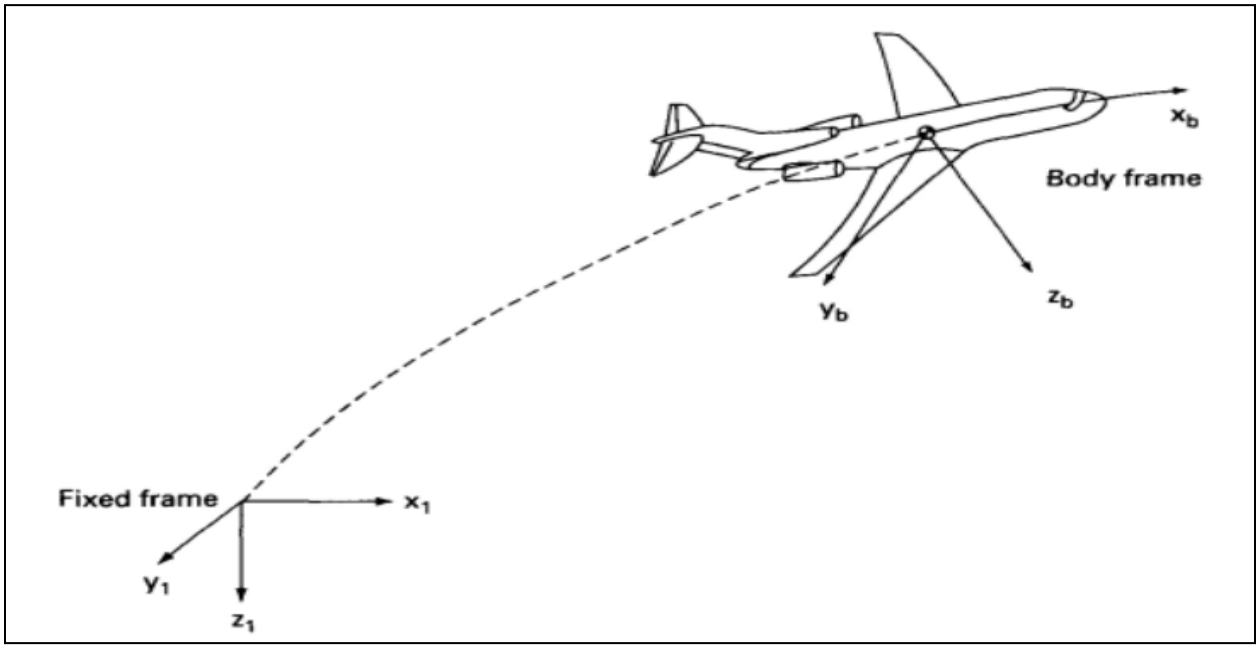


Figure 3.1.1. Inertial and body frames identification

3.1.2 Equations of motion

$$X - mgS\theta = m(\dot{u} + qw - rv)$$
$$Y + mgC\theta S\varphi = m(\dot{v} + ru - pw$$
$$Z + mgC\theta C\varphi = m(\dot{w} + pv - qu$$

Force Equations

$$L = I_x\dot{p} - I_{xz}\dot{r} + qr(I_z - I_y) - I_x$$
$$M = I_y\dot{q} + rq(I_x - I_z) + I_{xz}(p^2 -$$
$$N = -I_{xz}\dot{p} + I_z\dot{r} + pq(I_y - I_x) +$$

Moment Equations

$$p = \dot{\varphi} - \dot{\psi}S\theta$$
$$q = \dot{\theta}C\varphi + \dot{\psi}C\theta S\varphi$$
$$r = \dot{\psi}C\theta C\varphi - \dot{\theta}S\varphi$$

Body Angular Velocities in Terms of Euler Angles and Euler Rates

$$\dot{\theta} = qC\varphi - rS\varphi$$
$$\dot{\varphi} = p + qS\varphi T\theta + rC\varphi T\theta$$
$$\dot{\psi} = (qS\varphi + rC\varphi)sec\theta$$

Euler Rates in Terms of Euler Angles and Body Angular Velocities

3.1.3 Small disturbance Theory Linearization

The aircraft's motion is divided into two separate motions, longitudinal and lateral motions, with the X-force, Z-force, and M-Pitching moment representing the longitudinal motion and the Y-force, L-rolling moment, and N-yawing moment representing the lateral motion. Assuming a steady flight with small perturbations around the equilibrium point, the aerodynamic forces and moments are represented as a sum of Taylor series terms depending only on their derivatives and the disturbed states.

Along with a symmetric reference flying condition, the propulsive forces are also conveniently assumed to be constant. Additionally, the zero-initial conditions are as follows if the velocity vector direction is initially parallel to the x-axis:

$$v_o = p_o = q_o = r_o = \phi_o = \psi_o = w_o = 0$$

The linearized small-disturbance longitudinal and lateral rigid body equation of motion are:

Longitudinal Equations

$$(\frac{d}{dt} - X_u)\Delta u - X_w \Delta w + (gC\theta_o)\Delta\theta = X_{\delta_e}\Delta\delta_e + X_{\delta_T}\Delta\delta_T$$

$$-Z_u \Delta u + [(1 - Z_{\dot{w}})\frac{d}{dt} - Z_w]\Delta w - [(u_o + Z_q)\frac{d}{dt} - gS\theta_o]\Delta\theta = Z_{\delta_e}\Delta\delta_e + Z_{\delta_T}\Delta\delta_T$$

$$-M_u \Delta u - (M_{\dot{w}}\frac{d}{dt} + M_w)\Delta w + (\frac{d^2}{dt^2} - M_q\frac{d}{dt})\Delta\theta = M_{\delta_e}\Delta\delta_e + M_{\delta_T}\Delta\delta_T$$

Lateral Equations

$$(\frac{d}{dt} - Y_v)\Delta v - Y_p \Delta p + (u_o - Y_r)\Delta r - (gC\theta_o)\Delta\varphi = Y_{\delta_r}\Delta\delta_r$$

$$-L_v \Delta v + (\frac{d}{dt} - L_p)\Delta p - (\frac{I_{yz}}{I_x}\frac{d}{dt} + L_r)\Delta r = L_{\delta_a}\Delta\delta_a + L_{\delta_r}\Delta\delta_r$$

$$-N_v \Delta v - (\frac{I_{xz}}{I_z}\frac{d}{dt} + N_p)\Delta p + (\frac{d}{dt} - N_r)\Delta r = N_{\delta_a}\Delta\delta_a + N_{\delta_r}\Delta\delta_r$$

3.1.4 Stability Derivatives

Longitudinal Derivatives

u	$X_u = \frac{Q\,S}{m\,u_o}(2C_{Xo} + C_{X_u})$	$Z_u = \frac{Q\,S}{m\,u_o}(2C_{Zo} + C_{Z_u})$	$M_u = \frac{Q\,S\,c}{I_y\,u_o} C_{M_u}$
w	$X_w = \frac{Q\,S}{m\,u_o} C_{X_\alpha}$	$Z_w = \frac{Q\,S}{m\,u_o} C_{Z_\alpha}$	$M_w = \frac{Q\,S\,c}{I_y\,u_o} C_{M_\alpha}$
$\dot{w}$		$Z_{\dot{w}} = \frac{Q\,S\,c}{2m\,u_o^2} C_{Z_{\dot{\alpha}}}$	$M_{\dot{w}} = \frac{Q\,S\,c^2}{2I_y\,u_o^2} C_{Z_{\dot{\alpha}}}$
q		$Z_q = \frac{Q\,S\,c}{2m\,u_o} C_{Z_q}$	$M_q = \frac{Q\,S\,c^2}{2I_y\,u_o^2} C_{M_q}$
δ_e	$X_{\delta_e} = \frac{Q\,S}{m} C_{X_{\delta_e}}$	$Z_{\delta_e} = \frac{Q\,S}{m} C_{Z_{\delta_e}}$	$M_{\delta_e} = \frac{Q\,S\,c}{I_y} C_{M_{\delta_e}}$
δ_T	$X_{\delta_T} = \frac{Q\,S}{m} C_{X_{\delta_T}}$	$Z_{\delta_T} = \frac{Q\,S}{m} C_{Z_{\delta_T}}$	$M_{\delta_T} = \frac{Q\,S\,c}{I_y} C_{M_{\delta_T}}$

Table 3.1.1 Longitudinal Stability Derivatives

Lateral Derivatives

β	$Y_\beta = \frac{Q\,S}{m} C_{Y_\beta}$	$N_\beta = \frac{Q\,S\,b}{I_z} C_{N_\beta}$	$L_\beta = \frac{Q\,S\,b}{I_x} C_{L_\beta}$
p	$Y_p = \frac{Q\,S\,b}{2m\,u_o} C_{Y_\alpha}$	$N_p = \frac{Q\,S\,b^2}{2\,I_x\,u_o} C_{N_p}$	$L_p = \frac{Q\,S\,b^2}{2I_x\,u_o} C_{L_p}$
r	$Y_r = \frac{Q\,Sb}{2m\,u_o} C_{Y_r}$	$N_r = \frac{Q\,S\,b^2}{2\,I_x\,u_o} C_{N_r}$	$L_r = \frac{Q\,S\,b^2}{2I_x\,u_o} C_{L_r}$
δ_a	$Y_{\delta_e} = \frac{Q\,S}{m} C_{Y_{\delta_e}}$	$N_{\delta_a} = \frac{Q\,S\,b}{I_z} C_{N_{\delta_e}}$	$L_{\delta_a} = \frac{Q\,S\,b}{I_x} C_{L_{\delta_e}}$
δ_r	$Y_{\delta_T} = \frac{Q\,S}{m} C_{Y_{\delta_T}}$	$N_{\delta_r} = \frac{Q\,S\,b}{I_z} C_{N_{\delta_T}}$	$L_{\delta_r} = \frac{Q\,S\,b}{I_x} C_{L_{\delta_T}}$

Table 3.1.2 Lateral Stability Derivatives

3.1.5 Parameter Estimation

The system is put into state space formulation, with addition of altitude in the longitudinal mode.

$$x_{5x1}' = A_{5x5}x_{5x1} + B_{5x2}u_{2x1}$$

3.1.5.1 Longitudinal Mode

Where x is a 5-by-1 vector containing the states $\Delta u, \Delta w, \Delta q, \Delta\theta$, and Δh respectively.

$A =$

$$\begin{vmatrix} X_u & X_w & 0 & -g\cos(\theta_o) & 0 \\ \frac{Z_u}{1-Z_{\dot{w}}} & \frac{Z_w}{1-Z_{\dot{w}}} & \frac{u_o+Z_q}{1-Z_{\dot{w}}} & -\frac{gSin(\theta_o)}{1-Z_{\dot{w}}} & 0 \\ M_u+\frac{M_{\dot{w}}Z_u}{1-Z_{\dot{w}}} & M_w+\frac{M_{\dot{w}}Z_w}{1-Z_{\dot{w}}} & M_q+\frac{M_{\dot{w}}(u_o+Z_u)}{1-Z_{\dot{w}}} & -\frac{gM_{\dot{w}}Sin(\theta_o)}{1-Z_{\dot{w}}} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -Sin(\theta_o) & -Cos(\theta_o) & 0 & u_oCos(\theta_o)+w_oSin(\theta_o) & 0 \end{vmatrix}$$

$=$

$$\begin{vmatrix} -0.03645 & 0.2708 & 0 & -9.7712 & 0 \\ -1.1133 & -6.3299 & 16.8276 & -0.8839 & 0 \\ 0.26767 & 0.7402 & 12.1758 & 0.2125 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -0.090000 & -0.9960 & 0 & 18.0000 & 0 \end{vmatrix}$$

$B =$

$$\begin{vmatrix} X_{\delta_e} & X_{\delta_T} \\ \frac{Z_{\delta_e}}{1 - Z_{\dot{w}}} & \frac{Z_{\delta_T}}{1 - Z_{\dot{w}}} \\ M_{\delta_e} + \frac{M_{\dot{w}} Z_{\delta_e}}{1 - Z_{\dot{w}}} & M_{\delta_T} + \frac{M_{\dot{w}} Z_{\delta_T}}{1 - Z_{\dot{w}}} \\ 0 & 0 \\ 0 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 92.3979 \\ -13.1360 & 0 \\ -115.4777 & 3.5190 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}$$

These parameters were used to obtain the system's poles using MATLAB, and the poles of the longitudinal motion were discovered to be stable in both the short and long periods as shown below:

$p =$

0.0000 + 0.0000i

-13.8553 + 0.0000i

-4.6201 + 0.0000i

-0.0334 + 0.3624i

-0.0334 - 0.3624i

3.1.5.2 Lateral Mode

Where x is a 5-by-1 vector containing the states Δβ, Δp, Δr, Δφ, and Δψ respectively.

$A =$

$$\begin{vmatrix} \frac{Y_\beta}{u_o} & Y_p & -(u_o - Y_r) & g\cos(\theta_o) & 0 \\ L_v^* + \frac{I_{xz}}{I_x}N_v^* & L_p^* + \frac{I_{xz}}{I_x}N_p^* & L_r^* + \frac{I_{xz}}{I_x}N_r^* & 0 & 0 \\ N_v^* + \frac{I_{xz}}{I_z}L_v^* & N_p^* + \frac{I_{xz}}{I_z}L_p^* & N_r^* + \frac{I_{xz}}{I_z}L_r^* & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{vmatrix}$$

$$\begin{vmatrix} -3.8190 & -0.0007535 & -17.7590 & 9.7717 & 0 \\ 7.1050 & -5.1799 & 0.07723 & 0 & 0 \\ 11.2492 & -0.3579 & -0.7479 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{vmatrix}$$

$B =$

$$\begin{vmatrix} 0 & Y_{\delta_T} \\ L_{\delta_a}^* + \frac{I_{xz}}{I_x}N_{\delta_a}^* & L_{\delta_r}^* + \frac{I_{xz}}{I_x}N_{\delta_r}^* \\ N_{\delta_a}^* + \frac{I_{xz}}{I_z}L_{\delta_a}^* & N_{\delta_r}^* + \frac{I_{xz}}{I_z}L_{\delta_r}^* \\ 0 & 0 \\ 0 & 0 \end{vmatrix} = \begin{vmatrix} 0 & -2.3832 \\ 80.02653 & 0.8649 \\ 6.1344 & -7.6614 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}$$

These parameters were used to obtain the system's poles using MATLAB, and the Lateral Motion poles were discovered to be unstable in the spiral mode and stable in the roll and Dutch roll modes as follows:

$$p =$$

00000 + 0.0000i

-4.6792. + 0.0000i

0.0636 + 0.0000i

-2.5656 +14.0126i

-2.5656 -14.0126i

3.1.6 Flying Qualities

The flying qualities of an aircraft are indications of its stability and manageability; they most commonly reflect the pilot's feel of the aircraft's controllability. They are influenced by the stage of flight and the type of aircraft. The size and maneuverability of aircraft are used to classify them. There are three stages that define the flying qualities:

Level 1	Clearly suitable flying abilities for the mission flight phase.
Level 2	Flying abilities are sufficient to complete the mission flight phase, but with a little increase in the burden of the pilot and/or a decrease in the efficacy of the mission, or both.
Level 3	Flying abilities that allow for safe aircraft control but result in an excessive workload, poor mission efficacy, or both. It is safe to end Category A flying stages, and Category B and C flight phases can be finished.

Table 3.1.3 Levels of Flying Qualities

Classification of Aircrafts	
Class I	Small, light aircraft, such as light trainers, small observation aircraft, and light utility aircraft.
Class II	Aircraft with a medium weight and low to medium maneuverability, such as trainers, heavy attack aircraft, tactical bombers, heavy attack aircraft, heavy utility/search and rescue aircraft, and light or medium transport/cargo/tankers.

Class III	Large, heavy, low-to-medium maneuverability aircraft, including trainers, heavy bombers, and transport, cargo, and tankers.
Class IV	Aircraft with a high degree of maneuverability, including fighter/interceptor, assault, tactical reconnaissance, observation, and trainer.

Table 3.1.4 Classification of Aircrafts

Flight Phase Categories	
Nonterminal Flight Phase	
Category A	Non-terminal flying phase requiring quick movements, accurate monitoring, or accurate flight-path control. The following activities fall under this category: air-to-air combat ground attack, weapon delivery/launch, aerial recovery, reconnaissance, in-flight refueling (receiver), terrain-following, anti submarine search, and close-formation flying.
Category B	Non-terminal flying phases that are often carried out via gradated maneuvers and without precise monitoring, however it's possible that precise flight-path control is needed. The following actions fall under this category: ascent, cruise, loiter, aerial delivery, in-flight refueling (tanker), descent, emergency descent, and emergency deceleration.
Terminal Flight Phases	
Category C	Terminal flying phases are often completed with a series of progressive maneuvers and typically need for precise flight-path control. This group includes takeoffs, catapult takeoffs, approaches, wave-offs/go-arounds, and landings.

Table 3.1.5 Flight Phase Categories

The UAV used in this project is a Level 1, Class 1, and a Category C aircraft.

3.1.6.1 Long Period Mode

Pole	Damping	Frequency (rad/seconds)	Time Constant (seconds)
-3.34e-02 + 3.62e-01i	9.18e-02	3.64e-01	2.99e+01
-3.34e-02 - 3.62e-01i	9.18e-02	3.64e-01	2.99e+01

Table 3.1.6 Characteristics of the Long Period Mode

	Damping ξ
Level 1 Flying Qualities	> 0.04
Project's UAV	0.0918

Table 3.1.7 Damping of UAV and Link to Flying Qualities

3.1.6.2 Short Period Mode

Pole	Damping	Frequency (rad/seconds)	Time Constant (seconds)
-4.62e+00	1	4.62e+00	2.16e-01
-1.39e+01	1	1.39e+01	7.22e-02

Table 3.1.6 Characteristics of the Short Period Mode

	Damping ξ
Level 1 Flying Qualities	$0.35 < \xi < 1.3$
Project's UAV	1

Table 3.1.7 Damping of UAV and Link to Flying Qualities

3.1.7 External Loads Acting on the Aircraft

3.1.7.1 Aerodynamic Forces

$$F_x = m(X_u \Delta u + X_w \Delta w + X_{\delta_e} \Delta \delta_e + X_{\delta_T} \Delta \delta_T)$$

$$F_y = m(Y_v \Delta v + Y_p \Delta p + Y_r \Delta r + Y_{\delta_a} \Delta \delta_a + Y_{\delta_r} \Delta \delta_r)$$

$$F_z = m(Z_u \Delta u + Z_w \Delta w + Z_{\dot{w}} \Delta \dot{w} + Z_q \Delta q + Z_{\delta_e} \Delta \delta_e + Z_{\delta_T} \Delta \delta_T)$$

3.1.7.2 Aerodynamic Moments

$$L = I_x(L_\beta \Delta \beta + L_p \Delta p + L_r \Delta r + L_{\delta_a} \Delta \delta_a + L_{\delta_r} \Delta \delta_r)$$

$$M = I_y(M_u \Delta u + M_w \Delta w + M_{\dot{w}} \Delta \dot{w} + M_q \Delta q + M_{\delta_e} \Delta \delta_e + M_{\delta_T} \Delta \delta_T)$$

$$N = I_z(N_\beta \Delta \beta + N_p \Delta p + N_r \Delta r + N_{\delta_a} \Delta \delta_a + N_{\delta_r} \Delta \delta_r)$$

3.1.8 Velocities and Angular Rates

3.1.8.1 Velocities

$$\dot{u} = rv - qw - gS\theta + \frac{F_x}{m}$$

$$\dot{v} = qu + pw + gS\varphi + \frac{F_y}{m}$$

$$\dot{w} = qu - pv - gC\theta + \frac{F_z}{m}$$

3.1.8.2 Angular Rates

$$\dot{p} = (C_1 r + C_2 p)q + C_3 L + C_4 N$$

$$\dot{q} = C_5 pr - C_6(p^2 - r^2) + C_7 M$$

$$\dot{r} = (C_8 p - C_2 r)q + C_4 L + C_9 N$$

Where

$$C_1 = \frac{(I_y - I_z)I_z - I_{xz}^2}{I_x I_z - I_{xz}^2}$$

$$C_2 = \frac{(I_x - I_y + I_z)I_{xz}}{I_x I_z - I_{xz}^2}$$

$$C_3 = \frac{I_z}{I_x I_z - I_{xz}^2}$$

$$C_4 = \frac{I_{xz}}{I_x I_z - I_{xz}^2}$$

$$C_5 = \frac{I_z - I_x}{I_y}$$

$$C_6 = \frac{I_{xy}}{I_y}$$

$$C_7 = \frac{1}{I_y}$$

$$C_8 = \frac{(I_x - I_y)I_x + I_{xz}^2}{I_x I_z - I_{xz}^2}$$

$$C_9 = \frac{I_x}{I_x I_z - I_{xz}^2}$$

3.1.9 Euler Angles

$$\dot{\varphi} = p + T\theta(qS\varphi + rC\varphi)$$

$$\dot{\theta} = qC\varphi + rS\varphi$$

$$\dot{\psi} = qS\varphi + rC\varphi \, sec\theta$$

3.1.10 Positions with Respect to the Fixed Frame

$$\dot{x} = uC\theta C\psi + v(-C\varphi S\psi + S\varphi S\theta C\psi) + w(S\varphi S\psi + C\varphi S\theta C\psi)$$

$$\dot{y} = uC\theta S\varphi + v(C\varphi C\psi + S\varphi S\theta S\psi) + w(-S\varphi C\psi + C\varphi S\theta S\psi)$$

$$\dot{z} = -uS\theta + vS\varphi S\theta + wC\varphi C\theta$$

These equations are used to model the system. Details will be explained in the following chapter.

3.2. Quadrotor Modeling

The quadrotor modeling section aims to provide a comprehensive understanding of the dynamic behavior of a quadrotor system. Accurate modeling of the quadrotor is crucial for developing effective control strategies and optimizing its performance. This section focuses on deriving a mathematical model that captures the essential dynamics and kinematics of the quadrotor, considering factors such as aerodynamics, propulsion, and mechanical constraints. By establishing a reliable model, the subsequent control design and analysis can be based on a solid foundation. The modeling process involves incorporating physical principles, empirical data, and system identification techniques to ensure a realistic representation of the quadrotor's behavior. Through this modeling endeavor, a deeper insight into the quadrotor's dynamics will be obtained, enabling the subsequent sections of the thesis to explore control methodologies that can exploit the system's characteristics and address specific control objectives.

3.2.1. Overview

A quadcopter is a four-rotor helicopter. It is an under-actuated, dynamic vehicle with four input speeds (one for each rotor) and six degrees of freedom (6DOF). Unlike regular helicopters that have variable-pitch angle rotors, a quadcopter has four fixed-pitch and fixed-angle rotors. The motion of a quadcopter in 6DOF is controlled by varying the speed of each rotor, which changes the lift, and rotational forces (Jiřinec, 2011). The quadcopter tilts toward the direction of a slow-spinning motor, which enables it to roll and pitch. Roll and pitch angles divide the thrust into two directions due to which linear motion is achieved. The rotors rotate in clockwise-anticlockwise pairs, to control the yaw produced due to the drag force on the propellers.

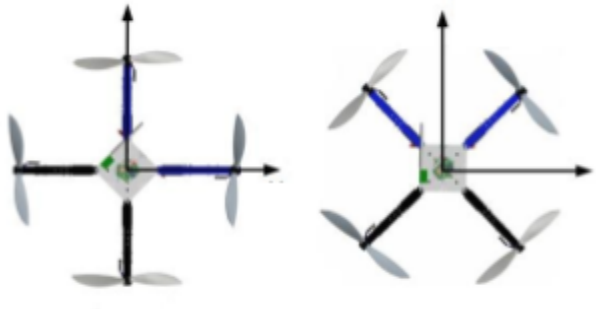


Fig.3.2.1. General quadrotor "plus" and "cross" configurations

The center of gravity (CG) lies almost in the same plane which contains all the rotors. Also, all four motors of the same class differ in efficiency. This differentiates it from helicopters,

and human control makes it very difficult to stabilize a quadcopter. Therefore, sophisticated control is essential for a balanced flight of a quadcopter (Jiřinec, 2011).

Quadrotors can vary in the setup configuration. The most popular two configurations are the plus and cross configurations, as illustrated in Fig. 3.2.1, for a general quadrotor. This study utilizes a cross-configuration quadrotor that is indicated in Fig. 3.2.2. The utilization of a cross configuration quadrotor offers a range of benefits in various applications. This type of quadrotor design, with its four rotors positioned in a cross-shaped configuration, as indicated in Fig. 3.2.2 and as stated by Niemiec and Gandhi (2016) provides enhanced stability and maneuverability. The balanced distribution of thrust and control enables the quadrotor to perform agile movements, making it suitable for tasks that require precise positioning and control in both indoor and outdoor environments.

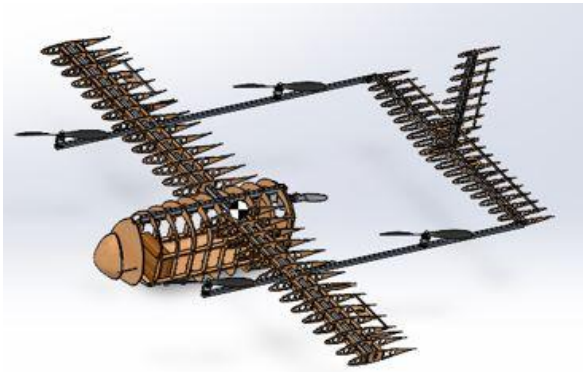


Fig.3.2.2. This study's VTOL has a cross-configuration quadrotor.

One area where the cross configurations quadrotor excels is in aerial photography and videography. Due to its stable flight characteristics and ability to hover in place, it can capture stunning aerial shots with smooth and steady camera movements. This makes it a valuable tool for filmmakers, photographers, and content creators who aim to capture breathtaking visuals from unique angles (Mellinger, & Warren, 2012).

In summary, the utilization of a cross configuration quadrotor offers immense potential across various domains. Its stability, maneuverability, and adaptability make it an invaluable tool for aerial photography, search and rescue operations, surveillance, inspection, and numerous other applications where precise control and versatility are essential.

3.2.2. Equations of Motion

The derivation of the mathematical model for the equations of motion in the domain of quadrotor dynamics serves as a fundamental step toward comprehending and analyzing the behavior and control of these aerial vehicles. This mathematical model provides a precise and quantitative representation of the quadrotor's motion, thereby enabling the development of robust control algorithms, rigorous simulation studies, and accurate performance evaluations.

3.2.2.1. Transformation Matrix

Firstly the transformation matrix that transforms from the inertial fixed frame to the quadrotors frame is defined by the Euler angles as

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$

Where R_x is the roll rotation matrix about the x axis

$$R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

Where R_y is the pitch rotation matrix about the y axis

$$R_z = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where R_z is the yaw rotation matrix about the z axis

Now, The trnasformation matrix $R(\phi,\ \theta,\ \psi)$ that transforms from the inertial fixed frame to the quadrotor's frame is

$$R_{x,y,z}(\phi,\ \theta,\ \psi) = R_x(\phi)R_y(\theta)R_z(\psi)$$

Hence, the equation relates the rates of the quadrotor’s position change with respect to the intirtial fixed inertial frame is

$$[x_n\ y_e\ z_d]^T = R_{x,y,z}(\phi,\ \theta,\ \psi)\ [u\ v\ w]^T$$

Where x_n is the x position in the inertial frame directed towards North, y_e is the y position in the inertial frame directed towards East, z_d is the z position in the inertial frame directed towards Down, hence the inertial frame directions are North-East-Down (NED), and the $u, v, and\ w$ are the velocity states in the quadrotor's frame.

3.2.2.2. Angular Rate Transformation

The transformation between angular rates of the quadrotor to inertial fixed frame is given by the relation (Cook, 1997)

$$d/dt\ ([\phi\ \theta\ \psi]^T) = R_y(\theta)\ *\ R_x(\phi)\ [p\ q\ r]^T$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

3.2.2.3. Linear Acceleration

The linear acceleration is defined in the inertial fixed frame by Newton's Second Law as

$$F = m\ d/dt\ (V)$$

Where m is the quradrotor's mass and V is the quadrotor's velocity vector that consistent of velocity 3 components $u, v,\ and\ w$. As the quadrotor moves and rotates it changes and rotates the velocity vector, as well. So, the derivative of the velocity vector needs to account for the rotational speed as well as the rate of the change of the velocity components.

$$F = m\ d/dt\ (V) + \omega \times mV$$

That is

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + m \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Furtherly analyzing the forces acting on the quadrotor, it is reached that the forces acting on the quadrotor are the weight force, the thrust force generated by the motors, neglecting the aerodynamics forces. Hence,

$$F_x = 0,\ F_y = 0,\ F_z = mg - T$$

Where g is the gravitational acceleration, and T is the thrust force.

As the force vector is defined in the inertial frame, it needs to be transformed to the quadrotor moving frame by multiplying the transformational matrix by the forces vector as,

$$F_b = R_{x,y,z}(\phi,\ \theta,\ \psi).F$$

Rewriting, and combining the force equation,

$$\dot{u} = rv - qw - g\sin\theta$$
$$\dot{v} = pw - ru + g\cos\theta\sin\phi$$
$$\dot{w} = qu - pv + g\cos\phi\cos\theta - \frac{T}{m}$$

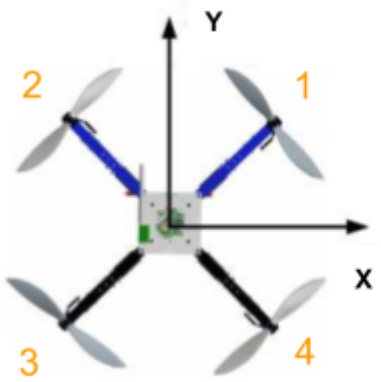


Fig.3.2.3. The quadrotor motors numbering

As Homann et al. (2007) indicated that with neglecting the motor dynamics, the thrust force is proportional to square the angular rate of propeller's rotation. Then the total thrust would be,

$$T = k(\Omega_1^{\ 2} + \Omega_2^{\ 2} + \Omega_3^{\ 2} + \Omega_4^{\ 2})$$

Where k is the thrust coefficient, and Ω_i is the angular speed of the i^{th} motor. The motors numberting is illustrated in Fig.3.2.3.

3.2.2.4. Angular Acceleration

As the change of angular momentum equals the external torque applied on the quadrotor, then

$$M = d/dt(H)$$

Hence the quadrotors changes in direction which implies that the angular momentum change in direction, as well, then the total derivative of angular momentum is,

$$M = d/dt(H) + \omega \times H$$

The angular momentum is evaluated by the product of the quadrotor's moment of inertia and its rotational speed

$$H = I\,\omega$$

Where I is the diagonal moment of inertia of the quadrotor.

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

Then after expanding the equations,

$$M_x = \dot{p}\,I_x + qr\,(I_z - I_y)$$
$$M_y = \dot{q}\,I_y + pr\,(I_x - I_z)$$
$$M_z = \dot{r}\,I_z + pq\,(I_y - I_x)$$

The external moments are just the torques generated by quadrotor motors, that are expressed as

$$M_x = l\,k(-\,\Omega_1^{\,2} + \Omega_2^{\,2} + \Omega_3^{\,2} - \Omega_4^{\,2})$$

$$M_y = l\,k(\Omega_1^{\,2} + \Omega_2^{\,2} - \Omega_3^{\,2} - \Omega_4^{\,2})$$

$$M_x = d\,(\Omega_1^{\,2} + \Omega_2^{\,2} + \Omega_3^{\,2} + \Omega_4^{\,2})$$

Where d is the drag coefficient.

Then,

$$\dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{1}{I_x} M_x$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{1}{I_y} M_y$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{1}{I_z} M_z$$

Now the whole mathematical model is constructed that has a 12 state which are

$$states = [x_n\ y_e\ z_d\ \dot{u}\ \dot{v}\ \dot{w}\ \phi\ \theta\ \psi\ p\ q\ r]^T$$

Chapter 4: Modeling Verification

4.1. Fixed Wing Model Verification

4.1.1 Kinematics Model (Nonlinear)

The block's inputs are the propelling and aerodynamic forces and moments, and its outputs are the aircraft's states, such as Euler angles, angular rates, velocities in body and fixed frames, and position with respect to a fixed frame. This block simulates the aircraft's nonlinear equations of motion and the conversion from the body to the inertial axis.

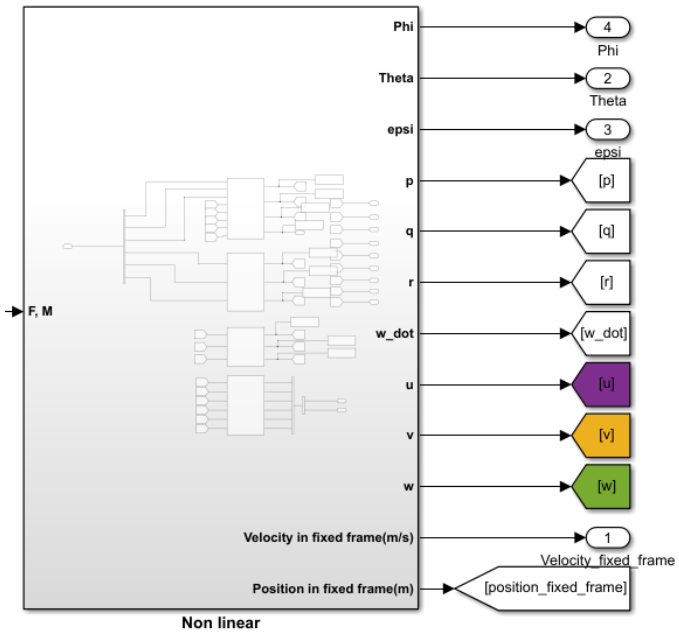


Figure 4.1.1. The Nonlinear Dynamics Block

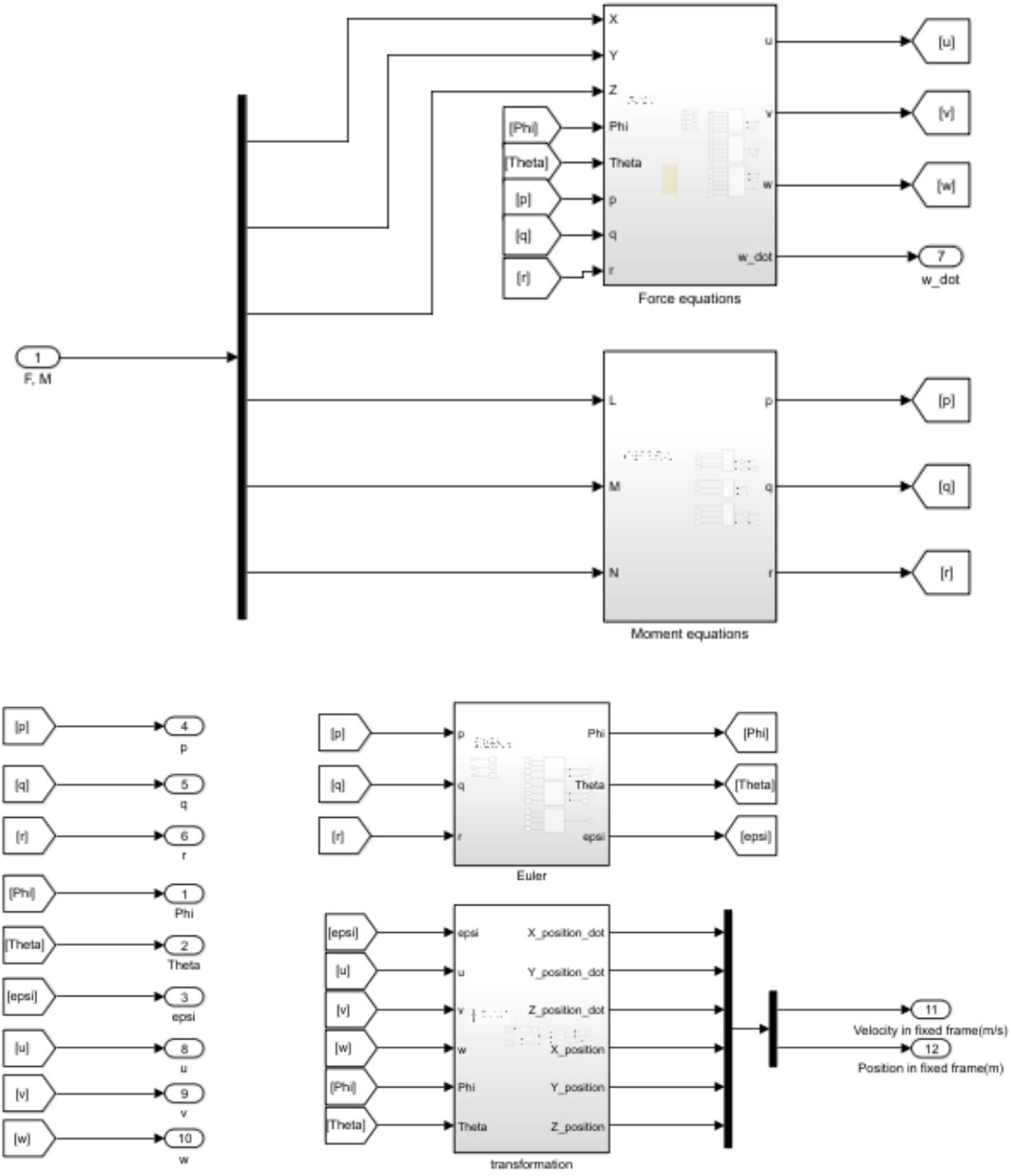


Figure 4.1.2. Nonlinear Dynamics Block Components

4.1.1.1 Euler Rates

This subsystem outputs the Euler angles (Pitch, Roll and Yaw).

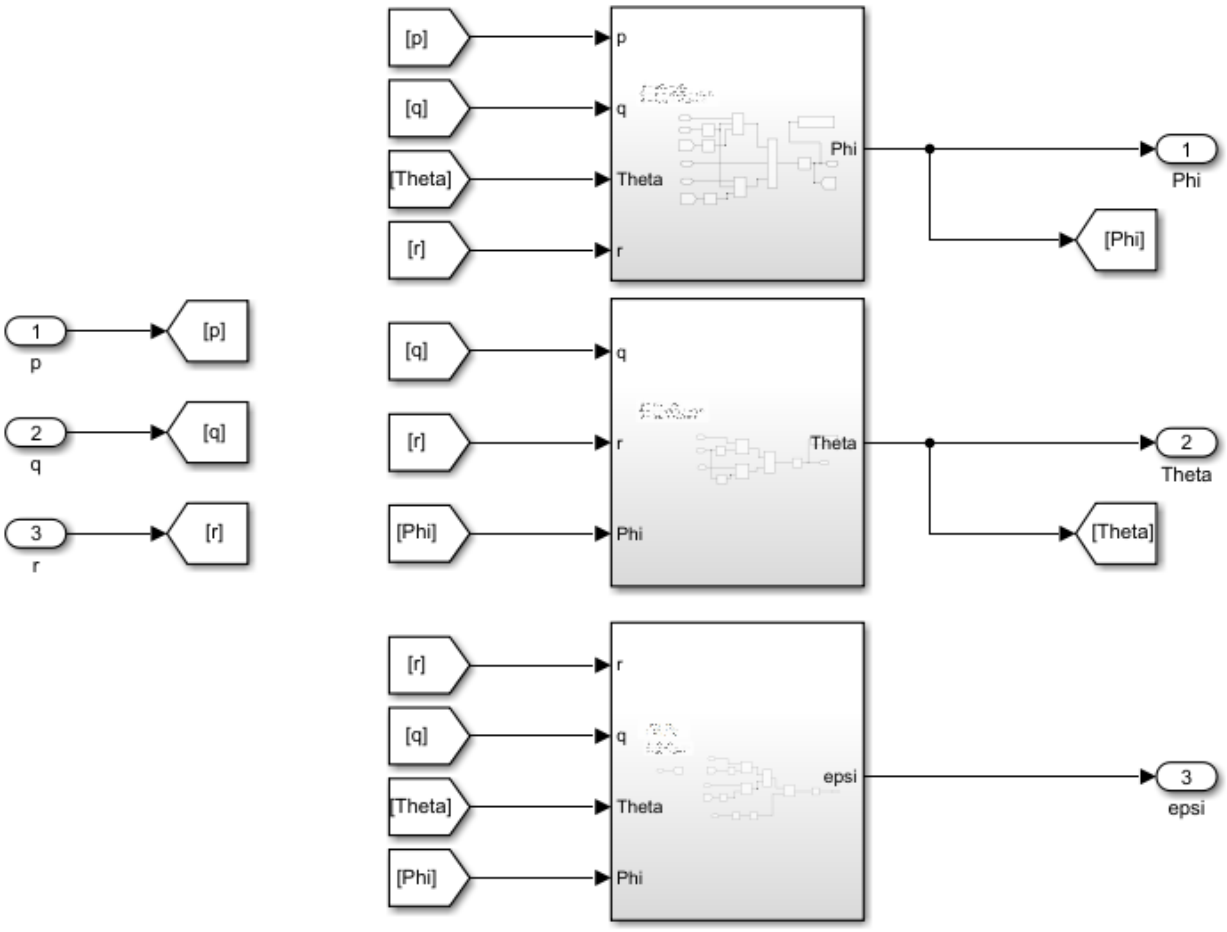


Figure 4.1.3. The Euler Rates Block

4.1.1.2 Force Equations

This sub-block models the force equations, giving velocities in the body frame (u, v, and w).

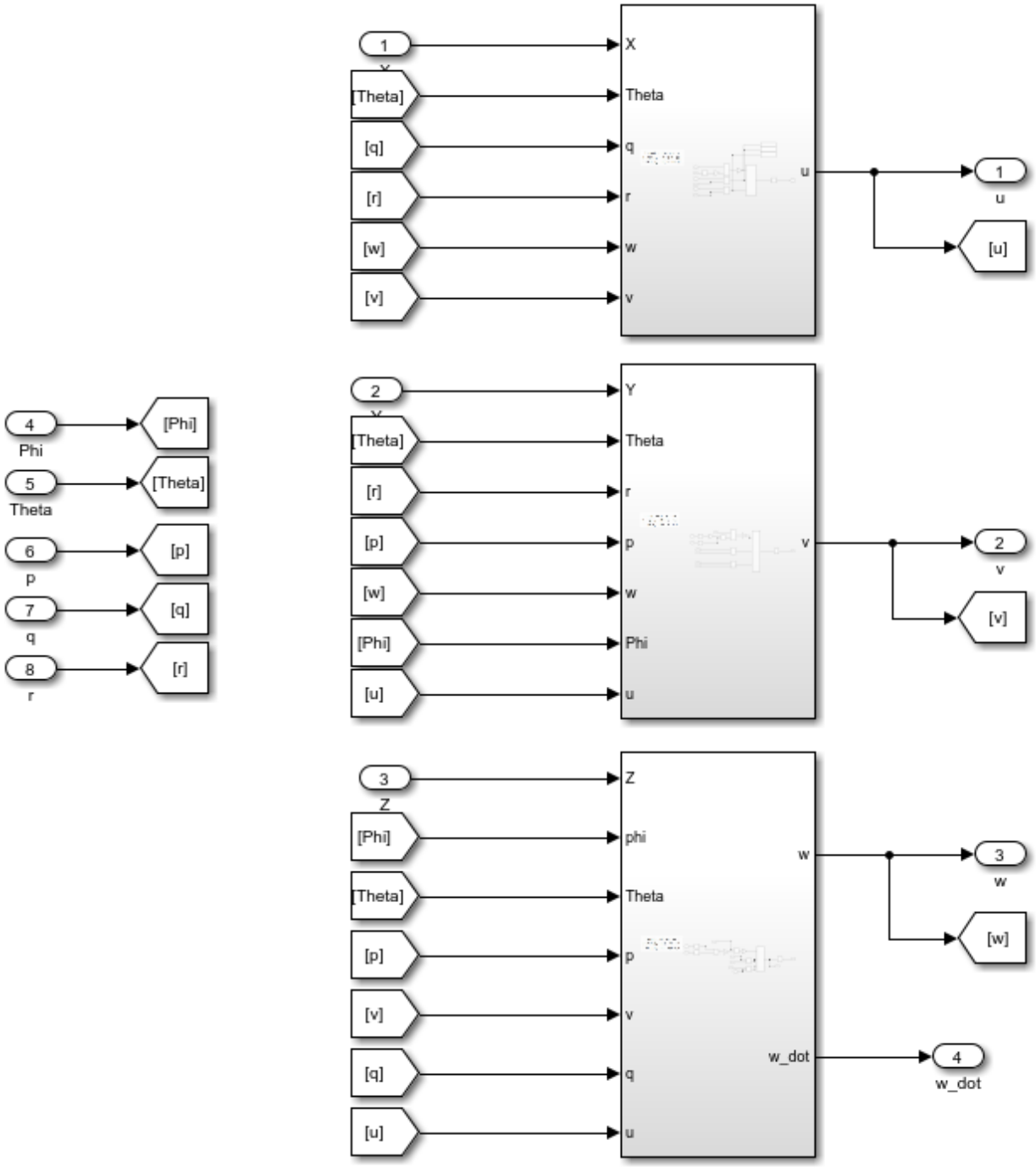


Figure 4.1.4. The Force Equations Block

4.1.1.3 Angular Velocities

The sub-block models the moment equations, giving the airplane's angular velocities (p, q, and r).

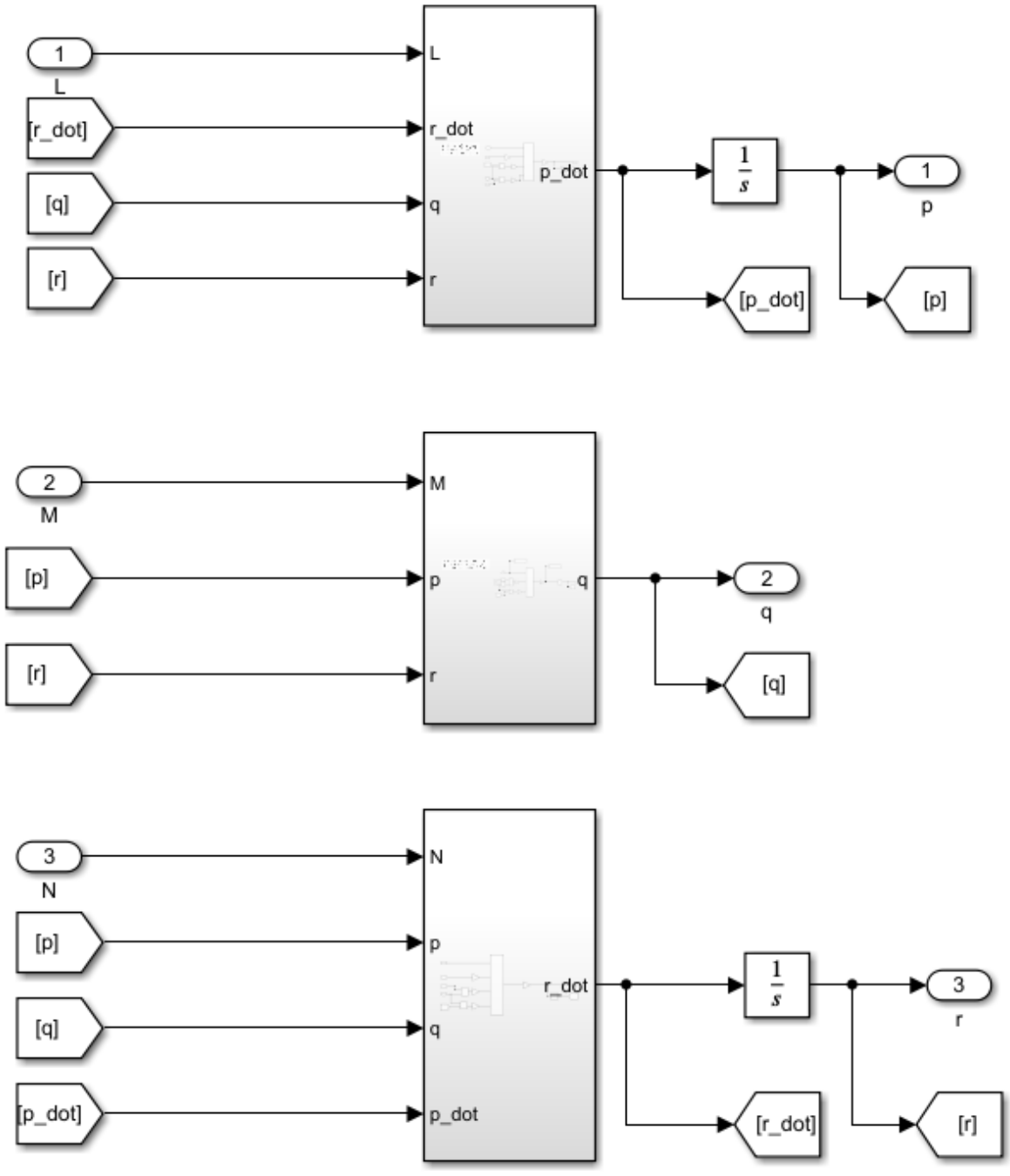


Figure 4.1.5. The Angular Velocities Block

4.1.1.4 Transformation to Fixed Axes

The block generates the Direct Cosine Matrix for translation from body to Fixed axis using the Euler angles and the body axis velocity vector, providing the velocities and location of the aircraft in the Fixed axis.

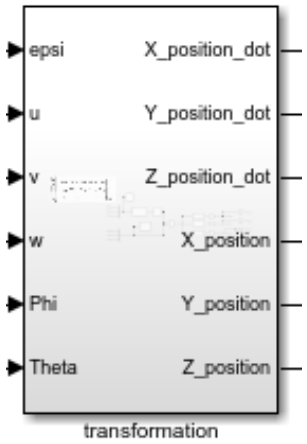


Figure 4.1.6. The Transformation to Fixed Axes Block

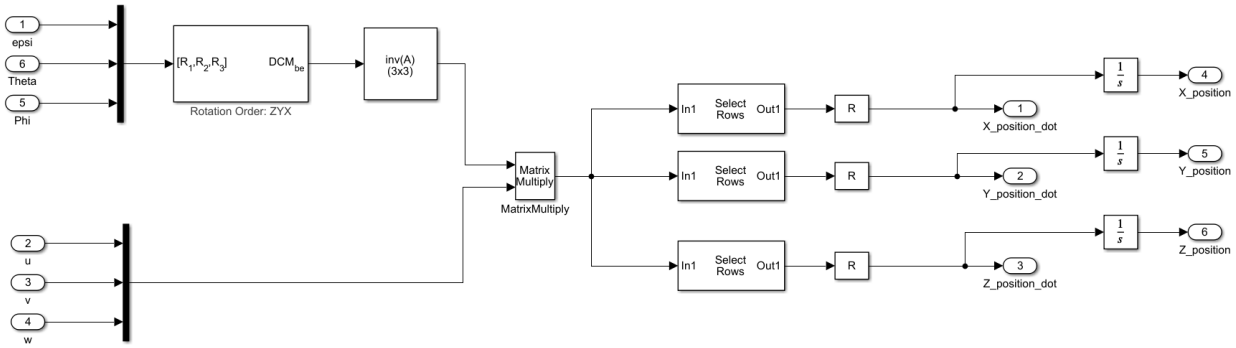


Figure 4.1.7. The Transformation to Fixed Axes Block Components

4.1.2 Aerodynamics and Propulsive Force and Moments Model

The block applies the small disturbance theory to the linearized force and moments model. The stability derivatives, the states, and the deflection of the control surfaces are used to describe the change in the force or moments resulting from a tiny disruption in the plane's trim conditions. This gives the overall force and moment operating on the airplane when added to the previously known forces and moments at the trim state.

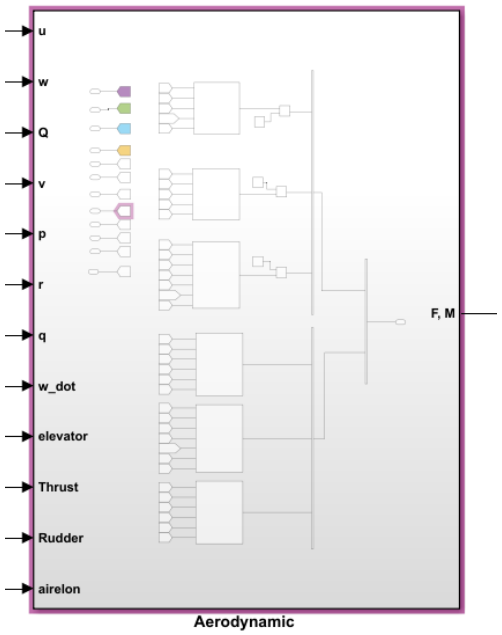


Figure 4.1.8. The Transformation to Fixed Axes Block Components

For example, one force and one moment subsystems will be shown.

4.1.2.1 ΔFx due to Disturbance

This sub-block calculates the change of the force in the x-direction using its stability derivatives.

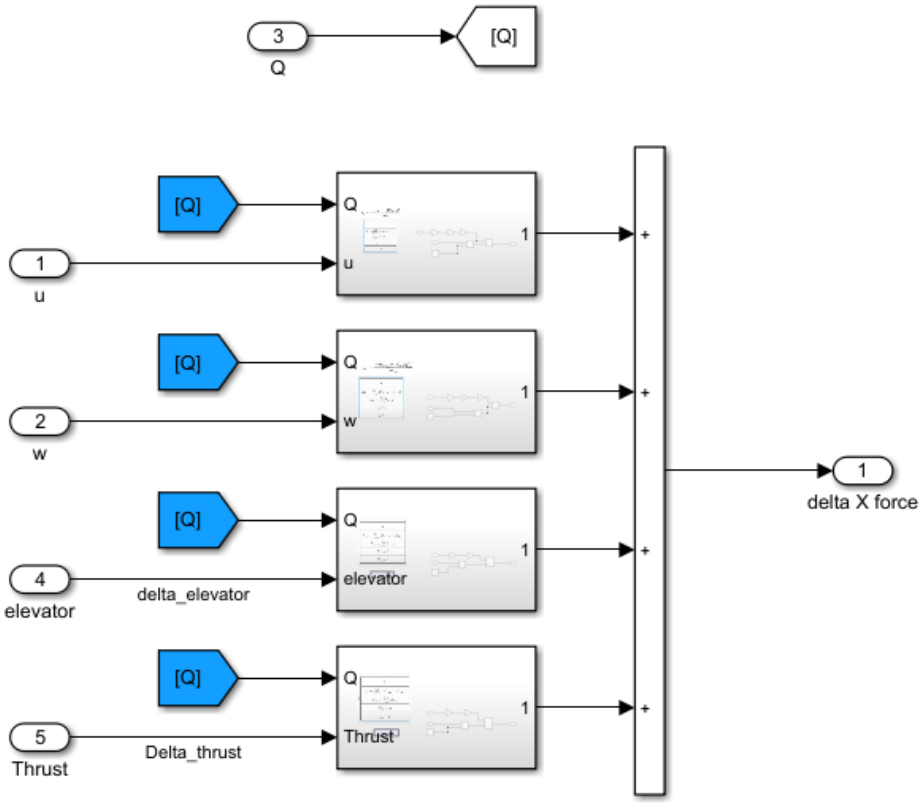


Figure 4.1.9. Force is x-direction due to Disturbance Block Components

4.1.2.2 ΔM (Moment around Y-axis)

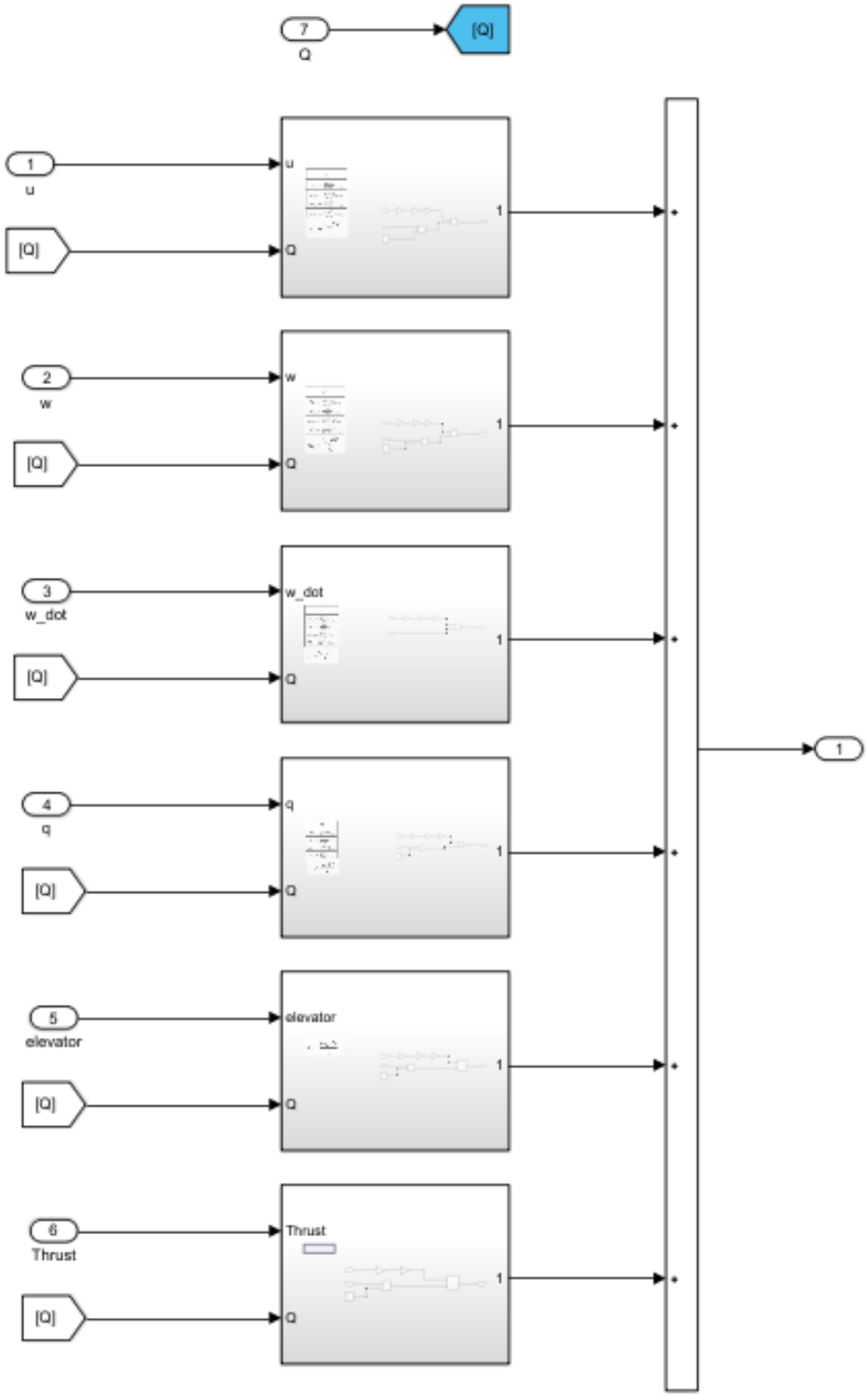


Figure 4.1.10. Moment around y-axis due to Disturbance Block Components

4.1.3 Atmospheric Properties

This block determines the density, temperature, and dynamic pressure of the atmosphere as they relate to changes in the aircraft's height.

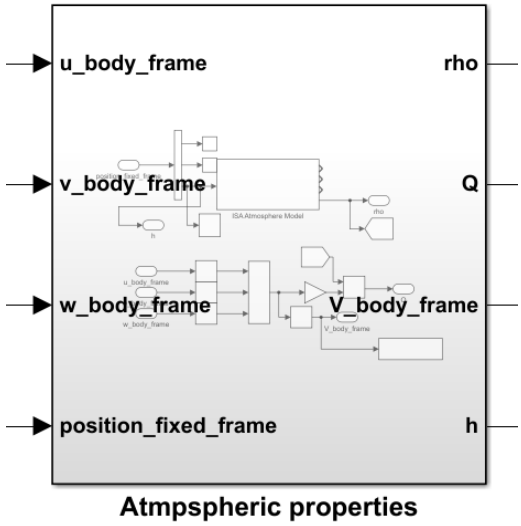


Figure 4.1.11. Atmospheric Properties Block

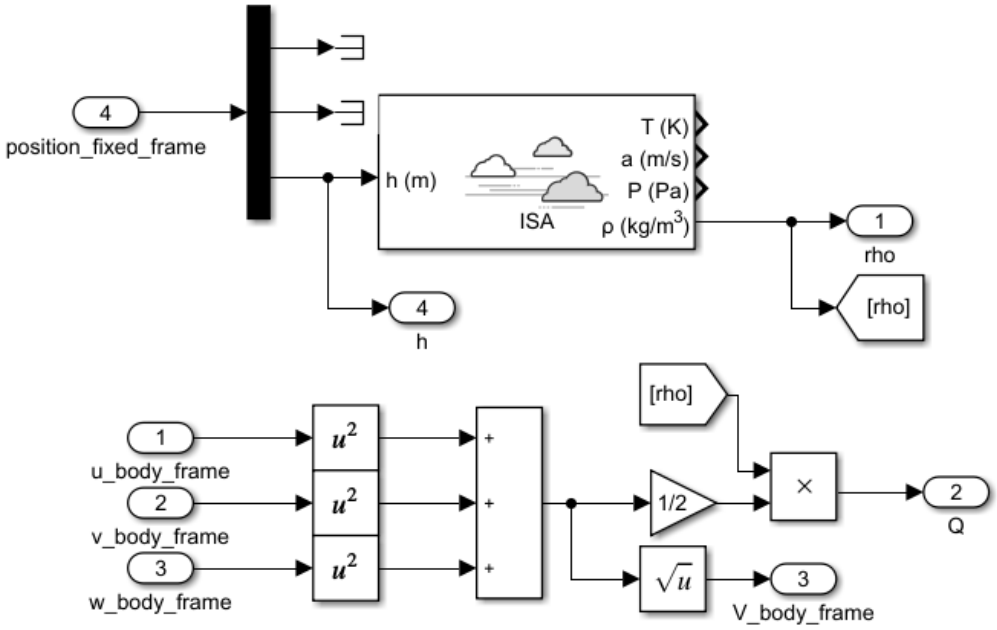


Figure 4.1.12. Atmospheric Properties Block Components

4.1.4 Aircraft Uncontrolled Response due to Disturbances from Trim Condition

4.1.4.1 Longitudinal Responses

4.1.4.1.1. Response due to +2 deg Disturbance in Pitch Angle (θ)

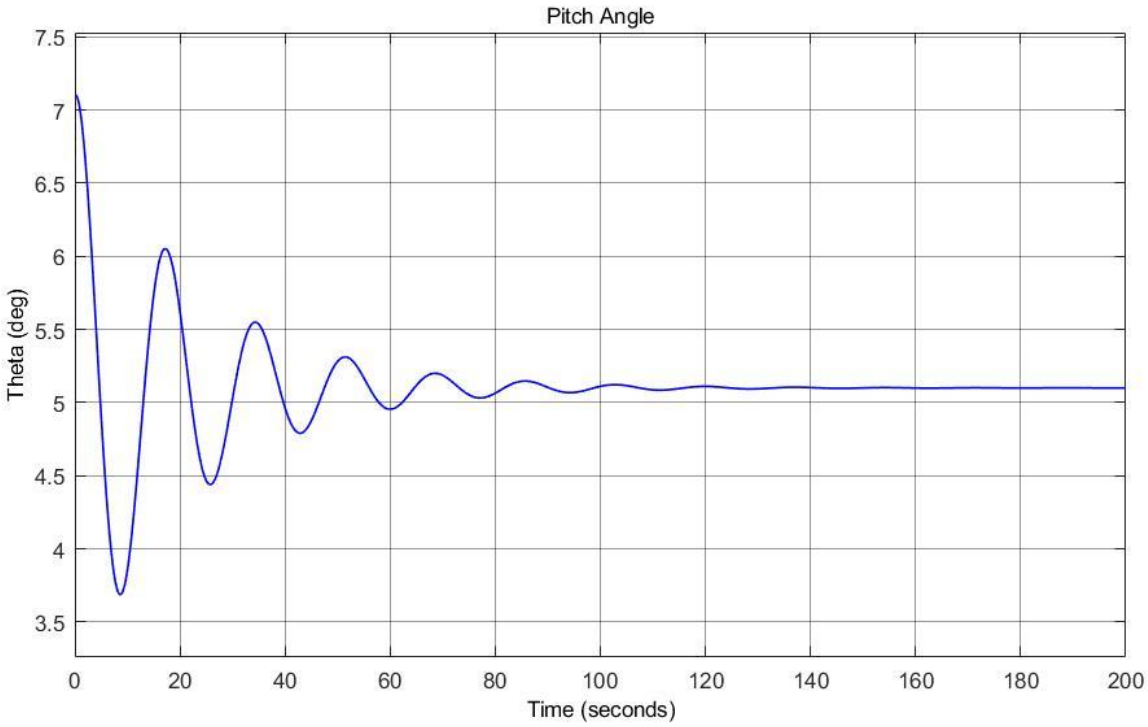


Figure 4.1.13. Pitch Response due to +2 deg Disturbance in Pitch Angle (θ)

Settling time is 87.26 seconds and the overshoot is 27.78%

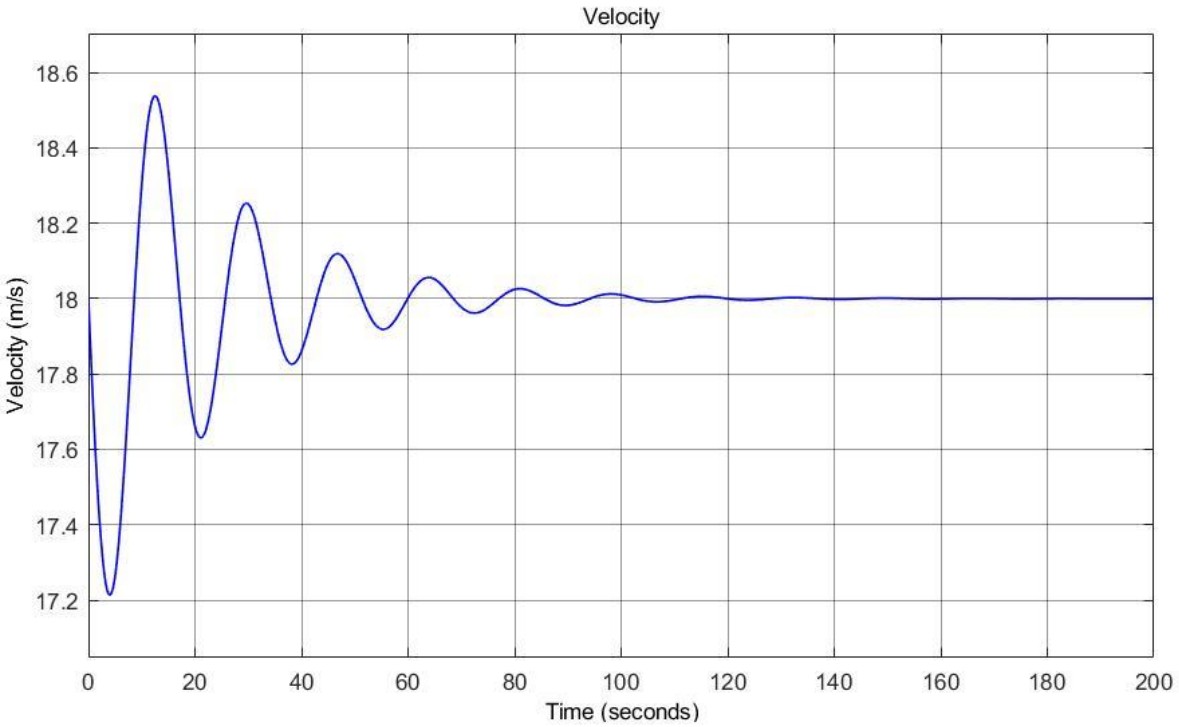


Figure 4.1.14. Velocity Response due to +2 deg Disturbance in Pitch Angle (θ)

Settling time is 91.90 seconds and the overshoot is 4.39%

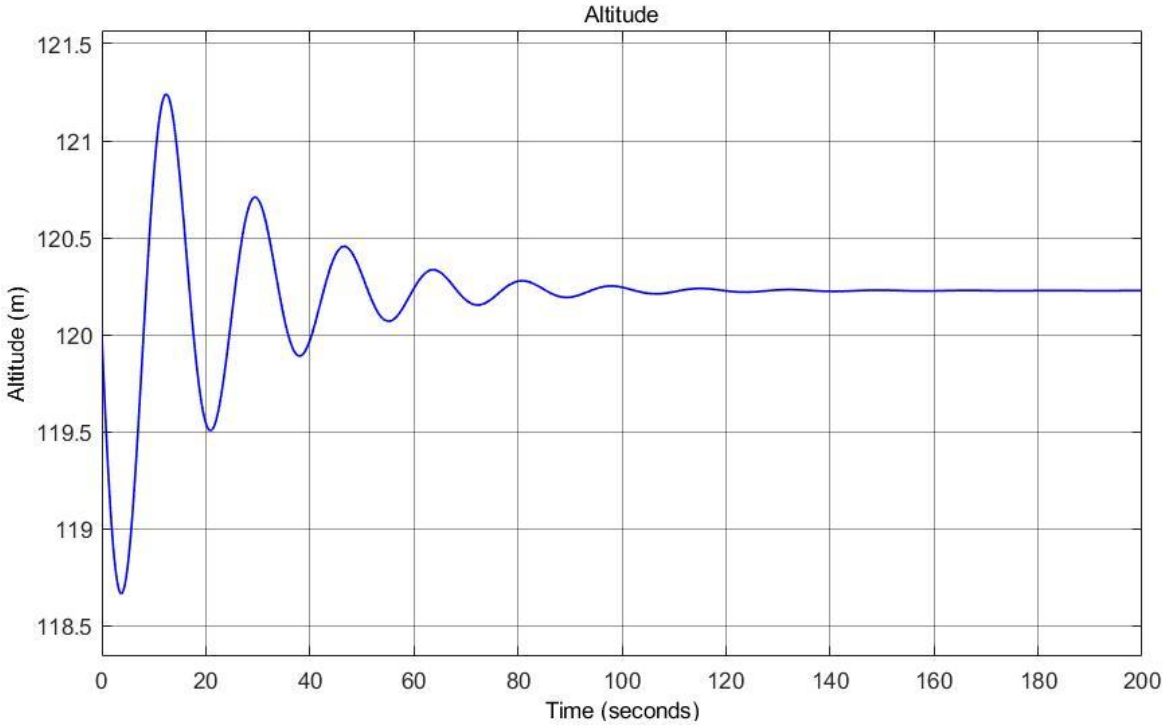


Figure 4.1.15. Altitude Response due to +2 deg Disturbance in Pitch Angle (θ)

Settling time is 90.73 seconds, the overshoot is 1.31%, and the steady state error is 0.229 meters.

4.1.4.1.2 Response due to +1 m/s Disturbance in Velocity

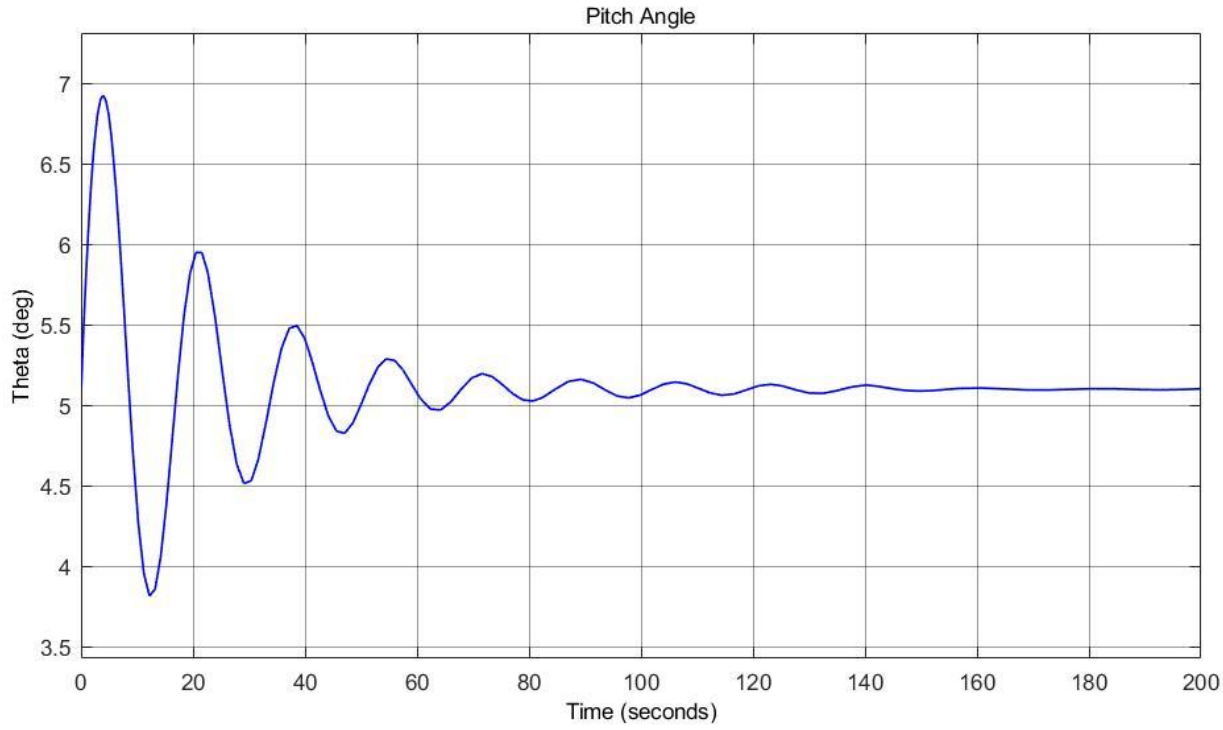


Figure 4.1.16. Pitch Response due to +1 m/s Disturbance in Velocity

Settling time is 91.90 seconds and the overshoot is 35.7%

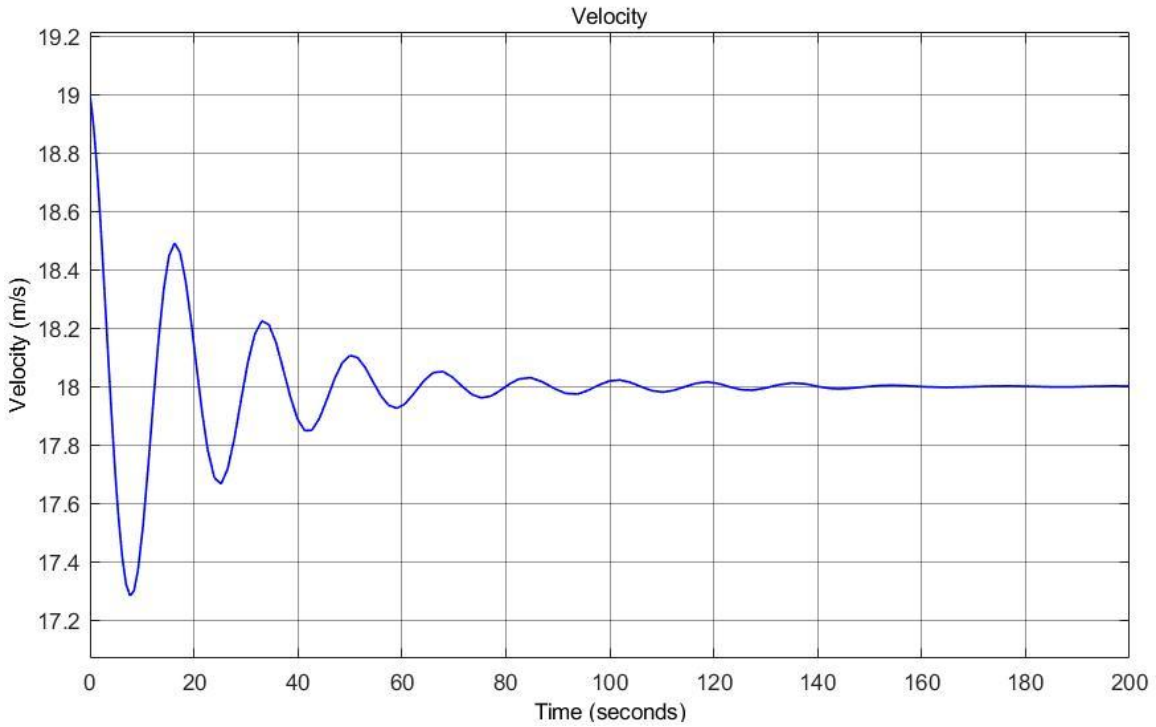


Figure 4.1.17. Velocity Response due to +1 m/s Disturbance in Velocity

Settling time is 86.52 seconds and the overshoot is 3.99%

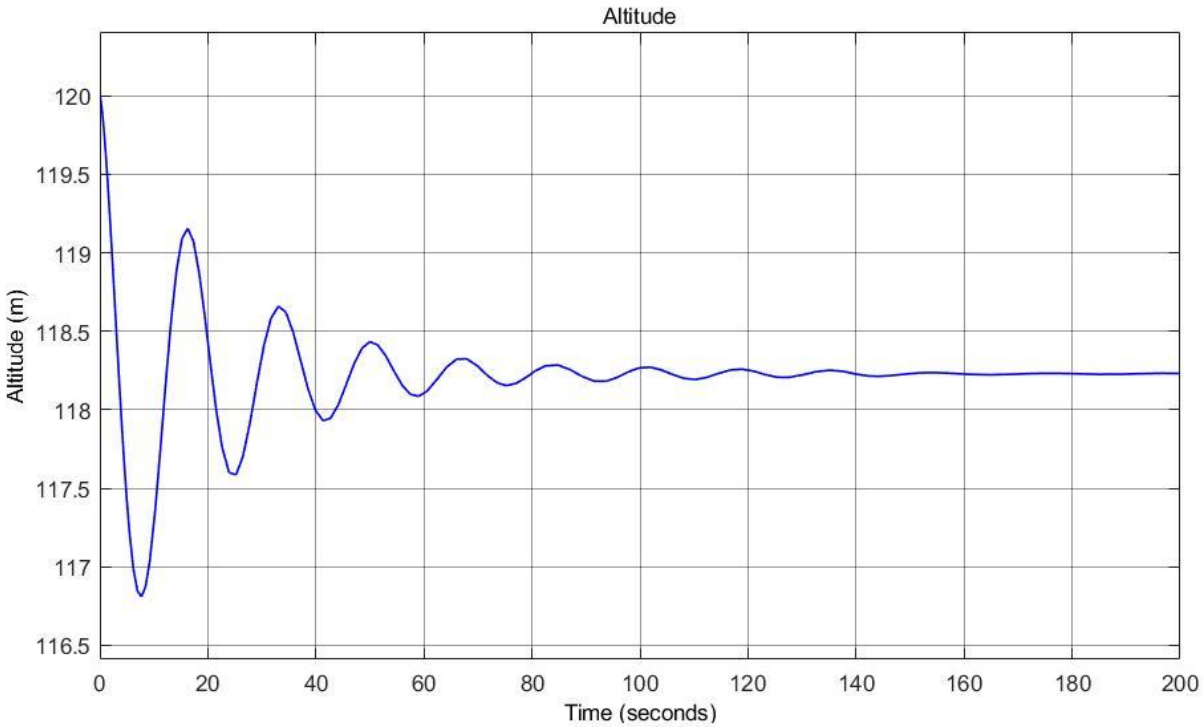


Figure 4.1.18. Altitude Response due to +1 m/s Disturbance in Velocity

Settling time is 86.62 seconds, the overshoot is 2.67%, and the steady state error is 1.77 meters.

4.1.4.2 Lateral Responses

4.1.4.2.1 Response due to +2 deg Disturbance in Roll Angle (ϕ)

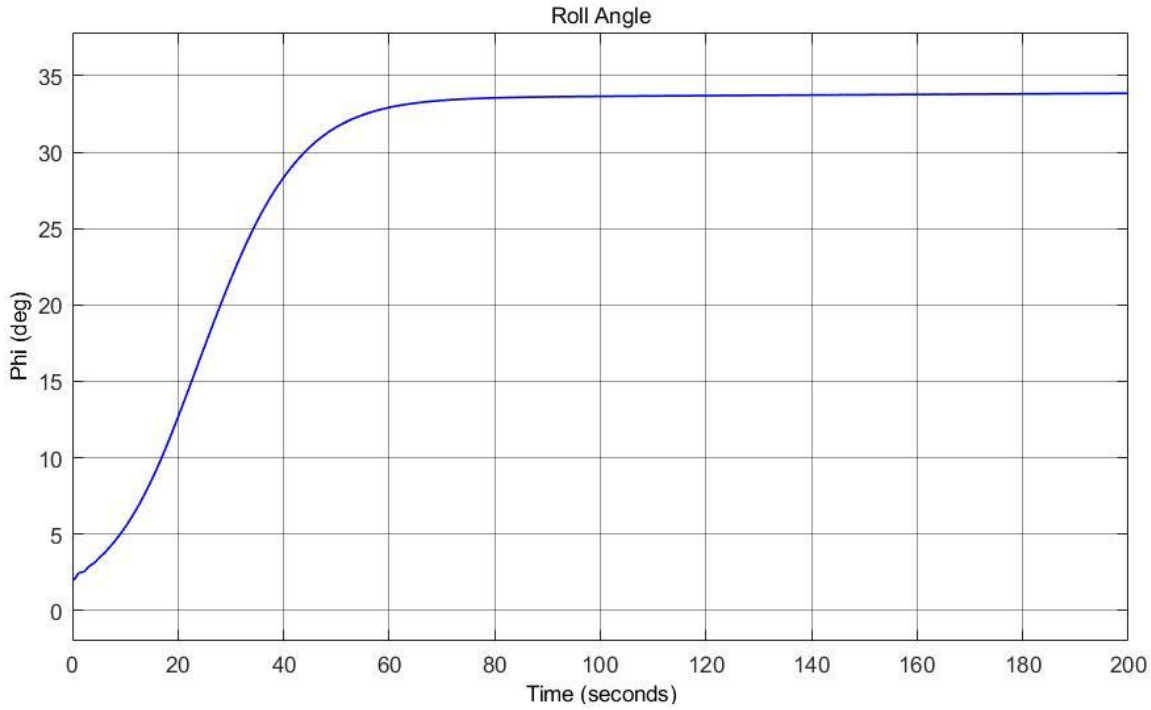


Figure 4.1.19. Roll Response due to +2 deg Disturbance in Roll Angle (ϕ)

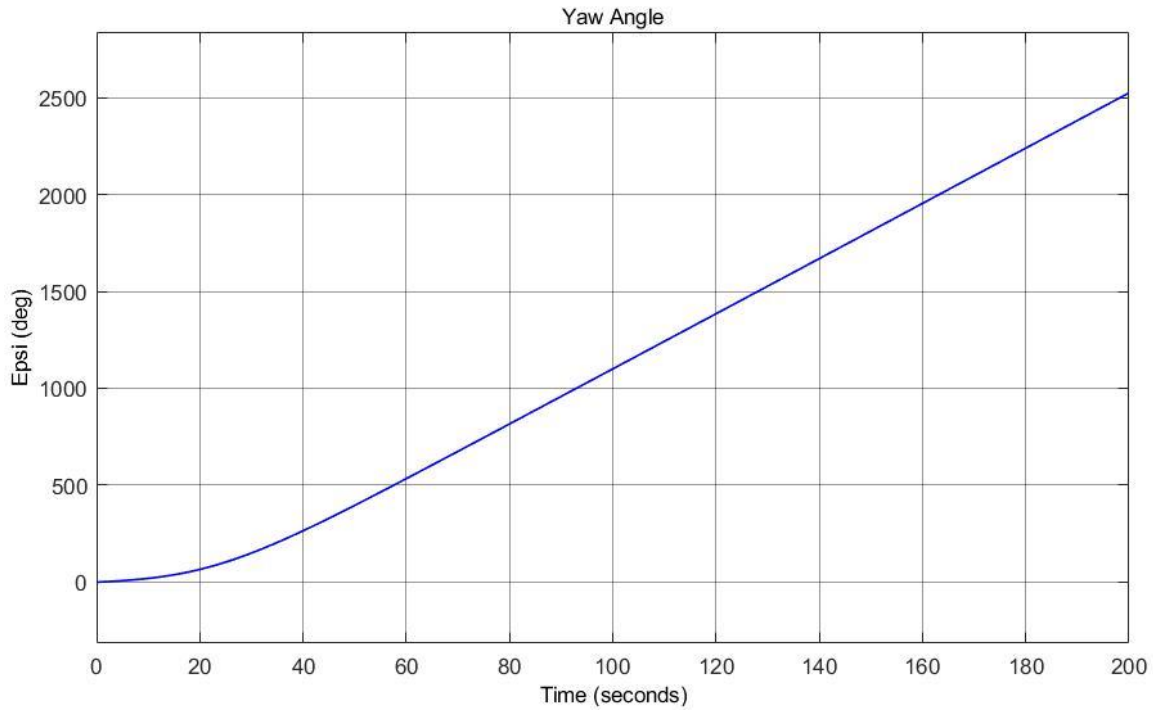


Figure 4.1.20. Yaw Response due to +2 deg Disturbance in Roll Angle (ϕ)

4.1.4.2.2 Response due to +2 deg Disturbance in Yaw Angle (ψ)

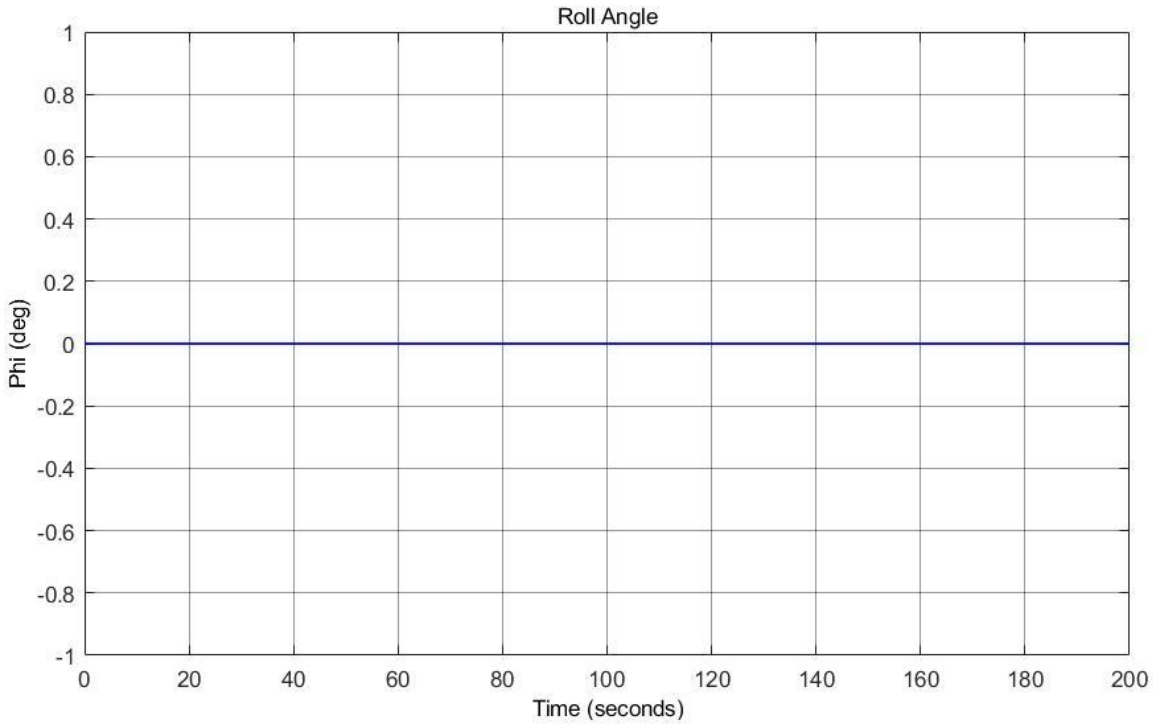


Figure 4.1.21. Roll Response due to +2 deg Disturbance in Yaw Angle (ψ)

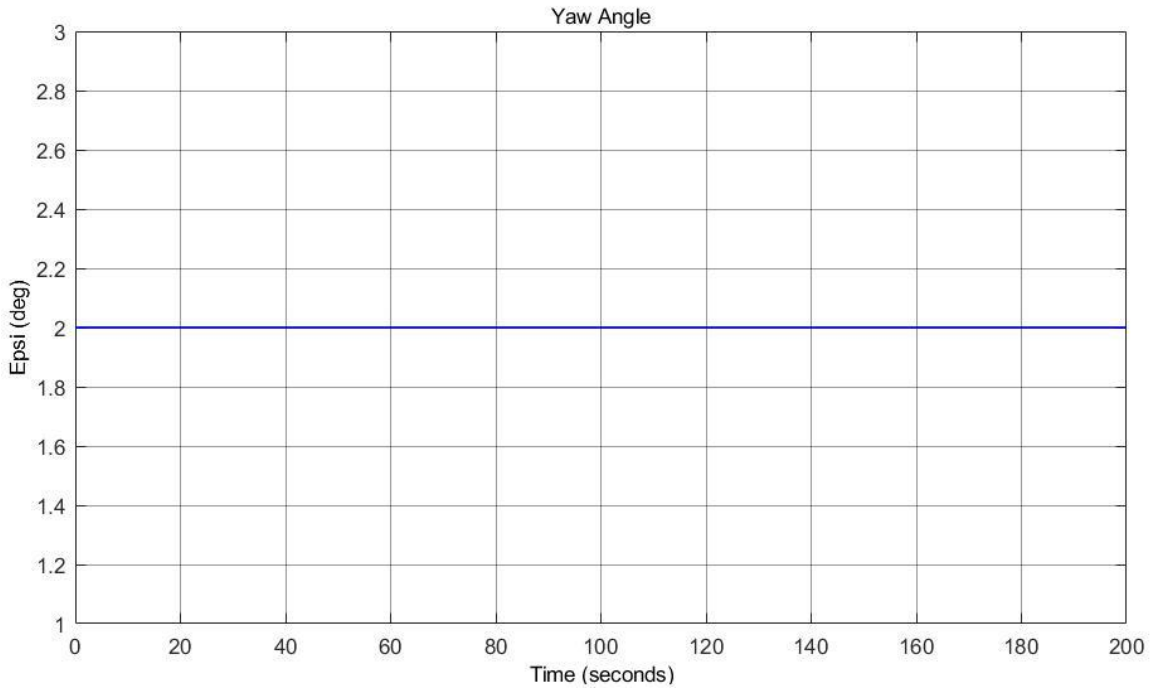


Figure 4.1.22. Yaw Response due to +2 deg Disturbance in Yaw Angle (ψ)

4.1.5 Linear Model

For study of dynamics and control law design, aircraft linear models are necessary. They serve as quick approximations for validating the nonlinear model. Six simultaneous nonlinear second order differential equations can be used to model an aircraft's motion as a rigid body. These equations depict the vehicle's translational motion.

After the nonlinear equations were linearized, the aircraft's properties can be computed, such as its short and long periods, to ensure that it is stable and that the linear model corresponds to the nonlinear model. This may be done by creating a MATLAB programme that accepts inputs for aircraft parameters and then calculates all of the aircraft's characteristics.

To test the stability of the aircraft, a 2^o disturbance in pitch angle is done. In figure 4.1.23, the aircraft stabilizes after around 50 sec as it does in the nonlinear model.

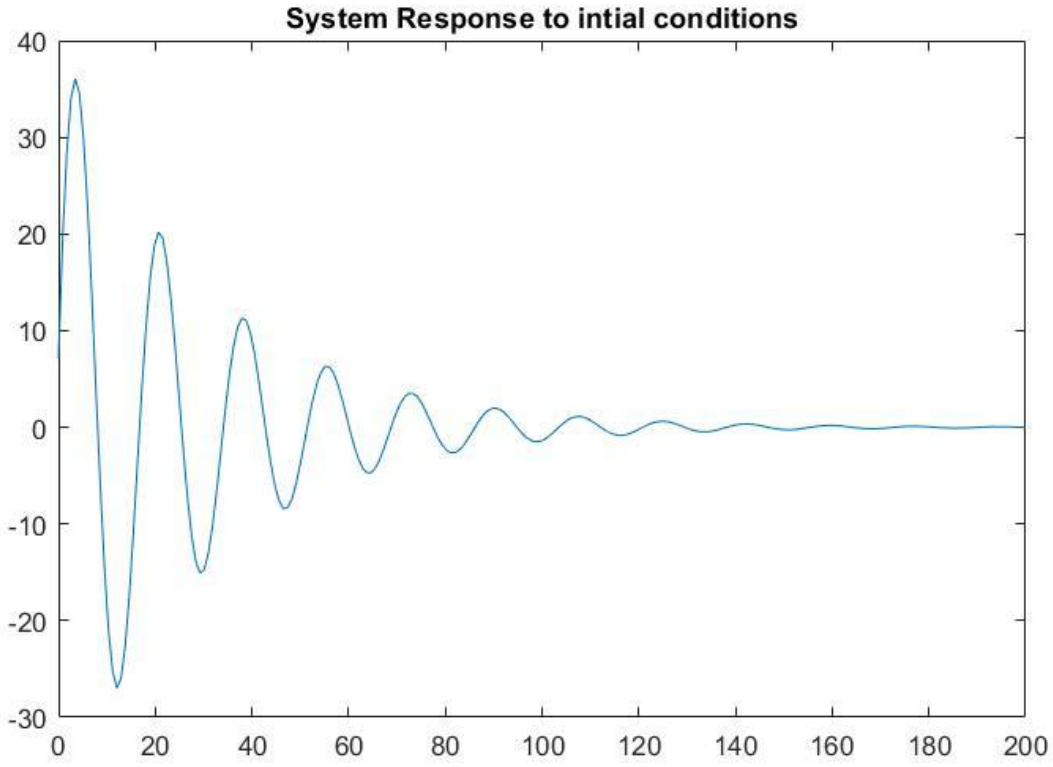


Figure 4.1.23. The Linear Model Response to +2 deg Disturbance in Pitch Angle (θ)

4.2. Quadrotor Model Verification

The model verification section focuses on rigorously examining the quadrotor model's validity and accuracy of representing the real system. This verification process entails conducting simulations that replicate the quadrotor's dynamic behavior across a range of operating conditions, thereby facilitating a comprehensive evaluation of the model's performance. These simulations involve applying control commands and analyzing the resulting outputs, including the quadrotor's position, attitude, and response to disturbances. Essential performance metrics, such as tracking errors, stability, and transient responses, are scrutinized to validate the model's fidelity. The outcomes of this meticulous model verification engender confidence in the accuracy and dependability of the quadrotor model, thus serving as a fundamental basis for subsequent control design and comprehensive performance evaluation in this study.

4.2.1. Quadrotor Elevating, Pitching, Rolling, and Yawing

This section presents an in-depth analysis and evaluation of the quadrotor model's performance in executing specific flight maneuvers. It focuses on testing the model's ability to accurately simulate the quadrotor main four action independently. Namely, simulating pure elevation, pitching while hovering, rolling while hovering, and yawning while hovering motions of the quadrotor. By isolating each maneuver, the effectiveness of the model's algorithms and control strategies in achieving precise and stable flight behaviors can be thoroughly examined.

The quadrotor's response to control inputs for elevating, pitching, rolling, and yawing will be analyzed, comparing the simulated results against expected theoretical behavior. This investigation aims to validate the quadrotor model's capability to accurately represent the physical dynamics of the system independently and provide valuable insights into its performance characteristics for each specific motion.

4.2.1.1. Elevating

In order for the quadrotor to do an elevation only without performing any other actions, the thrust force needs equal the quadrotor's weight.

$$T = k(u_1 + u_2 + u_3 + u_4) = mg$$

Where u_i is the square of the angular speed of the i^{th} motor

$$4u = 24 * 9.81$$

$$u = 58.86\, rpm^2$$

So for the conrol input $u_1 = u_2 = u_3 = u_4 = 58.86\, rpm^2$, the expected output is a zero in all of the 12 states on the model and for any values above $58.86\, rpm^2$, the quadrotor would only elevate in the z direction, which is the case as shown in Fig.4.2.1, and Fig.4.2.2. In Fig.4.2.2 thr z postion starts to increase indecating that the quadrotor is not vertically taking off while doing no other action as indicated by the zero values for all the other states except for the z velocity conpentnts that states the taking off speed.

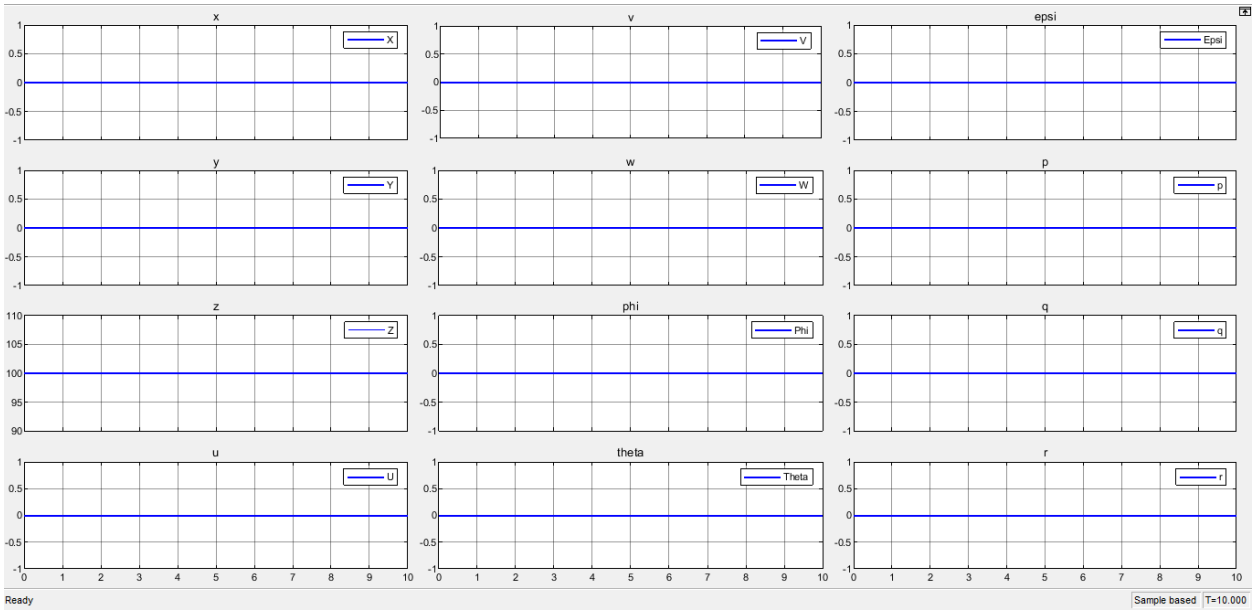


Fig.4.2.1. The model response for the hover speed at z = 100 m.

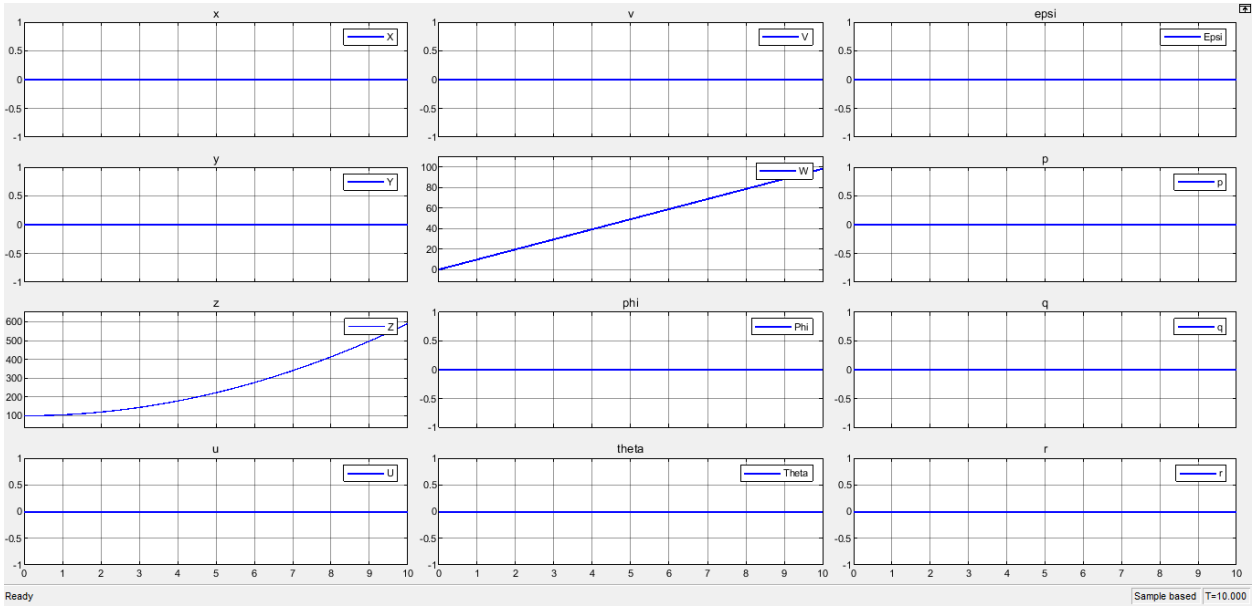


Fig.4.2.2. The model response for speed double the hover.

4.2.1.1. Pitching While Hovering

In order for the quadrotor to do a pitching while hovering only without performing any other actions, the thrust force needs to equal the quadrotor's weight, and motors 1 and 2 to have the same value that is slightly higher than that of motors 1 and 2. If motors 1 and 2 have a higher value the quadrotor will pitch up and move in the negative x direction and the theta value would increase, as indicated in fig.4.2.3. If motors 1 and 2 have a lower value the quadrotor will pitch

down and move in the positive x direction and the theta value would decrease, as indicated in Fig.4.2.4.

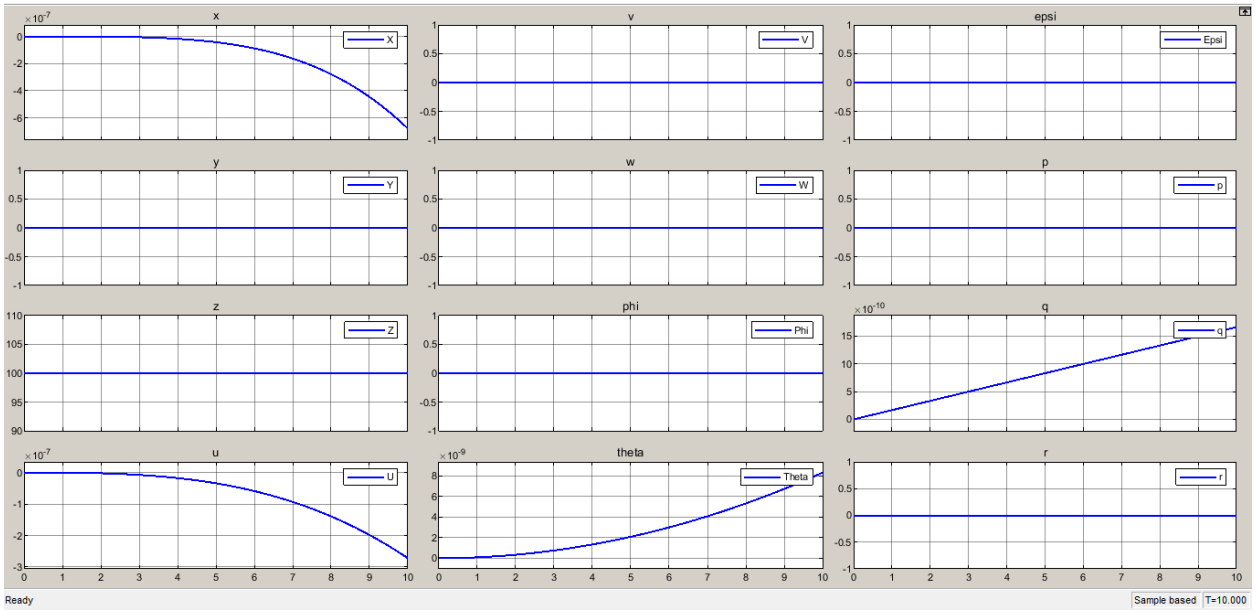


Fig.4.2.3. The model response for the pitch up while hovering at 100 m.

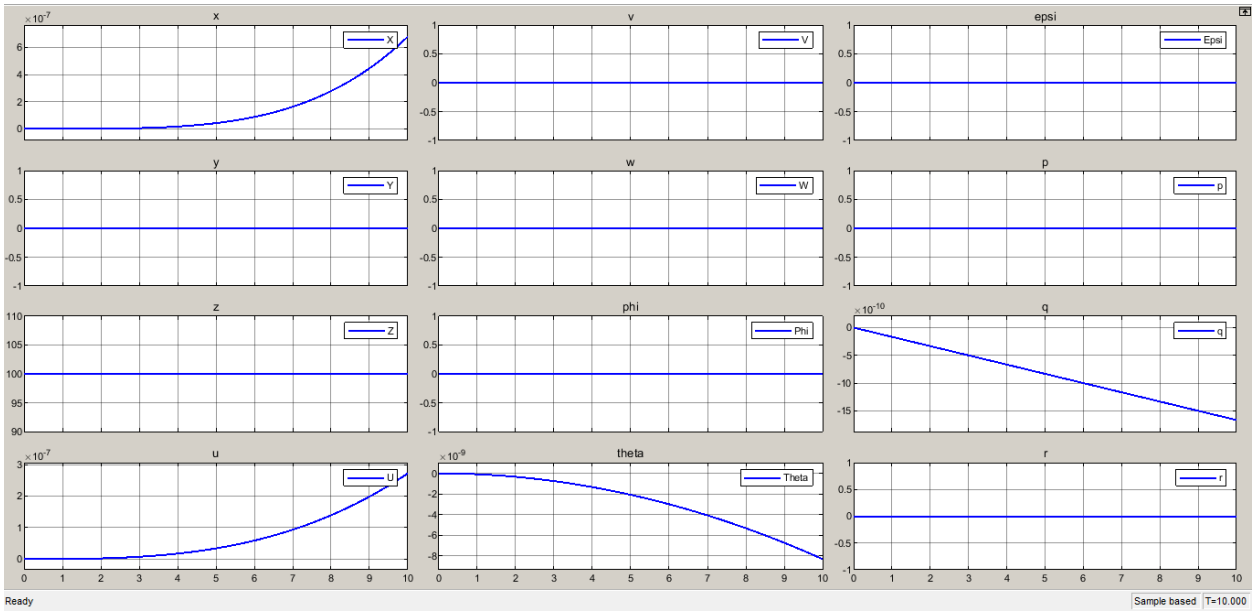


Fig.4.2.4. The model response for the pitching down while hovering at 100 m.

For the speeds needed to achieve pitching,

$$T = k(2u_1 + 2u_2) = mg$$

$$Assume\ u_1 = 60\ rpm^2$$

$$u_2 = 57.72\ rpm^2$$

Inputing motor 1 and 2 with $60\ rpm^2$ and motor 3 and 4 with $57.72\ rpm^2$ would pitch up, and vise versa.

4.2.1.1. Rolling While Hovering

In order for the quadrotor to do a rolling while hovering only without performing any other actions, the thrust force needs to equal the quadrotor's weight, and motors 2 and 4 to have the same value that is slightly higher than that of motors 1 and 3. If motors 1 and 3 have a higher value the quadrotor will roll up and move in the positive y direction and the theta value would increase, as indicated in fig.4.2.5, and vise versa, as indicated in Fig.4.2.6.

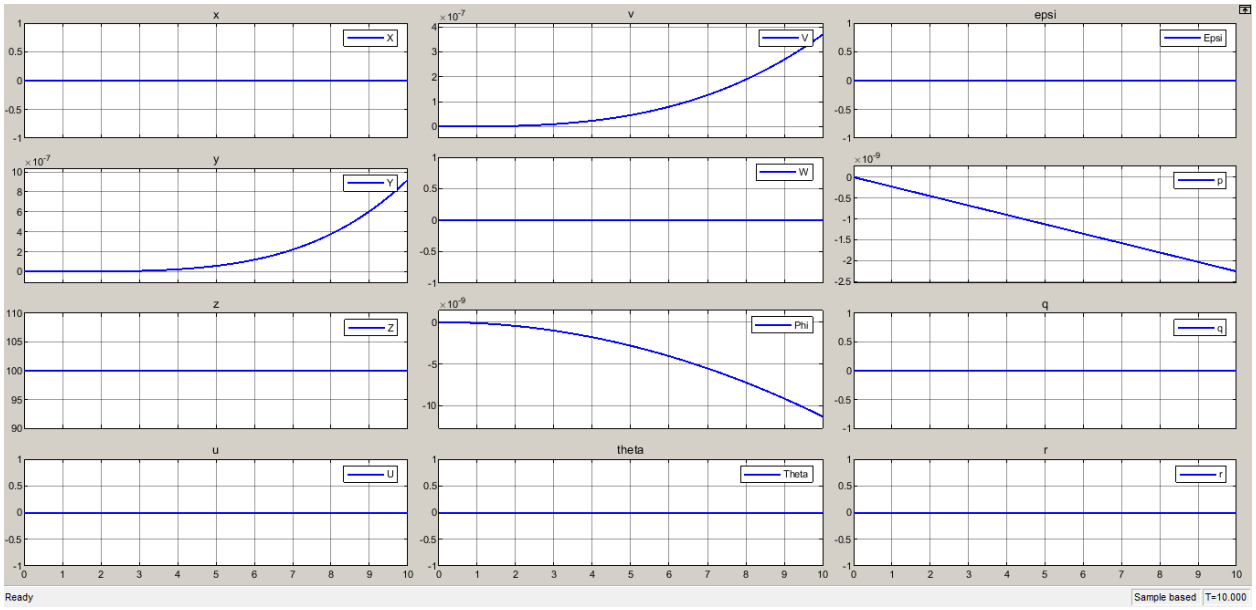


Fig.4.2.5. The model response for the rolldown while hovering at 100 m.

Inputing motor 1 and 3 with $60\ rpm^2$ and motor 2 and 4 with $57.72\ rpm^2$ would roll up, and vise versa.

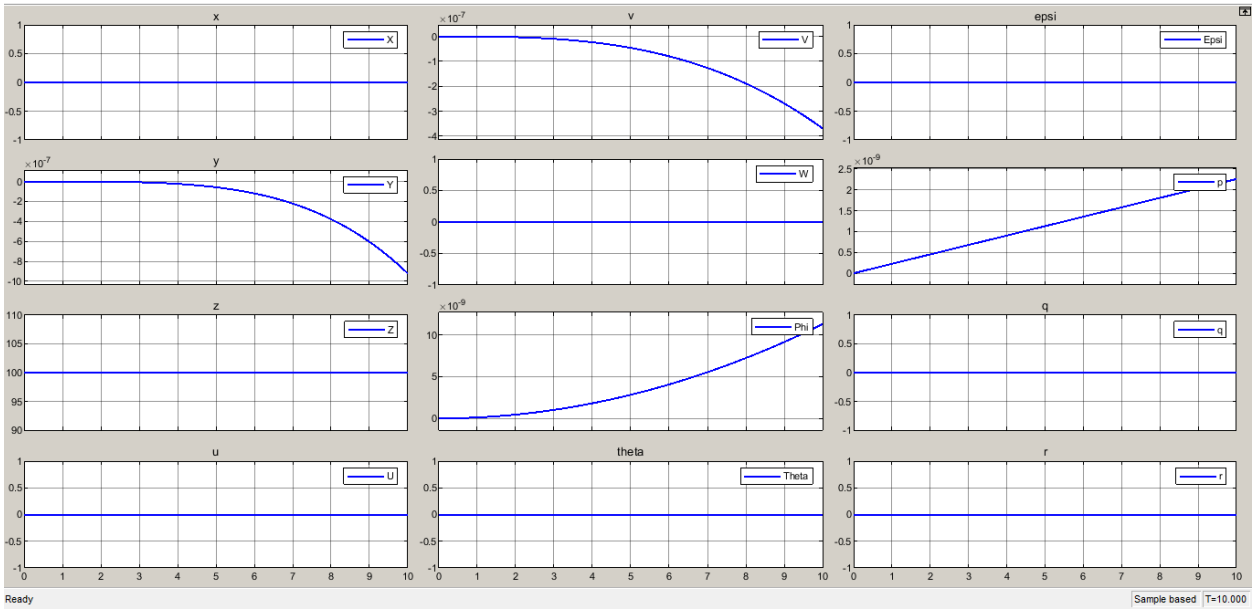


Fig.4.2.6. The model response for the roll down while hovering at 100 m.

4.2.1.1. Yawing While Hovering

In order for the quadrotor to do a roll while hovering only without performing any other actions, the thrust force needs to equal the quadrotor's weight, and opposite motors to have the same speed but differ from the other two. This means motors 1 and 3 have the same value that is slightly higher than that of motors 2 and 4. If motors 1 and 3 have a higher value the quadrotor will positively yaw in place without moving and the epsi angle value would increase, as indicated in fig.4.2.7, and vice versa, as indicated in Fig.4.2.8.

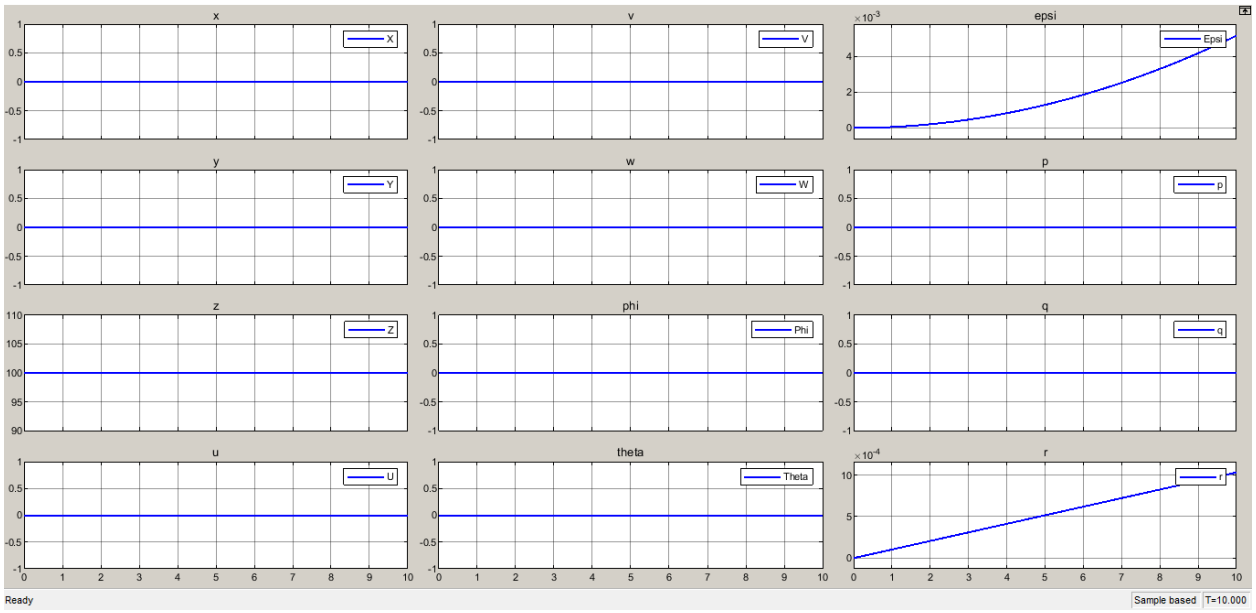


Fig.4.2.7. The model response for positively yawing while hovering at 100 m.

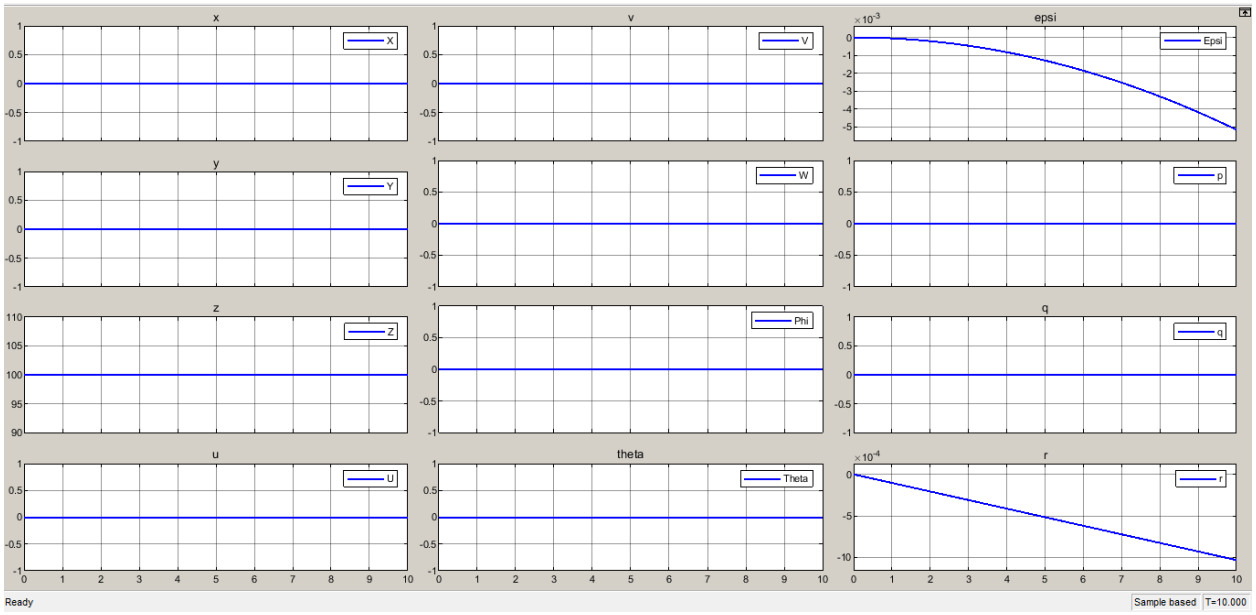


Fig.4.2.8. The model response for the positive yawing while hovering at 100 m.

4.2.2. Quadrotor Disturbances Response

This section delves into the analysis and evaluation of the quadrotor model's response when subjected to small disturbances in the angles phi, theta, and epsi. It aims to investigate whether the quadrotor is able to maintain stability and recover from disturbances that perturb its orientation or not. Disturbances in phi (roll angle), theta (pitch angle), and epsi (yaw angle) can

be caused by external factors such as wind gusts or sudden changes in payload. The quadrotor model's response to these disturbances will be carefully examined through simulations, where the disturbance inputs are applied, and the resulting behavior is analyzed. The simulated responses will be compared against theoretical expectations to study the model's nature and capture the quadrotor's dynamic behavior under disturbance conditions. The findings from this investigation will provide valuable information and insight into the quadrotor's control strategies and developing mechanisms to mitigate the effects of disturbances in real-world applications.

4.2.2.1. Disturbances in Phi

Introducing a 10 deg disturbance in the phi angle while hovering and observing the model response. As shown in Fig.4.2.9, the quadrotor loses altitude and starts falling. The quadrotor could not recover and hence the model is not stable for small angel disturbances in phi.

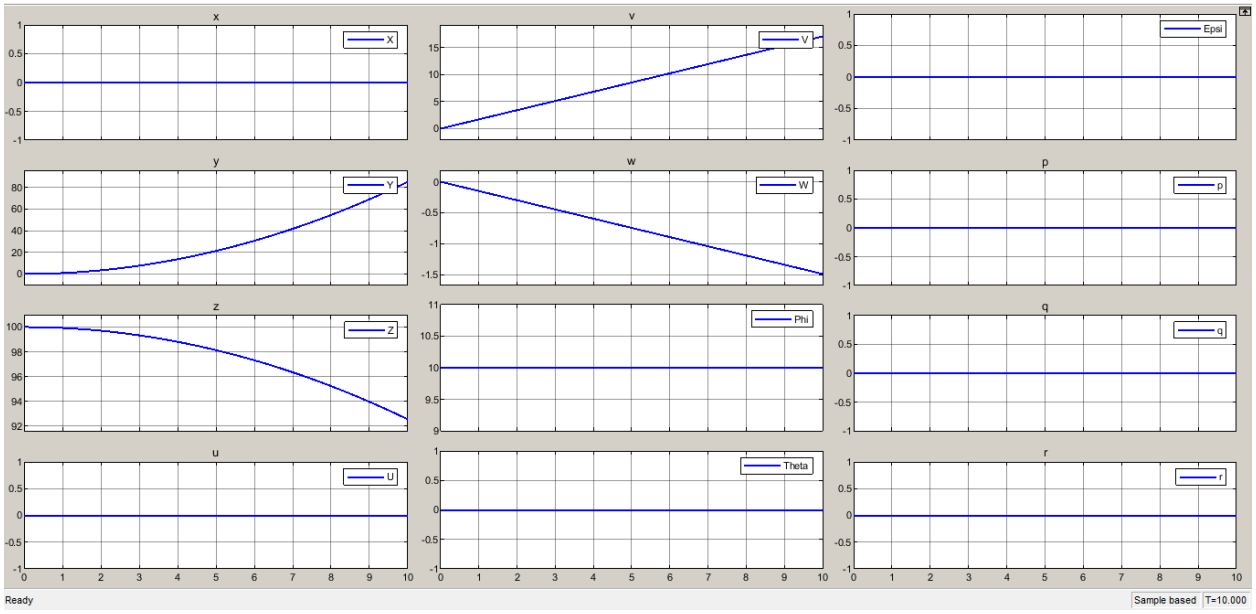


Fig.4.2.9. The model response for a 10 deg disturbance in the phi angle while hovering at 100 m.

4.2.2.1. Disturbances in Theta

Introducing a 10 deg disturbance in the theta angle while hovering and observing the model response. As shown in Fig.4.2.10, the quadrotor loses altitude and starts falling. The quadrotor could not recover and hence the model is not stable for small angel disturbances in theta.

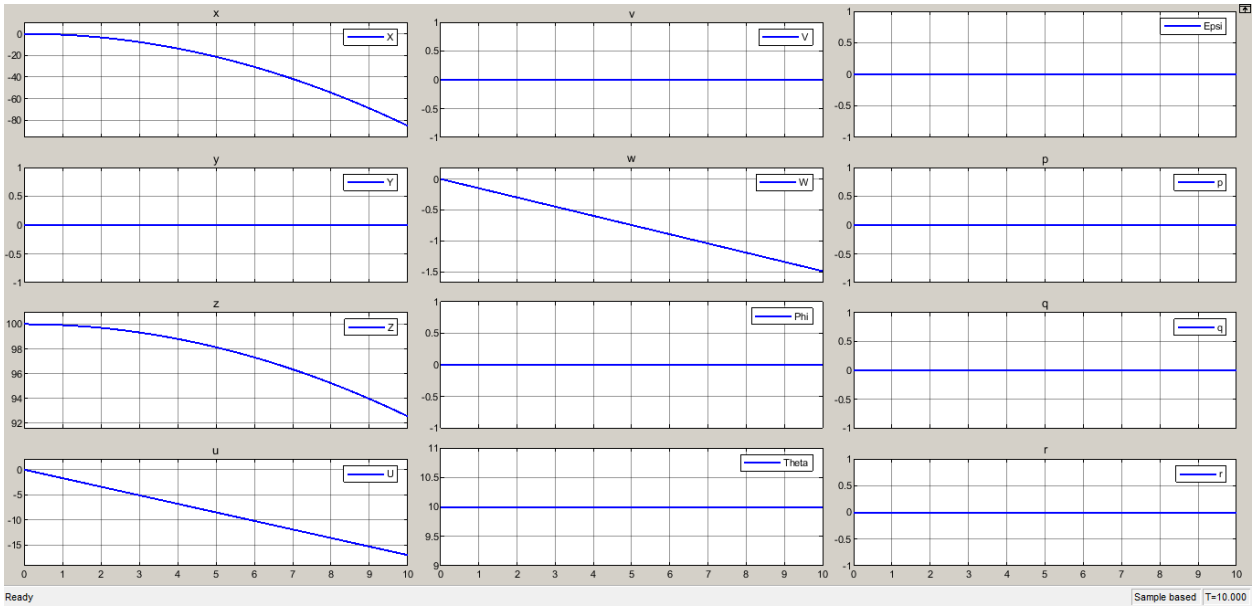


Fig.4.2.10. The model response for a 10 deg disturbance in the theta angle while hovering at 100 m.

4.2.2.1. Disturbances in Epsi

Introducing a 10 deg disturbance in the epsi angle while hovering and observing the model response. As shown in Fig.4.2.11, the quadrotor stays in place with the epsi angle it started. Hence, the quadrotor is stable for small angel disturbances in epsi.

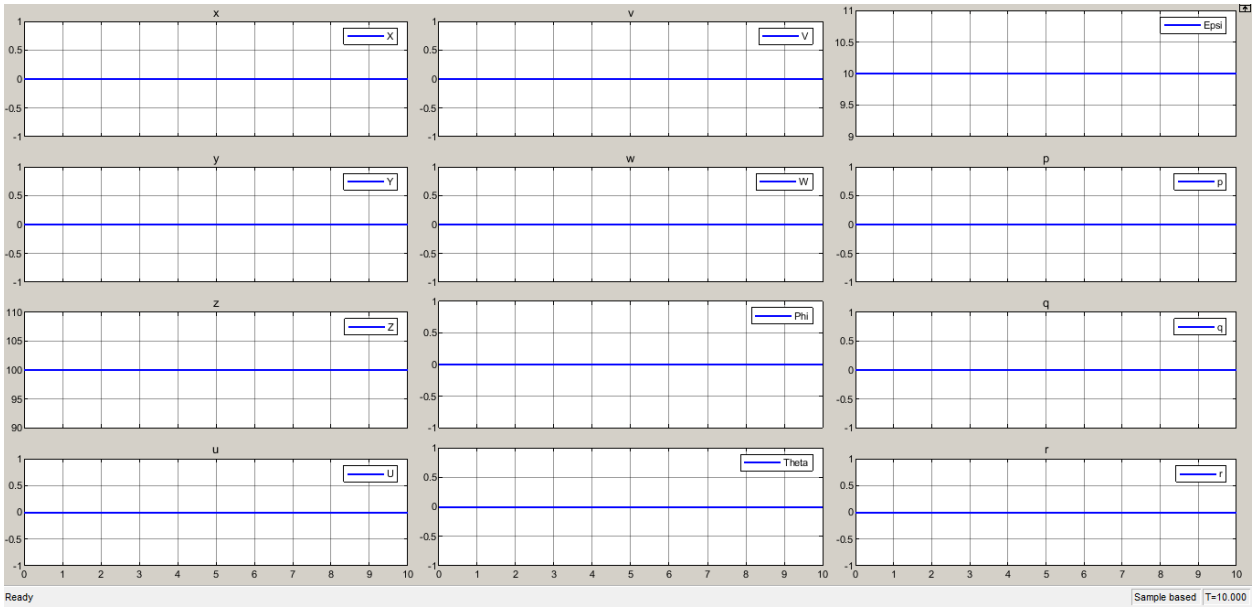


Fig.4.2.11. The model response for a 10 deg disturbance in the epsi angle while hovering at 100 m.

Chapter 5: Autopilot Controller Design and Implementation

This chapter presents the design and implementation of autopilot controllers for both the fixed-wing and quadrotor parts of this study's VTOL aircraft. It focuses on the development of controllers that enable autonomous flight and precise control of the VTOL during various operational modes.

For the fixed-wing component, multiple controllers have been designed and implemented for different flight modes. The controllers include a Linear Quadratic Regulator (LQR), Total Energy Control System (TECS), Model Predictive Control (MPC), and fuzzy logic controller. Each controller addresses specific requirements such as stability, energy management, and disturbances. These controllers are evaluated in terms of their performance, responsiveness, and ability to maintain desired flight characteristics during different mission profiles.

In addition to the fixed-wing controllers, the quadrotor part of the VTOL is equipped with two distinct controllers. The first controller is a Nonlinear Model Predictive Control (NMPC) that provides optimal control and trajectory tracking for the quadrotor. The NMPC accounts for the nonlinear dynamics and constraints of the quadrotor, enabling precise maneuvering and high-level autonomy. The second controller is a cascaded Proportional Integral Derivative (PID) controller, which offers a simpler but effective control strategy for stability and attitude control of the quadrotor.

The design and implementation of these autopilot controllers involve mathematical modeling, control algorithm development, simulation, and integration with the VTOL platform. The performance of each controller is rigorously evaluated through extensive simulations and various scenarios. The outcomes of this chapter provide insights into the effectiveness and applicability of the designed controllers for the autonomous and robust operation of the VTOL aircraft.

5.1. Fixed Wing Controllers

Different controllers have been designed,

5.1.1. Linear Quadratic Control LQR

In the LQR problem, a linear system with a quadratic cost function is used as a regulator in a LTV system. Finding the control input u_{LQR} that minimizes the performance index is the design target. Where K_{LQR} is the best feedback gain matrix.

The performance index J_{LQR} represents the performance characteristic requirement as well as the controller input limitation (Mohammed & Adulla, 2018). In LQR, the quadratic performance index is expressed in terms of weighting matrices Q and R as:

$$J_{LQR} = \int_0^\infty (x^T Q x + u^T R u)\, dt$$

where the matrix Q must be positive definite or positive semi-definite symmetry matrix, and R must be positive-definite Hermitian or real symmetric matrices (Nagarkar & Patil, 2016). A practical approach in tuning the weighting matrices is to begin by an identity matrix for each (Anjali et al., 2016).

The difference between the initial and target states is represented by the first term on the right-hand side of the last equation, while the energy used by the control signal is represented by the second term. The matrices Q and R establish the error's and the performance index's relative importance.

Gain matrix K_{LQR}, which minimizes J_{LQR} is

$$K_{LQR} = R^{-1} B^T P$$

Matrix P is evaluated by solving the Algebraic Riccati Equation

$$AP + A^T P - PBR^{-1} B^T P + Q = 0$$

In this project, Q and R were determined by trial and error.

In order to eliminate the steady state error, a feed-forward integral gain is added to the system. This is done by adding p more states to the original states. These states will be referred to by ξ_{state} and they present the error integral of the prospective state.

$$\dot{\xi}_p = C_p x - r_p$$

Hence the augmented system is in the form of

$$\begin{bmatrix} \dot{x} \\ \dot{\xi}_1 \\ \vdots \\ \dot{\xi}_p \end{bmatrix}_{(n+p)\times 1} = \begin{bmatrix} Ax + Bu \\ y_1 - r_1 \\ \vdots \\ y_p - r_p \end{bmatrix} = \begin{bmatrix} (Ax + Bu)_{n\times 1} \\ (Cx - r)_{p\times 1} \end{bmatrix}$$

Where p represents the number of states that are targeted to have zero steady-state error. The number should be less than or equal number of actuators available (i,e $p \leq m$).

Finally, the final compensator gain is in the form:

$u = - K_p x - K_I \xi$

5.1.1.1. Simulink Implementation

In this section the LQR implementation is reviewed in detail and its response. The LQR gains are calculated via MATLAB. The code is in appendix B. A saturation for the control demand has been implemented for the elevator, aileron, and rudder deflection angles to be from -15^o to 30^o. Another saturation was done for the thrust to have a positive value only (from 0 to ∞).

5.1.1.1 Longitudinal Control

LQR tuning was done manually. The following values are the best results to get a stable nonlinear response as well as minimum overshoot and settling time. Decreasing the ξ gain in the Q matrix results in longer settling time, but increasing it too much can result in high fluctuations and overshoot (OS) that might lead to instability. As θ is the integral of q, the later will always settle with zero with no steady state error. Increasing the net value of Q increases the OS, and the opposite with decreasing it. Increasing the net value of R, causes an increase in fluctuations.

$Q =$

10	*0*	*0*	*0*	*0*	*0*	*0*
0	*1*	*0*	*0*	*0*	*0*	*0*
0	*0*	*0.00001*		*0*	*0*	*0*
0	*0*	*0*	*100*	*0*	*0*	*0*
0	*0*	*0*	*0*	*1*	*0*	*0*
0	*0*	*0*	*0*	*0*	*1000*	*0*
0	*0*	*0*	*0*	*0*	*0*	*0.5*

$R =$

$$\begin{matrix} 0.01 & 0 \\ 0 & 0.01 \end{matrix}$$

$K =$

$$\begin{matrix} 0.0701 & -2.0443 & -2.0829 & -171.0664 & -15.2656 & 4.6298 & -7.0703 \\ 31.7301 & -0.0877 & 0.0109 & 1.5850 & 0.2055 & 316.1939 & 0.1035 \end{matrix}$$

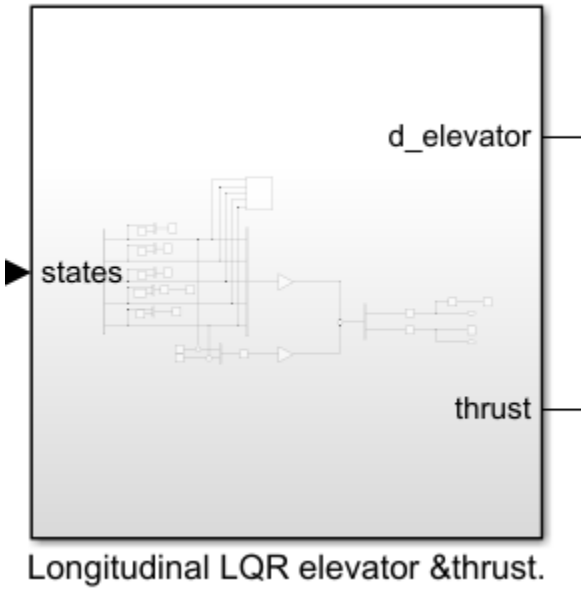


Figure 5.1.1. Longitudinal LQR Block

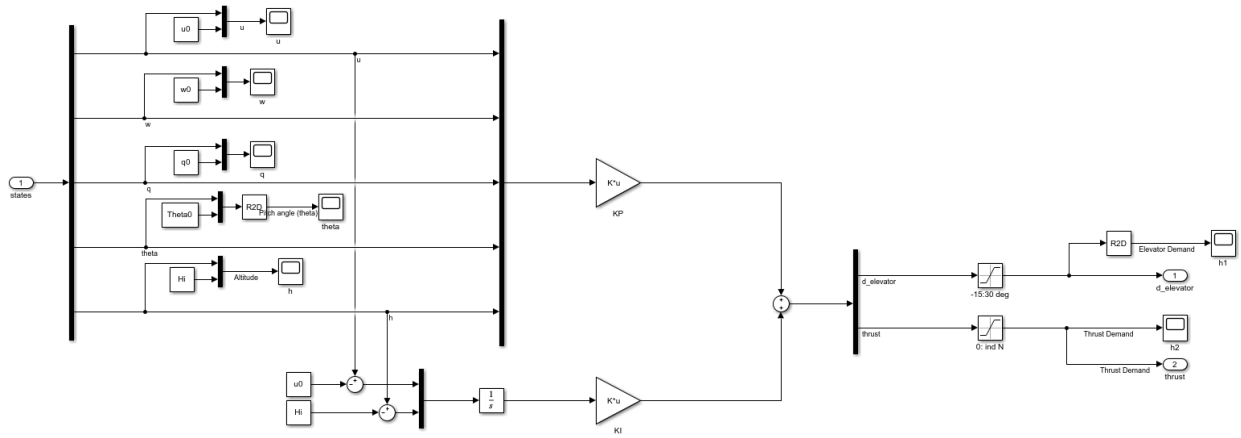


Figure 5.1.2. Longitudinal LQR Block Components

5.1.1.2. Lateral Control

$Q=$

$$\begin{matrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

$R=$

$$\begin{matrix} 100 & 0 \\ 0 & 100 \end{matrix}$$

$K=$

$$\begin{matrix} 0.0170 & 0.2767 & -0.0374 & 0.4365 & -0.0018 & 0.1000 \\ -0.0163 & 0.0170 & -0.1430 & 0.0963 & 0.1000 & 0.0018 \end{matrix}$$

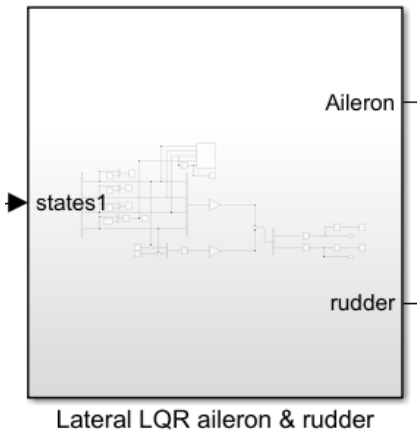


Figure 5.1.1.3. *Lateral LQR Aileron & Rudder Simulink Block.*

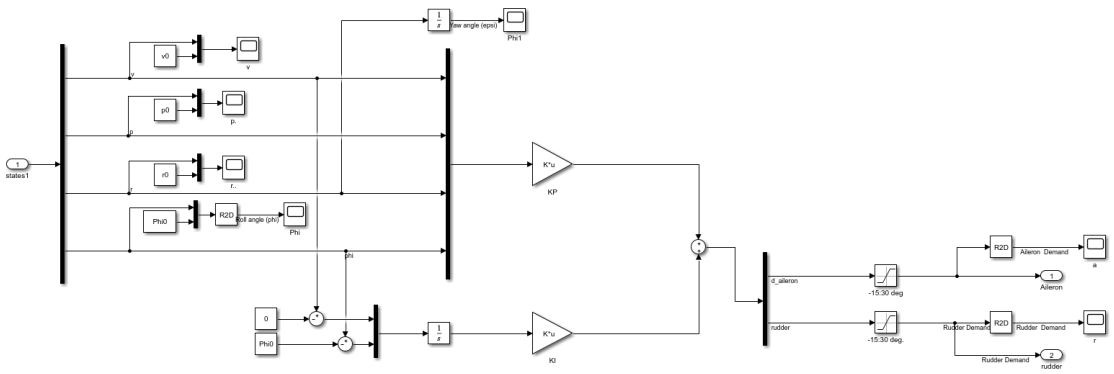


Figure 5.1.1.4. *Lateral LQR Aileron & Rudder Simulink Block.*

5.1.1.3. Longitudinal Response

[1] Response due to +2 deg Disturbance in Pitch Angle (θ)

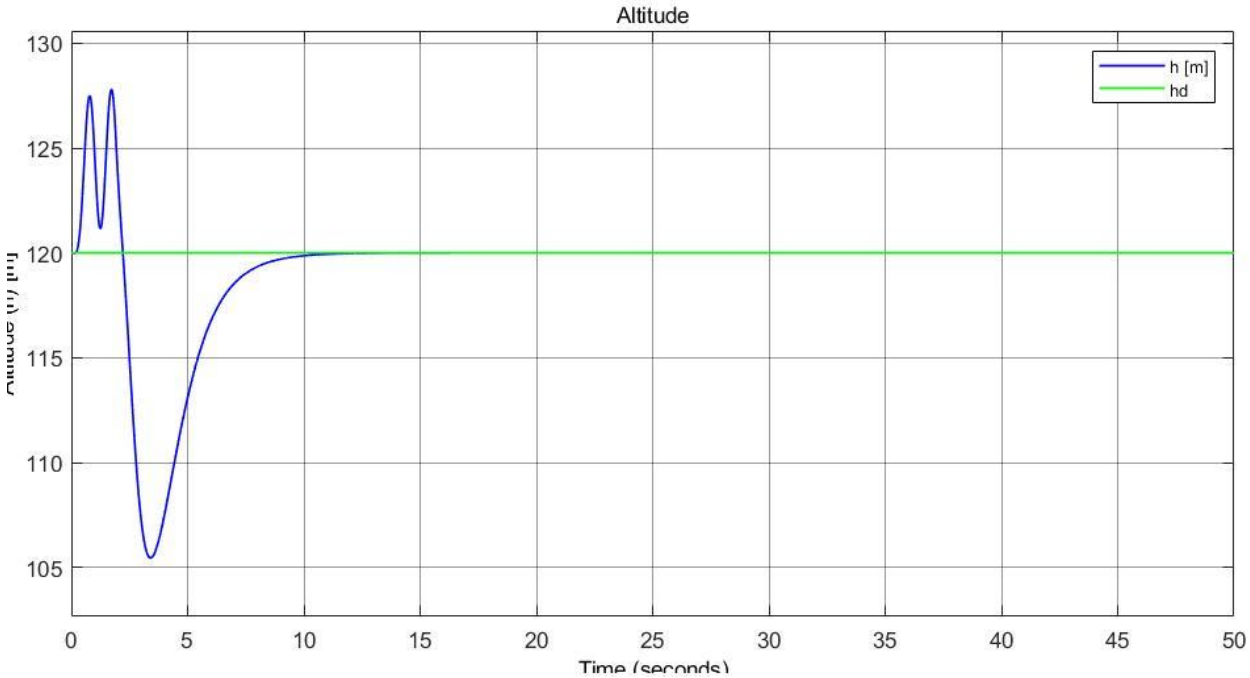


Figure 5.1.1.5. *Altitude Response.*

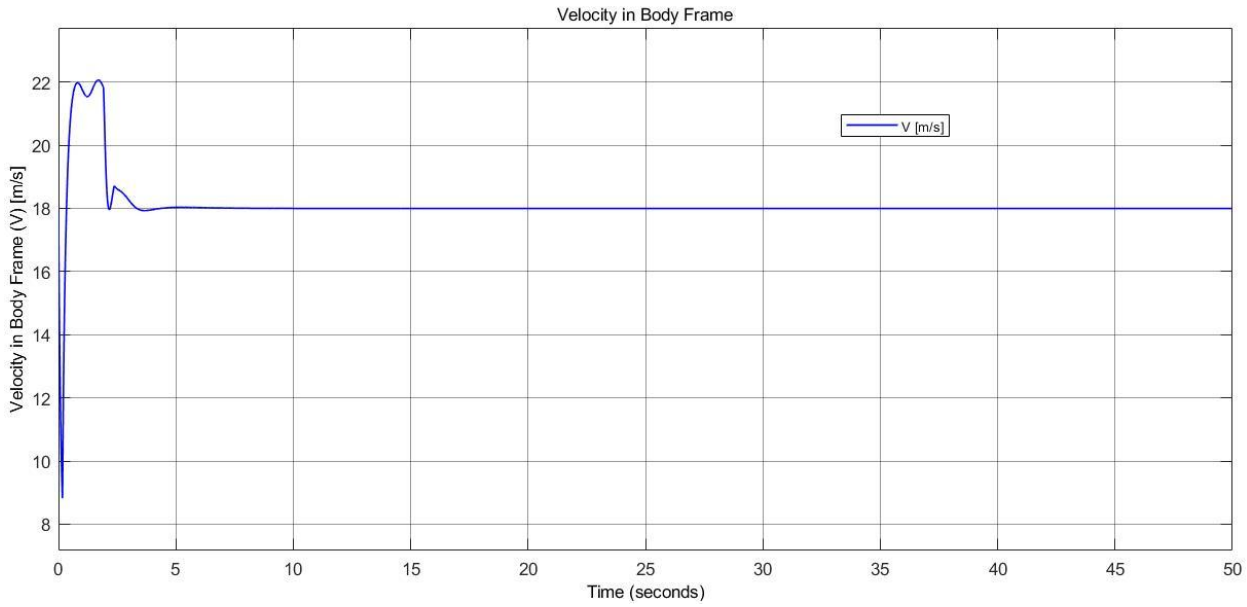


Figure 5.1.1.6. *Velocity in Body Frame.*

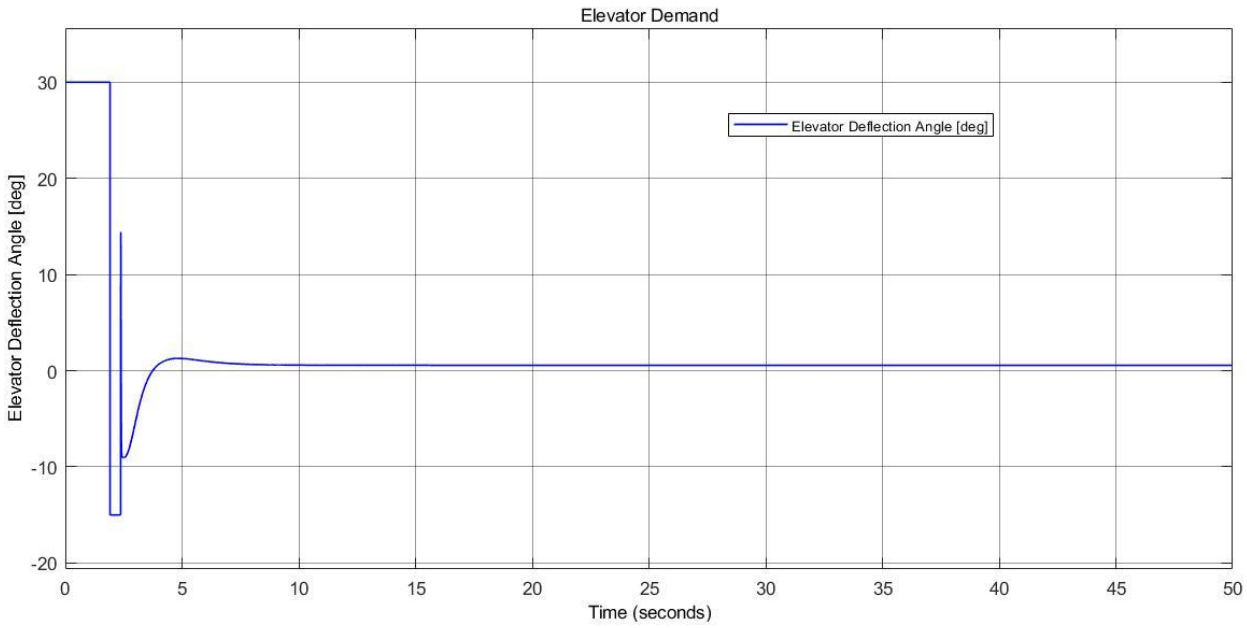


Figure 5.1.1.7. *Elevator Demand.*

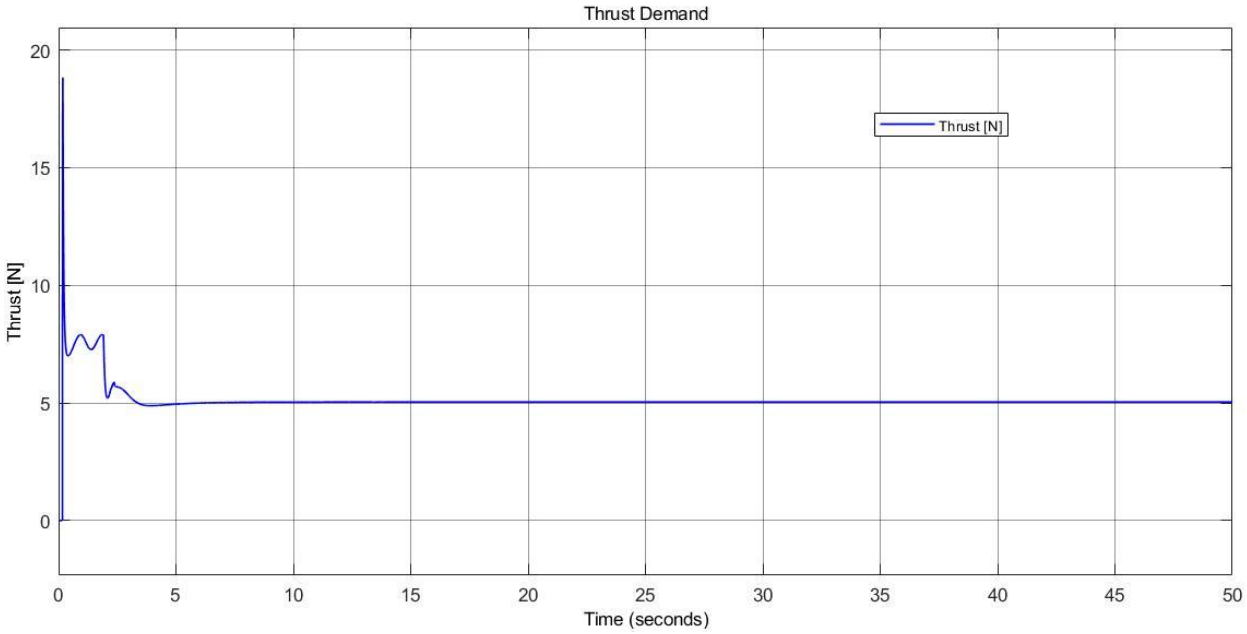


Figure 5.1.1.8. *Thrust Demand.*

[2] Response due to +1 m/s Disturbance in Velocity

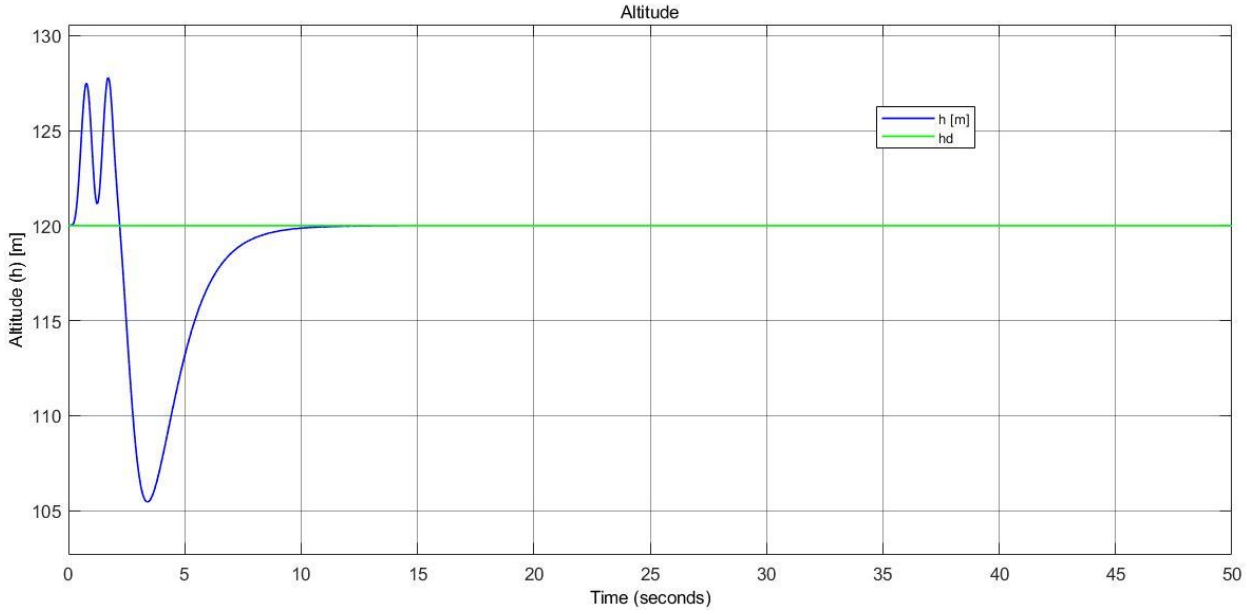


Figure 5.1.1.9. *Altitude Respnse.*

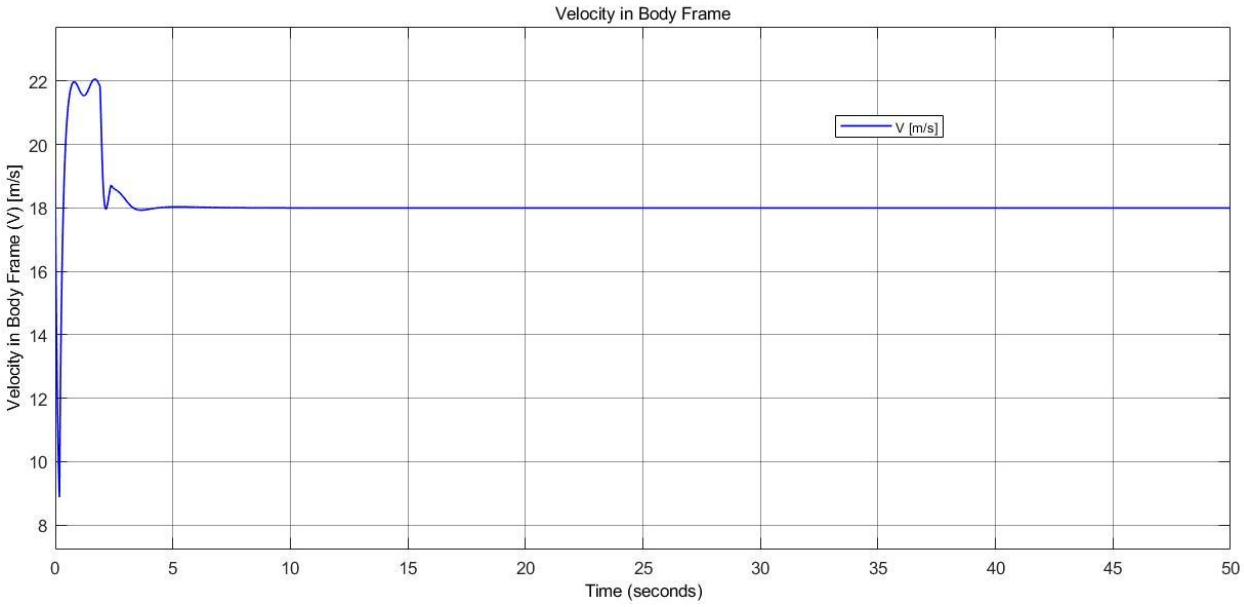


Figure 5.1.1.10. *Velocity in Body Frame.*

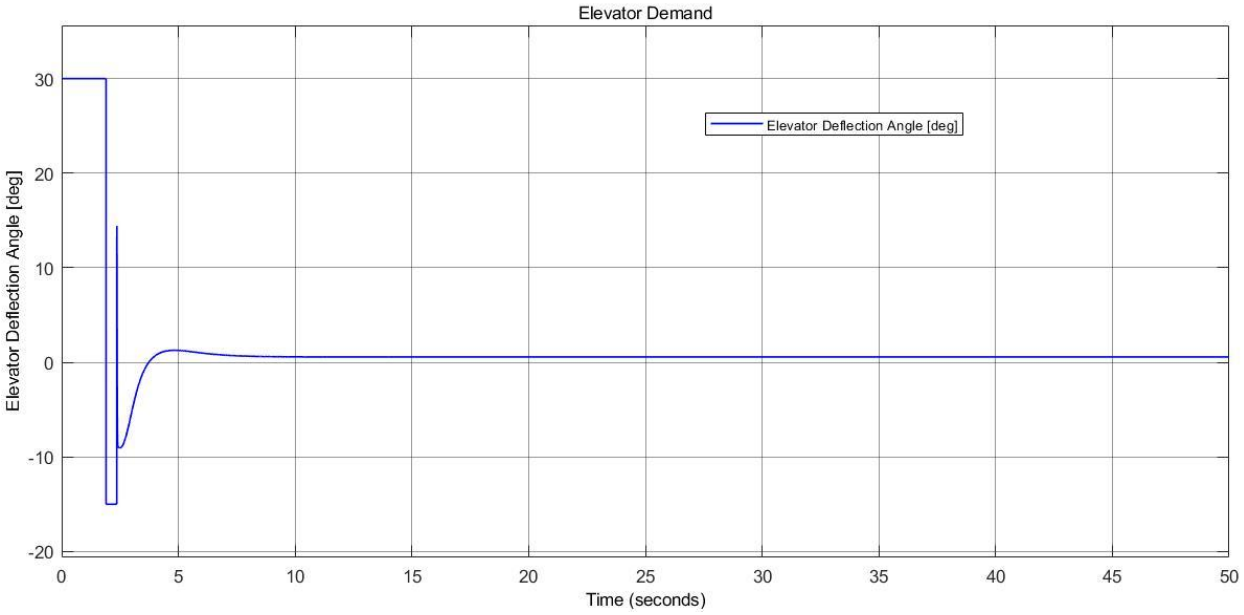


Figure 5.1.1.11.. *Elevator Demand.*

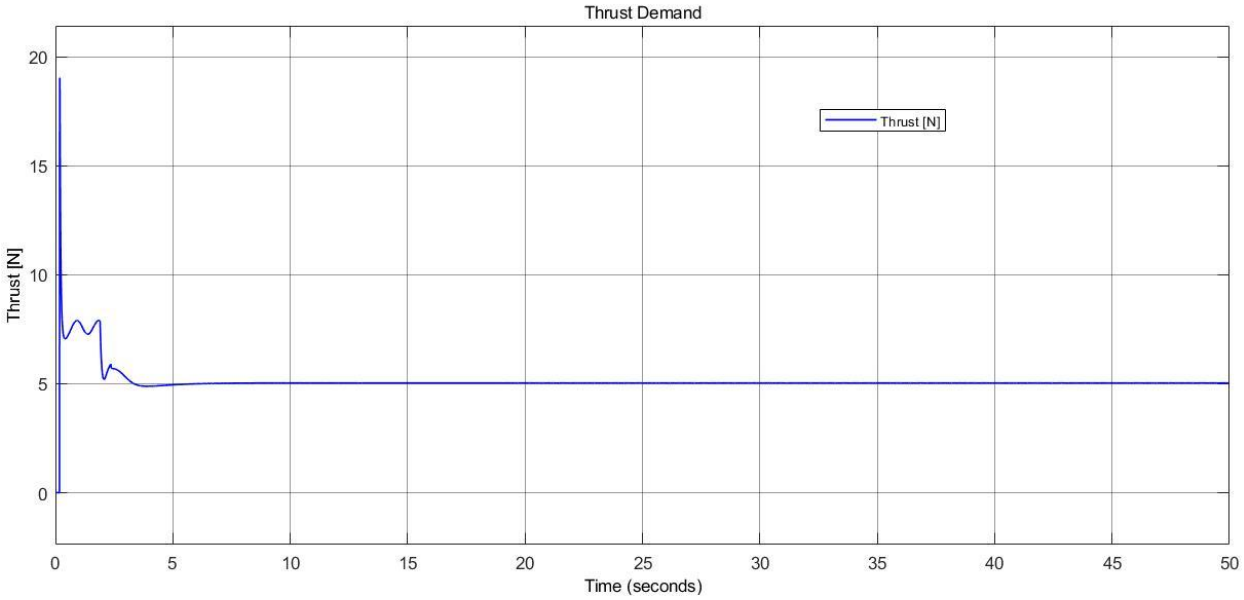


Figure 5.1.1.12. *Thrust Demand.*

[3] Altitude Tracking Response +15 m

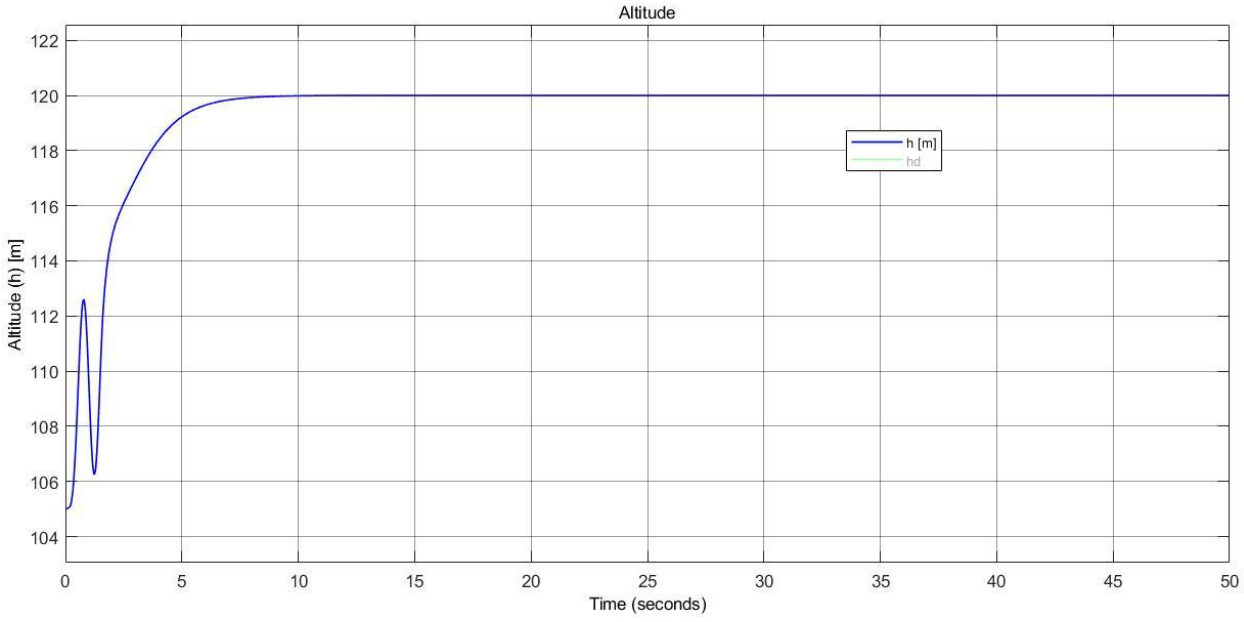


Figure 5.1.1.13. *Altitude Response.*

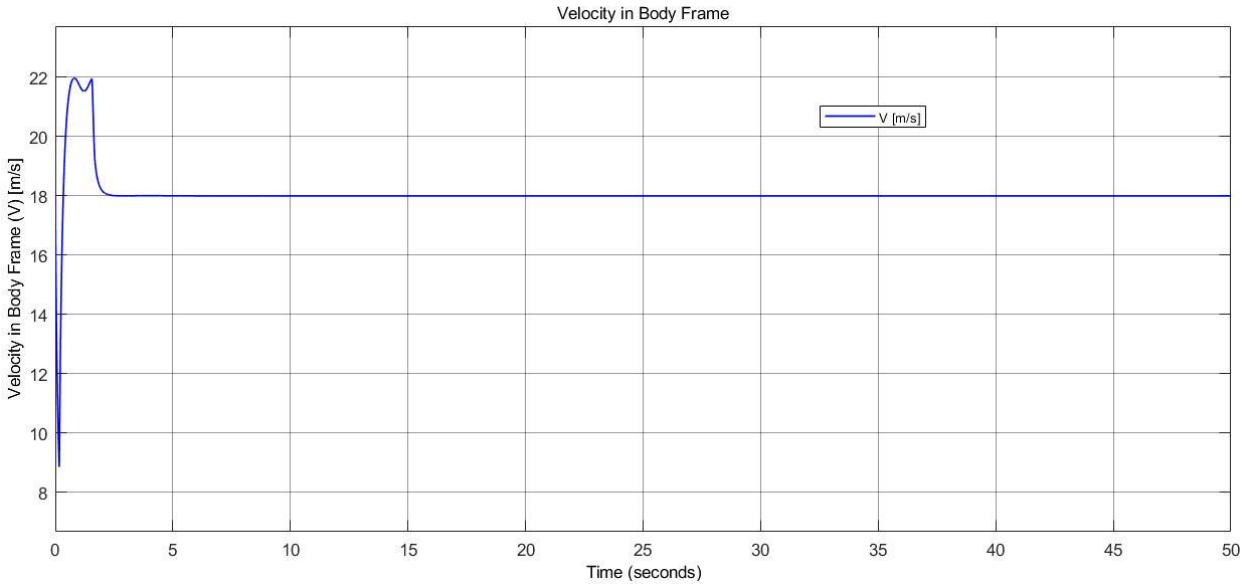


Figure 5.1.1.14. *Velocity in Body Frame.*

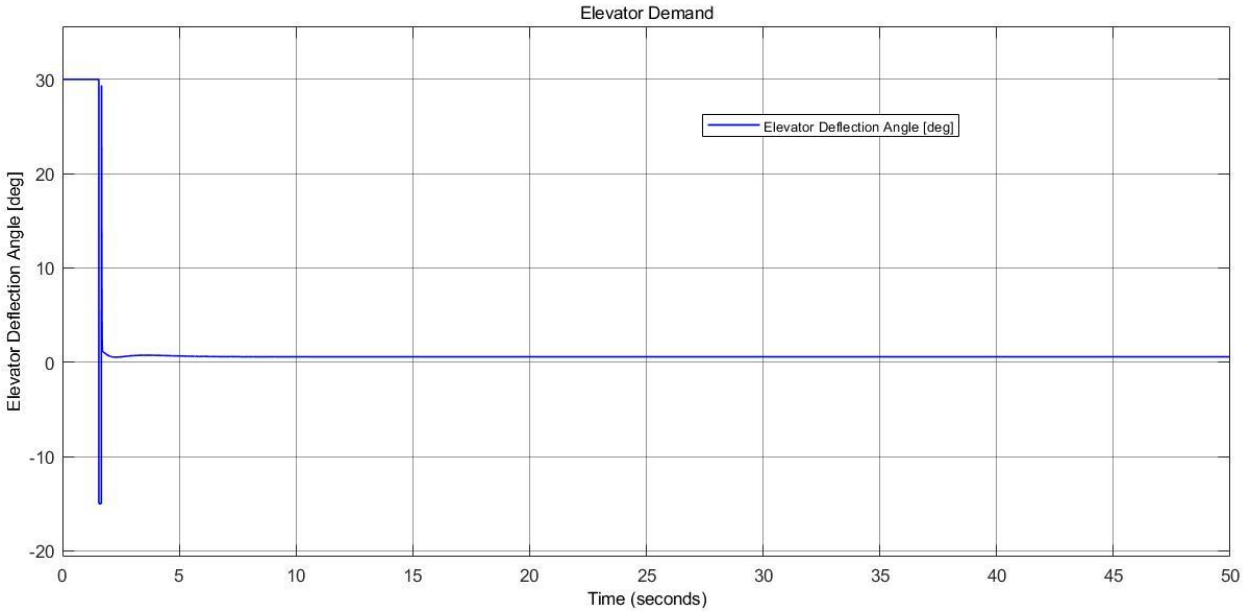


Figure 5.1.1.15. *Elevator Demand.*

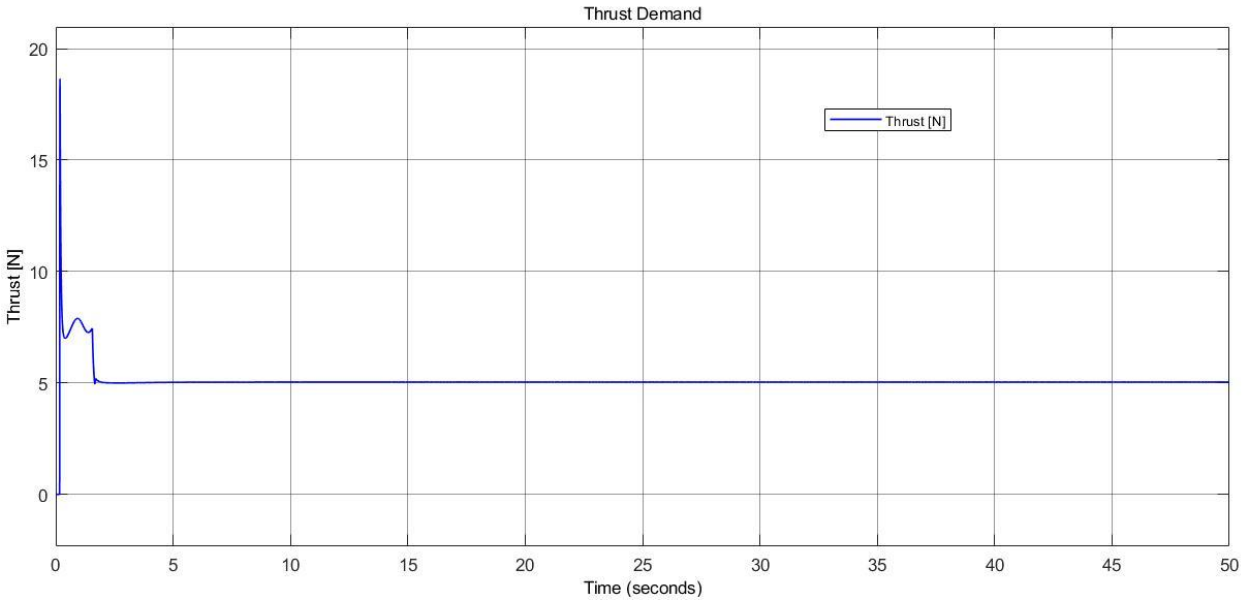


Figure 5.1.1.16. *Thrust Demand.*

5.1.1.4. Lateral Response

[1] Response due to +2 deg Disturbance in Roll Angle (ϕ)

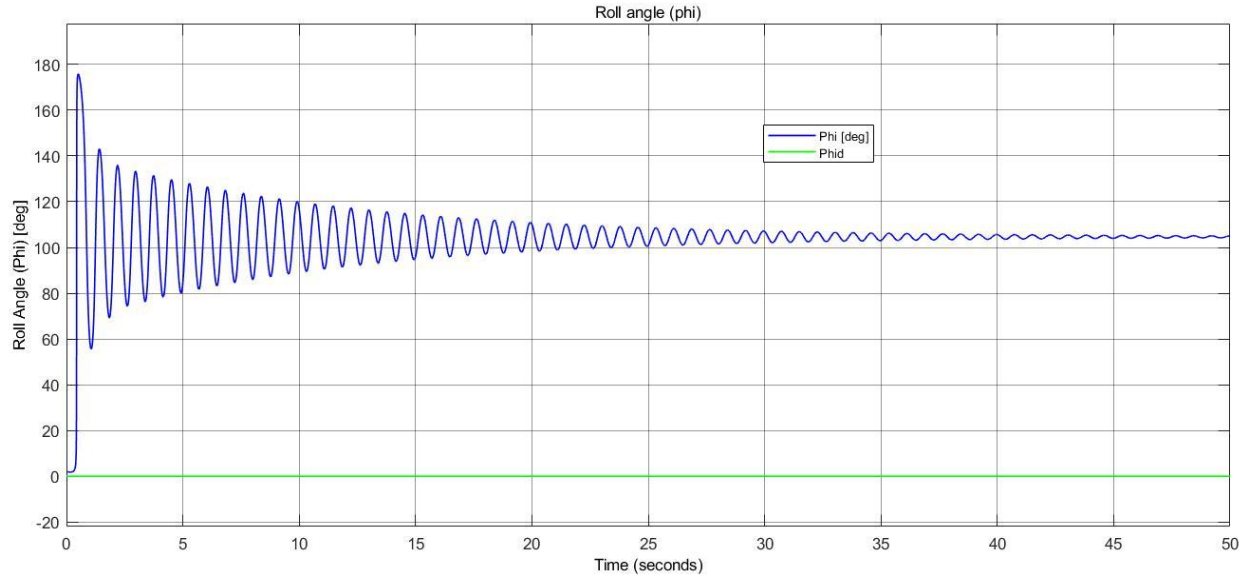


Figure 5.1.1.17. *Roll Angle Response.*

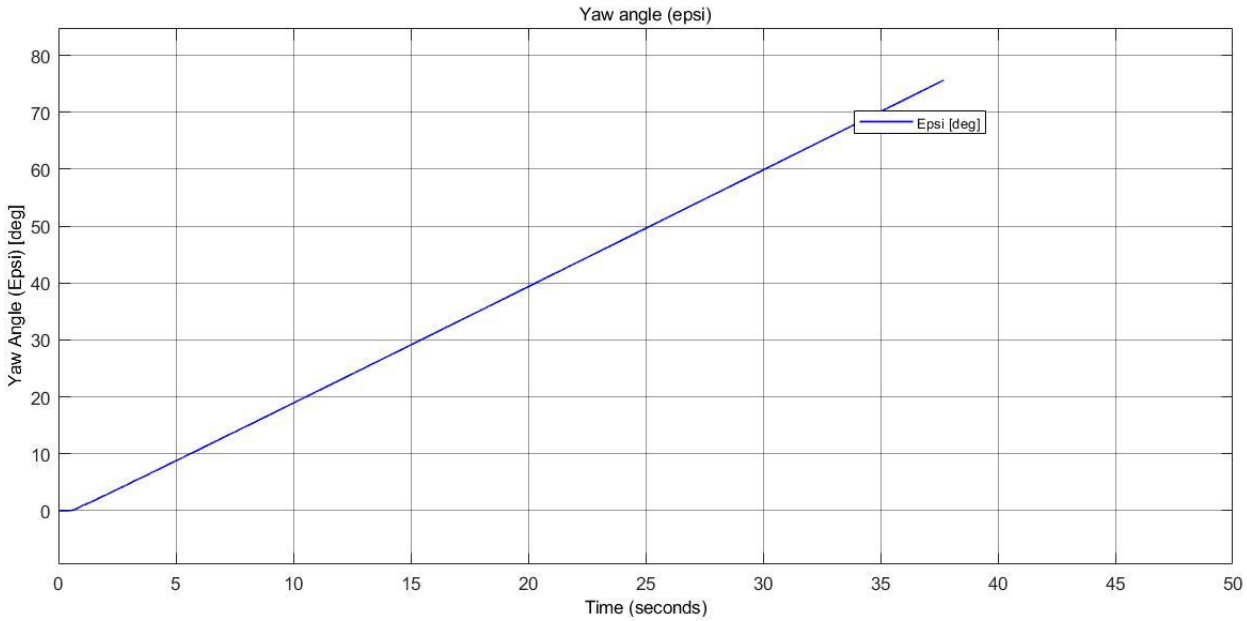


Figure 5.1.1.18. *Yaw Angle Response.*

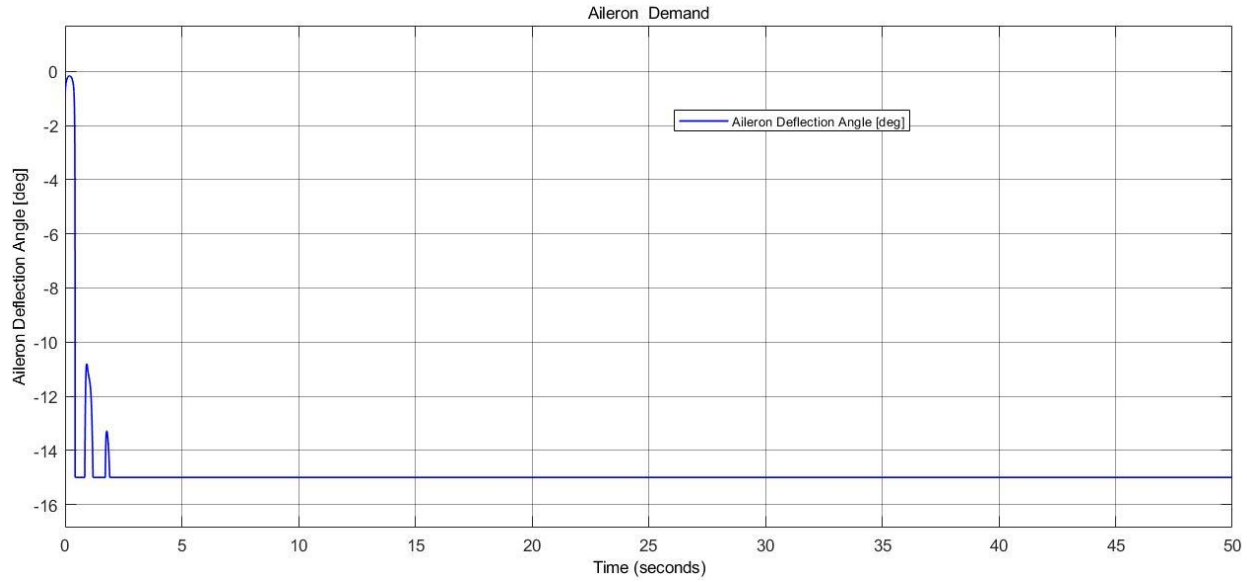


Figure 5.1.1.19. *Aileron Demand.*

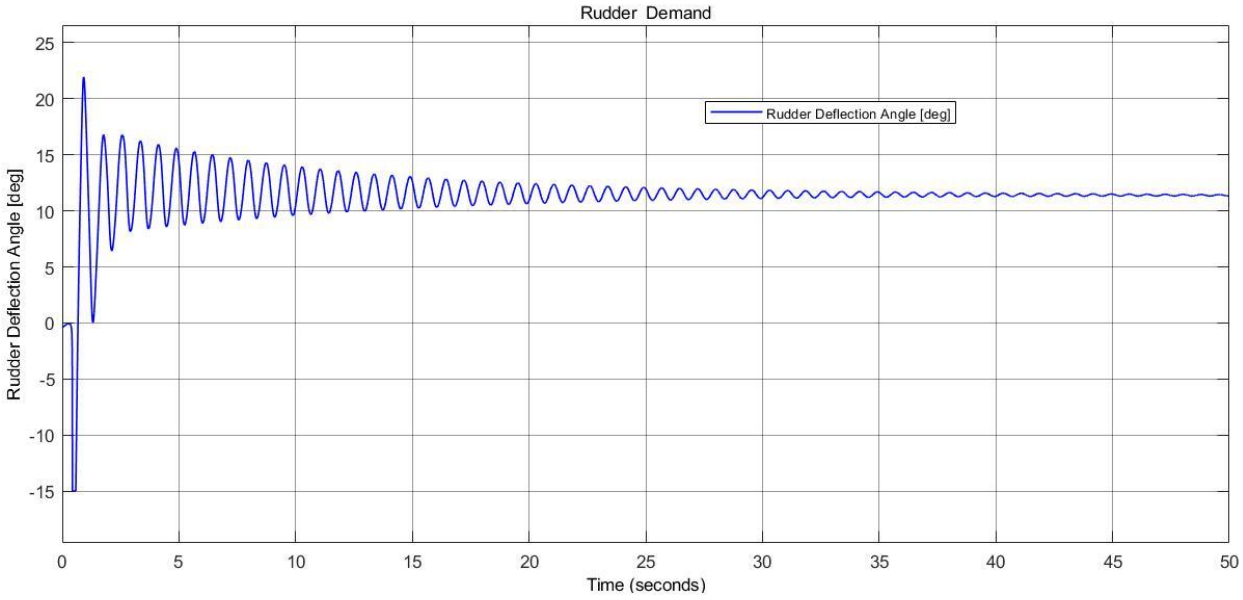


Figure 5.1.1.20. *Rudder Demand.*

5.1.1.5. Linear Longitudinal System

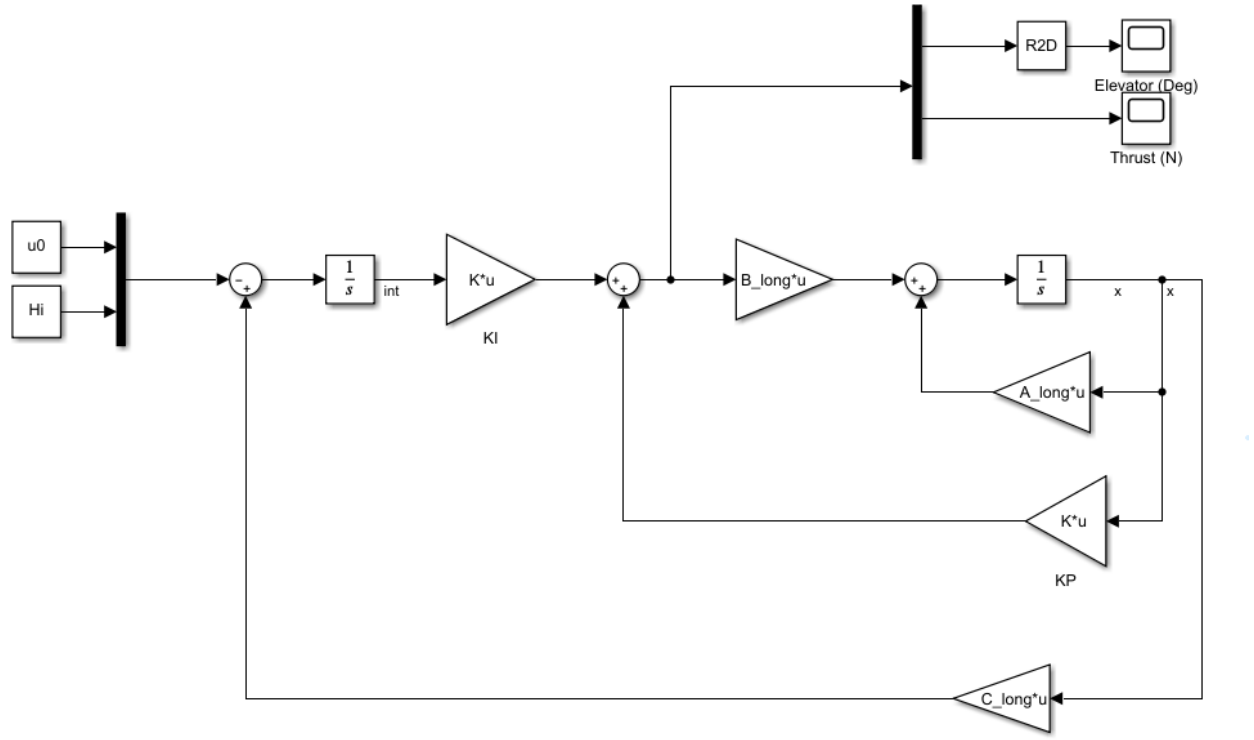


Figure 5.1.1.21. *Linear Longitudinal System Simulink Block.*

The linear implementation of the LQR was used to verify the methodology in tracking desired values in u and h.

5.1.1.6. Discussion

In conclusion, LQR provides robust control for linearized systems. Meanwhile, in nonlinear systems, it can produce an unsound response. The lateral mode is far more unstable than the longitudinal mode to control. Despite the good tracking ability of LQR in longitudinal mode, it still poses a great overshoot that makes it not the best choice when compared to other controllers. Tuning the parameters manually is an exhaustive process, especially when nonlinearities are present in the system; the response might be unstable.

5.1.2. Total Energy Control System

The Total Energy Control System (TECS) is an advanced flight control concept that combines the control of an aircraft's elevator and turbines. It is utilized in civil aircraft automation, operational control modes and Fly-By-Wire command augmentation. TECS manages multiple inputs and outputs, enabling it to simultaneously control the airplane's velocity and altitude by incorporating engine thrust and elevator while retaining the straightforward structure of a PID controller. Simultaneous control of altitude and velocity is not simple, as an increase in pitch angle will result in a rise in altitude but a decrease in speed. In contrast, increasing the propulsion throttle will result in an increase in both airspeed and altitude. The two inputs (pitch and throttle) therefore influence the two outputs (height and airspeed). TECS offers a solution by expressing setpoints in terms of energies as opposed to height and velocity. Increases in thrust and pitch angle convert kinetic energy to potential energy, respectively. Thus, thrust is used to modulate specific total energy, whereas pitch angle is used to maintain specific energy equilibrium between kinetic and potential energy.In addition to its use on commercial aircraft, TECS can also be applied to UAVs. In one study, TECS was proposed for the longitudinal height channel of a fixed-wing VTOL UAV. The system segregated speed control from track control and utilized an L1 nonlinear path tracking guidance algorithm for lateral trajectory tracking, resulting in enhanced curve tracking capability and wind resistance.

In order to decouple longitudinal motion in terms of flight path angle and airspeed, TECS employs a multivariable approach based on an energy controller. This allows for more precise

control of the flight path and airspeed of the aircraft, resulting in improved performance and safety.

Total energy =Kinetic energy +potential energy

$$E_T = \frac{1}{2}mV^2 + mgh$$

Taking the derivative of the equation w.r.t time

$$\dot{E}_T = mV\,V` + m\,g\,h`$$

Specific Energy rate is

$$\dot{E} = \frac{\dot{E}_T}{mgV} = \frac{V`}{g} + \frac{h`}{V} = \frac{V`}{g} + sin\,\gamma$$

For small flight path angle

$$\dot{E} \approx \frac{V`}{g} + \gamma$$

From the equation of motion of an aircraft in straight level flight

$$T - D = mg(\frac{V`}{g} + sin\,\gamma) \approx mg(\frac{V`}{g} + \gamma\ )$$

Since at Trim Thrust equals Drag, therefore change in thrust equals

$$\Delta T = mg(\frac{V}{g} + \gamma)$$

Since the change in Trust (ΔT) is related to the rate of specific total energy, thrust or throttle can be utilized to regulate total energy. Elevator Control, being energy-efficient and accurate, is employed to convert kinetic energy into potential energy and vice versa.

This is described by the specific energy balance equation:

$$B` = \gamma - \frac{V`}{g}$$

5.1.2.1. Simulink implementation:

The system is composed of two feedback loops: one for regulating total energy by adjusting the throttle setpoint, and another for maintaining energy balance by setting the pitch angle for attitude control.The attitude controller operates using a two-stage loop process. The first loop calculates the difference between the desired and actual attitude and uses a proportional

controller to generate a rate setpoint. The second loop then calculates the rate error and uses a proportional-integral controller to produce the desired angular acceleration.

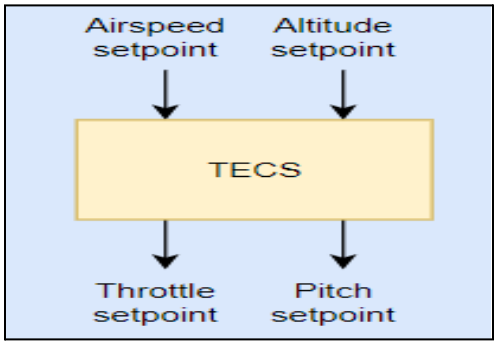


Figure 5.1.2.1 Total Energy control system

This acceleration is used to determine the control surface positions (ailerons, elevators, rudders, etc.) through control allocation or mixing, based on prior knowledge of the system. The controller is tuned for cruise speed and is scaled according to airspeed measurements (if available) since control surfaces are more effective at high speeds and less effective at low speeds.

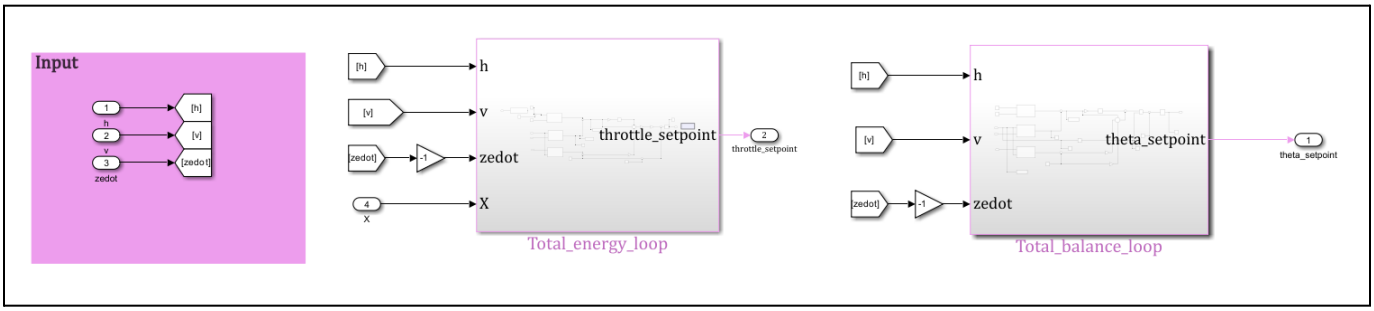


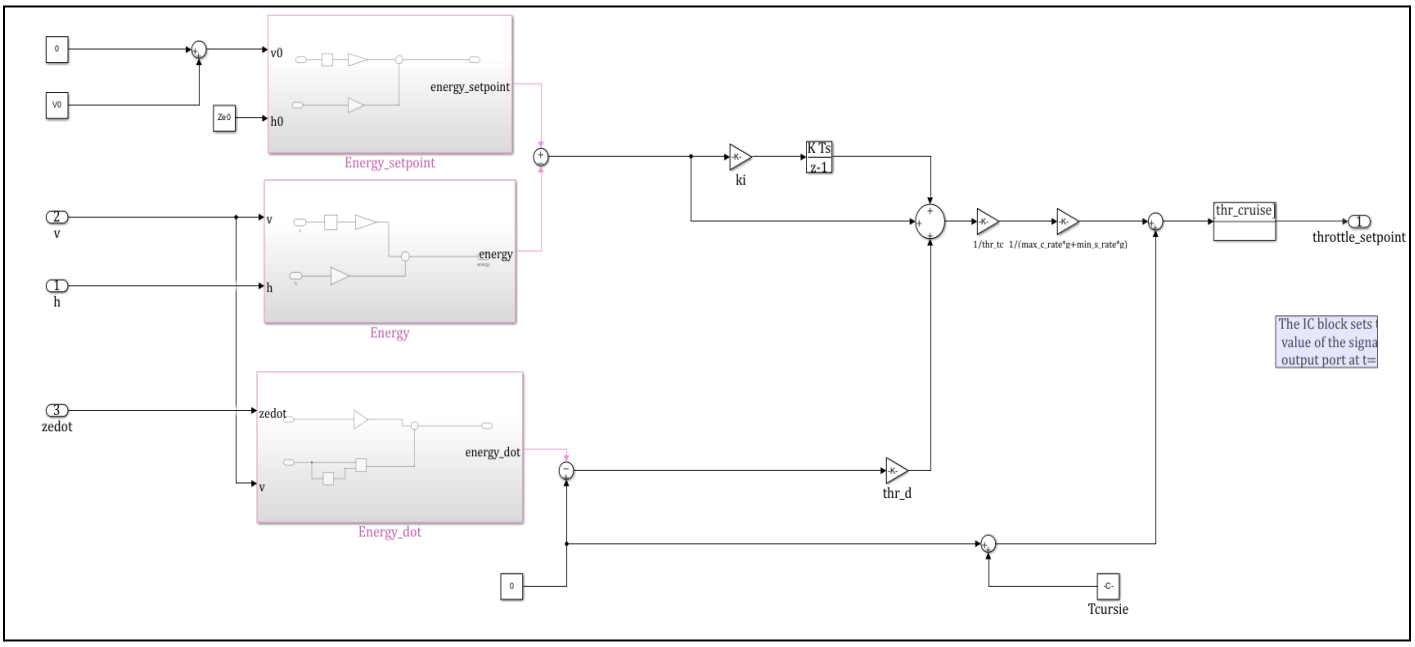
Figure 5.1.2.2:TECS simulink implementation

Specific Total Energy Loop:

This loop uses velocity, height, and their rates of change over time to determine the setpoints for total energy and energy rate. The resulting throttle setpoint is then sent to the engine

Figure 5.1.2.3: specific total energy loop

The loop includes damping and integrator gains as well as two additional gains located in the top right of the model. The first gain is the reciprocal of the time constant, while the second gain is a factor based on the aircraft's maximum climb and minimum sink rate. An aircraft's



maximum climb rate is the greatest vertical speed at which it can rise, while its minimum sink rate is the slowest vertical speed at which it can descend in calm air. These values vary for each aircraft and are influenced by factors such as its design and performance abilities

Specific Energy Balance Loop:

This loop uses velocity and height data, as well as their rates of change over time, to determine the setpoints for energy balance and energy balance rate. The resulting pitch angle setpoint is then sent to the attitude controller.

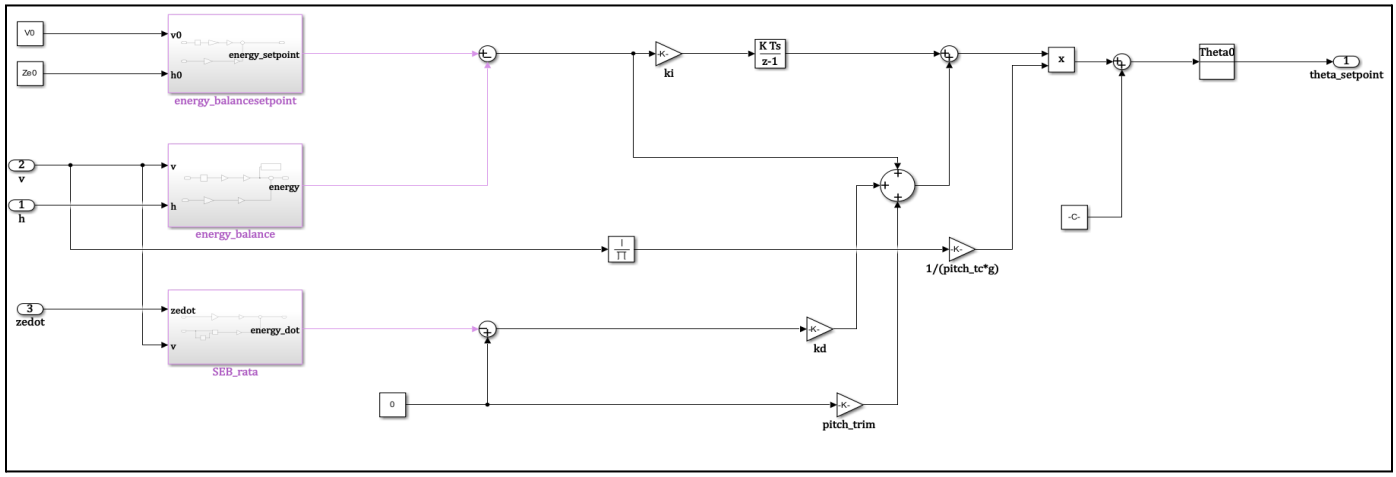


Figure 5.1.2.4:Specific Energy Balance Loop Simulink implementation

Integration of the controller with

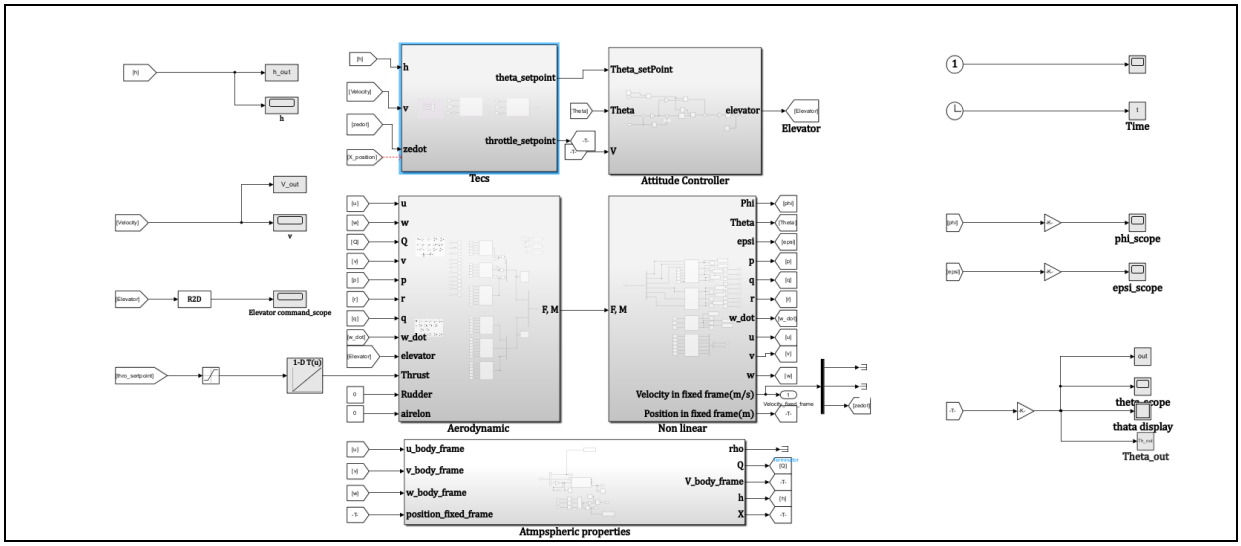


Figure 5.1.2.5 Integration of TECS controller with Fixed wing model

5.1.2.2. Gain Tuning

Due to the added complexity of this controller and its use in a MIMO system, traditional tuning methods such as root locus were not suitable. So we use different optimization techniques.

The optimization techniques are :

1- Genetic algorithm

2-particle swarm algorithm

3- Differential Evolution(DE)

4-Simulated Annealing (SA)

Optimizing on the following gains:

1- Pitch loop damping gain (Pitch_d)

2- Integrator gain (int_g) (the integrator gain is the same for both loops)

3- Pitch loop time constant (Pitch_tc)

4- Throttle loop damping gain (Thr_d)

5- Throttle loop time constant (Thr_tc)

The least squares method was employed to calculate the error for each response, and the total error served as the cost function to be minimized.

$$E_h = \sum_{i=0}^{n} (h_{setpoint} - h_i)$$

$$E_V = \sum_{i=0}^{n} (V_{setpoint} - V_i)^2$$

$$E_\theta = \sum_{i=0}^{n} (\theta_{setpoint} - \theta_i)^2$$

$$Erro_{total} = \sqrt{E_h^{\,2} + E_V^{\,2} + E_\theta^{\,2}}$$

5.1.2.3. Optimization Techniques

5.1.2.3.1 Genetic algorithm:

Genetic Algorithm (GA) is an optimization technique that uses principles of genetics and natural selection to find optimal or near-optimal solutions to complex problems . GA simulates natural selection and evolution by starting with an initial population of potential solutions encoded as binary strings (chromosomes). The fitness of each chromosome is evaluated using a fitness function, and the fittest chromosomes are selected for reproduction. Genetic operators such as crossover and mutation are applied to create the next generation of chromosomes. This process is repeated until a satisfactory solution is found or a stopping criterion is met.

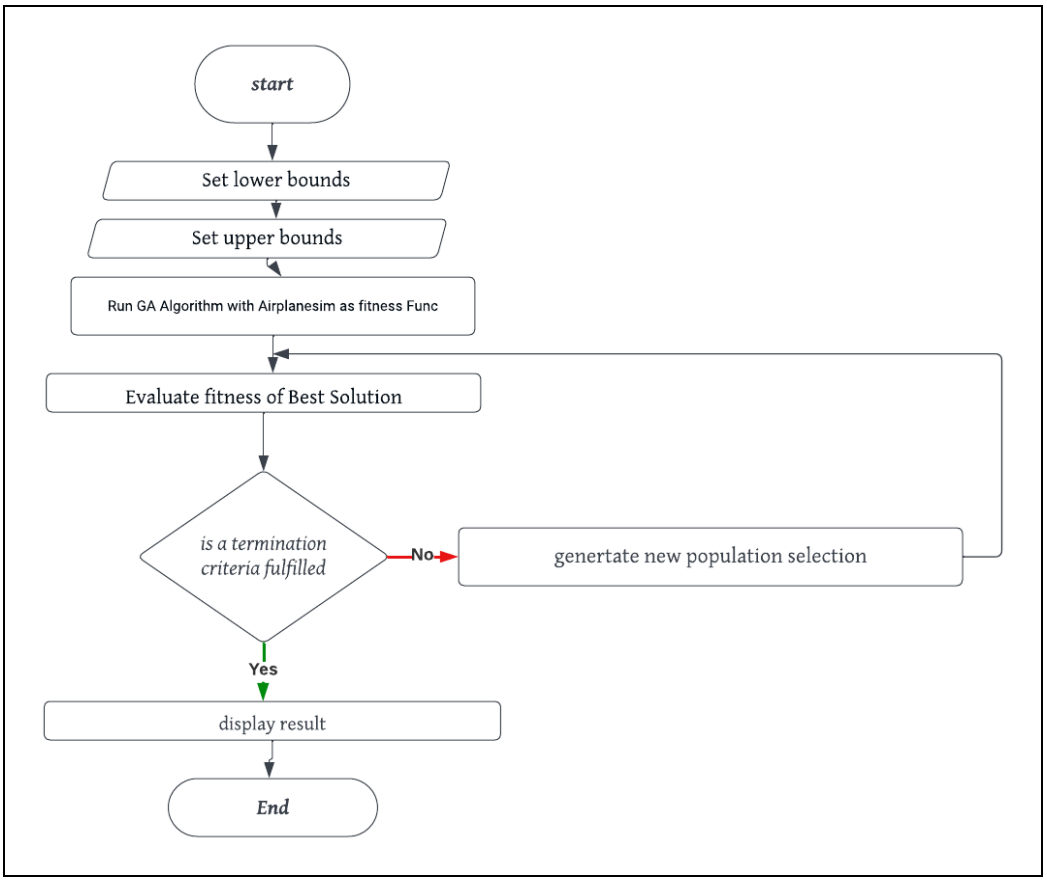


Figure 5.1.2.6. Flow chart of GA algorithm related to tune gains

5.1.2.3.1.1. Results of the genetic algorithm

The output gains of the genetic algorithm were as follows:

Gains	**Values**
Integrator gains	0.05
Throttle Damping	0
Throttle Time constant	1.5
Pitching Damping	0.1
Pitch Time constant	2.5

Table 5.1.2.1 Value of Gains tuning GA algorithm

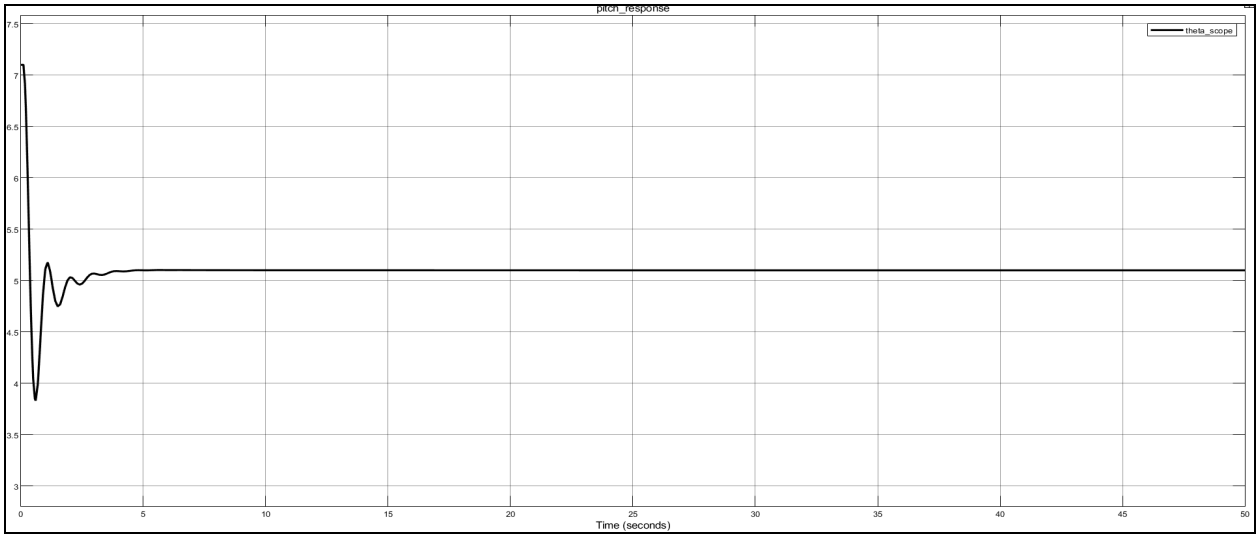


Figure 5.1.2.7.pitch angle response after using GA algorithm to tune gains

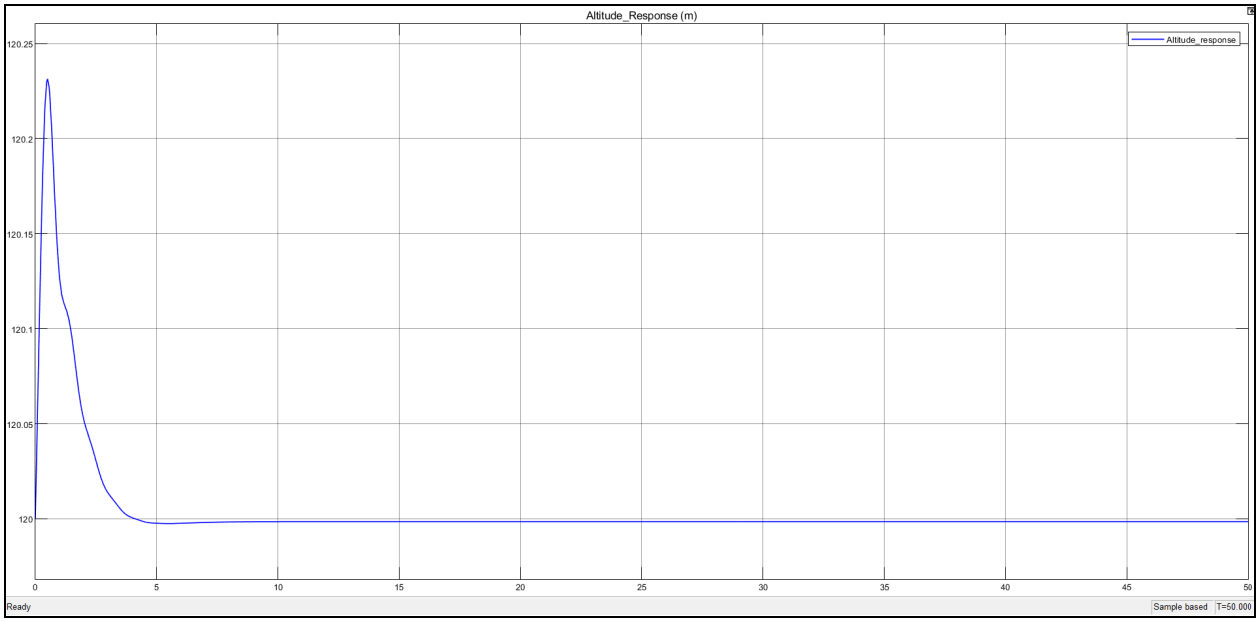


Figure 5.1.2.8.Altitude response after using GA algorithm to tune gains

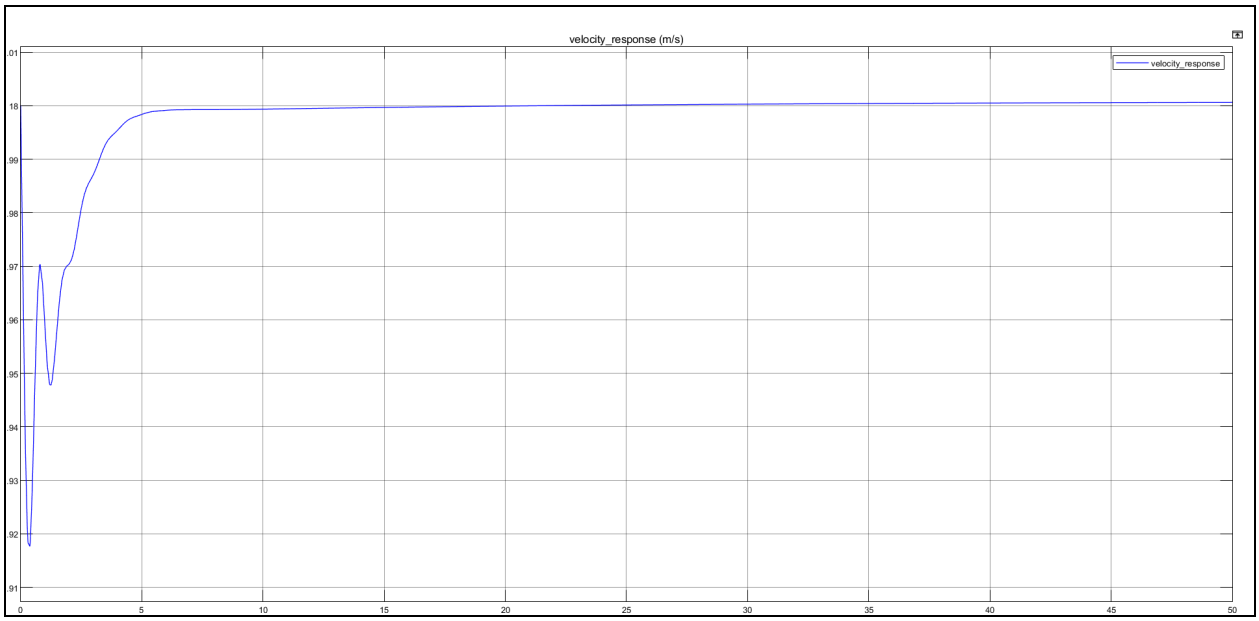


Figure 5.1.2.9.velocity response after using GA algorithm to tune gains

5.1.2.3.2. Particle swarm:

Particle Swarm Optimization (PSO) is a stochastic optimization algorithm that mimics the social behavior of animals such as insects, herds, birds, and fish. It uses a population of particles that move in the search space according to mathematical formulas based on their position and velocity. Each particle's movement is influenced by its local best position and the global best position found by other particles. This guides the swarm toward the best solutions. PSO is a metaheuristic algorithm that can search large spaces of candidate solutions without making assumptions about the problem being optimized. It does not use the gradient of the problem and does not require differentiability, unlike classic optimization methods such as gradient descent and quasi-Newton methods

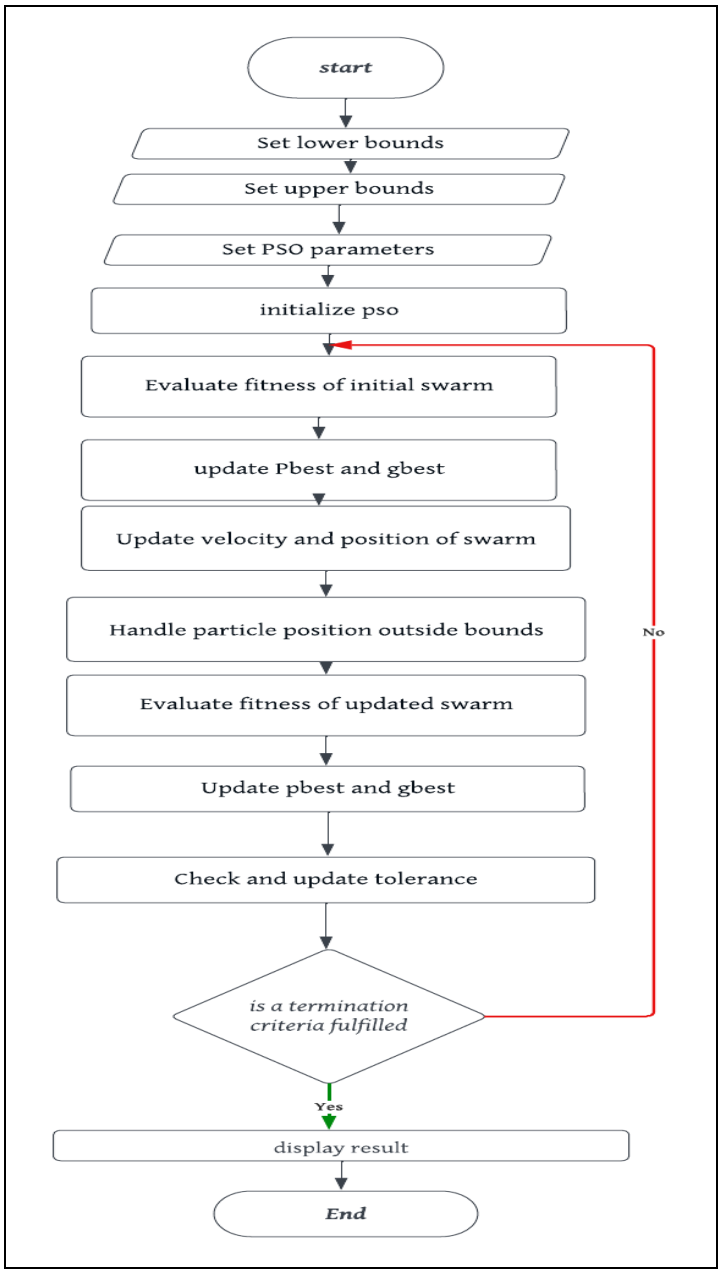


Figure 5.1.2.10. Flow chart of PSO algorithm related to tune gains

5.1.2.3.2.1. Results Of the PSO algorithm

The output gains of the PSO algorithm were as follows:

Gains	**Values**
Integrator gains	0.05
Throttle Damping	0.1
Throttle Time constant	5
Pitching Damping	0.15
Pitch Time constant	2

Table 5.1.2.2 Value of Gains tuning PSO algorithm

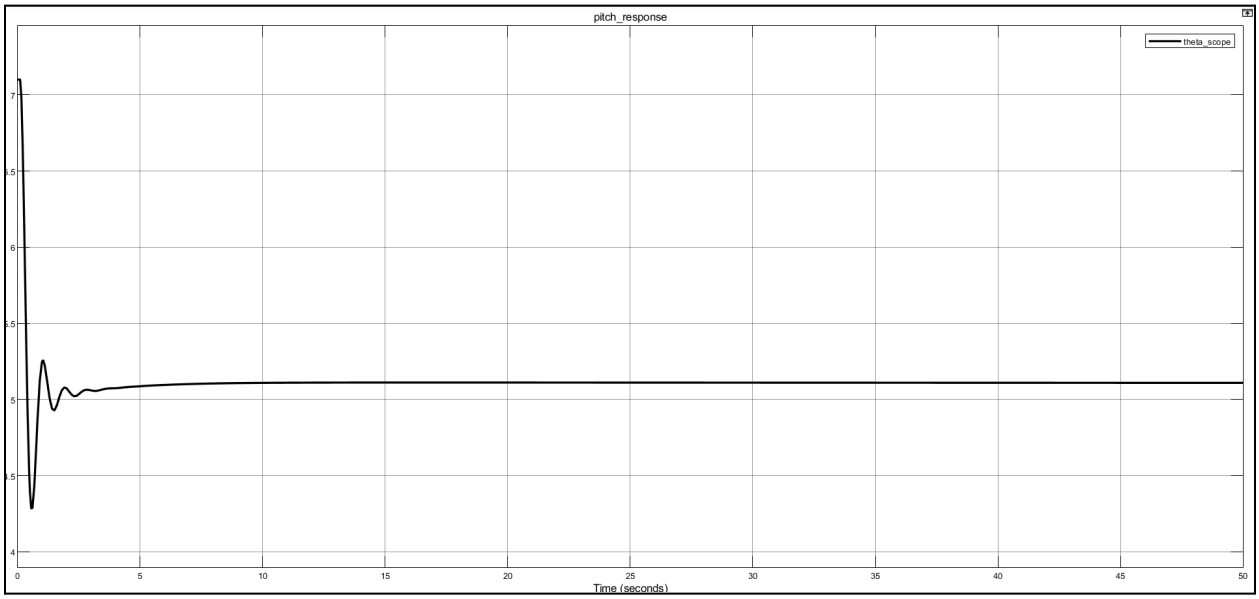


Figure 5.1.2.11. Pitch angle response after using PSO algorithm to tune gains

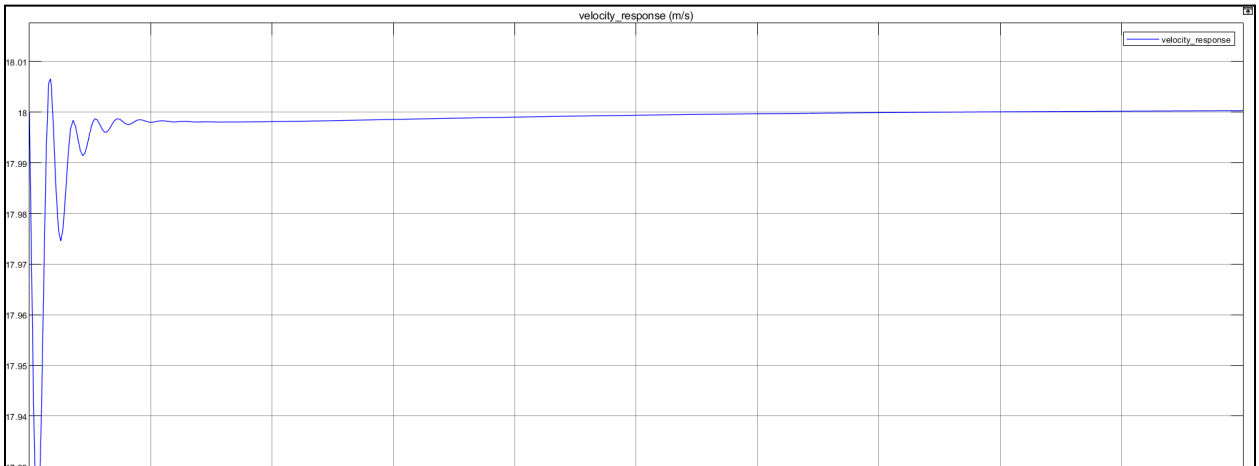


Figure 5.1.2.12. velocity response after using the PSO algorithm to tune gains

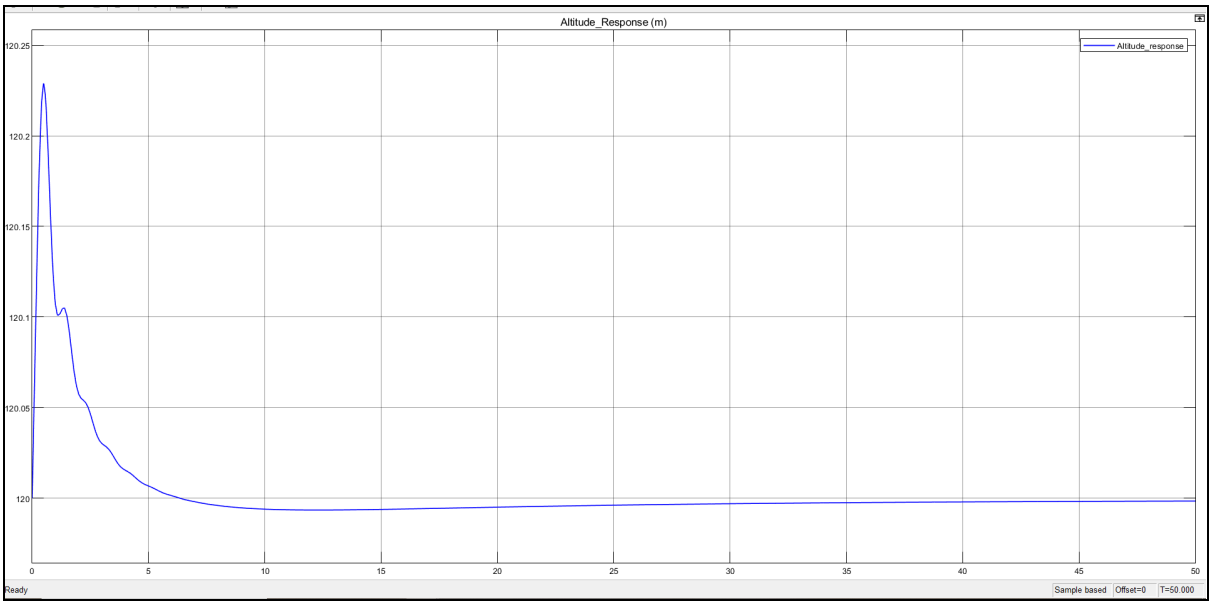


Figure 5.1.2.13. Altitude response after using PSO algorithm to tune gains

5.1.2.3.3. Simulated Annealing

Simulated Annealing (SA) is a probabilistic optimization technique based on the annealing process in metallurgy. It is often used to find the global optimum of a function, particularly when the search space is large or the problem is non-convex. SA starts with a random solution and improves it by making small changes. At each iteration, a new solution is generated by perturbing the current solution and evaluated using an objective function. If the new solution is better, it is accepted; if it is worse, it may still be accepted with a probability determined by a temperature parameter. This parameter controls the acceptance of worse solutions and decreases over time according to a cooling schedule. At high temperatures, the algorithm explores a wider range of solutions; as the temperature decreases, it becomes more selective and converges toward the global optimum. SA has been successfully applied to various optimization problems and can be easily adapted to different types of problems.

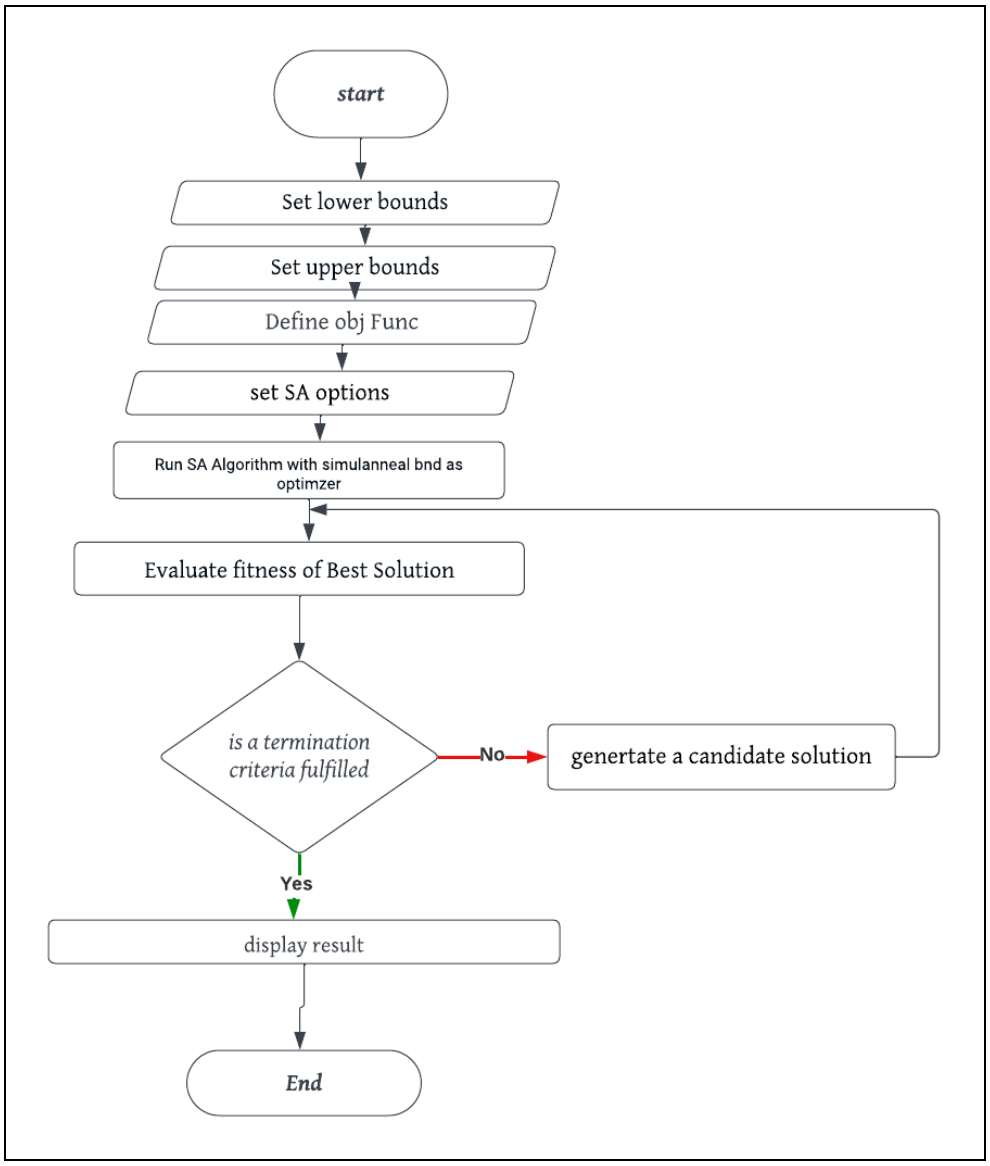


Figure 5.1.2.14. Flow chart of SA algorithm

5.1.2.3.3.1. Results of the SA algorithm

The output gains of the SA algorithm were as follows:

Gains	**Values**
Integrator gains	**0**

Throttle Damping	**.15**
Throttle Time constant	**3.5**
Pitching Damping	**.05**
Pitch Time constant	**3**

Table 5.1.2.3 Value of Gains tuning SA algorithm

Response:

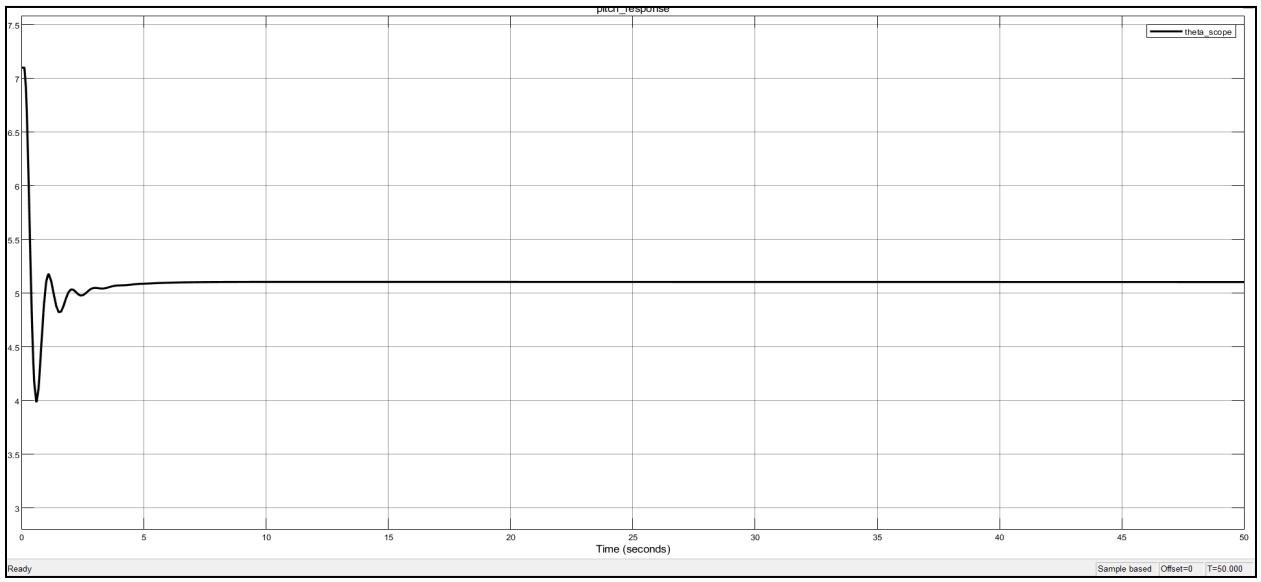


Figure 5.1.2.15. Pitch angle response after using SA algorithm to tune gains

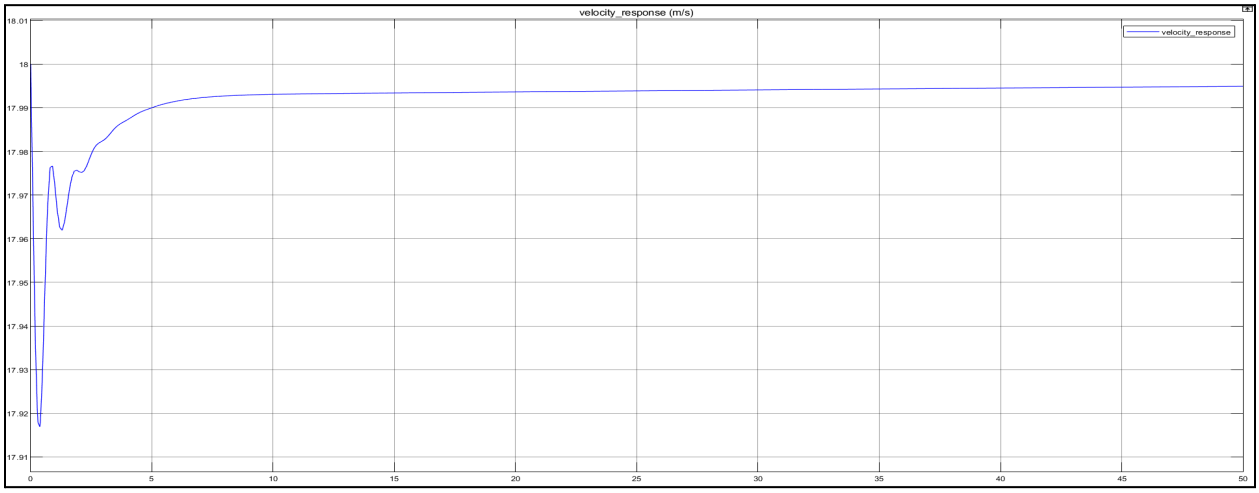


Figure 5.1.2.16. velocity response after using SA algorithm to tune gains

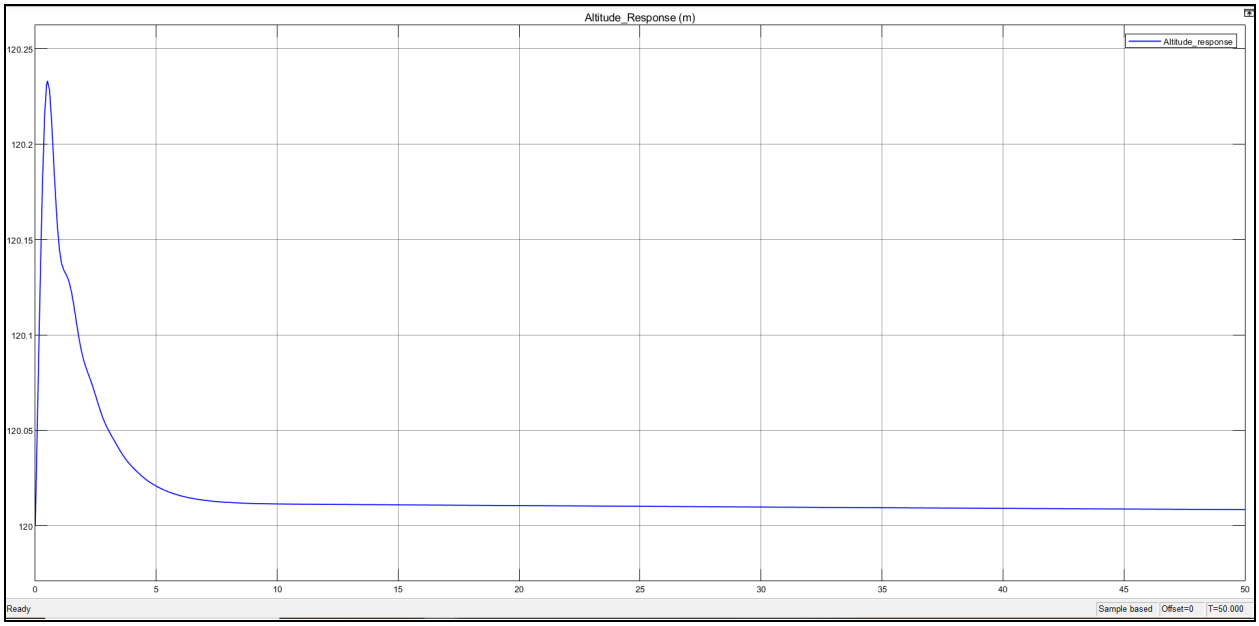
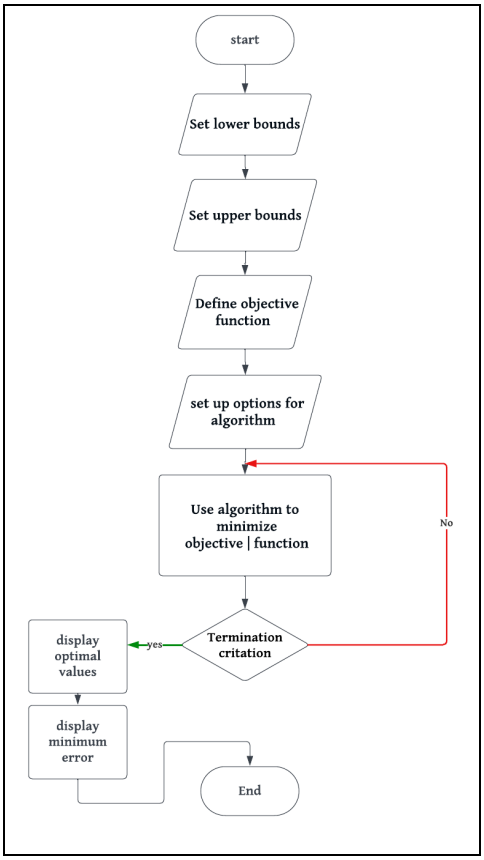


Figure 5.1.2.17. Altitude response after using SA algorithm to tune gains

5.1.2.3.4. Differential Evaluation

Differential Evolution (DE) is a population-based optimization algorithm introduced by Storn and Price in 1995. It is similar to other evolutionary algorithms but uses a different method to generate new solutions. DE iteratively improves a population of solutions through an evolutionary process. Pairs of solutions are selected and combined using a differential mutation operator to create trial solutions, which are then compared to existing solutions using a selection operator. The better solutions are kept for the next iteration. DE is simple and easy to use, with few control parameters, and can be applied to a wide range of problems. It has been shown to be effective at finding global optima in complex search spaces.



5.1.2.3.4.1. Results Of the DE algorithm

The output gains of the DE algorithm were as follows:

Gains	**Values**
Integrator gains	**.05**
Throttle Damping	**0.1**
Throttle Time constant	**3.5**
Pitching Damping	**.35**
Pitch Time constant	**2**

Table 5.1.2.4 Value of Gains tuning DE algorithm

Response:

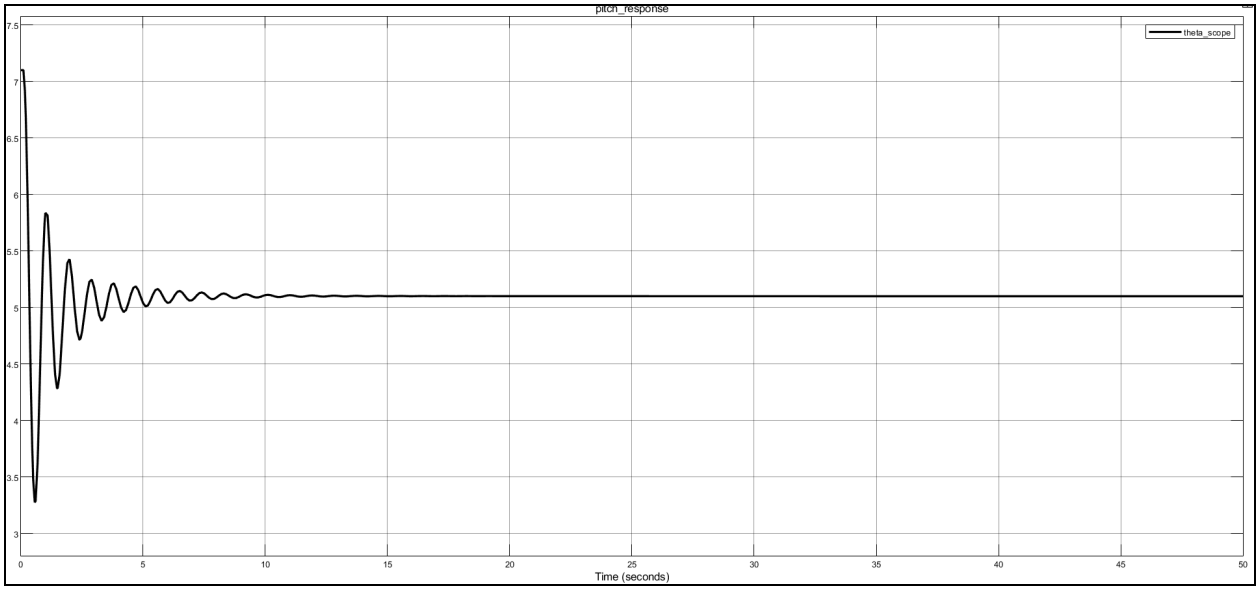


Figure 5.1.2.19. Pitch angle response after using DE algorithm to tune gains

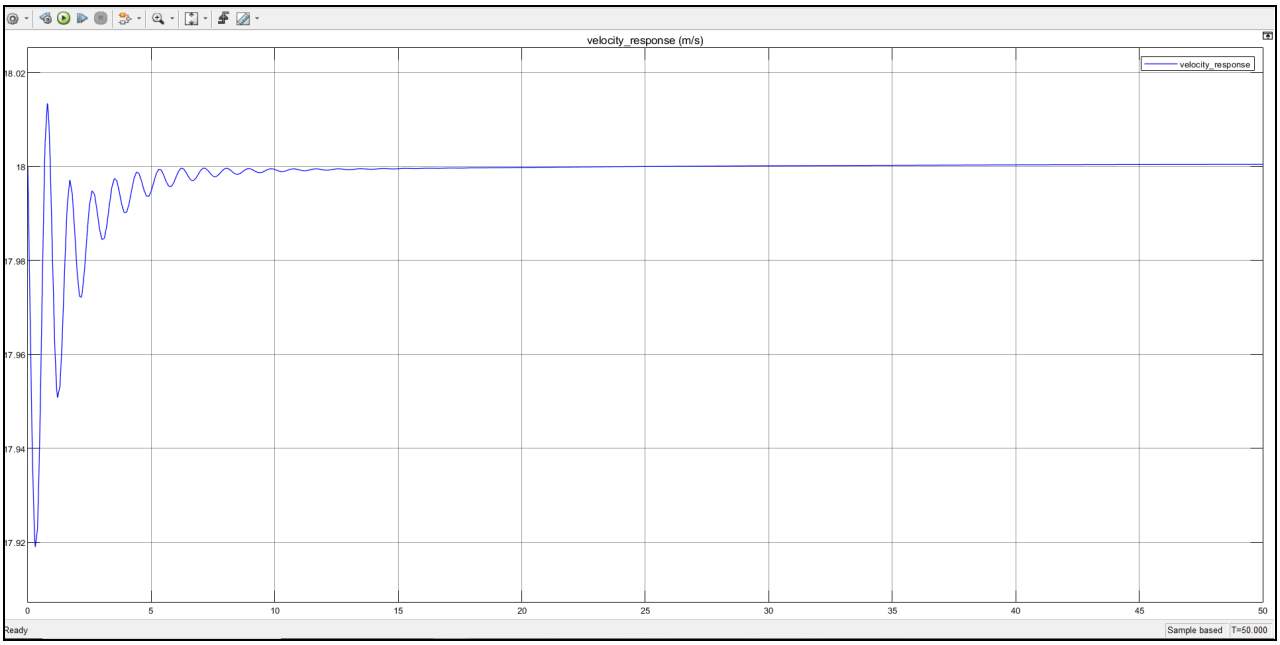
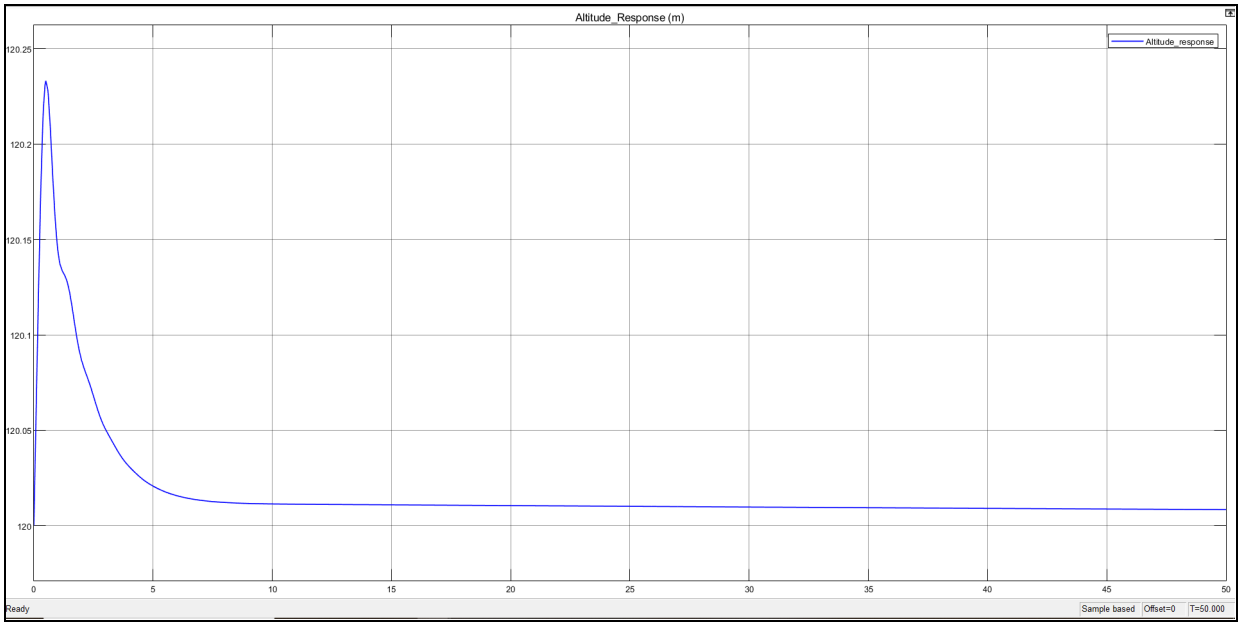


Figure 5.1.2.20. velocity response after using DE algorithm to tune gains

Figure 5.1.2.21. Altitude response after using DE algorithm to tune gains



5.1.2.3.5. Comparison between Different Methods:

All four algorithms discussed herein are categorized as metaheuristic optimization techniques. This implies that they do not rely heavily on assumptions regarding the problem being optimized, and are capable of exploring vast search spaces of candidate solutions. Furthermore, they are all population-based algorithms that employ a set of candidate solutions to navigate through the search space.

The primary distinctions between these algorithms lie in their approach to generating new candidate solutions and updating their populations. Genetic Algorithm (GA), for instance, uses selection, crossover, and mutation operators; Particle Swarm Optimization (PSO) employs personal best and global best positions; Simulated Annealing (SA) utilizes perturbations and a temperature parameter; and Differential Evolution (DE) relies on differential mutation and crossover operators.

It is important to note that each algorithm has its own strengths and weaknesses, and may be better suited for certain types of optimization problems. For example, GA is proficient at tackling discrete optimization issues, while PSO excels at handling continuous optimization problems. SA is particularly effective at escaping local optima, whereas DE is well-suited for identifying global optima in intricate search spaces.

5.1.2.3.5.1. Comparison in Response:

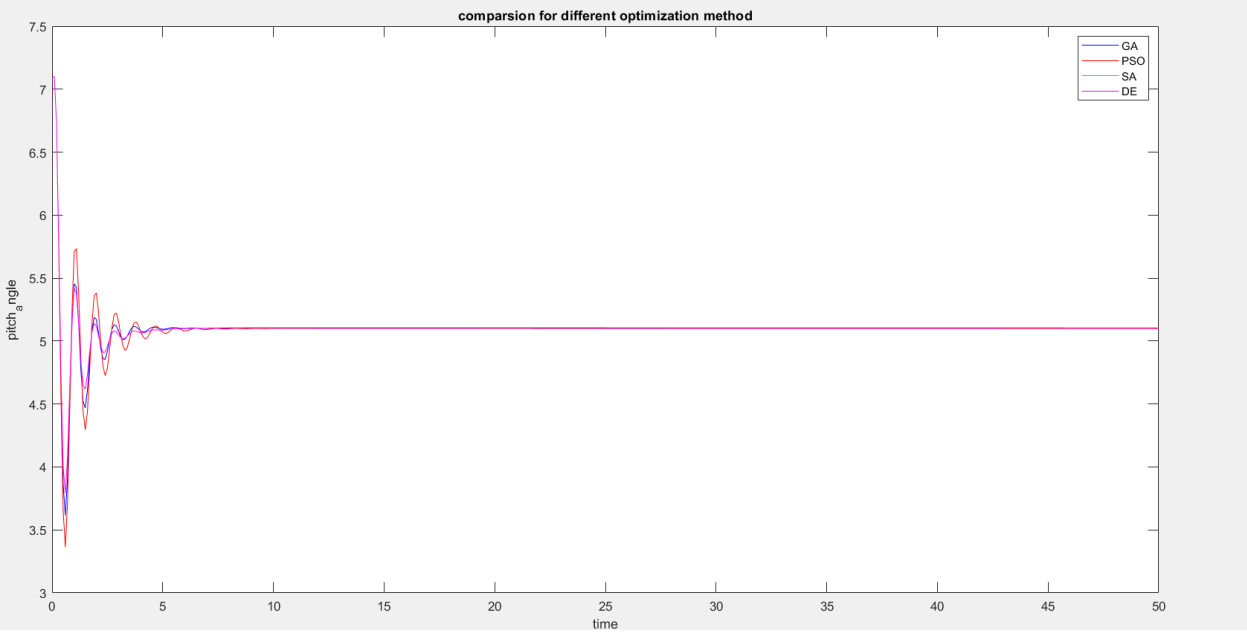


Figure 5.1.2.22. Pitch_angle response by four optimization method

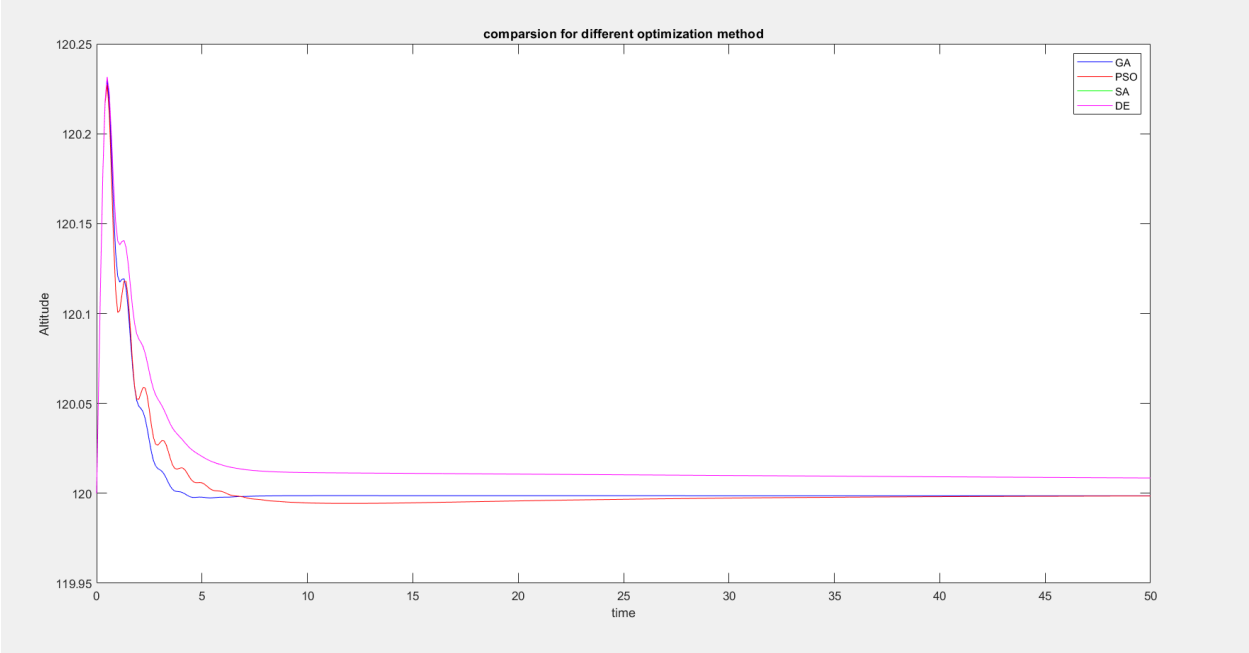


Figure 5.1.2.23. Altitude response by four optimization method

From Graphs it is shown that GA is the best one for optimization as it has the least oscillation.

5.1.3. Model Predictive Control MPC

Model Predictive Control (MPC) is a sophisticated approach for optimizing system performance under specified controlling constraints. These constraints represent the boundaries that are imposed on the system due to physical, economic, and environmental aspects. MPC is advantageous over the classical controllers since it can manage system constraints systematically and control more than one variable simultaneously. In that sense, it's widely used in various applications within different industries. MPC conducts control actions based on predicted future occurrences by just applying the present time step of an optimized finite time-horizon (Darwish et al., 2022).

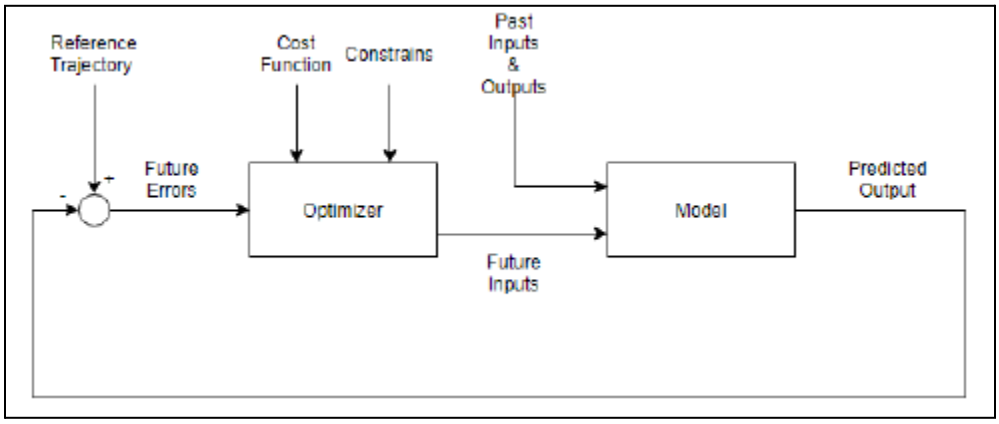


Figure 5.1.3.1. MPC block diagram

Unlike Classical feedback controllers, MPC modifies the control action before changing the output setpoint. And, by combining the capacity to predict with traditional feedback operation, the controller changes are smoother and better approach the values of an optimal control action.

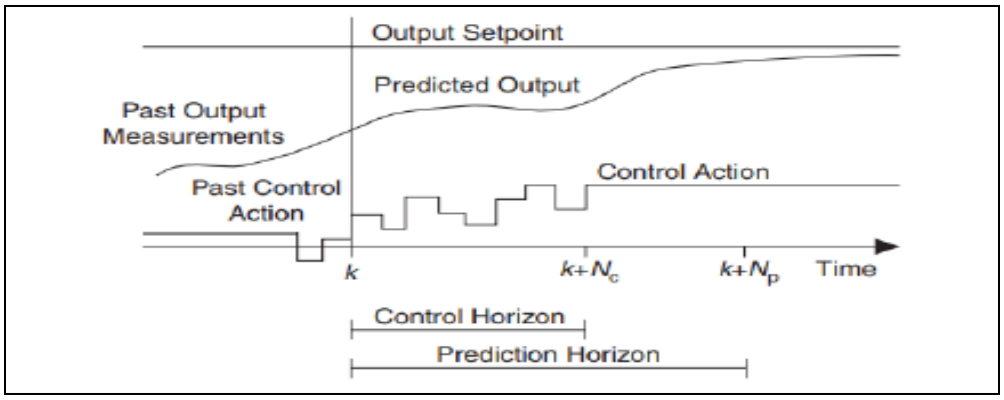


Figure 5.1.3.2. Prediction & control horizons

Where:

- Np is the prediction horizon is the number of samples in the future for which the MPC controller can predict the output.
- Nc is the Control horizon is the number of samples within the prediction horizon where the MPC controller can influence the control action.

The MPC controller predicts the planet's output at tk+Np for each time step tk. And the control action remains constant after the control horizon has passed. After each cycle, both the prediction and control horizons advance one step forward, allowing the MPC to predict the plant's next output, and so on.

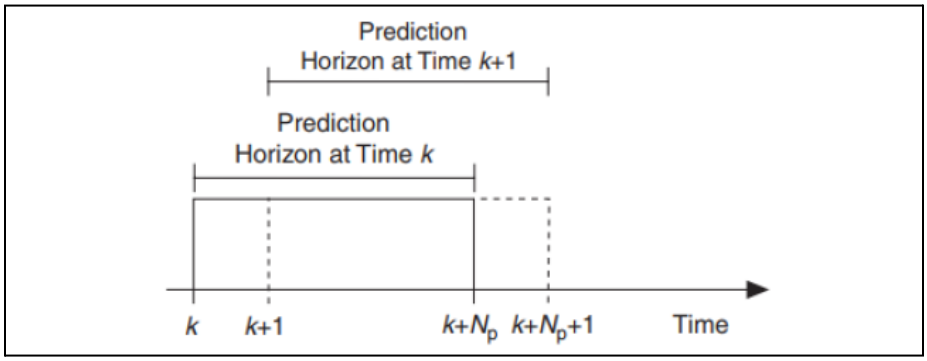


Figure 5.1.3.3. Prediction & control horizons

For the prediction horizon, National Instruments recommends setting the prediction horizon length based on the needs of the control problem. As for a short prediction horizon, the MPC does not get enough information to process and hence, the MPC acts as a conventional feedback controller. On the other hand, a long prediction horizon does increase the prediction ability of the MPC but it consumes lots of power and decreases the performance of the MPC by adding more data to analyze and process.

Because of the inability to change the control action after the control horizon ends, a short control horizon is recommended as it carefully changes the control action. Conversely, a long control horizon results in more aggressive changes to the control action and by extension, results in energy waste and/or oscillations.

The MPC controller, like other optimization approaches, is based on minimizing a cost function by computing the required sequence of future control actions. Nonlinear MPC, which requires the complete model (simulation) and uses Nonlinear Programming optimization (NLP), and Hybrid MPC, which deals with both discrete and parametric variables with a combination of dynamics and discrete mode using Mixed-Integer Quadratic Programming (MIQP) or Mixed-Integer Linear Programming (MIQP) optimization.

The general representation of the MPC problem for a linear, scalar, and discrete-time system is:

$$x[k+1] = x[k] + u[k]$$

where , x[0] is the initial state, and u is the control action.

And the goal of the MPC is to find the *uk for k*=0,1,2,… that minimize the cost function:

$$J = \sum_{k=0}^{N_p-1} x^2[k] + \rho u^2[k] + x^2[N_p]$$

Where, $\varrho > 0$ is the weight.

For instance, if Np = 1, then predicted future state is:

$$x[1] = x_0 + u[0]$$

And by minimizing the cost function:

$$\min J = x_o^2 + \rho u^2[0] + (x_0 + u[0])^2$$

$$\therefore \frac{\partial J}{\partial u} = 0 = (1+\rho)u[0] + x_0$$

$$\therefore u[0] = -\frac{x_0}{1+\rho}$$

$$\therefore x[1] = \left(1 - \frac{1}{1+\rho}\right)x_0 \approx \begin{cases} 0, & for\ \rho = 0 \\ x_0, & for\ \rho \ggg 1 \end{cases}$$

Solving Unconstrained MPC results in an analytical solution, but solving Control or State Constrained MPC yields a numerical solution. As a result, the constrained MPC problem is described as the discrete state space:

$$x_{k+1} = Ax_k + Bu_k$$

$$y_k = Cx_k + Du_k$$

With the cost function as a linear quadratic function like:

$$J = \sum_{k=0}^{N_p} (\hat{y} - r)^T Q(\hat{y} - r) + \Delta u^T R \Delta u$$

Where, r is the output set-point,$\hat{y}$ is the predicted process output, Δu is the predicted change in control value, $\Delta u = uk - uk-1$, Q is the output error weight matrix, and R is the control weight matrix.

To acquire the optimal future control action, the problem is solved for $\partial J \partial u = 0$.

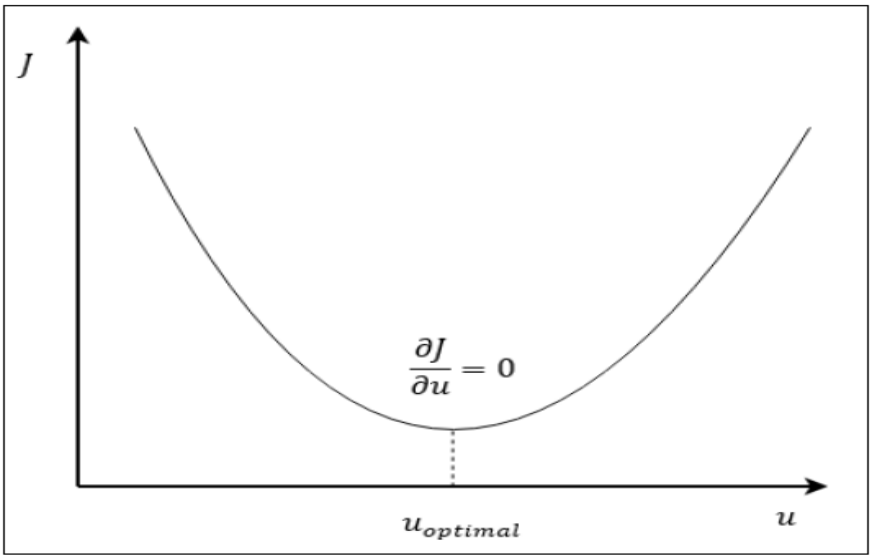


Figure 5.1.3.4. The relation between the Control Action & Cost function

Furthermore, the constraints are accounted for as constraints in the output: $ymin \leq y \leq ymax$, and constraints in the input: $umin \leq u \leq umax$.

Using the previously acquired information, the Simulink MPC toolbox was used to implement the controller to the aircraft model. Figure 5.1.3.5 shows the MPC block on Simulink and its inputs are mo which is the aircraft model output. and ref, the reference model the aircraft model needs to follow. And the MPC output, mv, represents the input control actions the system needs in order for the model output to follow the reference values.

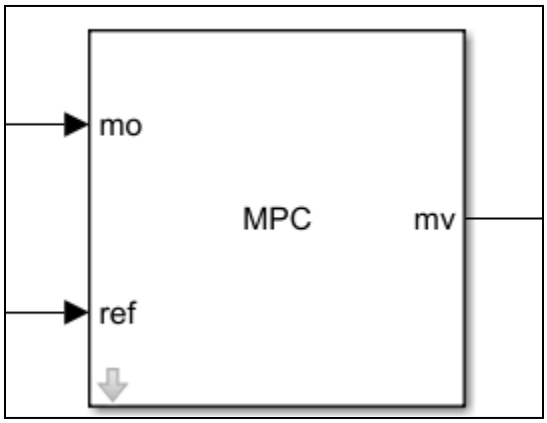


Figure 5.1.3.5.MPC simulink block

5.1.3.1. MPC for Longitudinal Model

In the Longitudinal Mode, The aircraft altitude and velocity represent the inputs of the MPC block. Both are set to follow reference values as step functions, The altitude setpoint is at 120m (trim altitude), while the velocity setpoint is 18 m/s. The outputs of the MPC block ate the change in the elevator and thrust as an input control action to the system. The first trial was tested by applying a 2-degree disturbance in the angle of attack, while the second trial is tested on the climb mode by starting at altitude 105m, in order to reach the 120m setpoint.

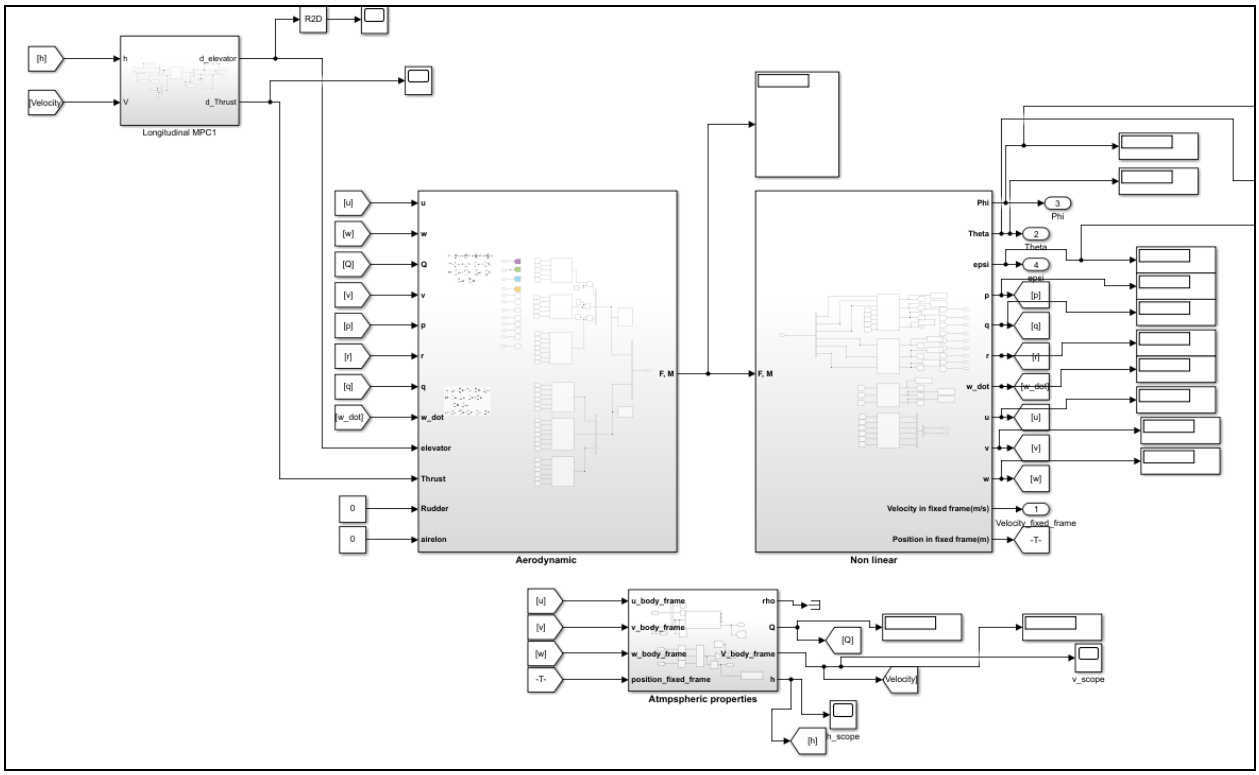


Figure 5.1.3.6. Integrating the MPC Controller into the Simulink Model

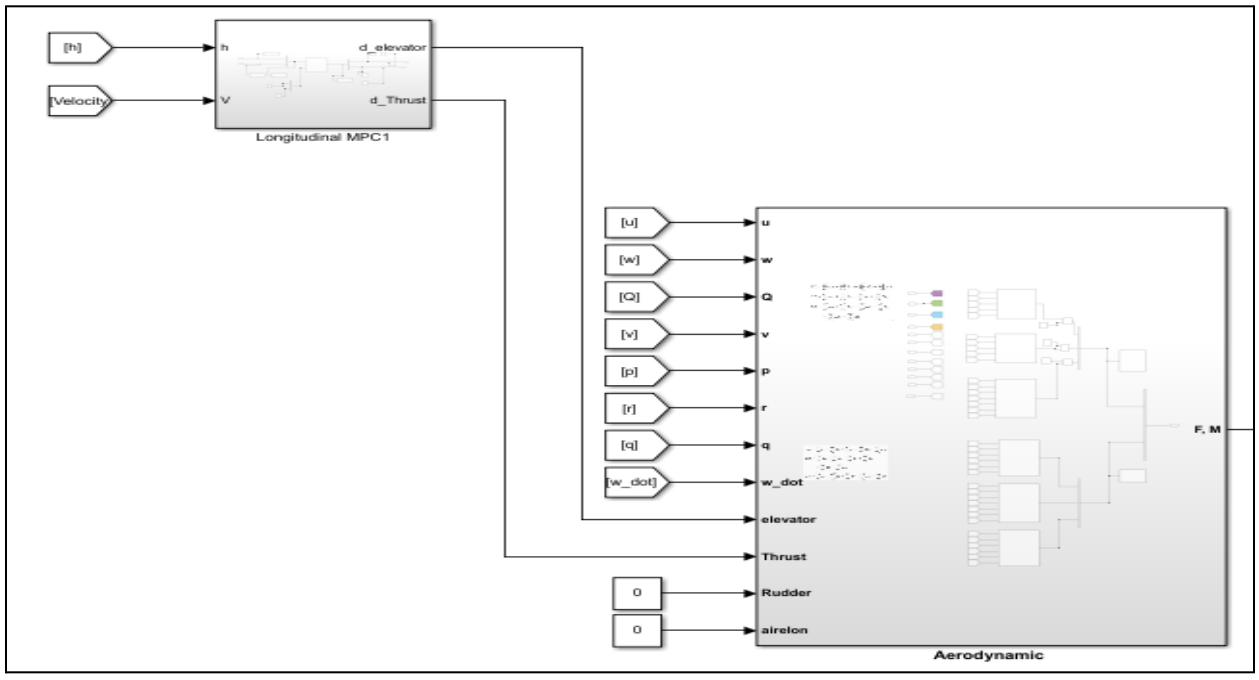


Figure 5.1.3.7.MPC longitudinal control using Both Elevator & Thrust Actions configuration

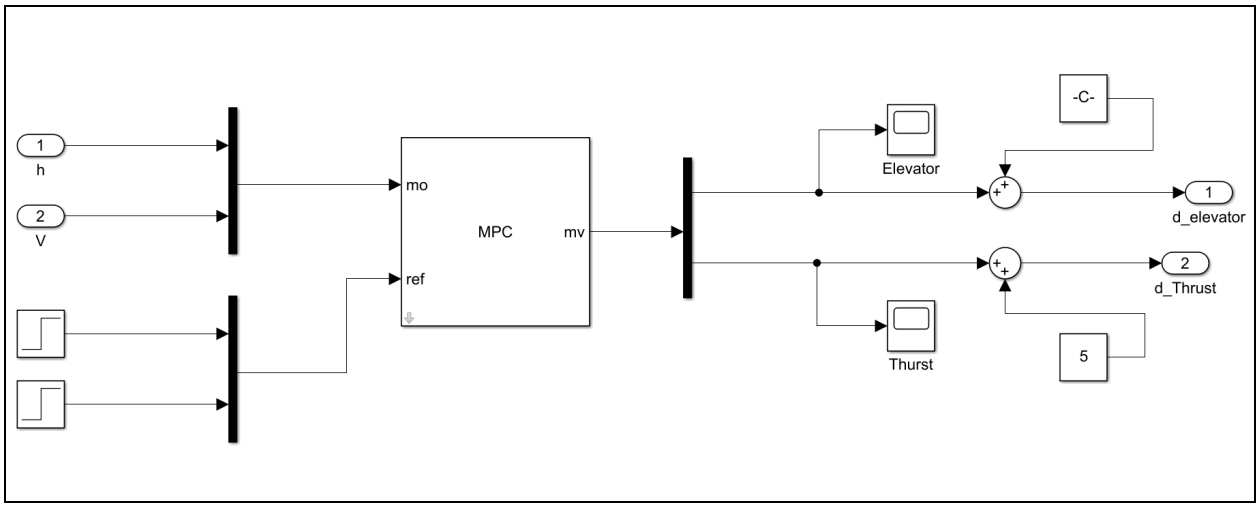


Figure 5.1.3.8. Longitudinal MPC Configuration

5.1.3.1.1. Tuning

Regarding the prediction horizon, It was found that by increasing the prediction horizon, the robustness and stability of MPC increase**.** The prediction horizons in general should be sufficiently large so that controlled output predictions can represent a large portion of the dynamics of the ongoing process. However, the computational cost of solving the MPC optimization problem increases as the prediction horizon increases(Alhajeri , 2021). A prediction horizon of 100 was used after iterations. While for the control horizon, the aggressiveness, the ability to stabilize unstable plants, and the computational cost of the controller increase, as the control horizons increase. However, the robustness of the control system decreases(Alhajeri , 2021). A control horizon of 2 was used as it was recommended for our model (Bagheri & Khaki Sedigh, 2015). The optimal sample time value in our case. Is 0.01, as shown in Figure 5.1.3.9.

Figure 5.1.3.10 shows the constraints used for both the elevator and thrust control surfaces. The maximum elevator deflection was set to 0.5235(30º) nose up and -0.2967 rad (17º) nose down, while the thrust control range was 0:16 N(-5:11 since an initial thrust value of 5N is added to the control value before affecting the model).

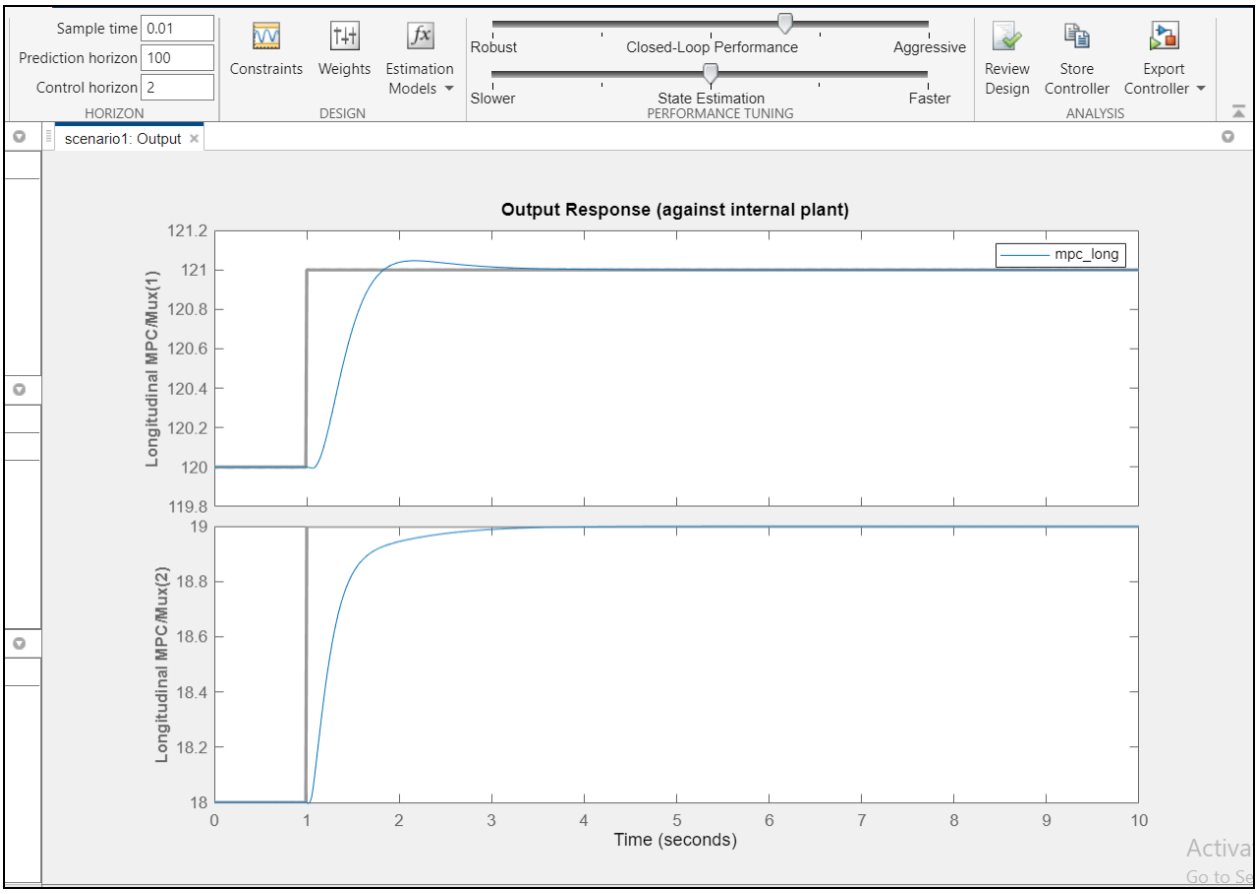


Figure 5.1.3.9.Output Response against the Commanded Response Using Both Elevator & Thrust

Input and Output Constraints

Channel	Type	Min	Max	RateMin	RateMax
▼ Inputs					
u(1)	*MV*	-0.26179	0.52359	-Inf	Inf
u(2)	*MV*	-5	11	-Inf	Inf
▼ Outputs					
y(1)	*MO*	-Inf	Inf		
y(2)	*MO*	-Inf	Inf		

Figure 5.1.3.10. MPC Input and output Constraints

The following formula developed by (Alhajeri , 2021) is used to get Input rate weights values of 3.344, as shown in Figure 5.1.3.11.

$$\rho = a\left(\frac{\theta}{\tau} + 0.94\right)^{0.15} \sigma^{0.9} K^2$$

Where,

K	is the steady state gain, and is set to 1
a and σ	are set according to(Alhajeri , 2021) with a recommended setting of (0.84,1)
Θ	is the prediction horizon which is 100
τ	is the sample time which is 0.01

Input Weights (dimensionless)

	Channel	Type	Weight	Rate Weight	Target
1	u(1)	MV	0	3.344	nominal
2	u(2)	MV	0	3.344	nominal

Output Weights (dimensionless)

	Channel	Type	Weight
1	y(1)	MO	1
2	y(2)	MO	1

Figure 5.1.3.11. MPC input & output weights

5.1.3.1.2. Responses:

[1] Responses after applying 2-degree disturbance in the Pitch angle:

- **Pitch Angle Response**

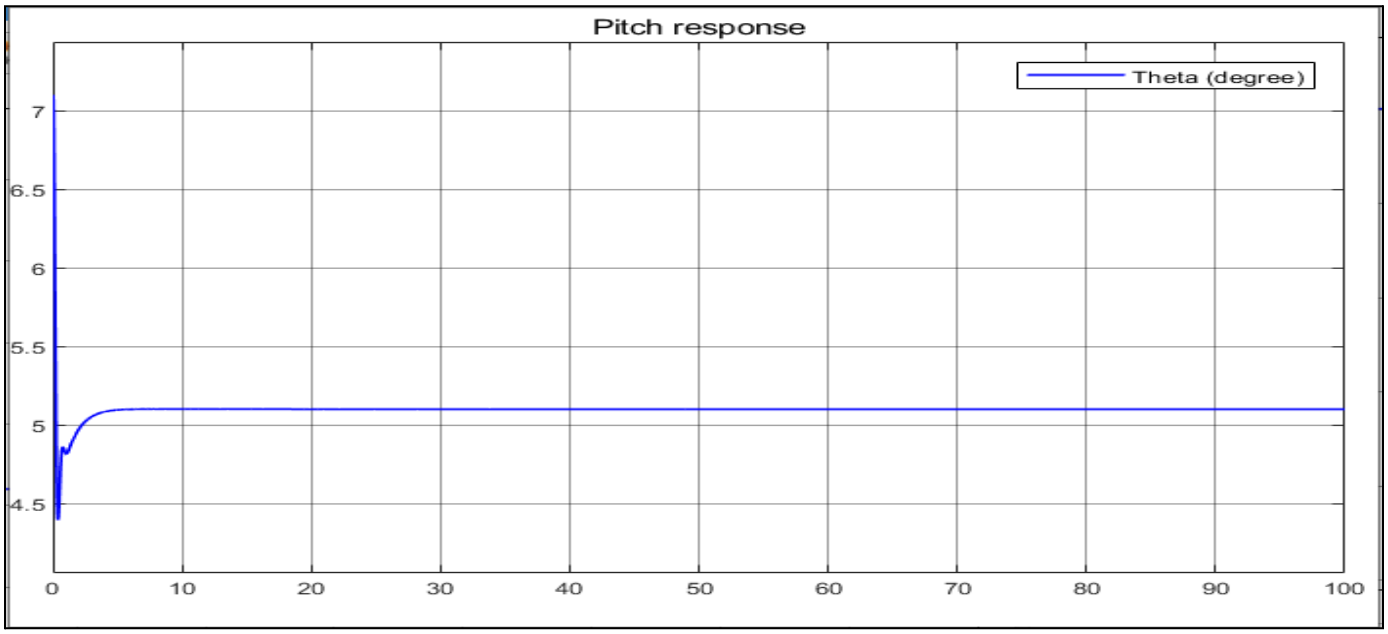


Figure 5.1.3.12. Pitch Angle Response using MPC, Elevator & Thrust Actions

- **Altitude Response**

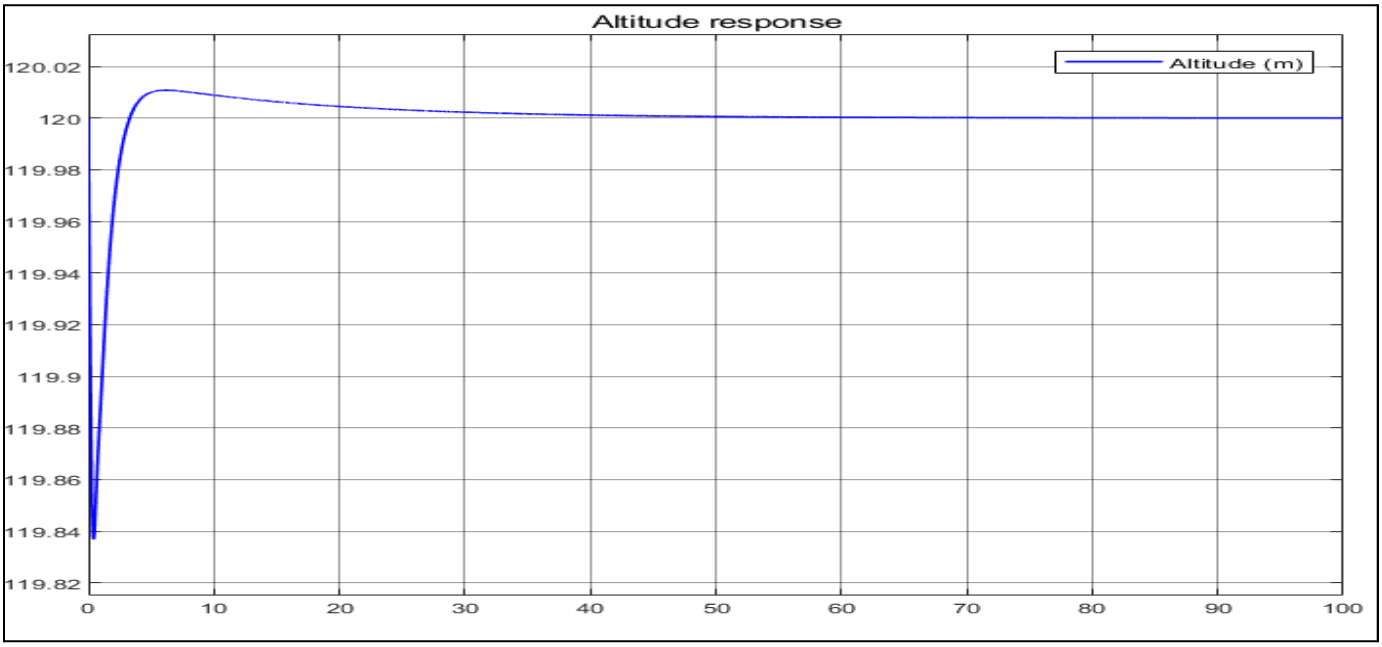


Figure 5.1.3.13..Altitude Response using MPC, Elevator & Thrust Actions

- **Velocity Response**

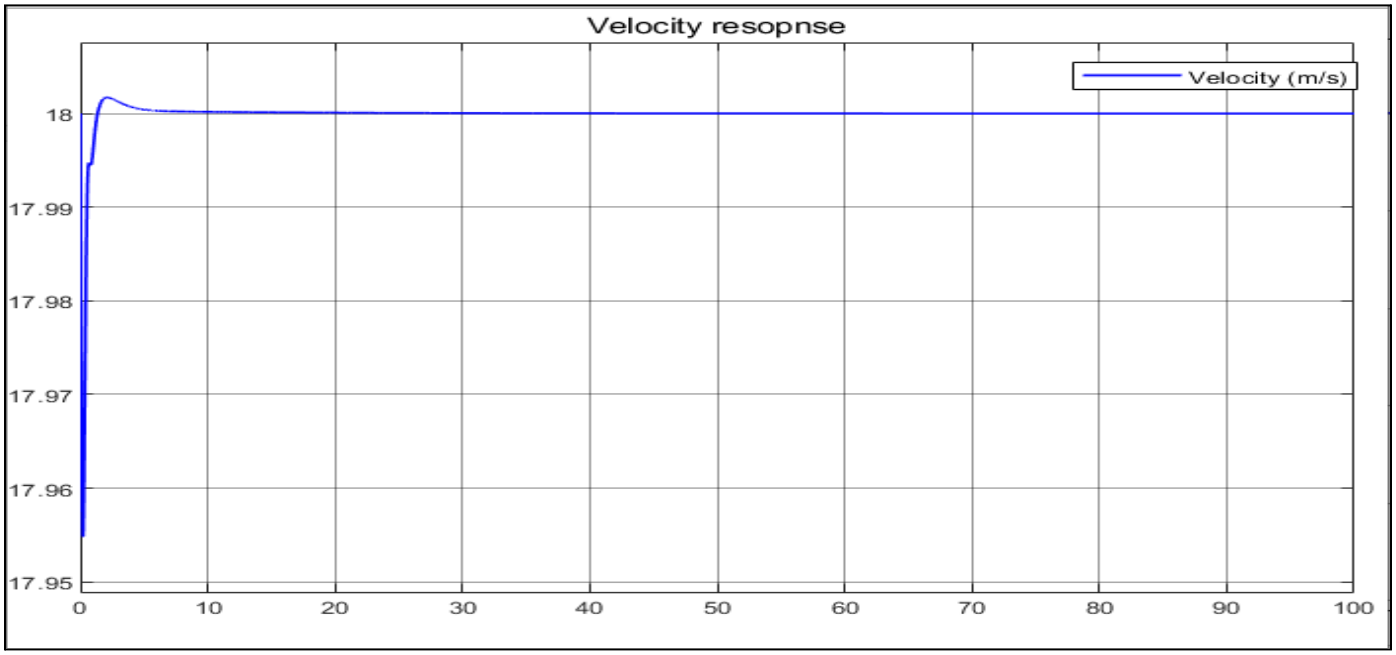


Figure 5.1.3.14.Velocity Response using MPC, Elevator & Thrust Actions

- **Thrust control Action**

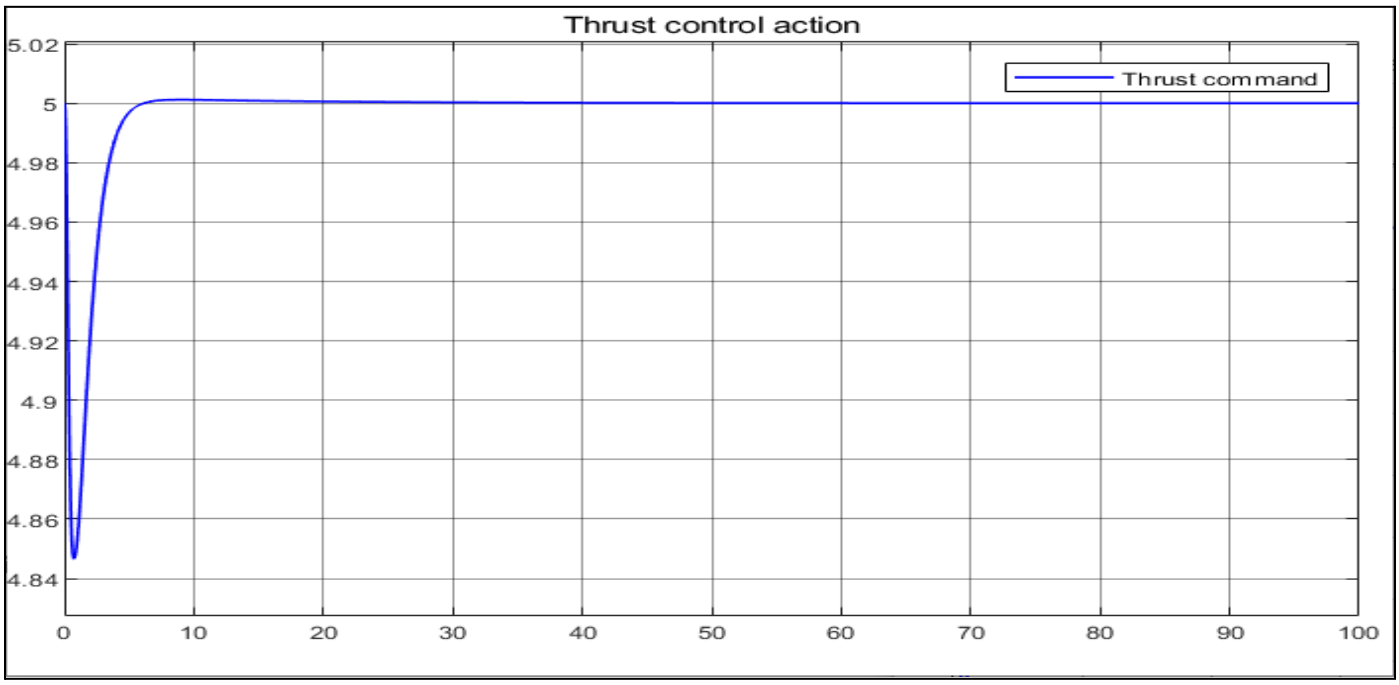


Figure 5.1.3.15. Thrust Control Action

- **Elevator control Action**

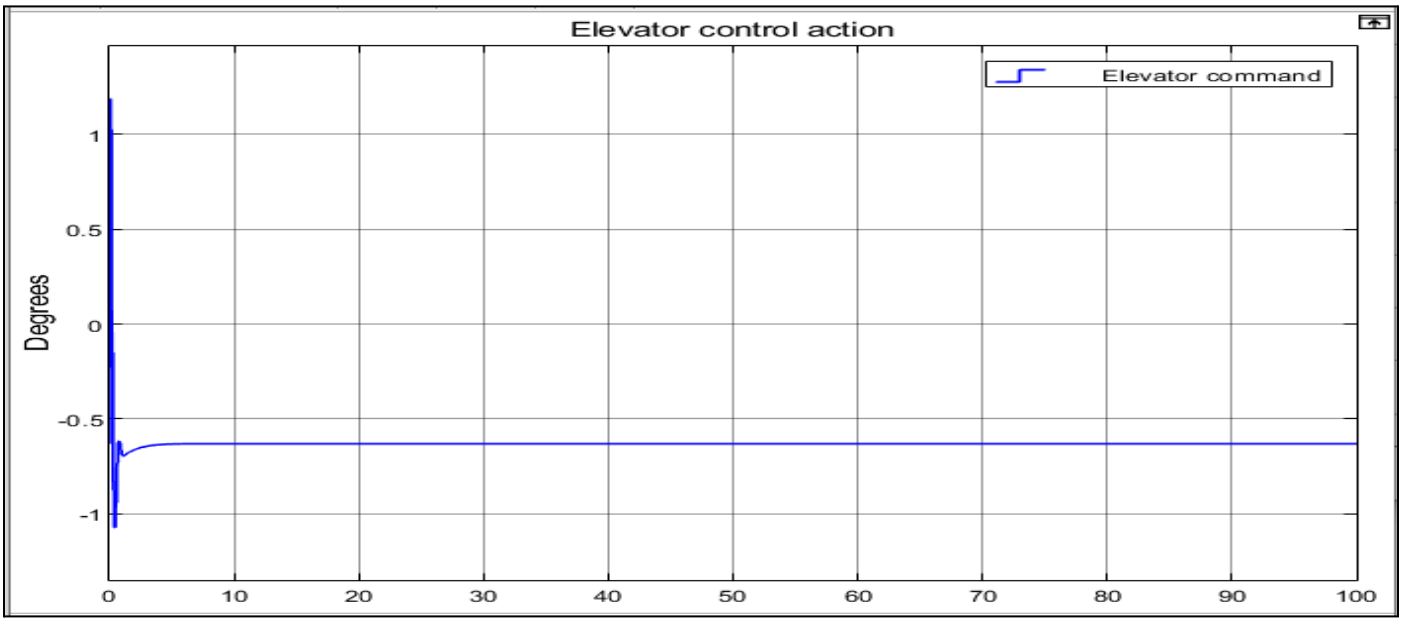


Figure 5.1.3.16. Elevator Control Action

[2] Responses during climb rate from 105m to the ref setpoint 120m:

- **Altitude Response**

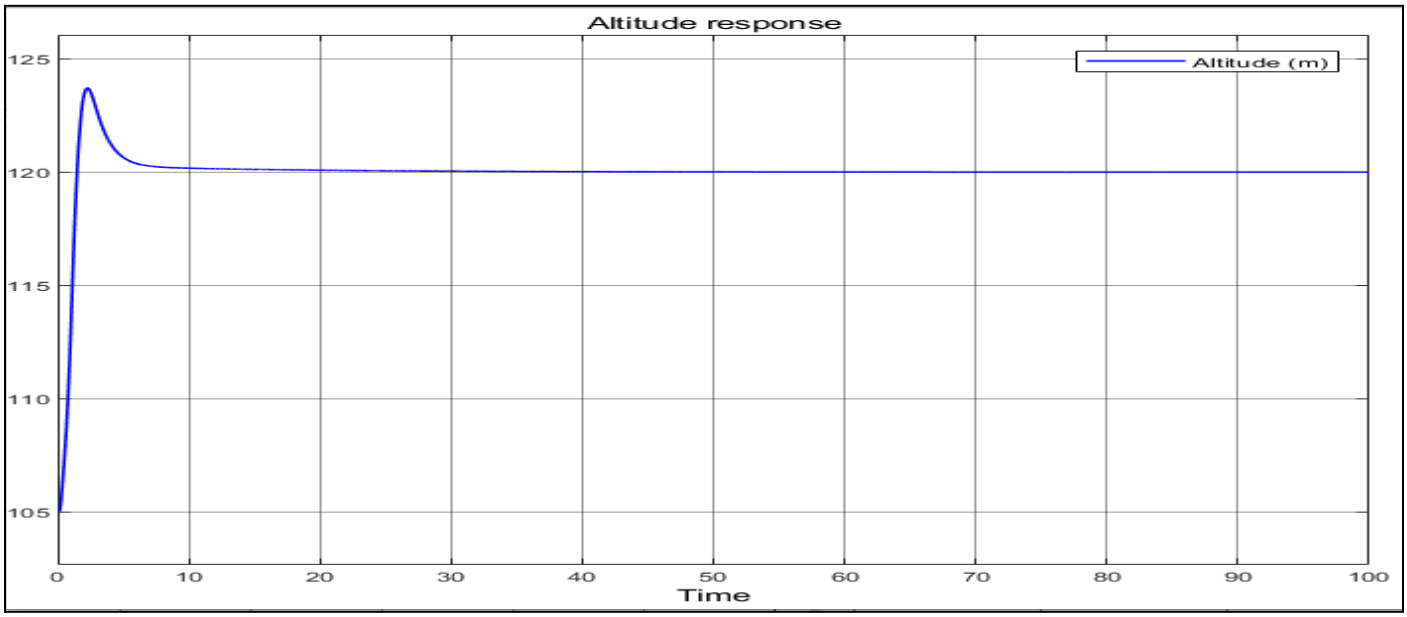


Figure 5.1.3.17. Altitude Response with climb rate using MPC, Elevator & Thrust Actions

- **Velocity Response**

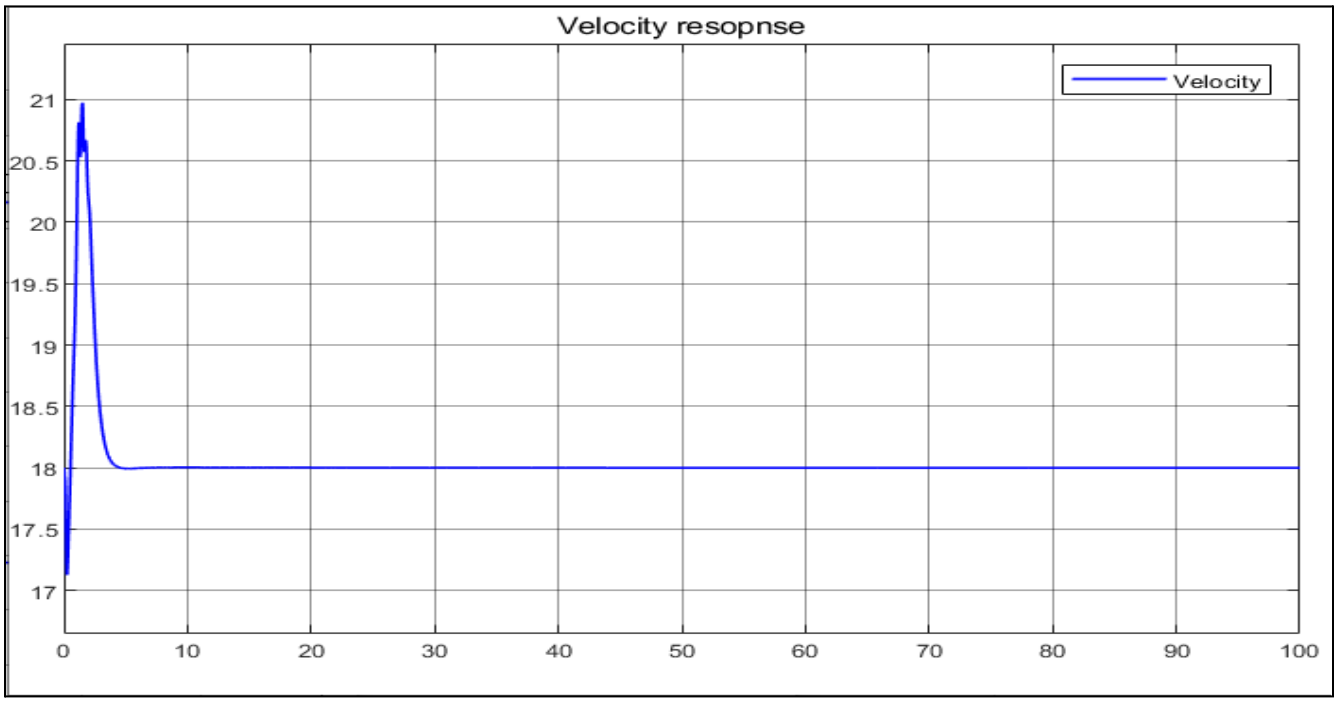


Figure 5.1.3.18.Velocity Response with climb rate using MPC, Elevator & Thrust Actions

- **Elevator control Action**

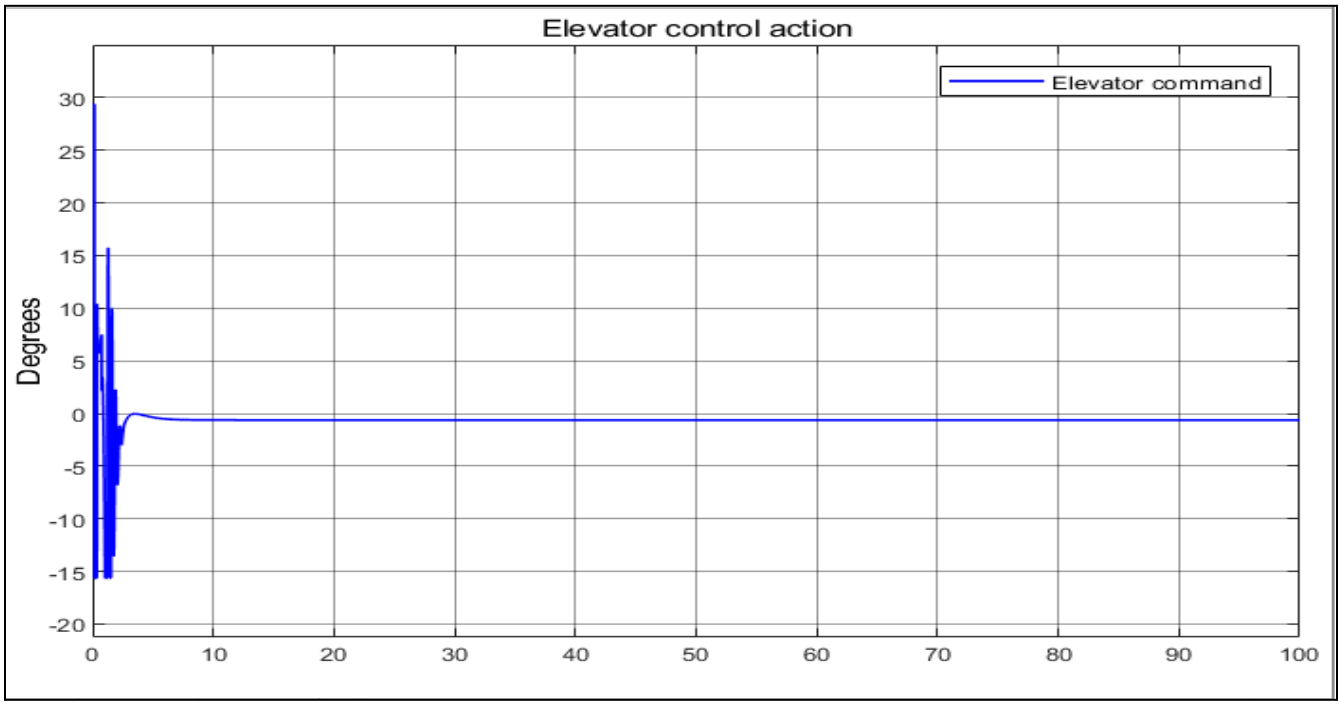


Figure 5.1.3.19. Elevator Control Action with climb rate

- **Thrust control Action**

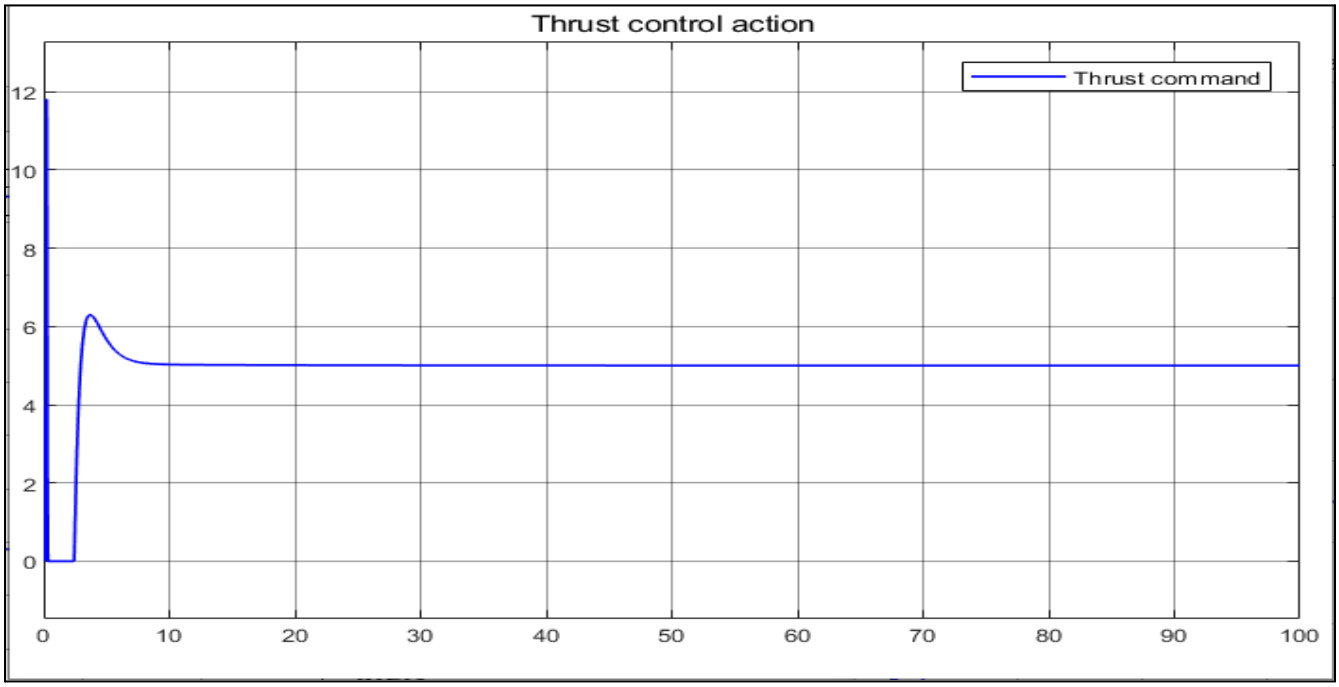


Figure 5.1.3.20. Thrust Control Action with climb rate

Analysis & Discussion:

From the above figure, The MPC works well with small disturbances in the pitch angle with the following Response specification, so does it during climb but its robustness with the elevator and thrust control surfaces increases with increasing the change in the altitude.

```
      RiseTime: 0
TransientTime: 3.1513
 SettlingTime: 2.2302
  SettlingMin: 4.3955
  SettlingMax: 7.1000
    Overshoot: 39.2161
   Undershoot: 0
         Peak: 7.1000
     PeakTime: 0.0100
```

5.1.3.2. MPC for Both Longitudinal & Lateral Models

The MPC ability to handle multiple inputs and outputs enables the MPC block on Simulink to control both the Longitudinal and Lateral modes at the same time.The roll and yaw angles both were given a 2 degree disturbance and the MPC uses both the aileron and the rudder to counteract these disturbances. The MPC is configured as follows.

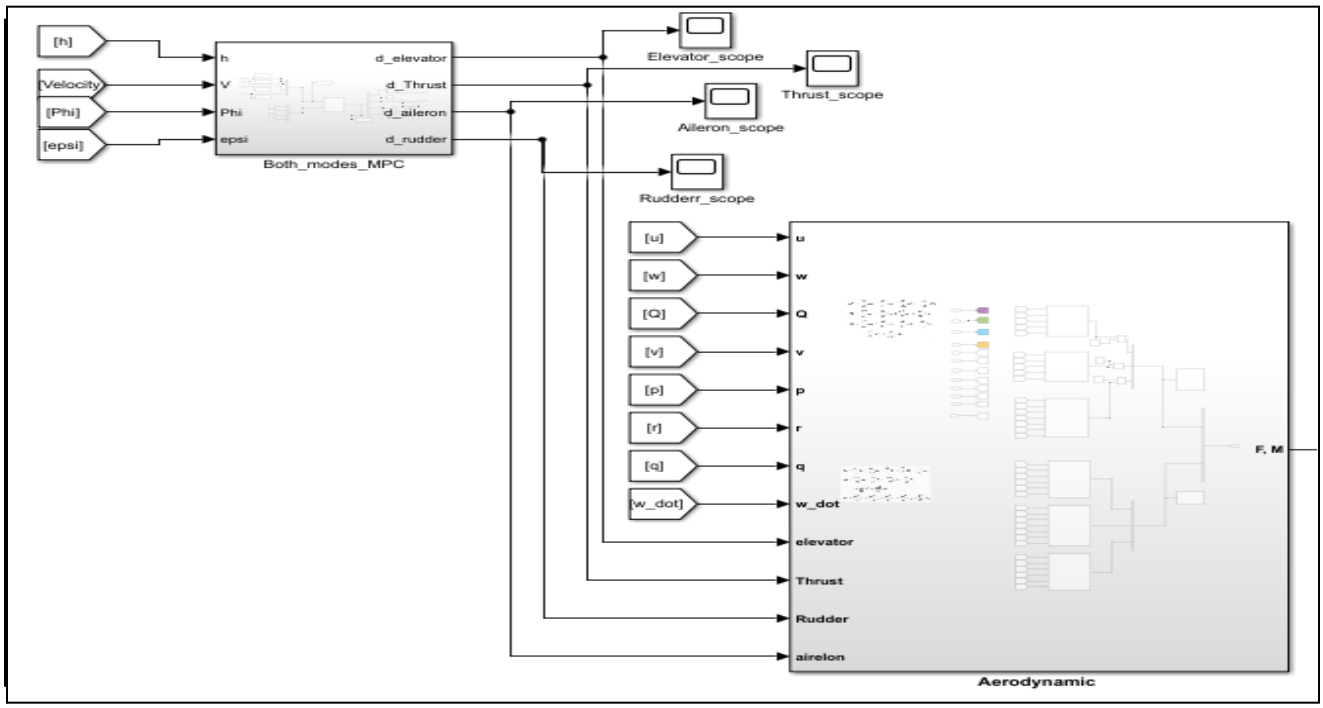


Figure 5.1.3.21.MPC configuration for Both Modes control using the 4control Actions

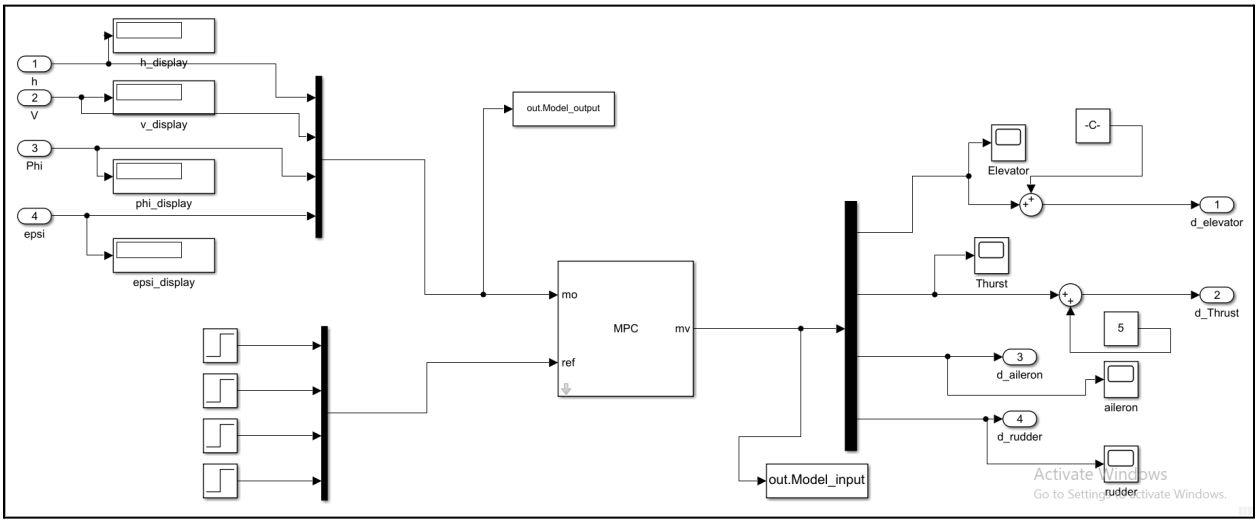


Figure 5.1.3.22. Longitudinal MPC Configuration

Tuning:

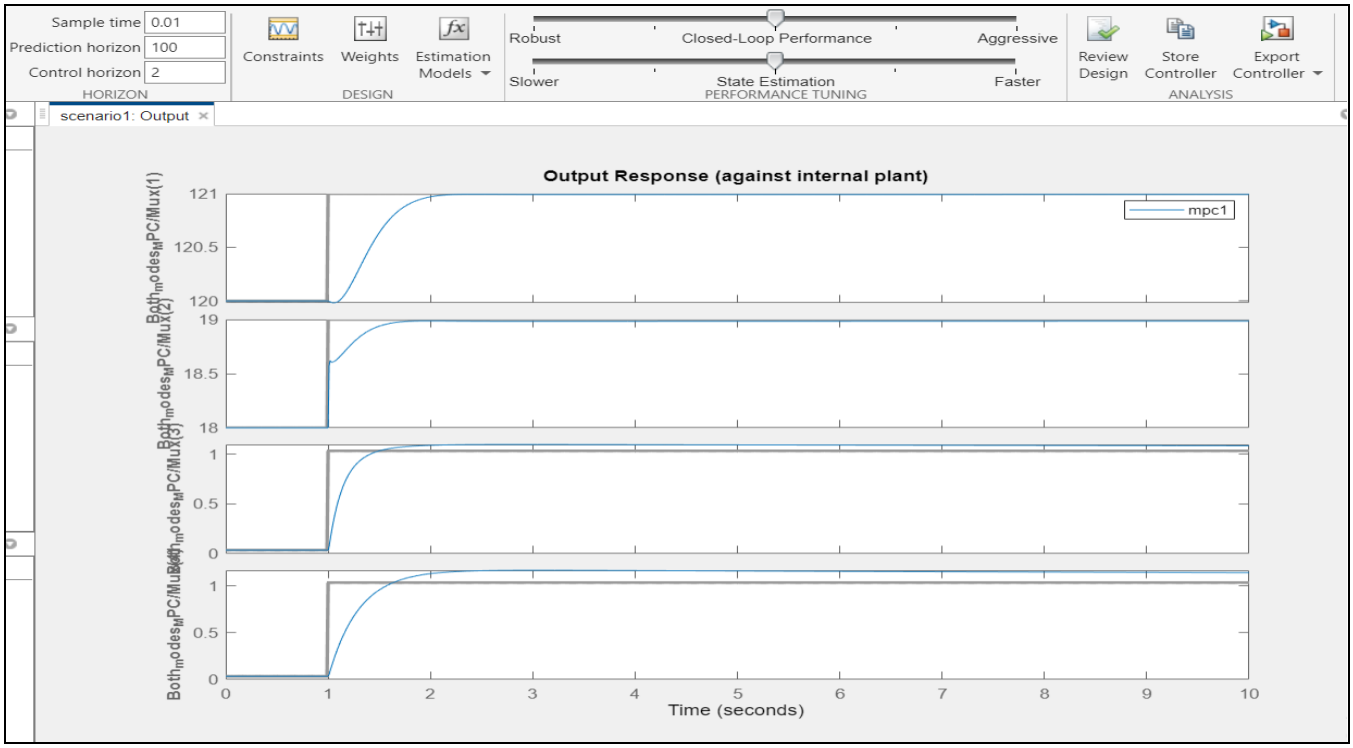


Figure 5.1.3.23. Output Response against the Commanded Response Using Longitudinal & Lateral

Responses:

- **Roll angle Response**

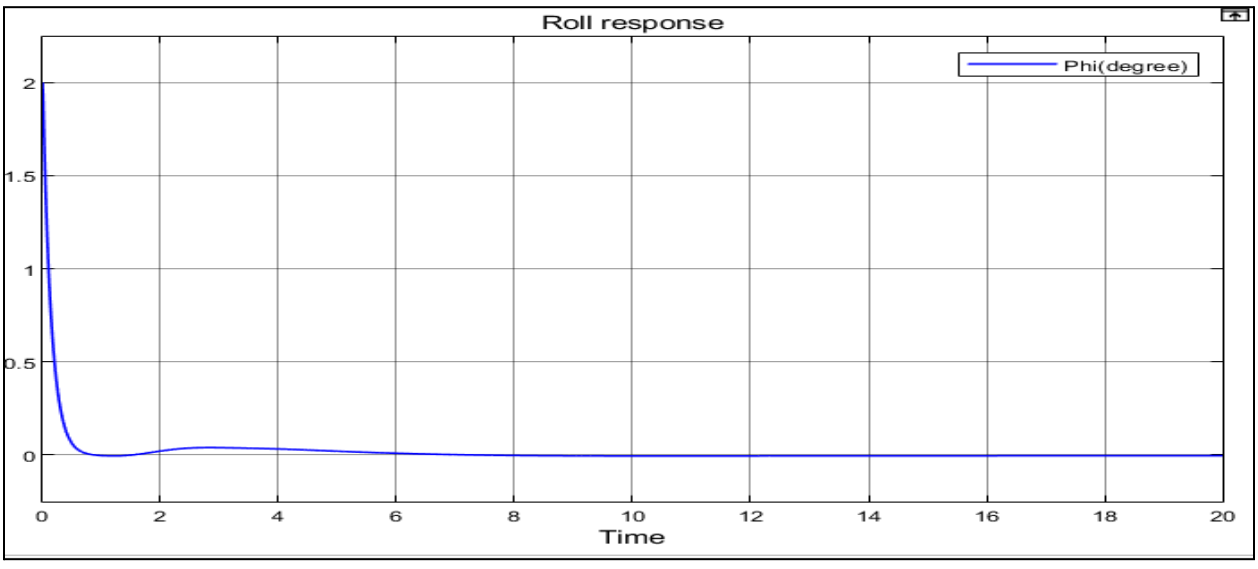


Figure 5.1.3.24. Roll angel Response Using MPC

- **Yaw angle Response**

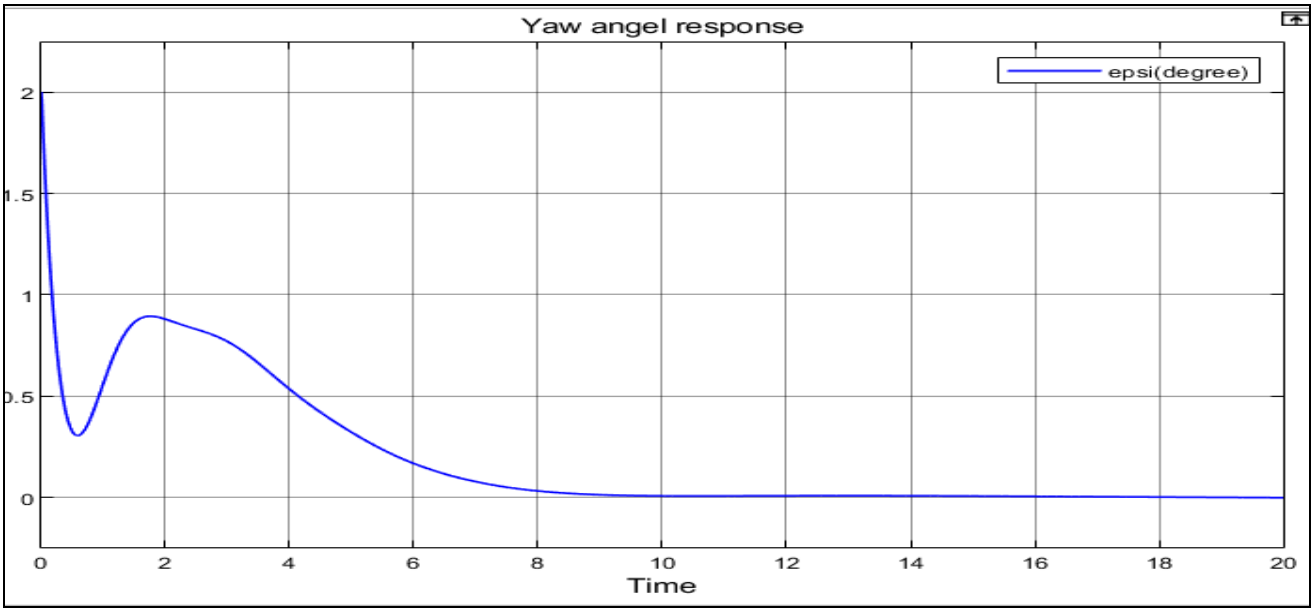


Figure 5.1.3.25. Yaw angel Response Using MPC

- **Aileron Control Action**

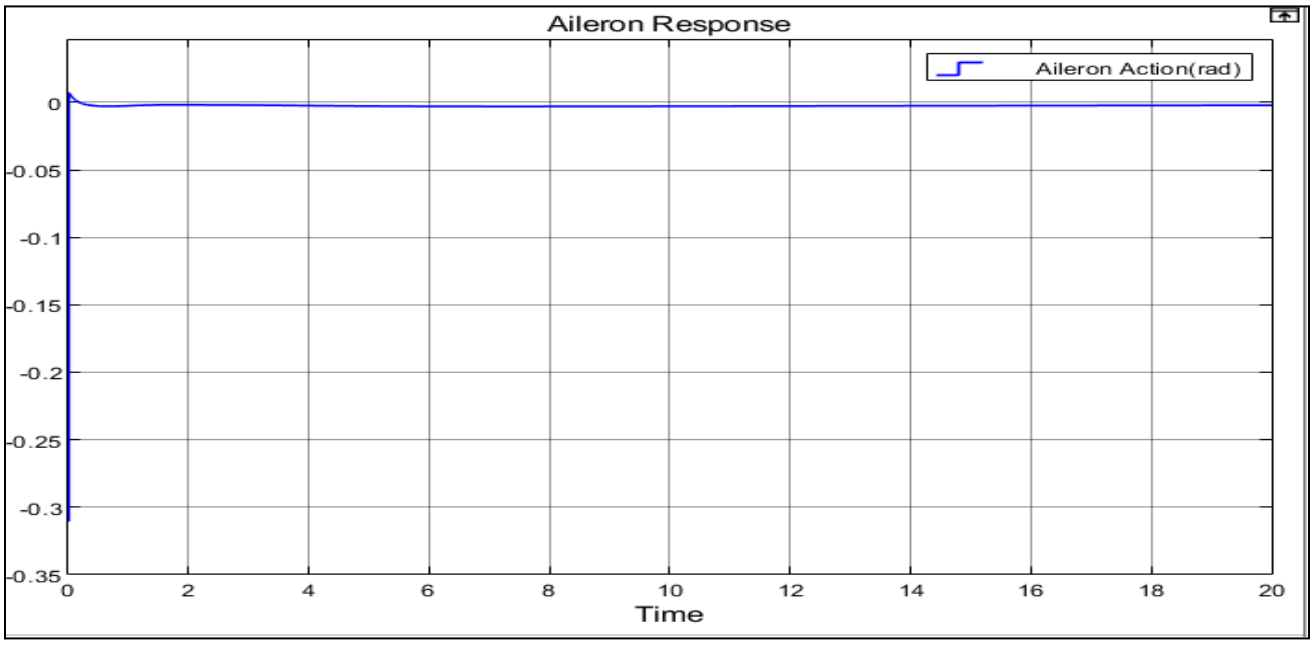


Figure 5.1.3.26.Aileron Control Action

- **Rudder Control Action**

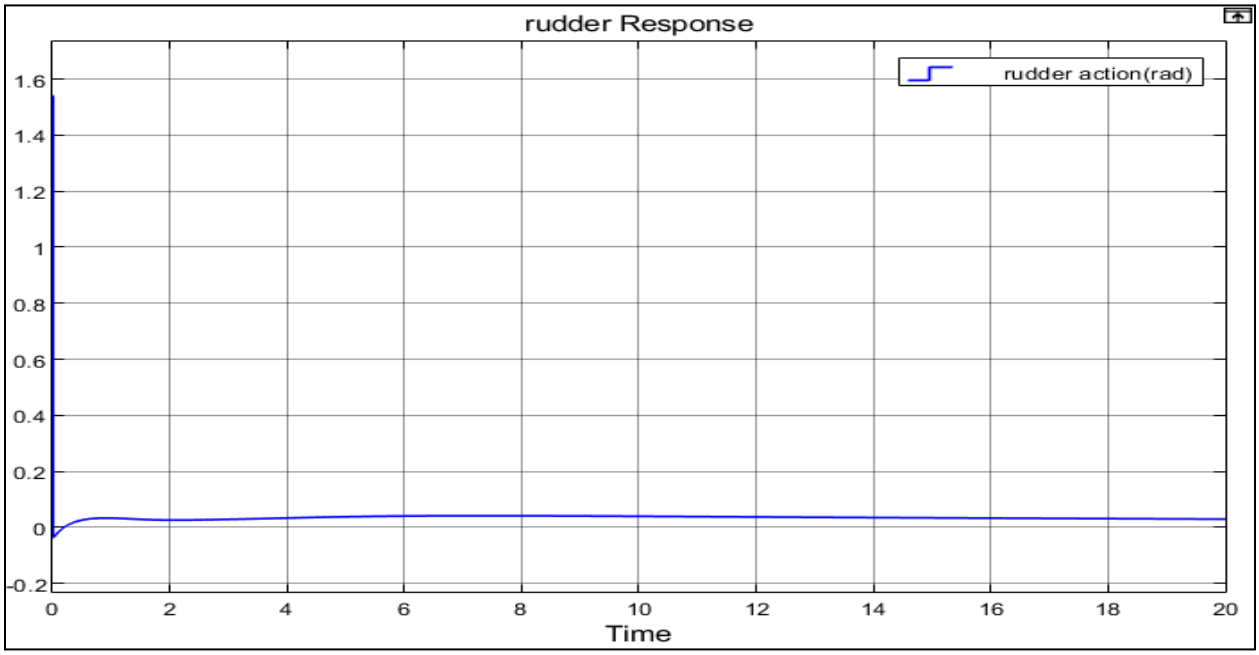


Figure 5.1.3.27.Aileron Control Action

5.1.4. Fuzzy Logic Controller

FLC (fuzzy logic controller) is a fast-developed technology over the last few years. In this study, FLC has been applied for controlling the pitch angle of the UAV. FLC is a method that was initially developed for sorting and handling data, but it has proven to be an excellent choice for a variety of control system applications due to its ability to handle nonlinear systems and complex mathematical computations. FLC uses an imprecise but descriptive language to process input data in a more human-like manner. It incorporates a simple rule-based "If X and Y then Z" approach to solving control problems rather than attempting to model a system mathematically.

Generally, a fuzzy control system consists of three basic sub-groups: fuzzification, fuzzy inference engine (decision logic), and defuzzification. The block diagram of FLC is illustrated in Figure 5.1.4.1. The first block shown is fuzzification. It converts input data into fuzzy membership.

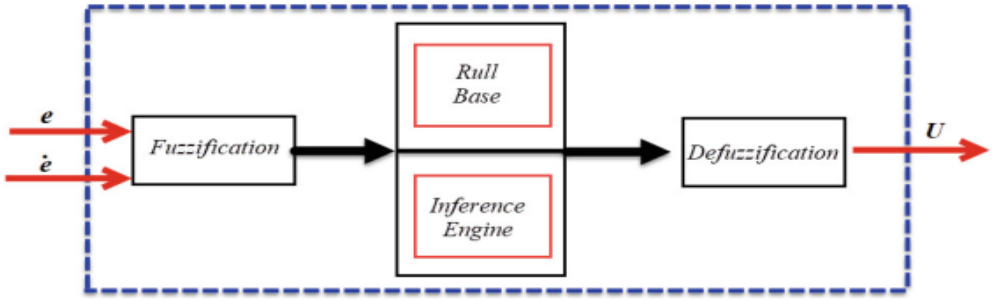


Figure 5.1.4.1. Structure of Fuzzy Logic Controller

The rule base and the inference engine performs as human decision-making, which depends on fuzzy rules from inference. The membership functions (MFs) as well as fuzzy control rules affect control performance substantially. The third block is defuzzification where results from the fuzzy set are defuzzified or converted into a control signal. The most frequent approach of the fuzzy inference system, the Mamdani model, is applied for inferring as shown in Figure 5.1.4.2.

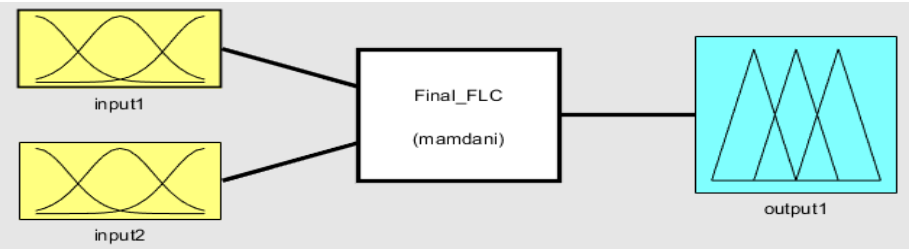


Figure 5.1.4.2. Mamdani fuzzy inference block

The Simulink model of the FLC with the pitch control for the aircraft is shown in Figures 5.1.4.3 and 5.1.4.4. The inputs to the fuzzy controller are the error (e) and the rate at which the error changes (de), whereas the output is the change of the control signal (du). From the figure, the error (e) is computed by comparing the reference point (desired angle) with the plant output. The change of error (de) is generated by the derivation of the error. The error and change of error is fed to the fuzzy controller through a multiplexer.

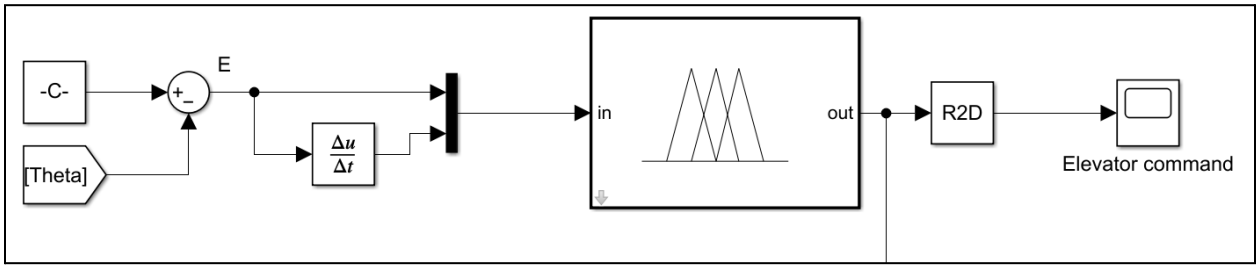


Figure 5.1.4.3. The fuzzy logic controller in a feedback loop of the pitch control system.

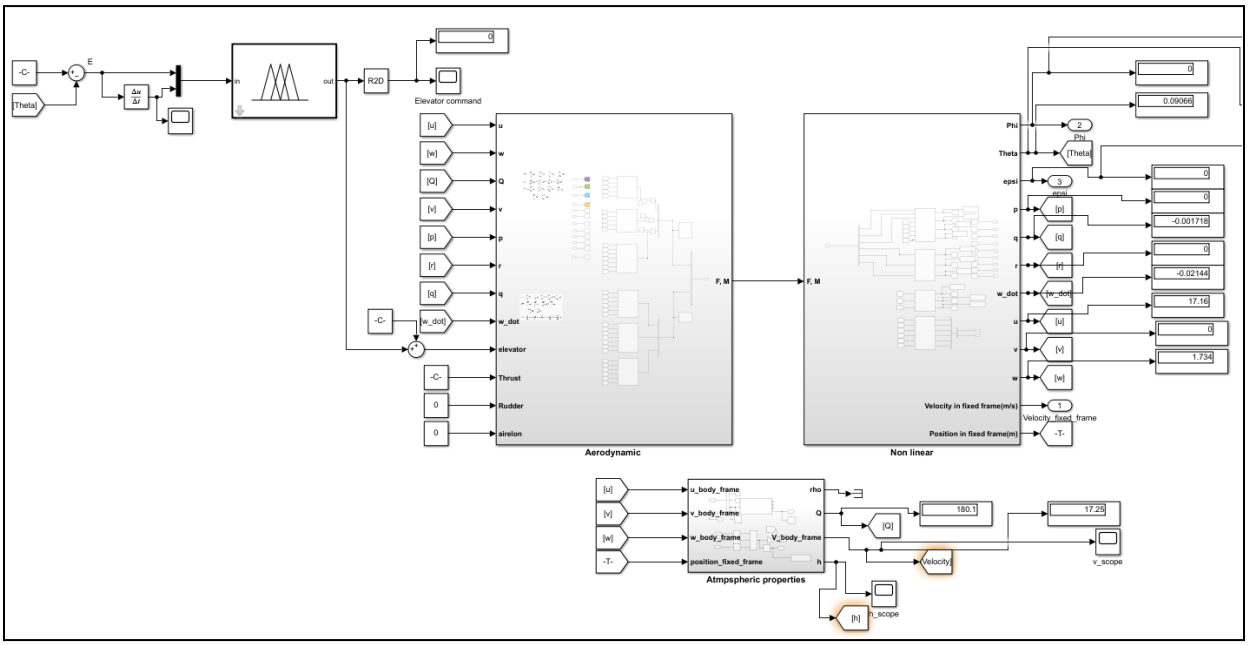


Figure 5.1.4.4.Integrating the FLC Controller into the Simulink Model.

Fuzzification involves the conversion of the input and output signal into a number of fuzzy represented values (fuzzy set). Each fuzzy set consists of three types of membership function, which is negative (N), zero (Z), and positive (P). The MFs of pitch angle and derivation of pitch angle are illustrated in Figure 5.1.4.5 respectively. FLC output and fuzzy control surface are

shown in Figure 5.1.4.6 The triangle membership is used for both input and output, so each membership function has three parameters (a modal point and two half-width). The obtained fuzzy set has to be transformed to the signal in such a way that it is sent to the control input. The centroid of the area is used as the defuzzification method.

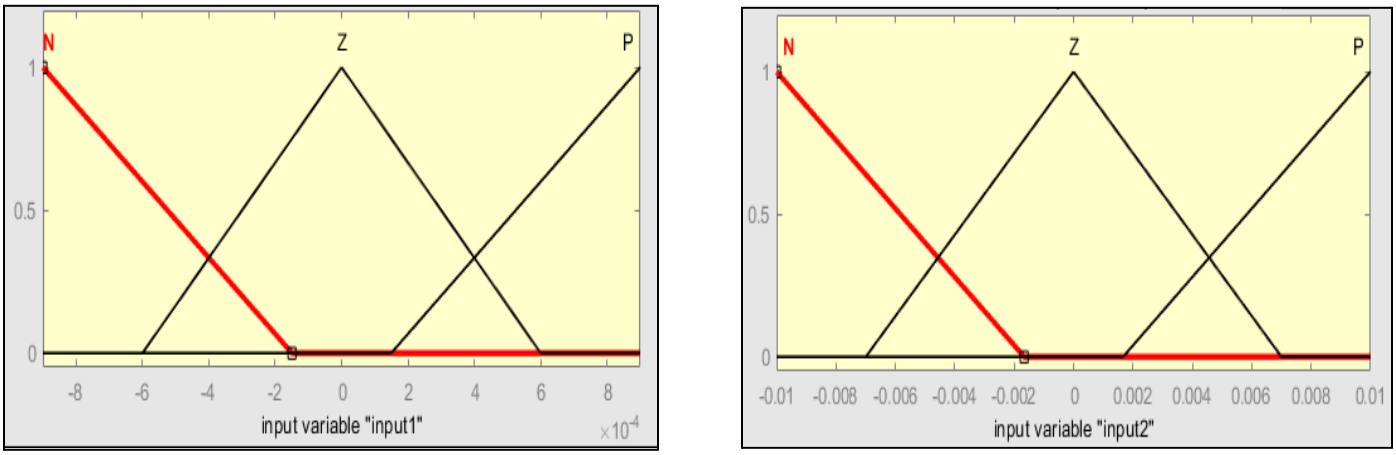


Figure 5.1.4.5. Membership functions of the linguistic variables for E(input1), dE(input2).

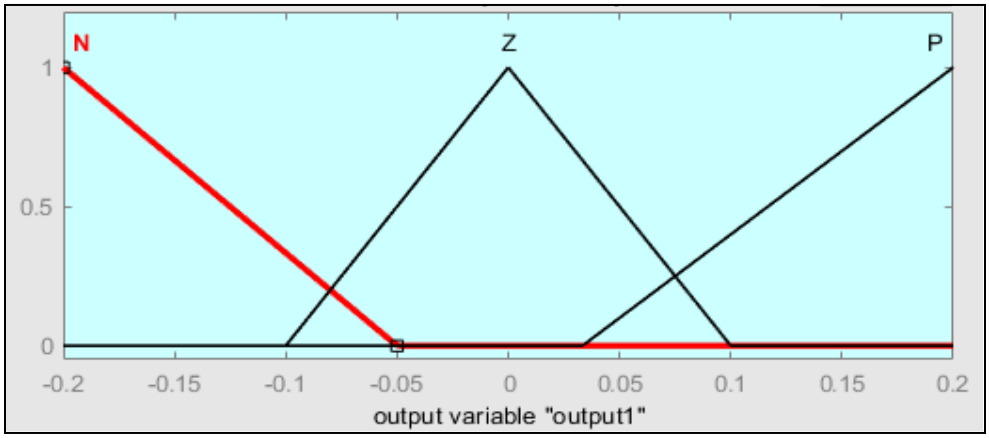


Figure 5.1.4.6. Membership functions of the linguistic variables for dElevator(output).

Since there are a total of three fuzzy variables (two inputs and one output), and each fuzzy variable has three membership functions, the fuzzy controller for pitch control of an aircraft has a total of nine membership functions. The expert knowledge is implemented in the form of an IF-THEN rule structure.

These are nine rules that have been utilized in designing the controller and the rule is defined in Table 5.1.4.1(Wahid, 2020).

	Error, e	Delta error. de	Delta u, du
1	N	N	N
2	N	Z	N
3	N	P	N
4	Z	N	N
5	Z	Z	Z
6	Z	P	P
7	P	N	P
8	P	Z	P
9	P	P	P

Table 5.1.4.1 Rules for the fuzzy controller.

The Response of the pitch angle due to a 2-degree disturbance:

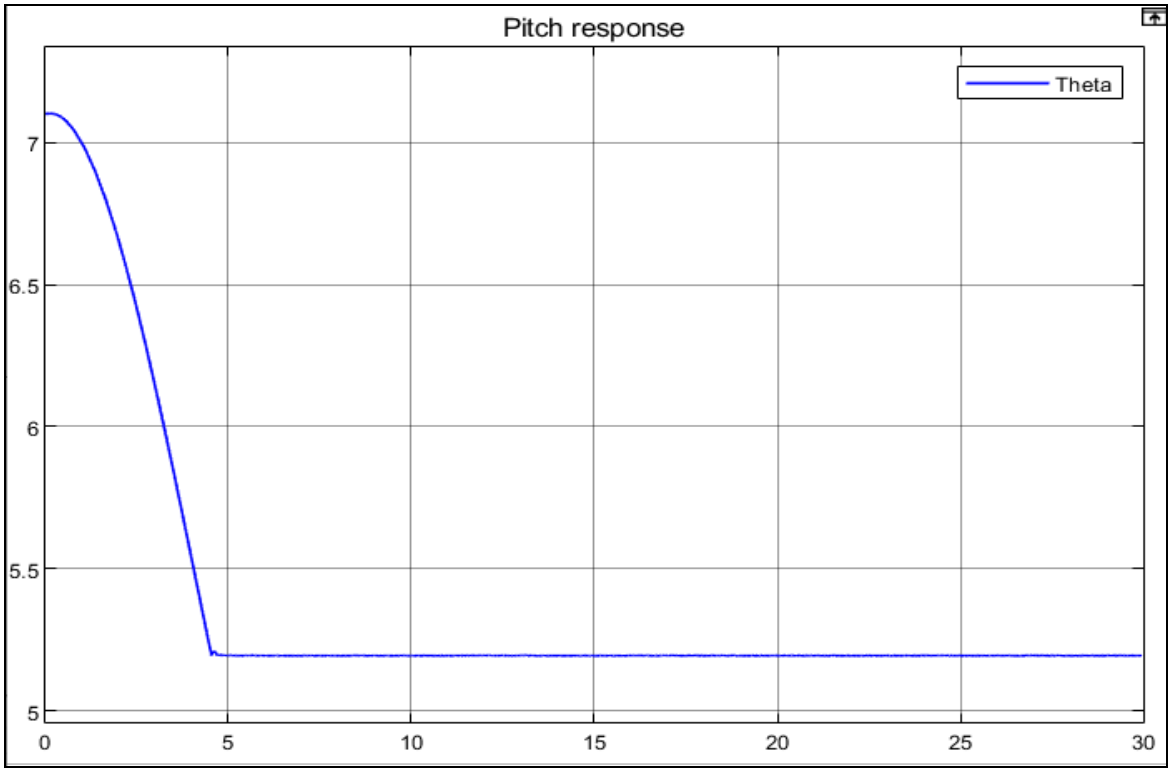


Figure 5.1.4.7. Pitch angle Response using FLC

From the above response shows that FLC has a very small overshoot and an acceptable settling time. However, its computational cost is very high.

5.1.5. Comparison between MPC & TECS

- Pitch angle Response due to 2-degree disturbance in the pitch angle

```
     RiseTime: 0
TransientTime: 7.5855
 SettlingTime: 20.2314
  SettlingMin: 4.7848
  SettlingMax: 24.3021
    Overshoot: 376.5126
   Undershoot: 0
         Peak: 24.3021
     PeakTime: 0.7000
```

Figure 5.1.5.1. Pitch Angle Response using Tecs

```
     RiseTime: 0
TransientTime: 3.1513
 SettlingTime: 2.2302
  SettlingMin: 4.3955
  SettlingMax: 7.1000
    Overshoot: 39.2161
   Undershoot: 0
         Peak: 7.1000
     PeakTime: 0.0100
```

Figure 5.1.5.2. Pitch Angle Response using MPC

- Altitude Response due to climb from 118m to 120m

```
     RiseTime: 0
TransientTime: 30.9725
 SettlingTime: 0
  SettlingMin: 118
  SettlingMax: 120.1416
    Overshoot: 0.1180
   Undershoot: 0
         Peak: 120.1416
     PeakTime: 5.6000
```

Figure 5.1.5.3. Altitude Response using Tecs

```
     RiseTime: 0
TransientTime: 23.3503
 SettlingTime: 0
  SettlingMin: 117.9451
  SettlingMax: 120.1287
    Overshoot: 0.1073
   Undershoot: 0
         Peak: 120.1287
     PeakTime: 5.6152
```

Figure 5.1.5.4. Altitude Response using MPC

From the above results it shows that the MPC works as a more efficient controller than TECS (with the GA gains) in the longitudinal mode, since its Response specifications are way better.

5.2. Quadrotor Controllers

5.2.1. Nonlinear Model Predictive Control

Nonlinear Model Predictive Control (NMPC) is a powerful control strategy that optimizes the performance of a nonlinear model subject to constraints over a finite prediction horizon. It combines system modeling, cost function formulation, optimization, and feedback to compute the optimal control inputs that drive the system toward desired behavior while satisfying constraints. Unlike linear control techniques, NMPC can handle systems with nonlinearity, constraints, and uncertainties, making it suitable for a wide range of applications.

In this study, NMPC is utilized to control the quadrotor motor speeds to follow a certain trajectory and to give the system a margin of noise immunity. NMPC is used in two cases; firstly by taking the 3 position parameters with their rates and the 3 angle parameters with their rates (total of 12 states) as output variables, and secondly by taking the 3 positions and the 3 angles without any rates (total of 6 states).

5.2.1.1. NMPC Setup and Tuning

NMPC require a different treatment than the linear MPC, NMPC requires the prediction model state function and it is advised and considered as best practice to give an analytical Jacobian formula for the prediction model as it significantly enhances the quality and accuracy of the responses and the applied control actions of the NMPC [1][2]. The prediction model state function and its Jacobian function can be specified and passed to the NMPC by their handle.

Before completing the rest of the NMPC setup parameters, the hovering speed is evaluated to be passed as the nominal control target to the NMPC. the hovering speed is evaluated by equating the quadrotor's total thrust with its weight and then solving for the motor hover speed.

$$T = k(u_1 + u_2 + u_3 + u_4) = mg$$

$$4u = 24 * 9.81$$

$$u = 58.86\, rpm^2$$

Back to the NMPC setup parameters, a prediction horizon of 10 to 20 steps is stable for NMPC to capture the hover full dynamics for small UAVs [1]. After several trials, a prediction of 18 steps with 2 steps of control action was found to be the most appropriate for the study of this

specific UAV. Since the hover speed was found to be $58.86\ rpm^2$, the control inputs constraints is set to be [0,70] to give the NMPC some flexibility in applying the appropriate action. As for the control action rates, they are set to [-2,2] to prevent aggressive motor responses that might harm the motors. For the cost function, the standard quadratic cost function is used for the NMPC as it is very suitable for reference tracking and noise reduction [1]. As for the output variables and manipulated variables weights, they are all kept to their default values of 1 for the output variables and 0.1 for the manipulated variables and their rates as well. A manipulated variables target is set to the hovering value of 58.86, and then the model was simulated to see NMPC in action using the ode45 solver and a fixed time step of 0.1s.

5.2.1.2. Results

For the first case where the parameter rates are not passed to the NMPC, the states' responses are delayed by almost 1.33s from the reference trajectory as shown in Fig.5.2.1.

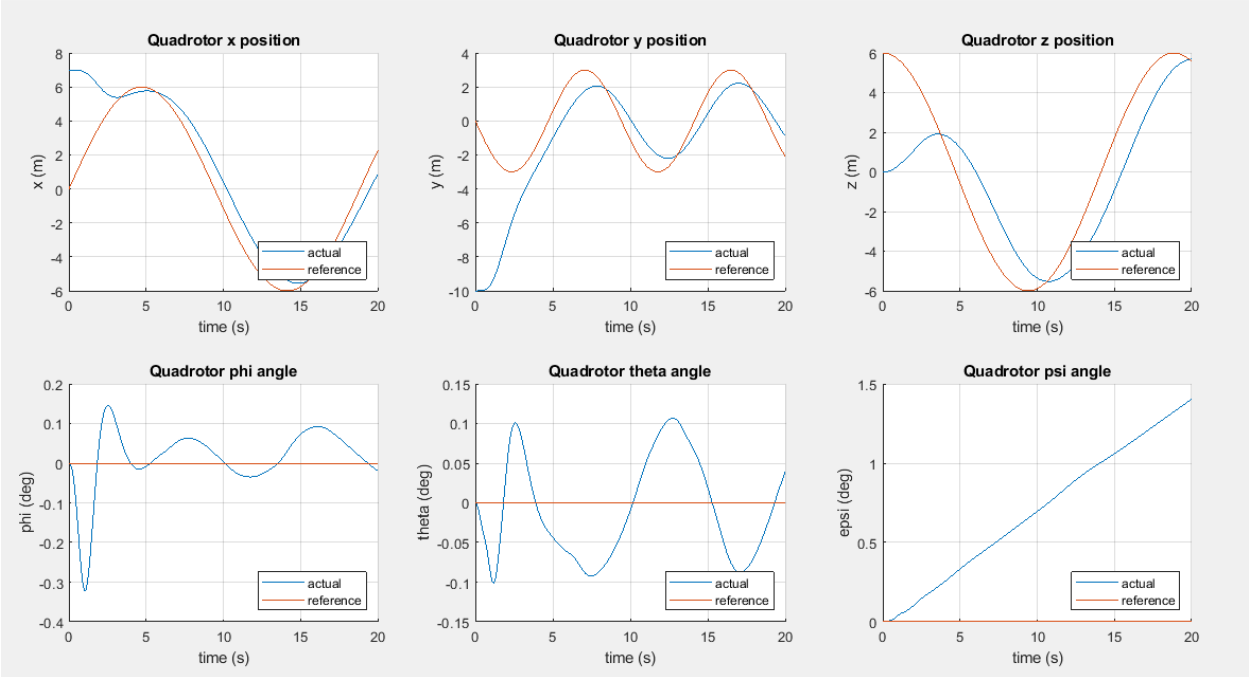


Fig.5.2.1. The states' responses for 6 states NMPC.

As for the control actions, the response is not observed to have any aggressive actions, and is kept to the minimum as shown in Fig.5.2.2.

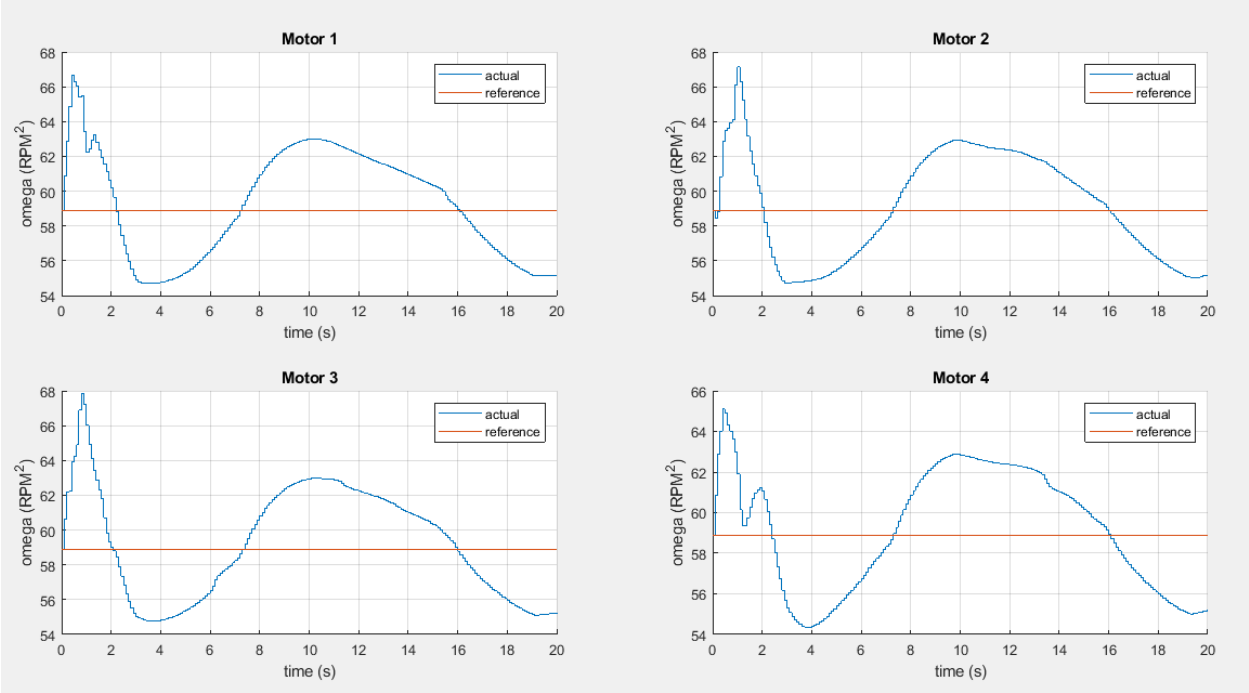


Fig. 5.2.2. The control actions responses for 6 states NMPC.

As for the second case where the parameters are utilized alongside their rates, the states' response is found to be significantly better but with an overshoot in the z position of the quadrotor of 19.32% and a minor overshoot in the y position of 4.67% (see Fig.5.2.3) due to the relatively high moments of inertia and mass of the whole VTOL.

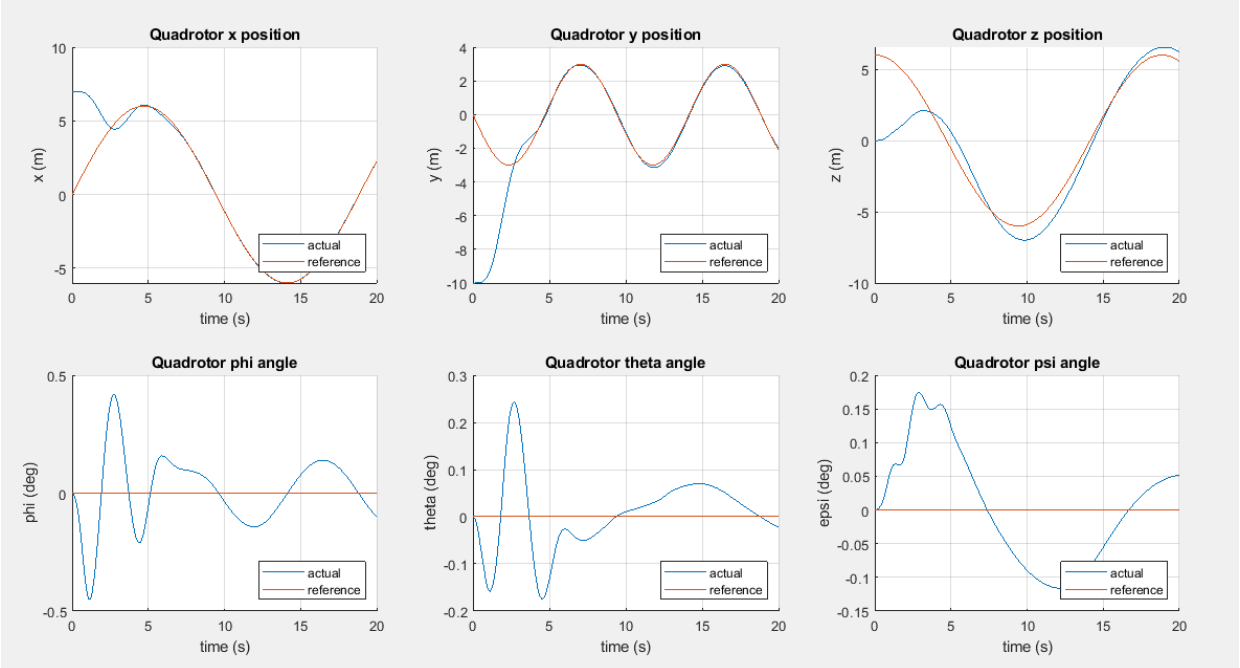


Fig. 5.2.3. The states' responses for 12 states NMPC.

Regarding the control actions, a slightly higher control action is applied than that of the previous case to prevent the 1.33s delay that is observed in the previous case, as indicated in fig.5.2.4.

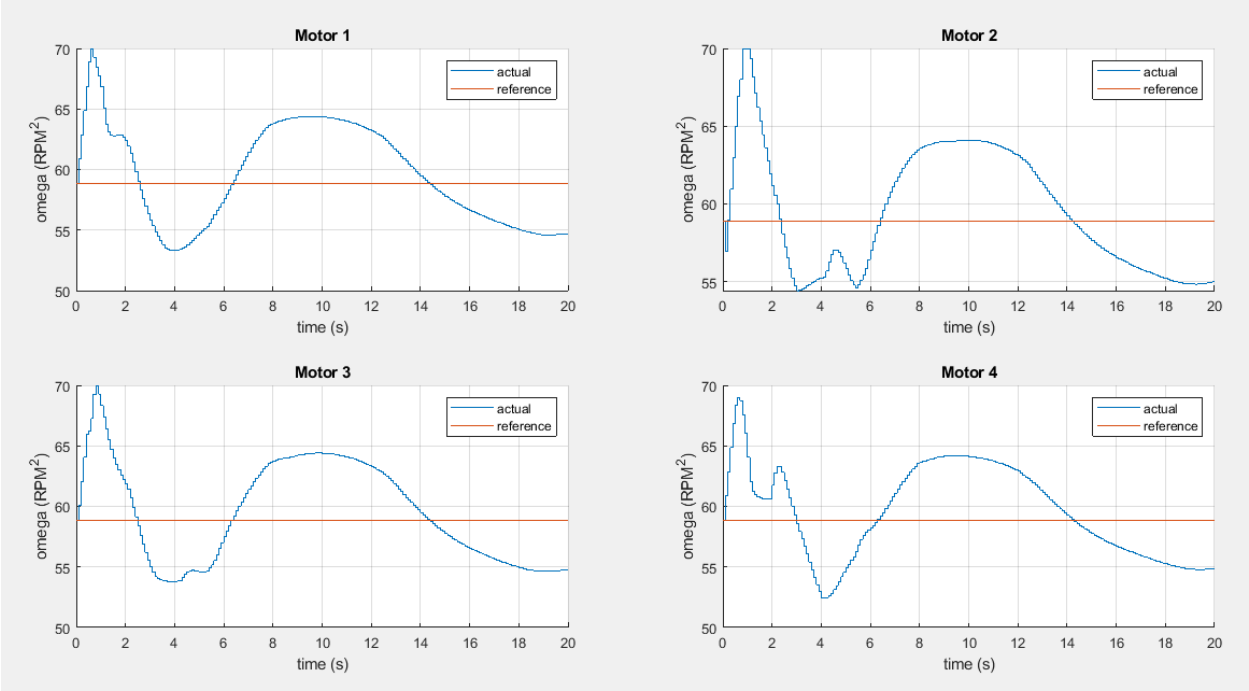


Fig. 5.2.4. The control actions responses for 12 states NMPC.

Overall, the NMPC gives significantly better control in the case of utilizing the model parameters alongside their rates and shows a good command of the quadrotor attitude response.

5.2.2. Proportional-Integral-Derivative Controller

PID control is a simple yet powerful method of controlling system processes. Provided the error signal as input ($e(t)$), which is the difference between the desired value and actual value of the process, the PID controller manipulates the error signal by evaluating three terms; proportional, integral, and derivative. The proportional term is calculated by multiplying the error signal by a proportional gain (k_P). As for the integral term, the error signal is integrated and multiplied by an integral gain (k_i). Finally, the derivative term is calculated by differentiating the error signal and multiplying it by a derivative gain (k_D). Thereafter, the terms are added together creating the control action $u(t)$ of the PID controller, as shown below:

$$\boldsymbol{u(t) = k_P * e(t) + k_i * \int_0^t e(\tau)\, d\tau + k_D * \frac{d}{dt} e(t)}$$

Each of these terms contributes to finding a suitable control action for the process. The proportional term is proportional to the error signal. It applies a corrective action based on the magnitude of the error. It helps the controller to respond quickly to quick variations in the error. But, on the other hand, it could lead to overshoot and steady-state errors. The integral term keeps track of the accumulated past errors, it provides a control action that gradually reduces the accumulated error, which helps in eliminating offset in controlled variables. Finally, the derivative term makes use of the error rate of change to anticipate future trends and dampen rapid changes and oscillations.

The following table summarizes the effect of PID parameters on signal performance characteristics; rise time, overshoot, settling time, and steady-state error.

Parameter	Rise Time	Overshoot	Settling Time	S-S Error
k_P	Decrease	Increase	Small Change	Decrease
k_i	Decrease	Increase	Increase	Decrease
k_D	Small Change	Decrease	Decrease	No change

Table 5.2.1. *PID Parameters Effect on Performance Characteristics (Nise, 2019).*

Performance Characteristics

Rise Time: Represents the time taken to reach around 90-95% of the desired setpoint. A smaller rise time indicates a fast response.

Overshoot: Represents the maximum deviation from the desired setpoint. It's mainly caused due to aggressive control actions.

Settling Time: Represents the time taken by the signal to reach and settle within an acceptable range of the setpoint and remains there indefinitely. It provides an indication of how quickly the system reaches a stable state after a change in the setpoint or disturbance.

Steady-state Error: Represents the deviation that remains after the response reaches its steady state. It provides a measure of the difference between the desired setpoint and the current final value of the controlled variable.

The following figure illustrates the performance characteristics for better visualization.

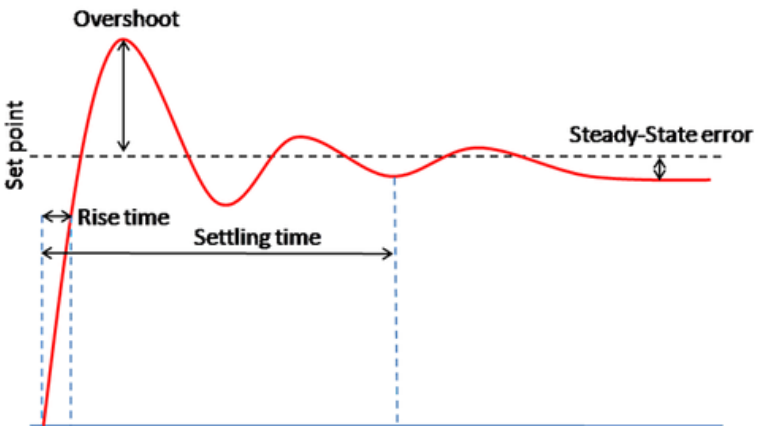


Figure 5.2.5. Performance Characteristics (WPI , 2021).

5.2.2.1 Cascaded PID Controller Architecture

A quadrotor is a highly nonlinear and underactuated system, which means that it has fewer control inputs (actuators) than the degrees of freedom that should be controlled. Namely, a quadrotor has only 4 actuators, meanwhile, it has 6 degrees of freedom, both translational and rotational.

A cascaded PID controller consists of several PID controllers that are arranged hierarchically, where each controller is responsible for controlling one of the system states. This arrangement in essence results in two control loops; outer control loop (position controller) and inner control loop (attitude controller).

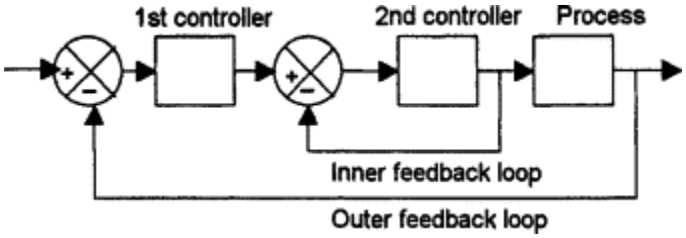


Figure 5.2.6. Cascaded PID Controller (Bolton, 2021).

Due to the decoupled nature of the system dynamics, the system states could be arranged such that the secondary process variables, that influence the primary process variables, are placed in the outer control loop. On the other hand, the primary process variables are placed in the inner control loop. In that sense, all the 6 degrees of freedom of the quadrotor are all controlled with only 4 actuator inputs.

The outer control loop is responsible for controlling the position (x & y) and altitude (z) of the quadrotor. The input to the states is the error signal computed as the difference between the

desired value and the current value (provided by the state estimator). The output of the outer loop serves as a setpoint to the inner controller in addition to the thrust command.

As for the inner control loop, it is responsible for controlling the quadrotor orientation (attitude) in terms of roll, pitch, and yaw. The output of the inner control loop serves as actuator commands.

Simulink Model

The closed control loop is shown below, where the cascaded PID controller is implemented resulting in two control loops; inner (attitude) and outer (position).

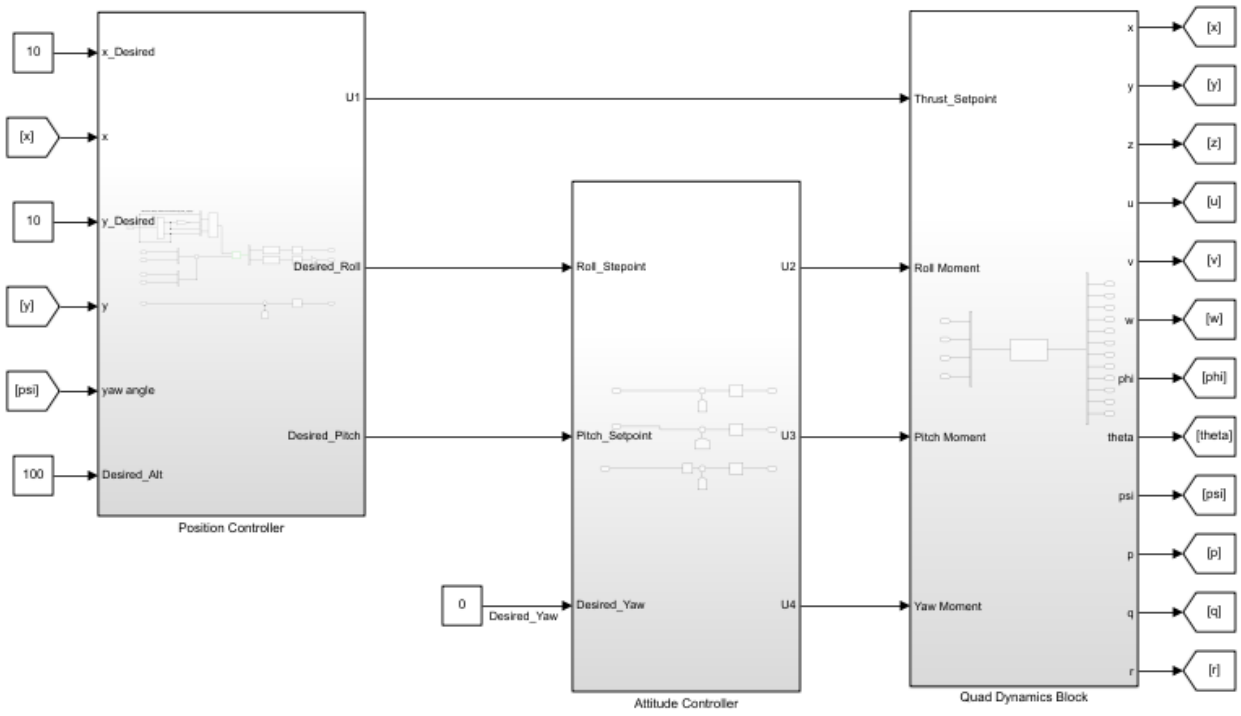


Figure 5.2.7. *Simulink PID Controller Block.*

Simulink Position Controller

The structure of the position controller is shown below, in which 3 PID controllers are implemented for each position state. In addition, the x & y error states are multiplied by a rotation matrix represented by the yaw angle. The rotation matrix works on obtaining the correct

representation of the position vector (x & y) in terms of the heading (yaw) angle of the quadrotor. Such that it's continually updated. In addition, a separate PID controller controls the altitude (z) of the quadrotor, which outputs the thrust setpoint (U1).

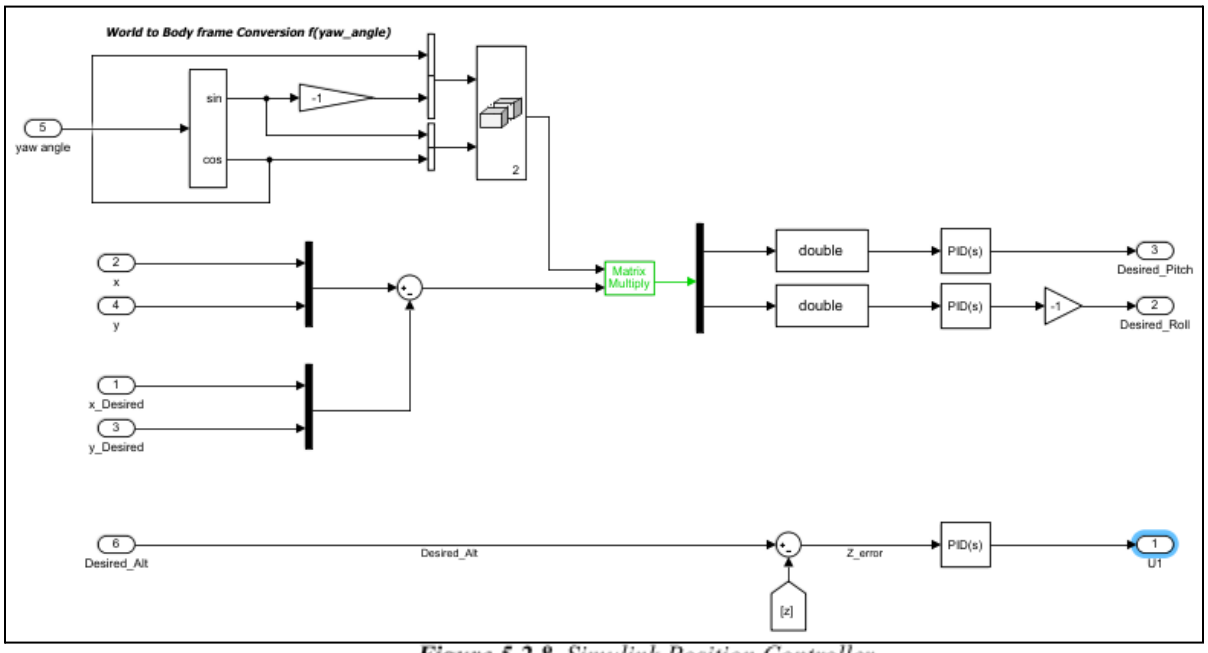


Figure 5.2.8. *Simulink Position Controller.*

Simulink Attitude Controller

The block structure is shown below, where each orientation state (roll, pitch, and yaw) is controlled by a PID controller. The input to both roll and pitch is the output setpoint of the position controller. The output of the inner loop controller is the actuator inputs (Moments) U2, U3, and U4.

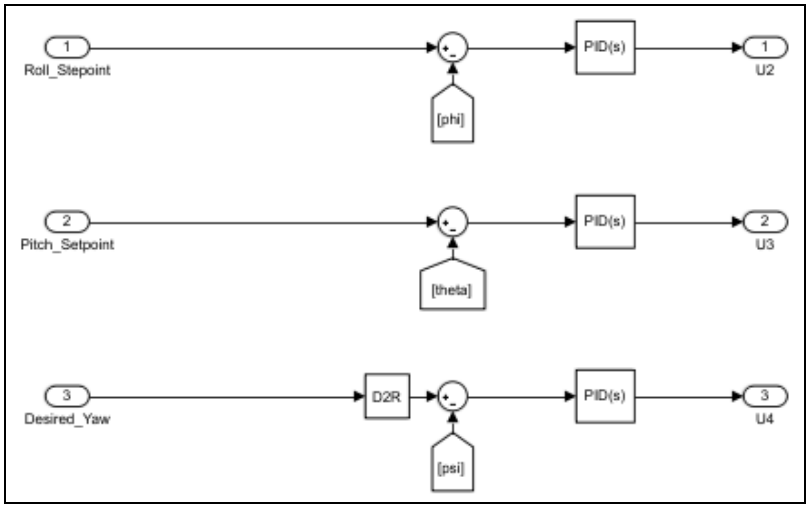


Figure 5.2.9. *Simulink Attitude Controller.*

5.2.2.2 Controller Tuning

There exist several methods that could be used to tune PID controllers, some of these methods are presented below:

Manual Tuning: this approach implies manually tweaking the PID controller gains until an acceptable response is obtained. However, in the case of a quadcopter, there are 6 PID controllers that should be adjusted. Accordingly, such an approach is deemed inefficient and impractical.

Heuristics & Optimization Algorithms: methods like Ziegler-Nichols and Cohen-Coon make use of the system response data and provide gain estimates using formulas. Such methods could be useful in achieving an acceptable response but not yet optimum. However, they could provide a good starting point or an initial guess for PID control parameters.

On the other hand, optimization algorithms like genetic algorithms and Particle Swarm Optimization can find the optimum or near-optimal PID gains. Their main working mechanism is iteratively minimizing a performance index (cost function) specified to the algorithm. While such an approach is more computationally intensive, it can effectively handle a complex system with nonlinear dynamics, as in the current case of a quadrotor.

Proposed Approach

In this work, Ziegler-Nichols Method was used to obtain a good initial guess for PID parameters. Thereafter, the gains were manually tweaked in order to test the response under small perturbations in PID parameters. In such a sense, an estimate for the search area was identified,

which was used together with the initial guess as an input to optimization algorithms in order to fine-tune the controller and reach the best performance possible.

Ziegler-Nichols Method

The method is applied to tune the PID controller in the following manner (Meshram & Kanojiya, 2012):

1. The integral gain is set to zero.
2. The proportional gain is increased incrementally while examining the system response until sustained oscillations are obtained.
3. Set the current proportional gain to K_u.
4. The period of oscillation, time from peak to peak, is measured and set to T_u.
5. The gains are then estimated using the formulas shown in the table below.

Controller	K_p	T_i	T_d	K_i	K_D
PID	$0.6 * K_u$	$T_u/2$	$T_u/8$	$1.2 * K_u/T_u$	$0.075 * K_u T_u$
P	$0.5 * K_u$	-	-	-	-
PI	$0.45 * K_u$	$T_u/1.2$	-	$0.54 * K_u/T_u$	-
PD	$0.8 * K_u$	-	$T_u/8$	-	$0.1 * K_u T_u$

Table 5.2.2. *PID ZN Controller Gain Estimation (Meshram & Kanojiya, 2012).*

In addition to the estimated gains, the model response could be fine-tuned manually as an attempt to improve the performance after the initial guess obtained from the ZN method.

Optimization Algorithms

PID gain optimization algorithms make use of mathematical optimization techniques to find the optimal values for PID parameters. Algorithms achieve this by iteratively searching for the combination of parameters that minimizes the performance index or cost function. This is achieved by relying on feedback from the performance index to update the parameters (Ahmmed et al., 2020). Therefore, they can help in improving the control performance and robustness of the system.

The choice of which optimization algorithm to use is dependent on the system characteristics. In the case of a quadrotor, the system dynamics are highly nonlinear. In addition, the optimization problem involves variables with high-dimensional search space. Moreover, the fitness (cost) function is non-differentiable. Therefore, the gradient is not available for use. Finally, the optimization problem is considered multimodal, meaning that several optimal or near-optimal solutions exist depending on the chosen search space.

All of this narrowed down the options to a few algorithms, among which Genetic Algorithm was suitable for our case.

Genetic Algorithm

The algorithm is inspired by the process of natural selection, in which the algorithm operates on a set of potential solutions, where individuals undergo genetic operations to reproduce better generations (Ahmmed et al., 2020). Reproduction is meant to produce new generations that carry better characteristics contributing to a better fitness value for the objective function. In that sense, the algorithm can help find the optimum PID parameters that minimize the performance index.

Algorithm Setup

Several vital factors should be considered carefully to achieve convergence with suitable computational time while achieving acceptable performance (Mathworks, 2023).

1. **Search Space:** while a large search space can help the algorithm explore further solutions, potentially improving the performance, it can come at the expense of high computational effort. Accordingly, the ZN method was initially used to narrow down to a viable search space.
2. **Population Size:** large population size can also help in better exploration of the search space. However, if it's too high, it may slow down the simulation due to computational overhead. As more individuals in this case will need to be evaluated and selected in each generation (iteration). Therefore, it's recommended to start from 50 to 200 initially and increase incrementally while monitoring performance.
3. **Max Number of Generations:** it determines how long the algorithm will run before termination. Setting a high number of generations can potentially improve the result, as the algorithm will continue searching for solutions. However, it's also computational

overhead. It's also recommended to start with 100 to 1000 and increase incrementally as well.

4. **Performance Index:** the choice of the fitness (cost) function contributes greatly to both the convergence of the algorithm and performance improvement. In this case, two cost functions are proposed; the least squares method and integral time absolute error.

Simulation Results

All the controllers were tuned as PD controllers, meaning that the integral gain was set to zero in all controllers. This was done for the following reasons:

1. **Simplicity:** removing the integral gain will reduce the problem size in terms of the number of variables to be tuned.
2. **Model Inaccuracies:** the mathematical model of the quadrotor may fail to capture the true dynamics exactly of the quadrotor in real-life operation. As the measurements are subject to sensor noises. The integral term is very sensitive to such uncertainties and may lead to instability.

Ziegler-Nichols Method

The method was applied to the model states while maintaining a zero desired yaw angle.

Inner Loop Controller		
State	K_u	T_u
Phi	0.1875	6.93 s
Theta	0.075	7.33 s
psi	0.625	1.6 s

Table 5.2.3. *ZN Results for Inner Loop.*

Outer Loop Controller		
State	K_u	T_u
X	$2.5 * 10^{-3}$	16 s
Y	$3.75 * 10^{-3}$	13.3 s
Z	18.75	3.73 s

Table 5.2.4. *ZN Results for Outer Loop.*

These results were then used to compute the proportional and derivative gains for all the controllers. In addition, some states were manually tuned further in an attempt to improve the response.

5.2.2.2.1 Ziegler-Nichols Results

Testing response of quadrotor given a setpoint in x and y directions (10, 10), starting initially at (0, 0), while altitude was given a setpoint at 110 m.

Outer Loop Response

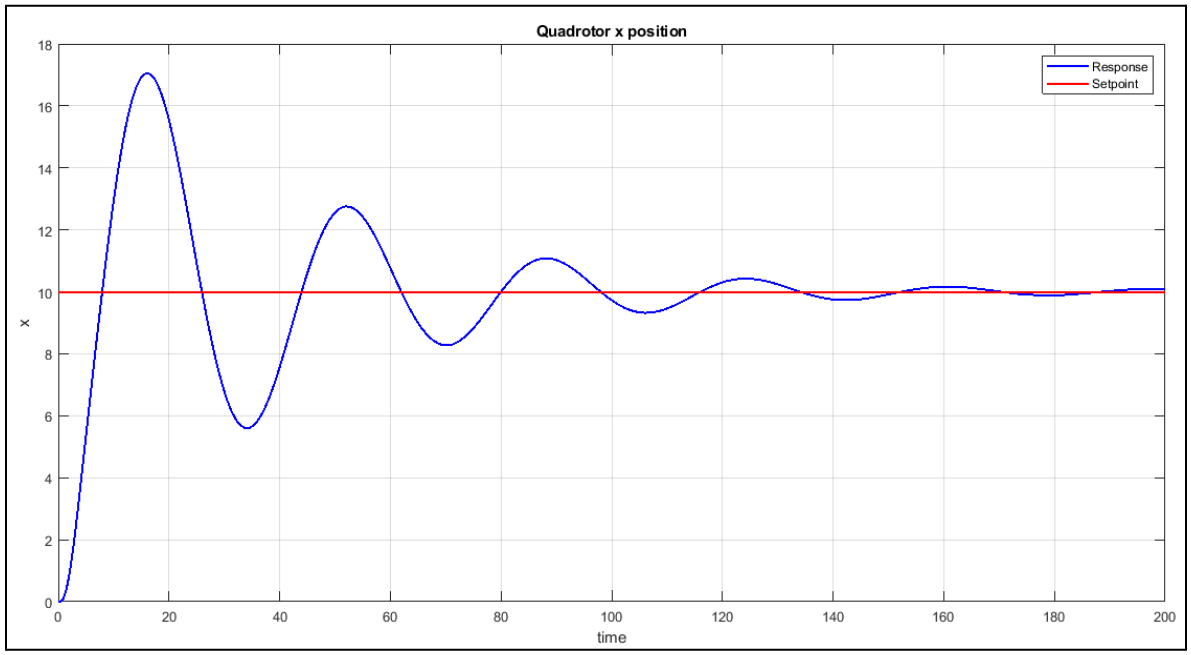


Figure 5.2.10. *X position response (ZN).*

Rise Time	5.1897
Settling Time	146.5576
% Overshoot	70.5037

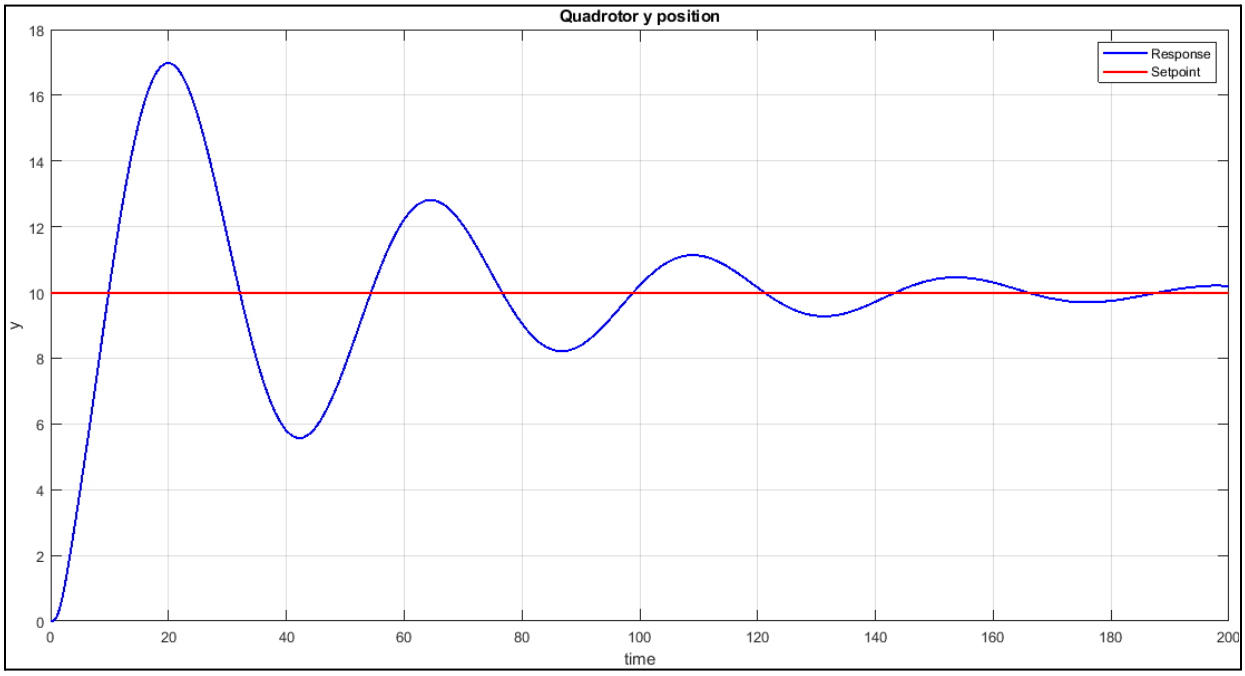


Figure 5.2.11. *Y position response (ZN).*

Rise Time	6.7438
Settling Time	202.47
% Overshoot	69.777

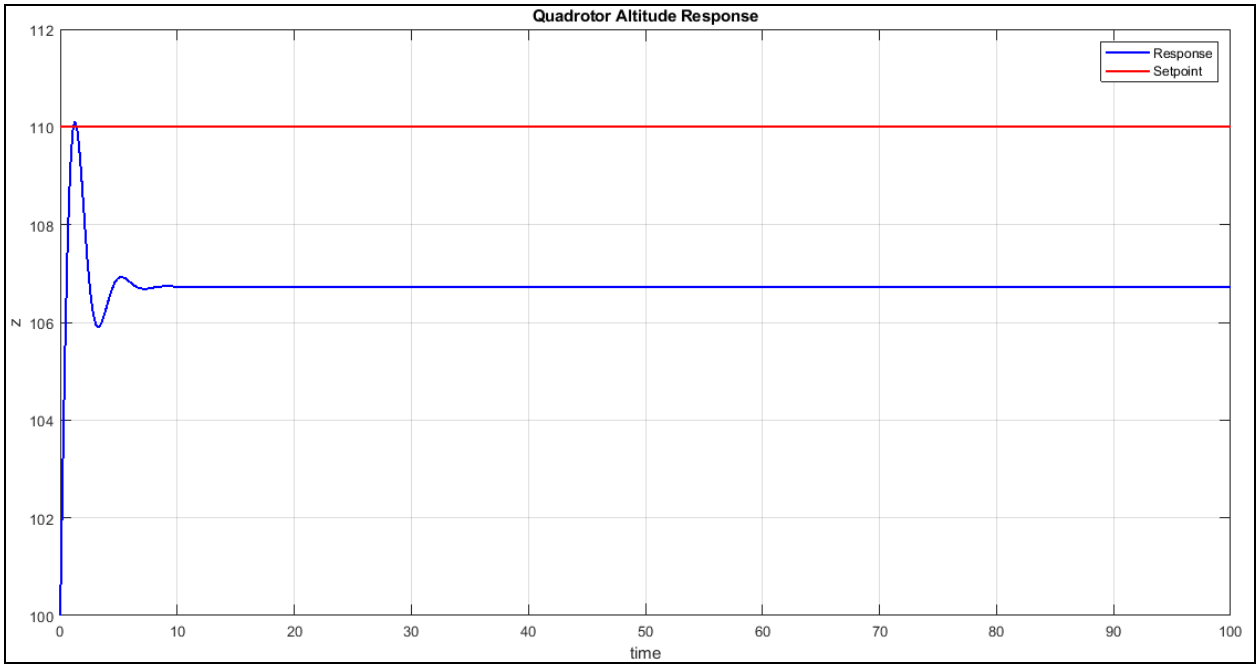


Figure 5.2.12. *Altitude response (ZN).*

Rise Time	0.7711
Settling Time	NAN
% Overshoot	0.0991

Inner Loop Response

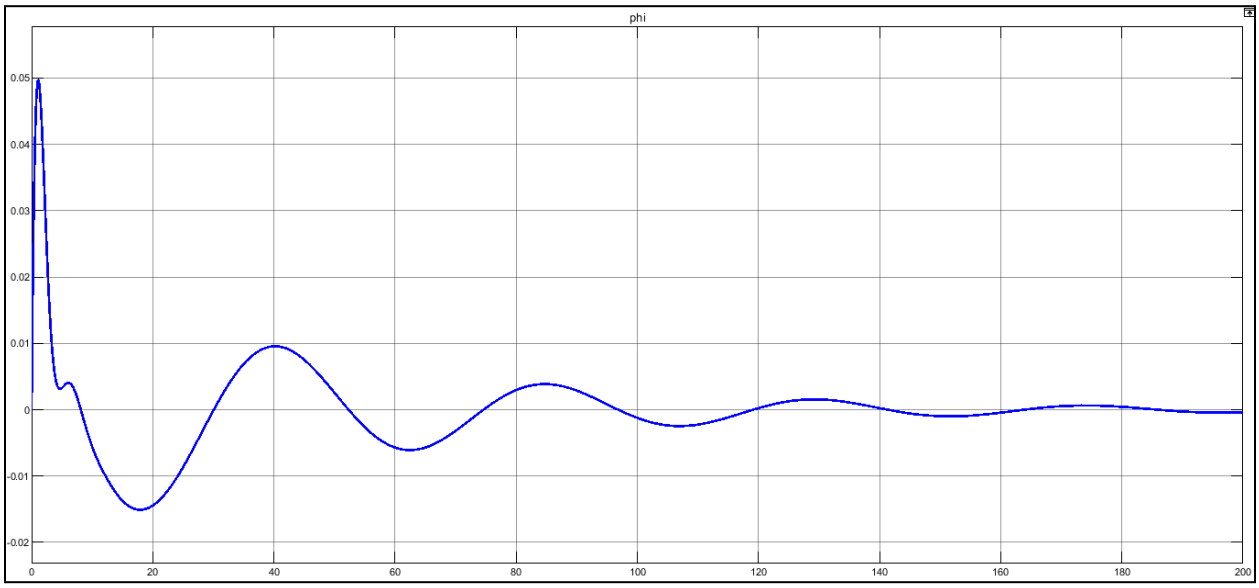


Figure 5.2.13. *Phi Angle Response (ZN).*

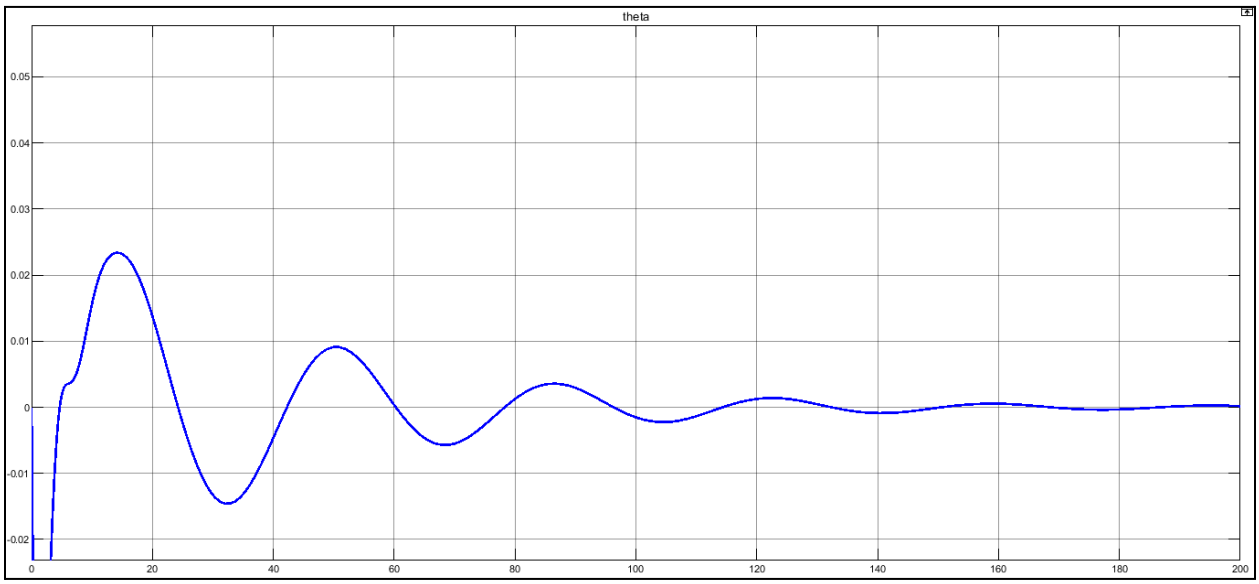


Figure 5.2.14. *Theta Angle Response (ZN).*

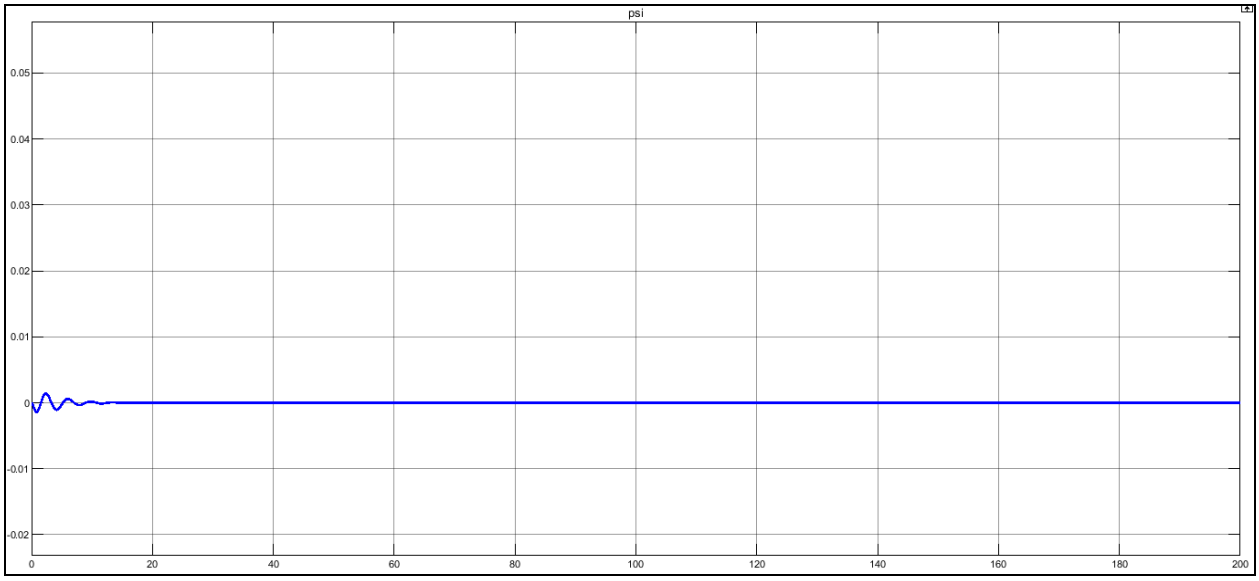


Figure 5.2.15. *Psi Angle Response (ZN).*

Disturbance Rejection Response

The quadrotor controller was tested to damp out 2 degrees of disturbance in both roll & pitch angles.

Outer Loop Response

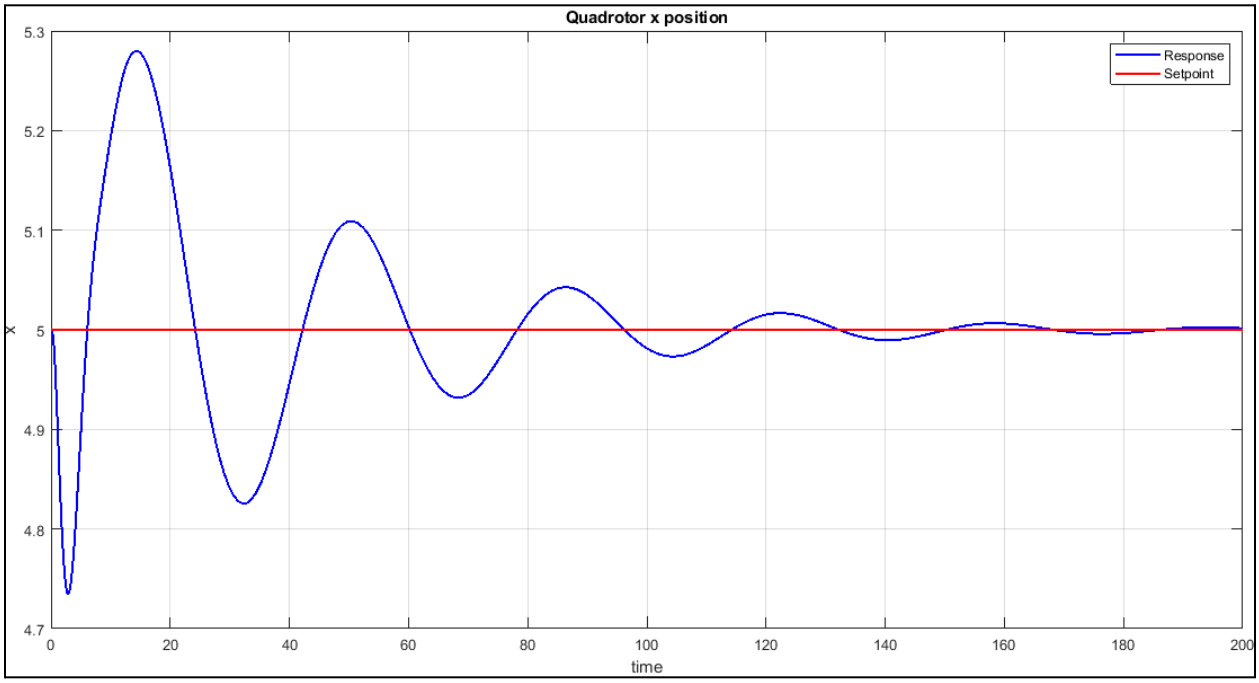


Figure 5.2.16. *X Disturbance Rejection (ZN).*

Settling Time	161.593
% Overshoot	5.5958

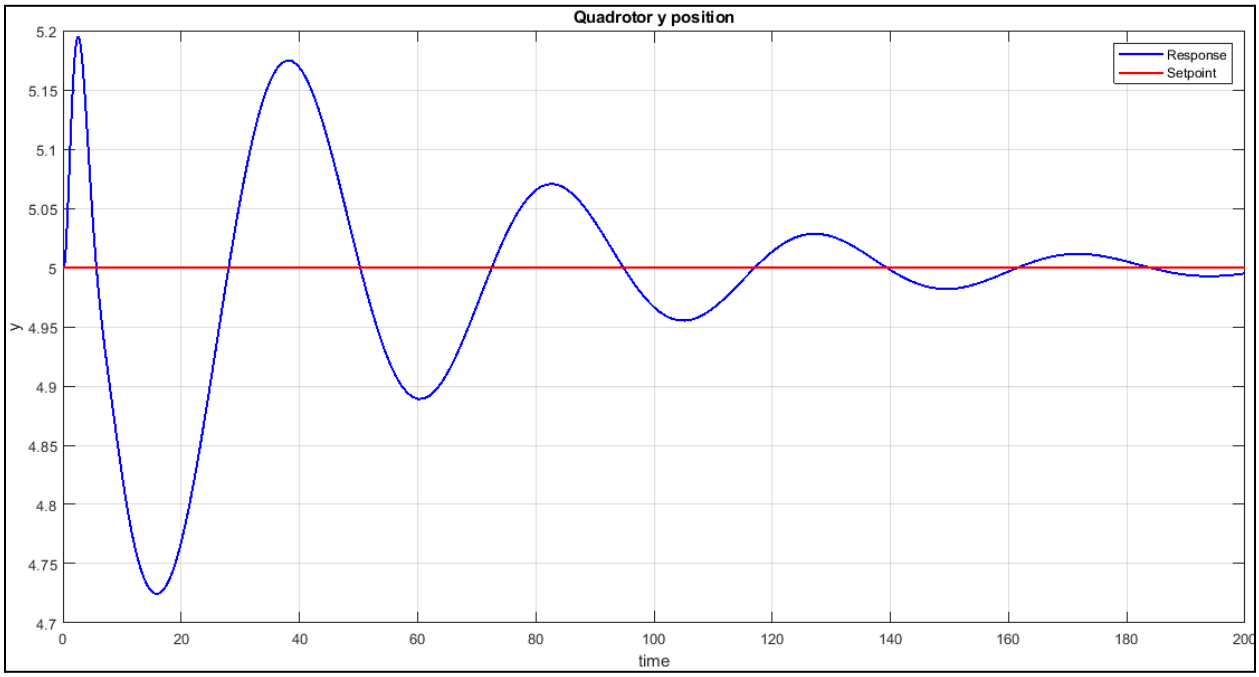


Figure 5.2.17. Y Disturbance Rejection (ZN).

Settling Time	199.1681 s
% Overshoot	3.8935

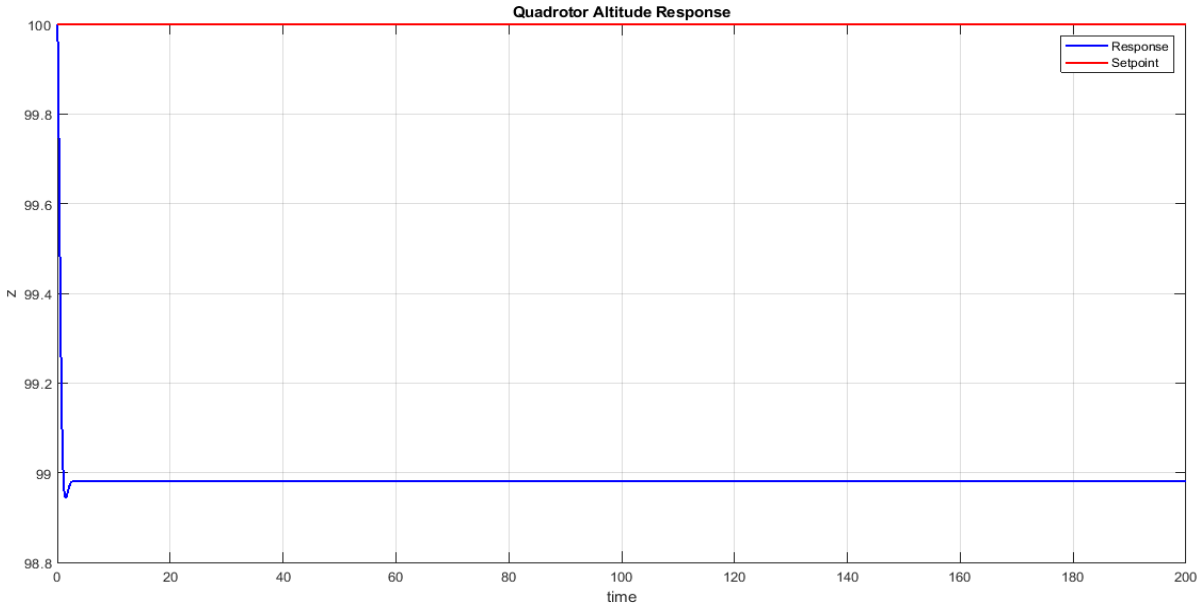


Figure 5.2.18. *Z Disturbance Rejection (ZN).*

Inner Loop Response

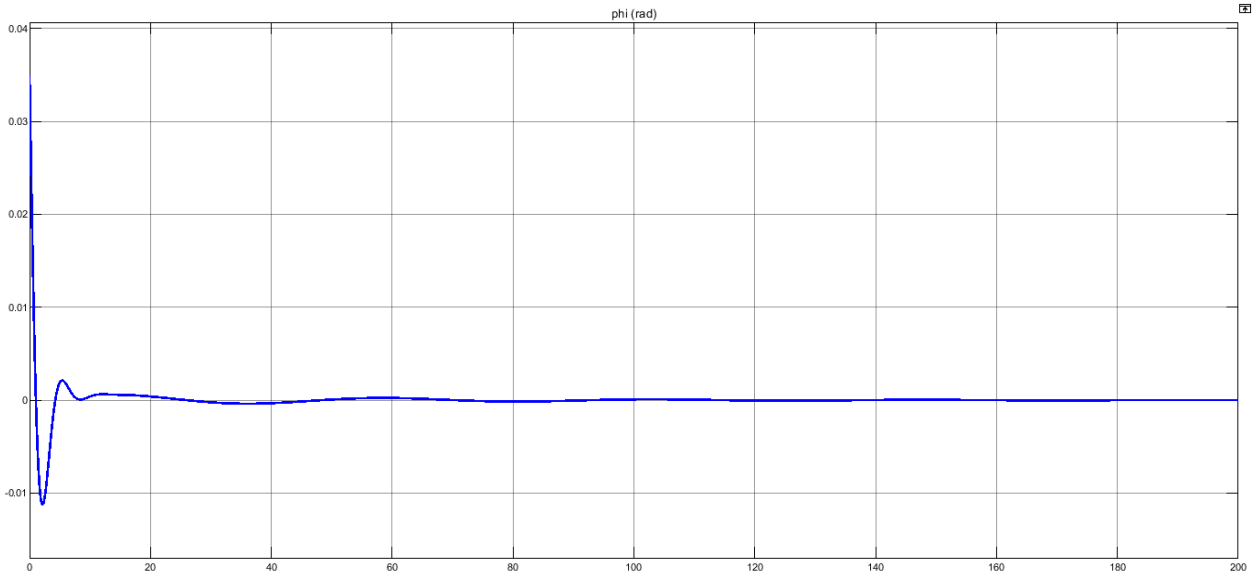


Figure 5.2.19. *Phi Angle Response (ZN).*

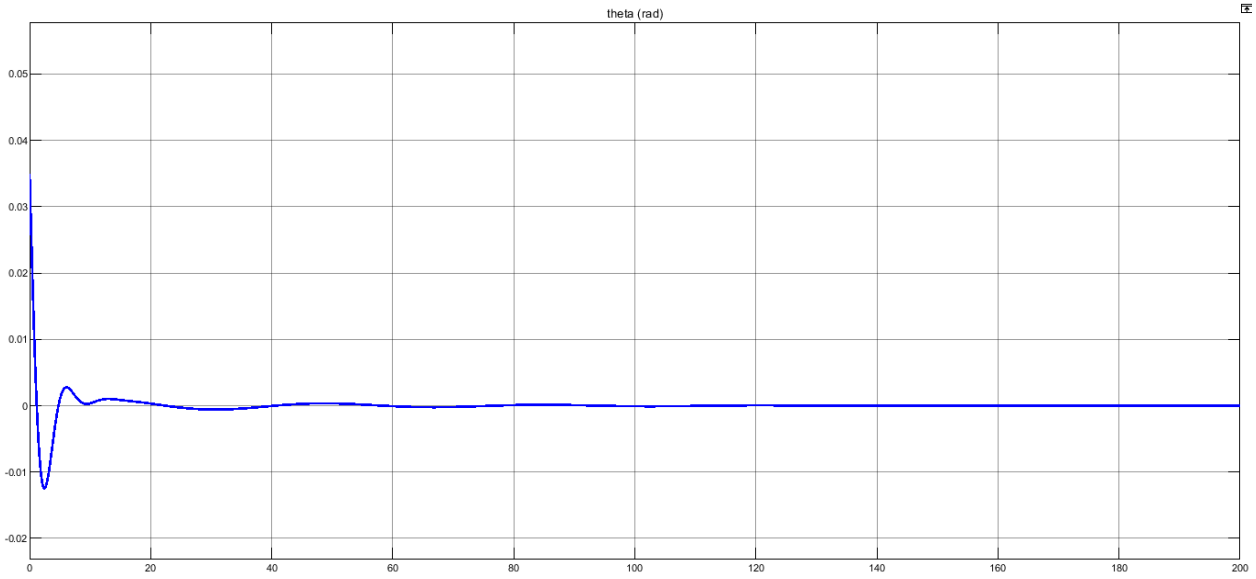


Figure 5.2.20. *Theta Angle Response (ZN).*

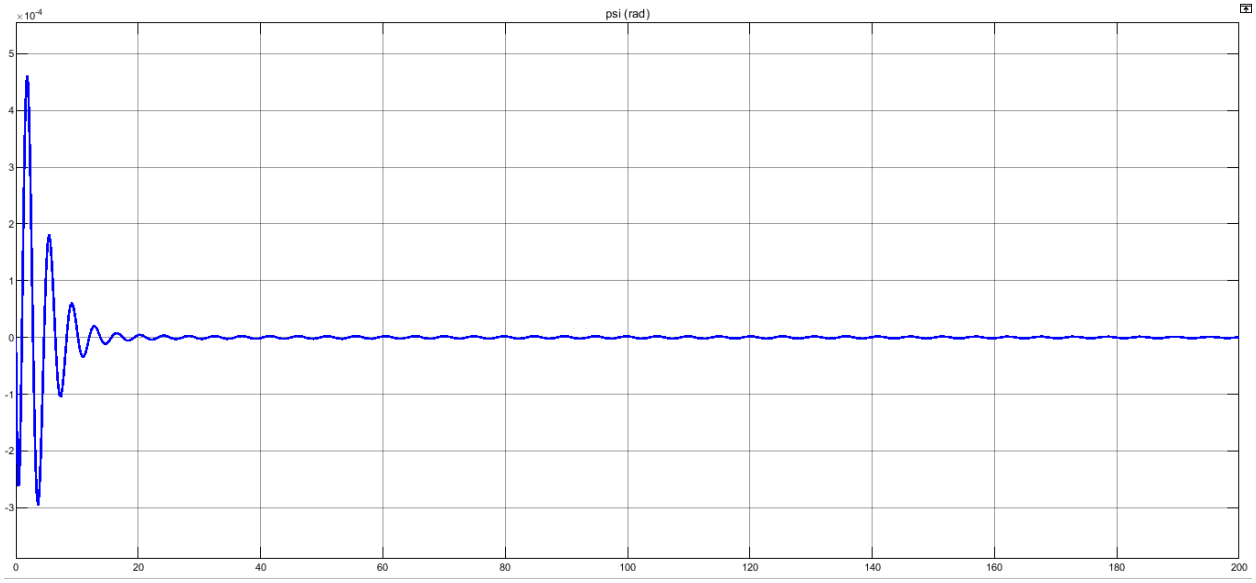


Figure 5.2.21. *Psi Angle Response (ZN).*

Discussion

As for the step response, high levels of overshoot and long settling times were spotted. However, the controller managed to settle eventually around the setpoint. As for disturbance rejection, the overshoot was acceptable, as it oscillates with small values across the setpoint. However, the

settling time problem persists in this case as well.

5.2.2.2.2 Genetic Algorithm Results

The quadrotor was given setpoints to move 10 m in x, y, and z directions.

Outer Loop Response

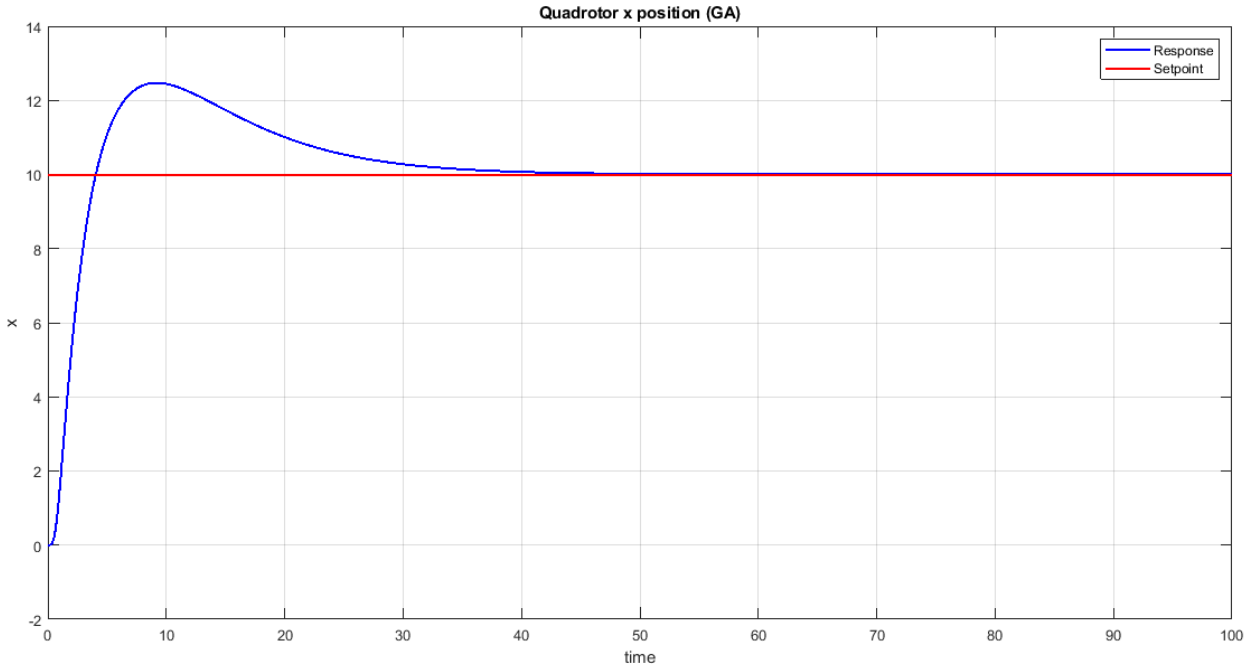


Figure 5.2.22. *X Position Response (GA).*

Performance	Value	Improvement
Rise Time	2.5844	50.2%
Settling Time	32.4265	77.87%
% Overshoot	24.7	64.97%

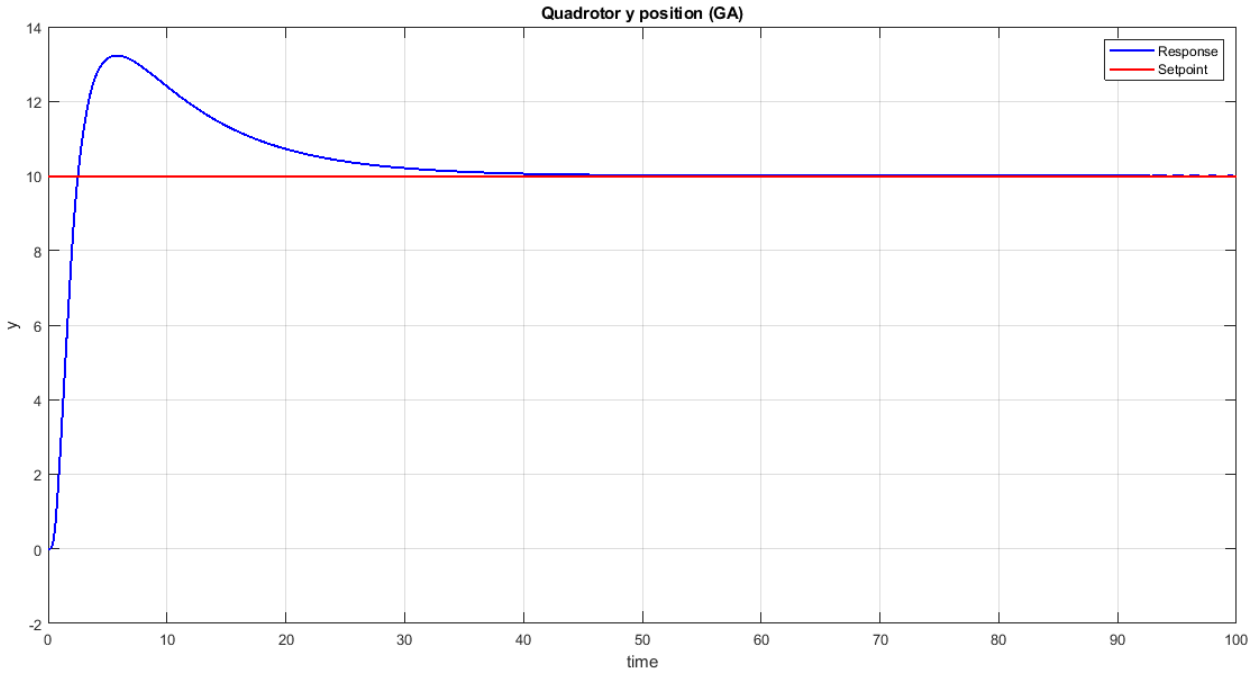


Figure 5.2.23. Y Position Response (GA).

Performance	Value	Improvement
Rise Time	1.5242	77.39%
Settling Time	30.3737	84.99%
% Overshoot	32.2996	53.71%

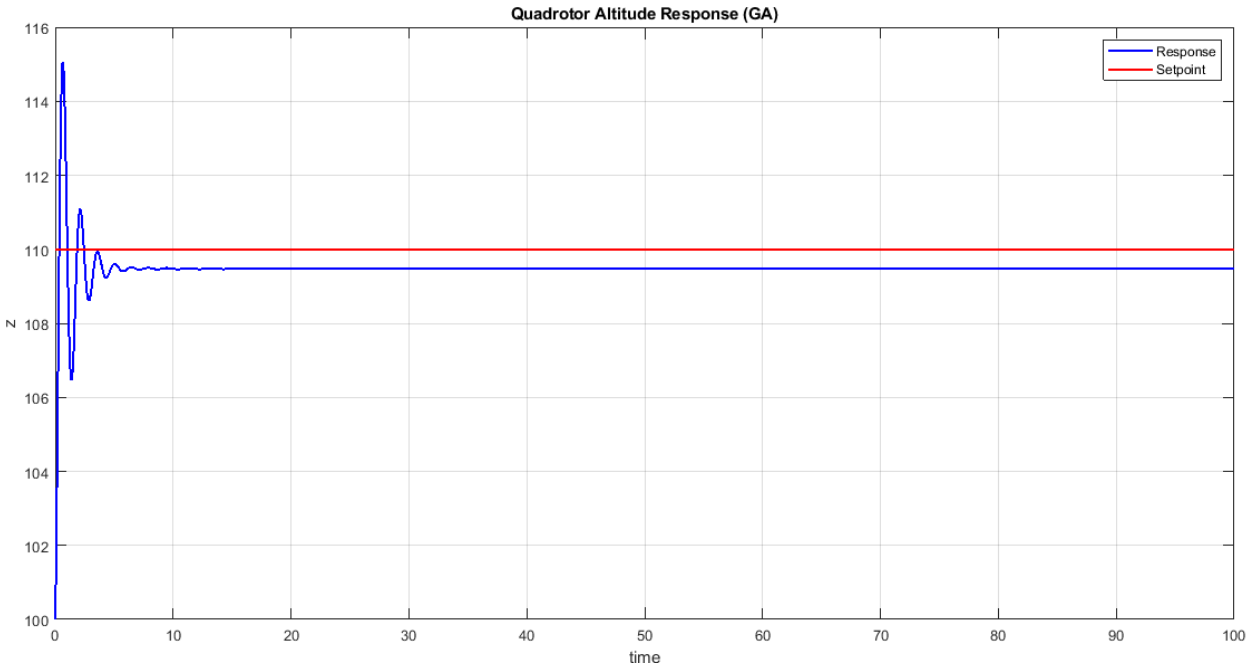


Figure 5.2.24. *Altitude Step Response (GA).*

Performance	Value	Improvement
Rise Time	0.2541	67.04%
Settling Time	NAN	NAN
% Overshoot	4.591	-45%

Inner Loop Response

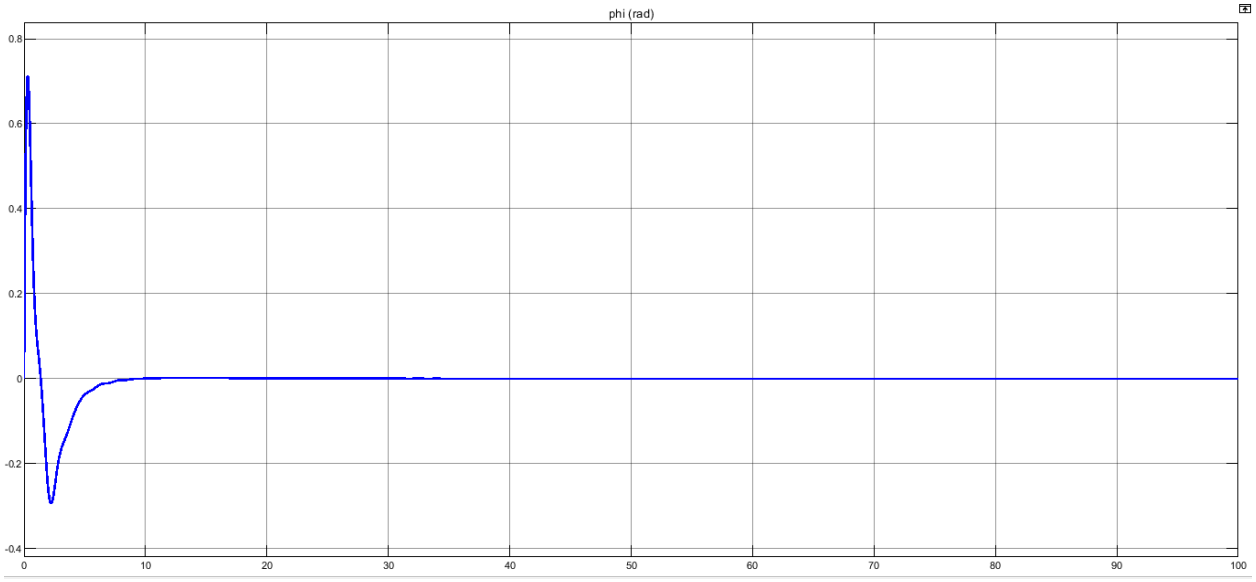


Figure 5.2.25. *Phi Angle Response (GA).*

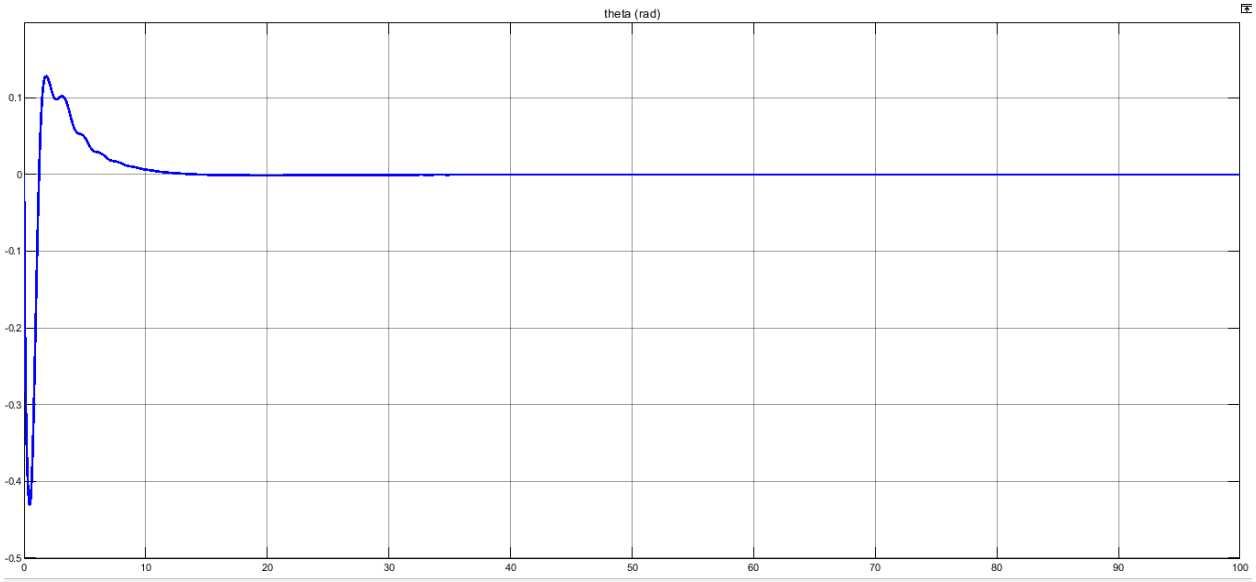


Figure 5.2.26. *Theta Angle Response (GA).*

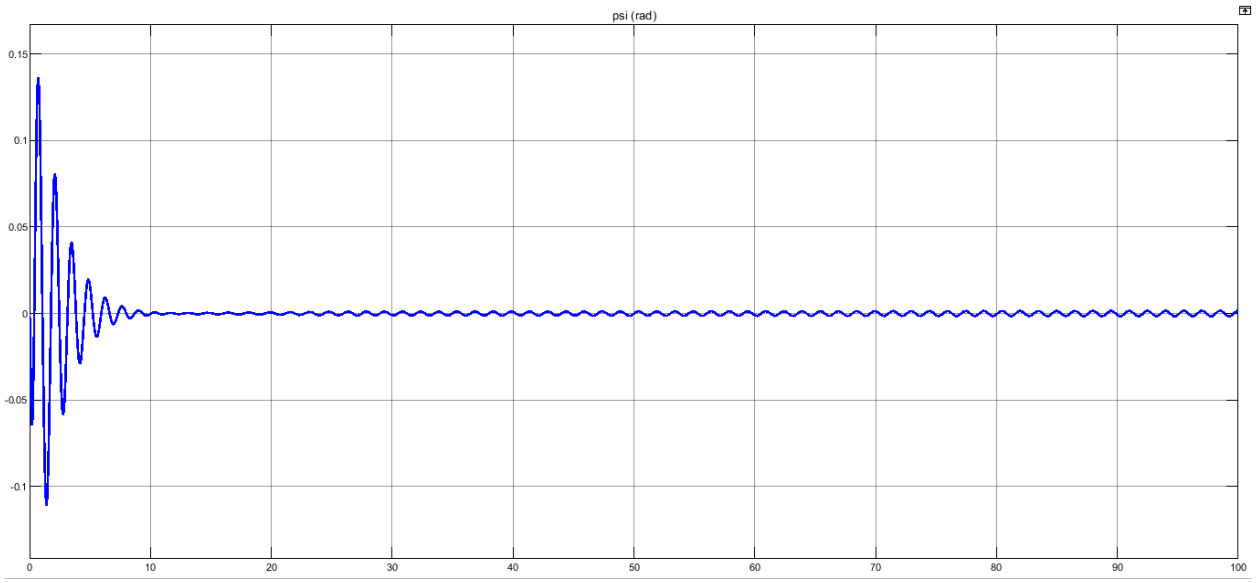


Figure 5.2.27. *Psi Angle Response (GA).*

Disturbance Rejection

The quadrotor, initially set at (5, 5, 100), was given a disturbance of 2 degrees in roll & pitch.

Outer Loop Response

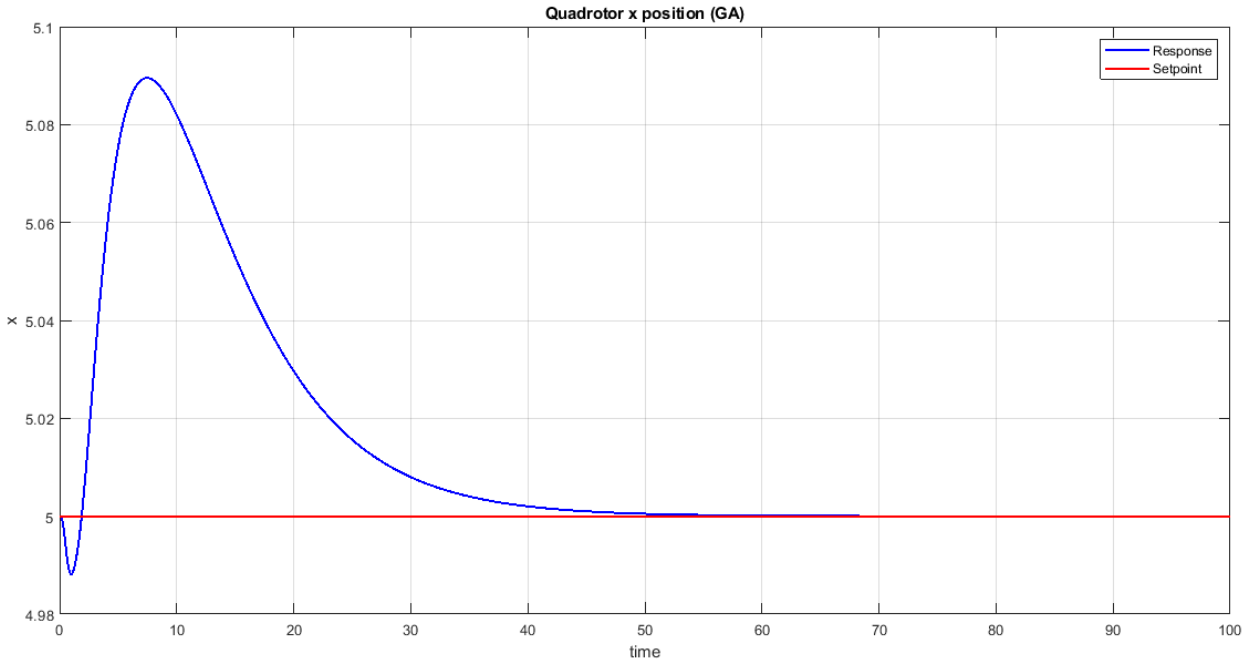


Figure 5.2.28. *X Position Response (GA).*

Performance	Value	Improvement
Settling Time	40.8368	74.73%
% Overshoot	1.79	68%

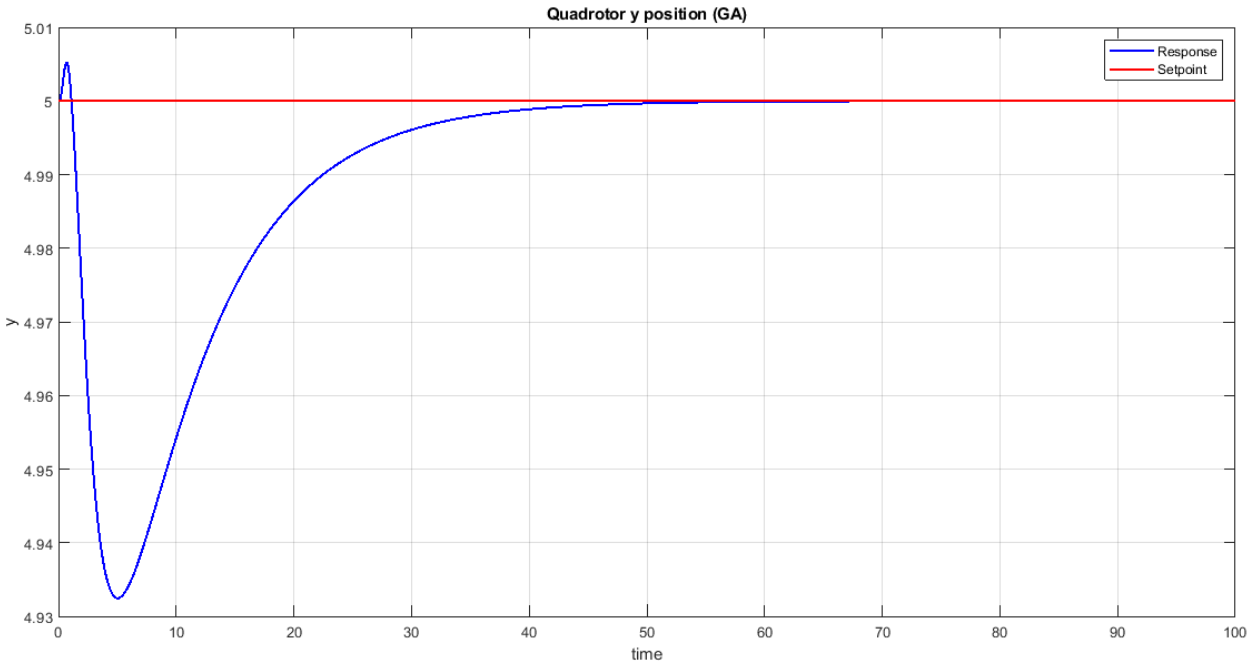


Figure 5.2.29. *Y Position Response (GA).*

Performance	Value	Improvement
Settling Time	38.5587	80.6%
% Overshoot	0.1048	97.3%

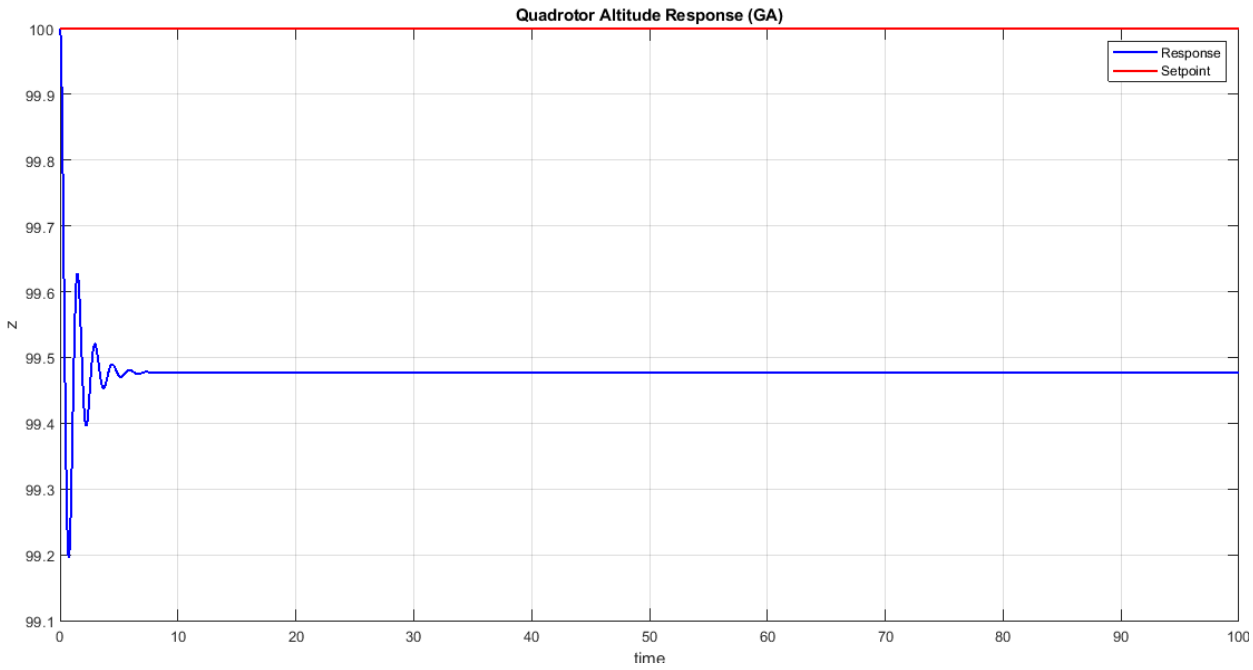


Figure 5.2.30. *Altitude Response (GA).*

Inner Loop Response

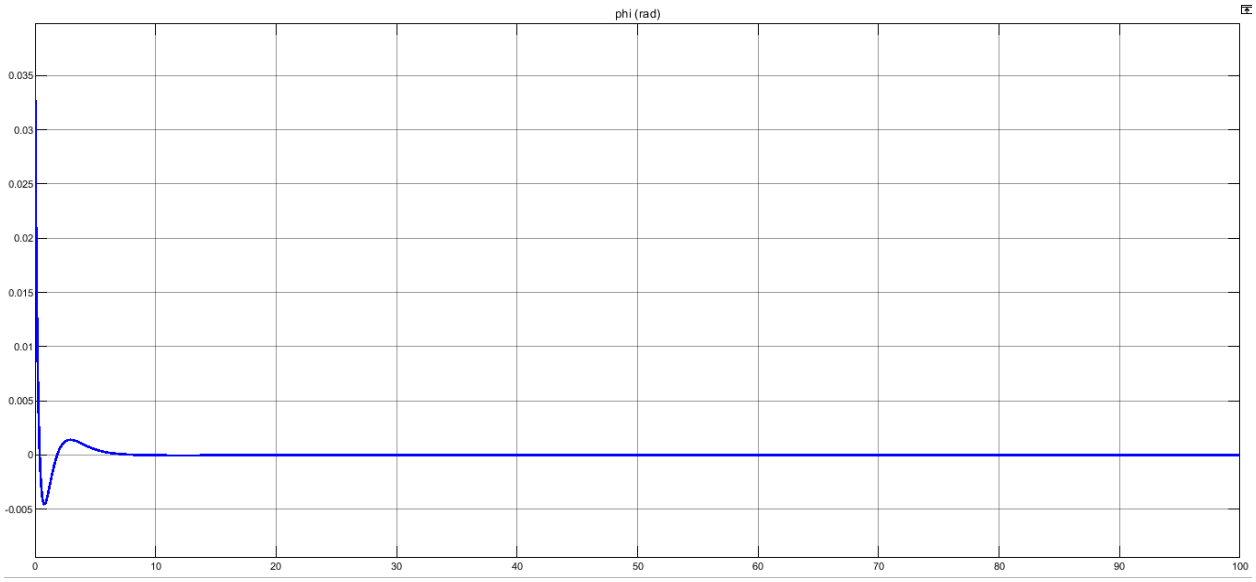


Figure 5.2.31. *Phi Angle Response (GA).*

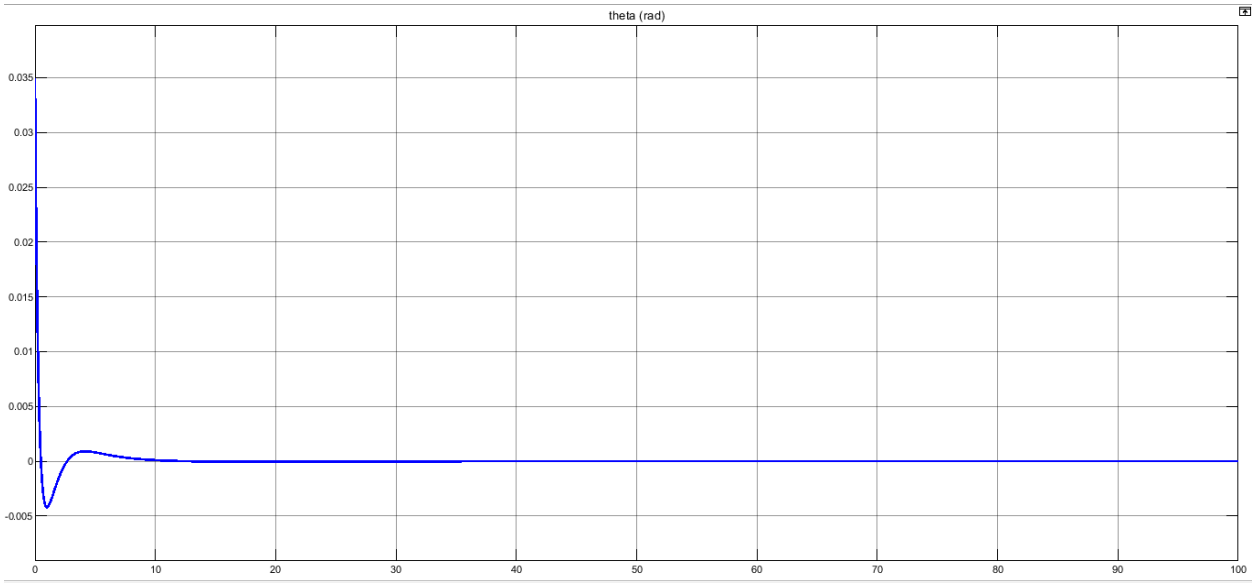


Figure 5.2.32. *Theta Angle Response (GA).*

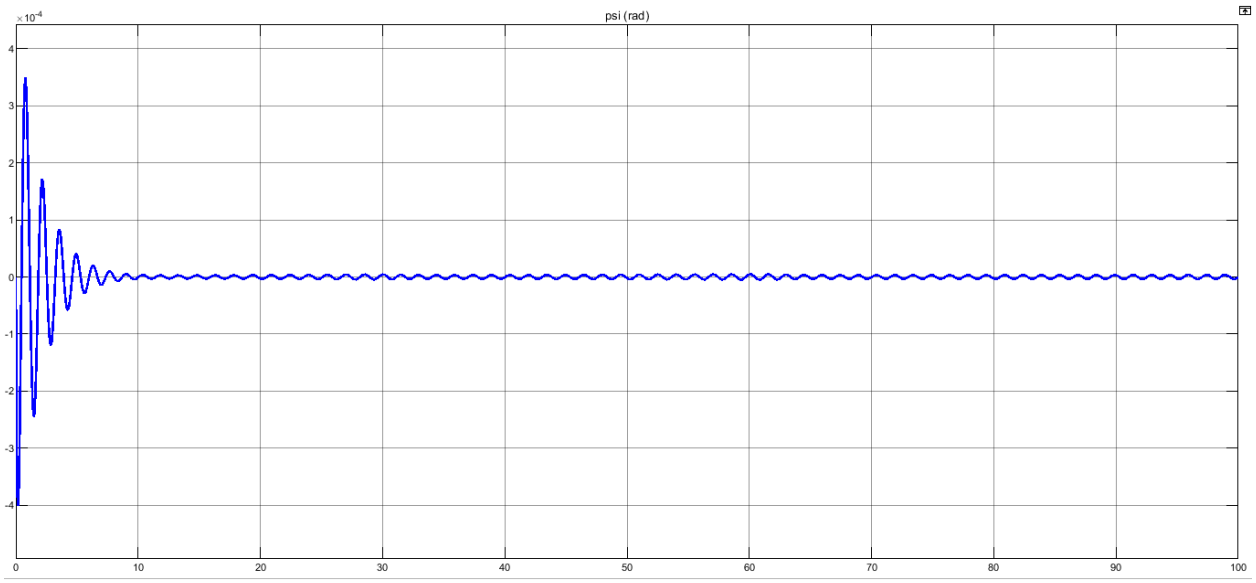


Figure 5.2.33. *Psi Angle Response (GA).*

Discussion

In both test scenarios; step response and disturbance rejection, the outer loop response (x & y) positions improved greatly and that was reflected in the performance criteria as shown above. As for altitude (z), the response did not settle at the setpoint desired exactly, but the error was in the range of 5% of the setpoint. Hence, the controller works best in XY planar motion. On the other hand, difficulties might be present in 3D motion.

Performance Further Improvement

The obtained results may be refined further to achieve better performance. This could be achieved by expanding the search area to include more potentially viable solutions. In addition, optimization settings could be adjusted to have more populations and generations. However, all of this will come as computational overhead.

Another approach is using hybrid algorithms in MATLAB. Instead of just relying on Genetic Algorithm, the algorithm starts with GA and once it's finished, it passes the obtained value to another optimization method. Many methods are available including: fminsearch, patternsearch, and fminunc.

5.2.2.3 Trajectory Following

In this section, the quadrotor was tested to follow aggressive planar trajectories. The quadrotor was tested on two trajectories; circle and infinity sign.

5.2.2.3.1 Circle Trajectory Results

Response to a circle trajectory of radius 5 is shown below. The figures below show a 3D demonstration of the trajectory in x, y, z directions. Then, modeling the response of outer loop elements followed by the inner loop.

Outer Loop Response

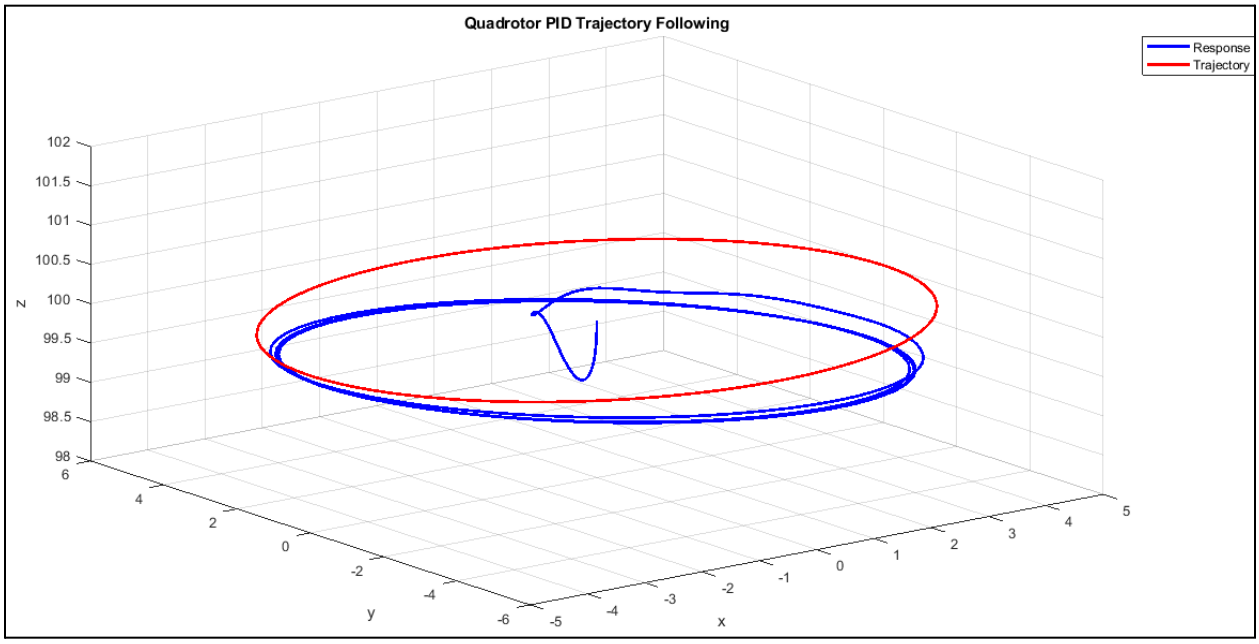


Figure 5.2.34. *3D Response of the Quadrotor.*

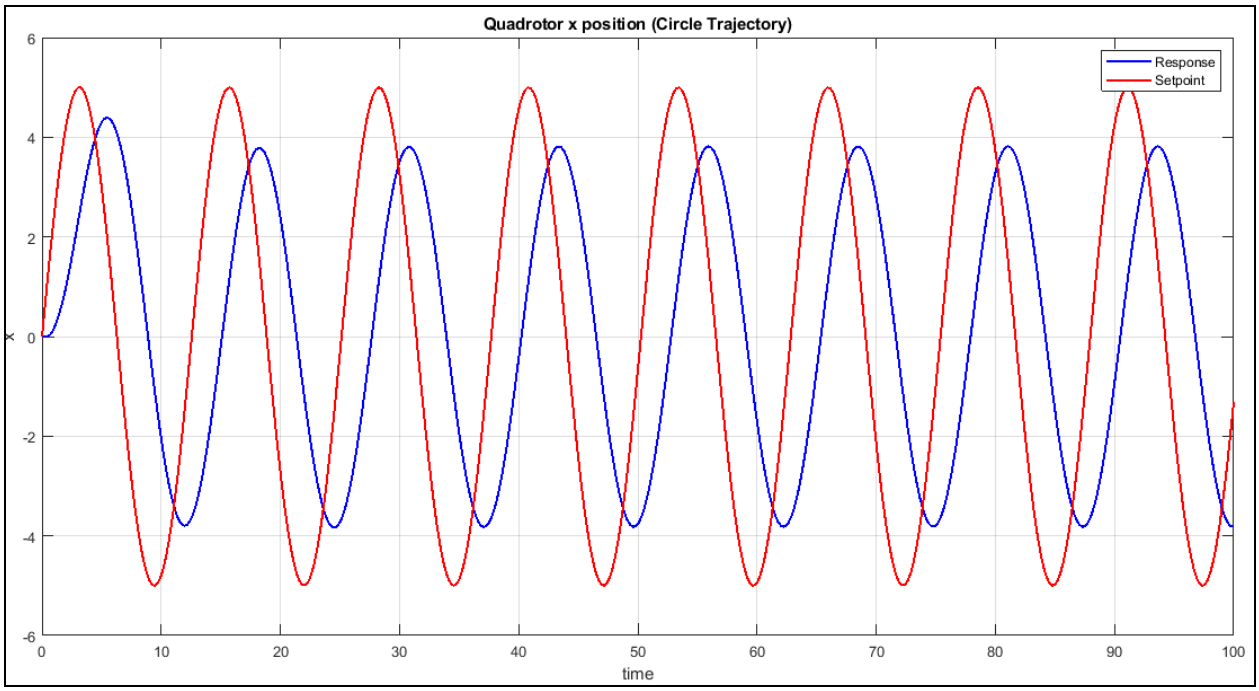


Figure 5.2.35. *X Position Response.*

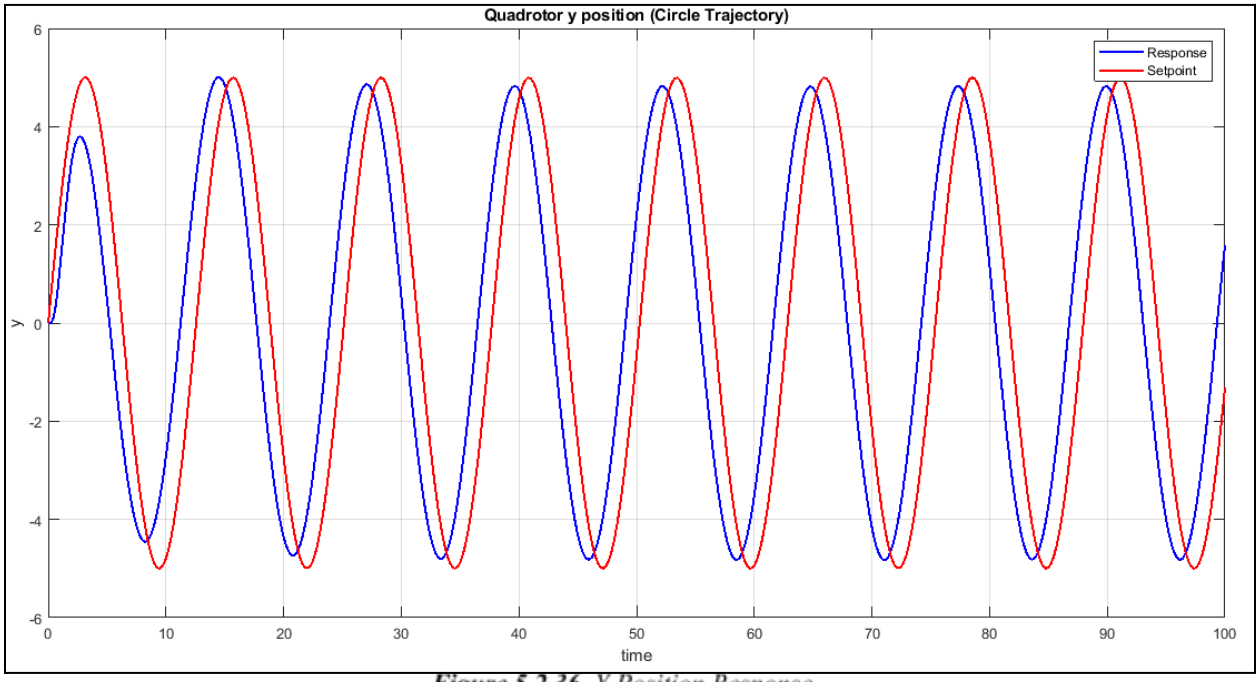


Figure 5.2.36. *Y Position Response.*

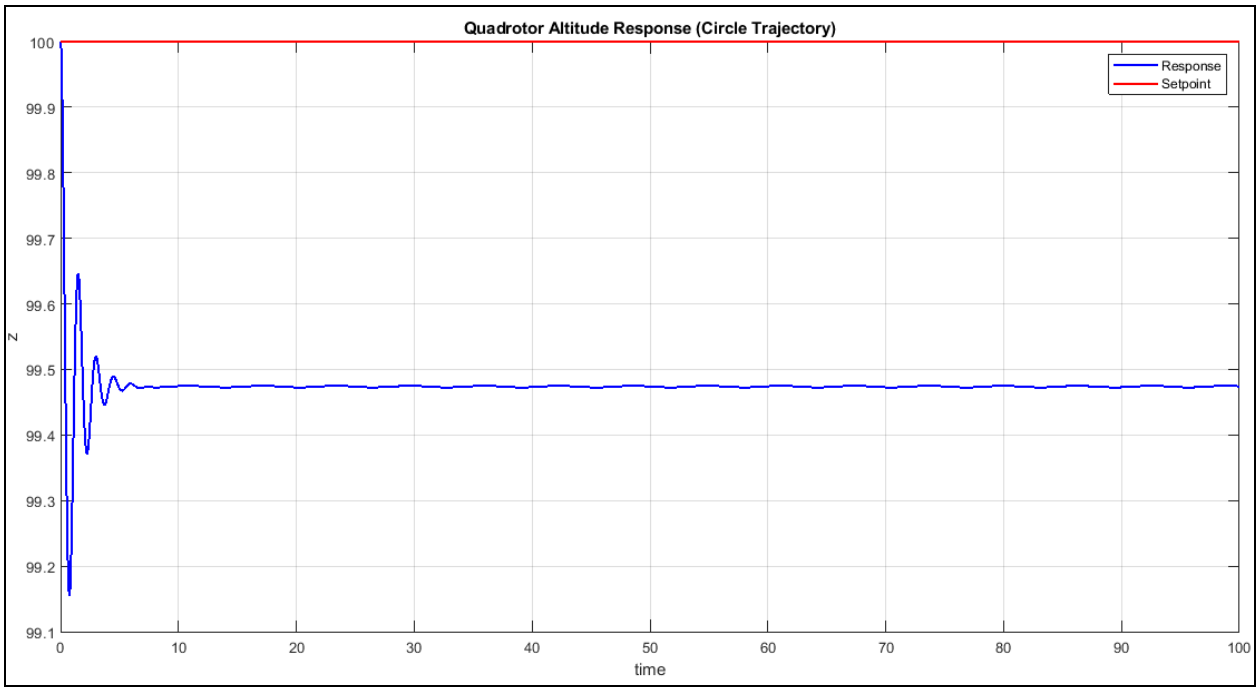


Figure 5.2.37. *Z Position Response.*

Inner Loop Response

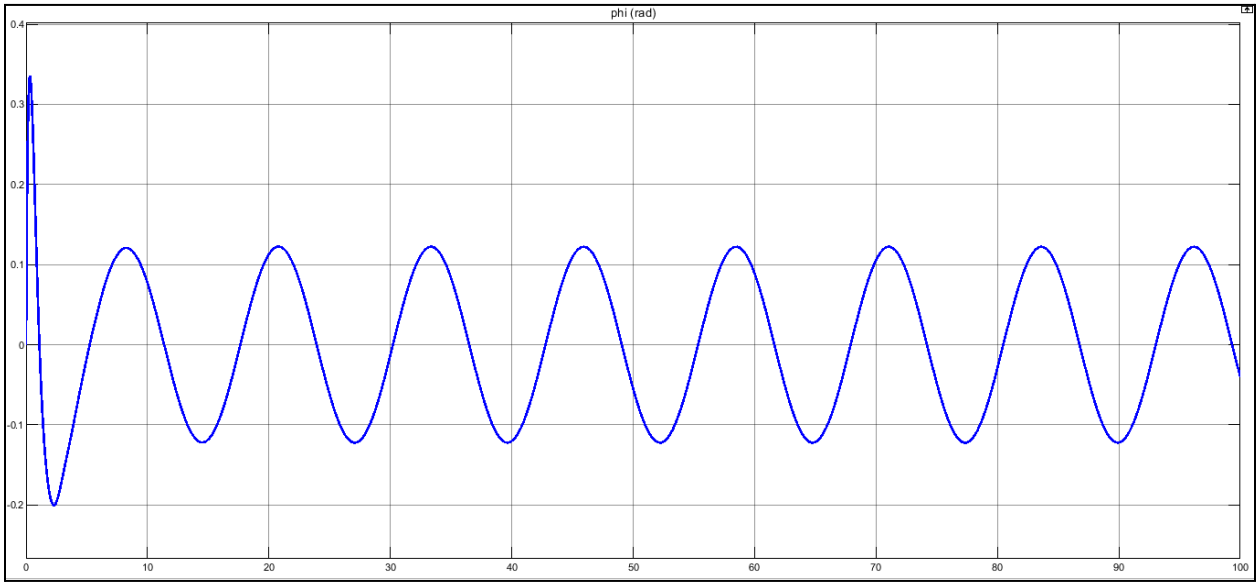


Figure 5.2.38. *Phi Angle Response.*

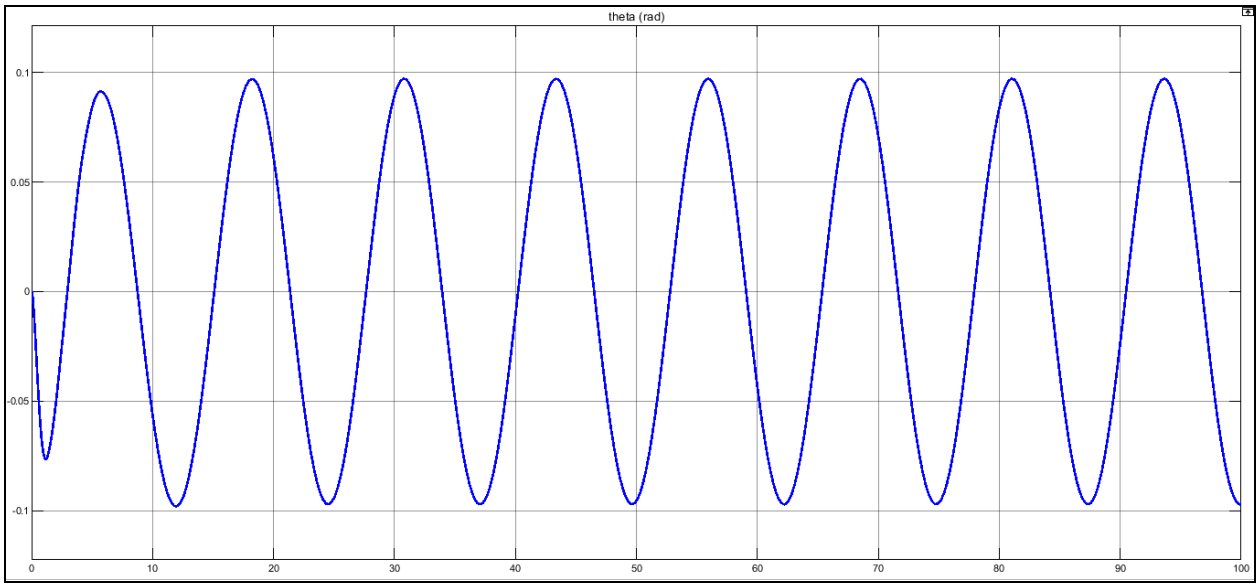


Figure 5.2.39. *Theta Angle Response.*

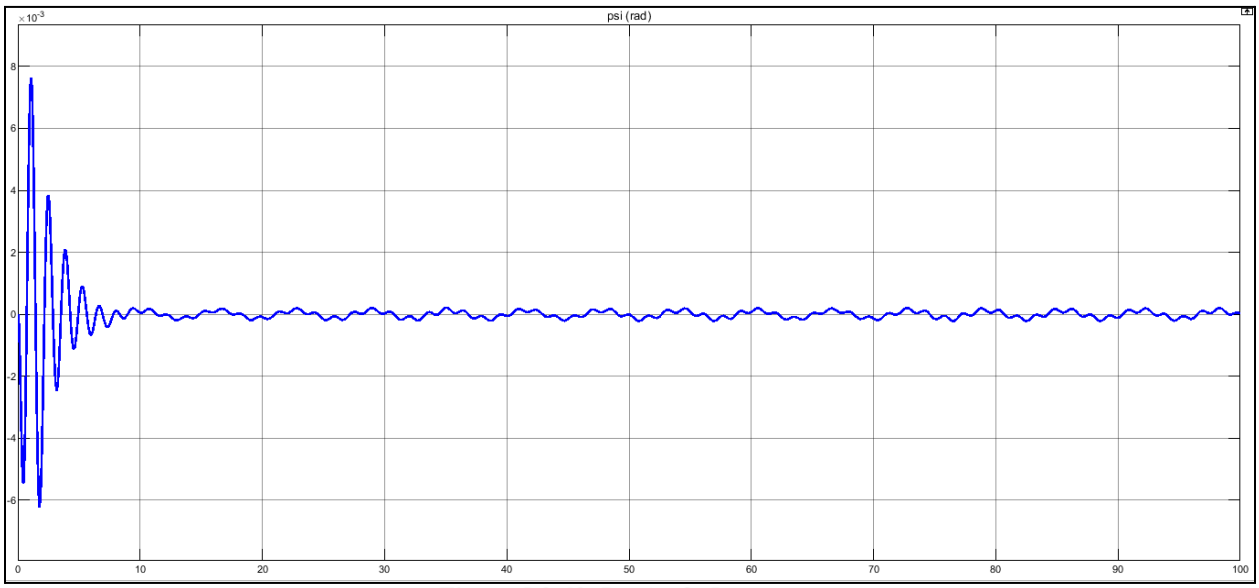


Figure 5.2.40. *Psi Angle Response.*

5.2.2.3.2 Infinity Trajectory Results

The quadrotor was also tested on an infinity-like trajectory. Results are shown below.

Outer Loop Response

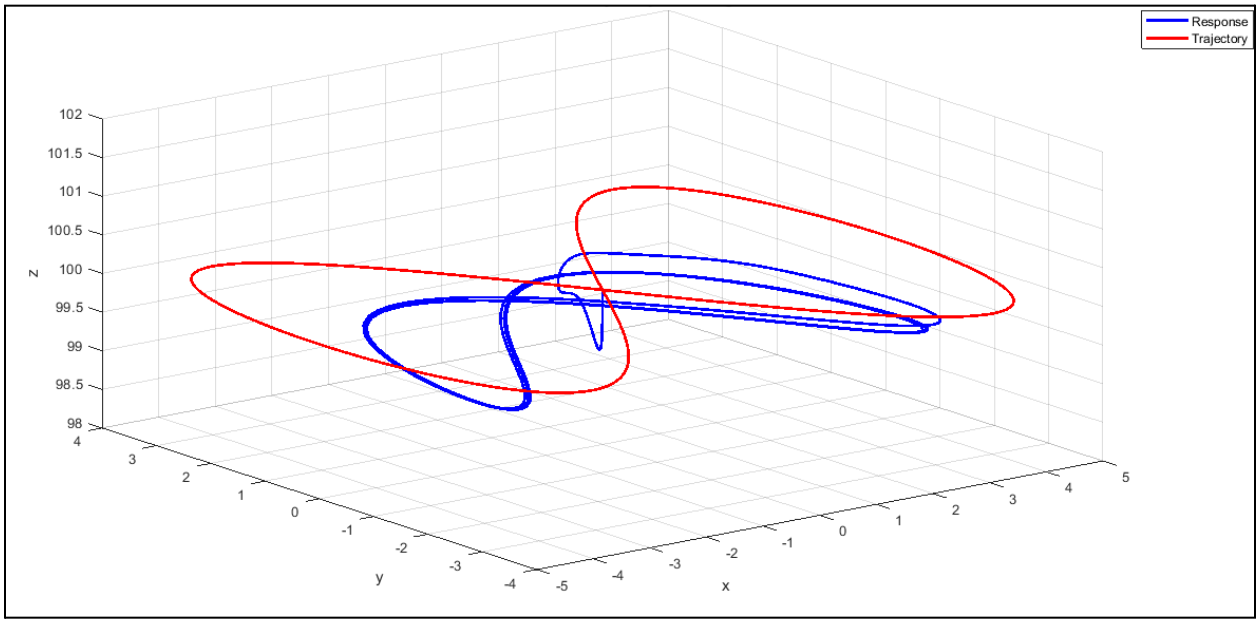


Figure 5.2.41. *3D Response of the Quadrotor.*

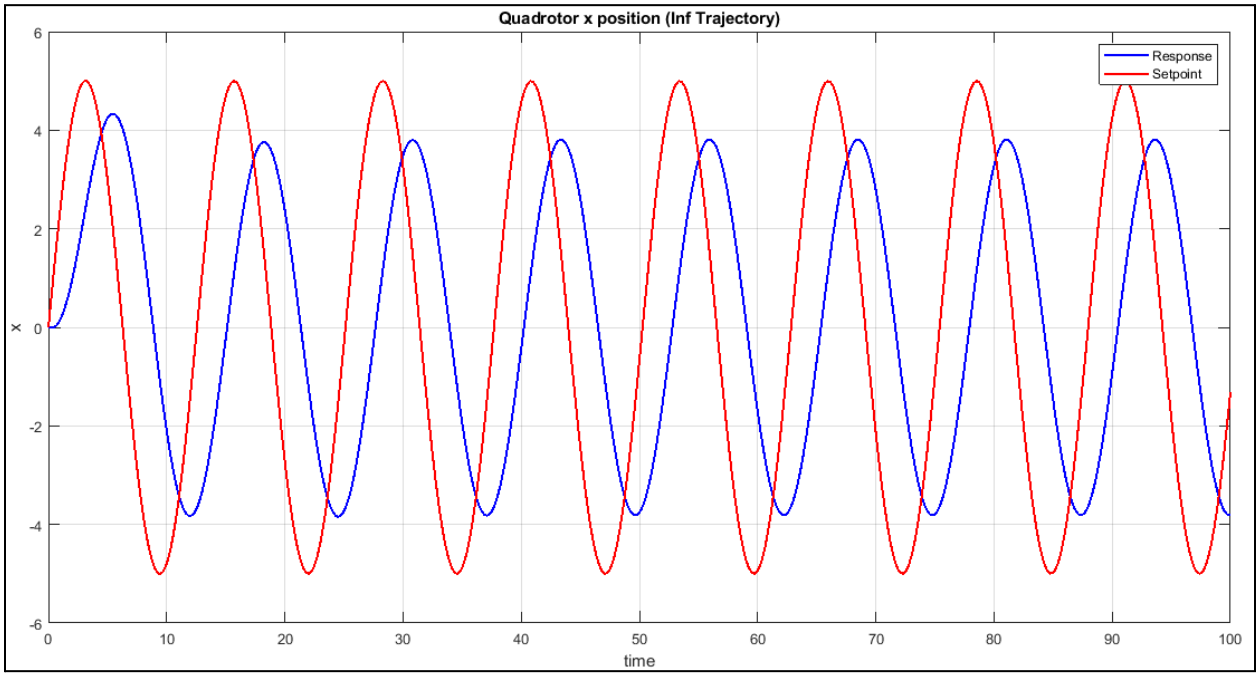


Figure 5.2.42. *X Position Response.*

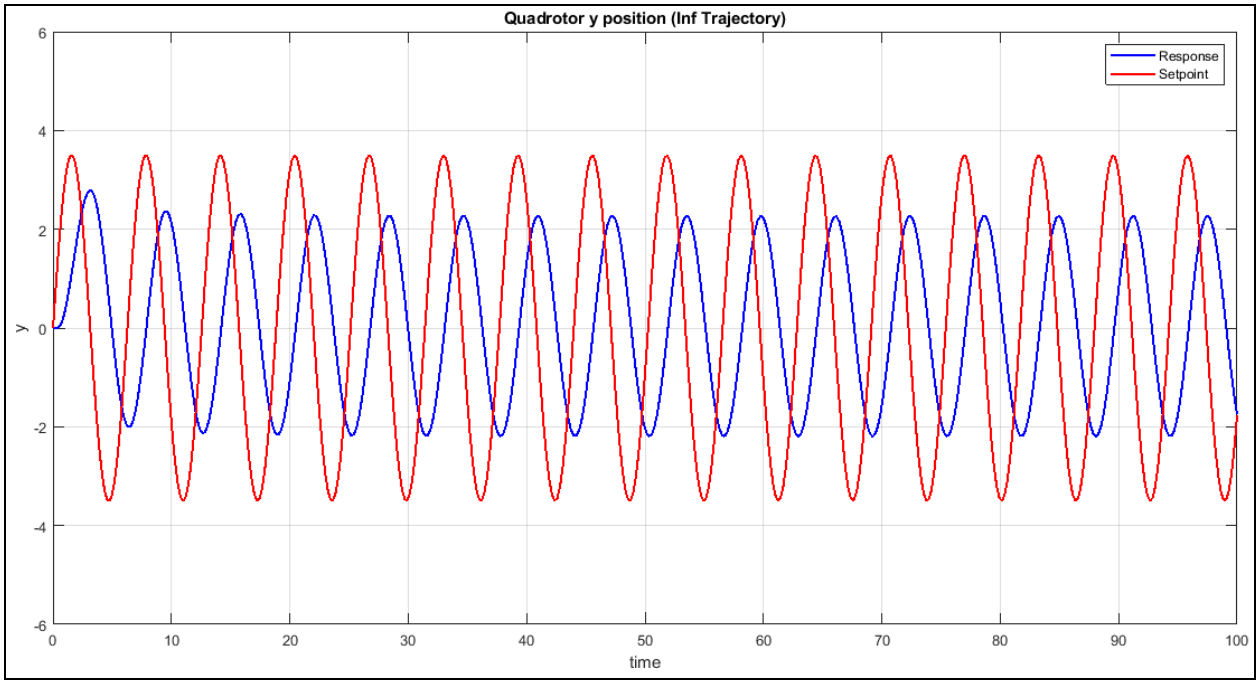


Figure 5.2.43. *Y Position Response.*

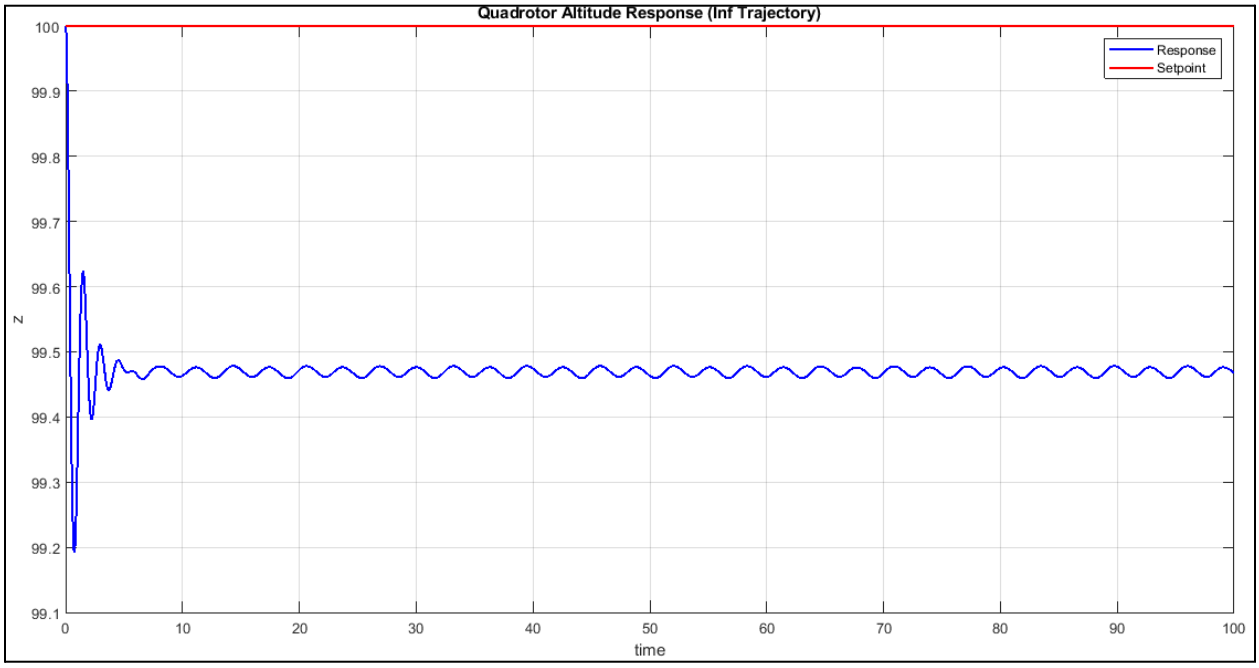


Figure 5.2.44. *Z Position Response.*

Inner Loop Response

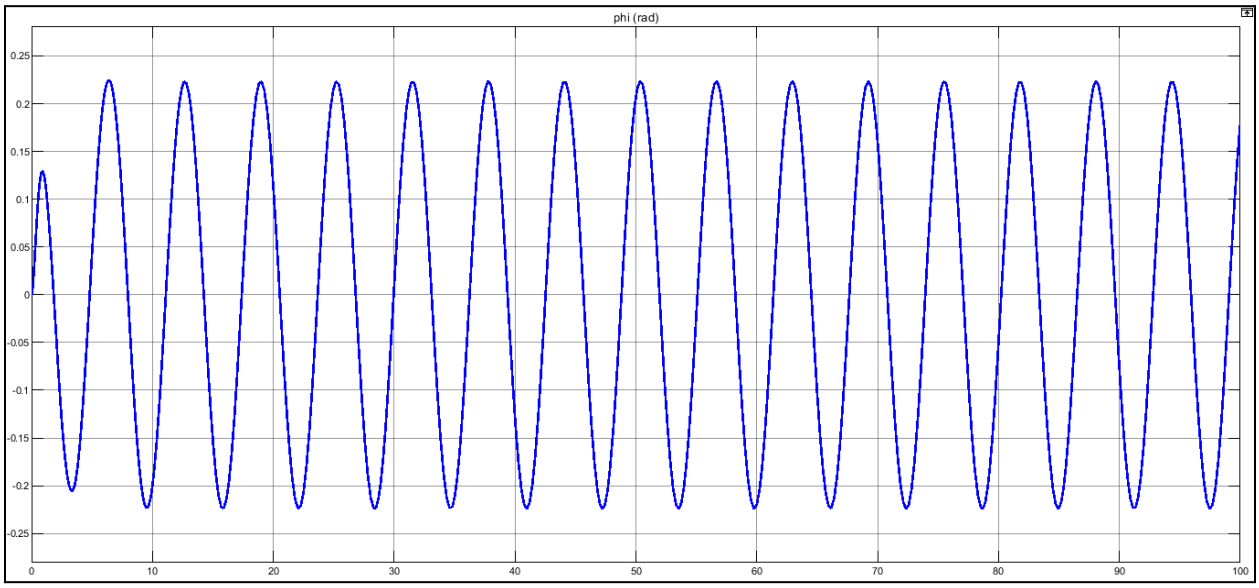


Figure 5.2.45. *Phi Angle Response.*

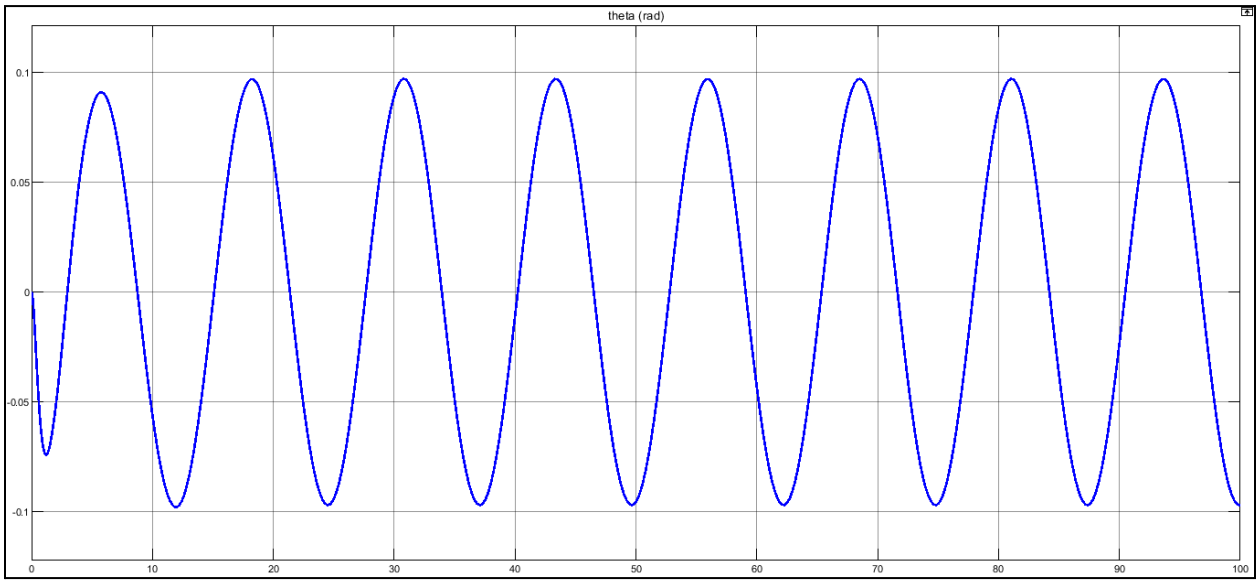


Figure 5.2.46. *Theta Angle Response.*

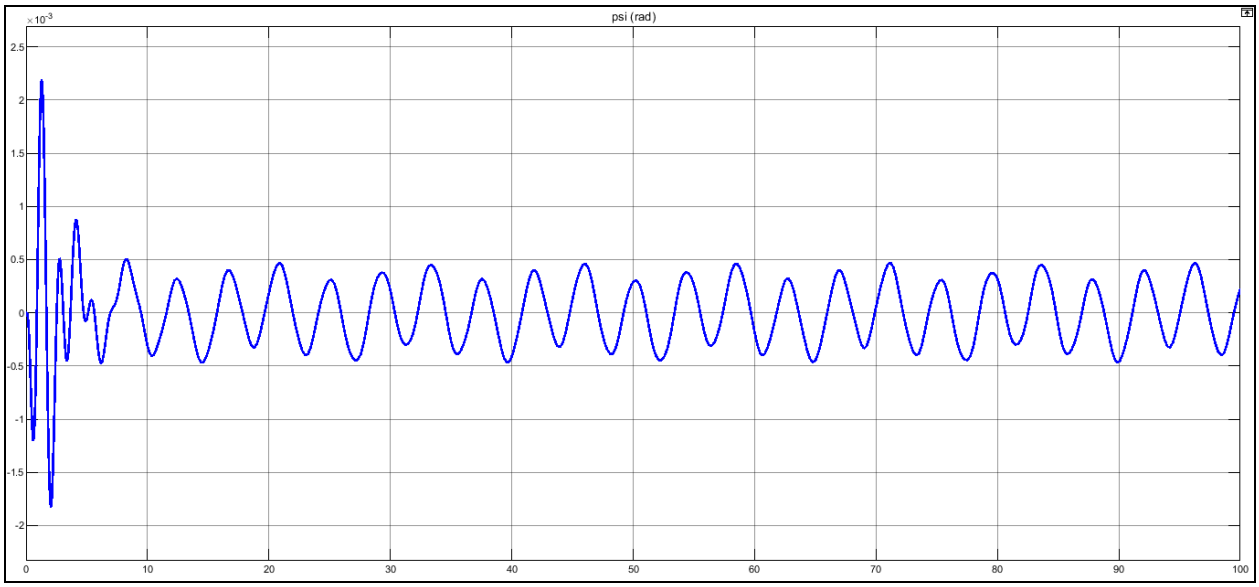


Figure 5.2.47. *Psi Angle Response.*

Discussion

The PID controller achieved satisfactory results tracking the circle trajectory, with errors in x, y, and z of 12%, 4%, and 0.5% respectively. The error in the x position could be reduced by further tuning of the controller so as to achieve better results.

5.2.3. Comparison between PID and NMPC

This comparison section focuses on contrasting the performance of the previously implemented cascaded PID controller and the aforementioned Nonlinear Model Predictive Controller approach for a quadrotor system. Three key aspects considered for comparison include reference tracking, disturbance rejection, and noise resilience.

Regarding reference tracking, the PID controller, known for its simplicity, exhibits satisfactory performance for basic reference trajectories due to its linear nature. However, when faced with more complex reference trajectories or nonlinear dynamics, the limitations of the PID controller become apparent (Spitzer, 2022). On the other hand, NMPC, with its ability to optimize control inputs over a finite prediction horizon, offers improved tracking performance, especially for intricate reference trajectories and challenging operating conditions (Spitzer, 2022) (Islam, & Okasha, 2019).

In terms of disturbance rejection, the PID controller struggles to effectively mitigate disturbances that impact the quadrotor's dynamics. Although the integral term can reduce

steady-state errors caused by disturbances, it may not adequately handle sudden or significant disturbances. In contrast, NMPC explicitly incorporates disturbance models or adapts the prediction model during the optimization process, enabling superior disturbance rejection. This feature makes NMPC more suitable for scenarios with anticipated or present disturbances.

Furthermore, the impact of noise addition on control performance is essential to consider. PID controllers, due to their linear nature, may exhibit limited resilience to noise, as they are unable to explicitly account for uncertainties (Islam, & Okasha, 2019). Conversely, NMPC demonstrates enhanced robustness against noise by incorporating state estimation techniques and employing robust optimization methods. This ability to effectively handle uncertainties and adapt control inputs contributes to improved control performance and stability in the presence of noise(Spitzer, 2022).

By evaluating the reference tracking capability, disturbance rejection, and noise resilience, a comprehensive analysis is conducted to assess the advantages and limitations of both PID control and NMPC for quadrotor control. This comparison provides valuable insights into the suitability of each control approach based on the specific requirements and challenges of the quadrotor system under investigation (Islam, & Okasha, 2019).

5.2.3.1. Reference Tracking

For the reference tracking comparison, both controllers are given the same reference trajectory to follow under the same constraints. Controllers are expected to follow the given reference trajectory with minimal overshoot and delay. Following an infinity-shaped trajectory while hovering at 100 m ($x = 5\sin(0.5t)$, $y = 3.5\sin(t)$, and $z = 100$).

The NMPC follows the trajectory with minor errors such as; an overshoot in the x direction of 15.02% and a steady state error in the z direction of 0.3 m (0.3%), which is shown in the states response for such a trajectory in Fig. 5.2.5. Noticeably, the NMPC control action makes the quadrotor motion very smooth following the trajectory however having sharp maneuvers to make. Besides, the control action response shown in Fig. 5.2.6 shows no

aggressive control action.

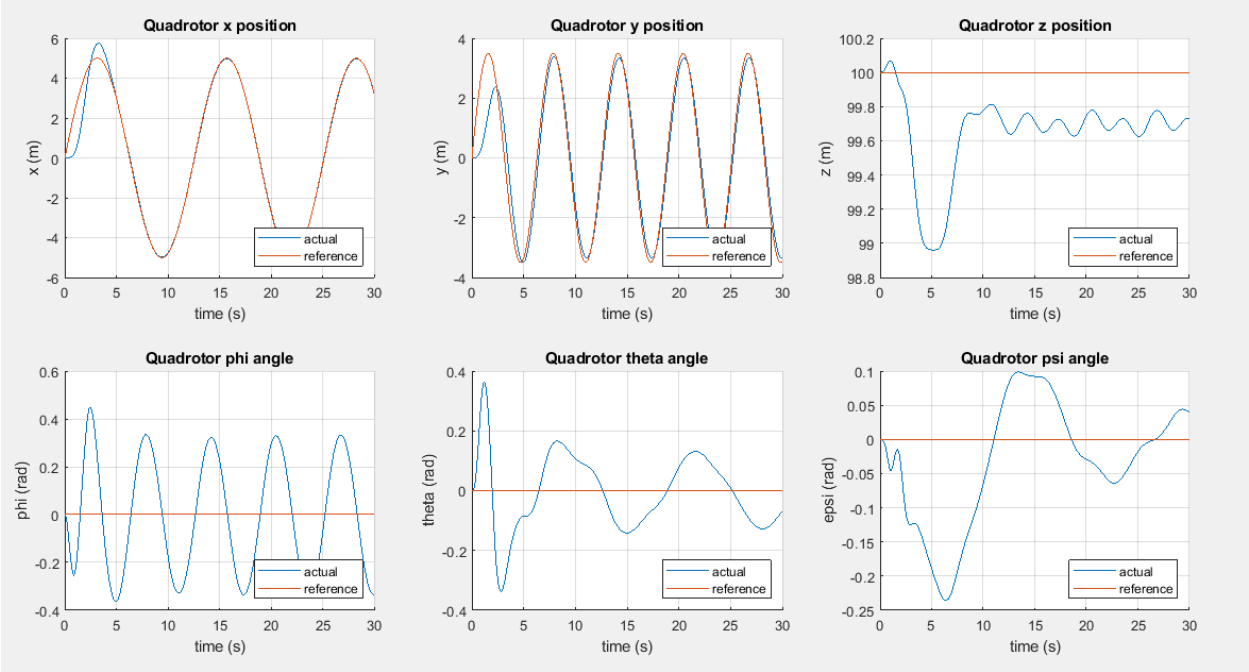


Fig. 5.2.48. The states' responses for an infinity-shaped trajectory while hovering at 100 m.

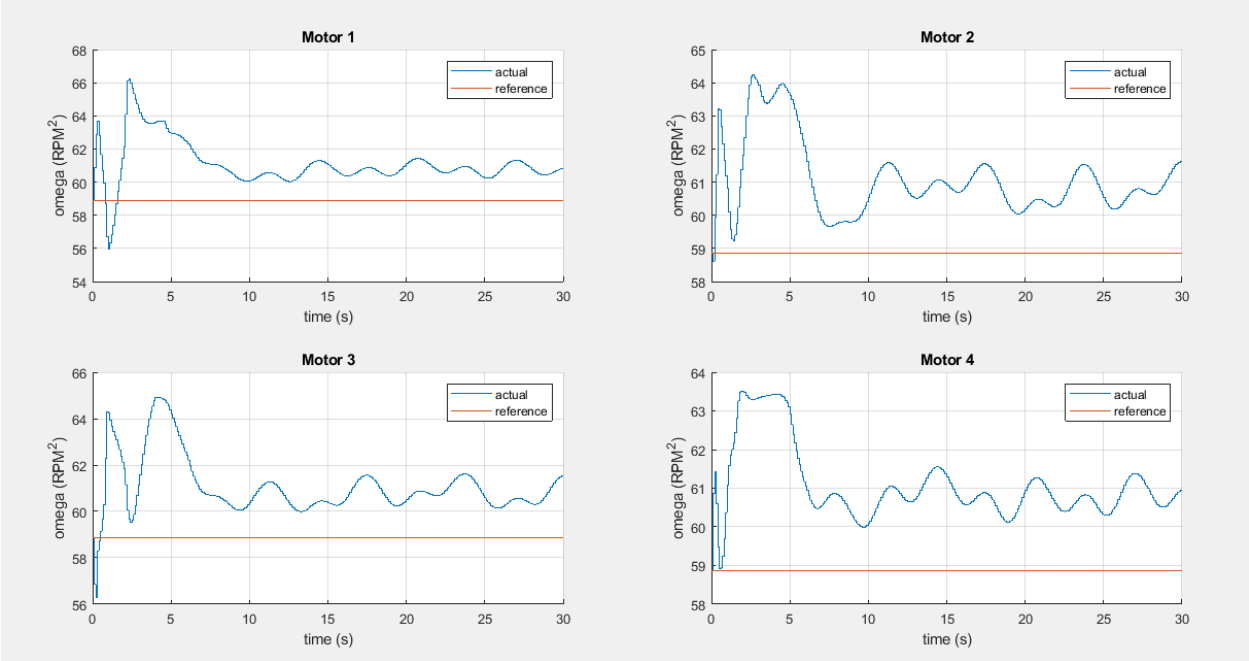


Fig. 5.2.49. The NMPC control actions to follow an infinity-shaped trajectory while hovering at 100 m.

As for the PID response, the controller did not cause any overshoot in tracking the position setpoints in x, y, and z. However, steady-state errors of around 12% occurred in both x

and y. As for the altitude response, the controller managed to maintain the aircraft at the desired setpoint but with an error of 0.5%. Accordingly, NLMPC outperformed the PID controller in tracking an aggressive trajectory.

5.2.3.2. Disturbance Rejection

For the disturbance rejection comparison, a minor step disturbance is introduced to one of the angles while hovering, the system then is observed to check the controllers' recovery of the hovering state rapidly and steadily. The responses of the PID and NMPC are expected to resist the disturbance introduced and recover the quadrotor hover state smoothly with no aggressive responses.

Introducing a 10-degree step disturbance to the phi angle while hovering at 100 m, the state response for 40 s of the NMPC control actions, as shown in Fig.5.2.7, and 5.2.8, indicates that the system y and z positions have settled already but the x position and the theta angle seem to be diverging, however, it still within very small values (magnitudes of $10^{-14} and\ 10^{-15}$), so a longer response is needed to judge.

Running another 10-degree step disturbance to the phi angle while hovering at 100 m for 1000s, the state response in Fig. 5.2.9 shows that the x position, z position, phi angle, and epsi angle have already totally settled, meanwhile, the y position and the theta angle are oscillating within very low magnitudes (vibrating). It is also worth noting that the motor speed (the NMPC actions) has already been set to its nominal hovering value very early in the simulation, as Fig. 5.2.10 states. This means that the vibrations happening in the system are not caused by the controller but rather inherited from the system model itself.

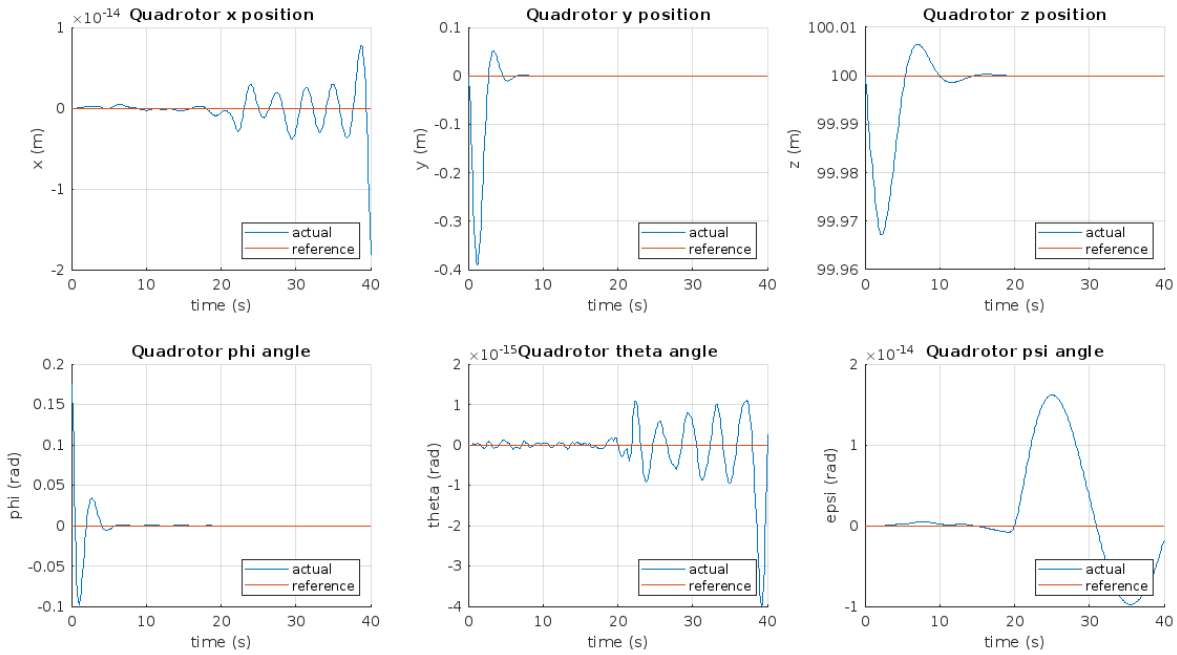


Fig. 5.2.50. The states' responses for a 10 deg disturbance in the phi angle while hovering at 100 m for 40 s.

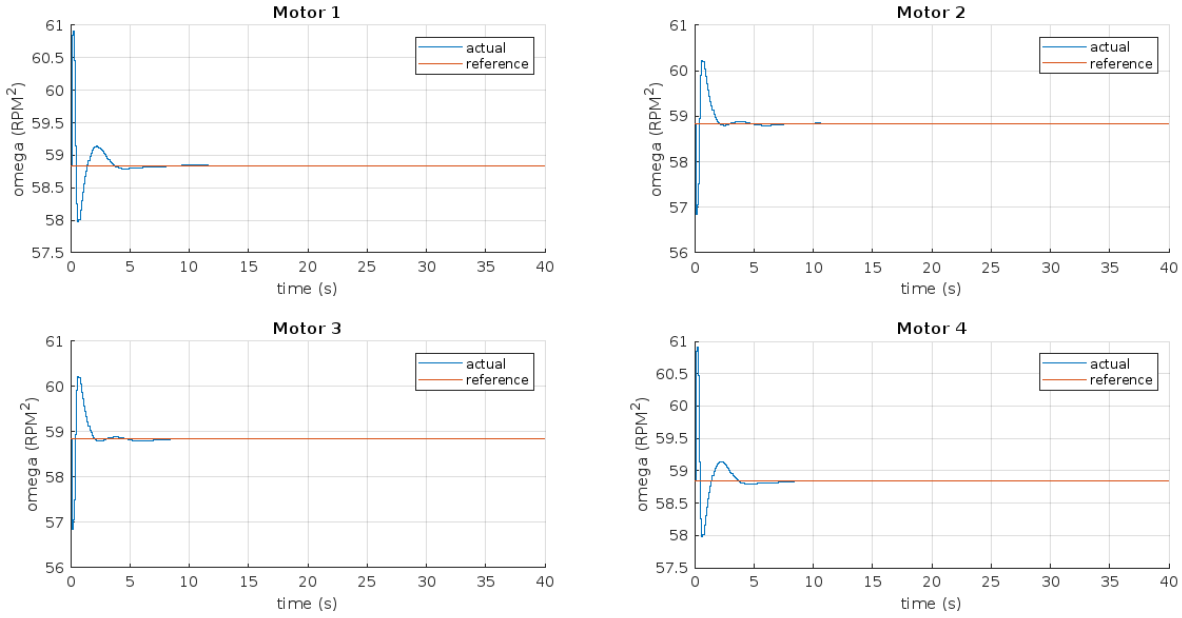


Fig. 5.2.51. The control action responses for a 10 deg disturbance in the phi angle while hovering at 100 m for 40 s.

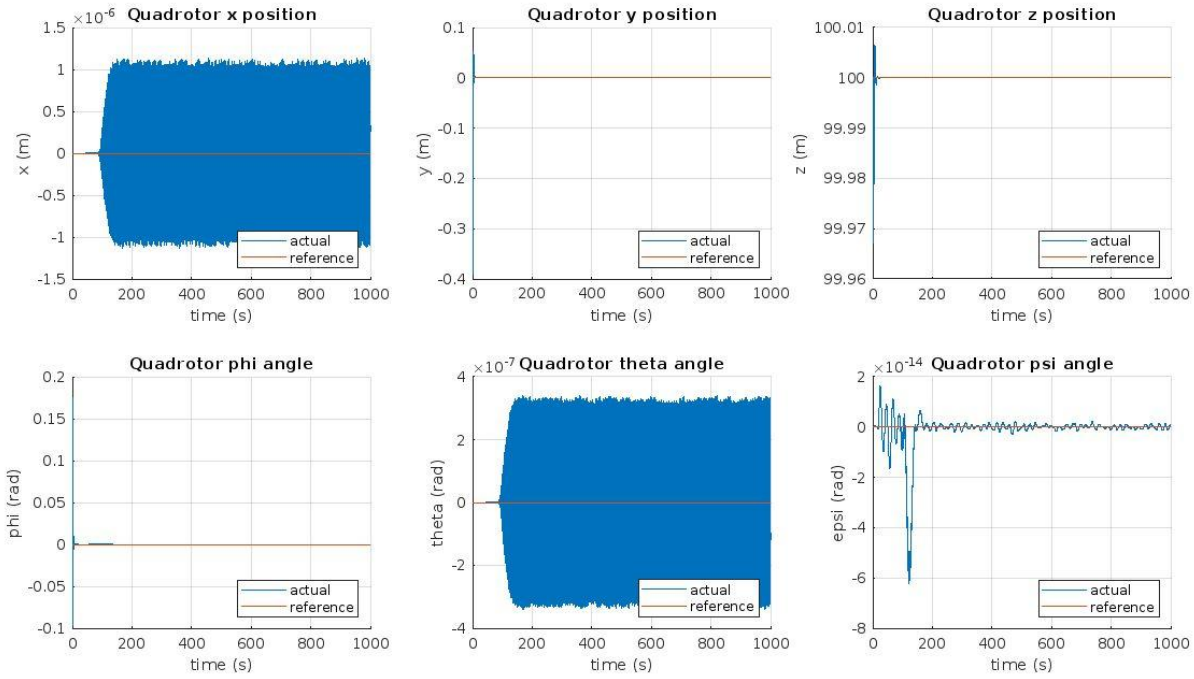


Fig.5.2.52. The states' responses for a 10 deg disturbance in the phi angle while hovering at 100 m for 1000 s.

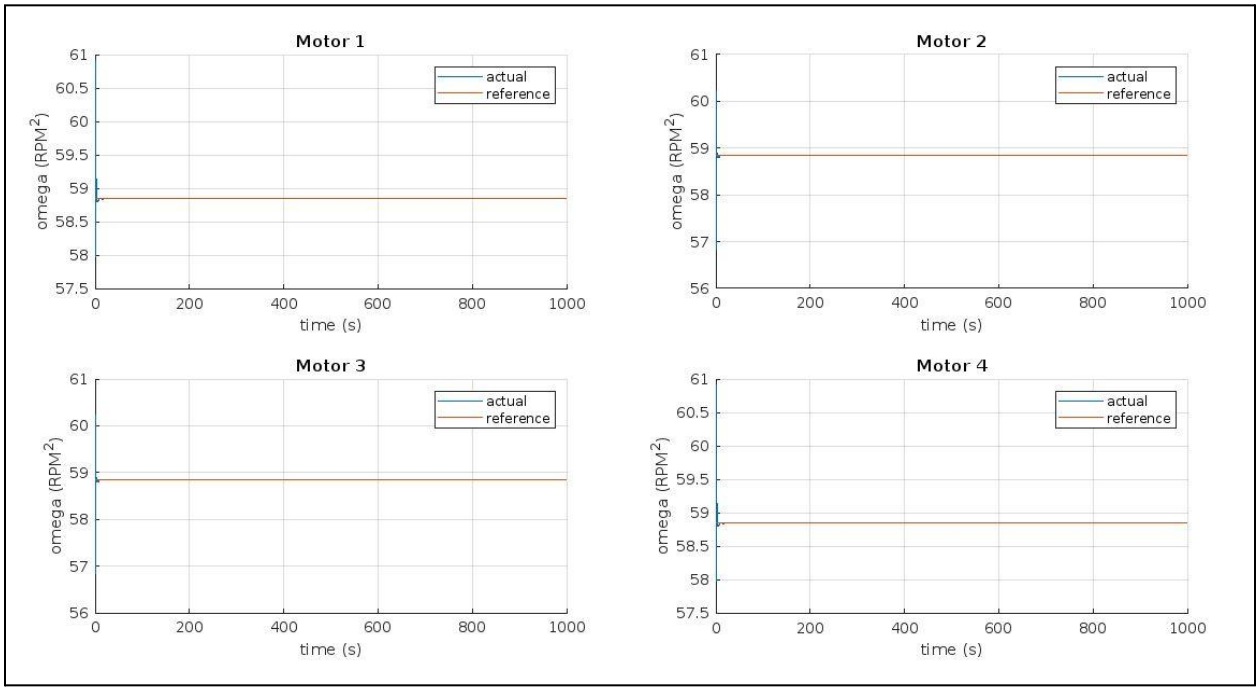


Fig.5.2.53. The control action responses for a 10 deg disturbance in the phi angle while hovering at 100 m for 1000 s.

The same procedure is repeated for the theta angle with a disturbance of 10 degrees. Fig.5.2.11, and 5.2.12 draw similar results as that of the phi angle disturbance, but this time the y position, epsi, and phi angles are the ones that seem to be diverging and need a longer response to judge while the other parameters settle already. Again, in Fig. 5.2.13, 5.2.14 when the simulation response is extended to 1000 s. It highlights the same conclusions reached with the phi angle disturbance responses, that the epsi keeps oscillating within a very low magnitude (vibrating) and again it is obvious that it is from the system model and not caused by a control action.

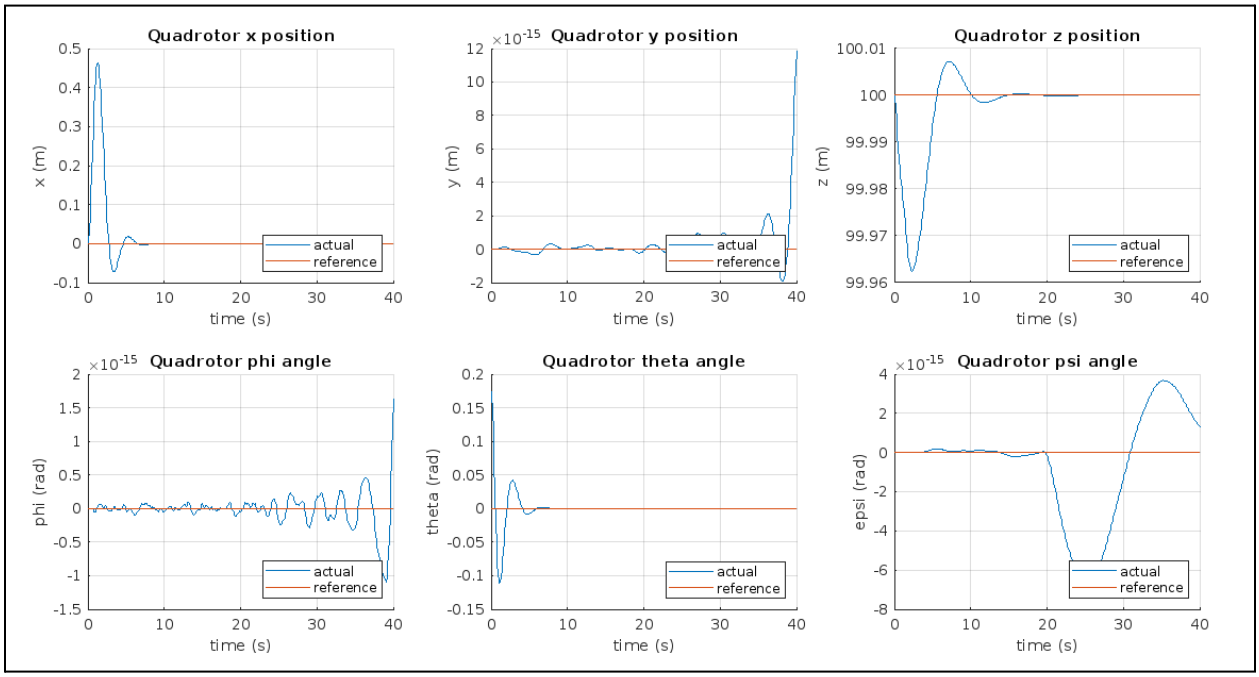


Fig.5.2.54. The states' responses for a 10 deg disturbance in the theta angle while hovering at 100 m for 40 s.

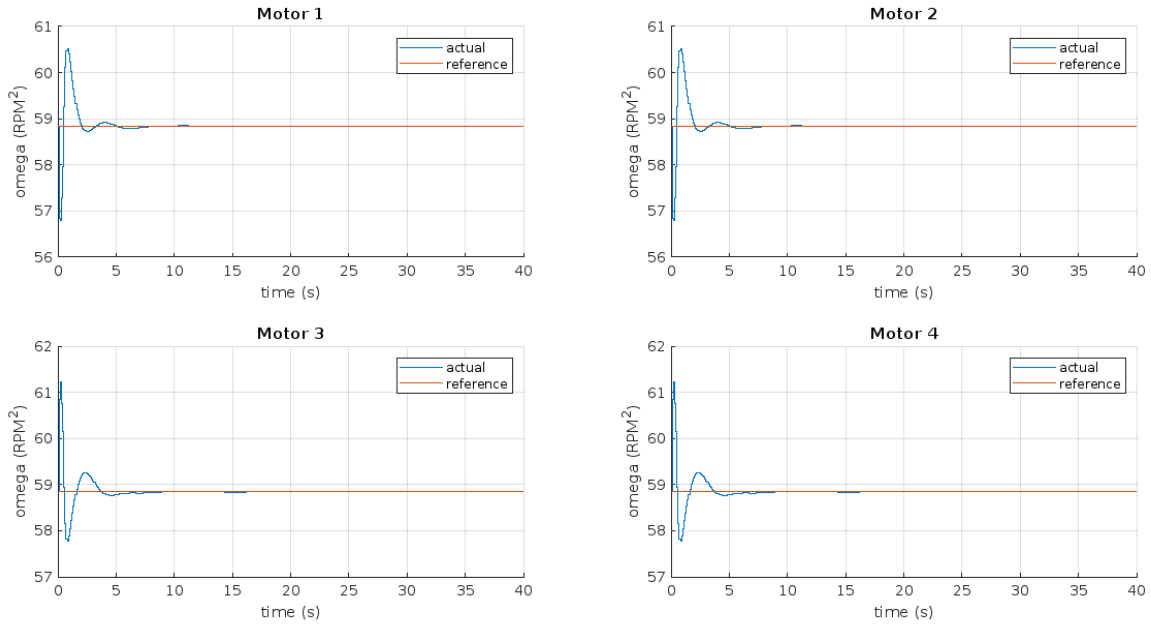


Fig.5.2.56. The control action responses for a 10 deg disturbance in the theta angle while hovering at 100 m for 40 s.

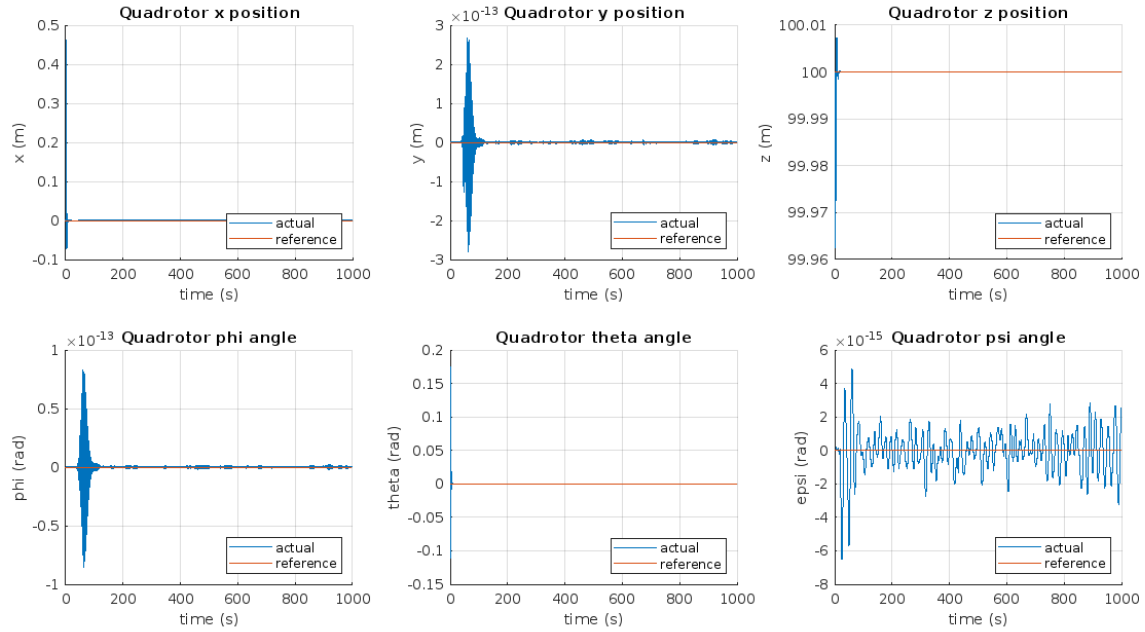


Fig.5.2.57. The states' responses for a 10 deg disturbance in the theta angle while hovering at 100 m for 1000 s.

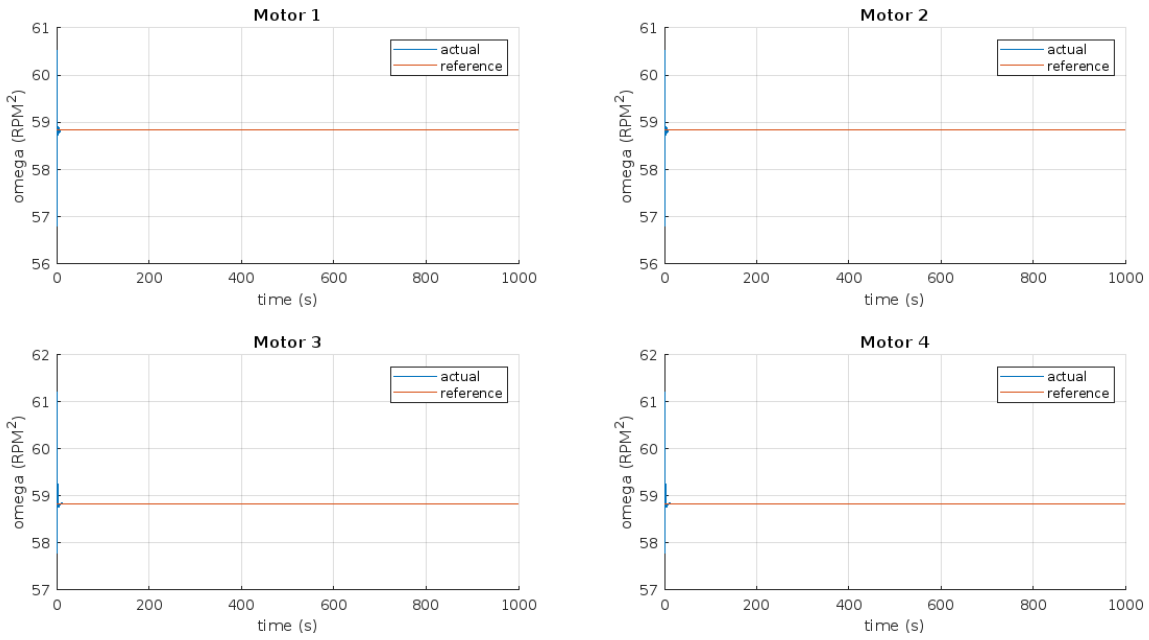


Fig.5.2.58. The control action responses for a 10 deg disturbance in the theta angle while hovering at 100 m for 1000 s.

Once more the procedure done with the phi and theta disturbances is repeated for the espi and a similar conclusion could be drawn, but this time for the x position and theta angle. The results are highlighted in Fig. 5.2.15, 5.2.16, 5.2.17, and 5.2.18 below.

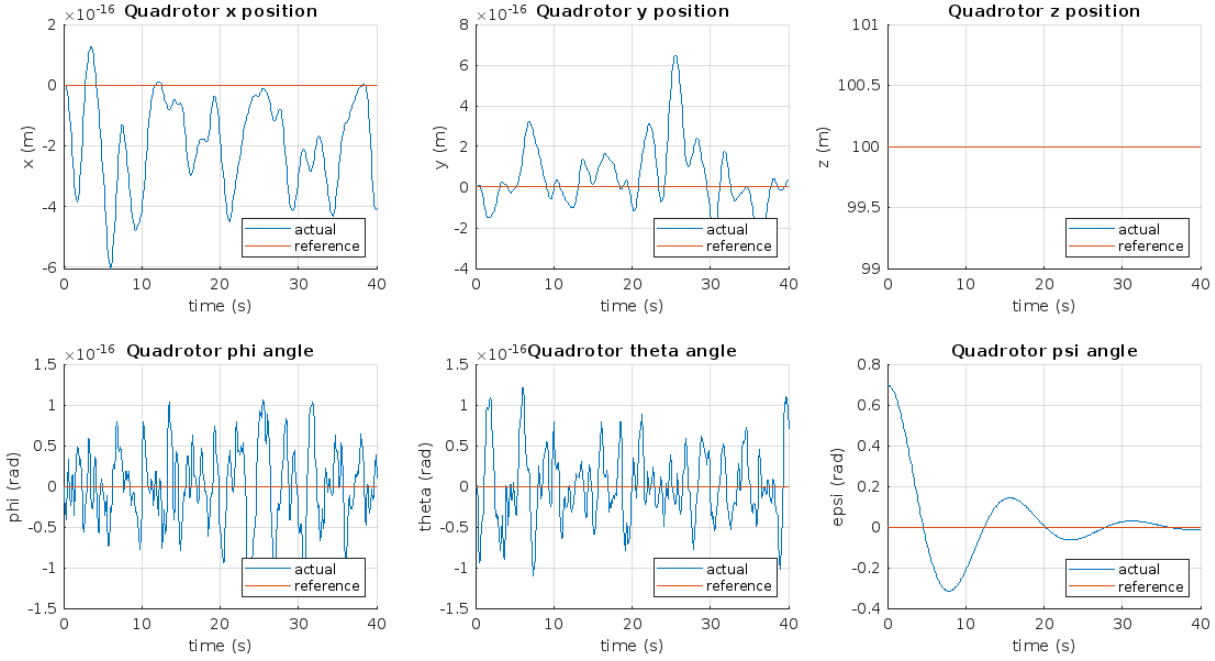


Fig.5.2.59. The states' responses for a 10 deg disturbance in the epsi angle while hovering at 100 m for 40 s.

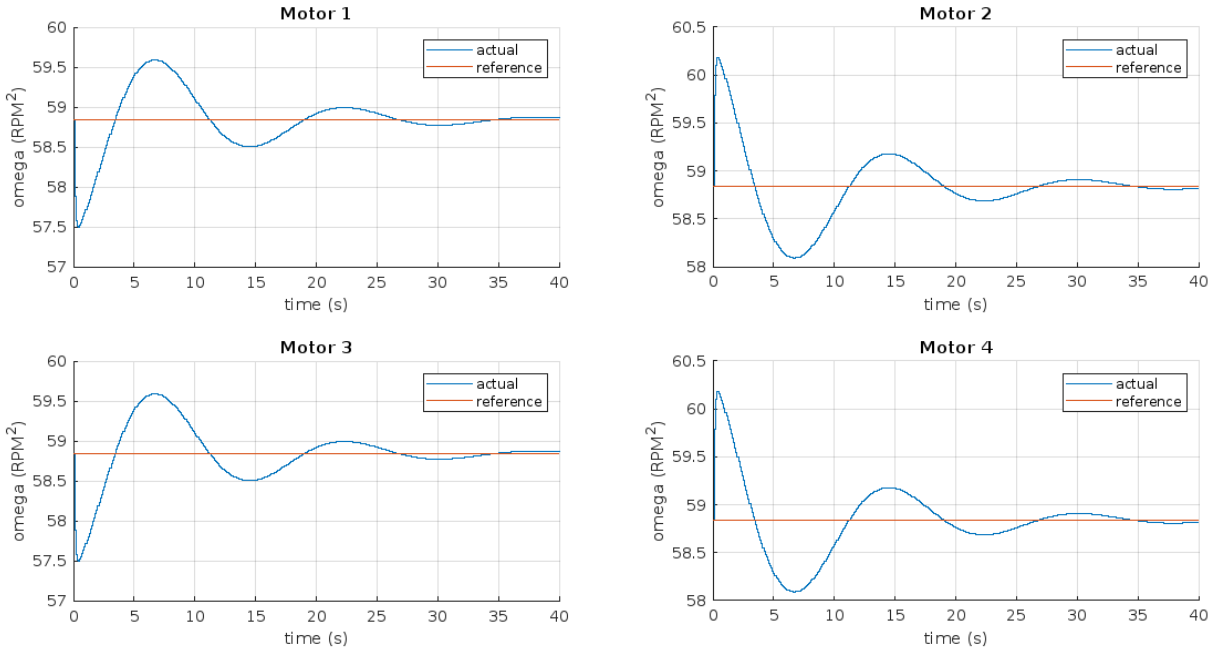


Fig.5.2.60. The control action responses for a 10 deg disturbance in the epsi angle while hovering at 100 m for 40 s.

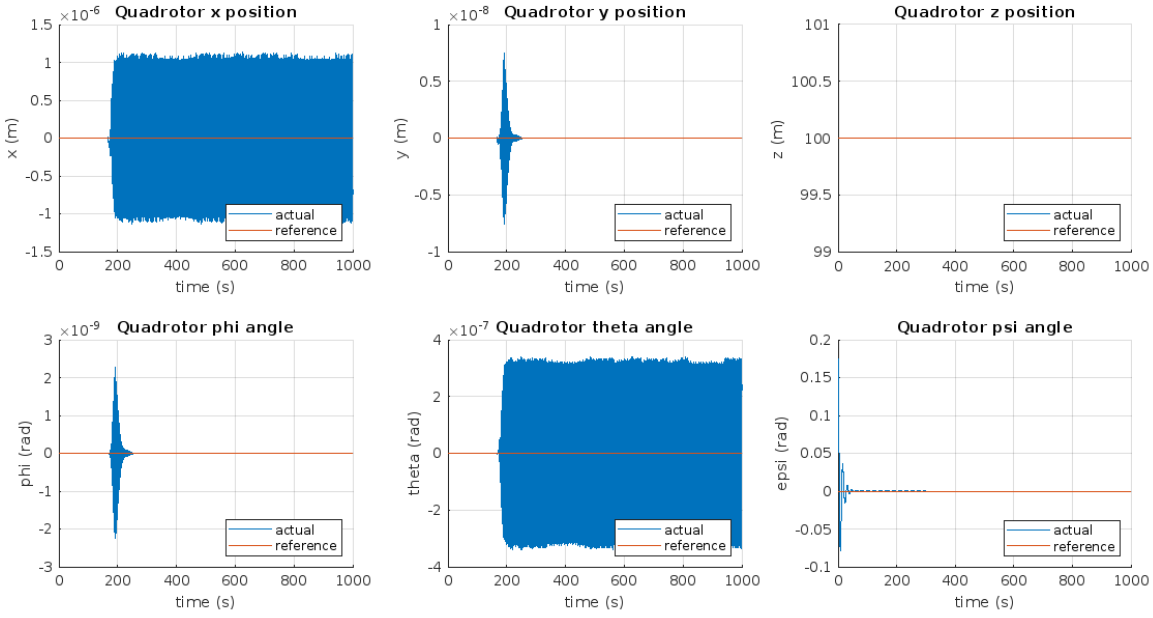


Fig.5.2.61. The states' responses for a 10 deg disturbance in the epsi angle while hovering at 100 m for 1000 s.

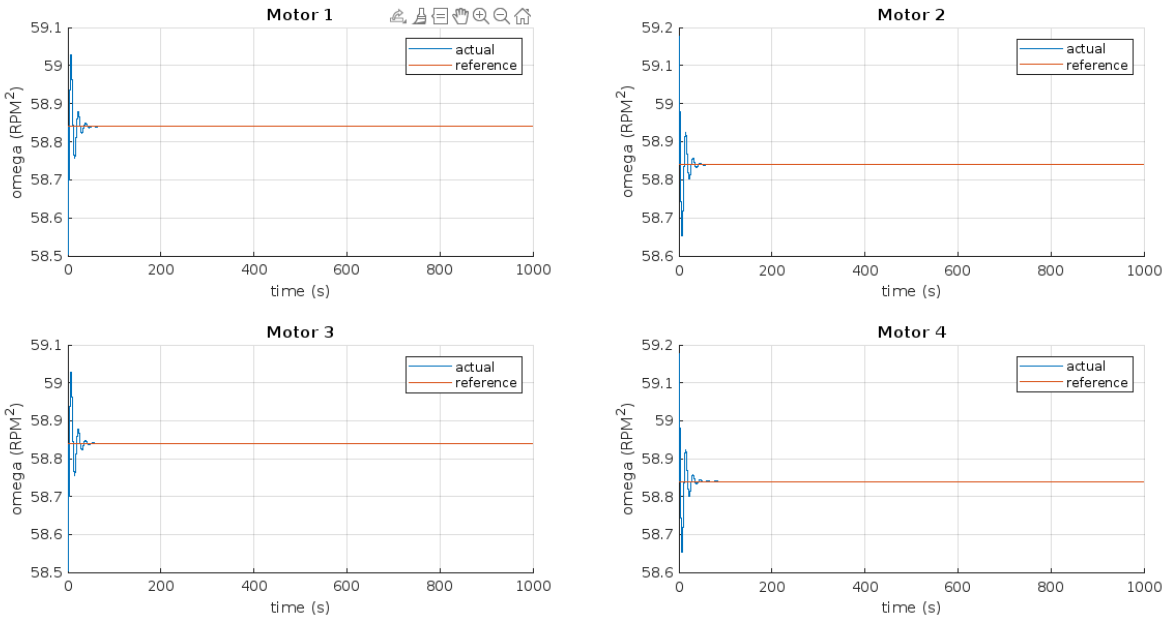


Fig.5.2.62. The control action responses for a 10 deg disturbance in the epsi angle while hovering at 100 m for 1000 s.

Another final disturbance for the three angles together shows, in Fig.5.2.19, and 5.2.20, that the quadrotor totally recovers its initial state of hovering without any vibrations as before.

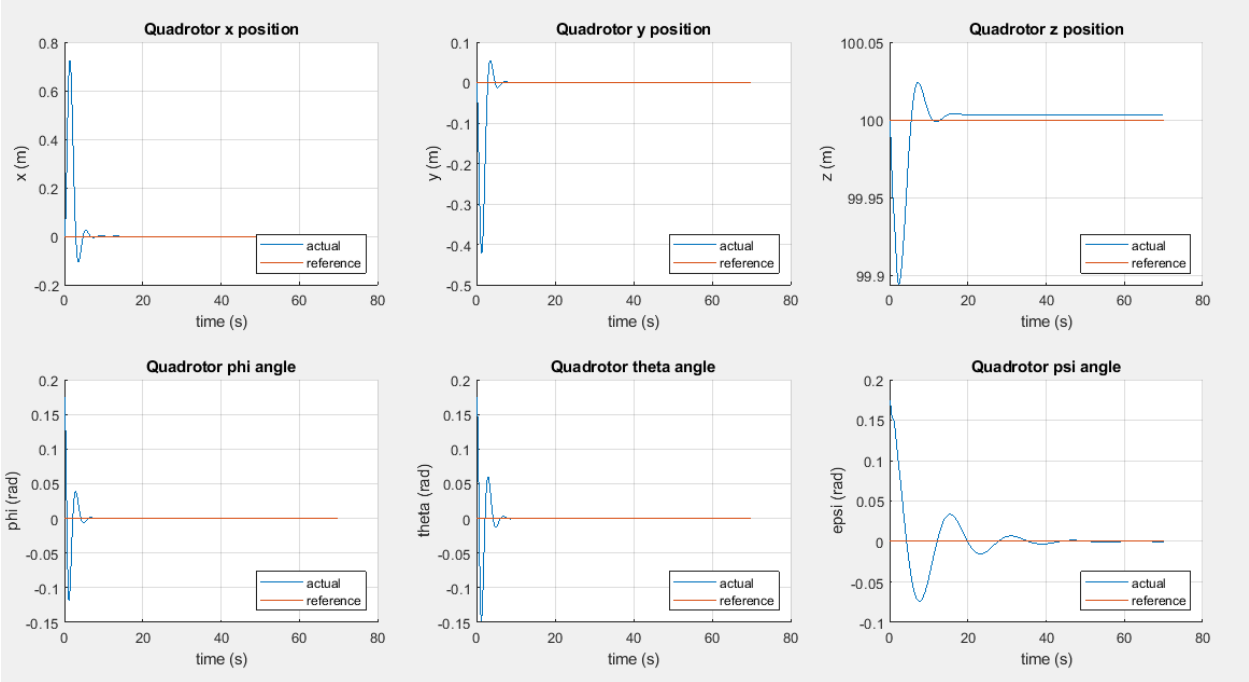


Fig.5.2.63. The states' responses for a 10 deg disturbance in all three angles while hovering at 100 m for 70 s.

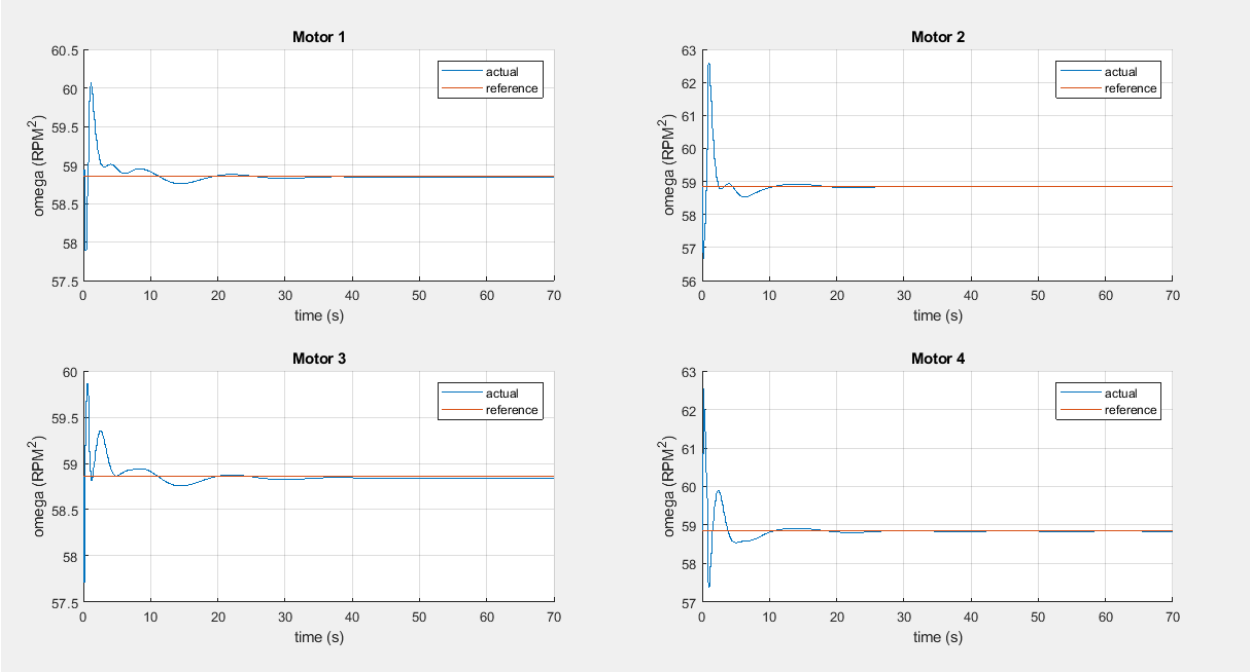


Fig.5.2.64. The control action responses for a 10 deg disturbance in all three angles while hovering at 100 m for 70 s.

As for the PID controller, it was able to perform well while experiencing an initial disturbance in both roll and pitch angles. The controller was able to recover quickly and return to the desired steady-state value. Accordingly, both PID and NLMPC performed well in disturbance rejection.

Chapter 6: Conclusion

6.1. Fixed Wing

Most of the controllers which have been implemented for the fixed wing UAV, have achieved their purpose of stabilizing the aircraft. However, in terms of their implementation, computational power, and response specifications, they are different. The obtained graphs and results from the Simulink have indicated that MPC has a better response with pitch angle disturbance compared to TECS using the GA gains. Moreover, MPC is a MIMO controller which makes it easier to control both longitudinal and lateral modes at the same time. Despite the LQR being a MIMO controller, the tuning process of LQR is exhaustive and the extension of using it in a nonlinear system is not efficient in this project: making it the least favorite candidate for fixed-wing control. Although the FLC algorithm is the simplest among them all, its computational power is higher, and as the input parameters number increases, the harder the FLC tuning process. So far, MPC has proven to be the better controller in our work.

MPC better response with pitch angle disturbance, controlling simultaneously both longitudinal and lateral modes.

LQR tuning process is exhaustive

FLC requires a high computational power, and the tuning process gets harder with the increase of input parameters

TECS

6.2. Quadrotor

For the quadrotor part of the VTOL, two controllers are applied to achieve stability, a high command of maneuverability, as well as disturbances immunity that might be caused due to a strong wind or noisy data read by the sensors.

Firstly, a Non-Linear Model Predictive Controller is designed for the quadrotor. The NMPC implementation required a special study of the system in order to make an appropriate design of the NMPC for the study's specific quadrotor. Keeping in mind that the study's quadrotor has a relatively high moment of inertia that makes the quadrotor a bit hard to control

and required more aggressive control action to achieve a good command of it. The NMPC results achieve a high level of controllability on the system, follow perfectly the commanded reference trajectories, and reject and recover quickly from small angle disturbances.

Secondly, a cascaded PID control architecture was implemented for the quadrotor. The tuning approach utilized the Ziegler-Nichols tuning method to obtain a good initial estimation for the PID gains. This helped in narrowing down on the search space for the Genetic Algorithm optimization method to refine the gains and produce better responses. The modified PID was able to act efficiently in terms of disturbance rejection and provided an acceptable response in transient response. Moreover, the controller was tested for trajectory tracking on two trajectories; a circle and an infinity sign. It performed relatively well with tolerable error for the circle trajectory, but the controller was unable to follow more aggressive trajectories like the infinity sign.

References

Chipade, V. S., Abhishek, Kothari, M., & Chaudhari, R. R. (2018). Systematic design methodology for development and flight testing of a variable pitch quadrotor biplane VTOL UAV for payload delivery. *Mechatronics*, *55*, 94–114. https://doi.org/10.1016/j.mechatronics.2018.08.008

Khuwaja, K., Lighari, N.-Z., Tarca, I. C., & Tarca, R. C. (2018). PID controller tuning optimization with genetic algorithms for a Quadcopter. *Recent Innovations in Mechatronics*, *5*(1.). https://doi.org/10.17667/riim.2018.1/11

Sheta, A., Braik, M., Maddi, D. R., Mahdy, A., Aljahdali, S., & Turabieh, H. (2021). Optimization of PID controller to stabilize quadcopter movements using meta-heuristic search algorithms. *Applied Sciences*, *11*(14), 6492. https://doi.org/10.3390/app11146492

Advanced Fixed-wing Position Tuning | PX4 User Guide. (n.d.). https://docs.px4.io/v1.12/en/config_fw/advanced_tuning_guide_fixedwing.html

Bilal, Pant, M., Zaheer, H., García-Hernández, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*, *90*, 103479. https://doi.org/10.1016/j.engappai.2020.103479

Cortés-Martínez, R., & Rodriguez-Cortes, H. (2019). A Total Energy Attitude Control System Strategy for Rigid Spacecraft. *IEEE Access*, *7*. https://doi.org/10.1109/access.2019.2934424

Darwish, M., Elgohary, A. A., Gomaa, M. M., Kaoud, A., Ashry, A., & Taha, H. E. (2022). A Comparative Assessment of the Performance of PID and MPC Controllers: A UAV Altitude Hold Autopilot Case Study. In *AIAA SCITECH 2022 Forum*. https://doi.org/10.2514/6.2022-1519

Degaspare, T. G., & Kienitz, K. H. (2022). Flight and engine control laws integration based on robust control and energy principles. *CEAS Aeronautical Journal*, *13*(4), 905–921. https://doi.org/10.1007/s13272-022-00599-x

Ducard, G., & Allenspach, M. (2021). Review of designs and flight control techniques of hybrid and convertible VTOL UAVs. *Aerospace Science and Technology*, *118*, 107035. https://doi.org/10.1016/j.ast.2021.107035

Fraga-Gonzalez, L. F., Fuentes-Aguilar, R. Q., Garcia-Gonzalez, A., & Sanchez-Ante, G. (2017). Adaptive simulated annealing for tuning PID controllers. *Ai Communications*, *30*(5), 347–362. https://doi.org/10.3233/aic-170741

GeeksforGeeks. (2023). Particle Swarm Optimization PSO An Overview. *GeeksforGeeks*. https://www.geeksforgeeks.org/particle-swarm-optimization-pso-an-overview/

Georgioudakis, M., & Plevris, V. (2020). A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization. *Frontiers in Built Environment*, *6*. https://doi.org/10.3389/fbuil.2020.00102

Haq, E. U., Ahmad, I., Hussain, A., & Almanjahie, I. M. (2019). A Novel Selection Approach for Genetic Algorithms for Global Optimization of Multimodal Continuous Functions. *Computational Intelligence and Neuroscience*, *2019*, 1–14. https://doi.org/10.1155/2019/8640218

Jiménez, P. A., Lichota, P., Agudelo, D. E., & Rogowski, K. (2019). Experimental Validation of Total Energy Control System for UAVs. *Energies*, *13*(1), 14. https://doi.org/10.3390/en13010014

Lambregts, A. A. (2013). TECS Generalized Airplane Control System Design – An Update. In *Springer eBooks* (pp. 503–534). https://doi.org/10.1007/978-3-642-38253-6_30

Singh, A., & Kumar, S. (2016). Differential Evolution: An Overview. In *Advances in intelligent systems and computing*. Springer Nature. https://doi.org/10.1007/978-981-10-0448-3_17

Wang, D., Tan, D., & Liu, L. (2017). Particle swarm optimization algorithm: an overview. Soft Computing, 22(2), 387–408. https://doi.org/10.1007/s00500-016-2474-6

Nise, N. S. (2019). Nise's control systems engineering. John Wiley & Sons.

WPI. (2021). PID Settings Tuning. World Precision Instruments. https://www.wpi-europe.com/solutions/fluid-handling-knowledgebase/setting-up-smartflo-when-using-a-flow-sensor.aspx

Bolton, W. (2021). Instrumentation and control systems. Elsevier, Newnes is an imprint of Elsevier.

Meshram, P.M., & Kanojiya, R.G. (2012). Tuning of PID controller using Ziegler-Nichols method for speed control of DC motor. IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012), 117-122.

Ahmmed, T., Akhter, I., Rezaul Karim, S. M., & Sabbir Ahamed, F. A. (2020). Genetic algorithm based PID parameter optimization. American Journal of Intelligent Systems, 10(1), 8–13. https://doi.org/10.5923/j.ajis.20201001.02

Mathworks. (2023). Genetic Algorithm Settings . Find the minimum of function using genetic algorithm - MATLAB. https://www.mathworks.com/help/gads/ga.html

Jiřinec, T. (2011). Stabilization and Control of Unmanned Quadcopter. Zech Technical University in Prague, Faculty of Electrical Engineering [Master's Thesis]. http://ltu.diva-portal.org/smash/record.jsf?pid=diva2:1020192

Mellinger, Warren, D. (2012). Trajectory Generation and Control for Quadrotors. University of Pennsylvania, Mechanical Engineering and Applied Mechanics.[Doctor of Philosophy Thesis]. https://repository.upenn.edu/edissertations/547

Niemiec, R., & Gandhi, F. (2016). A comparison between quadrotor flight configurations. Proceedings of the 42nd European Rotorcraft Forum, Lille, France. https://dspace-erf.nlr.nl/xmlui/handle/20.500.11881/3748

Cook, M.V., 1997. Flight Dynamics Principles 1st ed., John Wiley & Sons

Mughees, N., Jaffery, M. H., & Jawad, M. (2022). A new predictive control strategy for improving operating performance of a permanent magnet synchronous generator-based wind energy and superconducting magnetic energy storage hybrid system integrated with grid. Journal of Energy Storage, 55, 105515. https://doi.org/10.1016/j.est.2022.105515

Homann, G. M., Huang, H., Waslander, S. L., & Tomlin, C. J. (2007). Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment. In AIAA Guidance, Navigation and Control Conference and Exhibit. https://doi.org/10.2514/6.2007-6461

Darwish, M. H., Elgohary, A. A., Gomaa, M. M., Kaoud, A. M., Ashry, A. M., & Taha, H. E. (2022). A comparative assessment of the performance of PID and MPC controllers: A UAV altitude hold autopilot case study. AIAA SCITECH 2022 Forum. https://doi.org/10.2514/6.2022-1519

Alhajeri , M. (2021). Tuning Guidelines for Model-Predictive Control. Department of Chemical and Biological Engineering Drexel University. https://doi.org/https://par.nsf.gov/servlets/purl/10299968

Bagheri, P., & Khaki Sedigh, A. (2015). Robust tuning of dynamic matrix controllers for First Order Plus Dead Time models. Applied Mathematical Modelling, 39(22), 7017–7031. https://doi.org/10.1016/j.apm.2015.02.035

Sayar, E., & Ertunç, H. M. (2018). Fuzzy Logic Controller and PID controller design for Aircraft Pitch Control. EuCoMeS 2018, 53–60. https://doi.org/10.1007/978-3-319-98020-1_7

Elgohary, A. A., Ashry, A. M., Hesham, A. M., Gomaa, M. M., & Darwish, M. H. (2020). Autopilot Controller Design and Simulation for a Fixed Wing Unmanned Arial Vehicle (thesis).

Design of optimized linear quadratic regulator for capsule endoscopes based on artificial bee colony tuning algorithm. (2018). International Journal for Engineering Modelling. https://doi.org/10.31534/engmod.2018.1-2.ri.02_vjan

Nagarkar, M. P., & Patil, V. (2016). Multi-objective optimization of LQR control quarter car suspension system using genetic algorithm. FME Transaction, 44(2), 187–196. https://doi.org/10.5937/fmet1602187n

Anjali, B. S., Vivek, A., & Nandagopal, J. L. (2016). Simulation and analysis of Integral LQR controller for Inner Control Loop design of a fixed-wing Micro Aerial Vehicle (MAV). Procedia Technology, 25, 76–83. https://doi.org/10.1016/j.protcy.2016.08.083

Nelson, R. C. (2010). Flight stability and automatic control. McGraw-Hill Education (India) Private Limited.

Wahid, N. (n.d.). *Comparative assessment using LQR and Fuzzy Logic Controller for a pitch ...* Researchgate. https://www.researchgate.net/publication/267767915_Comparative_Assessment_using_LQR_and_Fuzzy_Logic_Controller_for_a_Pitch_Control_System

Wahid, N. (n.d.). *Comparative assessment using LQR and Fuzzy Logic Controller for a pitch ...* Researchgate. https://www.researchgate.net/publication/267767915_Comparative_Assessment_using_LQR_and_Fuzzy_Logic_Controller_for_a_Pitch_Control_System

User, S. (n.d.). Downloads:. Tornado. https://www.tornado.redhammer.se/index.php/download

Appendix A: Tornado Manual

A.1 Setup

After downloading the Tornado MATLAB program from (Gohary et al.,2020), the downloaded folder will be named “T135_export”. By exporting the folder, you should find a MATLAB file named “main” and open it with MATLAB.

```
>> main

 TORNADO  Version 136.007 Beta version
 build 2021 12 10
 Main Menu

Input operations.
   [1]. Aircraft geometry setup
   [2]. Flight condition setup
   [3]. Change rudder setting
   [4]. Move reference point

Lattice operations.
   [5]. Generate lattice.

Computation operations.
   [6]. Processor access

Post processing and interactive operations.
   [7]. Post processing, Result/Plot functions
   [8]. Keyboard access

Auxiliary operations.
  [10]. About / Release Info
  [100]. Help files
   [0]. Exit Tornado

   Please enter choice from above:
```

Figure A.1.1. Tornado’s Start menu

A.2 Input operations

I. Aircraft geometry setup.

II. Flight condition setup.

III. Change the rudder setting.

IV. Move the reference point

To choose an operation, type in the number on the left of the operation.

All the data presented forward are from this project.

A.2.1 Aircraft geometry setup

By typing 1 on the main menu, the geometry submenu shows in figure A.2.1.

```
Main Menu
    |---->Geometry setup menu


   [1]. Define new geometry
   [2]. Load geometry
   [3]. Edit current geometry
   [4]. Save current geometry


   [0]. Back / up menu
```

FIgure A.2.1. Geometry submenu

By typing 1, the program prompts the user by a series of questions. The **bolded** text is the answer. The green text represents comments.

```
Geometry name: last (enter any name)
Project Reference: . (enter any reference)
Center of gravity x-coordinate: 0.16
Center of gravity y-coordinate: 0
Center of gravity z-coordinate: 0.02
Reference point x-coordinate: 0.16
Reference point y-coordinate: 0
Reference point z-coordinate: 0.02
Number of Wings: 3 (for a conventional plane, there are 3 wings: main wing,
horizontal tail, and vertical tail)
```

```
_____________________________________________________

  Data regarding wing number :1 (main wing)

_____________________________________________________
Number of semispanwise partitions for this wing: 3 (depends on the shape of the
main wing itself in addition to the number and location of the control
surface(s) if any. Keep in mind that the control surface needs a partition for
itself)
Is the wing mirrored in the xz-plane [1 0]: 1 (1 for yes, 0 for no)
Apex x-coordinate: 0
Apex y-coordinate: 0
Apex z-coordinate: 0
Is this wing an all-moving control surface [1 0]: 0
_____________________________________________________

 Wing number:1
 Data entry for partition number:1
Root chord: 0.3 (in meters)

_____________________________
Base chord airfoil: 0016 (represents the NACA0016)
Number of panels chord wise: 20 (To enter a valid number of panels, chord wise,
is an iterative method in which the numbers are increased till the aerodynamic
derivatives converge to an acceptable accuracy. For our case, it was found to
be "20")
Partition dihedral [deg]: 2
Number of panels semi-span wise: 5
Span of partition: 0.14 (span of the first partition only, not the half span of
the main wing)
Taper ratio: 0.96667
Tip chord airfoil: 0016
Quarter chord line sweep [deg]: 3
Outboard twist [deg]: 0

 _____________________
 Available mesh distribution types:
   [1] Linear
   [2] Spanwise half-cosine
   [3] Spanwise half-cosine, chordwise cosine
   [5] Spanwise cosine
   [6] Chordwise cosine
   [7] 3:rd order centerpacking. (Not for wings)
Mesh type: 3 (best recommended by users)

 _____________________
Is partition flapped [1 0]:0
_____________________________________________________

 Wing number:1
 Data entry for partition number: 2
Partition dihedral [deg]: 2
Number of panels semi-span wise: 16
Span of partition: 0.64
Taper ratio: 0.8966

_____________________________
Tip chord airfoil: 0016
Quarter chord line sweep [deg]: 3
Outboard twist [deg]: 0

 _____________________
Mesh type: 3
```

```
 _____________________________
Is partition flapped [1 0]:1
Flap chord in fraction of local chord (0..1): 0.1481
Number of chord wise panels on flap: 20
Do control surfaces deflect symmetrically [1 0]:0
_____________________________________________________

 Wing number:1
 Data entry for partition number:3
Partition dihedral [deg]: 2
Number of panels semi-span wise: 12
Span of partition: 0.07
Taper ratio: 1

_____________________________
Tip chord airfoil: 0016
Quarter chord line sweep [deg]: 3
Outboard twist [deg]: 0

 _____________________________
Mesh type: 3

 _____________________________
Is partition flapped [1 0]:0
_____________________________________________________
 Data regarding wing number :2 (elevator)
_____________________________________________________
Number of semispanwise partitions for this wing: 1
Is the wing mirrored in the xz-plane [1 0]: 1

Apex x-coordinate: 0.78
Apex y-coordinate: 0
Apex z-coordinate: 0.06
Is this wing an all-moving control surface [1 0]: 0
 _____________________________________________________

 Wing number:2
 Data entry for partition number:1
Root chord: 0.2

_____________________________
Base chord airfoil: 0
Number of panels chord wise: 10
Base chord twist [deg]: 0
Partition dihedral [deg]: 0
Number of panels semi-span wise: 20
Span of partition: 0.325
Taper ratio: 0.6

_____________________________
Tip chord airfoil: 0
Quarter chord line sweep [deg]: 13
Outboard twist [deg]: 0

 _____________________________
Mesh type: 3

 _____________________________
Is partition flapped [1 0]:1
Flap chord in fraction of local chord (0..1): 0.25
Number of chord wise panels on flap: 5
Do control surfaces deflect symmetrically [1 0]:1

_____________________________________________________
  Data regarding wing number :3 (rudder)
```

```
_____________________________________________________
Number of semispanwise partitions for this wing: 1
Is the wing mirrored in the xz-plane [1 0]: 0
Apex x-coordinate: 0.78
Apex y-coordinate: 0
Apex z-coordinate: 0.06
Is this wing an all-moving control surface [1 0]: 0
 _____________________________________________________

 Wing number:3
 Data entry for partition number:1
Root chord: 0.235

_____________________________
Base chord airfoil: 0
Number of panels chord wise: 12
Base chord twist [deg]: 0
Partition dihedral [deg]: 90
Number of panels semi-span wise: 12
Span of partition: 0.195
Taper ratio: 0.266

_____________________________
Tip chord airfoil: 0
Quarter chord line sweep [deg]: 40
Outboard twist [deg]: 0

 _________________________
Mesh type: 3

 _________________________
Is partition flapped [1 0]:1
Flap chord in fraction of local chord (0..1): 0.2766
Number of chord wise panels on flap: 6
_____________________________________________
```

Type 4 to save the geometry. You can also type 3 to edit any part in the geometry as in Figure A.2.3.

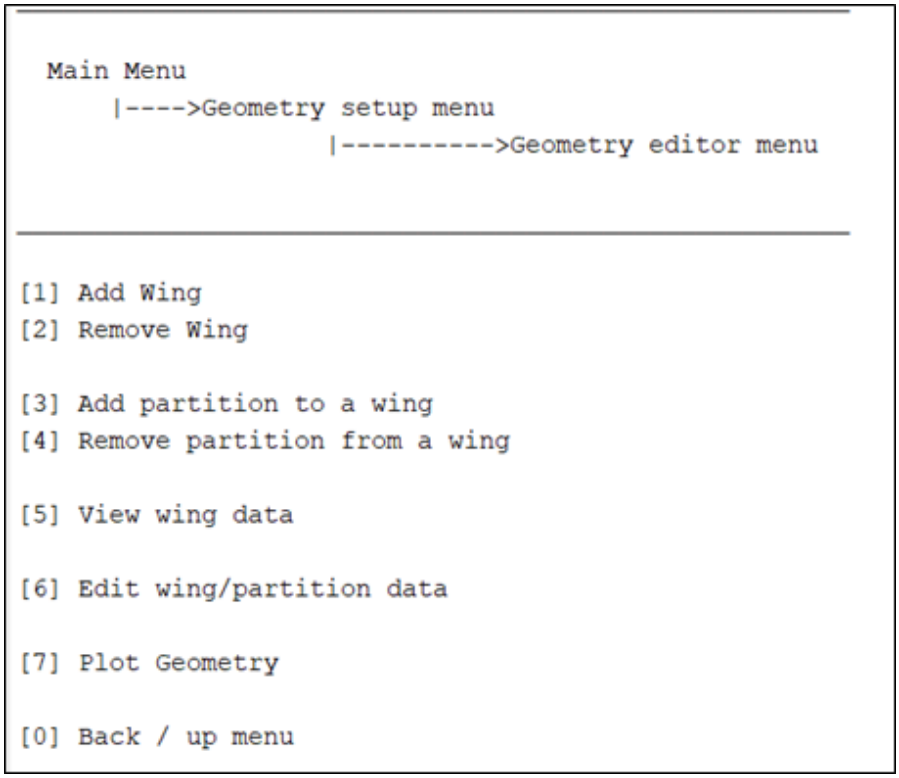


Figure A.2.3.. Geometry Editer Menu

A.2.2 Flight condition setup

By typing 2 in the main menu, the state setup submenu is opened as in Figure A.2.4.

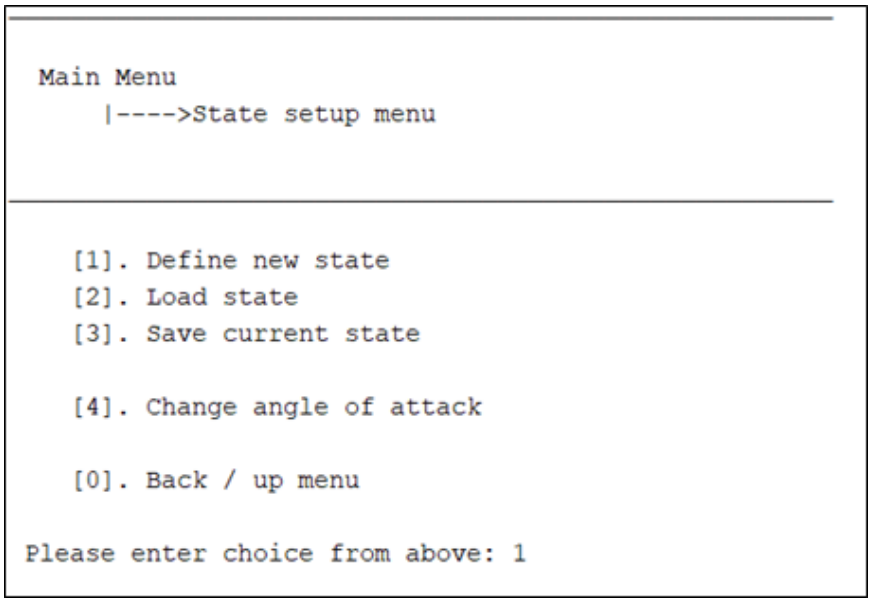


Figure A.2.4 State setup submenu

Type 1 to define a new state, or type 2 to load a previously saved state. In defining a new state, the user is prompted:

```
Alpha [deg]: 5.1 (The program asks for the angle of attack at which it will
process. To calculate the aerodynamic coefficients CLo, CDo, CMo, … etc., the
angle of attack should be "zero". However, it is more practical to process at
the trim conditions. Then the trim angle of attack is used, and as calculated,
is "5.1o".)
Beta [deg]: 0
Roll angular velocity [deg/s]: 0
Pitch angular velocity [deg/s]: 0
Yaw angular velocity [deg/s]: 0
(To fly at steady state cruise, all other values must be "0")
```

Then the speed submenu opens as in Figure A.2.5.

```
Enter which type of speed you whish to enter:

 International units:

 [1]. True airspeed (TAS) at SSL              [m/s]
 [2]. True airspeed (TAS) at altitude         [m/s, m]
 [3]. Equivalent airspeed (EAS) at altitude   [m, m/s]
 [4]. Calibrated air speed (CAS) at altitude  [m, m/s]
 [5]. Mach number at altitude                 [-, m]

 Imperial Units:

 [6]. True airspeed (TAS) at altitude         [kts, ft]
 [7]. Equivalent airspeed (EAS) at altitude   [kts, ft]
 [8]. Calibrated air speed (CAS) at altitude  [kts, ft]
 [9]. Mach number at altitude                 [-, ft]


        If unsure, select option number one.
```

Figure A.2.5. Speed submenu

Type 2 to enter the true airspeed and the altitude.

```
True airspeed [m/s]: 18
```

```
Altitude [m]: 120
Apply Prandtl-Glauert Correction [0 1]: 0 (Because of the low Mach number, it
is acceptable not to use this correction)
```

Finally, save the current flight state.

A.2.3 Change rudder setting

The trimming conditions are trim angle of attack and the trim elevator deflection angle δe *trim*. To account for this constraint, the rudder setting is altered. Tornado names all control surfaces by "rudder". The specify the elevator trim angle, we should choose the rudder on the second wing (the horizontal tail) and set the value to "−0.6319o, as calculated previously.

In the same manner, the first rudder is the aileron, and the third rudder is the rudder. By typing 4, the user is prompted:

```
      The current geometry has 3 trailing edge control surfaces
      and 0 all moving control surfaces
 Available operations:
 [1]. Change rudder/Trailing edge effector setting
 [2]. Change all moving surface setting
 [3]. Cancel / Upmenu.
option from above, please: 1
Change rudder number: [1..]: 2 (elevator)
New absolute control deflection [deg]: -0.6319
```

A.2.4 Move reference point

This option is used to change the point at which the aerodynamic moments are calculated.

A.3 Lattice operations.

A.3.1 Generate lattice

This lattice must be generated before moving to the process. access option to generate the panels specified previously over the aircraft. Because Tornado users recommend the "Freestream following wake, Tornado method", it will be used.

A.4 Computation operations.

A.4.1 Processor access

By typing 6 in the main menu, the processor access submenu is opened as in Figure A.4.1.

```
 Main Menu
     |---->Main processor menu


Low order solutions:
   [1]. Static computation at selected state.
   [2]. Sequential state parameter sweep menu:

High order methods:
   [3]. Trimmed aircraft polar point.
   [4]. Trimmed pitch sweep, polar.
   [5]. [Unused]
   [6]. [Unused]

Auxillary operations:
   [7]. Viscous Drag Estimation Methods Menu:
   [8]. Grid convergence study.
   [9]. Find stall angle of attack.
   [10]. Find alpha at prescribed CL.
   [11]. Compute static margin.

Aeroelastic operations
   [13]. Aeroelastic submenu.
```

Figure A.4.1. Processor submenu

A.4.1.1 Static computation at selected state

By typing 1, the static computation has started. After finishing the solution, the program goes back to the main menu. To view the solution, type 7 (post-processing submenu).

A.4.1.2 Sequential state parameter sweep menu

This option plots the change in the aerodynamic coefficients with respect to α, β, δa, δe, δr, p, q, *and* r. The job tags will be used later to access the results.

```
Available sequenses:
  [1]. Alpha sweep
  [2]. Beta sweep
  [3]. Delta sweep (control surface deflection.)
  [4]. Roll rate sweep
  [5]. Pitch rate sweep
  [6]. Yaw rate sweep

   [0]. Cancel / up menu.
       Sweep parameter: 3
```

A.4.2. Sequential state parameter sweep menu

After choosing a parameter, the user is prompted

```
_____________________________________________
      The current geometry has 3 trailing edge control surfaces
      and 0 all moving control surfaces
Sweeping either:
[1] Trailing edge control effector
[2] Allmoving wing effector

Selection [1..]: 1
Sweep TE effector number: [1..]: 1
    From delta [deg]: -5
    Increment [deg]: 0.5 (for a smooth response)
    To  delta [deg]: 10
_____________________________________________
```

The solution of the “Alpha Sweep” and “Delta Elevator Sweep” should give the same trim conditions. This procedure is repeated for β, δa, δe, δr, p, q, *and* r.

A.4.1.3 Find stall angle of attack

By typing 9, Tornado can find the stall angle of attack. The program displays the result in the command window after finishing the solution as shown in Figure A.4.3.

Given the maximum lift coefficient, *CLmax* , the stall angle of attack is found. For symmetric airfoils, *CLmax* ≈ 0.9.

```
Main wing profile maximum CL: 0.9
Converged

Stall angle of attack [deg]:10.6023


 Solution available in output/Stalla-Cx
```

Figure A.4.3. Result of Stall angle of attack

A.4.1.4 Compute static margin

By typing 11, the location of the static margin and the location of the aerodynamic center of the aircraft is obtained. The static margin results are displayed in MATLAB's command window as in Figure A.4.4.

```
**************************************************
Aerodynamic center located at :0.16953          0        0.02
Static margin                 :-0.034352
**************************************************
```

Figure A.4.4. Result of Static margin

A.5 Post-processing and interactive operations.

A.5.1 Post-processing, Result/Plot functions

By typing 7 in the main menu, the post-processing submenu is opened as in Figure A.5.1.

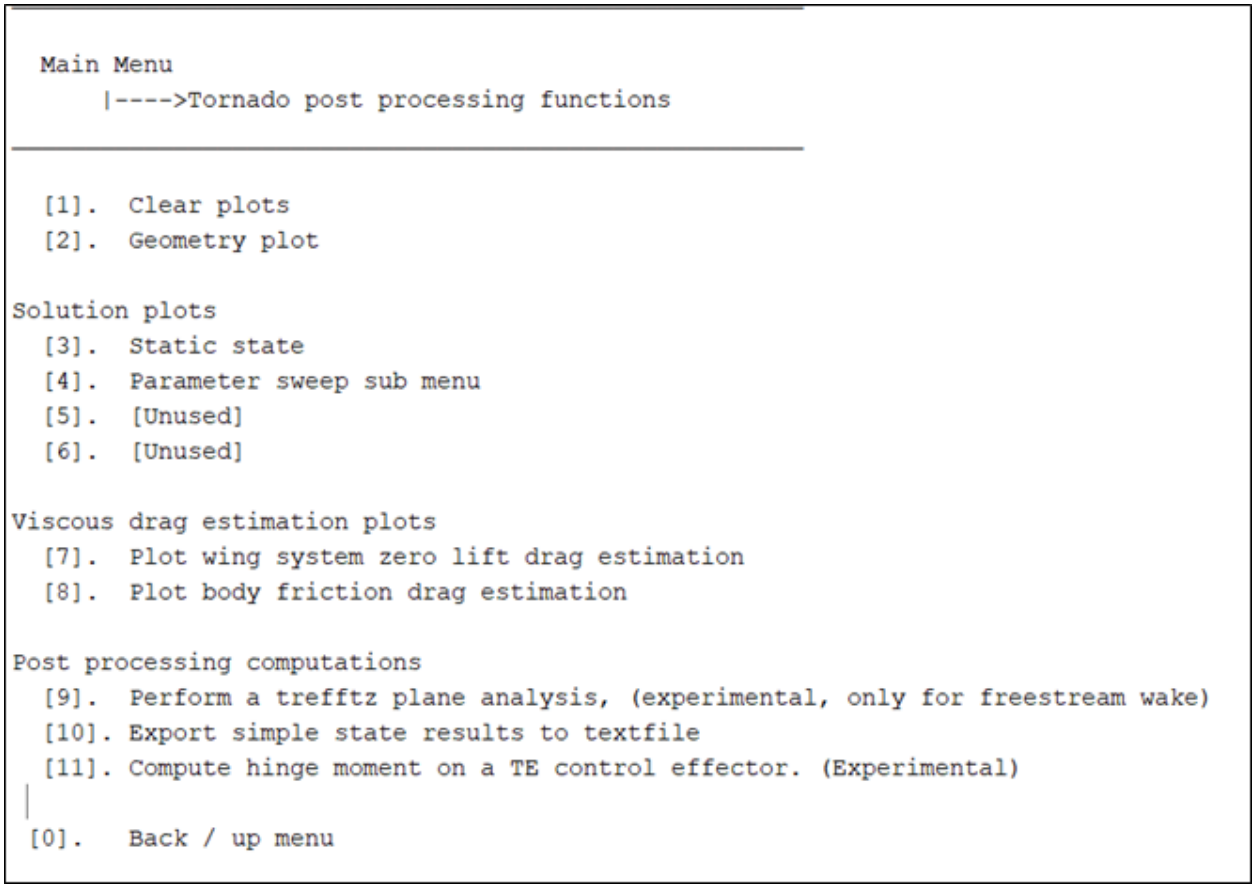


Figure A.5.1. Post processing submenu

A.5.1.1 Clear plots

To prevent the overlap between the new and the old results, use this option to clear any opened figures.

A.5.1.1.1 Geometry plot

This option plots the geometry figures of the loaded aircraft.

A.5.1.1.2 Static state

After choosing this option, the identification tag saved before is required to display the calculated results for this state.

A.5.1.1.3 Parameter sweep submenu

By typing 4, the parameter sweep submenu is opened as in Figure A.5.2.

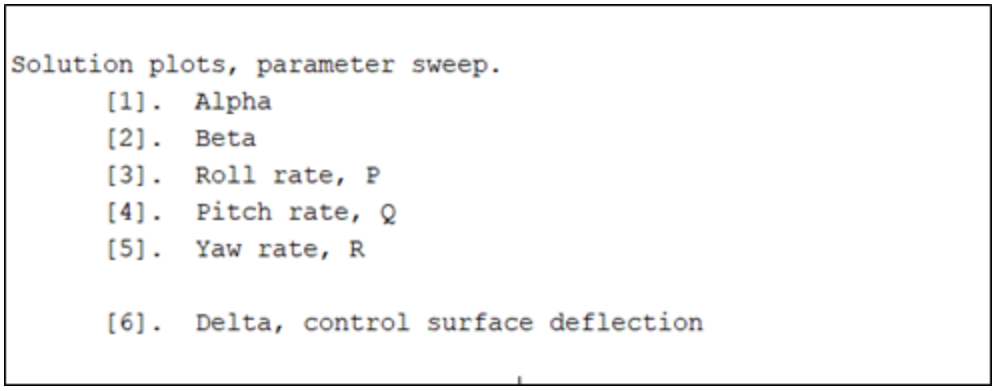


Figure A.5.2. Parameter sweep submenu

Choosing a parameter from this list opens its result.

Appendix B: Matlab Codes

LQR Matlab Code

This code is used in developing the linear model of the aircraft as well as calculating LQR gains. The code runs on Matlab and is subject to change in what state is needed to be controlled.

```
%Inputs to the NON_LINEAR_AicraftModel simulink
clear;clc
%------Gravity------
g=9.81;

%Mass Properties
mass=4;
Ix =0.6465;
Iy =0.3048;
Iz =0.9359;
Ixz=0.0006584;

%-----------Trim Conditions-------------%
Phi0=0;          %in rad
Theta0=5.1*pi/180; %in rad
Epsi0=0;         %in rad
V0=18;           %resultant
u0=sqrt(V0^2/(1+(tan(Theta0))^2));
v0=0;
w0=u0*tan(Theta0);
p0=0;
q0=0;
r0=0;
d_elevator0=-0.6319*pi/180;
d_Thrust0=5;     %0.8749; % 0.8749
d_rudder0=0;
d_aileron0=0;
w_d0=0;
X0=mass*g*sin(Theta0);
Y0=-mass*g*cos(Theta0)*sin(Phi0);
Z0=-mass*g*cos(Theta0)*cos(Phi0);

%-----------Initial Conditions-------------
Phi_initial=Phi0%+(2*pi/180);
Theta_initial=Theta0;%+(2*pi/180);
Epsi_initial=Epsi0+(2*pi/180);
u_initial=u0;
```

```
v_initial=v0;
w_initial=w0;
p_initial=0;
q_initial=0;
r_initial=0;
d_elevatori=d_elevator0;

%initial_position
Xe0=0;
Ye0=0;
Ze0=120; %% inital value, edit this
Hi=120; %%fixed desired value

%Airplane Properties
S=0.471;   %Wing Area
b=1.7;     %Wing Span
c=0.28;    %Mean Wing Chord
%Stability derivatives

%---------------- X force--------------------
CX0=-0.0094304;
CXu=-(0.158*0)-0.0094304;
CX_alpha=(0.42565-0.21544);

CXe=0;
CXt=1;

%---------------- Y force--------------------
CYB=-0.16533;
CYp=-0.00068802; %sign change
CYr=0.15502;    %sign Charge

CY_aileron=0.023693;
CY_rudder=-0.10317;   %sign Change12

%---------------- Z force--------------------
CZ0=-0.42565;
CZu=-((0.052968^2)/(1-(0.052968)^2))*0.42565;
CZ_alpha=-(4.8375+0.0094304);
CZ_alpha_dot=1.3359; %lt/c=2.5
CZq=-7.3583;    %-(-9.96/2.5)*2*u0/c;

CZe=-0.56118;
CZt=0;

%---------------- L Moment--------------------
CLb=0.029196;   %signChange2
CLp=-0.44965;
CLr=0.0067709;

CL_rudder=0.003592;
CL_aileron=0.32935;  %SignChange2
```

```
%---------------- M Moment--------------------
CMu=0.158*0;
CM_alpha=-0.1651;
CM_alpha_dot=-6.5034;
CMq=-11.7782;

CMe=-1.3977;
CMt=0;%0.18;

%---------------- N Moment--------------------
CNB=0.066996;   %sign Change
CNp=-0.030754;
CNr=-0.064937;

CN_rudder=-0.045652;
CN_aileron=0.036215;   %SignChange2

%--------------State space--------------------

Q0 = 196.1738447509;

Xu = Q0*S/(mass*u0)*(2*CX0+CXu);
Xw = Q0*S/(mass*u0)*CX_alpha;
%Xe = Q0*S/(mass)*CXe;
Xe = Q0*S*CXe; % edited no differernce%
%Xt = Q0*S/(mass)*CXt;
Xt = Q0*S*CXt; % edited %

Zu = Q0*S/(mass*u0)*(2*CZ0+CZu);
Zw = Q0*S/(mass*u0)*CZ_alpha;
Zw_d = Q0*S*c/(mass*2*u0^2)*CZ_alpha_dot;
Zq = Q0*S*c/(mass*2*u0)*CZq;
Ze = Q0*S/(mass)*CZe; % done without -%
%Zt = Q0*S/(mass)*CZt;
Zt = Q0*S*CZt; % edited no differernce%

Mu = Q0*S*c/(Iy*u0)*CMu;
Mw = Q0*S*c/(Iy*u0)*CM_alpha;
Mw_d = Q0*S*c^2/(2*Iy*u0^2)*CM_alpha_dot;
Mq = Q0*S*c^2/(2*Iy*u0)*CMq;
Me = Q0*S*c/(Iy)*CMe;
%Mt = Q0*S*c/(Iy)*CMt;
Mt = Q0*S*c*CMt; % edited no difference%

A_long=[ Xu                      Xw              0
-g*cos(Theta0)             0 ;...
     Zu/(1-Zw_d)                      Zw/(1-Zw_d)
(u0+Zq)/(1-Zw_d)           -g*sin(Theta0)/(1-Zw_d)          0 ;...
     Mu+Mw_d*Zu/(1-Zw_d)              Mw+Mw_d*Zw/(1-Zw_d)
Mq+Mw_d*u0/(1-Zw_d)        -g*Mw_d*sin(Theta0)/(1-Zw_d)
     0 ;...
```

```
     0                    0               1                       0
     0 ;...
     -sin(Theta0)    -cos(Theta0)         0
u0*cos(Theta0)+w0*sin(Theta0)   0   ];


B_long=[ Xe          Xt;...
     Ze/(1-Zw_d)           Zt/(1-Zw_d);...
     Me+Mw_d*Ze/(1-Zw_d)        Mt+Mw_d*Ze/(1-Zw_d);...
     0          0;...
     0          0];

longi = [ u_initial; w_initial; q_initial; Theta_initial; Ze0; 0 ;0];
longd=[u0; w0; q0; Theta0; Hi; 0; 0];

 C_long=[1 0 0 0 0;...
     0 0 0 0 1];

 Q_long= 1000*[1000/(u0+0)^2 0 0 0 0; 0 100/(w0+0)^2 0 0 0; 0 0
1/(1000000+q0)^2 0 0 ; 0 0 0 1000/(0+Theta0)^2 0; 0 0 0 0
100/(1+Hi)^2]


 R_long=0.01*[1 0; 0 1];

 K_long=lqr(A_long, B_long,Q_long,R_long);

 %%%state space with refrence tracking:
 A_long_hat=[A_long  zeros(5,2) ;...
          C_long   zeros(2,2) ];

 B_long_hat=[B_long;
          zeros(2,2)];
 C_long_hat=eye(6);
 D_long_hat=zeros(6,2);
 e_longi = longd- [longi; 0];   %[ 0; 0; 0; -2*pi/180; 0; 0];

%%%LQR:
%%% u; w; q; Theta; h; error theta
Q_long_hat= 1*[10 0 0 0 0 0 0; 0 1 0 0 0 0 0; 0 0 0.01 0 0 0 0; 0 0 0
100 0 0 0; 0 0 0 0 1 0 0; 0 0 0 0 0 1000 0;0 0 0 0 0 0 0.5];
K_long_hat=lqr(A_long_hat, B_long_hat,Q_long_hat,R_long);

%%Closed loop system
% reglator_sys = ss((A_long - B_long*K_long), B_long, C_long,
D_long);
 sys = ss((A_long_hat - B_long_hat* K_long_hat), B_long_hat,
C_long_hat, D_long_hat);

%%% simulation
%(1) Simulation without theta disturbance =2 degree with input zero
signal
```

```
 t=0:0.1:10;
u_input=[zeros(1,numel(t));zeros(1,numel(t))];
ff=d_elevatori+zeros(1,numel(t));
kk =d_Thrust0+zeros(1,numel(t));
%u_input=[ff;kk];


[Y,T,e]=lsim(sys,u_input,t,e_longi);
[y,t,x] = initial(sys, x0, t);
[y,t,e] = step(sys, t);
figure(1)
plot(t,Theta0*180/pi-e(:,4)*180/pi,'r',t,Theta0*180/pi+zeros(1,numel(
t)),'b')
title('Pitch angel Response with theta disturbance =2 degree and zero
inputs')
xlabel('time')
legend('theta','theta0')

figure(2)
plot(t,Ze0-e(:,5),'r',t,Ze0+zeros(1,numel(t)),'b')
title('H Response theta disturbance =2 degree and zero inputs')
xlabel('time')
legend('h','h0')

figure(3)
plot(t,u0-e(:,1),'r',t,u0+zeros(1,numel(t)),'b')
title('u Response theta disturbance =2 degree and zero inputs')
xlabel('time')
legend('u','u0')



% %-------------- Lateral LQR------------------

lati = [ v_initial; p_initial; r_initial; Phi_initial; Epsi_initial];
latd=[v0; p0; r0; Phi0; Epsi0; 0];

YB  = Q0*S*CYB/mass;
Yp  = Q0*S*b*CYp/(2*mass*u0);
Yr  = Q0*S*b*CYr/(2*mass*u0);
Yda = Q0*S*CY_aileron/mass;
Ydr = Q0*S*CY_rudder/mass;

NB  = Q0*S*b*CNB/Iz;
Np  = Q0*S*b^2*CNp/(2*Ix*u0);%%%%
Nr  = Q0*S*b^2*CNr/(2*Ix*u0);
Nda = Q0*S*b*CN_aileron/Iz;
Ndr = Q0*S*b*CN_rudder/Iz;

LB  = Q0*S*b*CLb/Ix;
%Lv= Q0*S*b*CLb/(Iz*u0);
Lp  = Q0*S*b^2*CLp/(2*Ix*u0);%%%%
```

```
Lr  = Q0*S*b^2*CLr/(2*Ix*u0);
Lda = Q0*S*b*CL_aileron/Ix;
Ldr = Q0*S*b*CL_rudder/Ix;

dem= 1-Ixz^2/(Ix*Iz);
LB_star= LB/dem;
%Lv_star=Lv/dem;
Lp_star= Lp/dem;
Lr_star= Lr/dem;
Lda_star= Lda/dem;
Ldr_star= Ldr/dem;

NB_star= NB/dem;
Np_star= Np/dem;
Nr_star= Nr/dem;
Nda_star= Nda/dem;
Ndr_star= Ndr/dem;

A21 = LB_star + Ixz/Ix*NB_star;
A22 = Lp_star + Ixz/Ix*Np_star;
A23 = Lr_star + Ixz/Ix*Nr_star;
A31 = NB_star + Ixz/Iz*LB_star;
A32 = Np_star + Ixz/Iz*Lp_star;
A33 = Nr_star + Ixz/Iz*Lr_star;

% states[ beta,p,r,phi,epsi]
A_lat= [YB      Yp    -(u0-Yr)   g*cos(Theta0)   ;...
     A21        A22        A23        0          ;...
     A31        A32        A33        0          ;...
     0          1          0          0          ];

B21=Lda_star+Ixz/Ix*Nda_star;
B22=Ldr_star+Ixz/Ix*Ndr_star;
B31=Nda_star+Ixz/Iz*Lda_star;
B32=Ndr_star+Ixz/Iz*Ldr_star;

% aileron, rudder
B_lat=[0        Ydr;...
     B21   B22;...
     B31   B32;...
     0     0];
 C_lat=[1 0 0 0 ;...
     0 0 0 1 ];
 D_lat=zeros(4,2);

 Q_lat=2*[1 0 0 0;...
     0 1 0 0;...
     0 0 1 0;...
     0 0 0 1];
  R_lat=100*[1 0;...
     0 1];
  K_lat=lqr(A_lat, B_lat,Q_lat,R_lat);
```

```
%%%state space with refrence tracking:
  A_lat_hat=[A_lat  zeros(4,2) ;...
          C_lat zeros(2,2) ];

 B_lat_hat=[B_lat;
          zeros(2,2)];
 C_lat_hat=eye(6);
 D_lat_hat=zeros(6,2);
 e_lati = latd-[lati;0]; %[ 0; 0; 0; -2*pi/180; 0; 0];

%%LQR:
 Q_lat_hat=1*[10  0  0  0   0   0 ;...
          0  10  0  0   0      0   ;...
          0  0  1  0   0 0 ;...
          0  0  0  10  0 0 ;...
          0  0  0  0   1    0 ;...
          0  0  0  0   0   1 ];


 K_lat_reglator=lqr(A_lat, B_lat,Q_lat,R_lat);
 K_lat_hat=lqr(A_lat_hat, B_lat_hat,Q_lat_hat,R_lat);

 %Closed loop system
 reglator_sys = ss((A_lat - B_lat*K_lat), B_lat, C_lat, D_lat);
  sys_lat = ss((A_lat_hat - B_lat_hat* K_lat_hat), B_lat_hat,
C_lat_hat, D_lat_hat);


%%% simulation
%(1) Simulation without theta disturbance =2 degree with input zero
signal
 t_lat=0:0.1:10;
 u_lat_input=[zeros(1,numel(t_lat));zeros(1,numel(t_lat))];
u_lat_input=[d_elevatori
+zeros(1,numel(t_lat));d_Thrust0+zeros(1,numel(t_lat))];

[Y_lat,T_lat,e_lat]=lsim(sys_lat,u_lat_input,t,e_lati);
[y,t,x] = initial(sys, x0, t);
[y,t,e] = step(sys, t);
figure(4)
plot(t,Phi0-e_lat(:,4)*180/pi,'r',t,Phi0*180/pi+zeros(1,numel(t_lat))
,'b')
title('Roll angel Response with phi disturbance =2 degree and zero
inputs')
xlabel('time')
legend('Phi','Phi0')

figure(5)
plot(t,Epsi0-e_lat(:,5),'r',t,Epsi0+zeros(1,numel(t_lat)),'b')
title('Epsi Response with Roll disturbance =2 degree and zero
inputs')
```

```
xlabel('time')
legend('Epsi','Epsi0')



% Plotting responses for sideslip angle
% Run response to initial condition
lati=[v_initial; p_initial; r_initial; Phi_initial; Epsi_initial];
[y,t,x] = initial(sys_lat, lati, 400);
figure;
plot(t,y(:,1));
title('System Response to intial conditions');

% Run response after controller
[y,t,x] = initial(sys_lat_C, lati, 400);
figure;
plot(t,y(:,1));
title('System Response to intial conditions with LQR Controller');
```