Hybrid lift UAV design and control for precision landing on a moving vessel in high sea state

Mancinelli, A.

DOI
10.4233/uuid:e2003e6a-6410-45bb-bf82-e36ef727d25c

Publication date
2025

Document Version
Final published version

Citation (APA)
Mancinelli, A. (2025). *Hybrid lift UAV design and control for precision landing on a moving vessel in high sea state*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:e2003e6a-6410-45bb-bf82-e36ef727d25c

Hybrid lift UAV design and control for precision landing on a moving vessel in high sea state

Alessandro Mancinelli

Hybrid lift UAV design and control for precision landing on a moving vessel in high sea state

Hybrid lift UAV design and control for precision landing on a moving vessel in high sea state

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Thursday 11 September 2025 at 17:30 o'clock

by

Alessandro MANCINELLI

Master of Science in Aeronautical Engineering, Sapienza University, Rome
born in Rome, Italy

This dissertation has been approved by the promotors.

Printed by: jouwboek.nl

Cover by: Hashim Karim

ISBN 978-94-6518-112-7

An electronic copy of this dissertation is available at https://repository.tudelft.nl/.

Anything one man can imagine,
other men can make real

- Jules Verne

CONTENTS

SUMMARY

Achieving precise and reliable autonomous landings on moving ship decks is a critical challenge for hybrid Unmanned Aerial Vehicles (UAVs), particularly under harsh maritime conditions. This dissertation addresses this challenge by developing a nonlinear control framework for a novel dual-axis tilting rotor quad-plane, designed specifically to enhance stability, maneuverability, and precision during landing operations on moving platforms. Operating in such harsh environments requires effective handling of varying wind conditions, ship motion, and rapid position changes.

The dual-axis tilting rotor concept offers unique capabilities that make it particularly suited for precision landing tasks on moving ship decks. Unlike conventional hybrid under-actuated UAV designs, the overactuated nature of the dual-axis tilting rotor quad-plane allows it to maintains full 6 Degrees Of Freedom (DOF) control authority during low airspeed flight. Moreover, its rapid thrust vectoring capability provides exceptional wind rejection performance, enabling precise control even under turbulent maritime conditions. This versatility is essential for maintaining stability and accuracy during complex landing maneuvers where environmental disturbances and ship motion are constantly changing. However, the novel propulsion system also presents significant control challenges due to its non-affine in the input dynamics. Developing advanced control strategies capable of handling these complexities is critical for achieving reliable and precise autonomous landings on moving platforms.

First, a nonlinear programming-based control allocation algorithm is developed to overcome the limitations of state-of-the-art methods that rely on linearized control effectiveness. This assumption may be too restrictive for vehicles with highly nonlinear effector dynamics, such as the dual-axis tilting rotor quad-plane. The proposed control allocation algorithm effectively manages the complex dynamic interactions inherent to the novel propulsion system, enabling smooth and precise control within the low-airspeed flight regime.

Second, the controller is extended to operate across the entire flight envelope, from hovering to forward flight. Unlike traditional hybrid UAV controllers that rely on distinct control strategies for each flight mode, the proposed framework enables seamless transitions by dynamically reallocating control objectives across the available actuators, including adjustments to vehicle attitude as airspeed varies.

Third, this dissertation introduces a real-time actuator state feedback system to enhance the controller’s robustness and adaptability in challenging maritime environments. By eliminating reliance on predefined actuator models, this improvement provides continuous situational awareness of the propulsion system's health, allowing the controller to dynamically adjust to varying operational conditions. The enhanced hardware architecture enables direct control of motor RPM and angular rotor tilt, significantly improving the system's resilience against environmental disturbances, actuator degradation, and battery voltage fluctuation. This capability forms the foundation for developing a robust fault-tolerant control framework essential for reliable autonomous landings on moving platforms.

Fourth, leveraging the improved hardware architecture, a fault-tolerant control framework is developed to ensure reliable operation under various actuator failure conditions. The dual-axis tilting rotor quad-plane's over-actuated design allows the controller to reallocate control efforts dynamically among functional actuators, maintaining stability and control authority even under severe actuator failures. Extensive flight tests validate the fault-tolerant framework's effectiveness, demonstrating resilience in challenging scenarios.

Finally, a real-time trajectory planning algorithm is developed to enable autonomous landing on moving ship decks. Using a Long Short-Term Memory (LSTM) neural network model, a prediction framework is created to estimate ship motion over a 7-second horizon. This prediction enables the generation of feasible landing trajectories that are continuously reassessed to account for prediction uncertainties and tracking errors. While simulation results demonstrated the approach's potential, future work will involve testing the algorithm in real-world conditions.

The proposed control framework, combined with the novel dual-axis tilting rotor quad-plane design, offers a robust and adaptable solution for achieving reliable autonomous landings on moving platforms. The methodologies developed in this thesis can be further extended to various hybrid UAV configurations, providing valuable insights for broader applications in complex operational environments.

1

INTRODUCTION

1.1. INTRODUCTION TO HYBRID LIFT UAVS

Unmanned Aerial Vehicles (UAVs) have undergone a rapid evolution due to advancements in Micro Electro-Mechanical Systems (MEMS), battery technology, high-performance microprocessors, and improvements in motor and motor controller systems. These innovations have enabled UAVs to be widely adopted in applications such as aerial delivery, environmental monitoring, search and rescue, and defense operations. A particularly promising category is hybrid lift UAVs, which combine the advantages of fixed-wing and rotary-wing aircraft. These vehicles offer Vertical Takeoff and Landing (VTOL) capabilities similar to helicopters while benefiting from the efficiency and endurance of traditional fixed-wing airplanes.

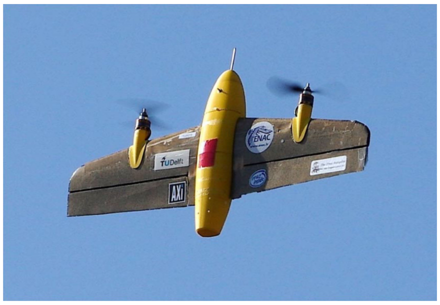
Hybrid lift UAV architectures vary significantly, each presenting distinct advantages and limitations. Tail-sitters, shown in Figure 1.1a, transition between vertical and horizontal flight by rotating their entire body. Tilt-rotors, illustrated in Figure 1.1b, use rotating propulsion units that allow transitions between hover and cruise modes. Tilt-wing UAVs, depicted in Figure 1.1c, employ rotating wings to enable both hover and forward flight. Meanwhile, quad-planes, shown in Figure 1.1d, feature dedicated vertical lift motors that enable VTOL capability. A comprehensive overview of manned and unmanned VTOL vehicles is available in the Miro board created for this research[1], offering insights into the trade-offs associated with each configuration and supporting the selection of optimal designs based on specific mission requirements.

While hybrid UAVs offer notable versatility, the inclusion of wings introduces significant control challenges—especially during landings in gusty environments. Unlike multi-rotor UAVs, hybrid designs must account for aerodynamic effects such as lift and drag forces acting on the wing. These forces can produce unwanted moments and disturbances, making precise position control at low airspeed particularly difficult.

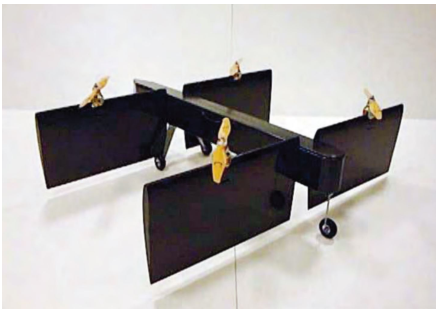
1.2. CHALLENGES OF HYBRID LIFT UAVS: PRECISION LANDING ON A MOVING TARGET IN HARSH CONDITIONS

One of the most critical challenges for hybrid UAVs is performing precision landings, particularly in environments where external disturbances are significant. This challenge is especially pronounced in shipboard operations, where the UAV must land on a moving platform subject to complex dynamic motion. Such operations represent one of the most demanding use cases for autonomous landing systems and form the primary application focus of this thesis.

[1] Link to miro board for VTOL vehicle categorization: https://shorturl.at/Qlw9S


(a) The Cyclone tail sitter UAV [1]

(b) The Dual-axis tilting rotor quad-plane, developed by Alessandro Mancinelli et. al. [2]


(c) Tilt-wing UAV developed by E. Cetinsoy et. al. [3]

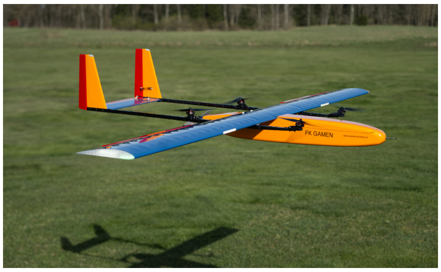

(d) Axel Brinkeby quad-plane. [4]

Figure 1.1: Several different UAV designs.

Shipboard landings involve managing continuously changing 6-Degrees Of Freedom (DOF) ship motion. These nonlinear dynamics create a highly challenging landing environment that demands real-time adaptation from the UAV.

The interaction between wind and the ship's superstructure generates localized gusts and turbulence, posing an additional challenge. These disturbances act on the aerodynamic surfaces of hybrid UAVs, such as wings and control surfaces, and must be compensated for to maintain accurate position control. Compared to pure multi-rotor UAVs, hybrid platforms are more susceptible to these disturbances due to their larger aerodynamic surface area. As a result, maintaining stable position control requires a fast-responding control system that can dynamically coordinate rotor-based thrust with aerodynamic surface deflections, making effective gust compensation significantly more complex.

Beyond these environmental factors, hybrid UAVs face significant control challenges during the transition between hover and forward flight. This phase introduces rapid changes in the vehicle's dynamics, as lift and control authority gradually shift from rotor-generated thrust to aerodynamic surfaces. During this transition, the vehicle operates in low-airspeed regimes where aerodynamic surfaces are near stall and thus less effective, while rotor-based control remains active but must compensate for increasing aerodynamic loads. When this is compounded by external factors such as ship motion and wind gusts, precise trajectory tracking becomes particularly difficult. Managing this phase therefore requires advanced flight control strategies capable of dynamically and effectively reallocating control inputs in real time.

Ensuring a safe and repeatable landing process further necessitates ship motion prediction models and robust fault-tolerant control strategies. The UAV must be capable of anticipating landing opportunities, dynamically adjusting its trajectory, and maintaining stability in the presence of external disturbances or system failures. This research addresses these challenges by developing a novel UAV design and control framework tailored for high-precision landings in extreme maritime conditions.

1.3. RESEARCH QUESTIONS

The primary research question of this work is:

Research Goal

How can a hybrid lift UAV be designed and controlled to ensure stable, precise, and fault-tolerant autonomous landings on moving ship decks in high sea states?

Addressing this question requires investigating key aspects of UAV design, control strategies, predictive landing algorithms, and fault tolerance mechanisms. A fundamental issue is enhancing UAV stability and wind rejection across all flight phases, particularly during landing. A highly agile UAV can adapt to the ship's deck dynamics while maintaining control authority against wind disturbances. Given the large and highly dynamic attitude changes of a vessel in rough sea conditions, achieving full 6 DOF control could enable the UAV to align its attitude with that of the vessel during touchdown. This leads to the first research question:

Research Question 1

How can a hybrid lift UAV be designed to ensure stable and precise autonomous landings on moving ship decks in sea state 5?

This research aims to identify key design trade-offs that optimize both stability and efficiency while ensuring the UAV retains sufficient control authority under extreme conditions.

Developing robust control strategies is critical for shipboard landing scenarios, where UAVs must remain stable despite wind disturbances and vessel motion. These demanding conditions require control allocation techniques capable of handling multiple, heterogeneous actuators with nonlinear and coupled effects. Hybrid UAVs pose unique challenges in distributing control inputs effectively across diverse control effectors while maintaining consistent performance across the full flight envelope. This motivates the second research question:

Research Question 2

How can control strategies and allocation methods be developed for nonlinear hybrid UAVs to ensure robust flight performance across the full flight envelope?

To ensure mission continuity during critical operations, fault tolerance is essential. Autonomous UAV landings must remain reliable even in the presence of actuator failures, sensor faults, or degraded subsystems. Fault-tolerant control strategies allow the system to maintain stability and control authority despite such internal issues. This leads to the third research question:

Research Question 3

What level of fault tolerance can be achieved in the chosen UAV to ensure safe landings?

Beyond precise and fault-tolerant control of the UAV, landing performance can be further improved by anticipating the future motion of the landing platform. In high sea states, the ship's deck undergoes rapid and nonlinear motion, making it difficult for reactive controllers to ensure safe touchdown. Predicting the vessel's motion a few seconds into the future can enable the UAV to plan more accurate and stable landing trajectories. Although some ship motion prediction models exist, they are typically designed for ship-based systems and are not readily suited for real-time integration into onboard UAV guidance algorithms. This leads to the fourth research question:

Research Question 4

How can ship motion be predicted to improve real-time adaptive landing trajectory planning?

By addressing these questions, this research aims to advance autonomous UAV precision landing and control in dynamic maritime environments, expanding UAV operational capabilities in high-risk, high-uncertainty scenarios.

1.4. DISSERTATION OUTLINE

This dissertation is structured as follows:

Chapter 2 introduces the dual-axis tilting rotor quad-plane, the vehicle ultimately chosen for precision landing on a moving target. This chapter presents the vehicle's architecture, explores simulation models, and compares its performance against conventional quad-planes, providing a foundation for understanding the advantages and trade-offs of this novel UAV design.

Chapter 3 delves into the development of a flight control strategy for the overactuated dual-axis tilt-rotor quad-plane in the low-airspeed configuration. It analyzes the limitations of the state-of-the-art Incremental Nonlinear Dynamic Inversion (INDI) control method and derives a more effective control algorithm for the vehicle.

Chapter 4 develops a unified nonlinear control framework tailored for managing the transition phases of hybrid UAVs. Starting from the hovering controller developed in Chapter 3, it introduces controller improvements that enable control across the vehicle's entire flight envelope.

Building upon this, Chapter 5 details hardware modifications and improvements to enable real-time feedback from motors and servos, emphasizing the importance of accurate state estimation and dynamic adaptation in ensuring robust UAV performance.

Chapter 6 shifts focus to fault tolerance, exploring how the chosen UAV maintains flight stability in the presence of actuator failures. By leveraging actuator redundancy and intelligent control allocation, the chapter presents strategies to enhance UAV resilience in mission-critical scenarios.

In Chapter 7, the discussion turns to ship motion prediction and trajectory planning, detailing how Long Short Term Memory (LSTM)-based models and polynomial path planning techniques are used to guide real-time landing maneuvers. By integrating ship motion forecasting into the control architecture, this chapter explores how UAVs can anticipate and adapt to unsteady landing conditions on moving vessels.

Finally, Chapter 8 concludes the dissertation by summarizing the key findings and addressing the research questions posed at the beginning of this work. It also outlines potential future research directions, identifying opportunities for further refinement of hybrid UAV technology and its applications in maritime and other dynamic environments.

REFERENCES

[1] E. Smeur, M. Bronz, and G. D. Croon. “Incremental Control and Guidance of Hybrid Aircraft Applied to a Tailsitter Unmanned Air Vehicle.” In: *Journal of Guidance Control and Dynamics* 42 N.2 (2020). doi: 10.2514/1.G004520.

[2] A. Mancinelli, B. D. W. Remes, G. C. H. E. de Croon, and E. J. J. Smeur. “Unified Incremental Nonlinear Controller for the Transition Control of a Hybrid Dual-Axis Tilting Rotor Quad-Plane”. In: *IEEE Transactions on Robotics* 41 (2025), pp. 306–325. doi: 10.1109/TRO.2024.3498372.

[3] E. Cetinsoy, S. Dikyar, C. Hancer, K. Oner, E. Sirimoglu, M. Unel, and M. Aksit. “Design and construction of a novel quad tilt-wing UAV”. In: *Mechatronics* 22.6 (2012). Special Issue on Intelligent Mechatronics, pp. 723–745. issn: 0957-4158. doi: 10.1016/j.mechatronics.2012.03.003.

[4] *Quad-plane developed by Axel Brinkeby.* Accessed 20-03-2025. url: https://axelsdiy.brinkeby.se/?p=2351.

2 VEHICLE DESIGN, MODELING, AND PRELIMINARY RESULTS

In the Introduction, we presented various hybrid lift UAV designs. This chapter focuses on the design and modeling of a novel dual-axis tilting rotor quad-plane—a configuration that combines vertical and forward flight capabilities through a propulsion system originally investigated to improve quadcopter agility, but not previously applied to quad-plane architectures.

The proposed vehicle serves as the foundational platform for this dissertation, supporting autonomous landing on moving ships. Preliminary wind tunnel tests were conducted to evaluate the system's wind rejection capabilities and compare its performance against a conventional quad-plane design, highlighting its potential for robust operation in challenging environments.

This chapter is based on the following article:
A. Mancinelli, E.J.J. Smeur, B. Remes and G.C.H.E. Croon, "Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design," 2022 International Conference on Unmanned Aircraft Systems (ICUAS), Dubrovnik, Croatia, 2022, pp. 197-206

2.1. INTRODUCTION

In the last twenty years, a boom in the Unmanned Aerial Vehicle industry has been experienced all over the world. In particular, the development has been focusing on VTOL UAVs. The development of brushless motors and MEMS, together with a huge leap in the battery and the microprocessor technology, allowed the development of small and cheap UAVs capable of controlled vertical takeoff and landing without the necessity of a complex swash plate system typical of rotary wing vehicles. Of particular interest are the so called 'hybrid' vehicles, able to combine the cruise efficiency of conventional fixed wing planes with hovering capability. One kind of hybrid VTOL vehicle is the quad-plane: a conventional fixed wing UAV equipped with four fixed rotors in a multi-copter configuration, allowing the UAV to takeoff and land vertically. An example of such a quad-plane can be found in [1]. Once in hovering flight, the quad-plane can transition to forward flight by gradually increasing the thrust of the forward motor and switching off the vertical lifting rotors.

Another design of the quad-plane, presented in 2012 by Gerardo Ramon Flores et al. [2], was able to make use of the same motors for both the hovering and the forward configuration. This was possible thanks to a mechanism able to synchronously tilt all the rotors of 90 degrees. An Cheng et al. [3] presented a quad-plane with only three rotors, two of which are independently tiltable. This feature improves the yaw authority of the vehicle in the hovering phase, relying on differential tilt rather than differential thrust.

However, as almost every VTOL platform with a non negligible wing surface, quad-plane designs are highly influenced by wind disturbances. In previous work, Hang Zhang et al. [4] successfully analyzed the steady wind rejection capabilities of such vehicles in detail. Unfortunately, the analysis is limited to the static rejection capability and does not consider the vehicle dynamics. Due to their limited degrees of freedom, conventional quad-plane platforms tend to reject the wind by changing the attitude. Because the attitude dynamics of such vehicles are typically slow, the disturbance rejection capability is also slow.

Another relevant limitation related to lacking full 6 degree of freedom authority, is the impossibility of independently achieving any desired rotation and translation in the 3D space. This limitation is particularly penalizing when a precision landing is attempted on a moving and tilting platform such as a ship at sea.

To address the maneuvering limitations of quadrotors, Ali Bin Junaid et al. [5] designed a quad-copter with dual-axis tilting rotors and full 6 DOF authority. Through various flight tests in constrained space achieving complex trajectory they proved the effectiveness of a tilting rotor system to enhance the quad-copter agility and mobility.

The contribution of this chapter is the extension of this concept to a quad-plane vehicle and the evaluation of the wind rejection capabilities and overall performance of the resulting vehicle.

We propose a novel quad-plane design consisting of a flying wing structure propelled with only four motors used for both the hovering and the forward flight condition. The special aspect of the rotors is their double axis tilting capability achieved thanks to two servomotors coupled with a specially developed gear mechanism. This gives the vehicle independent control over all 6 degrees of freedom. It is therefore possible to directly and independently achieve any combination of translation and rotation, leading to reactive and effective wind disturbance rejection.

The chapter is organized as follows; In Section 2.2 the novel vehicle design is presented. In Section 2.3, the mathematical model of the UAV is derived. In Section 2.4, the actuators are identified and modeled through several tests. In Section 2.5, the results of the simulations are analyzed. In Section 2.6, the results of a flight test campaign conducted on the UAV in the wind tunnel are commented. Finally, in Section 2.7 some conclusions are drawn.

2.2. VEHICLE DESIGN

The vehicle design consists of a blended wing body structure propelled with four dual-axis independently tiltable rotors acting as vertical and forward propellers. A render of the proposed vehicle prototype in hovering configuration is shown in Fig.2.1 while a summary of the vehicle characteristics is reported in Table 2.1.

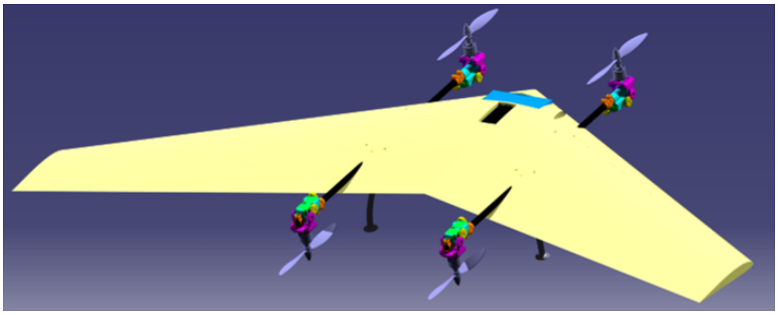


Figure 2.1: Render of the proposed vehicle.

2.2.1. WING DESIGN

The airfoil used for the wing design is a TL54, a reflex airfoil specifically developed for flying wing applications. In particular, since this wing was designed from scratch, a non-viscous analysis using the open-source

CG location	256mm from root L.E.
Takeoff mass	2.3 kg
Vehicle Inertia I_{xx} in Γ_b	0.1 $[kg \cdot m^2]$
Vehicle Inertia I_{yy} in Γ_b	0.15 $[kg \cdot m^2]$
Vehicle Inertia I_{zz} in Γ_b	0.25 $[kg \cdot m^2]$
Rotor Inertia I_{xx} in Γ_p	$1.3 \cdot 10^{-4}$ $[kg \cdot m^2]$
Rotor Inertia I_{yy} in Γ_p	$1.5 \cdot 10^{-4}$ $[kg \cdot m^2]$
Length l_1	185 $[mm]$
Length l_2	185 $[mm]$
Length l_3	376 $[mm]$
Length l_4	290 $[mm]$

Table 2.1: Physical characteristics of the prototype vehicle, see Fig.2.4 and Fig.2.5 for the nomenclature.

software XFLR5[1] was run in order to identify the aerodynamic properties and the aerodynamic center of the wing. The knowledge of the aerodynamic center was also used during the weight distribution of the vehicle, in such a way to achieve a static stability margin of 9.6%. A picture of the analyzed wing is visible in Fig.2.2 while the wing specifications are reported in Table 2.2. After being designed, the wing was cut from a block of foam using a Computer Numerical Control (CNC) machine.

Profile	TL54
Root chord length	45 cm
Tip chord length	21 cm
Mean aerodynamic chord	33.25 cm
Wing surface	0.57 m^2
Total wing-span	1.85 m
Twist angle tip	-5 deg
Dihedral angle	0 deg
Sweep angle	28.7 deg
Wing a.c.	288mm from root L.E.

Table 2.2: Specs of the designed wing

2.2.2. TILTING MECHANISM DESIGN

The tilting mechanism and the motor support were designed to be entirely 3D printed using a commercial Filament Deposition Material machine. As rotation transmission medium, a gearing system was

[1] http://www.xflr5.tech/xflr5.htm

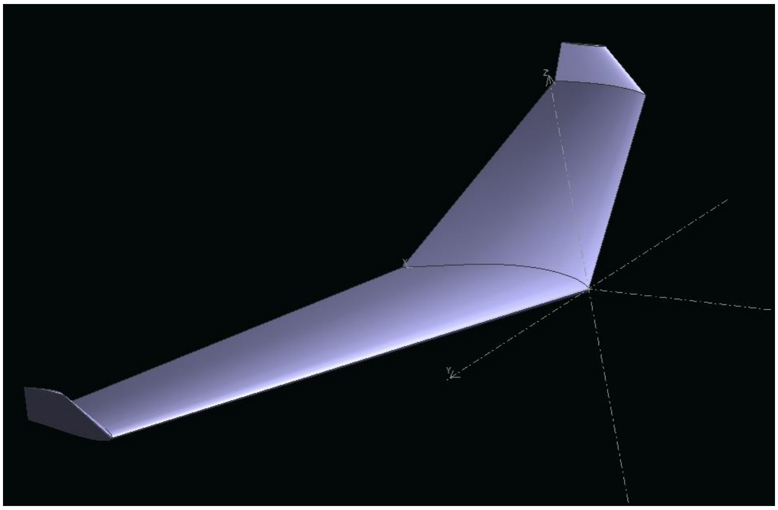


Figure 2.2: Wing shape of the vehicle.

preferred to a pull/push rod system to allow linear torque and rotational speed distribution between servo and rotor. Moreover, by adjusting the number of teeth it is possible to adjust the torque and speed ratio between the servo and the rotor system.
The properties of the gear system on the azimuth and elevation tilting rotation are reported in Table 2.3. As shown in Fig. 2.5, the azimuth rotor rotation is the rotation over the propeller x-axis (g_i angle) while the elevation rotor rotation is the rotation over the propeller y-axis (b_i angle). In Fig. 2.3, a render of an exploded and isometric view of the tilting mechanism is shown.

Azimuth gear module	1.39 mm
N. teeth azimuth servo gear	14
N. teeth azimuth rotor gear	13
Elevation gear modulus	1.39 mm
N. teeth elevation servo gear	14
N. teeth elevation rotor gear	14
Max elevation angle	25 deg
Min elevation angle	-120 deg
Max azimuth angle	45 deg
Min azimuth angle	-45 deg

Table 2.3: Specs of the designed tilting mechanism

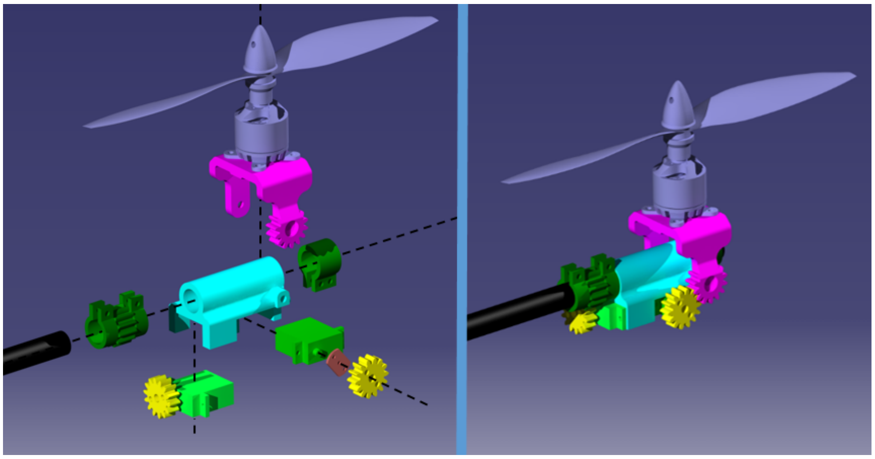


Figure 2.3: Render of the tilting mechanism, exploded view on the left side and isometric view on the right side.

2.3. MATHEMATICAL MODEL

An accurate mathematical model of the vehicle was derived with the aim of simulating the UAV performance and to predict the servomotor load. Notice that under the condition of $b_i = 0$ and $g_i = 0$, the same mathematical model also applies to the conventional quad-plane dynamics in the hovering configuration.

The following assumptions were made during the mathematical model development:

- Aerodynamic interaction between wind and propeller is neglected.
- Thrust is always aligned with the local rotor vertical axis.
- The change of the body inertia due to the rotor tilting is negligible and x_b, y_b and z_b are vehicle principal axes.
- x_p, y_p and z_p are principal axes for the propeller, and the inertia terms I^p_{xx} and I^p_{yy} are negligible.
- The inertia tensor of the tilting mechanism in the propeller reference frame Γ_p is a diagonal matrix.

The rotors identification, rotating direction and disposition with reference to the CG are shown in Fig. 2.4.

2.3.1. REFERENCE FRAMES AND TRANSPOSING MATRICES

The reference frames used are the following:

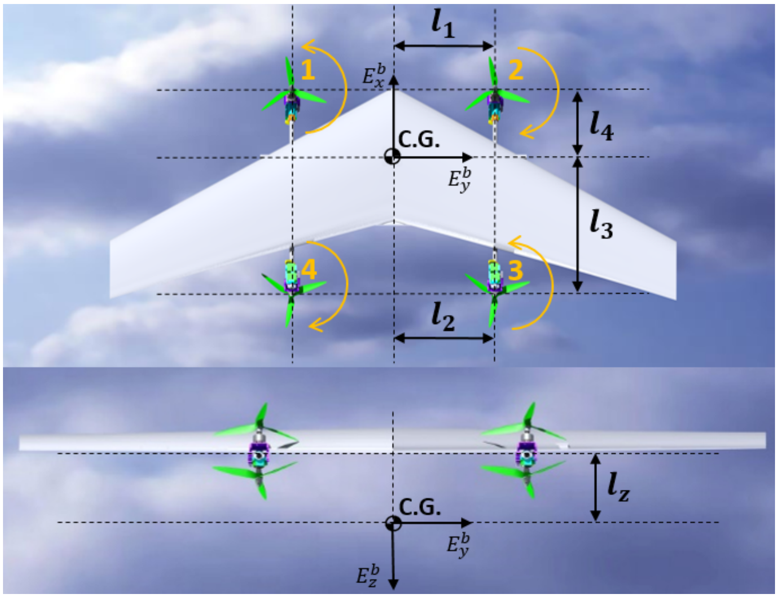


Figure 2.4: Assumptions and notation for rotor identification, disposition and rotating direction

- Earth Frame Γ_e
  - Origin: fixed to the surface of the Earth
  - x_e axis: positive in the direction of north
  - y_e axis: positive in the direction of east
  - z_e axis: positive towards the center of the Earth
- Body Frame Γ_b
  - Origin: airplane center of gravity
  - x_b axis: positive out the nose of the aircraft in the plane of symmetry of the aircraft
  - y_b axis: perpendicular to x_b and z_b, positive out of the right wing
  - z_b axis: perpendicular to the x_b axis, in the plane of symmetry of the aircraft, positive below the aircraft
- Propeller Frame Γ_p^i
  - Origin: center of rotation of the rotor, rotor number identified by the index i
  - x_p axis: positive out the nose of the aircraft(in hovering configuration) and perpendicular to z_p

  - y_p axis: perpendicular to x_p and z_p, positive pointing the right direction of the aircraft in hovering configuration
  - z_p axis: aligned with the motor axis, pointing in the opposite thrust direction

- Wind Frame Γ_w
  - Origin: airplane center of gravity
  - x_w axis: positive in the direction of the velocity vector of the aircraft relative to the air
  - y_w axis: perpendicular to x_w and z_w, positive to the right
  - z_w axis: perpendicular to the x_w axis, in the plane of symmetry of the aircraft, positive below the aircraft

An overview of the main reference frames used in the model is also shown in Fig. 2.5. The transformation matrix from the frame Γ_b to Γ_e is derived from the classical ZYX composition of elementary Euler angles rotation and is reported as follows:

$$R_{eb} = \begin{bmatrix} c_\theta c_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ c_\theta s_\psi & c_\phi c_\psi + s_\phi s_\theta s_\psi & -s_\phi c_\psi + c_\phi s_\theta s_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{2.1}$$

Such that:

$$\begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix} = R_{eb} \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} \tag{2.2}$$

Notice that c and s are the abbreviation respectively of the cos and sin functions.
The transformation matrix from the frame Γ_p to Γ_b is derived from the YX composition of elementary rotation of the tilting angles g and b expressed in Fig. 2.5. The final matrix R_{bp} is the following:

$$R^i_{bp} = \begin{bmatrix} c(b^i) & 0 & s(b^i) \\ s(g^i)s(b^i) & c(g^i) & -s(g^i)c(b^i) \\ -c(g^i)s(b^i) & s(g^i) & c(g^i)c(b^i) \end{bmatrix} \tag{2.3}$$

Such that:

$$\begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = R^i_{bp} \begin{pmatrix} x^p \\ y^p \\ z^p \end{pmatrix}_i \tag{2.4}$$

Where the suffix i indicates the rotor number. This specification is necessary since the rotors can tilt independently from each other.
Finally, the transformation matrix from the frame Γ_w to Γ_b is derived

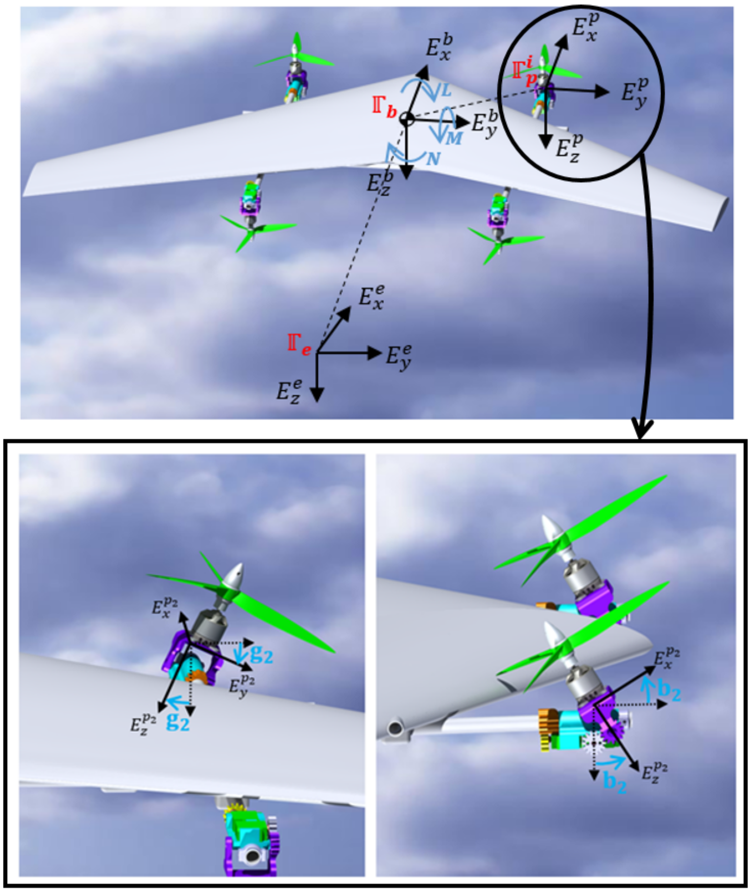


Figure 2.5: Overview of the Earth, Body and Propeller frames.

from the YZ composition of elementary rotation of the slide slip angle and the angle of attack. The final matrix R_{bw} is the following:

$$R_{bw} = \begin{bmatrix} c(\alpha)c(\beta) & -c(\alpha)s(\beta) & -s(\alpha) \\ s(\beta) & c(\beta) & 0 \\ s(\alpha)c(\beta) & -s(\alpha)s(\beta) & c(\alpha) \end{bmatrix} \tag{2.5}$$

Where α is the angle of attack and β is the sideslip angle.

2.3.2. EQUATIONS OF MOTION

After having defined the frames and the rotational matrices, it is possible to analyze all the forces and moments acting on the vehicle with the aim of determining the equations of motion. Considering the assumptions given above, the equations of motion for the vehicle are the following:

$$\begin{cases} \ddot{P}_e = \frac{1}{m}\left(F^p + F^a\right) + g\hat{z}_e \\ \dot{\omega} = I_b^{-1}\left(-\omega \times I_b\omega + M^t + M^d + M^i + \right. \\ \qquad \left. + M^p + M^a + M^r + M^{tilt}\right), \end{cases} \tag{2.6}$$

Where $\ddot{P}_e$ are the linear acceleration in the earth reference frame and $\dot{\omega}$ are the derivative of the vehicle body rate.

Each term in equation (2.6) refers to a specific effect and can be analyzed in detail as follows:

- F^p : Forces produced by the propeller thrust rotated to the earth frame:

$$F^p = \sum_{i=1}^{N} R_{eb} R_{bp}^i \begin{pmatrix} 0 \\ 0 \\ -K_p^T \Omega_i^2 \end{pmatrix} \tag{2.7}$$

  Where K_p^T is the thrust coefficient of the motor and Ω_i is the rotational speed of the motor. The thrust coefficient of the motor was identified by bench test of the motor-propeller system.

- F^a : Aerodynamic forces produced by the vehicle in the earth frame:

$$F^a = R_{eb} R_{bw} \begin{pmatrix} -D^{wb} \\ Y^{wb} \\ -L^{wb} \end{pmatrix} \tag{2.8}$$

  Where D_{wb}, Y_{wb} and L_{wb} represent the simplified forces acting on the vehicle in the wind reference frame and can be expressed as follows [6]:

$$\begin{pmatrix} D^{wb} \\ Y^{wb} \\ L^{wb} \end{pmatrix} = Q \begin{pmatrix} C_{D0} + k_{cd}(C_{L0} + C_{L\alpha}\alpha)^2 \\ C_{Y\beta}\beta \\ C_{L0} + C_{L\alpha}\alpha \end{pmatrix} \\ Q = \frac{1}{2}\rho S V_{tot}^2 \tag{2.9}$$

  Where ρ is the air density, S is the wing surface and V_{tot} is the airspeed.

- M^a : Aerodynamic moments produced by the vehicle in the body frame:

$$
M^{a} = \begin{pmatrix} M_L^{wb} \\ M_M^{wb} \\ M_N^{wb} \end{pmatrix} = \\
= Q \begin{pmatrix} \bar{b}(C_{l0} + C_{l\beta}\beta + \frac{\bar{b}}{2V_{tot}}(C_{lp}p + C_{lr}r)) \\ \bar{c}(C_{m0} + C_{m\alpha}\alpha) \\ \bar{b}(C_{np}\frac{\bar{b}}{2Vtot}p + C_{nr}\frac{\bar{b}}{2V_{tot}}r) \end{pmatrix} \tag{2.10}
$$

where M_L^{wb}, M_M^{wb} and M_N^{wb} represent respectively the vehicle roll, pitch and yaw moment induced by the wind [6]. The coefficients for the aerodynamic equations can be identified through test flight, CFD analysis or from geometrical vehicle properties [7]. Note that some coefficients are not present in the expression and considered negligible, because the vehicle is tailless and the CG is located very close to the quarter chord point [7].

- M^t : Torque generated by the rotors due to the propeller thrust:

$$
M^t = \sum_{i=1}^{N} \left(R_{bp}^i \cdot \begin{bmatrix} 0 \\ 0 \\ -K_p^T \Omega_i^2 \end{bmatrix} \right) \times \begin{pmatrix} l_x^i & l_y^i & l_z^i \end{pmatrix} \tag{2.11}
$$

Where (l_x^i, l_y^i, l_z^i) are the coordinate of the i-th rotor in the body reference frame.

- M^d : Torque generated by the rotors due to the propeller drag:

$$
M^d = \sum_{i=1}^{N} -R_{bp}^i \begin{pmatrix} 0 \\ 0 \\ K_p^M \Omega_i^2 \end{pmatrix} (-1)^i \tag{2.12}
$$

Where K_p^M is the torque coefficient of the motor and can be evaluated in the same way as K_p^T.

- M^i : Torque generated by the propeller inertia due to the propeller rate change:

$$
M^i = \sum_{i=1}^{N} -J_p R_{bp}^i \begin{pmatrix} 0 \\ 0 \\ \dot{\Omega}_i \end{pmatrix} (-1)^i \tag{2.13}
$$

- M^p : Torque generated by the rotor precession term due to the tilting rotation:

$$M^p = \sum_{i=1}^{N} R^i_{bp} \begin{pmatrix} 0 \\ 0 \\ J_p\Omega_i \end{pmatrix} \times \begin{pmatrix} \dot{g}_i \\ \dot{b}_i \\ 0 \end{pmatrix} (-1)^i \tag{2.14}$$

- M^{tilt} : Torque generated by the rotor inertial term due to the tilting rotation:

$$M^{tilt} = \sum_{i=1}^{N} R^i_{bp} \begin{pmatrix} \ddot{g}_i I^{tilt}_{xx_i} \\ \ddot{b}_i I^{tilt}_{yy_i} \\ 0 \end{pmatrix} (-1)^i \tag{2.15}$$

- M^r : Inertial term of the rotor due to the vehicle rates

$$M^r = \sum_{i=1}^{N} -\omega \times \left(R^i_{bp} I_p \begin{pmatrix} 0 \\ 0 \\ \Omega_i \end{pmatrix} (-1)^i \right) \tag{2.16}$$

  Where I_p is the propeller inertia in the propeller frame, in this case simplified to:

$$I_p = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & J_p \end{pmatrix}$$

2.3.3. SERVO LOAD EXPRESSION

It is possible to calculate the effective torque load on the servomotors. The total load on the servos is the addition of the rotor precession term (in propeller frame) and the inertial term related to the tilting moment of inertia. For every rotor, the servo load is thus given by:

$$\begin{aligned} S^i_{az} &= G^a_r(-J_r\Omega_i\dot{b}_i(-1^i) + \ddot{g}I^{tilt}_{xx_i}) \\ S^i_{el} &= G^e_r(-J_r\Omega_i\dot{g}_i(-1^i) + \ddot{b}I^{tilt}_{yy_i}) \end{aligned} \tag{2.17}$$

Where S^i_{az} and S^i_{el} are respectively the torque on the azimuth and elevation servo of the i-th rotor while G^a_r and G^e_r are the azimuth and elevation gear ratio between the servo and the rotating mechanism. Notice that in that expression, the torque related to the mass of the tilting system, as well as the damping term due to the propeller aerodynamic effects are not considered.

2.4. ACTUATOR MODEL IDENTIFICATION

To complete the vehicle modeling, the actuators were identified, modeled and mapped through several tests. Two types of actuators are present in the UAV. The first type of actuator is the propulsion system whose dynamics are driven mainly by the propeller-motor interaction. Specifically, a T-motor AT-2312 1150kV was used, coupled with a nylon 9x4.7 propeller, a Turnigy MultiStar 30A BLHeli-S Electronic Speed Controller(ESC) and a 3S 5000mAh 25C Turnigy LiPo battery.

The second type of actuator is the servo, found in the rotor tilt mechanism. This system is driven by two SAVOX 1232MG servos and its dynamics are related to the tilting system inertia, to the rotor precession term and to the propeller damping.

2.4.1. EXPERIMENTAL SETUP

To analyze the motor dynamics characteristics, the HOBBYWING RC Model Brushless RPM Sensor AC683 was used. This sensor is directly connected on two motor poles and generates a PWM signal associated to the rotational speed of the motor. For the motor step test, an arduino DUE was used to generate the Pulse Width Modulation (PWM) input to the ESC and to simultaneously monitor the RPM sensor output.

Concerning the tilting rotor dynamics, the most straightforward way to monitor the tilting angles would be through the servomotor feedback. However, the servos do not provide any telemetry information and therefore an MPU9255 Inertial Measurement Unit (IMU) rigidly attached to the tilting mechanism was used to record the tilting orientation. An arduino DUE was used to send commands to the actuators, to monitor the IMU and to log the data onto an external SD card. A scheme of the testing setup is visible in Fig.2.6.

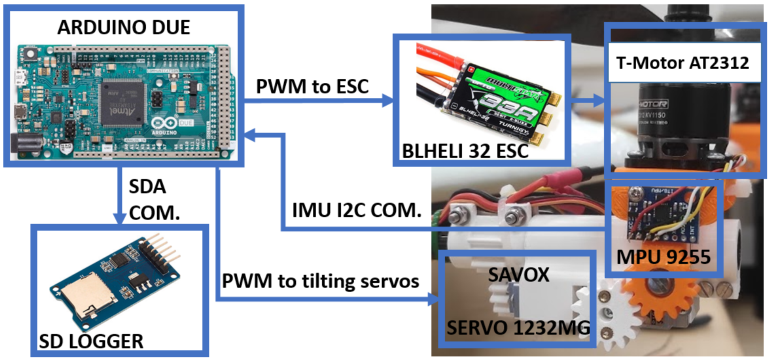


Figure 2.6: Scheme of the testing setup for the tilting mechanism dynamics acquisition.

2.4.2. RESULTS

For the motor dynamics identification, few step input were fed to the ESC. Fig. 2.7 shows the evolution of the fed PWM value, together with the recorded motor rotational speed.

Similar step inputs were fed to the azimuth and elevation servos of the tilting system[2]. In Fig. 2.8 the elevation and azimuth tilting angle response to a 60 degree step is shown while the motor (with propeller) was running. To evaluate the effect of the turning propeller on the tilting dynamics, the experiment was performed with the motor running at 25% and 100% thrust level. In Fig. 2.9 the same tilting angle dynamics is reported for a 70 degrees step input in the elevation axis. An interesting observation coming from the tilting test is the presence of the gyroscopic precession term of the rotor tilting. These dynamics are visible in the form of azimuth angle perturbation in the tilting elevation test and vice-versa. The effect increases with increasing throttle value, as analyzed in the servo load expression presented in Equation 2.17.

The motor and servomotor dynamics were modeled as first order system with delay:

$$H(s) = e^{-\tau_d \cdot s} \cdot \frac{\omega_c}{s + \omega_c} \tag{2.18}$$

where τ_d is the actuator delay and ω_c is the actuator cutting frequency. The actuators dynamics characteristics identified through the actuator tests for the servomotors and the motor dynamics are summarized in Table 2.4. Even though the servomotors used are the same, the azimuth and elevator dynamics are slightly different due to the different gear ratio used.

Motor delay τ_d^m	2 mS.
Elevation servo delay τ_d^e	15 mS.
Azimuth servo delay τ_d^a	15 mS.
Motor dynamics ω_c^m	30 rad/s
Azimuth servo dynamics ω_c^a	30.7 rad/s
Elevation servo dynamics ω_c^e	26.2 rad/s

Table 2.4: Actuator dynamics characteristics

2.5. WIND REJECTION CAPABILITIES AND SIMULATION

In this Section, using the mathematical model and the actuator dynamics as a reference, a set of simulations are performed to characterize and compare the performance of the presented over-actuated quad-plane

[2] https://youtu.be/QNZSxoJ1Gto

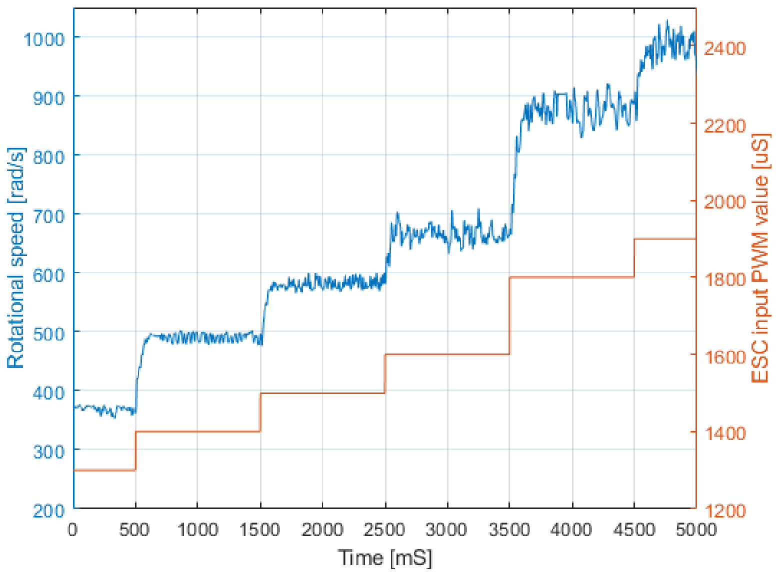


Figure 2.7: Test for the motor dynamics characterization. The red curve shows the PWM value fed to the ESC while the blue curve shows the rotational speed evolution of the motor.

with a conventional quad-plane design. Firstly, from the equations of motion established in Section 3, it is possible to characterize the system of equations at equilibrium for a stationary, symmetric flight condition ($\beta = 0$) for both the proposed and the conventional quad-plane in the hovering configuration. In Eq. 2.19 and Eq. 2.20, the system of equations at equilibrium for the conventional and for the over-actuated quad-plane vehicle are shown respectively.

$$\begin{cases} (F_1+F_2+F_3+F_4)sin(\theta) = -D_{wb} \\ (F_1+F_2+F_3+F_4)cos(\theta) = mg - L_{wb} \\ (F_3+F_4)l_3 - (F_1+F_2)l_4 = M_{wb} \\ F_1 = F_2 \\ F_3 = F_4 \end{cases} \tag{2.19}$$

$$\begin{cases} (F_1+F_2+F_3+F_4)sin(b_{tot}+\theta) = -D_{wb} \\ (F_1+F_2+F_3+F_4)cos(b_{tot}+\theta) = mg - L_{wb} \\ (F_3+F_4)l_3cos(b_{tot}) - (F_1+F_2)l_4cos(b_{tot}) = M_{wb} \\ F_1 = F_2 \\ F_3 = F_4 \end{cases} \tag{2.20}$$

Here D_{wb} and L_{wb} are already expressed in the earth reference frame and F_i represents the motor thrust in the propeller reference frame. b_{tot} represents the elevation tilt angle, assumed to be equal for all the rotors.

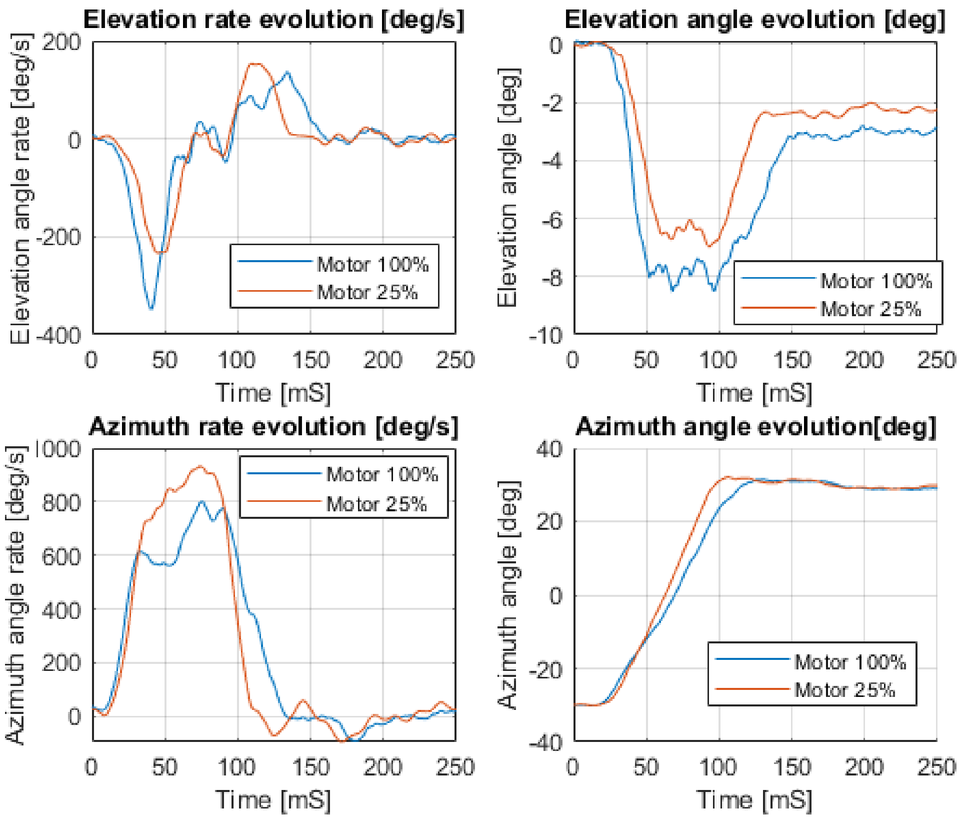


Figure 2.8: Tilting angles and rate evolution to an azimuth step response of around 60 degrees at different motor power level.

From these equations it is possible to determine the equilibrium pitch angle θ_{eq} (in conventional quad-plane configuration) and elevator tilting angle b_{eq} (in the over-actuated quad-plane configuration, supposing $\theta = 0$) to counteract a specific constant wind speed without losing altitude.

The lift and moment coefficients for the aerodynamics used in the simulations are derived from the XFLR5 analysis, the zero-lift drag coefficient C_{D0} is obtained from the wind tunnel test and the k_{cd} coefficient is calculated as follows: $k_{cd} = \frac{1}{\pi A_r e}$ [7]. An overview of the aerodynamic coefficients used for the simulation are reported in Table 2.5.

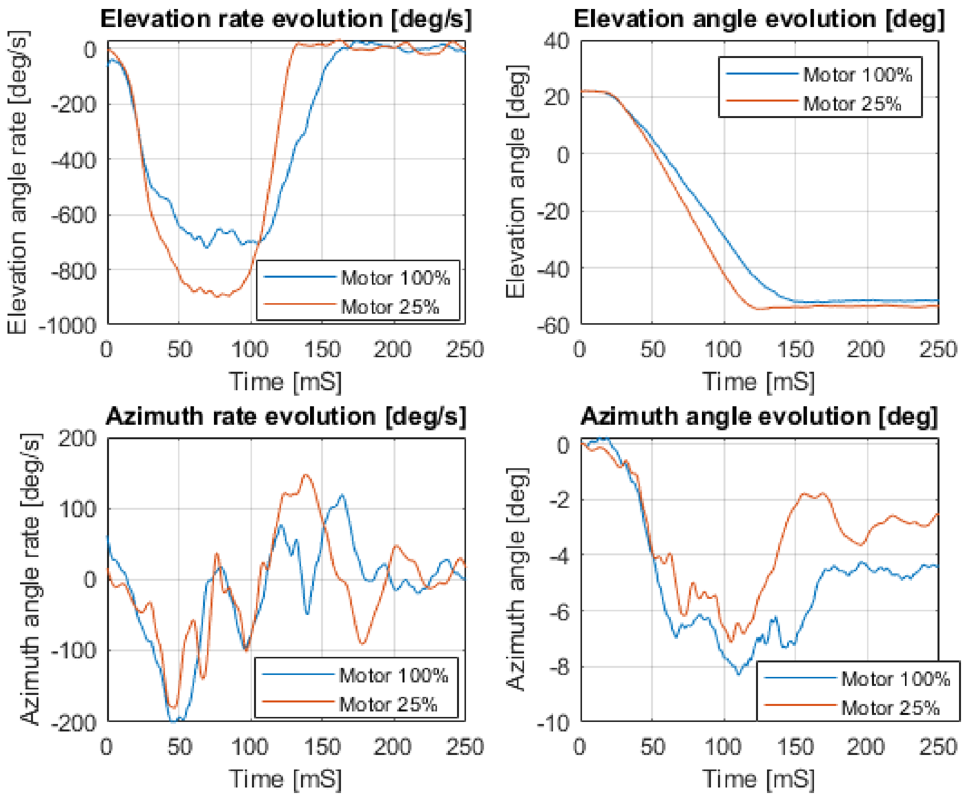


Figure 2.9: Tilting angles and rate evolution to an elevation step response of around 60 degrees at different motor power level.

Cl_α	4.2 rad^{-1}
Cl_0	-0.05
Cm_α	-0.58 rad^{-1}
Cm_0	0.002
k_{cd}	0.08
C_{D0}	0.55

Table 2.5: Aerodynamic coefficients for the vehicle

2.5.1. SIMULATION SETUP AND RESULTS

A diagram of the control system used to simulate the conventional quad-plane and the over-actuated quad-plane vehicle is reported respectively in Fig. 2.10 and Fig. 2.11. For clean comparison, the same altitude and attitude PID controller were used for the two vehicle configurations. The PID gains used for the altitude are $P = 600\frac{rad/s}{rad}$, $I = 100\frac{rad/s}{rad \cdot s}$ and

$D = 500\frac{rad/s}{rad/s}$, while the PID gains used for the theta controller are $P = 390\frac{rad/s}{rad}$, $I = 500\frac{rad/s}{rad\cdot s}$ and $D = 110\frac{rad/s}{rad/s}$. The PID coefficients for the theta controller were tuned using the open loop bode plot of the equivalent linear system as a main reference, ensuring a final open loop dynamics of 12.8 rad/s and a phase margin of 30 deg. The saturation points for the PID controllers on the desired motor rotational speed are $[-600, 600]$ rad/s for the theta controller and $[-500, 500]$ rad/s for the altitude controller.

The equilibrium point for theta (θ_{eq}) and the elevator tilting angle (b_{eq}) are a function of the wind speed and are computed using Eq. 2.19 and Eq. 2.20 respectively.

In Fig. 2.12, the simulated acceleration response with a frontal step wind of 5 m/s is shown for both the conventional and the over-actuated quad-plane. Similarly, Fig. 2.13 shows an acceleration test performed in absence of wind. During the tests, the conventional and over-actuated vehicles are requested to reach the previously determined θ_{eq} and b_{eq} condition. Considering the acceleration test in absence of wind, the acceleration peak is reached at 0.22 seconds on the over-actuated vehicle and at 0.75 seconds on the conventional quad-plane. With these simulations we can therefore conclude that tilting the rotors instead of the whole vehicle leads to a much faster acceleration response and a more effective wind gust rejection capability.

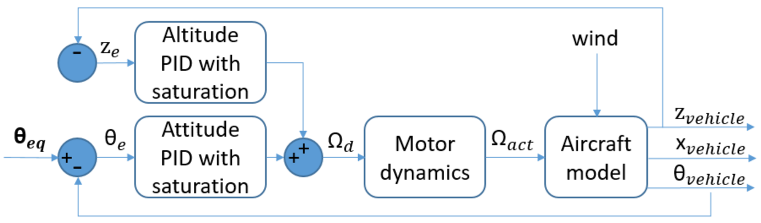


Figure 2.10: Control scheme for the conventional quad-plane configuration.

2.6. FLIGHT TEST RESULTS

In this section, the results of the wind tunnel experiments are analyzed. The main focus of the test flights was to validate the simulated response and to extend the vehicle gust response also to the lateral gust scenario. The wind tunnel experiments were also used to estimate the zero-lift drag coefficient of the vehicle.

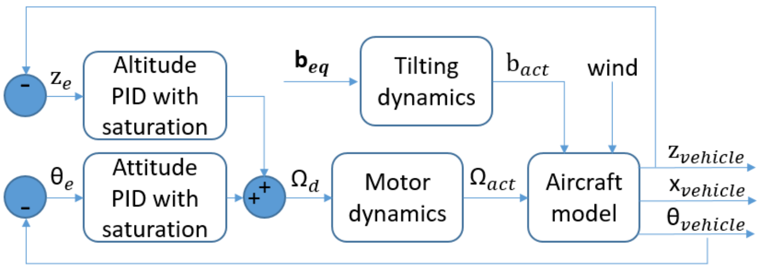


Figure 2.11: Control scheme for the over-actuated quad-plane vehicle. *b* is the overall elevation tilting angle of the rotors, assumed to be the same for all the four rotors.

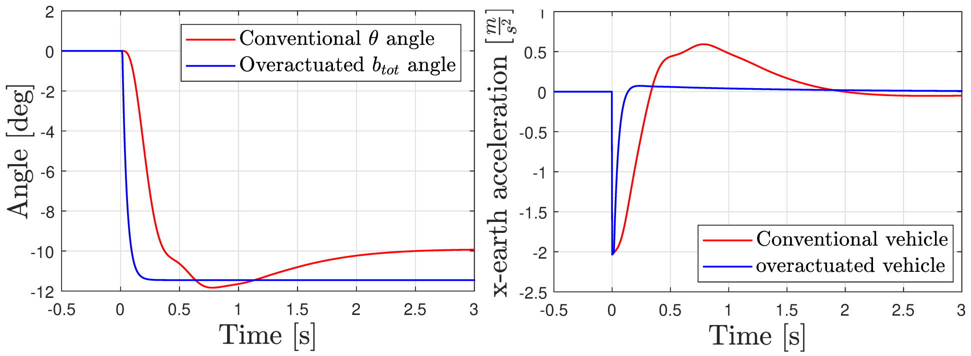


Figure 2.12: Simulation result to a 5 m/s frontal wind step. The blue curve represent the over-actuated vehicle while the red curve represent the conventional quad-plane.

2.6.1. EXPERIMENTAL SETUP

For the wind tunnel experiment, the Paparazzi open-source software[3] was used, running on a Holybro Pixhawk 4 flight computer. In order to simulate a discrete wind step response in the wind tunnel, the vehicle was introduced into the flow region from the bottom, to minimize the attitude change during the transition phase. For a better understanding of the wind tunnel testing setup, a picture of the wind tunnel flight test experiment is shown in Fig. 2.14.

For the over-actuated quad-plane setup, an INDI controller [8] was used to control the attitude of the vehicle using only the motors, the tilting system was controlled using a PD control with feedback of the vehicle position.

For the conventional quad-plane setup, the tilting angle g_i and b_i

[3]https://wiki.paparazziuav.org/wiki/Main_Page

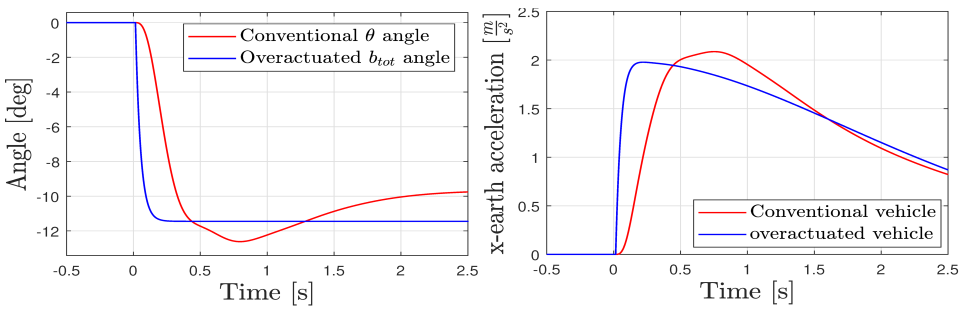


Figure 2.13: Simulation result to a forward acceleration command in absence of wind. The blue curve represent the over-actuated vehicle while the red curve represent the conventional quad-plane.

were set to zero for all the rotors. The attitude was controlled with the same INDI controller as in the over-actuated configuration. In that case, the position of the vehicle was controlled via an outer PD loop through attitude changes. For both the conventional and over-actuated quad-plane configuration, the yaw was controlled by means of differential azimuth tilting angle.

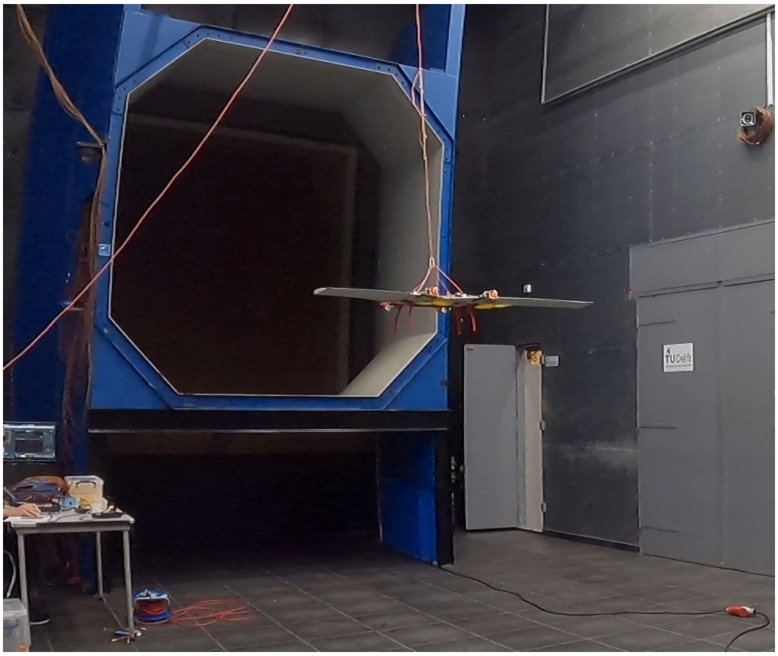


Figure 2.14: Picture of the wind tunnel test experiment.

2.6.2. TEST RESULTS

Frontal wind gust and lateral wind gust response experiments were performed at a wind speed of 5 m/s for both the over-actuated and the conventional quad-plane configuration[4]. Figure 2.15 shows the results of the frontal wind gust experiment. It can be observed that when the tilting system is active, the vehicle is able to hold its current position with a maximum displacement of around 20 cm and a maximum pitch perturbation of 5 degrees. On the other hand, the conventional quad-plane vehicle reached an equilibrium pitch angle of around 10 degrees with a maximum displacement of more than 1.4 meters.

Similar behavior can be observed for the lateral wind gust flight test, displayed in Fig. 2.16. Interestingly, during the lateral gust flight tests both the over-actuated and conventional quad-plane vehicle suffered from low frequency oscillations in the roll dynamics. This may be related to the lower roll authority of the vehicle with the present motor disposition. Saturation of the front right motor is observed during the maneuver.

Finally, considering the frontal gust tests of the conventional quad-plane, it is possible to identify an equilibrium pitch angle that the vehicle reaches to react to the wind gust. By knowing the wind speed, and observing stationary condition for the flight test, we can replace the equilibrium theta and the wind speed in equation (2.19). Substituting the lift and moment coefficients with the XFLR5 analysis result and the induced drag coefficient with $k_{cd} = \frac{1}{\pi A_r e}$, it is possible to estimate the vehicle zero-lift drag coefficient C_{D0}. The result of the fitted drag coefficient for the vehicle is equal to $C_{D0} = 0.55$. This completes the longitudinal aerodynamic model identification of the UAV.

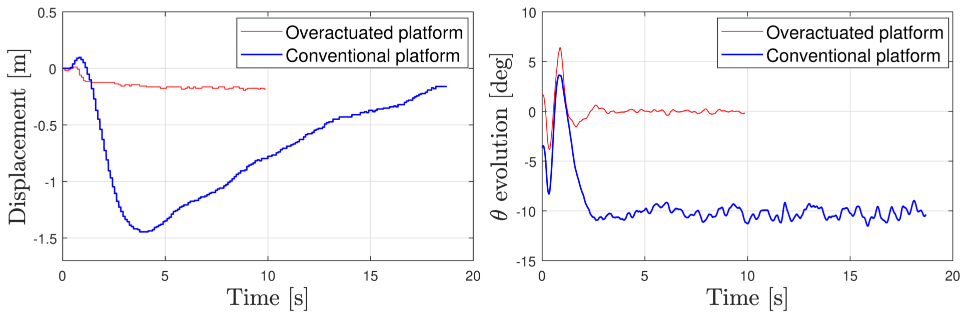


Figure 2.15: Position displacement (left) and pitch evolution (right) in response to a 5 m/s frontal wind gust. The blue curve represents the over-actuated vehicle response, while the red curve represents the conventional quad-plane response.

[4]https://youtu.be/xCXokABB4B0

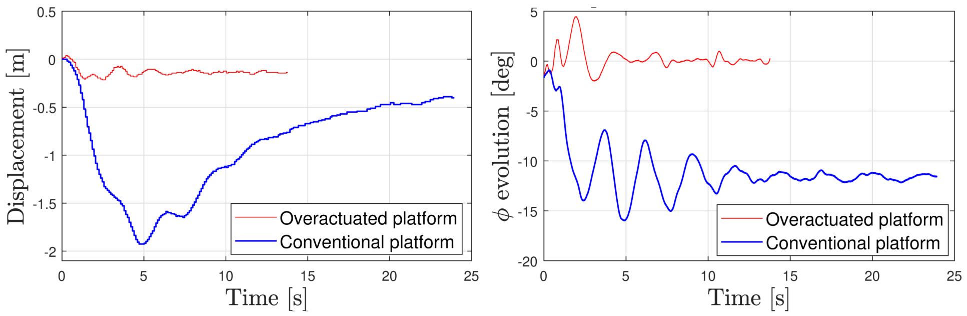


Figure 2.16: Position displacement (left) and roll evolution (right) in response to a 5 m/s lateral wind gust. The blue curve represents the over-actuated vehicle response, while the red curve represents the conventional quad-plane response.

2.7. CONCLUSIONS

In this chapter a novel quad-plane design capable of full 6DOF control was presented and modeled. Through wind tunnel experiments and simulations, the effectiveness of the rotor tilting strategy to handle rapid and unpredictable wind gusts and to achieve rapid linear accelerations in any direction was proved. In particular, considering the simulation results, the over-actuated quad-plane is 3.4 times more responsive than the conventional quad-plane in achieving linear acceleration. The spot hover capability in gusty environment was tested through a wind tunnel experiment simulating a frontal 5 m/s gust, leading to a displacement of just 0.2 meters for the over-actuated vehicle compared to the 1.4 meters of the conventional quad-plane. Similar behavior was observed in the lateral gust experiment. Dual-axis tilting rotors are therefore a good option to enhance the wind rejection capabilities of a conventional quad-plane vehicle.

Future research will focus on the development of a tailored control system for the UAV, able to homogeneously control the vehicle in both the forward and the hovering flight phase. Finally, flight performance of the vehicle will be tested at different unconventional attitude conditions.

REFERENCES

[1] A. R. Wang. “Conceptual Design of a QuadPlane Hybrid Unmanned Aerial Vehicle”. In: *AIAA Student Conference Region VII-AU*. 2017. doi: 10.13140/RG.2.2.23090.84163.

[2] G. R. Flores, J. Escareño, R. Lozano, and S. Salazar. “Quad-Tilting Rotor Convertible MAV: Modeling and Real-Time Hover Flight Control”. In: *Intelligent Robot Systems* (2012). doi: 10.1007/s10846-011-9589-x.

[3] A. Chen, Z. DaiBing, Z. JiYang, and L. TengXiang. “A New structural configuration of tilting rotor unmanned aerial vehicle modeling”. In: *35th Chinese Control Conference*. 2016. doi: 10.1109/ChiCC.2016.7553680.

[4] H. Zhang, B. Song, H. Wang, and J. Xuan. “A method for evaluating the wind disturbance rejection capability of a hybrid UAV in the quadrotor mode”. In: *International Journal of Micro Air Vehicles* (2019). doi: 10.1177/1756829319869647.

[5] A. B. Junaid, A. D. Sanchez, J. B. Bosch, N. Vitzilaios, and Y. Zweiri. “Design and Implementation of a Dual-Axis Tilting Quadcopter”. In: *Journal of Robotics* (2018). doi: 10.3390/robotics7040065.

[6] V. Klein and E. A. Morelli. *Aircraft System Identification: Theory and Practice*. Chapter 3. Blackburg, Virginia, 2006.

[7] F. O. Smetana, D. C. Summey, and W. D. Johnson. *Riding and Handling Qualities of Light Aircraft - A Review and Analysis*. Tech. rep. Washington DC: National Aeronautics and Space Administration, Mar. 1972.

[8] E. J. J. Smeur, P. Chu, and G. de Croon. “Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles”. In: *Journal of Guidance, Control, and Dynamics* (2015). doi: 10.2514/1.G001490.

3
LOW AIRSPEED flIGHT CONTROL SYSTEM DESIGN

The preliminary results of the dual-axis tilting rotor quad-plane presented in Chapter 2 were obtained using a simple, non-optimized Proportional-Integrative-Derivative (PID) controller that actuated the tilting system based solely on position error. While this approach confirmed basic functionality, it did not account for the vehicle's nonlinear dynamics. To improve robustness against external disturbances, a more advanced control strategy is required—one that explicitly considers the system's nonlinear behavior.

This chapter begins by addressing the limitations of traditional control allocation techniques when applied to vehicles with pronounced nonlinearities, such as the dual-axis tilting rotor quad-plane. Building on the incremental structure of INDI, a nonlinear programming-based control allocation method is integrated into the INDI framework to improve control performance. The resulting controller is tested onboard the vehicle, demonstrating smooth and precise flight in the low-airspeed regime, and confirming its effectiveness in managing coupled and nonlinear actuation.

This chapter is based on the following article:
A. Mancinelli, B.D.W. Remes, G.C.H.E. De Croon and E.J.J. Smeur, "Real-Time Nonlinear Control Allocation Framework for Vehicles with Highly Nonlinear Effectors Subject to Saturation," Journal of Intelligent Robotic System 108, 67 (2023).

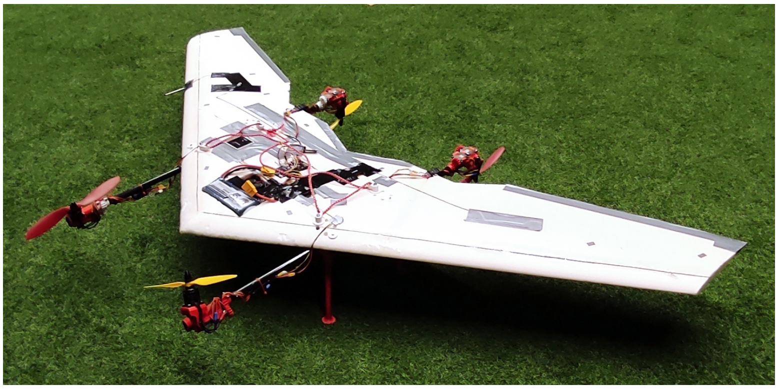


Figure 3.1: A picture of the dual-axis tilting rotor quad-plane used as test bench for the Control Allocation (CA) methods.

3.1. INTRODUCTION

In the last twenty years, a boom in the UAV industry has been experienced all over the world. UAVs prove to be flexible vehicles for several different tasks ranging from Mapping & Surveying to medical delivery. Of particular interest are the so-called 'hybrid' vehicles, able to combine the cruise efficiency of conventional fixed wing planes with the hovering capability typical of helicopters. This category of vehicles is characterized by the presence of a wing surface used to generate lift in the forward part of the flight. However, the presence of a wing brings significant disadvantages during the vertical take-off and landing phase. During vertical takeoff and landing, the wing surface interacts with the wind, generating strong and unpredictable forces and moments on the aircraft. If a precision landing has to be performed, these forces and moments need to be opposed by the vehicle.

A possible way to increase a hybrid air vehicle's gust disturbance capabilities could be through the addition of extra actuators on the vehicle. Using tilting mechanisms can enable the vehicle to direct the thrust and moments generated by the motors to oppose those induced by the wind. An example of this was proposed in [1], where the gust rejection capabilities of a conventional quad-plane were compared to the one of a dual-axis tilting rotor quad-plane, showing how effective the thrust vectoring capability is to minimize the influence of the wind on the vehicle in the hovering configuration.

Unfortunately, the large number of actuators increases the complexity of the CA problem for these vehicles. CA is the problem of distributing control effort over a set of actuators to achieve certain desired control forces and moments. These vehicles must solve the CA problem optimally to fully exploit their capabilities.

Several CA problem statements and resolution methods have been proposed over time. A good overview of the different CA methods is available in [2–4]. The unconstrained CA problem is most commonly solved with the pseudo inverse method, which inverts the mapping from control commands to the desired motion effects [5]. As for the constrained CA problem, redistributed pseudo-inverse [6] and direct allocation [7] are a few CA strategies based on the pseudo-inverse method, able to optimize the computed solution in case of actuator saturation. More elaborate iterative constrained CA methods, formulate the problem as a constrained linear or quadratic programming optimization problem [8–10]. Note that all these problem formulations presented so far still share the same linearized control effectiveness assumption.

Regrettably, when vehicles with strong nonlinear behavior are employed, linearized effectiveness approaches tend to fail. This happens especially when a large control objective needs to be realized with ineffective non-linear actuators.

A much more effective method, potentially capable of dealing with such a problem, is the nonlinear programming method. The very first attempt to approach the CA problem with a nonlinear programming method was presented in [11], with the aim of solving the allocation problem for reentry vehicles. During reentry, these vehicles are exposed to very high angle of attack, where the nonlinear dynamics of the effectors are too evident that the CA problem cannot be properly solved with a linearized approach. Simulation results confirmed the weaknesses of the linearized control effectiveness approaches in solving the CA problem for these kinds of vehicles, highlighting the potential of the nonlinear programming method.

Following the same idea, the nonlinear resolution of the CA problem was also applied to overactuated ground vehicles [12–14], gaining more and more popularity over time. Recently, learning-based approaches for the resolution of the nonlinear CA problem were proposed with the aim of reducing the computational load [15, 16]. Overall, although computationally intensive, the nonlinear programming approach for the resolution of the CA problem proved to be an extremely flexible and powerful method to determine control effectors solution for highly nonlinear and overactuated vehicles. However, so far, limited computational power of embedded computers, and the necessity of solving the problem in real time, prevented this CA method to be implemented and tested on a flying vehicle.

In this chapter, we address the CA problem of vehicles with highly nonlinear control effectors, such as thrust vectoring UAVs. Within the chapter we initially show the ineffectiveness of state-of-the-art linearized effectiveness approaches for these type of vehicles. Then, we propose a method to solve in real time the nonlinear CA problem using a Sequential Quadratic Programming (SQP) algorithm. The proposed method uses the nonlinear Equations Of Motion (EOM) inversion to compute the actuator solution, thus overcoming the control effectiveness linearization limitation. The control framework used to implement such a CA method is a modified INDI, adapted to include the incremental law through nonlinear EOM evaluation.

To prove the applicability of such a control strategy on a real time and computationally constrained platform, the control framework was implemented and tested on the dual-axis tilting rotor quad-plane depicted in Figure 3.1. The vehicle has a 2.4 kg take-off mass and features 8 servos and 4 motors for a total of 12 actuators controlling the 6 degrees of freedom. A commercial Raspberry Pi 4B Single Board Computer (SBC) was used to solve the Nonlinear CA problem in real time onboard the drone. Thanks to hardware and software optimization, it was possible to run the algorithm computing the actuator solution at an average refresh rate of 350 hz, while using just one of the four available cores of the SBC. Another of the available cores was dedicated to the serial communication with the main Flight Control Board (FCB), leaving a total of 2 free cores available for running algorithms supporting additional tasks.

To summarize, the research highlights of the chapter are:

- We demonstrated the inapplicability of state-of-the-art linearized effectiveness Control Allocation algorithms on overactuated UAVs with thrust vectoring capability.

- We proposed a Control Allocation algorithm based on Nonlinear EOM inversion able to adequately control overactuated UAVs with thrust vectoring capability.

- We demonstrated the applicability of the proposed Nonlinear Control Allocation method on a flying vehicle by implementing the algorithm on a commercial SBC onboard of a dual-axis tilting rotor quad-plane.

The chapter is structured as follows: In section 3.2, the nonlinear system is presented and the INDI control framework is introduced. Within this section, the linear and quadratic programming formulation of the CA problem are also derived. In section 3.3, the Nonlinear CA algorithm and the nonlinear INDI control framework are presented. In section 3.4, the Nonlinear CA module is firstly implemented in simulation and then tested on the flying double axis tilting rotor quad-plane. In section 3.5, some limitations of the Nonlinear CA methods are discussed. Finally, in section 3.6 conclusions are drawn.

3.2. PRELIMINARIES

3.2.1. SYSTEM DESCRIPTION AND INDI DERIVATION

Considering the generic nonlinear system:

$$\begin{aligned}\dot{x} &= f(x(t), u(t)) \\ y &= h(x(t))\end{aligned} \tag{3.1}$$

Where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control input vector and $y(t) \in \mathbb{R}^l$ is the output vector. $f_{nx1} : D_{x,u} \to \mathbb{R}^n$ is a nonlinear function representing the vehicle dynamics while $h_{lx1} : D_x \to \mathbb{R}^l$ is a nonlinear function representing the output dynamics.

For the derivation of the linearized CA methods, let's now consider the first order Taylor expansion of the first term in equation 3.1, centered around the current state x_0 and control input u_0:

$$\begin{aligned}\dot{x} &\simeq f(x_0, u_0) + \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=x_0, u=u_0} (x - x_0) \\ &\quad + \left.\frac{\partial f(x,u)}{\partial u}\right|_{x=x_0, u=u_0} (u - u_0) \\ &= \dot{x}_0 + \underbrace{F(x_0, u_0)(x - x_0)}_{F_{\Delta x}} + \underbrace{B(x_0, u_0)(u - u_0)}_{F_{\Delta u}},\end{aligned} \tag{3.2}$$

where $F_{\Delta x}$ is the state dependent term and $F_{\Delta u}$ is the control input dependent term.

Under the assumption of small controller sampling time and very fast actuators, it is possible to apply the time scale separation principle, letting us neglect the state dependent term Δx [17]. Based on the aforementioned assumptions, Equation 3.2 can than be be rewritten as follows:

$$\dot{x} \simeq \dot{x}_0 + B(x_0, u_0)(u - u_0) = \dot{x}_0 + B(x_0, u_0)\Delta u. \tag{3.3}$$

From Equation 3.3, it is now possible to derive the associated INDI control law for the vehicle. Assuming a direct expression between the state vector and the output vector (i.e. $y = h(x) = x$) and replacing $\dot{x}$ with the desired state derivative $\dot{x}_d$, the control law becomes:

$$\begin{aligned}\Delta u &= B^{\dagger}(x_0, u_0)(\dot{y}_d - \dot{y}_0) \\ u_s &= u_0 + \Delta u\end{aligned} \tag{3.4}$$

where $B^{\dagger}(x_0, u_0)$ represents the generalized inverse of the effectiveness matrix, linearized around the current state x_0 and current control input u_0. $\dot{y}_d - \dot{y}_0$ is the output derivative error $\dot{y}_e$, representing in this case the vehicle acceleration. The desired output derivative $\dot{y}_d$ is also known in literature as pseudo-control ν. The term u_s represents the control input solution of the CA problem.

Concerning the current output derivative value $\dot{y}_0$, it represents the vehicle linear and angular accelerations and a sensor-based estimation can be employed [18]. A scheme of the INDI control framework is depicted in Figure 3.2.

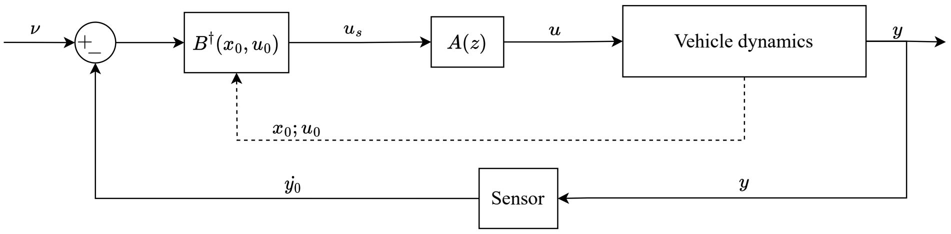


Figure 3.2: Scheme of the INDI control framework. In the diagram, the dashed lines represent variables estimated through sensor (for the vehicle state) or actuator model (for the current actuator state). The term $A(z)$ represents the actuator dynamics transfer function.

For completeness, it is possible to manipulate the terms in Equation 3.3 to directly compute the control input solution, rather than the control input increment:

$$u_s - u_0 = B^{\dagger}(x_0, u_0)(\dot{y}_d - \dot{y}_0) \tag{3.5}$$

Then, by bringing the current actuator state on the right side and including it into the inversion law it becomes:

$$u_s = B^{\dagger}(x_0, u_0)(\nu - \dot{y}_0 + B(x_0, u_0)u_0), \tag{3.6}$$

where the desired acceleration vector $\dot{y}_d$ was replaced by the pseudo-control vector ν.

Instead of minimizing the actuator solution norm, it is also possible to minimize the solution norm with respect to a desired control input state u_d [3]:

$$u_g = B^{\dagger}(x_0, u_0)\nu_g \tag{3.7}$$

with

$$\nu_g = \nu - \dot{y}_0 + B(x_0, u_0)(u_0 - u_d)$$
$$u_g = u_s + u_d$$

The problem formulation in Equation 3.7 is not yet a constrained problem and the solution cannot be optimized in case of actuator saturation. Within the chapter, we will refer to the CA strategy in Equation 3.7 as the Pseudo Inverse Unconstrained (PIU) CA algorithm.

3.2.2. QUADRATIC PROGRAMMING FORMULATION OF THE CA PROBLEM

Equation 3.3 can be reformulated as quadratic programming problem [10]. This brings two main benefits. The first benefit is the optimization of the control input solution in case of actuator saturation. The second benefit is the possibility to include multiple objectives in the cost function to be minimized during the resolution process.

Following the formulation presented by [18], the CA problem becomes:

$$\begin{gathered}
C(u) = ||W_u(u-u_d)||^2 + \gamma_\nu ||W_\nu(B(x_0,u_0)u - \nu_w)||^2 \\
= \left|\left| \begin{pmatrix} \gamma_\nu^{\frac{1}{2}} W_\nu B \\ W_u \end{pmatrix} u - \begin{pmatrix} \gamma_\nu^{\frac{1}{2}} W_\nu \, \nu_w \\ W_u u_d \end{pmatrix} \right|\right|^2 ; \\
u_s = \arg\min C(u) \\
\text{subject to} \\
u_{min} < u < u_{max}
\end{gathered} \tag{3.8}$$

with

$$\nu_w = \nu - \dot{y}_0 + B(x_0,u_0)u_0$$

where u_s is the control input solution. The two diagonal matrices W_u and W_ν represent respectively the weight for the control input and the pseudo-control. The role of these matrices will be more clear later on, when we will apply the CA methods on the flying vehicle. The term γ_ν is the scale factor assigning the preference on the two objectives included in the cost function. In order to prioritize the minimization of the control objective ν over the energy efficiency of the solution, this term should be chosen to be very high ($\gamma_\nu >> 1$). The solution of the least squares problem presented in Equation 3.8 can be efficiently determined with the active set method presented in [18].

Within this chapter, we will refer to this CA strategy as the Weighted Least Squares (WLS) CA algorithm.

3.3. NONLINEAR FORMULATION OF THE CA PROBLEM

So far, the methods presented to solve the CA problem associated with the INDI control framework make use of a linearized approximation of the actuator effectiveness. However, in some occasions, the vehicle effectors' behavior cannot be fully represented by a linear system. For those types of platforms, it is possible to formulate and solve the CA problem with an iterative nonlinear optimization process [11]. If we still assume very fast actuators, the Nonlinear CA problem can be reformulated as follows:

$$\begin{gathered} C(u) = ||\gamma_u^{\frac{1}{2}} W_u(u - u_d)||^2 + ||W_\nu(f(x_0, u) - \nu_n)||^2; \\ u_s = \arg\min C(u) \\ \text{subject to} \\ u_{min} < u < u_{max} \end{gathered} \tag{3.9}$$

with

$$\nu_n = \nu - \dot{y}_0 + f(x_0, u_0)$$

where γ_u is the control input optimality scale factor. In contrast to Equation 3.8, in this cost function, the scale factor is assigned to the control input rather than to the control objective. It is therefore suggested to assign a very small value ($\gamma_u << 1$) to keep the problem priority on the primary control objective.

The main noticeable difference between the WLS CA problem formulated in Equation 3.8 and the Nonlinear CA problem formulated in Equation 3.9 is the replacement of the linearized control effectiveness matrix with the nonlinear vehicle dynamics expression.

Concerning the two diagonal matrices W_u and W_ν, as in the WLS CA, they penalize respectively the associated actuator command and control objective. Within the chapter, we will refer to this CA strategy as the Nonlinear CA algorithm.

3.3.1. RESOLUTION METHOD OF THE NONLINEAR CA PROBLEM

The problem presented in Equation 3.9 is a constrained nonlinear optimization problem whose solution can be determined using the nonlinear programming approach. Among the different nonlinear programming algorithms proposed in literature [19], SQP has been shown to be the most efficient and accurate algorithm in almost all circumstances [20]. We therefore decided to employ this algorithm to solve the nonlinear problem presented in Equation 3.9.

Among all the different SQP implementations available, we decided to employ the line search based SQP algorithm implemented in the MATLAB *fmincon* function. A detailed derivation of the SQP method for the resolution of the Nonlinear CA problem is beyond the scope of this chapter. For the interested reader, [19] explains in detail how the optimization process works and how it can be implemented.

In brief, the strategy of the SQP optimization process is to reduce the nonlinear problem complexity by the formulation of a series of unconstrained sequential quadratic programming sub problems. Each sub problem solution determines the searching direction while the step size is determined by evaluating a merit function containing both the cost function and the problem constraints. Specifically, two main characteristics of the chosen SQP optimization strategy are particularly beneficial for the implementation of our CA problem:

- Strict feasibility of bounds: at every iteration, the implemented SQP algorithm never presents a solution that is out of bounds.
- Robustness to NaN or Inf results: In case a step is taken which results in an invalid evaluation of the cost function, the algorithm tries to take a smaller step with the aim of computing a valid solution.

3.4. IMPLEMENTATION OF THE CA METHODS ON THE DUAL-AXIS TILTING ROTOR QUAD-PLANE

In this section we will derive the INDI control framework for the dual-axis tilting rotor quad-plane in Figure 3.1. The vehicle was presented in our previous work [1] and has a total of 12 actuators including 8 servo-rotors. The number of actuators gives the vehicle a full 6 DOF and makes it overactuated. Furthermore, the presence of tiltable rotors gives the aircraft highly non-linear actuator dynamics. All these characteristics give the aircraft a high potential maneuverability but at the same time make the CA problem particularly complex to solve. We will use the vehicle platform as a test bed to compare the different previously presented linear and non-linear CA methods (PIU, WLS and Nonlinear).

The first step for the implementation of the presented CA methods on the flying vehicle is the derivation of the vehicle EOM. Once derived, the EOM can be replaced by the generic nonlinear system considered in Equation 3.2.

3.4.1. EQUATION OF MOTION DERIVATION

For the sake of brevity, we defer to Chapter 2.3 for the complete derivation of the equations of motion, the definition of the reference systems and the assumptions made. The equations of motion for the dual axis tilt rotor quad-plane are as follows:

$$\begin{cases} \ddot{P_e} = \dfrac{1}{m}\left(F^p + F^a\right) + g\hat{z}_e \\ \dot{\omega} = I_b^{-1}\big(-\omega \times I_b\omega + M^t + M^d + M^i + \\ \qquad +M^p + M^a + M^r + M^{tilt}\big), \end{cases} \tag{3.10}$$

where $\ddot{P_e}$ are the linear acceleration in the earth reference frame and $\dot{\omega}$ represent the body rates derivative. The forces vectors F^a and F^p

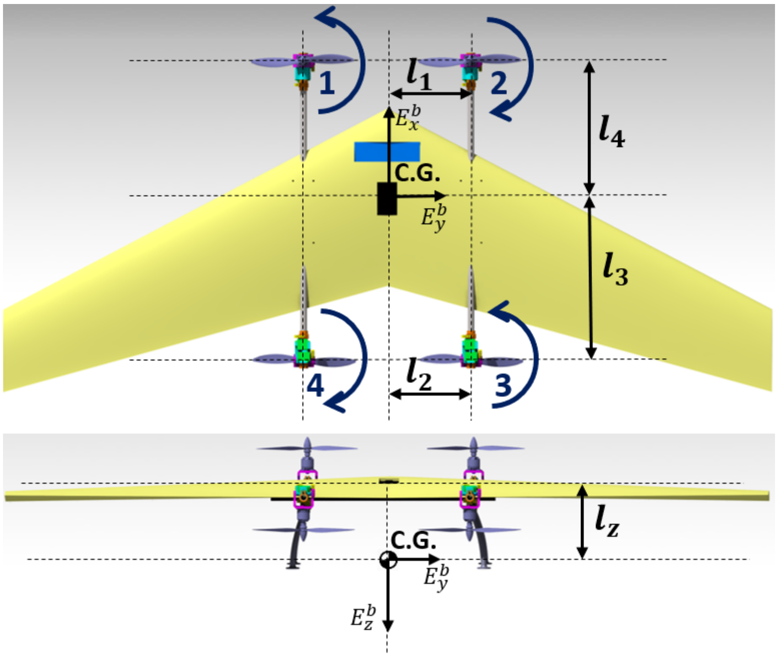


Figure 3.3: Motor spinning direction and vehicle geometry with respect to the Center of Gravity (CG)

represent respectively the aerodynamics forces and the thrust produced by the propellers, expressed in the earth reference frame. As for the moments equilibrium, the moment term M^t represents the torque generated by the rotors due to the propeller thrust, M^d represents the torque generated by the rotors due to the propeller drag, M^i represents the torque induced by the i-th propeller inertia due to the motor rotational rate change $\dot{\Omega}_i$. The term M^p represents the torque generated by the rotor precession term due to the tilting rotation, M^r models the inertial term of the rotors due to the vehicle rates and M^{tilt} represents the torque generated by the rotor inertia due to the tilting rotation. A more detailed explanation of each element of Equation 3.10 is covered in Chapter 2.3.

3.4.2. INDI FRAMEWORK FOR THE DUAL-AXIS TILTING ROTOR QUAD-PLANE

For a vehicle such as the dual-axis tilting rotor quad-plane, the control module scheme is slightly different from the one of a conventional under-actuated vehicle. In the hovering configuration, the dual-axis tilting rotor quad-plane has full 6 DOF control. Therefore, in this flight

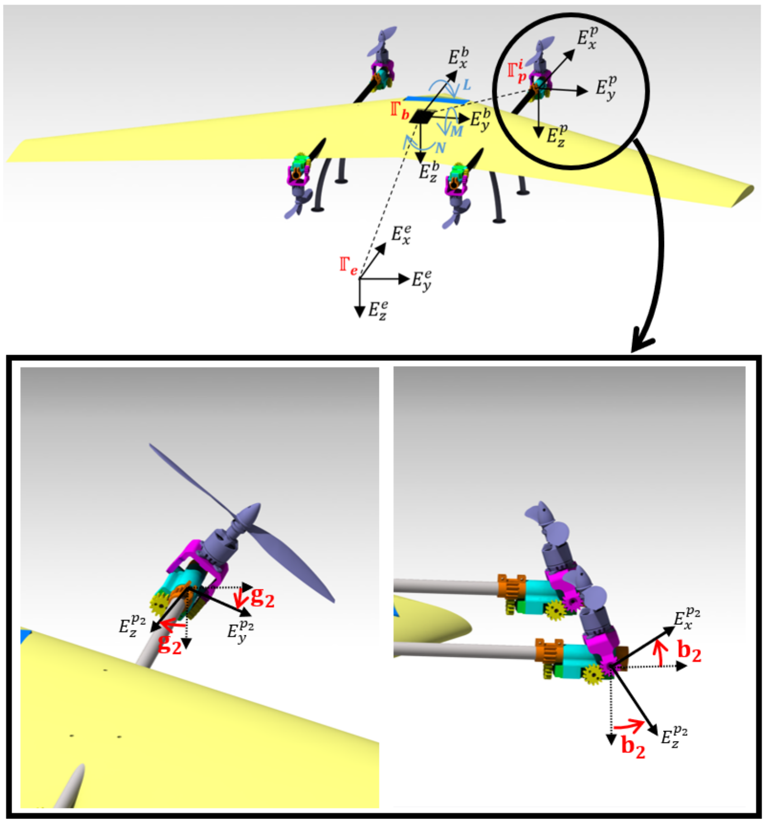


Figure 3.4: Overview of the Earth, Body and Propeller reference frames and rotor tilting angles.

configuration, an outer loop is not required and a single control loop can be employed to directly control attitude and position. The resulting INDI control scheme is presented in figure 3.5, where a linear error controller is employed for the generation of the desired linear and angular acceleration. A representation of the error controller scheme is depicted in figure 3.6.

As for the pseudo-control vector, it is composed of the following desired linear and angular accelerations, expressed in the earth reference frame Γ_e:

$$\nu = \begin{pmatrix} \ddot{x}_d & \ddot{y}_d & \ddot{z}_d & \dot{p}_d & \dot{q}_d & \dot{r}_d \end{pmatrix}^T . \tag{3.11}$$

For the control input vector, it is constituted of a total of 12 elements containing both the motor rotational speed and the rotor tilting angles:

$$u = \begin{pmatrix} \Omega_1 & \Omega_2 & \Omega_3 & \Omega_4 & b_1 & b_2 & b_3 & b_4 & g_1 & g_2 & g_3 & g_4 \end{pmatrix}^T . \tag{3.12}$$

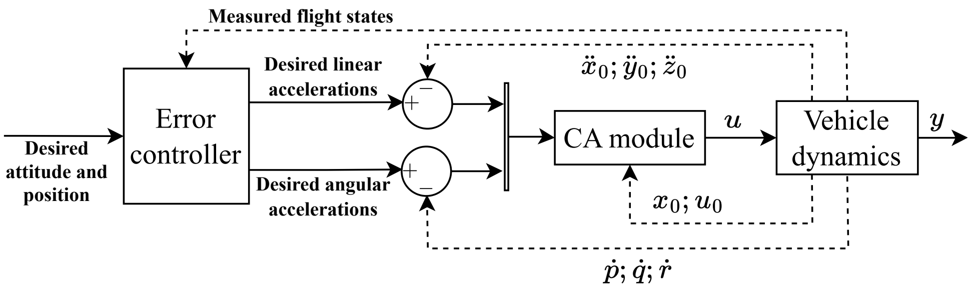

Measured flight states
Desired attitude and position
Error controller
Desired linear accelerations
Desired angular accelerations
$\ddot{x}_0; \ddot{y}_0; \ddot{z}_0$
CA module
u
Vehicle dynamics
y
$x_0; u_0$
$\dot{p}; \dot{q}; \dot{r}$

Figure 3.5: INDI control diagram for the dual-axis tilting rotor quad-plane in hovering configuration. In the diagram, the dashed lines represent variables estimated through sensor (for the vehicle state) or actuator model (for the current actuator state).

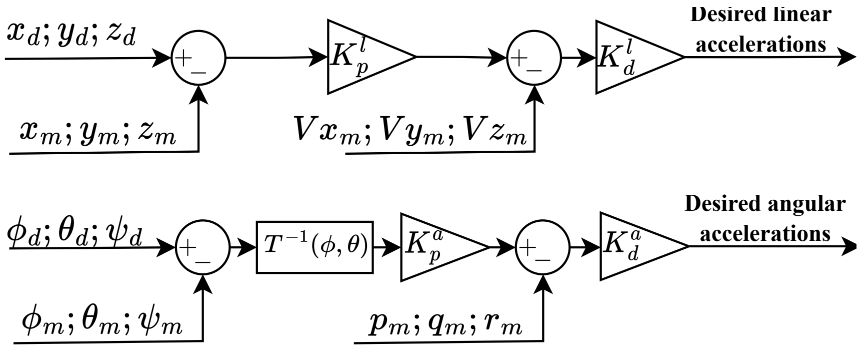


Figure 3.6: Error Controller for the 6 DOF INDI control scheme. The subscript *m* indicates measured values while the subscript *d* represents the desired variables.

DERIVATION OF THE EFFECTIVENESS MATRIX FOR THE LINEARIZED CA METHODS

For the PIU and the WLS control allocation methods, an estimate of the linear actuator effectiveness of the vehicle is required. As described in the previous section, the estimation of the linear control effectiveness matrix will be evaluated in the current state x_0 and current control input u_0 and then inverted to determine the control input set.

The effectiveness matrix is composed of the partial derivative of the EOM in Equation 3.10 with respect to the control input array defined in

Equation 3.12, for a total matrix size of 6x12:

$$B = \frac{\partial(\ddot{P}_e\,;\dot{\omega})}{\partial u} =$$
$$\begin{pmatrix}
\frac{\partial\ddot{x}}{\partial\Omega_1} & \frac{\partial\ddot{x}}{\partial\Omega_2} & \frac{\partial\ddot{x}}{\partial\Omega_3} & \frac{\partial\ddot{x}}{\partial\Omega_4} & \frac{\partial\ddot{x}}{\partial b_1} & \frac{\partial\ddot{x}}{\partial b_2} & \frac{\partial\ddot{x}}{\partial b_3} & \frac{\partial\ddot{x}}{\partial b_4} & \frac{\partial\ddot{x}}{\partial g_1} & \frac{\partial\ddot{x}}{\partial g_2} & \frac{\partial\ddot{x}}{\partial g_3} & \frac{\partial\ddot{x}}{\partial g_4} \\
\frac{\partial\ddot{y}}{\partial\Omega_1} & \frac{\partial\ddot{y}}{\partial\Omega_2} & \frac{\partial\ddot{y}}{\partial\Omega_3} & \frac{\partial\ddot{y}}{\partial\Omega_4} & \frac{\partial\ddot{y}}{\partial b_1} & \frac{\partial\ddot{y}}{\partial b_2} & \frac{\partial\ddot{y}}{\partial b_3} & \frac{\partial\ddot{y}}{\partial b_4} & \frac{\partial\ddot{y}}{\partial g_1} & \frac{\partial\ddot{y}}{\partial g_2} & \frac{\partial\ddot{y}}{\partial g_3} & \frac{\partial\ddot{y}}{\partial g_4} \\
\frac{\partial\ddot{z}}{\partial\Omega_1} & \frac{\partial\ddot{z}}{\partial\Omega_2} & \frac{\partial\ddot{z}}{\partial\Omega_3} & \frac{\partial\ddot{z}}{\partial\Omega_4} & \frac{\partial\ddot{z}}{\partial b_1} & \frac{\partial\ddot{z}}{\partial b_2} & \frac{\partial\ddot{z}}{\partial b_3} & \frac{\partial\ddot{z}}{\partial b_4} & \frac{\partial\ddot{z}}{\partial g_1} & \frac{\partial\ddot{z}}{\partial g_2} & \frac{\partial\ddot{z}}{\partial g_3} & \frac{\partial\ddot{z}}{\partial g_4} \\
\frac{\partial\dot{p}}{\partial\Omega_1} & \frac{\partial\dot{p}}{\partial\Omega_2} & \frac{\partial\dot{p}}{\partial\Omega_3} & \frac{\partial\dot{p}}{\partial\Omega_4} & \frac{\partial\dot{p}}{\partial b_1} & \frac{\partial\dot{p}}{\partial b_2} & \frac{\partial\dot{p}}{\partial b_3} & \frac{\partial\dot{p}}{\partial b_4} & \frac{\partial\dot{p}}{\partial g_1} & \frac{\partial\dot{p}}{\partial g_2} & \frac{\partial\dot{p}}{\partial g_3} & \frac{\partial\dot{p}}{\partial g_4} \\
\frac{\partial\dot{q}}{\partial\Omega_1} & \frac{\partial\dot{q}}{\partial\Omega_2} & \frac{\partial\dot{q}}{\partial\Omega_3} & \frac{\partial\dot{q}}{\partial\Omega_4} & \frac{\partial\dot{q}}{\partial b_1} & \frac{\partial\dot{q}}{\partial b_2} & \frac{\partial\dot{q}}{\partial b_3} & \frac{\partial\dot{q}}{\partial b_4} & \frac{\partial\dot{q}}{\partial g_1} & \frac{\partial\dot{q}}{\partial g_2} & \frac{\partial\dot{q}}{\partial g_3} & \frac{\partial\dot{q}}{\partial g_4} \\
\frac{\partial\dot{r}}{\partial\Omega_1} & \frac{\partial\dot{r}}{\partial\Omega_2} & \frac{\partial\dot{r}}{\partial\Omega_3} & \frac{\partial\dot{r}}{\partial\Omega_4} & \frac{\partial\dot{r}}{\partial b_1} & \frac{\partial\dot{r}}{\partial b_2} & \frac{\partial\dot{r}}{\partial b_3} & \frac{\partial\dot{r}}{\partial b_4} & \frac{\partial\dot{r}}{\partial g_1} & \frac{\partial\dot{r}}{\partial g_2} & \frac{\partial\dot{r}}{\partial g_3} & \frac{\partial\dot{r}}{\partial g_4}
\end{pmatrix} \tag{3.13}$$

For the determination of the solution, both the WLS and the PIU have to invert the constant effectiveness matrix in Equation 3.13. Unfortunately, for some current actuator values and states, the effectiveness matrix may contain small elements. This means that the associated actuator solution will be far away from the current actuator state, invalidating the linearization. The PIU and WLS CA algorithms have no information about the system's nonlinearities and therefore have to wait for the actuator movement to realize that the computed solution does not give the desired result.

This behavior is clearly visible by running the WLS CA algorithm with a desired vertical acceleration and all motors slightly tilted outward, as depicted in Figure 3.7:

$$\begin{aligned}
\nu &= \begin{pmatrix} 0 & 0 & -10 & 0 & 0 & 0 \end{pmatrix}^T \\
u_0 &= (\,700 \;\; 700 \;\; 700 \;\; 700 \\
&\qquad 0 \quad 0 \quad 0 \quad 0 \\
&\qquad -.1 \quad .1 \quad .1 - .1\,)^T.
\end{aligned} \tag{3.14}$$

Due to the linearization, the effectiveness matrix calculated with this actuator state contains some non-zero elements for the terms $\frac{\partial\ddot{z}}{\partial g_i}$. This means that the CA algorithm may use the lateral tilting to control the vertical acceleration without increasing the RPM of the motors, which would lead to an increase in the cost according to Equation 3.8. Unfortunately, the effectiveness matrix only contains a coefficient and does not have any information of the nonlinearities of the platform actuators. This means that the WLS CA algorithm will eventually compute a big lateral tilting angular change as an alternative to increasing the motor power. The actuator solution computed by the WLS CA algorithm, for the problem conditions in Equation 3.14 is

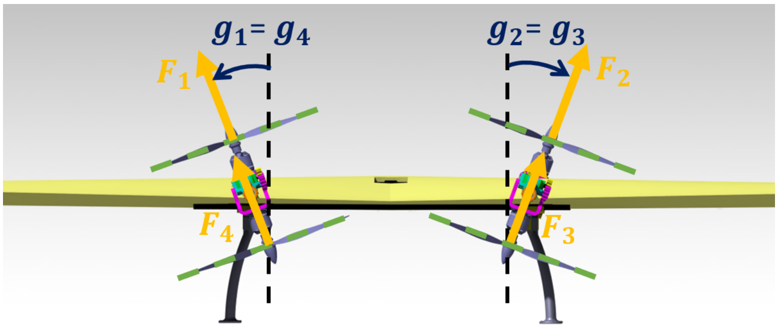


Figure 3.7: Initial actuator state defined in equation 3.14.

reported in Equation 3.15 and represented in Figure 3.8.

$$u_s = \begin{pmatrix} 950 & 950 & 950 & 950 \\ 0 & 0 & 0 & 0 \\ .78 & -.78 & -.78 & .78 \end{pmatrix}^T. \tag{3.15}$$

By comparing the computed solution with the saturation points of the

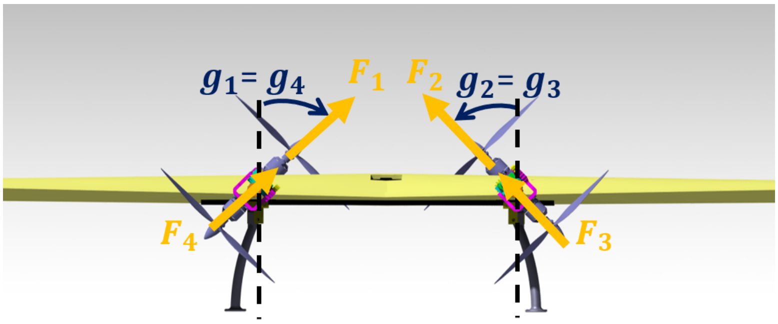


Figure 3.8: Actuator solution of the problem defined in Equation 3.14 determined using the WLS CA algorithm.

actuators in Table 3.2, it is visible that the proposed WLS actuator solution saturates both the motor rotational speed and the lateral tilting angle in an attempt to generate a vertical acceleration. This condition is clearly non-optimal, as a zero tilting angle would yield more vertical acceleration, and the controller will detect the ineffectiveness of the solution only once the tilting actuator will move. This will eventually trigger permanent oscillations in the angular solution computed by the WLS CA algorithm.

For the same scenario, the PIU CA algorithm computes an unfeasible actuator solution, with a motor rotational speed 1.2 times the saturation point and with the lateral tilt angle on a non-neutral condition:

$$\begin{aligned} u_s = (\, & 1161 \;\; 1161 \;\; 1161 \;\; 1161 \\ & 0 \quad 0 \quad 0 \quad 0 \\ & .11 \; -.11 \; -.11 \;\; .11 \,)^T. \end{aligned} \tag{3.16}$$

We can compare the results computed by WLS and PIU with the one computed by the Nonlinear CA algorithm:

$$\begin{aligned} u_s = (\, & 950 \;\; 950 \;\; 950 \;\; 950 \\ & 0 \quad 0 \quad 0 \quad 0 \\ & 0 \quad 0 \quad 0 \quad 0 \,)^T. \end{aligned} \tag{3.17}$$

In this case, the CA algorithm only saturates the motors, with both azimuth and elevation tilting angles equal to zero.

SIMPLIFICATION OF THE EOM FOR THE NONLINEAR CA METHOD FORMULATION

To decrease the computational load of the Nonlinear CA method, only the main terms of the EOM of Equation 3.10 are considered. The vehicle dynamics considered for the nonlinear optimization process in Equation 3.9 is the following:

$$\begin{cases} \ddot{P}_e = \dfrac{1}{m}\left(F^p + F^a\right) + g\hat{z}_e \\ \dot{\omega} = I_b^{-1}\left(-\omega \times I_b\omega + M^t + M^d + M^a\right) \end{cases} \tag{3.18}$$

The secondary terms of the EOM not considered for the inversion will be treated as unmodeled dynamics and handled by the error controller.

Note that this formulation of the EOM also considers the aerodynamic effects not considered in the linearized actuator effectiveness matrix formulated in Equation 3.13. It is now possible to insert the simplified EOM expression of Equation 3.18 into the cost function derived in Equation 3.9 to complete the Nonlinear CA problem formulation.

CA PARAMETERS CHOICE

For the implementation of the previously introduced CA methods, a few parameters have to be set and described. The parameter in common between all the CA algorithms is the choice of the desired actuator state.

For the tilting system, during the hovering test, the desired state of the tilting angles is set to the vertical condition ($b_i = g_i = 0$). Regarding the motor desired state, it was set to 100 *rad/s*, equivalent to the minimum applicable value.

For the WLS and Nonlinear CA algorithms, the weighting matrices W_u, W_ν and the penalty parameters γ_ν and γ_u have to be chosen. Starting with the control input weighting matrix W_u, it is a square diagonal matrix of the same size as the number of actuators (12x12 in that case). Each diagonal element of the matrix assigns a penalty on the usage of the respective actuator and it is located on the secondary objective of the cost function. Ideally, we prefer to prioritize the usage of the tilting system over the motor for a more energy efficient solution. Therefore, we decided to apply a unitary penalty coefficient to the tilting system and a penalty coefficient equal to 3 to the motors. The weighting matrix W_u used for the vehicle then becomes:

$$W_u = \text{diag}\ (3,\ 3,\ 3,\ 3,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1). \tag{3.19}$$

As for the pseudo-control weighting matrix W_ν, as already explained in [18], it should be set in such a way for the problem to firstly prioritize the pitch and roll changes, then the linear acceleration changes and the yaw angle. This leads to a pseudo-control weighting matrix W_ν of this form:

$$W_\nu = \text{diag}\ (0.01,\ 0.01,\ 0.02,\ 0.2,\ 0.2,\ 0.01). \tag{3.20}$$

Concerning the penalty parameter γ, we set the value of 1e5 to γ_ν and 1e-5 to γ_u while for both the Nonlinear and WLS problem formulations, a maximum number of 60 iterations was set and the current actuator state u_0 was used as starting point for the optimization process.

The last parameter that needs to be discussed for the constrained CA problems is the constraint definition. Ideally, the problem constraints are constant at every iteration and equal to the physical actuator limits. However, one may claim that with a limit on the maximum change in control input, the error that would be made by linearizing the system would become smaller. To investigate this, when using the WLS CA algorithm for the resolution of the CA problem, a different choice of the problem constraints will also be tested. The modified constraint, denoted as "local tilting constraint" later in the chapter, is of the following form:

$$\begin{pmatrix} \Omega^{min} \\ \Omega^{min} \\ \Omega^{min} \\ \Omega^{min} \\ max(b_1^{min}, b_1^0 - \delta_c) \\ max(b_2^{min}, b_2^0 - \delta_c) \\ max(b_3^{min}, b_3^0 - \delta_c) \\ max(b_4^{min}, b_4^0 - \delta_c) \\ max(g_1^{min}, g_1^0 - \delta_c) \\ max(g_2^{min}, g_2^0 - \delta_c) \\ max(g_3^{min}, g_3^0 - \delta_c) \\ max(g_4^{min}, g_4^0 - \delta_c) \end{pmatrix} \leq u \leq \begin{pmatrix} \Omega^{max} \\ \Omega^{max} \\ \Omega^{max} \\ \Omega^{max} \\ min(b_1^{max}, b_1^0 + \delta_c) \\ min(b_2^{max}, b_2^0 + \delta_c) \\ min(b_3^{max}, b_3^0 + \delta_c) \\ min(b_4^{max}, b_4^0 + \delta_c) \\ min(g_1^{max}, g_1^0 + \delta_c) \\ min(g_2^{max}, g_2^0 + \delta_c) \\ min(g_3^{max}, g_3^0 + \delta_c) \\ min(g_4^{max}, g_4^0 + \delta_c) \end{pmatrix} \tag{3.21}$$

Where the superscripts "0" indicates the current value, the superscripts *max* and *min* indicate respectively the maximum and minimum physical value of the actuator and δ_c represents the constant angular region of constraint. The maximum and minimum actuator values for the dual-axis tilting rotor quad-plane are reported in Table 3.2.

ACTUATOR MODEL IDENTIFICATION

The previously proposed CA methods all rely on an incremental control law and therefore require an estimate of the current actuator state to work. Accurate modeling of the actuator dynamics is also needed for the vehicle simulation.

For the motor propeller actuation system, we determined its dynamics through bench tests using an external RPM sensor. The result of a step test campaign on the motor propeller system is shown in Figure 3.9A. In the same plot, the motor rotational speed evolution was also fitted with a first and a second order discrete transfer function. As visible, the motor-propeller system can be modeled using a first order discrete transfer function. The analytical expression of the first order discrete transfer function associated with the motor dynamics is the following:

$$H_{motor}(z) = \frac{0.05824z^{-1}}{1 - 0.9418z^{-1}}. \tag{3.22}$$

The sampling frequency used to determine the transfer function in Equation 3.22 is 500 Hz. This is the update frequency of the main flight computer running onboard the vehicle.

Concerning the identification of the tilting dynamics, the approach of [1] was used. An IMU was mounted solidly to the tilting structure and

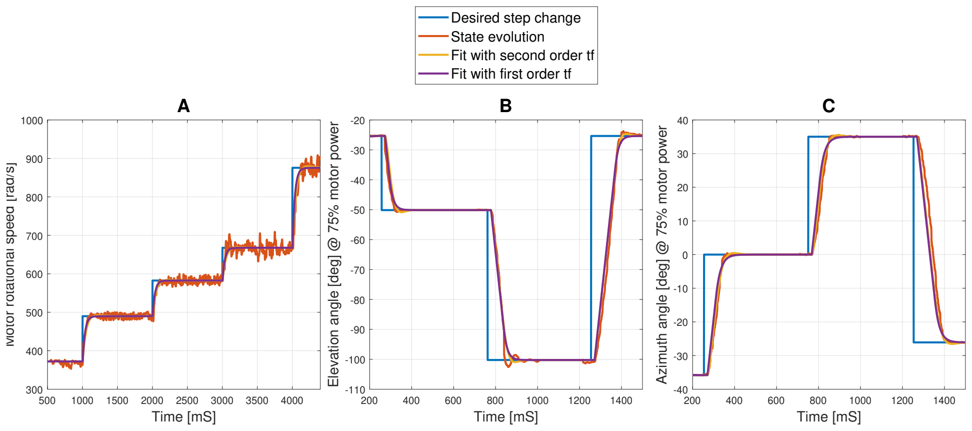


Figure 3.9: Step test actuator response for the identification of the dynamics. The left plot (A) is the motor step test, the central plot (B) is the azimuth tilting angle step test and the right plot (C) is the elevation tilting angle step test. The blue curve represents the desired step change and the red curve reports the variable evolution. The purple and yellow curves are a fit of the variable evolution using respectively a first and a second order discrete transfer function.

angle step changes were induced on the tilting servos. The dynamics is then recorded and analyzed to determine the tilting rotor dynamics at different motor speeds.

The motor spinning power chosen to identify the motor dynamics discrete transfer function was 75%. The tilting evolution for the azimuth and elevation rotor angle step test is displayed in Figure 3.9B and Figure 3.9C. The tilting angle evolution was fitted with a first and second order discrete transfer function featuring also a rate saturation limit. In this case, the second order transfer function gives a slightly better representation of the tilting dynamics. The analytical expressions of second order discrete transfer function used for the azimuth and elevation tilting angle dynamics are the following:

$$\begin{gathered} H_{az}(z) = (z^{-6})\frac{0.00386z^{-1} + 0.003679z^{-2}}{1 - 1.858z^{-1} + 0.8659z^{-2}}. \\ R_l^{az} = 9.95\,\frac{rad}{s} \end{gathered} \tag{3.23}$$

$$
\begin{aligned}
H_{el}(z) &= (z^{-6})\frac{0.006779z^{-1} + 0.006384z^{-2}}{1 - 1.822z^{-1} + 0.8353z^{-2}} \\
R_l^{el} &= 11.34\,\frac{rad}{s}
\end{aligned}
\quad . \tag{3.24}
$$

Where R_l represents the rate saturation limit value of the tilting system. The same sampling frequency of 500 Hz was used for the determination of the discrete transfer functions in Equation 3.23 and 3.24. For completeness, in Table 3.1, the characteristics of the equivalent continuous time system associated to the actuators are displayed.

Actuator	**Corner frequency ω_c**	**Damping ratio ζ**	**Rate limit**
Motor	30 rad/s	-	-
Tilt elevation	60 rad/s	1.5	11.34 rad/s
Tilt azimuth	45 rad/s	1.6	9.95 rad/s

Table 3.1: Characteristics of the equivalent continuous time system associated to the actuators.

ACTUATOR CONSTRAINT AND PROBLEM NORMALIZATION

For a complete CA problem formulation, two more steps are required. The first step is the determination of the physical limits of the actuators on the vehicle. The second step is to use the total travel of the actuators to normalize the CA problem, in such a way to equally compare the actuator effects during the CA resolution process. The physical limits of the actuators are displayed in Table 3.2.

With the maximum and minimum actuator limits determined, it is possible to scale the control input vector presented in Equation 3.12 for the normalization of the CA problem:

$$
u_n = \left(\frac{\Omega_1}{G_m}\ \frac{\Omega_2}{G_m}\ \frac{\Omega_3}{G_m}\ \frac{\Omega_4}{G_m} \right. \\
\frac{b_1}{G_{el}}\ \frac{b_2}{G_{el}}\ \frac{b_3}{G_{el}}\ \frac{b_4}{G_{el}} \\
\left. \frac{g_1}{G_{az}}\ \frac{g_2}{G_{az}}\ \frac{g_3}{G_{az}}\ \frac{g_4}{G_{az}} \right). \tag{3.25}
$$

Where the scaling factors G_m, G_{el} and G_{az} are displayed in Table 3.3. Using the normalized control input array u_n presented in Equation 3.25, the overall actuator travel will be the same for all the actuators and will be equal to 2. For a complete normalization of the CA problem, the maximum, minimum and desired actuator value, the effectiveness matrix B and the EOM were normalized as well.

	Max value	**Min value**
Motor rotational speed Ω_i	950 rad/s	100 rad/s
Azimuth tilting angle g_i	45 deg	-45 deg
Elevation tilting angle b_i	25 deg	-90 deg

Table 3.2: Actuators physical limit for the CA problem

Motor scaling factor G_m	425 rad/s
Azimuth tilting scaling factor G_{az}	0.785 rad
Elevation tilting scaling factor G_{el}	2.705 rad

Table 3.3: Scaling factor for the normalization of the control input vector

3.4.3. EXPERIMENTAL SETUP

The implementation of the previously presented CA methods on a real-time setup requires a reasonable amount of computational power that a microcontroller alone cannot provide. In order to solve the computational deficit problem, we decided to distribute the computational load among two boards. The main FCB used is a Pixhawk 4 flight controller running the open-source paparazzi software[1] [21]. The task of the Pixhawk 4 is to interact with the vehicle sensors and instruments, control the actuators and to generate the pseudo-control vector. Once the pseudo-control vector is generated, it is communicated over a UART interface to the Raspberry Pi 4B[2], whose task it is to solve the CA problem and communicate the actuator solution back to the Pixhawk 4.

The development of the CA modules operating on the SBC was characterized by a few steps. Initially, the PIU, WLS and Nonlinear CA algorithms were implemented and tested in Simulink. Secondly, using the Simulink Coder support package[3], the C functions implementing the CA methods were generated. Finally, the generated C code of the CA routines were manually optimized, compiled and then deployed on the Raspberry Pi 4B SBC.

For the communication between the Raspberry Pi 4 and the Pixhawk, a serial communication was employed. In order to maximize the computational speed of the CA solver, the serial communication on the

[1]https://wiki.paparazziuav.org/
[2]https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf
[3]https://www.mathworks.com/products/simulink-coder.html

raspberry Pi was implemented on a different thread. As for the rest of the components employed for the vehicle hardware such as the type of tilting servos and motors used, the reader can refer to [1].

3.4.4. SIMULATION RESULTS

With the knowledge of the vehicle dynamics, the actuator characteristics, and a proper identification of the control framework, it was possible to set up a simulation environment in Simulink. The simulation outcome could then be compared to the flight test result for the model validation.

A mixed maneuver engaging attitude, altitude and position changes was chosen. Initially, a pitch and roll angle of 20 degrees are commanded. Then, the vehicle is asked to track a desired altitude and position step change.

As a reference for the model validation, only the Nonlinear CA algorithm was used to compare the simulation outcome with the flight test results. In Figure 3.10 the attitude and position evolution between flight test and simulation is compared. As for the actuator evolution, for the sake of simplicity, only the tilting angles and motor power of the rotor number 2 are compared in Figure 3.11.

COMMENTS

The flight test results and the simulation outcome are matching, thus validating the modeling of the vehicle and actuator dynamics. The only visible mismatch happens around time 20 seconds, during the climbing phase of the maneuver. As also confirmed by the logs, this mismatch may have been caused by a sudden voltage drop related to a spike in power demand of the motors. This condition also influences the motor rotational speed response of the flight test which presents a damped oscillation not present in the simulation.

3.4.5. FLIGHT TEST RESULTS

To assess the behavior of the different CA methods on the flying vehicle, the same maneuver proposed for the simulation was programmed on board the drone. The flight test was carried out in the TUDelft Cyberzoo flight arena and the flying vehicle was always secured with a rope from the arena ceiling. The rope was manually handled in such a way to minimize the interference with the flight.

For an accurate estimation of the vehicle position and orientation, an Inertial Navigation System (INS) composed of an OptiTrack optical motion system, a 3-axis accelerometer, a 3-axis magnetometer and a 3-axis gyroscope was employed. Since the controller is designed to track accelerations in the earth reference frame Γ_e, an estimation of

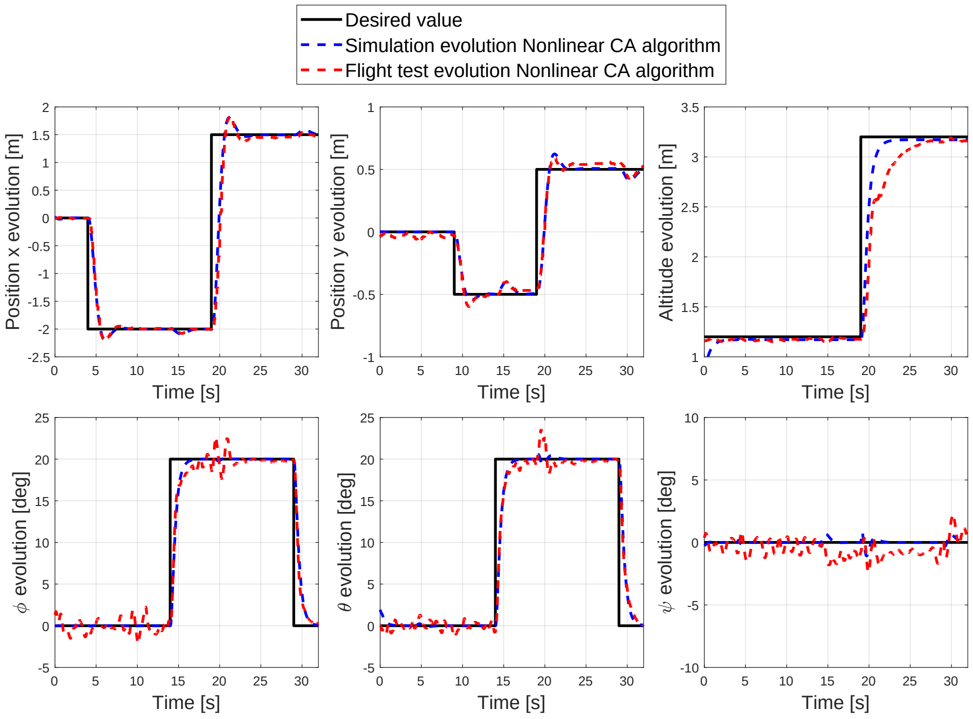


Figure 3.10: Position and attitude evolution comparison between simulation and flight test using the Nonlinear CA algorithm.

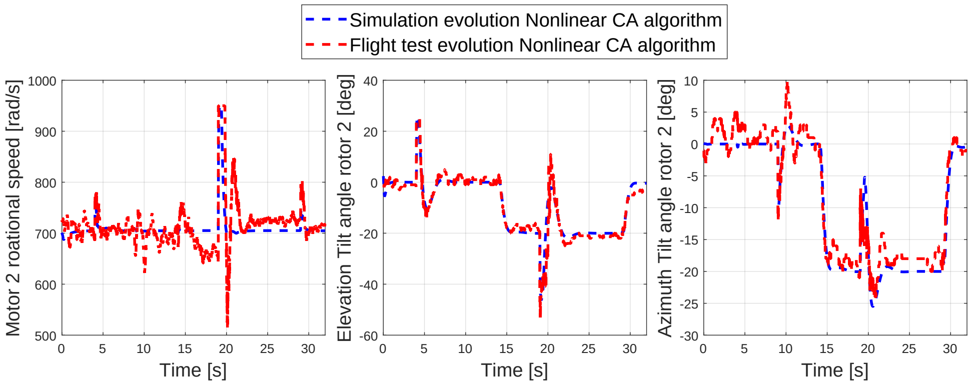


Figure 3.11: Tilting angles and motor rotational speed evolution of rotor number 2 computed by the Nonlinear CA algorithm. The dashed blue curve is the simulated actuator evolution while the dashed red curve is the actuator evolution experienced during the flight test.

the inertial accelerations of the vehicle in the earth reference frame is needed. This estimation is obtained by projecting the inertial body accelerations recorded by the accelerometer to the earth reference frame through Equation 2.3.1.

An overview of the different CA algorithms tested on the flying vehicle are summarized in Table 3.4 while in Figure 3.13, the attitude and position tracking for the successful and partially successful flights are displayed. The partially successful flight curves are truncated at the time the flight test was stopped. In Figure 3.14, the actuators evolution of the rotor number 2 are shown. As done for the analysis of the simulation results, the rotor number 2 was chosen as reference for the analysis of the solution computed by the different CA algorithms during the flight tests. Finally, in Figure 3.12, the plot containing the run-time of the three main CA algorithms is displayed. The run-time was defined as the time needed by the CA algorithm running on the Raspberry pi to compute the actuator solution. This value defines the actual update period of the desired actuator value and it is generally not constant.

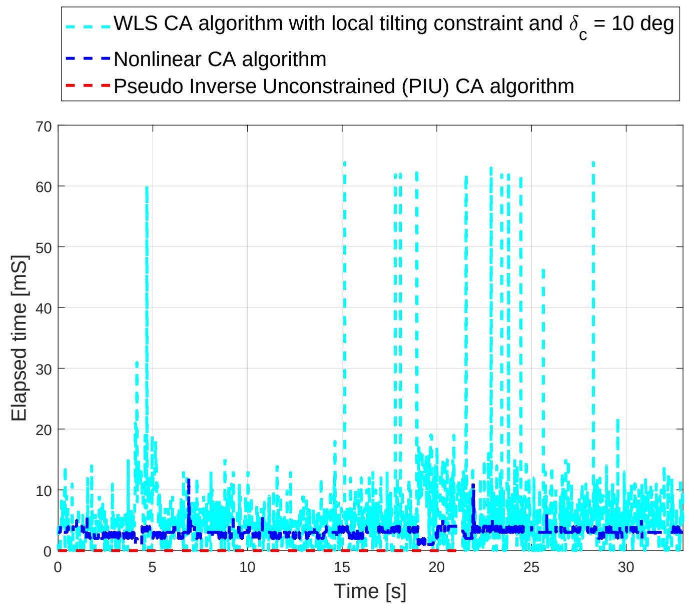


Figure 3.12: Run-time of the three main CA algorithms during the flight test maneuver.

COMMENTS

Several interesting observations can be made by looking at the flight

Algorithm tested	**Flight test outcome**	**Comments**	**Average run-time**
Nonlinear CA	Successful	Smooth actuator response, best position and attitude tracking.	2959 uS
WLS CA	Failed	Permanent and uncontrolled oscillations in the actuator solution. Not able to track neither the desired position nor the attitude.	-
WLS CA with identity control input weighting matrix W_u	Partially Successful	Motor saturation due to yaw tracking. Oscillations in the actuator solution once the rapid position change is commanded.	1711 uS
WLS CA with local tilting constraint and $\delta_c = 10$ deg	Successful	Permanent bounded oscillations in the tilting actuator solution. Position and attitude tracking comparable to the Nonlinear CA.	4690 uS
WLS CA with local tilting constraint and $\delta_c = 5$ deg	Partially successful	Permanent bounded oscillations in the tilting actuator solution. Overshoot in the position tracking.	5066 uS
PIU CA	Partially successful	Motor saturation due to yaw tracking. Oscillations in the actuator solution once the rapid position change is commanded.	11 uS
Video of the experiment			

Table 3.4: Overview of the different CA algorithms tested on the flying vehicle. Video link: https://youtu.be/OwoZQ9u_5EE

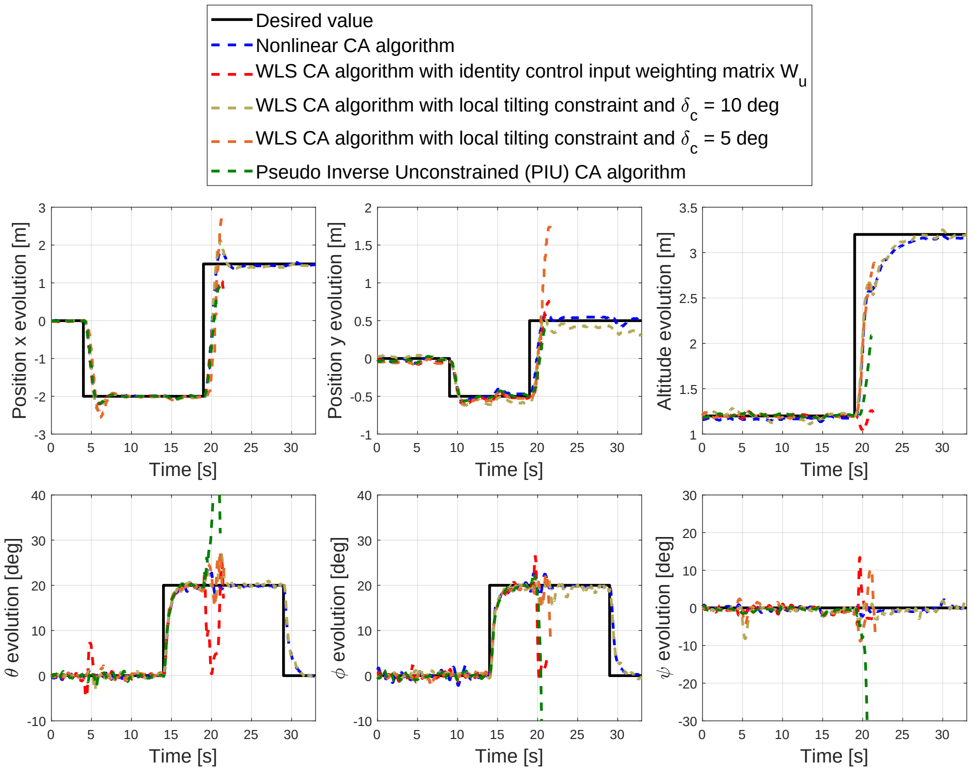


Figure 3.13: Position and attitude tracking using different CA algorithms for the same flight test maneuver.

test results for the tested CA methods. Both the Nonlinear and WLS CA methods provide a valid actuator solution at every time step. These methods are constrained optimization processes and therefore always provide an actuator solution within the actuator limits. On the other hand, the PIU CA method, being an unconstrained optimization problem, does not account for actuator saturation, providing a solution that is sometimes out of the available actuator range. This behavior is clearly visible at time 19 seconds, where the step change in altitude and position is requested. During that rapid maneuver, the PIU CA algorithm computes a motor speed solution way above the saturation point. This means that the controller tries to achieve the desired acceleration set by clipping a value to an already saturated actuator state, leading to a visible loss of attitude tracking.

Another major consideration has to be done when applying equal weight in the control input matrix W_u, as done for the PIU CA algorithm and the WLS CA algorithm with unitary W_u. With those two CA methods, there is no penalty on the usage of the motor over the tilting system. This means that it is mathematically more convenient to use motor power differential instead of rotor tilt to achieve yaw changes. This condition is observed around time 15 seconds, where the vehicle corrects a yaw error by decreasing the motor power of rotor 2 and 3 while increasing the motor power of rotors 1 and 4. Unfortunately, this practice leads to the saturation of rotors 1 and 4, therefore reducing the maneuverability of the vehicle.

Being able to prioritize the usage of the tilting system over motor power is a very important feature for the determination of an energy efficient solution. However, while it was possible to fly the vehicle using the Nonlinear CA method with the control input weighting matrix W_u in Equation (3.19), it was impossible to do so with the WLS CA algorithm. Assigning penalty on the motor power highlights the limitations of the linearized approximation of the actuator effectiveness. Under this condition, the WLS CA method tends to allocate the control objective to the tilting system as much as possible, leading to rapidly oscillating tilting angles solution coupled with an inaccurate motor power solution. To better understand the reasoning behind the tilting oscillations, the reader can refer to Section 3.4.2, where this condition was analyzed in detail.

It is possible to partially bound the oscillatory actuator solution by applying a local constraint to the tilting solution in the WLS CA problem formulation, as presented in Equation (3.21). However, this condition induces a delay in the acceleration tracking proportional to the speed of the actuators. The delay in the acceleration tracking leads to an overshoot in the position and attitude tracking of the vehicle. When a δ_c = 5 deg is applied, the overshoot in the position tracking becomes so pronounced that the flight test had to be stopped prematurely to avoid a crash with the lateral net.

Concerning the proposed Nonlinear CA method, it always computes a smooth, valid and effective actuator solution, correctly prioritizing the desired control input and actuator use according to the provided weighting matrices W_ν and W_u. Among the three analyzed algorithms, the proposed Nonlinear CA is the Control Allocation strategy providing the best tracking for both position and attitude, as visible from Figure 3.13.

As for the computational load of the different algorithms, the run-time needed by the algorithms to compute the actuator solution is displayed in Figure 3.12. The WLS CA algorithm is the most computational-intensive algorithm with an average of 4.7 milliseconds needed for the determination of the actuator solution. The large run-time needed by the WLS algorithm for the determination of the solution is mainly associated with the frequent actuator saturation condition experienced during the flight. In case of actuator saturation, the iterative active-set optimization process run by the WLS algorithm needs multiple iterations to determine the solution, leading to an increase in execution time. The second most computationally intensive algorithm is the Nonlinear CA algorithm, with an average run-time of 2.9 milliseconds. Interestingly, the Nonlinear CA algorithm, as opposed to the WLS CA, experiences a reduced run-time when a set of actuators is saturated. Finally, the computational time of the PIU CA algorithm is the smallest one, requiring an average of 11 microseconds to compute, and it is more or less constant during the whole flight.

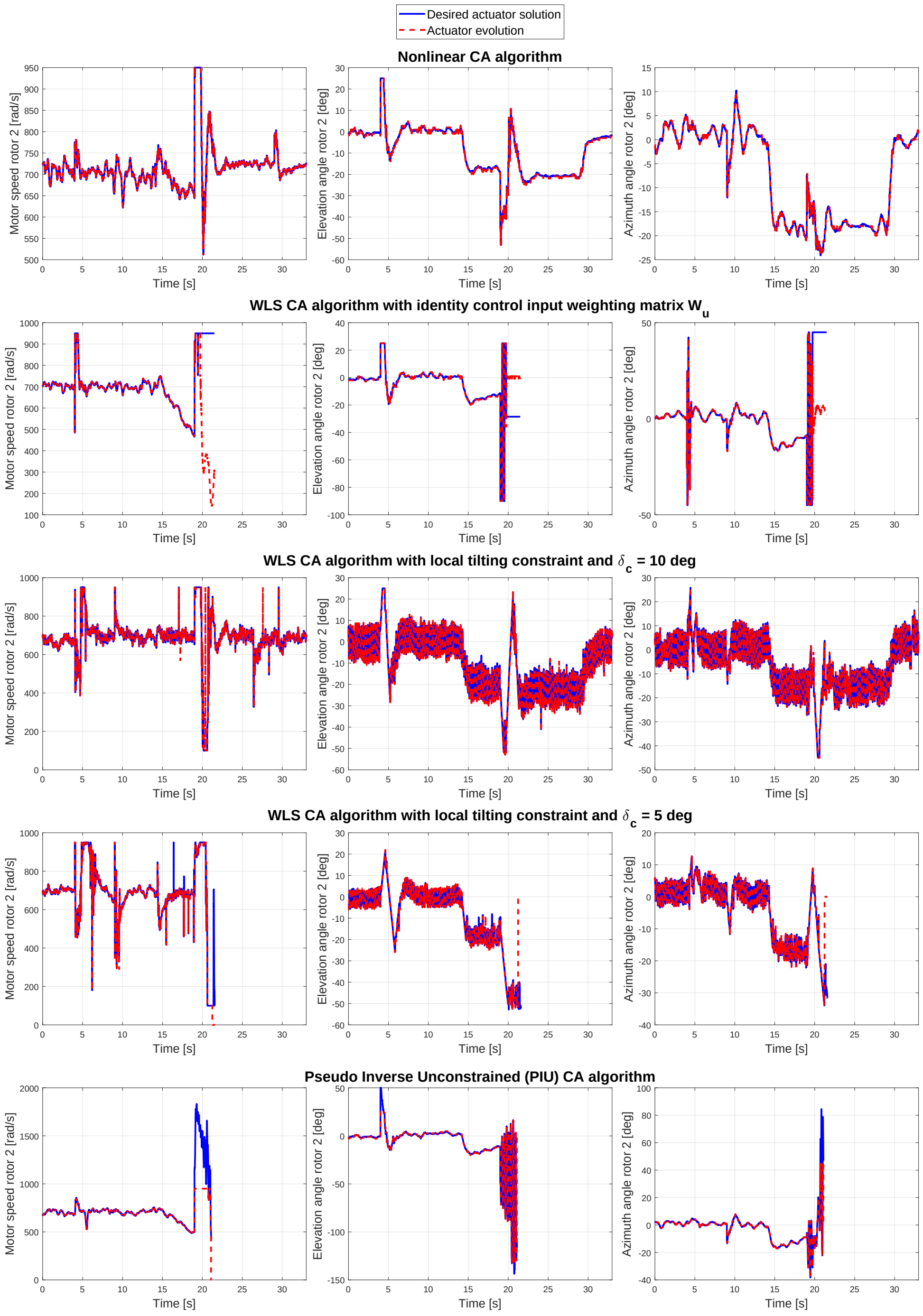


Figure 3.14: Tilting angles and motor rotational speed evolution of rotor number 2 computed by the different CA algorithms for the same flight test maneuver.

3.5. LIMITATIONS OF THE PROPOSED NONLINEAR CA METHOD

There are two main limitations concerning the applicability of the proposed Nonlinear CA on flying vehicles. The first limitation has to do with the computational power required by the Nonlinear CA algorithm to run. Depending on the number of actuators and the complexity of the EOM, the cost function formulation and its gradient might increase in complexity, leading to a longer run-time needed by the SQP algorithm for the computation of the actuator solution. A longer run-time will reduce the phase margin of the controller if it becomes excessive. Note that the nonlinear CA algorithm determines the actuator solution iteratively. Therefore, there is a direct relation between the number of iterations and the accuracy of the computed actuator solution. At the same time, the number of iterations also affects the overall run-time and computational load of the algorithm. The choice of the number of iterations is therefore a key parameter that must be chosen as a compromise between the computational load and the accuracy of the solution.

The second limitation affecting the Nonlinear optimization process is to ensure to find an actuator solution which is not in a local minimum of the cost function. The easiest way to prove the absence of local minima in the cost function, is to ensure global convexity of the function. Unfortunately, the complexity of the proposed cost function makes it difficult to analytically prove the global convexity. Even if the convexity of the cost function cannot be analytically determined, it is still possible to investigate if starting the optimization process using a different initial condition would change the cost function's final value. This gives a rough idea about the shape of the cost function and the presence of local minima.

In order to check if the cost function contains local minima, a monte-carlo simulation was performed varying the initial actuator set of the optimizer u_{init}, for different acceleration set, actuator condition and vehicle states. A total of 7000 tests were performed each time using 300 different initial actuator sets, for a total of more than 2 million optimization runs. A uniform statistical distribution was used to determine states and control objective values at every iteration. The constant and randomized variables used during the statistical analysis are displayed in Table 3.5. The choice of the variable ranges was set in such a way as to reproduce extreme conditions for the vehicle state, actuator state and for the control objective.

3.5.1. STATISTICAL ANALYSIS RESULTS

Out of the 7000 tests performed during the simulation, 98.6% of the time, the cost function value determined using $u_{init} = u_0$ is within 10%

of the minimum cost function value determined over the 300 runs with randomized u_{init}. Only 93 tests out of 7000 provided a final cost function mismatch of more than 10%, highlighting a potential local minimum condition.

In order to evaluate the primary objective term of the cost function, the residual norm of the solution determined using a different initial actuator set was compared with the solution providing the minimum residual norm for that test. The residual vector is defined as the difference between the desired accelerations and the accelerations achieved by the determined actuator solution. The norm of this vector will therefore be composed by a sum of angular and linear accelerations with units rad/s^2 and m/s^2.

In Figure 3.15, the residual norm of the "minimum norm actuator solution" is compared with the solution computed using different u_{init}. We define as "minimum norm actuator solution", the actuator solution providing minimum residual norm out of the 300 randomized u_{init} runs. The blue histogram in Figure 3.15 represents the residual norm error between the "minimum norm actuator solution" and the "maximum norm actuator solution" of the 300 optimization runs. In the same figure, the red histogram is the residual error between the "minimum norm actuator solution" and the solution coming from an arbitrary initial actuator set. The arbitrary actuator set was chosen as the value coming from the first of the 300 random optimization runs. Finally, the red histogram shows the residual norm error between the "minimum norm actuator solution" and the solution computed using the current actuator state as initial point for the optimizer.

Out of the 7000 tests performed, the maximum residual norm between the "minimum norm actuator solution" and the solution computed using $u_{init} = u_0$ is 3.5. This value increases to 7.2 when the arbitrary u_{init} is considered and reaches 10.4 when it is compared with the "maximum norm actuator solution". Interestingly, it was observed that in a total of 20 tests, the solution computed using $u_{init} = u_0$ leads to the lowest residual solution.

To evaluate the secondary objective term of the cost function, the actuator solution displacement from the current actuator state was also analyzed. A bigger actuator displacement in the solution involves a rapid and sudden change of the actuator state. In certain circumstances we would like to limit the actuator displacement in favor of a slightly bigger residual to avoid sudden jumps in the actuator solution and to achieve a smooth actuator evolution over time. This is particularly important in our case, where the actuators are mainly constituted of servo motors. To evaluate this parameter, the scaled actuator displacement of the solution computed using different u_{init} conditions is shown in Figure 3.16. In this figure, the magenta histogram shows the scaled solution displacement from u_0 of the "minimum norm solution", the red histogram represents the scaled solution displacement using the arbitrary initial actuator set while the green histogram is the scaled solution displacement when using the current actuator state as initial value for the optimizer.

Starting the optimization from different initial actuator states also provides different actuator solutions and therefore different residuals. However, the initial actuator state leading to a solution with minimal residual norm is not known in advance. The only two plausible options when running the Nonlinear optimization process in real time are either to use an arbitrary actuator state or to use the current actuator state as initial optimization point. The results in Figure 3.15 clearly shows that starting the optimizer from the current actuator state provides a lower residual norm than starting it from an arbitrary actuator point.

At the same time, it is clear from Figure 3.16 that initializing the Nonlinear CA algorithm from an initial actuator set different from the current one, leads to a solution with higher displacement from u_0. Therefore, even if by starting the optimization process at the current actuator state we might not be able to find the solution with the lowest norm, we expect to find the sub-optimal solution with minimum actuator displacement. This choice should mitigate sudden jumps in the actuator solution, guaranteeing a smooth actuator evolution over time.

Randomized variables	**Min value**	**Max value**
Motor rotational speed Ω_i	150 rad/s	950 rad/s
Elevation tilting angle b_i	-90 deg	25 deg
Azimuth tilting angle g_i	-45 deg	45 deg
Desired linear acceleration increment	-5 $\frac{m}{s^2}$	5 $\frac{m}{s^2}$
Desired angular acceleration increment	-5 $\frac{rad}{s^2}$	5 $\frac{rad}{s^2}$
Pitch angle θ	-20 deg	20 deg
Roll angle ϕ	-20 deg	20 deg
Forward speed V_x	0 m/s	3 m/s
Constant variables	**Value**	
Angular rates p, q, r	0	
Lateral and vertical speed V_y, V_z	0	
Yaw angle ψ	0	

Table 3.5: Variables definition for the statistical analysis of the Nonlinear CA solution

3.6. CONCLUSIONS

In this chapter we addressed the CA problem of hybrid UAV with nonlinear effectiveness actuators, such as tilting rotor vehicles.

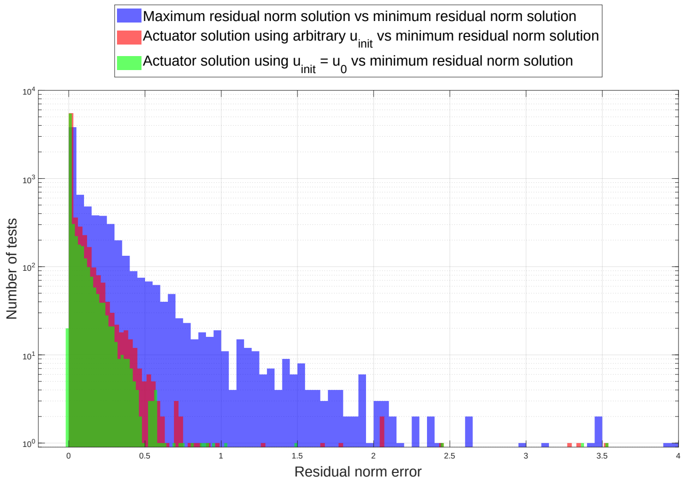


Figure 3.15: Residual norm error between the minimum residual norm solution and the solution computed using different u_{init} conditions. Notice that the y-axis is logarithmic and some points are not fully represented. One point of the red histogram and four points of the blue histogram are above the residual norm error of 4. These points are mentioned in the Statistical analysis results Section.

Through flight tests and simulations we proved that state-of-the-art linearized effectiveness CA algorithms like the PIU and WLS are not capable of determining a smooth and effective actuator solution for tilting rotor vehicles. We proposed a Nonlinear CA algorithm capable of computing an optimal and smooth actuator solution for such vehicles. Furthermore, the vehicle performance in terms of attitude and position tracking obtained using the Nonlinear CA algorithm outperformed the one obtained using any other tested CA method. It was observed that under high control objective commands or when an uneven control input weighting matrix W_u is used, the solution computed by linearized effectiveness algorithms is characterized by uncontrolled and wide oscillatory tilting angles, coupled with an inaccurate motor power solution. In an attempt to mitigate the oscillatory behavior of the WLS CA algorithm, a local tilting constraint was included in the WLS problem structure. Although potentially beneficial, this condition only bounds the oscillations of the tilting solution at the expense of an added acceleration response delay, inversely proportional to the applied angular region of constraint δ_c.

Concerning possible future development, even if no criticalities were found during the flight tests, a statistical analysis on the cost function used in the Nonlinear CA algorithm revealed the presence of some local

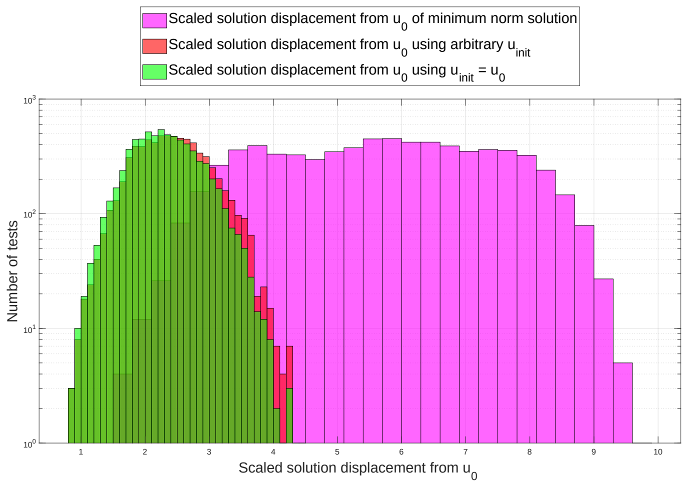


Figure 3.16: Scaled displacement of the computed actuator solution from the current actuator value for different u_{init} conditions. Notice that the y-axis is logarithmic. The displacement is defined as the norm of the difference in normalized control input $\left|u_n - u_{n0}\right|$, with u_n as defined in Equation 3.25.

minimum points. Future work should focus on this aspect, trying to modify the cost function in such a way to eliminate the presence of local minima in it. Furthermore, the Nonlinear CA algorithm could be extended to forward flight, with the aim of developing a unified control strategy for hovering, transitioning and forward flight of tilt rotor hybrid vehicles.

REFERENCES

[1] A.Mancinelli, E.J.J.Smeur, B.Remes, and G.DeCroon. “Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design.” In: *International Conference on Unmanned Aircraft Systems (ICUAS)* (2022). doi: 10.1109/ICUAS54217.2022.9836063.

[2] E.Sadien, C.Roos, M. A.Birouche, C.Grimault, L.E.Romana, and M.Basset. “A detailed comparison of control allocation techniques on a realistic on-ground aircraft benchmark.” In: *American Control Conference 2019* (2019). doi: 10.23919/ACC.2019.8814718.

[3] A.Johansen and I.Fossen. “Control allocation — A survey.” In: *Automatica* Volume 49, Issue 5 (2013). doi: 10.1016/j.automatica.2013.01.035.

[4] M.Bodson. “Evaluation of Optimization Methods for Control Allocation.” In: *Journal of Guidance, Control and Dynamics* Volume 25, No. 4 (2002). doi: 10.2514/2.4937.

[5] J.Barata and M.Hussein. “The Moore–Penrose Pseudoinverse: A Tutorial Review of the Theory.” In: *Brazilian Journal of Physics* Volume 42 (2012). doi: 10.1007/s13538-011-0052-z.

[6] J.Shi, W.Zhang, G.Li, and X.Liu. “Research on allocation efficiency of the redistributed pseudo inverse algorithm.” In: *Science China Information Sciences* Volume 53 (2010). doi: 10.1007/s11432-010-0032-x.

[7] W.C.Durham. “Constrained control allocation.” In: *Journal of Guidance, Control and Dynamics* Volume 16, No. 4 (1993). doi: 10.2514/3.21072.

[8] M.Bodson and S.A.Frost. “Load Balancing in Control Allocation.” In: *Journal of Guidance, Control and Dynamics* Volume 34, No. 2 (2011). doi: 10.2514/1.51952.

[9] J.A.M.Petersen and M.Bodson. “Constrained quadratic programming techniques for control allocation.” In: *IEEE Transactions on Control Systems Technology* Volume 14, No. 1 (2006). doi: 10.1109/TCST.2005.860516.

[10] O.Harkegard. “Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation.” In: *Proceedings of the 41st IEEE Conference on Decision and Control* Volume 2 (2002). doi: 10.1109/CDC.2002.1184694.

[11] V.L.Poonamallee, S.Yurkovich, A.Serrani, and D.B.Doman. “A nonlinear programming approach for control allocation.” In: *Proceedings of the 2004 American Control Conference* Volume 2 (2004). doi: 10.23919/ACC.2004.1386822.

[12] J.Wang and R.G.Longoria. “Coordinated and Reconfigurable Vehicle Dynamics Control.” In: *Transactions on Control Systems Technology* Volume 17, No. 3 (2009). doi: 10.1109/TCST.2008.2002264.

[13] Y.Chen and J.Wang. “Fast and Global Optimal Energy-Efficient Control Allocation With Applications to Over-Actuated Electric Ground Vehicles.” In: *IEEE Transactions on Control Systems Technology* Volume 20, No. 5 (2012). doi: 10.1109/TCST.2011.2161989.

[14] Y.Chen and J.Wang. “A global optimization algorithm for energy-efficient control allocation of over-actuated systems.” In: *Proceedings of the 2011 American Control Conference* (2011). doi: 10.1109/ACC.2011.5990899.

[15] H.Huan, W.Wan, C.We, and Y.He. “Constrained Nonlinear Control Allocation based on Deep Auto-Encoder Neural Networks.” In: *European Control Conference (ECC)* (2018). doi: 10.23919/ECC.2018.8550445.

[16] H.Khan, S.Mobeen, J.Rajput, and J.Riaz. “Nonlinear Control Allocation: A Learning Based Approach.” In: *arXiv* (2022). doi: 10.48550/arXiv.2201.06180.

[17] P. Simplício, M. Pavel, E. van Kampen, and Q. Chu. “An acceleration measurements-based approach for helicopter nonlinear flight control using Incremental Nonlinear Dynamic Inversion.” In: *Control Engineering Practice* 21 Issue 8 (2013). doi: 10.1016/j.conengprac.2013.03.009.

[18] E.J.J.Smeur, Q.Chu, and G. Croon. “Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles.” In: *Journal of Guidance, Control and Dynamics* Volume 39, No. 3 (2016). doi: 10.2514/1.G001490.

[19] J. Wright. “Numerical Optimization.” In: *Editor: Springer - ISBN: 978-0-38-730303-1* (1951).

[20] K. Schittkowski. “NLPQL: A fortran subroutine solving constrained nonlinear programming problems.” In: *Annals of Operations Research* Volume 5 (1986). doi: 10.1007/BF02739235.

[21] B. Gati. “Open source autopilot for academic research - The Paparazzi system.” In: *2013 American Control Conference* (2013). doi: 10.1109/ACC.2013.6580045.

4

FULL flIGHT ENVELOPE flIGHT CONTROL SYSTEM DESIGN

In Chapter 3, a nonlinear programming-based controller was developed to control the dual-axis tilting rotor quad-plane within the hovering flight regime. The next step, addressed in this chapter, is to extend that controller to operate across the vehicle's full flight envelope. Due to its hardware configuration, the quad-plane possesses full 6 DOF control authority in hover but effectively retains only 4 DOF during forward flight. This reduction in available control authority demands a control strategy capable of smoothly managing the transition between these flight modes.

In this chapter, the controller developed in Chapter 3 is extended to achieve smooth and robust control across the entire flight envelope, from hovering to cruise flight. The proposed framework ensures consistent performance by dynamically reallocating control objectives and adapting to the varying degrees of control authority available throughout the transition from low-speed to high-speed flight.

This chapter is based on the following article:
A. Mancinelli, B.D.W. Remes, G.C.H.E. de Croon and E.J.J. Smeur, "Unified Incremental Nonlinear Controller for the Transition Control of a Hybrid Dual-Axis Tilting Rotor Quad-Plane," in IEEE Transactions on Robotics, vol. 41, pp. 306-325, 2025.

4.1. INTRODUCTION

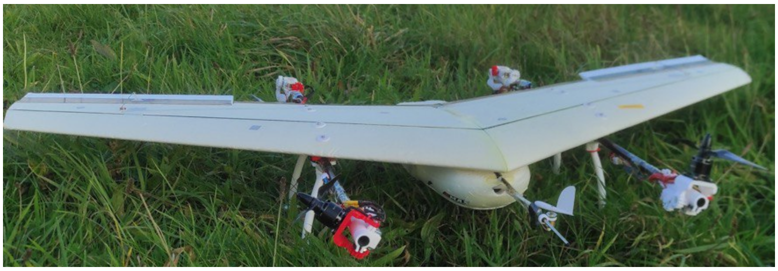


Figure 4.1: A picture of the hybrid dual-axis tilting rotor quad-plane. The unique characteristic of this vehicle lies in its dual-axis tilting rotors, which can rotate independently in both lateral and longitudinal directions, as illustrated in Fig. 4.2.

Hybrid UAVs represent a unique class of aircraft, seamlessly integrating VTOL capabilities typically associated with helicopters with the extended endurance and operational efficiency characteristic of fixed-wing planes. Categories of hybrid lift UAVs that have received significant attention in research include tailsitters [1], quad-planes [2], tilting wings [3], and tilting rotors [4].

Each of these vehicles has its own set of advantages and disadvantages. However, among this diverse spectrum of hybrid UAV configurations, one particular type of vehicle has demonstrated exceptional suitability for operations in gusty environments: the Tilt Rotor Unmanned Aerial Vehicle (TRUAV). TRUAVs distinguish themselves by their ability to rapidly adjust rotor orientation, enabling them to respond swiftly to wind disturbances and minimize the impact of wind on the vehicle's position [5]. Additionally, TRUAVs can be designed as overactuated systems, allowing for independent control of both their position and attitude. This characteristic enhances their versatility and adaptability [6, 7].

Nonetheless, the control of TRUAVs comes with several challenges that have somewhat constrained their widespread adoption. The primary challenge in controlling TRUAVs comes from the fact that their dynamics are non-affine in the control input, which presents a significant hurdle. Unlike conventional UAVs, where thrust can be assumed to be consistently applied in the same direction in the body frame, TRUAVs have the capacity to alter the direction of thrusters within the body frame. This characteristic results in a pronounced nonlinear relationship between the control inputs and the generated vehicle accelerations. Consequently, when the tilting dynamics are particularly rapid, the CA problem becomes increasingly complex and cannot be adequately addressed using conventional algorithms relying on linearized control effectiveness simplification [8].

In a notable study, Papachristos et al. [9] addressed the challenge of controlling the control input non-affine dynamics of rotor tilting. Constrained by the computational cost of a pure Model Predictive Control (MPC) controller, they implemented a hybrid MPC/PID controller. They utilized a nonlinear MPC to generate tilting and attitude commands. Subsequently, the tilting commands were sent to the actuators, while the attitude commands were executed through a gain-scheduled PID controller controlling the motor commands. A similar approach was also proposed by Zelong Yu et al. [10], who used a back-stepping Sliding Model controller to solve the control allocation problem. In a recent study [8], we developed a nonlinear incremental CA algorithm employing a nonlinear solver capable of directly controlling both linear and angular accelerations through the generation of motor and tilt commands. However, it is important to note that these works primarily focused on controlling the vehicle in the hovering configuration, with limited analysis of the transition to forward flight. Consequently, this limitation significantly restricts the applicability of these controllers, mainly to low airspeeds.

Another significant challenge encountered by hybrid TRUAVs is the transition from low-speed to high-speed flight regimes. This transition necessitates efficient allocation of control objective accelerations between direct actuator commands and the vehicle's attitude. To illustrate this challenge, let's consider the transition control from low-speed to high-speed flight of the dual-axis tilt rotor quad-plane, presented in [5] and depicted in Figure 4.1.

At low airspeed, the vehicle's 4 motors and 8 tilting servos provide direct control over all 6 DOF, enabling the achievement of any linear and angular acceleration through direct actuator commands. However, as the vehicle transitions to high airspeed, roll control must be achieved using ailerons, and vertical acceleration is more effectively accomplished through pitch adjustments rather than motor or tilt commands. Additionally, lateral acceleration can only be attained through roll adjustments, as the lateral tilt actuator becomes ineffective due to rotor gimbal lock. The gimbal lock condition of the lateral rotor tilt can be understood by referring to Figure 4.2, where the tilting angles of the dual-axis tilting rotor quadplane are defined. Assuming the rotors are fully oriented forward (e.g., with the tilt angle b set at -90 degrees), a lateral tilt angle rotation g results in a rotation around the spinning axis of the motor, rendering it ineffective for tilting the thrust. Consequently, despite the vehicle possessing 6 DOF in the hovering configuration, it retains only 4 DOF authority in the high-speed regime. This necessitates the development of a distinct control strategy to achieve lateral and vertical accelerations in different flight regimes.

The challenge of allocating accelerations across different actuators

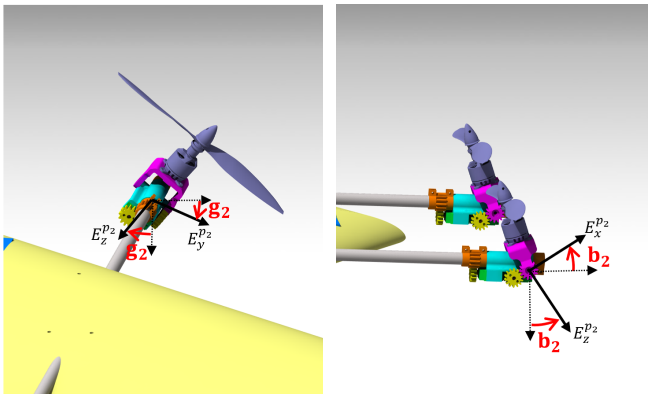


Figure 4.2: Close-up of the rotor tilting mechanism for rotor number two (front right) on the dual-axis tilting rotor quad-plane. On the left side of the figure, the lateral tilt or azimuth angle *g* is defined. On the right side of the figure, the longitudinal tilt or elevation angle *b* is defined.

or states in various flight regimes is well-documented in the literature. For vehicles like tail-sitters or quad-copters, where the thrust vector is aligned with the body reference frame, this issue can be effectively addressed using a differential flatness-based controller, which enables smooth trajectory control and agile maneuvering [11–13]. More broadly, the most common method for addressing this challenge involves an outer-inner loop control framework. In this framework, the outer loop generates attitude commands based on position references, which are then allocated to the physical actuators by the inner loop [14, 15]. Another popular solution involves separating the vehicle's transition control using one or multiple intermediate states. These states, combined with a switched logic controller, allow for separate management of each flight phase [16–18]. The controllers developed for these approaches range from Linear–Quadratic Regulator (LQR) [19, 20] to simple PID [21].

Other interesting works that focus on creating a uniform control framework for the transition control of tilt-wing UAVs have utilized H-infinity control theory [22, 23]. Recently, Raab et al. [24] developed a uniform controller based on Incremental Nonlinear Dynamic Inversion (INDI) specifically for TRUAV vehicles. This proposed control solution employs a single-loop INDI control structure. At the core of their incremental control strategy is the generation of pitch and roll commands directly within the CA inversion loop, referred to as virtual commands. These virtual commands are derived from the linearized effectiveness of the pitch and roll angles and are then applied to the system using an INDI control scheme.

While this control architecture shows promise and has yielded favorable results, it does have certain limitations. The primary limitation of this study is that the proposed CA law is based on a linearized control effectiveness and solely utilizes motor and control surface commands to achieve the desired control actions. This approach treats the tilting angle as an external variable rather than incorporating it as a control input. Consequently, this controller may not be suitable for vehicles equipped with quick-tilting actuators that can be directly employed into the vehicle's control. Furthermore, the CA algorithm employed in this framework does not facilitate independent attitude and position control. This capability is particularly crucial for precision landing on a tilting platform, such as a moving vessel.

In this chapter, we propose a unified nonlinear control framework for a hybrid dual-axis tilting rotor quad-plane, capable of addressing the previously analyzed TRUAV control challenges. Inspired by the virtual commands approach proposed in [24], we modified the cost function of the nonlinear optimization process presented in [8] to compute the references for the roll and pitch angles within the nonlinear CA algorithm. Additionally, we incorporated an Angle of Attack (AoA) protection system and an accelerometer-based sideslip controller into the nonlinear INDI control laws to minimize sideslip during coordinated turns in the high-speed flight regime. The proposed control strategy also allows for the achievement of desired pitch and roll angles at low airspeed when the vehicle maintains 6 DOF control authority.

The developed control strategy has been implemented and tested on the dual-axis tilting rotor quad-plane depicted in Figure 4.1. Real-time computation for the 8 tilt rotor servos, 4 motor commands, ailerons servos and the two virtual attitude commands was performed using a Raspberry Pi 4B companion computer, at an average refresh rate of 220 Hz.

Results from flight tests have demonstrated the effectiveness of the control strategy. Throughout the flight test maneuvers, the vehicle exhibited precise tracking of linear and angular acceleration and effectively managed the loss of DOF during the transition. It is important to highlight that we were unable to perform comparisons with other algorithms, as no other existing algorithms are capable of performing the same task under the same conditions. This emphasizes the novelty of the approach and the gap it fills in the field.

To summarize, the key contributions of this research are as follows:

1. We present a unified single-loop incremental nonlinear controller capable of seamlessly controlling hybrid TRUAVs across their full flying envelope. Unlike state-of-the-art control algorithms based on linearized effectiveness, the proposed controller is capable of producing smooth and effective control input commands,

accounting for the nonlinearities of the tilting system.

2. We demonstrate the applicability of this nonlinear programming-based controller by implementing and running it in real-time on a small 2.5 kg UAV with limited computational capabilities, opening the source code to the community.

The chapter is structured as follows: in section 4.2, we derive the incremental Nonlinear Unified CA problem which generates the commands for both the attitude and the physical vehicle actuators. Section 4.3 introduces the control framework supporting the Incremental Nonlinear Unified CA algorithm for the dual-axis tilt rotor quad-plane. Section 4.4 completes the setup of the unified controller and provides some characteristic parameters of the flying vehicle. Section 4.5 offers insights into the controller's working principle through simulation results and includes a statistical local minima analysis on the cost function. Section 4.6 is dedicated to presenting and analyzing flight test results. In section 4.7, we discuss some limitations of the Nonlinear Unified CA algorithm. Finally, Section 4.8 summarizes our conclusions.

4.2. NONLINEAR CONTROL ALLOCATION PROBLEM DEFINITION

Starting from the nonlinear control allocation formulation defined in Equation 3.9, we can rewrite the associated optimization problem, based on the same cost function, using the normalized control input $\tilde{\boldsymbol{u}}$:

$$\begin{gathered}\tilde{\boldsymbol{u}}_{\boldsymbol{cmd}} = \arg\min \tilde{C}(\tilde{\boldsymbol{u}}) \\ \text{subject to} \\ \tilde{\boldsymbol{u}}_{min} < \tilde{\boldsymbol{u}} < \tilde{\boldsymbol{u}}_{max}.\end{gathered} \tag{4.1}$$

Where $\tilde{C}(\tilde{\boldsymbol{u}})$ represents the nonlinear cost function as defined in equation 3.9, modified to use normalized control input, while $\tilde{\boldsymbol{u}}_{max}$ and $\tilde{\boldsymbol{u}}_{min}$ specify the constraints on the normalized control input solution. The use of the normalized control input $\tilde{\boldsymbol{u}}$ is essential to enable the optimization algorithm to equally iterate over actuators with different units of measure and travel. This is due to the fact that each element of the control input array uses different units of measurement. For example, motors rotation is expressed in rad/s while tilting angles are expressed in radians. The normalized input vector $\tilde{\boldsymbol{u}}$ is defined as follows:

$$\tilde{\boldsymbol{u}} = \boldsymbol{u} \oslash \frac{\boldsymbol{u}_{max} - \boldsymbol{u}_{min}}{2}. \tag{4.2}$$

Where the symbol $\oslash$ represents the Hadamard division. In this normalized input vector, $\tilde{\boldsymbol{u}}$, all actuators are represented with a value between -1 and 1, irrespective of their actual physical range or unit of measurement. As a consequence, also the optimization algorithm will output a normalized actuator solution, which must then be re-scaled to their original dimensions before being applied to the actuators. It's important to note that the modified cost function, $\tilde{C}(\tilde{\boldsymbol{u}})$, accepts a normalized control input, $\tilde{\boldsymbol{u}}$, but the cost value is not normalized.

To reduce the complexity of the problem, the CA problem associated with our vehicle employs a simplified version of the system dynamics, denoted as $\boldsymbol{f}_s(\boldsymbol{x},\boldsymbol{u})$, instead of the full system dynamics $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$. The simplified system dynamics used in this work are defined as follows:

$$\begin{cases} \ddot{P_c} = \frac{1}{m}(F^p + F^a) + g\hat{z}_e \\ \dot{\omega} = I_b^{-1}\left(-\omega \times I_b\omega + M^t + M^d + M^a + M^{\delta_a}\right) \\ f_s(\boldsymbol{x},\boldsymbol{u}) = (\ddot{P_c}\ ,\ \dot{\omega})^T \end{cases} \tag{4.3}$$

A detailed derivation of each term in Equation 4.3 is provided in Chapter 2.3. The only term not described in that chapter is M^{δ_a}, which represents the torque generated by the ailerons. It is defined as:

$$M^{\delta_a} = \begin{pmatrix} M_L^{\delta_a} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\rho S V_a^2\ \bar{c}\ C_{M_L}^{\delta_a}\ \delta_a \\ 0 \\ 0 \end{pmatrix}, \tag{4.4}$$

where ρ is the air density, S is the wing surface area, V_a is the airspeed, $\bar{c}$ is the mean aerodynamic chord of the wing, δ_a is the aileron deflection angle and $C_{M_L}^{\delta_a}$ is the aileron moment coefficient, which can be identified through flight test experiments.

Note that the term $\ddot{P_c}$ models the inertial accelerations in the control reference frame, rather than in the Earth reference frame as used in Chapter 2.

The coordinate transformation between the Earth reference frame Γ_e and the control reference frame Γ_c is described by the following rotation matrix:

$$R_{ec} = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.5}$$

$$\text{such that } \bar{x}_e = R_{ec} \cdot \bar{x}_c$$

where ψ is the yaw Euler angle in the conventional ZYX order.

4.2.1. CONTROL INPUT VECTOR DEFINITION

Regarding the definition of the control input vector $\boldsymbol{u}$ utilized in the Control Allocation Optimization problem, we have adopted the subsequent control input vector:

$$\boldsymbol{u} = (\boldsymbol{u}_a, \phi, \theta)^T. \tag{4.6}$$

Here, the first element, $\boldsymbol{u}_a$, represents an array containing the physical command to each of the actuators, while the last two elements, ϕ and θ, are referred to as virtual commands.

The unconventional definition of the control input vector in Equation 4.6, encompassing both physical and virtual actuators, is essential to devise a control law capable of seamlessly allocating the acceleration increments across varying airspeed conditions.

In the hovering configuration, at low airspeed, the vehicle's actuators can directly generate all three forces and three moments required for the 6 DOF control of the vehicle. However, as the airspeed increases, we need to incorporate lift as a means of controlling lateral and vertical acceleration.

To accomplish this, we incorporated roll and pitch angles as active optimization control inputs. These angles are no longer treated as current vehicle states but are considered as variables to be optimized, similar to the physical actuators of the system. Leveraging the vehicle's aerodynamic properties and the choice of elements in $\boldsymbol{W}_{\boldsymbol{u}}$, the optimization process determines whether to achieve a specific linear acceleration by actuating the physical actuators or by adjusting the vehicle's attitude. The adoption of both attitude commands and physical actuator commands in the control allocation problem is made feasible because the nonlinear optimization process re-evaluates the EOM at every iteration and thus continuously adjusts the real actuator control inputs to the new attitude.

Including the generation of roll and pitch commands in the control input array provides an additional advantage. It enables the integration of desired roll and pitch commands into the desired actuator command array, $\boldsymbol{u}_d$, which is part of the secondary objective in the cost function. This means that the control input commands calculated by the optimizer will always aim to meet the roll and pitch conditions specified in the $\boldsymbol{u}_d$ array.

However, as airspeed increases, the vehicle will increasingly rely on roll or pitch changes to achieve the lateral or vertical acceleration increments. As acceleration increments are a component of the primary objective within the cost function, the roll and pitch commands will be primarily computed to fulfill lateral and vertical accelerations, therefore discarding the desired pitch and roll inputs in $\boldsymbol{u}_{\boldsymbol{d}}$.

4.3. UNIFIED INCREMENTAL NONLINEAR CONTROL FRAMEWORK

In this section, we present the unified incremental nonlinear control framework, which consists of three main blocks.

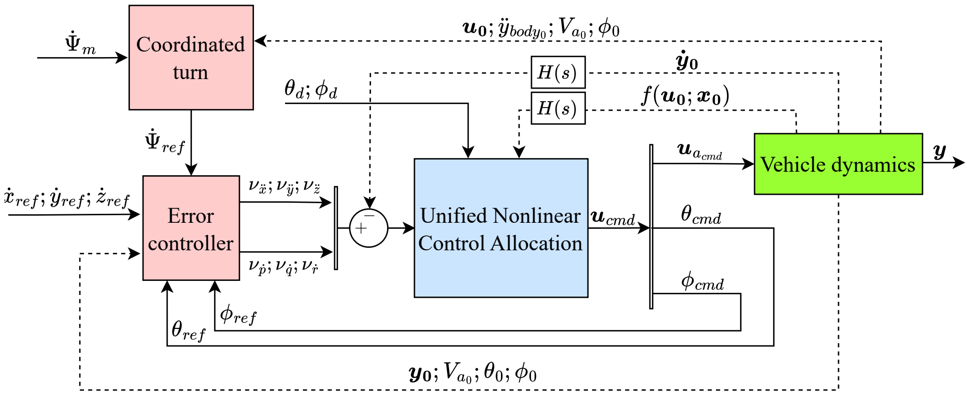

$\dot{\Psi}_m$
Coordinated turn
$\boldsymbol{u_0}; \ddot{y}_{body_0}; V_{a_0}; \phi_0$
$H(s)$
$\dot{\boldsymbol{y}}_0$
$\theta_d; \phi_d$
$H(s)$
$f(\boldsymbol{u_0}; \boldsymbol{x_0})$
$\dot{\Psi}_{ref}$
$\boldsymbol{u}_{a_{cmd}}$
Vehicle dynamics
$\boldsymbol{y}$
$\dot{x}_{ref}; \dot{y}_{ref}; \dot{z}_{ref}$
$\nu_{\ddot{x}}; \nu_{\ddot{y}}; \nu_{\ddot{z}}$
Error controller
$-$
$+$
Unified Nonlinear Control Allocation
$\boldsymbol{u}_{cmd}$
θ_{cmd}
$\nu_{\dot{p}}; \nu_{\dot{q}}; \nu_{\dot{r}}$
ϕ_{cmd}
ϕ_{ref}
θ_{ref}
$\boldsymbol{y_0}; V_{a_0}; \theta_0; \phi_0$

Figure 4.3: Diagram of the Unified Incremental Nonlinear Controller. In the diagram, dashed lines represent variables estimated through sensors (for vehicle states and accelerations) or the actuator model (for the current actuator state vector). The red blocks run on the primary flight computer, the blue block runs on the Raspberry Pi, and the green block represents the vehicle dynamics evolution. The control scheme presented is capable of controlling the vehicle's full 6 DOF in a single loop under any flight condition. This capability is achieved through the generation of attitude commands directly in the control allocation block. The transfer functions $H(s)$ represent a second-order low pass filter used to remove noise from the sensor readings.

Coordinated Turn Block: This block is dedicated to generating the yaw rate reference. Its role is to control turns during high-airspeed flight and minimize sideslip.

Error Controller block: This block is responsible for generating the control objective array $\boldsymbol{\nu}$ composed of linear and angular accelerations, derived from speed references and Euler angle references.

Unified Nonlinear Control Allocation Block: This block takes as input the current modeled accelerations and the control objective accelerations increments $\boldsymbol{\nu} - \dot{\boldsymbol{y}}_0$. By solving the CA problem, it outputs attitude and physical actuators commands.

A schematic representation of the unified incremental nonlinear control framework is provided in Figure 4.3. Further elaboration on each block is presented in the subsequent subsections.

4.3.1. COORDINATED TURN BLOCK

The coordinated turn block generates the yaw rate reference, which is then fed into the error controller. This yaw rate reference plays a crucial role at high airspeed and during transition, as it enables the execution of a coordinated turn while minimizing sideslip. The governing equation for coordinated turn block is as follows:

$$\dot{\Psi}_{ref} = 9.81\frac{\tan(\phi)}{V_{a_c}}K_{\dot{\psi}_{air}} - \dot{y}_c K_\beta + \dot{\Psi}_m \tag{4.7}$$

Where $\dot{\Psi}_m$ is the manual hovering yaw rate, $\dot{y}_c$ is the corrected body lateral acceleration, V_{a_c} is a the corrected airspeed and $K_{\dot{\psi}_{air}}$ and K_β are the yaw rate and sideslip correction gains, all defined later in the section. It is possible to identify three main components in Equation 4.7.

The first component is a roll angle feed-forward, which is essential for executing coordinated turns effectively, as elaborated in detail in [25]. To prevent undesired yaw actions at low speed, the gain $K_{\dot{\psi}_{air}}$ was introduced. This gain scales the roll angle feed-forward component, and it is defined as follows:

$$K_{\dot{\psi}_{air}} = \begin{cases} 0 & V_a < V_{\dot{\psi}_{min}} \\ \dfrac{V_a - V_{\dot{\psi}_{min}}}{V_{\dot{\psi}_t} - V_{\dot{\psi}_{min}}} & V_{\dot{\psi}_{min}} < V_a < V_{\dot{\psi}_t} \\ 1 & V_a > V_{\dot{\psi}_t}. \end{cases} \tag{4.8}$$

This gain is crucial because, at low airspeed, the vehicle can achieve a desired roll and pitch angle included in $\boldsymbol{u_d}$. When a desired roll angle is attained, the feed-forward component of the control law in equation 4.7 may generate an undesirable yaw rate command. The $K_{\dot{\psi}_{air}}$ gain is designed to reduce the roll feed-forward term to zero below a minimum speed $V_{\dot{\psi}_{min}}$. The gain gradually increases to 1 between $V_{\dot{\psi}_{min}}$ and $V_{\dot{\psi}_t}$ to ensure smooth transitions in the yaw rate reference generation. For the dual-axis tilting rotor quad-plane, the reference speeds for the coordinated turn block are chosen as follows:

$$V_{\dot{\psi}_t} = 4\,\frac{m}{s} \quad , V_{\dot{\psi}_{min}} = 6\,\frac{m}{s}. \tag{4.9}$$

Note that the feed-forward roll angle law component in equation 4.7 assumes a consistently positive AoA. This assumption implies that a distinct yaw rate law is necessary for scenarios where the vehicle executes turns at a negative AoA. Furthermore, the term V_{a_c} within equation 4.7 represents a corrected airspeed value, defined as follows:

$$V_{a_c} = \underset{(}{\max}\ 10, V_a);$$

This correction is essential to prevent the feed-forward term from reaching infinity when V_a is equal to zero.

The second component of the yaw rate generation law in equation 4.7 is the side-slip correction term. This term utilizes feedback from the corrected body lateral acceleration $\dot{y}_c$ to mitigate side-slip at high airspeed. The corrected body lateral acceleration $\dot{y}_c$ comprises a sensed y-body acceleration term $\dot{y}_0^{body}$, which is adjusted to eliminate the acceleration contribution induced by the sideways tilting of the rotors:

$$\begin{aligned}\dot{y}_c = \dot{y}_0^{body} - \frac{K_p^T}{m}\Big(\Omega_1^2 \sin(g_1)\cos(b_1) + \Omega_2^2 \sin(g_2)\cos(b_2) + \\ + \Omega_3^2 \sin(g_3)\cos(b_3) + \Omega_4^2 \sin(g_4)\cos(b_4)\Big).\end{aligned} \tag{4.10}$$

Where Ω represents the motor rotation, b denotes the longitudinal tilt of the rotor, and g indicates the lateral tilt of the rotor, as illustrated in Figure 4.2. For the rotor numbering definition, the reader can refer to Figure 4.4. The assumption of a proportional relation between the

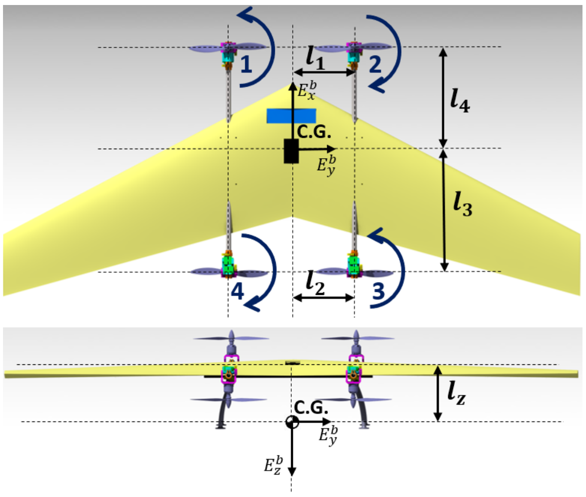


Figure 4.4: Motor spinning direction, rotor numbering and vehicle geometry with respect to the Center of Gravity (CG).

side-slip angle and the corrected body lateral acceleration is derived from a detailed analysis carried out by Smeur et al. [1].

The third and final component of the yaw rate generation law in equation 4.7 is a manual yaw rate $\dot{\Psi}_m$. This component can be used to manually input a heading change at low airspeed to complete the 6 DOF control.

4.3.2. ERROR CONTROLLER BLOCK

For the generation of linear and angular acceleration commands, a linear error controller is employed. The error controller plays a crucial role in correcting inaccuracies in the model and compensating for external disturbances [26]. This is especially important because we are using a simplified version of the system dynamics. Consequently, the secondary terms of the system dynamics that are not considered in the Control Allocation problem are treated as unmodeled dynamics and managed by the error controller. A scheme of the error controller is depicted in Figure 4.5.

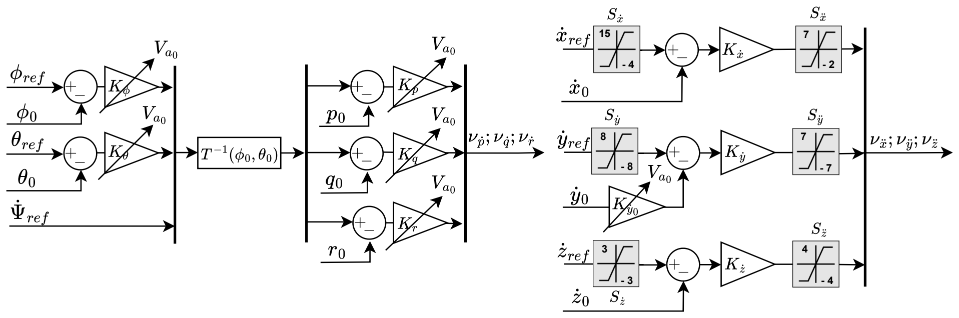


Figure 4.5: Diagram of the Linear Error Controller block included in the Unified Incremental Nonlinear Controller, as presented in Figure 4.3. The right side of the diagram generates the control objective for the linear accelerations, while the left side provides insight into how the control objectives for the angular accelerations are generated. The numerical values of the gains used for the dual-axis tilt rotor quad-plane are reported in Table 4.1.

The reference frame employed for generating linear accelerations is the control reference frame. This reference frame corresponds to an Earth reference frame, but it is rotated around the Z axis by the vehicle's yaw angle ψ, as described in Equation 4.5.

Manipulating the linear acceleration and speed terms in this reference frame offers two advantages. Firstly, it simplifies the direct application of speed and acceleration constraints to the aircraft's lateral and longitudinal motion using saturation blocks. This is particularly valuable when imposing maximum forward or lateral speeds during hovering since the components of the control reference frame align with those of the vehicle's body frame, making it possible to directly apply a maximum forward speed using a saturation block. Secondly, it provides the flexibility to eliminate the lateral speed feedback term as airspeed increases. This can be achieved by applying zero gain to the $\dot{y}_0$

feedback, as illustrated in Figure 4.5.

Disabling the lateral speed feedback as airspeed increases is essential to prevent the vehicle from accumulating and maintaining a constant lateral speed during high-airspeed flight. This is crucial because, while the vehicle can maintain a constant lateral speed by tilting the rotors sideways at low airspeed, retaining a constant lateral speed in the high-airspeed regime is undesirable and inefficient, as it can lead to side-slip. By deactivating the lateral speed feedback line $\dot{y}_0$, the reference for the lateral speed automatically becomes a lateral acceleration control objective, which can be directly generated through roll commands, initiating a coordinated turn maneuver. The deactivation of the $\dot{y}_0$ feedback line can be accomplished through a gain scheduling process that gradually reduces the $K_{\dot{y}_0}$ gain to zero as airspeed increases. The definition of the $K_{\dot{y}_0}$ gain is provided in Table 4.1.

We implemented a similar gain scheduling strategy to reduce the angular acceleration gains with the airspeed, in response to simulations and field tests. These tests revealed an undesired oscillatory behavior around the pitch axis when the same hovering gains for the attitude control were used in the fast forward flight condition. This behavior can be attributed to the distinct actuators employed by the vehicle to achieve angular accelerations at various airspeeds. For example, during hovering, the vehicle primarily relies on motor differential thrust to create pitch and roll changes. However, at high airspeeds, pitch control is mainly achieved through elevation tilting differential, while roll control relies on aileron deflection. Furthermore, as airspeed increases, aerodynamic effects play a role in dampening the vehicle's attitude dynamics.

For detailed numerical values of all the gains and saturation blocks associated with the error controller, specific to the dual-axis tilt rotor quad-plane, please refer to Table 4.1 and Table 4.2. The numerical values of the gains and saturation blocks result from an initial simulation campaign and were subsequently refined through field tests to achieve a damped and oscillation-free system response.

4.3.3. UNIFIED NONLINEAR CONTROL ALLOCATION BLOCK

The Unified Nonlinear Control Allocation block implements and solves the Control Allocation problem presented in Equation 4.1, using the control input array defined in Equation 4.6. This block provides outputs for the commands of the physical actuators $\boldsymbol{u}_{a_{cmd}}$ and the virtual commands ϕ_{cmd} and θ_{cmd}.

The inputs to this block include the current vehicle state and control input vector $\boldsymbol{x}_0$ and $\boldsymbol{u}_0$, which are required for evaluating the simplified system dynamics $\boldsymbol{f}_s(\boldsymbol{x}_0, \boldsymbol{u}_0)$. Additionally, it takes as inputs the control objective composed of linear and angular acceleration increments and

Error controller block gains	
$K_\phi = K_v \quad s^{-1}$	$K_\theta = K_v \quad s^{-1}$
$K_p = 4K_v \quad s^{-1}$	$K_q = 4K_v \quad s^{-1}$
$K_r = 5K_v \quad s^{-1}$	$K_{\dot{x}} = 1 \quad s^{-1}$
$K_{\dot{y}} = 1 \quad s^{-1}$	$K_{\dot{z}} = 3 \quad s^{-1}$
$K_{\dot{y}_0} = 1 - K_{\psi_{air}}$	$K_v = (1 - 0.03 \cdot V_a)$
Coordinated turn block gains	
$K_\beta = 0.15$ s/m	

Table 4.1: Gains used by the dual-axis tilting rotor quad-plane for the Error Controller block and the coordinated turn block. The speed V_a is expressed in m/s and $K_{\psi_{air}}$ is defined in Equation 4.7.

Saturation block	Min value	Max value
$S_{\dot{x}}$	-4 m/s	15 m/s
$S_{\dot{y}}$	-8 m/s	8 m/s
$S_{\dot{z}}$	-3 m/s	3 m/s
$S_{\ddot{x}}$	-2 m/s^2	7 m/s^2
$S_{\ddot{y}}$	-7 m/s^2	7 m/s^2
$S_{\ddot{z}}$	-4 m/s^2	4 m/s^2

Table 4.2: Maximum and minimum values of the Error Controller saturation blocks depicted in Figure 4.5.

the desired pitch and roll angles, θ_d and ϕ_d. These desired pitch and roll angles are incorporated into the desired control input array $\boldsymbol{u}_d$, which is used in the control allocation problem.

4.4. CONTROLLER SETUP AND PARAMETERS

4.4.1. ACTUATOR VECTOR DEFINITION

The $\boldsymbol{u}_a$ vector for the dual-axis hybrid quad-plane is composed of the following elements:

$$\begin{aligned}\boldsymbol{u}_a = (\, &\Omega_1\ \Omega_2\ \Omega_3\ \Omega_4 \\ &b_1\ b_2\ b_3\ b_4 \\ &g_1\ g_2\ g_3\ g_4\ \delta_a\,),\end{aligned} \tag{4.11}$$

where Ω_i is the rotational speed of the i-th rotor, b_i is the longitudinal tilting angle of the i-th rotor (elevation tilting angle), g_i is the lateral tilting angle of the i-th rotor (azimuth tilting angle) and δ_a is the aileron deflection angle, positive for a positive generation of roll rate. The rotor deflection angles are defined in Figure 4.2 while the rotor numbering definition is displayed in Figure 4.4.

The desired actuator vector, denoted by $\boldsymbol{u_d}$, is incorporated into the secondary objective of the cost function and it represents a preferred state of the actuators. This state is pursued in accordance with the primary objective, with priorities set by the weighting matrices and the control input optimality scale factor. For our platform, the desired normalized control input array is defined as follows:

$$\begin{aligned} \boldsymbol{u}_d = (&150,\ 150,\ 150,\ 150,\\ &0,\ 0,\ 0,\ 0,\\ &0,\ 0,\ 0,\ 0,\\ &0,\ \theta_d,\ \phi_d). \end{aligned} \tag{4.12}$$

The elements in equation 4.12 are arranged according to the same format as detailed in equation 4.6. We choose a non-zero value for the desired motor rotational speed to ensure that the motor does not come to a complete stop during flight while the values ϕ_d and θ_d are treated as external input.

4.4.2. ACTUATOR DYNAMICS IDENTIFICATION

The primary element of the cost function used in the CA problem and presented in Equation **??** necessitates an estimation of the current control input state for its evaluation. A model of the actuators is used to estimate the current actuator state based on the actuator commands. Compared to our prior work [5], we made changes in this chapter, such as adopting a different propeller and battery to better support high-speed forward flight. Consequently, a new system identification process, akin to the one conducted in [5], was executed to identify the actuator dynamics of the vehicle. The results of this system identification process are presented in Table 4.3. For the estimation of θ and ϕ, they are determined using an Extended Kalman Filter (EKF) INS that runs on the autopilot.

4.4.3. THRUST AND TORQUE COEFFICIENTS IDENTIFICATION

Another crucial aspect of thrust modeling involves identifying the propeller thrust and torque coefficients, denoted as K_p^T and K_p^M respectively. To streamline the derivation of the system dynamics in 2.3, we made the assumption that the inflow angle's influence on

Actuator	**Corner frequency ω_c**	**Damping ratio ζ**	**Rate limit**	**Delay**
Motor	25 rad/s	-	-	1 mS
Ailerons	20 rad/s	-	-	15 mS
Tilt elevation	60 rad/s	1.5	11.34 rad/s	15 mS
Tilt azimuth	45 rad/s	1.6	9.95 rad/s	15 mS

Table 4.3: Continuous time characteristics of the dual-axis tilt rotor quad-plane. The ailerons and motors dynamics are identified using a first order transfer function while a second order transfer function with rate limiter is used for the tilting dynamics.

thrust and torque generation is negligible. However, even under this assumption, the propeller thrust and torque coefficients cannot be considered constant across the entire flight envelope. These coefficients still exhibit a dependence on airspeed. To capture this characteristic, a series of wind tunnel motor-propeller tests were conducted to determine the propeller thrust and torque coefficients at various airspeed. These tests were performed with the propeller disk oriented perpendicular to the airspeed direction, and the resulting coefficients are presented in Equation 4.13. It is worth mentioning that this model was developed based on a maximum airspeed of 20 m/s, and as such, it is not applicable to airspeed exceeding this value.

$$\begin{aligned} K_p^T &= 0.55e^{-5}(1 - V_a * 0.025) \\ K_p^M &= 0.94e^{-7}(1 - V_a * 0.025), \end{aligned} \tag{4.13}$$

Actuator	**Min value**	**Max value**
Motors (Ω_i)	150 rad/s	1400 rad/s
Tilt elevation (b_i)	- 120 deg	25 deg
Tilt azimuth (g_i)	- 45 deg	45 deg
Ailerons (δ_a)	- 25 deg	25 deg

Table 4.4: Maximum and minimum physical actuator limits. For the definition of the tilting angles, please refer to Figure 2.5.

4.4.4. CONTROL INPUT CONSTRAINT DEfiNITION

The maximum and minimum values for the physical actuators Ω, b, g and δ_a are defined in Table 4.4. Regarding the virtual commands θ and ϕ, these constraints are interpreted as aircraft attitude limits. A constant constraint of $\phi^{max} = 40$ deg and $\phi^{min} = -40$ deg is applied to the roll angle. The pitch angle constraint, however, has a more nuanced interpretation. If a relationship between the pitch angle and the aircraft's AoA is established, an AoA protection logic can be implemented by enforcing pitch angle constraints at each control iteration. As a result, the maximum and minimum pitch angle values can be dynamically evaluated at every time step using the AoA protection algorithm, which is detailed in the following subsection. This protection ensures the wing can generate maximum lift in the low to mid airspeed regime, preventing wing stall and the associated drag penalty. This enables the effective utilization of all thrust power for forward acceleration, without diverting rotor power to contribute to vertical force generation.

AOA ESTIMATION AND PROTECTION

For small roll angles and in the absence of external wind, a direct relationship can be established between the AoA and the pitch angle:

$$\alpha = \theta - \gamma. \tag{4.14}$$

Here, α denotes the AoA, and γ represents the flight path angle, defined as follows:

$$\gamma = \begin{cases} 0 & V_a \leq 3 \quad or \quad V_a \leq \dot{z} \\ sin^{-1}(-\frac{\dot{z}}{V_a}) & else \end{cases} \tag{4.15}$$

In Equation 4.15, $\dot{z}$ is the vertical speed in the control reference frame, while V_a denotes the vehicle airspeed read by the pitot tube. The piece-wise function definition is necessary to avoid division by zero, moreover the negative sign in Equation 4.15 is due to the fact that the control reference frame's z-axis points downward.

Additionally, it's worth highlighting that the relationship between the AoA and the pitch angle, as described in Equation 4.14, and the definition of the flight path angle γ in Equation 4.15, are also utilized in the Unified Nonlinear Control Allocation block for estimating the relation between the pitch angle and the aerodynamic forces generated by the vehicle.

Based on the relationship described in Equation 4.14, when the flight path angle γ is known, it becomes feasible to determine a pitch angle corresponding to a specific AoA value. Consequently, the pitch angle box constraint applied at each iteration can be computed based on the

AoA constraint as follows:

$$\theta^{min} = \begin{cases} \theta^{hard}_{min} & V_a \leq V^{min}_{\alpha_{prot}} \\ max(\theta^{hard}_{min}, \alpha_{min} + \gamma_0) & V_a > V^{min}_{\alpha_{prot}} \end{cases}$$
$$\theta^{max} = \begin{cases} \theta^{hard}_{max} & V_a \leq V^{min}_{\alpha_{prot}} \\ min(\theta^{hard}_{max}, \alpha_{max} + \gamma_0) & V_a > V^{min}_{\alpha_{prot}} \end{cases} . \tag{4.16}$$

Here, $V^{min}_{\alpha_{prot}}$ represents the speed threshold above which the AoA protection algorithm becomes active. Additionally, α_{min} and α_{max} denote the minimum and maximum angle of attack limits. γ_0 represents the current flight path angle, while θ^{hard}_{min} and θ^{hard}_{max} are the pitch angle minimum and maximum hard limits applied across the entire flight envelope. The specific values chosen for $V^{min}_{\alpha_{prot}}$, θ^{hard}_{min}, θ^{hard}_{max}, α_{min}, and α_{max} for the dual-axis tilting rotor quad-plane are provided in Table 4.5.

$V^{min}_{\alpha_{prot}}$	6	m/s
θ^{hard}_{min}	-20	deg
θ^{hard}_{max}	80	deg
α_{min}	2	deg
α_{max}	15	deg

Table 4.5: Dual-axis tilting rotor quad-plane parameters required for the evaluation of the pitch angle constraint in Equation 4.16

4.4.5. CHOICE AND TUNING OF WEIGHTING MATRICES $\boldsymbol{W_u}$,$\boldsymbol{W_\nu}$ AND OPTIMALITY SCALE FACTOR γ_u

This section discusses the selection of numerical values for the control input weighting matrices $\boldsymbol{W_u}$ and $\boldsymbol{W_\nu}$, as well as the optimality scale factor γ_u. The numerical values specifically identified for the dual-axis tilting rotor quad-plane were determined through extensive tuning in simulation, with the goal of achieving a smooth transition and appropriate control input commands.

CONTROL INPUT WEIGHTING MATRIX $\boldsymbol{W_u}$

The matrix $\boldsymbol{W_u}$ is used to penalize the relative actuator command with respect to $\boldsymbol{u_d}$. We have selected the control input weighting matrix $\boldsymbol{W_u}$ with the intention of discouraging the utilization of lateral tilt angles at high airspeed, while simultaneously promoting the utilization of roll and pitch angles as airspeed increases:

$$\begin{aligned} \boldsymbol{W_u} =& diag(3\,,\,3\,,\,3\,,\,3\,,\,0\,,\,0\,,\,0\,,\,0 \\ & 1.5V_a\,,\,1.5V_a\,,\,1.5V_a\,,\,1.5V_a \\ & 0.5\,,\,100-15V_{a_b}\,,\,100-15V_{a_b}). \end{aligned} \tag{4.17}$$

Where V_a represents the airspeed measured by the Pitot tube in m/s. Note that the last two diagonal elements of the $\boldsymbol{W_u}$ use a different definition of the airspeed, named V_{a_b}. V_{a_b} is a bounded airspeed value within 0 and 100/15 m/s, to avoid the elements of $\boldsymbol{W_u}$ to become negative.

CONTROL OBJECTIVE WEIGHTING MATRIX $\boldsymbol{W_\nu}$

The control objective weighting matrix $\boldsymbol{W_\nu}$ is utilized to prioritize certain acceleration increments over others. Specifically, we chose to give precedence to angular acceleration, followed by vertical acceleration, and lastly, lateral and longitudinal accelerations. The reason to choose this order is that the attitude is used to control the linear acceleration, especially in forward flight. The control objective weighting matrix W_ν chosen for the vehicle is therefore given by:

$$\boldsymbol{W}_\nu = \text{diag}\ (.005,\ .005,\ .008,\ .015,\ .015,\ .015). \tag{4.18}$$

The relatively small magnitudes of the elements in $\boldsymbol{W_\nu}$ were selected because it has been observed that a cost function with smaller magnitudes facilitates faster convergence to the cost function's minimum. This behavior is likely connected to how the merit function is defined in the fmincon implementation of the SQP algorithm, as discussed in [8].

OPTIMALITY SCALE FACTOR γ_u

The optimality scale factor γ_u, as discussed in the previous subsection, differentiates the primary objective from the secondary objective of the cost function. This parameter is critically important: if γ_u is too small, the vehicle control input solution will not be sufficiently attracted to $\boldsymbol{u_d}$; if it is too large, the solution will partially overlook the primary objective of the cost function, leading to a sub-optimal solution with large acceleration residuals. This parameter underwent extensive tuning in simulations, after which a value of $\gamma_u = 1e^{-6}$ was chosen for the vehicle.

4.4.6. Maximum specific vertical force constraint

In the mid to high airspeed ranges, even when we enforce the AoA to always be positive, as illustrated in Table 4.5, situations may arise where, due to the incremental nature of the control law, the control input commands generated through the nonlinear optimization process may seek to induce a vertical acceleration surpassing gravity. In this scenario, the control inputs are designed to generate a net positive (downward) force on the control z-axis, a condition achievable, for example, by tilting the motors below -90 degrees. Consequently, in this particular condition, a different yaw rate reference model should be employed, since the model presented in Equation 4.7 assumes an always negative z-control generated force. Additionally, this condition can lead to the generation of extreme roll angle commands, as the effectiveness in controlling the lateral acceleration significantly diminishes. To prevent this, the following constraint was added to the control input optimization problem presented in Equation 4.1:

$$f(\boldsymbol{x}_0, \boldsymbol{u_{cmd}}) \cdot [0,0,1,0,0,0]^T < 9.81 - \sigma_z. \tag{4.19}$$

Here, the column vector $[0,0,1,0,0,0]^T$ specifies that the constraint is applied exclusively to the third component, which is the vertical acceleration. In the additional margin, σ_z, chosen in our case to be $2.5\ \frac{m}{s^2}$, is introduced to prevent the roll effectiveness from dropping too low, thereby avoiding the computation of extreme roll commands for achieving lateral acceleration control objectives.

4.4.7. Resolution of the control optimization problem

For the solution of the normalized control problem presented in Equation 4.1, we utilized the SQP algorithm. This gradient-based iterative optimization method demonstrated the shortest runtime and the fastest solution convergence for our specific problem among various available optimization algorithms.

A detailed derivation of the SQP algorithm is beyond the scope of this chapter. Interested readers can refer to the work of K. Schittkowski, where the SQP algorithm is thoroughly explained [27].

One of the primary advantages of using the SQP algorithm is its compatibility with the MATLAB *fmincon* function [1]. Furthermore, the *fmincon* function seamlessly integrates with the MATLAB Coder toolbox[2]. This compatibility facilitates the smooth transition of the same SQP algorithm developed in Simulink for simulation to practical testing on a real flying drone. This transition can be achieved by employing an external companion computer, such as a Raspberry Pi 4 running a Linux

[1]https://nl.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html

[2]https://nl.mathworks.com/products/matlab-coder.html

OS. The C code generated by MATLAB can be executed onboard the Raspberry Pi 4.

To ensure that the Unified Nonlinear Control Allocation algorithm runs in real-time we decided to limit the maximum runtime of the optimization to 5 mS. While this constraint may result in a less accurate solution, it guarantees that we achieve an actuator solution with a frequency of at least 200 Hz.

4.4.8. CHOICE OF THE INITIAL POINT FOR THE OPTIMIZER

To conclude the setup of the optimization problem, another crucial aspect to consider is the choice of the initial actuator solution as a starting point for the optimization process. This parameter holds significant importance because an incorrect initialization of the initial control input array can potentially lead to the computation of a sub-optimal control input solution. This situation can occur due to the non-global convexity of the cost function or due to the premature stop of the optimization process as a consequence of an iteration limit or a maximum run-time constraint.

To ensure the shortest possible convergence time for the optimizer, we decided to use the current control input array, $\boldsymbol{u}_0$, as the initial control input state for the optimization process $\boldsymbol{u}_{init}$.

4.4.9. SIGNAL FILTERING

In the Nonlinear Unified Controller scheme depicted in Figure 4.3, we've integrated two second order Butterworth filters, labeled as $H(s)$. The filter on $\dot{\boldsymbol{y}}_0$ serves the role of removing high-frequency sensor noise. Given the incremental nature of the controller, the same Butterworth filter is also applied to the modeled acceleration term $f(\boldsymbol{x}_0, \boldsymbol{u}_0)$, as described in [1]. For our specific vehicle, we utilized a second-order Butterworth filter with a cutoff frequency of $\omega_n = 13$ rad/s.

4.5. SIMULATION ANALYSIS

To gain a comprehensive understanding of the controller's working principle and to tune the weights and gains in the controller framework, a Simulink simulation environment was created. This environment encompasses the vehicle dynamic model, and the controller outlined in the preceding sections. The Simulink model includes a synthetic representation of the vehicle, facilitating the visualization of both current and commanded states for actuators and attitude.

4.5.1. CONTROLLER'S WORKING PRINCIPLE

In Figure 4.6, an overview of the control input solution $\boldsymbol{u_{cmd}}$, generated by the optimization process detailed in Equation 4.1, is presented. This overview emphasizes variations across different control objective increments $\boldsymbol{\nu} - \dot{\boldsymbol{y}}_{\boldsymbol{0}}$, desired control input arrays $\boldsymbol{u_d}$, and airspeed. Simulations 1 and 2 provide insight into how the controller processes the desired control input array $\boldsymbol{u_d}$. Simulations 3 and 4 reveal the controller's strategy shift from utilizing tilt and motor power to leveraging lift for the generation of control objective acceleration increments at different airspeed. Lastly, simulations 5 and 6 offer a perspective on how the optimization algorithm integrates the use of motor tilt and motor power to create pitch moments under varying flight conditions.

In Simulation 1, which operates at zero airspeed and starts from a hovering and leveled condition, a desired pitch and roll angle of 25° results in an attitude control input command of 25° for both pitch and roll. The associated tilt and actuator commands are synchronized with this attitude command, ensuring a neutral linear acceleration outcome. In Simulation 2, conducted at high airspeed and also starting from a hovering condition with θ_d and ϕ_d set to 25°, the optimizer outputs a neutral roll command and a pitch command of 3.3°. This occurs due to the impracticality of attaining the desired attitude while also achieving neutral linear accelerations at high airspeed. Consequently, the optimizer prioritizes the acceleration increments in the primary objective of the cost function and disregards the desired attitude inputs, which belong to the secondary objective. The presence of a nonzero pitch command and the rotor commands indicates the controller's intention to leverage wing lift to reduce motor power.

In Simulation 3, carried out at zero airspeed, a linear sideways and vertical acceleration increment is directly achieved through a combination of motor tilt and motor power. In Simulation 4, conducted at an airspeed of 10 m/s, the same control objective in terms of linear accelerations is achieved through executing a pitch-up and roll-right command, with the tilt commands fully forward and a significant reduction in motor power.

Simulation 5 shows what the control input solution looks like to achieve a positive pitch acceleration at zero airspeed. In this scenario, the optimizer solely utilizes motor power differential, commanding greater motor power to the front motors and reducing power to the rear ones. In Simulation 6, where the vehicle is initialized in a fully forward configuration, achieving the same control objective in terms of angular acceleration results in minimal motor power change but significant tilt deflection of the rotors.

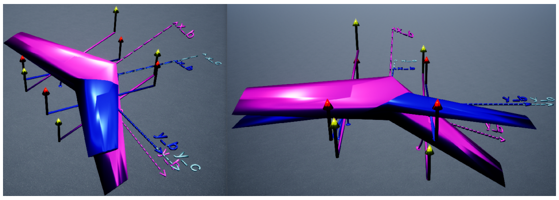
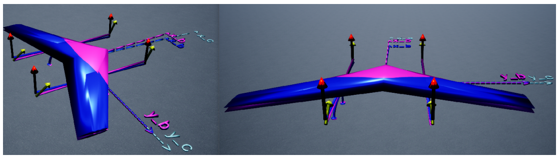
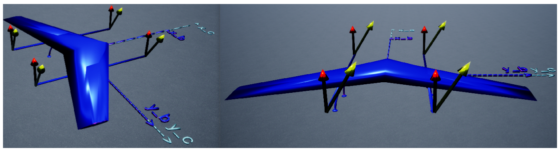
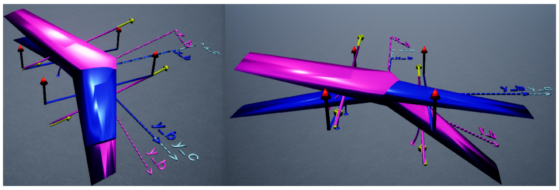
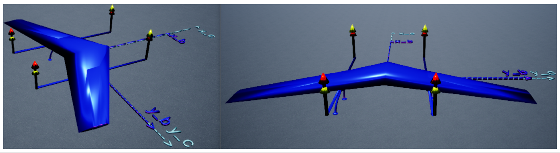
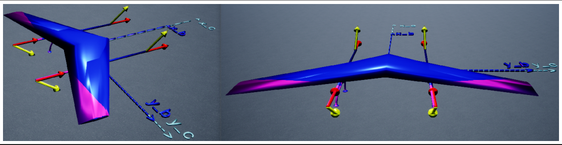
Simulation visualization	Simulation detail
	**Simulation number 1** Control input initial condition: u_0 = [600 600 600 600 ; 0 0 0 0 ; 0 0 0 0 ; 0 0 ; 0] Desired control input array: u_d = [150 150 150 150 ; 0° 0° 0° 0° ; 0 0 0 0 ; 25° 25° ; 0] Control objective acceleration increment: Δv = [0 0 0 ; 0 0 0] Airspeed: V_a = 0 m/s Computed control input command: u_c = [600 600 600 600; -25° -25° -25° -25°; -25° -25° -25° -25°; 25° 25°; 0°]
	**Simulation number 2** u_0 = [600 600 600 600 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 0° 0° ; 0°] u_d = [150 150 150 150 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 25° 25° ; 0°] Δv = [0 0 0 ; 0 0 0] V_a = 10 m/s u_c = [150 150 150 150; -30° -30° -56° -56°; 0° 0° 0° 0°; 3.3° 0°; 0°]
	**Simulation number 3** u_0 = [600 600 600 600 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 0° 0° ; 0°] u_d = [150 150 150 150 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 0° 0° ; 0°] Δv = [0 5 -5 ; 0 0 0] V_a = 0 m/s u_c = [1042 1042 1042 1042; 0° 0° 0° 0° ; 31° 31° 31° 31°; 0° 0°; 0°]
	**Simulation number 4** u_0 = [600 600 600 600 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 0° 0° ; 0°] u_d = [150 150 150 150 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 0° 0° ; 0°] Δv = [0 5 -5 ; 0 0 0] V_a = 10 m/s u_c = [378 378 383 383; -87° -87° -98° -98°; 0° 0° 0° 0°; 13.2° 30°; 0°]
	**Simulation number 5** u_0 = [600 600 600 600 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 0° 0° ; 0°] u_d = [150 150 150 150 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 0° 0° ; 0°] Δv = [0 0 0 ; 0 10 0] V_a = 0 m/s u_c = [737 737 408 408; 0° 0° 0° 0°; 0° 0° 0° 0°; 0° 0°; 0°]
	**Simulation number 6** u_0 = [600 600 600 600 ; -90° -90° -90° -90° ; 0° 0° 0° 0° ; 0° 0° ; 0°] u_d = [150 150 150 150 ; 0° 0° 0° 0° ; 0° 0° 0° 0° ; 0° 0° ; 0°] Δv = [0 0 0 ; 0 10 0] V_a = 10 m/s u_c = [665 665 646 646; -51° -51° -128° -128°; 0° 0° 0° 0°; 0° 0°; 0°]

Figure 4.6: Simulation results for varying control objective increments, desired control input arrays, and airspeed. The synthetic visualization features a vehicle depicted in blue with red arrows indicating the current attitude and actuator states, and another in magenta with yellow arrows showing the attitude and actuator commands calculated by the nonlinear controller. In the simulation details column, u_0 represents the current control input array $\boldsymbol{u_0}$, u_d denotes the desired control input array $\boldsymbol{u_d}$, and u_c signifies the control input command array $\boldsymbol{u_{cmd}}$ resulting from the optimization process described in Equation 4.1. The elements within the control input array are organized as outlined in Equations 4.6 and 4.11. Regarding the control objective acceleration increments Δv, they correspond to the array $\dot{\boldsymbol{y}}_{\boldsymbol{0}} - \boldsymbol{\nu}$ in Equation **??**, with the acceleration order mirroring that in Equation **??**.

4.5.2. Local minima analysis

Due to the algorithm used for solving the optimization problem described in Equation 4.1, combined with the uncertain global convexity of the cost function, the nonlinear controller may produce sub-optimal results. The complexity of the cost function makes it impossible to verify its convexity analytically. To identify local minima, a statistical analysis approach was therefore adopted. Initially, 2000 pairs of reference states, current states, desired control input and wind condition were generated using a uniform statistical distribution to simulate the entire flight envelope, with the range of these values documented in Table 4.6. For each pair, 1000 feasible control input states, labeled to as u_{rand}, along with one additional control input state representing the current actuator state, u_0, were created. These feasible control input states are arrays whose values lie within the maximum and minimum values defined in Section 4.4.4, and were uniformly generated to cover all possible operational scenarios.

For the detection of local minima, for each of the 2000 generated reference state, the final cost function values obtained using the current control input state u_0 as the starting point of the optimizer were compared against those obtained using the 1000 different u_{rand} states. The optimizer was configured to allow up to 10,000 iterations to prevent premature termination. Ideally, in the absence of local minima, the final cost function value should not depend on the optimizer's initial control input condition, u_{init}. Figure 4.7 displays the analysis results, where red dots indicate the error between the cost function value obtained with $u_{init} = u_0$ and the lowest value achieved using $u_{init} = u_{rand}$. A threshold of 1e-4 for the minimum cost error was established to identify points as local minima.

Out of the 2000 evaluated instances, 161 were identified as having local minima. To further evaluate the impact of these local minima on vehicle control, these instances were isolated and simulated for 0.5 seconds to check if the local minima would persist. Remarkably, after just 0.5 seconds of simulation, all previously identified local minima were resolved, with the minimum cost errors dipping below the predetermined threshold. These instances are denoted in blue in Figure 4.7. This observed behavior suggests that the optimizer may initially settle on a sub-optimal solution due to specific combinations of reference and current vehicle states. However, as the vehicle dynamics evolve and the error controller adjusts the control objectives, these initially detected local minima are effectively mitigated over time, resulting in the resolution of such conditions.

variable	max value	min value	std	mean
$\dot{x}_{ref}$	15.0 m/s	-4.0 m/s	5.5 m/s	5.4 m/s
$\dot{y}_{ref}$	8.0 m/s	-8.0 m/s	4.6 m/s	-0.1 m/s
$\dot{z}_{ref}$	3.0 m/s	-3.0 m/s	1.7 m/s	0.0 m/s
$\dot{x}_0$	19.1 m/s	-3.7 m/s	4.7 m/s	7.2 m/s
$\dot{y}_0$	5.0 m/s	-5.0 m/s	2.0 m/s	0.0 m/s
$\dot{z}_0$	0.5 m/s	-0.5 m/s	0.3 m/s	0.0 m/s
$W^c_{x_0}$	4.9 m/s	-5.0 m/s	2.0 m/s	0.0 m/s
$W^c_{y_0}$	5.0 m/s	-5.0 m/s	2.0 m/s	0.0 m/s
$W^c_{z_0}$	0.5 m/s	-0.5 m/s	0.3 m/s	0.0 m/s
p_0	2.0 rad/s	-2.0 rad/s	1.1 rad/s	0.0 rad/s
q_0	2.0 rad/s	-2.0 rad/s	1.2 rad/s	0.0 rad/s
r_0	2.0 rad/s	-2.0 rad/s	1.2 rad/s	0.0 rad/s
θ_d	15.0 deg	-15.0 deg	8.7 deg	0.0 deg
ϕ_d	15.0 deg	-15.0 deg	8.6 deg	0.0 deg

Table 4.6: Maximum, minimum, standard deviation and mean of the 2000 realizations of current states, reference states, and desired control input used for the local minima analysis. with W^c we refer to the components of the wind in the control reference frame. All the current and reference states not mentioned in the table are initialized to zero.

4.6. FLIGHT TEST EXPERIMENT AND RESULTS

4.6.1. EXPERIMENTAL SETUP

The dual-axis tilting rotor vehicle used in the experiments has the physical characteristics outlined in Table 4.7.

For state estimation, reference generation, and ground station communication, we employed the open-source Paparazzi UAV [3] autopilot, running on the Primary Flight Computer Pixhawk 4. The flight code is available on GitHub [4].

4.6.2. FLIGHT TEST DESIGN

To assess the capabilities of the presented Unified Incremental Nonlinear Controller, two distinct autonomous maneuvers were designed and

[3]https://wiki.paparazziuav.org/
[4]https://github.com/tudelft/paparazzi/tree/dual_axis_tr_quadplane_TRO

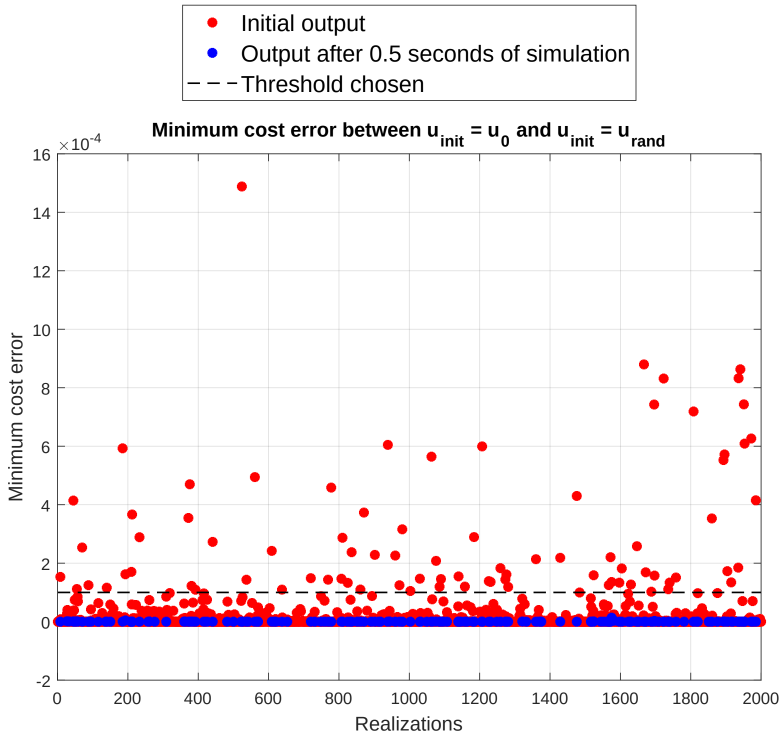


Figure 4.7: Outcome of the statistical analysis on the cost function local minima. Red dots indicate the error between the cost function value obtained with $u_{init} = u_0$ and the lowest value achieved using $u_{init} = u_{rand}$. Instances where the minimal cost error exceeds the threshold dotted black line are simulated for 0.5 seconds and reassessed, yielding the blue dots.

executed aboard the vehicle.

The first maneuver involved accelerating to 15 m/s, followed by a climb at a constant vertical speed of 4 m/s for 5 seconds. Subsequently, a right-hand turn was executed targeting a lateral acceleration of 4 $\frac{m}{s^2}$. Finally, after completing the turn, the vehicle was instructed to level off and decelerate back to a hover state. In the second maneuver, we replicated the first maneuver while keeping a desired pitch angle of $\theta_d = 25°$ throughout the entire duration of the maneuver. This particular maneuver was designed to evaluate the controller's response when a desired vehicle attitude is demanded during a flight phase where the vehicle lacks full 6-DOF control capabilities, such as during fast forward flight. In such scenarios, the controller is expected to prioritize pitch control actions to achieve vertical acceleration, effectively overlooking

I_{xx}	$0.156 \quad Kg \cdot m^2$
I_{yy}	$0.161 \quad Kg \cdot m^2$
I_{zz}	$0.259 \quad Kg \cdot m^2$
$Mass$	$2.44 \quad Kg$
S	$0.43 \quad m^2$
$\bar{c}$	$0.3 \ m$
b	$1.4 \ m$
l_1	$0.228 \ m$
l_2	$0.228 \ m$
l_3	$0.38 \ m$
l_4	$0.38 \ m$
l_z	$0 \ m$

Table 4.7: Vehicle physical characteristics. The inertia was estimated through a meticulous cad modeling process that encompasses each hardware component of the vehicle. For the definition of the rotor displacement l_i please refer to Figure 4.4.

the desired pitch angle.

The hovering capabilities of the vehicle were not intentionally tested, as they were extensively examined in our previous work [8].

4.6.3. FLIGHT TEST RESULTS

After analyzing the flight test results of the two maneuvers reported from Figure 4.8 to Figure 4.14, several observations can be made:

As evident from Figure 4.8 and 4.11, it is clear that all reference speeds, lateral acceleration, attitude angles, and yaw rates are well tracked. Minimal error is observed between the reference and current states throughout both the first and second maneuvers.

The Control Allocation problem is consistently solved correctly, as evidenced by the computed actuator solution in Figure 4.9 and Figure 4.12. The actuator solution remains smooth throughout the maneuvers for all actuators (motors, tilt, and ailerons) with no noticeable oscillations or abrupt changes. Additionally, the controller's adaptive use of various actuators to implement attitude commands across different airspeeds is clearly visible. At low airspeed, pitch and roll actions are primarily controlled through motor thrust commands, while yaw rate is controlled through tilting action. In fast forward flight, the pitch angle is controlled through differential elevation tilt commands, and roll actions are controlled by the ailerons. As for the yaw rate, in forward flight, it is controlled with differential motor thrust commands.

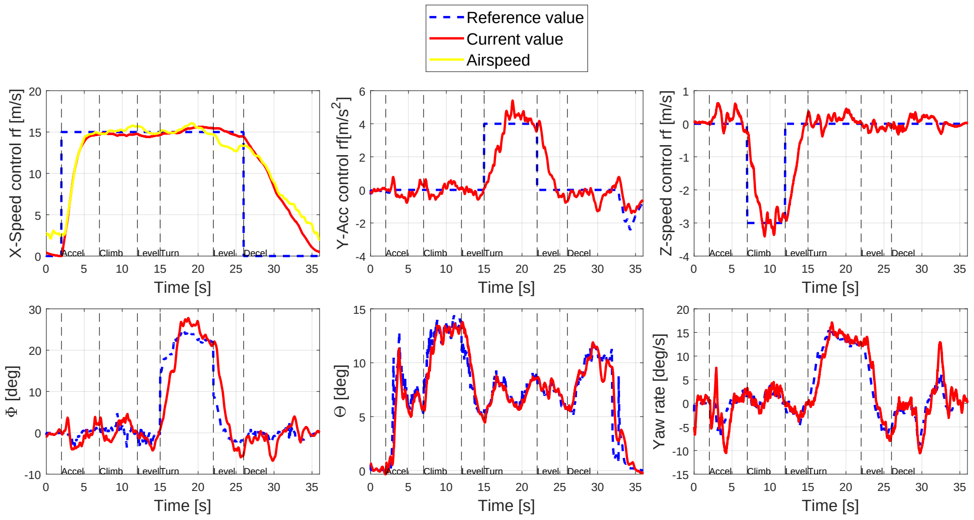


Figure 4.8: Evolution of vehicle states and reference state values for the first maneuver. At the top of the figure, you can see the forward speed, lateral acceleration, and vertical acceleration over time. In the lower section, the evolution of roll, pitch, and yaw rate is displayed. The vertical dotted line indicates the vehicle's actions throughout the maneuver, while the yellow line represents the airspeed pitot tube reading.

As demonstrated in Figure 4.10 and Figure 4.13, the AoA protection algorithm consistently maintains the AoA within the range of ± 15 degrees when the airspeed exceeds 6 m/s.

Thanks to the coordinated turn block, which generates the yaw rate reference as expressed in equation 4.7, the turn maneuvers are effectively executed, and the sideslip angle is consistently minimized. The sideslip angle remains within 10 degrees during the forward flight phase of both maneuvers, as evident from Figure 4.10 and Figure 4.13.

The transition from hovering to forward flight is effectively managed. During transition, the vertical speed is initially controlled by motor thrust and then, as airspeed increases, by pitch actions.

In the second maneuver, the desired pitch angle of 25 degrees is appropriately applied during hovering and low airspeed flight phases, allowing for full 6 DOF control. In fast forward mode, where the vehicle has only 4 DOF, the desired pitch angle is smoothly discarded in favor of AoA protection and lift generation.

The run-time for the optimizer solving the Control Allocation problem consistently remains below or equal to 5 milliseconds, as shown in Figure 4.14, which displays the optimizer performance during the second maneuver. The controller maintains an average update rate of 224 Hz

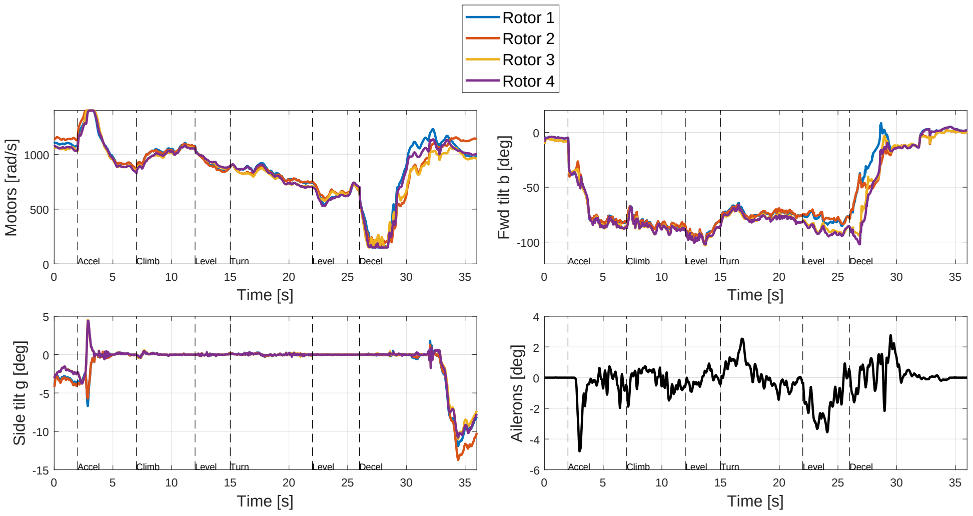


Figure 4.9: Estimated actuator evolution during the first maneuver is depicted. The vertical dotted line marks the vehicle's actions throughout the maneuver. In the top-left plot, the rotational speeds of each motor are illustrated; in the top-right plot, the evolution of the longitudinal tilt angle b for all rotors is displayed. The bottom-left plot shows the evolution of the lateral tilting angle g for all four rotors. The bottom-right plot details the evolution of the ailerons' deflection. For definitions of the tilting angles and rotor numbering, the reader is directed to Figure 4.2 and 4.4, respectively.

in the first maneuver and 222 Hz in the second one, ensuring minimal delay in the control loop.

Lateral tilting commands are not used in fast forward flight to achieve lateral acceleration, in favor of roll angle commands.

For interested readers, a video of the experiment, which includes both the first and second maneuver as well as a detailed close-up of the tilting mechanism, is available at the following link: https://youtu.be/WAbYVZCk_FY.

4.7. LIMITATIONS OF THE UNIFIED INCREMENTAL NONLINEAR CONTROLLER

Even if the presented Unified Incremental Nonlinear Controller was capable of smoothly controlling the dual-axis tilting rotor quad-plane, some limitations can be still identified. Those limitations are still currently a source of research and are the following:

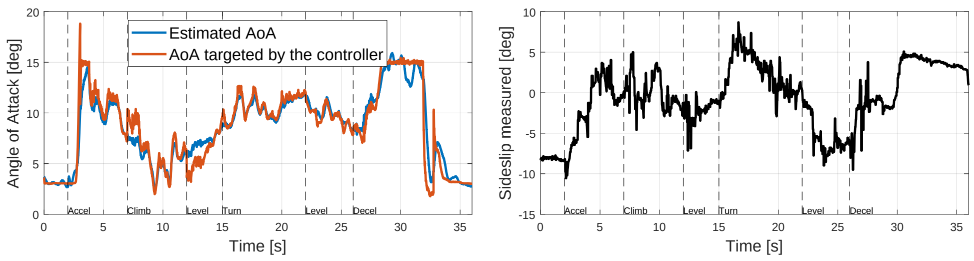


Figure 4.10: On the right side of the plot, the sideslip angle evolution measured using the sideslip vane during the first maneuver is shown. On the left side of the plot, the AoA evolution for the first maneuver is presented. In the AoA plot, the red curve represents the AoA targeted by the nonlinear control allocation block through the theta command θ_{cmd}, while the blue curve illustrates the evolution of the AoA, estimated as presented in equation 4.14. Notably, the AoA never exceeds the maximum value of 15 deg, adhering to the imposed limit. Additionally, the sideslip angle is effectively minimized during the high airspeed phases of the flight.

Limited Actuator Prioritization: The current cost function definition in Equation 4.1 incorporates two weighting matrices, $\boldsymbol{W}_u$ and $\boldsymbol{W}_\nu$, designed to either penalize or prioritize specific actuator usage and acceleration components in the control objective array. Nevertheless, the current configuration lacks the capacity to prioritize the utilization of a particular actuator for achieving a specific component of the control objective $\boldsymbol{\nu}$. In other words, it is currently not feasible to assign, for example, exclusive priority to the back rotors elevation tilt for achieving the roll rate while primarily utilizing the front rotors elevation tilt for pitch rate control. Instead, the system allows only for prioritizing a specific set of rotors elevation tilts for the allocation of all desired accelerations. This limitation became relevant as we noted that the impact of the back tilting rotors on the vehicle's aerodynamics was relatively minor. At high airspeed, it would be more effective to allocate the roll rate over the back rotor's elevation tilt, while maintaining the use of the front rotor's elevation tilt for pitch rate control. However, this is not currently possible within the existing configuration. Consequently, we opted to incorporate ailerons on the vehicle to enable proper and efficient turns in high airspeed conditions.

Inability to Account for Different Actuator Bandwidths: The Control Allocation algorithm in Equation 4.1 assumes uniform bandwidth for all actuators to determine the actuator solution. However, in our system, various actuators possess different bandwidths, as evident in Table 4.3. Moreover, in addition to the physical actuators, we also

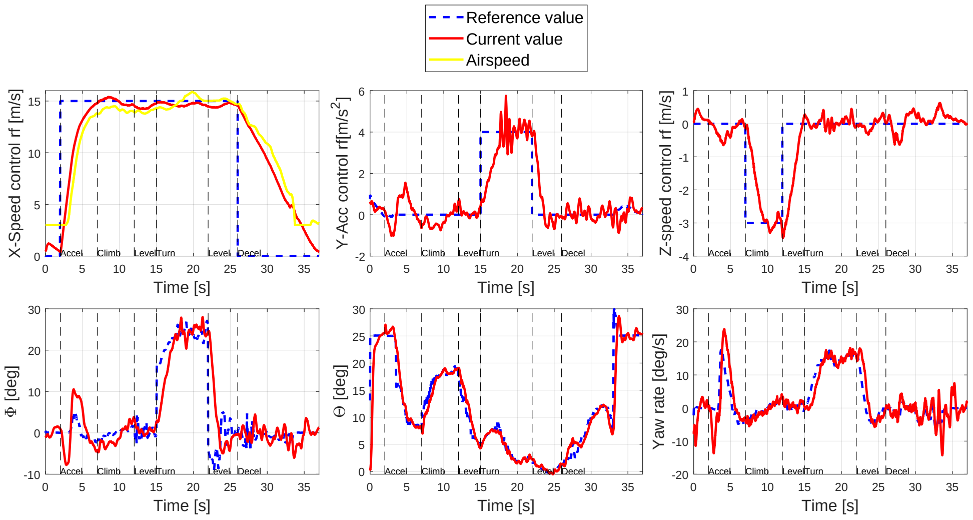


Figure 4.11: Evolution of vehicle states and reference state values during the second maneuver. At the top of the figure, the forward speed, lateral acceleration, and vertical acceleration over time are shown. In the lower section, the evolution of roll, pitch, and yaw rates is presented. The vertical dotted line indicates the vehicle's actions throughout the maneuver, while the yellow line represents the airspeed pitot tube reading. As evident from the pitch plot in the bottom center portion of the figure, the desired pitch angle, input throughout the entire flight maneuver, is effectively commanded and achieved only at the beginning and the end of the maneuver. This is when the airspeed is relatively low, allowing the vehicle to retain effective 6 DOF control.

produce commands for the roll and pitch angles, which have significantly slower dynamics compared to the physical actuators. Consequently, some actuator commands reach their desired values faster than others, resulting in transient undesired acceleration components that require correction. While tweaking the error controller gains or the elements of the control input weighting matrix $\boldsymbol{W_u}$, offers a way to lessen this dynamics mismatch, it is essential to explore more effective methods to account for the physical properties of each actuator.

4.8. CONCLUSION

Even with an imperfect aerodynamic and actuator model, the ability to effectively and accurately control the vehicle has been demonstrated.

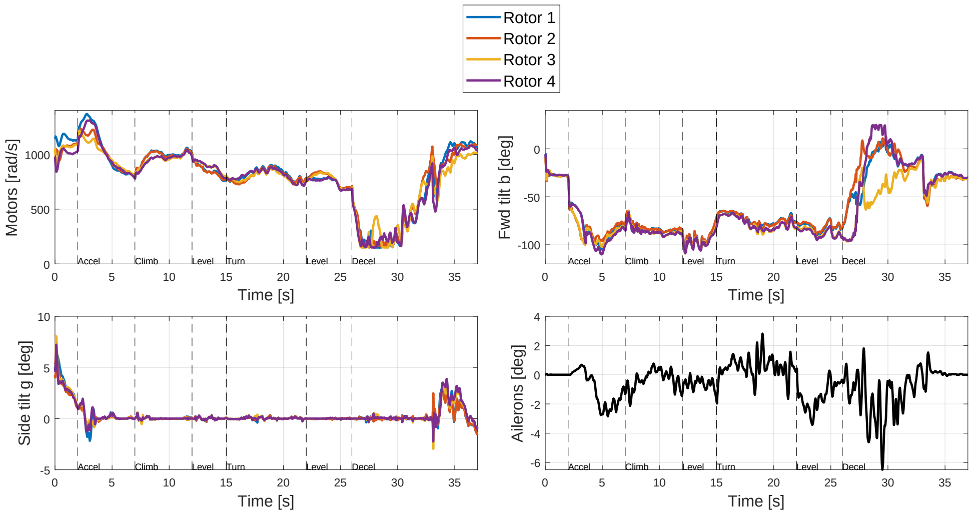


Figure 4.12: Estimated actuator evolution during the second maneuver is depicted. The vertical dotted line marks the vehicle's actions throughout the maneuver. In the top-left plot, the rotational speeds of each motor are illustrated; in the top-right plot, the evolution of the longitudinal tilt angle b for all rotors is displayed. The bottom-left plot shows the evolution of the lateral tilting angle g for all four rotors. The bottom-right plot details the evolution of the ailerons' deflection. For definitions of the tilting angles and rotor numbering, the reader is directed to Figure 4.2 and 4.4, respectively.

This achievement can be largely attributed to the incremental nature of the control laws, which can readily compensate for any offsets in the model parameters and external disturbances.

The AoA protection algorithm played a crucial role in safeguarding the vehicle's pitch angle, minimizing the risk of wing stall, especially during transition phases. The vehicle's transition capabilities, coupled with the ability to produce lift and thrust simultaneously, have led to improved performance and minimal altitude loss during these critical flight phases.

Pitch and roll angles were effectively utilized during fast forward flight to control vertical and lateral acceleration. Moreover, the controller adeptly discarded desired attitude commands when transitioning between hovering and high-speed configurations, where only 4 DOF are effectively controlled. Furthermore, the yaw rate commands generated by the coordinated turn block proved effective in managing turns while minimizing sideslip.

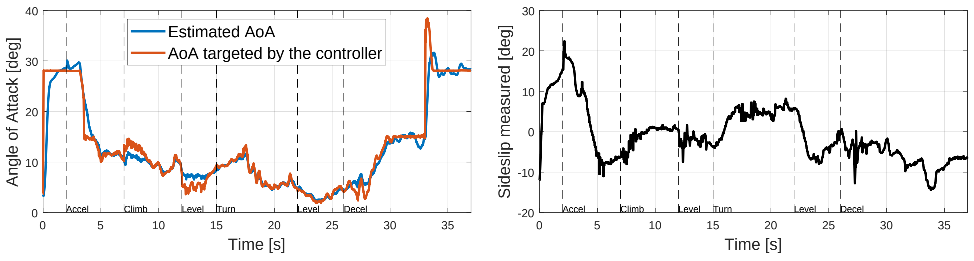


Figure 4.13: On the right side of the plot, the sideslip angle evolution, measured using the sideslip vane during the second maneuver, is displayed. On the left side of the plot, the AoA evolution for the second maneuver is shown. In the AoA plot, the red curve shows the AoA targeted by the nonlinear control allocation block via the theta command, while the blue curve depicts the evolution of the estimated AoA, as presented in equation 4.14. Unlike the first maneuver, where the AoA was consistently maintained below 15 degrees, in this maneuver, both the targeted and current AoA exceed the maximum limit of 15 degrees. This occurrence is noted during the low-speed flight phases, where the AoA protection logic is deactivated, and the vehicle successfully targets and achieves the desired theta angle in $\boldsymbol{u_d}$. Similar to the previous maneuver, the sideslip is effectively minimized during the high airspeed phase of the flight.

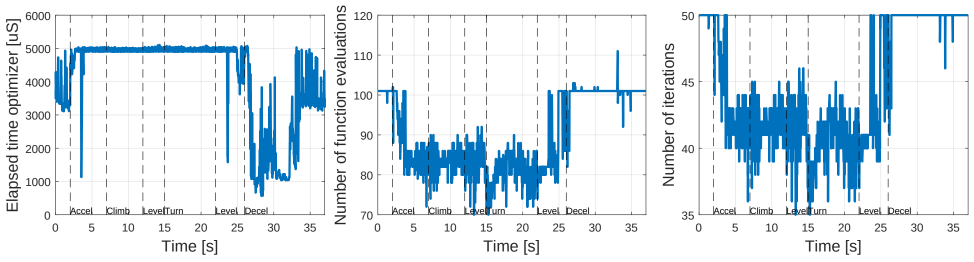


Figure 4.14: Run-time, function evaluations, and the number of iterations required by the Unified Nonlinear Control Allocation algorithm running on the Raspberry pi 4, to compute the actuator solution during the second maneuver.

A detailed statistical analysis of the cost function's shape across the entire flight envelope of the vehicle identified a few local minima. Fortunately, the presence of local minima in the cost function does not pose any significant issues for vehicle control. We demonstrated that these local minima do not persist over time and are automatically resolved within 0.5 seconds, thanks to the error controller and the vehicle dynamics.

In future work, we aim to enhance vehicle performance through a system identification models that consider propeller-inflow angle and the interactions between the wing and rotor-induced inflow. This will possibly enable the vehicle to achieve roll rate control in fast forward flight solely through tilting commands, potentially eliminating the need for ailerons.

Furthermore, we are currently investigating a method to incorporate and account for different actuator bandwidths within the nonlinear controller framework.

REFERENCES

[1] E. Smeur, M. Bronz, and G. D. Croon. "Incremental Control and Guidance of Hybrid Aircraft Applied to a Tailsitter Unmanned Air Vehicle." In: *Journal of Guidance Control and Dynamics* 42 N.2 (2020). doi: 10.2514/1.G004520.

[2] X. Wang and S. Sun. "Incremental fault-tolerant control for a hybrid quad-plane UAV subjected to a complete rotor loss." In: *Journal of Aerospace Science and Technology* 125 (2022). doi: 10.1016/j.ast.2021.107105.

[3] K. Masuda and K. Uchiyama. "Robust Control Design for Quad Tilt-Wing UAV." In: *MDPI* (2018). doi: 10.3390/aerospace5010017.

[4] G. Flores and R. Lozano. "Transition flight control of the quad-tilting rotor convertible MAV." In: *International Conference on Unmanned Aircraft Systems (ICUAS)* (2013). doi: 10.1109/ICUAS.2013.6564761.

[5] A. Mancinelli, E. Smeur, B. Remes, and G. Croon. "Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design." In: *International Conference on Unmanned Aircraft Systems (ICUAS)* (2022). doi: 10.1109/ICUAS54217.2022.9836063.

[6] C. Papachristos, K. Alexis, and A. Tzes. "Technical Activities Execution with a TiltRotor UAS employing Explicit Model Predictive Control." In: *IFAC Proceedings* 47 Issue 3 (2014). doi: 10.3182/20140824-6-ZA-1003.02692.

[7] M. Allenspach, K. Bodie, M. Brunner, and al. "Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight." In: *International Journal of Robotics Research* (2020). doi: 10.1177/0278364920943654.

[8] A. Mancinelli, B. D. W. Remes, G. C. H. E. De Croon, and E. J. J. Smeur. "Real-Time Nonlinear Control Allocation Framework for Vehicles with Highly Nonlinear Effectors Subject to Saturation". In: *Journal of Intelligent & Robotic Systems* 108 (July 2023). doi: 10.1007/s10846-023-01865-8.

[9] C. Papachristos, K. Alexis, and A. Tzes. "Model predictive hovering-translation control of an unmanned Tri-TiltRotor." In: *International Conference on Robotics and Automation* (2013). doi: 10.1109/ICRA.2013.6631355.

[10] Y. Zelong, J. Zhang, and X. Wang. "Thrust Vectoring Control of a Novel Tilt-Rotor UAV Based on Backstepping Sliding Model Method." In: *MDPI sensors* (2023). doi: 10.3390/s23020574.

[11] E. Tal and S. Karaman. "Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness." In: *IEEE Transactions on Control Systems Technology* (2021). doi: 10.1109/TCST.2020.3001117.

[12] E. Tal and S. Karaman. "Global Incremental Flight Control for Agile Maneuvering of a Tailsitter Flying Wing." In: *Journal of Guidance Control and Dynamics* (2022). doi: 10.2514/1.G006645.

[13] M. Faessler, A. Franchi, and D. Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories." In: *IEEE Robotics and Automation Letters* (2018). doi: 10.1109/LRA.2017.2776353.

[14] G. D. Francesco and M. Mattei. "Modeling and Incremental Nonlinear Dynamic Inversion Control of a Novel Unmanned Tiltrotor." In: *Journal of Airfraft* (2015). doi: 10.2514/1.C033183.

[15] H. Wang, S. Wenhao, Z. Changli, Z. Sujie, and H. Jianda. "Dynamic Modeling and Control for Tilt-Rotor UAV Based on 3D Flow Field Transient CFD." In: *MDPI Drones* (2022). doi: 10.3390/drones6110338.

[16] X. Wang and L. Cai. "Mathematical modeling and control of a tilt-rotor aircraft." In: *Aerospace Science and Technology* 47 (2015). doi: 10.1016/j.ast.2015.10.012.

[17] M. Sato and K. Muraoka. "Flight Controller Design and Demonstration of Quad-Tilt-Wing Unmanned Aerial Vehicle." In: *Journal of Guidance Control and Dynamics* 34 (2015). doi: 10.2514/1.G000263.

[18] D. Rohr, T. Stastny, S. Verling, and R. Siegwart. "Attitude and Cruise Control of a VTOL Tiltwing UAV." In: *IEEE Robotics and Automation Letters* (2019). doi: 10.1109/LRA.2019.2914340.

[19] K. Oner, E. Cetinsoy, M. Unel, M. Aksit, I. Kandemir, and K. Gulez. "Dynamic Model and Control of a New Quadrotor Unmanned Aerial Vehicle with Tilt-Wing Mechanism." In: *International Journal of Aerospace and Mechanical Engineering* 9 No. 9 (2008). doi: 10.5281/zenodo.1074431.

[20] Z. Chen and H. Jia. “Design of Flight Control System for a Novel Tilt-Rotor UAV.” In: *Complexity* (2020). doi: 10.1155/2020/4757381.

[21] E. Cetinsoy, S. Dikyar, C. Hancer, K. Oner, E. Sirimoglu, M. Unel, and M. Aksit. “Design and construction of a novel quad tilt-wing UAV”. In: *Mechatronics* 22.6 (2012). Special Issue on Intelligent Mechatronics, pp. 723–745. issn: 0957-4158. doi: 10.1016/j.mechatronics.2012.03.003.

[22] K. Masuda and K. Uchiyama. “Flight Controller Design Using mu-synthesis for Quad Tilt-Wing UAV.” In: *AIAA Scitech 2019 Forum* (2019). doi: 10.2514/6.2019-1918.

[23] J. J. D. et al. “H-infinity Hover-to-Cruise Conversion for a Tilt-Wing Rotorcraft.” In: *Proceedings of the 44th IEEE Conference on Decision and Control* (2005). doi: 10.1109/CDC.2005.1583202.

[24] S. Raab, J. Zhang, P. Bhardwaj, and F. Holzapfel. “Proposal of a Unified Control Strategy for Vertical Take-off and Landing Transition Aircraft Configurations.” In: *Applied Aerodynamics Conference* (2018). doi: 10.2514/6.2018-3478.

[25] J. Lv, J. Li, and D. An. “Coordinated-turn control law design of aircraft based on optimal tracking.” In: *International Conference on Automatic Control and Artificial Intelligence* (2012). doi: 10.1049/cp.2012.1081.

[26] T. Pollack and E. V. Kampen. “Robust Stability and Performance Analysis of Incremental Dynamic Inversion-based Flight Control Laws.” In: *AIAA Scitech 2022 Forum* (2022). doi: 10.2514/6.2022-1395.

[27] K. Schittkowski. “NLQPL: A FORTRAN-Subroutine Solving Constrained Nonlinear Programming Problems.” In: *Annals of Operations Research* 5 (1985).

5 AUTOPILOT FRAMEWORK WITH INDI RPM CONTROL AND REAL-TIME ACTUATOR FEEDBACK

In Chapter 4, a unified control framework was developed to enable smooth and robust control of the dual-axis tilting rotor quad-plane across its entire flight envelope, from hovering to cruise flight. While the controller demonstrated effective performance in handling the vehicle's complex dynamics, it relied on predefined models of the actuators, which limited its robustness and adaptability in real-world operations.

This chapter addresses these limitations by introducing a real-time actuator state feedback system, enabling direct control of both motor RPM and angular tilt without relying on an accurate actuator model. This improvement enhances the controller's performance and provides continuous situational awareness of the propulsion system's health.

This chapter is based on the following article:
A. Mancinelli, E. Van Der Horst Bart D.W. Remes and Ewoud J.J. Smeur, "Autopilot framework with INDI RPM control, real-time actuator feedback, and stability control on companion computer through MATLAB generated functions," in IMAV 2024 Conference and Competition, 2024

5.1. INTRODUCTION

UAVs are aircraft designed to operate without a human pilot on board. UAVs have captivated a vast audience and found profound significance through their unmatched versatility, accessibility, and diverse range of applications across industries. From breathtaking aerial photography and aerial manipulation to efficient delivery and logistics.

In recent years, with growing interest from both industry and research communities, the field of UAVs has witnessed rapid advancements. The integration of a large number of actuators in UAVs has increasingly become prevalent, as exemplified by notable studies such as [1–4]. The inclusion of an increasing number of actuators in UAVs serves a crucial role in achieving propulsion redundancy and enhancing overall performance. A prime example illustrating this is the dual-axis tilt rotor quad-plane depicted in Figure 4.1, where a total of 14 actuators, specifically 4 motors and 10 servomotors, are utilized to independently control the vehicle's 6 degrees of freedom.

Managing a significant number of actuators presents a complex challenge for modern commercial flight controllers, such as The Cube[1] , Pixhawk 6c[2] , Pixracer[3] or the Lisa M[4]. These flight controllers offer limited PWM actuator outputs, which are sometimes further restricted due to the shared PWM lines with auxiliary peripherals and sensors. Consequently, the integration of an external subsystem becomes indispensable to effectively handle the growing quantity of actuators in modern UAVs. Moreover, this external flight system must possess the capability to monitor the state of the actuators in real-time. The significance of precise real-time estimation of the actuator state becomes particularly evident in contemporary UAV design, where the INDI control method is widely used[5, 6]. This advanced control method heavily relies on accurate actuator state estimation to ensure optimal performance. Traditionally, given the lack of actuator state feedback, the estimation of the actuator state is obtained by feeding the desired actuator input into an actuator model. The actuator model must be identified through laborious tests, which can introduce inaccuracies and consequently impact the quality of the estimation.

The ability to achieve reliable real-time measurements of the actuator state yields substantial advantages that extend beyond enhanced control performance. It assumes a pivotal role in promptly detecting actuator faults, facilitating the design and implementation of effective fault-tolerant control strategies. By possessing a dependable estimation of the actuator state, UAV systems can swiftly identify deviations or anomalies in actuator behavior, enabling timely interventions.

[1] https://www.cubepilot.com/#/cube/features
[2] https://docs.px4.io/main/en/flight_controller/pixhawk6c.html
[3] https://docs.px4.io/main/en/flight_controller/pixracer.html
[4] https://wiki.paparazziuav.org/wiki/Lisa/M_v2.0

Furthermore, a notable constraint of commercial flight controllers lies in their limited computational power. Typically, these flight controllers utilize ARM Cortex M-family processors, which are specifically optimized for embedded applications. However, when compared to ARM A-family processors found in SBCs like the Raspberry Pi [7], their computational power falls short. Embracing more powerful processors presents a promising opportunity to significantly enhance the computational capabilities of commercial flight controllers, enabling the execution of complex algorithms to support the UAV control and operations.

One potential way to improve the computational power of commercial flight controllers is through the adoption of an external companion computer, as demonstrated by previous works such as [8–10]. In particular, in our previous work [11], we utilized a Raspberry Pi 4B to solve the non-linear control allocation problem for the dual-axis tilting rotor quad-plane. This approach has proven to be highly successful and reliable.

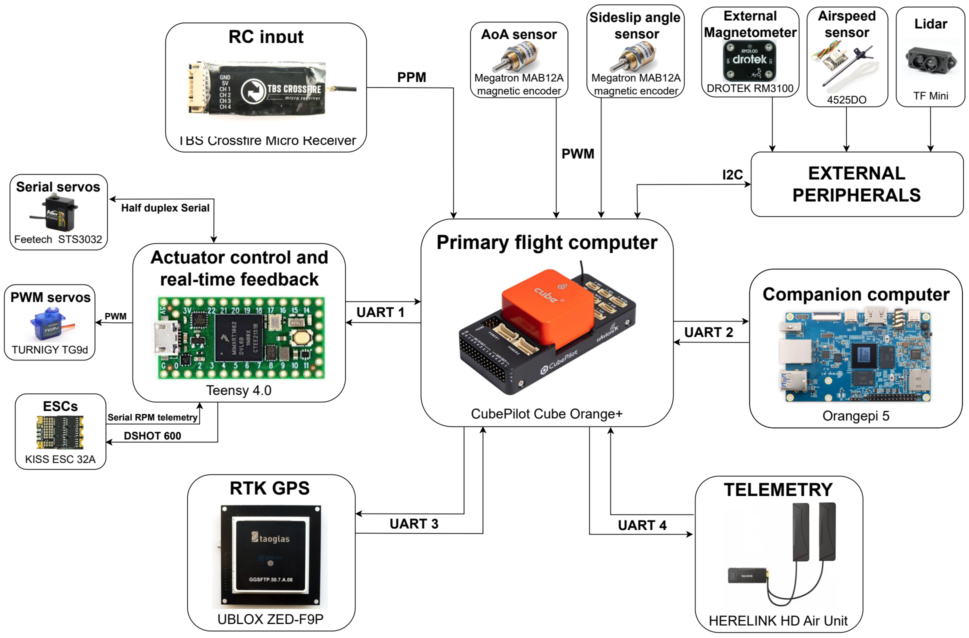


Figure 5.1: Hardware components and their intercommunication.

In this chapter, we present a modular autopilot framework that utilizes separate subsystems to address the aforementioned limitations of modern flight control. The primary flight computer employed is a Cube Orange, which runs the popular Paparazzi UAV autopilot software [12]. The main task of the primary flight computer is to estimate the vehicle states, run the autopilot routines and coordinate all the other sub-modules. As a companion computer sub-module, an OrangePi 5[5] SBC was utilized, featuring a Rockchip RK3588S octa-core 64-bit processor. A notable feature of the OrangePi 5 SBC is its ability to run a Linux operating system, enabling the compilation and execution of functions developed in the MATLAB environment using the MATLAB C-coder toolbox[6]. For the actuator control and real-time feedback subsystem, a Teensy 4.0[7] micro-controller was employed. The primary flight computer communicates with all the subsystems through custom modules developed in Paparazzi UAV , utilizing UART communication.

The chapter is structured as follows: Section 5.2 presents the Modular Autopilot framework, which encompasses the Paparazzi UAV autopilot running on the primary flight computer. It also explores the information flow among all the subsystems and the primary flight computer. Section 5.3 provides a detailed analysis of the actuator control and real-time feedback subsystem, providing insights into its functionality and performance. Within this section, an INDI RPM controller is also presented. Section 5.4 focuses on the companion computer sub-module and explores its setup for running MATLAB functions. Section 5.5 discusses and showcases real flight results obtained from the implemented framework. Finally, Section 5.6 draws conclusions based on the presented findings and discussions.

5.2. AUTOPILOT STRUCTURE

This section presents the Autopilot structure, covering both hardware and software aspects. Special attention is given to the working principles of Paparazzi UAV, the software running on the primary flight computer. The Paparazzi UAV flight software, running on the primary flight computer, is made available to the reader[8].

5.2.1. MAIN AUTOPILOT HARDWARE STRUCTURE

The overall Autopilot structure is illustrated in Figure 5.1. It can be observed that the AP structure consists of several parts. The main part is the Primary flight computer, where the estimation, stability, guidance, and navigation routines are implemented and executed. To achieve accurate estimation of flight states and position, the primary flight computer interfaces with multiple sensors, located both internally within the Orange Cube module and externally.

The internal sensors of the Orange Cube module include an Invensense ICM42688, an Invensense ICM20948, an Invensense MS5611

[5]http://www.orangepi.org/
[6]https://nl.mathworks.com/help/coder/index.html
[7]https://www.pjrc.com/store/teensy40.html
[8]https://github.com/alessandro-mancinelli/Modular_AP_PPZ.git

I2C barometer, and an AK099916 magnetometer. All the internal sensors are mounted on a temperature-controlled, vibration-isolated board.

For state estimation purposes, the primary flight computer utilizes also the following external sensors:

- UBLOX ZED-F9P RTK GPS connected through a Serial connection.
- Angle of Attack sensor employing a Megatron MAB12A magnetic encoder connected through PWM input.
- Sideslip angle sensor employing a Megatron MAB12A magnetic encoder connected through PWM input.
- External Drotek RM3100 magnetometer connected through the I2C bus.
- 4525DO Airspeed sensor employing a differential MS5611 barometer connected through the I2C bus.
- TFMini lidar with a maximum range of 15 meters, connected through the I2C bus.

All the sensor readings, except for the Sideslip and Angle of Attack readings are fused together using an Extended Kalman Filter based on the Estimation and Control Library (ECL) from PX4[9], running on the Paparazzi UAV flight software.

Regarding the Remote Control (RC) communication with the safety pilot of the UAV, an RC input source is established using a TBS Crossfire Micro receiver connected through a PPM signal. Additionally, a Herelink HD Air unit connected through UART is utilized to establish communication with the Ground Control Station.

A serial communication is utilized for the communication between the Primary Flight Computer and both the Actuator control and the real-time feedback subsystem, and the Companion computer subsystem. In our specific case, the companion computer is responsible for solving the control allocation problem in real-time. Since the solution needs to be communicated to all the actuators, this requires a fast and reliable communication channel. To minimize overhead, we implemented a UART communication protocol with one starting byte and one checksum byte. The baud rate chosen for the Serial communication is 1.5 megabaud.

5.2.2. Paparazzi UAV flight software

Paparazzi UAV is a versatile and highly customizable flight software that offers users a wide range of options for configuring their aircraft. At the core of the Paparazzi software is the airframe XML file, which serves as the central configuration hub for selecting and setting up all the

[9] https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html

necessary components and modules required for a successful flight. This modular approach empowers users to customize their aircraft according to specific requirements by choosing the appropriate sensors, actuators, and other peripherals. A comprehensive overview of the information flow in Paparazzi UAV is available on the wiki [10].

One of the key strengths of Paparazzi UAV is its support for custom modules, which facilitate communication and collaboration among different components. These modules can interact with each other using external global variables or, in a more structured manner, through an intermediary routine known as the Application Binary Interface (ABI) routine. This enables seamless integration and coordination between various functionalities, enhancing the overall capabilities and adaptability of the system.

THE ABI PUBLISHER PUBLISH/SUBSCRIBE MIDDLE-WARE

ABI is a custom publish/subscribe middleware that facilitates the exchange of data between software components. Within the Paparazzi UAV software, two different modules can exchange information through ABI. Once the ABI message structure is defined, each module can utilize the ABI routine as either a publisher or a subscriber of information. The role of a publisher is to update the content of the ABI node with new information to be broadcasted in the autopilot. One or more modules can then subscribe to that ABI node. An interesting aspect is that the subscriber node is notified whenever the ABI message contains fresh data and automatically triggers a function to post-process the data. In the design of the Modular Autopilot framework, we extensively utilized the ABI middleware. Specifically, we associated an ABI message with both the Actuator Control and Real-time Feedback subsystem and the Companion Computer subsystem, enabling interaction between our control module and these subsystems.

CUSTOM TEENSY ACTUATORS PAPARAZZI MODULE

The custom module called serial_act_t4, developed within the Paparazzi UAV software, serves as the software back-end running on the Primary Flight Computer. Its purpose is to enable communication between the Actuator Control and Real-time Feedback subsystem running on the Teensy 4.0 and the Primary Flight Computer. The messages exchanged through the Serial UART channel between the Teensy 4.0 and the Primary Flight Computer are defined in the ca_am7.h file, which is located in the "Modular_AP_PPZ/sw/airborne/modules/sensors" path. It should be noted that the "Modular_AP_PPZ" refers to the Paparazzi repository available at the TUDelft github page[8]. In the serial_act_t4.h file, two

[10]https://wiki.paparazziuav.org/wiki/DevGuide/DesignOverview

structures are defined: serial_act_t4_data_in and serial_act_t4_data_out, representing the inbound and outbound messages, respectively. In our case, we have decided to include a cyclically updated message with lower priority in the main message struct. Whenever a message is sent, one element of a float array with a length of 255 is also transmitted. Consequently, the update rate of the float array is approximately 0.4% of the refresh rate of the main message.

This module is associated with a publishable ABI message named "SERIAL_ACT_T4_OUT". Users can publish to this message from any location within the Paparazzi software, allowing the outbound message structure to be sent over the hardware serial interface to the Teensy.

Similarly, an ABI broadcast message called "SERIAL_ACT_T4_IN" is triggered whenever a new inbound message structure is received from the Teensy. This broadcast message can also be accessed from any location within the Paparazzi software.

To gain a comprehensive understanding of how to subscribe to or publish an ABI message, readers are encouraged to consult the Paparazzi UAV ABI documentation[11]

CUSTOM ORANGEPI PPZ MODULE

Similarly to the implementation of the serial_act_t4 module, a custom module called ca_am7 module was developed within the Paparazzi UAV software as a back-end running on the Primary flight computer to enable communication with the OrangePi 5. The inbound and outbound message structures exchanged with the OrangePi are defined in the ca_am7.h file, located in the "Modular_AP_PPZ/sw/airborne/modules/sensors" path. Just like the serial_act_t4 module, we also included a rolling message with lower priority in the main message struct.

The ABI broadcast message containing the inbound data structure from the OrangePi is named "AM7_IN", while the publishable ABI message for the outbound data transmission is named "AM7_OUT".

5.3. ACTUATOR CONTROL AND REAL-TIME FEEDBACK SUBSYSTEM

In this section, we present the code of the Actuator control and real-time feedback subsystem running on the Teensy 4.0. We also provide a brief overview of the protocol used to control both the serial servos, PWM servos, and the ESCs. The source code for generating the binary file for the Teensy 4.0 is available in the Teensy_actuators_PPZ GitHub repository[12]. The software is developed using the Arduino IDE and the

[11]https://paparazzi-uav.readthedocs.io/en/stable/developer_guide/abi.html
[12]https://github.com/tudelft/Teensy_actuators_PPZ.git

Teensyduino add-on[7]. The electrical scheme of the actuator control and real-time feedback subsystem is depicted in Figure 5.2.

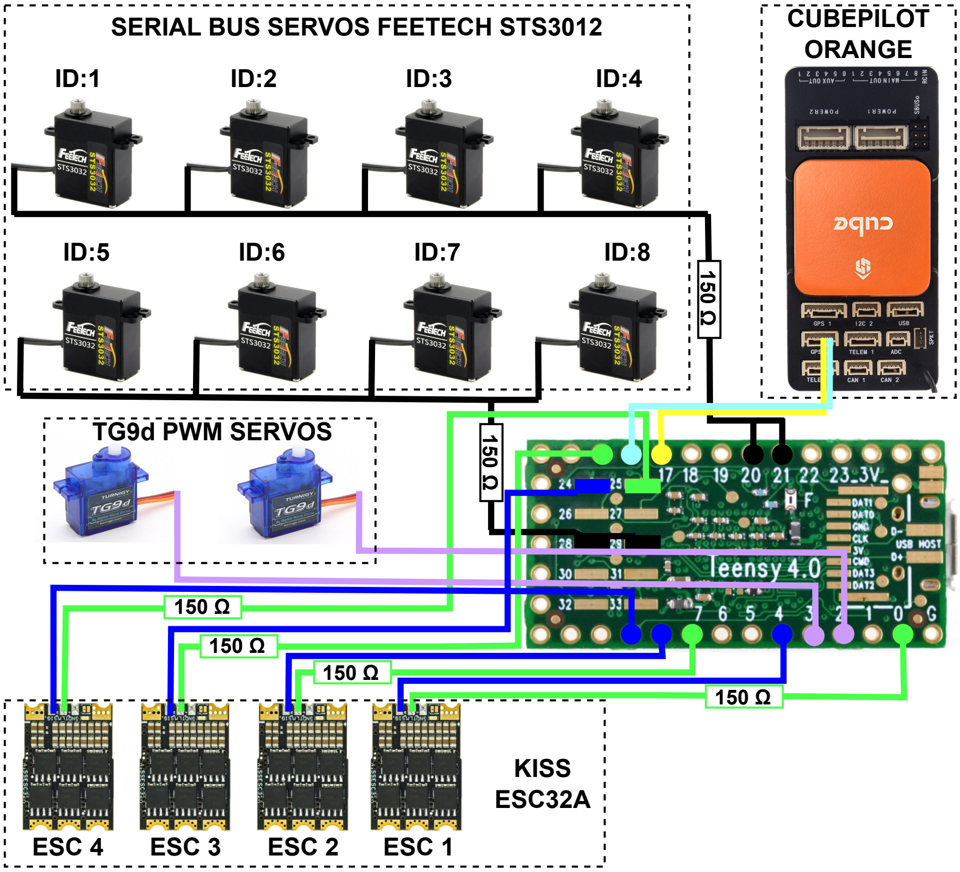


Figure 5.2: Electrical schematic of the actuator control and real-time feedback subsystem. The 150 Ω resistors are utilized to safeguard the bus from potential transmission conflicts.

5.3.1. SERIAL BUS SERVOS CONTROL

In the current configuration, the Teensy 4.0 microcontroller controls a total of 8 Feetech STS3012 serial servos. These 8 servos are controlled using two separate buses, with each bus connected to 4 servos. The protocol used to communicate with the serial servos is a single-wire half-duplex serial protocol. It involves standardized packet transmission between a master device (the Teensy 4.0) and slave devices (the STS3012 servos). An overview of the packet frame is provided in Table

Header	ID number	Data Length	Command	Parameter	Checksum
0x55 0x55	ID	Packet Length	Cmd code	Prm 1... Prm N	Checksum

Table 5.1: Command packet frame for the communication between the Teensy 4.0 and the Feetech STS3012 servos.

5.1.

The master device sends an instruction packet on the bus, addressing a specific servo ID. All devices on the bus receive the packet, but the servos discard it if their ID does not match. Only the servo with the matched ID executes the instruction and then sends a confirmation packet.

The servo ID is a value ranging from 0 to 253, which means that a maximum theoretical number of 254 servos per bus is supported. However, since the bus is shared among all servos and the bus bandwidth is limited to 1 Mega baudrate, increasing the number of servos will limit the refresh rate of the instructions that can be sent to each servo.

Furthermore, it should be noted that if we want to both command a servo position and obtain servo position feedback, two instruction packets have to be sent to the servo. The first instruction contains the desired servo position, to which the servo will respond with an ack packet. Then, we need to send another instruction requesting the servo's current angular position, to which the servo will respond with a packet containing its current position. This can be extended to include additional information such as load, tension, or angular speed. For a detailed explanation of all the possible commands for the serial servos, the reader can refer to the Documentation/Serial_servo_protocol.pdf document in the Teensy_actuators_PPZ GitHub repository[12].

We observed that typically each servo takes 20 to 50 microseconds to respond to a Teensy request. Due to this response time and our requirement for high-speed position control and feedback from the servos, we decided to use two separate buses to control the 8 servos. This approach allowed us to achieve a refresh rate of 350 Hz for both the angular position control and angular position feedback of each servo, while providing enough time to avoid bus conflicts. As an additional safety feature, we also added a 150 Ω resistor to the serial line. This resistor helps limit the maximum current flow on the Teensy serial pin to 22 mA in the event of conflicts on the serial line. Without the resistor, the current flow on the Teensy serial pin in the case of a conflict would exceed the maximum limit of 25 mA tolerated by the microcontroller, potentially resulting in chip damage.

5.3.2. PWM SERVOS CONTROL

On top of controlling Serial bus servos, the Teensy_actuators_PPZ software running on the Teensy 4.0 can also control conventional PWM servos. In our current setup, we have utilized two PWM TGYd micro servos, which are assigned to the ailerons of the airframe. These servos, due to their communication technology, do not feature any feedback regarding their angular position. Therefore, an estimation of the servo position is carried out on the Teensy board. The estimation is based on a first-order transfer function that incorporates the command and the user-provided first-order dynamics characteristics of the servo. These parameters can be identified through a step response test using external equipment, such as an IMU mounted on the servo arm, as demonstrated in [3].

The routine responsible for PWM servo control and estimation is implemented in the *writeEstimatePwmServos()* function, which can be found in the servo_esc_control_w_feedback_T4_PPZ.ino file in the Teensy_actuators_PPZ GitHub repository[12].

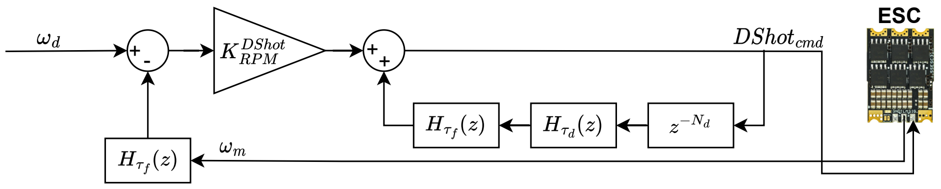


Figure 5.3: Scheme of the RPM INDI controller.

5.3.3. CONTROL OF THE ESCS

The KISS ESC 32A in our system are controlled using Digital Shot (DShot) commands at a speed of 600 bit/s. The DShot protocol has become widely adopted as a digital communication protocol specifically designed for controlling ESCs. It offers several advantages over conventional UART communication.

One notable advantage of the Dshot protocol over the UART protocol lies in how zeros and ones are identified in the communication. In DShot, the differentiation between zeros and ones is based on the duration of the high state voltage, rather than relying on traditional binary representation through voltage levels. This utilization of timing information allows for more precise encoding of data.

The DShot packet frame consists of a total of 16 bits. The initial 11 bits are dedicated to identifying the throttle command. A value of all zeros indicates that the motor is disarmed, while values 1-47 are reserved for special purposes. Consequently, the throttle command can range from 0 to 2000. Following the 11-bit throttle command, one bit is allocated for telemetry request, and an additional 4 bits contain the cyclic redundancy check (CRC) for message integrity.

When the telemetry bit is enabled, the ESC outputs a telemetry packet through a dedicated serial line operating at a baud rate of 115200. The telemetry packet comprises 10 bytes of information, including temperature, voltage, current, and RPM data from the ESC. For a more detailed explanation of the telemetry protocol, the reader can refer to the Documentation/KISS_ESC_telemetry_protocol.pdf file available in the GitHub repository[12].

Due to the limited baud rate of this serial line, we decided to employ a refresh rate of 500 Hz for both DShot throttle commands and telemetry transmission. Keeping within this limit ensures reliable and timely telemetry updates from the ESCs.

MOTOR RPM CONTROL

The high-speed command and telemetry capabilities of the ESCs have allowed for the design and implementation of an RPM controller for the motors. The ability to reliably control the motor's RPM is a crucial feature for modern controllers that rely on model inversion to compute the actuator solution. By directly setting the actuator RPM, the need to accurately model external factors such as battery discharge or propeller inflow is eliminated. This simplifies the control process and enhances the overall performance of the system.

The INDI RPM controller scheme used to control the motor RPM is illustrated in Figure 5.3, where ω_m is the RPM value measured by the ESC and ω_d is the desired RPM. The transfer function $H_\tau(z)$ represents a low-pass filter in the discrete-time domain with a time constant of τ:

$$H_\tau(z) = \frac{\tau}{z-(1-\tau)}, \tag{5.1}$$

where z denotes the discrete-time step delay operator.

Through experimentation, we determined that this filter with a time constant of $\tau_f = 0.1131$ effectively filters out the noise present in the ESC readings. With the ESC telemetry frequency of 500 Hz, this corresponds to a first order low-pass filter with a cutoff frequency of 60 rad/s in the Laplace domain.

Another filter of the form of Equation 5.1 with parameter τ_d represents the first-order dynamics of the motor system, while N_d in Figure 5.3 represents the discrete-time steps of delay introduced by the ESC for processing the DShot command. To determine these parameters, we analyzed the step response for various DShot commands. The analysis

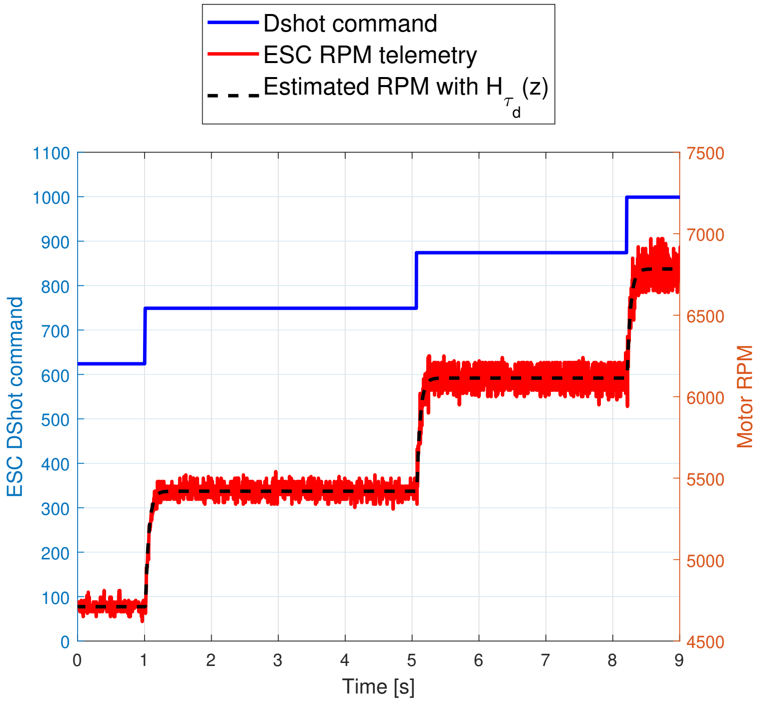


Figure 5.4: Identification of the motor dynamics.

of the system response, as shown in Figure 5.4, yielded an estimated value of 0.039 for τ_d and 3 for N_d. It is important to note that these values are specific to the tested set of propellers and to an RPM control loop update rate of 500 Hz, matching the ESC telemetry refresh rate.

The gain K_{RPM}^{DShot}, on the other hand, represents the inverse of the control effectiveness, in this case the mapping from RPM to DShot commands. We opted for a constant gain of $K_{RPM}^{DShot} = 0.096$, which corresponds to the ratio between the maximum RPM and the maximum DShot command.

To illustrate the performance of the INDI RPM controller, a step response from an initial RPM of 5000 to a desired RPM of 10000 is depicted in Figure 5.5.

5.3.4. COMMUNICATION WITH THE PRIMARY FLIGHT COMPUTER

For communication with the serial_act_t4 module in Paparazzi UAV presented in Section 5.2.2, the Teensy also utilizes UART communication using the same inbound and outbound structures defined in the serial_act_t4_data_in and serial_act_t4_data_out data structures. These data structures can be found in the Definitions.h file in the

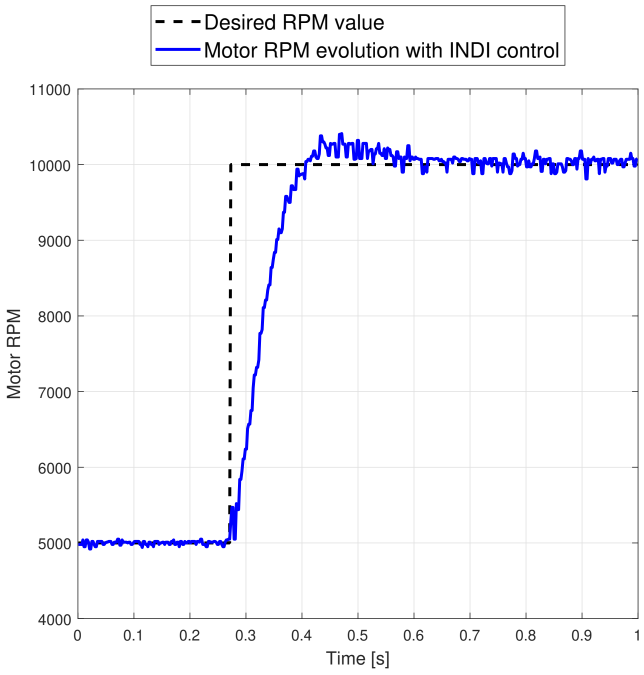


Figure 5.5: RPM step response using the INDI RPM controller.

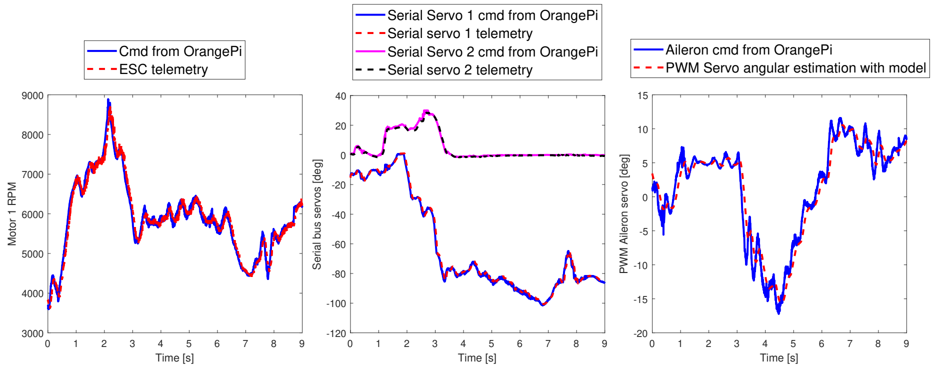


Figure 5.6: Actuators evolution during a portion of the dual-axis tilting rotor quadplane flight test.

Teensy_actuators_PPZ GitHub repository[12].

It is crucial to ensure that the structures defined in the Paparazzi module and in the Teensy firmware are identical. Any inconsistencies between the structures will result in improper communication between the hardware modules.

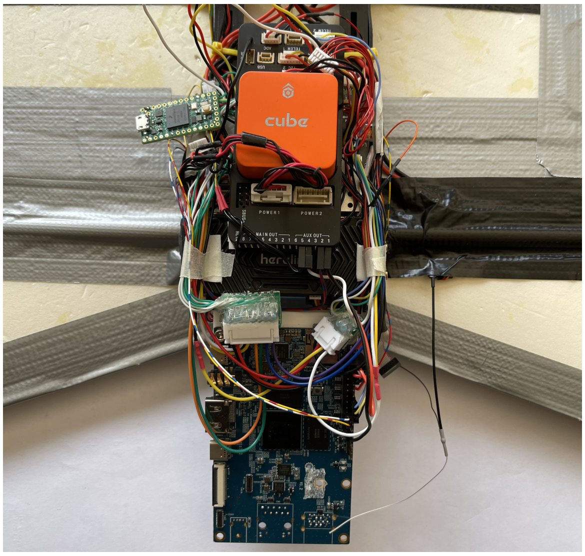


Figure 5.7: A picture of the Autopilot hardware mounted on the dual-axis tilting rotor quad-plane.

5.4. COMPANION COMPUTER SUBSYSTEM

As mentioned in the introduction, the utilization of ARM-A family processors in conjunction with the embedded ARM-M family processor significantly enhances the computational capabilities of any robotic platform. In our specific implementation, we employed an OrangePi 5 to tackle the complex Nonlinear Control Allocation problem for the dual-axis tilting rotor quad-plane.

The communication with the custom ca_am_7 module running on the Primary Flight Computer through UART is implemented on a separate thread on the OrangePi 5. Unlike the Teensy 4.0, which has a single-core processor, the OrangePi 5 features an Octa-Core processor. This enables us to distribute the serial communication routine and the control routine across different threads running on different cores.

The software developed for this purpose is available in the GitHub repository OrangePi_PPZ[13]. The software is structured around the am7x.h and am7x.c files. The am7x.h file contains the inbound and outbound data structures for serial communication with the Primary Flight Computer. As mentioned in the previous section, it is crucial for these data structures to match those implemented in the Paparazzi UAV ca_am_7 module, as presented in Section 5.2.2.

The am7x.c file implements UART communication within one thread and executes the MATLAB-generated function (or any other C function

[13]https://github.com/tudelft/OrangePi_PPZ

useful for the autopilot) on a separate thread. In our case, we execute a C-function automatically generated by the MATLAB C-coder toolbox. For information on generating a C function from a MATLAB function, please refer to the MATLAB documentation[6].

All the MATLAB-generated files are located in the MATLAB_generated_files folder of the OrangePi_PPZ repository[13] and can be invoked from the am7x.c file.

5.5. FLIGHT TEST RESULTS

To demonstrate the efficacy of the proposed Autopilot framework, we conducted tests using the dual-axis tilting rotor quad-plane depicted in Figure 4.1. During the flight, the companion computer subsystem computed the actuator solution, which was then transmitted to the actuator control and real-time feedback subsystem. Figure 5.7 presents a photograph of the Autopilot hardware, while Figure 5.6 illustrates the observed evolution of the actuators command and state during the flight test.

5.6. CONCLUSION

In this chapter, we introduced a cutting-edge modular Autopilot framework designed to handle a multitude of actuators while providing real-time, high-frequency feedback on their state. We presented comprehensive software and hardware details necessary for implementing the framework on a wide range of UAVs using the Paparazzi UAV platform. We have also developed an INDI RPM controller leveraging the high-frequency feedback, enabling precise control of motor RPM. Furthermore, the modular Autopilot includes a companion computer subsystem, capable of efficiently solving complex real-time tasks to support the Primary Flight Computer. The effectiveness of the Autopilot functionalities was then demonstrated through a successful flight test of a dual-axis tilting rotor quad-plane equipped with the system.

REFERENCES

[1] C.DeWagter, B.Remes, E.Smeur, F.VanTienen, and R.Ruijsink. "The NederDrone: A hybrid lift, hybrid energy hydrogen UAV." In: *International Journal of Hydrogen Energy* Volume 46, Issue 29 (2021). doi: 10.1016/j.ijhydene.2021.02.053.

[2] A. Bin Junaid, A. Diaz De Cerio Sanchez, J. Betancor Bosch, N. Vitzilaios, and Y. Zweiri. "Design and Implementation of a Dual-Axis Tilting Quadcopter". In: *Robotics* 7.4 (2018). issn: 2218-6581. doi: 10.3390/robotics7040065.

[3] A. Mancinelli, E. Smeur, B. Remes, and G. Croon. "Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design." In: *International Conference on Unmanned Aircraft Systems (ICUAS)* (2022). doi: 10.1109/ICUAS54217.2022.9836063.

[4] C. Papachristos, K. Alexis, and A. Tzes. "Efficient force exertion for aerial robotic manipulation: Exploiting the thrust-vectoring authority of a tri-tiltrotor UAV". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 4500–4505. doi: 10.1109/ICRA.2014.6907516.

[5] E. Smeur, G. de Croon, and Q. Chu. "Cascaded incremental nonlinear dynamic inversion for MAV disturbance rejection". In: *Control Engineering Practice* 73 (2018), pp. 79–90. issn: 0967-0661. doi: 10.1016/j.conengprac.2018.01.003.

[6] G. D. Francesco, M. Mattei, and E. D'Amato. "Incremental Nonlinear Dynamic Inversion and Control Allocation for a Tilt Rotor UAV". In: *AIAA Guidance, Navigation, and Control Conference* (2014). doi: 10.2514/6.2014-0963.

[7] G. H. Eben Upton. *Raspberry Pi User Guide*. Wiley, 2016.

[8] J. Yang, A. Thomas, S. Singh, S. Baldi, and X. Wang. "A Semi-Physical Platform for Guidance and Formations of Fixed-Wing Unmanned Aerial Vehicles." In: *MDPI sensors* (2020). doi: 10.3390/s20041136.

[9] G. Gargioni, M. Peterson, J. B. Persons, K. Schroeder, and J. Black. "A Full Distributed Multipurpose Autonomous Flight System Using 3D Position Tracking and ROS". In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)* (2019), pp. 1458–1466. doi: 10.1109/ICUAS.2019.8798163.

[10] H. Mohr. “UAV Implementation of Distributed Robust Target Location in Unknown Environments”. In: *2020 IEEE Aerospace Conference* (2020), pp. 1–10. doi: 10.1109/AERO47225.2020.9172459.

[11] A. Mancinelli, B. D. W. Remes, G. C. H. E. De Croon, and E. J. J. Smeur. “Real-Time Nonlinear Control Allocation Framework for Vehicles with Highly Nonlinear Effectors Subject to Saturation”. In: *Journal of Intelligent & Robotic Systems* 108 (July 2023). doi: 10.1007/s10846-023-01865-8.

[12] B. Gati. “Open source autopilot for academic research - The Paparazzi system.” In: *2013 American Control Conference* (2013). doi: 10.1109/ACC.2013.6580045.

6 FAULT TOLERANT CONTROL FOR THE DUAL-AXIS TILTING ROTOR QUAD-PLANE

In Chapter 5, a significant upgrade to the hardware architecture of the dual-axis tilting rotor quad-plane was introduced, incorporating real-time actuator state feedback to enhance control performance and provide continuous situational awareness of the propulsion system's health. This enhancement laid the foundation for developing a robust fault-tolerant control framework capable of maintaining reliable operation despite potential hardware failures.

Building on the improved hardware architecture, this chapter presents the development and implementation of a fault-tolerant control strategy for the dual-axis tilting rotor quad-plane. By leveraging the real-time actuator feedback system, the vehicle's over-actuated design, and the model-based control structure, the platform is able to adapt to degraded actuator conditions. The controller dynamically reallocates control efforts among the remaining functional actuators to preserve flight stability. The robustness of the fault-tolerant framework is validated through extensive flight tests, demonstrating its ability to maintain stable operation even under severe actuator failures.

This chapter is based on the following article:
A. Mancinelli, N. Voss and E.J.J. Smeur, "Fault Tolerant Control in Over-Actuated Hybrid Tilt-Rotor Unmanned Aerial Vehicles," in AIAA SCITECH 2025 Forum, 2025.

6.1. INTRODUCTION

The development of VTOL aircraft has been driven by the benefit of combining efficient high-speed, long-range flight with the capability of landing on unprepared or size-restricted areas. Before the era of unmanned systems, several projects investigated full-scale concepts to achieve this combination by using separate or multi-purpose propulsion systems. Despite the increased mechanical and control complexity, using a single propulsion system for both modes of flight is desirable for efficiency. Tilt-wing and tilt-rotor systems have established themselves as the current industry standard, enabling a single propulsion system across hover, transition, and forward flight regimes by adjusting the thrust vector accordingly.

For smaller and more agile systems, research has expanded toward utilizing tilt mechanisms for additional functions, such as gust rejection or physical interaction tasks [1, 2]. At the same time, powerful System-on-Chip (SoC) solutions offer affordable computing power to implement complex real-time CA algorithms within limited power and weight budgets [3]. The addition of independent rotor tilt axes can increase the degrees of freedom the vehicle can independently control.

By adding only two additional servos and spherical joints to a quadcopter, Zheng et al. [4] were able to control the common motor tilt and demonstrated attitude-independent thrust. This configuration tilts the common thrust vector via a gimbal linkage to control motor rotation parallel to the body's pitch and roll axes.

Several other studies have previously investigated the increased maneuverability resulting from tilt configurations, including Junaid et al. [5], who specifically noted improvements in maneuvering flight and obstacle avoidance. A step further in complexity is the independent tilt of individual rotors: [6] developed a quad-plane with independent single-axis tilt along the vehicle's pitch axis on all four rotors and investigated motor failure in hover and forward flight. They successfully simulated recovery from motor failure in both hover and forward flight modes.

In previous research [7], we took this concept one step further and developed a quad-plane with independent dual-axis tilt of each motor, shown in Figure 6.1. The ability to directly generate forces independent of moments gives the vehicle increased disturbance rejection and maneuverability capabilities. Because of the nonlinearities in tilting each thrust vector, a nonlinear optimization control allocation approach is necessary. Compared to the previously introduced quad-plane [6], the additional tilting of motors parallel to the vehicle's roll axis allows for lateral thrust vectoring and reorientation in roll and pitch under motor failure.

In order to tolerate actuator failures, two major approaches to Fault Tolerant Control (FTC) can be discerned. In active FTC, the system is

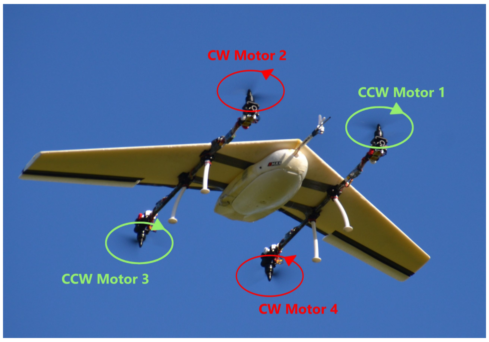


Figure 6.1: Dual-axis tilt TRUAV quad-plane developed by Mancinelli et al. [7]

actively monitored, and faults need to be detected and communicated to the controller. With information on the failure, which relies on a Fault Detection and Identification (FDI) mechanism, the controller takes action to mitigate the failure. In passive FTC, the controller is instead designed to be robust to failures within established limits. This usually results in a more conservative design and reduced overall performance but does not require any active monitoring component [8–10]. As both introduced tilt-rotor quad-planes rely on an onboard dynamics model for their control allocation, a means to inform the controller of the changed configuration is required. For this reason, active fault tolerance is applied to update the model accordingly.

Wang and Sung [11] focused on the effect a lifting body has on more classical types of recovery for quadrotors and proposed a novel Incremental Adaptive Sliding Mode Control (I-ASMC) approach to mitigate uncertainties in the modeled aerodynamic forces and interactions of rotor and wing [11]. The proposed solution simulated flight with only three rotors and had the spinning quad-plane follow a rectangular reference trajectory. This relaxed hover state, allowing yaw rotation, is a proven approach to overcome the inherent under-actuation of quadcopters [12]. Sun et al. previously demonstrated flight with two rotors for a commercial drone [13], and Zhang et al. designed a vehicle capable of tracking position with only a single actuator [14].

In this chapter, we demonstrate fault-tolerant control of a dual-axis tilt quadplane, with onboard optimization of a new static hover condition. Maintaining static hover has the advantage of avoiding the need to model and adapt to the complex aerodynamic interactions of a spinning wing. We developed a cascaded nonlinear control allocation strategy that separates attitude control from actuator command optimization. By still including the actuators in the attitude control loop, the obtained pitch and roll angle commands respect the actuator limits, even though the actuator commands are not used and are an output of the inner control loop. Test flights demonstrate the capability of the control framework to deal with actuator faults of the dual-axis tilt quad-plane.

6.2. METHOD

6.2.1. CONTROLLER LAYOUT

The vehicle's controller is based on the INDI implementation of [15] and consists of two main components. The primary component is a single-loop Control Allocation algorithm, which generates commands for 13 physical actuators (8 tilting servos, 4 motor RPM commands, and 1 aileron servo pair) as well as two virtual actuators: the vehicle's roll and pitch angles. The Control Allocation algorithm determines the optimal actuator commands to achieve the desired accelerations by minimizing a cost function that includes a model of the vehicle dynamics, as derived in the previous section. The Control Allocation algorithm receives angular and linear acceleration inputs generated by a linear error controller.

The error controller provides acceleration references for both linear and angular accelerations. For linear acceleration references, the error controller receives a setpoint for the desired vehicle position. It then uses feedback from the vehicle's current position and speed to generate the necessary linear acceleration references. A similar method is applied for angular acceleration generation, where a PD (Proportional-Derivative) error controller, with feedback on body rates and Euler angles, is used to generate the angular acceleration references. A schematic representation of the architecture can be seen in Figure 6.2.

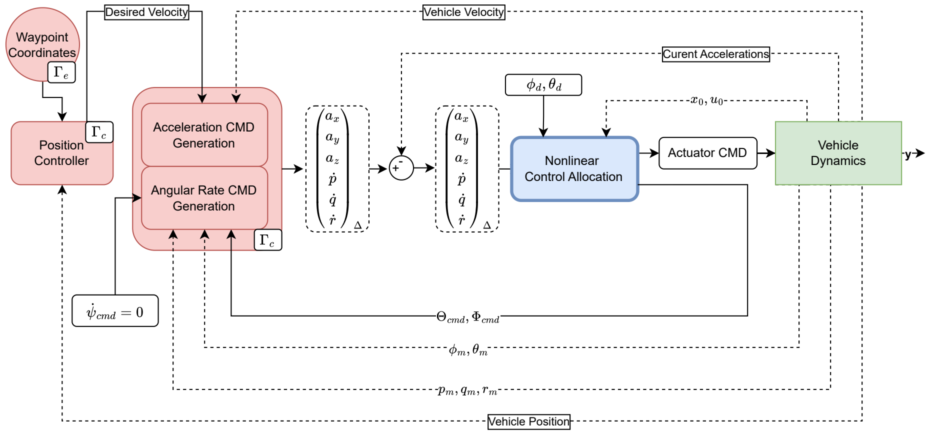


Figure 6.2: The original controller layout including the error controllers.

6.2.2. THE COST FUNCTION

The CA algorithm generates the control input commands by minimizing a cost function. By adjusting the cost function, specific control objectives can be prioritized. This approach integrates with the previously presented dynamics and control scheme by aiming to match the acceleration increments generated by the upstream error controllers, using the equations of motion in Equation 4.3.

The following equation shows a breakdown of the cost function structure and its individual components:

$$C(u) = ||W_v(f_s(x_0, u) - \nu_n)||^2 + \gamma_u ||W_u(u - u_d)||^2 \tag{6.1}$$

$$u_s = \text{ arg min } C(u)$$

$$\text{subject to} \\ u_{min} < u < u_{max}, \tag{6.2}$$

where the goal is to minimize the cost $C(u)$ by finding a u_s that minimizes the difference between the desired and predicted acceleration increment, $f_s(x_0, u) - \nu_n$. Here, $f_s(x_0, u)$ represents the modeled linear and angular acceleration, according to Equation 4.3 from the control input u and the vehicle states x_0, while ν_n is the vector of linear and angular acceleration increments targeted by the error controller.

As a secondary objective for the optimization process, the cost function also minimizes the difference between commanded and desired actuator settings, $u - u_d$. The control input vector u containing the computed commanded controls and attitudes, is structured as follows:

$$\begin{aligned} u = (&\Omega_1, \Omega_2, \Omega_3, \Omega_4, b_1, b_2, b_3, b_4, \\ &g_1, g_2, g_3, g_4, \delta_a, \theta_{cmd}, \phi_{cmd}), \end{aligned} \tag{6.3}$$

where Ω_i denote individual motor rotational speed in rad/s, b_i the motor gimbal elevation, g_i the motor gimbal azimuth and δ_a the aileron deflection. The definition of the tilt angles can be seen in Figure 2.5. The desired control input vector is chosen primarily to minimize motor usage, as follows:

$$\begin{aligned} u_d = (&150, 150, 150, 150, 0, 0, 0, 0, \\ &0, 0, 0, 0, 0, \theta_d = 0, \phi_d = 0). \end{aligned} \tag{6.4}$$

The primary and secondary objectives of the optimization are differentiated using the scaling factor γ_u, which is set to a very small value. The weighting matrices W_v and W_u are used to prioritize or penalize specific control objectives or actuators respectively. Finally, u_{min} and u_{max} provide the control input constraints for the optimization problem, as specified in Table 6.1.

6.2.3. CASCADED CONTROL STRUCTURE

When a motor fails, the required attitude change that allows static hover can be very significant. A limitation of the combined control allocation as in Equation 6.1, is that a new attitude is found with actuator commands that satisfy the linear and angular acceleration control objective at that new attitude.

However, the forces generated by the physical actuators will be valid for the associated computed attitude, which has a much lower bandwidth. Therefore, accelerations initially occur in the wrong frame when big attitude changes are commanded.

To address this limitation, we developed a new control framework that separates the computation of attitude commands from physical actuator commands in a cascaded manner. The controller sequentially solves two distinct optimization problems. The first optimization problem, similar to the control problem in Equation 6.1, determines the optimal attitude command for the vehicle in alignment with physical actuator limitations, based on linear acceleration target increments. In this initial optimization run, the angular acceleration targets are set to zero, assuming that the vehicle will reach a steady state at the new attitude. Although the first optimization run also produces preliminary physical control input commands, these are disregarded. However, it is important to include them, such that the optimizer can take the required actuator inputs (and limits) at this final state into account.

The attitude commands from the first optimization run are then fed to the attitude error controller, which generates the required angular acceleration commands to achieve the targeted attitude. These angular acceleration commands are subsequently combined with the initial linear acceleration targets and passed to the second optimization run, which computes the final physical actuator commands.

The derivation of the updated control laws used in the first optimization run is as follows:

$$\begin{pmatrix}\nu_n^p & u_d & u_0\end{pmatrix} \xRightarrow[\text{CA-1}]{} \begin{pmatrix}\cancel{\Omega_{1-4}} & \cancel{b_{1-4}} & \cancel{g_{1-4}} & \cancel{\delta_a} & \theta_{cmd} & \phi_{cmd},\end{pmatrix} \tag{6.5}$$

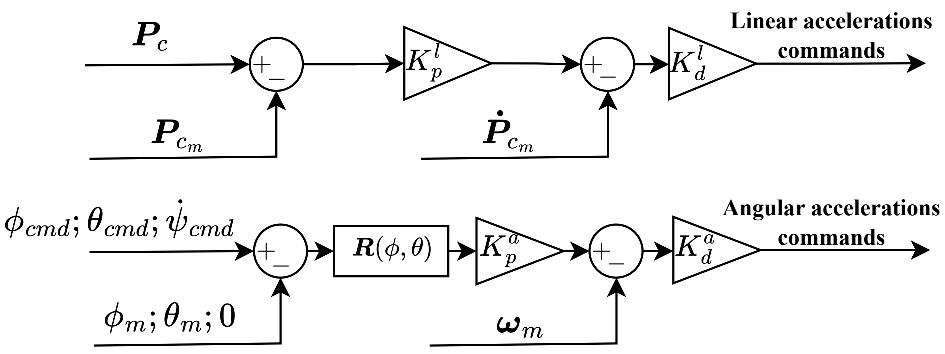
where the term ν_n^p represents the modified pseudo-control vector, containing only the linear acceleration components of the error controller, defined as follows:

$$\nu_n^p = \Delta\begin{pmatrix}a_x & a_y & a_z & 0 & 0 & 0\end{pmatrix}. \tag{6.6}$$

The commanded attitude determined through Equation 6.5 is then fed to an intermediate error controller. This error controller incorporates a proportional gain on the Euler angle error to produce Euler angles derivatives commands. The Euler angle kinematics are then used to convert the Euler angle rates commands to body rates commands:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix}_{cmd} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_{cmd} = \boldsymbol{R}(\phi,\theta) \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_{cmd} \quad (6.7)$$

A scheme of the linear error controller is shown in Figure 6.3.



$\boldsymbol{P}_c$
$\boldsymbol{P}_{c_m}$
K_p^l
$\dot{\boldsymbol{P}}_{c_m}$
K_d^l
Linear accelerations commands
$\phi_{cmd}; \theta_{cmd}; \dot{\psi}_{cmd}$
$\phi_m; \theta_m; 0$
$\boldsymbol{R}(\phi,\theta)$
K_p^a
$\boldsymbol{\omega}_m$
K_d^a
Angular accelerations commands

Figure 6.3: Linear error controller used to determine the linear and angular acceleration commands.

The angular acceleration increments generated through the error controller, are then added to the initial linear acceleration increments to form the final acceleration increment vector for the second optimization run:

$$\nu_n^s = \Delta\begin{pmatrix} a_x & a_y & a_z & \dot{p} & \dot{q} & \dot{r} \end{pmatrix}. \quad (6.8)$$

This stage has been modified to only compute outputs for the physical actuators, and the attitude angles are not optimized. Instead, only the attitude generated by the first stage optimizer is used.

$$\begin{pmatrix} \nu_n^s & u_d^s & u_0^s \end{pmatrix} \xRightarrow[CA-2]{} \begin{pmatrix} \Omega_{1-4} & b_{1-4} & g_{1-4} & \delta_a \end{pmatrix}_{cmd} \quad (6.9)$$

Here, u_d^s and u_0^s represent respectively the desired and current control input vectors used in the second optimization run. These vectors are identical to those from the first optimization run, except that they exclude the last two attitude elements.

The revised architecture can be seen in Figure 6.4. Despite the additional computation step, the sampling frequency for the optimization algorithm consistently remains above 200 Hz.

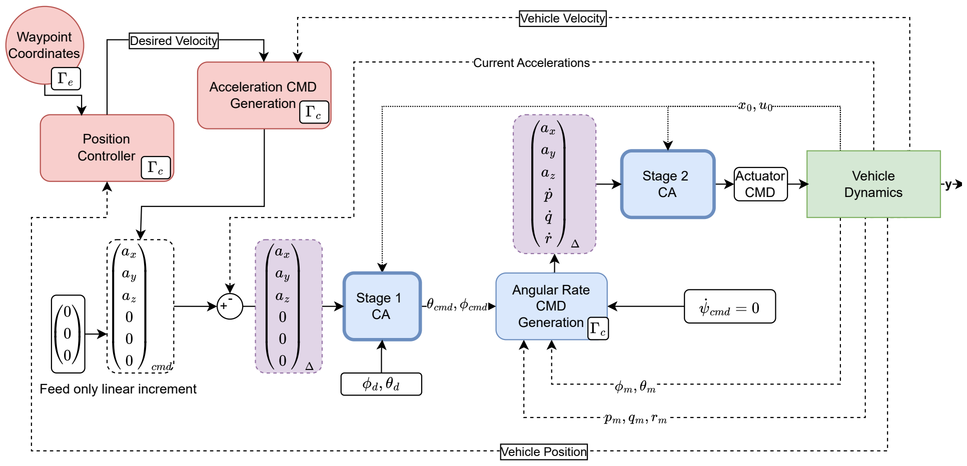


Figure 6.4: The cascaded layout optimizing attitude and actuators in different steps.

6.2.4. THE CONTROL ALLOCATION SOLVER

The presented optimization problem is solved using the SQP approach, and makes use of the Matlab *fmincon* function. This implementation includes suggested improvements to K. Schittkowski's well-documented explanation of the algorithm [16] and is built on the work of Nocedal and Wright [17]. The choice of using the Matlab function also allows for the use of the Coder toolbox, accelerating the process of implementing the developed controllers on the drone.

6.2.5. TEST SCENARIO AND OPTIMAL ATTITUDE IDENTIFICATION

As the most power-intensive flight condition, motor failure during hover was selected as the primary failure case for investigation. While forward flight offers the advantage of lift generated by the wing and effective use of aerodynamic control surfaces, these benefits are absent in hover. Additionally, the failure of a motor renders the respective tilt servos ineffective, as no thrust vector can be generated.

With an identified thrust coefficient of $K_T^P = 1.106465 \times 10^{-5}$, N/(rad/s)^2 and a maximum motor speed of 1000 rad/s recorded with the current power system, the maximum thrust each motor can provide is approximately 11.1 N. For a vehicle mass of 2.5 kg, this results in a hover thrust-to-weight ratio of about 1.8 with all engines operational. The failure of a single motor reduces the maximum thrust and thrust-to-weight ratio to 33.2 N and 1.35, respectively. Further motor failures would leave insufficient thrust, without accounting for the need to generate moments and linear accelerations.

This implies that current actuator fault control strategies, which rely on switching off the opposing motor in the event of a motor failure, would not provide enough thrust for sustained hover flight post-failure. Instead, the vehicle must fully utilize the remaining three operational motors, reorienting its attitude to achieve a new symmetric thrust equilibrium that ensures moment balance.

To analyze the behavior of the first CA optimizer in more detail, we examined the failure of motor 3 (back right). The cost function output from the first optimization run was evaluated across different attitude values. The resulting surface plot is shown in Figure 6.5. From the plot, we observe that the optimal attitude after the failure of the back right motor corresponds to $\theta = 67°$ and $\phi = 90°$. Another significant observation is that the shape of the cost function may direct the optimization process toward a local minimum located at $\theta = -15°$ and $\phi = -90°$. This aspect is crucial when defining the attitude constraints under actuator failure conditions.

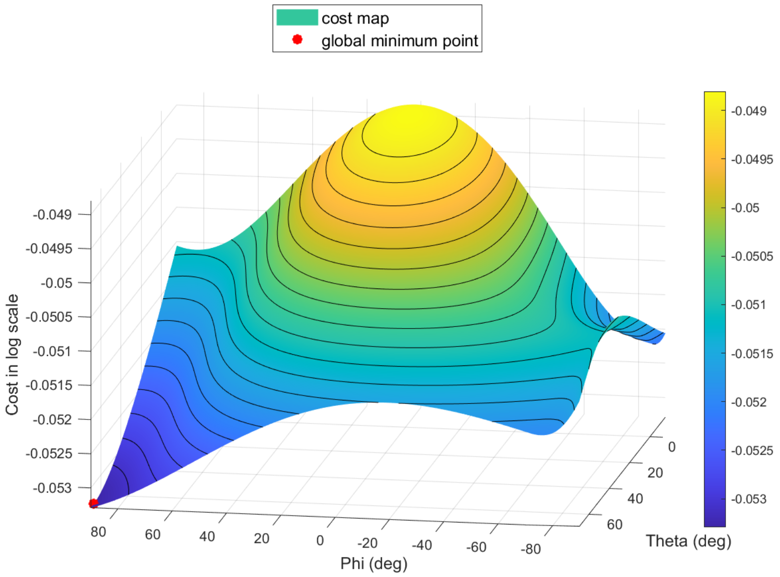


Figure 6.5: Cost function value for different attitude angles. The cost function numerical value was obtained using logarithm with base 10.

6.2.6. FAILURE INFORMATION: CONSTRAINT SETS AND WEIGHTING MATRICES

It is essential to implement a mechanism that informs the controller of failures and prevents the allocation algorithm from utilizing ineffective actuators. This fault information mechanism is straightforward to implement, as the platform uses very high-speed feedback from all its actuators, as described in [18]. The failure information is communicated to the controller, which subsequently restricts the control input constraints during both the first and second optimization runs. The constraints for the non-faulty actuators can be found in Table 6.1, which are applied during regular flight. It is important to note that pitch and roll limits were established to prevent significant angular ambiguities that could arise from the ZYX rotation order.

Once the failure is triggered, the controller internally switches to a different set of constraints, causing the selected motor to cut thrust and tilt the faulty rotor into a neutral orientation. This activated set of constraints is shown in Table 6.2. The motor speed was not fully reduced to zero, as the framework did not allow the motor speed Ω_i to drop below 120 rad/s at idle. Additionally, adjustments were made to the roll angle constraint, which was forced to compute a value in the positive roll region to prevent the optimizer from converging to a local minimum, as discussed in the previous subsection.

Each motor failure case requires a unique constraint. For example, in the event of a back-left motor failure, the roll angle constraint would range between $-\phi_{max}$ and zero, since the shape of the cost function mirrors the one shown in Figure 6.5.

Constraint	Minimum	Maximum
Motor speed	120 rad/s	1000 rad/s
Tilt Elevation	-130°	20°
Tilt Azimuth	-95°	95°
Theta cmd	-15°	45°
Phi cmd	-65°	65°

Table 6.1: Default Actuator and Attitude Constraints.

Considering the different weights, it is essential to discuss how the weighting matrices and scaling factors from Equation 6.1 affect drone behavior in nominal and failure conditions. A secondary objective is minimizing power consumption during failure, making motor cost an important factor alongside primary acceleration tracking. The motor cost influences the computed optimal attitude and motor orientation; for instance, a small motor weight may result in motors not pointing

No Fault	**Failed M3**
$150 \leq \Omega_{m_3} \leq 1000$ rad/s	$\Omega_{m_3} = 150$ rad/s
$-130 \leq \delta_{el3} \leq 20\ ^\circ$	$\delta_{el3} = 0\ ^\circ$
$-95 \leq \delta_{az3} \leq 95\ ^\circ$	$\delta_{az3} = 0\ ^\circ$
$-65 \leq$ Phi cmd $\leq 65\ ^\circ$	$0 \leq$ Phi cmd $\leq 65\ ^\circ$

Table 6.2: Adjustment in Case of M3 Failure.

directly upward during hover. A high motor weight minimizes motor use and indirectly promotes vertical alignment, efficiently utilizing the available thrust.

The attitude and servo costs have a similar purpose during hover, both costs stabilize the system back towards pointing the motors and vehicle straight and level. The ailerons were largely unused during the flight test, as the airspeed was set to zero, rendering them ineffective.

Various interactions are involved, most of which change in the event of a failure. During motor failure, the drone should avoid expending unnecessary effort in keeping a stable yaw reference. To facilitate this, the weights are adjusted as shown in Table 6.3. Lowering the weights on attitude enables the cascaded optimizer to explore orientations far from the initial hover configuration without incurring a large cost from attitude deviation. As a result, the first stage optimizer is less constrained and can find an attitude significantly different from ϕ_d and θ_d.

During the tests, the cost on absolute servo use was set to zero, allowing the motor orientation to be optimized by considering the commanded attitude and minimizing motor power.

6.3. RESULTS AND DISCUSSION

This section presents the data gathered from the flight tests. Prior to the flight test campaign, the algorithm was rigorously tested and refined within a simulation environment developed in Simulink. Once the Simulink simulation produced promising results, the code was generated from the MATLAB functions and compiled to run in real time on the UAV's onboard single-board computer. For a more detailed analysis of the hardware used in these tests, refer to [18].

6.3.1. FLIGHT TEST SETUP

For the test flights, the system was informed of the failure through a switch activated by the pilot. The drone was flown to a pre-selected hover point, and once stable hover at the designated position was achieved, the controller was notified of the failure of motor number

	Component	Stage 1 Optimization Run		Stage 2 Optimization Run		Effect
		Default	Motor Failure	Default	Motor Failure	
W_u	Motor	$W_\Omega = 10$	$W_\Omega = 70$	$W_\Omega = 20$	$W_\Omega = 20$	Penalize deviation from u_d.
	Servos	$W_{\delta_{az}} = 0$ $W_{\delta_{el}} = 0$ $W_{\delta_{ail}} = 0.5$	$W_{\delta_{az}} = 0$ $W_{\delta_{el}} = 0$ $W_{\delta_{ail}} = 0.5$	$W_{\delta_{az}} = 0$ $W_{\delta_{el}} = 0$ $W_{\delta_{ail}} = 0.5$	$W_{\delta_{az}} = 0$ $W_{\delta_{el}} = 0$ $W_{\delta_{ail}} = 0.5$	
	Attitude	$W_\theta = 100$ $W_\phi = 100$	$W_\theta = 1$ $W_\phi = 1$	—	—	
W_v	Accelerations	$W_{ax} = 0.01$ $W_{ay} = 0.01$ $W_{az} = 0.05$ $W_{\dot p} = 0.1$ $W_{\dot q} = 0.1$ $W_{\dot r} = 0.1$	$W_{ax} = 0.1$ $W_{ay} = 0.1$ $W_{az} = 0.1$ $W_{\dot p} = 0.01$ $W_{\dot q} = 0.01$ $W_{\dot r} = 0.01$	$W_{ax} = 0.01$ $W_{ay} = 0.01$ $W_{az} = 0.05$ $W_{\dot p} = 0.1$ $W_{\dot q} = 0.1$ $W_{\dot r} = 0.1$	$W_{ax} = 0.01$ $W_{ay} = 0.01$ $W_{az} = 0.05$ $W_{\dot p} = 0.1$ $W_{\dot q} = 0.1$ $W_{\dot r} = 0.05$	Penalize acceleration residuals.
γ_u	Cost function	$\gamma_u = 3e^{-7}$	$\gamma_u = 3e^{-7}$	$\gamma_u = 3e^{-7}$	$\gamma_u = 3e^{-7}$	Scales the secondary objective in the cost function.

Table 6.3: Default and Failure Weighting Matrices Used in the Stage 1 and Stage 2 Optimization runs.

3 (back-left motor). The system response and onboard data were recorded, and the failure was introduced repeatedly.

A representation of the flight test plan is shown in Figure 6.6, which illustrates the selected failure case of losing motor 3 (back-right). Following this, a waypoint tracking scenario was conducted to evaluate the maneuvering capabilities with the remaining three motors. During the flight test, a safety rope was used to ensure a safe environment and to secure the drone in case of unexpected behavior. The rope was continuously managed by an operator to maintain sufficient slack, ensuring it did not interfere with the flight test results.

6.3.2. FLIGHT TEST RESULTS

During the test flights, the controller successfully stabilized the vehicle after a failure was introduced on motor 3 and was able to follow waypoint changes. The resulting commands and other metrics logged by the onboard computer were analyzed and are presented in this section. A video showcasing the flight test experiment, including a synthetic 3D vehicle visualization to better analyze the forces generated by the motors, has been uploaded to the MAVLab YouTube page [1].

By analyzing the flight test data from the repeated failure and moving waypoint scenarios, the following observations can be made:

[1]Video of the experiment: https://youtu.be/7fKJa7_T6L0

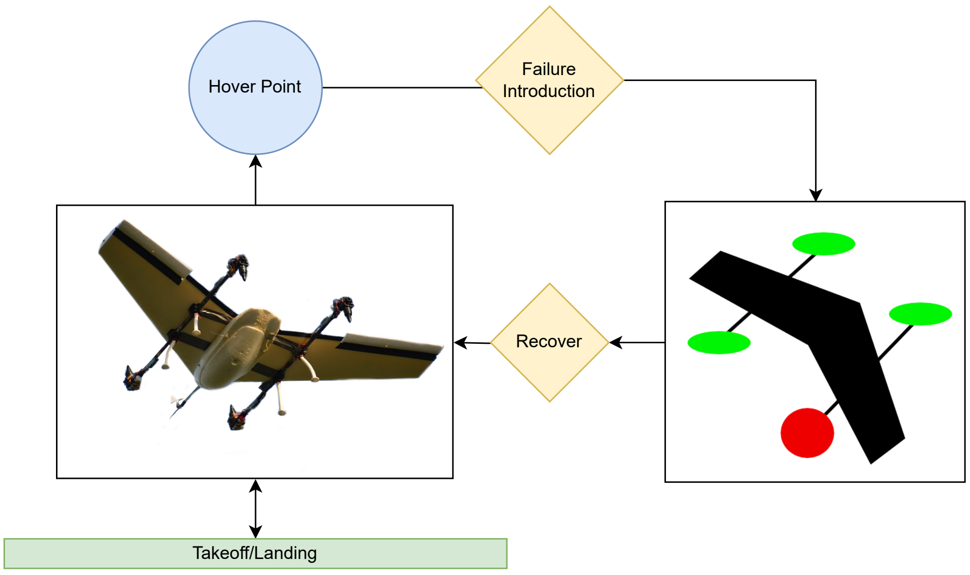


Figure 6.6: Test flight representation of maneuver.

- Tracking of the position and attitude setpoints remains consistent throughout the flight test, as shown in Figure 6.7. An initial displacement in position occurs immediately following the fault, but the error controller quickly corrects it. The maximum horizontal position error, recorded during the third failure test, was less than 1 meter. Additionally, there is a slight loss of altitude due to the fault; however, this loss consistently remains below 0.5 meters across all three failure maneuvers. It is worth noting that this loss is more evident during the second and third failure tests, as the first failure was induced before the vehicle reached the initial altitude target.

- Following the failure, the vehicle adjusts its attitude toward the global minimum identified in Figure 6.5. However, the attitude setpoints are constrained by enforced limits of $\phi = 65°$ and $\theta = 45°$.

- Overall, the transitions to and from the failure state are consistently well-recovered, with the vehicle smoothly switching configurations. The attitude target references generated by the first-stage optimizer remain stable and do not oscillate, indicating a well-conditioned optimization process.

- Looking at Figure 6.8, which plots the actuator evolution, the continuous failure demonstrates similar stability in the actuator solution during both failure and recovery transitions. However,

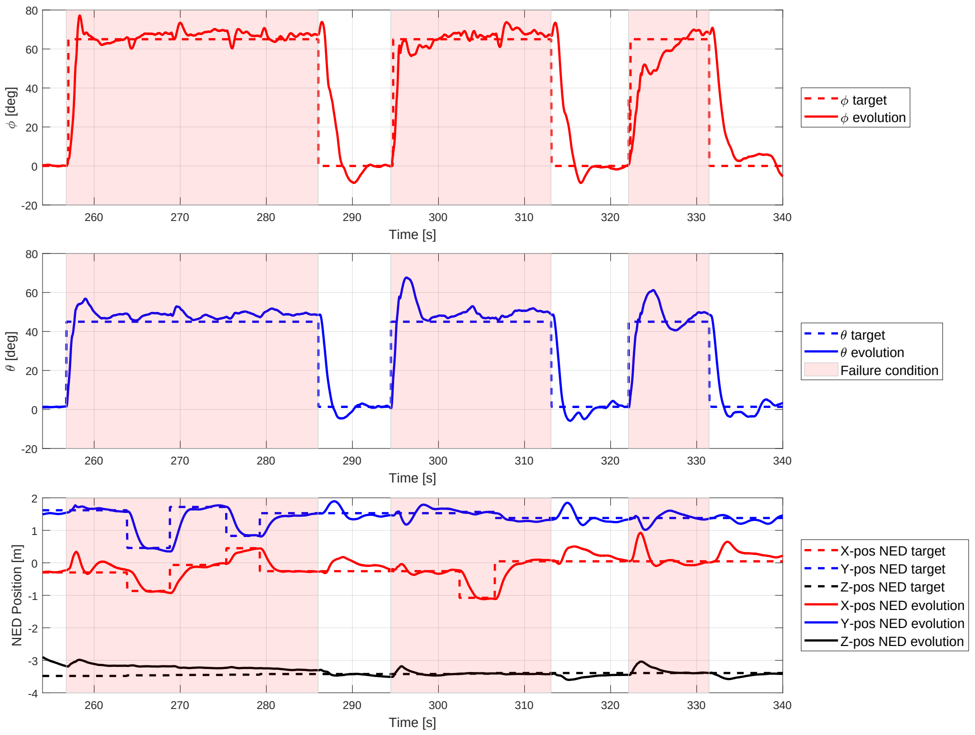


Figure 6.7: Attitude and position tracking during the flight test. The red shaded areas indicate when the simulated failure was active.

slight oscillations are observed, particularly in the elevation servos immediately after the failure. A deeper analysis, including simulations, confirms that these oscillations are caused by the redistribution of roll moment generation from the saturated azimuth tilt angle of rotor 1 to the remaining unsaturated actuators.

- Another relevant observation is that rotor 4 operates more intensively than rotors 1 and 2 during the failure condition. This discrepancy can be attributed to the constrained attitude configuration the drone adopts following the failure. Despite motor 4 reaching saturation and a transient saturation of rotor 1's azimuth angle, the other actuators remain well within their operational ranges.

- The recovery from a failure state to a non-failure state shows a smoother actuator response compared to the transition from a non-failure state to a failure state. Nevertheless, the smoothness of the transition could be improved by relaxing the attitude gains, albeit at the cost of increased positioning error.

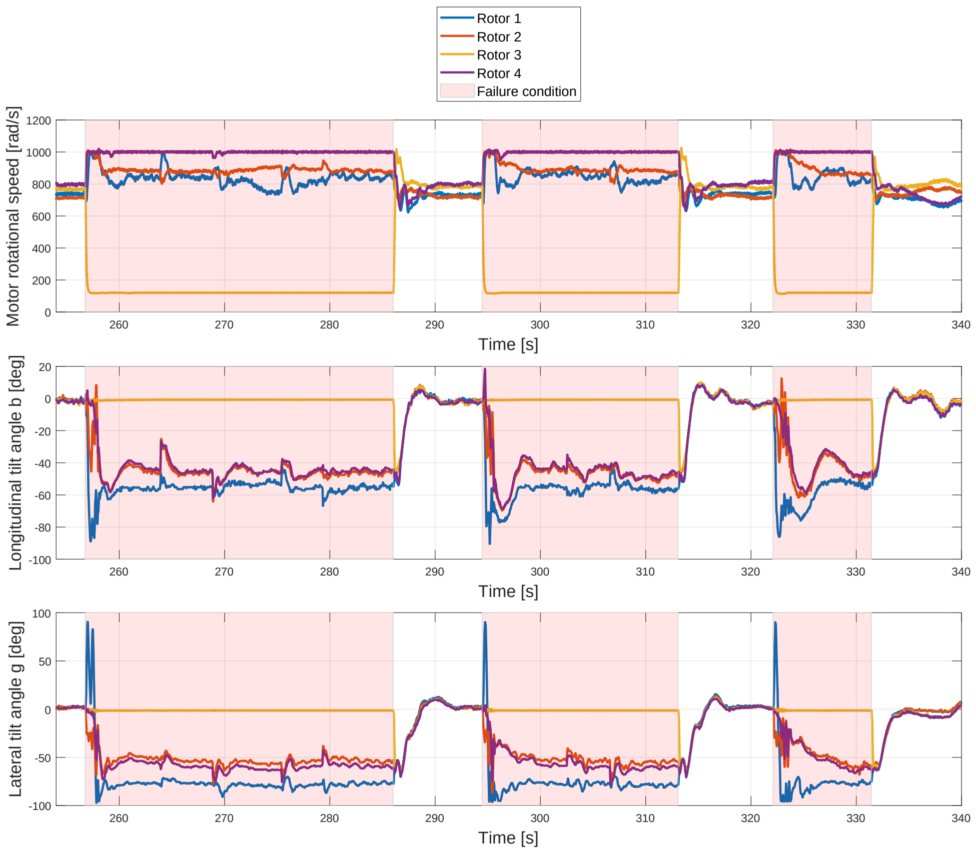


Figure 6.8: Actuator evolution during the flight test.

- Figure 6.9 illustrates the measured motor power and estimated thrust during the flight test. As anticipated, during the induced failure, motor 3's power and thrust drop to near zero, while the remaining motors, particularly motor 4, compensate by increasing their power output and thrust.

- The total thrust generated during failure conditions exceeds the thrust generated during normal hover (non-failure conditions) and is also greater than the vehicle weight. This is essential to counteract the imbalance caused by the inoperative motor and to stabilize the vehicle under the constrained attitude angles. Without these constraints, the total thrust generated would likely be closer to the vehicle's weight, as the vehicle would reach the optimal attitude configuration shown in Figure 6.5, thereby reducing the additional demand on the functioning motors.

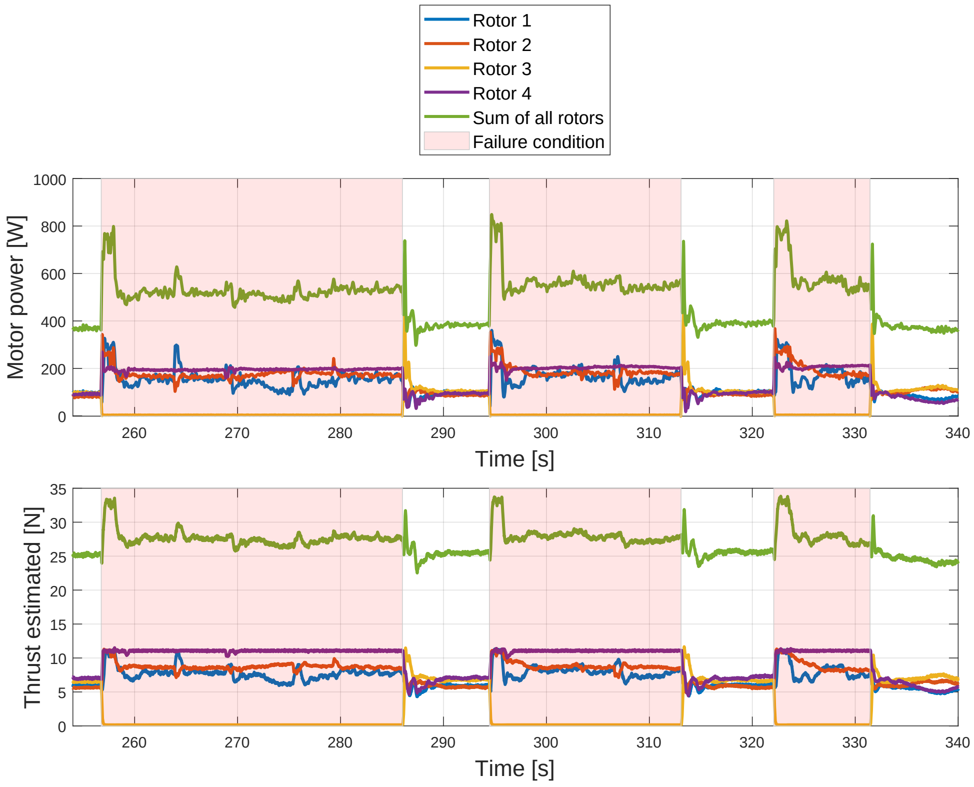


Figure 6.9: Measured motor power and estimated thrust during the flight test.

- During failure, the total power consumed by the remaining motors is significantly higher than during non-failure hovering because not all of the rotor thrust was oriented in the earth-vertical direction. This increase is primarily due to the system generating a higher amount of thrust, exceeding the drone's weight, to compensate for the angular acceleration balance. Furthermore, the remaining motors operate closer to their saturation points, where the efficiency of both propellers and motors decreases, further contributing to the higher power consumption.

6.4. CONCLUSION & RECOMMENDATIONS

This project set out to investigate the capabilities of the proposed unified non-linear controller with SQP CA on the dual-axis tilt quad-plane under actuator faults. As the most challenging case for this vehicle, the fault of an engine in hover was chosen to be the prime failure case to be considered. Failure dependent constraints on the faulty actuator were introduced to inform the CA of the changed operating conditions.

With the separation of attitude and actuator command optimization the previously problematic recursion loop between commanded attitude and desired angular acceleration increment was removed. Because of this, the actuator orientation is now always computed with respect to the actual vehicle attitude instead of the desired one. This is especially relevant when big attitude changes are commanded, such as in the case of an actuator failure.

Experiments show that the vehicle can successfully and repeatedly recover from a motor failure. Successful tracking of a changing reference waypoint was demonstrated with one failed motor. This chapter highlights the potential of over-actuated tilt-rotor configurations in establishing new ways of recovery by utilizing the available thrust vectoring to re-orient the vehicle into a new hover configuration.

6.4.1. LIMITATIONS & RECOMMENDATIONS

The flight tests were conducted in a controlled indoor environment, minimizing disturbances but limiting the ability to evaluate robustness under more unpredictable conditions. While the tests demonstrated repeated recovery, a longer and more comprehensive campaign is necessary to assess recovery performance across a wider range of scenarios.

Additionally, the flight tests did not include transitions from low airspeed to high airspeed regimes. In such conditions, aerodynamic forces could be leveraged to balance the vehicle's weight, reducing the impact of motor failure. Future work could explore the algorithm's effectiveness in forward flight mode, validating its capability to handle failures during high-speed operations.

Additional limitations of the project include the following points:

- It is possible that there are still more efficient hover attitudes available under failure. The available angles in pitch and roll were limited due to possibility of Euler angle gimbal lock and due to limitations on the vehicle tilt mechanism. With the ZYX rotation order, pitch angles close to or exceeding 90° cause instability in the controller. A cost function based on quaternions or switching roll and pitch rotation order could open up new possibilities.

- The tests were limited to the failure case of motor 3, more tests should be performed on different (types and combinations of) actuator failures.

- The motor speed cost was chosen to stimulate motor tilt alignment. A better flight efficiency could be achieved if the motor cost would be based on consumed power instead of the motor speed.

- This study focused heavily on the mitigation of a motor failure, but stuck servos and other failure scenarios should be looked at in real test-flights in the future.

REFERENCES

[1] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski. “The Voliro Omniorientational Hexacopter: An Agile and Maneuverable Tiltable-Rotor Aerial Vehicle”. In: *IEEE Robotics & Automation Magazine* 25.4 (Dec. 2018), pp. 34–44. doi: 10.1109/mra.2018.2866758.

[2] W. Myeong and H. Myung. “Development of a Wall-Climbing Drone Capable of Vertical Soft Landing Using a Tilt-Rotor Mechanism”. In: *IEEE Access* 7 (2019), pp. 4868–4879. doi: 10.1109/ACCESS.2018.2889686.

[3] C. Papachristos, K. Alexis, and A. Tzes. “Model predictive hovering-translation control of an unmanned Tri-TiltRotor”. In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 5425–5432. doi: 10.1109/ICRA.2013.6631355.

[4] P. Zheng, X. Tan, B. B. Kocer, E. Yang, and M. Kovac. “TiltDrone: A Fully-Actuated Tilting Quadrotor Platform”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6845–6852. doi: 10.1109/LRA.2020.3010460.

[5] A. Bin Junaid, A. Diaz De Cerio Sanchez, J. Betancor Bosch, N. Vitzilaios, and Y. Zweiri. “Design and Implementation of a Dual-Axis Tilting Quadcopter”. In: *Robotics* 7.4 (2018). issn: 2218-6581. doi: 10.3390/robotics7040065. url: https://www.mdpi.com/2218-6581/7/4/65.

[6] M. Mousaei, J. Geng, A. Keipour, D. Bai, and S. Scherer. “Design, Modeling and Control for a Tilt-rotor VTOL UAV in the Presence of Actuator Failure”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 4310–4317. doi: 10.1109/IROS47612.2022.9981806.

[7] A. Mancinelli, E. J. Smeur, B. Remes, and G. d. Croon. “Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design”. In: *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2022, pp. 197–206. doi: 10.1109/ICUAS54217.2022.9836063.

[8] J. Jiang, Y. Zhang, and X. Yu. “Fault-tolerant control systems: A comparative study between active and passive approaches”. In: *Annual Reviews in Control* 36 (1 2012), pp. 60–72. doi: 10.1016/j.arcontrol.2012.03.005.

[9] T. Jain, J. J. Yame, and D. Sauter. “Active Fault-Tolerant Control Systems: A Behavioral System Theoretic Perspective”. In: *null* (2017). doi: 10.1007/978-3-319-68829-9.

[10] A. Abbaspour, S. Mokhtari, A. Sargolzaei, and K. K. Yen. “A Survey on Active Fault-Tolerant Control Systems”. In: *Electronics* 9.9 (2020). issn: 2079-9292. doi: 10.3390/electronics9091513.

[11] X. Wang and S. Sun. “Incremental fault-tolerant control for a hybrid quad-plane UAV subjected to a complete rotor loss”. In: *Aerospace Science and Technology* 125 (2021), pp. 1–9. doi: 10.1016/j.ast.2021.107105.

[12] S. Bouabdallah and R. Siegwart. “Full control of a quadrotor”. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 153–158. doi: 10.1109/IROS.2007.4399042.

6

[13] S. Sun, X. Wang, Q. Chu, and C. de Visser. “Incremental Nonlinear Fault-Tolerant Control of a Quadrotor With Complete Loss of Two Opposing Rotors”. In: *IEEE Transactions on Robotics* 37.1 (2021), pp. 116–130. doi: 10.1109/TRO.2020.3010626.

[14] W. Zhang, M. W. Mueller, and R. D’Andrea. “Design, modeling and control of a flying vehicle with a single moving part that can be positioned anywhere in space”. In: *Mechatronics* 61 (2019), pp. 117–130. issn: 0957-4158. doi: 10.1016/j.mechatronics.2019.06.004.

[15] A. Mancinelli, B. D. W. Remes, G. C. H. E. de Croon, and E. J. J. Smeur. *Unified incremental nonlinear controller for the transition control of a hybrid dual-axis tilting rotor quad-plane*. 2023. doi: 10.48550/arXiv.2311.09185. arXiv: 2311.09185 [eess.SY].

[16] K. Schittkowski. “On the convergence of a sequential quadratic programming method with an augmented lagrangian line search function”. In: *Mathematische Operationsforschung und Statistik. Series Optimization* 14.2 (1983), pp. 197–216. doi: 10.1080/02331938308842847.

[17] J. Nocedal and S. J. Wright. “Sequential Quadratic Programming”. In: *Numerical Optimization*. New York, NY: Springer, 2006, pp. 529–562. isbn: 978-0-387-40065-5. doi: 10.1007/978-0-387-40065-5_18.

[18] A. Mancinelli, E. van der Horst, B. Remes, and E. Smeur. “Autopilot framework with INDI RPM control, real-time actuator feedback, and stability control on companion computer through MATLAB generated functions”. In: *14th annual international micro air vehicle conference and competition*. 2023, pp. 109–116. url: https://2023.imavs.org/.

7 PRECISION UAV LANDING ON A MOVING VESSEL IN HIGH SEA STATE

In the previous chapters, we developed a robust control framework for the dual-axis tilting rotor quad-plane, capable of maintaining stable operation across a wide range of flight conditions and effectively handling actuator failures. While these advancements established a reliable control foundation, the challenge of precise and resilient autonomous landing on a moving ship remains unaddressed.

This chapter focuses on developing a real-time trajectory planning algorithm aimed at enabling the quad-plane to autonomously land on a moving ship under harsh weather conditions. Leveraging an LSTM artificial neural network model, a prediction framework is formulated to estimate ship motion over a 7-second horizon. This prediction serves as the basis for generating feasible landing trajectory that are continuously reassessed to account for prediction uncertainties and tracking errors. The iterative evaluation and adaptation of landing trajectory during the maneuver ensure resilience against rapidly changing environmental conditions. The proposed strategy is validated through extensive simulation testing, demonstrating its potential for enabling robust autonomous landing operations.

7.1. INTRODUCTION

The capability for aircraft to land on moving ships has long been of significant interest, particularly in the military sector. Over the decades, the scientific community has developed various landing techniques to enhance reliability and precision, enabling rotary-wing manned vehicles to operate under extreme wind and sea conditions [1, 2]. With the rapid expansion of the Unmanned Aerial Vehicle (UAV) market, these techniques have been adapted for autonomous landings of Micro Aerial Vehicles (MAVs) [3–8]. The ship landing problem encompasses three primary challenges:

The first challenge in the ship landing problem is the estimation of the relative position and orientation of the landing pad with respect to the UAV. Traditional methods involve active equipment on the landing pad, such as sound-based estimation [1], Real Time Kinematics (RTK) GPS combined with an Inertial Measurement Unit (IMU) [9], Infra Red (IR) beacon systems [10], Ultra Wide Band (UWB) systems [11], and ground-based vision systems [12]. More recent approaches utilize onboard cameras and sensors, often employing fiducial markers like ArUco or AprilTag for pose estimation [13–15]. Advanced markerless methods leverage machine learning-based vision models to detect the landing pad, but these require pre-training and tend to be ship-specific [6, 16–18]. However, vision-based techniques are highly dependent on lighting conditions and struggle in low visibility scenarios, such as fog. Infrared-based systems [19, 20] offer a potential solution, but they too face limitations under extreme environmental conditions.

The second challenge in the ship landing problem is the prediction of the ship motion. Once the current motion of the landing pad is estimated, predicting its future motion is essential for identifying feasible landing windows. Since UAVs have strict constraints on linear and angular accelerations, forecasting ship motion enables trajectory planning that ensures a safe descent within these operational limits. One of the earliest approaches for ship motion prediction used Fast Fourier Transform (FFT) decomposition [3, 21, 22]. While computationally lightweight, these methods are generally limited to short-term predictions of up to 1 second [23]. With the rise of Artificial Neural Networks (ANNs), machine learning-based time series prediction models have emerged as a more robust solution. Long Short Term Memory (LSTM) Recurrent Neural Networks (RNNs) have been particularly effective in predicting nonlinear ship motions over longer horizons [24–26]. Studies show that LSTM-based models can predict ship motion up to 8 seconds ahead, achieving a maximum prediction error of less than 15% [27]. Recently, Transformer models have been explored for ship motion prediction due to their ability to capture long-range dependencies in sequential data [28, 29]. While some research suggests that LSTMs still outperform Transformers in specific contexts [30], Transformer-based models offer computational efficiency and are gaining traction in this field.

The third challenge in the ship landing problem is the generation of a feasible trajectory capable of safely landing the UAV on the moving landing pad. Once the relative pose estimation and prediction of the ship motion is available, the third element for a successful landing is the approach and landing trajectory generation. Typically, in literature, the ship prediction is used to identify an optimal landing window, characterized by minimal energy states of the ship's motion, to facilitate safe UAV touchdown. Once the landing time is determined, various methods can be employed to generate an appropriate trajectory. Polynomial-based trajectory generation techniques are commonly used, either in three-dimensional space [31] or focusing solely on the vertical (z) axis [32]. Alternatively, linear programming approaches have been applied to optimize UAV trajectories, ensuring adherence to kinematic constraints and obstacle avoidance [33]. A significant challenge in these methods is assessing the feasibility of the generated trajectory, particularly concerning the UAV maximum allowable accelerations. Ensuring that the planned trajectory remains within the UAV performance limits is essential for a successful and safe landing. Additionally, due to the continuously changing nature of ship motion, the trajectory must be re-evaluated and refined in real-time as new ship motion predictions become available and the UAV follows its descent trajectory.

In this chapter, we present an integrated landing algorithm that is both ship-agnostic and UAV-agnostic. The proposed approach specifically addresses the second and third challenges discussed earlier: ship motion prediction and trajectory planning generation. For ship motion prediction, we leverage insights from the literature and employ an LSTM network. The model is trained with a learning window of 400 seconds, generating ship motion predictions up to 7 seconds ahead. The LSTM network is implemented in MATLAB, running on consumer-grade laptop hardware with an evaluation interval of 0.3 seconds and retraining every 6 seconds. This iterative training process allows the model to dynamically adapt to changing sea states and ship motion characteristics. For trajectory planning, we developed an iterative method that computes a feasible landing trajectory by matching the UAV's current position and velocity with the predicted landing pad position and velocity. The feasibility of the generated trajectory is ensured by optimizing acceleration, speed, and landing time constraints, ensuring the UAV remains within its operational limits.

The proposed algorithm was validated using real-world ship motion data recorded onboard the Guardian ship, an offshore safety vessel with a length of 65 meters and a width of 15 meters, operating under sea state 5 conditions. The data was collected using an RTK GPS + IMU system and replayed in a real-time simulation environment in Simulink. Within the same simulation framework, both the ship motion prediction algorithm and the landing trajectory planning algorithm were tested using a high-fidelity dual-axis tilt-rotor quad-plane UAV model. A reference image of the Guardian vessel is shown in Figure 7.1, while Figure 4.1 depicts the UAV used in this study.

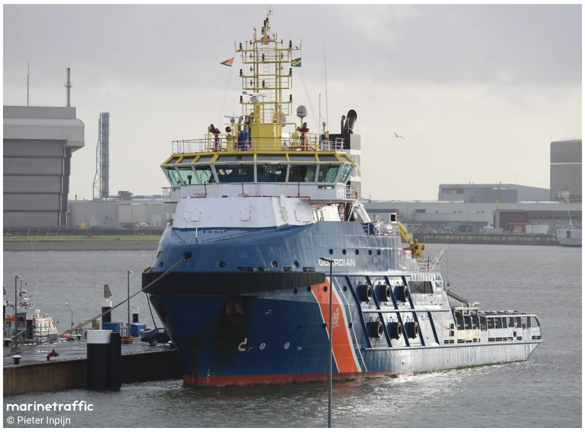


Figure 7.1: The Guardian, an offshore safety vessel used to collect ship motion data for testing the landing algorithm. Photo courtesy of Pieter Inpijn.

The main contributions of this chapter are as follows:

- We implemented and validated an LSTM-based model for ship motion prediction, trained on real ship motion data, capable of forecasting 3D velocity states.
- We developed an iterative, reconfigurable landing strategy based on a 3D feasible polynomial trajectory, ensuring smooth and reliable UAV landings.

The chapter is structured as follows. Section 7.2 describes the GPS RTK + IMU system used to estimate and record ship motion onboard. Section 7.3 details the LSTM network employed for ship motion prediction. Section 7.4 introduces the landing strategy and trajectory planning algorithm. Section 7.5 presents the results of the landing strategy, evaluated in a real-time simulation environment using recorded ship data and a dual-axis tilt-rotor quad-plane UAV model. Finally, Section 7.6 concludes the chapter with key findings and recommendations for future research.

7.2. ESTIMATION AND ACQUISITION OF SHIP MOTION DATA

During navigation, the ship experiences six degrees of freedom (6-Degrees Of Freedom (DOF)) motion, encompassing both angular and linear components. The angular motion, defined by roll, pitch, and yaw, occurs around the ship's center of gravity Center of Gravity (CG), as illustrated in Figure 7.2. The same figure also depicts the Earth reference frame, serving as a basis for describing the ship's linear motion. In this reference frame, the linear motion is represented by the components ${}^{e}X$, ${}^{e}Y$, and ${}^{e}Z$, corresponding to the ship's translation along the longitudinal, lateral, and vertical axes, respectively.

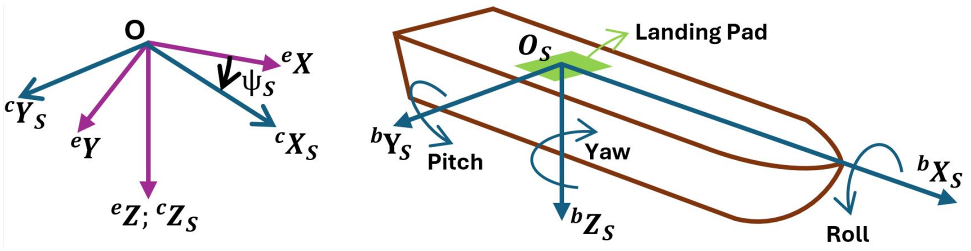


Figure 7.2: 6-DOF motion of a navigating ship, including the definition of the Ship Body, Ship Control and Earth reference frames. In this figure we assume the CG of the ship to be aligned with the landing pad.

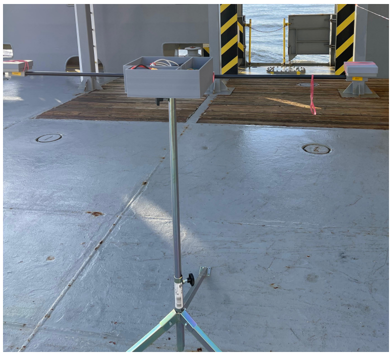
To collect and record ship motion data, a system based on a dual ZED F9P RTK-capable GPS receivers were employed. The GPS units were configured such that one transmitted RTCM correction data to the other, with a known positional offset. This configuration enhances the absolute accuracy of position estimation and enables precise determination of the ship's heading.

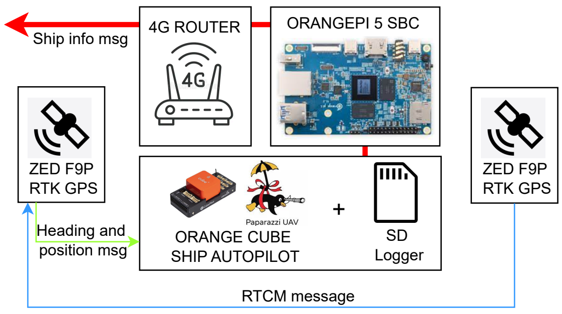
These data, combined with an IMU and a dedicated Extended Kalman Filter (EKF), provide an accurate estimation of the Euler angles and position of the ship, as well as the inertial velocity of the landing pad. The filter for estimating the ship's motion states was implemented on a commercial Orange Cube autopilot running the Paparazzi UAV software, specifically modified for ship-based deployment. The autopilot also logged the estimated ship motion states at a high refresh rate of 500 Hz and, through an intermediary Orange Pi 5 Single Board Computer (SBC), transmitted these estimations in real time to the UAV via a User Datagram Protocol (UDP) stream. The ship motion variables estimated by the system are described in Table 7.1.

A picture of the system is shown in Figure 7.3a, while Figure 7.3b presents a schematic diagram of the ship motion acquisition system.

Variable name	Description
${}^{c}\dot{X}_s$	Inertial velocity of the landing pad, in meters per second in the X-control reference frame.
${}^{c}\dot{Y}_s$	Inertial velocity of the landing pad, in meters per second in the Y-control reference frame.
${}^{c}\dot{Z}_s$	Inertial velocity of the landing pad, in meters per second in the Z-control reference frame.
Lat_{O_s}	Latitude of the landing pad in the LLA geodetic coordinate system, in degrees.
$Long_{O_s}$	Longitude of the landing pad in the LLA geodetic coordinate system, in degrees.
Alt_{O_s}	Altitude of the landing pad in the LLA geodetic coordinate system, in meters.
ϕ_s	Ship Roll angle in degrees.
θ_s	Ship Pitch angle in degrees.
ψ_s	Ship Yaw angle in degrees.

Table 7.1: Description of the variables estimated by the RTK GPS and IMU system onboard the ship during navigation.


(a) A picture of the RTK GPS and IMU system used to collect ship motion data onboard the Guardian vessel.



(b) Schematic of the RTK GPS and IMU system used for ship motion data collection and recording.

Figure 7.3: RTK GPS and IMU setup used for ship motion data collection. (a) Onboard installation of the system on the Guardian vessel. (b) Schematic representation of the system's configuration and data recording setup.

7.3. SHIP MOTION PREDICTION MODEL

Once the ship motion data is acquired, the next step in the landing process is predicting future ship motion states based on current observations. LSTM networks have proven to be an effective RNN structure for time series prediction. Unlike conventional RNNs, LSTMs mitigate the well-known issues of gradient vanishing and exploding during training while still capturing long-term dependencies in time series data [34].

At the core of the prediction model lies the LSTM cell, or neuron, whose structure is illustrated in Figure 7.4. Each LSTM cell consists of

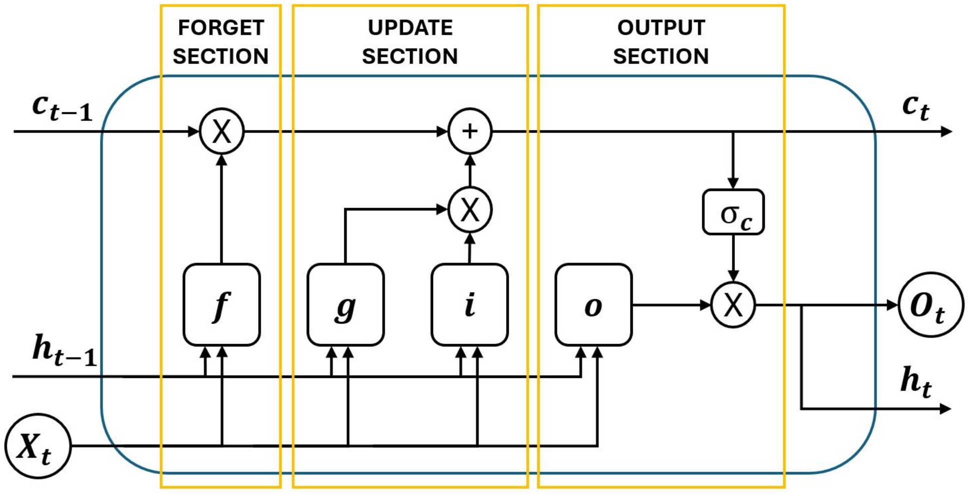


Figure 7.4: LSTM cell

a set of recurrent mathematical operations involving learnable weights and biases, which are tuned during the training process. The cell receives an input state, denoted as X_t and generates an output state and two internal states: the cell state c_t and the hidden state h_t, which are computed as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \tag{7.1}$$

$$h_t = f_t \odot \sigma_c(c_t). \tag{7.2}$$

where $\odot$ represents the Hadamard product (element-wise multiplication), and σ_c denotes the state activation function, defined as the hyperbolic tangent function:

$$\sigma_c(x) = (e^x - e^{-x})/(e^x + e^{-x}). \tag{7.3}$$

The remaining components of the LSTM cell, which regulate information flow, are defined as follows:

$$\begin{aligned} i_t &= \sigma_g(W_i x_t + R_i h_{t-1} + b_i), \\ f_t &= \sigma_g(W_f x_t + R_f h_{t-1} + b_f), \\ g_t &= \sigma_c(W_g x_t + R_g h_{t-1} + b_g), \\ o_t &= \sigma_g(W_o x_t + R_o h_{t-1} + b_o). \end{aligned} \tag{7.4}$$

Here, the subscripts *i, f, g* and *o* correspond to the input gate, forget gate, cell candidate gate, and output gate, respectively. The weights *W, R* and the biases *b* in equation 7.4 represent the learnable parameters that are adjusted during training. The gate activation function, denoted by σ_g is chosen to be a sigmoid function, defined as:

$$\sigma_g(x) = 1/(1+e^{-x}). \tag{7.5}$$

7.3.1. SHIP MOTION DATA SELECTION AND PREPARATION

The ship motion information message collected by the RTK GPS and IMU system, as defined in Table 7.1, must be stored and pre-processed to enable the training and evaluation of the prediction model. The primary objective of the motion prediction algorithm is to predict the x-y-z velocity components of the landing pad in the control reference frame. The control reference frame was selected for velocity estimation and prediction because, in this frame, and in the absence of ocean currents, the Y-axis velocity component of the landing pad remains small.

To minimize the training and evaluation time of the prediction model, we selected a subset of the estimated ship motion variables from Table 7.1 to be used during both the training phase and the evaluation phase. The selected time-discrete state variables are defined as:

$$\mathbf{X_p}(t) = \left[{}^{c}V_{xs}(t) \quad {}^{c}V_{ys}(t) \quad {}^{c}V_{zs}(t) \quad \phi_s(t) \quad \theta_s(t)\right]^T. \tag{7.6}$$

Once the ship motion array is defined, it must be normalized to ensure proper operation of the prediction model. The normalized ship motion array is given by:

$$\tilde{\boldsymbol{X}}_p(t) = \frac{\boldsymbol{X}_p(t) - \mu_{\boldsymbol{X}_p}}{\sigma_{\boldsymbol{X}_p}}, \quad \text{for } t \in [t-T, t] \tag{7.7}$$

where

$$\mu_{\boldsymbol{X}_p} = \frac{1}{T}\sum_{\tau=t-T}^{t} \boldsymbol{X}_p(\tau), \quad \sigma_{\boldsymbol{X}_p} = \sqrt{\frac{1}{T}\sum_{\tau=t-T}^{t} \left\|\boldsymbol{X}_p(\tau) - \mu_{\boldsymbol{X}_p}\right\|^2}$$

and *T* is the duration of the selected training window.

7.3.2. PREDICTION MODEL STRUCTURE, TRAINING AND EVALUATION

The prediction model was implemented in MATLAB using the Machine Learning Toolbox and consists of three main layers. The first layer is an input layer with a dimension of 5, equal to the size of the $\tilde{\mathbf{X}}_{\mathbf{p}}$ vector. The middle layer is an LSTM layer composed of a total of 50 LSTM cells, as defined in Figure 7.4. Finally, the last layer is a fully connected layer with an output size of 5, providing a one-sample prediction estimate of the ship motion state $\tilde{\mathbf{X}}_{\mathbf{p}}$. An overview of the prediction model is shown in Figure 7.5.

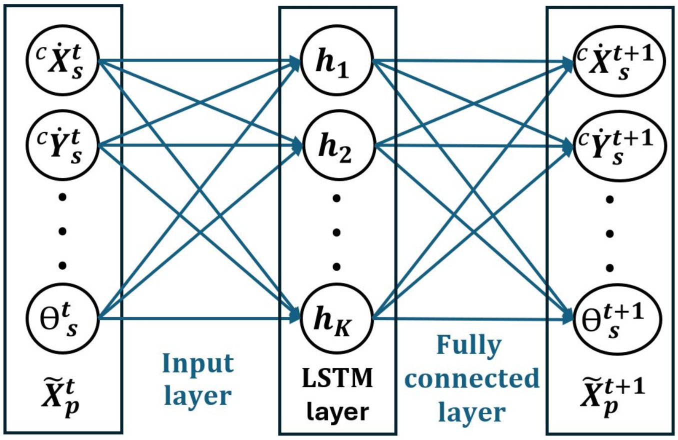


Figure 7.5: Prediction model implemented in MATLAB using the Machine Learning Toolbox.

The model presented in Figure 7.5 is then trained and evaluated using historical ship motion data collected by the RTK GPS and IMU system, following the scheme in Figure 7.6.

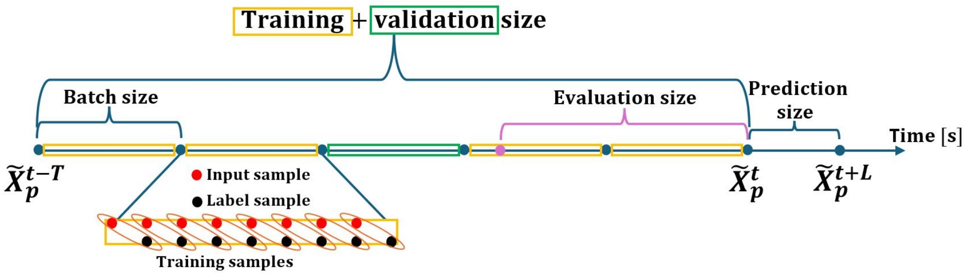


Figure 7.6: Selection of the time series for training and evaluation of the prediction model.

We adopted a training and validation window of 400 seconds, a batch size of 50 seconds, and an evaluation window of 150 seconds. The ship motion state was sampled at 0.2 second intervals, yielding five samples per second. The dataset was partitioned into training and validation sets, with four-fifths used for training and one-fifth randomly selected for validation. To ensure the model was trained on the most recent data, validation samples were prevented from including the final batch segment.

As illustrated in Figure 7.5, the prediction model generates one time step at a time. To extend this to a multi-step forecast, we employed a closed-loop prediction strategy, where each output is fed back as input for the next step. However, in this setup, accuracy tends to degrade progressively since the model operates on its own predictions rather than actual observations. To contain the impact of this compounding error, we limited the prediction horizon to 7 seconds, an interval that we considered sufficient for close-distance landing in our scenario.

The algorithm works as follows: initially, the model takes as input the most recent ship motion state array $\tilde{\boldsymbol{X}}_{\boldsymbol{p}}^{\boldsymbol{t}}$, which serves as the initial input for generating the first predicted sample. The LSTM hidden state S is updated accordingly, and the first prediction $\hat{X}$ is stored in the predicted sequence $\hat{X}_{\text{pred}}$. Then the prediction is extended to L seconds using the following process:

The algorithm proceeds as follows: it begins with the most recent ship motion state array, $\tilde{\boldsymbol{X}}_{\boldsymbol{p}}^{\boldsymbol{t}}$, as the initial input to generate the first prediction. The LSTM hidden state S is updated accordingly, and the result $\hat{X}$ is stored in the prediction sequence $\hat{X}_{\text{pred}}$. The remaining predictions are generated recursively through the following steps:

1. The current predicted state $\hat{X}_{\text{pred}}(t)$ is used as input to the prediction model.
2. The prediction model processes this input and outputs a new predicted state $\hat{X}$ while updating the hidden state S_{net}.
3. The new predicted state $\hat{X}$ is appended to the prediction sequence $\hat{X}_{\text{pred}}$, extending the forecast window by one step.
4. Steps 1-3 are repeated until the full prediction time window L is reached.

At the end of this process, the algorithm outputs the predicted sequence $\hat{X}_{\text{pred}}$, covering the entire L-second prediction horizon.

The training and validation batches were fed into MATLAB's Adam (Adaptive Moment Estimation) stochastic optimizer, which determined the optimal weights and biases for the prediction model. The mean square error (MSE) loss function was used, with training executed for a total of 250 epochs. Table 7.2 summarizes the hyper-parameters used to build and train the LSTM ship motion prediction model.

Parameter	Value	Description
Data sampling period	0.2 [s]	Time interval for ship motion data sampling.
Training and Validation size *T*	400 [s]	Historical ship motion data selected for training and validation.
Evaluation size	150 [s]	Time window for evaluating the trained prediction model.
Prediction size *L*	7 [s]	Duration of the prediction window.
Batch size	50 [s]	Size of the training and validation batches.
Training to validation ratio	5:1	Proportion of training to validation data.
LSTM cells *k*	50	Number of LSTM cells in the prediction model.
Training epochs	250	Number of epochs used for training.
Training optimizer	Adam	Adaptive Moment Estimation (Adam) optimizer used for training.
Loss function	MSE	Mean Square Error (MSE) loss function applied during training.
Learning rate	0.002	Rate at which the model adjusts its parameter at every optimization step.

Table 7.2: Hyper-parameters used in the training and evaluation of the LSTM ship motion prediction model.

7.3.3. PREDICTION MODEL ACCURACY AND PERFORMANCE

To evaluate the prediction accuracy and performance of the proposed model, it was implemented in MATLAB and tested on an off-the-shelf laptop equipped with an NVIDIA GeForce GTX 1650 Ti GPU with 4 GB of dedicated memory, an Intel i9-10885H CPU with a base frequency of 2.4 GHz, and 64 GB of RAM.

To simulate a real-time prediction scenario, the evaluation was performed using a multi-threaded MATLAB script. The first thread was responsible for continuously training the model, recursively updating the weights of the prediction network. The second thread focused on evaluating the model's performance and generating prediction results.

The prediction accuracy was initially evaluated using a pure sinusoidal input, as illustrated in Figure 7.7, where the reconstructed state obtained through the prediction algorithm is compared against the simulated signal. Subsequently, the model was validated using real-world ship motion data acquired from the RTK GPS + IMU system shown in

Figure 7.3a, installed onboard the *Guardian* vessel (Figure 7.1), which was operating under sea state 5 conditions. An overview of the model's performance on this real-world dataset is presented in Figure 7.8.

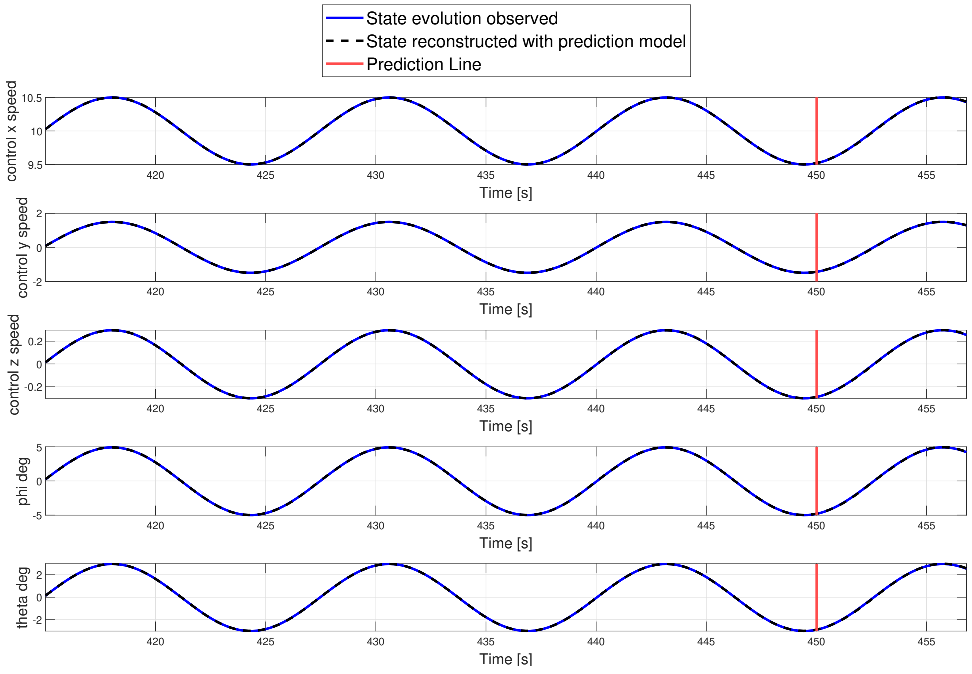


Figure 7.7: Prediction model running on a simulated sinusoidal ship motion input at time t=450 seconds.

To assess the model's accuracy, three commonly used evaluation metrics were employed. The first metric, Root Mean Square Error (RMSE), measures the standard deviation of the prediction errors and is defined as:

$$RMSE = \sqrt{\frac{1}{N_s}\sum_{i=1}^{N_s}(y_i - Y_i)^2}, \tag{7.8}$$

where y_i represents the predicted sample, Y_i is the observed sample, and N_s is the number of available predicted samples per inference.

The second metric, Mean Absolute Error (MAE), quantifies the average absolute difference between predicted and actual values:

$$MAE = \frac{1}{N_s}\sum_{i=1}^{N_s}|y_i - Y_i|. \tag{7.9}$$

The last metric, Root Mean Squared Percentage Error (RMSPE), expresses

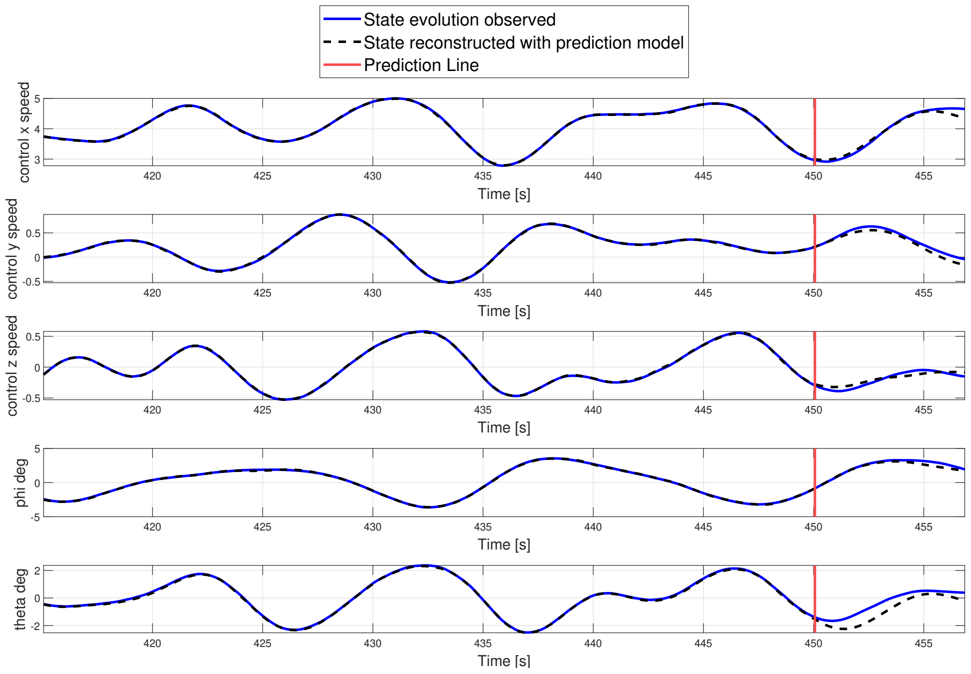


Figure 7.8: Prediction model running on real ship motion data collected by the RTK GPS and IMU system onboard the Guardian ship sailing under sea state 5 at time t=450 seconds.

the prediction error as a percentage of the actual values:

$$RMSPE = 100\% \sqrt{\frac{\sum_{i=1}^{N_s}(y_i - Y_i)^2}{\sum_{i=1}^{N_s}(y_i)^2}}. \tag{7.10}$$

It is important to note that RMSPE is sensitive to the magnitude of the observed signal. As a result, it may not always be the most appropriate metric for evaluating the absolute deviation between predictions and observations, particularly when dealing with signals of varying scales.

The prediction accuracy of the model, assessed using Equations 7.8, 7.9 and 7.10 is reported in Table 7.3 for a 100-second testing window. A visualization of the model prediction evolution on both the simulated sinusoidal input and the real-world ship motion data is provided in the accompanying video[1].

It is noteworthy that the prediction accuracy on real ship motion data, measured in terms of RMSE and MAE, consistently remains

[1]Prediction example with simulated sinusoidal and recorded ship motion: https://youtu.be/ESEuLAxQf6E

within one degree for attitude and well below 0.3 m/s for velocity components. However, due to the limited dynamic range of certain ship motion states—particularly lateral and vertical velocities, as well as attitude—the relative error reported by the RMSPE metric appears disproportionately high for these components.

Whether the achieved accuracy is adequate for the landing task ultimately depends on the strategy adopted for trajectory planning. As prediction uncertainty increases, the trajectory planner must compensate by re-evaluating the trajectory more frequently to avoid relying on outdated or inaccurate predictions.

Parameter	${}^{c}\dot{X}_s$ (m/s)	${}^{c}\dot{Y}_s$ (m/s)	${}^{c}\dot{Z}_s$ (m/s)	ϕ_s (deg)	θ_s (deg)
RMSE (first 3 sec)	0.14	0.10	0.09	0.62	0.39
RMSE (full 7 sec)	0.25	0.20	0.14	0.94	0.61
MAE (first 3 sec)	0.12	0.09	0.08	0.54	0.34
MAE (full 7 sec)	0.21	0.16	0.12	0.79	0.51
RMSPE (first 3 sec)	3.57%	52.37%	69.37%	65.52%	67.05%
RMSPE (full 7 sec)	6.11%	70.18%	83.11%	61.63%	82.79%

Evaluation Performance	
Number of training runs	18
Number of evaluation runs	376
Average training time	5.5 s
Average evaluation time	0.27 s

Table 7.3: Prediction model accuracy over a testing window of 100 seconds, including RMSE, MAE, and RMSPE metrics, using the recorded ship motion data sailing under sea state 5.

7.3.4. IMPLEMENTATION OF THE PREDICTION MODEL ON THE UNMANNED AERIAL SYSTEM (UAS)

To minimize the amount of data transmitted to the UAV, the full time series of predicted ship motion states is not sent. Instead, only the three components of the predicted landing pad velocity are selected. These samples are then approximated using a seventh-order polynomial fitted via a least squares procedure. This approach significantly reduces bandwidth requirements, as only the polynomial coefficients need to be transmitted rather than the full set of velocity samples. Additionally, it provides a smooth estimate that can be analytically differentiated or integrated to obtain both inertial acceleration and position predictions of the landing pad.

The Euler angles included in the full ship motion model are not used at this stage, as they were only incorporated into the prediction model to enhance performance by retaining information about the ship motion.

To facilitate onboard computation, the polynomial coefficients of the landing pad velocity components are transmitted in the Earth reference frame rather than the ship control reference frame. The transformation between these two frames is given by:

$$
\begin{aligned}
{}^{\boldsymbol{e}}\hat{\boldsymbol{V}}_{\boldsymbol{s}}(t) &= R_{ec}\,{}^{\boldsymbol{c}}\hat{\boldsymbol{V}}_{\boldsymbol{s}}(t) = R_{ec}\begin{pmatrix} {}^{c}\hat{V}_{xs}(t) \\ {}^{c}\hat{V}_{ys}(t) \\ {}^{c}\hat{V}_{zs}(t) \end{pmatrix}; \\
R_{ec} &= \begin{bmatrix} \cos(\psi_{s_0}) & \sin(\psi_{s_0}) & 0 \\ -\sin(\psi_{s_0}) & \cos(\psi_{s_0}) & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\end{aligned}
\tag{7.11}
$$

where ψ_{s_0} is the current ship yaw angle , and the hat symbol indicates an estimated prediction. Here, we assume that the yaw remains constant throughout the prediction window.

The polynomial function approximating the landing pad velocity prediction in the Earth reference frame is therefore given by:

$$
\boldsymbol{\gamma}(t) = \begin{pmatrix} \sum_{k=0}^{7} s_{1+k}\, t^{7-k} \\ \sum_{k=0}^{7} s_{9+k}\, t^{7-k} \\ \sum_{k=0}^{7} s_{17+k}\, t^{7-k} \end{pmatrix} \approx {}^{\boldsymbol{e}}\hat{\boldsymbol{V}}_{\boldsymbol{s}}(t), \tag{7.12}
$$

where s_i are the polynomial coefficients obtained from the least squares fit applied to the predicted time series of ${}^{\boldsymbol{e}}\hat{\boldsymbol{V}}_{\boldsymbol{s}}(t)$, and t represents time with a domain of $t \in [0, L]$.

Due to evaluation and transmission delays, by the time the prediction reaches the UAV, it may already be outdated. To compensate for this, the prediction must be adjusted using the most recent ship motion states. The correction is applied by modifying only the zero-degree polynomial term (bias) to align its initial value with the latest observed ship motion state. The corrected estimate is given by:

$$
\begin{aligned}
\boldsymbol{\gamma}_{\boldsymbol{c}}(t) &= \boldsymbol{\gamma}(t + \Delta t_d) + {}^{e}\boldsymbol{\Delta V}_{\boldsymbol{s}}, \\
\textit{where} \quad {}^{e}\boldsymbol{\Delta V}_{\boldsymbol{s}} &= \boldsymbol{\gamma}(\Delta t_d) - {}^{\boldsymbol{e}}\boldsymbol{V}_{\boldsymbol{s}}(0).
\end{aligned}
\tag{7.13}
$$

where ${}^{\boldsymbol{e}}\boldsymbol{V_s}(0)$ represents the latest observed ship velocity vector in the Earth reference frame, and Δt_d accounts for the combined evaluation and transmission delay.

After applying this correction, $t = 0$ now represents the actual current time, rather than the time at which the prediction was originally computed. Consequently, the new domain is $t \in [0, L - \Delta t_d]$.

From the corrected velocity estimate polynomials, the position and acceleration predictions of the landing pad are obtained as:

$$ {}^{\boldsymbol{e}}\hat{\boldsymbol{P}}_{\boldsymbol{s}}(t) \approx {}^{\boldsymbol{e}}\boldsymbol{P_s}(0) + \int_0^t \boldsymbol{\gamma_c}(\tau)\, d\tau. \tag{7.14} $$

$$ {}^{\boldsymbol{e}}\hat{\boldsymbol{A}}_{\boldsymbol{s}}(t) \approx \frac{\partial \boldsymbol{\gamma_c}}{\partial t}. \tag{7.15} $$

where ${}^{\boldsymbol{e}}\boldsymbol{P_s}(0)$ represents the latest observed ship position vector in the Earth reference frame.

An example of the landing pad forward velocity in the ship control reference frame, including the polynomial fit and the real-time adjustment for a 1-second evaluation and transmission delay, is shown in Figure 7.9.

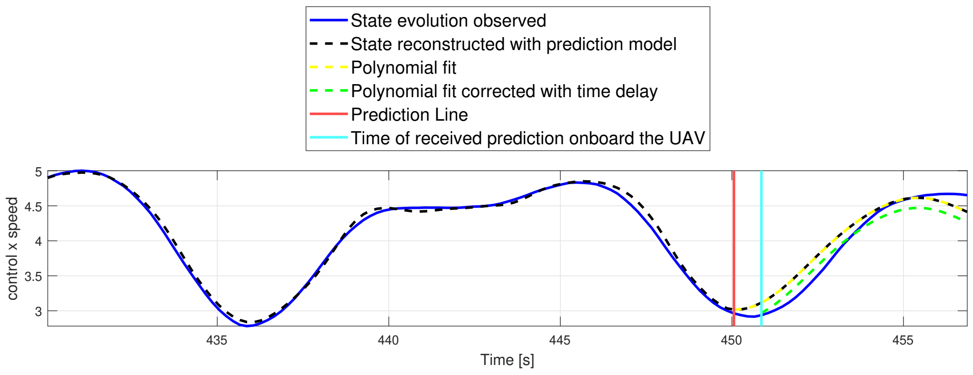


Figure 7.9: Close-up view of the landing pad forward velocity prediction in the ship control reference frame, with the applied polynomial fit used for transmission to the UAV. A 1-second delay correction has been applied to the signal. Note that, prior to 450 seconds, the black dotted line represents a one-timestep-ahead prediction of the ship's velocity.

7.4. LANDING STRATEGY AND TRAJECTORY PLANNING

In the previous sections, we focused on the estimation and prediction of the ship motion. This section shifts the focus to the UAV aspects, detailing the strategy developed to guide the UAV from cruise conditions to safe landing. The landing strategy is structured as a state machine that generates an inertial velocity target vector $\mathbf{v_t}$ from cruise flight until touchdown. An overview of the landing strategy is provided in Figure 7.10. The approach consists of four main stages, transitioning from cruise conditions to touchdown:

1. **Approach Mode:** The UAV begins by approaching a predefined holding point $\mathbf{P_H}$, defined by the operator and visible in Figure 7.10, along a line.

2. **Holding Point Hovering Mode:** Once the UAV reaches a distance of 6 meters or less from $\mathbf{P_H}$, it enters a hovering state at the holding point.

3. **Landing Trajectory Tracking:** If a feasible landing trajectory is found, the UAV tracks the generated landing trajectory.

4. **Flare Mode and Touchdown:** If the UAV's relative altitude is below 0.5 meters, it initiates a flare maneuver until a safe landing is achieved.

The following subsections provide a detailed analysis of each landing mode.

7.4.1. APPROACH MODE

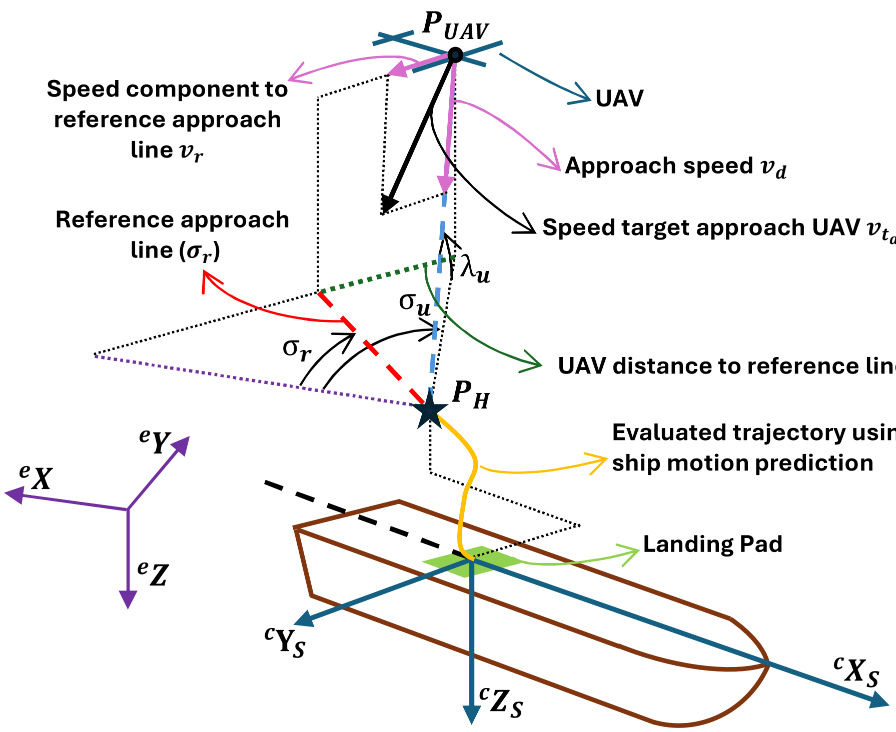
The first phase of the landing strategy consists of an approach to a predefined holding point, denoted as $\mathbf{P_H}$. This point has fixed coordinates relative to the ship and moves with it as a rigid body. The location of the $\mathbf{P_H}$ can therefore be adjusted based on obstacles or specific operational requirements. Typically, to minimize gust-induced perturbations, this point is positioned toward the stern and slightly to the side of the ship, ensuring a safe altitude. In our case, the holding point $\mathbf{P_H}$ is defined as:

$$ {}^{c}\mathbf{P_H} = \begin{pmatrix} -5 \\ -3 \\ -5 \end{pmatrix} \tag{7.16}$$

where the values are expressed in meters, and the superscript c indicates that the position is defined in the ship's control reference frame, as illustrated in Figure 7.2, with its origin located at the landing pad.

In this approach mode, the velocity target vector for the UAV is computed as:

$$ {}^{e}\mathbf{V_t} = {}^{e}\bar{\mathbf{V}}_\mathbf{s} + {}^{e}\mathbf{V_d} + {}^{e}\mathbf{V_r}. \tag{7.17}$$



P_{UAV}
UAV
Speed component to reference approach line v_r
Approach speed v_d
Speed target approach UAV v_{t_a}
Reference approach line (σ_r)
λ_u
σ_u
σ_r
UAV distance to reference line D_l
P_H
Evaluated trajectory using ship motion prediction
Landing Pad
${}^eY$
${}^eX$
${}^eZ$
${}^cY_S$
${}^cZ_S$
${}^cX_S$

Figure 7.10: Landing strategy scheme.

Where the first component, ${}^e\bar{\mathbf{V}}_\mathbf{s}$ represents the filtered average inertial velocity of the ship in the earth reference frame. This vector has zero vertical speed component, while its horizontal speed components are smoothed using a moving average filter to eliminate the oscillatory ship motion component. This element of the velocity target vector ensures that the UAV follows the ship horizontal velocity during the approach.

The second component of Equation 7.17, ${}^e\mathbf{V}_\mathbf{d}$, is the velocity which brings the UAV towards the moving holding point $\mathbf{P}_\mathbf{H}$ and is defined as follows:

$$ {}^e\mathbf{V}_\mathbf{d} = v_d \begin{pmatrix} cos(\lambda_u)sin(\sigma_u) \\ cos(\lambda_u)cos(\sigma_u) \\ sin(\lambda_u) \end{pmatrix}. \tag{7.18}$$

With σ_u and λ_u being the current UAV azimuth and elevation angle in the earth reference frame and v_d being the desired diagonal approach speed.

The third component of Equation 7.17, ${}^e\mathbf{V}_\mathbf{r}$, is a velocity component aimed at bringing the UAV towards the approach line. The approach line is identified by its azimuth angle σ_r, measured with respect to true north. The velocity component ${}^{e}\mathbf{V_r}$, which corrects lateral deviation from the reference approach line, is defined as follows:

$$
{}^{e}\mathbf{V_r} = G \cdot D_l \begin{pmatrix} \sin(\sigma_u) \\ -\cos(\sigma_u) \\ 0 \end{pmatrix}. \tag{7.19}
$$

Where D_l is the UAV distance to the reference diagonal line, defined as:

$$
D_l = |\mathbf{P_H} - \mathbf{P_{UAV}}| \cdot \cos(\lambda_u) \cdot \sin(\sigma_u - \sigma_l), \tag{7.20}
$$

while G is the line speed gain and is chosen to be proportional to the reciprocal of D_l, with a saturation to prevent it from approaching infinity as the UAV nears the holding point $\mathbf{P_H}$:

$$
G = \min\left(G_{max}, \frac{K_g}{|\mathbf{P_H} - \mathbf{P_{UAV}}| \cdot \cos(\lambda_u)}\right). \tag{7.21}
$$

In our case, G_{max} was set to 0.5, while K_g was chosen as 6. This gain structure prevents excessive corrections when the UAV is far from the holding point $\mathbf{P_H}$, as $\mathbf{P_H}$ moves with the rigid body of the ship and is highly sensitive to small heading changes.

7.4.2. HOLDING POINT HOVERING MODE

Upon reaching a distance of 6 meters or less from the holding point $\mathbf{P_H}$, the UAV transitions to hovering mode. In this mode, the UAV velocity target vector is defined as:

$$
{}^{e}\mathbf{V_t} = K_{p_h}(\mathbf{P_H} - \mathbf{P_{UAV}}) + {}^{e}\bar{\mathbf{V}}_\mathbf{s}. \tag{7.22}
$$

Where K_{p_h} is an hovering proportional gain that ensures smooth convergence to the holding point. In our case, K_{p_h} is set to 1.

7.4.3. FEASIBLE LANDING TRAJECTORY GENERATION AND TRACKING USING THE SHIP MOTION PREDICTION MODEL

While the UAV remains in the holding point hovering mode, the trajectory planning algorithm is activated. This algorithm continuously iterates using the ship motion prediction model to identify a feasible landing trajectory. A landing trajectory is considered feasible if it can bring the UAV to a safe touchdown while ensuring that the required accelerations and velocities remain within the vehicle's operational limits. The algorithm developed to generate this landing trajectory is outlined in Algorithm 1.

The function *ProduceTrajectory*() in Algorithm 1 generates a fifth-order polynomial trajectory that connects the UAV's current position,

```
Algorithm 1: Trajectory planning algorithm
  Input: Max prediction window $L-\Delta_{t_d}$, ship motion prediction
         coefficients $\boldsymbol{\gamma_c(t)}$, current positions, velocities, and
         accelerations of the ship and UAV and the UAV velocity and
         acceleration limits expressed in the control reference
         frame.
  Output: Trajectory feasibility flag $\chi_{found}$, estimated landing time
          $T_{land}$ and trajectory polynomial coefficients $\boldsymbol{\chi(t)}$
1 $\chi_{found} = 0$ // Initialize trajectory feasibility flag
2 while $\chi_{found} == 0$ and $T_{land} \leq L-\Delta_{t_d}$ do
3     $\boldsymbol{\chi(t)}$ = ProduceTrajectory()
4     if IsFeasible($\boldsymbol{\chi(t)}$) then
5         $\chi_{found} = 1$ // Verify if the trajectory satisfies the UAV constraints
6     else
7         $T_{land} = T_{land} + 0.1$ // Increase landing time by 0.1s and retry
8 return $\chi_{found}, T_{land}, \boldsymbol{\chi(t)}$
```

velocity, and acceleration ${}^e\mathbf{P_{UAV}}(0)$, ${}^e\mathbf{V_{UAV}}(0)$, ${}^e\mathbf{A_{UAV}}(0)$ to the predicted landing pad state at T_{land}, denoted as ${}^e\mathbf{\hat{P}_s}(T_{land})$, ${}^e\mathbf{\hat{V}_s}(T_{land})$, ${}^e\mathbf{\hat{A}_s}(T_{land})$. The resulting trajectory is expressed as:

$$\boldsymbol{\chi(t)} = \begin{pmatrix} {}^eX_{trajectory}(t) \\ {}^eY_{trajectory}(t) \\ {}^eY_{trajectory}(t) \end{pmatrix} = \begin{pmatrix} \sum_{k=0}^{7} m_{1+k}\, t^{7-k} \\ \sum_{k=0}^{7} m_{9+k}\, t^{7-k} \\ \sum_{k=0}^{7} m_{17+k}\, t^{7-k} \end{pmatrix}, \tag{7.23}$$

subject to the boundary conditions:

$$\begin{cases} \boldsymbol{\chi}(0) = {}^e\mathbf{P_{UAV}}(0) \\ \dot{\boldsymbol{\chi}}(0) = {}^e\mathbf{V_{UAV}}(0) \\ \ddot{\boldsymbol{\chi}}(0) = {}^e\mathbf{A_{UAV}}(0) \\ \boldsymbol{\chi}(T_{land}) = {}^e\mathbf{\hat{P}_s}(T_{land}) \\ \dot{\boldsymbol{\chi}}(T_{land}) = {}^e\mathbf{\hat{V}_s}(T_{land}) \\ \ddot{\boldsymbol{\chi}}(T_{land}) = {}^e\mathbf{\hat{A}_s}(T_{land}) \end{cases}. \tag{7.24}$$

Where m_i are the polynomial coefficients defining the landing trajectory, and $\boldsymbol{\chi(t)}$ represents the landing trajectory function over the domain $t \in [0, T_{land}]$. Note that the trajectory if fully determined with the boundary conditions in Equation 7.24.

The function *IsFeasible*() in Algorithm 1 evaluates whether the generated trajectory $\boldsymbol{\chi}(\boldsymbol{t})$ adheres to the UAV's operational constraints by comparing its speed and acceleration profiles against the UAV's maximum allowable limits. To streamline this verification process, the UAV's acceleration and speed limits are expressed and assessed in the UAV's control reference frame.

An example of a trajectory generated using Algorithm 1 is illustrated in Figure 7.11.

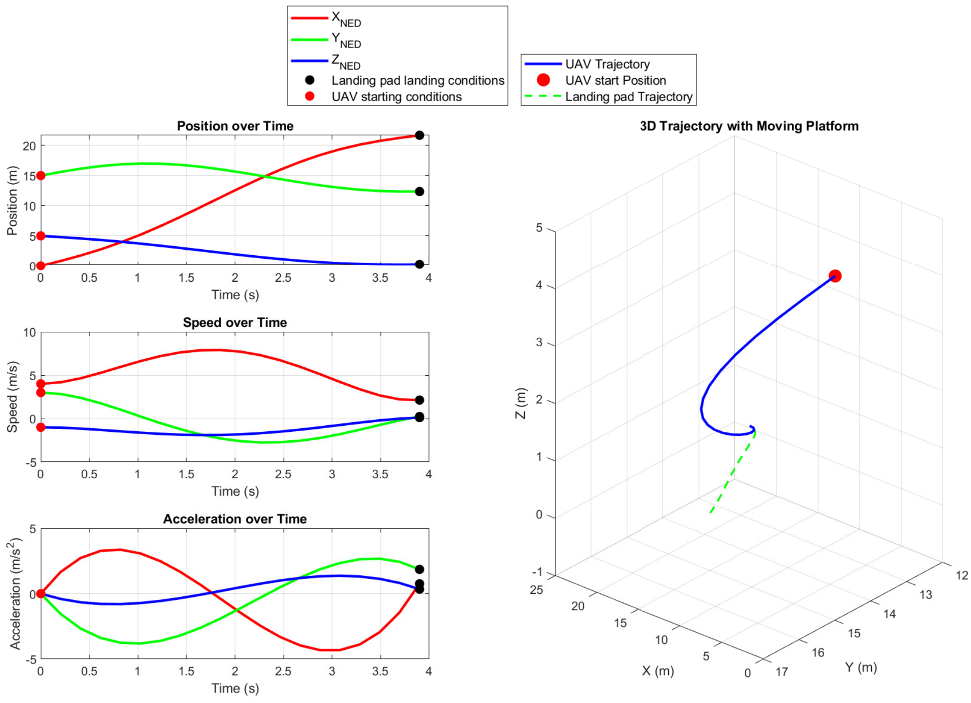


Figure 7.11: Generation of a feasible trajectory from the UAV position towards the landing pad using a simulated ship motion.

Once the algorithm determines an initial landing trajectory, the UAV computes its target velocity vector as:

$$^{e}\mathbf{V_t}(t) = K_{p_p}(\boldsymbol{\chi}(\boldsymbol{t}) - \mathbf{P_{UAV}}(t)) + \dot{\boldsymbol{\chi}}(\boldsymbol{t}). \tag{7.25}$$

where K_{p_p} is the proportional trajectory tracking gain, chosen as 5 in our case.

To mitigate inaccuracies in landing trajectory prediction and tracking, the trajectory planning algorithm in Algorithm 1 is executed multiple times throughout the tracking phase. The algorithm is re-evaluated every $T_{land}/2$ seconds using the latest available prediction and ship motion data. This ensures a higher update frequency as the UAV approaches the landing pad.

If at any point during trajectory tracking a feasible trajectory cannot be determined, the UAV is instructed to return to the holding point $\mathbf{P_H}$ and await a new valid trajectory.

Under normal conditions, the computed trajectory brings the UAV to the landing pad at a velocity and acceleration matching those of the ship at T_{land}. Finally, when the altitude above the deck falls below 0.5 m, the drone enters the flare maneuver.

7.4.4. FLARE MANEUVER

When the UAV's relative altitude falls below 0.5 meters, it transitions into the flare maneuver mode. In this phase, the UAV is required to track the following velocity vector:

$$^{e}\mathbf{V_t} = {}^{e}\mathbf{V_s} + \begin{pmatrix} 0 \\ 0 \\ v_{\text{touch-down}} \end{pmatrix}. \tag{7.26}$$

Where $v_{\text{touch-down}}$ represents the desired vertical descent speed at which the UAV makes contact with the landing pad. For our application, we set this value to 0.5 m/s to ensure a controlled touchdown.

For overactuated vehicles, such as the dual-axis tilting rotor quad-plane shown in Figure 4.1, the UAV is additionally required to align its attitude angles with the ship's pitch θ_s and roll ϕ_s to maintain stability during landing.

In this final landing phase, position control is deliberately de-prioritized to prevent the UAV from sliding on the landing pad. Instead, the focus remains on matching relative velocity and attitude, reducing the risk of tipping over due to a speed mismatch between the UAV and the landing surface.

7.5. SIMULATION RESULTS

To evaluate the effectiveness of the landing strategy, the proposed algorithm was tested in a high-fidelity simulation environment developed primarily in MATLAB and Simulink.

7.5.1. SIMULATION FRAMEWORK

The simulation framework consists of three main components: a MATLAB script implementing the LSTM-based ship motion prediction model, a MATLAB script responsible for loading and transmitting pre-recorded ship motion logs, and a Simulink model featuring an accurate dual-axis tilt-rotor quad-plane model along with its flight controller. An overview of the simulation architecture is shown in Figure 7.12.

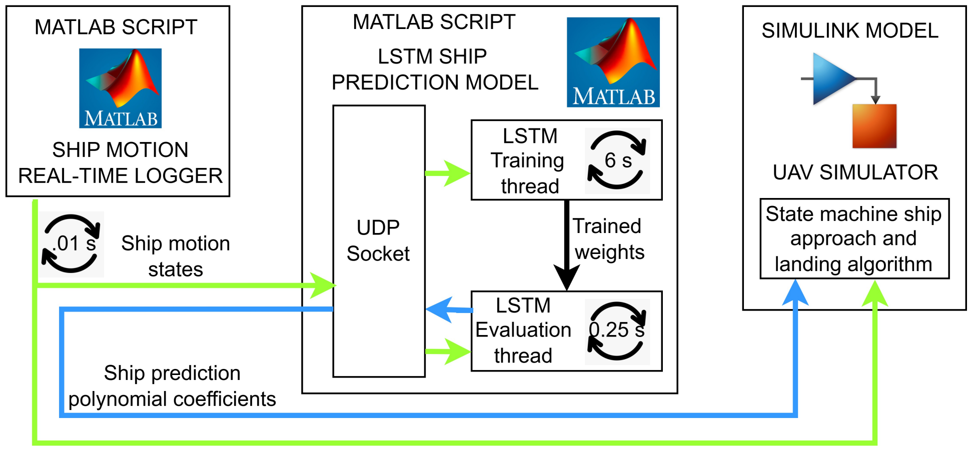


Figure 7.12: Simulation architecture to test the landing algorithm.

LSTM SHIP PREDICTION MODEL MATLAB SCRIPT

The LSTM ship prediction model MATLAB script receives real-time ship motion states via a UDP socket. These states are then passed to an internal thread responsible for training the LSTM prediction model. This training thread runs at an average interval of 6 seconds, producing updated neural network weights and biases.

Once the training process is completed, the trained network is forwarded to another internal thread, along with the latest received motion states, to generate a polynomial fit of the predicted ship motion time series, as described in Section 7.3.4. The polynomial coefficients of the ship motion prediction $\gamma(t)$ are then transmitted through another UDP socket, making them accessible to other components of the simulation framework. This thread runs at an average frequency of 4 Hz.

SHIP MOTION LOGGER MATLAB SCRIPT

This MATLAB script is responsible for retrieving and transmitting ship motion data recorded onboard the Guardian ship while sailing in sea state 5. The ship motion vector, as described in Table 7.1, is transmitted via a UDP port at a frequency of 100 Hz. The script is designed to simulate real-time operations on the ship, ensuring that messages are transmitted in a time-synchronized manner to replicate actual conditions.

UAV SIMULATION SIMULINK MODEL

This Simulink model simulates the state evolution of the dual-axis tilting rotor quad-plane depicted in Figure 4.1 using the flight controller streamlined in our previous work [35]. The high-fidelity 6-DOF model of the vehicle was developed based on wind tunnel experiments and flight test data, incorporating a wind model and rotor-wing interactions [36].

For wind turbulence modeling, we utilized the continuous Dryden wind turbulence model available in the Simulink Aerospace Blockset. The properties of this model are thoroughly described in the official documentation[2]. The block was configured with the physical properties of the UAV, modifying only the *wind intensity at 6 meters* and the *predominant wind direction* properties. This setup allowed us to evaluate the landing algorithm under various wind gust conditions.

7.5.2. RESULTS

To evaluate the landing algorithm, the simulation framework depicted in Figure 7.12 was executed multiple times under varying wind gust conditions, using real ship motion logs recorded onboard the Guardian ship while sailing in the North Sea under sea state 5.

The simulations were conducted with the Dryden gust block *wind intensity at 6 meters* parameter set to 3, 6, 9, 12, and 15 m/s. For each wind speed scenario, 10 different values for the Dryden gust block *predominant wind direction* covering all 360 degrees. The results of this simulation campaign are presented in Table 7.4, where, for each wind speed intensity, the mean, standard deviation, and maximum values of several key landing parameters are reported. The relevant performance metrics include horizontal position offset, horizontal speed offset, vertical speed offset with respect to the required landing descent speed, and attitude angle offset.

All offsets are computed as the difference between the UAV's landing condition and the ship's motion state at the time of touchdown. The results indicate that wind has a limited impact on the landing performance, with the UAV always landing within 0.45 meters from the center of the landing pad. The maximum observed horizontal speed offset is 0.16 m/s, while the maximum vertical speed offset is 0.25 m/s.

Figure 7.13 shows the UAV and ship trajectories under a predominant wind speed of 15 m/s and a relative wind direction of 16 degrees with respect to the ship's heading, from the moment the landing algorithm is engaged until touchdown. The figure highlights the capability of both the vehicle and the algorithm to successfully manage strong wind conditions.

Another notable scenario occurred with a predominant wind speed of 3m/s and relative wind direction of 160 degrees, as illustrated in

[2]Dryden wind turbulence model: https://shorturl.at/fBQkT

Figure7.14. Among all simulation runs, this was the only case in which the UAV aborted the landing during the trajectory tracking phase. The vehicle returned to the holding point and re-entered hovering mode before initiating a new trajectory planning sequence, which ultimately resulted in a successful landing. The abort was primarily caused by an initially inaccurate ship motion prediction combined with UAV tracking deviations, which made the originally planned trajectory infeasible. This behavior further underscores the robustness of the landing algorithm and its ability to adapt and recover from unexpected disturbances.

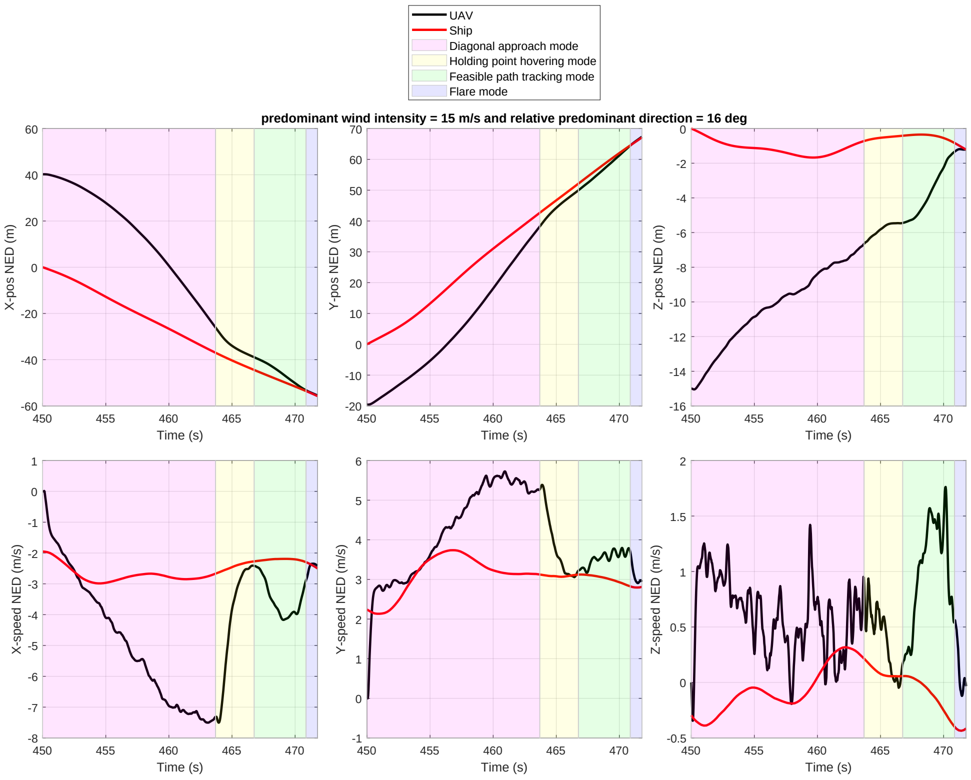


Figure 7.13: Simulation output with wind intensity of 15 m/s and predominant wind direction with respect of the ship's heading of 16 degrees.

7.6. CONCLUSION

This study presents an integrated approach for UAV landing on a moving vessel in high sea states, focusing on ship motion prediction and real-time trajectory planning. The proposed method combines

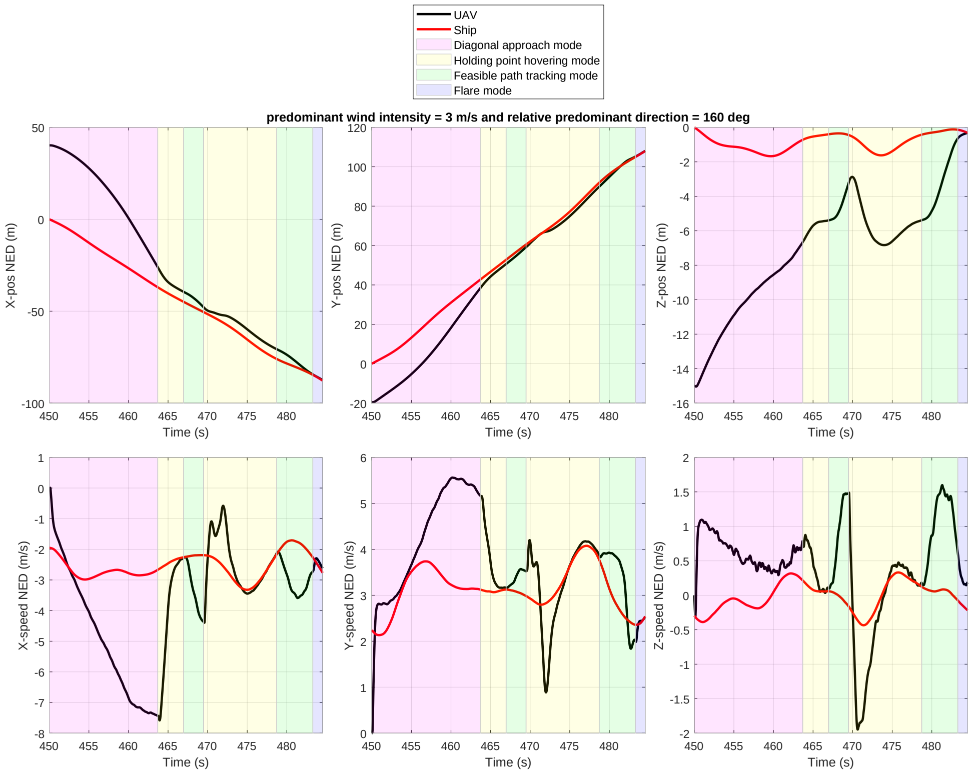


Figure 7.14: Simulation output with wind intensity of 3 m/s and predominant direction with respect of the ship's heading of 160 degrees. This is the only simulation run in which the UAV aborted the landing during the trajectory tracking mode, returned to the holding point hovering mode, and then re-initiated a new trajectory planning maneuver leading to a successful landing.

an LSTM-based prediction model with an iterative trajectory planning strategy, ensuring that the UAV can land safely while adhering to operational constraints. The algorithm was validated using real-world ship motion data from an offshore safety vessel, demonstrating its ability to predict ship movements up to seven seconds ahead and generate feasible landing trajectories. The results indicate that the approach remains effective even under challenging wind conditions, with minimal landing offsets and the capability to adapt dynamically to changing ship motion.

Despite these promising results, several aspects require further refinement. One key area for improvement is the enhancement of prediction accuracy. While the LSTM-based ship motion prediction model

Metric	Predominant Wind Intensity (m/s)				
	3	6	9	12	15
Mean land offset horizontal distance (m)	0.358	0.350	0.355	0.315	0.297
Std land offset horizontal distance (m)	0.045	0.049	0.041	0.047	0.081
Max land offset horizontal distance (m)	0.438	0.429	0.406	0.379	0.390
Mean land offset horizontal speed (m/s)	0.074	0.080	0.100	0.095	0.096
Std land offset horizontal speed (m/s)	0.029	0.024	0.027	0.032	0.036
Max land offset horizontal speed (m/s)	0.150	0.136	0.146	0.135	0.158
Mean land offset vertical speed (m/s)	0.036	0.047	0.053	0.086	0.068
Std land offset vertical speed (m/s)	0.026	0.029	0.030	0.091	0.072
Max land offset vertical speed (m/s)	0.104	0.081	0.115	0.244	0.235
Mean land offset attitude (deg)	2.113	2.085	1.882	1.675	1.623
Std land offset attitude (deg)	0.216	0.513	0.614	0.835	0.954
Max land offset attitude (deg)	2.608	2.770	2.848	2.737	3.608

Table 7.4: Landing Performance Metrics Under Various Wind Conditions.

performs well, integrating alternative architectures, such as Transformer-based networks or hybrid models combining LSTM with other ship motion prediction approaches, could provide more robust predictions under extreme sea states. Another important step is optimizing the system for real-time execution on UAV-embedded processors. The current implementation was tested in a simulated environment using consumer-grade hardware, and deploying it on an onboard computational system will be crucial for practical applications.

Future work should also focus on conducting real-world experiments to validate the system beyond simulation environments. Although the use of real ship motion data in the simulations provides a strong foundation, actual UAV landing trials on moving vessels are necessary to confirm performance under real operational conditions. Additionally, refining the system's wind compensation strategies could further enhance landing precision, particularly in highly turbulent environments where ship motion becomes more unpredictable.

Another avenue for advancement lies in the integration of vision-based landing aids. Currently, the landing strategy relies heavily on datalink communication and GPS, but combining this approach with onboard visual tracking, such as fiducial markers or infrared detection, could further refine the UAV's ability to land accurately, even in case of loss of GPS or datalink communication. Addressing these challenges will contribute to the development of a more reliable and adaptable UAV landing system for maritime operations, improving both safety and operational efficiency in high sea state environments.

REFERENCES

[1] J. Ross, M. Seto, and C. Johnston. “Autonomous Landing of Rotary Wing Unmanned Aerial Vehicles on Underway Ships in a Sea State”. In: *Journal of Intelligent Robot System* 104.1 (2022). doi: 10.1007/s10846-021-01515-x.

[2] A. E. Baitis. “The Influence of Ship Motions on Operations of SH-2F Helicopters from DF-1052-Class Ships: Sea Trial with USS Bowen”. In: *USA gov report* (1975).

[3] P. M. Gupta, E. Pairet, T. Nascimento, and M. Saska. “Landing a UAV in Harsh Winds and Turbulent Open Waters”. In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 744–751. doi: 10.1109/LRA.2022.3231831.

[4] D. N. Tri and S. Cornel. “Nonlinear Helicopter and Ship Models for Predictive Control of Ship Landing Operations”. In: *AIAA Guidance, Navigation, and Control Conference* (2014). doi: 10.2514/6.2014-1298.

[5] S. Lee, J. Lee, S. Lee, H. Choi, Y. Kim, S. Kim, and J. Suk. “Sliding Mode Guidance and Control for UAV Carrier Landing”. In: *IEEE Transactions on Aerospace and Electronic Systems* 55.2 (2019), pp. 951–966. doi: 10.1109/TAES.2018.2867259.

[6] Y. Meng, W. Wang, H. Han, and J. Ban. “A visual/inertial integrated landing guidance method for UAV landing on the ship”. In: *Aerospace Science and Technology* 85 (2019), pp. 474–480. issn: 1270-9638. doi: 10.1016/j.ast.2018.12.030.

[7] R. Polvara, S. Sharma, J. Wan, A. Manning, and R. Sutton. “Vision-Based Autonomous Landing of a Quadrotor on the Perturbed Deck of an Unmanned Surface Vehicle”. In: *Drones* 2.2 (2018). doi: 10.3390/drones2020015.

[8] G. Xu, Y. Zhang, S. Ji, Y. Cheng, and Y. Tian. “Research on computer vision-based for UAV autonomous landing on a ship”. In: *Pattern Recognition Letters* 30.6 (2009), pp. 600–605. issn: 0167-8655. doi: 10.1016/j.patrec.2008.12.011.

[9] A. D. Jordan, M. Rydalch, T. McLain, and M. Williamson. “Precision Maritime Localization and Landing with Real-time Kinematic GNSS”. In: *AIAA SCITECH 2023 Forum* (2023). doi: 10.2514/6.2023-0488.

[10] K. Wenzel, A. Masselli, and A. Zell. “Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle”. In: *Journal of Intelligent Robot System* 61 (2010). doi: 10.1007/s10846-010-9473-0.

[11] K. Xia, M. Shin, W. Chung, M. Kim, S. Lee, and H. Son. “Landing a quadrotor UAV on a moving platform with sway motion using robust control”. In: *Control Engineering Practice* 128 (2022), p. 105288. doi: 10.1016/j.conengprac.2022.105288.

[12] N. P. Santos, V. Lobo, and A. Bernardino. “A ground-based vision system for UAV tracking”. In: *OCEANS 2015 - Genova*. 2015, pp. 1–9. doi: 10.1109/OCEANS-Genova.2015.7271349.

[13] R. Polvara, S. Sharma, J. Wan, A. Manning, and R. Sutton. “Towards autonomous landing on a moving vessel through fiducial markers”. In: *2017 European Conference on Mobile Robots (ECMR)*. 2017, pp. 1–6. doi: 10.1109/ECMR.2017.8098671.

[14] P. Chiranjeev, R. Rahul, S. Ritwik, A. A. Jitendra Singh, and K. S. Venkatesh. “Vision-Based Autonomous Ship Deck Landing of an Unmanned Aerial Vehicle Using Fractal ArUco Markers”. In: *AIAA SCITECH 2025 Forum* (2025). doi: 10.2514/6.2025-2345.

[15] G. Cho, J. Choi, G. Bae, and H. Oh. “Autonomous ship deck landing of a quadrotor UAV using feed-forward image-based visual servoing”. In: *Aerospace Science and Technology* 130 (2022), p. 107869. issn: 1270-9638. doi: 10.1016/j.ast.2022.107869.

[16] G. Anitha and R. G. Kumar. “Vision Based Autonomous Landing of an Unmanned Aerial Vehicle”. In: *Procedia Engineering* 38 (2012). INTERNATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING, pp. 2250–2256. issn: 1877-7058. doi: 10.1016/j.proeng.2012.06.271.

[17] L. Coutard and F. Chaumette. “Visual detection and 3D model-based tracking for landing on an aircraft carrier”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 1746–1751. doi: 10.1109/ICRA.2011.5979771.

[18] B. Damas, N. P. Santos, and M. Correia Vieira. “A visual tracking system for UAV landing on ships”. In: *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*. 2024, pp. 2627–2632. doi: 10.1109/CoDIT62066.2024.10708450.

[19] O. Yakimenko, I. Kaminer, W. Lentz, and P. Ghyzel. “Unmanned aircraft navigation for shipboard landing using infrared vision”. In: *IEEE Transactions on Aerospace and Electronic Systems* 38.4 (2002), pp. 1181–1200. doi: 10.1109/TAES.2002.1145742.

[20] Y. Gui, P. Guo, H. Zhang, Z. Lei, X. Zhou, J. Du, and Q. Yu. “Airborne Vision-Based Navigation Method for UAV Accuracy Landing Using Infrared Lamps”. In: *Journal of Intelligent robot system* (2013). doi: 10.1007/s10846-013-9819-5.

[21] L. G. L. de Paula, V. S. Dwivedi, H.-S. Shin, and A. Tsourdos. “Towards Autonomous Landing on Dynamic Naval Surfaces Under Harsh Sea State Motion”. In: *AIAA SCITECH 2025 Forum* (2025). doi: 10.2514/6.2025-2607.

[22] J. Neupert, T. Mahl, B. Haessig, O. Sawodny, and K. Schneider. “A heave compensation approach for offshore cranes”. In: *2008 American Control Conference*. 2008, pp. 538–543. doi: 10.1109/ACC.2008.4586547.

[23] X. Zhao, R. Xu, and C. Kwan. “Ship-motion prediction: algorithms and simulation results”. In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 5. 2004, pp. V–125. doi: 10.1109/ICASSP.2004.1327063.

[24] Y. Jiang, X.-R. Hou, X.-G. Wang, Z.-H. Wang, Z.-L. Yang, and Z.-J. Zou. “Identification modeling and prediction of ship maneuvering motion based on LSTM deep neural network”. In: *Journal of Marine Science and Technology* (2021). doi: 10.1007/s00773-021-00819-9.

[25] Q. Sun, Z. Tang, J. Gao, and G. Zhang. “Short-term ship motion attitude prediction based on LSTM and GPR”. In: *Applied Ocean Research* 118 (2022), p. 102927. doi: 10.1016/j.apor.2021.102927.

[26] D. Zhang, X. Zhou, Z.-H. Wang, Y. Peng, and S.-R. Xie. “A data driven method for multi-step prediction of ship roll motion in high sea states”. In: *Ocean Engineering* 276 (2023), p. 114230. issn: 0029-8018. doi: 10.1016/j.oceaneng.2023.114230.

[27] Z. Chen, X. Liu, X. Ji, and H. Gui. “Real-Time Prediction of Multi-Degree-of-Freedom Ship Motion and Resting Periods Using LSTM Networks”. In: *Journal of Marine Science and Engineering* 12.9 (2024). issn: 2077-1312. doi: 10.3390/jmse12091591.

[28] M. Zhang, G. Taimuri, J. Zhang, and S. Hirdaris. “A deep learning method for the prediction of 6-DoF ship motions in real conditions”. In: *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment* 237.4 (2023), pp. 887–905. doi: 10.1177/14750902231157852.

[29] L. Hou, X. Wang, H. Sun, Y. Sun, and Z. Wei. “A long sequence time-series forecasting model for ship motion attitude based on informer”. In: *Ocean Engineering* 305 (2024), p. 117861. issn: 0029-8018. doi: 10.1016/j.oceaneng.2024.117861.

[30] I. Sonata and Y. Heryadi. “Comparison of LSTM and Transformer for Time Series Data Forecasting”. In: *2024 7th International Conference on Informatics and Computational Sciences (ICICoS)*. 2024, pp. 491–495. doi: 10.1109/ICICoS62600.2024.10636892.

[31] B.-M. Min, M.-J. Tahk, H.-C. D. Shim, and H. Bang. “Guidance Law for Vision-Based Automatic Landing of UAV”. In: *The Korean Society for Aeronautical and Space Sciences* (2007). doi: 10.5139/IJASS.2007.8.1.046.

[32] S. Saripalli. “Vision-Based Autonomous Landing of an Helicopter on a Moving Target”. In: *AIAA Guidance, Navigation, and Control Conference* (2009). doi: 10.2514/6.2009-5660.

[33] C. Wu, J. Qi, D. Song, and J. Han. “LP Based Path Planning for Autonomous Landing of An Unmanned Helicopter on A Moving Platform”. In: *Journal of unmanned system and technology* (2013).

[34] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory.” In: *Neural Computation* (1997). doi: 10.1162/neco.1997.9.8.1735.

[35] A. Mancinelli, B. D. W. Remes, G. C. H. E. de Croon, and E. J. J. Smeur. “Unified Incremental Nonlinear Controller for the Transition Control of a Hybrid Dual-Axis Tilting Rotor Quad-Plane”. In: *IEEE Transactions on Robotics* 41 (2025), pp. 306–325. doi: 10.1109/TRO.2024.3498372.

[36] N. Wechtler, A. Mancinelli, and E. Smeur. “Implications of Propeller-Wing Interactions on the Control of Aerodynamic-Surface-Free Tilt-Rotor Quad-Planes.” In: *AIAA SCITECH 2025 Forum* (2025). doi: 10.2514/6.2025-2129.

8 CONCLUSION

In this final chapter, I recap the work presented throughout the thesis and highlight the answers to the research questions posed in the Introduction. Following this, I will discuss the scientific contribution and potential future work related to this research.

8.1. RESEARCH QUESTIONS

Recalling the first research question from the Introduction:

Research Question 1

How can a hybrid lift UAV be designed to ensure stable and precise autonomous landings on moving ship decks in sea state 5?

In Chapter 2, following a review of all available hybrid lift Unmanned Aerial Vehicle (UAV) platforms, we chose to focus on the dual-axis tilting rotor concept. In that chapter, we analyzed the potential of such a propulsion system to power a quad-plane and demonstrated improved performance compared to a conventional quad-plane design. However, we did not fully exploit all the system's capabilities, as the control system used for testing was based on a simple Proportional-Integrative-Derivative (PID) controller and did not leverage detailed knowledge of the vehicle model. Therefore, while the results were promising, a more suitable control system still needed to be developed.

The second research question was posed as follows:

Research Question 2

How can control strategies and allocation methods be developed for nonlinear hybrid UAVs to ensure robust flight performance across the full flight envelope?

In Chapter 3, we examined the limitations of state-of-the-art control approaches, such as Incremental Nonlinear Dynamic Inversion (INDI), when applied to vehicles with highly nonlinear effectors dynamics like the dual-axis tilting rotor quad-plane. Due to computational constraints, these methods often rely on linearized control effectiveness, which can be overly restrictive in systems with strong actuator coupling and nonlinear behavior. To address this, we developed a custom controller that retains the incremental structure of INDI but solves the control allocation problem using a nonlinear programming solver. This enhancement enables the controller to handle the platform's strongly nonlinear actuator dynamics while preserving the responsiveness and disturbance rejection benefits of the incremental approach.

In Chapter 4, we extended the hovering controller from Chapter 3 to cover the entire flight envelope, ensuring smooth and stable control across all airspeed ranges. The final control algorithm, detailed in Chapter 4, provides full 6 Degrees Of Freedom (DOF) control of the vehicle, enabling shipboard operations and effective wind rejection. The controller was successfully tested under challenging real-world weather conditions, further validating the preliminary results from Chapter 2.

The third research question was:

Research Question 3

What level of fault tolerance can be achieved in the chosen UAV to ensure safe landings?

In Chapter 5, we modified the vehicle's hardware architecture to include real-time feedback from all actuators. This enhancement improved control authority by eliminating uncertainties associated with modeled actuator states and established a foundation for designing an informed fault-tolerant controller. In Chapter 6, leveraging the new hardware components that enable actuator state monitoring, we adapted the unified vehicle controller to handle complete rotor failures while maintaining position control in hovering configurations. Flight tests successfully demonstrated the effectiveness of this fault-tolerant control framework, enhancing the vehicle's reliability in challenging environments.

Finally, the fourth research question was:

Research Question 4

How can ship motion be predicted to improve real-time adaptive landing trajectory planning?

In Chapter 7, we designed and implemented a Long Short Term Memory (LSTM) Artificial Neural Network (ANN) model to predict ship motion over a 7-second horizon. This prediction was used to identify feasible landing trajectory for the UAV. The trajectory planning process served as the core maneuver for safely landing the UAV on a moving ship. To ensure resilience against prediction uncertainties and tracking errors, trajectory generation is continuously re-evaluated and reassigned throughout the landing maneuver.

8.2. DISCUSSION

The primary objective of this thesis was to design a hybrid lift UAV capable of performing stable and precise autonomous landings on moving ship decks in high sea states. To address this challenge, four research questions were formulated, focusing on vehicle design, control strategy development, real-time trajectory planning, and fault tolerance.

In Chapter 2, we briefly analyzed existing hybrid lift vehicle design options and directed our attention toward a particularly agile propulsion system: the dual-axis tilting rotor design. When this research began in 2021, the dual-axis tilt rotor propulsion concept had only been developed to enhance the maneuverability of quadcopters and had never been applied to power a hybrid lift vehicle. Through flight tests, we demonstrated the potential of this propulsion system when applied to a quad-plane, achieving enhanced wind rejection capabilities and improved overall maneuverability. However, the initial implementation relied on a simple PID controller that was unable to account for the vehicle's nonlinear dynamics or to operate effectively across the full flight envelope. These limitations underscored the need for a control strategy capable of capturing these interactions and ensuring reliable performance around the full flight envelope, motivating the developments presented in the following chapters.

In Chapter 3, we implemented the state-of-the-art INDI control method and other similar control methods based on the linearization of the control effectiveness, highlighting their limitations when applied to vehicles with highly nonlinear dynamics, such as the dual-axis tilting rotor quad-plane. As a solution, we developed a nonlinear programming-based controller tailored to the vehicle's nonlinear dynamics. Implementing this controller in real time demanded extensive hardware and software optimizations to meet the system's performance constraints. However, this controller was designed exclusively for hovering flight, where the vehicle retains full 6 DOF control authority. It lacked the ability to effectively manage the vehicle during forward flight, where only 4 DOF are actively controllable.

Building upon this foundation, Chapter 4 introduced a unified control framework capable of effectively managing the transition from hovering to forward flight. Despite relying on an imperfect aerodynamic and actuator model, the controller demonstrated robustness and accuracy due to the incremental nature of the control laws, which effectively compensated for model inaccuracies and external disturbances. A key feature of the proposed controller is its ability to seamlessly manage all 6 DOF in the hovering configuration while adapting to the reduced control authority available during fast forward flight, where only 4 DOF are actively controllable. During high-speed flight, the controller effectively managed vertical and lateral accelerations by dynamically reallocating control objectives and disregarding unachievable desired attitude commands. This adaptability allowed for consistent control performance across the entire flight envelope. Additionally, despite occasional local minima in the cost function, the controller remained resilient, quickly resolving these issues through the error controller and vehicle dynamics.

In Chapter 5, we introduced a significant upgrade to the tilting and motor system, enabling direct control of both motor RPM and angular tilt through real-time actuator state feedback, thereby eliminating the need for an accurate actuator model. This hardware improvement enhanced the controller's performance while providing continuous situational awareness of the propulsion system's health. The real-time feedback capability was instrumental in enabling fault-tolerant control, establishing a solid foundation for the developments presented in Chapter 6.

Building upon the improved hardware architecture, Chapter 6 presented a robust fault-tolerant control framework for the dual-axis tilting rotor quad-plane. The platform's over-actuated design made it particularly suitable for accommodating multiple actuator failures. Additionally, the model-based structure of the controller required only minor adjustments to maintain control during rotor failures, even in challenging scenarios such as a complete rotor failure. The enhanced controller successfully maintained position control during rotor failures, including in the hovering configuration, thereby demonstrating its robustness and adaptability. This fault-tolerant capability was further validated through flight tests, proving the system's reliability and effectiveness under adverse operational conditions.

Finally, in Chapter 7, we addressed the challenge of real-time trajectory planning by developing an LSTM artificial neural network model capable of predicting ship motion over a 7-second horizon. This prediction framework enabled the generation of feasible landing trajectory, which were continuously reassessed to account for prediction uncertainties and vehicle tracking errors. The iterative evaluation and reallocation of new trajectories during the landing maneuver proved effective in maintaining resilience against dynamic environmental conditions and ensuring successful landing operations. Due to time constraints, the landing algorithm was only tested in simulation; therefore, a more thorough flight test campaign is necessary to further validate this concept under real-life conditions.

8.3. FUTURE WORK

The research presented in this thesis demonstrated the feasibility of achieving stable and precise autonomous landings of a dual-axis tilting rotor UAV on moving ship decks. However, several aspects of this work warrant further investigation and development to enhance robustness, reliability, and applicability across broader operational scenarios.

The most immediate step for future research involves implementing and testing the landing algorithm presented in Chapter 7 onboard a moving ship. While simulation results demonstrated the algorithm's potential, real-world testing is essential to validate its performance under the dynamic and unpredictable conditions of a maritime environment. This step would provide valuable insights into the control framework's resilience when exposed to real-life disturbances and operational constraints.

Additionally, further refinement of the cost function used in the nonlinear programming controller developed in Chapters 3 and 4 is necessary. A more detailed analysis of the cost function's shape and characteristics across various flight conditions could enhance solution stability and mitigate the risk of becoming trapped in local minima.

The current landing strategy presented in Chapter 7 relied exclusively on a single sensing technology, specifically the GPS-IMU system, to estimate the relative attitude and position between the UAV and the ship. This approach also depended on maintaining a reliable data-link between the drone and the landing pad. However, dependence on a single technology introduces potential vulnerabilities, especially in defense-related applications where RF communication links may be disrupted or jammed. To address these limitations, developing a redundant and robust method for estimating the relative pose between the ship and the drone is essential. Incorporating complementary sensing technologies, such as vision-based systems or LiDAR, would enhance resilience and reliability under adverse conditions.

Furthermore, applying the proposed nonlinear programming-based control algorithm developed in Chapter 4 to different hybrid UAVs presents a valuable research direction. Assessing the algorithm's portability across various configurations would offer insights into its generalization capabilities and potential improvements over existing linearized effectiveness-based techniques, such as the conventional INDI. Comparing the performance of the proposed approach across multiple hybrid platforms could also help identify its strengths and limitations, guiding further refinements and optimizations.

In conclusion, while the methodologies developed in this thesis have shown promising results, extending their application through real-world testing, cost function refinement, redundancy enhancements, and broader applicability to different platforms remains essential. These future developments will contribute to establishing a robust, versatile, and resilient control framework capable of addressing the diverse challenges associated with autonomous landing on moving platforms.

ACKNOWLEDGEMENTS

The work presented in this dissertation is not solely the result of my own effort. It has been made possible thanks to the support of many people who have accompanied me on this remarkable journey—a journey filled with both great rewards and challenges, which has allowed me to grow in ways I could never have imagined. During these four and a half years of doctoral research, I have had the privilege of meeting extraordinary individuals, with whom I was able to build bonds that go well beyond professional collaboration. This achievement belongs to them as much as it does to me.

First and foremost, I wish to thank all the members of the MAVLab, who provided invaluable advice and generously dedicated their time and energy to help me overcome complex challenges, ultimately enabling my drone to fly as I had envisioned. A special thanks goes to Erik, for accompanying me countless times to Valkenburg for flight tests and for believing in my ideas even more than I did myself; to Freek, for always finding solutions to even the most intricate problems; and to Christophe, for his guidance in selecting critical components.

I am deeply grateful to my daily supervisors, Ewoud and Bart, who consistently kept me on track and encouraged me to focus on what truly mattered to achieve this milestone. I will always remember the many afternoons spent in Ewoud's office, working together to unravel electrical and control issues that at first seemed unsolvable.

A very special acknowledgment goes to my promotor, Guido. Thank you for guiding me through the most important stages of my PhD, for carefully reviewing each of my papers, for your constant encouragement, and for ensuring that my doctoral journey has always been a positive and enriching experience.

Beyond the MAVLab, I would also like to thank all my fellow PhD candidates, with whom I shared not only academic collaboration but also genuine mutual support. Thank you for every fruitful discussion at the coffee corner, for every beer we enjoyed together, and for all the activities and BBQs we organized outside of work.

I would also like to express my sincere gratitude to the Master students Noah, Pietro, and Nico, whom I had the pleasure of supervising during these years. Guiding them through their projects was both a responsibility and a privilege, and I learned a great deal from their curiosity, dedication, and fresh perspectives. Their enthusiasm and collaboration not only contributed to my research but also enriched my own journey as a PhD student, making it even more rewarding.

I would also like to warmly thank Jack and Tomaso, with whom I have shared both research and the exciting challenge of entrepreneurship. Together, we took the first steps in transforming the Lab's drone prototype into a real and scalable product, pushing our work beyond the academic context. Collaborating with them has been — and I am certain will continue to be — an extraordinary experience that has combined technical challenges, innovation, and friendship. I am deeply grateful for their trust, dedication, and vision throughout this journey.

I am especially grateful to my friends, both in Italy and here in the Netherlands. Thank you to Youssef, Alessio, Herman, Yildiray, and all the others who stood by me during these years in Delft. I will never forget our BBQs and all the unforgettable moments we shared—your presence truly made a difference.

I also wish to extend my gratitude to all the people who, in one way or another, crossed my path during these four and a half years. Whether through brief conversations, words of encouragement, or shared experiences, each of you has contributed to making this journey richer and more meaningful than I could ever have imagined.

Most importantly, I would like to thank my family. Thank you, Mom Tina, Dad Salvatore, and my brother Carmine, for your unwavering support, for always making me feel valued, for helping me overcome every obstacle, and for celebrating every achievement by my side.
This accomplishment belongs to all of you as well.

Per aspera ad astra.

CURRICULUM VITæ

Alessandro Mancinelli

06-02-1995	Born in Rome, Italy.

EDUCATION

2008–2013	De Pinedo Aeronautical Technical School Rome, IT
2013–2016	BSc in Aerospace Engineering Sapienza University of Rome, IT
2016–2019	MSc in Aeronautical Engineering Sapienza University of Rome, IT Erasmus exchange student at TUDelft, NL (2017–2018) MSc thesis exchange at TUDelft, NL (2018–2019)
2020–2021	Master in Business Administration (MBA) Collège des Ingénieurs (Munich, Paris, Turin)
2021–2025	PhD. Aerospace Engineering Delft University of Technology, NL *Thesis:* Hybrid lift UAV design and control for precision landing on a moving vessel in high sea state *Promotor:* Prof. dr. G.C.H.E. De Croon *Copromotor:* Prof. dr. E.J.J. Smeur

WORK EXPERIENCE

2013-2013	Flight Operations Manager Mistral Air S.r.l. - Rome, IT
2017-2018	Software and telecommunication Engineer Sapienza flight team - Rome, IT
2019-2020	Flight control laws engineer Airbus - Toulouse, FR
2020-2021	Mentor and project leader

	Vento - Turin, IT
2020-2021	Project Manager CIM4.0 - Turin, IT
2021-2023	External Advisor CIM4.0 - Turin, IT
2023-	Chief Technical Officer (CTO) and Co-Founder AerogridUAV - Delft, NL

LIST OF PUBLICATIONS

JOURNAL ARTICLES

2. **A. Mancinelli**, B. D. W. Remes, G. C. H. E. de Croon, and E. J. J. Smeur. "Unified Incremental Nonlinear Controller for the Transition Control of a Hybrid Dual-Axis Tilting Rotor Quad-Plane". In: *IEEE Transactions on Robotics* 41 (2025), pp. 306–325. doi: 10.1109/TRO.2024.3498372

1. **A. Mancinelli**, B. D. W. Remes, G. C. H. E. De Croon, and E. J. J. Smeur. "Real-Time Nonlinear Control Allocation Framework for Vehicles with Highly Nonlinear Effectors Subject to Saturation". In: *Journal of Intelligent & Robotic Systems* 108 (July 2023). doi: 10.1007/s10846-023-01865-8

CONFERENCE PROCEEDING

4. **A. Mancinelli**, N. Voss, and E. Smeur. "Fault Tolerant Control in Over-Actuated Hybrid Tilt-Rotor Unmanned Aerial Vehicles." In: *AIAA SCITECH 2025 Forum* (2025). doi: 10.2514/6.2025-0317

3. N. Wechtler, **A. Mancinelli**, and E. Smeur. "Implications of Propeller-Wing Interactions on the Control of Aerodynamic-Surface-Free Tilt-Rotor Quad-Planes." In: *AIAA SCITECH 2025 Forum* (2025). doi: 10.2514/6.2025-2129

2. **A. Mancinelli**, E. V. D. Horst, B. D. Remes, and E. J. Smeur. "Autopilot framework with INDI RPM control, real-time actuator feedback, and stability control on companion computer through MATLAB generated functions". In: *14th annual International Micro Air Vehicle Conference and Competition*. Aachen, Germany, 2023, pp. 109–116. url: http://www.imavs.org/papers/2023/13.pdf

1. **A. Mancinelli**, E. Smeur, B. Remes, and G. Croon. "Dual-axis tilting rotor quad-plane design, simulation, flight and performance comparison with a conventional quad-plane design." In: *International Conference on Unmanned Aircraft Systems (ICUAS)* (2022). doi: 10.1109/ICUAS54217.2022.9836063