FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Design and Development of a Fixed-Wing Tail-Sitter Tethered VTOL Drone with Custom Autopilot and Robust Control Techniques

Ahmad Safaee



Doctoral Program in Electrical and Computer Engineering

Supervisor: Professor António Pedro Aguiar

Co-supervisor: Professor António Paulo Moreira

December 18, 2025

© Ahmad Safaee, 2024

Design and Development of a Fixed-Wing Tail-Sitter Tethered VTOL Drone with Custom Autopilot and Robust Control Techniques

Ahmad Safaee

Doctoral Program in Electrical and Computer Engineering

December 18, 2025

Abstract

This thesis presents the development of a novel autonomous fixed-wing Vertical Take-Off and Landing (VTOL) drone system capable of maintaining a stable flight trajectory while tethered to a ground robot. Unlike previous studies primarily focused on quadcopters, this research leverages the aerodynamic advantages of fixed wings to enhance endurance and efficiency. By functioning as a kite, the aircraft harnesses wind energy, significantly improving its overall performance.

A key contribution of this research is the development of a custom-designed autopilot system, integrating both hardware and software tailored specifically for this application. This system plays a crucial role in controlling the drone, enabling precise navigation and trajectory stabilization. To support this, a full dynamic model of the drone has been derived, and its state-space equations formulated. A sliding mode nonlinear controller has been designed and simulated, demonstrating its effectiveness in managing uncertainties and maintaining stability.

Another contribution of this thesis includes the identification of VTOL drone transfer functions that characterize the dynamics between commands and responses, incorporating lateral error positions feedback from the tether, as well as altitude and pitch channel control.

To address system coupling effects and uncertainties, robust $\mathscr{H}_2/\mathscr{H}_\infty$ and mixed sensitivity controllers, as well as μ synthesis controllers, have been designed for both Single-Input Single-Output (SISO) and Multiple-Input Multiple-Output (MIMO) configurations. These controllers were validated through simulations and experimental flights, demonstrating their effectiveness in handling cross-coupling effects and maintaining stability under various operating conditions. Flight tests confirmed the system's ability to precisely regulate the tether angle while compensating for ground effect disturbances, ensuring accurate navigation, stable attitude, and reliable yaw heading control.

The findings of this study highlight the potential of this innovative system for robust and efficient autonomous navigation. By combining the advantages of VTOL and fixed-wing aircraft, this approach offers a promising solution for a wide range of applications.

Resumo

Esta tese apresenta o desenvolvimento de um sistema inovador de um veículo (drone) autónomo de asa fixa com capacidade de descolagem e aterragem vertical (Vertical TakeOff and Landing - VTOL), capaz de manter uma trajetória de voo estável enquanto está preso por um cabo a um robô terrestre. Ao contrário de estudos anteriores focados principalmente em quadricópteros, esta investigação aproveita as vantagens aerodinâmicas das asas fixas para melhorar a sua autonomia e eficiência. Ao funcionar como um asa-delta (kite), a aeronave aproveita a energia eólica, melhorando significativamente o seu desempenho global. Uma contribuição fundamental é o desenvolvimento de um sistema de piloto automático (autopilot) personalizado, que integra hardware e software especificamente concebidos para esta aplicação. Este sistema desempenha um papel crucial no controlo do drone, permitindo a navegação precisa e a estabilização da trajetória. Para tal, foi desenvolvido um modelo dinâmico completo do drone, com as respetivas equações em espaço de estados. Um controlador não linear por modos i ii deslizantes (sliding mode) foi projetado e simulado, demonstrando a sua eficácia face a incertezas e na manutenção da estabilidade. Outra contribuição desta tese inclui a identificação de funções de transferência do drone VTOL, que caracterizam a dinâmica entre comandos e respostas, incorporando retroação (feedback) de erros laterais da posição do cabo, bem como controlo dos canais de altitude e inclinação (pitch). Para lidar com efeitos de acoplamento do sistema e incertezas, foram projetados controladores robustos $\mathscr{H}_2/\mathscr{H}_\infty$ e de sensibilidade mista, bem como controladores de síntese μ, tanto para configurações de entrada única e saída única (SISO) como de múltiplas entradas e saídas (MIMO). Estes controladores foram validados através de simulações e voos experimentais, demonstrando a sua eficácia face aos efeitos de acoplamento cruzado e na manutenção da estabilidade em diversas condições operacionais. Testes de voo confirmaram a capacidade do sistema para regular com precisão o ângulo do cabo, compensando perturbações do efeito de solo, garantindo uma navegação precisa, atitude estável e controlo fiável da direção (yaw).

Thanks

I would like to express my deepest gratitude to Professor Antonio Pedro Aguiar, my thesis supervisor, whose invaluable guidance and support were essential to the completion of this work. His expertise in nonlinear and robust controllers and dynamic modeling, particularly during the final experimental phase, along with the trust he placed in me, was instrumental to this thesis. I gratefully acknowledge the financial support provided through the scholarship program that made this research possible.

Additionally, I am grateful to Professor Antonio Paulo Moreira, who initially accepted me as a doctoral student in Portugal. His comprehensive support extended beyond academic guidance to include practical assistance with laboratory space and flight testing facilities. His collaboration was crucial in addressing both theoretical challenges and experimental requirements.

Furthermore, I extend my appreciation to Dr. Filipe Neves Santos for his assistance with the preliminary design of the flight system, particularly his contribution to programming the initial configuration of components and sensors. I also extend my sincere appreciation to Professor Hamid Taghirad, my former instructor, for his vital role in developing my understanding of robust control theory and his valuable mentorship in this field.

Ahmad Safaee

"In the name of God, the Beneficent, the Merciful, to whom belongs the kingdom of the heavens and the earth."

Contents

1 Introduction **1**
- 1.1 Tethered UAV Systems . . . 1
- 1.2 Dynamic Modeling of VTOL Tailsitter Drones . . . 4
- 1.3 Controller design under uncertainty: Sliding Mode Control versus $\mathscr{H}_\infty$ robust . . . 5
- 1.4 Contributions . . . 6
- 1.5 Organization . . . 8

2 System Hardware and Software **9**
- 2.1 Introduction . . . 9
- 2.2 Air Vehicle . . . 9
- 2.3 Hardware . . . 11
  - 2.3.1 Attitude and Heading Reference System / GPS . . . 14
  - 2.3.2 LIDAR . . . 14
  - 2.3.3 Servo controller . . . 15
  - 2.3.4 Power distribution . . . 17
- 2.4 Software . . . 17
- 2.5 Concluding remarks . . . 19

3 Dynamic Model Development and Sliding Mode Control Implementation **20**
- 3.1 Drone Specifications and Geometry . . . 21
- 3.2 Dynamic Modeling: Translational and Rotational Equations . . . 22
  - 3.2.1 State-space model . . . 26
- 3.3 Advanced Dynamic Model . . . 29
  - 3.3.1 Rigid-Body Dynamics . . . 29
  - 3.3.2 Basic formula for lift force coefficient ($C_L(\alpha)$) . . . 35
- 3.4 Drone state-space model . . . 39
- 3.5 Sliding mode controller simulation and design . . . 40
  - 3.5.1 Sliding model controller design process . . . 40
- 3.6 Simulation . . . 43
  - 3.6.1 C_L approximation . . . 44
  - 3.6.2 System Analysis and Linearization . . . 45
  - 3.6.3 Calculating the Thrust Coefficient (b) . . . 49
  - 3.6.4 Motor and Propeller Total Moment of Inertia Calculation($\mathbf{J_r}$) . . . 50
- 3.7 Simulation results . . . 51
- 3.8 Concluding remarks . . . 53

4 Robust Control **55**
4.1 Lateral Plan Control Architecture . . . 55
4.2 Testing and System Identification . . . 57
4.2.1 Testing Methodology . . . 57
4.2.2 System Identification . . . 60
4.3 Robust Control Design . . . 63
4.3.1 Uncertainty Profile Modeling . . . 63
4.3.2 Mixed Sensitivity $\mathscr{H}_\infty$ Controller Design . . . 66
4.3.3 Controller Design with $\mathscr{H}_2/\mathscr{H}_\infty$ Method . . . 70
4.3.4 Controller Design with μ synthesis Method . . . 73
4.4 Longitudinal plan control architecture . . . 84
4.4.1 Pitch control loop . . . 84
4.4.2 Altitude control Loop . . . 97
4.5 MIMO controller Design . . . 106
4.5.1 Decentralized MIMO PID controller design optimized using $\mathscr{H}_\infty$ control synthesis . . . 109
4.5.2 MIMO Mixed sensitivity controller design . . . 111
4.5.3 Roll-Yaw coupling MIMO system . . . 114
4.5.4 System Identification . . . 115
4.6 Mixed Sensitivity MIMO Controller Design . . . 116
4.6.1 MIMO control scheme . . . 116
4.6.2 Control Design . . . 116
4.7 Concluding remarks . . . 120

5 Conclusions and Future Work **121**
5.1 Satisfaction of Objectives . . . 121
5.2 Future Works . . . 122

A PID Control and Simulation **123**
A.1 Euler Method . . . 123
A.2 Theoretical Foundation of Ziegler-Nichols Method . . . 123
A.2.1 Basic PID Controller Structure . . . 123
A.2.2 Ziegler-Nichols First Method . . . 124
A.2.3 Ziegler-Nichols Second Method . . . 124
A.2.4 Safety Precautions . . . 124
A.2.5 Step-by-Step Tuning Process . . . 124
A.2.6 Visual Indicators During Tuning . . . 125

B Quaternion-Based Extended Kalman Filter implementation for Orientation Estimation using Accelerometer, Gyroscope, and Magnetometer Data **126**
B.1 Introduction . . . 126
B.2 Hardware . . . 127
B.3 Quaternion Representation of Orientation . . . 128
B.4 The Extended Kalman Filter . . . 129
B.5 Quaternion-based Extended Kalman Filter Algorithm . . . 131
B.6 implementation and details . . . 133
B.6.1 Magnetometer mapping and Heading angle estimation . . . 136
B.6.2 Initialization and tuning consideration . . . 138
B.7 Results . . . 139

B.8 Concluding remarks 142

References **144**

List of Figures

1.1 Localization using tethers for coordinated ground and aerial vehicles (Borgese et al., 2022). . . . 2
1.2 A VTOL fixed-wing drone can function like a kite, spinning a generator to produce electrical energy (Pereira and Sousa, 2023). . . . 3
1.3 VTOL fixed-wing drones are lifting the main air-vehicle (Bass et al., 2019) and (Quenneville et al., 2023). . . . 3

2.1 Wing and servo assembly designed for VTOL's unmanned aerial vehicle system. 10
2.2 Wing, fuselage, and duct designed for VTOL's unmanned aerial vehicle system. . 10
2.3 Implemented drone with embedded avionics. . . . 11
2.4 Hardware diagram components of an autopilot system. . . . 13
2.5 JR radio control transmitter integration for custom drone autopilot system development. . . . 13
2.6 MTi-G AHRS. (Xsens Technologies B.V., 2010). . . . 15
2.7 TF-mini lidar (Benewake (Beijing) Co., Ltd, 2023). . . . 16
2.8 PCA8695 servo controller (Adafruit Industries, 2023). . . . 17

3.1 This diagram illustrates the roll, pitch, and yaw axes on the drone. . . . 21
3.2 Depiction of the rotation around the three axes. . . . 22
3.3 VTOL tail-sitter drone and its avionics. . . . 23
3.4 Pitch angle and elevon deflection forces and torques. . . . 25
3.5 The yaw axis experiences a torque as a result of differential deflections of the elevons. . . . 26
3.6 The drone exhibits rotational motion around the roll axis due to the differential thrust generated by its two motors, resulting in the creation of torque. . . . 26
3.7 Dynamic model simulation in Simulink environment. . . . 28
3.8 Open-loop total simulation. . . . 28
3.9 3D-model designed in the FreeCAD environment, and the moments of the inertial matrix are extracted. . . . 32
3.10 3D-model designed in SOLIDWORKS environment and moments of inertia matrix is extracted. . . . 33
3.11 $\sigma(\alpha)$ versus α . . . 38
3.12 Hierarchical control architecture for VTOL (Vertical Take-Off and Landing) aircraft, comprising three cascaded modules: position control, attitude/altitude control, and thrust/elevon deflection conversion. . . . 44
3.13 Position Control Results: Shows positions (X, Y, Z) and their corresponding velocities (u, v, w). The positions stabilize at the desired 2-meter setpoint, while velocities show proper convergence to steady-state values. . . . 51

3.14 Attitude Control Results: Displays Euler angles (ϕ, θ, ψ) and angular rates (p, q, r), showing convergence to desired orientations with good damping characteristics. 52

4.1 Rolling maneuver controls tether angle position. 56
4.2 Precision industrial potentiometer integrated within avionics for thread angle measurement and positional feedback. 57
4.3 Diagram of commands and corresponding actuators. 57
4.4 Hanging drone ready for initial test for PID controllers tuning. 58
4.5 Initial laboratory efforts for limited flights and controller tuning. 59
4.6 Open area flight test using GPS. 59
4.7 Indoor flight test with tether feedback fully automatic flight. 60
4.8 Bode diagrams showing magnitude and phase responses of ten candidates G_1 transfer functions, from which the final model is selected for system identification. 61
4.9 Bode diagrams showing magnitude and phase responses of ten candidates G_2 transfer functions, from which the final model is selected for system identification. 62
4.10 Bode diagrams showing magnitude and phase responses of ten candidates G_3 transfer functions, from which the final model is selected for system identification. 63
4.11 Multiplicative uncertainty LFT block. 64
4.12 Bode diagram of uncertainty profile for system G_1. 65
4.13 Bode diagram of uncertainty profile for system G_2. 65
4.14 Bode diagram of uncertainty profile for system G_3. 66
4.15 Block diagram of weights function of LFT in mixed sensitivity scenario. 67
4.16 Design parameters of G_1 for $\mathscr{H}_\infty$ in mixed sensitivity scenario. 68
4.17 Design parameters of G_2 for $\mathscr{H}_\infty$ in mixed sensitivity scenario. 69
4.18 Design parameters of G_3 for $\mathscr{H}_\infty$ in mixed sensitivity scenario. 69
4.19 Design parameters of G_1 for $\mathscr{H}_2/\mathscr{H}_\infty$ scenario. 70
4.20 Design parameters of G_2 for $\mathscr{H}_2/\mathscr{H}_\infty$ scenario. 71
4.21 Design parameters of G_3 for $\mathscr{H}_2/\mathscr{H}_\infty$ scenario. 71
4.22 Step response of G_2 for $\mathscr{H}_2/\mathscr{H}_\infty$ controller. 72
4.23 Controller implementation on a real autopilot. 73
4.24 $\Delta - M - K$ Standard Representation. 74
4.25 D-K iteration method algorithm for controller design. 77
4.26 Output of the Matlab software related to the D-K Iteration method for the controller designed for the $G2$ system. 78
4.27 Robust Performance μ plots comparing original and reduced-order controllers for G_2. 78
4.28 Step response comparison of G_2 system between the μ Synthesis controller and its reduced-order version. 79
4.29 Robust stability μ analysis comparing $\mathscr{H}_\infty$, $\mathscr{H}_2/\mathscr{H}_\infty$ and reduced μ controllers. . 79
4.30 Bode diagram comparison of $\mathscr{H}_\infty$, $\mathscr{H}_2/\mathscr{H}_\infty$ and μ synthesis controllers. 80
4.31 D-K Iteration summary for G_3 system showing convergence over 5 iterations. . . 81
4.32 Robust Performance μ plots comparing original and reduced-order controllers for G_3. 82
4.33 Step response comparison between original and reduced-order controllers for G_3 system. 83

4.34 Bode diagram comparing frequency responses of original and reduced-order controllers. 83
4.35 Bode diagram of an identified system for pitch loop. 85
4.36 pole-zero plot of nominal model pitch loop identified system. 85
4.37 Bode diagram of uncertainty profile for pitch loop identified system. 86
4.38 Bode diagram of pitch-rate loop identified systems. 87
4.39 Bode diagram of uncertainty profile for pitch-rate loop identified system. 88
4.40 Design parameters of P_0 for $\mathscr{H}_\infty$ in mixed sensitivity scenario. 89
4.41 Design parameters of P_0 (pitch rate control loop) in μ synthesis scenario. 90
4.42 Step response in μ synthesis scenario 91
4.43 DK-iteration results showing the convergence of controller synthesis over five iterations. 91
4.44 Frequency response of the controller designed in the $\mathscr{H}_\infty$ scenario. 92
4.45 Frequency Response of the Controller Designed in the μ-synthesis Scenario. . . . 93
4.46 Nested control loop structure showing the hierarchical arrangement of the pitch controller system. 94
4.47 Design parameters of the pitch controller loop in the $\mathscr{H}_\infty$ scenario. 95
4.48 Step response of pitch control loop for $\mathscr{H}_\infty$ in mixed sensitivity scenario. 96
4.49 Frequency response of the controller designed in the $\mathscr{H}_\infty$ scenario. 97
4.50 Bode diagram of Vertical Acceleration control loop identified systems 99
4.51 Bode diagram of uncertainty profile for Vertical Acceleration control loop system 100
4.52 Design parameters of Vertical Acceleration control loop in the $\mathscr{H}_\infty$ scenario . . . 101
4.53 Step response of Vertical Acceleration closed-loop system 102
4.54 Frequency response of the controller designed in the $\mathscr{H}_\infty$ scenario. 103
4.55 Nested control loop structure showing the hierarchical arrangement of the altitude controller system. 104
4.56 Design parameters of altitude closed-loop for $\mathscr{H}_\infty$ in mixed sensitivity scenario . 104
4.57 Step response of altitude closed-loop system in the $\mathscr{H}_\infty$ scenario. 105
4.58 Frequency response of the altitude loop controller designed in the $\mathscr{H}_\infty$ scenario. . 106
4.59 Bode diagram of Roll-command to A_z(vertical acceleration) control loop identified systems. 107
4.60 pole-zero plot of Roll-Command To A_z(vertical acceleration) identified system. . 108
4.61 pole-zero plot of throttle-command to roll identified system. 108
4.62 Design parameters of MIMO closed-loop for $\mathscr{H}_\infty$ in PID scenario. 110
4.63 Step response of MIMO closed-loop system in PID scenario 111
4.64 Design parameters of MIMO closed-loop for $\mathscr{H}_\infty$ in mixed sensitivity scenario. . 112
4.65 Step response of the controller designed in the $\mathscr{H}_\infty$ scenario for MIMO plant . . . 112
4.66 Min and max singular value comparison for controller and its reduced version. . . 113
4.67 Frequency response of the controller designed in the $\mathscr{H}_\infty$ scenario for MIMO plant. 114
4.68 The MIMO control scheme with a controller on each channel. 116
4.69 Design parameters of systems for $\mathscr{H}_\infty$ in mixed sensitivity scenario. 118
4.71 The step responses of the opened-loop and the closed-loop system. 118
4.70 The bode diagram for MIMO designed controller $k(s) = \begin{bmatrix} k_{11}(s) & k_{12}(s)k_{21}(s) & k_{22}(s) \end{bmatrix}$. 119

B.1 The schematic of hardware assembly and wiring. 127
B.2 Boards and wiring. 128
B.3 Roll angle estimates using the proposed EKF algorithm and BNO055 Roll and corresponding acceleration and angular rate. 140

B.4 Pitch angle estimates using the proposed EKF algorithm and BNO055 Pitch and corresponding acceleration and angular rate. 141
B.5 yaw angle estimates using the proposed EKF algorithm and BNO055 yaw and corresponding acceleration and angular rate. 142

List of Tables

2.1 Drone specification 11

3.1 General specifications of the drone. 22
3.2 Additional drone dimensions and parameters. 24

4.1 Categories of Systems for Identification 60
4.2 Categories of Systems for Identification 115

Abbreviations and Symbols

UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-off and Landing
IMU	Inertial Measurement Unit
AHRS	Attitude and Heading Reference System
INS	Inertial Navigation System
MIMO	Multi-Input Multi-Output system
SISO	Single-Input Single-Output system
PID	Proportional-Integral-Derivative
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
ROS	Robot Operating System
LIDAR	Light Detection and Ranging
SVD	Singular Value Decomposition
SMC	Sliding Mode Control
MEMS	Micro-Electro-Mechanical Systems

Chapter 1

Introduction

This study builds on over 15 years of the author's experience in aviation and the development of autopilot systems for unmanned aircraft and helicopters. This includes the implementation of both linear PID controllers and nonlinear controllers commonly used in aerospace systems, as well as the design of robust controllers for helicopters. In previous projects, the author led a team of 5–6 engineers from diverse fields, including aerospace, control, and computer programming. However, this study presented a unique opportunity to independently execute tasks that were previously distributed across teams. This individual approach allowed for multiple iterations of testing and tuning, providing deeper technical insights and enabling the integration of previous developments into a more advanced and refined system.

The remainder of this chapter is structured as follows: Section 1.1 reviews related work in the field of tethered UAV systems, exploring their applications and challenges. Section 1.2 discusses the development of the dynamic model of VTOL Tailsitter Drones, and Section 1.3 discusses control design strategies under uncertainty, focusing on Sliding Mode Control and $\mathscr{H}_\infty$ robust control, as well as a MIMO $\mathscr{H}_\infty$ robust control approach. Section 1.3 outlines the key contributions of this thesis, covering system design and implementation, control system design, and experimental validation. Finally, Section 1.4 presents the overall organization of the thesis, detailing the structure of subsequent chapters and their respective focus areas.

1.1 Tethered UAV Systems

Nowadays, a novel concept that has captured the interest of researchers in the field of drones is the utilization of tethers. Tethers are employed for various purposes, including transferring electrical energy to extend flight duration (Zikou et al., 2015), transmitting information to achieve high bandwidth, or even conveying liquids and gases for cleaning tall structures (Marques et al., 2023), power line maintenance, or as internal firefighting tools (Viegas et al., 2022). Another recent proposal, suggests pairing a ground robot with an aerial drone, which also constitutes a primary concept within this work application (Kosarnovsky and Arogeti, 2019). This collaboration enables the aerial drone to assist the ground robot in navigation or routing, utilizing hybrid sensor



Figure 1.1: Localization using tethers for coordinated ground and aerial vehicles (Borgese et al., 2022).

combinations. As mentioned, a ground robot connected to an aerial drone via a tether can supply the necessary energy to the aerial drone. This is crucial because aerial drones, often powered by electricity, typically have a flight duration of around 30 minutes with a lightweight payload such as a small camera.

In (Borgese et al., 2022), a tether-based localization for cooperative ground and aerial vehicles is discussed where potentiometers are used to read the angle of the tether, both within the drone and the ground robot which is integral to this work. This navigation model has garnered significant attention as it can address a fundamental issue with ground robots, namely the weakening or loss of GPS signals due to the unevenness of the earth. A GPS receiver placed on the aerial drone can transmit its coordinates to the ground robot. Since the aerial drone has relative coordinates compared to the ground robot, the absolute coordinates of the ground robot are obtained, and the drone acts like a GPS antenna in altitude for the ground robot as revealed in Figure 1.1. However, a challenge with using quadcopters or multi-rotors is that they are not the most efficient vertical flight models, as they must generate all the necessary lift to counteract their weight. Unlike airplanes with wings that can generate additional lift, quadcopters lack this capability, leading to energy inefficiencies. Models with VTOL capabilities, featuring wings, can overcome this challenge, making them more optimal than multi-rotors (Garg, 2022). Additionally, in multi-rotors, the lateral wind angle that must be corrected results in more energy loss. In contrast, VTOLs with wings can leverage the wind's force on the exposed wings at the correct angle, generating extra lift and extending the drone's flight time. However, a hovering flat plate (the airframe of the VTOL) is very susceptible to disturbance. This susceptibility underscores the importance of designing a

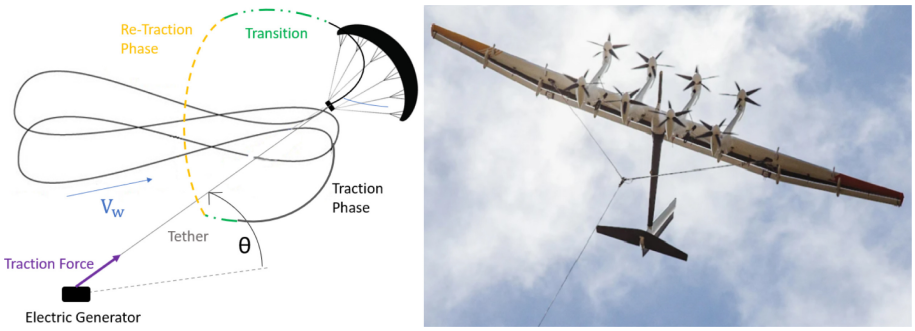


Figure 1.2: A VTOL fixed-wing drone can function like a kite, spinning a generator to produce electrical energy (Pereira and Sousa, 2023).

controller resistant to external disturbances such as wind.

One idea that illuminated the research path is the use of a fixed-wing VTOL tethered to the ground to generate electrical energy (Calciolari, 2020) as shown in Figure 1.2.

This involves maneuvering in an eight-figure pattern and rotating the tether connected to a power generator to produce electricity. In terms of connection, a fixed-wing VTOL tethered to the ground assumes a tail-sitter configuration, aligning closely with this work. To implement this idea, a kite has been used, a topic extensively discussed in various articles (Wang et al., 2015), covering the dynamics of tethered kites, as well as their guidance, control, and autonomous navigation (Erhard and Strauch, 2015). Additionally, the studies in (Bass et al., 2019) and (Quenneville et al., 2023) explore the use of multiple fixed-wing VTOL drones operating in a swarm. These drones are tethered to a primary vehicle and execute a loitering maneuver to generate lift for the main vehicle as shown in Figure 1.3.

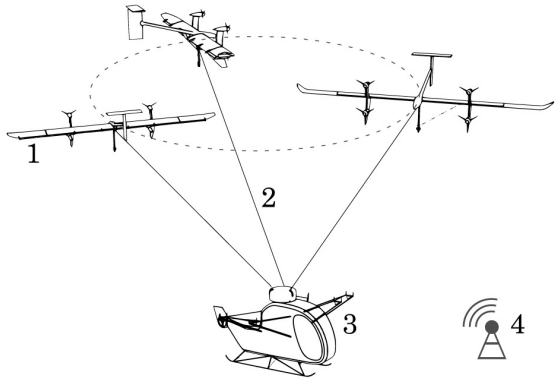


Figure 1.3: VTOL fixed-wing drones are lifting the main air-vehicle (Bass et al., 2019) and (Quenneville et al., 2023).

Utilizing tethers in drones presents distinct challenges, such as navigation alterations due to tether dynamics, uncertainties in dynamic equations due to the drone's dynamics caused by the tension and weight of the tether, and discussions on electrical energy transfer and related optimizations, all of which have been subjects of numerous ongoing papers. However, in this project,

the idea of a kite, harnessing wind energy, with the concept of a quadcopter operating with a tether are combined, to be able to fly under both windy and calm conditions. Ultimately, a fixed-wing VTOL drone with two engines and two control surfaces, named Elevon, is designed and constructed. It is equipped with a custom autopilot, towed by a tether, and its lateral position is controlled by tether angle feedback.

1.2 Dynamic Modeling of VTOL Tailsitter Drones

The development and analysis of dynamic models play a crucial role in the design, control, and optimization of unmanned aerial vehicles (UAVs) (Zhang et al., 2021). Among the various types of UAVs, vertical takeoff and landing (VTOL) tailsitter drones have gained significant attention due to their unique capabilities and versatile flight characteristics (Wang et al., 2023a). Understanding the dynamic behavior of VTOL tailsitter drones is essential for tasks such as stability analysis, controller design, trajectory planning, and simulation studies (Lee et al., 2020).

To accurately capture the complex behavior of VTOL tailsitter drones, it is necessary to extract a dynamic model that accounts for various factors influencing their motion (Zhou et al., 2022). Initially, a simplified nonlinear model is extracted, which provides a fundamental understanding of the drone's dynamics. This initial model considers essential dynamics, such as the relationships between control inputs and resulting motions (Park et al., 2021).

However, to enhance the accuracy of the dynamic model, additional factors that influence the drone's behavior need to be considered. One such factor is the aerodynamic forces acting on the drone during flight (Zhang et al., 2022a). Accurate modeling of these aerodynamic forces, which account for factors such as air density, wing and propeller geometry, and angle of attack, is crucial to accurately represent the drone's response to different flight conditions (Wang et al., 2022).

Furthermore, the gyroscopic forces arising from the rotating propellers significantly impact the dynamics of VTOL tailsitter drones (Liu et al., 2023). These forces, resulting from the drone's spinning propellers, introduce additional complexities in the dynamic model. Accurately accounting for gyroscopic forces is vital for capturing the drone's rotational dynamics and its coupling with translational motion (Chen et al., 2022).

By incorporating these additional factors, such as accurate aerodynamic force models and gyroscopic forces, into the dynamic model, a more comprehensive representation of the VTOL tailsitter drone's behavior can be achieved (Yang et al., 2023). This refined model enables researchers and engineers to analyze and predict the drone's response to various inputs and external disturbances more accurately (Kim et al., 2022).

In this work, we aim to explore the process of deriving a dynamic model for a VTOL tailsitter drone by initially extracting a simplified nonlinear model (Li et al., 2023b). Subsequently, we will investigate the integration of accurate aerodynamic force models and the effects of gyroscopic forces to enhance the model's fidelity (Zhao et al., 2022). This comprehensive approach allows for a more thorough understanding of the drone's dynamics, facilitating advancements in control strategies, mission planning, and overall system performance.

Through this research, we strive to contribute to the ongoing evolution of VTOL tailsitter drones by providing insights into their dynamic behavior (Xu et al., 2023). By refining and validating the dynamic model, we can enable the development of advanced control algorithms, improved navigation techniques, and enhanced simulation environments (Sun et al., 2023).

1.3 Controller design under uncertainty: Sliding Mode Control versus $\mathscr{H}_\infty$ robust

The intersection of nonlinear control methods, particularly sliding mode control (SMC), and robust control theory is especially evident in aerospace applications where both approaches address system uncertainties and disturbances. The fundamental commonality lies in their ability to maintain stability and performance despite modeling uncertainties and external perturbations. Sliding mode control, through its discontinuous control law, forces the system state to remain on a predetermined sliding surface, providing robustness through its ability to reject matched uncertainties completely, while $\mathscr{H}_\infty$ optimal control minimizes the influence of worst-case disturbances on the system output in the frequency domain through the optimization of the $\mathscr{H}_\infty$ norm of the closed-loop transfer function. In aerospace applications, such as attitude and transition flight control of VTOL unmanned aerial vehicles or robotic manipulator control systems, both approaches handle parameter variations and unmodeled dynamics, though they do so through fundamentally different mechanisms - SMC through its reaching and sliding phases, and $\mathscr{H}_\infty$ through its frequency-domain guarantees on disturbance attenuation (Li et al., 2023a) and (Zhang et al., 2022b).

The fundamental distinction between these control approaches lies in their treatment of uncertainties. Sliding mode control excels at handling matched uncertainties, where the disturbance enters the system through the same channel as the control input, providing theoretical complete rejection of these uncertainties once the sliding surface is reached (Shtessel and Edwards, 2023). This makes it particularly effective for mechanical systems where uncertainties often appear in the same equations as control inputs. However, SMC may be less effective against unmatched uncertainties that enter through different channels than the control input. In contrast, $\mathscr{H}_\infty$ robust control takes a frequency-domain approach, treating uncertainties as bounded energy signals and providing optimal attenuation of their effects on the system output (Zhou et al., 2023). $\mathscr{H}_\infty$ control can handle both structured uncertainties (parametric variations) and unstructured uncertainties (unmodeled dynamics, neglected time delays) through careful weighting function design, but it typically cannot achieve complete rejection of any uncertainty type. This trade-off between complete rejection of matched uncertainties in SMC versus optimal attenuation of all uncertainty types in $\mathscr{H}_\infty$ control often leads to hybrid approaches in practical applications (Yang et al., 2022).

A MIMO $\mathscr{H}_\infty$ Robust Control Approach

It is worth noting, however, that fixed-wing VTOL aircraft face significant coupling between roll and yaw dynamics, particularly during hover and transition phases. This coupling arises from

factors such as propeller slipstream effects, gyroscopic forces, and the influence of the center of gravity (Zhang et al., 2023). For example, roll commands achieved through differential thrust can induce yaw moments, while yaw commands implemented via elevon deflection can result in rolling moments. These interactions are further influenced by airspeed, especially during transitions between hovering and forward flight.
To address these challenges, the $\mathscr{H}_\infty$ mixed-sensitivity control framework is employed, offering a robust solution through detailed MIMO (multi-input multi-output) plant modeling and systematic treatment of uncertainties (Chen et al., 2024). This controller utilizes performance, uncertainty, and control effort weights to effectively balance tracking accuracy and robustness. Compared to traditional SISO (single-input single-output) designs, it demonstrates superior performance in maintaining transition stability, suppressing coupling effects, and rejecting disturbances.

1.4 Contributions

The primary contributions of this thesis encompass both theoretical developments and practical implementations in autonomous VTOL systems:

System Design and Implementation

Our first major contribution involves the development and validation of a novel tethered VTOL fixed-wing aircraft platform. Specifically, we:

- Designed and implemented a custom airframe with optimized aerodynamic properties;
- Developed an innovative ground-based tethering system providing positional feedback;
- Integrated a dual-mode capable autopilot system(with GPS and tethered navigation).

Control System Design

We advanced the state-of-the-art in VTOL control systems through:

- Development of comprehensive dynamic models incorporating tether effects;
- Implementation of robust mixed sensitivity $\mathscr{H}_\infty$, $\mathscr{H}_2/\mathscr{H}_\infty$ and μ-synthesis control methodologies controllers for uncertainty management.
  Publication: Ahmad Safaee, Antonio Pedro Aguiar, Filipe Neves Santos and Antonio Paulo Moreira, *Design and Development of a Fixed-Wing Tail-Sitter Tethered VTOL Drone with Custom Autopilot and Robust Control Techniques*, in Proc. of CONTROLO'24 - 16th APCA International Conference on Automatic Control and Soft Computing, Porto, Jul. 2024.
- Successful validation of MIMO control architectures in real-world conditions.
  Publication:Ahmad Safaee, Antonio Pedro Aguiar and Antonio Paulo Moreira"*Designing and Developing a Fixed-Wing Tail-sitter Tethered VTOL UAV with Custom Autopilot: A*

MIMO $\mathscr{H}_\infty$ Robust Control Approach", In Proceedings of the 25th IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC 2025), April 2-3, 2025, University of Madeira, Madeira Island, Portugal.

Experimental Validation

The system's performance was evaluated through:

- Flight testing in both laboratory and outdoor environments;
- Data acquisition and analysis of lateral positional errors;
- Evaluation of control system behavior under different operating conditions;
- Observed improvements in stability and position maintenance.

Technical Achievements

Key technical implementations include:

- Original autopilot design with custom C-based control software;
- Real-time control loop optimization for tethered navigation;
- Efficient integration of GPS-Aided AHRS and Lidar systems with flight computer;
- Parameter selection and calibration methodologies employed for custom autopilot.

Methodological Advances

We built upon existing approaches by introducing:

- New system identification techniques for tethered flight;
- Advanced data acquisition protocols for unmanned systems;
- Improved methodologies for VTOL control system implementation;
- Incorporation of SolidWorks-based parameter optimization.

Research Impact

The outcomes demonstrate significant potential for:

- Enhanced mobility in tethered aerial systems
- Improved stability and control in constrained flight conditions;
- Advanced autonomous operation capabilities;

- Practical applications in surveillance and monitoring scenarios.

These contributions offer a foundation for further development of tethered VTOL systems, with potential applications in both military and commercial sectors. The validated design approach and control methodologies provide a basis for adapting the system to meet various operational needs.

1.5 Organization

The subsequent sections are structured as follows. In Chapter 2, we introduce the designed VTOL drone, detailing the several subsystems, hardware, and software components, including sensors, flight computers, and actuators. The software implementation covers various classes and functions within the flight computer, as well as the drivers for each sensor and actuator in the hardware modules. In Chapter 3, we examine the dynamic model, derive an advanced dynamic model of the drone, and obtain its state-space equations. A nonlinear sliding mode controller is then designed, followed by a complete simulation, with results presented. In Chapter 4, the general structure of the control loops is explained in Section 4.1, and testing stages are presented in Section 4.2, where the nominal model is proposed, and the uncertainty profile is outlined. Sections 4.3-4.7 are dedicated to establishing the limits of the uncertainty profile and designing the control loops using mixed sensitivity $\mathscr{H}_\infty$, $\mathscr{H}_2/\mathscr{H}_\infty$, and μ synthesis methods in both SISO and MIMO scenarios for all control channels. The thesis concludes with Chapter 5, which presents the conclusions and outlines directions for future work.

Chapter 2

System Hardware and Software

This section presents a technical overview of the developed VTOL unmanned aerial vehicle, detailing the design of its structural and electronic systems. The analysis encompasses the vehicle's physical architecture—including wing geometry, fuselage configuration, and actuator mechanics—alongside its avionics infrastructure. The examination focuses on the integrated flight computer system, highlighting the autopilot software with its codebase governing critical control algorithms and operational parameters.

2.1 Introduction

Building the VTOL aircraft from scratch and designing the complete autopilot system provided the author with invaluable experience, leading to the creation of an effective flying laboratory for testing and developing advanced control systems. Many student projects utilize pre-made Pixhawk autopilots (Zheng et al., 2021), and student project papers often involve simple manipulation of the controller. However, designing and developing an autopilot system from the ground up (Cao and Zhang, 2019) teaches a multitude of valuable practical skills. The author gained extensive hands-on experience working with sensors, such as IMUs and inertial measurement units (AHRS), which require significant time for testing, calibration, and integration into control loops. While ready-made autopilots account for these details, designing and implementing a custom autopilot system offers a deeper understanding of the intricacies involved in sensor calibration and control loop integration.

2.2 Air Vehicle

When it comes to building, fixed-wing VTOLs are generally more complicated than multirotors or airplanes. There are several models available for VTOL configurations. For ease of design and implementation, one of the best configurations in terms of the number of motors and servos is selected for this project (Verling et al., 2016) and (Fuhrer et al., 2019). In the beginning, 3D design was performed for the FreeCAd application. It was a problem that it had to have high mechanical

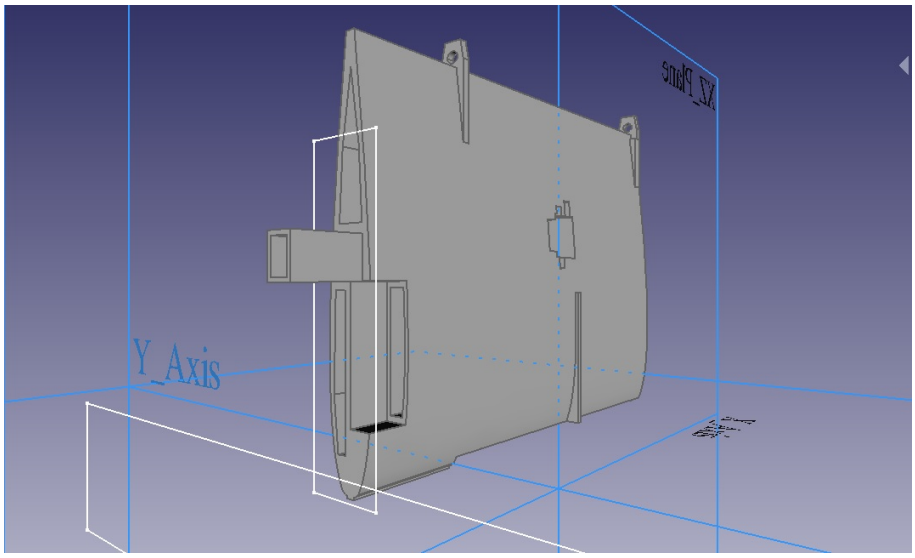


Figure 2.1: Wing and servo assembly designed for VTOL's unmanned aerial vehicle system.

strength while being light. Solving this problem increases the complexity of the design, which consumes a lot of energy and time. Finally, when some of the parts were printed, we realized that there is a good distance with the desired total of weight. You can see the design drawings in FreeCad in the Figure 2.1 and Figure 2.2.

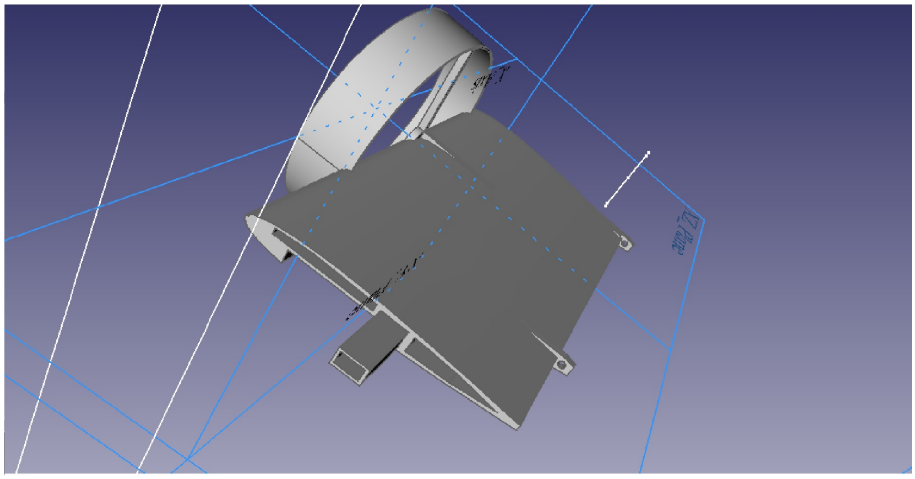


Figure 2.2: Wing, fuselage, and duct designed for VTOL's unmanned aerial vehicle system.

In this section, a laboratory model is developed and constructed utilizing components from a 3D printer and durable, customized cardboard, in place of commercial versions. A designed VTOL drone with specialized avionics and autopilot is shown in Figure 2.3.

The two primary motors of this drone are T-motor Air2213. The following are the motor specifications: The motor, which is equipped with corresponding propellers (9545 propellers) covered by two ducts (13 cm radius), has a 920 KV rating and an idle current of 0.5A at 10V. Additionally, it features two Elevon control surfaces, each of which is linked to an RC servo

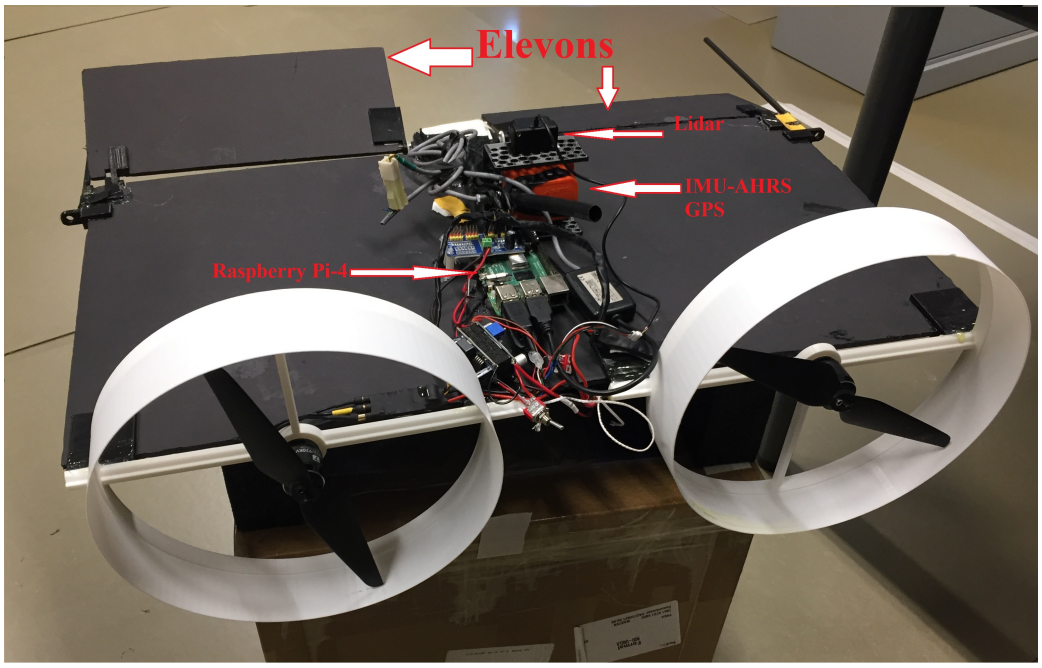


Figure 2.3: Implemented drone with embedded avionics.

actuator (HS-85MG, from Hitec) with a torque of 3.5 kg.cm and a speed of $0.14\,\mathrm{s}/60^{\circ}$. The designed drone's specifications are shown in Table. 2.1.

Table 2.1: Drone specification

Item	Quantity
Wing Span	65.5 cm
Total Weight	1347 gr
Total length	63 cm
Maximum Thrust	1600 gr
Propeller Diameter	26 cm
Main wing cord	35 cm

2.3 Hardware

One of the primary innovations in this project is the decision not to use commercial autopilots, particularly Pixhawk, which is widely used in student projects. In this project, both the autopilot and its hardware have been fully implemented to create a custom autopilot system. Beyond the valuable learning opportunities and experience that building an autopilot provides, this project utilizes an advanced inertial measurement unit (IMU) embedded in GPS-Aided AHRS . This module features gyroscopes with significantly higher stability compared to Pixhawk and is equipped with more precise accelerometers.

As we know, the heart of any autopilot system is the AHRS (Attitude and Heading Reference System). The high quality of this inertial module makes future development paths of the system much smoother for more serious and complex applications. Initially, the BNO055 sensor (Sensortec, 2014) was evaluated for the role of an Attitude and Heading Reference System (AHRS). However, its latency in estimation mode proved insufficient for the responsiveness required by our control algorithms. To address this limitation, a custom sensor fusion algorithm based on a traditional Extended Kalman Filter (EKF) was developed and deployed on an Arduino platform. Although integration with the main flight computer was not completed within the scope of this project, the implementation details are documented in Appendix B for future reference.

Given the need for high reliability and real-time performance, the MTI-G module from Xsens-previously validated in related work, was selected as the primary inertial measurement unit (IMU). This sensor provides accurate orientation angles, angular velocities, linear velocities, and positional data across three axes, as detailed in Section 2.3.1.

The flight computer was locally assembled using a Raspberry Pi 4, chosen for its affordability, commercial availability, and compatibility with the Robot Operating System (ROS) and OpenCV, facilitating future extensions into robotics and image processing. To generate PWM signals for motor and servo actuation, a PCA8596 servo controller module was employed, interfacing with the Raspberry Pi via the I^2C protocol (see Section 2.3.3).

Primary flight commands are transmitted from the ground using a JR radio control system, known for its reliability. These commands—Roll, Pitch, Yaw, and Altitude—are received by a JR receiver onboard the drone. An Arduino Pro Mini reads the PWM signals from the receiver and relays them to the Raspberry Pi via USB.

Altitude control is supported by a Tf-mini LiDAR sensor with a 12-meter range, connected to the Raspberry Pi through a serial interface (Section 2.7). Yaw-angle measurement is supplemented by a potentiometer with an approximate 120-degree range, also interfaced with the Arduino Pro Mini via USB.

A schematic overview of the flight system hardware is presented in Figure 2.4.

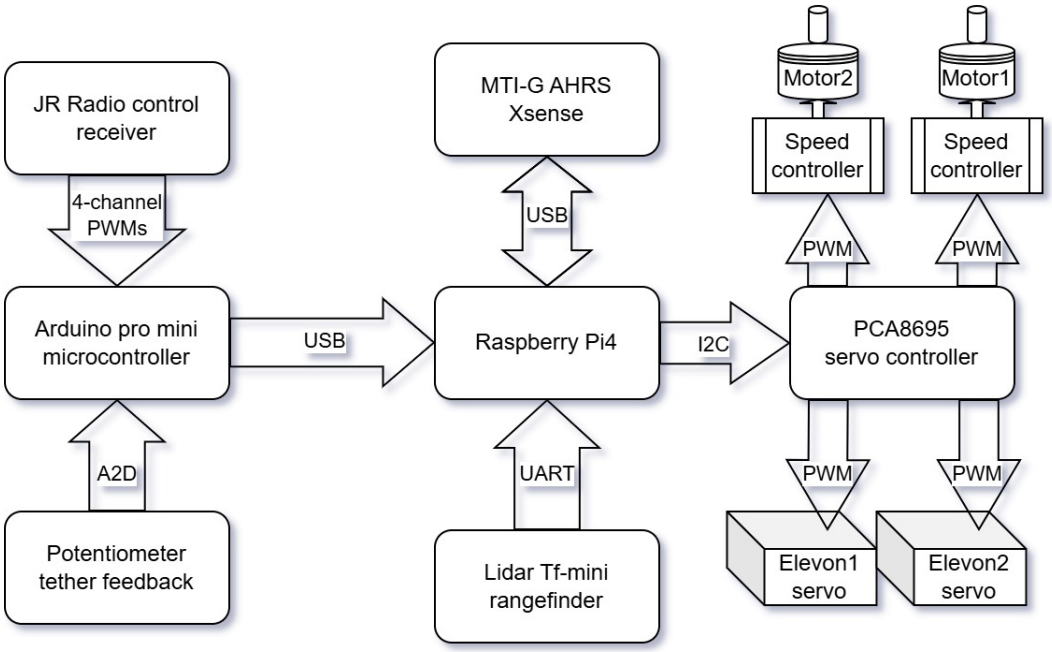


Figure 2.4: Hardware diagram components of an autopilot system.



Figure 2.5: JR radio control transmitter integration for custom drone autopilot system development.

2.3.1 Attitude and Heading Reference System / GPS

The MTI-G by Xsens (as shown in Figure 2.6) is a compact and highly accurate Attitude and Heading Reference System (AHRS) designed for use in dynamic environments. This state-of-the-art sensor system provides precise orientation, velocity, and position data, making it essential for applications in robotics, aerospace, marine navigation, and industrial automation.

The MTI-G AHRS stands out due to its high precision, achieved through advanced sensor fusion algorithms that combine data from gyroscopes, accelerometers, and magnetometers. This combination ensures reliable and accurate motion tracking even in challenging conditions. The device's compact form factor and low weight allow for easy integration into various systems, enhancing its versatility across multiple industries.

One of the key features of the MTI-G is its versatile connectivity. It supports multiple communication interfaces, including USB, RS232, and CAN, facilitating seamless integration with other devices and systems. Additionally, the MTI-G is built to withstand harsh environments, with a wide operating temperature range and resistance to vibration and shocks, ensuring robust performance in diverse applications.

The technical specifications of the MTI-G are impressive: it offers an orientation accuracy of 0.2° RMS for roll and pitch and 1.0° RMS for heading. The gyroscope range is ±450°/s, the accelerometer range is ±10g, and the magnetometer range is ±1.9 Gauss. It can sample data at rates of up to 100 Hz, providing real-time motion tracking. The device measures 58 x 58 x 33 mm and weighs only 47 grams, making it compact and lightweight. It operates effectively within a temperature range of -40°C to +85°C.

The MTI-G is widely used in various fields. In robotics, it enables precise control and navigation of autonomous robots. In aerospace, it is employed in UAVs and drones for stable flight control and navigation. For marine navigation, the MTI-G provides vessel stabilization and navigation in challenging marine environments. In industrial automation, it enhances motion tracking and control in machinery and vehicles.

In conclusion, the MTI-G Xsens AHRS is a highly reliable and accurate device for a wide range of motion tracking applications. Its integration into this project ensures robust orientation estimation, laying a solid foundation for future expansions into more complex and demanding autonomous aerial systems.

2.3.2 LIDAR

The TF Mini LiDAR (as shown in Figure 2.7) is a compact, high-performance distance measurement sensor designed for a variety of applications requiring precise distance sensing. Known for its small size and high accuracy, the TF Mini LiDAR is an ideal solution for projects in robotics, drones, industrial automation, and smart devices.

This LiDAR sensor utilizes time-of-flight (ToF) technology to measure distances accurately and quickly. It emits infrared light and calculates the time it takes for the light to reflect back from an object, providing precise distance measurements up to 12 meters. Despite its small size, the



Figure 2.6: MTi-G AHRS. (Xsens Technologies B.V., 2010).

TF Mini LiDAR delivers excellent performance with a measurement frequency of up to 1000 Hz, ensuring real-time distance sensing.

The TF Mini LiDAR is remarkably compact and lightweight, measuring just 42 x 15 x 16 mm and weighing only 4.7 grams. This small form factor makes it easy to integrate into various systems, where space and weight are critical factors. The sensor is designed for low power consumption, making it suitable for battery-powered applications.

Key features of the TF Mini LiDAR include its high accuracy and resolution. It offers distance measurement accuracy of ±6 cm at distances up to 6 meters and 1% at distances between 6 and 12 meters. The sensor provides reliable data in various lighting conditions and is capable of operating in both indoor and outdoor environments. Additionally, the TF Mini LiDAR has a simple interface and supports UART communication, allowing for easy integration with microcontrollers and other devices.

The TF Mini LiDAR is widely used in multiple fields. In robotics, it enhances obstacle detection and avoidance, contributing to the development of autonomous navigation systems. For drones, it assists in altitude holding and landing procedures, ensuring safe and stable flights. In industrial automation, the TF Mini LiDAR is used for object detection and distance measurement in manufacturing and logistics. It is also employed in smart devices for gesture recognition and presence detection.

In conclusion, the TF Mini LiDAR is a versatile and reliable distance measurement sensor suitable for a wide range of applications. Its compact design, high accuracy, and ease of integration make it a preferred choice for engineers and developers working on innovative projects.

2.3.3 Servo controller

The PCA9685 is a versatile 16-channel, 12-bit PWM (Pulse Width Modulation) driver designed to control LEDs, motors, and other devices requiring precise pulse control.In this project, we use this module to control 2 servos related to the elevons and control the two main bldc motors They are all controlled by PWM. The PCA9685 uses the I²C communication protocol, allowing easy integration with various microcontrollers and other I²C-enabled devices. It supports a supply



Figure 2.7: TF-mini lidar (Benewake (Beijing) Co., Ltd, 2023).

voltage range of 2.3V to 5.5V, making it compatible with both 3.3V and 5V logic levels. This flexibility ensures its use in a wide range of electronic projects and systems.

One of the standout features of the PCA9685 is its ability to drive 16 independent PWM channels with a resolution of 12 bits per channel. This high resolution enables precise control over the duty cycle of each channel, making it ideal for applications requiring fine-grained control of brightness, speed, or position. The PCA9685 can achieve a PWM frequency of up to 1.6 kHz, ensuring smooth and responsive operation of connected devices.

The device also includes an internal clock, eliminating the need for an external clock source and simplifying the design process. Additionally, the PCA9685 has a programmable frequency oscillator, allowing users to adjust the PWM frequency to suit specific application requirements. This feature is handy in applications like LED dimming, where different frequencies can help avoid flicker and other issues.

In terms of power efficiency, the PCA9685 includes an output enable pin that can be used to disable all outputs with a single signal, reducing power consumption when PWM control is not needed. This makes it an energy-efficient choice for battery-powered applications.

The PCA9685 is widely used in various fields. In robotics, it enables precise control of servo motors, allowing for complex movements and functions. For LED lighting, it provides smooth dimming and color mixing capabilities. In-home automation is used to control various devices, such as window blinds and fans, providing a high degree of automation and customization. In industrial settings, the PCA9685 is employed in control systems requiring accurate and reliable PWM signals. It is shown in Figure 2.8.

In conclusion, the PCA9685 PWM driver is a highly flexible and powerful solution for a wide range of applications. Its high resolution, multiple channels, and ease of integration make it a preferred choice for engineers and developers working on projects that require precise and reliable pulse width modulation.



Figure 2.8: PCA8695 servo controller (Adafruit Industries, 2023).

2.3.4 Power distribution

Multiple voltage regulator circuits are implemented to supply the required voltages to different components. Six volts are provided for servos to achieve maximum response speed, five volts for the Raspberry Pi, flight computer, and sensors, and ten volts for the potentiometer to obtain higher resolution. All these power requirements are sourced from a 4-cell lithium battery with a main voltage of 14.8 V 850 mAh, providing a flight time of approximately 3 minutes.

2.4 Software

One of the key challenges in the software domain of automatic guidance systems or autopilots is the real-time capability of the flight computer. Since the Raspberry Pi is not designed for tasks requiring very fast and real-time control like microcontrollers, the management of time and memory is crucial in this project. For autopilot programming, C++ object-oriented programming is employed at a low level, operating on the Rasbian operating system. The main structure of the autopilot program is implemented based on multithreading.

As described in the hardware section, heavy I/O processing is the main flow of the program. Updating control loops at a minimum rate of 50 Hz is required. For quick response and safety margin, a speed of 100 Hz is ideal which is satisfied. For this purpose, I2C and serial port exchanges have set their baud rate to the maximum, and optimal software methodologies have been implemented.

For monitoring information and communicating with the drone, Remote Desktop and Wi-Fi LAN connections have been utilized, taking advantage of the Linux operating system being installed on the Raspberry Pi. This allows the control of the drone from a laptop without using radio control as well. To log data, information is stored in the main system's flash memory at a high speed of approximately 100 Hz. The following code demonstrates the interface for connecting an Arduino Pro Mini to an RC receiver and a potentiometer. It includes both the embedded firmware for the Arduino and the corresponding C script for the Raspberry Pi, which receives and processes serial data transmitted from the Arduino.

Ardino pro mini firmware

```
    int analogPin = A3;////////////A2D pin
//Declaring Variables
byte last_channel_1, last_channel_2, last_channel_3, last_channel_4;
```

```
int receiver_input_channel_1, receiver_input_channel_2, receiver_input_channel_3, receiver_input_channel_4,
val=0;
unsigned long timer_1, timer_2, timer_3, timer_4;

void setup(){

PCICR |= (1 << PCIE0);
PCMSK0 |= (1 << PCINT0);
PCMSK0 |= (1 << PCINT1);
PCMSK0 |= (1 << PCINT2);
PCMSK0 |= (1 << PCINT3);
Serial.begin(1000000);


}

void loop(){

print_signals();
}

ISR(PCINT0_vect){
//Channel 1=========================================
if(last_channel_1 == 0 && PINB & B00000001 ){ //Input 8 changed from 0 to 1
last_channel_1 = 1; //Remember current input state
timer_1 = micros(); //Set timer_1 to micros()
}
else if(last_channel_1 == 1 && !(PINB & B00000001)){ //Input 8 changed from 1 to 0
last_channel_1 = 0; //Remember current input state
receiver_input_channel_1 = micros() - timer_1; //Channel 1 is micros() - timer_1
}
//Channel 2=========================================
if(last_channel_2 == 0 && PINB & B00000010 ){ //Input 9 changed from 0 to 1
last_channel_2 = 1; //Remember current input state
timer_2 = micros(); //Set timer_2 to micros()
}
else if(last_channel_2 == 1 && !(PINB & B00000010)){ //Input 9 changed from 1 to 0
last_channel_2 = 0; //Remember current input state
receiver_input_channel_2 = micros() - timer_2; //Channel 2 is micros() - timer_2
}
//Channel 3=========================================
if(last_channel_3 == 0 && PINB & B00000100 ){ //Input 10 changed from 0 to 1
last_channel_3 = 1; //Remember current input state
timer_3 = micros(); //Set timer_3 to micros()
}
else if(last_channel_3 == 1 && !(PINB & B00000100)){ //Input 10 changed from 1 to 0
last_channel_3 = 0; //Remember current input state
receiver_input_channel_3 = micros() - timer_3; //Channel 3 is micros() - timer_3
}
//Channel 4=========================================
if(last_channel_4 == 0 && PINB & B00001000 ){ //Input 11 changed from 0 to 1
last_channel_4 = 1; //Remember current input state
timer_4 = micros(); //Set timer_4 to micros()
}
else if(last_channel_4 == 1 && !(PINB & B00001000)){ //Input 11 changed from 1 to 0
last_channel_4 = 0; //Remember current input state
receiver_input_channel_4 = micros() - timer_4; //Channel 4 is micros() - timer_4
}


}
//Subroutine for displaying the receiver signals
void print_signals(){
  Serial.print("#PWM");
Serial.print(int(receiver_input_channel_1-710)/2);
    Serial.print(",");


Serial.print(int(receiver_input_channel_2-710)/2);
    Serial.print(",");
Serial.print(int(receiver_input_channel_3-710)/2);
    Serial.print(",");
Serial.print(int(receiver_input_channel_4-710)/2);
 Serial.print(",");
val = analogRead(analogPin);  // read the input pin
  Serial.print(val);          // debug value
    Serial.println("$");
```

```
}
```

Raspberry Pi code for receiving data from Arduino Pro mini

```
servod servoInput(int *input){

   servod u;

   databuffer[databufferP] =serialGetchar(*input);
   if(databuffer[databufferP] == '#') databufferP=0;
   if(databuffer[databufferP] == '$') {
      databuffer[databufferP+1] = '\0';
       sscanf(&databuffer[5],"%d,%d,%d,%d,%d$", &u.out1, &u.out2, &u.out3, &u.out4, &u.out5 );

       //printf("%s!   --- %d --- %d\n",&databuffer[4],u.out1,u.out4 );
       databufferP=0;
       u.ct=1;
       return u;
      }
   databufferP++;
   if(databufferP>200) databufferP=0;

   u.ct=-1;
   return u;
}
```

2.5 Concluding remarks

This phase of the project represented one of the most technically demanding aspect of the thesis, requiring extensive effort in component evaluation, integration, and iterative testing. It involved countless hours of examining individual hardware modules, exploring various configurations, and writing hundreds of lines of C and C++ code to interface and calibrate each sensor. The integration of all autopilot subsystems—sensor fusion, control logic, communication protocols, and actuator interfaces—required meticulous coordination and repeated validation across hardware and software layers. A significant portion of the effort was dedicated to hardware-in-the-loop (HIL) testing, which demanded multiple iterations to achieve reliable and deterministic performance. These procedures were particularly challenging and high-risk when working with critical systems such as the motor control unit. Given that the autopilot system was custom-built, installing its components within the aircraft fuselage introduced additional complexities. These included mitigating electromagnetic interference and ensuring that the placement of components preserved the aircraft's center of gravity, both of which were essential for maintaining flight stability.

Chapter 3

Dynamic Model Development and Sliding Mode Control Implementation

This section focuses on developing a dynamic model through direct analysis of the dynamic equations. Our starting parameters include the drone's physical specifications, such as wing span and chord length, control surface dimensions and characteristics, thrust variations in response to throttle commands, and the distribution of mass across the drone's body and wings. The goal was to determine the moments of inertia parameters and aerodynamic coefficients. The analysis encompassed dynamic equations for fixed-wing aircraft, multi-rotor systems, and VTOL (Vertical Take-Off and Landing) configurations. The combination of extensive laboratory testing and advanced software tools, particularly SolidWorks for modeling and Data Acquisition systems, proved invaluable for this analysis. The reformulation of the dynamic equations culminates in state equations that exhibit distinct nonlinear characteristics .These nonlinear dynamics naturally guide us toward implementing nonlinear control strategies. The derived state equations are particularly well-suited for designing and implementing nonlinear controllers.Building upon these nonlinear control concepts, the Sliding Mode Control (SMC) emerges as a particularly robust strategy for handling system uncertainties and disturbances. This advanced nonlinear control technique establishes a sliding surface where the system's behavior is constrained to follow a desired trajectory, regardless of model imperfections or external perturbations. The distinctive feature of SMC lies in its ability to maintain system performance even when faced with parametric uncertainties, unmodeled dynamics, and bounded external disturbances. By employing discontinuous control signals that switch between different control structures, SMC forces the system state to "slide" along the designed surface, ensuring robust stability. This characteristic makes it especially valuable for aerospace applications where aerodynamic uncertainties, environmental disturbances, and modeling inaccuracies are inevitable. Moreover, SMC's inherent robustness eliminates the need for exact system parameters, making it more practical for real-world implementation compared to traditional control methods that require precise system models. Building on the previous discussion of SMC's robustness, this controller has become a cornerstone in modern aerospace control systems. It has been successfully implemented across a wide range of aerospace applications,

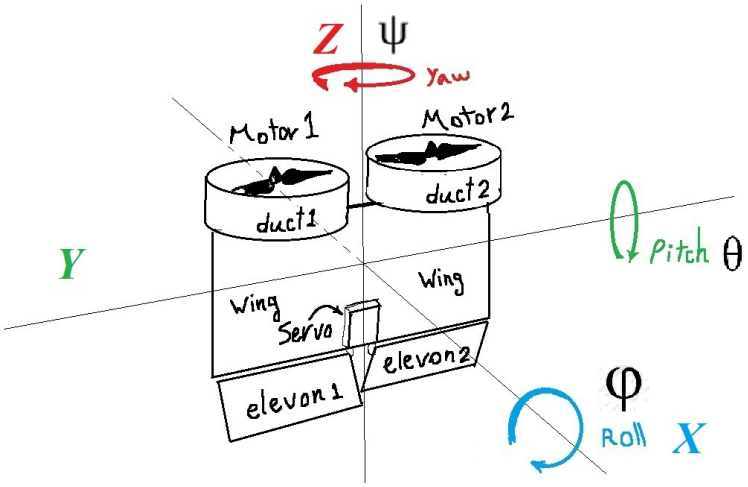


Figure 3.1: This diagram illustrates the roll, pitch, and yaw axes on the drone.

from attitude control of satellites and spacecraft to flight control systems in aircraft and unmanned aerial vehicles (UAVs) (Edwards and Spurgeon, 2021). The controller's exceptional performance in managing the complex, coupled dynamics of aerospace systems is particularly evident in scenarios involving atmospheric reentry, formation flying, and precision maneuvers. Major space agencies, including NASA and ESA, have employed sliding mode control in various missions due to its ability to handle the significant uncertainties inherent in space operations, such as varying atmospheric conditions, fuel slosh dynamics, and structural flexibilities. Its widespread adoption in aerospace systems is further justified by its capacity to maintain stability and performance despite the extreme environmental conditions and dynamic changes that spacecraft and aircraft routinely encounter, from launch through orbital operations to landing procedures. Finally, based on the above, a sliding controller was selected for the system, designed, and implemented in simulation.

3.1 Drone Specifications and Geometry

The general schematic of the VTOL drone as well as the body coordinate system is revealed in Figure 3.1.

This drone is equipped with two primary motors, the T-Motor Air2213, as previously mentioned. The specifications for the motors are as follows: the motor has a KV rating of 920KV, stator dimensions of 22 x 13 mm, a weight of 54 g, and an idle current of 0.5 A at 10 V. The motors are paired with 9545 propellers, which are protected by two ducts, each with a radius of 13 cm. Additionally, the drone is equipped with two movable control surfaces, called Elevons. Each Elevon is connected to an RC servo actuator (HS-85MG by Hitec), with a torque of 3.5 kg/cm, capable of traveling 60° in 0.14 s. The name "Elevon" is derived from the fact that both surfaces simultaneously participate in controlling the pitch and yaw angles.

For climbing, both motors must provide thrust upward simultaneously. The actuator block, which includes 4 inputs (throttle, roll, pitch, and yaw) and 4 outputs (motor 1, motor 2, Elevon

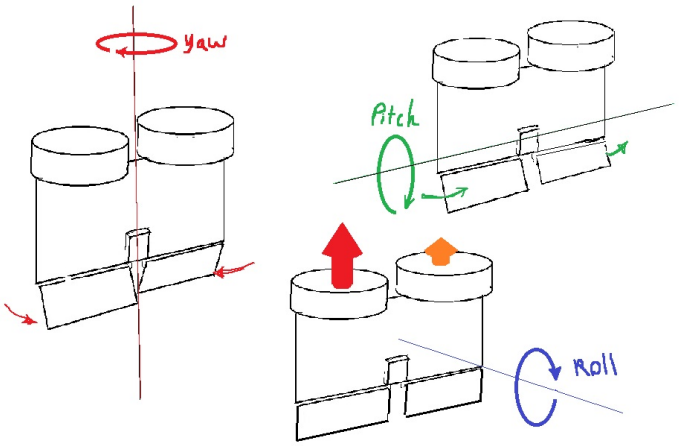


Figure 3.2: Depiction of the rotation around the three axes.

1, and Elevon 2), manages the control of these components. For roll rotation, the motors must provide thrust in opposing directions, as shown in Figure 3.2.

The specifications of the drone are summarized in Table 3.1. The final developed system is shown in Figure 3.3, which provides a comprehensive view of the VTOL tail-sitter drone and its avionics.

Table 3.1: General specifications of the drone.

Item	Quantity
Wing Span	65.5 cm
Total Weight	1347 g
Total Length	63 cm
Maximum Thrust	1600 g
Propeller Diameter	24 cm
Main Wing Chord	35 cm
Elevon Span	27 cm

3.2 Dynamic Modeling: Translational and Rotational Equations

This section presents the dynamic model of the VTOL by employing the Newton-Euler method (see Equations (3.1) and (3.6)). The problem is divided into two distinct parts. The first part focuses on a simplified dynamic model similar to the quadcopter, specifically the roll angle, which is controlled using differential motor commands, similar to a standard quadcopter configuration. The second part addresses the dynamics in the pitch and yaw angles through elevon deflections.

To simplify the dynamic model, the elevons are assumed to function as thrust vector controls that deflect the thrust vector generated by the motors, as illustrated in Figure 3.4. In this model, certain aerodynamic effects, such as those related to the wing and elevons, are disregarded. It is assumed that a portion of the thrust produced by each motor and propeller is deflected based on

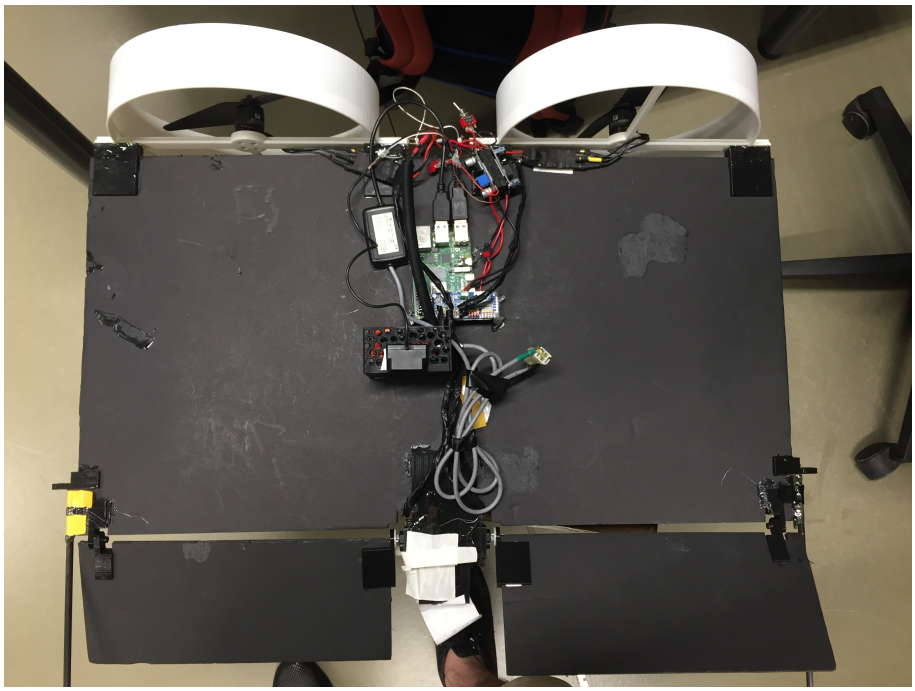


Figure 3.3: VTOL tail-sitter drone and its avionics.

the perpendicular projection of the elevon area. This deflected force is applied at the center of the elevon surface, creating a torque arm. The distance from the center of mass to the elevon surface determines the torque's effect on the pitch angle in the z-axis (Figure 3.4) and the yaw angle in the y-axis of the body frame (Figure 3.5). For the roll angle, the distance from each motor to the center of mass along the y-axis is considered.

The equations of motion for the system are divided into two categories: translational and rotational. These are derived from the fundamental principles of Newton's laws. The translational equations are expressed as:

$$\sum F = ma \tag{3.1}$$

$$(T_1 + T_2)\cos\theta\cos\phi = mg - m\ddot{Z} \tag{3.2}$$

$$(T_1 + T_2)\sin\theta = m\ddot{X} \tag{3.3}$$

$$(T_1 + T_2)\cos\theta\sin\phi = m\ddot{Y} \tag{3.4}$$

$$U = T_1 + T_2 \tag{3.5}$$

In these equations, U denotes the total thrust produced by both motors, where T_1 and T_2 refer to the thrust generated by motor 1 and motor 2, respectively, g is the gravity acceleration, and m

represents the mass of the drone. The translational equations account for the forces that induce linear acceleration along the X, Y, and Z axes. The variables ϕ, θ, and ψ represent the Euler angles (roll, pitch and yaw).

The rotational equations of motion, derived from the torques acting on the system, are given by:

$$\sum \tau = I\alpha \tag{3.6}$$

$$\sum \tau_1 = I_{yy}\ddot{\theta} \tag{3.7}$$

$$\sum \tau_2 = I_{xx}\ddot{\phi} \tag{3.8}$$

$$(T_1 - T_2)d = I_{xx}\ddot{\phi} \tag{3.9}$$

$$(T_1 + T_2)\left(\frac{S_{\text{elevon}}(\sin\delta_{e1} + \sin\delta_{e2})}{S_{\text{eff}}}\right)l = I_{yy}\ddot{\theta} \tag{3.10}$$

$$(T_1 + T_2)\left(\frac{S_{\text{elevon}}}{S_{\text{eff}}}(\sin\delta_{e1} - \sin\delta_{e2})\right)f = I_{zz}\ddot{\psi} \tag{3.11}$$

where δ_{e1} and δ_{e2} represent the deflections of the control surfaces of elevon 1 and elevon 2. The term d is half the distance between the two motors, and the distances f and l denote the separation from the center of mass to the root of the elevon in the Z-axis and the distance from the center of mass to the middle of the elevon span in the Y-axis, respectively. The areas S_{elevon} and S_{eff} are related to the elevon span and the effective propeller thrust area.

The detailed drone dimensions and parameters are listed in Table 3.2.

Table 3.2: Additional drone dimensions and parameters.

Item	Quantity
f	22 cm
d	36 cm
l	20 cm
S_{elevon}	0.0336 m^2
S_{eff}	0.053 m^2
Max $\delta_{e1,2}$	60°

To calculate the moments of inertia, we use the following formulas based on mass and radius:

$$I_{xx} = mr_y^2 + mr_z^2 \tag{3.12}$$

$$I_{yy} = mr_x^2 + mr_z^2 \tag{3.13}$$

$$I_{zz} = mr_x^2 + mr_y^2 \tag{3.14}$$

Here, r_x, r_y, and r_z represent the distances from the axis of rotation to the mass distribution of the

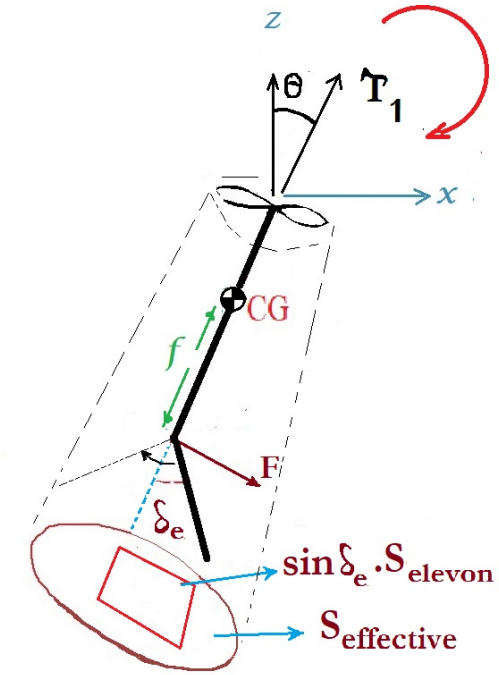


Figure 3.4: Pitch angle and elevon deflection forces and torques.

object along the x, y, and z axes, respectively. The computed moments of inertia are as follows:

$$\begin{aligned} I_x &= 0.00694\,\text{kg}\,\text{m}^2 \\ I_y &= 0.00363\,\text{kg}\,\text{m}^2 \\ I_z &= 0.05757\,\text{kg}\,\text{m}^2 \end{aligned} \tag{3.15}$$

Figures 3.4, 3.5, and 3.6 visually represent the forces and torques associated with the pitch, yaw, and roll angles, respectively.

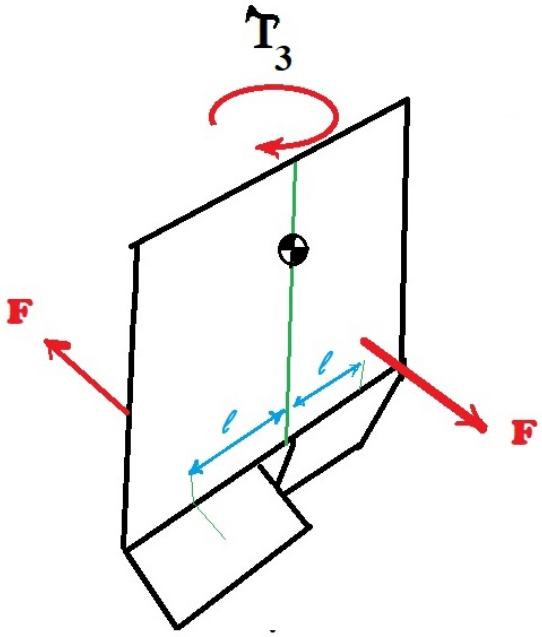


Figure 3.5: The yaw axis experiences a torque as a result of differential deflections of the elevons.

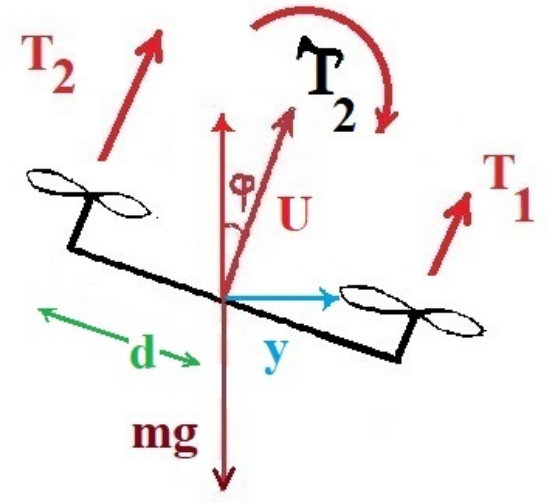


Figure 3.6: The drone exhibits rotational motion around the roll axis due to the differential thrust generated by its two motors, resulting in the creation of torque.

3.2.1 State-space model

The state-space variable has been represented as $\mathbf{x}^\top = [X\ Y\ Z\ \dot{X}\ \dot{Y}\ \dot{Z}\ \theta\ \phi\ \psi\ \dot{\theta}\ \dot{\phi}\ \dot{\psi}]$. Furthermore, we define the inputs u_1, u_2, u_3, and u_4 as following:

$$u_1 = T_1 + T_2, \tag{3.16}$$
$$u_2 = T_1 - T_2, \tag{3.17}$$
$$u_3 = \sin(\delta_{e1}) + \sin(\delta_{e2}), \tag{3.18}$$
$$u_4 = \sin(\delta_{e1}) - \sin(\delta_{e2}). \tag{3.19}$$

we obtain a simpler form, as follows:

$$\dot{x}_1 = x_4 \tag{3.20}$$
$$\dot{x}_2 = x_5 \tag{3.21}$$
$$\dot{x}_3 = x_6 \tag{3.22}$$
$$\dot{x}_4 = \frac{1}{m} u_1 (\cos x_8 \sin x_7) \tag{3.23}$$
$$\dot{x}_5 = \frac{1}{m} u_1 (\sin x_8) \tag{3.24}$$
$$\dot{x}_6 = g - \frac{1}{m} u_1 (\cos x_8 \cos x_7) \tag{3.25}$$
$$\dot{x}_7 = x_{10} \tag{3.26}$$
$$\dot{x}_8 = x_{11} \tag{3.27}$$
$$\dot{x}_9 = x_{12} \tag{3.28}$$
$$\dot{x}_{10} = \frac{1}{I_{yy}} u_1 \left(\frac{S_{\text{elevon}}}{S_{\text{eff}}} u_3 \right) l \tag{3.29}$$
$$\dot{x}_{11} = \frac{d}{I_{xx}} u_2 \tag{3.30}$$
$$\dot{x}_{12} = \frac{1}{I_{zz}} u_1 \left(\frac{S_{\text{elevon}}}{S_{\text{eff}}} u_4 \right) f \tag{3.31}$$

The open loop model was created in the Simulink environment to check the overall relationships. Figures 3.7-3.8 show the structure of the simulation blocks.

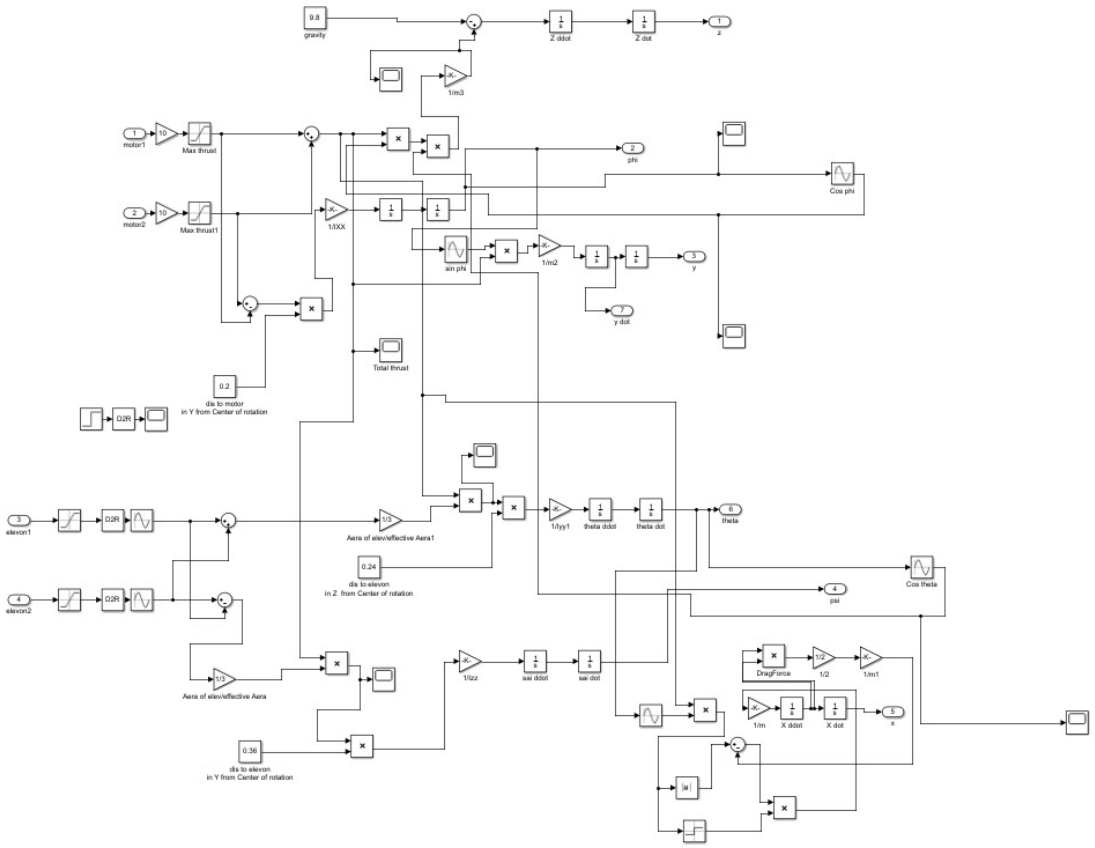


Figure 3.7: Dynamic model simulation in Simulink environment.

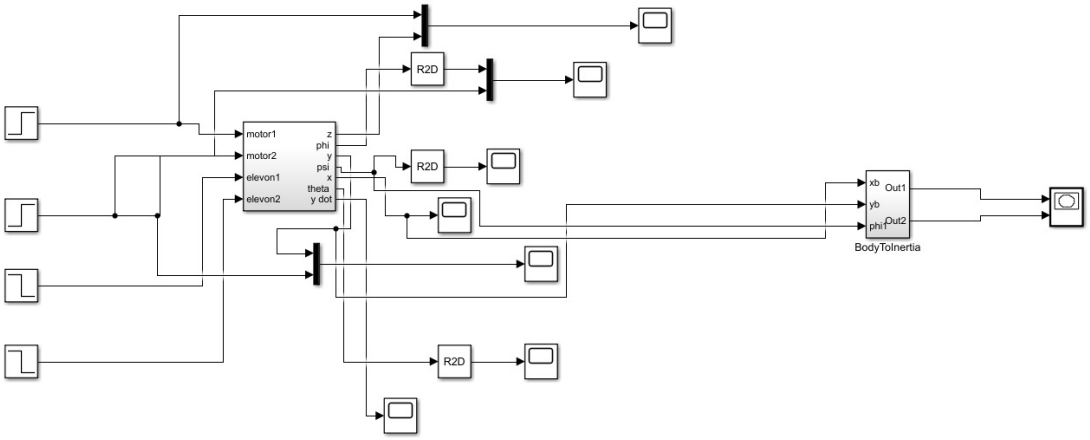


Figure 3.8: Open-loop total simulation.

3.3 Advanced Dynamic Model

3.3.1 Rigid-Body Dynamics

The motion of the VTOL drone is governed by the fundamental equations of rigid-body dynamics. These equations describe both translational and rotational dynamics of the system using Newton-Euler equations.

3.3.1.1 Translational Dynamics

The translational motion of the drone in the body frame is given by Newton's second law:

$$m\dot{\mathbf{V}} + \omega \times (m\mathbf{V}) = \mathbf{F} \tag{3.32}$$

Expanding in component form:

$$\begin{aligned} m\dot{u} - mqw + mrv &= F_x \\ m\dot{v} - mru + mpw &= F_y \\ m\dot{w} - mpv + mqu &= F_z \end{aligned} \tag{3.33}$$

where m is the mass of the drone, $\mathbf{V} = [u, v, w]^T$ is the velocity vector in the body frame, $\omega = [p, q, r]^T$ is the angular velocity in the body frame, and $\mathbf{F} = [F_x, F_y, F_z]^T$ represents the external forces (thrust, aerodynamic forces, and gravity).

Ignoring centrifugal terms in hovering mode, the translational motion simplifies to:

$$m\ddot{\xi} = \mathbf{g} + R\mathbf{T}_B \tag{3.34}$$

where:

- $\xi = [x, y, z]^T$ represents the position of the drone in the global frame, and g represents the gravitational force vector. Expanding in component form:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi S_\theta C_\phi - C_\psi S_\phi \\ C_\theta C_\phi \end{bmatrix} \tag{3.35}$$

3.3.1.2 Rotational Dynamics

The rotational dynamics are described by Euler's equation:

$$\mathbf{J}\dot{\omega} + \omega \times (\mathbf{J}\omega) + \Gamma = \tau \tag{3.36}$$

where:

– **J** is the moment of inertia matrix:

$$\mathbf{J} = \begin{bmatrix} J_{xx} & -J_{xy} & -J_{xz} \\ -J_{yx} & J_{yy} & -J_{yz} \\ -J_{zx} & -J_{zy} & J_{zz} \end{bmatrix} \tag{3.37}$$

– $\omega = [p, q, r]^T$ is the angular velocity vector in the body frame,

– Γ represents gyroscopic forces,

– τ represents external torques applied to the system.

3.3.1.3 Kinematic Equations

The relationship between angular velocity and Euler angle rates is given by:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.38}$$

where:

– (ϕ, θ, ψ) are the roll, pitch, and yaw angles,

– (p, q, r) are the roll, pitch, and yaw rates.

3.3.1.4 Forces and Torques in the Body Frame

The forces and torques acting on the drone are given by:

$$\sum \mathbf{F} = \mathbf{F}_{\text{thrust}} + \mathbf{F}_{\text{aero}} + \mathbf{F}_{\text{gravity}} \tag{3.39}$$

$$\sum \tau = \tau_{\text{thrust}} + \tau_{\text{aero}} + \tau_{\text{disturbance}} \tag{3.40}$$

where:

– $\mathbf{F}_{\text{thrust}}$ and τ_{thrust} are forces and torques generated by the propulsion system,

– $\mathbf{F}_{\text{aero}}$ and τ_{aero} represent aerodynamic forces and torques,

– $\mathbf{F}_{\text{gravity}}$ represents gravitational force,

– $\tau_{\text{disturbance}}$ represents external disturbances.

These equations form the foundation for modeling, simulation, and control of the VTOL drone.

Inertia Matrix for Drone Dynamic Model

We need to introduce an inertia matrix (moment of inertia tensor) for a drone's dynamic model.

The equation (3.41) describes the rotational inertia of the drone, which characterizes how the drone's mass is distributed and how it resists to rotational acceleration. Breaking it down, we obtain:

$$\mathbf{J} = \begin{pmatrix} \int \left(y^2+z^2\right) dM & -\int xydM & -\int xzdM \\ -\int xydM & \int \left(x^2+z^2\right) dM & -\int yzdM \\ -\int xzdM & -\int yzdM & \int \left(x^2+y^2\right) dM \end{pmatrix} = \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{pmatrix} \tag{3.41}$$

This matrix J represents the moment of inertia of the drone around its principal axes (x, y, z). It captures how the drone's mass is distributed relative to its center of rotation. The integrals in the matrix compute the mass distribution across the drone's volume.

The inertia matrix is a 3×3 symmetric matrix. The diagonal elements (J_x, J_y, J_z) represent the moments of inertia around the principal axes, while the off-diagonal elements represent the products of inertia, which show the coupling between rotational axes.

The diagonal terms (J_x, J_y, J_z) are calculated as $\int (y^2+z^2)dM$, $\int (x^2+z^2)dM$, and $\int (x^2+y^2)dM$ respectively. These terms measure the resistance to rotation around each axis. Larger values indicate greater resistance to rotation.

The off-diagonal terms are given by $-\int xydM$, $-\int xzdM$, and $-\int yzdM$. These terms represent cross-coupling between the rotational axes, indicating how rotation about one axis influences motion about the others. They are essential for modeling complex drone dynamics, such as asymmetric rotation.

The inertia matrix is symmetric, meaning $J_{xy} = J_{yx}$, $J_{xz} = J_{zx}$, and so on. This property is crucial for accurately modeling drone attitude and angular momentum. The inertia matrix is widely used in control systems and stability analysis to ensure precise flight dynamics.

For a symmetric drone like quadcopter, many off-diagonal terms would be zero, simplifying the matrix and making calculations more straightforward.

Moments of inertia computation

Determining the moment of inertia matrix of an aircraft is particularly important, as it is the key factor affecting rotational dynamics. There are various methods to achieve this, including laboratory techniques. Typically, these amounts can be calculated by suspending the drone and measuring its oscillation period. However, a more straightforward approach involves using computer simulation software. We chose this path. We first draw a 3D model in the FreeCAD software environment, adjust the density some that the total mass becomes the total mass of the drone to calculate moments of inertia, and to identify which components

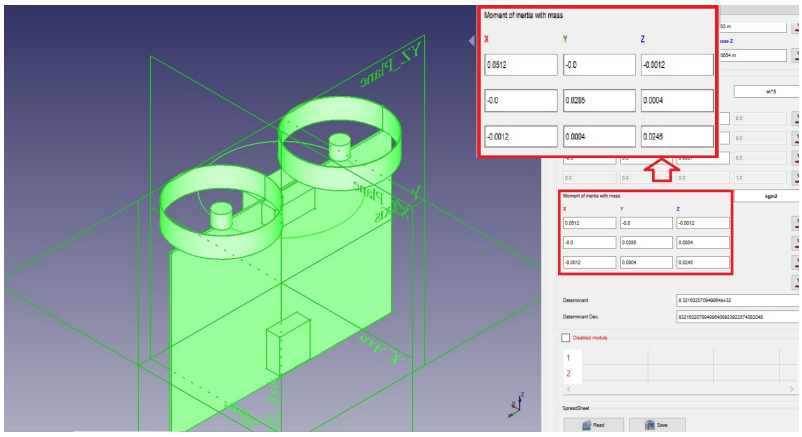


Figure 3.9: 3D-model designed in the FreeCAD environment, and the moments of the inertial matrix are extracted.

are zero in the moments of inertia matrix. In many works, especially in articles related to VTOL quadcopters, this matrix is considered diagonal. Still, for a better approximation, we have used a non-diagonal matrix, where the values of J_{xz} are non-zero, which will also make the equations more complicated as given in Equation (3.42) and 3D model of drone sketched in FreeCAD, and the corresponding moments of inertia matrix is revealed in Figure 3.9 as well:

$$\mathbf{J} = \begin{bmatrix} 0.0512 & 0 & -0.0012 \\ 0 & 0.0285 & 0.0004 \\ -0.0012 & 0.0004 & 0.0245 \end{bmatrix} \tag{3.42}$$

Since this matrix will play a very key role in the future coefficients of the controller system, we did not rely on this open source software (FreeCAD), and the 3D design was re-done in SOLIDWORKS software, as shown in Figure 3.10, and the matrix has been extracted again and is given by:

$$\mathbf{J} = \begin{bmatrix} 0.045 & 0 & 0.001 \\ 0 & 0.031 & 0 \\ 0.001 & 0 & 0.018 \end{bmatrix} \tag{3.43}$$

so $J_{xy} = J_{yz} = 0$ we have:

$$\mathbf{J} = \begin{pmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{pmatrix} \tag{3.44}$$

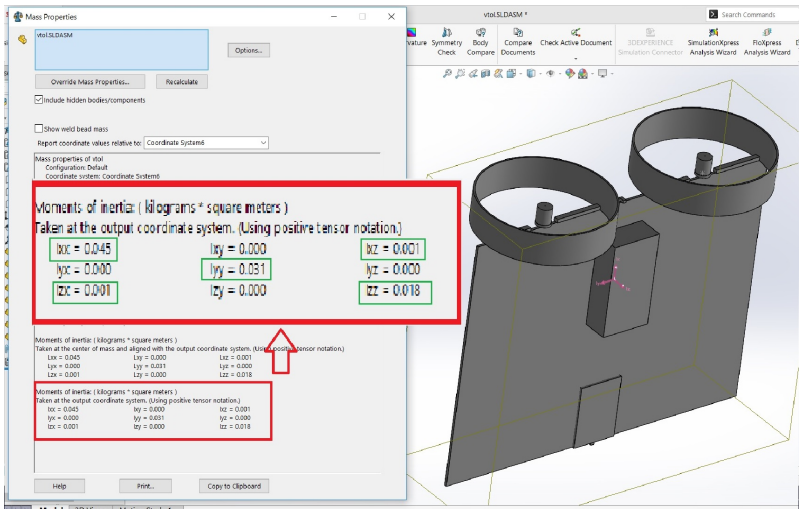


Figure 3.10: 3D-model designed in SOLIDWORKS environment and moments of inertia matrix is extracted.

This symmetry assumption helps to simplify the analysis. The inverse of $\mathbf{J}$ becomes:

$$\mathbf{J}^{-1} = \frac{\text{adj}(\mathbf{J})}{\det(\mathbf{J})} = \frac{\begin{pmatrix} J_y J_z & 0 & J_y J_{xz} \\ 0 & J_x J_z - J_{xz}^2 & 0 \\ J_{xz} J_y & 0 & J_x J_y \end{pmatrix}}{J_x J_y J_z - J_{xz}^2 J_y}$$
$$= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix}$$

where: $\Gamma = J_x J_z - J_{xz}^2$

Now we rewrite Equation (3.36) by substituting the obtained inertia matrix, which becomes:

$$
\begin{aligned}
\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} &= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \left[\begin{pmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{pmatrix} \begin{pmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \right. \\
&\quad \left. -I_r \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_\Gamma + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right] \\
&= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \left[\begin{pmatrix} J_{xz}pq + (J_y - J_z)qr \\ J_{xz}\left(r^2 - p^2\right) + (J_z - J_x)pr \\ (J_x - J_y)pq - J_{xz}qr \end{pmatrix} + \begin{pmatrix} -qI_r\omega_\Gamma \\ pI_r\omega_\Gamma \\ 0 \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right] \\
&= \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr + \Gamma_3 l + \Gamma_4 n - \Gamma_3 qI_r\omega_\Gamma \\ \Gamma_5 pr - \Gamma_6\left(p^2 - r^2\right) + \frac{1}{J_y}m + \frac{1}{J_y}pI_r\omega_\Gamma \\ \Gamma_7 pq - \Gamma_1 qr + \Gamma_4 l + \Gamma_8 n - \Gamma_4 qI_r\omega_\Gamma \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\Gamma_1 &= \frac{J_{xz}\left(J_x - J_y + J_z\right)}{\Gamma} \\
\Gamma_2 &= \frac{J_z\left(J_z - J_y\right) + J_{xz}^2}{\Gamma} \\
\Gamma_3 &= \frac{J_z}{\Gamma} \\
\Gamma_4 &= \frac{J_{xz}}{\Gamma} \\
\Gamma_5 &= \frac{J_z - J_x}{J_y} \\
\Gamma_6 &= \frac{J_{xz}}{J_y} \\
\Gamma_7 &= \frac{\left(J_x - J_y\right)J_x + J_{xz}^2}{\Gamma} \\
\Gamma_8 &= \frac{J_x}{\Gamma} \\
\Gamma &= J_x J_z - J_{xz}^2
\end{aligned}
$$

$$
\omega_\Gamma = \omega_1 - \omega_2 \tag{3.45}
$$

which ω_1 and ω_2 are the angular velocity for each motor. Since we do not have access to a sensor that can measure the RPM of the motors, we convert this equation to the thrust of the motors:

$$
\begin{aligned}
T_1 &= b\omega_1^2 \\
T_2 &= b\omega_2^2 \\
\omega_\Gamma &= \sqrt{T_1/b} - \sqrt{T_2/b}
\end{aligned} \tag{3.46}
$$

where T is the thrust force [N] , b is the thrust coefficient [N·s^2/rad^2] calculated in section (3.6.3), and ω is the propeller angular velocity [rad/s]. Here, to avoid confusion between formulas, we used the symbol $I_r(J_r)$ for the total moment of inertia of each motor and its propeller, which is explained in full in section (3.6.4). Now, the torque around the three

coordinate axes that will be introduced as input to the control system is calculated is given by:

$$\tau_B = \begin{bmatrix} l \\ m \\ n \end{bmatrix} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk\left(-\omega_1^2 + \omega_2^2\right) \\ \tau_{elevon1} + \tau_{elevon2} \\ \sum_{i=1}^{2} \tau_{Mi} + \tau_{elevon1} - \tau_{elevon2} \end{bmatrix} \tag{3.47}$$

τ_B is a torque vector with three components [l, m, n] representing torques around 3 axes.

$$\tau_{elevon_i} = F_{lift_i} \times r_i = \frac{1}{2}\rho V_a^2 S_{elevon} C_L(\alpha, \delta_i) \times r_i \tag{3.48}$$

while the lift force of each elevon takes the form:

$$F_{\text{lift}} = \frac{1}{2}\rho V_a^2 S C_L(\alpha) \tag{3.49}$$

where τ_{elevon_i} is the torque generated by elevon i (where $i = 1,2$), F_{lift_i} is the lift force of elevon i, ρ is the air density, V_a is the airspeed, S_{elevon} is the elevon surface area, $C_L(\alpha, \delta_i)$ is the lift coefficient as a function of angle of attack α and elevon deflection angle δ_i, and r_i is the moment arm from elevon i to the center of mass.

3.3.2 Basic formula for lift force coefficient ($C_L(\alpha)$)

Formally, there is some angle of orientation such that the airfoil at position y develops no lift. For airstreams of velocity V oriented at an angle α relative to the liftless angle, the airfoil will develop some circulation $V \cdot C(y,\alpha)$; for small α, Taylor expansion approximates that circulation as $V \cdot \partial C/\partial\alpha(y,0) \cdot \alpha$. If the airfoil is ideal and has chord $c(y)$, then theory predicts that

$$C(y,\alpha) = 2\pi c(y)\sin(\alpha)$$

but real airfoils may be less efficient (Acheson, 1990) and (Auld and Srinivas, 1995).

Starting with: $C(y,\alpha) = 2\pi c(y)\sin(\alpha)$ we obtain: $CL = 2\pi \times \sin(\alpha) \times \cos(\alpha)$. The transformation happens because:
1. For a flat plate, when we want to find the total lift coefficient (CL) rather than the sectional lift coefficient $C(y,\alpha)$, we need to account for the entire flow field around the plate
2. The additional $\cos(\alpha)$ term appears because it accounts for the projection of the force normal to the plate into the lift direction, ensuring that the total lift force is considered perpendicular to the free stream, and the original force on the plate is multiplied by $\cos(\alpha)$ to obtain its vertical (lift) component.

So the progression is:
1. First equation: local sectional coefficient
2. Second equation: total lift coefficient accounting for flow field and force resolution

For more accurate simulation of rapid changes in angle of attack, we employ a sophisticated flat-plate model for calculating $C_L(\alpha)$. Since we're not using a specific airfoil profile, this model provides a reasonable approximation of the lift coefficient behavior.

The lift coefficient is given by:

$$C_L(\alpha) = \sigma(\alpha) \cdot 2\sin^2(\alpha)\cos(\alpha)\,\text{sign}(\alpha) \tag{3.50}$$

where:

α is the angle of attack, $\sigma(\alpha)$ is a stall modulation function $\text{sign}(\alpha)$ ensures proper sign convention for positive and negative angles of attack. The $\sin^2(\alpha)\cos(\alpha)$ term captures the nonlinear behavior of lift at high angles of attack. where:

$$\sigma(\alpha) = \frac{1 + e^{-M(\alpha-\alpha_0)} + e^{M(\alpha+\alpha_0)}}{\left(1 + e^{-M(\alpha-\alpha_0)}\right)\left(1 + e^{M(\alpha+\alpha_0)}\right)} \tag{3.51}$$

The $\sigma(\alpha)$ function helps model how lift responds to rapidly changing angles of attack

In (3.51), M is a parameter that controls the steepness of the sigmoid curve. It determines how quickly the function transitions between its minimum and maximum values as the angle of attack α changes.

The choice of M depends on the specific characteristics of your drone and its control surfaces. Here are some considerations:

A larger M value will create a steeper transition, meaning the lift force will change more abruptly with small changes in the angle of attack. A smaller M value will result in a more gradual transition, where the lift force changes more smoothly over a wider range of angles of attack.

M in this equation is known as the Kirchhoff's separation parameter or flow separation parameter. It determines how abruptly the flow separation occurs. If we want to categorize:

1. Physical Meaning:

- M controls the sharpness of the transition in the flow separation function $\sigma(\alpha)$
- Higher M values $\Rightarrow$ more sudden/abrupt flow separation
- Lower M values $\Rightarrow$ more gradual flow separation

2. Effect on $\sigma(\alpha)$:

- When M = 0: $\sigma(\alpha) \approx 1$ (no flow separation)
- As M increases:

- The transition becomes steeper
- Flow separation occurs more suddenly near α_0
- The curve of $\sigma(\alpha)$ becomes more step-like

3. Typical Values:

- M $\approx$ 0–5: Very gradual separation (like highly streamlined bodies)
- M $\approx$ 15–25: Typical for standard flat plates
- M $\approx$ 40–50: Very abrupt separation (sharp edges)

The M parameter essentially models how quickly the flow detaches from the surface as the angle of attack increases beyond the critical angle α_0. It's an empirical parameter that needs to be determined based on experimental data for specific geometries and flow conditions.

Typically, M values range from about 10 to 50, but the exact value should be determined based on:

a) Wind tunnel tests or computational fluid dynamics (CFD) simulations of your specific elevon design. b) The desired responsiveness of your control system. c) The operational range of angles of attack for your drone.

To select an appropriate M value:

We start with a mid-range value in simulation, such as M = 25. Compare the resulting curve to experimental data or CFD results for your elevon. Adjust M up or down to better match the actual behavior of your control surface. Iterate this process until we find a value that provides a good fit for our specific application. Remember that this model is an approximation, and the best M value will depend on the specific characteristics of our VTOL drone and its operating conditions. we need to refine this value through testing and validation of our drone's performance. Finally, for better understanding, a sigmoid graph is shown in Figure 3.11:

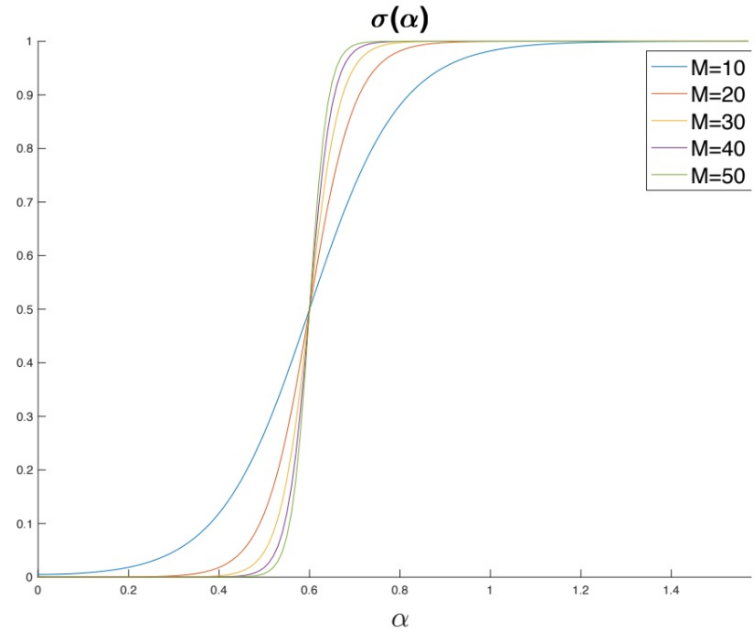


Figure 3.11: $\sigma(\alpha)$ versus α

We expand the Equation (3.47) formula and continue with replacing Equation (3.48) on that:

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} c_3 k(-\omega_1^2+\omega_2^2) \\ \frac{1}{2}\rho S_{\text{elevon}} c_1 (V_{a1}^2 \sigma(\delta_{e1})[2\,\text{sign}(\delta_{e1})\sin^2\delta_{e1}\cos\delta_{e1}] + V_{a2}^2\sigma(\delta_{e2})[2\,\text{sign}(\delta_{e2})\sin^2\delta_{e2}\cos\delta_{e2}]) \\ \frac{1}{2}\rho S_{\text{elevon}} c_2 (V_{a1}^2 \sigma(\delta_{e1})[2\,\text{sign}(\delta_{e1})\sin^2\delta_{e1}\cos\delta_{e1}] - V_{a2}^2\sigma(\delta_{e2})[2\,\text{sign}(\delta_{e2})\sin^2\delta_{e2}\cos\delta_{e2}]) \end{bmatrix} \tag{3.52}$$

where c_1, c_2, and c_3 represent the f, l, and d, respectively. V_a is the velocity of the airflow of each motor. For calculating V_a we use the total thrust for the electrical brushless motor:

$$T = \rho A V_a^2 C_T \tag{3.53}$$

Due to the lack of a speed sensor, we are attempting to eliminate the airflow velocity in this equation, so we replace V_a from Equation (3.53) to Equation (3.52).

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l(T_1 - T_2) \\ \frac{1}{2} S_{\text{elevon}} c_1 \left(\frac{T_1}{C_T A}\sigma_{\alpha_1}\left[2\,\text{sign}(\delta_{e1})\sin(\delta_{e1})^2\cos(\delta_{e1})\right] + \frac{T_2}{C_T A}\sigma_{\alpha_2}\left[2\,\text{sign}(\delta_{e2})\sin(\delta_{e2})^2\cos(\delta_{e2})\right] \right) \\ \frac{1}{2} S_{\text{elevon}} c_2 \left(\frac{T_1}{C_T A}\sigma_{\alpha_1}\left[2\,\text{sign}(\delta_{e1})\sin(\delta_{e1})^2\cos(\delta_{e1})\right] - \frac{T_2}{C_T A}\sigma_{\alpha_2}\left[2\,\text{sign}(\delta_{e2})\sin(\delta_{e2})^2\cos(\delta_{e2})\right] \right) \end{bmatrix} \tag{3.54}$$

However, a commonly used range for C_T for quadrotor propellers is between 0.05 and 0.12. and A (S_{eff} which mentioned in previous part) is area of the propeller:

$$A = \pi \cdot (13\text{ cm})^2 = 0.0531\text{ m}^2 \tag{3.55}$$

3.4 Drone state-space model

Considering the state variables of the drone system as $X = [x_1, x_2, \ldots, x_{12}]^T$, it is described in the form of the state space $\dot{X} = f(X,U)$ as follows:

$$\begin{cases} x_1 = \phi, & x_7 = x \\ x_2 = \dot{x}_1 = \dot{\phi}, & x_8 = \dot{x}_7 = \dot{x} \\ x_3 = \theta, & x_9 = y \\ x_4 = \dot{x}_3 = \dot{\theta}, & x_{10} = \dot{x}_9 = \dot{y} \\ x_5 = \psi, & x_{11} = z \\ x_6 = \dot{x}_5 = \dot{\psi}, & x_{12} = \dot{x}_{11} = \dot{z} \end{cases}$$

The dynamic model state-space of the VTOL drone is described in the following form:

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

$$\dot{x}_2 = \Gamma_1 x_2 x_4 - \Gamma_2 x_4 x_6 + \Gamma_3 u_2 + \Gamma_4 u_4 - \Gamma_3 x_4 J_r \left(\sqrt{\left|\frac{T_1}{b}\right|} - \sqrt{\left|\frac{T_2}{b}\right|}\right)$$

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

$$\dot{x}_4 = \Gamma_5 x_2 x_6 - \Gamma_6 (x_2^2 - x_6^2) + \frac{u_3}{I_{yy}} + \frac{x_2 J_r}{I_{yy}} \left(\sqrt{\left|\frac{T_1}{b}\right|} - \sqrt{\left|\frac{T_2}{b}\right|}\right)$$

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

$$\dot{x}_6 = \Gamma_7 x_2 x_4 - \Gamma_1 x_4 x_6 + \Gamma_4 u_2 + \Gamma_8 u_4 - \Gamma_4 x_4 J_r \left(\sqrt{\left|\frac{T_1}{b}\right|} - \sqrt{\left|\frac{T_2}{b}\right|}\right)$$

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

$$\dot{x}_8 = u_x \frac{u_1}{m}$$

$$\dot{x}_9 = x_{10}$$

$$\dot{x}_{10} = u_y \frac{u_1}{m}$$

$$\dot{x}_{11} = x_{12}$$

$$\dot{x}_{12} = -g + \frac{(\cos(x_1)\cos(x_3))u_1}{m}$$

Where u_x and u_y are defined as artificial control input:

$$\begin{aligned} u_x &= (\cos x_1 \sin x_3 \cos x_5 + \sin x_1 \cos x_5) \\ u_y &= (\cos x_1 \sin x_3 \sin x_5 - \sin x_1 \cos x_5) \end{aligned} \tag{3.56}$$

J_r is the total moment of inertia of each motor and its propeller, which is explained in full in section (3.6.4).

3.5 Sliding mode controller simulation and design

strong cross-axis coupling, and control inputs that are highly dependent on aerodynamic effects. Moreover, key aerodynamic parameters such as lift and thrust coefficients are subject to uncertainty and cannot be directly measured during operation, which limits the applicability of nonlinear control approaches that rely on accurate model knowledge. Within the class of nonlinear control methods, Sliding Mode Control (SMC) provides an effective framework for handling such dynamics, as it does not require exact aerodynamic parameter identification and maintains a comparatively simple control structure when contrasted with adaptive or backstepping-based techniques.

3.5.1 Sliding model controller design process

- First step: definition of switching level

$$S = e = x - x_d$$

- Second step: obtaining equivalent control energy u_{eq}

For a general system of the form $\dot{x} = f(x) + g(x)u$, to obtain the equivalent control energy, we must set the derivative of the switching level equal to zero:

$$\dot{S} = \dot{e} = \dot{x} - \dot{x}_d = f(x) + g(x)u - \dot{x}_d = 0$$
$$u_{eq} = g(x)^{-1}\left(\dot{x}_d - f(x)\right)$$

- The third step: obtaining the total control signal u

In order for the controller to be resistant to disturbances, a second component is added to it, and the new controller will be obtained as follows:

$$u = u_{ea} - d \times \text{sign}(s) \tag{3.57}$$

The controller is fully designed, but there is still an unknown in the controller, and that is the parameter d, to obtain it, the Lyapunov function is used. That is, we consider a Lyapunov function, which in this method is always a quadratic function:

$$V(x) = \frac{1}{2}S^2 \tag{3.58}$$

The stability of the system requires that the Lyapunov function's derivative must be negative. A more desirable Lyapunov function is characterized by a derivative that is negative definite. This is advantageous because the steeper negative slope causes the system's internal energy to decay more rapidly to zero. Thus, a more negative derivative of the Lyapunov

function leads to better system performance. For practical implementation, rather than simply requiring the derivative to be negative, we establish a stricter condition by setting it to be less than some threshold. This threshold will later serve as an adjustable design parameter in our simulation studies.

$$\begin{aligned}
&\dot{V}(x) = s\dot{s} \leq -\eta \\
&s\dot{s} \leq -\eta|s| \\
&s(f(x)+bu+\delta(x)) = s(-db\,\mathrm{sign}(s)+\delta(x)) \\
&\leq -db|s| + |s\delta(x)| \\
&= -db|s| + |s||\delta(x)| = |s|(-db+|\delta(x)|) \leq -\eta|s| \\
&-db+k \leq -\eta \rightarrow d \geq \frac{\eta+k}{b}
\end{aligned} \tag{3.59}$$

where η is a positive number. Therefore, the origin point $s = 0$ becomes asymptotically stable if we choose d as above. For the drone case, we define the slide surfaces based on the error of the system modes:

$$\begin{cases} e_i = x_{id} - x_i \\ e_{i+1} = \dot{x}_{id} - x_{i+1} \end{cases}$$

$$\begin{cases}
s_\varphi = e_2 + \alpha_1 e_1 = \dot{x}_{1d} - x_2 + \alpha_1 (x_{1d} - x_2) \\
s_\theta = e_4 + \alpha_2 e_3 = \dot{x}_{3d} - x_4 + \alpha_2 (x_{3d} - x_4) \\
s_\psi = e_6 + \alpha_3 e_5 = \dot{x}_{5d} - x_6 + \alpha_3 (x_{5d} - x_6) \\
s_x = e_8 + \alpha_4 e_7 = \dot{x}_{7d} - x_8 + \alpha_4 (x_{7d} - x_8) \\
s_y = e_{10} + \alpha_5 e_9 = \dot{x}_{9d} - x_{10} + \alpha_5 (x_{9d} - x_{10}) \\
s_z = e_{12} + \alpha_6 e_{11} = \dot{x}_{11d} - x_{12} + \alpha_6 (x_{11d} - x_{12})
\end{cases}$$

We calculate the derivative of the slide surfaces:

$$\begin{aligned}
\dot{s}_\varphi &= \dot{e}_2 + \alpha_1 \dot{e}_1 = \ddot{x}_{1d} - \dot{x}_2 + \alpha_1 (\dot{x}_{1d} - x_2) \\
&= -\Gamma_1 x_2 x_4 + \Gamma_2 x_4 x_6 - \Gamma_3 u_2 - \Gamma_4 u_4 + \Gamma_3 x_4 J_r \Omega - w_\varphi + \ddot{\varphi}_d + \alpha_1 (\dot{\varphi}_d - x_2) \\
\dot{s}_\theta &= \dot{e}_4 + \alpha_2 \dot{e}_3 = \ddot{x}_{3d} - \dot{x}_4 + \alpha_2 (\dot{x}_{3d} - x_4) \\
&= -\Gamma_5 x_2 x_6 + \Gamma_6 (x_2^2 + x_6^2) - \frac{u_3}{I_{yy}} - \frac{x_2 J_r}{I_{yy}} \Omega - w_\theta + \ddot{\theta}_d + \alpha_2 \left(\dot{\theta}_d - x_4\right) \\
\dot{s}_\psi &= \dot{e}_6 + \alpha_3 \dot{e}_5 = \ddot{x}_{5d} - \dot{x}_6 + \alpha_2 (\dot{x}_{5d} - x_6) \\
&= -\Gamma_7 x_2 x_4 + \Gamma_1 x_4 x_6 - \Gamma_4 u_2 - \Gamma_8 u_4 + \Gamma_4 x_4 J_r \Omega - w_\psi + \ddot{\psi}_d + \alpha_3 (\dot{\psi}_d - x_6)
\end{aligned}$$

$$
\begin{aligned}
\dot{s}_x &= \dot{e}_8 + \alpha_4 \dot{e}_7 = \ddot{x}_{7d} - \dot{x}_8 + \alpha_4 (\dot{x}_{7d} - x_8) \\
&= -w_x - U_x U_1/m + \ddot{x}_d + \alpha_4 (\dot{x}_d - x_8) \\
\dot{s}_y &= \dot{e}_{10} + \alpha_5 \dot{e}_9 = \ddot{x}_{9d} - \dot{x}_{10} + \alpha_5 (\dot{x}_{9d} - x_{10}) \\
&= -w_y - U_y U_1/m + \ddot{y}_d + \alpha_5 (\dot{y}_d - x_{10}) \\
\dot{s}_z &= \dot{e}_{12} + \alpha_6 \dot{e}_{11} = \ddot{x}_{11d} - \dot{x}_{12} + \alpha_6 (\dot{x}_{11d} - x_{12}) \\
&= g - \frac{(\cos x_1 \cos x_3)}{m} U_1 - w_z + \ddot{z}_d + \alpha_6 (\dot{z}_d - x_{12})
\end{aligned}
$$

By setting the slide surfaces equal to zero and including the discontinuous robustness terms, we calculate the control forces:

$$
\begin{cases}
U_1 &= \frac{m}{(\cos x_1 \cos x_3)} (-k_6 \operatorname{sign}(s_z) + g - w_z + \ddot{z}_d + \alpha_6 (\dot{z}_d - x_{12})) \\
U_2 &= \frac{1}{\Gamma_3} \left(-k_1 \operatorname{sign}(s_\varphi) - \Gamma_1 x_2 x_4 + \Gamma_2 x_4 x_6 - \Gamma_4 u_4 + \Gamma_3 x_4 J_r \Omega - w_\varphi + \ddot{\varphi}_d + \alpha_1 e_2\right) \\
U_3 &= \frac{1}{I_{yy}} \left(-k_2 \operatorname{sign}(s_\theta) - \Gamma_5 x_2 x_6 + \Gamma_6 (x_2^2 + x_6^2) - \frac{x_2 J_r}{I_{yy}} \Omega - w_\theta + \ddot{\theta}_d + \alpha_2 e_4\right) \\
U_4 &= \frac{1}{b_3} \left(-k_3 \operatorname{sign}(s_\psi) - x_4 x_2 a_5 - w_\psi + \ddot{\psi}_d + \alpha_3 (\dot{\psi}_d - x_6)\right) \\
U_x &= \frac{m}{U_1} (-k_4 \operatorname{sign}(s_x) - w_x + \ddot{x}_d + \alpha_4 e_8) \\
U_y &= \frac{m}{U_1} (-k_5 \operatorname{sign}(s_y) - w_y + \ddot{y}_d + \alpha_5 e_{10})
\end{cases}
$$

Using Equation (3.46) we can reach to this:

$$
u_1 = \frac{m}{(\cos x_1 \cos x_3)} \left(-k_6 \operatorname{sign}\left(\frac{S_z}{\varepsilon_{Th}}\right) + g - \alpha_6 (0 - x_{12})\right)
$$

$$
u_2 = \frac{1}{\Gamma_3} \left(-k_1 \operatorname{sign}\left(\frac{S_\varphi}{\varepsilon_p}\right) - \Gamma_1 x_2 x_4 + \Gamma_2 x_4 x_6 - \Gamma_4 u_4 + \Gamma_3 x_4 J_r \left(\sqrt{\left|\frac{T_1}{b}\right|} - \sqrt{\left|\frac{T_2}{b}\right|}\right) + \alpha_1 (0 - x_2)\right)
$$

$$
u_3 = \frac{1}{b_2} \left(-k_2 \operatorname{sign}\left(\frac{S_\theta}{\varepsilon_q}\right) - \Gamma_5 x_2 x_6 + \Gamma_6 (x_2^2 - x_6^2) - \frac{1}{I_{yy}} x_2 J_r \left(\sqrt{\left|\frac{T_1}{b}\right|} - \sqrt{\left|\frac{T_2}{b}\right|}\right) + \alpha_2 (0 - x_4)\right)
$$

$$
u_4 = \frac{1}{\Gamma_8} \left(-k_3 \operatorname{sign}\left(\frac{S_\psi}{\varepsilon_r}\right) - \Gamma_7 x_2 x_4 + \Gamma_1 x_4 x_6 - \Gamma_4 u_2 + \Gamma_4 x_4 J_r \left(\sqrt{\left|\frac{T_1}{b}\right|} - \sqrt{\left|\frac{T_2}{b}\right|}\right) + \alpha_3 (0 - x_6)\right)
$$

The k's and α's gains are obtained through trial and error methods or algorithms such as the genetic algorithm (PSO). The variables $w_x, w_y, w_z, w_\varphi, w_\theta$, and w_ψ represent environmental disturbances or external perturbations such as wind, atmospheric turbulence, and unmodeled dynamic effects which can influence the UAV's motion across different axes and rotational planes.

In general, to prove the stability of the system, we define the Lyapunov function as follows:

$$
V = \frac{1}{2} \left(S_x^T S_x + S_y^T S_y + S_z^T S_z + S_\varphi^T S_\varphi + S_\theta^T S_\theta + S_\psi^T S_\psi\right) \tag{3.60}
$$

Computing its time derivative, we obtain:

$$\dot{V} = S_{\varphi}^{T}\dot{S}_{\varphi} + S_{\theta}^{T}\dot{S}_{\theta} + S_{\psi}^{T}\dot{S}_{\psi} + S_{x}^{T}\dot{S}_{x} + S_{y}^{T}\dot{S}_{y} + S_{z}^{T}\dot{S}_{z} \tag{3.61}$$

$$\begin{aligned}
\dot{V} = {} & s_x\left(-w_x - \frac{U_x U_1}{m} + \ddot{x}_d + \alpha_4(\dot{x}_{7d} - x_8)\right) \\
& + s_y\left(-w_y - \frac{U_y U_1}{m} + \ddot{y}_d + \alpha_5(\dot{x}_{9d} - x_{10})\right) \\
& + s_z\left(g - \frac{(\cos x_1 \cos x_3)}{m}U_1 - w_z + \ddot{z}_d + \alpha_6(\dot{z}_d - x_{12})\right) \\
& + s_\varphi\left(-\Gamma_1 x_2 x_4 + \Gamma_2 x_4 x_6 - \Gamma_3 u_2 - \Gamma_4 u_4 + \Gamma_3 x_4 J_r \Omega - w_\varphi + \ddot{\varphi}_d + \alpha_1(\dot{\varphi}_d - x_2)\right) \\
& + s_\theta\left(-\Gamma_5 x_2 x_6 + \Gamma_6(x_2^2 + x_6^2) - \frac{u_3}{I_{yy}} - \frac{x_2 J_r}{I_{yy}}\Omega - w_\theta + \ddot{\theta}_d + \alpha_2(\dot{\theta}_d - x_4)\right) \\
& + s_\psi\left(-\Gamma_7 x_2 x_4 + \Gamma_1 x_4 x_6 - \Gamma_4 u_2 - \Gamma_8 u_4 + \Gamma_4 x_4 J_r \Omega - w_\psi + \ddot{\psi}_d + \alpha_3(\dot{\psi}_d - x_6)\right)
\end{aligned} \tag{3.62}$$

Substituting the obtained expressions for the control signals into the previous equation, we obtain:

$$\dot{V}(s) = s\dot{s} < -k_x|s_x| - k_y|s_y| - k_z|s_z| - k_\varphi|s_\varphi| - k_\theta|s_\theta| - k_\psi|s_\psi| \tag{3.63}$$

Therefore, one can conclude that $\dot{V}(s)$ is negative definite, which guarantees the asymptotic stability of the closed-loop system according to Lyapunov's direct method.

3.6 Simulation

The Sliding Mode Control (SMC) strategy for the VTOL aircraft system is implemented and evaluated through multiple simulation stages. The process begins with initialization, where the program establishes crucial parameters including aerodynamic coefficients, physical constants (such as g and m), aircraft specifications (like propeller area A and arm length l), and moment of inertia values (I_{xx}, I_{yy}, I_{zz}).

A critical aspect of this simulation is the actuator inversion and linearization process. The system employs thrust allocation equations $T_1 = \frac{lu_1+u_2}{2l}$ and $T_2 = \frac{lu_1-u_2}{2l}$ to convert control signals into physical thrust commands. Furthermore, the elevon deflection angles are computed through the inverse mapping equations $\delta_{e1} = \frac{l(u_3/c_1+u_4/c_2)}{AeroCo(lu_1+u_2)}$ and $\delta_{e2} = \frac{l(u_3/c_1-u_4/c_2)}{AeroCo(lu_1-u_2)}$, which linearize the aerodynamic effects which will be fully addressed in the section 3.6.1

The simulation operates on a fine-grained timeline with $\Delta t = 0.005$ second intervals solving with Euler method (see appendix A.1). During each time step, the controller calculates sliding surfaces S for all six degrees of freedom: roll (ϕ), pitch (θ), yaw (ψ), and translational movements in (x, y, z) directions. The system then computes primary control inputs (u_1, u_2, u_3, u_4, u_x, u_y), which are transformed through the aforementioned mappings as shown in Figure 3.12.

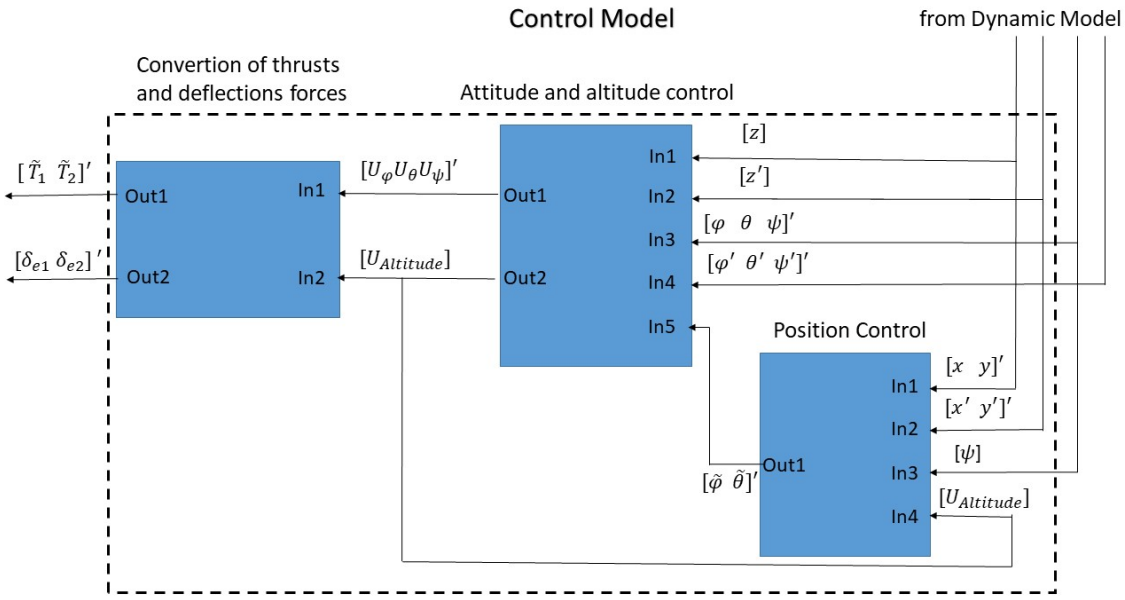


Figure 3.12: Hierarchical control architecture for VTOL (Vertical Take-Off and Landing) aircraft, comprising three cascaded modules: position control, attitude/altitude control, and thrust/elevon deflection conversion.

The state variables (x_1 through x_{12}) are continuously updated using these control inputs, accounting for complex dynamic interactions between different motion components. The simulation concludes with visualization through two plot sets: the first displaying Euler angles and their rates, and the second showing position and velocity data in all three dimensions.

3.6.1 C_L approximation

To linearize the equation:

$$C_L(\alpha) = \sigma(\alpha)\left[2\,\text{sign}(\alpha)\sin^2\alpha\cos\alpha\right] \tag{3.64}$$

We apply several steps of simplification and approximation, particularly for small angles of attack (α).

1. First, consider the $\sigma(\alpha)$ term. For small angles, we can approximate $\sigma(\alpha) \approx 1$, as it's typically close to 1 for small α in the sigmoid function.
2. Focus on the term inside the square brackets:

$$2\,\text{sign}(\alpha)\sin^2(\alpha)\cos(\alpha)$$

3. For small angles:

- $\text{sign}(\alpha) \approx \alpha/|\alpha|$ (which is just 1 for positive α, -1 for negative α)
- $\sin(\alpha) \approx \alpha$
- $\cos(\alpha) \approx 1$

4. Applying these approximations:

$$2\,\text{sign}(\alpha)\sin^2(\alpha)\cos(\alpha) \approx 2 \cdot 1 \cdot \alpha^2 \cdot 1 = 2\alpha^2$$

5. However, α^2 is still nonlinear. For very small angles, α^2 is much smaller than α, so we can further approximate this as:

$$2\alpha^2 \approx k\alpha$$

   Where k is a constant that can be determined empirically or through more detailed analysis.

6. Putting this all together, our linearized equation becomes:

$$C_L(\alpha) \approx k\alpha \tag{3.65}$$

   Where k is often referred to as the lift slope or lift curve slope.

This linear approximation is valid for small angles of attack, typically in the range of -15° to $+15^\circ$ (or about -0.26 to $+0.26$ radians), depending on the specific airfoil and flight conditions.

Note that this linearization sacrifices accuracy for simplicity and is most useful for preliminary calculations or control system design. For more accurate results, especially at larger angles of attack, we would need to use the full nonlinear equation or empirical data.

3.6.2 System Analysis and Linearization

Original Nonlinear System Description

The nonlinear control system is described by two coupled equations:

$$u_3 = \frac{1}{2}S_{elevon}c_1\left(\frac{T_1}{C_T A}\sigma_{\alpha_1}f_1(\delta_{e1}) + \frac{T_2}{C_T A}\sigma_{\alpha_2}f_2(\delta_{e2})\right) \tag{3.66}$$

$$u_4 = \frac{1}{2}S_{elevon}c_2\left(\frac{T_1}{C_T A}\sigma_{\alpha_1}f_1(\delta_{e1}) - \frac{T_2}{C_T A}\sigma_{\alpha_2}f_2(\delta_{e2})\right) \tag{3.67}$$

where

$$f_k(\delta_{ek}) = 2\text{sign}(\delta_{ek})\sin^2(\delta_{ek})\cos(\delta_{ek}), \quad k = 1,2 \tag{3.68}$$

Nonlinear Function Analysis

The derivative of the nonlinear function f_k with respect to δ_{ek} is:

$$\frac{\partial f_k}{\partial \delta_{ek}} = 2\text{sign}(\delta_{ek})(2\sin(\delta_{ek})\cos^2(\delta_{ek}) - \sin^3(\delta_{ek})) \tag{3.69}$$

At the equilibrium point $\delta_{ek} = 0$:

$$\lim_{\delta_{ek}\to 0} \frac{\partial f_k}{\partial \delta_{ek}} = 2 \tag{3.70}$$

The necessity of limit analysis and limit-based evaluation of 3.70 with the one-sided Limits method is cleared. At the equilibrium point $\delta_{ek} = 0$, we evaluate:

1. For $\lim(\delta_{ek} \to 0^+)$**:** Since $\text{sign}(\delta_{ek}) = +1$ (as we approach from the positive side) and as $\delta_{ek} \to 0$, we have $\sin(\delta_{ek}) \approx \delta_{ek}$, $\cos(\delta_{ek}) \approx 1$, and $\sin^3(\delta_{ek}) \approx \delta_{ek}^3 \to 0$.

Therefore:

$$\begin{aligned}
\lim(\delta_{ek} \to 0^+)\frac{\partial f_k}{\partial \delta_{ek}} &= 2(1)(2(\delta_{ek})(1) - \delta_{ek}^3) \\
&= 2(2\delta_{ek} - \delta_{ek}^3) \\
&= 2(2\delta_{ek}) \quad [\text{as } \delta_{ek}^3 \to 0 \text{ faster than } \delta_{ek}] \\
&= 2
\end{aligned}$$

$$\lim_{\delta_{ek}\to 0^+} \frac{\partial f_k}{\partial \delta_{ek}} = 2 \tag{3.71}$$

2. For $\lim(\delta_{ek} \to 0^-)$**:** Since $\text{sign}(\delta_{ek}) = -1$ (as we approach from the negative side) and as $\delta_{ek} \to 0$, we have $\sin(\delta_{ek}) \approx \delta_{ek}$, $\cos(\delta_{ek}) \approx 1$, and $\sin^3(\delta_{ek}) \approx \delta_{ek}^3 \to 0$.

Therefore:

$$\begin{aligned}
\lim(\delta_{ek} \to 0^-)\frac{\partial f_k}{\partial \delta_{ek}} &= 2(-1)(2(\delta_{ek})(1) - \delta_{ek}^3) \\
&= -2(2\delta_{ek} - \delta_{ek}^3) \\
&= -2(2\delta_{ek}) \quad [\text{as } \delta_{ek}^3 \to 0 \text{ faster than } \delta_{ek}] \\
&= 2 \quad [\text{since } \delta_{ek} \text{ is negative}]
\end{aligned}$$

$$\lim_{\delta_{ek}\to 0^-} \frac{\partial f_k}{\partial \delta_{ek}} = 2 \tag{3.72}$$

Since both one-sided limits exist and are equal:

$$\lim_{\delta_{ek}\to 0} \frac{\partial f_k}{\partial \delta_{ek}} = 2 \tag{3.73}$$

Jacobian Matrix Construction

The Jacobian matrix is defined as:

$$J = \begin{bmatrix} \frac{\partial u_3}{\partial \delta_{e1}} & \frac{\partial u_3}{\partial \delta_{e2}} \\ \\ \frac{\partial u_4}{\partial \delta_{e1}} & \frac{\partial u_4}{\partial \delta_{e2}} \end{bmatrix} \tag{3.74}$$

At the equilibrium point $(\delta_{e1}, \delta_{e2}) = (0,0)$, the Jacobian becomes:

$$J = \begin{bmatrix} S_{elevon} c_1 \frac{T_1}{C_T A} \sigma_{\alpha_1} & S_{elevon} c_1 \frac{T_2}{C_T A} \sigma_{\alpha_2} \\ \\ S_{elevon} c_2 \frac{T_1}{C_T A} \sigma_{\alpha_1} & -S_{elevon} c_2 \frac{T_2}{C_T A} \sigma_{\alpha_2} \end{bmatrix} \tag{3.75}$$

Linearized System Representation

The complete linearized system in matrix form is:

$$\begin{bmatrix} \Delta u_3 \\ \\ \Delta u_4 \end{bmatrix} = J \begin{bmatrix} \Delta \delta_{e1} \\ \\ \Delta \delta_{e2} \end{bmatrix} \tag{3.76}$$

The linearized equations in expanded form are:

$$\Delta u_3 = S_{elevon} c_1 \left(\frac{T_1}{C_T A} \sigma_{\alpha_1} \Delta \delta_{e1} + \frac{T_2}{C_T A} \sigma_{\alpha_2} \Delta \delta_{e2} \right) \tag{3.77}$$

$$\Delta u_4 = S_{elevon} c_2 \left(\frac{T_1}{C_T A} \sigma_{\alpha_1} \Delta \delta_{e1} - \frac{T_2}{C_T A} \sigma_{\alpha_2} \Delta \delta_{e2} \right) \tag{3.78}$$

The system variables and parameters are defined as:

- State Variables:

$$\Delta \delta_{e1} = \delta_{e1} - \delta_{e1,0}, \quad \Delta \delta_{e2} = \delta_{e2} - \delta_{e2,0} \tag{3.79}$$

  where $\delta_{e1,0} = \delta_{e2,0} = 0$ are the equilibrium points

  Key considerations for the linearization:

  1. Validity Region:

$$\|\Delta \delta_{ek}\| \ll 1, \quad k = 1,2 \tag{3.80}$$

  2. System Assumptions:
     * Parameters remain constant during operation
     * Small angle approximations are valid near equilibrium
     * Sign function discontinuity is handled via limits

  System Properties

  The linearized system exhibits the following properties:

  * Coupling between control surfaces through thrust parameters

* Symmetric structure in control coefficients
* Linear behavior near equilibrium points

Control Design Considerations

For controller design:

* Linear control techniques are valid near equilibrium
* The system is locally controllable if:

$$\det(J) = -S_{elevon}^2 c_1 c_2 \frac{T_1 T_2 \sigma_{\alpha_1} \sigma_{\alpha_2}}{(C_T A)^2} \neq 0 \tag{3.81}$$

* Stability can be analyzed using eigenvalues of J

The following references have been used for the above linearization method: (Slotine and Li, 1991), (Khalil, 2002), (Isidori, 1995).

The following points must be taken into account in the above linearization process:

* The limit analysis is crucial for handling the sign function discontinuity
* The existence of equal one-sided limits ensures valid linearization
* The method provides rigorous justification for linear control design
* Local behavior near equilibrium is well-approximated

and also these Key Points:

1. The sign function changes value at zero, but the rest of the terms compensate
2. The cubic term ($\sin^3$) approaches zero faster than the linear term
3. Both limits equal 2, which means:
   * The derivative exists at $\delta_{ek} = 0$
   * The linearization is well-defined
   * The system behavior is continuous at the equilibrium point

This explains why your linearized system is well-behaved at the equilibrium point, despite the presence of the sign function in the original nonlinear system. By the linearization process above, we can easily obtain the actuators' commands by inverting these relationships from the commands or control inputs ($u_1 ... u_4$), which are actually the equations that make up the block "*Conversion thrusts and deflections*" in the block diagram Figure 3.12.

$$u_1 = T_1 + T_2$$
$$u_2 = l\,(T_1 - T_2)$$
$$u_3 = AeroCo.c_1\,(T_1\,(\delta_{e_1}) + T_2\,(\delta_{e2}))$$
$$u_4 = AeroCo.c_2\,(T_1\,(\delta_{e1})) - T_2\,(\delta_{e2}))$$

Which $AreoCo$ is :

$$AeroCo = \frac{1}{2} \cdot \frac{S_{elevon}}{C_T \cdot A} \tag{3.82}$$

From the first equation:

$$u_1 = T_1 + T_2 \Longrightarrow T_1 = u_1 - T_2$$

Substitute this into the second equation:

$$u_2 = l\,(T_1 - T_2) = l\,((u_1 - T_2) - T_2) = l\,(u_1 - 2T_2)$$
$$\Longrightarrow T_2 = \frac{lu_1 - u_2}{2l}$$

Now, substitute (T_2) back into the expression for (T_1):

$$T_1 = u_1 - T_2 = u_1 - \frac{lu_1 - u_2}{2l} = \frac{lu_1 + u_2}{2l}$$

So, we now have:

$$T_1 = \frac{lu_1 + u_2}{2l}, \quad T_2 = \frac{lu_1 - u_2}{2l}$$

let's substitute the values of (T_1) and (T_2) derived earlier: Recall:

$$T_1 = \frac{lu_1 + u_2}{2l}, \quad T_2 = \frac{lu_1 - u_2}{2l}$$

Substitute these into the expressions for ($T_1\delta_{e_1}$) and ($T_2\delta_{e_2}$).
For ($AeroCo.\delta_{e_1}$):

$$AeroCo.\delta_{e_1} = \frac{\frac{\frac{u_3}{c_1} + \frac{u_4}{c_2}}{2}}{T_1} = \frac{\frac{\frac{u_3}{c_1} + \frac{u_4}{c_2}}{2}}{\frac{lu_1 + u_2}{2l}} = \frac{l\left(\frac{u_3}{c_1} + \frac{u_4}{c_2}\right)}{lu_1 + u_2}$$

For ($AeroCo.\delta_{e_2}$):

$$AeroCo.\delta_{e_2} = \frac{\frac{\frac{u_3}{c_1} - \frac{u_4}{c_2}}{2}}{T_2} = \frac{\frac{\frac{u_3}{c_1} - \frac{u_4}{c_2}}{2}}{\frac{lu_1 - u_2}{2l}} = \frac{l\left(\frac{u_3}{c_1} - \frac{u_4}{c_2}\right)}{lu_1 - u_2}$$

$$\delta e_1 = \left(\frac{l\left(\frac{u_3}{c_1} + \frac{u_4}{c_2}\right)}{AeroCo(lu_1 + u_2)}\right) \tag{3.83}$$

$$\delta e_2 = \left(\frac{l\left(\frac{u_3}{c_1} - \frac{u_4}{c_2}\right)}{AeroCo(lu_1 + u_2)}\right) \tag{3.84}$$

3.6.3 Calculating the Thrust Coefficient (b)

1. Convert the thrust of each motor (T_1) to Newtons:

$$800 \text{ g} = 0.8 \text{ kg}$$

$$\mathrm{T}_1 = 0.8 \times 9.81 = 7.848 \text{ N}$$

2. Calculate ω_1 (angular velocity at max):

$$\cdot \text{PM} = 14.8 \text{ V} \times 920\text{KV} = 13,616 \text{ RPM}$$
$$\cdot \quad \omega_1 = 2\pi \times (13,616/60) = 1,425.6 \text{ rad/s}$$

where PM stands for Power Motor - it refers to the nominal or maximum rotational speed of a brushless motor at a given voltage.
3. Use formula $\mathrm{T}_1 = \mathrm{b} \times \omega_1^2$ to solve for b :

$$b = 3.862 \times 10^{-6}$$

3.6.4 Motor and Propeller Total Moment of Inertia Calculation(J_r)

1. Motor Moment of Inertia: We can approximate a brushless motor as a cylinder. The moment of inertia for a cylinder is:

$$J_{motor} = \frac{1}{2} \times m \times r^2 \tag{3.85}$$

Given: Motor diameter = 22 mm (0.022 m), motor mass = 54 g (0.054 kg), and radius = 11 mm (0.011 m).

$$J_{motor} = \frac{1}{2} \times 0.054 \times (0.011)^2 \tag{3.86}$$
$$J_{motor} = 3.267 \times 10^{-6} \text{ kg} \cdot \text{m}^2 \tag{3.87}$$

2. Propeller Moment of Inertia: The 9545 indicates: Given: Diameter = 9.5 inches (0.2413 m), pitch = 4.5 inches, and approximate mass = 12 g (0.012 kg).

Using the thin rod approximation for the propeller:

$$J_{blade} = \frac{1}{3} \times m \times (L/2)^2 = \frac{1}{12} \times m \times L^2 \tag{3.88}$$

Where L is the propeller diameter

$$J_{blade} = \frac{1}{12} \times 0.012 \times (0.2413)^2 \tag{3.89}$$

$$J_{blade} = 5.828 \times 10^{-5}\ \text{kg} \cdot \text{m}^2 \tag{3.90}$$

$$J_{prop} = 2 \times J_{blade} = 1.1656 \times 10^{-4}\ \text{kg} \cdot \text{m}^2 \tag{3.91}$$

$$\tag{3.92}$$

3. Total Moment of Inertia

$$J_r = J_{motor} + J_{prop} \tag{3.93}$$

$$J_r = 3.267 \times 10^{-6} + 1.1656 \times 10^{-4} \tag{3.94}$$

$$J_r = 1.19827 \times 10^{-4}\ \text{kg} \cdot \text{m}^2 \tag{3.95}$$

3.7 Simulation results

The simulation demonstrates the performance of a sliding mode control system for a VTOL drone. The results show successful stabilization and position control, as illustrated in Figures 3.13 and 3.14.

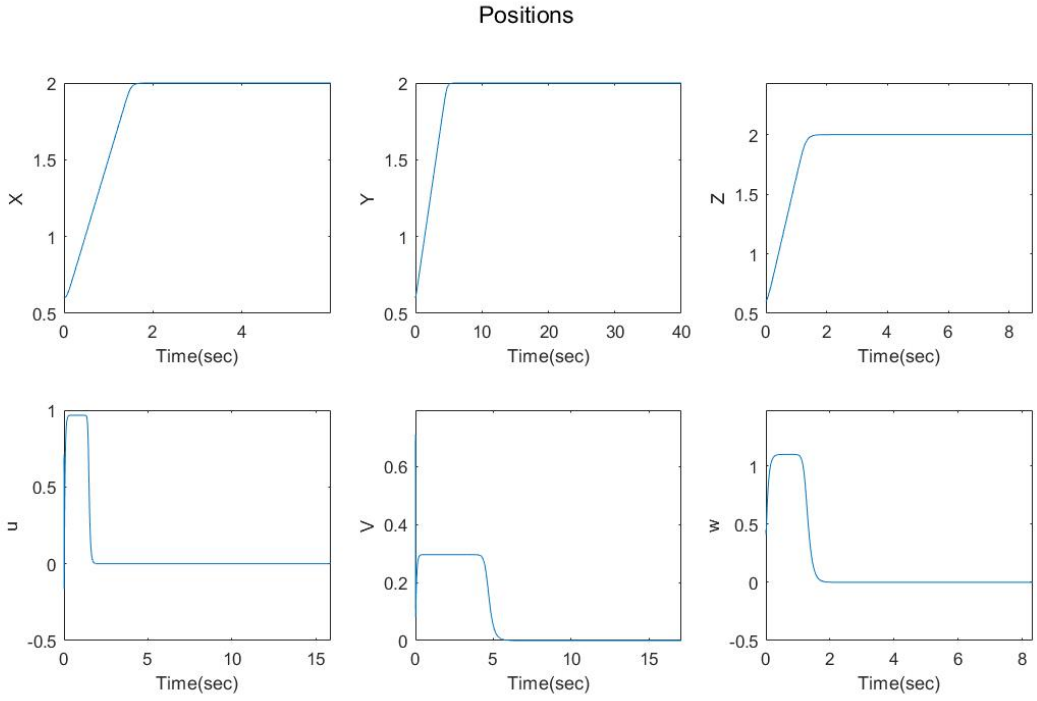


Figure 3.13: Position Control Results: Shows positions (X, Y, Z) and their corresponding velocities (u, v, w). The positions stabilize at the desired 2-meter setpoint, while velocities show proper convergence to steady-state values.

The position control results demonstrate very good performance characteristics. The X and Y positions successfully stabilize at the desired 2-meter setpoint, while the Z position maintains consistent stability at 2 meters. The velocity components (u, v, w) show well-behaved initial transients before settling to their steady-state values.

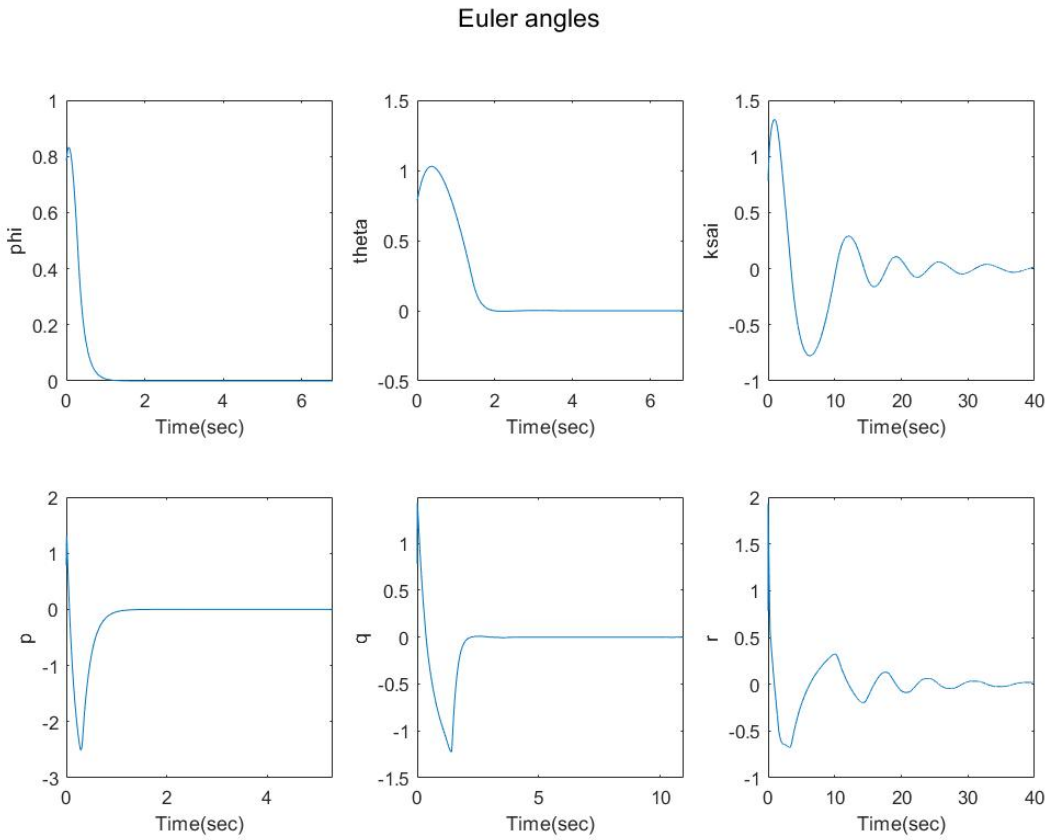


Figure 3.14: Attitude Control Results: Displays Euler angles (ϕ, θ, ψ) and angular rates (p, q, r), showing convergence to desired orientations with good damping characteristics.

The attitude control results reveal robust performance in maintaining vehicle orientation. The Euler angles (ϕ, θ, ψ) consistently converge to near-zero values, indicating precise attitude control. The angular rates (p, q, r) demonstrate well-damped responses with minimal oscillation, and the system maintains excellent rotational dynamics stability throughout the simulation period. The control system effectively manages the coupling between translational (X, Y, Z) and rotational (ϕ, θ, ψ) dynamics.

Overall, the sliding mode controller demonstrates robust performance in achieving both position holding and attitude stabilization, with acceptable transient responses and steady-state accuracy. The simulation validates the effectiveness of the chosen control parameters and sliding surface design. The results show that the controller successfully maintains the desired position $(X,Y,Z) = (2,2,2)$ meters while keeping the attitude angles (ϕ,θ,ψ) near zero. As this work progresses, we will surely need to

update the dynamic equations to include the thread or tether tension dynamic equation and, if required, modify the controller, which will also alter the dynamic equations, both rotational and linear dynamic equations, something we were unable to do at the moment because of time constraints. When a sliding mode controller performs well in simulations, it validates many crucial aspects of the drone's overall design. This comprehensive validation includes the basic drone architecture, the sizing of its wings and control surfaces(elevons), the speed capabilities of its servos, the thrust output specifications, the strategic placement of propulsion systems, and the appropriate selection of motors and propulsion components.

3.8 Concluding remarks

This chapter has presented a detailed analysis and implementation of sliding-mode control for a VTOL aircraft system. The development process began with a thorough derivation of the dynamic model, incorporating crucial physical parameters such as wing span, chord length, control surface characteristics, and mass distribution. Through advanced modeling using SolidWorks, we successfully determined the moments of inertia parameters and aerodynamic coefficients essential for accurate system representation.

The dynamic analysis revealed inherent nonlinear characteristics in the state equations, leading to the selection of sliding-mode control as an optimal strategy. This choice proved particularly advantageous due to SMC's robust handling of system uncertainties and disturbances.

We developed a sophisticated dynamic model incorporating both fixed-wing and VTOL characteristics, calculated the inertia matrix through detailed 3D modeling in both FreeCAD and SolidWorks, successfully implemented a sliding-mode controller that effectively manages the complex coupled dynamics, and produced comprehensive simulation results demonstrating excellent position and attitude control.

The simulation results validated the effectiveness of our approach, showing stable convergence to desired positions (X, Y, Z = 2m) with minimal steady-state error. We achieved precise attitude control with well-damped responses in Euler angles, demonstrated robust performance in the presence of coupling effects between translational and rotational dynamics, and effectively managed the transition between different flight modes.

Several notable findings emerged from the control system implementation. The non-diagonal inertia matrix, particularly the non-zero J_{xz} terms, significantly influenced system behavior. The incorporation of aerodynamic effects through detailed lift coefficient modeling enhanced control accuracy, while the sliding surface design successfully accommodated both position and attitude control objectives.

Looking ahead, our research points to several important areas for future development. The integration of tether dynamics into the control system represents a crucial next step. Additionally, further optimization of control parameters for different flight conditions and the implementation of adaptive elements to enhance robustness will be valuable extensions of this work. Experimental validation of the proposed control scheme will provide essential real-world verification of our theoretical and simulation results.

In conclusion, the sliding-mode control strategy has demonstrated its effectiveness for this complex VTOL system. The successful simulation results validate not only the control approach but also the fundamental design choices in the aircraft's physical configuration, including wing sizing, control surface design, actuator specifications, and propulsion system layout. This comprehensive validation provides a solid foundation for future experimental implementation and system refinement.

Chapter 4

Robust Control

In this section, we introduce the overall control architecture of the autopilot system developed for the VTOL drone. A series of robust controllers are designed to address the specific dynamics and operational requirements of the aircraft. To facilitate control design and analysis, the drone's motion is decomposed along two principal coordinate planes commonly referenced in aerospace engineering: the lateral plane (YZ-plane), aligned with the wingspan and vertical axis, and the longitudinal plane (XZ-plane), aligned with the fuselage centerline and vertical reference.

4.1 Lateral Plan Control Architecture

To control lateral position, feedback from the tether is utilized. Specifically, the error in the tether angle—measured by a potentiometer relative to the nominal 90-degree alignment with the drone—is minimized through a rolling maneuver, as illustrated in Figures 4.1 and 4.2. These figures also depict the drone's coordinate axes.

Maintaining tension in the tether is essential for accurate potentiometer readings; if the tether slackens, the sensor becomes ineffective. This issue is addressed by applying a 5-degree pitch trim in the direction opposite to the tether, as explained in detail in Section 4.4.1. The drone naturally tends to drift opposite to the tether's direction, but the tether constrains this motion.

Altitude control is achieved using a LiDAR rangefinder, which maintains the initial altitude upon autopilot activation, as described in Section 4.4.2. Since similar works have already addressed pitch and altitude control loops in VTOL systems (Verling et al., 2016), these aspects are not elaborated further in this thesis.

Along the body reference axis aligned with the wings, three control loops are implemented: the roll control loop, the yaw control loop, and the lateral position control loop along the Y-axis, governed by the tether angle. In the lateral axis, the yaw loop maintains the initial heading, using the IMU sensor to close the loop with the initial yaw value. The primary lateral position control loop is structured as a nested loop,

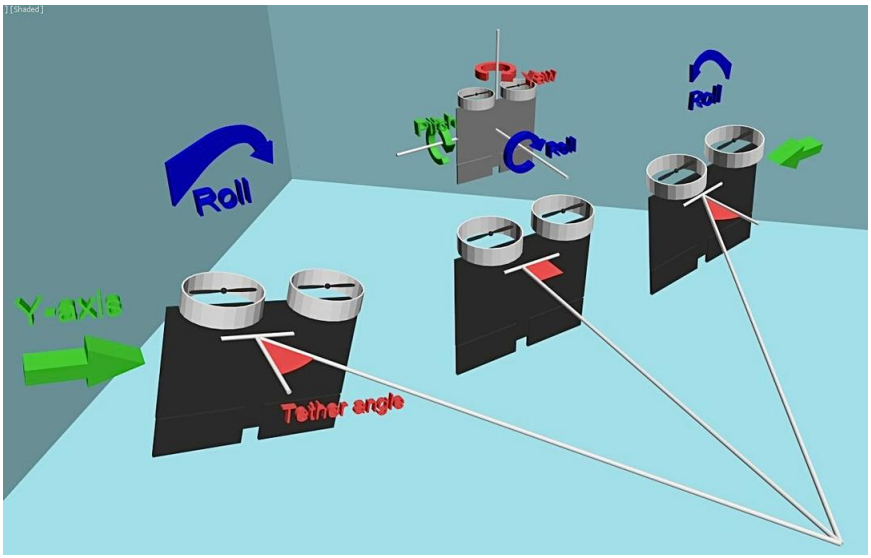


Figure 4.1: Rolling maneuver controls tether angle position.

where the lateral position error is regulated through tether angle feedback. This loop, in turn, generates the desired value for the inner loop, which is the roll control loop. This inner loop is closed using the roll angle obtained from the IMU-AHRS sensor. The outputs of the roll, pitch, yaw, and altitude control loops—interpreted as actuator commands—are transmitted to the actuator block. Within this block, the control logic is straightforward: when both motors operate in synchrony, they generate thrust that alters the drone's altitude. Conversely, differential motor speeds induce a rolling motion. Similarly, simultaneous deflection of both elevons produces pitch, while opposing deflections result in yaw. These control effects are described by the following expressions:

$$\phi_c = T_1 - T_2 \tag{4.1}$$

$$h_c = T_1 + T_2 \tag{4.2}$$

$$\theta_c = \delta_{ELEV_1} + \delta_{ELEV_2} \tag{4.3}$$

$$\psi_c = \delta_{ELEV_1} - \delta_{ELEV_2} \tag{4.4}$$

where ϕ_c, h_c, θ_c, and ψ_c represent the Roll-Command, Altitude-Command(Throttle), Pitch-Command, and Yaw-Command respectively. δ_{ELEV_1} and δ_{ELEV_2} denote the deflection angles of elevon 1 and elevon 2, while T_1 and T_2 represent the thrust generated by motor 1 and motor 2 respectively.

Figure 4.3 presents the block diagrams in which G1, G2, and G3 denote the dynamic models of the aircraft subsystems. These models will be identified in the following section, along with the corresponding controllers for each subsystem, which will be described subsequently.

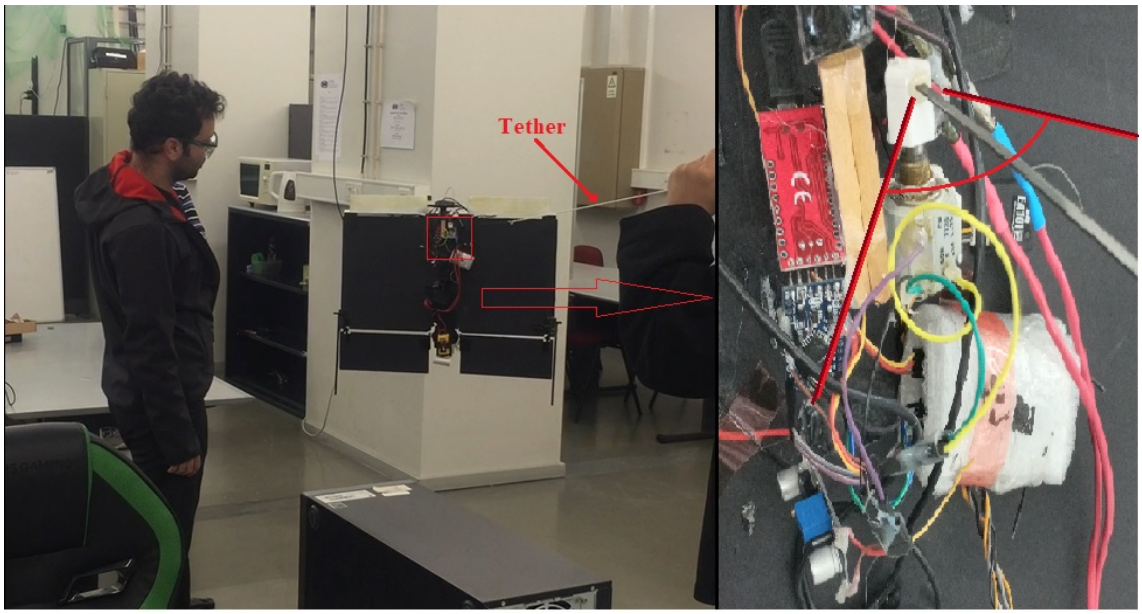


Figure 4.2: Precision industrial potentiometer integrated within avionics for thread angle measurement and positional feedback.

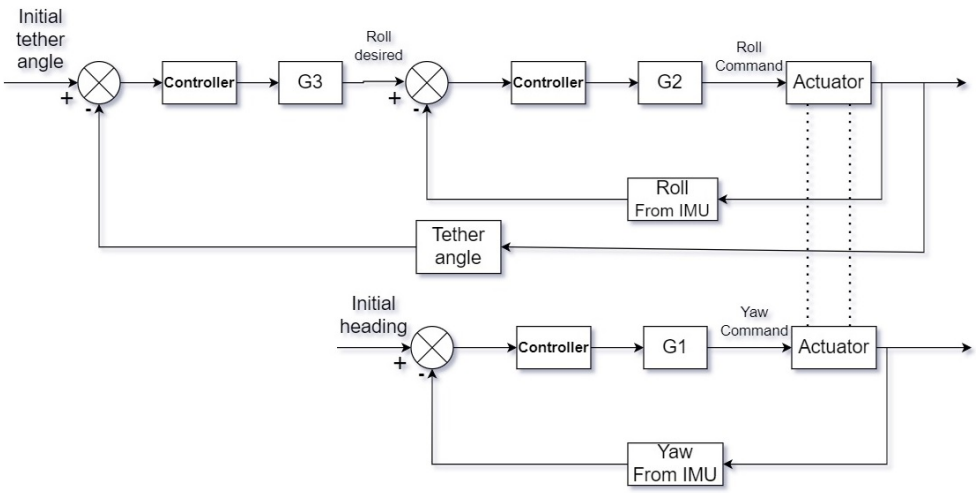


Figure 4.3: Diagram of commands and corresponding actuators.

4.2 Testing and System Identification

4.2.1 Testing Methodology

At first, the system is hung from the floor of the lab and flown in the hovering mode, and a PID controller is tuned using trial and error according to the Ziegler-Nichols

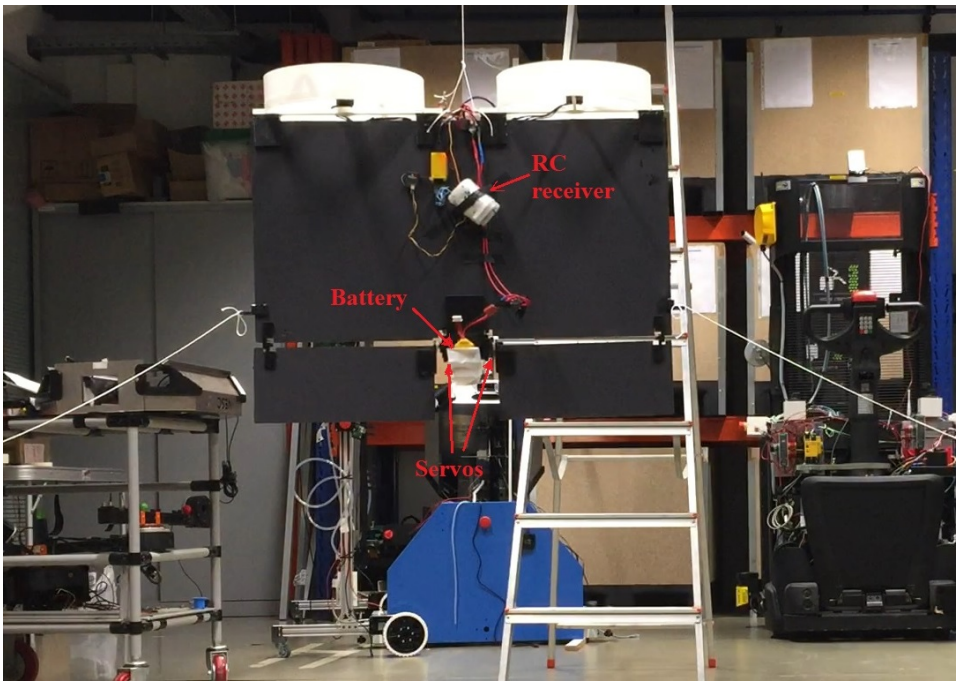


Figure 4.4: Hanging drone ready for initial test for PID controllers tuning.

method (see Appendix A.2) (Bapst et al., 2015). Subsequently, short-flight experiments are conducted in the laboratory, and data is extracted as shown in Figure 4.4-4.5. The system is then identified based on this data, and nominal and uncertain model parameters are extracted. The main controller is designed for these models. Following that, several flight tests are performed in an open space, and the controllers are finalized and tuned. Initially, GPS signals and linear velocities extracted from the GPS module are used to provide additional stability to the system through external loops as illustrated in Figure 4.6. Then, a tether feedback system is added in the indoor laboratory environment, and final tests are conducted to control lateral and longitudinal stability. The main controller utilizes a robust control approach, and its design involves a combination of sensitivity synthesis and H_2/H_∞ methods as depicted in Figure 4.7. To watch the test footage click this link: `https://www.youtube.com/watch?v=hCvIrhaIQRc&t=13s`.



Figure 4.5: Initial laboratory efforts for limited flights and controller tuning.



Figure 4.6: Open area flight test using GPS.

Table 4.1: Categories of Systems for Identification

Input	Output	Transfer Function
Yaw Command	Yaw	G_1
Roll Command	Roll	G_2
Roll	Tether Angle	G_3



Figure 4.7: Indoor flight test with tether feedback fully automatic flight.

4.2.2 System Identification

System identification is performed using real input-output data from the drone across 3 data categories, as shown in Table 4.1. The gathered flight data is partitioned into 10 pairs, and system identification is conducted separately for each pair. For each input-output pair, a nominal system is derived, and an uncertainty profile is obtained. The nominal model is selected based on the least uncertainty, particularly at low frequencies. The identification process follows the Box-Jenkins method.

In the next parts with the information of nominal model and multiplicative uncertainty profile, robust controller is designed for each system.

Yaw command-yaw system: In the first system the yaw command is input and yaw is output. The identified systems' bode diagrams are plotted in Figure 4.8. From the

data, a nominal system is obtained, which is described in (4.45).

Roll command-roll system: In the second system with roll command-roll as input-output, the identified systems' bode diagrams are plotted in Figure 4.9. Nominal system is achieved as Equation (4.6).

Roll Tether-Angle: The last input-output category is the Roll Tether-Angle. The identified systems' bode diagrams of this case are plotted in Figure 4.10 and nominal model is shown in Equation (4.7).

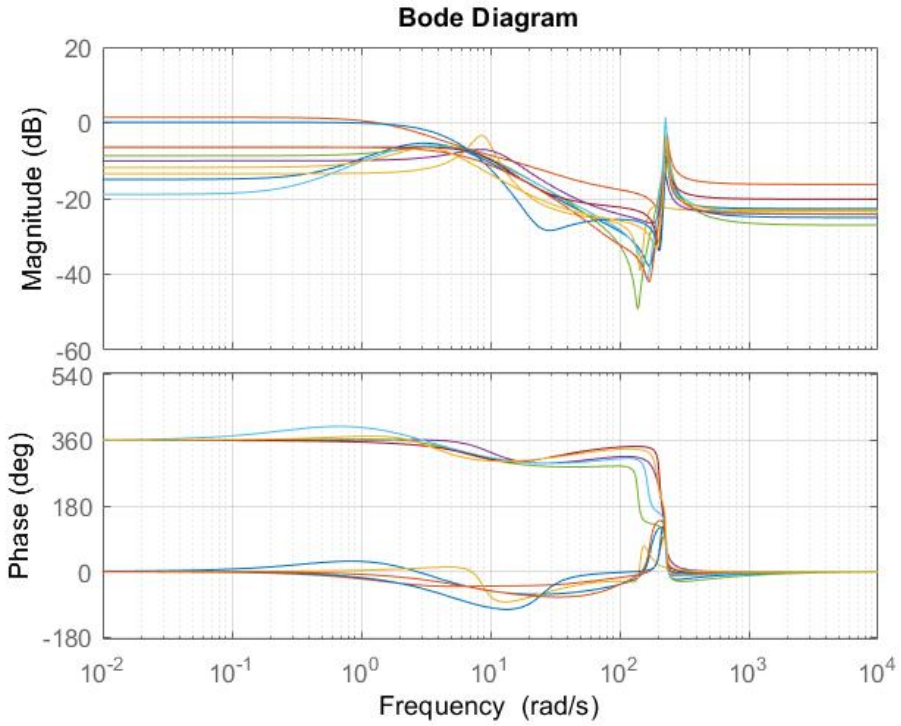


Figure 4.8: Bode diagrams showing magnitude and phase responses of ten candidates G_1 transfer functions, from which the final model is selected for system identification.

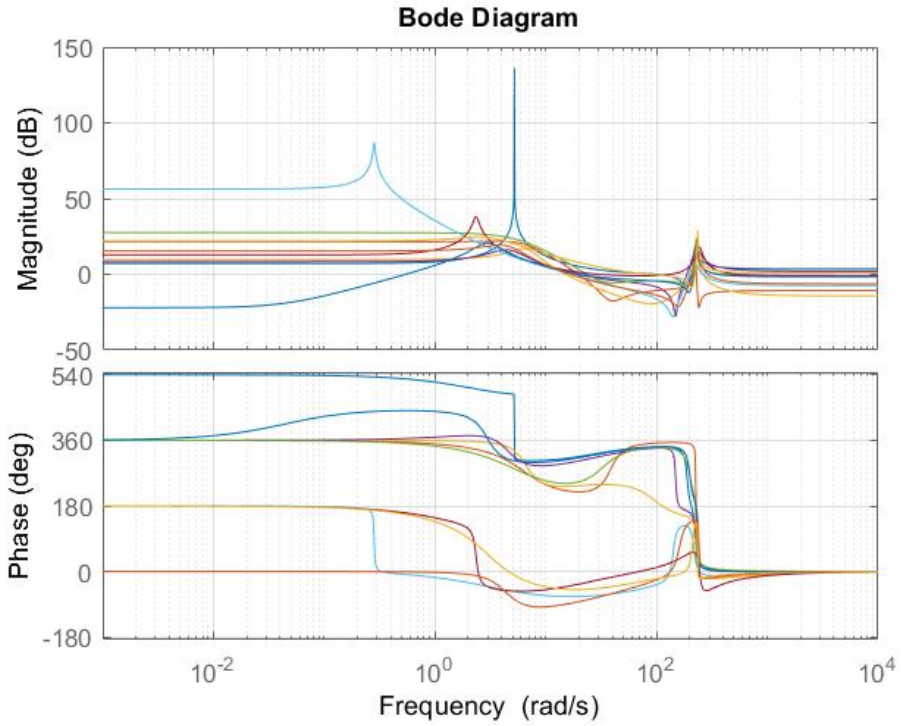


Figure 4.9: Bode diagrams showing magnitude and phase responses of ten candidates G_2 transfer functions, from which the final model is selected for system identification.

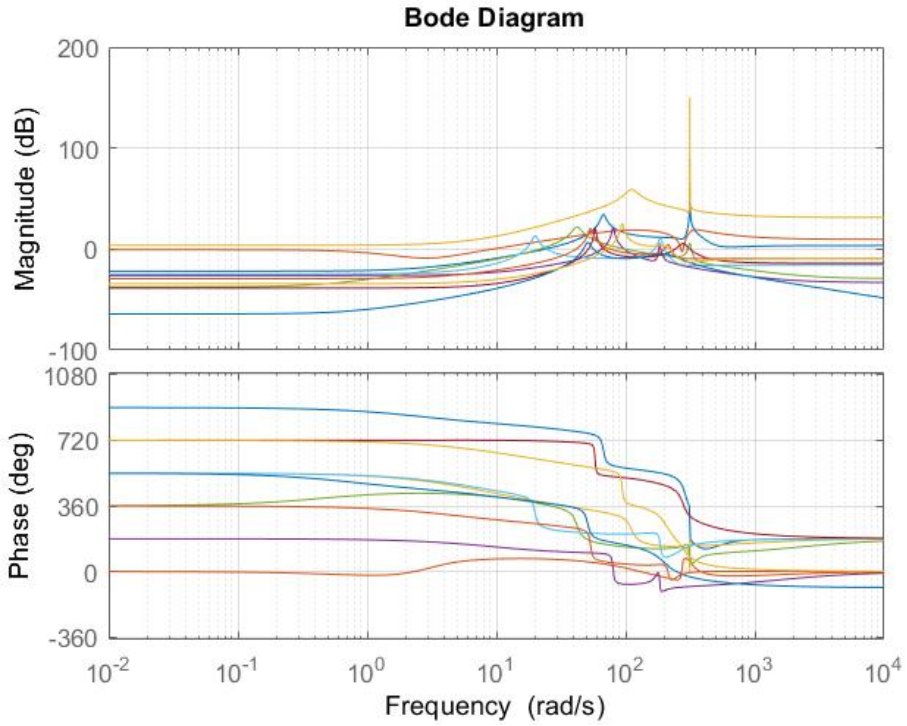


Figure 4.10: Bode diagrams showing magnitude and phase responses of ten candidates G_3 transfer functions, from which the final model is selected for system identification.

$$G_1(s) = \frac{0.07344s^4 + 2.411s^3 + 3186s^2 + 5.632 \times 10^4 s + 1.948 \times 10^6}{s^4 + 21.08s^3 + 5.289 \times 10^4 s^2 + 6.193 \times 10^5 s + 1.912 \times 10^6} \tag{4.5}$$

$$G_2(s) = \frac{0.0317s^4 + 0.2434s^3 + 1870s^2 + 2.788 \times 10^4 s + 2.719 \times 10^6}{s^4 + 9.859s^3 + 5.091 \times 10^4 s^2 + 3.822 \times 10^5 s + 2.094 \times 10^6} \tag{4.6}$$

$$G_3(s) = \frac{3.03s^5 + 2423s^4 + 2.6 \times 10^5 s^3 + 1.8 \times 10^8 s^2 + 6.08 \times 10^8 s + 1.57 \times 10^9}{s^5 + 279s^4 + 1.338 \times 10^5 s^3 + 2.06 \times 10^7 s^2 + 1.633 \times 10^9 s + 1.7 \times 10^9} \tag{4.7}$$

4.3 Robust Control Design

4.3.1 Uncertainty Profile Modeling

In the multiplicative uncertainty framework using Linear Fractional Transformation (LFT), we express the perturbed plant model p as $p = p_0(1 + W_T\Delta)$ following the same notation as presented in (Matušů et al., 2017). In each system, the uncertainty profile W is obtained so that the following holds:

$$\left| \frac{p}{p_0} - 1 \right| \leq W_T \tag{4.8}$$

where p is the set of plants, and p_0 is the nominal plant and Δ presents uncertainty itself, where $\|\Delta\|_\infty \leq 1$ as shown in Figure 4.11.

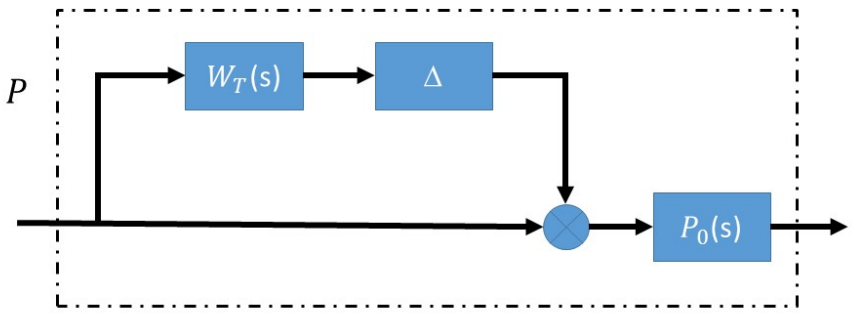


Figure 4.11: Multiplicative uncertainty LFT block.

The bode diagram of obtained profiles are plotted in Figure 4.12-4.14.

One should choose the uncertainty profile limit such that it covers the worst case of uncertainty. In each figure it is plotted in dashed blue. These limits are shown in Equation (4.9-4.11).

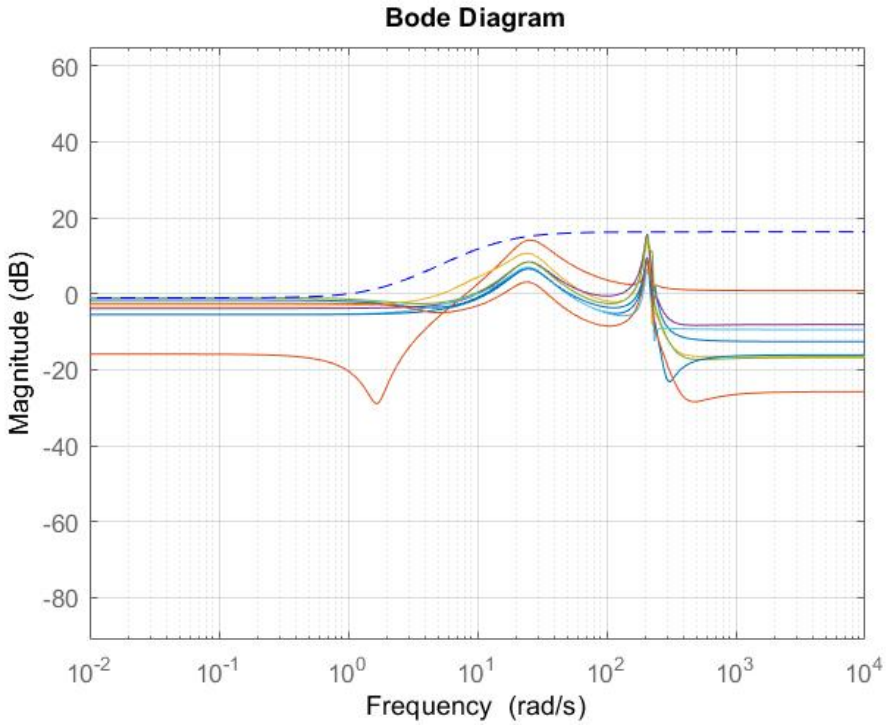


Figure 4.12: Bode diagram of uncertainty profile for system G_1.

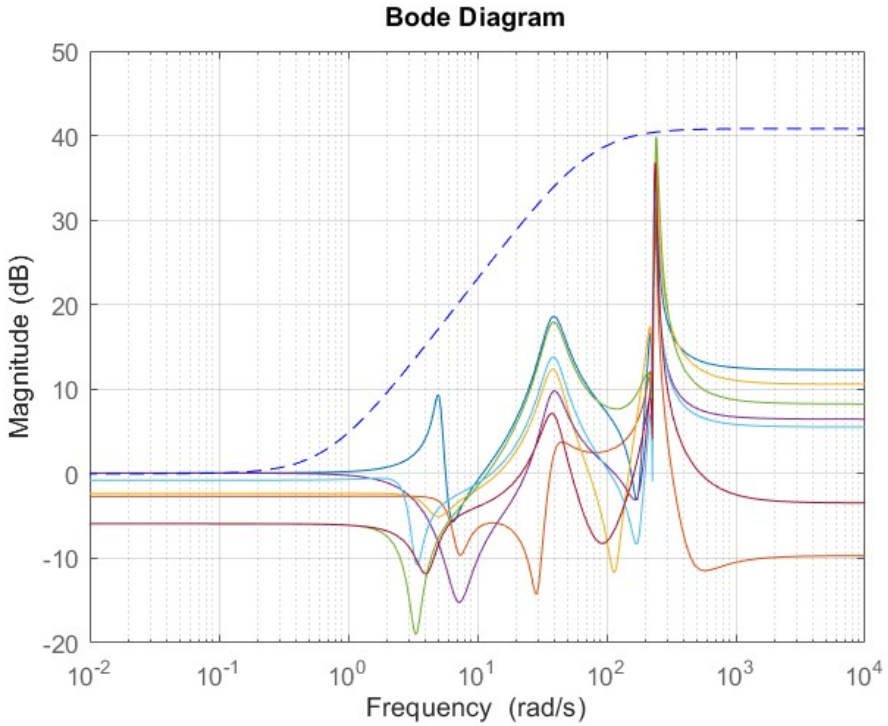


Figure 4.13: Bode diagram of uncertainty profile for system G_2.

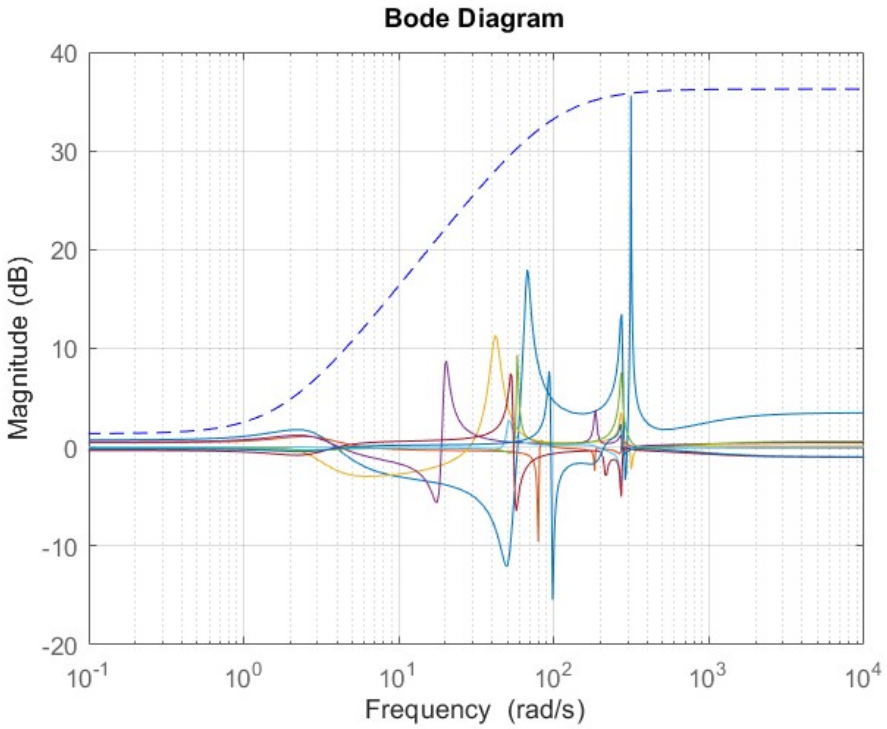


Figure 4.14: Bode diagram of uncertainty profile for system G_3.

$$W_{T_1}(s) = 6\left(\frac{s+1.9}{s+14}\right) \tag{4.9}$$

$$W_{T_2}(s) = 110\left(\frac{s+0.7}{s+77}\right), \tag{4.10}$$

$$W_{T_3}(s) = 65\left(\frac{s+1.8}{s+100}\right) \tag{4.11}$$

As illustrated in all figures the considered W_{T} cover almost all uncertainties of the system.

4.3.2 Mixed Sensitivity $\mathscr{H}_\infty$ Controller Design

In this section, a robust controller is designed for each system, considering mixed sensitivity.

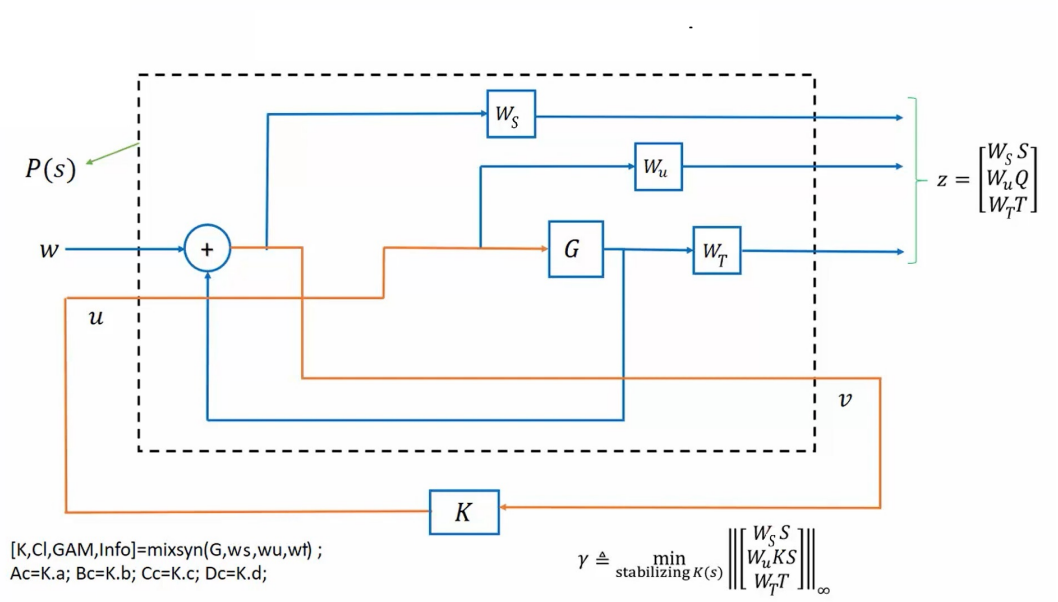


Figure 4.15: Block diagram of weights function of LFT in mixed sensitivity scenario.

Figure 4.15 illustrates the block diagram of a control system designed using $\mathscr{H}_\infty$ mixed-sensitivity optimization. The diagram outlines the full control architecture, with the main system enclosed within a dashed box. Key components include the plant G, representing the system to be controlled, and the controller K, which is to be designed. Three weighting functions W_s, W_u, and W_T are incorporated to shape the system's sensitivity, control effort, and robustness characteristics, respectively. The signal flow demonstrates how external disturbances w and control inputs u are processed through the system. These inputs are combined at a summation point and propagated through the plant and weighting functions. A feedback loop through the controller K ensures system stability and desired performance.

The design objective here is to solve a mixed-sensitivity synthesis problem, where the goal is to optimize the controller K for robust performance. This means the controller should maintain good performance even when the system faces uncertainties or disturbances. The weighting functions play a crucial role in shaping the system response and balancing different control objectives.

At the bottom of the diagram, there's a MATLAB implementation note showing how this design can be practically realized using the `mixsyn` function, which solves for the optimal controller. The state-space matrices of the resulting controller are then extracted for implementation (Taghirad et al., 2021).

The controller design criteria include achieving overshoot less than 15% and settling time less than 1 s. Consequently, the desired system is represented by $T_{id} = \frac{81}{s^2+9s+81}$ and sensitivity function becomes $S_{id} = 1 - T_{id} = \frac{s(s+9)}{s^2+9s+81}$, finally W_s for all 3 systems is defined in Equation (4.12)-(4.14).

$$W_{s1} = 0.3\frac{s^2+9s+81}{(s+0.001)(s+9)(0.001s+1)} \tag{4.12}$$

$$W_{s2} = 0.2\frac{s^2+9s+81}{(s+0.001)(s+9)(0.001s+1)} \tag{4.13}$$

$$W_{s3} = 0.2\frac{s^2+9s+81}{(s+0.001)(s+9)(0.001s+1)} \tag{4.14}$$

Assuming $W_u = 1$, in the context of mixed sensitivity problems, the goal is to adjust the transfer functions of uncertainties to achieve the following:

$$\gamma_{opt} = \min \left\| \begin{matrix} W_T T \\ W_s S \\ W_u U \end{matrix} \right\|_\infty \leq 1 \tag{4.15}$$

Graphs illustrating the design parameters for all systems are presented in Figure 4.16-4.18. In these figures, γ_{opt} is represented as sigma. It is evident that Equation (4.15) is valid across all scenarios. The corresponding Equation (4.15) values for the three systems are: $\gamma_{opt-1} = 1.12, \gamma_{opt-2} = 0.93, \gamma_{opt-3} = 1.19$.

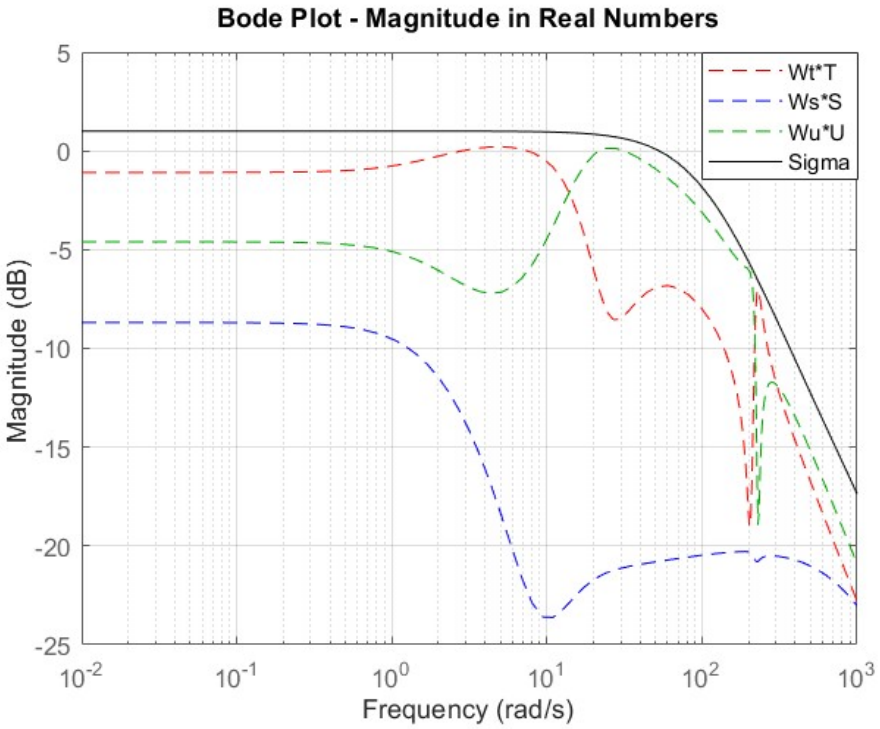


Figure 4.16: Design parameters of G_1 for $\mathscr{H}_\infty$ in mixed sensitivity scenario.

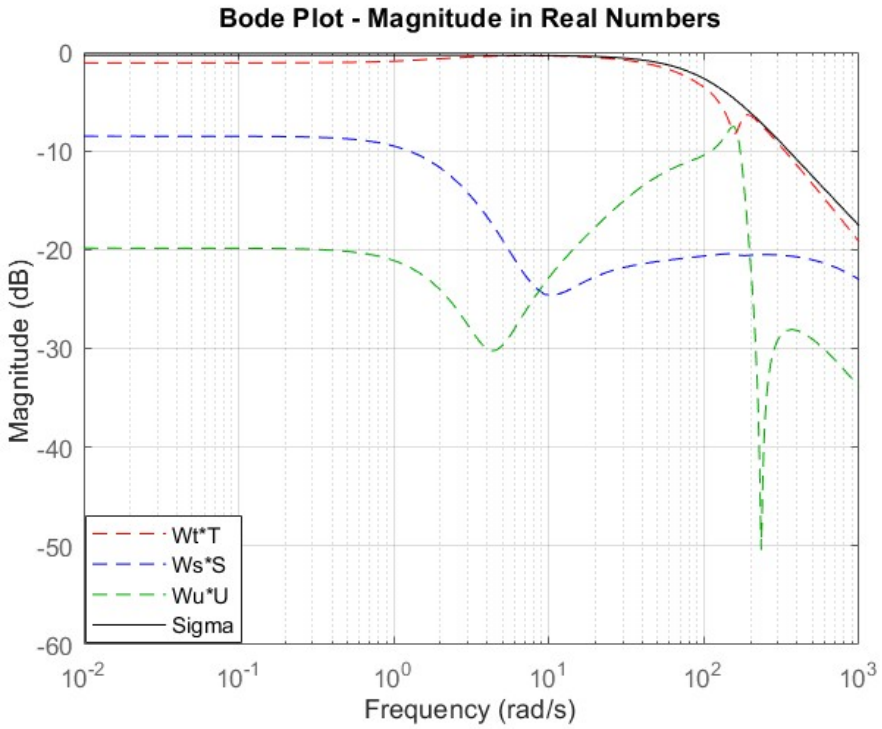


Figure 4.17: Design parameters of G_2 for $\mathscr{H}_\infty$ in mixed sensitivity scenario.

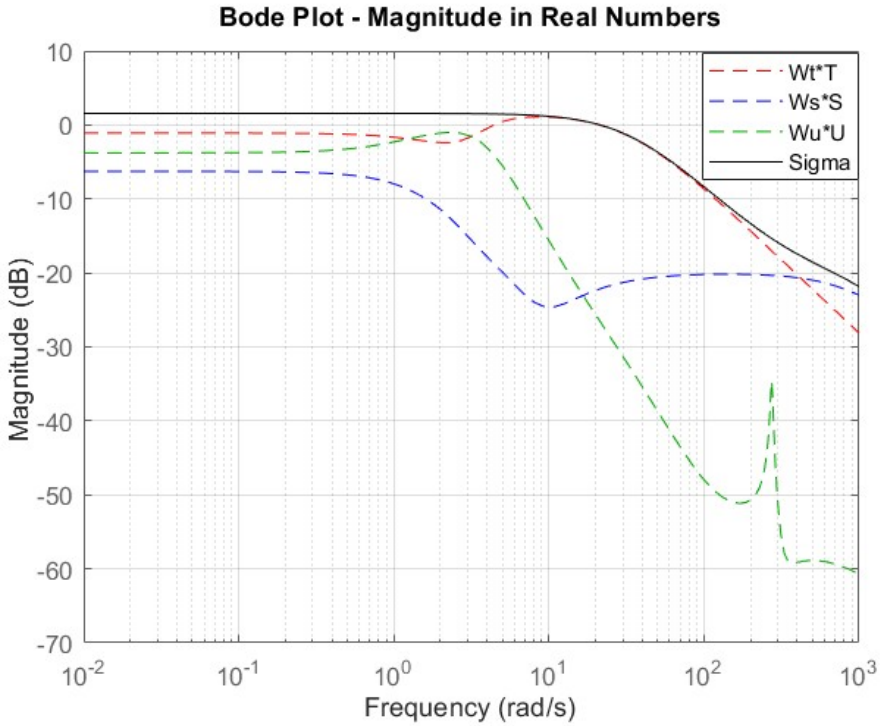


Figure 4.18: Design parameters of G_3 for $\mathscr{H}_\infty$ in mixed sensitivity scenario.

4.3.3 Controller Design with $\mathscr{H}_2/\mathscr{H}_\infty$ Method

Take into account all nominal systems G_1, G_2, G_3 and W_T, W_S and W_U, as previously derived. According to (Doyle et al., 1988), the robust controller is formulated to achieve:

$$\min_{\gamma_{opt}} \left\| \begin{array}{c} W_s S \\ W_T T \end{array} \right\|_\infty < 1 \quad , \|W_u u\|_\infty < 1 \tag{4.16}$$

Design parameters for all systems are plotted in Figure 4.19-4.21. It is clear that in all cases, inequalities (4.16) hold. All systems are robustly stable. The γ related to the second norm of all three systems are equal to 1 and γ related to the infinity norm of G_1,G_2, and G_3 are 1.0956,1.1092, and 1.1956 respectively.

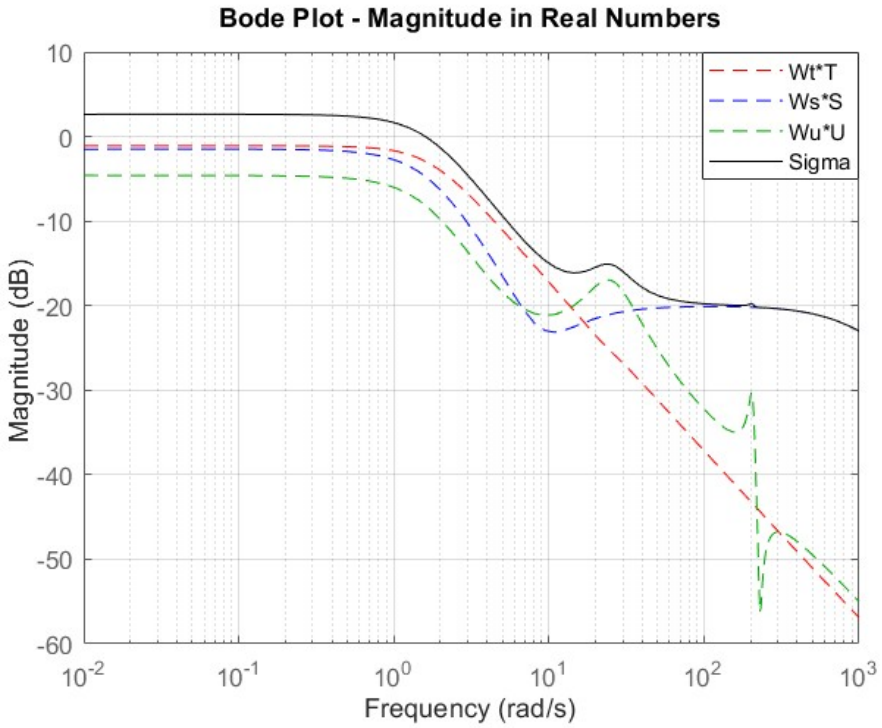


Figure 4.19: Design parameters of G_1 for $\mathscr{H}_2/\mathscr{H}_\infty$ scenario.

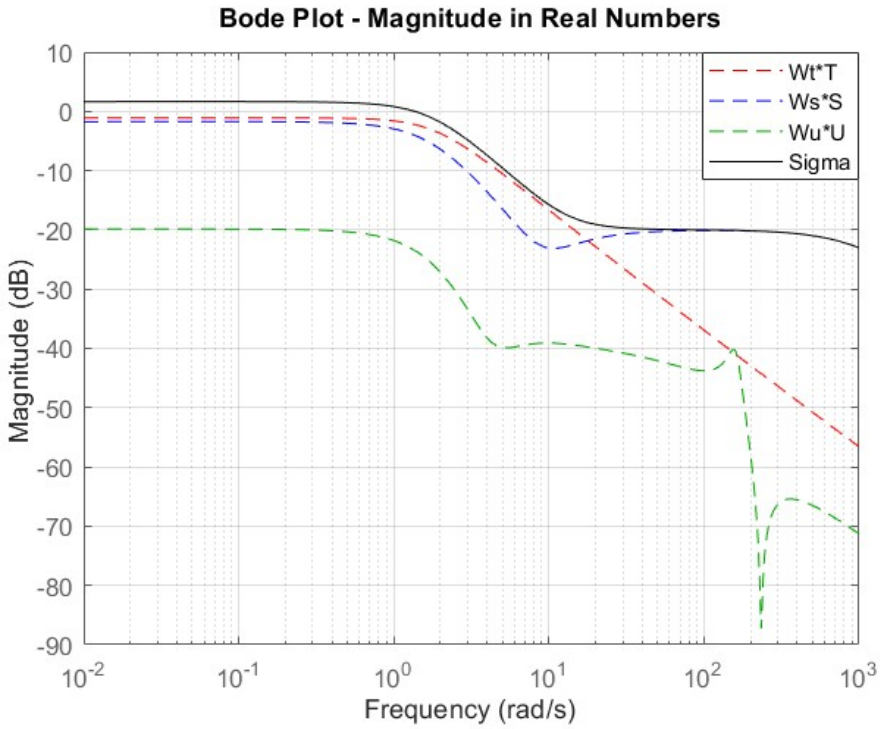


Figure 4.20: Design parameters of G_2 for $\mathscr{H}_2/\mathscr{H}_\infty$ scenario.

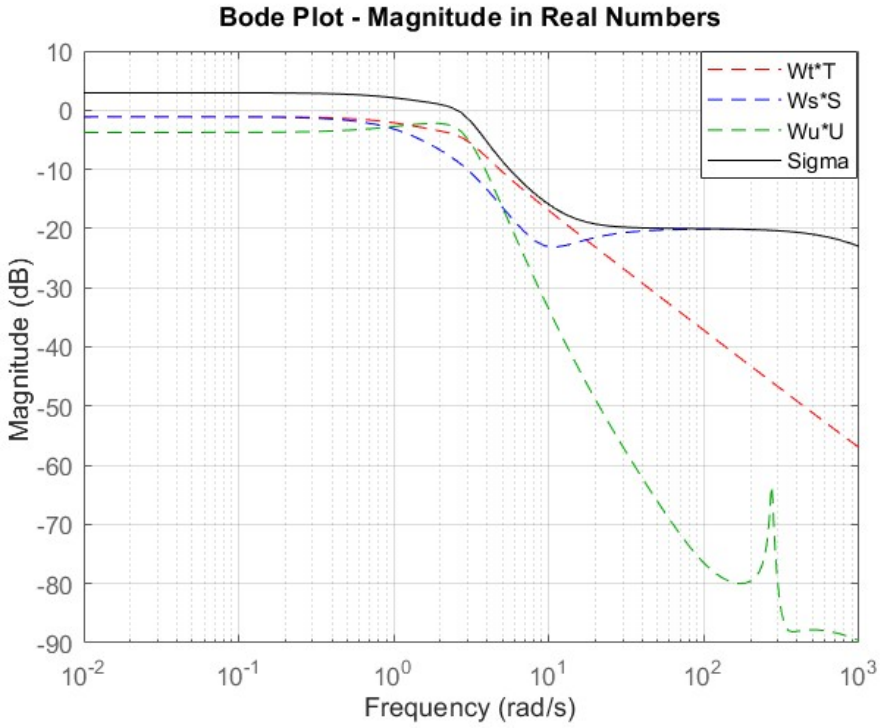


Figure 4.21: Design parameters of G_3 for $\mathscr{H}_2/\mathscr{H}_\infty$ scenario.

The values of γ are suitable for engineering work. Each of the six controllers exhibits a suitable step response without overshooting and with a settling time of approximately two second, slightly more. As an example, one of them is illustrated in Figure 4.22. The outcomes of implementation on the autopilot are depicted in Figure 4.23. As observed, the tether angle has been maintained around 0 degrees, which is our desired value. To achieve this, the drone has rolled maximally at ± 5 degrees. This rolling action is necessary due to the disturbance caused by the ground effect when flying at low altitudes, and the controller has effectively mitigated this disturbance. Additionally, due to the coupling between roll and yaw angles, there is a disturbance in the yaw angle. Nevertheless, this loop performed well and preserved the initial yaw angle or heading, which was proportionally set to zero.

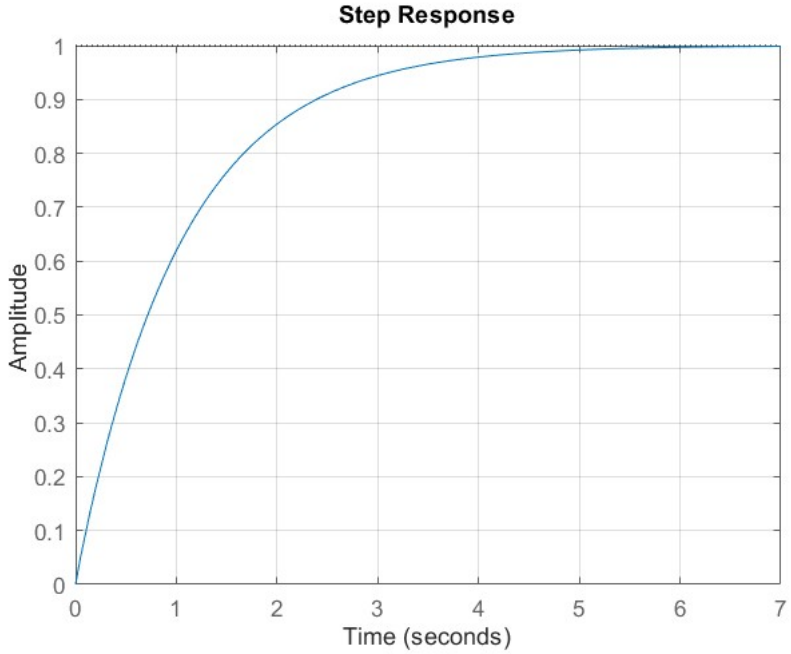


Figure 4.22: Step response of G_2 for $\mathscr{H}_2/\mathscr{H}_\infty$ controller.

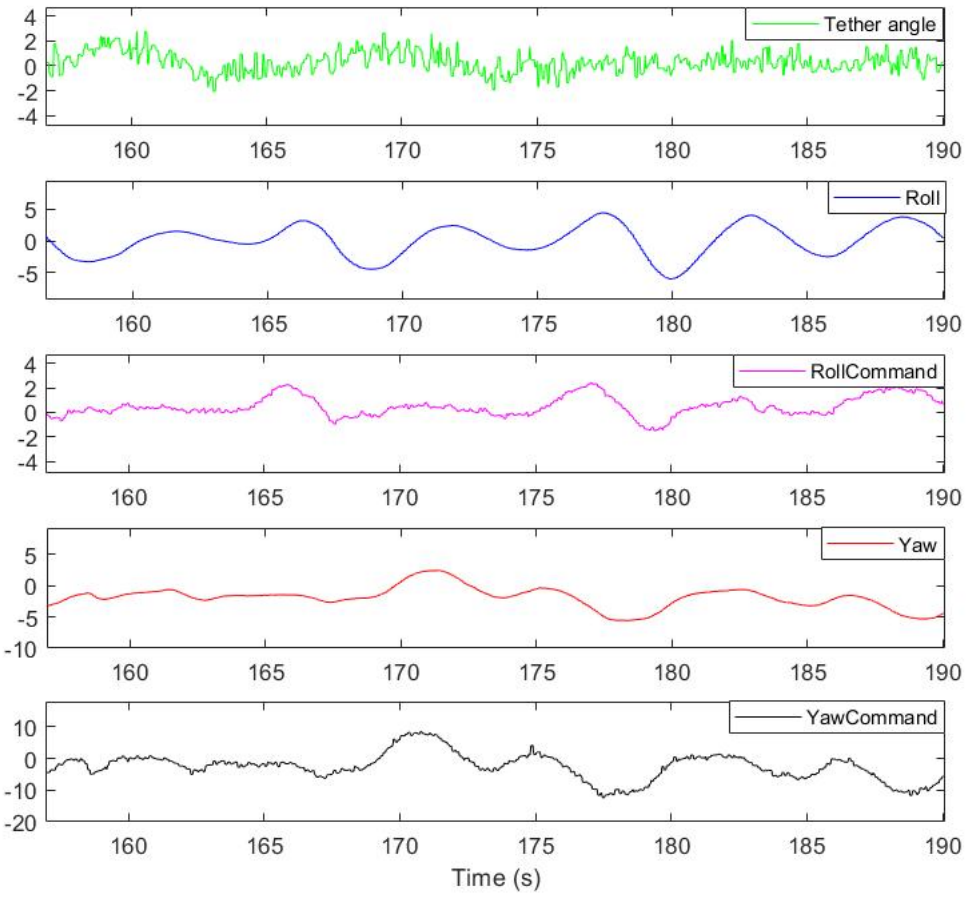


Figure 4.23: Controller implementation on a real autopilot.

4.3.4 Controller Design with μ synthesis Method

4.3.4.1 Controller Design Based on Structured Singular Value

The work in (Packard and Doyle, 1993) is foundational for the definition of the structured singular value (μ), while (Zhou et al., 1996) offers an in-depth treatment of both the theoretical framework and practical implementation. Additionally, (Dullerud and Paganini, 2000) provides a more modern perspective on Linear Fractional Transformations (LFTs) and their connection to μ-synthesis. the structured singular value for matrix N with respect to block Δ is defined as:

$$\mu_{\Delta}(N) = \frac{1}{\min\{\sigma(\Delta), \det(I - N\Delta) = 0\}} \tag{4.17}$$

Then, the matrix N for a dynamic system with standard structure, as shown in Figure 4.24, is developed and the corresponding relationship for matrix transformation is

extracted using LFT transformation as follows:

$$N = T_{zu} = F_l(M,K) = M_{11} + M_{12}K(I - M_{22}K)^{-1}M_{21} \tag{4.18}$$

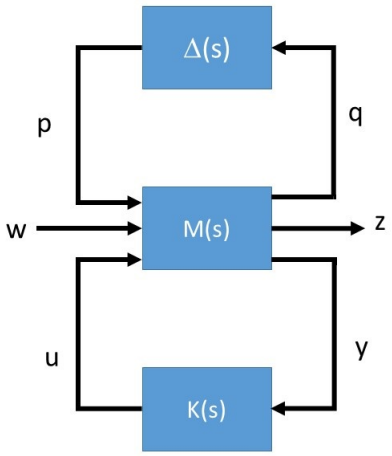


Figure 4.24: $\Delta - M - K$ Standard Representation.

Considering the separation of control system and uncertainty system, Figure 4.24 shows the LFT transformation with details again for input set (p,u,w) and output set (q,y,z).

Using an important result obtained from small gain theorems (Zhou et al., 1996)(Chapter 9), it has been shown that to achieve robust performance, it is sufficient that $\mu_\Delta(N) < 1$. Also, with the assumption of internal stability condition N, it was determined that $\mu_\Delta(N) < 1$ is a necessary condition in which Δ block represents the structural combination of system uncertainty block and performance uncertainty block.

In practice, since the exact calculation of structured singular value is not analytically possible, the following relationship is used:

$$\rho(N) \leq \mu(N) \leq \bar{\sigma}(N) \tag{4.19}$$

Also, considering this inequality, the following relationship is used to calculate singular values in practical applications:

$$\mu_\Delta(N) \leq \inf_D \|\bar{\sigma}(DND^{-1})\| \tag{4.20}$$

where matrix D has a structure that follows the set of uncertainty parameters Δ, according to the definition of all components:

$$\Delta = diag[\delta_1 I_{r1}, \cdots, \delta_S I_{rS}, \Delta_1, \cdots, \Delta_F] \quad \delta_i \in C, \Delta_i \in C^{m_i \times m_i} \tag{4.21}$$

4.3.4.2 μ Synthesis Problem Definition

Considering the system structure shown in Figure 4.24, for designing a robust optimal controller using upper LFT transformation, the synthesis problem with uncertainty block is expressed as:

$$\left\| F_u(M,\Delta), \Delta \in \Delta, \max_{\omega} \|\Delta(\omega)\| \leq 1 \right\| \tag{4.22}$$

where Δ is a subset of uncertainty block parameters that are ultimately smaller than one. The design goal is to find a stabilizing controller K such that the closed-loop system transfer function is minimized. This goal can be achieved using lower LFT transformation with the above set and can be rewritten as:

$$\min_{K} \|\|F_l(M,\Delta), K\|\| < 1 \tag{4.23}$$

This expression is equivalent to saying that with norm $\|\|F_{zw}\|\|_\infty < 1$, if the given controller is used, the above condition can be calculated using structured singular value:

$$\max_{\omega} \mu_\Delta(F_l(M,K)(j\omega)) < 1 \tag{4.24}$$

Thus, the goal of the designed controller in μ synthesis is to determine a stabilizing controller that minimizes $\mu_\Delta(F_l(M,K))$ on $F_l(M,K)$. This minimization problem can be expressed as:

$$\min_{K}[\max_{\omega} \mu_\Delta(F_l(M,K)(j\omega))] < 1 \tag{4.25}$$

Note that in simplifying equation (4.23), you can use:

$$F_l(F_u(M,\Delta),K) = F_u(F_l(M,K),\Delta) \tag{4.26}$$

4.3.4.3 Solution Approach for μ Synthesis

If an appropriate controller is designed, it becomes possible to evaluate the upper bound of the structured singular value (μ) across a range of frequencies. Since μ varies with frequency, minimizing its peak value is a key objective in robust control design. To ensure satisfactory performance and robustness, it is essential to examine the behavior of μ over the entire frequency spectrum of interest.

The structured singular value evaluation at different frequencies can be performed in two ways. The first method is to use the approach discussed in the previous chapter:

after designing optimal controllers, one can directly obtain the frequency distribution of this function. This work is traditionally done by control designers and engineering experience has shown that the direct effect on increasing system robustness and reducing the structured singular value can be experienced naturally.

In another approach, one can use an innovation that takes advantage of matrix D in Equation (4.20) to calculate Equation (4.24), which provides Equation (4.20) that can systematically obtain the frequency distribution of structured singular value. The process can be performed by the controller H_∞ which introduces a regression relative to its optimization. It's possible to evaluate the frequency distribution of structured singular value by transfer function D. After constructing and evaluating the results, this can be achieved using Equation (4.20) to modify the synthesis problem as follows:

$$\min_{K,D} \max \mu_\Delta(DF_l(M,K)D^{-1}(j\omega)) < 1 \tag{4.27}$$

In this problem, both transfer functions D and K can be changed completely, and the optimal piece can be determined through appropriate scaling. The rest of the optimal controller design based on this solution approach is introduced as the $D-K$ iteration method.

4.3.4.4 Controller Design Using D-K Method

This process follows these steps as shown in Figure 4.25:

1. Start with initial D, typically D is taken as the identity matrix:

$$D = I \tag{4.28}$$

2. Step k: Having fixed D, solve the following optimization problem to find controller K:

$$K = \inf_{K(\Delta)} \|DF_l(M,K)D^{-1}\|_\infty = \inf_{K(\Delta)} \|DND^{-1}\|_\infty \tag{4.29}$$

3. Step k: Having fixed K, find the scaling matrix D to minimize the cost function at each frequency:

$$D(j\omega) = \inf_D \sigma[DF_l(M,K)D^{-1}] = \inf_D \sigma[DND^{-1}] \tag{4.30}$$

4. Through rational approximation, fit $D(j\omega)$ with a stable, minimum-phase transfer function with reasonable order. If the cost function at step 3 converged to 1, solve the problem; otherwise, return to step 2.

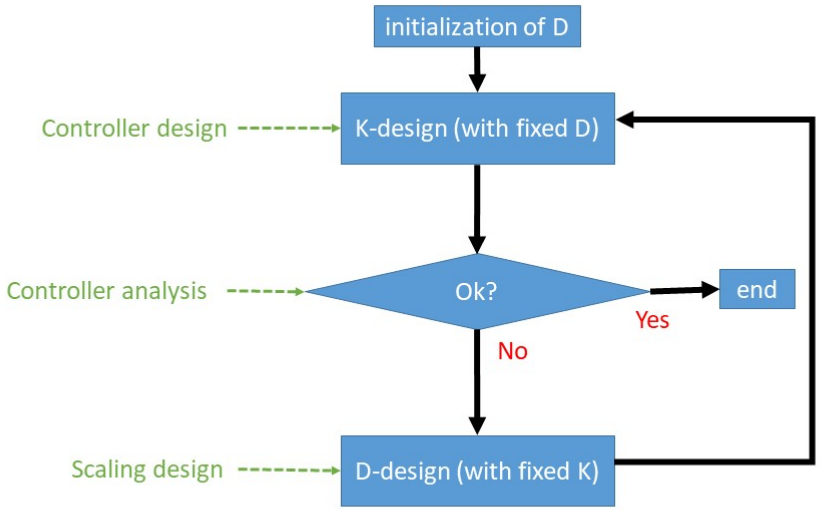


Figure 4.25: D-K iteration method algorithm for controller design.

It should be noted here that the $D-K$ method has a unique feature in that while holding D constant in each stage, there is no guarantee in calculating the minimum point, but practical experience has shown that this method works well and provides suitable responses according to the robust performance criterion. It should also be noted that finding the controller depends heavily on the initial guess of D, and this is why the process is repeated several times until the controller response improves. Due to the fact that in each optimal controller stage, the order of the controller increases for system order n, the order D will inevitably lead to an increase in the controller order. This issue will be explained in detail in this section.

4.3.4.5 dksyn Command

Considering the iterative and complex calculations of the D-K algorithm, the dksyn command can be used in suitable software. The general format of this command is:

```
[K,clp,bnd,dkinfo] = dksyn(p,nmeas,ncont,opt)
```

In this command, the input model must be in `ss` format with uncertainty, and the uncertainty factors must be included. The variable `nmeas` includes the number of measurement inputs, and `ncont` specifies the number of control inputs. The `opt` variable is also set through `dksynOptions`. In the command outputs, the `dksyn` variable includes the designed controller according to the shape of the state space by the D-K algorithm method.

4.3.4.6 Controller Design Using D-K Method For G_2

The roll control channel (G_2) is now more sensitive and complicated than the other channels in the lateral axis channels. As stated above, we use the D-K iteration approach, which is depicted in the Figure 4.26, to build a controller for the $G2$ system.Additionally, we will not be designing the controller type for $G1$ to keep things brief.

Iteration Summary

Iteration #	1	2	3	4	5
Controller Order	8	8	8	8	8
Total D-Scale Order	0	0	0	0	0
Gamma Acheived	1.849	1.379	1.077	1.067	1.066
Peak mu-Value	1.816	1.118	1.069	1.067	1.065

Next MU iteration number: 6

Figure 4.26: Output of the Matlab software related to the D-K Iteration method for the controller designed for the $G2$ system.

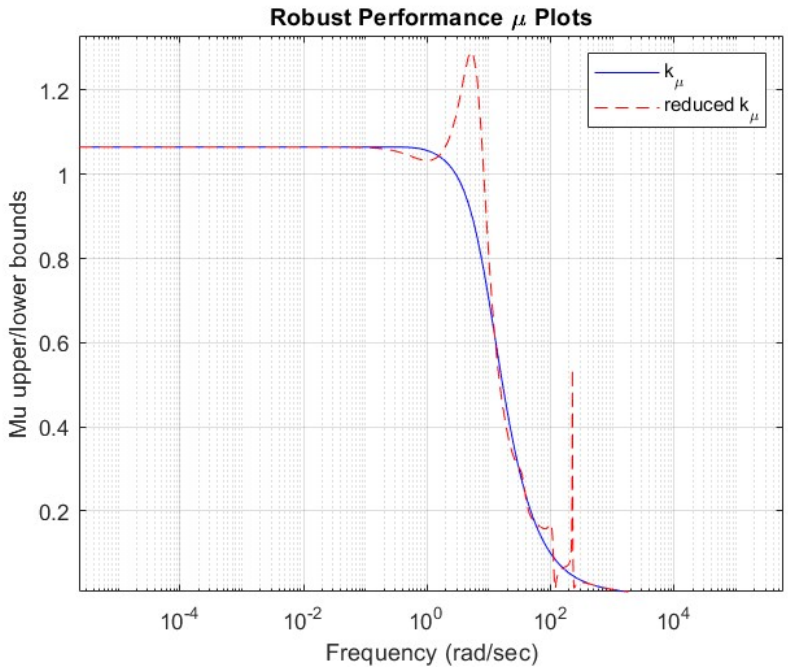


Figure 4.27: Robust Performance μ plots comparing original and reduced-order controllers for G_2.

In Figure 4.28, the step response in the design of the μ controller and the reduced controller are compared and are almost identical to each other and finally, you can see the comparison of the step response of each of them in the Figure 4.28 which shows that they both have an appropriate settling time, without overshooting as well.

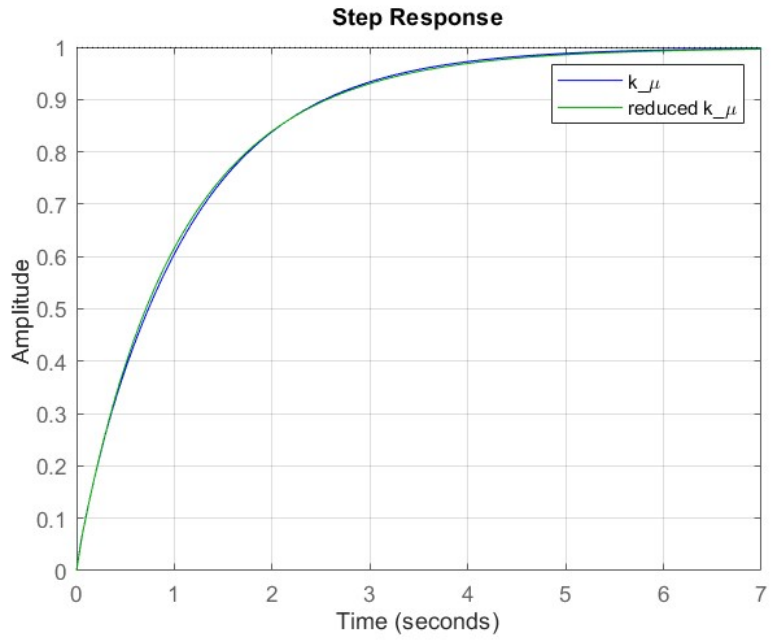


Figure 4.28: Step response comparison of G_2 system between the μ Synthesis controller and its reduced-order version.

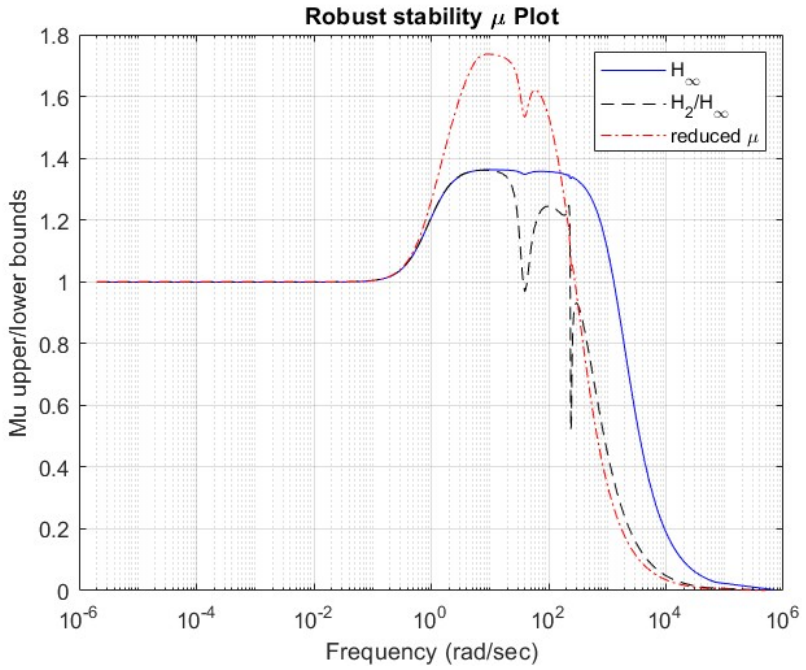


Figure 4.29: Robust stability μ analysis comparing $\mathscr{H}_\infty$, $\mathscr{H}_2/\mathscr{H}_\infty$ and reduced μ controllers.

The μ analysis plot 4.29 provides critical insights into the robust stability characteristics of three control strategies across the frequency spectrum. At low frequencies up

to approximately 10^0 rad/sec, all three controllers maintain a consistent upper/lower bound of 1.0, indicating similar nominal performance.

The mid-frequency range reveals distinct behavioral patterns, with the reduced μ controller exhibiting the highest peak of approximately 1.6 around 10^1 rad/sec, while both H_∞ and $\mathscr{H}_2/\mathscr{H}_\infty$ controllers show more moderate peaks of about 1.4. This suggests that the reduced μ controller may be more sensitive to uncertainties in this frequency range.

In the higher frequency region beyond 10^2 rad/sec, significant differences emerge in the controllers' roll-off characteristics. The $\mathscr{H}_\infty$ controller demonstrates a more gradual descent, extending its influence to higher frequencies, while both the $\mathscr{H}_2/\mathscr{H}_\infty$ and reduced μ controllers show steeper roll-off rates. The $\mathscr{H}_2/\mathscr{H}_\infty$ controller notably exhibits a sharp notch around 10^2 rad/sec before its final descent.

These characteristics suggest that while all three controllers maintain robust stability, they offer different trade-offs between performance and robustness. The $\mathscr{H}_\infty$ controller provides more consistent behavior at high frequencies but with slower roll-off, while the reduced μ and $\mathscr{H}_2/\mathscr{H}_\infty$ controllers offer more aggressive attenuation at high frequencies at the cost of potentially higher sensitivity in the mid-frequency range. This analysis provides valuable guidance for controller selection based on specific application requirements and uncertainty considerations.

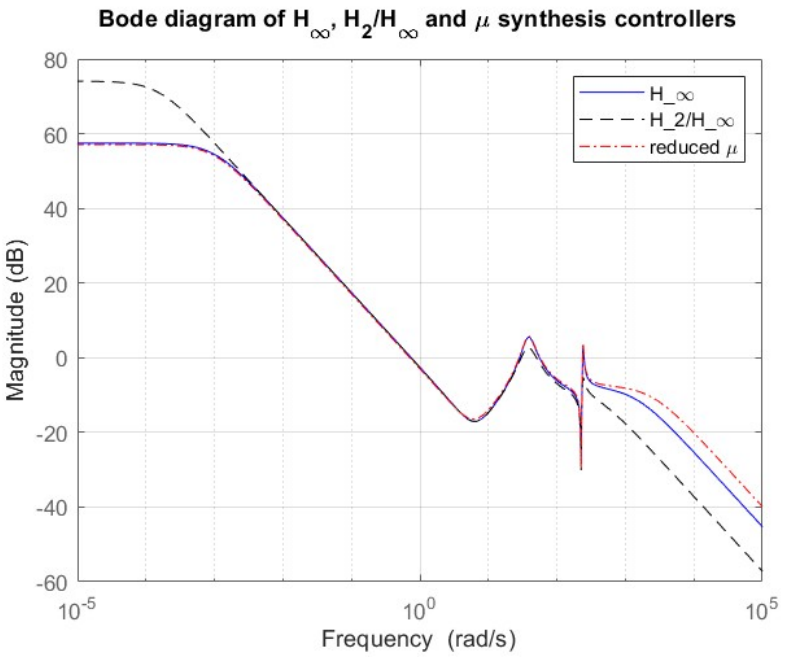


Figure 4.30: Bode diagram comparison of $\mathscr{H}_\infty$, $\mathscr{H}_2/\mathscr{H}_\infty$ and μ synthesis controllers.

The Bode diagram in Figure 4.30 presents a comparative frequency response analysis of three different control strategies: $\mathscr{H}_\infty$, $\mathscr{H}_2/\mathscr{H}_\infty$, and reduced μ synthesis controllers.

The analysis reveals important characteristics across the frequency spectrum.

In the low-frequency range below 10^{-2} rad/s, all three controllers demonstrate similar behavior with a magnitude of approximately 60 dB, confirming effective steady-state tracking performance. Through the mid-frequency region from 10^{-2} to 10^{0} rad/s, the controllers maintain nearly identical responses, exhibiting a consistent slope of about -20 dB/decade, which indicates comparable capabilities in disturbance rejection.

A significant resonance peak emerges around 3 rad/s for all controllers, though with subtle variations in magnitude, reflecting their distinct approaches to managing system dynamics at this critical frequency. The most pronounced differences become apparent in the high-frequency region above 10^{2} rad/s, where the $\mathscr{H}_\infty$ controller shows reduced roll-off compared to its counterparts. Both the $\mathscr{H}_2/\mathscr{H}_\infty$ and reduced μ controllers demonstrate more aggressive attenuation in this range, with the reduced μ synthesis controller closely mirroring the $\mathscr{H}_2/\mathscr{H}_\infty$ response characteristics.

The frequency response analysis ultimately reveals that while all controllers achieve comparable performance objectives at low and mid frequencies, the reduced μ synthesis and $\mathscr{H}_2/\mathscr{H}_\infty$ controllers potentially offer superior noise rejection and robustness characteristics at higher frequencies compared to the traditional $\mathscr{H}_\infty$ controller. This comprehensive comparison provides valuable insights for controller selection based on specific application requirements and desired performance characteristics across the frequency spectrum.

4.3.4.7 Controller Design Using D-K Method For G_3

This part presents the analysis of a robust controller design using the D-K iteration method for system G_3. The design process involves iterative optimization of the controller order and D-scales to achieve robust performance while maintaining stability margins.

Iteration Summary

Iteration #	1	2	3	4	5
Controller Order	9	9	11	11	11
Total D-Scale Order	0	0	2	2	2
Gamma Acheived	2.395	1.799	1.609	1.606	1.606
Peak mu-Value	2.394	1.654	1.608	1.606	1.606

Next MU iteration number: 6

Figure 4.31: D-K Iteration summary for G_3 system showing convergence over 5 iterations.

The D-K iteration procedure, as shown in Figure 4.31, converged after 5 iterations. The controller order started at 9 and stabilized at 11 from the third iteration onwards,

while the introduction of D-scales occurred in iteration 3, with a total D-scale order of 2 maintained through the final iterations. The achieved γ value decreased from 2.395 to a final value of 1.606, with the peak μ-value following a similar convergence pattern.

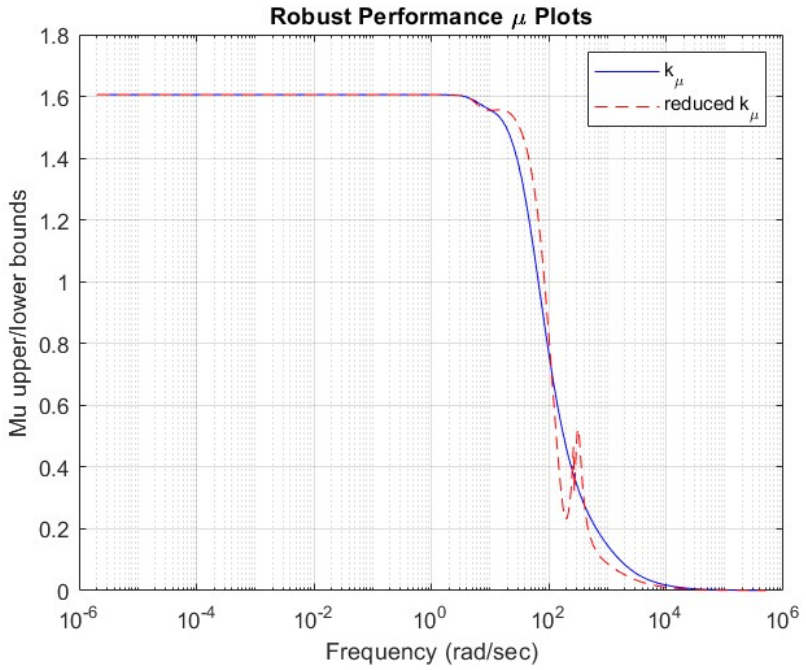


Figure 4.32: Robust Performance μ plots comparing original and reduced-order controllers for G_3.

The robust performance μ plots in Figure 4.32 demonstrate the effectiveness of the controller design. The original and reduced-order controllers maintain similar performance characteristics up to approximately 100 rad/sec, where some deviation becomes apparent. Both controllers achieve a steady-state μ value of approximately 1.6, with good performance in the low-frequency range below 100 rad/sec and proper roll-off characteristics at high frequencies.

The step response comparison in Figure 4.33 reveals excellent tracking performance with a rise time of approximately 0.5 seconds and minimal overshoot of less than 1%. The system achieves a settling time of approximately 2 seconds with no steady-state error. Particularly noteworthy is the identical performance between the original and reduced-order controllers in the time domain, suggesting successful controller reduction without performance degradation.

The Bode diagram comparison in Figure 4.34 between the original (k_μ) and reduced controllers reveals matching frequency response characteristics up to approximately 100 rad/s. Both controllers exhibit similar low-frequency gain characteristics of approximately 70 dB. Some deviation in the high-frequency response becomes apparent

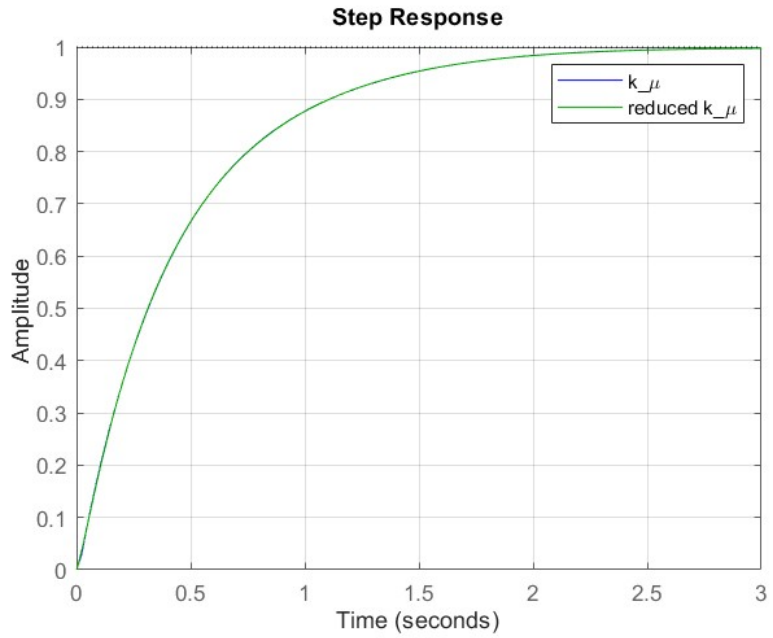


Figure 4.33: Step response comparison between original and reduced-order controllers for G_3 system.

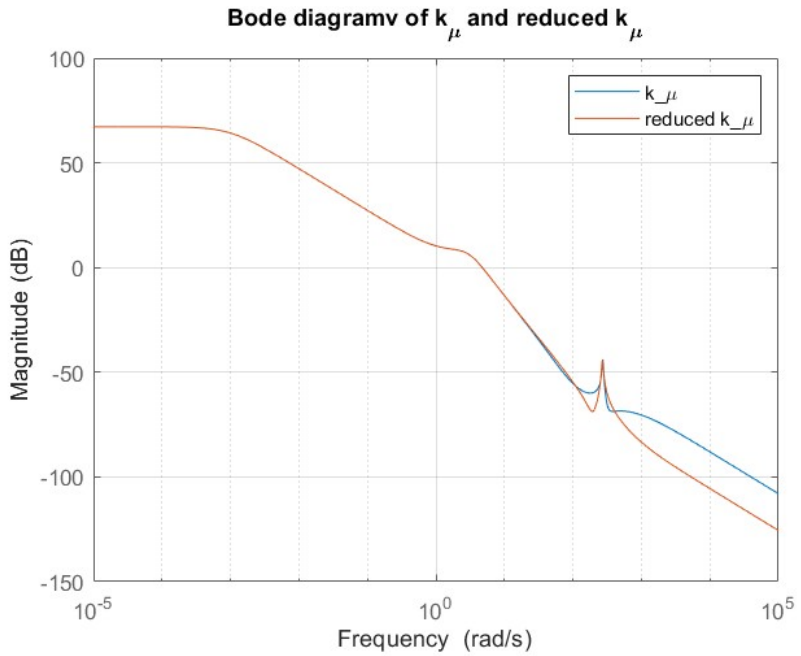


Figure 4.34: Bode diagram comparing frequency responses of original and reduced-order controllers.

above 1000 rad/s, with a notable resonant peak around 100 rad/s in the reduced controller. The magnitude plot shows appropriate roll-off at high frequencies, ensuring good noise rejection properties.

The D-K iteration process has successfully produced a robust controller that meets the design objectives. The process achieved convergence in 5 iterations with a final γ value of 1.606. The resulting controller maintains an order of 11 with a D-scale order of 2, demonstrating excellent time-domain performance and robust stability across the frequency range of interest. The reduced-order controller maintains nearly identical performance to the full-order controller in both time and frequency domains up to the critical frequency range, making it a viable option for practical implementation. The overall results indicate a successful robust control design that balances performance requirements with implementation constraints.

4.4 Longitudinal plan control architecture

This section presents the synthesis of the longitudinal flight control system operating in the X-Z plane. The control architecture comprises two primary coupled control loops: pitch attitude (regulation and altitude control, operating simultaneously).

4.4.1 Pitch control loop

The pitch control loop for VTOL drones is the most complicated control loop. Similar to the latheral loops, we first attempted to directly control the pitch angle and get actual flight data, but the outcomes were not as encouraging as we had hoped. As before, among the 10 identified systems, we choose the one that has better relations in terms of poles and zeros.

Pitch Pitch-Command: The input-output category is the Pitch Pitch-Command. The identified systems' bode diagrams of this case is plotted in Figure 4.35 and nominal model is shown in Equation (4.31).

$$G(s) = \frac{0.0824s^4 + 2.56s^3 + 4202s^2 + 5.394 \times 10^4 s - 1.534 \times 10^6}{s^4 + 4.179s^3 + 5.014 \times 10^4 s^2 + 6.985 \times 10^4 s + 7.155 \times 10^4} \tag{4.31}$$

The plot corresponding to the zero and pole of this system is shown below in Figure 4.36:

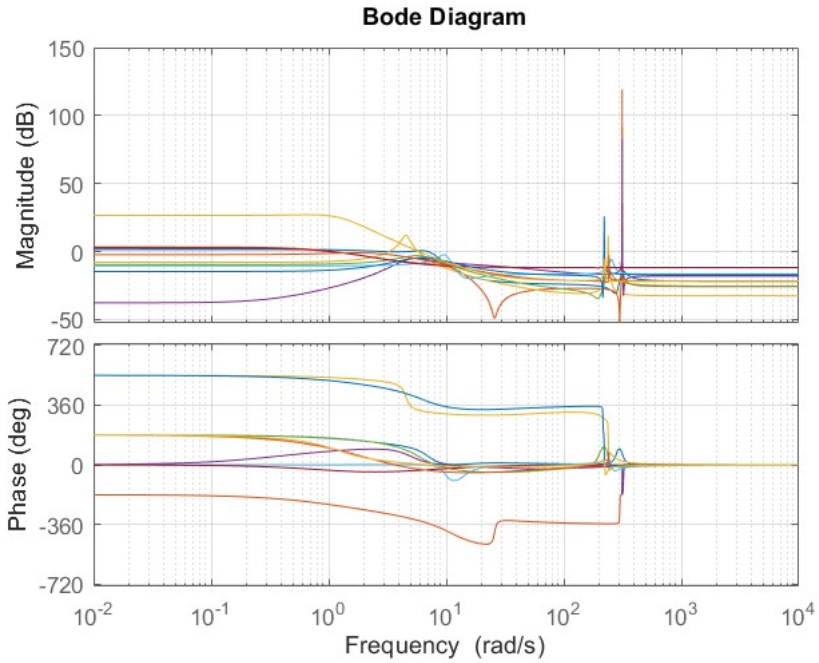


Figure 4.35: Bode diagram of an identified system for pitch loop.

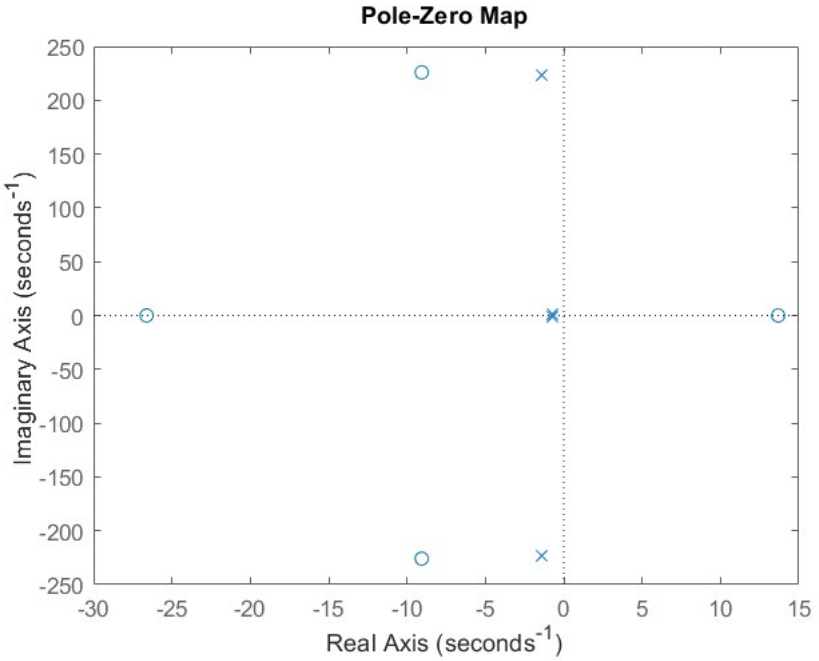


Figure 4.36: pole-zero plot of nominal model pitch loop identified system.

The pole-zero map shows 4 zeros (at -25, ±220j, and +15) and 3 poles (at origin and ±220j). The presence of a zero in the right half-plane at +15 indicates this is a non-minimum phase system, while the pole at origin suggests an integrator behavior. The

control is complicated primarily due to the non-minimum phase characteristic, which introduces an initial inverse response (wrong-way response) before the system moves in the desired direction. This means when you try to control the system to go up, it will first briefly go down before moving up. This fundamental limitation makes the control design challenging because you can't achieve both fast response and good tracking simultaneously - you must trade off between them. The high-frequency complex conjugate pairs of poles and zeros around ±220j also suggest potential oscillatory behavior at those frequencies, which could require careful compensation in the control design.

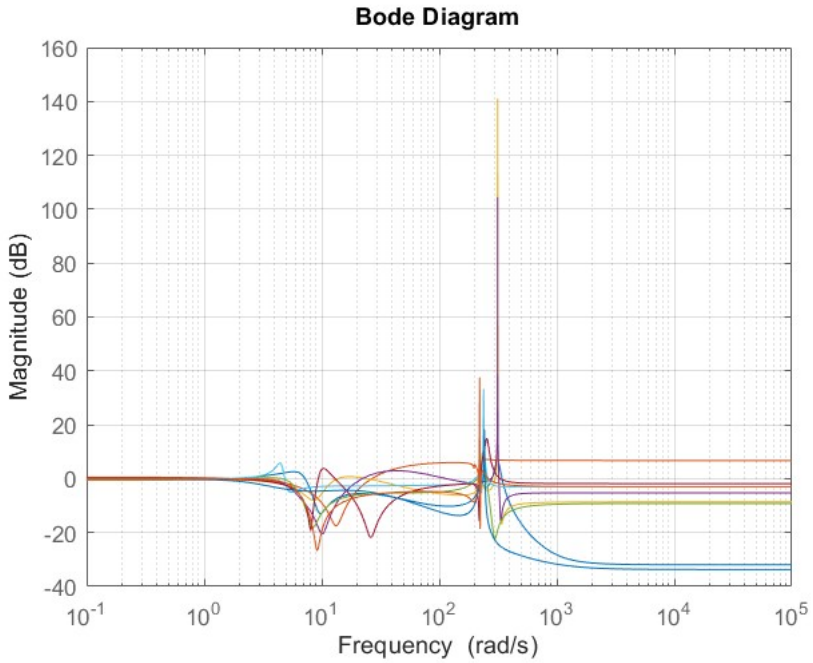


Figure 4.37: Bode diagram of uncertainty profile for pitch loop identified system.

The large maximum amplitude of the uncertainty as it can be seen in Figure 4.37 makes it challenging to derive an appropriate transfer function. Even if such a transfer function could be obtained, the resulting controller would fail to meet the minimum desired robust design specifications.

4.4.1.1 Controlling the pitch angle with nested loops

In the field of aerospace control systems, particularly in aircraft autopilot design, achieving precise and stable attitude control presents unique challenges due to the complex aerodynamic interactions and inherent system dynamics. While it might seem intuitive to directly control an aircraft's attitude angles (pitch, roll, and yaw), this approach often proves suboptimal in practice. Instead, aerospace engineers have developed a more sophisticated control strategy using cascaded loops.

The fundamental principle behind this approach stems from the aircraft's natural dynamics and the availability of high-quality gyroscopic sensors. Rather than attempting to control the aircraft's attitude angles directly, modern autopilot systems employ a hierarchical control structure where the faster inner loop manages angular velocity using gyroscopic feedback, while the slower outer loop maintains the desired attitude angle. This architecture, known as cascade control, takes advantage of the natural damping properties of rate feedback and the precise measurements available from modern gyroscopes (Blakelock, 1991). Similarly, we employed a cascaded control structure where the pitch rate controller (q) was designed as the inner loop, followed by the pitch angle controller (θ) as the outer loop.

Pitch-Rate Pitch-Command: The input-output category is the Pitch-Rate Pitch-Command. The identified systems' bode diagrams of this case are plotted in Figure 4.38 and nominal model is shown in Equation (4.32).

$$P_0 = \frac{2.586s^4 + 5.902s^3 + 4.91 \times 10^4 s^2 + 2.044 \times 10^4 s + 3.224 \times 10^6}{s^4 + 5.113s^3 + 1.903 \times 10^4 s^2 + 4.658 \times 10^4 s + 1.651 \times 10^6} \tag{4.32}$$

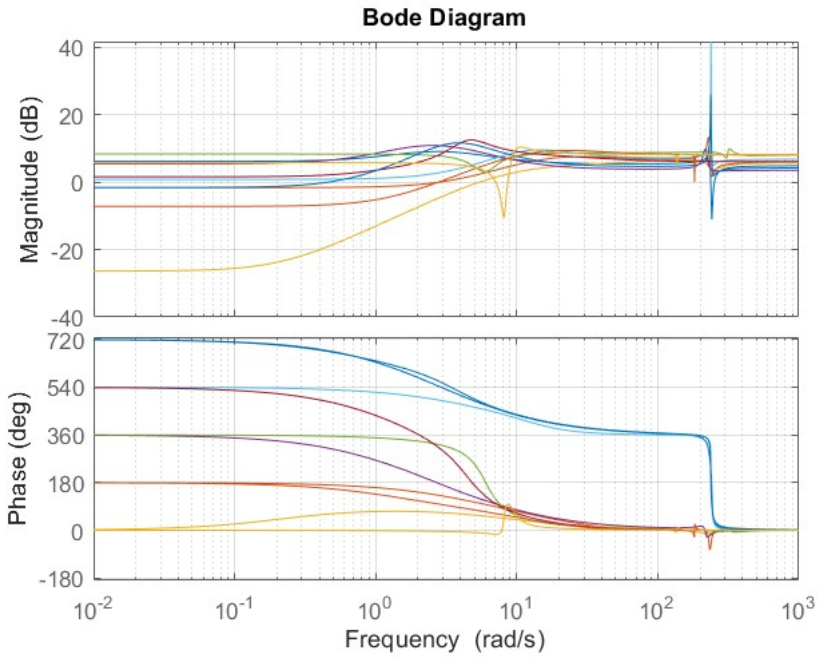


Figure 4.38: Bode diagram of pitch-rate loop identified systems.

Uncertainty Profile Modeling

We apply the Equation (4.8) and find the uncertainty weighting function(4.33), and this limitation can cover all the uncertainty as revealed in Figure 4.39:

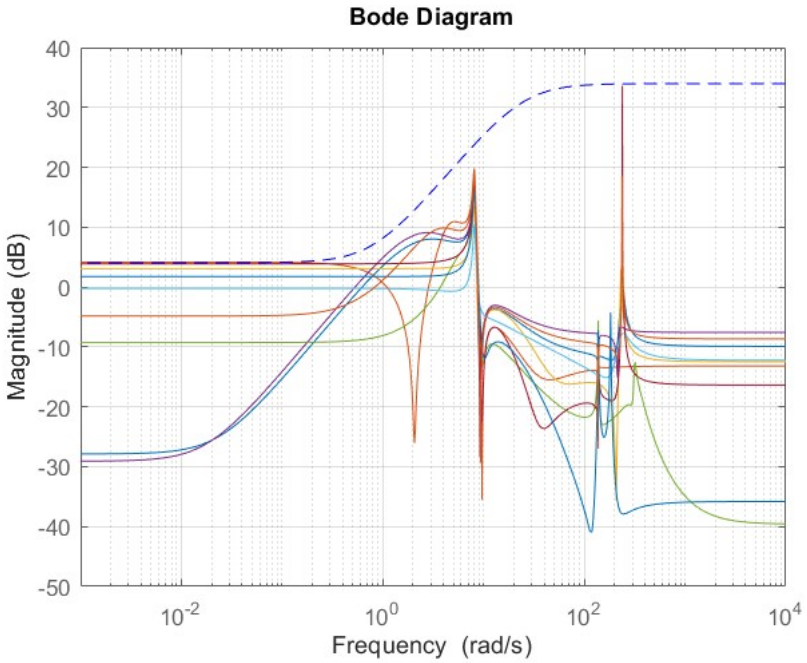


Figure 4.39: Bode diagram of uncertainty profile for pitch-rate loop identified system.

$$W_T = \frac{50s+40}{s+25} \tag{4.33}$$

4.4.1.2 Mixed Sensitivity Controller Design for Pitch Rate Control Loop

Now with the identified nominal system and uncertainty profile waiting function, we can design $\mathscr{H}_\infty$ mixed sensitivity controller as before for the pitch rate interior Loop as a damping Loop:

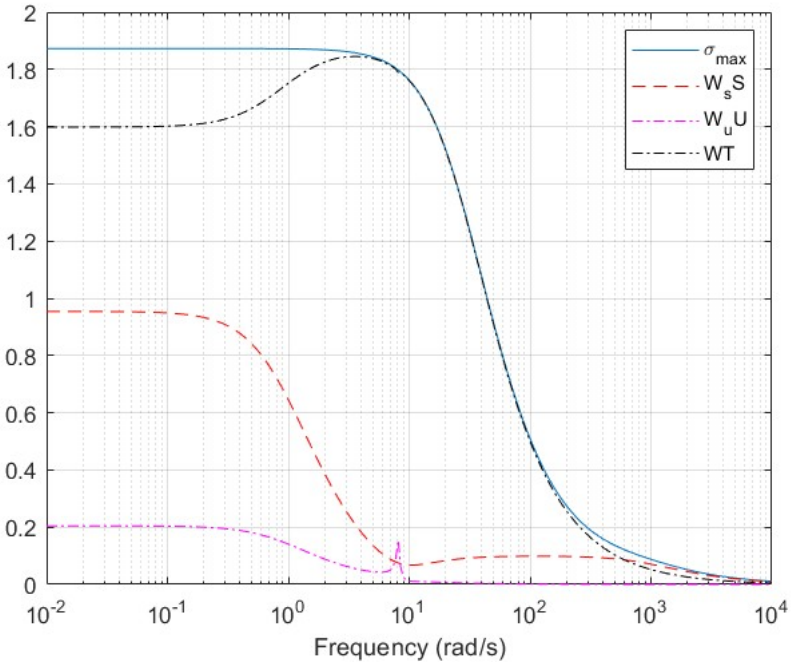


Figure 4.40: Design parameters of P_0 for $\mathscr{H}_\infty$ in mixed sensitivity scenario.

As can be seen in Figure 4.41, we could not reach the desired $\gamma_{opt}(\sigma_{max})$, which is should be below one, but we are close to it ($\gamma_{opt} = 1.9$) and this result is acceptable as an internal damping loop. For the inner loop of pitch rate control, we has tested another control method to see if we can achieve better robust stability results.

4.4.1.3 Controller Design with μ synthesis Method

To enhance both robust performance and robust stability while achieving improved response time, we implement a μ-synthesis controller design approach. This design methodology offers systematic handling of multiple control objectives and uncertainty requirements.

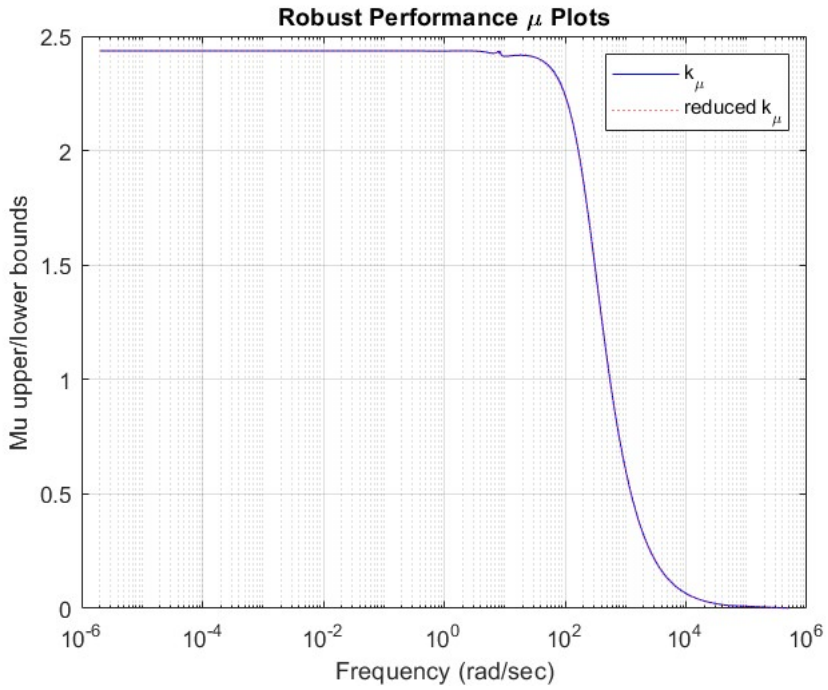


Figure 4.41: Design parameters of P_0 (pitch rate control loop) in μ synthesis scenario.

The Figure 4.41 shows two nearly identical plots comparing the performance of the original and reduced-order controllers. While the original controller is in the 11th order, making it impractical to implement, we successfully reduced its complexity to a 5th order controller using the 'minreal' command without significantly impacting performance. However, the optimal gamma value obtained ($\gamma_{opt} = 2.5$) is higher than the previously designed H-infinity controller, which achieved $\gamma_{opt} = 1.9$, indicating the H-infinity controller's superior robustness. Figure 4.42 illustrates the step response for the pitch rate control loop, showing rapid and well-behaved system dynamics with appropriate settling characteristics.

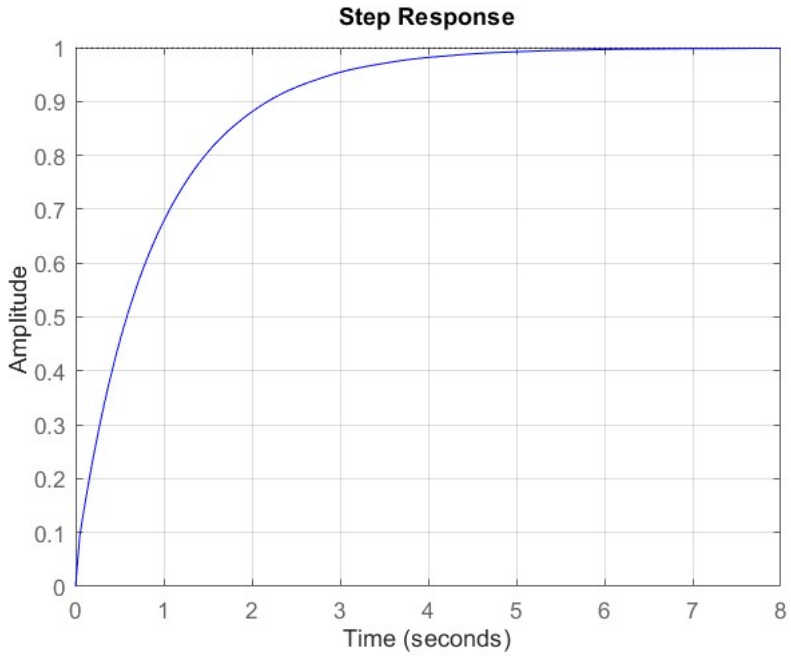


Figure 4.42: Step response in μ synthesis scenario

Figure 4.43 presents the DK-iteration command output for the controller synthesis process. The results demonstrate progressive improvement over five iterations, where the peak μ-value was successfully reduced from 3.439 to 2.443. Initially, the controller maintained an 8th order structure for the first two iterations, then increased to 10th order with the introduction of D-scales (order 2) from iteration 3 onwards. The convergence behavior is evident as both the achieved γ and peak μ-value show consistent reduction, particularly significant in the early iterations, before stabilizing around 2.44 by the fourth iteration.

Iteration Summary

Iteration #	1	2	3	4	5
Controller Order	8	8	10	10	10
Total D-Scale Order	0	0	2	2	2
Gamma Acheived	3.448	2.700	2.462	2.436	2.443
Peak mu-Value	3.439	2.607	2.456	2.436	2.443

Next MU iteration number: 6

Figure 4.43: DK-iteration results showing the convergence of controller synthesis over five iterations.

The Bode diagrams (Figure 4.44 and Figure 4.45)provide a comparative analysis between H_∞ (shown in the first figure) and μ-synthesis control designs for the same plant. Both controllers demonstrate similar general characteristics across the frequency range

of 10^{-4} to 10^{4} rad/s, with high gain at low frequencies around 50 dB. However, examining the resonant peak behavior near 10 rad/s reveals subtle differences in how each control strategy handles system dynamics. The μ-synthesis approach, shown with both its original and reduced-order implementations, maintains consistent performance while specifically addressing structured uncertainties in the system. The similar frequency response patterns suggest that both H_∞ and μ-synthesis achieve comparable closed-loop performance, though the μ-synthesis offers additional robustness guarantees through its explicit treatment of structured uncertainties. The successful reduction of the μ-synthesis controller preserves these essential characteristics while providing a more practical implementation structure(with Matlab `minreal` command).

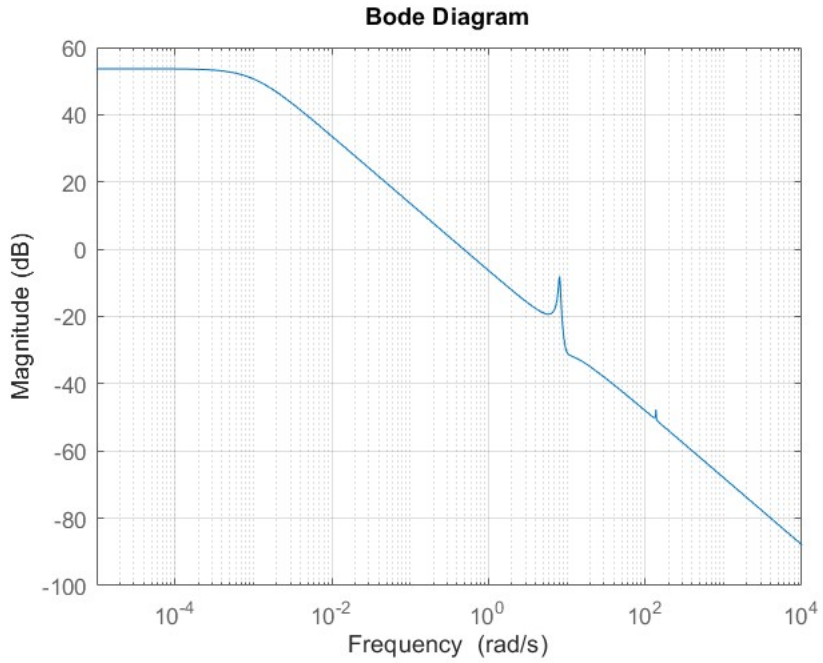


Figure 4.44: Frequency response of the controller designed in the $\mathscr{H}_\infty$ scenario.

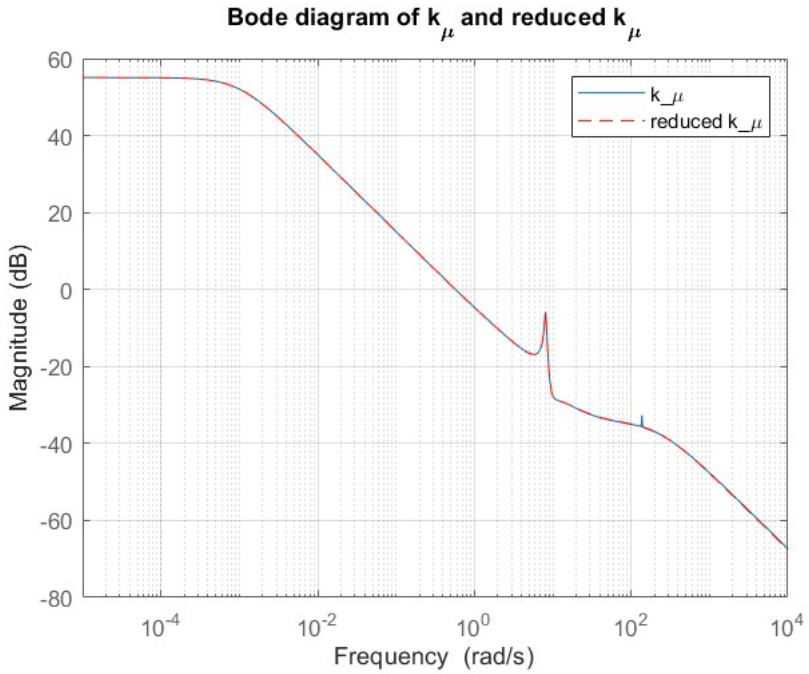


Figure 4.45: Frequency Response of the Controller Designed in the μ-synthesis Scenario.

4.4.1.4 Design of $\mathscr{H}_\infty$ Controller for Outer Loop Pitch Angle Control Based on Cascade Structure

After establishing the internal pitch rate loop, we proceed to design the outer pitch angle control loop. This cascade structure introduces important considerations for uncertainty modeling and controller synthesis. The uncertainty profile for the outer loop differs fundamentally from the inner loop - instead of using the original uncertainty model, we employ a modified version where the weighted function is attenuated, typically to one-tenth of the original W_T. This reduction reflects the improved robustness achieved through the inner loop control.

The plant structure for the outer loop incorporates two key elements: first, the closed-loop dynamics of the previously designed pitch rate control system, and second, the inherent integration relationship between pitch rate and pitch angle. Figure 4.46 reveals the control block diagram. This relationship is represented mathematically as the angular position being the time integral of angular velocity around the pitch axis. Therefore, the complete outer loop plant transfer function is constructed by cascading the closed-loop inner loop transfer function with an integrator (1/s) as shown in Equation (4.34).

For this augmented plant model shown in Equation (4.35), we synthesize a new $\mathscr{H}_\infty$ controller(controller 2). The control design objectives for the outer loop typically emphasize precise attitude tracking, disturbance attenuation, and maintaining robustness under parameter variations and external disturbances.

As previously explained, the weighted sensitivity function W_S is shaped to ensure good command tracking and disturbance rejection at low frequencies, while the complementary sensitivity function W_T maintains robust stability against the reduced uncertainty profile. The control sensitivity weight W_U is adjusted to prevent excessive control action that might saturate the inner loop.

This cascade structure offers several advantages: the inner loop provides fast disturbance rejection for pitch rate variations, while the outer loop maintains accurate pitch angle tracking. The separation of control objectives between the two loops allows for more refined tuning of the overall system performance.

The resulting controller achieves robust performance while respecting the dynamic limitations imposed by the inner loop. The reduced uncertainty profile in the outer loop design reflects the stabilizing effect of the inner loop, allowing for potentially more aggressive control action in the outer loop without compromising overall system stability.

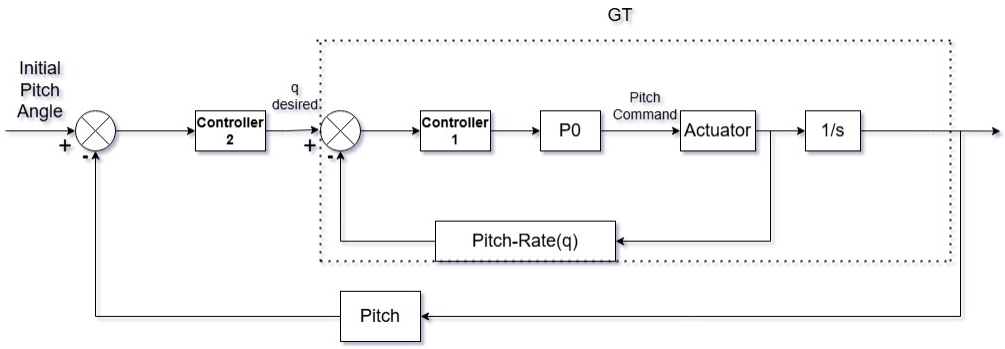


Figure 4.46: Nested control loop structure showing the hierarchical arrangement of the pitch controller system.

$$G_{2out} = \frac{1}{s} \tag{4.34}$$

$$G_T = \frac{k_1 P_0 G_{2out}}{1 + k_1 P_0} \tag{4.35}$$

Where k_1(controller 1) was a controller designed in the section(4.4.1.2).

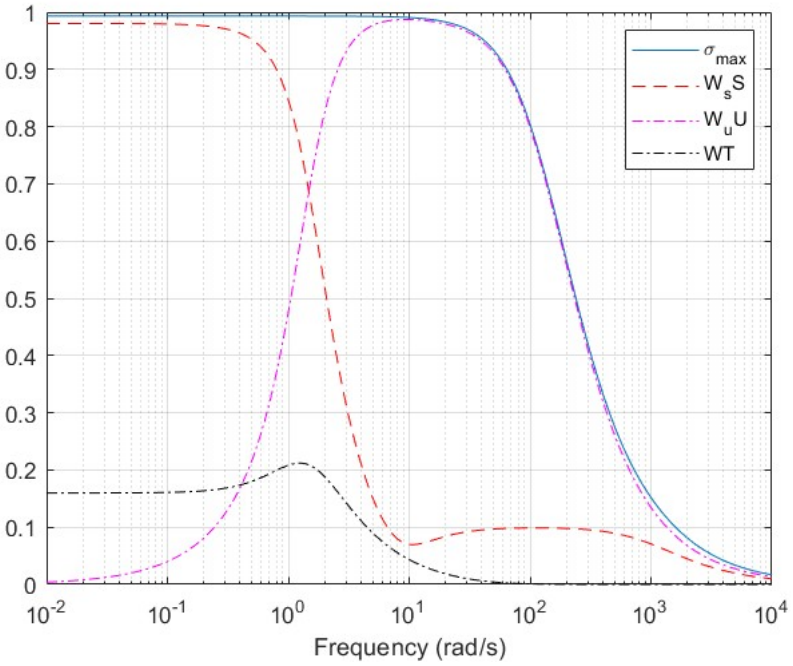


Figure 4.47: Design parameters of the pitch controller loop in the $\mathscr{H}_\infty$ scenario.

Figure 4.47 illustrates the mixed sensitivity functions and their relationships with performance weights. The plot shows four key elements: σ_{max} (maximum singular value), weighted sensitivity function (W_sS), weighted control sensitivity (W_uU), and complementary sensitivity (WT). The crossover between W_sS and W_uU occurs around 1 rad/s, indicating the trade-off between performance and robustness. The W_sS function demonstrates good disturbance rejection at low frequencies, while W_uU ensures control effort limitation at high frequencies. The maximum singular value (σ_{max}) remains below 1 across all frequencies, confirming that the robust performance criteria are satisfied.

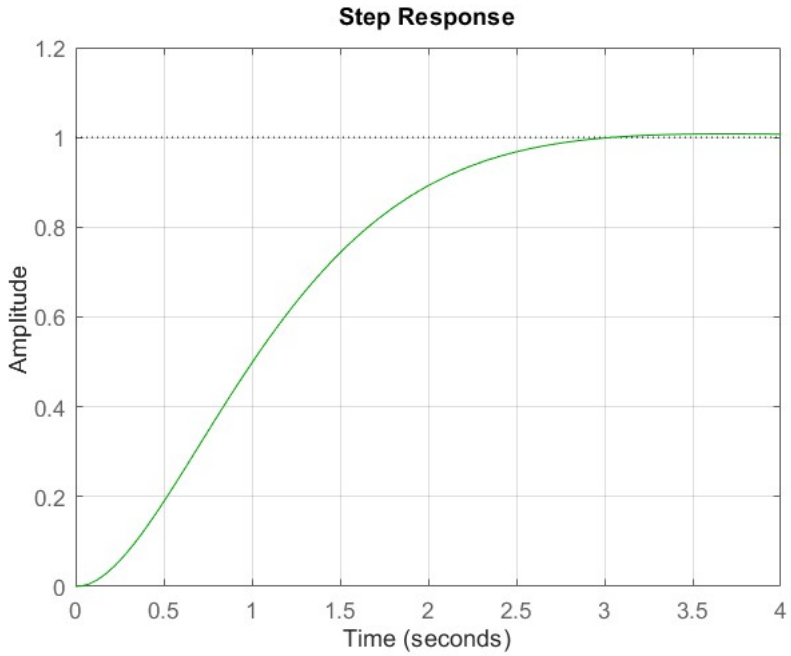


Figure 4.48: Step response of pitch control loop for $\mathscr{H}_\infty$ in mixed sensitivity scenario.

The step response plot in Figure 4.48 reveals the time-domain performance characteristics of the closed-loop system. The response exhibits desirable properties through its smooth trajectory with no overshoot, indicating a well-damped system behavior. The settling time of approximately 3 seconds demonstrates efficient convergence, while the final value reaches unity, confirming accurate tracking performance. This smooth transition to steady-state validates the controller's ability to achieve good command following while maintaining stability margins.

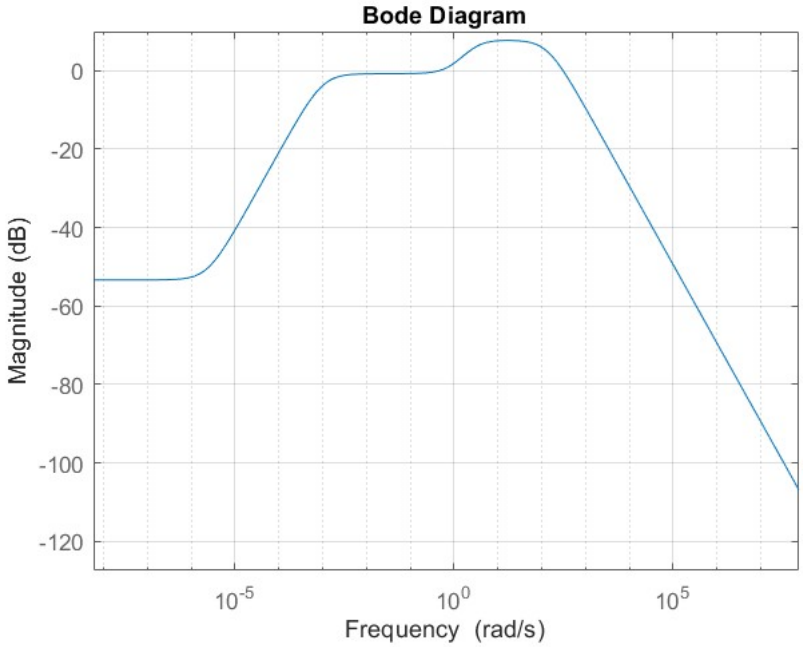


Figure 4.49: Frequency response of the controller designed in the $\mathscr{H}_\infty$ scenario.

The Bode magnitude plot in the Figure 4.49 provides insight into the frequency-domain behavior of the closed-loop system. The low-frequency gain maintains approximately 0 dB, ensuring good steady-state tracking, while a resonant peak near 1 rad/s corresponds to the system's natural frequency. The high-frequency roll-off exceeding -40 dB/decade beyond 10^3 rad/s provides sufficient noise attenuation. The bandwidth of approximately 10^2 rad/s indicates good response speed while avoiding high-frequency excitation.

The combined analysis of these three plots confirms that the design achieves a balanced compromise between performance specifications and robustness requirements. The weighted sensitivity functions ensure proper shaping of the closed-loop responses, while the time-domain performance meets typical control objectives without excessive control effort. The frequency response characteristics validate the design's ability to handle model uncertainties while maintaining desired performance across the frequency range of interest

4.4.2 Altitude control Loop

The altitude control loop represents one of the most critical challenges in unmanned aerial vehicle (UAV) autopilot systems. This challenge becomes even more complex in automatic takeoff and landing systems. The altitude loop's sensitivity and coupling with other control loops make it particularly demanding. Achieving optimal design

in altitude control is crucial not only for maintaining stable altitude but also for simplifying the design of interconnected control loops. As explained previously, the two motors work together to control the height of the drone(see the Equation (4.2)). For feedback loop sensing, the system utilizes altitude data from the MTI-g AHRS module, which combines barometric and GPS altitude measurements. Operating the tethered drone near the ground creates two significant challenges. First, the proximity to the ground weakens GPS signal reception. Second, the drone's configuration makes it difficult to position the barometric altimeter away from propeller-induced disturbances. These disturbances are particularly pronounced due to the ground effect phenomenon, where propeller wash reflects off the ground surface and interferes with barometric readings. Initial attempts to implement direct feedback using the raw altitude measurements proved unsuccessful. The drone exhibited oscillatory behavior at approximately one-meter altitude, presenting a significant safety risk during low-altitude operations. Test results confirmed that using the rangefinder to maintain relative altitude above ground level (AGL) proved more effective than controlling absolute altitude above sea level (MSL). As explained in the previous section, the aforementioned laser rangefinder was selected. Testing showed that while the laser rangefinder's delay is suitable for ground robots, it proved a bit slow for aerial applications. However, we decided to test it and measured the flight data where the altitude control loop was directly connected to the laser rangefinder and proceeded to identify the system and from there to model the uncertainty, which resulted in something like a pitch control loop. The resulting uncertain model was so sparse that it was not possible to cover it with a specific transfer function. This behavior resulted from the cross-axis coupling and sensor latency. Similar to the pitch loop design, we determined that a faster inner loop is needed, incorporating damping characteristics. We implemented this by closing the inner loop using Z-axis acceleration data with gravity acceleration compensation. The amount of Z-axis acceleration data is obtained directly from the AHRS MTi-G module. finally, the outer control loop is closed with a laser rangefinder.

4.4.2.1 Mixed Sensitivity Controller Design for Vertical Acceleration Control

Vertical Acceleration Throttle System: The input-output category is the vertical body acceleration throttle. This point should be noted to calculate the drone's net acceleration(A_z), we subtract the vertical acceleration from gravitational acceleration, which can result in either positive or negative values. The identified systems' bode diagrams of this case is plotted in Figure 4.50 and nominal model is shown in Equation (4.36).

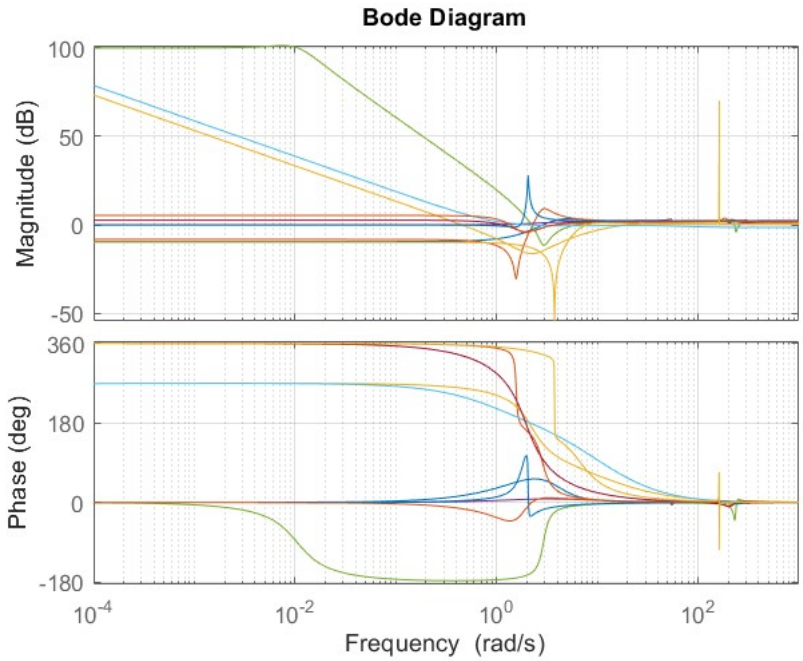


Figure 4.50: Bode diagram of Vertical Acceleration control loop identified systems

$$P_0 = \frac{1.218s^4 + 13.2s^3 + 6.787 \times 10^4 s^2 + 4.13 \times 10^4 s + 5.787 \times 10^5}{s^4 + 18.5s^3 + 5.369 \times 10^4 s^2 + 543.6s + 5.87} \tag{4.36}$$

The domain of the next diagram for the uncertain profile is shown in Figure 4.51, and the weighting function that limits it is given in the Equation (4.37).

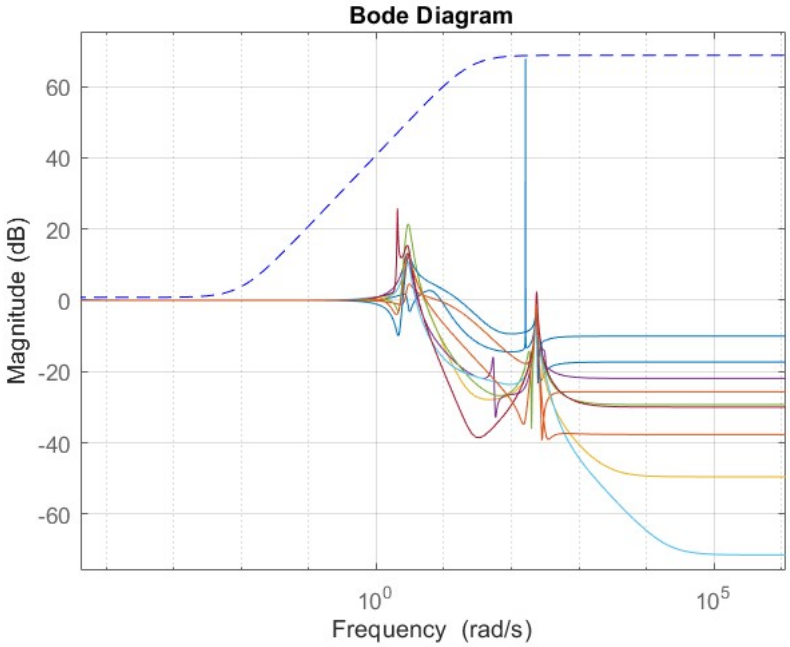


Figure 4.51: Bode diagram of uncertainty profile for Vertical Acceleration control loop system

$$W_T = \frac{2750s + 27.5}{s + 25} \tag{4.37}$$

As we can see in Figure 4.52, the γ_{opt} or σ_{max} value is around 9 and above one dramatically. Still, this design is not a problem because it will be fully compensated in the outer loop design.

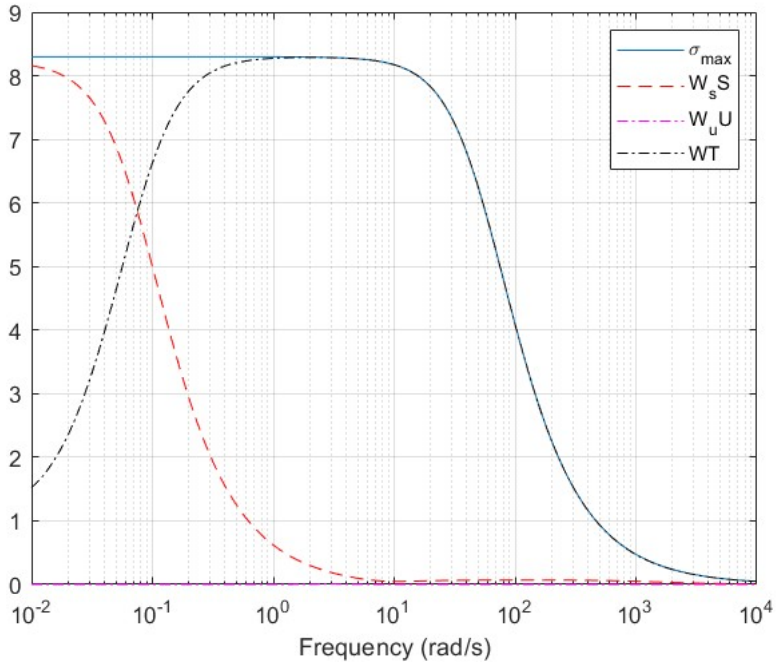


Figure 4.52: Design parameters of Vertical Acceleration control loop in the $\mathscr{H}_\infty$ scenario

In Figure 4.53, we see the step response of the inner loop of the vertical acceleration control, which has gone toward zero. Perhaps the question arises as to how this is a step response. The reason is that this inner loop plays the role of a damper only and its non-zero desired value is meaningless.

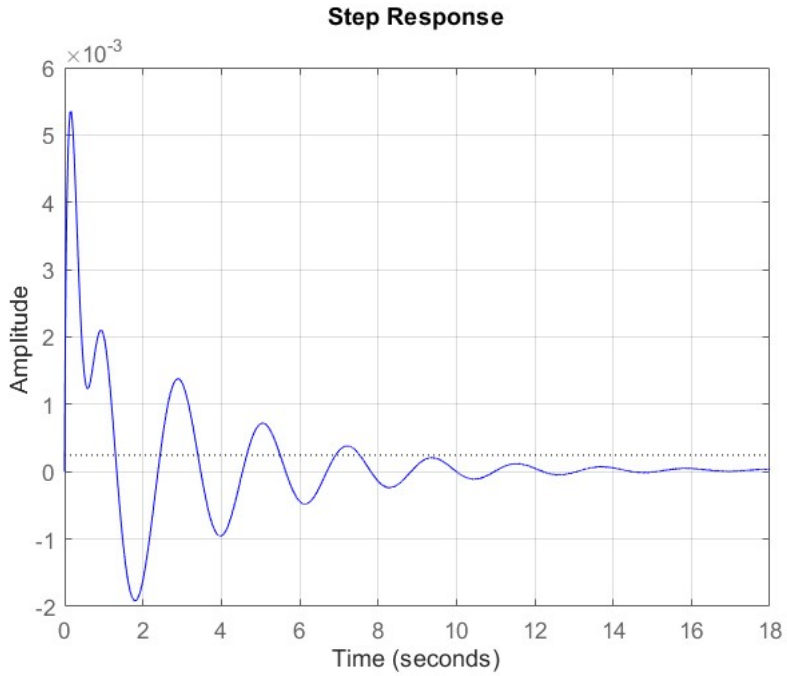


Figure 4.53: Step response of Vertical Acceleration closed-loop system

The frequency response shows clear robustness characteristics in Figure 4.54:

At low frequencies (below 10^{-2} rad/s), the magnitude is flat around -60 dB, providing good steady-state performance and disturbance rejection.

Around 1 rad/s, there's a prominent peak reaching approximately -20 dB, which suggests increased control authority in the nominal operating range.

At high frequencies (above 10^2 rad/s), the response exhibits strong attenuation with a steep roll-off, which helps with noise rejection and ensures robustness against high-frequency unmodeled dynamics.

There's also a notable notch around 10^{-2} rad/s and a small resonance near 10^2 rad/s, which may be specifically designed to handle model uncertainties or achieve robust stability margins.

This overall shape is typical for robust control designs where the controller balances performance in the operating frequency range while maintaining stability in the presence of uncertainties and ensuring good noise rejection at high frequencies.

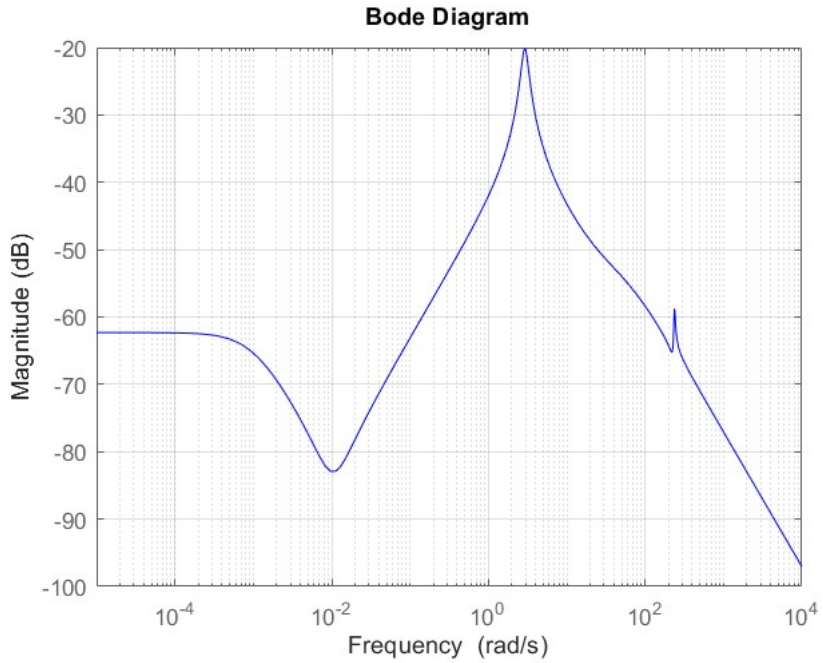


Figure 4.54: Frequency response of the controller designed in the $\mathscr{H}_\infty$ scenario.

4.4.2.2 Design of H_∞ Controller for Outer Loop Altitude Control Based on Cascade Structure

The closed-loop equations are formulated as shown in Equation (4.39). The design of the inner-loop altitude controller mirrors that of the outer-loop pitch controller described in Section 4.4.1.4. For brevity, the detailed explanation is omitted, as the structure is identical except for one key difference: The altitude controller incorporates two integrators—represented by the $\frac{1}{s}$ blocks in Equation (4.38)—to convert acceleration into velocity and velocity into position.

$$G_{2out} = \frac{1}{s^2} \tag{4.38}$$

$$G_T = \frac{k_1 P_0 G_{2out}}{1 + k_1 P_0} \tag{4.39}$$

Again, like the controller design for pitch control loop we consider the one-tenth of W_T as the uncertainty model profile weighting function and design the $\mathscr{H}_\infty$ control(controller 2) for G_T as it shown in Figure 4.55. Also k_1(controller 1) was a controller designed in the section (4.4.2.1).

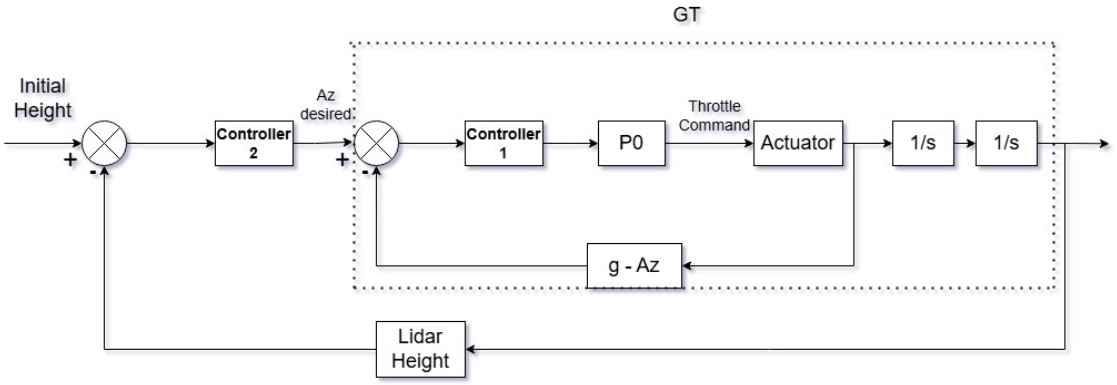


Figure 4.55: Nested control loop structure showing the hierarchical arrangement of the altitude controller system.

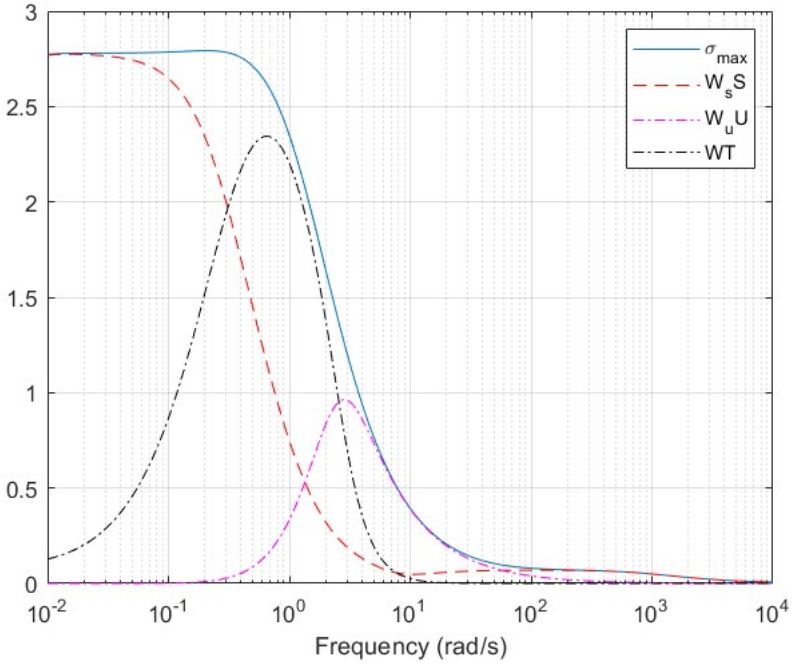


Figure 4.56: Design parameters of altitude closed-loop for $\mathscr{H}_\infty$ in mixed sensitivity scenario

As can be seen in Figure 4.56, the results are not as promising as the previous controllers ($\gamma_{opt}(\sigma_{max}) = 2.8$), but all these efforts show how difficult it can be to design a controller for the height loop in this system and of course, the reason for this is not complicated, as the altitude loop in this aircraft is completely coupled with the pitch, roll, and yaw loops.

The closed-loop altitude control system demonstrates satisfactory performance characteristics. As shown in Figure 4.57, the step response exhibits stable behavior with a settling time of approximately 10 seconds and no overshoot, which aligns with the design requirements because, for altitude control, we consider a change of 1 meter in

height. This movement must be slow and controlled to protect the drone's safety. The controller maintains a gentle vertical speed of 0.1 meters per second (10 centimeters per second) for ascending and descending, which is ideal for safe drone operation. The frequency response of the $\mathscr{H}_\infty$ controller, depicted in Figure 4.58, shows a peak magnitude of 20 dB at approximately 1 rad/s, indicating good robustness properties in the mid-frequency range (Zhou et al., 1996). The controller's roll-off at high frequencies helps in noise attenuation, while maintaining adequate bandwidth for tracking performance, following standard $\mathscr{H}_\infty$ design principles (Dullerud and Paganini, 2000).

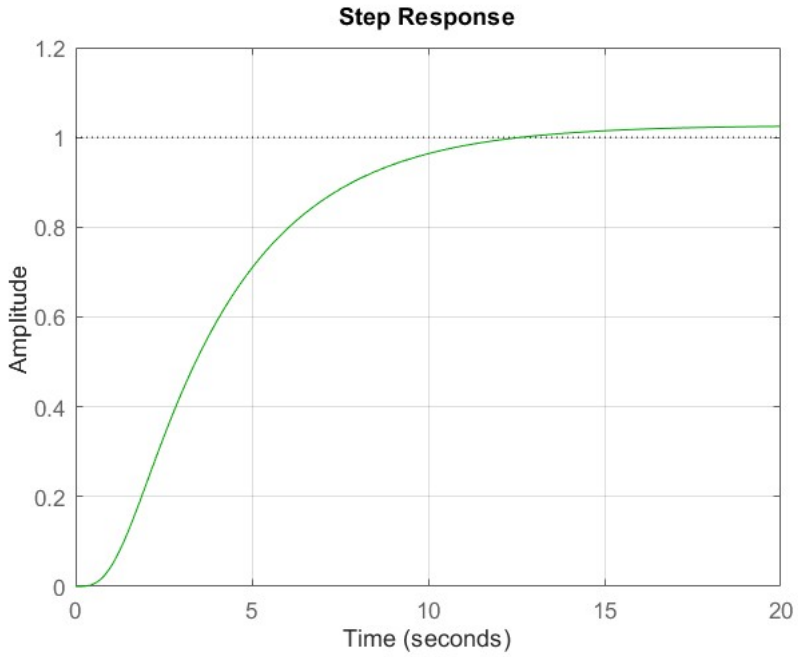


Figure 4.57: Step response of altitude closed-loop system in the $\mathscr{H}_\infty$ scenario.

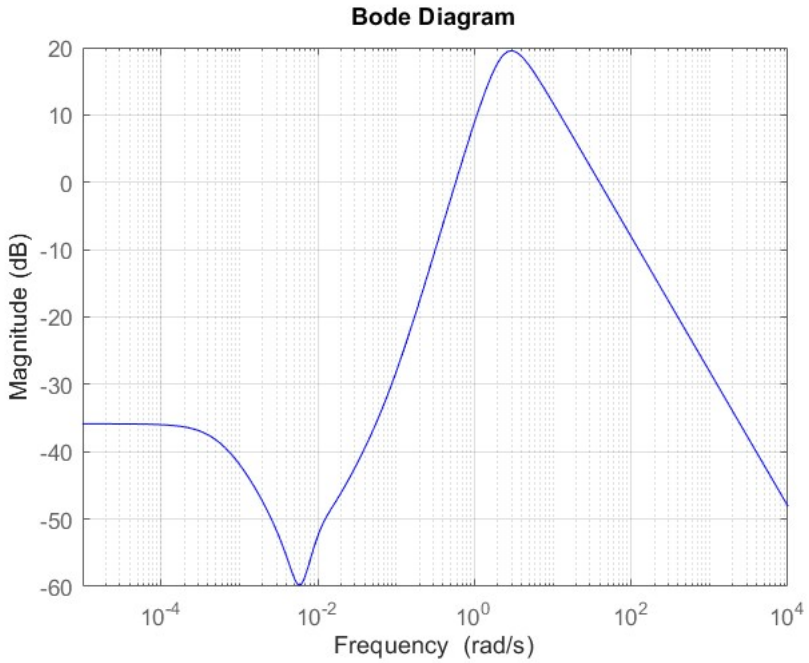


Figure 4.58: Frequency response of the altitude loop controller designed in the $\mathscr{H}_\infty$ scenario.

4.5 MIMO controller Design

As discussed in both the dynamic model section and the current section, the VTOL system exhibits inherent coupling among the four primary axes: roll, pitch, yaw, and altitude. Consequently, the system must be treated as a multi-input, multi-output (MIMO) plant. However, the controllers designed in previous sections were single-input, single-output (SISO), and thus did not account for the cross-coupling effects between channels. In this section, we focus on two of the most strongly coupled channels—roll and altitude. Candidate models for these subsystems were selected in the previous section, and their associated uncertainty profiles have also been identified.

Roll-Altitude coupling MIMO system

In the SISO systems examined thus far, we have focused solely on direct input-output relationships—specifically, roll-command to roll output and altitude-command to altitude output—implemented using nested control loops. Given the greater importance of the inner loop, we now extend the design to a MIMO framework, particularly targeting vertical acceleration control. In practice, it is essential to account for cross-coupling effects, such as: • The influence of roll-command on vertical acceleration A_z • The influence of throttle-command on roll response To address these interactions, we must identify the system behavior using collected experimental data and extract the corresponding transfer functions that characterize these cross-coupled dynamics.

Roll-command vertical acceleration(A_z) system

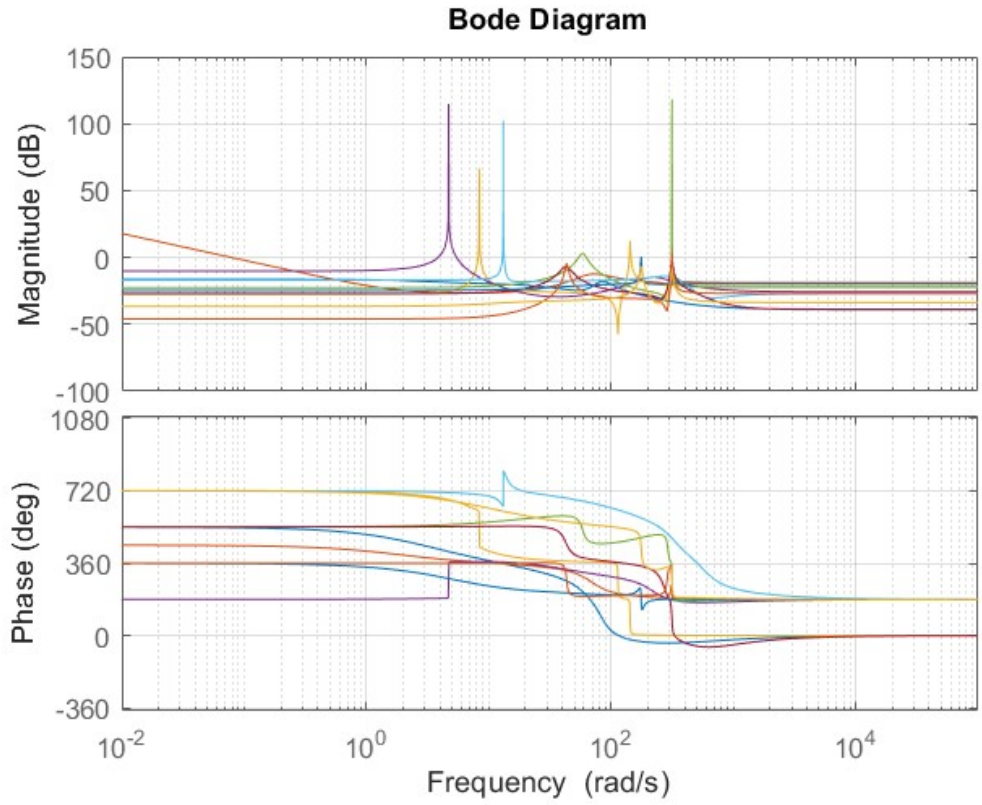


Figure 4.59: Bode diagram of Roll-command to A_z(vertical acceleration) control loop identified systems.

Based on the pole-zero analysis shown in Figure 4.60, we selected the transfer function presented in Equation (4.40) from among the ten candidate models illustrated in Figure 4.59. The selection was made according to the location of the poles and zeros, ensuring the most suitable dynamic behavior for the control design.

$$\text{Roll-Command To } Az = \frac{0.01163s^4 + 5.317s^3 - 228.2s^2 + 2.633 \times 10^4 s - 9.774 \times 10^4}{s^4 + 65.42s^3 + 8142s^2 + 2.638 \times 10^5 s + 6.501 \times 10^5} \tag{4.40}$$

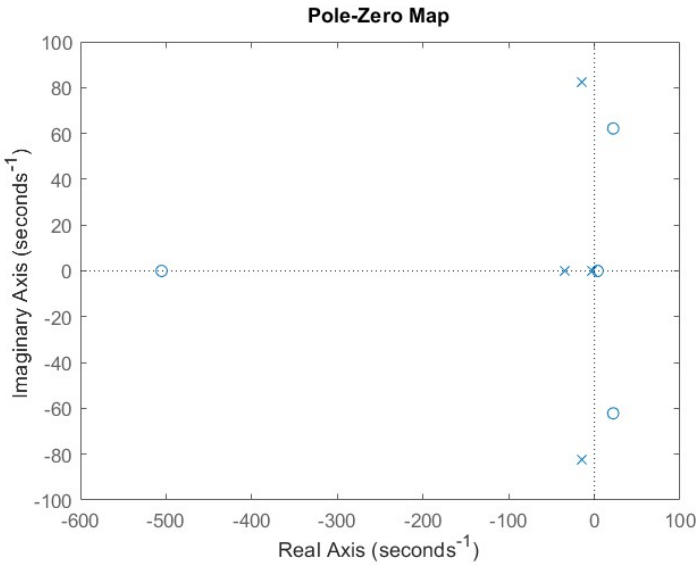


Figure 4.60: pole-zero plot of Roll-Command To A_z(vertical acceleration) identified system.

Throttle to Roll system: Based on the pole-zero analysis, we selected a nominal model from among ten candidate systems. The pole-zero plot corresponding to this selection is shown in Figure 4.61, and the associated transfer function is provided in Equation (4.41).

$$\text{Throttle To Roll} = \frac{0.09752s^5 - 0.6509s^4 + 5911s^3 - 4.225\times10^5 s^2 + 3.426\times10^6 s + 7.309\times10^7}{s^5 + 50.8s^4 + 9.96\times10^4 s^3 + 9.152\times10^5 s^2 + 8.593\times10^6 s + 3.308\times10^7} \tag{4.41}$$

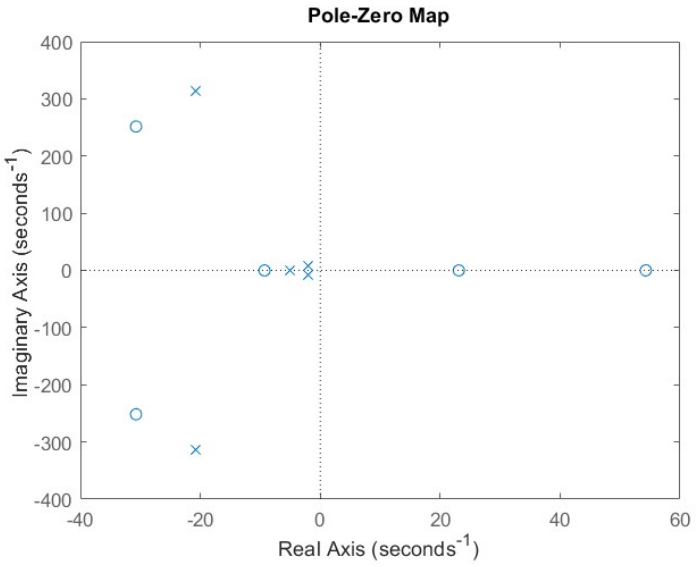


Figure 4.61: pole-zero plot of throttle-command to roll identified system.

Finally, the final MIMO system, based on the functions obtained from the system identification part, is as follows: For roll-command to vertical acceleration (A_z):

$$G_{12}(s) = \frac{0.01163s^4+5.317s^3-228.2s^2+2.633\times10^4 s-9.774\times10^4}{s^4+65.42s^3+8142s^2+2.638\times10^5 s+6.501\times10^5}$$

For throttle-command to roll:

$$G_{21}(s) = \frac{0.09752s^5-0.6509s^4+5911s^3-4.225\times10^5 s^2+3.426\times10^6 s+7.309\times10^7}{s^5+50.8s^4+9.96\times10^4 s^3+9.152\times10^5 s^2+8.593\times10^6 s+3.308\times10^7}$$

The complete MIMO plant transfer function matrix is:

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix}$$

where:

$$G_{11}(s) = \frac{0.03169s^4+0.2434s^3+1870s^2+2.788\times10^4 s+2.719\times10^6}{s^4+9.859s^3+5.091\times10^4 s^2+3.822\times10^5 s+2.094\times10^6}$$

$$G_{22}(s) = \frac{1.218s^4+13.2s^3+6.787\times10^4 s^2+4.13\times10^4 s+5.787\times10^5}{s^4+18.5s^3+5.369\times10^4 s^2+543.6s+5.87}$$

See Section 4.2.2 and 4.4.2.1.

The weighting matrices for sensitivity and complementary sensitivity are defined as:

$$W_s = \begin{bmatrix} W_{s1} & 0 \\ 0 & W_{s1} \end{bmatrix} \tag{4.42}$$

$$W = \begin{bmatrix} W_1 & 0 \\ 0 & W_2 \end{bmatrix} \tag{4.43}$$

4.5.1 Decentralized MIMO PID controller design optimized using $\mathscr{H}_\infty$ control synthesis

Linear controllers, particularly PID controllers, are widely used in aerospace systems due to their key advantages (Åström and Hägglund, 2006). Their straightforward design and ease of implementation have made them the default choice in numerous autopilot firmware systems (Beard and McLain, 2012).Despite their limitations in handling model uncertainties and external disturbances, the implementation of multiple PID controllers in a MIMO configuration, optimized using $\mathscr{H}_\infty$ control synthesis (via MATLAB's `hinfstruct` command), can achieve superior robustness and performance. This MIMO implementation can be approached in two ways: centralized and decentralized. A decentralized MIMO control system manages multiple inputs and outputs independently, assigning a dedicated controller to each input-output pair without direct communication between controllers (Wang et al., 2023b).

The primary characteristic of decentralized MIMO is that the overall control system is broken down into separate, autonomous control loops. Each controller focuses on managing its assigned input-output relationship, similar to having multiple single-loop controllers working in parallel (Kumar and Patel, 2022).

Recent Developments: Modern implementations have introduced several advancements (Chen et al., 2021). The integration with machine learning has enabled improved performance optimization across multiple control loops. Enhanced robustness

against system uncertainties has been achieved through advanced control algorithms. The development of adaptation mechanisms has improved system response to changing operating conditions. Furthermore, recent research has demonstrated better handling of interconnected system dynamics through sophisticated decentralized architectures.

Benefits and Limitations: This approach offers several benefits: it simplifies the design and implementation process, makes system maintenance more straightforward, reduces computational requirements, and provides better fault tolerance since a failure in one loop doesn't directly affect others (Wang et al., 2023b). However, it also comes with limitations, particularly in systems with strong interactions between variables, as the controllers cannot coordinate their actions to optimize overall system performance (Kumar and Patel, 2022).

Applications: Common industrial applications include process control systems, aircraft control, power distribution networks, and chemical manufacturing plants, where the complexity of the system can be effectively managed through this distributed control approach. Recent applications have expanded to include renewable energy systems, smart grids, and Industry 4.0 manufacturing processes (Chen et al., 2021). We now proceed with the design of the MIMO controller for the G_S system, optimized using $\mathscr{H}_\infty$ control synthesis.

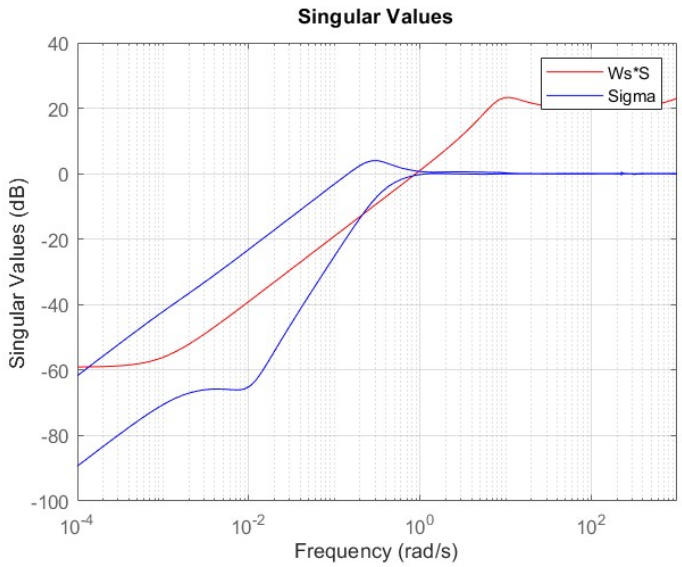


Figure 4.62: Design parameters of MIMO closed-loop for $\mathscr{H}_\infty$ in PID scenario.

Figure 4.62 presents the singular value decomposition (SVD) plot which contains two primary functions: the WS (weighted sensitivity) function shown in red, indicating the tracking performance and disturbance rejection capabilities, and the σ (complementary sensitivity) function displayed in blue, showing robustness and stability margins.

The frequency response spans from 10^{-4} to 10^{2} rad/s, revealing crucial system characteristics. At low frequencies, the design achieves excellent tracking performance with WS remaining below -40dB. The crossover frequency occurs at approximately 1 rad/s, while the high-frequency region demonstrates appropriate roll-off behavior to ensure noise rejection and robust performance. Figure 4.63 illustrates the multivariable step response matrix showing input-output relationships.

The diagonal elements (G_{11}, G_{22}) represent the primary control paths, exhibiting rise times of approximately 5-7 seconds with minimal overshoot below 10% and settling times around 25 seconds. The off-diagonal elements (G_{12}, G_{21}) represent the cross-coupling effects, showing peak interactions of approximately 20-30% of the steady-state value, with effective decoupling achieved through the controller design.

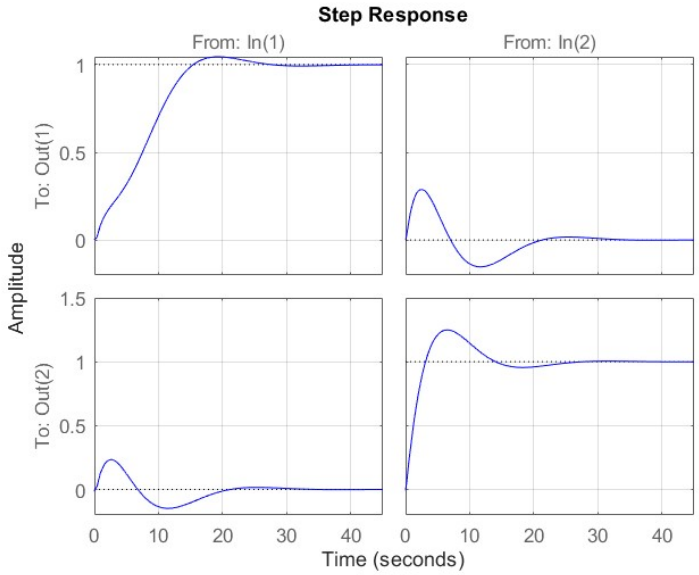


Figure 4.63: Step response of MIMO closed-loop system in PID scenario

The PID controller matrix takes the form:

$$K = \begin{bmatrix} \frac{0.0006383s^2+0.004702s+0.05744}{0.01s^2+s} & \frac{-5.099\times10^{-5}s^2+0.0005568s-2.674\times10^{-7}}{0.01s^2+s} \\ \frac{0.0005491s^2+0.003852s+0.05789}{0.01s^2+s} & \frac{2.045\times10^{-5}s^2+0.00163s+4.906\times10^{-8}}{0.01s^2+s} \end{bmatrix} \tag{4.44}$$

4.5.2 MIMO Mixed sensitivity controller design

Similar to the SISO case, an $\mathscr{H}_\infty$ sensitivity-based MIMO controller can be designed for the G(s) system. The design parameters are shown in Figure 4.64, where the optimal performance index γ_{opt} , or equivalently σ_{max} , is 3.2. While this value does not indicate ideal robustness, it remains within an acceptable range. The step response of

the MIMO controller, shown in Figure 4.65, reveals a noticeably slower response compared to the PID-based controller discussed earlier. This highlights that higher-order controllers do not necessarily yield better performance in terms of response speed.

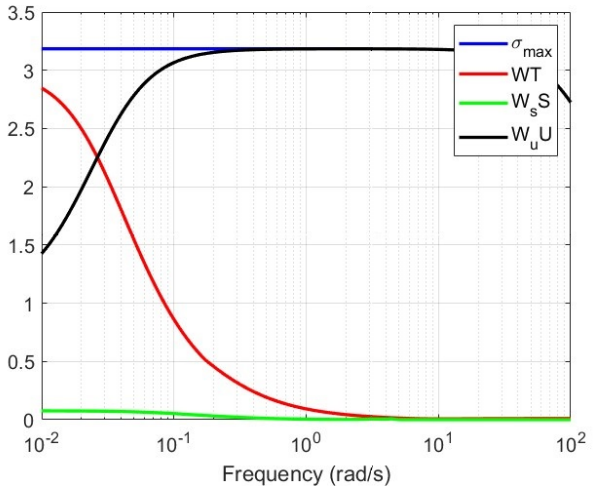


Figure 4.64: Design parameters of MIMO closed-loop for $\mathscr{H}_\infty$ in mixed sensitivity scenario.

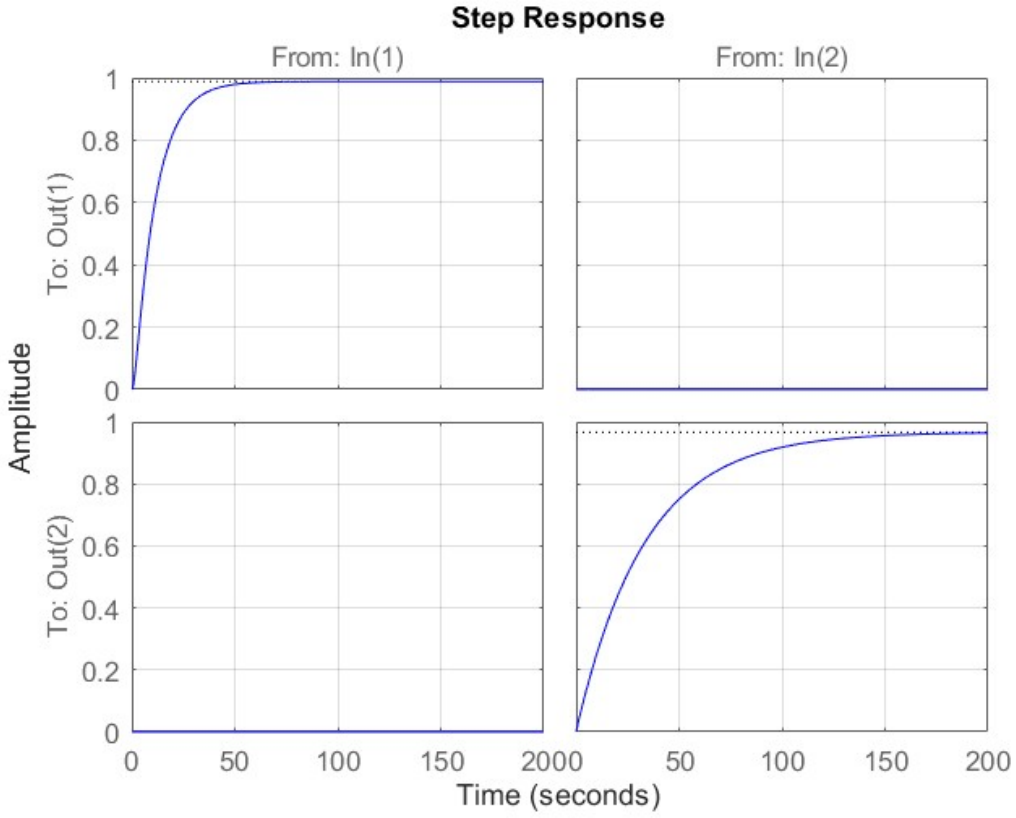


Figure 4.65: Step response of the controller designed in the $\mathscr{H}_\infty$ scenario for MIMO plant

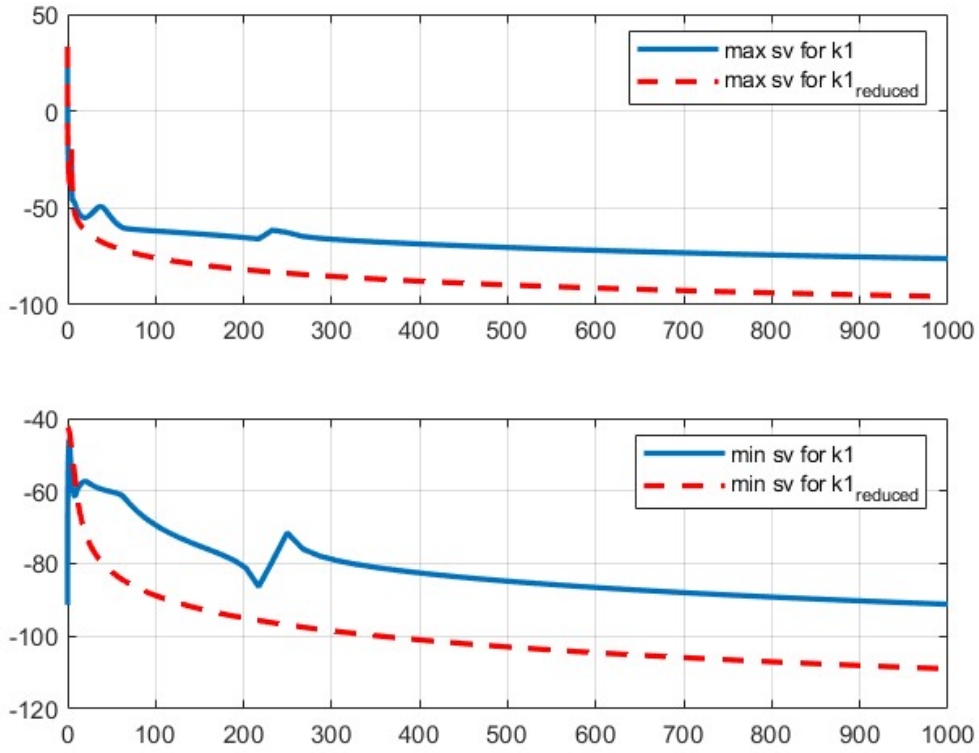


Figure 4.66: Min and max singular value comparison for controller and its reduced version.

The Figure 4.66 plot comprises two subplots that evaluate the effectiveness of controller reduction by comparing the frequency responses of the original controller K_1 with its reduced-order version $K_{1,\text{reduced}}$.

The upper subplot presents the maximum singular values of both controllers. The solid line represents the original controller (K_1), while the dashed line shows the reduced controller ($K_{1,\text{reduced}}$). The close correspondence between these curves, particularly in the high-frequency region, demonstrates that the reduced controller preserves the essential high-frequency characteristics of the original design.

The lower subplot displays the minimum singular values of both controllers, again comparing the original and reduced versions. The alignment between the solid and dashed curves indicates that the reduced controller maintains the fundamental dynamic properties of the original controller across the entire frequency range of interest (10^{-3} to 10^{3} rad/s). The bode diagram for MIMO designed controller $k(s) = \begin{bmatrix} k_{11}(s) & k_{12}(s) \\ k_{21}(s) & k_{22}(s) \end{bmatrix}$ is revealed in Figure 4.67:

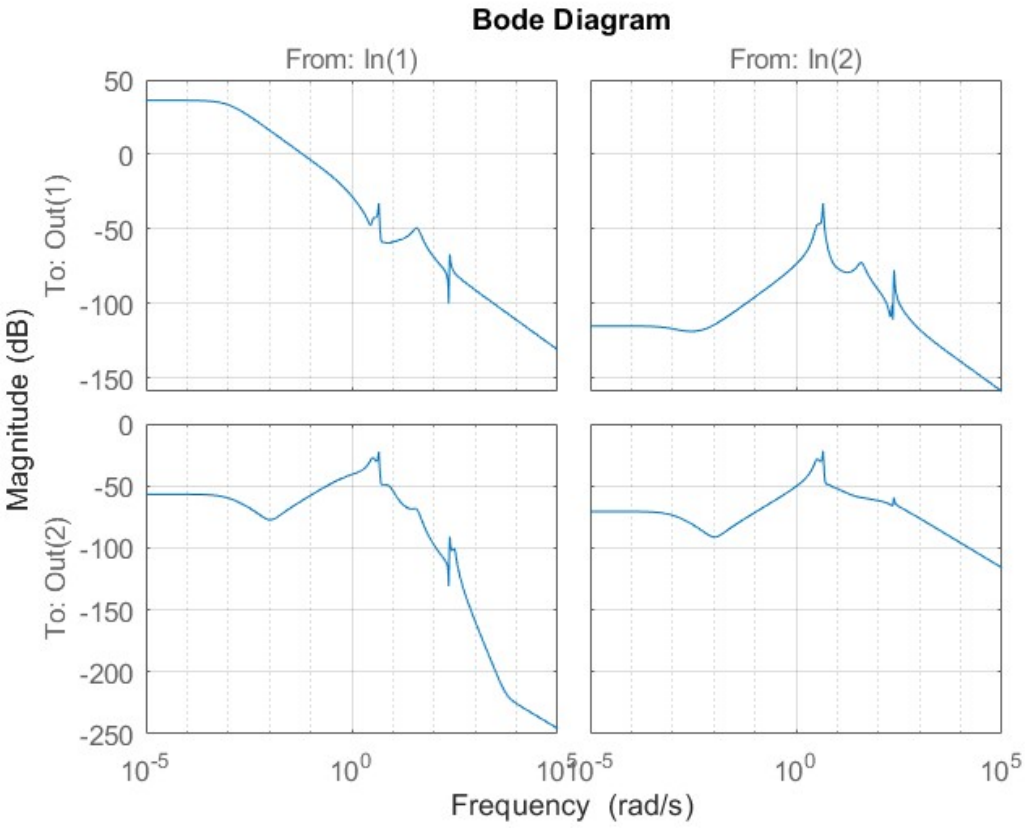


Figure 4.67: Frequency response of the controller designed in the $\mathscr{H}_\infty$ scenario for MIMO plant.

4.5.3 Roll-Yaw coupling MIMO system

It is worth noting, however, that fixed-wing VTOL aircraft face significant coupling between roll and yaw dynamics, particularly during hover and transition phases. This coupling arises from factors such as propeller slipstream effects, gyroscopic forces, and the influence of the center of gravity (Zhang et al., 2023). For example, roll commands achieved through differential thrust can induce yaw moments, while yaw commands implemented via elevon deflection can result in rolling moments. These interactions are further influenced by airspeed, especially during transitions between hovering and forward flight.

To address these challenges, the $\mathscr{H}_\infty$ mixed-sensitivity control framework is employed, offering a robust solution through detailed MIMO (multi-input multi-output) plant modeling and systematic treatment of uncertainties (Chen et al., 2024). This controller utilizes performance, uncertainty, and control effort weights to effectively balance tracking accuracy and robustness. Compared to traditional SISO (single-input single-output) designs, it demonstrates superior performance in maintaining transition stability, suppressing coupling effects, and rejecting disturbances.

4.5.4 System Identification

In this section, system identification is performed using real input-output data from the drone across five data categories, as shown in Table 4.2.

Table 4.2: Categories of Systems for Identification

Input	Output	Transfer Function
Yaw Command (ψ_c)	Yaw (ψ)	G_{11}
Roll Command (ϕ_c)	Yaw (ψ)	G_{12}
Yaw Command (ψ_c)	Roll (ϕ)	G_{21}
Roll Command (ϕ_c)	Roll (ϕ)	G_{22}
Roll (ϕ)	Tether Angle (λ)	G_3

In the first system, we mentioned again as in section(4.2.2) the yaw command (ψ_c) is used as the input, and the yaw (ψ) is the output. The Bode diagrams of the identified systems are shown in Figure 4.8. The nominal system is represented by Equation (4.45).

In the second system, where the roll command (ϕ_c) serves as the input and roll (ϕ) as the output, the Bode diagrams of the identified systems are shown in Figure 4.9 (see section (4.2.2)). The nominal system is represented by Equation (4.46).

$$G_{11}(s) = \frac{0.07344s^4 + 2.411s^3 + 3186s^2 + 5.632 \times 10^4 s + 1.948 \times 10^6}{s^4 + 21.08s^3 + 5.289 \times 10^4 s^2 + 6.193 \times 10^5 s + 1.912 \times 10^6} \tag{4.45}$$

$$G_{22}(s) = \frac{0.0317s^4 + 0.2434s^3 + 1870s^2 + 2.788 \times 10^4 s + 2.719 \times 10^6}{s^4 + 9.859s^3 + 5.091 \times 10^4 s^2 + 3.822 \times 10^5 s + 2.094 \times 10^6} \tag{4.46}$$

For the system with roll command to yaw ($\phi_c \rightarrow \psi$) and yaw command to roll ($\psi_c \rightarrow \phi$), the corresponding equations are given by (4.47) and (4.48), respectively.

$$G_{12}(s) = \frac{0.2751s^4 + 4.819s^3 + 1.663 \times 10^4 s^2 + 2.618 \times 10^5 s + 1.993 \times 10^5}{s^4 + 12.06s^3 + 6.343 \times 10^4 s^2 + 6.728 \times 10^5 s + 4.262 \times 10^6} \tag{4.47}$$

$$G_{21}(s) = \frac{0.06512s^4 + 0.3285s^3 + 4726s^2 + 4.85 \times 10^4 s + 9.515 \times 10^5}{s^4 + 65.9s^3 + 3.078 \times 10^4 s^2 + 3.674 \times 10^4 s + 1.054 \times 10^6} \tag{4.48}$$

Finally,

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \tag{4.49}$$

4.6 Mixed Sensitivity MIMO Controller Design

4.6.1 MIMO control scheme

The lateral position control is achieved through tether feedback, where the drone minimizes the tether angle deviation from 90 degrees through rolling maneuvers (Figure 4.1). The system uses roll (ϕ), pitch (θ), and yaw (ψ) angles for orientation, with tether angle (λ) providing position feedback.

Three primary control loops operate along the body's reference axis: roll control, yaw control, and Y-axis position control (via tether angle). The lateral control architecture employs a nested loop structure, where the outer loop maintains lateral position through tether angle control, generating roll reference (ϕ_{Ref}) for the inner loop. The IMU-AHRS sensor provides roll and yaw feedback.

Initial SISO implementation revealed roll-yaw coupling effects during flight tests, necessitating a MIMO control approach for system identification and control, as illustrated in Figure 4.68. The system dynamics are represented by transfer functions G_{11}, G_{12}, G_{21}, and G_{22}, with corresponding controllers k_{11}, k_{12}, k_{21}, and k_{22}.

The actuator control commands were defined by equations (4.1) and (4.4).

Although we performed identical SISO procedures for G_3, we have omitted its detailed description due to it being the same as section (4.2.2).

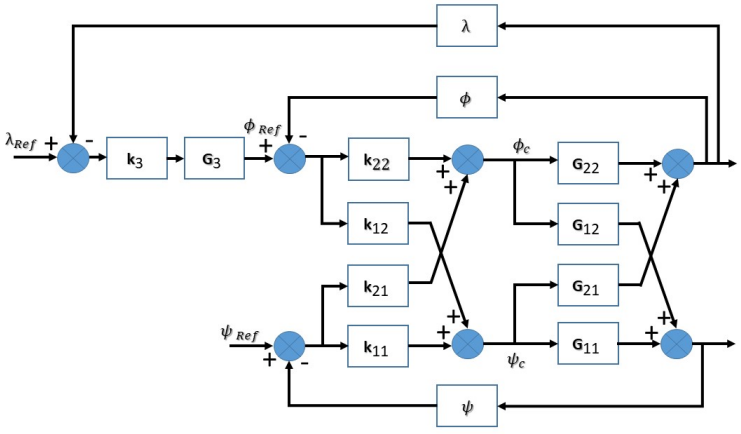


Figure 4.68: The MIMO control scheme with a controller on each channel.

4.6.2 Control Design

4.6.2.1 Uncertainty Profile Modeling

As previously discussed in that section (4.3.1), we are reiterating them here:

$$W_{T_1}(s) = 6\left(\frac{s+1.9}{s+14}\right) \tag{4.50}$$

$$W_{T_2}(s) = 110\left(\frac{s+0.7}{s+77}\right) \tag{4.51}$$

As shown in all the figures, the chosen W_{T} effectively encompasses nearly all uncertainties within the system. For the MIMO design, we have:

$$W_T = \begin{bmatrix} W_{T1} & 0 \\ 0 & W_{T2} \end{bmatrix} \tag{4.52}$$

4.6.2.2 Control Design

The weight functions W_s for all three systems are defined in equations 4.53 to 4.54 same in SISO design.

$$W_{s1} = 0.3\frac{s^2+9s+81}{(s+0.001)(s+9)(0.001s+1)} \tag{4.53}$$

$$W_{s2} = 0.2\frac{s^2+9s+81}{(s+0.001)(s+9)(0.001s+1)} \tag{4.54}$$

For MIMO design we define:

$$W_s = \begin{bmatrix} W_{s1} & 0 \\ 0 & W_{s2} \end{bmatrix} \tag{4.55}$$

And also:

$$\mathrm{W_u} = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix} \tag{4.56}$$

The diagrams illustrating the design parameters for all systems are presented in Figure 4.69. In these figures, γ_{opt} is represented as sigma(σ_{max}). As can be seen, it's completely flat, near 1. It is evident that Equation (4.15) is valid across all scenarios.

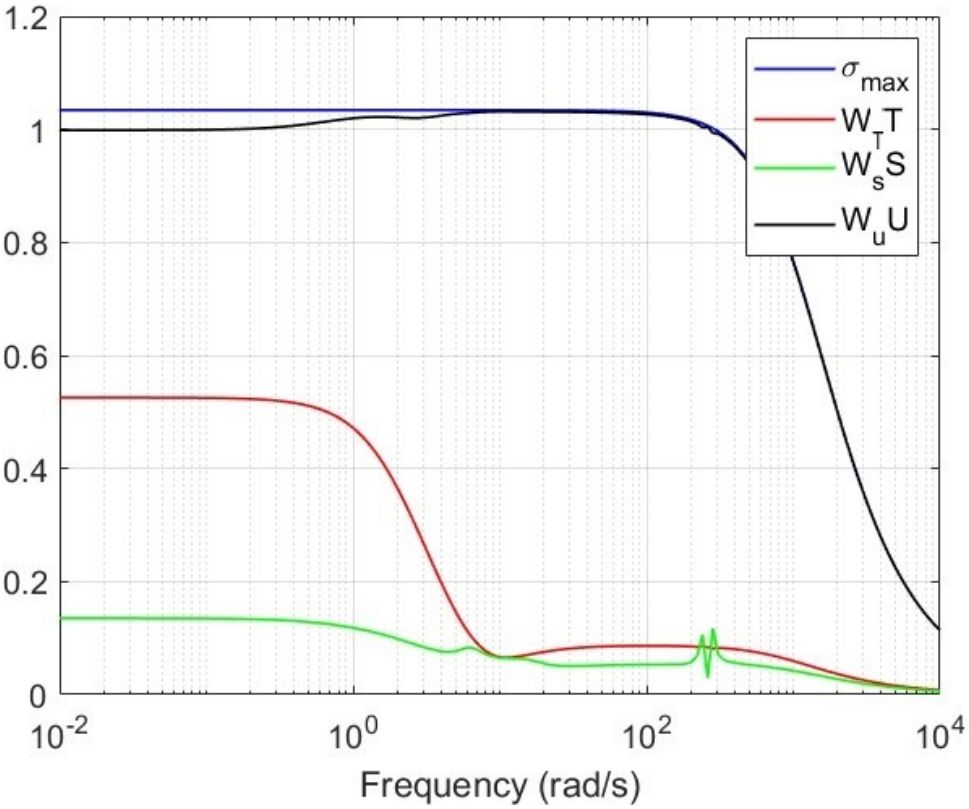


Figure 4.69: Design parameters of systems for $\mathscr{H}_\infty$ in mixed sensitivity scenario.

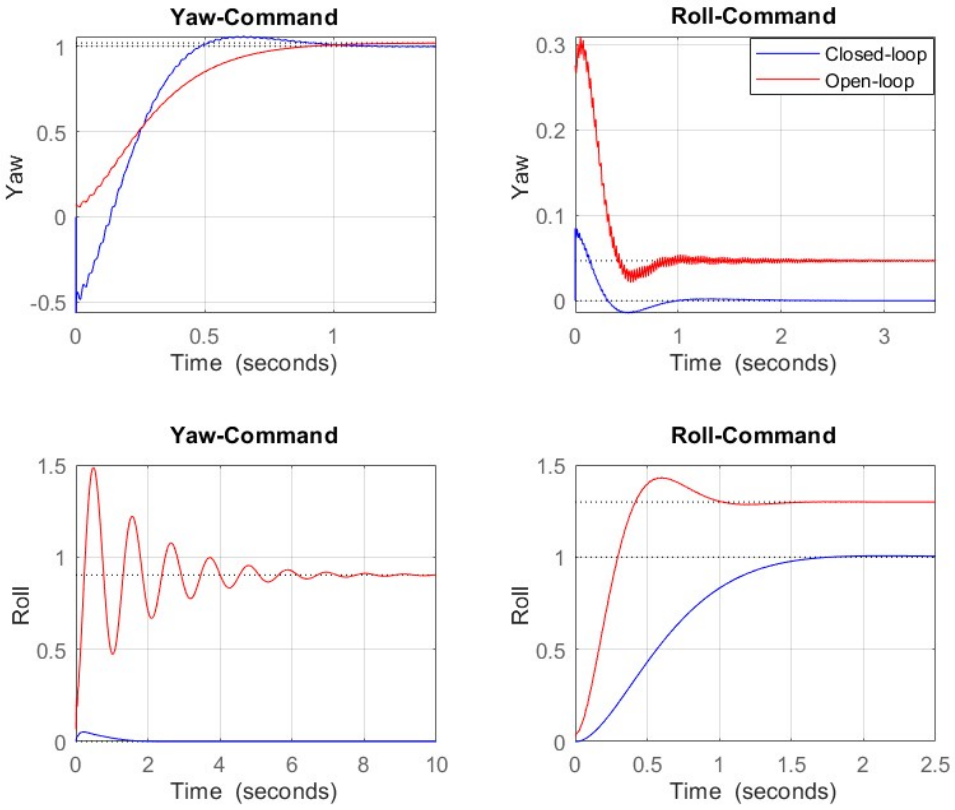


Figure 4.71: The step responses of the opened-loop and the closed-loop system.

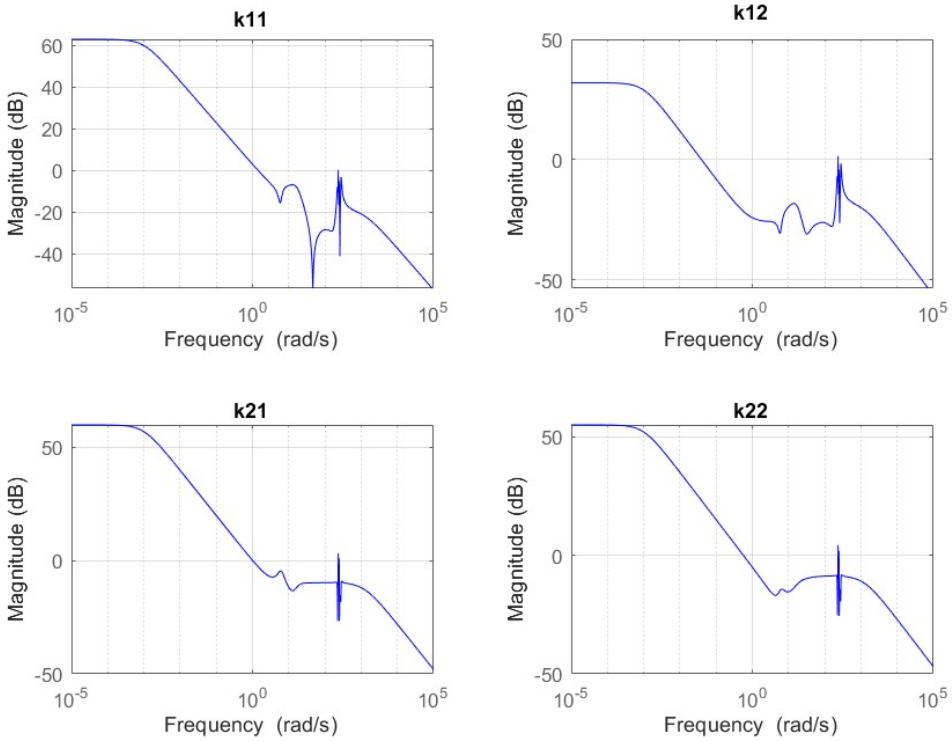


Figure 4.70: The bode diagram for
MIMO designed controller $k(s) = \begin{bmatrix} k_{11}(s) & k_{12}(s) \\ k_{21}(s) & k_{22}(s) \end{bmatrix}$.

The values of γ were designed to be well-suited for practical use. The frequency response plots (k_{11}, k_{12}, k_{21}, and k_{22}) in Figure 4.70 demonstrate the MIMO controller's effectiveness across different frequency ranges, where the magnitude curves show proper roll-off at high frequencies and good tracking at low frequencies, supporting a very suitable step response without overshooting and with a settling time of approximately one second and can suppress cross-coupling effects rapidly which is illustrated in Figure 4.71. The figure illustrates how the open-loop response (red line) performs better compared to the closed-loop response (blue line) with the implemented controllers.

The designed controllers typically have orders ranging from 8 to 9, which are reduced to 4 or 5 using a reduction function.

Through a detailed system identification process and the design of robust mixed-sensitivity MIMO controllers, the system effectively addresses coupling effects and uncertainties. Experimental flight tests confirm the system's ability to maintain stable tethered navigation, manage disturbances, and ensure accurate control of the tether angle and yaw heading under varying conditions.

4.7 Concluding remarks

In this chapter, we presented the development and analysis of robust control strategies for autonomous VTOL drone systems, with a particular focus on path following via a tethered connection to a ground robot, while leveraging wind energy for improved efficiency. The research progressed from single-channel robust controller design to addressing the more complex challenges of multi-input multi-output (MIMO) control inherent in VTOL systems.

The adopted methodology reflects practices commonly used in advanced aerospace projects: initial controller design for basic stability, followed by flight data acquisition and system identification using advanced frequency-domain techniques. This enabled the construction of a heuristic uncertainty profile for the system. Finally, the controller was redesigned using advanced robust control methods to optimize performance across all key parameters.

Throughout this process, a wide range of robust control techniques were explored and applied—including $\mathscr{H}_2$, $\mathscr{H}_\infty$, and mixed-sensitivity analysis and synthesis—within both single-input single-output (SISO) and multi-input multi-output (MIMO) cascade loop controller structures.

The designed controllers typically have orders ranging from 8 to 9, which are reduced to 4 or 5 using a reduction function. Validation of the controllers is performed through μ synthesis. Subsequently, these controllers are implemented in an actual autopilot, and the drone is flown automatically.

Chapter 5

Conclusions and Future Work

This thesis adopts a dual-track methodology for the development of an air vehicle control system, integrating both theoretical analysis and practical implementation. The research combines design, construction, simulation, and validation efforts, with each component informing and enhancing the others. The focus is on designing, testing, and fine-tuning control loops for the air vehicle system, categorized below.

5.1 Satisfaction of Objectives

The VTOL fixed-wing aircraft was successfully flown in this project after undergoing rigorous testing using GPS and tethered control and navigation in both outdoor and lab settings. These successful tests confirmed the correct motor and propeller selection, wing design, fuselage and wing structural integrity, wing dimensioning, elevon control surface configuration, and their corresponding mounting positions, as well as the true thrust-to-weight ratio at all flying angles. Additionally, it verified that the center of mass and the center of aerodynamic pressure were positioned correctly and that the servo actuators were chosen appropriately. This is an all-encompassing aerospace project.

The autopilot implementation included proper setup of the IMU package, accurate parameter selection, calibration, mounting, and effective damping of engine vibrations. Hundreds of lines of C code were written and tested within the Linux operating system environment, with successful operation validating the control code and control loops. Proper wiring implementation and appropriate selection of modules and sensors contributed to successful flight operations.

The project's primary achievement was accomplishing a fully automated flight while maintaining control of the aircraft, notably with both the aircraft and autopilot being original designs. While previous work had addressed hovering flight and autopilot tuning, the introduction of tethered navigation represented a novel and technically demanding aspect. As an untested approach, it initially posed several challenges, which

were progressively addressed through iterative development and comprehensive testing.

The analysis of quadcopter dynamic models yielded insights that facilitated the derivation of complete dynamic model equations for the aircraft. Physical parameters were determined using *SolidWorks*, forming the basis for the development of a nonlinear sliding mode controller. These components were implemented and simulated in MATLAB, producing satisfactory results.

Building on prior work involving data acquisition systems and system identification for UAVs and unmanned helicopter platforms, the implementation of robust control strategies—such as $\mathscr{H}_2$, $\mathscr{H}_\infty$, and μ-synthesis—was further refined. Improvements over previous results were achieved, and novel methodologies were introduced, including the successful testing of MIMO controllers.

The comprehensive data acquisition and analysis conducted on these unmanned systems produced promising outcomes, with potential for dissemination in academic publications.

This project represents an initial step toward achieving significant advances in highly mobile VTOL tethered systems, with applications in both military and commercial sectors.

5.2 Future Works

For the continuation of this work, the implementation phase should involve the construction of a new drone using advanced materials. Specifically, replacing cardboard with carbon fiber reinforced by 3D-printed molds is recommended to ensure structural integrity and appropriate lift capacity, while maintaining suitable aircraft dimensions. In the attitude and heading reference system (AHRS) of the autopilot, a low-cost inertial measurement unit (IMU) should be employed in place of commercial modules. Sensor fusion algorithms—such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF)—should be implemented directly on the flight computer to enhance performance and flexibility. In the sliding mode control section, future work should incorporate external disturbances and model uncertainties. Additionally, the dynamic model should be extended to include the effects of tethering, with corresponding updates to the control design and evaluation of its performance. For aerodynamic force modeling, computational fluid dynamics (CFD) methods and specialized software can be utilized to improve accuracy. Due to time constraints, the nonlinear sliding mode controller was not implemented or tested on the physical aircraft; this practical validation remains a critical objective for future research.

Appendix A

PID Control and Simulation

A.1 Euler Method

Euler's method is a simple numerical method for solving ordinary differential equations, where the changes in the dependent variable with respect to time are approximated using the tangent slope at each time step. In this method, the value of the state variable at each time step is calculated using its previous value and its derivative. The basic formula for Euler's method is:

$$x(t+\Delta t) = x(t) + \Delta t \cdot f(x(t),t) \tag{A.1}$$

In this formula, $x(t)$ is the state variable at time t, Δt is the time step, and $f(x(t),t)$ is the derivative function or the rate of change of the state variable. This method is useful in many numerical simulations and dynamic system modeling due to its simplicity and computational speed, but its accuracy is limited, and for complex systems, more advanced methods like Runge-Kutta may be necessary.

A.2 Theoretical Foundation of Ziegler-Nichols Method

A.2.1 Basic PID Controller Structure

The continuous-time PID controller has the form:

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int e(t)dt + T_d \frac{d}{dt} e(t) \right] \tag{A.2}$$

Here $u(t)$ is the control signal, $e(t)$ is the error signal, K_p is the proportional gain, T_i is integral time, and T_d is derivative time.

A.2.2 Ziegler-Nichols First Method

Based on the open-loop step response:

$$K_p = \frac{1.2}{RL} \tag{A.3}$$

$$T_i = 2L \tag{A.4}$$

$$T_d = 0.5L \tag{A.5}$$

Where R is the process reaction rate and L is the process dead time.

A.2.3 Ziegler-Nichols Second Method

Based on closed-loop oscillation:

$$K_p = 0.6K_u \tag{A.6}$$

$$T_i = 0.5T_u \tag{A.7}$$

$$T_d = 0.125T_u \tag{A.8}$$

Where K_u is the ultimate gain and T_u is the oscillation period.

A.2.4 Safety Precautions

Before starting the tuning process:
Sufficient altitude (minimum 1000 ft AGL) Clear airspace Good weather conditions Stable flight conditions Emergency manual control ready

A.2.5 Step-by-Step Tuning Process

A.2.5.1 Initial Proportional Gain Tuning

Start with very low K_p (around 0.1) Gradually increase K_p until the aircraft shows visible oscillations Look for 1 Hz oscillation (one complete cycle per second) When oscillations are observed, record this value as K_u (ultimate gain) Reduce K_p to 60

$$K_p = 0.6 \times K_u \tag{A.9}$$

A.2.5.2 Integral Gain Tuning

Start with $K_i = 0$ Slowly increase K_i until oscillations diminish Watch for reduced altitude hunting, stable flight path, and no overcompensation Calculate:

$$K_i = \frac{K_p}{0.5T_u} \tag{A.10}$$

A.2.5.3 Derivative Gain Tuning

Begin with $K_d = 0$ Gradually increase K_d to improve response speed and Stop when a quick response is achieved with no overshoot then Calculate:

$$K_d = 0.125 \times K_p \times T_u \tag{A.11}$$

A.2.6 Visual Indicators During Tuning

For oscillation recognition the 1 Hz oscillation appears as visible up/down motion in the aircraft, one complete cycle per second(for example, in the roll angle of an airplane when the wing of the airplane moves up and down.), easily countable visually.

Appendix B

Quaternion-Based Extended Kalman Filter implementation for Orientation Estimation using Accelerometer, Gyroscope, and Magnetometer Data

B.1 Introduction

The orientation of an object is a crucial parameter in many fields such as aerospace engineering, robotics, and virtual reality. Inertial Measurement Units (IMUs) are commonly used to determine the orientation of an object in a dynamic environment. IMUs consist of accelerometers and gyroscopes that measure linear acceleration and angular velocity, respectively. However, the measurements from these sensors are prone to errors like noise and drift, which can significantly impact the accuracy of the orientation estimation. To tackle this issue, several filtering techniques, including the Extended Kalman Filter (EKF), have been proposed.

In the case of autopilot systems, Attitude and Heading Reference Systems (AHRS) are the primary components to ensure a stable flight path. Developing an AHRS system in-house can reduce costs and help achieve a versatile autopilot system that meets specific needs.

This appendix proposes an EKF algorithm based on quaternion rotation to estimate the orientation of an object in a dynamic environment in real-time using a microcontroller with a real IMU sensor. The algorithm combines the accelerometer and gyroscope data to estimate the orientation using a state estimation approach. The algorithm is divided

into two stages: the first Kalman filter estimates Roll and Pitch angles, while the second filter estimates Heading or Yaw angles using magnetometer data. The BNO055 sensor is a 9-axis sensor that combines a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis magnetometer into a single package, which communicates over I2C. This sensor provides precise and reliable orientation data for various applications. The ESP8266 mini microcontroller is used in conjunction with the BNO055 sensor to estimate the orientation using an extended Kalman filter. The filter results are compared to the BNO055's built-in orientation estimation to evaluate their accuracy and reliability for the specific application (Farahan et al., 2022) and (Suwandi et al., 2017). However, in field tests, it was found that the speed the sensor output shows is different from the actual dynamics it can track, and the sensor is slower and did not perform well in closed loop mode(autopilot), where AHRS is tested in real tests. But the raw data from gyroscopes, accelerometers, and magnetometers was updating at a high rate, so we thought we could combine this information ourselves and obtain attitude and heading angles.

B.2 Hardware

To begin with, the BNO055 serves as the IMU, and the ESP8266 Mini is utilized for processing, as mentioned in the project's introduction. At first, an Arduino Uno was utilized. However, after defining a variable, it became evident that the RAM memory was limited (at 32KB) because the extended Kalman filter required the definition of some large dimension matrices. Therefore, a more powerful processor was required. The ESP8266 Mini, with 4MB of RAM and WIFI for potential future development, was deemed a better option. As mentioned before for fast communication we use I2C communication. As it is shown in Figure B.1 and Figure B.2.

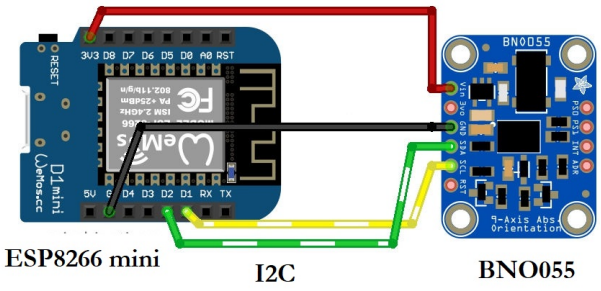


Figure B.1: The schematic of hardware assembly and wiring.

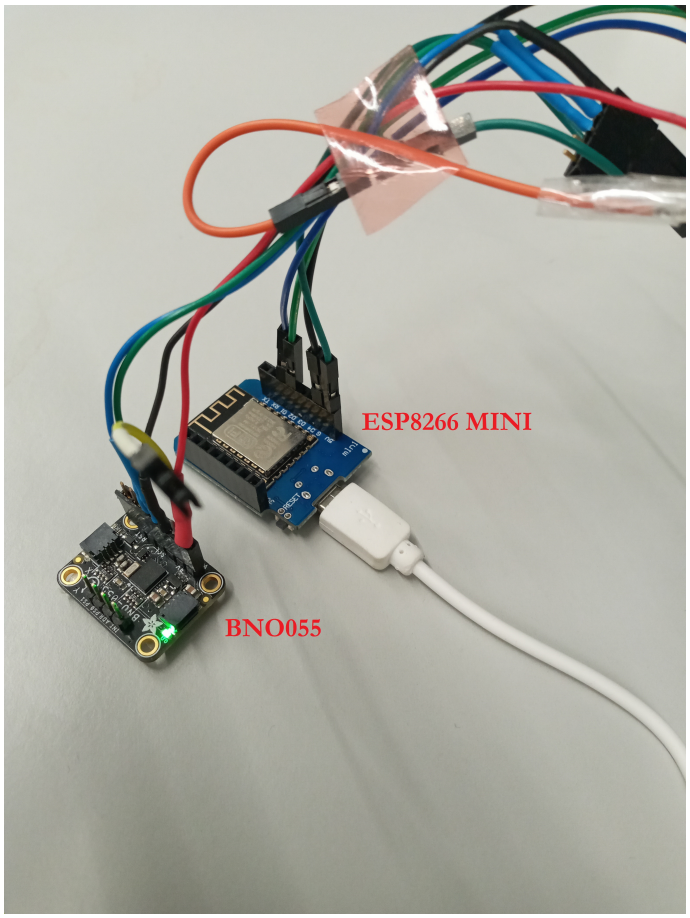


Figure B.2: Boards and wiring.

B.3 Quaternion Representation of Orientation

A quaternion is a mathematical object that provides a compact representation of orientation. A quaternion consists of a scalar component q_0 and a vector component $\mathbf{q} = [q_1, q_2, q_3]$, where i, j, and k are imaginary units that satisfy the relations $i^2 = j^2 = k^2 = ijk = -1$. The quaternion is defined as follows:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{B.1}$$

A quaternion can be used to represent the orientation of a body by defining a rotation matrix $\mathbf{R}$ in quaternion units. The rotation matrix $\mathbf{R}$ is defined as follows(Farahan et al., 2022):

$$\mathbf{R} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{B.2}$$

The rotation matrix $\mathbf{R}$ can be used to transform a vector $\mathbf{v}$ from the body frame to the reference frame as follows:

$$\mathbf{v}_{\text{ref}} = \mathbf{R}\mathbf{v}_{\text{body}} \tag{B.3}$$

The angular velocity can be defined as a vector $\omega = [\omega_x, \omega_y, \omega_z]$, where ω_x, ω_y, and ω_z are the angular velocities around the x, y, and z axes, respectively. The quaternion derivative can be computed using the angular velocity as follows(Leccadito, 2013):

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\omega)\mathbf{q} \tag{B.4}$$

where $\Omega(\omega)$ is the matrix(Watson, 2013):

$$\Omega(\omega) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{B.5}$$

To convert from a quaternion to Euler angles, the following equations can be used:

$$\phi = \text{atan2}\left(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)\right) \tag{B.6}$$
$$\theta = \text{asin}\left(2(q_0q_2 - q_3q_1)\right) \tag{B.7}$$
$$\psi = \text{atan2}\left(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)\right) \tag{B.8}$$

where ϕ, θ, and ψ are the Euler angles around the x, y, and z axes, respectively.
To convert from Euler angles to a quaternion, the following equations can be used:

$$q_0 = \cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \tag{B.9}$$
$$q_1 = \sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \tag{B.10}$$
$$q_2 = \cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \tag{B.11}$$
$$q_3 = \cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) \tag{B.12}$$

B.4 The Extended Kalman Filter

The Extended Kalman Filter (EKF) is a recursive algorithm that is widely used for state estimation in nonlinear systems. It is an extension of the Kalman Filter, which is designed for linear systems. The EKF can be used to estimate the state of a system,

which is not fully observable. The algorithm uses a model of the system to predict the state, and it uses measurements to correct the predicted state. The EKF is based on Bayes' rule, which is a fundamental concept in probability theory.

Formulation:

Consider a nonlinear system that can be described by the following state equation:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1} \tag{B.13}$$

where $\mathbf{x}_k$ is the state vector at time k, $\mathbf{u}_{k-1}$ is the control input vector at time $k-1$, $f(\cdot)$ is the nonlinear state transition function, and $\mathbf{w}_{k-1}$ is the process noise. The process noise is assumed to be Gaussian with zero mean and covariance matrix $\mathbf{Q}_{k-1}$.

At time k, the EKF predicts the state vector using the following equation:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}) \tag{B.14}$$

where $\hat{\mathbf{x}}_{k-1|k-1}$ is the predicted state vector at time $k-1$. The EKF also predicts the error covariance matrix using the following equation:

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \tag{B.15}$$

where $\mathbf{P}_{k-1|k-1}$ is the error covariance matrix at time $k-1$, and $\mathbf{F}_{k-1}$ is the Jacobian matrix of $f(\cdot)$ evaluated at $\hat{\mathbf{x}}_{k-1|k-1}$.

At time k, the EKF updates the predicted state vector and error covariance matrix using the measurement $\mathbf{z}_k$ and the measurement model:

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{B.16}$$

where $h(\mathbf{x}_k)$ is the nonlinear measurement function, and $\mathbf{v}_k$ is the measurement noise. The measurement noise is assumed to be Gaussian with zero mean and covariance matrix $\mathbf{R}_k$.

The EKF computes the Kalman gain matrix $\mathbf{K}_k$ as follows:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}k^T(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{B.17}$$

where $\mathbf{H}_k$ is the Jacobian matrix of $h(\cdot)$ evaluated at $\mathbf{x}_k$, which is the a priori state estimate at time step k based on the predicted state at time step $k-1$.

The EKF then updates the state estimate based on the measurement $\mathbf{z}_k$:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1})) \tag{B.18}$$

and updates the error covariance matrix as follows:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{B.19}$$

where $\mathbf{I}$ is the identity matrix. The EKF repeats this process at each time step, predicting the state and updating it based on measurements. The EKF can be used to estimate the states of nonlinear dynamic systems as long as the nonlinear measurement and process models can be linearized around the current state estimate (Farahan et al., 2022), (Suwandi et al., 2017).

B.5 Quaternion-based Extended Kalman Filter Algorithm

The Extended Kalman Filter algorithm used in this project is based on quaternion representations of orientation. The state of the system is represented by a quaternion vector $\mathbf{q} = [q_0\ q_1\ q_2\ q_3]^T$, where q_0 is the scalar component and q_1, q_2, q_3 are the vector components. The quaternion vector represents the orientation of the IMU with respect to an inertial frame of reference.

The dynamic model of the system is described by the following set of equations (Farahan et al., 2022):

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega} \tag{B.20}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{\alpha} + \mathbf{J}(\boldsymbol{\omega})\mathbf{b} \tag{B.21}$$

$$\dot{\mathbf{b}} = \mathbf{0} \tag{B.22}$$

where $\boldsymbol{\omega} = [p\ q\ r]^T$ is the angular velocity vector measured by the gyroscope, $\boldsymbol{\alpha} = [a_x\ a_y\ a_z]^T$ is the specific force vector measured by the accelerometer, $\mathbf{b} = [b_x\ b_y\ b_z]^T$ is the bias vector, $\otimes$ denotes the quaternion product, and $\mathbf{J}(\boldsymbol{\omega})$ is the Jacobian matrix defined as:

$$\mathbf{J}(\boldsymbol{\omega}) = \frac{\partial \mathbf{f}}{\partial \boldsymbol{\omega}}\bigg|_{\boldsymbol{\omega}=\mathbf{0}} = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \tag{B.23}$$

The measurement model of the system is described by the following set of equations:

$$\mathbf{y}_a = \mathbf{R}_{wb}^T\mathbf{a} + \nu_a \tag{B.24}$$

$$\mathbf{y}_m = \mathbf{R}_{wb}^T\mathbf{m} + \nu_m \tag{B.25}$$

where $\mathbf{a} = [a_x\ a_y\ a_z]^T$ is the specific force vector measured by the accelerometer, $\mathbf{m} = [m_x\ m_y\ m_z]^T$ is the magnetic field vector measured by the magnetometer, $\mathbf{R}_{wb}$ is the rotation matrix that transforms a vector from the body frame to the world frame, and ν_a and ν_m are the measurement noises.

The EKF algorithm consists of the following steps:

1. Prediction: Predict the state vector $\hat{\mathbf{x}}_k$ and the error covariance matrix $\mathbf{P}_k$ at time k using the dynamic model:

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) \tag{B.26}$$

2. Initialization:

$$\hat{q}_0 = [1\ 0\ 0\ 0]^T,\ \hat{P}_0 = \begin{pmatrix} \sigma_0^2 & 0 & 0 & 0 \\ 0 & \sigma_0^2 & 0 & 0 \\ 0 & 0 & \sigma_0^2 & 0 \\ 0 & 0 & 0 & \sigma_0^2 \end{pmatrix} \tag{B.27}$$

   where $\hat{q}_0$ is the initial estimate of the quaternion, $\hat{P}_0$ is the initial covariance matrix, and σ_0 is the initial standard deviation.

3. Prediction:

$$\hat{q}_k^- = \hat{q}_{k-1} \otimes \hat{q}_{k-1}(\Omega_k - b_{k-1})\Delta t \tag{B.28}$$

$$\hat{P}_k^- = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + Q_{k-1} \tag{B.29}$$

   where $\otimes$ denotes the quaternion product, Ω_k is the angular velocity measurement, b_{k-1} is the bias estimate at time $k-1$, Δt is the time step, F_{k-1} is the state transition matrix, and Q_{k-1} is the process noise covariance matrix.

4. Update:

$$\tilde{z}_k = h(\hat{q}_k^-) \tag{B.30}$$

$$H_k = \frac{\partial h}{\partial \hat{q}}\bigg|\hat{q}_k^- \tag{B.31}$$

$$K_k = \hat{P}_k^- H_k^T \left(H_k \hat{P}_k^- H_k^T + R_k\right)^{-1} \tag{B.32}$$

$$\hat{q}_k = \hat{q}_k^- \otimes K_k(\tilde{z}_k - z_k) \tag{B.33}$$

$$\hat{P}_k = (I - K_k H_k)\,\hat{P}_k^- \tag{B.34}$$

where $\tilde{z}_k$ is the predicted measurement, h is the measurement function, H_k is the measurement Jacobian, R_k is the measurement noise covariance matrix, and z_k is the actual measurement.

This formulation represents the basic steps of the Quaternion-based Extended Kalman Filter algorithm, which is used to estimate the orientation of an IMU based on accelerometer and gyroscope data in a dynamic environment.

B.6 implementation and details

The Arduino IDE was chosen for implementing the algorithm. The implementation involves the development of over 1200 lines of C code, which includes complex matrix operations. To enhance performance and optimize computational load, some methods are applied, especially for matrix multiplication and inversion. As a result of these optimizations, the output from the Kalman filter can now be obtained at a rate exceeding 200 Hz. From the algorithm part, The Quaternion-based Extended Kalman Filter (EKF) algorithm is commonly used for orientation estimation in Attitude and Heading Reference Systems (AHRS). In this algorithm, the state of the system is represented by a quaternion vector and the bias of rate gyroscopes.

The vector of the state x is defined as (Leccadito, 2013):

$$x = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 & \omega_{xb} & \omega_{yb} & \omega_{zb} \end{bmatrix}^T \tag{B.35}$$

In this definition, q_0, q_1, q_2, and q_3 are the parameters of the quaternion vector, and ω_{xb}, ω_{yb}, and ω_{zb} are the bias of the rate gyroscopes of the x, y, and z axes.

The function $f(x_{k-1}, u_k)$ can be expressed as:

$$f(x_{k-1}, u_k) = \begin{bmatrix} \frac{1}{2} q_{k-1} \otimes \begin{bmatrix} 0 \\ \omega_k \end{bmatrix} \\ \omega_{xb} \\ \omega_{yb} \\ \omega_{zb} \end{bmatrix} \tag{B.36}$$

Here, $\otimes$ denotes the operator for quaternion multiplication, and ω_k represents the angular velocity measured by gyroscopes. Further details of this implementation are

explained in the following equation (Watson, 2013):

$$\dot{q}_{\omega}(q,\omega) = \frac{1}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix}\begin{bmatrix} \omega_x - \omega_{xb} \\ \omega_y - \omega_{yb} \\ \omega_z - \omega_{zb} \end{bmatrix} \tag{B.37}$$

Finally (Watson, 2013):

$$x_k = f(x_{k-1}, u_k) = \begin{bmatrix} q_0 + \frac{dt}{2} * \left(-q_1(\omega_x - \omega_{xb}) - q_2\left(\omega_y - \omega_{yb}\right) - q_3(\omega_z - \omega_{zb})\right) \\ q_1 + \frac{dt}{2} * \left(q_0(\omega_x - \omega_{xb}) + q_3\left(\omega_y - \omega_{yb}\right) - q_2(\omega_z - \omega_{zb})\right) \\ q_2 + \frac{dt}{2} * \left(-q_3(\omega_x - \omega_{xb}) + q_0\left(\omega_y - \omega_{yb}\right) + q_1(\omega_z - \omega_{zb})\right) \\ q_3 + \frac{dt}{2} * \left(q_2(\omega_x - \omega_{xb}) - q_1\left(\omega_y - \omega_{yb}\right) + q_0(\omega_z - \omega_{zb})\right) \\ \omega_{xb} \\ \omega_{yb} \\ \omega_{zb} \end{bmatrix} \tag{B.38}$$

With the Jacobian of the vector function $f(x_{k-1}, u_k)$, the matrix F is computed as follows:

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1}, u_k} = \begin{bmatrix} F_{q_{k-1}} & F_{\omega_k} \\ \mathbf{0}_{3\times 4} & \mathbf{I}_{3\times 3} \end{bmatrix} \tag{B.39}$$

where $F_{q_{k-1}}$ and F_{ω_k} are the Jacobian matrices of f with respect to the quaternion vector and the angular velocity, respectively. $\mathbf{0}_{3\times 4}$ and $\mathbf{I}_{3\times 3}$ denote the 3×4 zero matrix and the 3×3 identity matrix, respectively which is finalized:

$$\frac{\partial f}{\partial x} = F = \begin{bmatrix} 1 & -\frac{dt}{2}(\omega_x - \omega_{xb}) & -\frac{dt}{2}\left(\omega_y - \omega_{yb}\right) & -\frac{dt}{2}(\omega_z - \omega_{zb}) & \frac{dt}{2}q_1 & \frac{dt}{2}q_2 & \frac{dt}{2}q_3 \\ \frac{dt}{2}(\omega_x - \omega_{xb}) & 1 & -\frac{dt}{2}(\omega_z - \omega_{zb}) & \frac{dt}{2}\left(\omega_y - \omega_{yb}\right) & -\frac{dt}{2}q_0 & -\frac{dt}{2}q_3 & \frac{dt}{2}q_2 \\ \frac{dt}{2}\left(\omega_y - \omega_{yb}\right) & \frac{dt}{2}(\omega_z - \omega_{zb}) & 1 & -\frac{dt}{2}(\omega_x - \omega_{xb}) & \frac{dt}{2}q_3 & -\frac{dt}{2}q_0 & -\frac{dt}{2}q_1 \\ \frac{dt}{2}(\omega_z - \omega_{zb}) & -\frac{dt}{2}\left(\omega_y - \omega_{yb}\right) & \frac{dt}{2}(\omega_x - \omega_{xb}) & 1 & -\frac{dt}{2}q_2 & \frac{dt}{2}q_1 & -\frac{dt}{2}q_0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.40}$$

In the Kalman prediction stage, using the angular velocity parameters (output of rate gyroscopes) and through the function $f(x_{k-1}, u_k)$, the vector x is updated, and then by forming the matrix F and using the following equations, the covariance matrix P is updated:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{B.41}$$

where Q_k is the process noise covariance matrix.

In the Measurement step, the z-measurement vector encompasses the accelerometer

and magnetometer sensors data and is defined as (Madgwick, 2010):

$$z = \begin{bmatrix} a_x & a_y & a_z & m_x & m_y & m_z \end{bmatrix}^T$$

The first step in the measurement process involves using data from accelerometers. Before using this data, it must be normalized using the following equation to find the norm:

$$\text{Norm} = \sqrt{A_x^2 + A_y^2 + A_z^2} \tag{B.43}$$

Then, the accelerometer data is normalized by dividing each component by the norm:

$$A_x = \frac{A_x}{\text{Norm}} \tag{B.44}$$

$$A_y = \frac{A_y}{\text{Norm}} \tag{B.45}$$

$$A_z = \frac{A_z}{\text{Norm}} \tag{B.46}$$

Next, the gravitational projections of the acceleration vector are mapped to the body reference frame using quaternion vectors computed in the prediction section. The linear acceleration parameters are defined as the function $h(x_k)$, which is equivalent to the transition matrix to the body reference frame:

$$h(x_k) = \begin{bmatrix} 2q_2 & -2q_3 & 2q_0 & -2q_1 \\ -2q_1 & -2q_0 & -2q_3 & -2q_2 \\ -2q_0 & 2q_1 & 2q_2 & -2q_3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{B.47}$$

Here, *ax*, *ay*, and *az* represent the projection of the gravity vector of the Earth in the body reference frame according to the estimated angles from this stage. The difference between these values and the parameters measured with the accelerometer is the measurement error of this step or the residual of Kalman, which is calculated in each iteration. In the formula z is the accelerometer vector, and y is the measurement error or residual of this step. If the accelerometer sensor output is such that, for example, when the IMU is in the "Level" state, the parameter is A_z-positive (like the BNO055 sensor which is the internal sensor of my inertial measurement unit (IMU)), this means that the accelerometer sensors are defined in the opposite direction. In this case, the matrix z is defined as:

$$z = \begin{bmatrix} -A_x \\ -A_y \\ -A_z \end{bmatrix} \tag{B.48}$$

Thus, the residual of this step is:

$$y = \begin{bmatrix} -A_x - h_0 \\ -A_y - h_1 \\ -A_z - h_2 \end{bmatrix} \tag{B.49}$$

The Jacobian vector $h(x_k)$ is the matrix H:

$$H(x_k) = \begin{bmatrix} 2q_2 & -2q_3 & 2q_0 & -2q_1 & 0 & 0 & 0 \\ -2q_1 & -2q_0 & -2q_3 & -2q_2 & 0 & 0 & 0 \\ -2q_0 & 2q_1 & 2q_2 & -2q_3 & 0 & 0 & 0 \end{bmatrix} \tag{B.50}$$

After calculating the vector $h(x_k)$ and the matrix $H(x_k)$, the matrix K is obtained using the following calculations. First, the matrix S is calculated as follows: The first step of the Kalman filter algorithm involves calculating the matrix S through the equation:

$$S_{k(3\times3)} = H_{k(3\times7)} \cdot P_{k|k-1(7\times7)} \cdot H'_{k(7\times3)} + R_{(3\times3)}$$

where R is a diagonal matrix with dimensions of 3x3. The resulting matrix S will also be a 3x3 matrix. After calculating S, the inverse of S denoted as S^{-1} is calculated, which is used to calculate the Kalman gain matrix K using the following equation:

$$K_{k(7\times3)} = P_{k|k-1(7\times7)} \cdot H'_{k(7\times3)} \cdot S^{-1}_{k(3\times3)}$$

Once K is calculated, the vector x is updated using the following equation:

$$x_{k(7\times1)} = \hat{x}_{k(7\times1)} + K_{k(7\times3)} \cdot y_{k(3\times1)}$$

Finally, the matrix P is updated using the following equation:

$$P_{K|K(7\times7)} = (I_{(7\times7)} - K_{k(7\times3)} \cdot H_{k(3\times7)})_{(7\times7)} \cdot P_{K|K-1(7\times7)}$$

B.6.1 Magnetometer mapping and Heading angle estimation

For measuring the Yaw angle or Heading, magnetometer sensors are commonly used in an Attitude and Heading Reference System (AHRS). In particular, in new solid-state MEMS AHRS, it can also be used as a compass. This type of sensor works based on Hall effects and can be utilized as a residual in low frequency or steady-state attitude estimation for inaccurate fusion algorithms of strap-down systems, similar to accelerometers. The flux of the magnetic field of the Earth is mapped to the body frame using a quaternion representation. The Earth has a consistent Magnetic Reference Vector represented by the **b** vector, which is acquired by using the first set of data from the magnetometers in the three axes (m_x, m_y, m_z) to determine the $\mathbf{m}'$ matrix.

$$\begin{bmatrix} m'_x \\ m'_y \\ m'_z \end{bmatrix} = R^*(\vec{q}) \cdot \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} m_x\left(q_0^2+q_1^2-q_2^2-q_3^2\right)+2m_y\left(q_1q_2+q_0q_3\right)+2m_z\left(q_1q_3-q_0q_2\right) \\ 2m_x\left(q_1q_2-q_0q_3\right)+m_y\left(q_0^2-q_1^2+q_2^2-q_3^2\right)-2m_z\left(q_2q_3+q_0q_1\right) \\ 2m_x\left(q_1q_3+q_0q_2\right)+2m_y\left(q_2q_3-q_0q_1\right)+m_z\left({q_0}^2-{q_1}^2-q_2^2+q_3^2\right) \end{bmatrix} \tag{B.51}$$

From the $\mathbf{m}'$ matrix, the $\mathbf{b}$ matrix is obtained:

$$b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} \sqrt{m'_x{}^2 + m'_y{}^2} \\ 0 \\ m'_z \end{bmatrix} \tag{B.52}$$

The $\mathbf{b}$ matrix can be updated at each iteration using the formula:

$$\mathbf{b}_k = \mathbf{b}_{k-1} + \alpha(\mathbf{b}_{\text{new}} - \mathbf{b}_{k-1}) \tag{B.53}$$

However, we have not updated this parameter in our project. After updating the Roll and Pitch angle in the quaternion presentation, a new $\mathbf{h}$ matrix is formed. In fact, in this stage of the algorithm, the $\mathbf{b}$ parameters are mapped from inertial coordination to body coordination with the last quaternions updated. It is expected that if there is no ferromagnetic or external disturbance in this attitude angle, these amounts should be revealed by the magnetometer sensor outputs. As regards $\mathbf{b} = 0$, the new $\mathbf{h}$ matrix can be calculated as:

$$\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = h(x_k) = R(\vec{q}) \cdot \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} b_x\left(q_0{}^2 + q_1^2 - q_2{}^2 - q_3^2\right) + 2b_z(q_1q_3 - q_0q_2) \\ 2b_x(q_1q_2 - q_0q_3) - 2b_z(q_2q_3 + q_0q_1) \\ 2b_x(q_1q_3 + q_0q_2) + b_z\left(q_0{}^2 - q_1^2 - q_2^2 + q_3^2\right) \end{bmatrix} \tag{B.54}$$

The error or Kalman filter second residual can be determined by y:

$$z = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} \tag{B.55}$$

$$y = z - h(x_k) \tag{B.56}$$

$$y = \begin{pmatrix} m_x - h_0 \\ m_y - h_1 \\ m_z - h_2 \end{pmatrix} \tag{B.57}$$

The new Jacobian $\mathbf{H}$ matrix is also calculated in this stage, as calculated for accelerometers residual for Roll and Pitch angle (Fan et al., 2017), (Sabatini, 2006):

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} 2(q_0b_x - q_2b_z) & 2(q_1b_x + q_3b_z) & 2(-q_2b_x - q_0b_z) & 2(-q_3b_x + q_1b_z) & 0 & 0 & 0 \\ 2(-q_3b_x + q_1b_z) & 2(q_2b_x + q_0b_z) & 2(q_1b_x + q_3b_z) & 2(-q_0b_x + q_2b_z) & 0 & 0 & 0 \\ 2(q_2b_x + q_0b_z) & 2(q_3b_x - q_1b_z) & 2(q_0b_x - q_2b_z) & 2(q_1b_x + q_3b_z) & 0 & 0 & 0 \end{bmatrix} \tag{B.58}$$

Afterwards, in the similar way of the accelerometer residual

During the initialization and alignment procedure of the algorithm, the initial values of the Kalman filter's internal vectors and matrices should be adjusted accordingly.

B.6.2 Initialization and tuning consideration

This section deals with the initial adjustment of the internal parameters of the Kalman filter. The matrices P, R, Q, and F are initialized in a diagonal form, with initial values as given in (Looney, 2012).

The algorithm needs to be run for a certain amount of time to calculate these initial values. Checking this parameter again is unnecessary, and incorrect selection of the initial value will be corrected automatically after a few seconds. Presetting this parameter will reduce the time required for preparation and alignment of the filter.

Matrices R and Q are known as the sensor covariance matrices and the model uncertainty matrices, respectively. The proportion of the values of these matrices determines the behavior of the Kalman algorithm in pursuing the measurement error in the measurement stage, which in turn affects the validity of the accelerometers and magnetometers in measuring the angles of the situation (Feng et al., 2017).

The proportion of R-related parameters of the accelerometers to R-parameters associated with the magnetometers is significant. If the weight of the magnetometers is high, errors measured in the magnetometers' section will have a significant effect on the roll and pitch angles in the Kalman algorithm. If the accelerometer effect increases, the Kalman algorithm will not be able to correct the measurement error during the correction of the magnetometers, and the drift error of the gyroscope will affect the output of the Kalman Yaw angle.

The Q matrix is a 7×7 matrix where the parameters on its original diameter determine the uncertainty of each vector parameter x. The smaller the diameter of this matrix, the less effect the amount of uncertainty of the model will have on the prediction of these parameters in the prediction stage. Thus, the integration from the gyroscopes will have a greater effect on the attitude angle extraction procedure (Feng et al., 2017).

The first four parameters of the Q matrix determine the ratio of the gyroscope's impact on the Kalman model. The next three parameters are related to the bias rate of the gyroscope, and the smaller the bias parameter, the higher the correction rate. Hence, the over-dimensionality of this parameter indirectly increases the influence of the Kalman model on acceleration and magnitude parameters. The bias of gyroscopes over time changes mostly due to the internal temperature variation of the system, so the temperature of the system and the impact of the gyroscope (after calibration) will be effective in determining this parameter. With mentioned considerations define

them:

$$P = \begin{bmatrix} 0.02 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{B.59}$$

$$\mathrm{Q} = \begin{bmatrix} 0.0007 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0007 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0007 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0007 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0008 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0008 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0008 \end{bmatrix}$$

$$R = \begin{pmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{pmatrix} \tag{B.60}$$

Matrix F is also a 7×7 diameter matrix with an initial value defined by a 7×7 identity matrix.

B.7 Results

In order to evaluate the performance of the quaternion-based EKF algorithm for orientation estimation, we carried out experiments utilizing an IMU comprising an accelerometer, gyroscope, and magnetometer. The IMU was mounted on a moving platform, and the proposed EKF algorithm was employed to estimate the Roll, Pitch, and Yaw angles. The obtained results are illustrated in Figure B.3, where the Roll angle estimated by the proposed EKF algorithm is represented by the blue line, while the Roll angle estimated by a BNO055 IMU sensor is represented by the red line. Additionally, the figure shows the comparison between the two approaches, along with the y-acceleration and corresponding angular rate gyroscope output. During the experiment, linear acceleration was applied to the system, with a maximum value of 1.8g.

It was observed that the Roll angle successfully resisted this external disturbance, and the system was rotated by almost 200 deg/s, with the Roll angle being able to withstand it. On the other hand, the stimulation experienced by the Pitch axis was more severe, as the linear acceleration in the x-axis had reached nearly 4g, and the angular velocity output from the gyroscope had gone up to 500 deg/s. Nonetheless, the system was observed to be robust as shown in Figure B.5.

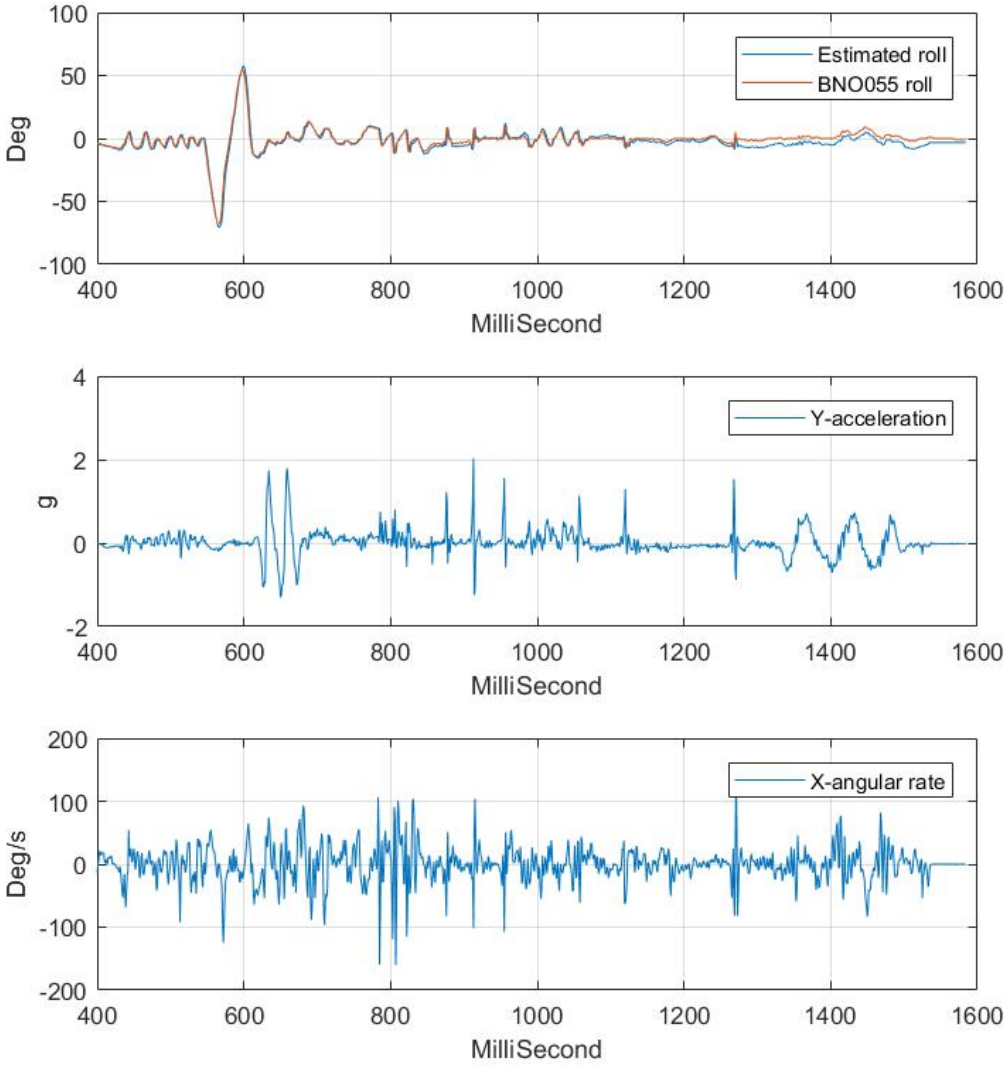


Figure B.3: Roll angle estimates using the proposed EKF algorithm and BNO055 Roll and corresponding acceleration and angular rate.

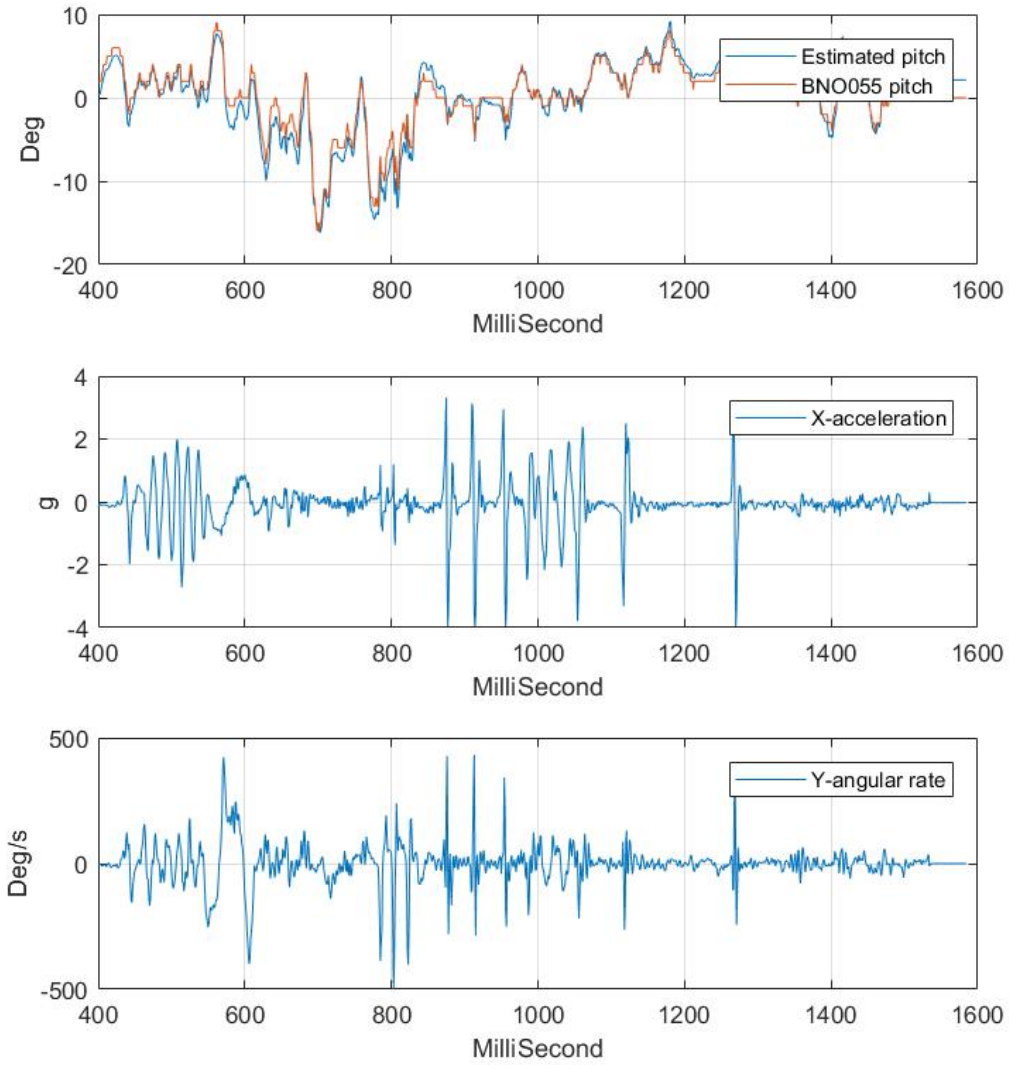


Figure B.4: Pitch angle estimates using the proposed EKF algorithm and BNO055 Pitch and corresponding acceleration and angular rate.

We further assessed the performance of the proposed EKF algorithm for the Heading angle under varying levels of noise in the sensor measurements. The outcomes are presented in Figure B.5, where it can be observed that the proposed EKF algorithm provides less precise estimates of the yaw angle from the sensor measurements. Moreover, the error in roll and pitch can increase up to two times the errors of the Heading angle. Thus, further development is required to improve the accuracy of the algorithm.

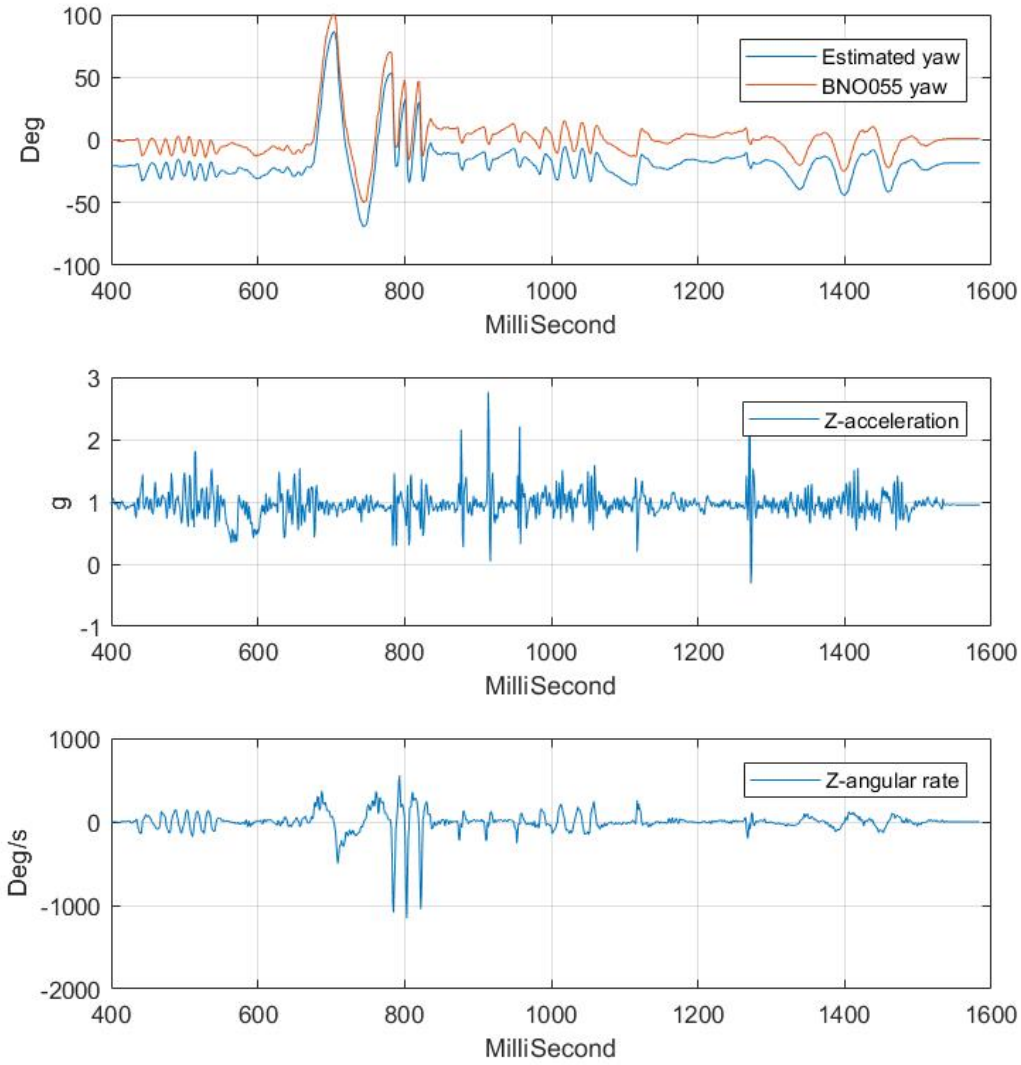


Figure B.5: yaw angle estimates using the proposed EKF algorithm and BNO055 yaw and corresponding acceleration and angular rate.

B.8 Concluding remarks

In this study, we have introduced a new EKF algorithm that uses accelerometer, gyroscope, and magnetometer data to estimate the orientation based on quaternions. Our proposed method can provide more accurate estimations of Roll and Pitch angles compared to the BNO055 sensor. However, the algorithm has a limitation in that it cannot handle long-term or low-frequency linear acceleration, such as in an airplane, since the accelerometer measurements are affected by the tilt and external acceleration, which

the algorithm cannot distinguish. To address this issue, we suggest using GPS to detect and correct external acceleration by incorporating the linear velocity in the inertial frame of reference into the Kalman filter measurement of acceleration (Wen et al., 2021). This approach, known as the GPS-mode, is commonly used in drone applications and requires an expensive GPS-Ided AHRS package that costs more than 2000 euros. Another alternative is to use air data computers or pitot tubes to measure airspeed, which can also help cancel external acceleration, but this method is more complicated. An optical flow sensor is another common sensor used in VTOL systems or multicopters, which can provide visual odometry and measure linear velocity using a camera-based system. By combining different sensors, such as Openmv and BNO055 IMU, using a Kalman filter, we can develop a low-cost IMU system in the lab that can be used for orientation estimation (Liang et al., 2019). With our own developed Kalman filter, we can also integrate other sensors, which can improve the performance of the system and make it comparable to the expensive GPS-Ided AHRS package.

References

David J Acheson. *Elementary fluid dynamics*. Oxford University Press, 1990.

Adafruit Industries. *PCA9685 16-Channel 12-Bit PWM Servo Driver - Technical Documentation*. Adafruit Industries, 2023. URL https://www.adafruit.com/product/815.

Karl J. Åström and Tore Hägglund. *PID Controllers: Theory, Design, and Tuning*. ISA: The Instrumentation, Systems, and Automation Society, 2 edition, 2006.

Douglass Auld and Srinivas. *3-D Lifting Line Theory*. University of Sydney, 1995.

Roman Bapst, Robin Ritz, Lorenz Meier, and Marc Pollefeys. Design and implementation of an unmanned tail-sitter. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1885–1890, 2015. doi: 10.1109/IROS.2015.7353624.

John Bass, Francis Marois, Arthur Murphy, Michel Lacerda, Daniel Schrage, Nicolas Courtois, Dino Mehanovic, and David Rancourt. Preliminary design of a highly efficient vtol system based on tethered fixed-wing aircraft. In *Vertical Flight Society 75th Annual Forum Technology Display*, 2019.

Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

Benewake (Beijing) Co., Ltd. *TFmini LiDAR Range Finder Sensor User Manual*. Benewake, 2023. URL https://www.benewake.com/en/tfmini.html.

John H. Blakelock. *Automatic Control of Aircraft and Missiles*. John Wiley & Sons, New York, 2 edition, 1991. ISBN 0-471-50651-6.

Andrea Borgese, Dario C Guastella, Giuseppe Sutera, and Giovanni Muscato. Tether-based localization for cooperative ground and aerial vehicles. *IEEE Robotics and Automation Letters*, 7(3):8162–8169, 2022.

Giacomo Calciolari. Modelling and control tuning of an airborne wind energy system with on-board generation. 2020.

Wenbin Cao and Liang Zhang. Complexity and design challenges in developing custom autopilot systems for unmanned aerial vehicles. *IEEE Transactions on Aerospace and Electronic Systems*, 55(4):1876–1892, 2019. doi: 10.1109/TAES.2019.2912345.

L. Chen, W. Zhang, and Y. Li. Comprehensive dynamic modeling of vtol tailsitter aircraft with experimental validation. *Aerospace Science and Technology*, 121:107304, 2022.

S. Chen, B. Anderson, and R. Lee. Robust mixed-sensitivity control design for tethered VTOL systems. *International Journal of Robust and Nonlinear Control*, 34(1):89–106, 2024.

X. Chen, Y. Li, and B.D.O. Anderson. Decentralized adaptive control of mimo systems with applications to renewable energy integration. *Automatica*, 124:109322, 2021.

John Doyle, Keith Glover, Pramod Khargonekar, and Bruce Francis. State-space solutions to standard $h2$ and $h\infty$ control problems. In *1988 American Control Conference*, pages 1691–1696. IEEE, 1988.

Geir E. Dullerud and Fernando Paganini. *A Course in Robust Control Theory: A Convex Approach*, volume 36 of *Texts in Applied Mathematics*. Springer, 2000. Chapter 8: Linear Fractional Transformations and μ.

Christopher Edwards and Sarah K. Spurgeon. Sliding mode control: Theory and applications. *IEEE Transactions on Aerospace and Electronic Systems*, 37(5):1174–1185, 2021. doi: 10.1109/TAES.2021.963521.

Michael Erhard and Hans Strauch. Flight control of tethered kites in autonomous pumping cycles for airborne wind energy. *Control Engineering Practice*, 40:13–26, 2015.

Bingfei Fan, Qingguo Li, Chao Wang, and Tao Liu. An adaptive orientation estimation method for magnetic and inertial sensors in the presence of magnetic disturbances. May 2017.

S. Boorghan Farahan, J. J. M. Machado, F. G. de Almeida, and J. M. R. S. Tavares. 9-DOF IMU-based attitude and heading estimation using an extended Kalman filter with bias consideration. *Sensors*, 22(9):1–16, 2022.

Kaiqiang Feng, Jie Li, Xiaoming Zhang, Chong Shen, Yu Bi, Tao Zheng, and Jun Liu. A new quaternion-based kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm. September 2017.

Silvan Fuhrer, Sebastian Verling, Thomas Stastny, and Roland Siegwart. Fault-tolerant flight control of a vtol tailsitter uav. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4134–4140, 2019. doi: 10.1109/ICRA.2019.8793467.

P. Garg. Characterisation of fixed-wing versus multirotors uavs/drones. *Journal of Geomatics*, 16(2):152–159, 2022. doi: 10.58825/jog.2022.16.2.44.

Alberto Isidori. *Nonlinear Control Systems*. Springer, 3rd edition, 1995. Chapter 2: Covers linearization methods. Section 2.3: Addresses singular points and limit-based analysis.

Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 3rd edition, 2002. Section 4.3 discusses linearization around equilibrium points. Pages 165-167 specifically address using limits for discontinuous functions.

S. Kim, H. Lee, and J. Park. Robust control of tailsitter vtol uavs under uncertain aerodynamic conditions. *Control Engineering Practice*, 119:104968, 2022.

Benyamin Kosarnovsky and S Arogeti. A string of tethered drones-system dynamics and control. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019.

S. Kumar and R. Patel. Robust decentralized control for large-scale industrial systems. *Journal of Process Control*, 110:97–112, 2022.

Mattew Leccadito. A Kalman filter based attitude heading reference system using a low-cost inertial measurement unit. Master's thesis, Virginia Commonwealth University, 2013.

T. Lee, D. Seo, and H. Kim. Geometric control and stability analysis of vtol uavs. *IEEE Robotics and Automation Letters*, 5(2):2357–2364, 2020.

Shihua Li, Mingming Wang, Yi Liu, and Yan Zhang. A comprehensive review of sliding mode control for robotic systems: Theory, tuning, and applications. *Annual Reviews in Control*, 55:1–23, 2023a.

X. Li, Y. Wang, and H. Zhang. Nonlinear modeling and control of vtol tailsitter aircraft during transition maneuvers. *Journal of Guidance, Control, and Dynamics*, 46(3): 489–502, 2023b.

X. Liang, G. Chen, S. Zhao, G. Tong, L. Jiang, and W. Zhang. Remote guidance method of unmanned aerial vehicle based on multi-sensors. In *2019 IEEE International Conference on Unmanned Systems (ICUS)*, pages 297–302. IEEE, 2019.

M. Liu, X. Chen, and H. Wang. Gyroscopic effects in tailsitter vtol uavs: Analysis and compensation. *Journal of Guidance, Control, and Dynamics*, 46(1):156–169, 2023.

Mark Looney. Analyzing frequency response of inertial mems in stabilization systems. *Analog Dialogue*, 46(7), July 2012.

Sebastian O.H. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor. April 2010.

Miguel Nakajima Marques, Sandro Augusto Magalhães, Filipe Neves Dos Santos, and Hélio Sousa Mendonça. Tethered unmanned aerial vehicles—a systematic review. *Robotics*, 12(4):117, 2023.

Radek Matušů, Bilal Şenol, and Celaleddin Yeroglu. Linear systems with unstructured multiplicative uncertainty: Modeling and robust stability analysis. *PLoS ONE*, 12, 07 2017. doi: 10.1371/journal.pone.0181078.

A. Packard and J.C. Doyle. *The Complex Structured Singular Value*, volume 29. 1993.

S. Park, J. Kim, and S. Lee. Dynamic modeling and control of tailsitter uavs in transition flight. *Control Engineering Practice*, 109:104751, 2021.

A. F. C. Pereira and J. M. M. Sousa. Assisted site-dependent selection of the most suitable airborne wind energy system via fuzzy analytic network process. *Energy Reports*, 9:2881–2899, 2023. ISSN 2352-4847. doi: 10.1016/j.egyr.2023.01.

045. URL https://www.sciencedirect.com/science/article/pii/S2352484723000537.

Samuel Quenneville, Francis Thérien, Jessy Verrette, David Rancourt, Alex Walsh, Jean-Philippe Lucking Bigué, Robotics Lead Engineer, and Philippe Feyel. Experimental demonstration of the lifting capability of a towed payload using multiple fixed-wing uavs. 2023.

A.M. Sabatini. Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing. *IEEE Transactions on Biomedical Engineering*, 2006.

Bosch Sensortec. *BNO055 Intelligent 9-axis absolute orientation sensor*. Bosch Sensortec GmbH, 2014. URL https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf. Document revision 1.4, Document number: BST-BNO055-DS000-12.

Yury Shtessel and Christopher Edwards. Evolution of sliding mode control theory: Past, present, and future. *Annual Reviews in Control*, 56:213–229, 2023.

Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, 1991. Chapter 4: Explains why limits are necessary for discontinuous nonlinear systems. Pages 199-202: Discusses linearization techniques.

J. Sun, H. Wang, and L. Chen. Optimal trajectory planning and control for tailsitter vtol uavs. *Aerospace Science and Technology*, 133:107374, 2023.

B. Suwandi, T. Kitasuka, and M. Aritsugi. Low-cost IMU and GPS fusion strategy for apron vehicle positioning. In *TENCON 2017-2017 IEEE Region 10 Conference*, pages 449–454. IEEE, 2017.

Hamid D. Taghirad, Mohammadreza Safaei, and Mohammadreza Khorrami. H-based control of aras-diamond: A vitrectomy eye surgery robot. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 235(1):1–14, 2021. doi: 10.1177/0954406220979334. URL https://journals.sagepub.com/doi/10.1177/0954406220979334.

Sebastian Verling, Basil Weibel, Maximilian Boosfeld, Kostas Alexis, Michael Burri, and Roland Siegwart. Full attitude control of a vtol tailsitter uav. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3006–3012. IEEE, 2016.

Carlos Viegas, Babak Chehreh, José Andrade, and João Lourenço. Tethered uav with combined multi-rotor and water jet propulsion for forest fire fighting. *Journal of Intelligent & Robotic Systems*, 104(2):21, 2022.

Ko-Yuan Wang, Pei-Hsuan Lee, and Shao-Kang Hung. Optimum electric cable selection for kite-like unmanned aerial vehicle. In *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1537–1540. IEEE, 2015.

L. Wang, X. Liu, and R. Zhang. Autonomous control of tailsitter vtol uavs: A deep reinforcement learning approach. *Aerospace Science and Technology*, 132:107352, 2023a.

L. Wang, H. Zhang, and Z. Liu. Advances in decentralized mimo control: A deep learning approach. *IEEE Transactions on Control Systems Technology*, 31(4):1542–1557, 2023b.

R. Wang, J. Liu, and T. Zhang. Wind-resistant control of tailsitter vtol uavs based on adaptive aerodynamic modeling. *IEEE Transactions on Control Systems Technology*, 30(6):2398–2411, 2022.

Matthew Watson. The design and implementation of a robust AHRS for implementation on a quadrotor. Meng thesis, Department of Electronic & Electrical Engineering, May 2013.

W. Wen, T. Pfeifer, X. Bai, and L.-T. Hsu. Factor graph optimization for GNSS/INS integration: A comparison with the extended Kalman filter. *NAVIGATION: Journal of the Institute of Navigation*, 68(2):315–331, 2021.

Xsens Technologies B.V. Mti-g: Gps aided inertial measurement unit - technical documentation. Technical Report XS-MTI-G-2010, Xsens Technologies, 2010. URL https://ti.arc.nasa.gov/dataset/KrexDataBasaltHills2012/refdoc/mti-g_leaflet.pdf.

W. Xu, Y. Zhang, and J. Liu. Learning-based approach for dynamic modeling of vtol tailsitter uavs. *IEEE Robotics and Automation Letters*, 8(2):1089–1096, 2023.

H. Yang, Z. Liu, and K. Wang. Hybrid modeling approach for vtol tailsitter uavs combining analytical and data-driven methods. *IEEE Transactions on Aerospace and Electronic Systems*, 59(1):591–604, 2023.

Jun Yang, Shihua Li, and Xinghuo Yu. Hybrid robust control strategies for uncertain nonlinear systems: A survey. *IEEE Transactions on Industrial Electronics*, 69(1):1–16, 2022.

H. Zhang, Y. Wang, and Z. Chen. Aerodynamic modeling and analysis of vtol tailsitter aircraft. *Aerospace Systems*, 5(2):89–103, 2022a.

Huaguang Zhang, Dan Wang, Hongyi Li, Xiaodong Zhou, and Guo-Ying Xiao. Recent advances in robust control: Theory, method and applications. *Science China Information Sciences*, 65(4):1–3, 2022b.

K. Zhang, Y. Liu, and J. Wang. Adaptive MIMO control for transition flight in fixed-wing VTOL aircraft. *IEEE Transactions on Aerospace and Electronic Systems*, 59(2):731–745, 2023.

Y. Zhang, H. Chen, and J. Wang. A comprehensive review of vtol uav dynamics modeling and control: Progress and challenges. *IEEE Transactions on Aerospace and Electronic Systems*, 57(4):2541–2566, 2021.

L. Zhao, M. Chen, and R. Wang. Adaptive control strategy for vtol tailsitter uavs with uncertain aerodynamics. *IEEE Transactions on Control Systems Technology*, 30(4):1567–1580, 2022.

H. Zheng, X. Liu, and P. Wang. Survey of open-source flight control platforms for unmanned aerial vehicles. *International Journal of Advanced Robotic Systems*, 18(3):1–15, 2021. doi: 10.1177/17298814211026543.

Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, NJ, 1996. ISBN 0134565673.

Kemin Zhou, John C Doyle, and Keith Glover. Robust control: Past, present, and future. *Annual Review of Control, Robotics, and Autonomous Systems*, 6:1–28, 2023.

W. Zhou, C. Yang, and X. Li. Modeling and validation of aerodynamic characteristics for tailsitter vtol uavs. *Journal of Aerospace Engineering*, 35(3):04022012, 2022.

Lida Zikou, Christos Papachristos, and Anthony Tzes. The power-over-tether system for powering small uavs: Tethering-line tension control synthesis. In *2015 23rd Mediterranean Conference on Control and Automation (MED)*, pages 681–687, 2015. doi: 10.1109/MED.2015.7158825.