Data-Driven Real-Time Model Identification of UAS for Adaptive Control

by

Sean Bazzocchi

B.Sc., Politecnico di Torino, Italy, 2016
M.Sc., Politecnico di Torino, Italy, 2018
M.Sc., Instituto Superior Técnico, Portugal, 2018

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Mechanical Engineering

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

Data-Driven Real-Time Model Identification of UAS for Adaptive Control

by

Sean Bazzocchi

B.Sc., Politecnico di Torino, Italy, 2016
M.Sc., Politecnico di Torino, Italy, 2018
M.Sc., Instituto Superior Técnico, Portugal, 2018

Supervisory Committee

Dr. Afzal Suleman, Supervisor
(Department of Mechanical Engineering)

Dr. Yang Shi, Departmental Member
(Department of Mechanical Engineering)

Dr. Phalguni Mukhopadhyaya, External Member
(Department of Civil Engineering)

ABSTRACT

This dissertation presents a comprehensive investigation into the development, modeling, and control of novel unmanned aerial vehicles (UAVs) within the Eusphyra project. Structured as a thesis-by-publication, the work delivers significant advancements in UAV design, flight dynamics modeling, autopilot tuning, and adaptive control, offering innovative methodologies to enhance performance and autonomy.

The research begins with the design and airworthiness assessment of the Eusphyra UAV, detailing an iterative development process that culminates in the validation of an innovative tri-rotor VTOL configuration. A high-fidelity flight dynamics model is then developed using limited onboard sensor data and state-of-the-art system identification techniques to capture the complex aero-propulsive coupling inherent in the system. This model is rigorously validated against out-of-sample flight data, confirming its reliability and predictive capability.

Building on these foundational insights, an automated offline autopilot tuning framework is introduced that leverages a simplified system identification process in conjunction with genetic algorithms. This approach minimizes human oversight and enables rapid retuning in response to design modifications. Further extending the scope of the work, the dissertation explores real-time system identification by integrating unsupervised learning techniques to dynamically update UAV models during flight. This capability is advanced into the development of a Model Identification Adaptive Controller (MIAC), which combines Sparse Identification of Nonlinear Dynamics (SINDy) with Model Predictive Control (MPC) for adaptive, online control under varying flight conditions.

Comprehensive hardware-in-the-loop simulations and flight tests confirm the feasibility and performance of MIAC, marking a significant step forward in UAV autonomy and adaptability, and laying the groundwork for future research in advanced adaptive control for complex aerial systems.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those who have supported me throughout my PhD journey—both professionally and personally. This dissertation is not only the result of my own hard work but also a testament to the collective support, guidance, and friendship I have received over the years.

First and foremost, I owe an immense debt of gratitude to my supervisor, Dr. Afzal Suleman. Dr. Suleman not only provided an environment abundant in resources and research opportunities but also maintained an unwavering belief in my abilities—from our work during my master's studies to the completion of this PhD. His encouragement has been a constant source of inspiration, and his generosity in enabling students to conduct research abroad continues to motivate me every day.

I am also grateful to the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Defence Research and Development Canada (DRDC), the University of Victoria (UVic), Canadian Aviation Electronics (CAE Inc.), Aeromagnetic Solutions Inc., and the National Research Council Canada (NRC), along with all the individuals at these organizations whose contributions have been instrumental to this project.

My heartfelt thanks go to my colleagues and collaborators, whose technical expertise and companionship have been instrumental to the success of my projects:

To João Figueira: Your engineering ingenuity during the development of the Mini-E 5050B FDM, along with your invaluable insights during our MIAC project discussions, have been a beacon of inspiration. I have deeply cherished our brainstorming sessions and the friendship that has flourished beyond our work hours.

To Nicolas Castellani: Your remarkable ability to transform a napkin sketch into a functional prototype is nothing short of extraordinary. Your contributions to the detailed design, manufacturing, and successful flight tests of the Mini-E and MIMIQ prototypes have been pivotal, and your passion for mechanical development has made every collaboration a joy.

To Grant Howard: As a true jack-of-all-trades in avionics, you played a crucial role in integrating the MIAC controller and establishing the companion computer for the PX4. Your expertise and willingness to assist, even amid a busy schedule, were essential for the flight tests of the Eusphyra aircraft.

To Dr. Maxym Rukosuyev: Thank you for turning the concept of the Eusphyra into a tangible design. Your innovative solutions, meticulous attention to detail, and even your enthusiasm for titanium parts have challenged and inspired me. Your thoughtful gestures—like timely coffee breaks during long work sessions—made a big difference.

To Dr. José Vale: You were the anchor of the Eusphyra project. Your passion for UAV design and your deep expertise in aerodynamics and structural design were critical to the development of both the Mini-E 5050A and the full-scale Eusphyra. I am grateful for your mentorship and for constantly reminding me to focus on the fundamentals.

To Jack Baker: Your readiness to dive into the most demanding phases of our project, especially during the MAD sensing missions, provided tremendous support. Your hardworking spirit, infectious optimism, and unwavering encouragement helped carry the team through challenging times.

To António Arco: Your work on developing and testing the tri-rotor configuration of the Eusphyra demonstrated a drive and passion that I deeply admire. Our engaging conversations about both work and life have been a great source of motivation.

To Stephen Warwick: As the flight operations manager at CfAR, you have been a role model in every sense. Your guidance in flight operations and your steady support through the ups and downs of the project have been indispensable—not only as a colleague but also as a friend.

To Dr. Jenner Richards: Our brief encounter left a lasting impact on me, inspiring my decision to pursue this PhD journey. Your intelligence and efficiency continue to serve as a benchmark for what I aspire to achieve.

I would also like to acknowledge everyone at the Center for Aerospace Research (CfAR) of the University of Victoria and Quaternion Engineering. Special thanks go to John Raffaelli for his ingenuity in manufacturing, commissioning, and troubleshooting, along with his countless bright ideas; Julia Colasurdo for her steadfast support with electrical issues on the Mini-E 505B; Jay Matlock for always finding time for our projects during the busiest periods; and Kieran Warren for his exemplary leadership and all-around kindness.

I am equally grateful to all the UVic and IST students whose research contributions to the Eusphyra project, as detailed in Chapter 2, have been indispensable.

On a more personal note, my PhD journey would not have been possible without the emotional and moral support of my loved ones:

To my dear parents, Baldovino and Roisin: Your unwavering love, countless sacrifices, and steadfast belief in me have been the foundation of my journey. Even when my dreams led me far from home, your support remained my guiding light. I can only hope to one day be as nurturing, inspiring, and devoted a parent as you have been to me.

To my sisters, Sarah and Laura: Your constant presence, care, and the joyful chaos of our family (complete with dramatic updates from overseas) have kept me grounded and connected, no matter the distance.

To the Gawley extended family (Brian, Margot, Jerram, Brynn, Almudena, and Roy): Thank you for welcoming me with open arms, providing comfort during times of homesickness, and enriching my life with countless memories in nature. Your kindness has helped shape who I am.

To Josie: Though you've only recently entered my life, your passion and determination have ignited a spark within me that guides me forward. Our study sessions and shared adventures have been both a source of inspiration and sanity during the challenging moments of writing this dissertation. Thank you for filling my days with joy and for reminding me to keep striving with grace and perseverance.

I must also thank the many friends from the UVic Sailing Team and MyTai Sailing Team, whose companionship and adventures on the water provided laughter, exercise, and balance throughout this journey. Special thanks to Nathan, Maclain, Ethan, Maggie, Shima, Dallas, Hugh, Kim, Liam, Hunter, and Barry for the joy and distraction you generously shared with me.

To Inês: Even our brief encounter brought laughter and support during some of the busiest times of my research, reminding me to take life a little less seriously.

To Alexander: Your guidance on sailing, match racing, and our long conversations on the water enriched my sporting life and provided clarity for many important decisions regarding my future.

To Ana: Thank you for the many hours of discussion about the highs and lows of PhD life. Your curiosity and thoughtful insights into what it means to pursue a doctoral journey have been invaluable. Our coffees and sauna sessions with João have been a true balm for my mental health.

And finally, to my best friend, roommate, coworker, and acquired brother, Ben: Your constant presence, support, and unwavering belief in me have been the backbone of my PhD experience. From everyday challenges to monumental moments, you have been there through it all. Without you, both my time in Canada and the completion of this dissertation would have been unimaginable.

To everyone who has walked this path with me, thank you. Each of you has contributed in unique and significant ways to the realization of this work, and I am forever grateful for your support, mentorship, friendship, and love.

DEDICATION

To my beloved parents, Baldovino and Roisin,

whose unwavering support and the sacrifices of letting me pursue my dreams far from home have been my guiding light.

"Nella consapevolezza delle tue radici, per aspera ad astra."

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) have emerged as indispensable assets in a wide range of applications, including environmental monitoring, disaster response, infrastructure inspection, and defense operations. Their versatility, operational flexibility, and capacity to access hazardous or remote areas render them particularly suitable for tasks that require real-time sensing and data acquisition [1, 2]. In recent years, small-scale UAVs have also gained prominence due to their rapid deployment capabilities, cost-effectiveness, and adaptability in prototyping advanced flight technologies. As research in UAVs continues to advance, there is an increasing emphasis on enhancing autonomy, flight performance, and adaptability under varying operational conditions. The Eusphyra project exemplifies this evolution by investigating novel UAV configurations tailored for challenging mission profiles, such as magnetic anomaly detection. By leveraging recent advancements in data-driven flight dynamics modeling, autopilot tuning, and online system identification, this research develops an integrated framework for UAV automation. Despite notable progress in UAV technology, fundamental challenges persist in flight dynamics modeling and control. The unconventional nature of emerging UAV designs, coupled with the stringent requirements of rapid prototyping environments, necessitates robust methodologies that balance modeling fidelity with computational efficiency.

1.1 Background and Motivation

Small-scale UAVs offer a valuable experimental platform for aerospace research by enabling rapid iteration of design concepts, thereby reducing development time and mitigating risks while maintaining a rigorous experimental framework. Such platforms have been instrumental in advancing aerospace technologies through real-world testing and validation.

The evolution of UAV technology is driven by the growing need for autonomous, reliable, and adaptable aerial systems. Traditional fixed-wing and rotary-wing UAVs, although widely deployed, exhibit inherent limitations in addressing the diverse requirements of modern missions. Hybrid configurations, such as Vertical Takeoff and Landing (VTOL) UAVs, seek to reconcile these limitations by combining the endurance of fixed-wing aircraft with the maneuverability of multirotors. However, these unconventional designs introduce substantial complexities in flight dynamics and control, thereby demanding the development of novel system identification and control methodologies.

A critical challenge in UAV development lies in achieving accurate flight dynamics modeling using limited onboard sensor data. High-fidelity models often depend on extensive flight testing, computationally intensive simulations, or wind tunnel experiments, all of which are resource-intensive. This dissertation advances the state-of-the-art in system identification by proposing a structured methodology for high-fidelity UAV modeling using minimal sensor information, thereby enhancing predictive accuracy while reducing reliance on costly empirical methods.

Robust and accurate control systems are vital for UAVs, particularly when executing autonomous missions or stabilizing inherently unstable configurations. Although significant progress has been made in control theory, many high-performance control strategies are challenging to implement on aerial platforms due to their sensitivity to model inaccuracies [3]. Consequently, autopilot design has traditionally relied on robust control methods that tolerate modeling uncertainties, often at the expense of optimal performance or requiring extensive modeling and tuning efforts.

Conventional UAV control development typically involves constructing comprehensive aerodynamic models through iterative ground-based testing, extensive flight maneuvers, and meticulous post-flight analysis [4–8]. While effective, this approach is both time-consuming and costly, with inherent fidelity limitations arising from factors such as wind tunnel scale effects, structural interference, Reynolds number

discrepancies, and necessary computational approximations.

Moreover, rapid prototyping environments demand flexible and automated tuning strategies for UAV autopilots. Traditional tuning methods generally require significant human intervention, specialized expertise, and numerous flight tests. To address these challenges, this research introduces an optimization framework that integrates system identification with genetic algorithms, thereby enabling efficient autopilot tuning with minimal manual intervention. This approach not only enhances UAV performance but also significantly expedites the development cycle.

An additional challenge addressed in this work is the development of real-time adaptive control strategies. Conventional UAV control laws are often designed for predetermined operating conditions and may struggle to accommodate variations in aerodynamic properties, payload configurations, or environmental factors. The proposed Model Identification Adaptive Control (MIAC) framework integrates Sparse Identification of Nonlinear Dynamics (SINDy) with Model Predictive Control (MPC) to facilitate real-time system identification and adaptive control. The feasibility and effectiveness of this approach are rigorously validated through hardware-in-the-loop simulations and flight testing, underscoring its potential to enhance UAV autonomy.

The implications of this research extend well beyond UAV prototyping. As Morelli [9] highlights, the development of online adaptive dynamic models and full-envelope flight controllers holds significant promise for applications in fault detection, self-learning aerial systems, morphing wing control, flight envelope protection, rapid flight testing, and high-fidelity aerodynamic modeling from flight data. Collectively, these advancements contribute to the broader objective of improving UAV reliability, safety, and operational flexibility.

1.2 Literature Review

In this section, we introduce the literature pertinent to the topics addressed in this dissertation. The discussion is organized into four subsections: 1 Small-scale UAV Development for Prototyping, 2 Flight Dynamic Modeling and System Identification, 3 Online System Identification, and 4 Model Identification Adaptive Control.

1.2.1 Small-scale UAV Development for Prototyping

Small unmanned aerial vehicles (UAVs) have emerged as a pivotal tool in aerospace research and development, primarily due to their capacity for rapid prototyping of innovative technologies. As noted by Owens [10], sub-scale UAVs offer a cost-effective and lower-risk alternative to full-scale crewed aircraft, permitting the safe execution of aggressive maneuvers and the evaluation of high-risk scenarios. This approach not only facilitates the experimental implementation of novel control laws and failure simulations [11] but also serves as an essential intermediary between simulation and full-scale flight testing [12].

The iterative design process central to small-scale UAV development has significantly enhanced prototyping methodologies. Frameworks such as the UAV Development Framework introduced by Dantsker et al. [13] exemplify how iterative cycles of design, modeling, simulation, and testing can expedite development timelines. These processes are further optimized by the reuse of validated components and the integration of Commercial Off-The-Shelf (COTS) parts and open-source autopilot systems, which collectively foster innovation while reducing costs and development time.

Additionally, the advent of additive manufacturing (most notably 3D printing) has revolutionized the prototyping phase. This technology enables rapid fabrication of airframe components, thereby allowing for swift design iterations and reducing reliance on traditional manufacturing processes. Krznar et al. [14] demonstrate that additive manufacturing can considerably diminish both development time and costs by facilitating the quick production and testing of multiple prototype parts. The versatility of 3D-printed components is particularly advantageous for experimental UAVs, which benefit from low-volume production and the capacity for rapid modification [15].

Despite these advancements, challenges remain in extrapolating the results from small-scale tests to full-scale aircraft, primarily due to scaling effects and payload limitations. Nonetheless, numerous global research initiatives have validated the effectiveness of small UAV platforms in expediting the development and maturation of new aerospace technologies [16].

1.2.2 Flight Dynamic Modeling and System Identification for UAVs

Accurate flight dynamic models are essential for the simulation, control design, and safe operation of UAVs. These models are commonly derived from a nonlinear 6-degree-of-freedom (DOF) framework that integrates rigid-body equations of motion with aerodynamic, propulsion, and inertial characteristics. For example, Triputra [17] demonstrated that classical aerodynamic structures can be effectively adapted to construct state-space models, capturing UAV performance across a variety of flight envelopes. Similarly, Wang et al. [18] showcased the application of dynamic models in VTOL UAVs to inform flight strategies and evaluate performance.

Typically, these physics-based models are initially developed using computational fluid dynamics (CFD), or wind tunnel data and then linearized about a trimmed flight condition. This linearization process not only isolates the dominant states but also incorporates stability derivatives calculated from aerodynamic coefficients. Subsequent validation through flight testing (where maneuvers are specifically designed to excite multiple motion axes) enables the estimation and refinement of aerodynamic derivatives. In one study, Low-Hansen et al. [19] modeled a Skywalker X8 UAV using a classical aerodynamic framework, later employing stepwise regression to identify a comprehensive set of 44 aerodynamic coefficients spanning the longitudinal, lateral, and directional dynamics. Likewise, Simmons [20] integrated computational aerodynamics with empirical adjustments to develop a nonlinear model for a small fixed-wing UAV, which ultimately informed control law design, risk analysis, and the creation of pilot training simulators. Furthermore, the work of Dorobantu et al. [21] illustrates the application of a systematic frequency-domain system identification (SID) procedure on the Ultra Stick 25e, thereby confirming the practical relevance of these methodologies.

Because small UAVs fly in a regime where they encounter challenges such as low Reynolds number aerodynamics and propeller slipstream effects that complicate purely analytical modeling, SID becomes a crucial tool. System identification refines these models using flight test data, as detailed comprehensively by Morelli and Grauer [22]. Foundational contributions by authors like Morelli and Tischler have significantly advanced SID techniques. Morelli's work, particularly the development of the SIDPAC toolset for NASA [23], standardized the estimation of stability and control derivatives using methods such as maximum likelihood estimation. In parallel,

Tischler's frequency-domain approaches (notably exemplified by the CIFER system) have proven highly accurate for rotorcraft and flexible aircraft, where coupled modes are prevalent [24]. These data-driven methods, often utilizing multisine and chirp inputs to stimulate both rigid-body and aeroelastic modes, underscore the indispensable role of extensive flight test data in achieving high-fidelity models.

Although these approaches yield models with high fidelity, they necessitate extensive flight testing to collect sufficient data, followed by rigorous post-flight analysis to identify and process valid flight segments. The subsequent section will explore methods and projects that aim to perform system identification in-flight, thereby reducing reliance on manual data processing and enabling faster, unsupervised analysis.

1.2.3 Online System Identification

Online system identification (SID) presents several challenges as it must operate autonomously, with limited computational resources and in the presence of noisy or incomplete data [25]. One promising approach is real-time frequency response estimation. By injecting small excitation signals during flight and measuring the corresponding response, the dynamic characteristics of the UAV can be identified in real time [26]. Holzel and Morelli further demonstrated the feasibility of computing frequency response functions on the fly from actual flight data [27]. Building upon these foundations, Grauer and Morelli developed methods that not only estimate the frequency response in real time but also quantify the uncertainty associated with these estimates, thereby enabling an onboard system to assess its confidence in the identified dynamics during flight [28]. Extending these techniques to closed-loop systems, Grauer applied orthogonal multisine signals to excite multiple control loops simultaneously without compromising aircraft stability [29]. Such real-time SID methods have even been employed in NASA's X-56A flight tests to monitor changes in aeroelastic modes [30].

Despite these advances, real-time SID still faces significant obstacles. Small UAVs are particularly vulnerable to turbulence and sensor noise, which can corrupt data and degrade model quality [20]. In addition, the limited precision of low-cost sensors (often characterized by higher noise floors or biases) necessitates robust filtering or data fusion strategies. Research by Heim and colleagues at NASA has highlighted that for minimally instrumented UAVs, disturbances such as wind gusts can impair the accuracy of the identified models. Consequently, careful design of filtering tech-

niques and flight maneuvers (such as repetitive or synchronized multi-axis inputs) is critical for maximizing the information content of the data. Furthermore, the design of input perturbations (e.g., doublets, chirps, or multisines) must strike a balance between eliciting sufficient dynamic response for identification and maintaining flight safety. Recent studies propose optimized multisine signals that concentrate energy at frequencies of interest while minimizing amplitude excursions, thereby enhancing the signal-to-noise ratio for SID [31, 32].

The inherent computational constraints of in-flight processing demand that identification algorithms be both efficient and robust. While full nonlinear batch estimations are typically too time-consuming for real-time application, computationally light techniques—such as Morelli's frequency-domain methods or recursive least squares—offer a viable alternative. In recent years, there has been a growing interest in leveraging machine learning for system identification and control. Brunton and collaborators have extensively explored data-driven techniques, including NARMAX, the Eigensystem Realization Algorithm (ERA), Genetic Programming (GP), Neural Networks (NN), and Koopman with control [33]. Proctor et al. introduced Dynamic Mode Decomposition with control (DMDc), which is capable of extracting low-order models from complex systems while effectively isolating the influence of control inputs [34]. Among these novel approaches, the Sparse Identification of Nonlinear Dynamics (SINDy) framework has shown significant promise. Initially proposed in 2016 [35] and later extended to incorporate external inputs and feedback control (SINDYc) [36], SINDy employs sparsity-promoting techniques to extract governing equations directly from noisy measurement data. Kaiser demonstrated the potential of SINDy for online SID by applying it to a range of problems—including a simulated chaotic Lorenz system, a nonlinear pitch problem of an F8, and an HIV model—thereby underscoring its applicability to control optimization in noisy and discrete data environments [37]. Further, Quade proposed a fault detection framework that leverages sparse regression to update models with minimal modifications in response to abrupt system changes [38], while Fasel refined the method with ensemble capabilities to better address challenges posed by limited and noisy data [39].

Collectively, these developments suggest that online SID is evolving towards the capability of enabling UAVs to learn and adapt their own dynamics in flight. The integration of these real-time identification techniques with adaptive control strategies is embodyed by initiatives such as NASA's "Learn-to-Fly" program, which aims to develop onboard algorithms that update system models without the need for extensive

ground testing [40]. This convergence of real-time identification and adaptive control lays the groundwork for the next generation of UAVs that can autonomously optimize their performance in dynamic environments.

1.2.4 Model Identification Adaptive Control

Model Identification Adaptive Control (MIAC) encompasses control architectures that update a system's model in real time and adjust the corresponding control laws to maintain optimal performance. The overarching vision is to develop an aircraft that "learns to fly" by autonomously identifying changes in its dynamics (whether due to damage, failures, or transitions within the flight envelope) and then adapting its controller accordingly. This concept is not entirely new; early efforts to extend autopilot functionality across the entire flight envelope date back to World War II [41], and subsequent decades have witnessed continual advances in adaptive control theories.

Early implementations of MIAC can be seen in Self-Designing Control (SDC) systems. For instance, a reconfigurable control scheme was demonstrated on the *VISTA F-16* using regularized sequential least-squares parameter estimation to compute the necessary stability and control derivatives for solving an optimal control problem [42]. In that work, several simplifications were adopted, nonlinear dynamics were linearized to yield a state-space model, and inertia-ratio parameters were held at nominal values to reduce the identification burden. An innovative aspect of the approach was the clamping of parameter updates to restrict abrupt variations, although this limitation also reduced the system's ability to react swiftly to sudden changes. Flight tests, which simulated a missing control surface by gradually transitioning an actuator to a "float" state, revealed issues such as a divergent nose-down transient when the elevator-to-pitch acceleration parameter approached zero. These discrepancies, particularly within the SID process, led to recommendations for incorporating nonlinear dynamics in future identification algorithms [43].

Parallel to these developments, research on control allocation for reconfigurable systems was advancing. Several studies proposed methods to enhance parameter estimation under rapid dynamics changes. For example, Chandler introduced an SID technique that incorporated known information into least-squares estimation, while Buffington proposed a dynamic inversion controller designed to prevent actuator saturation by scheduling controller parameters off-line according to flight conditions [44–

46]. Neural network–based adaptive controllers were also explored; Brinker and Wise developed a dynamic inversion scheme that employed an online neural network to regulate plant inversion errors, building upon earlier work by Kim and Calise [47, 48]. These adaptive control solutions underwent extensive evaluation in programs such as the Air Force's *RESTORE* and Boeing's *Robust Adaptive Controller Experiment* (RACE).

The drive to ensure aircraft resilience in the presence of component malfunctions prompt the evolution of fault-tolerant control systems (FTCS), which inherently accommodate component failures. Comprehensive reviews, such as that by Zhang and Jiang [49], classify and contextualize the myriad FTCS methods developed over the past decade, further underscoring the relevance of adaptive control in modern aerospace systems.

More recently, MIAC has been reinvigorated through initiatives like NASA's Learn-to-Fly program [50], which integrates online identification and adaptive control to retune flight control systems in real time. This "learn-on-the-fly" approach involves executing a series of small excitation maneuvers to update the aircraft model, followed by the retuning of control gains during flight. Grauer [40] demonstrated the viability of this concept by adaptively tuning the stability augmentation system of a subscale aircraft. In one implementation, a nonlinear dynamic inversion method augmented by an adaptive disturbance rejection module was combined with an L1 adaptive control law (introduced to address robustness and performance issues in adaptive controllers [51, 52]) to compensate for high uncertainty levels. Subsequent extensions to L1 adaptive control enabled the linear reference systems to be updated in real time and parametrized according to different flight conditions [53]. While successful applications have been demonstrated on platforms such as the Learjet [54], the F-16, and the AirSTAR GTM [55], critiques have noted that the L1 adaptive controller can exhibit behavior similar to a full-state feedback, linear time–invariant proportional integral (PI) controller with decaying additive disturbances [56, 57].

Other research efforts, by the L2F team, have explored alternative strategies to mitigate adverse model–control coupling. As Morelli [58] explains, the reliance on certain key control variables (e.g., angle of attack) for modeling can compromise identification quality. A real-time global aerodynamic modeling technique was developed to attribute dynamic responses to individual control surfaces, compute stability and control derivatives, and allocate commands based on estimated control effectiveness. Although flight tests (e.g., the E1 and Woodstock campaigns) validated aspects of

this approach, its dependency on frequency response estimation (necessitating steady-state data and pre-programmed excitation inputs) highlighted limitations that require further refinement [59, 60]. Nonetheless, the successes achieved in the L2F project have had a profound impact on the field and have influenced the structure of subsequent research in this dissertation.

In parallel with traditional adaptive control methods, recent advances in artificial intelligence have stimulated interest in applying Reinforcement Learning (RL) to UAV control tasks. RL techniques have shown promise in domains such as perception, planning, and localization [61, 62]. Notable studies include those by Zhang et al. [63] and Kelchtermans et al. [64], in which model predictive control was used to generate training data for deep neural network (DNN) policies. However, the non-interpretable, "black-box" nature of RL–derived policies, along with challenges in verification and extensive training requirements (as illustrated in [65]), limit their direct application to low-level motion control. Active research in explainable artificial intelligence [66–68] seeks to address these issues, but RL remains more suited as an optimization or identification tool within a broader adaptive control framework.

Central to the MIAC framework is the system identification process. Recent advances in data–driven modeling techniques (most notably the Sparse Identification of Nonlinear Dynamics (SINDy) framework [35, 36]) have provided a compact and interpretable means of capturing nonlinear system behavior. SINDy's computational efficiency and robustness to noise make it particularly attractive for onboard implementation. Kaiser [37] demonstrated MIAC using a SINDy-based model predictive control (SINDY–MPC) approach on systems including the longitudinal dynamics of a vintage F-8 Crusader model, while NASA's adaptive flight control experiments on the X-36 and X-48 platforms further exemplify the MIAC philosophy [69].

In summary, MIAC represents the convergence of real-time system identification and adaptive control. Pioneering contributions by researchers such as Morelli, Tischler, Heim, Grauer, and Brunton have collectively advanced the field from offline identification and fixed–gain control toward self–modeling, self–correcting flight control systems. Real–world demonstrations, from NASA's subscale testbeds to simulated full–scale aircraft, underscore the potential of MIAC to support the rapid development of aircraft novel configurations as well as potential for a certifiable adaptive control technique, marking a significant step toward the future of autonomous flight.

1.3 Research Problem and Objectives

This research addresses several interrelated challenges that arise during the rapid prototyping and development of a novel Vertical Takeoff and Landing (VTOL) UAV for the MAD sensing project. The project explores different configurations with fully tilting propellers and hybrid engines, designs chosen for their potential to improve airworthiness and operational agility. However, this unconventional designs introduces a set of research problems that must be overcome to achieve reliable performance and robust control.

A primary challenge is the development of a high-fidelity flight dynamic model for a UAV that is equipped with a limited onboard sensor suite. Traditional system identification techniques, which often rely on extensive flight testing or wind tunnel experiments, are impractical in a rapid prototyping environment. Thus, there is a critical need for a structured methodology that can extract accurate dynamic models from minimal sensor data.

Furthermore, the rapid prototyping process inherently involves frequent design changes. Each alteration necessitates retuning and validating the UAV's controller, imposing a significant workload and increasing the potential for human error. To address this, an automated tuning architecture is required, one that leverages unsupervised system identification methods in conjunction with genetic algorithms to optimize controller parameters efficiently and consistently.

Lastly, the dynamic nature of UAV operations and the potential for system faults call for an adaptive control strategy. The development of a Model Identification Adaptive Controller (MIAC) is proposed to meet this need. By continuously identifying the UAV's dynamics online and integrating these insights with a model-based control scheme, the MIAC framework aims to improve performance, enhance fault tolerance, and accommodate morphing aerodynamic effects.

In summary, the primary objectives of this research are to:

1. Develop a novel VTOL UAV platform for the MAD sensing project by advancing control system design to streamline autopilot development and tuning.

2. Formulate and validate a robust methodology for generating high-fidelity flight dynamic models from the limited sensor data typical of small-scale UAVs.

3. Design and implement an automated controller tuning framework that integrates unsupervised system identification with genetic algorithm optimization

to reduce manual retuning efforts.

4. Demonstrate the feasibility of a Model Identification Adaptive Controller (MIAC) by adapting state-of-the-art online system identification techniques and integrating the identified model with a Model Predictive Control (MPC) scheme for robust adaptive control.

By addressing these critical challenges, the work in this dissertation makes substantial contributions to the field of UAV automation. It presents a systematic approach to data-driven flight dynamics modeling, tuning optimization, and adaptive control strategies.

1.4 Research Contributions

This dissertation presents a systematic and integrated approach to addressing the challenges inherent in rapid prototyping and adaptive control of unconventional UAV designs. The key contributions of this work are as follows:

- **Airworthiness Analysis and Novel Configuration:** A comprehensive evaluation of the airworthiness of a novel tri-rotor VTOL UAV configuration is presented. Detailed analyses of attitude control during hovering, transition, and forward flight are provided, with both ground tests and flight experiments confirming the feasibility of the front and rear rotor tilt mechanisms.

- **High-Fidelity Flight Dynamics Modeling:** A high-fidelity flight dynamics model is developed based on limited onboard sensor data. Advanced system identification techniques are employed to capture the nonlinear aero-propulsive coupling inherent in VTOL configurations, with rigorous validation performed using out-of-sample flight data.

- **Automatic Autopilot Tuning Framework:** An automated framework for offline autopilot tuning is introduced, integrating a simplified system identification process with genetic algorithm optimization. This framework minimizes manual intervention, enabling rapid retuning in response to design modifications while effectively handling the complexities of unconventional vehicle dynamics.

- **Online System Identification for Adaptive Control:** The dissertation explores real-time, unsupervised system identification methods, specifically, leveraging the Sparse Identification of Nonlinear Dynamics (SINDy) framework, to dynamically update UAV models during flight. The study systematically quantifies the impact of sensor noise and training time on model fidelity.
- **Model Identification Adaptive Controller (MIAC):** A comprehensive adaptive control architecture is designed and implemented by integrating the online system identification process with a Model Predictive Control (MPC) framework. The resulting MIAC is validated through high-fidelity hardware-in-the-loop simulations and real flight tests, representing a significant advancement in UAV autonomy and fault-tolerant control.

1.5 Thesis Organization

This dissertation is structured as a thesis-by-publication, wherein each main chapter corresponds to one or more publications. Each chapter begins with a brief introductory section that situates the work within the broader research agenda of the Eusphyra VTOL UAV project, outlining its objectives and contextual significance. This structure guides the reader through the logical progression of the research.

Following this introductory chapter, the dissertation is organized as follows:

- **Chapter 2: Project Background and UAV Development**
  This chapter outlines the overarching mandate and objectives of the project in which the Eusphyra VTOL UAV development took place. It details the aircraft development process implemented to address the principal research questions and meet defined performance objectives. The chapter introduces each prototype developed to achieve the required Technology Readiness Level (TRL) and illustrates how iterative design modifications contributed to the evolving system architecture. It culminates with a published article examining the airworthiness of one prototype, serving as a representative case study of the concept-to-prototype development pathway.
- **Chapter 3: Flight Dynamics Modeling**
  Chapter 3 presents the development of a high-fidelity flight dynamics model for the most advanced Eusphyra prototype used in key technical demonstrations.

Derived from state-of-the-art system identification techniques applied to limited onboard sensor data, the model accurately captures the aero-propulsive coupling of the vehicle. Rigorous validation against out-of-sample flight data confirms the model's predictive capabilities.

- **Chapter 4: Automatic Autopilot Tuning Framework**
  This chapter introduces an automated framework for offline autopilot tuning. By integrating a simplified system identification process with a genetic algorithm-based optimization scheme, the framework minimizes manual intervention. It enables rapid retuning of the controller in response to design modifications or the introduction of new prototypes, thereby reducing the engineering workload associated with conventional tuning methods.

- **Chapter 5: In-Flight Nonlinear System Identification for Adaptive Control**
  Chapter 5 explores a novel unsupervised system identification (SID) technique for real-time application onboard the UAV. Following an overview of system identification and its role in adaptive control, the chapter investigates both the real-time SID algorithm and the corresponding control perturbation strategy. Numerical simulations based on the high-fidelity flight dynamics model illustrate the impact of sensor noise and training duration on model fidelity.

- **Chapter 6: Model Identification Adaptive Control**
  In this chapter, the online SID process is integrated into a real-time model-based controller framework, marking a significant advance in adaptive control for UAVs. The chapter details algorithmic enhancements, hardware considerations, and software implementations that support online model identification. The fully realized MIAC is validated through high-fidelity simulations, hardware-in-the-loop testing, and real flight experiments, demonstrating its capacity to adapt to varying flight conditions and outperform static control laws.

- **Chapter 7: Conclusions and Future Work**
  The final chapter synthesizes the main findings and contributions from Chapters 2 through 6, critically examining the limitations of the current research. It proposes future directions, including the extension of the adaptive control framework to other applications and the integration of advanced sensing modalities to further enhance online system identification accuracy.

Each chapter has been designed to stand alone as a publication while contributing to a coherent and integrated research narrative throughout the dissertation.

Chapter 2

Project Background and UAV Development

This chapter presents a high-level overview of the work performed during a multi-year research effort titled *UAV Based Magnetic Anomaly Detection System for Remote Sensing*. The project was supported by multiple stakeholders, including the Natural Sciences and Engineering Research Council of Canada (NSERC), Defence Research and Development Canada (DRDC/RDDC), the University of Victoria (UVic), Canadian Aviation Electronics (CAE Inc.), Aeromagnetic Solutions Inc., and the National Research Council Canada (NRC). Under the supervision of Dr. Afzal Suleman as the principal investigator, the project aimed to explore the feasibility of using unmanned aerial vehicle (UAV) platforms to carry magnetic anomaly detection (MAD) sensors for submerged target detection.

The overarching motivation was to enable efficient and accurate detection of submerged metallic targets, such as submarines or other vessels, using advanced sensor payloads. UAVs naturally lend themselves to missions involving repetitive or hazardous flight profiles, thus offering several advantages over conventional crewed aircraft in terms of cost, risk, and operational flexibility. However, achieving the stringent magnetic signature, flight performance, and sensor integration requirements imposed by a MAD-based mission necessitated a series of iterative UAV designs and prototypes.

This chapter is organized into three main sections. First, a *Project Overview* outlines the original mandate and objectives, offering insight into the operational and performance constraints that shaped the UAV development process. Next, the

Eusphyra Aircraft Development section presents a chronological account of the various prototypes built and tested to reach the required Technology Readiness Level (TRL) and highlights key design choices made in response to evolving project requirements. Each prototype subsection describes how lessons learned in previous iterations guided changes in aerodynamic configuration, structural design, propulsion arrangement, and control systems. Lastly, the chapter culminates in a detailed description of a *Tri-Rotor Airworthiness Investigation* manuscript, which serves as a representative case study.

More details about the magnetic anomaly detection research, the VTOL configuration research, the Eusphyra design, and the multi-agent work may be found in the following references [70–84]. Although these topics are connected to the broader body of work, the present chapter aims to highlight the rationale behind designing the new vehicle platform, the incremental development through multiple prototypes, and how these informed subsequent research directions such as automated controller tuning and the creation of a self-learning adaptive controller.

2.1 Project Overview

The original mandate for this project was to research and demonstrate the viability of UAV platforms for carrying magnetic anomaly detection (MAD) sensors, specifically for the detection of submerged metallic vehicles. By integrating sensors such as CAE's MAD-XR [85] (see Figure 2.1), the project sought to capitalize on the unique advantages of UAVs for surveillance tasks that are monotonous, repetitive, or hazardous for human-occupied aircraft. Through this overarching goal, the research team pursued a specialized UAV design with vertical takeoff and landing (VTOL) capabilities, capable of long-endurance flight in challenging environments.

Figure 2.1: CAE Magnetic Anomaly Detection sensor (MAD-XR).

Due to the particular operating conditions required by the mission and the complexity of integrating such a payload on an unmanned platform, the project was further decomposed into a series of incremental milestones:

1. Characterize the magnetic signature of a ready-to-fly "general-purpose" UAV.
2. Fly a smaller, lighter, and easier-to-install magnetometer (with lower sensitivity than the MAD-XR) on the ready-to-fly UAV to assess potential roadblocks.
3. Integrate the MAD-XR sensor on the UAV.
4. Conduct flight testing with the MAD-XR-equipped traditional UAV.
5. Design a specialized VTOL UAV to carry the MAD-XR with minimal magnetic signature and structural displacement (also satisfying mission requirements such as cruise speed, dash speed, endurance, and maximum wind capability).
6. Develop and flight-test a sub-scale version of the specialized VTOL UAV.
7. Conduct flight trials with the specialized UAV and onboard MAD sensor over known magnetic targets.

These milestones guided the systematic exploration of sensor integration challenges, UAV flight performance, and the need for customized airframe and propulsion

configurations. The first four objectives were addressed using a Nebula N1 aircraft (Figure 2.2), which served as a testbed for installing and operating the MAD-XR sensor on a wingtip pod to minimize magnetic interference. Early ground and flight tests using this platform (detailed in [70–72]) revealed several challenges:

- Persistent vehicle-induced magnetic interference (Figure 2.3).
- Degradation of sensor data quality due to the dynamic relative movement of the sensor with respect to the vehicle center of gravity.
- No VTOL capabilities and a suboptimal flight envelope for capturing MAD data in the given operating conditions.

Figure 2.2: Nebula N1 aircraft with MAD-XR integrated on the wingtip pod.

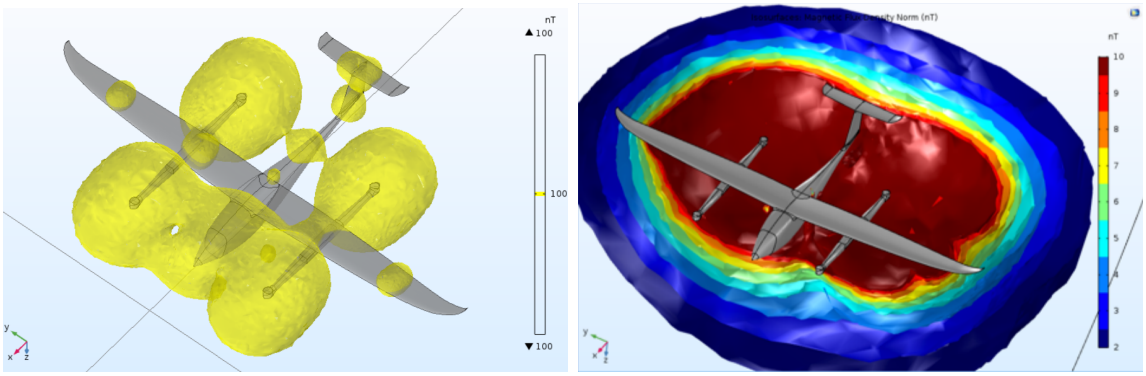


Figure 2.3: Isosurfaces of Magnetic Flux Density norm: Left shows 100 nT magnitude, and Right displays the colormap of magnetic flux density.

As a result of these initial tests, project members concluded that a specialized UAV design was necessary to minimize magnetic interference and accommodate the required mission profile. Specifically, the new airframe needed to satisfy the following key requirements:

- **VTOL capabilities**: The aircraft must perform vertical takeoff and landing to enable shipborne deployment and operation in confined environments.
- **Low magnetic signature**: Ideally, the UAV-generated magnetic flux density near the sensor should be less than 2 nT (with a maximum tolerable limit of 3 nT).
- **Rigid sensor mounting**: Minimizing relative movement of the sensor with respect to the center of gravity and IMU reduces complexity in data post-processing.
- **High wind tolerance**: Arctic operations often face winds of up to 40 knots, requiring robust aerodynamic and control characteristics.
- **Cruise and dash speeds**: Must achieve a dash speed of 115 knots and a cruise speed of 70 knots for effective transit and mission efficiency, with a stall speed of 40 knots.
- **Endurance**: Achieve a mission endurance of at least 5 hours, with a preference for extended flight times.
- **Multi-agent readiness**: The platform should be adaptable to cooperative missions, potentially scaling to simultaneous deployments for wider surveillance.

- **Ship-based operations**: The UAV must be deployable from moving platforms, requiring precision navigation (e.g., differential GPS, optical flow for moving bases).

These requirements established the design envelope and performance targets for the specialized UAV, referred to as Eusphyra, which prioritized magnetic invisibility, robust flight performance, and modularity for sensor integration. Subsequent sections will describe the development and refinement of multiple Eusphyra prototypes, culminating in a mature configuration that meets the mandated performance and airworthiness standards. In the final section of this chapter, an airworthiness investigation of a representative tri-rotor prototype is presented as a case study, illustrating the concept-to-prototype development pathway.

2.2 Eusphyra Aircraft Development

With the preliminary milestones achieved through integrating the MAD sensor on the Nebula N1 UAV and a set of refined requirements established by key stakeholders, the next phase involved designing a specialized UAV uniquely tailored for carrying a MAD sensor. The development strategy adopted leveraged an iterative, *hands-on* approach that has been a longstanding practice at the Center for Aerospace Research at the University of Victoria. In this methodology, promising configurations are rapidly prototyped at a sub-scale and flight-tested. Such physical trials typically yield more actionable insights than purely analytical or simulation-based studies in a similar timeframe, particularly given the laboratory's comprehensive design, manufacturing, and flight-testing capabilities.

The guiding philosophy is to produce an inexpensive, easy-to-manufacture *proof-of-concept* (POC) prototype once a candidate configuration is deemed viable. This allows early detection of potential aerodynamic, structural, and control issues under realistic conditions. Following successful evaluation of the POC, progressively more complex prototypes can be built to meet stricter performance criteria. Sub-system testing (e.g., propulsion modules, control surfaces, avionics) is conducted wherever possible in isolation before integration into the airframe. Ultimately, once sub-scale platforms and critical subsystems are validated, the final (and more costly) full-scale prototype is produced with greater confidence in its performance and safety.

In the next subsections, a concise overview is provided of the key design decisions and prototypes that guided the project toward a final configuration capable of meeting both the magnetic interference constraints and the operational performance targets.

2.2.1 Configuration Research

Before any airframe design work commenced, a thorough market survey was conducted to evaluate whether any off-the-shelf UAV systems could fulfill the stringent requirements of this project. Although numerous surveillance-oriented UAV platforms (e.g., those carrying electro-optical (EO), infrared (IR), LiDAR, or radar sensors) exist, few address the unique challenges posed by integrating a highly sensitive MAD sensor. In particular, standard powertrains (e.g., internal combustion engines, electric motors), actuators, avionics, high-current wiring, ferromagnetic components, and even batteries can each introduce magnetic noise, compromising the accuracy of MAD sensor readings.

Some UAVs do carry or have demonstrated integration with MAD sensors (e.g., PD-1 UAV [86], Brican TD 100 [87], HAASW UTAS [88], MQ-8 Fire Scout [89], Geoscan 201 [90], and GeoSurv II [91]). However, these platforms typically enforce less strict magnetic signature requirements than demanded by the latest generation of MAD sensors such as MAD-XR. In addition, many such UAVs remain unavailable for direct purchase, with design details often classified due to ongoing military or governmental applications. Given these constraints, no existing UAV met all the necessary performance criteria. A bespoke design was therefore required.

To achieve the desired combination of low magnetic signature, VTOL capability, and rigid mounting of the MAD sensor, the design team employed an iterative methodology in line with established aircraft design principles [92, 93]. Detailed in [72], the configuration research focused on identifying an aerodynamic layout that would:

1. Maximize distance between magnetic sources (e.g., motors, batteries, ferromagnetic components) and the sensitive MAD sensor.

2. Offer VTOL functionality for ship-based operations and remote site deployments.

3. Minimize relative sensor motion with respect to the aircraft's center of gravity (and IMU), thereby simplifying data post-processing for target localization.

After comparing a range of fuselage and wing arrangements, a *canard* configuration with a pusher propeller was selected, driven by the need to isolate major sources of magnetic interference near the aft section of the fuselage and free the nose for integrating the MAD sensor. Canards also provide the advantage of placing most of the control surfaces far from the sensor, which aids in maintaining minimal ferromagnetic presence in the forward section of the aircraft. Alternative concepts, such as wingtip sensor placement or a long tail-boom, were deemed less optimal due to higher structural flex, complex mass balancing, and potential degradation of magnetic data fidelity. A *tail-sitter* approach was also considered but ultimately discarded, primarily because landing in high winds or on a moving platform becomes more challenging for tail-sitter configurations with control surfaces oriented perpendicular to the wind.

An equally critical design choice was the VTOL propulsion layout. As illustrated in Figure 2.4, four candidate propulsion architectures were explored in detail (see [82]). After static bench testing each option, the decision was made to adopt a *quad-rotor lift + pusher* layout for the first prototype, appreciating its mechanical simplicity and well-understood control dynamics. In parallel, work proceeded on a smaller-scale, tri-rotor POC with a tilting pusher assembly. Although more complex, the tri-rotor approach can reduce the aircraft's overall power overhead during cruise, an important factor for extending flight endurance.

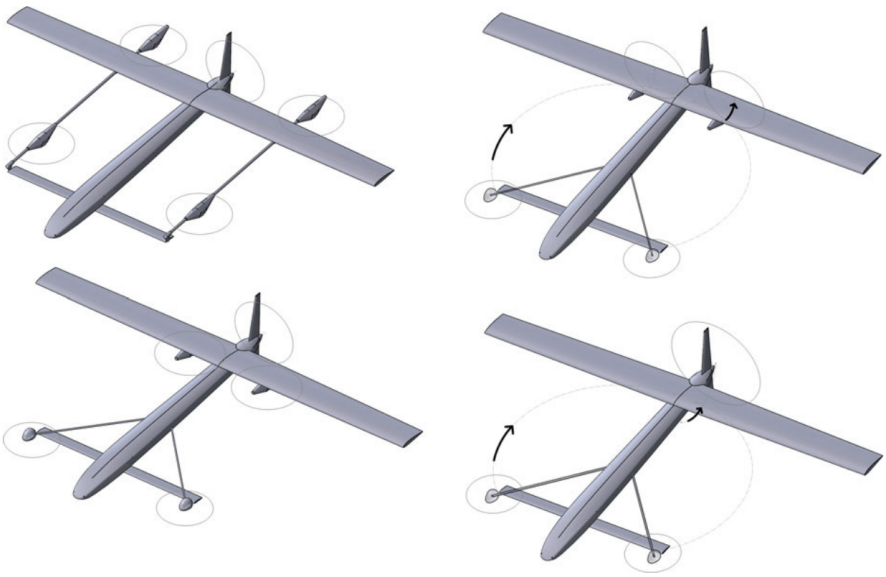


Figure 2.4: Four candidate VTOL propulsion configurations considered for the Eusphyra vehicle.

The resulting platform was named Eusphyra, referencing the Greek term for the wing-headed shark, *Eusphyra blochii*. The final airframe integrates a forward-mounted canard reminiscent of the shark's cephalofoil and, like this marine creature, can effectively "hover" for vertical takeoff and landing without requiring continuous forward motion.

In the following subsections, the progression from initial sub-scale prototypes to more sophisticated vehicles is outlined, illustrating how iterative testing provided critical insights into aerodynamic performance, system integration, and magnetic signature management. Each prototype contributed lessons that refined subsequent designs, ultimately converging on a final configuration optimized to carry the MAD-XR sensor in demanding operational environments.

2.2.2 Mini-Eusphyra 5050A

Once the overall aircraft configuration had largely been determined, a subscale, simplified model was constructed to serve as a proof of concept. This model was sized to accommodate the QuSpin Total-Field Magnetometer (QTFM) [94] (the smaller, lighter, but less sensitive magnetometer previously flown on the Nebula N1 UAV), while remaining compact enough to be manufactured and tested locally. Of the candidate VTOL propulsion architectures discussed in [73], the simplest—a quadrotor arrangement with a dedicated pusher motor and symmetric lift distribution (50% of the takeoff weight carried by the forward motors and 50% by the rear motors, hence "5050")—was selected.

The prototype was developed and manufactured using rapid, cost-effective methods to identify any major issues with the chosen configuration. The airframe comprised primarily 3D-printed parts joined by carbon-fiber spars, avoiding the need for time-consuming composite layups. Although structurally minimalistic, this approach allowed researchers to gather valuable data on aerodynamic performance, stability, and control. Details on the design, manufacturing, and initial flight testing of this specific prototype are documented in [74]. Figure 2.5 shows the vehicle undergoing pre-flight checks.

Figure 2.5: Mini-E 5050A prototype during a pre-flight IMU calibration.

Most of the avionics were commercial off-the-shelf (e.g., autopilot, differential pressure sensor, inertial measurement unit, current and voltage sensors, LiDAR, and GPS receivers). The system ran a modified version of PX4 1.12 firmware, incorporating a tailored mixer matrix for the 5050A's propulsion layout. Test flights with this prototype exposed several design issues, notably with the landing gear, VTOL propellers, vertical empennage, and the canard assembly, including its range of actuation and mechanical drive components. Structural resonance was also identified, requiring subsequent stiffening strategies. These findings drove modifications and design iterations to improve stability, control authority, and sensor accuracy. Each modification necessitated a corresponding assessment of flight-controller performance and occasional retuning of control gains.

Upon resolving critical deficiencies, the 5050A successfully completed the planned test matrix for both the VTOL and fixed-wing (FW) phases of flight. With these objectives met, the team proceeded to integrate the QTFM payload. To accommodate further sensor trials and improve flight performance, a second version, the Mini-E 5050B, was developed with improved structural properties.

2.2.3 Mini-Eusphyra 5050B

The Mini-E 5050B introduced a carbon-composite primary structure, which required the fabrication of custom molds but provided significantly higher rigidity and reduced structural mass relative to the 3D-printed approach. This mass savings allowed for an expanded payload, determined in consultation with project stakeholders. In addition to the QuSpin QTFM, a Billingsley TFM100-G2 sensor [95] was installed in the aircraft nose, along with five current sensors for monitoring the control surfaces and pusher motor. A dedicated onboard computer (provided by the National Research Council, NRC) equipped with a separate GPS receiver was also integrated to record, time-synchronize, and store data from all newly installed devices.

Although the improved structural efficiency partially offset the increased system weight, additional payload capacity inevitably raised the aircraft's overall takeoff mass. A series of incremental flight tests with substituted ballast masses confirmed the feasibility of carrying the combined sensors. Minor modifications to the VTOL propeller pitch and a thorough retuning of the flight controller were sufficient to restore acceptable handling qualities. Following validation under both VTOL and FW operating modes, the Mini-E 5050B performed several magnetometer-based missions over targets with known magnetic signatures. These missions verified the airframe's suitability for MAD applications, demonstrating reliable detection and highlighting the viability of this configuration as a small-scale research platform. Figure 2.6 shows the vehicle taking off in one of the MAD sensing missions.

Figure 2.6: Mini-E 5050B in the VTOL phase during flight testing.

Concluding the 5050B testing, the team confirmed that the core design principles could indeed support an aircraft specialized for magnetic anomaly detection. Development efforts then shifted to exploring subsystems for the full-scale Eusphyra, focusing on alternative propulsion architectures, hybrid power configurations, and multi-agent capabilities, as well as evaluating the performance of these VTOL systems. Studies also extended to detailed aerodynamic and structural analyses of the ultimate full-scale vehicle.

2.2.4 Mini Tri-Rotor Eusphyra

As mentioned previously, the project's configuration research encompassed several VTOL propulsion layouts, initially favoring the simpler quadrotor-plus-pusher concept for the Mini-E 5050 series. Nevertheless, a novel tri-rotor architecture with tilting propellers was also investigated, owing to its potential for reduced drag and improved efficiency during forward flight. This layout redirects the primary propulsive force from the FW motor and thereby eliminates the need for dedicated lift motors during cruise, though at the cost of increased mechanical and control-system complexity.

In particular, tri-rotor configurations feature fewer redundant propulsors and additional failure points introduced by tilting mechanisms [77, 96]. These trade-offs necessitated a focused study on airworthiness. A small-scale tri-rotor proof-of-concept was thus developed to rigorously evaluate flight performance, mechanical integrity, and control responsiveness. The details of the tri-rotor airworthiness investigation, including design rationale and test results, are provided in Section 2.3 of this chapter.

2.2.5 Hybrid Sub-System

Achieving the endurance requirements demanded by this project proved infeasible with a purely electric powertrain. For this reason, alternative power sources were evaluated to fulfill the minimum threshold of five hours. This led to a side research effort aimed at identifying and testing viable hybrid power solutions for an airframe of the Eusphyra's scale. Various architectures were studied and examined on a dedicated test bench, including series hybrid internal combustion engine (ICE) systems [97], parallel ICE configurations [98], and hydrogen fuel cells [99]. An example test bench setup for a parallel hybrid system is shown in Figure 2.7.

Figure 2.7: Hybrid research test bench evaluating a parallel ICE configuration.

Based on performance results from these bench tests, a series ICE hybrid emerged as the most promising solution, balancing energy density, system complexity, and reliability. Detailed findings and the final system sizing for the full-scale aircraft are reported in [78].

2.2.6 MIMIQ

Development of the MIMIQ vehicle marked a significant step toward realizing the full-scale Eusphyra platform (Figure 2.8). MIMIQ—an acronym for *Mass and Inertia Matching Innovative Quadcopter*—is a modular UAS designed to replicate the take-off weight and inertia characteristics of the envisioned full-scale Eusphyra. Equipped with the same VTOL propulsive hardware (motors, propellers, electronic speed controllers (ESCs), and associated high-voltage components), MIMIQ enables rigorous testing of flight controllers, onboard avionics, and power subsystems under representative load conditions [80].

Figure 2.8: MIMIQ vehicle configured with a series ICE hybrid system.

The team designed MIMIQ to facilitate rapid swapping among different hybrid configurations, including both series ICE powertrains and hydrogen fuel cells, in order to evaluate their suitability for charge-sustaining operations and extended flight endurance. Flight tests with MIMIQ uncovered several significant issues, such as high-frequency vibrations transmitted from the ICE to avionics components, necessitating multiple redesigns of the engine mounts, adoption of vibration isolation strategies for critical avionics, and refinement of IMU filter settings.

Despite these challenges, MIMIQ achieved a projected endurance of 7.38 h for the full-scale Eusphyra when incorporating the VTOL and dash flight segments [78], an outcome that greatly informed the final design direction and validated the selection of a series ICE hybrid powertrain.

2.2.7 Multi-Agent Research

In parallel with the vehicle-specific prototypes, a dedicated subproject focused on developing and demonstrating multi-agent capabilities as a standalone subsystem. This strategy enabled rapid prototyping and testing of new cooperative control concepts without the flight vehicles already engaged in previously mentioned milestones. The initial phase of this research involved simulation-based studies of various coordination algorithms, aiming to enable a fleet of UAVs to navigate cooperatively without reliance on a permanent ground station. A *virtual leader* coordination scheme, in which a centralized guidance unit resides on one of the air vehicles, formed the basis of the first implementation [81].

Integrating this multi-agent control approach demanded a re-design of the command-and-control links and communication protocols. Notably, the Microhard P900 radio was selected for its air-to-air mesh networking capability, enabling each vehicle in the fleet to communicate with every other node and allowing ground-station access when within range. This initial multi-agent navigation system was installed on three smaller quadcopters, each outfitted with the same avionics suite used in the Eusphyra platform (including autopilot, LiDAR, and IMU) alongside a companion computer for the custom navigation controller and updated radio. The off-the-shelf Tarot 650 Ironman served as the primary airframe for these demonstrations (Figure 2.9).

Figure 2.9: Three Tarot 650 UAVs taking off for a multi-agent flight coordination demonstration.

To stress-test the proposed framework, a close-formation flight scenario was selected, requiring higher positional accuracy and real-time coordination updates under realistic flight conditions. Further details on the flight test results are reported in [84]. Building upon this proof of concept, Xu introduced an adaptive distributed model predictive control scheme for multi-agent formation tracking, engineered to be robust against communication faults [100–102]. Initially validated in simulation, this advanced cooperative control system is planned for future flight tests to confirm its performance in operational environments.

2.2.8 Full-Scale Eusphyra

In parallel with subsystem development, a comprehensive design effort began for the full-scale Eusphyra prototype, starting with a multidisciplinary optimization approach. Aerodynamic performance predictions benefited from prior flight data gathered on the Mini-E platforms, supplemented with computational fluid dynamics (CFD) studies. Figure 2.10 shows a representative CFD run for the full-scale airframe.

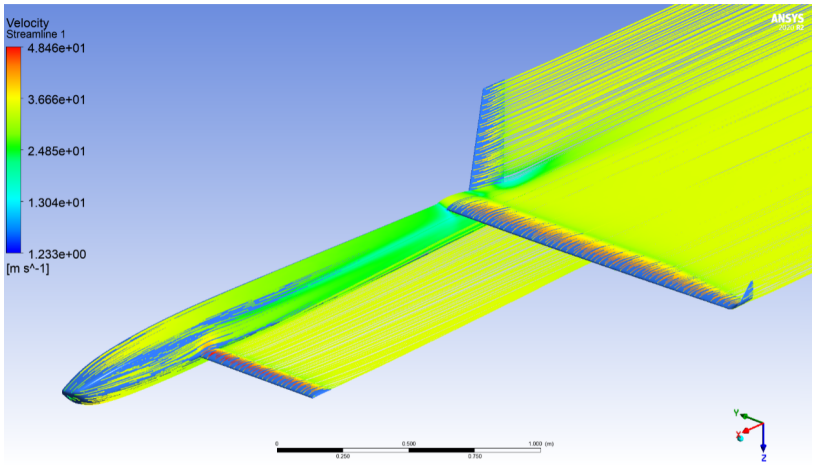


Figure 2.10: CFD analysis of the full-scale Eusphyra model in Ansys.

Control studies relied on high-fidelity simulations to assess autopilot performance during critical maneuvers and mission segments. Meanwhile, structural properties were investigated via finite element analysis (FEA) to ensure sufficient stiffness and load-bearing capability. These predictions were verified by manufacturing a full-scale wing section and subjecting it to physical load and failure testing (Figure 2.11). The wing tests validated numerical estimates of deflection and overall structural integrity.

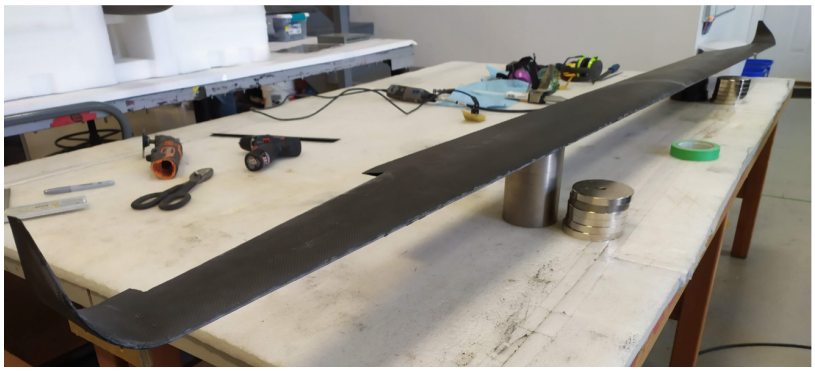


Figure 2.11: Full-scale Eusphyra wing under fabrication for structural testing.

Although the fundamental project mandate was satisfied by demonstrating a magnetically clean, VTOL-capable UAV with extended endurance, significant work continued to advance the Eusphyra platform toward Technology Readiness Level (TRL) 7. This phase includes refining the airframe, propulsion, and avionics subsystems for reliable operation in relevant environments, such as maritime or remote Arctic conditions. The resultant full-scale Eusphyra preliminary design represents a culmination of sub-scale validations, hybrid power development, and extensive simulation-based verification, illustrating a robust path from concept through near-operational readiness.

2.3 Investigation on the Airworthiness for the Tri-Rotor Configuration of the Eusphyra aircraft

The earlier sections of this chapter introduced the Eusphyra project, detailing the general aircraft development methodology and providing an overview of the various prototypes created and the rationale behind each design. These prototypes represent iterative steps in achieving a highly capable and efficient VTOL UAV, with each iteration contributing to the overall refinement of the platform.

This final section serves as a case study, showcasing the development and investigation of one of these prototypes: a tri-rotor configuration for the Eusphyra aircraft. This configuration was proposed to exploit the surplus power of the pusher propeller used in fixed-wing flight, with the aim of reducing the weight of the propulsive system and minimizing the vehicle's wetted surface. However, this design also introduced new challenges, including increased complexity and reduced redundancy.

The tri-rotor configuration was explored through a cost-effective, rapidly constructed prototype to evaluate its airworthiness and feasibility.

The investigation included the conceptual design, development, and testing of the tri-rotor prototype, as well as the application of a frequency-response-based system identification technique to generate a flight dynamics model. Ground and flight testing phases validated the operational capability of this novel configuration and informed future developments in VTOL UAV design. Additionally, custom flight control firmware in PX4 was developed to support the unique requirements of this configuration, demonstrating its potential for broader applications in multi-rotor/VTOL systems.

The contents of this section are based on the following publications:

- Arco, A., Lobo Do Vale, J., Bazzocchi, S., Suleman, A. (2023). *Investigation on the Airworthiness of a Novel Tri-Rotor Configuration for a Fixed Wing VTOL Aircraft.* International Journal of Aviation Science and Technology, 04(02), 53-62. [https://doi.org/10.23890/IJAST.vm04is02.0201]
- Arco, A., do Vale, J.L., Bazzocchi, S., Suleman, A. (2024). *Conceptual Design, Development, Test and System Identification of a Novel Tri-Rotor Configuration for a VTOL Fixed Wing Aircraft.* In: Karakoc, T.H., et al. Novel Techniques in Maintenance, Repair, and Overhaul. ISATECH 2022. Sustainable Aviation. Springer, Cham. https://doi.org/10.1007/978-3-031-42041-2_28

Abstract

In this paper, a novel tri-rotor configuration is proposed with the goal of granting vertical take-off and landing (VTOL) capabilities to a new concept of tiltrotor, fixed-wing aircraft, while minimizing both the overall mass of the propulsive system and the aerodynamic drag during horizontal flight. The novelty of the presented configuration lies not only in the thrust vectoring capabilities of all three rotors but also in the constraints placed on the rear rotor, which must provide thrust in both vertical and horizontal flight stages while drawing power from an internal combustion engine fixed inside the aircraft's fuselage. Additionally, this configuration features a 20/80 thrust distribution between the front and rear rotors during vertical flight, departing from the more conventional approach of evenly loading all rotors.

The proposed configuration was realized in a test vehicle that underwent multiple stages of ground and flight testing to evaluate the airworthiness of both the vehicle and

the concept. This process also involved developing a custom flight control firmware in PX4, enabling the operation of this or similar multi-rotor/VTOL systems. Finally, a frequency-response-based system identification technique was applied to the collected flight data to derive a suitable flight dynamics model for future autopilot tuning.

Keywords

Unmanned Aerial Vehicle, Vertical Take-Off and Landing, Tilt-rotor, Tri-rotor, System Identification

2.3.1 Introduction

The propulsive system configuration proposed in this paper is the result of a series of design constraints which were imposed during the conceptual design of the new, canard configuration, fixed-wing (FW) unmanned aerial vehicle (UAV) for which such a system is being devised (Fig. 2.12). The main requirement and motivation for this investigation is that the vehicle should be capable of performing vertical take-off and landing (VTOL) maneuvers, as well as being able to hover steadily upon request.

The propulsive system was conceptualized in a way that the two front rotors are to be powered by electric motors, with the frontal arms on which these are mounted retrieving into the fuselage of the aircraft during horizontal flight to optimize the aerodynamic efficiency of the vehicle. These rotors are meant to operate only in the vertical flight stages and are responsible for generating, in nominal flight conditions, around 10% of the total vertical thrust required to hover (i.e., enough thrust to balance 10% of the total aircraft's weight each). The third, rear tilting rotor, which will provide thrust in both the vertical and horizontal stints of the aircraft's mission should, in turn, be powered by an internal combustion engine. This engine is to be fixed inside the aircraft's fuselage, in a way that the rear rotor, responsible for generating enough thrust to balance the remaining 80% of the vehicle's weight in vertical flight, shall assume a tilt-shaft configuration. In order to achieve this, a tilting 90° gearbox system will be used. The tilting axis of said gearbox shall be aligned with the motor's output shaft (gearbox's input shaft) in order to allow for the proper transmission of power between motor and propeller regardless of the angle assumed by the propeller shaft.

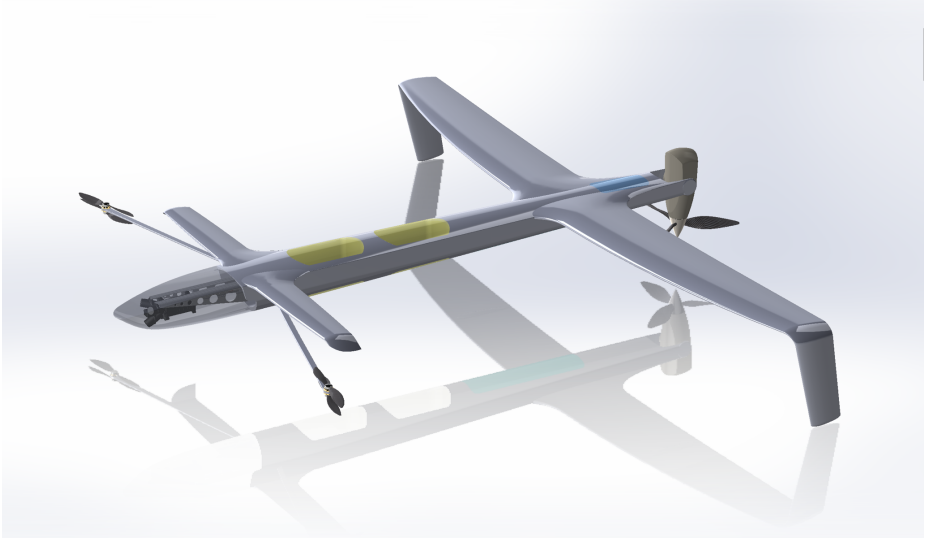


Figure 2.12: Preliminary CAD model of the conceptualized VTOL UAV.

When dealing with a multi-rotor configuration where the number of propellers is not even, and thus propeller pairing cannot be accomplished, a problem arises regarding the balance of the drag torques developed by each rotating propeller. While in multi-rotors with an even number of propellers these are usually paired among themselves (i.e., the same number of vertically pointing propellers rotating clockwise as counterclockwise), yielding that a balance of torques along the yaw axis is achievable as long as the rotational speed of any given clockwise rotor is matched to the one of a counterclockwise rotor. Furthermore, in such cases, these rotational speeds can be manipulated by the flight controller (as is often done) to maneuver the aircraft in yaw. Tri-rotor configurations, however, having an uneven number of rotors, must achieve yaw stability through other means, the most common one being thrust vectoring. This approach requires that one of the rotors (usually the aft one in a "Y" configuration) is capable of tilting laterally, thus developing a lateral thrust component and an associated torque along the yaw axis [103–106].

However, in the proposed configuration, the aft rotor shaft will already be required to tilt between the initial vertical stance and a longitudinal orientation during the forward flight stages. Hence, it was decided that the yaw attitude should be managed by a similar thrust vectoring approach but performed by the two frontal rotors. These will thus tilt along the longitudinal axis of the frontal arms of the tri-rotor simultaneously, which allows minimization of the tilting angle that is required from each front rotor to achieve the necessary equilibrium of torques in yaw. Furthermore,

the sense of rotation of the frontal rotors (counterclockwise) shall be opposite to that of the rear one (clockwise). This will allow minimization of the natural torque imbalance and, consequently, of the lateral thrust vector needed to achieve equilibrium, the tilting angle of the frontal rotors (assuming that the required vertical thrust is constant), and thus the amount of power required to operate the front rotors.

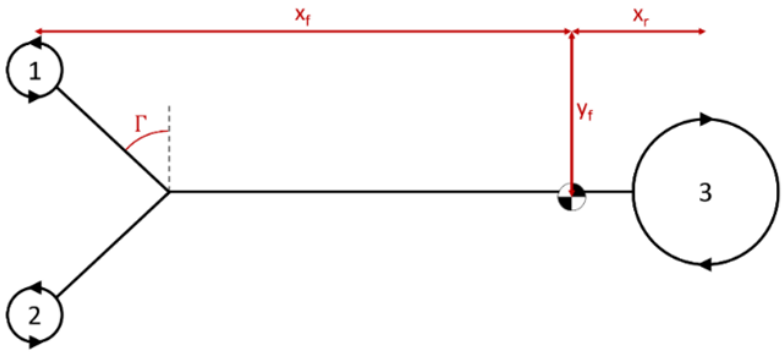


Figure 2.13: Schematic representation of the proposed tri-rotor configuration (not to scale).

Γ: Arms' opening angle
x_f: Longitudinal distance between front rotors and aircraft center of gravity
x_r: Longitudinal distance between rear rotor and aircraft center of gravity
y_f: Lateral distance between each front rotor and aircraft center of gravity

Given that the objective of the present study was to explore a new multi-rotor configuration for application to a VTOL, fixed-wing aircraft, it was deemed necessary to start the process by deducing the equations that would later support the development of a preliminary flight dynamics model (FDM). Once these equations were obtained, trim condition studies were performed in order to establish the conditions under which this aircraft configuration could perform stable hovering flight. Once an initial understanding of the configuration's capabilities was gained, the development of the test vehicle commenced with particular focus on the mechanisms required to perform thrust vectoring with all three rotors. These mechanisms, when built, were tested to map their actuation and to explore their performance. In parallel, a personalized PX4 autopilot firmware was obtained in order to control this novel-configuration aircraft in the subsequent test flights. Once the vehicle was completed, it was subjected to a battery of test flights to assess the airworthiness of both the vehicle itself and of the configuration as a concept. Finally, with the collected data, a frequency-response system identification method was applied to obtain a suitable dynamics model from

experimental data.

In the following subsections, this sequence of steps is explored in more detail.

2.3.2 Configuration Study

Flight Dynamics Model

With the goal of better comprehending the expected dynamics of the future fixed-wing (FW) vehicle that encompasses the proposed multi-rotor configuration, it was first necessary to derive the appropriate flight dynamics model (FDM). For the deduction of the equations that sustain this model, the standard body reference frame was considered. The vector that describes the attitude of the aircraft relative to a fixed North-East-Down referential is provided by the commonly used Euler angles, $\mathbf{\Phi} = [\phi\ \theta\ \psi]$, while the angular rates are $\boldsymbol{\omega} = [p\ q\ r]$ and the local body-speeds are $\mathbf{V} = [u\ v\ w]$.

The dynamic equations, describing the motion of the aircraft, were derived from the typical Newton–Euler equations [107]. Given that the next iteration of the vehicle will be electric (despite the final one having a hybrid propulsion system), the equations can be manipulated in such a way that a constant mass and inertia tensor are considered. The main distinction between the derived FDM and a common multi-rotor FDM resides in the formulation of the propulsive forces and moments.

$$
\begin{cases}
F_{x_{\text{prop}}} &= (-T_1 + T_2)\ \sin(\delta_{\text{arms}})\cos(\Gamma) + T_3\ \cos(\mu) \\
F_{y_{\text{prop}}} &= (T_1 + T_2)\ \sin(\delta_{\text{arms}})\ \sin(\Gamma) \\
F_{z_{\text{prop}}} &= -\left(T_1 + T_2\right)\ \cos(\delta_{\text{arms}})\ -\ T_3\ \sin(\mu) \\
M_{x_{\text{prop}}} &= (-T_1 + T_2)\, y_f\ \cos\left(\delta_{\text{arms}}\right)\ +\ (\tau_1 - \tau_2)\ \sin\left(\delta_{\text{arms}}\right)\cos(\Gamma)\ +\ \tau_3\ \cos(\mu) \\
M_{y_{\text{prop}}} &= (T_1 + T_2)\, x_f\ \cos\left(\delta_{\text{arms}}\right)\ -\ T_3\, x_r\ \sin(\mu) \\
&\quad -\ (\tau_1 + \tau_2)\ \sin(\delta_{\text{arms}})\ \sin(\Gamma) \\
M_{z_{\text{prop}}} &= (T_1 + T_2)\, x_f\ \sin(\delta_{\text{arms}})\ \sin(\Gamma)\ +\ (T_1 + T_2)\, y_f\ \sin(\delta_{\text{arms}})\ \cos(\Gamma) \\
&\quad +\ (\tau_1 + \tau_2)\ \cos\left(\delta_{\text{arms}}\right)\ -\ \tau_3\ \sin(\mu)
\end{cases}
\tag{2.1}
$$

Here, T_n and τ_n represent the thrust and drag torque magnitudes generated by rotor n, according to the numbering provided in Fig. 2.13, while x_f, x_r, and y_f

represent the distances depicted in the same figure. Furthermore, δ_{arms} is the tilting angle of the front rotors (0° for vertical), and μ is the rear rotor tilting angle (90° at vertical).

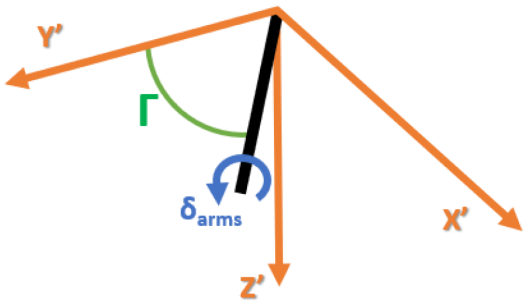


Figure 2.14: Schematic representation of the right frontal arm rotation angles.

In addition to the propulsive contributions in (2.1), the effects of aerodynamics [82], gravity, and gyroscopic moments [107] were also included to capture the complete dynamics of the aircraft.

Trim Conditions Analysis

Once the dynamics equations were derived, a trim condition analysis was carried out using a non-linear solver algorithm with the purpose of determining the attitude and actuator outputs required for the conceptualized test vehicle to hover steadily under various constant wind conditions. This algorithm, being based on an optimization approach, required the definition of a cost function. The methodology considered consisted of minimizing the sum of the mechanical power developed by the three rotors. Thus, the program was to find values for a set of predefined variables (regarding the vehicle's attitude and actuation) that would enable this minimization. The optimization problem was accordingly defined as follows:

$$\min \sum_{i=1}^{3} P_i, \quad \text{w.r.t.} \left[\phi,\ \theta,\ \Omega_1,\ \Omega_2,\ \Omega_3,\ \delta_{\text{arms}} \right] \tag{2}$$

subject to

$$\begin{cases} F_{x_a} + F_{x_{\text{prop}}} + F_{x_g} = 0, \\ F_{y_a} + F_{y_{\text{prop}}} + F_{y_g} = 0, \\ F_{z_a} + F_{z_{\text{prop}}} + F_{z_g} = 0, \\ M_{x_a} + M_{x_{\text{prop}}} = 0, \\ M_{y_a} + M_{y_{\text{prop}}} = 0, \\ M_{z_a} + M_{z_{\text{prop}}} = 0. \end{cases} \tag{4}$$

As for the ranges over which the optimization variables could vary, these are provided in Table 2.1.

Table 2.1: Study variables' boundaries.

Variable	**Lower Bound**	**Upper Bound**
ϕ	$-30°$	$30°$
θ	$-30°$	$30°$
Ω_1	0 rpm	8200 rpm
Ω_2	0 rpm	8200 rpm
Ω_3	0 rpm	6374 rpm
δ_{arms}	$-15°$	$15°$
δ_r	$-30°$	$30°$

Once the algorithm had been established, several simulations were performed in which the magnitudes and directions of (constant) incoming wind gusts were varied. The resulting attitude of the aircraft and expected actuation were recorded. This enabled conclusions on the expected tendencies of this novel configuration under different hovering flight scenarios, as well as providing an initial vertical flight envelope. That envelope contains the maximum allowable gust magnitudes for each direction, based on the actuation and attitude limits established (i.e., the allowable roll and pitch angles, the maximum tilt angle of the frontal rotors, and the maximum allowable rotational speed of all three rotors, as shown in Table 2.1).

The first study using the developed optimization tool focused on determining the attitude and actuation assumed by the fixed-wing vehicle (FW VTOL) under zero-wind conditions. The results of this simulation are presented in Table 2.2.

Table 2.2: Attitude and actuation during steady hovering flight under zero-wind conditions.

Variable	**FW VTOL**
ϕ	$-0.338°$
θ	$0°$
Ω_1	6047 rpm
Ω_2	6047 rpm
Ω_3	4909 rpm
δ_{arms}	$2.484°$
δ_r	$0°$

From the analysis of these results, it is possible to draw several conclusions regarding the behavior of the configuration in hover. Even though this may not hold for all vehicles that assume this multi-rotor configuration, for the present study case it was concluded that the drag torque developed by the rear rotor in hovering flight conditions surpassed the accumulated drag torques of the front rotors. Given the sense of rotation defined for the various propellers in Fig. 2.13, this results in a net drag torque that is non-zero in magnitude and assumes negative values along the vertical (yaw) axis of the aircraft's local reference frame.

Consequently, to counterbalance this phenomenon by means of thrust vectoring, the front rotors must tilt to the right, producing a lateral (positive Y) thrust component and an associated positive torque along the yaw axis. This balances the net drag torque, and is manifested in the positive value of δ_{arms}. However, while this lateral thrust vectoring balances yaw torque, it introduces an imbalance of forces along the local Y axis. To avoid drifting to the right, the vehicle simultaneously rolls slightly to the left, allowing the lateral thrust to be offset by a component of the aircraft's weight. Meanwhile, the pitch angle remains approximately zero. The rotational speeds of the different rotors depend on the powertrain characteristics of each specific vehicle. Should the balance of torques along the yaw axis shift (e.g., if the frontal rotors produce more drag torque than the rear rotor), the exact hovering attitude and actuation required will naturally change, but the underlying logic driving this behavior remains the same.

The devised optimization tool was also used to obtain a preliminary vertical flight envelope regarding the gust magnitudes (and directions) under which the conceptualized vehicle could operate while hovering in a fixed position and respecting the

operational boundaries from Table 2.1. Such results are shown in Table 2.3.

Table 2.3: Maximum allowable gust magnitudes by direction for the FW VTOL in hover, given the operational limits of Table 2.1.

Variable	**FW VTOL**
Front	> 10
Rear	> 10
Left	4
Right	4
Up	7
Down	4

Finally, an additional study on the relation between the angle assumed by the rear rotor and the forward speed achieved during a hypothetical transition maneuver was carried out using a modified version of the same optimization tool. In this scenario, we included the rear rotor tilting angle as an extra optimization variable, aiming to minimize total mechanical power while maintaining level flight at constant altitude under no-wind conditions. The aerodynamic model assumed a null pitch angle throughout the transition and a canard incidence matching that of the trimmed flight condition at cruise speed. The resulting optimal transition curve is plotted in Fig. 2.15.

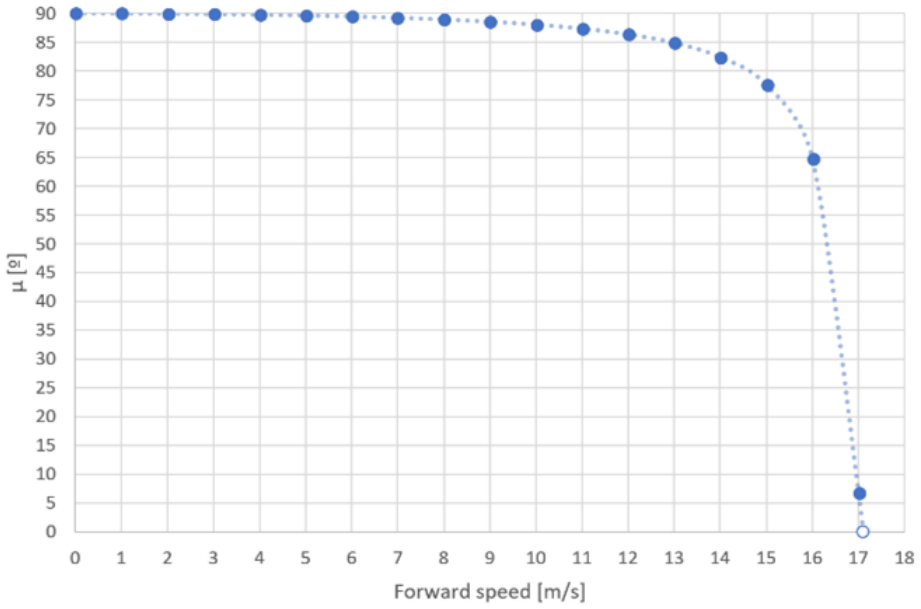


Figure 2.15: Relation between rear rotor tilting angle (μ) and the forward speed achieved by the aircraft for the optimal transition scenario.

μ: Rear rotor's tilting angle

Considering that the theoretical stall speed for the conceptualized FW VTOL aircraft has been established at 13 m/s [82], it was found that such forward flight condition could be reached with a rear rotor tilt angle as low as 5°, assuming sufficient time for the vehicle to accelerate. From Fig. 2.15, once the theoretical stall speed is achieved, there is a noticeable ramp-up in the progression of the rear rotor's tilting angle with forward speed. This behavior arises from two factors: (1) increasing forward velocity augments the contribution of aerodynamic surfaces (wing and canard) toward lift generation, thereby reducing the required vertical thrust from the rotors (in fact, rotor speed is gradually reduced for all rotors during transition); and (2) as forward speed grows, the horizontal drag likewise increases, requiring more horizontal thrust to maintain speed, which means reducing the tilting angle of the rear rotor in the later stages of the transition.

2.3.3 Vehicle Design

Flight Controller Development

In order to perform proof-of-concept flights to attest the airworthiness of the proposed configuration, a test vehicle was built, and a custom flight control firmware was developed. The latter step was carried out in the PX4 environment by creating a custom airframe configuration. Although two pre-existing "Y"-shape tri-rotor flight controllers were available in the PX4 repository, neither was applicable to the proposed configuration for three main reasons:

1. Both pre-existing models assumed that all three rotors were equidistant from the aircraft's center of gravity (CG), whereas the presented case features a 20/80 thrust distribution between the front and rear rotors.

2. The sense of rotation of the rotors differed from those envisioned for the proposed configuration.

3. The pre-existing controllers presumed that yaw control was dependent on lateral tilting of the rear rotor, while in our configuration, this task is assigned to the frontal rotors' tilting mechanism.

Once the characteristics of the proposed configuration were translated into the required PX4 files and the firmware was compiled, the resulting autopilot was flashed onto a Pixhawk 4 board. Communication with the hardware was achieved via the QGroundControl software.

A primary concern during this development was the neutralization of possible coupled dynamics. This is particularly relevant because the sense of rotation of the frontal rotors is opposite to that of the rear rotor. Although this design choice was intentional to minimize the natural drag torque imbalance along the yaw axis (and thus reduce the lateral thrust vector needed to achieve torque equilibrium, consequently minimizing overall power in hover), it presented a potential undesirable effect. This effect stemmed from the conventional approach of multirotor flight controllers toward yaw attitude management, which typically involves adjusting the rotational speed of specific rotors (depending on their sense of rotation) to induce yaw. In the proposed configuration, however, applying a similar yaw control method would inadvertently couple into the pitch axis.

To solve this, and to exploit the thrust vectoring capabilities of the front rotors, complete yaw control authority was assigned to the front tilt mechanism. Meanwhile, the flight controller was prevented from varying the rotational speeds of the rotors when yaw attitude changes were required. Consequently, pitch, roll, and heave remain dependent on the manipulation of rotor speeds (as is standard practice in multirotor controllers), whereas the yaw axis is governed solely by the frontal tilt mechanism.

Tilt Mechanisms Development

In the context of the test vehicle's design, the two subsystems that warrant particular attention, given their novelty and importance to the function of the proposed concept, are the tilt mechanisms for the front and rear rotors. These mechanisms were developed from an initial conceptual sketch through to the final prototyping stage.

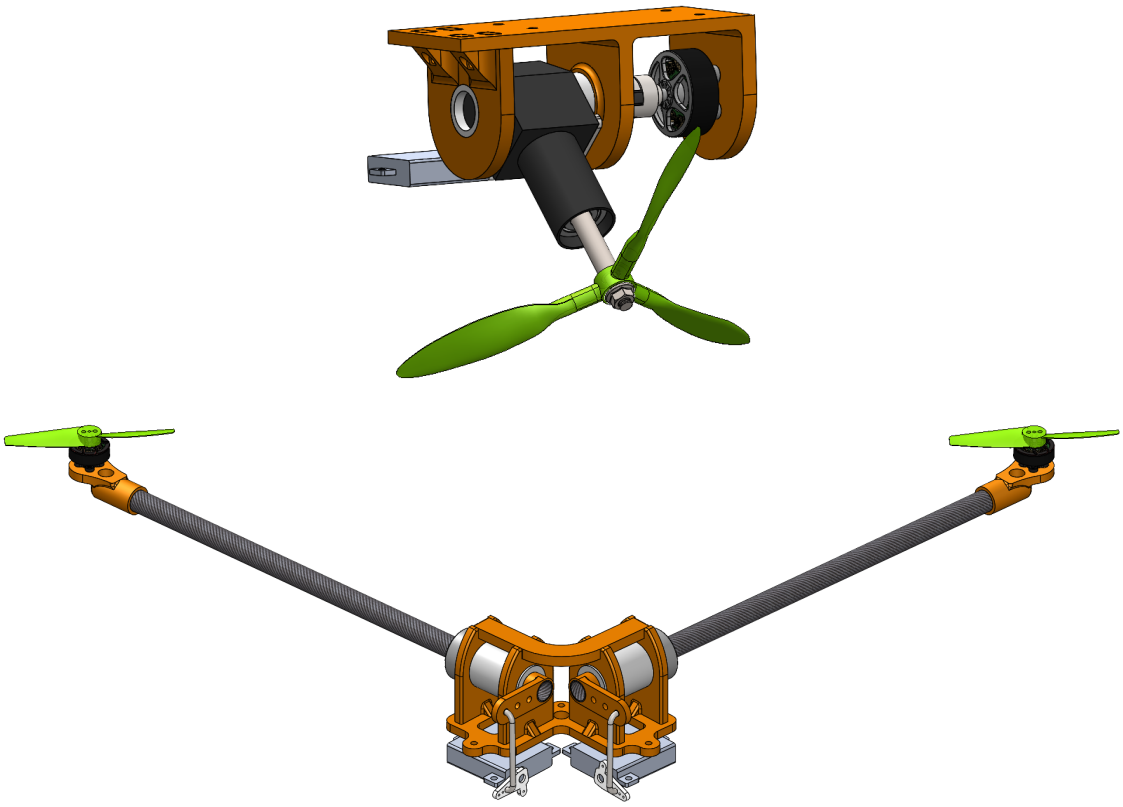


Figure 2.16: CAD models for both the rear (top) and front (bottom) rotor tilting mechanisms.

Rear Rotor Mechanism. The rear rotor design was pursued with the overarching intention of emulating the use of an internal combustion engine (ICE) to power the rear rotor, as required in the future, larger-scale version of the fixed-wing VTOL aircraft. It was also determined that this power unit should remain fixed relative to the airframe, thus necessitating a 90° gearbox to transfer mechanical power from the stationary power source to the output shaft of the system, which tilts. The propeller is mounted on this tilting output shaft. Naturally, as in any mechanical system, this gearbox will introduce losses that affect the overall efficiency of the rear rotor assembly. This aspect will be examined in a later stage of this work.

Front Rotors Mechanism. Regarding the front rotors' tilting mechanism, one of the main considerations was that the chosen design should allow for future modifications enabling a retraction of the frontal arms of the tri-rotor during horizontal flight in the full-scale FW-VTOL aircraft. Consequently, it was deemed necessary to equip each arm with its own dedicated actuator (servo). This choice not only provides redundancy in the yaw attitude management system (in case of a servo failure) but also enables the support structure of the tilt mechanism (shown in orange in Fig. 2.16) to be separated into two symmetrical parts capable of rotating independently, facilitating future integration of a suitable retraction mechanism. This additional mechanism was not developed for the current multirotor because it has negligible influence on the vehicle's dynamics (beyond the added weight contribution), and thus was not critical to proof-of-concept tests.

Ground Testing

Once the different mechanisms were built, their actuation was measured and mapped before being installed on the airframe. This process made it possible to update the previously devised flight dynamics model (FDM) and collect valuable data for future interpretation of flight logs. Static thrust testing was likewise conducted on both the front and rear rotor systems with the same objective. In addition, the study on the performance loss caused by introducing a 90° gearbox between the rear electric motor and the propeller shaft was performed, comparing the power required to generate a given thrust in two scenarios: a direct drive configuration and the complete rear tilt mechanism system.

To obtain these results, a set of static thrust tests was first conducted with the rear rotor's propeller mounted conventionally on the motor shaft. Then, once the tilt

mechanism had been assembled, the gearbox was fixed in place using two wedges to align the output propeller shaft with the static thrust test bench, allowing further thrust tests to be performed. The results are shown in Figure 2.17.

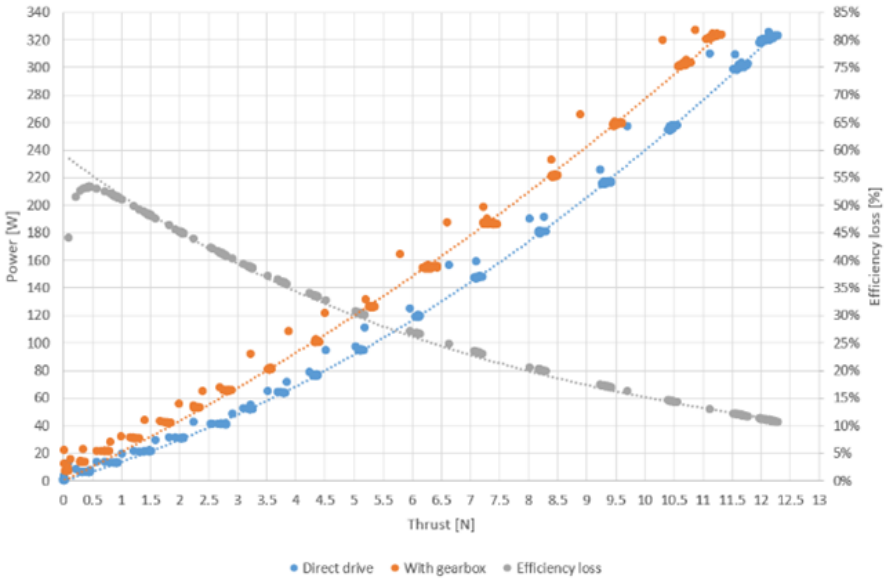


Figure 2.17: Static thrust test results for the rear ESC/motor/propeller system in direct drive and when fitted to the rear tilting mechanism.

From these data, there is a clear increase in the electrical power required by the motor in the presence of the complete rotor tilt system, as expected. It was estimated, for instance, that to generate the rear rotor thrust required for hovering flight, the additional power demanded by the tilt-mechanism-equipped system (relative to the direct drive case) is approximately 19%. This reduction in efficiency is primarily due to losses introduced by the 90° gearbox. In fact, the efficiency-loss curve obtained matches the typical gearbox efficiency behavior reported by several authors [108, 109], who observed that mechanical transmissions tend to reduce losses under increased loading conditions.

Additional ground tests were performed to assess continuous-operation static thrust for the rear rotor system. In particular, these tests monitored the gearbox's gears and the electric motor's temperature under sustained operation. The main motivation was the possibility that, with the motor no longer positioned in the propeller wake (as originally designed), higher temperatures could arise and risk magnet demagnetization [110, 111]. However, the measured temperatures stayed well below the manufacturer's maximum allowable values, indicating that the subsystem could be

considered safe and was thus integrated into the vehicle's airframe.

2.3.4 Flight Testing

After all subsystems were tested individually and the prototype was fully assembled, flight testing commenced. The initial flights were conducted indoors with the prototype tethered to ensure the safety of the personnel involved and to protect the vehicle during the early test phases (see Fig. 2.18). These indoor flights allowed tuning of the various aircraft subsystems, from the controller parameters to airframe and power distribution refinements, all of which underwent continual upgrades throughout the preliminary testing phase. Once the vehicle's performance and controllability were deemed satisfactory, the tethers were removed, and the team proceeded with the first untethered flight. In this test, the aircraft demonstrated stable hovering capability while maintaining position (as required for a vertical take-off or landing maneuver) and was also able to maneuver effectively around all three attitude axes. The pilot reported that the vehicle was easily controllable and displayed predictable dynamics despite its unconventional design.

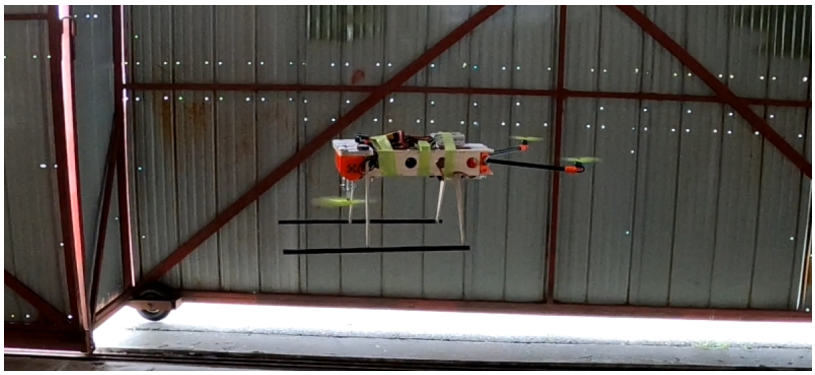


Figure 2.18: Test vehicle during indoor, untethered flight testing.

Encouraged by these results, the team proceeded with outdoor flight tests (see Fig. 2.19). The first set of outdoor experiments involved five flights, each comprising maneuvers such as chirps, doublets, and periods of static hovering. The objective was to collect data for subsequent system identification. An initial practice flight allowed the pilot to become familiar with the maneuvers and the aircraft's tendencies. This was followed by three flights, each focusing on one attitude axis (roll, pitch, or yaw). In each flight, the pilot executed multiple chirps and doublets on that specific axis, interspersed with periods of hovering. Finally, a validation flight was conducted in

which chirps and doublets were performed on alternating axes, ensuring at least one example of each maneuver across all three axes. These validation flight data would later be used to check the robustness of the flight dynamics models derived from system identification, providing a test scenario different from the ones that generated the models.

The only remaining configuration capability to be tested was initiating and maintaining stable forward flight by tilting the rear rotor system. Since the experimental vehicle lacks the fixed-wing surfaces of a conventional aircraft, a complete transition from vertical to horizontal flight was not possible. Instead, the goal of this final test flight was solely to assess whether the multi-rotor configuration could commence a simulated transition maneuver. To that end, the aft rotor was tilted by 5°, based on prior parametric studies suggesting that such thrust vectoring would allow the aircraft to maintain level attitude and constant altitude while gaining forward momentum, as would be required in a true transition maneuver.

Figure 2.19: Test tri-rotor during outdoor flight testing.

The results were highly satisfactory, with the aircraft proving capable of initiating and sustaining stable forward flight and requiring minimal pilot input for attitude corrections. These findings support the feasibility of the proposed multi-rotor configuration for both stationary and transition flight phases, thereby validating the fundamental design concepts.

2.3.5 System Identification

After collecting the necessary flight data from the tests described in Section 2.3.4, the next objective was to derive a suitable dynamics model from these experimental data. The chosen approach for this work was based on a frequency-response model, following the methodology described by Tischler and Remple [24]. In practice, this method was implemented in a custom transfer-function estimation tool that took as inputs:

- The rate setpoints of the flight controller for each of the three attitude axes (roll, pitch, and yaw),
- The corresponding angular rate responses recorded by the onboard gyroscope.

Additionally, for each flight test data file, the user supplied the time intervals corresponding to specific maneuvers within that test. The data from the three system-identification (SID) flights (one per axis) were then partitioned so that each data segment encompassed a single maneuver on a single axis. Next, all possible combinations of these maneuvers for a given axis were compiled, yielding several simulated "flight examples." Each such example was used to estimate a single transfer function relating the relevant input-output pairs.

Afterward, each derived transfer function was tested against the separate validation flight dataset (which was not used during transfer-function estimation) for the same axis. The transfer function that provided the highest correlation (i.e., best reproduced the validation flight outputs when driven by the same inputs) was deemed to represent the aircraft's dynamic behavior on that axis.

It should be noted that this model is linear and thus ignores coupled dynamics, which may reduce fidelity, given that the proposed concept exhibits some notable couplings. Although certain coupling tendencies were mitigated in the flight controller design (as discussed in Section 2.3.3), others could not be fully neutralized. One of the more prominent couplings was observed between pitch and yaw, particularly noticeable during high-frequency pitch maneuvers. In such cases, the rapid variation in thrust required of the front rotors to execute pitch chirps inadvertently induced an uncommanded yaw-rate chirp (of the same frequency but smaller amplitude). This was a direct consequence of the thrust-vectoring scheme used for yaw control, which, while beneficial for managing yaw in vertical flight, also introduces coupling paths.

Nonetheless, the frequency-response-based dynamics model obtained via this approach is sufficiently accurate for many practical purposes, such as axis-specific autopilot tuning and optimization. For configurations less prone to dynamic coupling, this method yields an even closer match between simulated and real flight dynamics. The goodness-of-fit values (G) achieved in the present study are shown in Figures 2.20, 2.21, and 2.22.

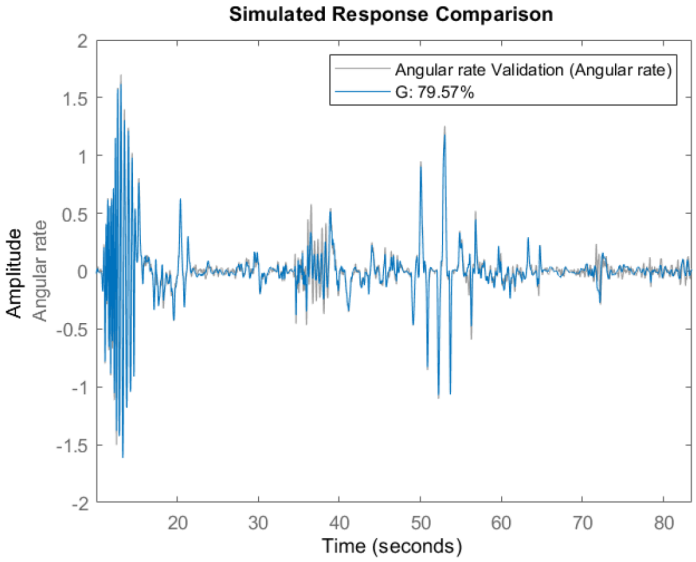


Figure 2.20: Comparison between replicated and recorded **roll** dynamics during the validation flight.

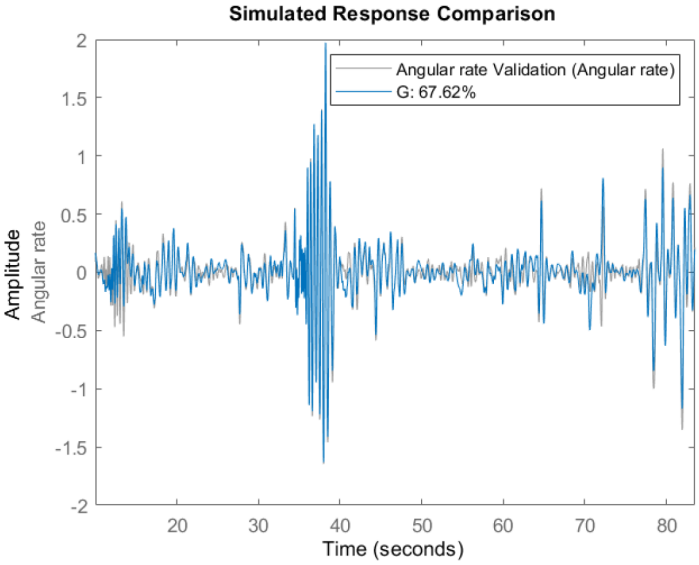


Figure 2.21: Comparison between replicated and recorded **pitch** dynamics during the validation flight.

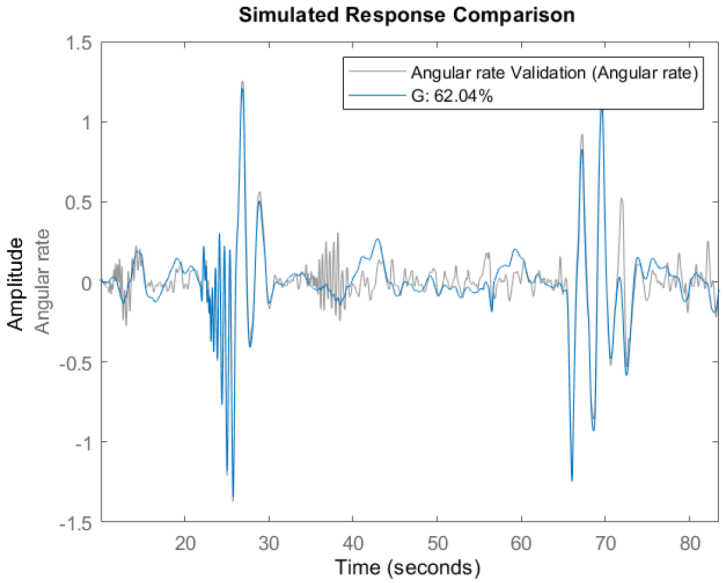


Figure 2.22: Comparison between replicated and recorded **yaw** dynamics during the validation flight.

2.3.6 Conclusions

The work presented in this paper characterizes the airworthiness of the proposed multi-rotor configuration for future application in the VTOL system of a fixed-wing aircraft. An initial study on the expected behavior of a potential fixed-wing VTOL aircraft equipped with the proposed tilt tri-rotor propulsive system was performed. This study provided:

- An understanding of the vehicle's attitude and actuation during hovering, stationary flight under various wind conditions,
- A preliminary vertical flight envelope regarding the maximum acceptable gust magnitudes (from multiple directions) that still permit stable hover,
- Insights into the expected attitude and actuation of the conceptualized vehicle during a transition from vertical to horizontal flight (via the rear rotor tilting).

Additionally, the proposed solutions for both the front and rear rotor tilt mechanisms were successfully developed and validated through ground tests. The complete prototype was then flight-tested, both indoors and outdoors, demonstrating:

- The ability to sustain continuous and stable hovering flight under external disturbances,
- Adequate maneuverability along all three attitude axes over a range of frequencies,
- The successful initiation and maintenance of stable forward flight by tilting the rear rotor, thus paving the way for future transition-condition studies of the larger hybrid aircraft under development.

Finally, a custom frequency-response-based system identification tool was applied to the collected flight data. Despite neglecting some of the coupled dynamics inherent to the tilt tri-rotor configuration, the resulting model still provided satisfactory fidelity when replicating real flight test dynamics, showing the practicality of this approach for future autopilot optimization and simulation studies.

Chapter 3

Flight Dynamics Modeling

In the chapter preceding this, we explored the development of the Eusphyra project and its associated prototypes, culminating in the Mini-E 5050B. This prototype achieved Technology Readiness Level 6 (TRL 6), demonstrating readiness for operational environments. Achieving this milestone required mission simulations and refined control system tuning to meet the stringent flight requirements of its payload. To address this, a classic control development cycle was employed, beginning with an initial flight dynamic model (FDM) generated from computational fluid dynamics (CFD) data. This model was preliminarily tuned and validated using the Mini-E 5050A prototype. The same controller was then used for the initial flights of the Mini-E 5050B vehicle that incorporated pilot-generated system identification maneuvers.

This chapter focuses on developing a nonlinear flight dynamic model for the fixed-wing portion of the eVTOL aircraft, using flight test data collected from the first 20 flights of the Mini-E 5050B. The methodology accounts for the limited data available from onboard systems, especially regarding the aero-propulsive system, and emphasizes generating a high-fidelity model validated against out-of-sample flight data. While this approach yields highly accurate models, it requires substantial data, time, and human supervision. Despite these challenges, it remains one of the most reliable methods for achieving high model fitness. Subsequent chapters will explore alternative methods aimed at reducing the workload and data requirements associated with this process.

The content of this chapter is based on the following published journal article:

- Figueira, J. C., Bazzocchi, S., Warwick, S., Suleman, A. (2024). *Nonlinear*

Aero-Propulsive Modeling for Fixed-Wing eVTOL UAV from Flight Test Data. In: *Journal of Aircraft*, pp. 1-13. [https://doi.org/10.2514/1.C037964]

Abstract

This paper presents a methodology for constructing an aero-propulsive system identification model for a fixed-wing propeller-driven aircraft with limited flight data. Developing a flight dynamics model is an iterative process involving costly and time-consuming flight testing to collect relevant data. To maximize the utilization of available data, this study employs a time-domain system identification procedure on flight data from various maneuvers and flights. The methodology incorporates multivariate orthogonal functions to capture the nonlinear terms representing the coupled effects of the propeller and airframe dynamics. A stepwise regression is then employed to identify the most pertinent model parameters. Estimation of variables associated with the propulsive contribution is accomplished through an electrical propulsive model, identified using command and battery data from static thrust and flight tests. Aero-propulsive derivatives are determined using the output-error method, where the accuracy is assessed based on a colored noise correction. To validate the predictability of the flight dynamics model, out-of-sample data is employed. The construction of the electrical propulsive model and identification of aerodynamic derivatives from flight data were specifically carried out for a particular eVTOL flight test vehicle; however, the techniques employed are generalizable and applicable to other aircraft models.

Nomenclature

a_x, a_y, a_z	=	measured body-axis translational acceleration, m/s^2
b	=	projected wingspan, m
C_Q	=	propeller torque coefficient
C_X, C_Y, C_Z	=	body-axis force coefficients
C_l, C_m, C_n	=	body-axis moment coefficients
$\bar{c}$	=	mean aerodynamic chord, m
D	=	propeller diameter, m
g	=	gravitational acceleration, m/s^2

I_x, I_y, I_z, I_{xz}	=	aircraft moments of inertia, kg· m ^{2}
J	=	advance ratio
L, M, N	=	applied body-axis moments, N·m
m	=	aircraft mass, kg
p, q, r	=	body-axis angular velocity, rad/s
$\bar{q}$	=	dynamic pressure, N·m^2
S	=	wing reference area, m^2
u, v, w	=	body-axis translational velocity, m/s
u_p, v_p, w_p	=	local propeller velocity, m/s
V	=	true airspeed, m/s
V_N, V_E, V_D	=	Earth-fixed translational velocity components, m/s
X, Y, Z	=	applied body-axis forces, N
α	=	angle of attack, rad
β	=	angle of sideslip, rad
$\delta_c, \delta_a, \delta_r$	=	control surface deflections, rad
η_t	=	throttle command, %
φ, θ, ψ	=	inertial orientation Euler angles, rad
Ω_p	=	propulsor rotational speed, rad/s

Superscripts

^{T}	=	transpose
$\dot{}$	=	time derivative
$^{-1}$	=	matrix inverse
$_{CG}$	=	center of gravity

Subscripts

$_c$	=	centered value
$_o$	=	bias or reference value

3.1 Introduction

System Identification (SID) from flight data has proven to be the most accurate approach for constructing a high-fidelity simulation model of an aircraft [112, 113]. Understanding the dynamics of an aircraft is crucial for its successful flight. However,

building, optimizing and validating a Flight Dynamics Model (FDM) is often an iterative and resource-intensive process [114].

The focus of this investigation centers on the Eusphyra aircraft (Figure 3.1), an electric Vertical Take-Off and Landing (eVTOL) platform featuring a canard configuration, intricately designed to meet stringent requirements [82]. When in VTOL mode, the aircraft functions as a quadcopter. In Fixed-Wing mode, the propulsive system consists of a rear propeller. This specific Unmanned Aerial Vehicle (UAV) has been chosen as the primary subject for conducting extensive research aimed at testing and validating advanced control strategies and online system identification techniques, as outlined in [115].

The aircraft uses an off-the-shelf autopilot tuned with a model derived from analytical analysis. The same aerodynamic model did not align with real flight data. To advance ongoing research, a new FDM is essential to facilitate testing novel techniques in a simulation environment, validated through subsequent flight tests.

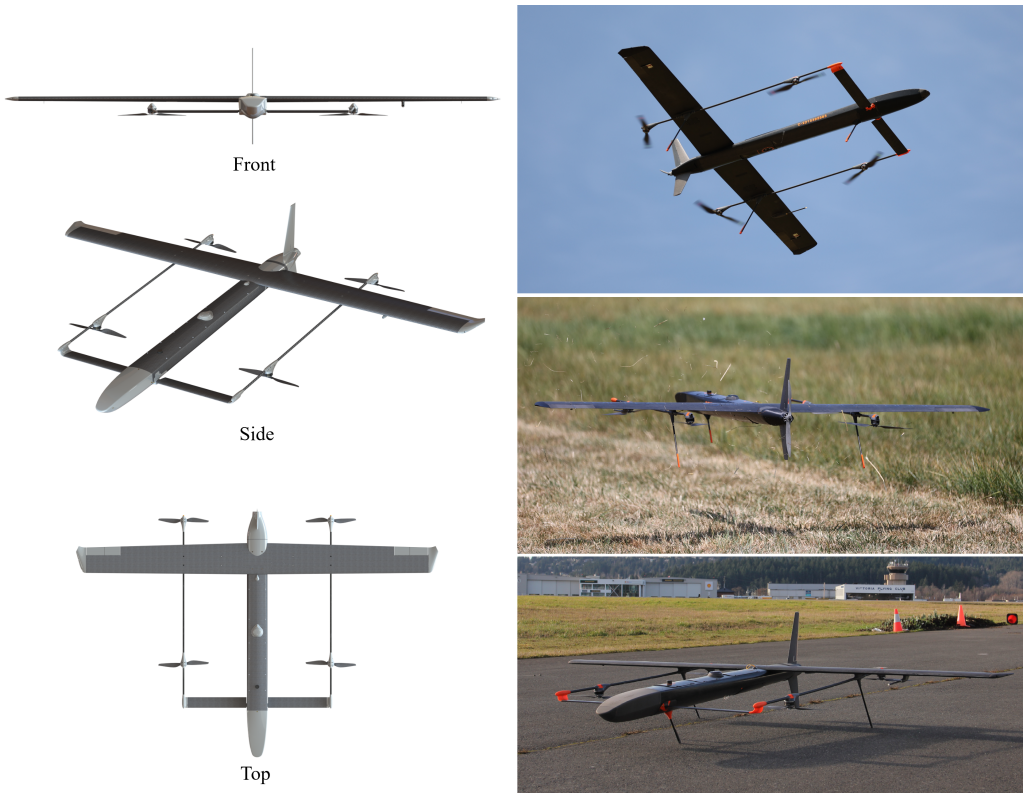


Figure 3.1: Eusphyra aircraft.

The development of an FDM for the fixed-wing configuration of the Eusphyra faced two main challenges: determining the propulsion contribution and addressing

constraints during flight testing due to a lack of accurate *a priori* data.

The Eusphyra's fixed-wing configuration relies on a propeller-driven system, introducing potential interactions between the airframe and propeller airflow. Achieving a coupled propulsion and airframe model enhances simulation accuracy [112]. Reference [116] outlines two approaches for coupled and decoupled aero-propulsive model identification from flight data. The coupled approach involves exciting all control surfaces along with the throttle, while the decoupled approach uses data from maneuvers at no throttle to capture bare-airframe dynamics. This paper follows the coupled approach due to the unmet conditions required for the second approach.

In the present propulsion configuration, the propeller rotation rate cannot be directly measured during flight. To address this issue, the propulsive system comprising an Electronic Speed Controller (ESC), electrical motor, and propeller blade is modeled. ESC losses are estimated as the propulsive system underperformed during flight testing compared to expectations. The model uses the commanded throttle and battery voltage logged in the flight data as inputs.

Multiple-input designs, such as orthogonal multisine input design, have been employed in SID procedures to enhance aircraft dynamics excitation [117]. Although multisine inputs offer advantages over traditional inputs [118], they necessitate an automated system to command the aircraft. Considering safety concerns, the maneuvers were conducted in closed-loop (stabilized mode), supervised by the pilot, to ensure a certain level of safety during perturbation injection. Closed-loop maneuvers, while safe, introduce challenges such as input distortion, data collinearity, and limited excitation beyond the feedback control. However, it is still feasible to employ open-loop parameter estimation processes from closed-loop data [112]. To mitigate potential issues, a comprehensive data analysis is conducted to ensure that the quality of available data aligns with the assumptions required for parameter identification.

The goal of this paper is to employ offline system identification techniques on a specific eVTOL aircraft when operating in a fixed-wing configuration. The paper is organized as follows: Section II describes the aircraft and general aerodynamics. Section III details the model responsible for outputting parameters related to the propulsion contribution. Section IV presents the techniques used to determine the structure of the aero-propulsive model and its parameter estimation. In Section V, the analysis and processing of flight data are explained to ensure data quality meets the assumptions of the presented techniques. Finally, Section VI validates and discusses the obtained propulsive system modeling and aero-propulsive parameters.

The methods applied in this work were adapted from the System IDentification Programs for AirCraft (SIDPAC) software toolbox [119] and supplemental software materials from reference [113][1].

3.2 Aircraft Properties and Vehicle Dynamics

The modeling of flight dynamics involves computing aerodynamic forces (X,Y,Z) and moments (L,M,N), which are influenced by the vehicle's state and control inputs. The equations of motion, derived from Newton's second law, govern the translational and rotational dynamics. The specific equations utilized in this project are as follows:

$$\begin{aligned} \dot{u} &= rv - qw + X/m - g\sin\theta \\ \dot{v} &= pw - ru + Y/m + g\cos\theta\sin\phi \\ \dot{w} &= qu - pv + Z/m + g\cos\theta\cos\phi \end{aligned} \tag{3.1}$$

$$\begin{aligned} \dot{p} - \frac{I_{xz}}{I_x}\dot{r} &= \frac{L}{I_x} - \frac{(I_z - I_y)}{I_x}qr + \frac{I_{xz}}{I_x}qp \\ \dot{q} &= \frac{M}{I_y} - \frac{(I_x - I_z)}{I_y}pr - \frac{I_{xz}}{I_y}\left(p^2 - r^2\right) - \frac{I_p}{I_y}\Omega_p r \\ \dot{r} - \frac{I_{xz}}{I_z}\dot{p} &= \frac{N}{I_z} - \frac{(I_y - I_x)}{I_z}pq - \frac{I_{xz}}{I_z}qr + \frac{I_p}{I_z}\Omega_p q \end{aligned} \tag{3.2}$$

$$\begin{aligned} \dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ \dot{\theta} &= q\cos\phi - r\sin\phi \\ \dot{\psi} &= q\sin\phi\sec\theta + r\cos\phi\sec\theta \end{aligned} \tag{3.3}$$

The equations related to the moments (Eq. 3.2) incorporate the gyroscopic contribution of the rotor. Ω_p is the rotational rate of the propulsive system and I_p is the moment of inertia of the motor and propeller combined. Notably, the transient response's roll contribution is neglected due to the constant throttle command in selected flight segments.

Force coefficients (C_x, C_y, C_z) are derived from linear accelerations at the center of gravity (Eq. 3.4), with the non-dimensional form eliminating dynamic and pressure dependency.

[1]Information available online at `https://arc.aiaa.org/doi/suppl/10.2514/4.102790` [retrieved February 2024]

$$C_x = \frac{ma_x}{\bar{q}S}, \quad C_y = \frac{ma_y}{\bar{q}S}, \quad C_z = \frac{ma_z}{\bar{q}S} \tag{3.4}$$

To correct measured accelerations from the accelerometers displaced from the center of gravity, adjustments are made as follows:

$$\begin{aligned} a_x^{CG} &= a_x^{IMU} + \left(q^2 + r^2\right)\Delta x - (pq - \dot{r})\Delta y - (pr + \dot{q})\Delta z \\ a_y^{CG} &= a_y^{IMU} - (pq + \dot{r})\Delta x + \left(p^2 + r^2\right)\Delta y - (qr - \dot{p})\Delta z \\ a_z^{CG} &= a_z^{IMU} - (pr - \dot{q})\Delta x - (qr + \dot{p})\Delta y + \left(p^2 + q^2\right)\Delta z \end{aligned} \tag{3.5}$$

Moment coefficients (C_l, C_m, C_n) are calculated from Eq. 3.2 in their non-dimensional form:

$$\begin{aligned} C_l &= \frac{1}{\bar{q}Sb}\left[I_x\dot{p} - I_{xz}\dot{r} + (I_z - I_y)\,qr - I_{xz}pq\right] \\ C_m &= \frac{1}{\bar{q}S\bar{c}}\left[I_y\dot{q} + (I_x - I_z)\,pr + I_{xz}\left(p^2 - r^2\right) + I_p\Omega_p r\right] \\ C_n &= \frac{1}{\bar{q}Sb}\left[I_z\dot{r} - I_{xz}\dot{p} + (I_y - I_x)\,pq + I_{xz}qr - I_p\Omega_p q\right] \end{aligned} \tag{3.6}$$

Here, $\bar{q} = \frac{1}{2}\rho V^2$ represents dynamic pressure, and Table 3.1 details the inertial and geometric properties of the Eusphyra aircraft. The inertial properties of the aircraft were determined through a combination of experimental and computational methods. The principal moments of inertia (I_x, I_y, and I_z) were measured using a bifilar pendulum test. In this test, the aircraft is suspended by two parallel strings, and its oscillatory motion is analyzed to extract the moments of inertia about its principal axes. The off-diagonal terms were derived from a detailed CAD model of the vehicle, where the masses and positions of each component were measured, and the inertia values were computed using the CAD software's built-in application of the parallel axis theorem.

Table 3.1: Inertial and geometric properties of the Eusphyra aircraft.

Property	Value	Units
m	8.152	kg
I_x	1.159	$\text{kg} \cdot \text{m}^2$
I_y	1.230	$\text{kg} \cdot \text{m}^2$
I_z	2.386	$\text{kg} \cdot \text{m}^2$
I_{xz}	0.000	$\text{kg} \cdot \text{m}^2$
$\bar{c}$	0.208	m
b	2.400	m
S	0.499	m^2

The Eusphyra aircraft, controlled by a PixHawk autopilot, utilizes avionic instrumentation, including an Attitude and Heading Reference System (AHRS), GPS, LIDAR altimeter, and an airspeed sensor. The PX4 software uses an Extended Kalman Filter (EKF) process to estimate the vehicle state variables, including attitude angles, Earth-relative linear velocity, inertial position and wind estimates. The logging sampling rate for raw acceleration, angular rates, attitude angles and commanded controls is 200 Hz. The inertial velocities estimated by the EKF have a sampling rate of 10 Hz. The remaining states, the wind velocities and battery data are logged at 5 Hz to 1 Hz.

The actuated surfaces of the vehicle are a canard, two ailerons and a rudder. The control signals, generated by the control laws, are non-dimensional (from -1 to 1). The surface actuation process, from the command output to the surface deflection, is modeled in the FDM as a first-order system. This includes the mixing matrix that converts the control output into a PWM signal, the resolution of the protocol and servo-actuator, the linkage geometry, the hysteresis of the system, the actuation speed, the deflection limits and the response delay. The actuator model parameters were derived from bench tests, where the servo-motor and actuator surfaces were tested under varying load conditions (e.g., no load, cruise load, VNE load, and maximum actuator load). These tests involved sending command signals to the actuator and measuring the output shaft position with an external angular encoder synchronized with the command signal. This setup allowed for precise measurements of delay, gain, hysteresis, and the transfer function representing the actuator's dynamic behavior. To simplify the simulation model, we used the cruise load transfer function

for all simulations. Cruise load conditions were replicated on the test bench using springs to simulate the in-flight loads. The time constant corresponds to the coefficient in the transfer function. This, along with the response delay, quantization, rate limiter, hysteresis, gain, surface limits, and linkage ratio, is used to generate the final actuator model in the simulation. The parameters of the servo-motor actuation and the actuator surfaces were determined from bench tests and are summed in Table 3.2 and Table 3.3, respectively.

Table 3.2: Servo-motor parameters.

Model	Gain [°]	Time constant [s]	Response delay [s]	Rate Limit [°/s]
MKS HBL 6625	51	0.0132	0.0132	530

Table 3.3: Actuator surface configurations.

Surface	Linkage ratio [°]	Upper limit [°]	Lower limit [°]
Aileron	20.3	20.5	-20
Canard	5.85	5.8	-5.9
Rudder	-46.3	44.7	-45

Due to the noise and unreliability of the airspeed sensor [120], speed V and the flow angles, α and β, are calculated from the body velocities (u, v, w), with the following relations:

$$V = \sqrt{u^2 + v^2 + w^2} \qquad \alpha = \tan^{-1}\left(\frac{w}{u}\right) \qquad \beta = \sin\left(\frac{v}{V}\right) \tag{3.7}$$

The body velocities are calculated from the rotated inertial velocities (V_N, V_E, V_D) by removing the wind components (W_N, W_E, W_D), see Eq. 3.8.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\phi\sin\psi & \sin\theta\cos\phi\sin\psi - \sin\phi\cos\psi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} V_N - W_N \\ V_E - W_E \\ V_D - W_D \end{bmatrix} \tag{3.8}$$

After analyzing the power spectrum densities, the frequency interval of [0 5Hz] was identified to contain all the rigid body dynamics. The angular accelerations were then derived from the angular rates after being properly smoothed [121]. The filter

used to accomplish this is the optimal Fourier smoother filter available in the SIDPAC toolbox [23]. This filter is described in detail in reference [112].

The key parameters that characterize the FDM are the aerodynamic derivatives from equations 3.4 and 3.6. These are functions of the aircraft states and the parameters related to the propulsive system.

3.3 Modeling of the Propulsive System

The methodology for obtaining parameters related to the contribution of the propulsive system is outlined in this section. The propeller dynamics and their related variables used in the System Identification of the aero-propulsive model are detailed in reference [116] and briefly summarized here.

3.3.1 Propeller Dynamics

The propeller's axis of rotation aligns with the aircraft's longitudinal axis. According to the model proposed in reference [116], the primary variable describing the contribution of the thrust is the inverse advance ratio, denoted as:

$$\mathcal{J} = \frac{2\pi\Omega_p D}{V} \tag{3.9}$$

The value of the inverse advance ratio is centered around $\mathcal{J}_o = 1.12$:

$$\mathcal{J}_c = \mathcal{J} - \mathcal{J}_o \tag{3.10}$$

For the Eusphyra aircraft, $\mathcal{J}_o$ is determined from the average of several longitudinal steady-state flights.

Propeller forces and moments depend on the angle between the freestream airflow and the propeller rotation axis, known as the propeller incidence angle, i_p, calculated as:

$$i_p = \cos^{-1}\left(\frac{u_p}{\sqrt{u_p^2 + v_p^2 + w_p^2}}\right) \tag{3.11}$$

The projections of the airflow incidence on the propeller disk, represented by i_z and i_y, influence their corresponding axes. This projection is determined by the angle

ε_p:

$$\varepsilon_p = \tan^{-1}\left(\frac{v_p}{w_p}\right) \tag{3.12}$$

The projections of the propeller incidence angle characterize the off-axis force and moment component due to incidence relative to the airflow. i_z affects C_Z and C_n and i_y influences C_Y and C_m. These variables are given by:

$$i_z = i_p \cos\varepsilon_p \qquad i_y = i_p \sin\varepsilon_p \tag{3.13}$$

Local velocities (u_p, v_p, w_p) are derived from the aircraft's body velocities using the relation [122]:

$$\begin{aligned} u_p &= u + q\Delta z_p - r\Delta y_p \\ v_p &= v + r\Delta x_p - p\Delta z_p \\ w_p &= w + p\Delta y_p - q\Delta x_p \end{aligned} \tag{3.14}$$

where $\Delta x_p = -0.6458$ m and $\Delta y_p = \Delta z_p = 0$ denote the displacements between the center of gravity and the propeller's disk. Δx_p is negative due to the pusher configuration with the propeller located behind the center of gravity.

The rotation rate (Ω_p) is a crucial measurement for the characterization of the propeller dynamics. In this case, Ω_p is estimated from the model comprising the ESC, brushless motor, and blade propeller.

3.3.2 Electrical System Model

The motor is modeled as a combination of an ideal motor, armature resistance (R_m) and no-load current (I_0). The motor velocity constant, K_v, describes the relation between the armature voltage and the output rotational speed of the ideal motor. The inductance contribution is neglected since the motor is considered in a steady-state condition. Although this value is not usually provided by the manufacturer it could be measured experimentally if necessary. Figure 3.2 illustrates the propulsive system.

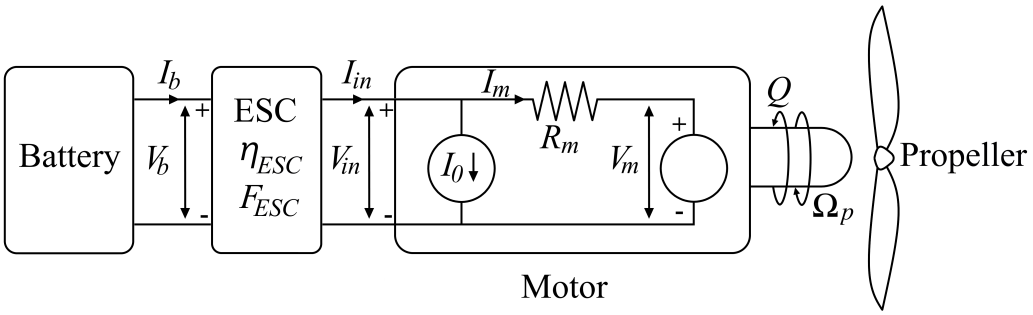


Figure 3.2: Schematic of the propulsive system, including the representation of the equivalent circuit for a DC motor.

This model takes as inputs the throttle command and battery voltage. The current is calculated within the model itself. The motor dynamics are described by the equation:

$$\dot{\Omega}_p = \frac{1}{I_p}\left(\frac{1}{K_v}I_m - Q - B_m\Omega_p\right) \tag{3.15}$$

The rotational acceleration of the motor, $\dot{\Omega}_p$, is integrated to obtain the rotation rate in rad/s. Viscous friction is neglected ($B_m = 0$), and the current I_m is calculated as:

$$I_m = \frac{V_{in} - V_m}{R_m} \tag{3.16}$$

V_{in} is the voltage provided by the ESC, equal to the battery voltage V_b for an ideal ESC. However, as discussed in section D of this chapter, the efficiency losses of the ESC cannot be neglected. V_m is the voltage in the motor, directly related to the rotation rate:

$$V_m = \frac{1}{K_v}\Omega_p \tag{3.17}$$

To complete the model, the remaining variable to calculate is the torque Q.

3.3.3 Propeller Torque Estimation

A propeller is conventionally characterized by thrust, power, and torque coefficients. In the proposed method, as thrust is integrated into the aerodynamic model, the propeller model focuses solely on torque, defined by the coefficient C_Q:

$$C_Q = \frac{Q}{4\pi^2 \rho \Omega_p^2 D^5} \tag{3.18}$$

Here, D is the diameter of the propeller disk. Given that Q is the output of the motor model, and considering the steady-state case ($\dot{\Omega}_p = 0$), the torque is obtained from Eq. 3.15 as:

$$Q = \frac{1}{K_v} I_m \tag{3.19}$$

C_Q is a function of the advance ratio, J, and the propeller geometry. While small-dimension propellers ($D < 9$in) typically require consideration of Reynolds number and viscosity effects [123], the Eusphyra's propeller, despite not being small, exhibits a calculated Reynolds number below 100,000 for the considered flight conditions. The final torque model is expressed as:

$$C_Q = C_{Q_0} + C_{Q_J} J + C_{Q_{Re}} \Omega_p \tag{3.20}$$

The Reynolds effect can be characterized by the variable Ω_p as this variable is directly proportional to the Reynolds number [124].

3.3.4 Static Thrust Test

The validation of the motor model is conducted through static thrust tests, measuring propulsor rotational speed, thrust, and torque generated by the propulsive system. These tests are carried out in a steady-state condition at various throttle command levels (in increments of 10%).

The bench tests showed that the actual RPM values were lower than expected for the given K_v value. References [125, 126] employed similar motor models and confirmed that the motor constant given by the manufacturer is accurate. Assuming the correctness of the tabulated K_v value, system losses are estimated. In reference [127], these losses are modeled by determining the ESC voltage ratio function, $F_{ESC\%}$, and the efficiency of the ESC, η_{ESC}. The ESC voltage ratio function depends on the commanded throttle:

$$F_{ESC\%}(\eta_{th}) = \frac{V_{in}}{V_b} \tag{3.21}$$

The ESC efficiency, η_{ESC}, is calculated for different η_{th} and V_b using:

$$\eta_{ESC}\left(V_b, \eta_{th}\right) = \frac{V_{in} I_{in}}{V_b I_b} = F_{ESC\%}\left(\eta_{th}\right) \frac{\left(QK_v + I_0\right)}{I_b} \tag{3.22}$$

As demonstrated in [126], this method might inaccurately model the current across the entire throttle range. Therefore, parameters are determined only for the throttle range used during system identification maneuvers: [70 100] %.

After establishing the model defined by equation 3.15, the output Ω_p is compared with the measured Ω.

With the ESC and electrical motor components modeled, the Ω_p is estimated from the flight data by incorporating the model for the torque of the propeller blade as described in the previous subsection. The propeller model parameters are estimated with the same methods employed for aerodynamic derivatives in the following section.

3.4 Determination and Estimation of Model Parameters

An overview of model structure determination, parameter estimation and model validation methods is provided in this section. Detailed descriptions of these techniques and other system identification methods are available in references [112, 113].

3.4.1 Parameter Estimation

Output-Error Method

The Output-Error Method (OEM) is the most common method applied to time-domain system identification for aircraft cases [113]. The model parameters are adjusted to minimize the error between the model and predicted outputs through numerical integration of the dynamic model. The model, described as a deterministic and time-invariant system, is represented in state space as follows [112, 113]:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f[\boldsymbol{x}(t), \boldsymbol{u}(t), \Theta], \quad x\left(t_0\right) = x_0 \\ \boldsymbol{y}(t) &= g[\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\Theta}] \end{aligned} \tag{3.23}$$

Here, $\boldsymbol{x}$ is the $n_x \times 1$ state variables vector, $\boldsymbol{u}$ is the $n_u \times 1$ control input vector,

$\boldsymbol{y}$ is the $n_y \times 1$ system output vector, Θ is the $n_p \times 1$ system parameter vector, and f and g are nonlinear functions representing the model dynamics and the observer transformation, respectively.

The parameter vector Θ is estimated from N discrete measurements of the model outputs $\boldsymbol{y}$. The measurement equation is given as:

$$\boldsymbol{z}(k) = \boldsymbol{y}(k) + \boldsymbol{v}(k), \quad k = 1, 2, \ldots, N \tag{3.24}$$

With $\boldsymbol{z}$ the $n_y \times 1$ measurement vectors and the $\boldsymbol{v}$ the measurement noise vector, assumed to be characterized by a sequence of n_y independent Gaussian random variables with zero mean and measurement noise covariance $\boldsymbol{R}$.

The measurement noise comes from N samples of m random variables, where m is the number of measured outputs.

The maximum likelihood estimation consists of searching the vector $\boldsymbol{\Theta}$ that maximizes the probability distribution of the measurements. The likelihood function is maximized by minimizing a cost function given by [113]:

$$J(\boldsymbol{\Theta}) = \det(\boldsymbol{R}) \tag{3.25}$$

Equation 3.25 is derived under the assumption that the measurement noise covariance matrix is unknown *a priori*. The estimation of $\boldsymbol{R}$ is defined as:

$$\boldsymbol{R} = \frac{1}{N} \sum_{k=1}^{N} [\boldsymbol{z}(k) - \boldsymbol{y}(k)] [\boldsymbol{z}(k) - \boldsymbol{y}(k)]^T \tag{3.26}$$

The minimization of the cost function is achieved using the Gauss-Newton method. At each iteration, the parameters are updated by:

$$\boldsymbol{\Theta}_{i+1} = \boldsymbol{\Theta}_i + \boldsymbol{\Delta\Theta} \tag{3.27}$$

$$\boldsymbol{\Delta\Theta} = -\boldsymbol{\mathcal{F}}^{-1} \boldsymbol{\mathcal{G}} \tag{3.28}$$

The information matrix $\boldsymbol{\mathcal{F}}$ and the gradient vector $\boldsymbol{\mathcal{G}}$ are defined as:

$$\boldsymbol{\mathcal{F}} = \sum_{k=1}^{N} \left[\frac{\partial \boldsymbol{y}(k)}{\partial \boldsymbol{\Theta}} \right]^T R^{-1} \left[\frac{\partial \boldsymbol{y}(k)}{\partial \boldsymbol{\Theta}} \right] \tag{3.29}$$

$$\boldsymbol{\mathcal{G}} = \sum_{k=1}^{N} \left[\frac{\partial \boldsymbol{y}\,(k)}{\partial \boldsymbol{\Theta}} \right]^{T} R^{-1} \left[\boldsymbol{z}\,(k) - \boldsymbol{y}\,(k) \right] \tag{3.30}$$

Equations 3.29 and 3.30 utilize first-order sensitivities calculated with a forward difference approximation [113].

The statistical accuracy of the parameters is evaluated based on the Cramér-Rao lower bound [112, 113]. The Cramér-Rao inequality for the parameter error covariance matrix is obtained with the inverse of the information matrix:

$$\mathrm{Cov}\,(\boldsymbol{\Theta}) \geq \boldsymbol{\mathcal{F}}^{-1} \tag{3.31}$$

Factors that can prevent the complete modeling of all dynamics present in a flight test can arise from the limitations of the experiment. Insufficient excitation of all aircraft modes can produce low signal-to-noise ratio on some aircraft dynamics. The non-modeled dynamics show up in the residuals as sequences that resemble colored noise [112, 113]. A method that corrects the accuracy of the parameter error bounds for a time-domain analysis is described in reference [128]. The white residual assumption is replaced and the correction is obtained by including an estimate of the residual coloring in the expression for the Cramér-Rao bound estimate. The residual coloring is quantified by the correlation between the residuals with their adjacent neighbors [112]. The colored noise correction is applied at the end of the Gauss-Newton optimization procedure.

The N discrete measurements result from concatenating n_E time segments corresponding to individual maneuvers. The colored noise correction from reference [128] is adapted so that the residual coloring is calculated for each time segment, modifying the inequality of equation 3.31 as follows:

$$\mathrm{Cov}(\boldsymbol{\Theta}) \geq \boldsymbol{\mathcal{F}}^{-1} \left[\sum_{i=1}^{n_E} \sum_{j=1}^{N_i} \left[\frac{\partial \boldsymbol{y}\,(j)}{\partial \Theta} \right] \boldsymbol{R}^{-1} \sum_{k=1}^{N_i} \boldsymbol{\mathcal{R}}_{\boldsymbol{vv}}(j-k) \boldsymbol{R}^{-1} \left[\frac{\partial \boldsymbol{y}\,(k)}{\partial \Theta} \right] \right] \boldsymbol{\mathcal{F}}^{-1} \tag{3.32}$$

where $\boldsymbol{\mathcal{R}}_{vv}$ is the autocorrelation matrix for the output residual vector, calculated as:

$$\boldsymbol{\mathcal{R}}_{\boldsymbol{vv}}\,(l) = \frac{1}{N_i} \sum_{i=1}^{N_i - l} \boldsymbol{v}_i \boldsymbol{v}_{i+l} = \boldsymbol{\mathcal{R}}_{\boldsymbol{vv}}\,(-l) \quad l = 0, 1, 2, \ldots, r \tag{3.33}$$

Here, r represents the maximum time difference used in the computation of the

residual autocorrelation and is set to be equal to N_i.

The parameter error is obtained from the standard deviation $s(\boldsymbol{\Theta})$:

$$s(\boldsymbol{\Theta}) = \sqrt{\text{diag}\left[\text{Cov}\left(\boldsymbol{\Theta}\right)\right]} \tag{3.34}$$

The standard deviation is represented as a relative quantity, denoted as the error of the estimates:

$$\text{Error}\,[\%] = 100 \times \frac{\boldsymbol{s}(\boldsymbol{\Theta})}{\boldsymbol{\Theta}} \tag{3.35}$$

To ensure convergence of the Gauss-Newton method to the global maximum of the likelihood function, the initial parameter vector estimate must be near the value of $\boldsymbol{\Theta}$ corresponding to the minimum of the cost function $J(\boldsymbol{\Theta})$ [112, 113]. The initial parameter vector is obtained using the Equation-Error Method.

Equation-Error Method

The Equation-Error Method (EEM) is a simple and computationally efficient technique chosen for obtaining an initial guess of the parameter vector. Parameters are estimated using ordinary least-squares regression. The unknown model is given by $\boldsymbol{y} = \boldsymbol{\Theta}\boldsymbol{X}$, where $\boldsymbol{X}$ is the regressor matrix and each column is the x_i regressor vector. The equation that relates the output of the model $\boldsymbol{y}$ and the response variable $\boldsymbol{z}$ obtained from the measurements is equivalent to the output equation (Eq. 3.24). For this method, z is the force and moment coefficients calculated with equations 3.4 and 3.6.

For least-squares parameter estimation, the optimal estimate of the unknown parameter vector $\boldsymbol{\Theta}$ is determined by minimizing the cost function:

$$J\left(\boldsymbol{\Theta}\right) = \frac{1}{2}\left[\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\Theta}\right]\left[\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\Theta}\right]^T \tag{3.36}$$

The minimum of the cost function provides an optimal estimate of the unknown parameters, given by:

$$\boldsymbol{\Theta} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{z} \tag{3.37}$$

The statistical accuracy of the least-squares estimates is measured by the standard deviation $s(\boldsymbol{\Theta})$ as expressed in equation 3.34. The parameter error covariance matrix

is calculated with a colored noise correction, resulting in the following expression:

$$\mathrm{Cov}(\boldsymbol{\Theta}) = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\left[\sum_{i=1}^{N}\boldsymbol{x}_i\sum_{j=1}^{N}\boldsymbol{\mathcal{R}}_{\boldsymbol{vv}}(i-k)\boldsymbol{x}_j^T\right]\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \tag{3.38}$$

The autocorrelation matrix $\boldsymbol{\mathcal{R}}_{\boldsymbol{vv}}$ is estimated following Equation 3.33. Equation 3.38 presents the general form of the parameter error covariance matrix, corrected for colored noise. Understanding the importance of parameter error results from the EEM is crucial for determining the feasibility of progressing with the OEM optimization. Consequently, extra attention is given to the calculation of the error. Reference [121] provides a comprehensive equation applicable when dealing with multiple concatenated maneuvers. By defining $\boldsymbol{\Sigma}_i$ as the covariance matrix of the i-th maneuver, the total covariance is calculated as:

$$\mathrm{Cov}\left(\boldsymbol{\Theta}\right) = \left[\sum_{i=1}^{n_E}\boldsymbol{\Sigma}_i^{-1}\right]^{-1} \tag{3.39}$$

Calculation of the parameter error covariance matrix individually accounts for differences in the noise sequence $\boldsymbol{v}$ during each maneuver. This approach is crucial to avoid assuming the noise $\boldsymbol{v}$ has the same characteristics for concatenated maneuvers, which could lead to inaccuracies in estimated parameter uncertainties [121]. The EEM provides a good parameter estimation solution if the correct regressors are considered. To obtain a good model and take advantage of the least-squares regression solution, a dedicated model structure determination technique is used.

3.4.2 Model Determination

The model structure determined analytically proved to be inconsistent with flight data. Additionally, incorporating propulsive terms in the aerodynamic model necessitates a new model structure. Candidate regressors include linear terms corresponding to state variables and propulsive system variables, while regressors related to the coupling of the propulsive system and airframe surfaces, along with other non-linearities, are identified through multivariate orthogonal function (MOF) modeling.

The MOF modeling, explained in references [112, 129], has proven successful in fixed-wing aircraft system identification cases using flight data [116, 130]. The orthogonalization of a predefined set of candidate regressors is the starting point for

this approach. Regressors are ranked based on their ability to improve the model, and the final set is constituted by polynomial terms defining the selected orthogonal functions. The selected functions are the ones that minimize a specific metric that accounts for the mean squared fit error and overfitting [112], obtained through automated software [23]. The algorithm was set to generate the orthogonal functions with the state variables and combinations up to the third order.

The resulting polynomial regressors form an extensive list, sometimes lacking expected linear terms. To refine this list to include only significant regressors, both state variables and polynomials from the MOF modeling are used in a stepwise regression procedure. The stepwise regression is detailed in references [112, 113].

In the stepwise regression, regressors are manually selected to incorporate the model based on several statistical metrics:

$$\boldsymbol{r}_i = \frac{\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i\right)^T \left(\boldsymbol{z} - \bar{\boldsymbol{z}}\right)}{\sqrt{\left[\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i\right)^T \left(\boldsymbol{x}_i - \bar{\boldsymbol{z}}_i\right)\right] \left[(\boldsymbol{z} - \bar{\boldsymbol{z}})^T (\boldsymbol{z} - \bar{\boldsymbol{z}})\right]}} \tag{3.40}$$

$$F_{0_i} = \frac{\boldsymbol{\Theta}_i^2}{s\left(\boldsymbol{\Theta}_i\right)^2} \tag{3.41}$$

$$R^2 = \frac{\hat{\boldsymbol{y}}^T \boldsymbol{z} - N\bar{\boldsymbol{z}}^2}{\boldsymbol{z}^T \boldsymbol{z} - N\bar{\boldsymbol{z}}^2} \tag{3.42}$$

Equation 3.40 expresses the partial correlation coefficient, r_i. The partial F statistic (Eq. 3.41), F_0, indicates the significance of the added regressor. The coefficient of determination R^2, (Eq. 3.42) measures the goodness of the fit between the measurements and modeled data. The regressor with the highest r_i is added first. To verify that the term added contributes significantly to the model, F_0 should be higher than the given cut-off value. The addition of a new term is stopped if the R^2 does not increase more than 0.5%. If a nonlinear term would have a bigger r_i than the linear terms it constitutes, then the addition of both individually should be analyzed. If the difference in the goodness of fit between a more complex model and a simpler one is not significant, then the simplest model should be chosen. Knowing the final model, the EEM is applied to estimate the respective coefficients.

3.5 Flight Data Analysis

The flight data is analyzed to obtain a data set that meets the assumptions that enable the parameter estimation techniques used. The main concerns of this analysis are the collinearity assessment and kinematic compatibility of the flight data. The analysis techniques are described in this section.

3.5.1 Data Collinearity Diagnosis

Collinearity in the regressors can negatively impact parameter estimation procedures [112, 113]. The system identification maneuvers were executed in stability mode with active feedback control. For instance, in the presence of roll and yaw motion coupling, when a manual pilot initiates a roll excitation, the active feedback control adjusts the rudder deflection to correct the yaw moment response. In this scenario, at least three variables (δ_a, δ_r, r) may exhibit some level of linear dependency.

To assess the severity of linear dependencies, an eigensystem analysis of the standardized regressor matrix is conducted. Variance decomposition proportions are then calculated to identify the specific regressors involved in the dependency [112].

The regressor vectors are standardized using unit length scaling, defined as:

$$\boldsymbol{x}_i^* = \frac{\boldsymbol{x}_i}{\sqrt{\sum_{k=1}^{N} \boldsymbol{x}_i^2(k)}} \tag{3.43}$$

Reference [131] discusses whether the data should be centered; however, as this paper utilizes the entire measured dataset, the regressor vectors are not centered to avoid masking dependencies related to the constant term [131, 132].

Utilizing Singular Value Decomposition (SVD), $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{T}^2$, the condition index (k_j) is obtained as:

$$k_j = \frac{\mu_{max}}{\mu_j} \quad j = 0, 1, 2, \ldots, n \tag{3.44}$$

Here, μ_j corresponds to the diagonal elements of matrix $\boldsymbol{D}$ with n elements, each associated with an eigenvalue. The largest condition index is called the condition number. A condition number between 10 and 30 indicates collinearity, and when exceeding 30, collinearity is considered strong [133].

After detecting collinearity, variance proportion decomposition helps identify the

involved regressors. The variance for each estimated parameter can be decomposed into a sum of components, each corresponding to one of the n eigenvalues of the regressor matrix [112].

Using SVD of the regressor matrix, the i, j variance proportion π_{ij} is defined as the proportion of the variance of the i-th estimated parameter, associated with the j-th eigenvalue. Given,

$$\phi_{ij} \equiv \frac{t_{ij}^2}{\mu_j^2} \quad \text{and} \quad \phi_i \equiv \sum_{j=1}^{n} \frac{t_{ij}^2}{\mu_j^2} = \sum_{j=1}^{n} \phi_{ij} \tag{3.45}$$

t_{ij} is the i-th element of the j-th eigenvector associated with μ_j, and the j-th eigenvector is the j-th column of $\boldsymbol{T}$ [131]. The variance decomposition proportions are obtained as follows:

$$\pi_{ij} = \frac{\phi_{ij}}{\phi_j} \tag{3.46}$$

If two or more variance decomposition proportions corresponding to condition indices higher than 10 to 30 exceed 80%, collinearity is determined between the respective explanatory variables [133].

3.5.2 Flight Path Reconstruction

Flight Path Reconstruction (FPR) validates data accuracy in measured aircraft responses. Systematic errors, such as constant bias in the Inertial Measurement Unit (IMU), can cause mismatches between measured and reconstructed responses [112]. Measured responses take the form:

$$\boldsymbol{y}_m(t) = \boldsymbol{y}(t) + b_y \tag{3.47}$$

Systematic errors, dependent on factors like temperature and vibrations, are time-variant. Bias terms are independently determined for each time segment to enhance reconstructed state accuracy.

Incorrect initial conditions can lead to integrated errors over time. Following the suggestion in reference [113], initial conditions regarding body velocities (u_0, v_0, w_0) and attitude angles (ϕ_0, θ_0, ψ_0), and altitude above ground (h_0) are optimized. The Output-Error Method, a suitable parameter estimation method, is employed to obtain systematic sensor errors and optimal initial conditions.

The state equations for the FPR procedure are as follows:

$$
\begin{aligned}
\dot{u} &= -\left(q_m - b_q\right) w + \left(r_m - b_r\right) v - g \sin\theta + \left(a_{xm} - b_{ax}\right), && u\left(t_0\right) = u_0 \\
\dot{v} &= -\left(r_m - b_r\right) u + \left(p_m - b_p\right) w + g \cos\theta \sin\phi + \left(a_{ym} - b_{ay}\right), && v\left(t_0\right) = v_0 \\
\dot{w} &= -\left(p_m - b_p\right) v + \left(q_m - b_q\right) u + g \cos\theta \cos\phi + \left(a_{zm} - b_{az}\right), && w\left(t_0\right) = w_0 \\
\dot{\phi} &= \left(p_m - b_p\right) + \left(q_m - b_q\right) \sin\phi \tan\theta + \left(r_m - b_r\right) \cos\phi \tan\theta, && \phi\left(t_0\right) = \phi_0 \\
\dot{\theta} &= \left(q_m - b_q\right) \cos\phi - \left(r_m - b_r\right) \sin\phi, && \theta\left(t_0\right) = \theta_0 \\
\dot{\psi} &= \left(q_m - b_q\right) \sin\phi \sec\theta + \left(r_m - b_r\right) \cos\phi \sec\theta, && \psi\left(t_0\right) = \psi_0 \\
\dot{h} &= u \sin\theta - v \cos\theta \sin\phi - w \cos\theta \cos\phi, && h\left(t_0\right) = h_0
\end{aligned}
\tag{3.48}
$$

Equation 3.48 corresponds to the force and kinematic equations (Eq. 3.1), modified to have as input the measured linear accelerations and angular rates. Table 3.4 shows the input, state variables, measured/model output vectors, and unknown estimated variables for the flight path reconstruction procedure.

Table 3.4: Input, State, Output Vectors and Unknown Variables in the Flight Path Reconstruction for the Output Error Method

Symbol	Description	Variables
$\boldsymbol{u}$	Input Vector	$[a_x, a_y, a_z, p, q, r]$
$\boldsymbol{x}$	State Variable Vector	$[u, v, w, \phi, \theta, \psi]$
$\boldsymbol{y}$	Model Observation Vector	$[u_m, v_m, w_m, \phi_m, \theta_m, \psi_m]$
$\boldsymbol{\theta}$	Unknown Variables	$[u_0, v_0, w_0, \phi_0, \theta_0, \psi_0, b_{ax}, b_{ay}, b_{az}, b_q, b_q, b_r]$

The measured output data corresponds to the EKF data coming from the Pixhawk autopilot. The body velocities come from the inertial velocities which are sampled at 10 Hz. The bias estimation is conducted at 10 Hz, since that is the rate with available data. After the bias was estimated, the states are reconstructed using acceleration and angular rates data sampled at 200Hz.

The FPR reconstructed states serve as inputs for aero-propulsive parameter estimation. The reconstructed states, with bias correction, aim for a fit evaluated by the R^2 coefficient of at least 95%, given the expected accuracy of the Extended Kalman Filter (EKF) data.

3.5.3 Data Set

The data set is constructed from SID maneuvers, including commanded doublets and chirps. The maneuvers were performed at a target altitude of 120 m above mean sea level, and reference airspeed ranging from 23 to 27 m/s. During the flight tests the wind conditions were calm, 3kts variable. Collinearity diagnosis is performed separately for state variables describing longitudinal and lateral-directional modes. Flight path reconstruction filters out segments violating the kinematic laws and provides systematic error corrections.

Unlike commonly used perturbation data, the entire measured dataset is used, including information about trim conditions. This inclusion is advantageous as the dataset consists of segments from different flights with potentially divergent trim conditions. The disadvantage of using the entire measured data is that the estimator can have difficulty predicting the constant terms [113]. However, it is considered that the advantage of utilizing this type of data outweighs the estimation difficulties.

The final data set incorporates 26 segments, encompassing eleven pitch excitation maneuvers, thirteen lateral- directional maneuvers, and two pitch, roll, and yaw doublet sequences for validation.

3.6 Results and Discussion

3.6.1 Propulsive System Model

ESC Parameters

Static thrust tests were conducted for battery charge levels of 30%, 50%, and 100%, corresponding to average battery voltages of 22.2V, 23V, and 24.5V, respectively. The motor parameters, namely K_v, R_m, and I_0, were adopted from the manufacturer's specifications. The voltage ratio function ($F_{ESC\%}$) was computed and averaged for each throttle percentage. Figure 3.3 illustrates minimal divergence in $F_{ESC\%}$ for varying throttle percentages. Notably, $F_{ESC\%}$ was found to be relatively constant, regardless of battery voltage, aligning with findings in [127]. Theoretically, ESC efficiency (η_{ESC}) is a function of both throttle and battery voltage. However, Figure 3.4 demonstrates negligible variations in η_{ESC} across the considered throttle range and different battery voltages. Consequently, η_{ESC} is treated as a constant, with an average value of $\eta_{ESC} = 83.947\%$.

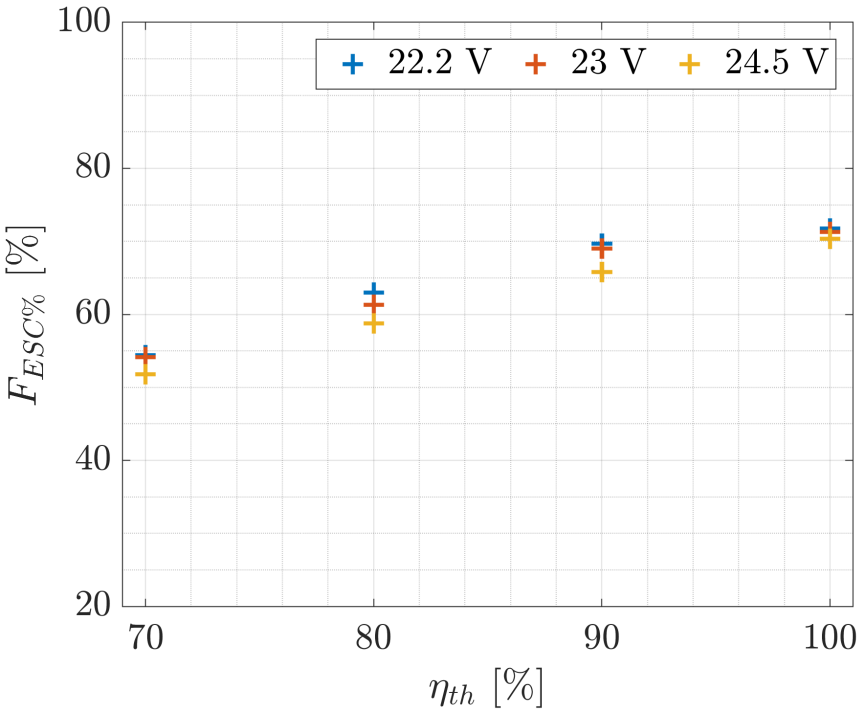


Figure 3.3: ESC voltage ratio function map.

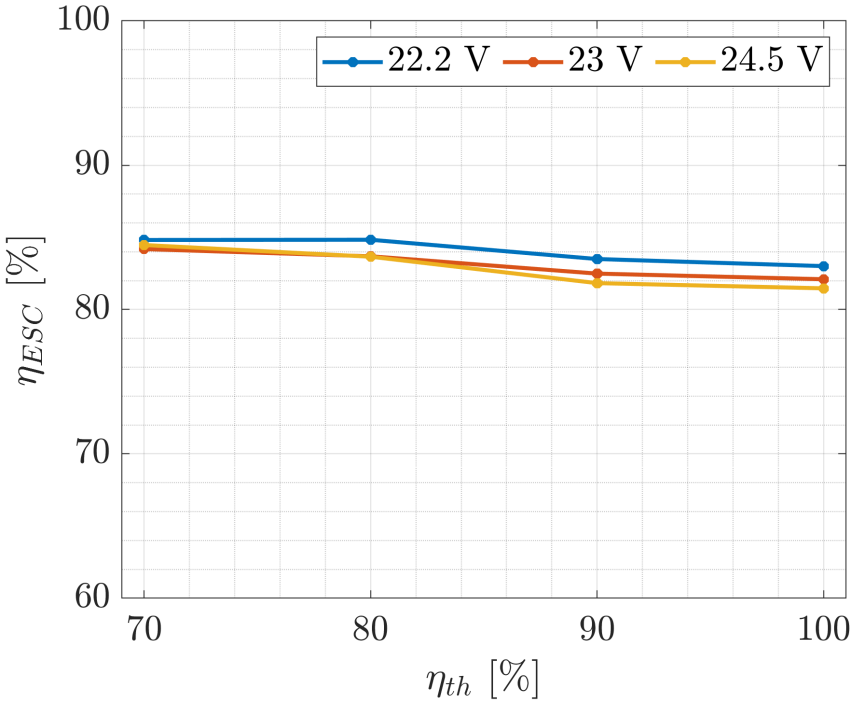


Figure 3.4: ESC efficiency map.

Motor Model Validation

Having estimated the ESC parameters, the motor model is subjected to validation. The propulsor rotation speed is directly measured in the test bench. The armature current is derived from the measured battery voltage, I_b , and the ESC mappings. From equations 3.19 and 3.22, the measured derived current is obtained as:

$$I_m = \frac{\eta_{ESC}}{F_{ESC\%}\left(\eta_{th}\right)} I_b - I_0 \tag{3.49}$$

The test bench measurements are compared against the simulated model results. The propulsor rotation speed, Ω_p, is obtained by integrating the motor's dynamic equation (Eq. 3.15). The model's armature current is obtained by combining Eq. 3.16, 3.17 and 3.21, with this variable being function of the rotational speed, input throttle and battery voltage:

$$I_m = \frac{1}{R_m}\left(V_b F_{ESC\%}\left(\eta_{th}\right) - \frac{\Omega_p}{K_V}\right) \tag{3.50}$$

The Root Mean Squared Error (RMSE) is employed to quantify the fit error, and it is normalized to express the relative error, defined as follows:

$$\text{RMSE} = \sqrt{\frac{(\boldsymbol{z} - \hat{\boldsymbol{y}})^T(\boldsymbol{z} - \hat{\boldsymbol{y}})}{N}} \quad \text{and} \quad \text{NRMSE} = 100 \times \frac{\text{RMSE}}{(z_{\max} - z_{\min})} \tag{3.51}$$

Figure 3.5 presents the time series of measured and predicted rotation rates in RPM, as well as armature current. The model exhibits an acceptable fit, simulating armature current with an NRMSE of 3.52% and rotation rate with an NRMSE of 5.39%. This level of error is deemed acceptable given the model's ability to predict an averaged Ω_p for a specific throttle level.

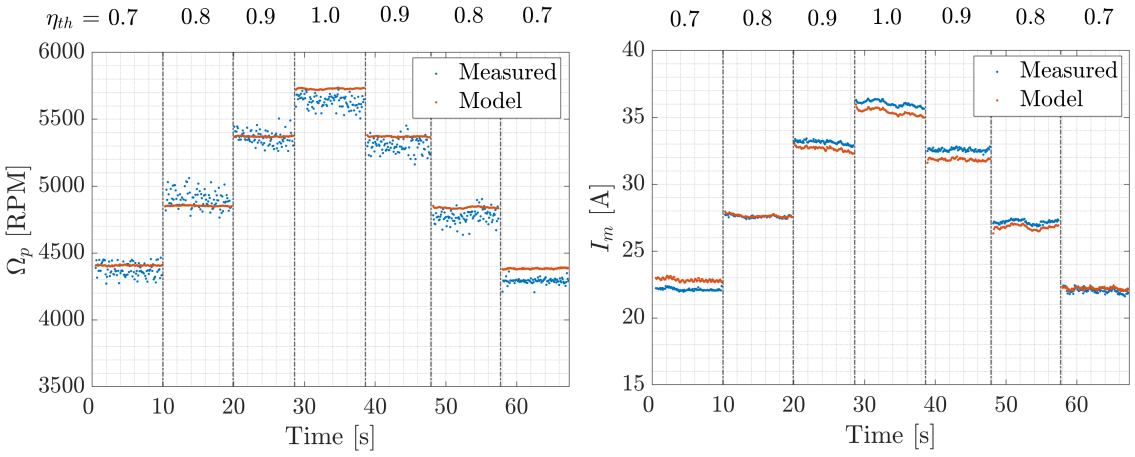


Figure 3.5: Model prediction of the rotation rate Ω_p (left) and the armature current I_m (right).

3.6.2 Torque Model

To determine the parameters of the torque coefficient, the equation-error method was employed on flight data segments where the throttle command remains approximately constant. Figure 3.6 illustrates the throttle command, rotation rate, airspeed, and calculated advance ratio for the dataset.

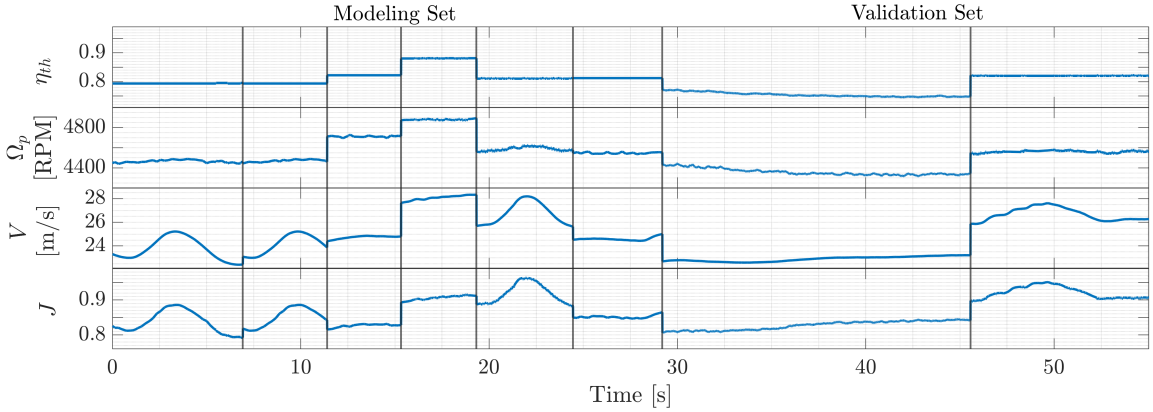


Figure 3.6: Throttle command, RPM, airspeed and advance ratio of the training and validation data sets.

The rotation rate of the motor is derived from the throttle, battery voltage, and current. Table 3.5 presents the coefficients obtained along with their respective errors for the modeling set.

Table 3.5: Parameters of the torque coefficient estimated using the Equation-Error Method.

Parameter	Value	Error %
C_{Q_0}	1.229e-02	0.1
C_{Q_J}	-5.72853e-03	9.7
$C_{Q_{Re}}$	-1.27613e-06	8.2

Figure 3.7 shows the time series of the torque coefficient. For the modeling set, the R^2 obtained for the C_Q model is 83.5%. The RMSE is 0.00016, equivalent to 1.3% of the average value of C_Q. In the validation set, the model fitting yields $R^2 = 81.6\%$ with an RMSE of 0.00021, indicating that the model is a reliable predictor. Notably, the flight data is noisy, but the model is capable of predicting an averaged trend.

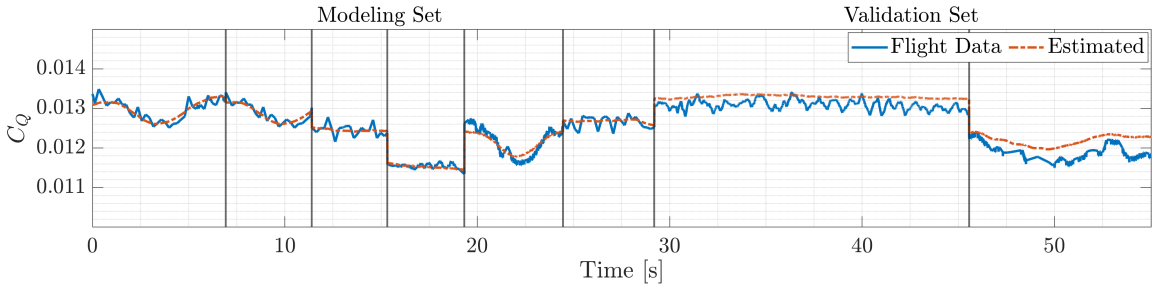


Figure 3.7: Comparison between modeling and validation data and model fit of the torque coefficient.

The evaluation of the model involves predicting the output Ω_p. The measurement of Ω_p used to compare against the model's output is obtained with Eq. 3.50, where the armature current is the battery data and throttle command derived from Eq. 3.49:

$$\Omega_p = K_v \left(V_b F_{ESC\%}\left(\eta_{th}\right) - \frac{\eta_{ESC}}{F_{ESC\%}\left(\eta_{th}\right)} I_b + I_0 \right) \tag{3.52}$$

Figure 3.8 illustrates the plots of Ω_p calculated from flight data and estimated using the Equation-Error Method for the entire dataset.

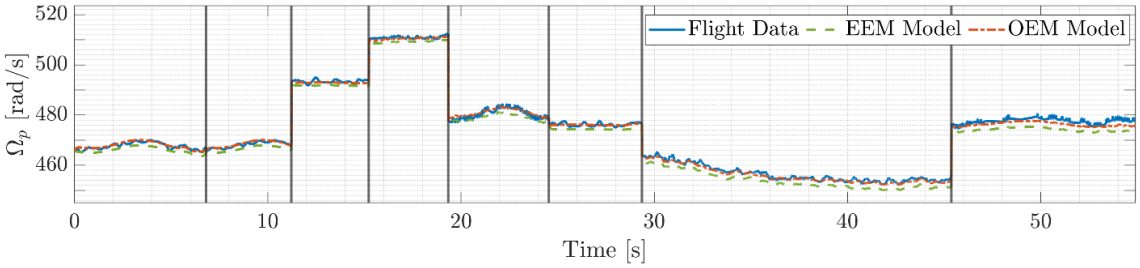


Figure 3.8: Model prediction of the rotation rate for the training and validation sets.

The Output-Error Method is then applied to optimize the results obtained by the EEM. Table 3.6 displays the updated parameters, and Table 3.7 shows the RMSE and NRMSE for both models obtained from the Equation-Error and Output-Error methods. This optimization improves the agreement between the measured and predicted outputs while maintaining the predictability of the model.

Table 3.6: Parameters of the torque coefficient estimated using the Output-Error Method.

Parameter	Value	Error %
C_{Q_0}	1.232e-02	0.2
C_{Q_J}	-8.213e-03	6.8
$C_{Q_{Re}}$	-1.743e-06	8.0

Table 3.7: RMSE and NRMSE for the rotation rate estimation.

	EEM		OEM	
	Training	Validation	Training	Validation
RMSE [rad/s]	1.30	1.25	0.85	0.96
NRMSE [%]	1.4	4.4	0.9	3.4

The determined propulsive model yields Ω_p for the dataset used for determining and estimating the aerodynamic model.

3.6.3 Aerodynamic Model

The coefficients for longitudinal forces and moments (C_X, C_Z, C_m) and lateral-directional forces and moments (C_Y, C_l, C_n) are determined for each respective data

subset. The predefined set of candidate regressors corresponds to the variables that define the longitudinal mode (α, $\hat{q}$, δ_c) and the lateral-directional mode (β, $\hat{p}$, $\hat{r}$, δ_a, δ_r). The propulsive parameters ($\mathcal{J}_c$, i_y, i_z) are considered for both modes, and Ω_p is included to account for Reynolds effects related to thrust.

The MOF method generated candidate model terms from these variables and their combinations up to the third order. Then, a stepwise regression is applied to select the more significant terms. Following the stepwise regression, the resulting model is:

$$\begin{aligned}
C_X &= C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 + C_{X_q}\hat{q} + C_{X_{\delta_c}}\delta_c + C_{X_{\mathcal{J}_c}}\mathcal{J}_c + C_{X_0} \\
C_Y &= C_{Y_\beta}\beta + C_{Y_{\delta_r}}\delta_r + C_{Y_0} \\
C_Z &= C_{Z_\alpha}\alpha + C_{Z_{\delta_c}}\delta_c + C_{Z_{\mathcal{J}_c}}\mathcal{J}_c + C_{Z_0} \\
C_l &= C_{l_\beta}\beta + C_{l_p}\hat{p} + C_{l_r}\hat{r} + C_{l_{\delta_a}}\delta_a + C_{l_{\beta\delta_r}}\beta\delta_r + C_{l_0} \\
C_m &= C_{m_\alpha}\alpha + C_{m_q}\hat{q} + C_{m_{\delta_c}}\delta_c + C_{m_{\alpha\delta_c}}\alpha\delta_c + C_{m_0} \\
C_n &= C_{n_\beta}\beta + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + C_{n_{\mathcal{J}_c\delta_r^2}}\mathcal{J}_c\delta_r^2 + C_{n_0}
\end{aligned} \tag{3.53}$$

The derivative coefficients of the model are determined using the Equation-Error Method. The longitudinal parameters are summarized in Table 3.8, and the lateral-directional coefficients are presented in Table 3.9.

Table 3.8: Longitudinal aerodynamic coefficient parameters results with the Equation-Error method.

Term	Value	Error %	Term	Value	Error %	Term	Value	Error %
C_{X_α}	0.4401	5.0	C_{m_α}	-1.6357	3.2	C_{Z_α}	-6.1813	1.0
$C_{X_{\alpha^2}}$	4.5146	3.8	C_{m_q}	-4.1543	38.5	$C_{Z_{\delta_c}}$	-0.5621	6.0
C_{X_q}	-1.6160	14.6	$C_{m_{\delta_c}}$	0.8528	6.9	$C_{Z_{\mathcal{J}_c}}$	-0.3675	25.5
$C_{X_{\delta_c}}$	-0.0473	9	$C_{m_{\alpha\delta_c}}$	-9.2875	9.5	C_{Z_0}	-0.7714	0.8
$C_{X_{\mathcal{J}_c}}$	0.1195	3.6	C_{m_0}	-0.0932	3.8			
C_{X_0}	-0.0148	5.3						
R^2	88.0%		R^2	91.2%		R^2	92.8%	
NRMSE	4.38%		NRMSE	4.16%		NRMSE	4.34%	

Table 3.9: Lateral-directional aerodynamic coefficient parameters results with the Equation-Error method.

Term	Value	Error %	Term	Value	Error %	Term	Value	Error %
C_{Y_β}	0.0733	1.9	C_{l_β}	-0.0364	3.2	C_{n_β}	0.0315	3.6
$C_{Y_{\delta_r}}$	-0.4103	3.0	C_{l_p}	-0.4937	2.3	$C_{n_{\delta_a}}$	-0.0108	3.6
C_{Y_0}	-0.0020	46.5	C_{l_r}	0.1277	10.8	$C_{n_{\delta_r}}$	-0.0158	2.4
			$C_{l_{\delta_a}}$	0.1058	1.8	$C_{n_{\mathcal{J}_c\delta_r^2}}$	0.0618	19.9
			$C_{l_{\beta\delta_r}}$	0.0268	11.8	C_{n_0}	-0.0001	131.3
			C_{l_0}	-0.0008	16.1			
R^2	92.4 %		R^2	90.8 %		R^2	83.6 %	
NRMSE	4.06 %		NRMSE	3.22 %		NRMSE	5.77 %	

The longitudinal and vertical force coefficients incorporate the propulsive term $\mathcal{J}_c$. Commonly, propeller blade thrust coefficient models present a quadratic form. The absence of the term associated to $\mathcal{J}_c^2$ could be attributed to poorly excitation of the propulsive system dynamic response. The presence of $C_{n_{\mathcal{J}_c\delta_r^2}}$ signifies a coupling effect between the rudder surface and the propulsive system. The influence of i_y, i_z, and Ω_p did not merit inclusion in the model.

In a broader context, actuation surface deflections are found to contribute more significantly to the model than their corresponding angular rates. For both lateral force and yaw moment coefficient models, the model determination step appears less sensitive to the effects of angular rates. A possible explanation could be collinearity. Table 3.10 indicates low pair-wise correlations between candidate regressors, suggesting no significant collinearity. Similarly, Table 3.11 demonstrates no evidence of multicollinearity based on the eigenvalues and condition indices.

Table 3.10: Pair-wise correlation between C_n and C_Y candidate regressors.

	β	$\hat{p}$	$\hat{r}$	δ_a	δ_r	$\mathcal{J}_c\delta_r^2$
β	1.0	-0.082	-0.216	0.172	0.215	-0.003
$\hat{p}$		1.0	-0.139	0.705	-0.051	0.016
$\hat{r}$			1.0	-0.138	-0.150	-0.027
δ_a				1.0	0.039	-0.011
δ_r					1.0	0.604
$\mathcal{J}_c\delta_r^2$						1.0

Table 3.11: Variance Proportions of C_n and C_Y candidate regressors.

μ_i	k_i	π_{ij}						
		1	β	$\hat{p}$	$\hat{r}$	δ_a	δ_r	$\mathcal{J}_c\delta_r^2$
1.694	1.0	0.032	0.028	0.000	0.013	0.010	0.032	0.028
1.283	1.3	0.002	0.001	0.140	0.018	0.098	0.032	0.028
0.966	1.8	0.001	0.021	0.019	0.620	0.007	0.006	0.007
0.910	1.9	0.068	0.076	0.006	0.276	0.019	0.028	0.184
0.531	3.2	0.536	0.049	0.270	0.014	0.329	0.177	0.045
0.493	3.4	0.010	0.066	0.162	0.015	0.132	0.741	0.604
0.443	3.8	0.350	0.758	0.403	0.043	0.405	0.004	0.019

The results of the Equation-Error Method serve as an initial guess for the Output-Error Method procedure. Model outputs include accelerations at the CG, angular accelerations, angular rates, body velocities, and Euler angles. Bias terms are incorporated for each equation of motion (3.1-3.3), estimated alongside other parameters after the simulated model drifts from flight data. These bias terms account for systematic errors undetected during the Flight Path Reconstruction procedure.

An initial iteration for each longitudinal and lateral-directional channel was conducted. During longitudinal parameter optimization, lateral-directional states were used as inputs and vice-versa. Due to systematic errors in measured data, the optimization was conducted by covering the entire 6-Degree-of-Freedom (6DoF) model. When the complete 6DoF model was used, the longitudinal and lateral-directional parameters were optimized iteratively for their respective dataset until convergence. Tables 3.12 and 3.13 present parameters determined using the output-error method for longitudinal and lateral-directional modes, respectively. ΔC denotes the change relative to initial values generated from the EEM (Tables 3.8 and 3.9), reflecting greater changes for parameters with initial higher uncertainty.

Table 3.12: Longitudinal aerodynamic coefficient parameters from the Output-Error Method.

Term	Value	Error %	ΔC %	Term	Value	Error %	ΔC %	Term	Value	Error %	ΔC %
C_{X_α}	0.4193	15.2	4.7	C_{m_α}	-1.6902	2.8	2.8	C_{Z_α}	-6.7902	3.8	9.9
$C_{X_{\alpha^2}}$	4.5176	10.0	0.1	C_{m_q}	-3.5464	51.2	14.6	$C_{Z_{\delta_c}}$	0.0753	266.7	113.4
C_{X_q}	-0.6025	115.8	62.7	$C_{m_{\delta_c}}$	0.7507	5.1	12.0	$C_{Z_{\mathcal{J}_c}}$	-0.3669	4.9	0.2
$C_{X_{\delta_c}}$	-0.0384	25.9	18.9	$C_{m_{\alpha\delta_c}}$	-9.7356	6.2	4.8	C_{Z_0}	-0.8169	2.5	5.9
$C_{X_{\mathcal{J}_c}}$	0.1168	9.3	2.2	C_{m_0}	-0.0984	4.5	5.6				
C_{X_0}	-0.0159	14.1	7.7								

Table 3.13: Lateral-directional aerodynamic coefficient parameters from the Output-Error Method.

Term	Value	Error %	ΔC %	Term	Value	Error %	ΔC %	Term	Value	Error %	ΔC %
C_{Y_β}	-0.5784	12.5	41.0	C_{l_β}	-0.0434	4.4	19.2	C_{n_β}	0.0376	4.1	19.5
$C_{Y_{\delta_r}}$	0.0713	16.8	2.7	C_{l_p}	-0.5132	0.9	4.0	$C_{n_{\delta_a}}$	-0.0072	10.2	33.2
C_{Y_0}	0.0144	82.1	825.7	C_{l_r}	0.1226	11.7	4.0	$C_{n_{\delta_r}}$	-0.0154	2.5	2.4
				$C_{l_{\delta_a}}$	0.1007	1.1	4.9	$C_{l_{J_c\delta_r^2}}$	0.0597	15.1	3.3
				$C_{l_{\beta\delta_r}}$	0.0326	10.4	21.8	C_{n_0}	-0.0007	63.9	724.8
				C_{l_0}	-0.0001	365.2	81.2				

The changes in parameter values could be due to low signal-to-noise ratios and insufficient excitation of the aircraft dynamic response. The bias terms, specifically for the lateral-directional parameters, show a large parameter error in percentage, because these parameters are small in magnitude. Additionally, C_{X_q} and C_{Z_c} present a large parameter uncertainty.

On a follow-up iteration, the mentioned parameters were removed and the Output-Error Method was applied again. Table 3.14 indicates the results of the new longitudinal parameters, and in this table ΔC denotes the change relative to the previous OEM results (Table 3.12).

Table 3.14: Longitudinal aerodynamic coefficient parameters from the Output-Error Method after removal of high uncertainty terms.

Term	Value	Error %	ΔC %	Term	Value	Error %	ΔC %	Term	Value	Error %	ΔC %
C_{X_α}	0.4183	15.6	0.2	C_{m_α}	-1.6987	3.0	0.5	C_{Z_α}	-6.7561	2.2	0.2
$C_{X_{\alpha^2}}$	4.5914	10.6	1.6	C_{m_q}	-3.6466	48.1	2.8	$C_{Z_{J_c}}$	-0.3682	4.5	0.5
$C_{X_{\delta_c}}$	-0.0456	11.8	18.8	$C_{m_{\delta_c}}$	0.7384	4.8	1.6	C_{Z_0}	-0.8152	2.2	0.4
$C_{X_{J_c}}$	0.1182	9.1	1.1	$C_{m_{\alpha\delta_c}}$	-9.8573	5.8	1.3				
C_{X_0}	-0.0163	12.3	2.5	C_{m_0}	-0.0984	4.6	0.6				

Most of the longitudinal terms remained consistent with the previous OEM results, with no significant deviations. The most significant change was observed in $C_{X_{\delta_c}}$, which showed an increase in magnitude and a reduction in percentage error. This change could potentially be attributed to collinearity between $\hat{q}$ and δ_c; the pairwise correlation of 0.599 suggests this is unlikely in this case. Another plausible explanation is that the previously identified term C_{X_q} may have incorrectly attributed the effects of canard deflection on longitudinal dynamics to the pitch rate.

To assess the predictive capability of the developed model, simulations were conducted on the validation set. The goodness of fit was quantified using the R^2 coefficient (Equation 3.42). Additionally, Theil's Inequality Coefficient (TIC), a metric for comparing time histories [113, 134], was employed as a quality criterion, as observed in prior aerodynamic modeling works [120, 135, 136]. TIC is a normalized metric, where 0 indicates a perfect match, and is defined as follows:

$$\text{TIC} = \frac{\sqrt{\frac{1}{N}\left(\mathbf{z}-\widehat{\mathbf{y}}\right)^T\left(\mathbf{z}-\widehat{\mathbf{y}}\right)}}{\sqrt{\frac{1}{N}\mathbf{z}^T\mathbf{z}}+\sqrt{\frac{1}{N}\widehat{\mathbf{y}}^T\mathbf{z}}} \tag{3.54}$$

To mitigate potential issues with TIC caused by nonzero means in time series [120, 134–136], the mean of both measured ($\mathbf{z}$) and estimated ($\widehat{\mathbf{y}}$) time histories was subtracted. A TIC in the range of 0.25-0.3 signifies good agreement between time series [113].

Table 3.15 presents the R^2 and TIC coefficients for both the modeling and validation sets. Metrics were computed separately for longitudinal and lateral-directional variables. Overall, the TIC values indicate a good match for all variables. While the R^2 values are generally satisfactory, longitudinal acceleration (a_x) exhibits a lower fit compared to other states for the out-of-sample data.

Table 3.15: R^2 and TIC coefficients for observation variables in the Output-Error Method.

	Modeling		Validation			Modeling		Validation	
Variable	R^2	TIC	R^2	TIC	Variable	R^2	TIC	R^2	TIC
a_x	0.847	0.182	0.538	0.273	a_y	0.762	0.216	0.912	0.114
a_z	0.952	0.093	0.825	0.137	$\dot{p}$	0.930	0.215	0.935	0.131
$\dot{q}$	0.926	0.141	0.950	0.116	$\dot{r}$	0.867	0.197	0.931	0.139
q	0.953	0.099	0.951	0.107	p	0.954	0.135	0.975	0.078
θ	0.945	0.107	0.864	0.171	r	0.867	0.164	0.941	0.120
u	0.949	0.094	0.895	0.156	ϕ	0.973	0.105	0.976	0.075
w	0.869	0.151	0.869	0.169	ψ	1.000	0.107	0.929	0.137
h	0.969	0.064	0.928	0.119	v	0.875	0.234	0.925	0.135

Figures 3.9 and 3.10 display time series plots of model outputs, flight data, and corresponding surface excitations for longitudinal and lateral-directional states, respectively, for the validation set. Visual inspection reveals discrepancies in simulated and measured longitudinal and lateral accelerations, consistent with the observed metrics in Table 3.15. The timeseries of simulated linear and angular accelerations exhibit some inconsistencies when compared to the actual data. These discrepancies could come from atmospheric perturbations or inaccurately modeled dynamics. The potential causes of modeling inaccuracies include the assumptions of rigid-body dynamics, limitations in the function library used for system identification, insufficient excitation of the vehicle dynamics, or low signal-to-noise ratio in the measurements. Additionally, in fixed-wing mode, the VTOL aircraft's quadcopter propellers are not locked in place, potentially contributing to unmodeled dynamics. The largest discrepancy is observed in the longitudinal acceleration, a_x, suggesting the model struggles most with predicting this variable. Other states show a more satisfactory alignment between the simulated outputs and the flight data.

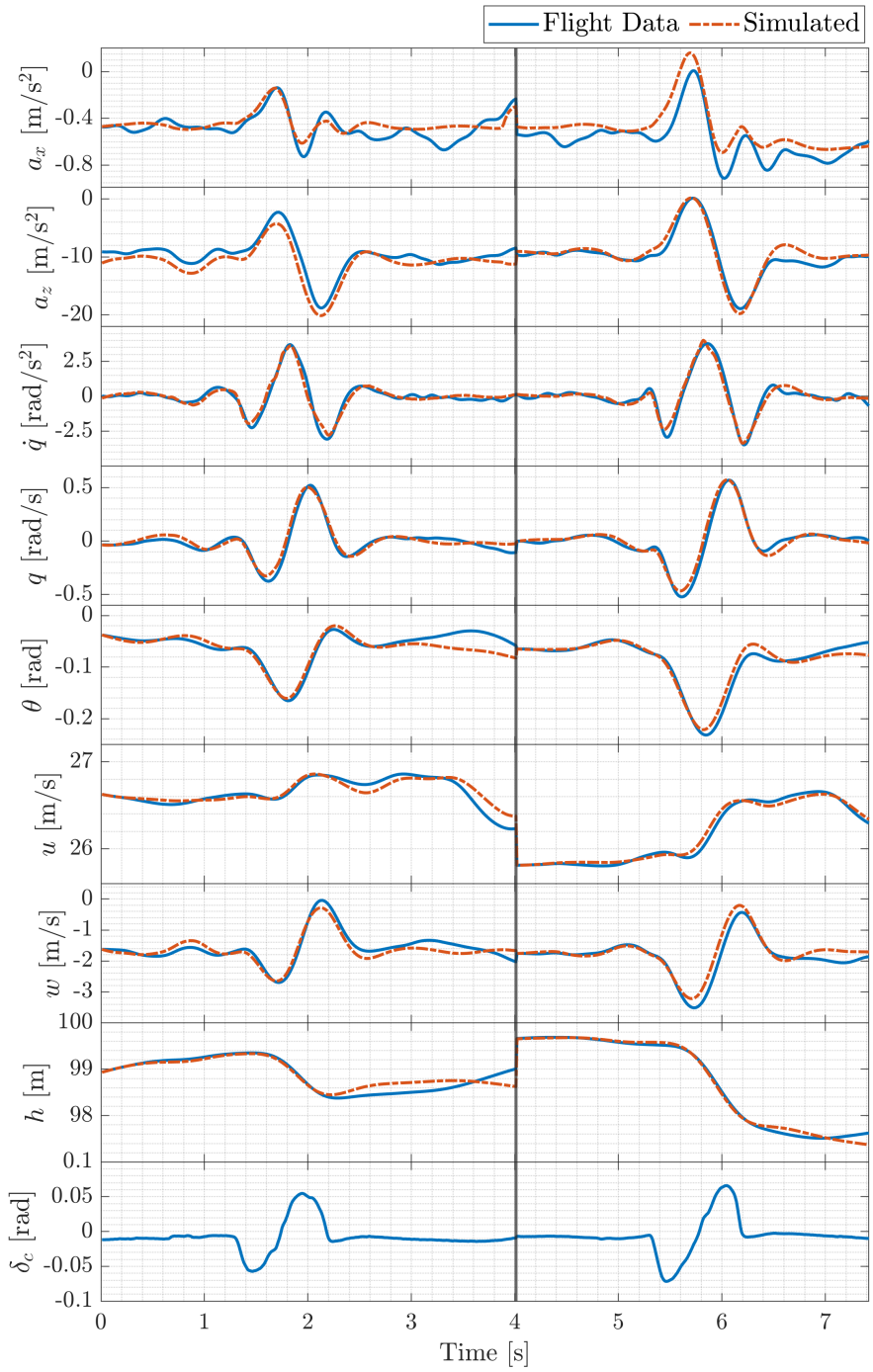


Figure 3.9: Comparison between the model and flight data for the longitudinal states of the validation set.

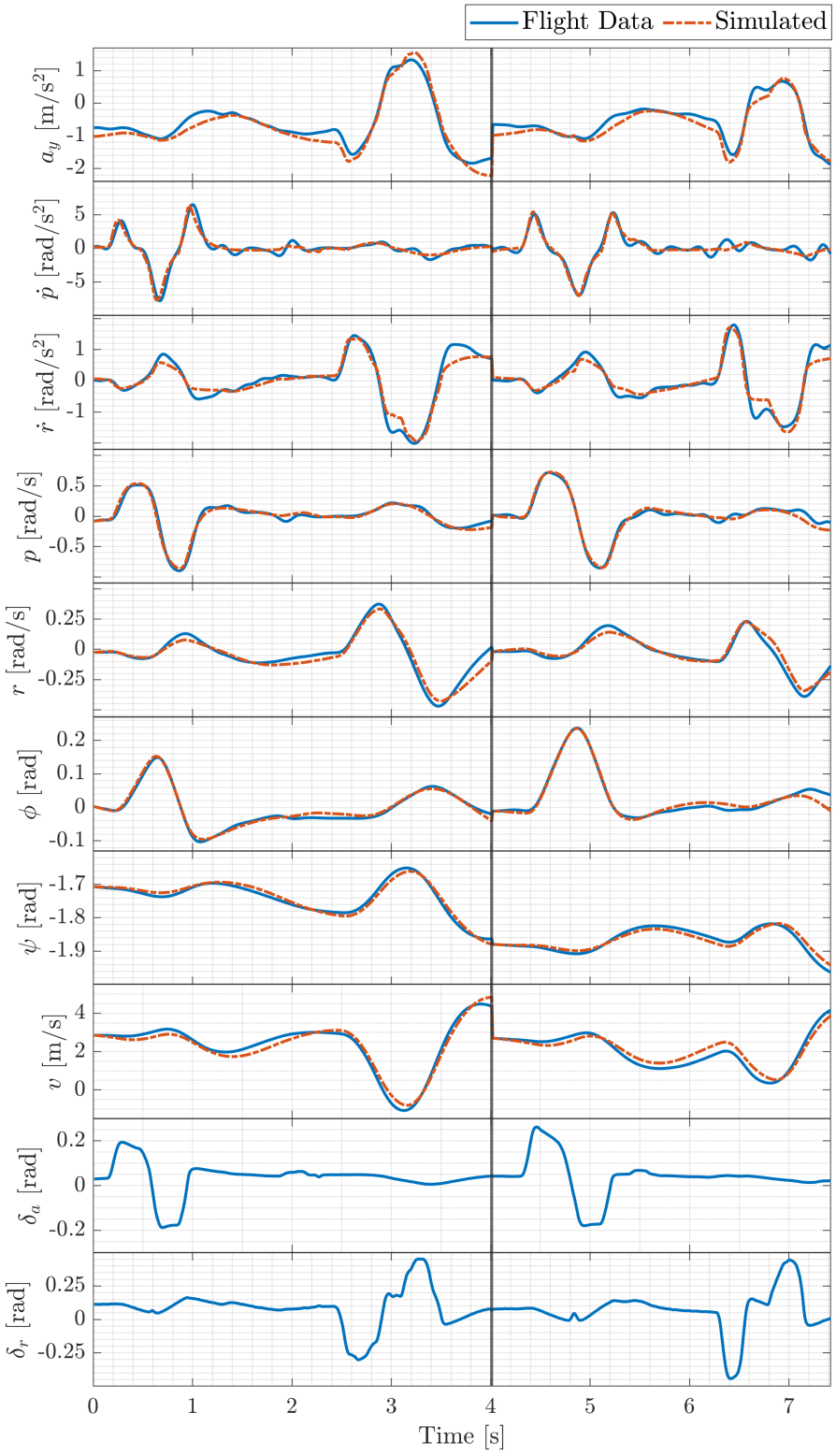


Figure 3.10: Comparison between the model and flight data for the lateral-directional states of the validation set.

3.7 Conclusions

This paper introduced a systematic approach to identifying parameters of a model representative of the fixed-wing configuration of a propeller-based eVTOL aircraft.

Simple static thrust tests were performed to validate the motor model and determine the losses of the electronic speed controller. The contribution of the propeller to the electrical propulsion model was extracted from flight data, utilizing information such as commanded throttle, battery voltage, and logged airspeed during flight. This propeller-based electrical propulsion model facilitated the identification of coupling effects between the propeller and airframe dynamics.

The Multivariant Orthogonal Functions method proved to be crucial in determining nonlinear aero-propulsive variables. The Stepwise Regression and Equation-Error Method established the model and provided an initial prediction of the model parameters, which were ultimately refined with good accuracy using the Output-Error Method.

Factors such as safety constraints during flight testing may have impacted the identifiability of certain aerodynamic terms, particularly angular rate derivatives. This was most notable in the lateral force and yaw moment coefficients. Additionally, the model exhibited reduced predictive capability for longitudinal acceleration, likely due to a low signal-to-noise ratio in the accelerometer data and insufficient excitation of the propulsive system. Nonetheless, the identified models demonstrated strong predictive capability for the other output variables.

To enhance the Eusphyra flight dynamic model, the methodologies and techniques outlined in this paper can be applied to future flight test data with appropriate excitation. The insights gained from this system identification effort offer the potential to reduce safety constraints, expand the flight envelope, and allow for more sophisticated input designs. Despite these potential improvements, the current model provides a realistic representation of the aircraft flight dynamics, serving as a foundation for future research.

Chapter 4

Automatic Autopilot Tuning Framework

In the previous chapter, we explored the development of a physics-based flight dynamic model (FDM), highlighting the significant effort required to produce such a high-fidelity model. The benefits of having this model extend beyond its initial development, as it enables flight simulation in various conditions, allowing for the testing of different controllers and tunings without the necessity of costly and risky flight tests. Furthermore, real-time and accelerated simulations open the door to advanced optimization techniques that generate controllers tailored to specific flight performance requirements.

This chapter introduces a framework for automated autopilot tuning. While leveraging a high-fidelity physics-based FDM remains the preferred approach when available, the true strength of this framework lies in its adaptability to faster modeling techniques. When modifications are made to the aircraft but it remains flight-capable, this method allows for the rapid generation of a new model and subsequent controller tuning. The technique has demonstrated satisfactory results for rate and attitude controllers across different UAV platforms and autopilot systems. Consequently, the extensive FDM development described in the previous chapter is not always necessary if the primary goal is to derive an updated controller tuning efficiently.

The chapter details the full development, testing, and validation of the complete framework, which integrates a simplified system identification process with controller tuning optimization via genetic algorithms. The Mini-E 5050A VTOL rate controller is used as a case study, though the methodology has been successfully applied to

various vehicles and control systems. While the simplified system identification approach forms the foundation of the framework, it can and should be replaced with a higher-fidelity FDM when available to further enhance accuracy.

This chapter is based on the following published journal article:

- Bazzocchi, S., Warwick, S., Suleman, A. (2024). *Automatic autopilot tuning framework using genetic algorithms and system identification.* In: *Aerospace Science and Technology*, Volume 157, 2025, 109779, ISSN 1270-9638. [`https://doi.org/10.1016/j.ast.2024.109779`]

Abstract

This paper presents a comprehensive framework for offline optimization of tuning parameters in unmanned aerial vehicle (UAV) flight controllers. The framework uses system identification to create a simplified flight dynamics model, followed by control law matching to ensure the simulated controller's output closely replicates real-world autopilot commands. The optimization phase employs genetic algorithms to tune parameters based on a defined cost function that incorporates performance requirements. Each stage, from flight dynamics model development to optimization, is validated to ensure enhanced controller performance. Finally, real-world flight tests confirm the effectiveness of the optimized controller, demonstrating the validity of the proposed framework for autopilot tuning optimization.

4.1 Introduction

The integration of autopilot systems has become increasingly common in the development of new vehicle configurations, particularly with sub-scale models. This approach enables the development of progressively more complex vehicles, thereby facilitating significant advancements in vehicle design. However, designing and tuning these systems pose substantial challenges, requiring an optimal balance among metrics such as stability, efficiency, safety, and overall performance.

Several approaches exist for tuning flight controllers in autopilots, including hand-tuning, simulation-based optimization, and machine learning techniques. While hand-tuning remains prevalent in UAV prototyping with off-the-shelf autopilots, it is time-

consuming and often challenging to achieve robust tuning that ensures satisfactory performance.

In environments characterized by rapid prototyping, hand-tuning becomes impractical due to the frequent tuning changes required for modifications in the vehicle's structure, controls, actuation, or sensors. The need for a structured tuning framework that enhances autopilot performance and robustness motivates the exploration of alternative tuning techniques.

This paper presents an autopilot tuning framework developed and tested on the Eusphyra vehicle, a highly specialized Vertical Takeoff and Landing (VTOL) UAV [82, 137]. The project involved several demonstrators, including the Mini-E 5050A vehicle with a flexible structure (Figure 4.1a), which serves as an illustrative example in this paper. Although a single vehicle is used as an example, the proposed autotuning framework has demonstrated success across various other vehicles (Figure 4.1) and control laws. Further details on the development and tuning of these demonstrators can be found in related papers [80, 83, 84, 96, 138].

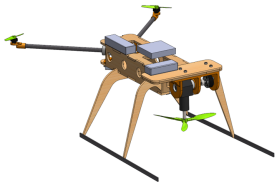
Autopilot tuning can be approached in two ways: online or offline. Online tuning involves adjusting autopilot parameters during flight, allowing real-time optimization based on current operating conditions [115]. Although advantageous for adapting to changing conditions, online tuning carries risks related to potential impacts on stability and safety during flight. Offline tuning, on the other hand, adjusts parameters when the vehicle is not in flight, using data collected under various conditions for subsequent analysis and control parameter optimization. This paper primarily adopts offline tuning due to its safety, reliability, and compatibility with off-the-shelf solutions.

Several methods exist for offline tuning, such as Ziegler-Nichols, Cohen-Coon, Frequency Response Analysis, and Pole Placement. However, many of these methods require a simplified control law, typically PID or dynamic inversion, which may not align with more sophisticated controllers designed for complex vehicle dynamics or higher control levels, such as Total Energy Control Systems or Navigation Controllers. An optimization-based approach overcomes these limitations by accommodating any control law design without requiring controller modifications. Therefore, this paper adopts an optimization-based tuning method.

To solve the optimization problem, various methods can be used, depending on the problem's complexity, the ability to calculate gradients of the objective function, and the number of tuning parameters. Since this architecture is intended for use with

(a) Eusphyra Mini-E 5050A flexible structure

(b) Mini-E 5050B rigid



(c) Mini-E 8020 Tri POC

(d) Tarot 650 Swarm

(e) MIMIQ Hybrid Quad

Figure 4.1: Vehicles used to test the tuning optimization architecture

different types of Flight Dynamics Models (FDM), only optimization methods that do not require explicit knowledge of the system equations were considered. These include Particle Swarm Optimization, Simulated Annealing, Nelder-Mead Direct Search, and Genetic Algorithms. For the simple case study presented in this paper, involving a few tuning parameters, all of these methods successfully converged to the same set of tuning parameters that minimized the cost function. Ultimately, Genetic Algorithm was selected as a robust choice capable of adapting to a wide range of tuning optimization problems and for its ease of implementation, as highlighted in other research papers comparing the performance of nature-inspired optimization methods

[139, 140].

A noteworthy paper that aligns with our requirements is by Pereira D. S. and Pinto J. O. [141], who proposed a framework using Genetic Algorithms for both System Identification (SID) and tuning. However, the work presented in that paper is largely theoretical, lacking real-world application and specificity to UAVs. Other researchers, such as Khuwaja [142] and Tran [143], have explored similar problems in simulation but did not consider critical factors like delays and sensor noise. Tishler and his team have conducted extensive research in the frequency domain, providing valuable tools for system identification (CIFER) [24] and controller tuning (CONDUIT) [144], but their closed-source software limits its applicability. Faced with the absence of a comprehensive architecture for identification and tuning of a generic off-the-shelf autopilot validated through flight tests, this project was initiated.

The remainder of this paper is structured as follows: the methodology section outlines the steps of the proposed approach, and the results section demonstrates the efficacy of the proposed method using simulation and real flight data. The specific example presented involves tuning the rate controller of a PX4 autopilot controlling the Mini-E 5050A UAV during the vertical take-off phase.

4.2 Methodology

Overall Architecture

The proposed tuning optimization architecture, illustrated in Figure 4.2, begins with the input of flight test data. This data undergoes pre-processing, including filtering and interpolation, to ensure consistency for the subsequent steps. Next, both the control laws and the flight dynamic model (FDM) are implemented within the simulation environment. For the control implementation, two alternatives are considered: encoding the control laws directly into the simulation environment or wrapping the autopilot source code. The flight dynamic model is developed using system identification (SID). Alternatively, if a physics-based model of the aircraft is already available, it can be directly integrated into the simulation. Both the flight dynamic model and the control implementation are validated through flight dynamic matching and control matching techniques, ensuring the simulated behavior aligns accurately with real-world performance. At the core of this architecture is the optimization of the con-

troller tuning parameters. This process involves defining relevant states and control signals, designing a cost function, and tuning the hyperparameters. The final stage of validation assesses the performance of the new tuning. Initially, this validation is performed in a simulated environment; however, if available, flight test data is also used to verify the tuning under real-world conditions. The output of this process is a set of optimized autopilot tuning parameters, tailored to the specific vehicle and the desired control objectives.

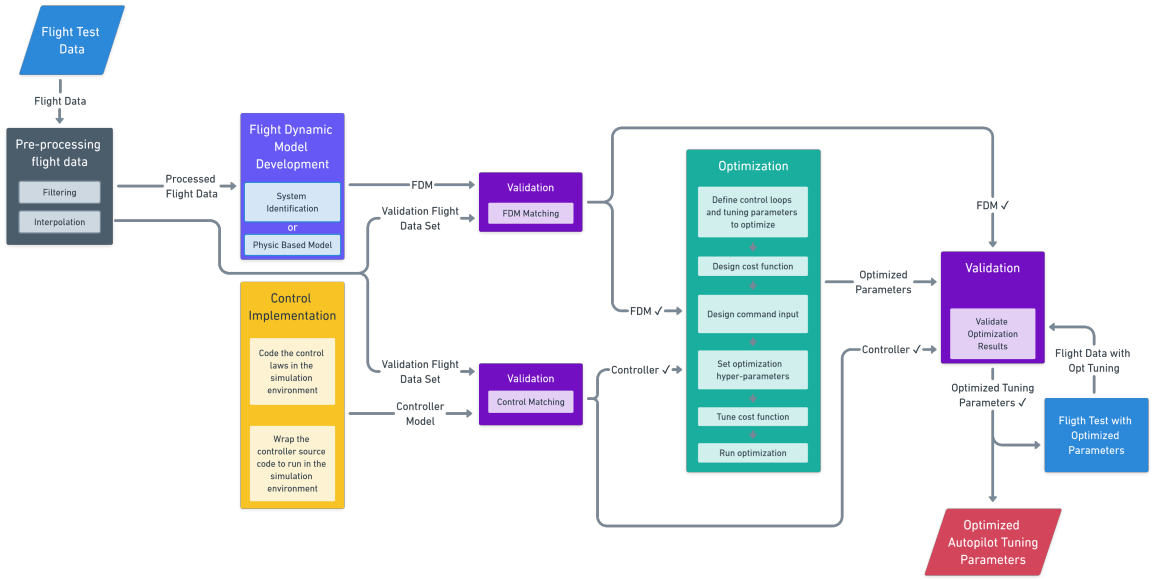


Figure 4.2: Overview of the proposed tuning optimization architecture

The source code developed in this manuscript is available in the the GitHub repository: "Automatic-Autopilot-Tuning-Optimization-PX4-MC-RateController" [145].

The subsequent sections delve into specific steps for developing this tuning framework, exemplified by a rate controller tuning for a multicopter flight-phase of a VTOL aircraft. However, this method is applicable to various vehicles and controllers.

4.2.1 Pre-processing the Flight Log

The first step in the data preparation process involves importing flight data into the chosen programming environment. In this study, data from real flight tests and hardware-in-the-loop (HIL) simulations was used. The data serves two main purposes: it's used for control law matching and, in the case of real flight data, to support the development of a flight dynamics model. To ensure that the dataset is comprehen-

sive, it must contain maneuvers that sufficiently excite the vehicle's dynamics. The selection of autopilot and programming tools, such as PX4 and Matlab/Simulink, influences the data import process.

Once the data is loaded, it undergoes processing through filtering and interpolation techniques. Filtering is applied to eliminate sensor noise, while interpolation ensures uniform timestamp alignment across all data points. Several filtering methods were tested, including the low-pass Butterworth filter, Savitzky-Golay filter, and Extended Kalman Filter (EKF). For the IMU data in this example, a second-order low-pass Butterworth filter with a cutoff frequency of 20 Hz was selected. This choice was made as the low-pass filter effectively removed high-frequency noise while preserving the integrity of data below the cutoff frequency. Although the low-pass filter produced satisfactory results, other filters remain valuable alternatives depending on the application. The EKF is particularly useful when attitude estimation is required. The Savitzky-Golay filter offers stronger smoothing, which may be beneficial when dealing with more significant noise, though with a potential trade-off in altering the dynamic response of the system.

Interpolation is applied to resample the data at a consistent timestep, ensuring compatibility with certain Matlab/Simulink libraries. In this study, simple linear interpolation was used to resample all data to a uniform timestep of 0.0033 seconds (300 Hz). Care must be taken during this process to avoid introducing artifacts that could distort the re-sampled signals.

4.2.2 Control Law Implementation

Efficient automatic tuning requires running the controller faster than real-time within the simulation environment. Although connecting the physical autopilot through a hardware-in-the-loop (HIL) architecture ensures fidelity in the controller's outputs, the main advantage of this method lies in its ability to initiate simulations from specific flight conditions and execute control laws at speeds higher than real-time.

Two approaches are considered for integrating the controller into the simulation environment: wrapping the autopilot source code or directly encoding the control laws into the integrated development environment (IDE).

The first approach involves accessing the autopilot's source code, identifying the relevant control law scripts, and generating a wrapper function around these scripts. This method facilitates easier transitions to newer firmware versions but limits de-

bugging capabilities and requires full access to the autopilot's internal software architecture, which can be restrictive in some cases.

The second approach, adopted in this project, involves directly coding the specific control logic into the simulation environment. This method isolates and optimizes a specific controller while enabling the use of advanced debugging and probing techniques within the IDE. However, any changes made to the control law in future autopilot firmware releases must be manually updated in the simulation model.

For this project, the PX4 angular rate control law from firmware version 1.12.3 was implemented in a Simulink model. A block diagram of the controller is shown in Figure 4.3.

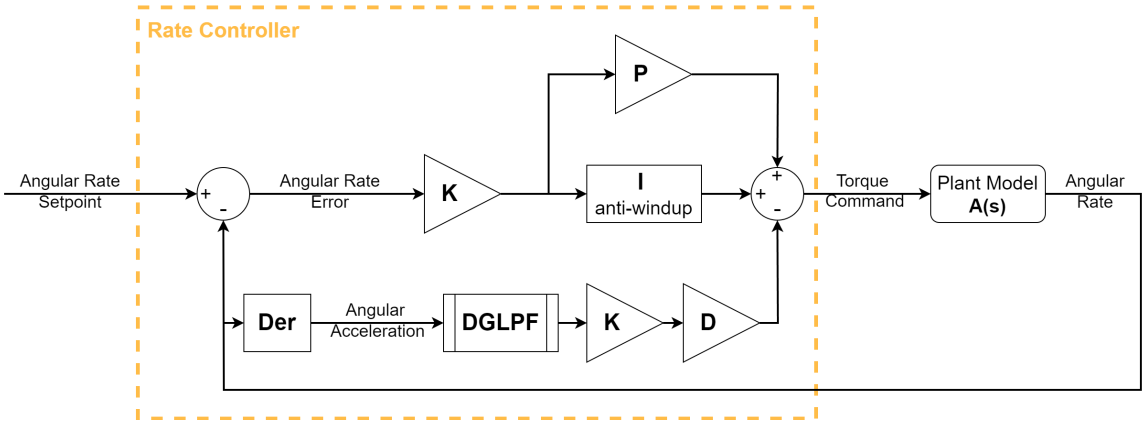


Figure 4.3: Block diagram of the PX4 angular rate control law implemented in Simulink

In this model, K is the proportional gain in the standard form, P is the proportional gain in the parallel form, I is the integrator gain for the anti-windup integrator, and D is the differential gain. The term $Der = \frac{s}{1e10 \cdot s+1}$ represents the derivative term, while $DGLPF = \frac{1}{1+\frac{s}{(2\pi f_c)}}$ is the Differential Gyro Low-Pass Filter applied to angular acceleration data, with a cutoff frequency of $f_c = 20\,\text{Hz}$. Finally, $A(s)$ simulates the dynamic response of the vehicle, and its development is described in the next section.

4.2.3 Flight Dynamic Model Development

This section outlines the development of the flight dynamic model (FDM), which can be achieved through two distinct approaches: physics-based modeling and data-driven modeling.

A physics-based FDM is formulated using mathematical equations that describe the vehicle's dynamics. In the context of a multicopter, essential parameters such as mass, moments of inertia, geometric properties, motor characteristics, and propeller specifications are considered. Submodules are constructed to compute forces and moments, which are then used to solve the equations of motion. For fixed-wing models, aerodynamic forces are also incorporated. This approach allows for adjusting various vehicle parameters to evaluate their impact on the vehicle's response and to assess controller robustness. The model can be further refined by introducing more detailed submodules to increase fidelity.

Notably, during the development of the Mini-E flight dynamic model, a correlation between yaw dynamics and the torsional elasticity of the frame was discovered. System identification was subsequently employed to refine the fixed-wing physics-based model, particularly focusing on certain aero-propulsive parameters [79]. This iterative refinement, though time-consuming, significantly improved model accuracy.

The second approach, which is employed in this work, involves using a data-driven model represented by a single-input single-output (SISO) transfer function. This approach, based on system identification, enables the rapid extraction of dynamic models from measured responses to specific input excitations [146]. While it offers advantages such as speed and accuracy, it has certain limitations, including the inability to modify physical parameters for robustness analysis and the requirement for the vehicle to be flown prior to model tuning and optimization. Nevertheless, in cases where a rapid and accurate dynamic model is prioritized over the ability to manipulate physical characteristics, the system identification method is the preferred choice within this framework.

Transfer Function Estimation with System Identification

The transfer function (TF) identification process is illustrated in Figure 4.4.

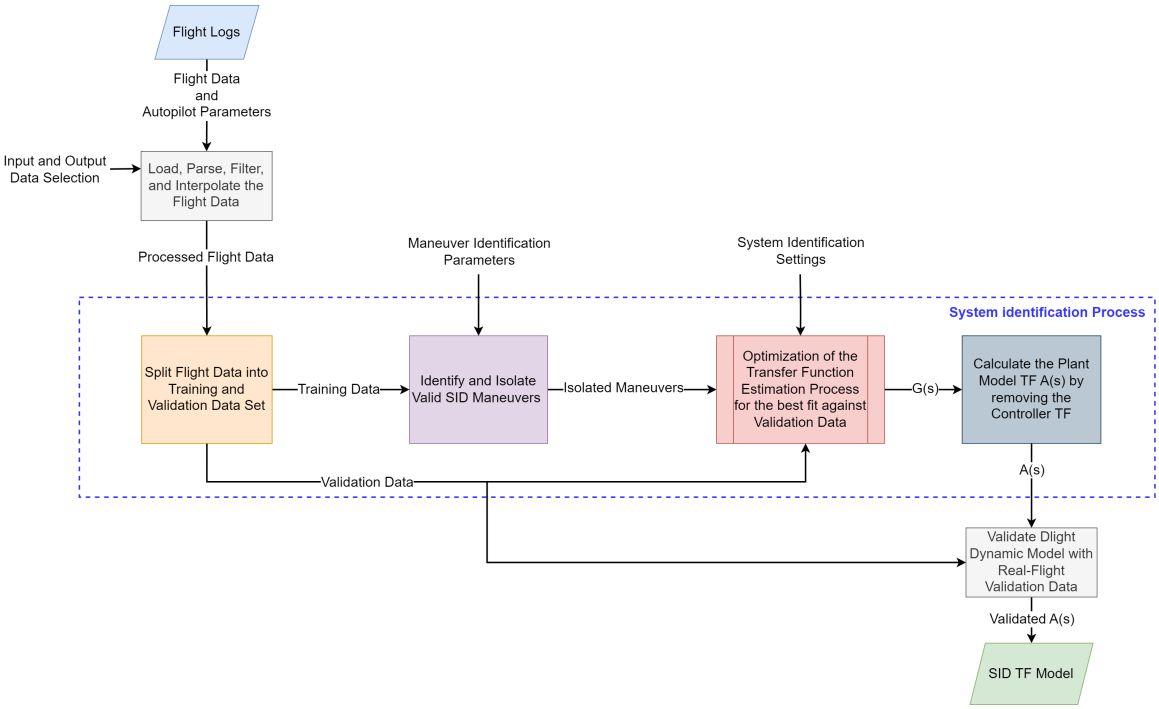


Figure 4.4: Block diagram of the transfer function estimation process

In the presented case study, the system identification (SID) process aims to identify the angular rate response from actuator commands. To execute the SID process, four flight logs are used: one for each axis (roll, pitch, yaw), and one for out-of-sample validation. Parsing these logs isolates the autopilot commands and the vehicle's response, measured as angular rates from gyroscopes. The parsing script also extracts information about control law parameters and filters used during the flights. While four flights were used to segregate the data, it is possible to collect all necessary test points in a single flight and segment the log afterward.

After loading and parsing the data, the pre-processing tasks are executed. Signals are filtered using low-pass filters and re-sampled to synchronize the data on a common timestamp. The data, now a time-series object, is converted into an *iddata* object, a suitable Matlab structure for the SID process, and is then split into training data and validation data to avoid overfitting.

Before the actual TF estimation, the time-series flight data is processed via an algorithm that identifies and isolates flight segments during which maneuvers were executed. This process is automated using a threshold-based logic that isolates flight segments containing maneuvers exceeding a predefined command threshold and a signal-to-noise ratio (SNR) within a maximum response time. The function identifies

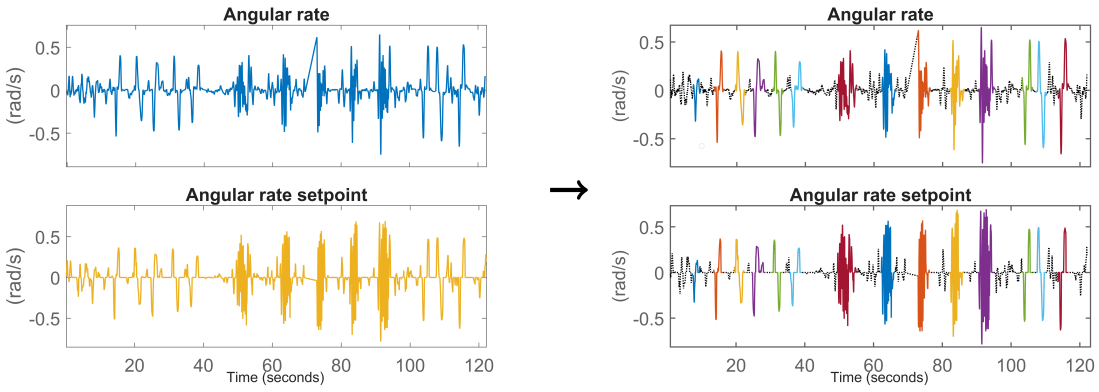
when the maneuver ends by detecting the vehicle's return to nominal trim conditions and only saves maneuvers that exceed a minimum time duration.

For the example presented in this manuscript, the threshold values used for maneuver identification are provided in Table 4.1.

Axis	Command Threshold	SNR Threshold [dB]	Min Duration [s]	Max Response Time [s]
Roll	0.3	10	1	1
Pitch	0.3	10	1	1
Yaw	0.065	5	1	1

Table 4.1: Threshold values for the maneuver identification process for each axis

Figure 4.5 illustrates the maneuver identification process, highlighting the identified maneuvers using different colors.



(a) Roll angular rate response and setpoint from flight test data

(b) Maneuvers detected by the auto-identification function

Figure 4.5: Maneuver isolation process for system identification

The discrete-time transfer function (TF) estimation, using time-domain data, is performed using the Output-Error (OE) model estimation algorithm [147]. The initialization begins with an Autoregressive with Exogenous Variables (ARX) model [148], followed by nonlinear least-squares updates to minimize a weighted prediction error norm [149]. Equation (4.1) outlines the relationship between the output $y(t)$, input $u(t)$, transfer function $G(s) = \frac{B(s)}{F(s)}$, and error $e(t)$.

$$y(t) = \frac{B(s)}{F(s)}u(t - nk) + e(t) \tag{4.1}$$

Adjusting the SID process involves modifying the polynomial orders n_b and n_f of the polynomials $B(s)$ and $F(s)$ in Equation (4.2), which determines the number of poles and zeros of the transfer function $G(s)$.

$$\begin{cases} B(s) = b_{n_b} s^{n_b-1} + b_{n_b-1} s^{n_b-2} + \cdots + b_1 \\ F(s) = s^{n_f} + f_{n_f} s^{n_f-1} + \cdots + f_1 \end{cases} \tag{4.2}$$

The SID process adjustments also involve incorporating transport delay and manipulating the input-output data for estimating polynomial coefficients. System delay is introduced by multiplying the transfer function $G(s)$ by $e^{-d \cdot s}$, where d represents the transport delay in seconds. This typically improves the model's fit to the data. The SID algorithm allows for setting the delay to zero, a known value, or letting the initialization process estimate the delay by setting it to Not a Number (NaN).

The SID process further involves adjusting the training data used for estimating the transfer function. A grid search method is used to explore different combinations of parameters, such as the number and type of maneuvers, the three delay options, and different numbers of poles and zeros. The resulting multidimensional matrix is populated with fitness values for each estimated model. Model fitness, assessed against validation data to ensure out-of-sample optimization, is calculated using the Normalized Root Mean Square Error (NRMSE) between the validation data output y and the identified system output $\hat{y}$, as per Equation (4.3).

$$\text{fit} = 100 \left(1 - \frac{\|y - \hat{y}\|}{\|y - \text{mean}(y)\|} \right) \tag{4.3}$$

The best-fitting models are highlighted within the matrix and presented to the user. Generally, it is preferable to select lower-order models that provide satisfactory fitness levels. While the process is automated, visual inspection of the simulated time-series response of these models is recommended to ensure that the selected model exhibits no undesirable behavior that may not be captured by the goodness-of-fit metric.

In the initial Mini-E 5050A flights, the autopilot's actuator command signal exhibited noise due to a poorly tuned controller and filters. To address this, the angular rate setpoint, a cleaner signal generated by the pilot in "acro-mode", was used as the input for the transfer function estimation (instead of using the actuator commands directly). However, this required an additional step: subtracting the controller model from the estimated transfer function. From the control law shown in

Figure 4.3, the transfer function estimated by the SID process can be expressed as $G(s) = \frac{\text{Angular Rate}}{\text{Angular Rate Setpoint}}$, which can be decomposed as $G(s) = C(s) \cdot A(s)$, where $C(s) = \frac{\text{Torque Command}}{\text{Angular Rate Setpoint}}$ is the controller transfer function and $A(s) = \frac{\text{Angular Rate}}{\text{Torque Command}}$ is the plant model transfer function. Using block diagram reduction rules (series, parallel, and feedback connections), the diagram can be simplified as shown in Figure 4.6.

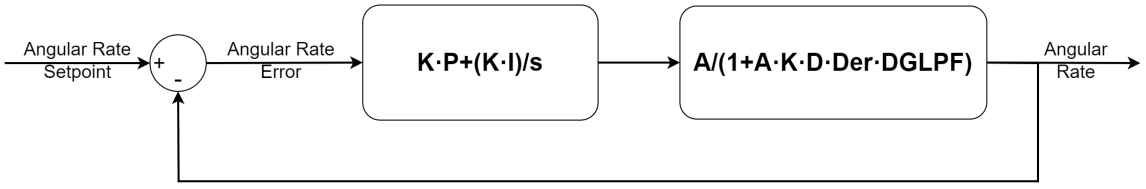


Figure 4.6: Reduced block diagram of the angular rate controller and plant model

By further reducing the block diagram, $G(s)$ can be expressed as per Equation (4.4).

$$G(s) = \frac{\left(K \cdot P + \frac{K \cdot I}{s}\right) \frac{A}{1+A \cdot K \cdot D \cdot Der \cdot DGLPF}}{1 + \left(K \cdot P + \frac{K \cdot I}{s}\right) \frac{A}{1+A \cdot K \cdot D \cdot Der \cdot DGLPF}} \tag{4.4}$$

Note that the "I anti-windup" PX4 logic was simplified to a basic integrator term. This assumption holds true as long as the I term does not reach saturation. By solving for $A(s)$, we obtain Equation (4.5), which provides the isolated dynamic model of the vehicle for a specific axis based on the identified TF and controller parameters.

$$A(s) = -\frac{G}{G\left(K \cdot D \cdot Der \cdot DGLPF + K \cdot P + K \cdot \frac{I}{s}\right) - K \cdot P - K\frac{I}{s}} \tag{4.5}$$

Reducing noise and vibration from the data used for identification is crucial. Investigating potential sources of vibration before starting the SID process, including propeller balancing, checking for loose components, and employing onboard filtering methods, can significantly simplify and improve the system identification process.

Noise and Vibration Design

While noise and vibration suppression are important for the system identification process, reintroducing realistic noise that the controller will encounter in real-life scenarios is equally crucial for controller tuning.

To achieve this, the FDM outputs are passed through sensor models, which add white noise and biases to replicate onboard sensor measurements. In this study, noise and vibrations are introduced by using a white noise source, a gain to set the appropriate amplitude, and a filter to mimic the onboard filtering performed by the autopilot. The values used are derived from real flight data collected during a trim condition without commanded perturbations. The noise is added as the standard deviation from the mean of the response signal, with values of 1×10^{-4} for roll, 3×10^{-4} for pitch, and 7×10^{-5} for yaw. To improve the design, noise and vibration should be separated into different subsystems: the first subsystem should account only for sensor noise using sensor data from ground tests, while the second should model vibrations as a function of the motor RPMs. Introducing noise into the feedback signals allows the tuning and optimization of filters while preventing the controller from overreacting to noise. For robustness analysis, noise and vibration may be artificially increased to ensure that the optimized controller is resilient to elevated noise levels.

4.2.4 Optimization Framework

After integrating the control laws into the simulation environment and identifying the flight dynamic model of the vehicle, it becomes possible to evaluate the vehicle's performance with different tuning parameters. This process is encoded in a script that automates simulations, enabling the exploration of various parameter configurations and the identification of the optimal tuning set. A schematic representation of this optimization algorithm is illustrated in Figure 4.7.

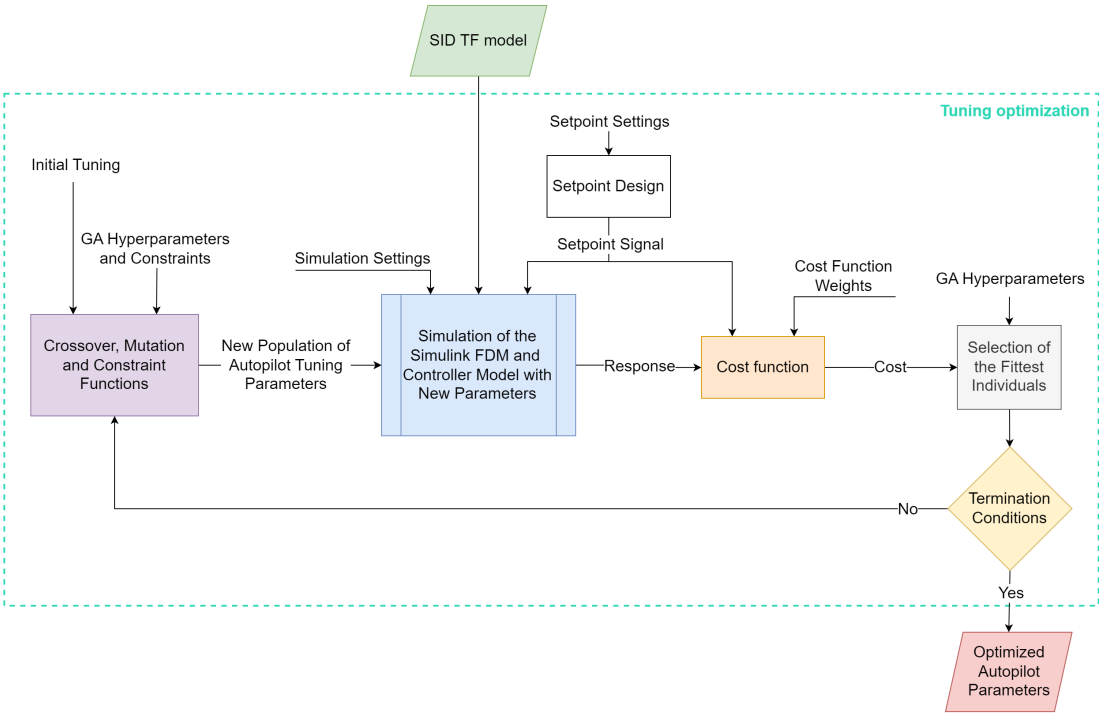


Figure 4.7: Block diagram of the optimization algorithm

Steps for Developing a Controller Tuning Optimization Framework:

1. **Define Control Loops and Tuning Parameters to Optimize**

   The first step involves selecting which control loops will be optimized at the same time during the tuning process. To simplify the optimization, it is recommended to isolate the control loops both laterally (optimizing one axis at a time) and vertically (optimizing inner loops separately from outer loops). In this study, the angular rate controller was optimized independently from the attitude controller, and each axis (roll, pitch, and yaw) was optimized in isolation. This approach reduces the complexity of the cost function and improves convergence. Simultaneously optimizing related controllers would expand the design space, requiring a more complex cost function. Tischler [144] discusses the benefits of sequential versus simultaneous optimization for nested control architectures. For this study, isolation was preferred to focus on optimizing the attitude and rate controllers independently for each axis. In more complex control systems, such as the Total Energy Control System (TECS), isolating control loops can be more challenging due to the numerous inter-dependencies between loops. For example, in the PX4 firmware, TECS involves multiple

interconnected loops, complicating the isolation process.

The next aspect of this step is determining which parameters of the defined control loops should be optimized. Emphasizing essential parameters reduces the design space and, thus, the time needed for convergence. In the case of the angular rate controller, only four parameters were considered: P, D, I, and $DGLPF$. Other parameters, such as K, i_{windup}, FF, and $GLPF$, were excluded from the optimization. K was set to 1 to maintain the PID in the parallel form instead of the standard one [150]. The integral anti-windup parameter i_{windup} was not included since it does not directly affect controller performance during nominal simulations, assuming the integral term does not saturate. If saturation occurs, it indicates an imbalance in the model that should be addressed separately. The feed-forward term FF was excluded due to its potential to cause instability if there are discrepancies between the flight dynamic model and the real vehicle, as demonstrated by Sanchez-Cuevas [151]. Given that acceptable performance was achievable with a standard PID architecture, FF was set to zero. The low-pass filter cut-off frequencies, $DGLPF$ and $GLPF$, were also considered: $GLPF$ was manually set to minimize noise while maintaining minimal latency, and $DGLPF$, which applies to the derivative term, was included as a parameter for optimization.

Properly setting bounds and resolutions for each variable is crucial to avoid an unnecessarily large design space and to enhance convergence speed. Considering the limitations imposed by the autopilot documentation or tuning parameters for similar vehicles can help in defining suitable upper and lower bounds. The bounds and resolution used in this study are presented in Table 4.2.

Parameter	Lower Bound	Upper Bound	Resolution
P	0.01	5	0.01
I	0	5	0.01
D	0	1	0.0005
$DGLPF$	12	120	1

Table 4.2: Bounds and resolution constraints for optimization

2. **Design the Cost Function**

A critical part of the optimization process is designing the cost function, which balances the performance and robustness of the controller. The cost function quantifies the objectives the control engineer aims to achieve, with the primary component typically being the error between the setpoint and the vehicle response. In this case, the error is represented as a modified Integrated Time-weighted Squared Error (ITSE) between the angular rate setpoint and the estimated angular rate, where the error is doubled in the event of an overshoot. To further reduce controller aggressiveness and improve robustness, additional components, such as the deviation of the command signal from the trim condition and the rate of change of the command signal, were incorporated. The cost function J for this study is presented in Equation (4.6).

$$J = Q\cdot\int_0^{t_{sim}} \left[(e + \min(0, e)) \cdot t\right]^2 dt + R\cdot\int_0^{t_{sim}} u^2 dt + C\cdot\int_0^{t_{sim}} \left(\frac{du}{dt}\cdot t\right)^2 dt \quad (4.6)$$

The tunable weights (Q, R, and C) balance the contributions of each component. The ITSE was chosen after analyzing various objective functions [152–154]. The deviation from the trim point was not time-weighted to avoid penalizing a steady-state command necessary to reach the setpoint.

The cost function complexity can vary depending on the controller being optimized. Metrics such as overshoot, settling time, steady-state error, and rise time may also be included in the cost function to fulfill specific project requirements. For more complex control systems, additional signals can be included to prevent undesirable behavior.

3. **Design the Command Input**

Control optimization commonly employs command inputs such as steps, singlets, and doublets. The selection of input shape, amplitude, and timing should ensure that the amplitude aligns with the vehicle's expected response under nominal conditions. An amplitude that is too small may be overwhelmed by noise, while an excessively large amplitude may lead to overly aggressive tuning. Additionally, the maneuver duration must be sufficient to capture the full response and should remain consistent across optimizations to maintain the relative weighting of steady-state error. In this study, a simple step input was

used with amplitudes of 1 rad/s for roll and pitch, and 0.8 rad/s for yaw. The step durations are 1.5 s for roll and pitch, and 1.6 s for yaw.

4. **Set Optimization Hyperparameters**

   Selecting appropriate hyperparameters, such as population size, initial population, termination conditions, design variable properties, and evolutionary properties, is essential for effective optimization. A well-distributed and well-sized population is crucial for adequately exploring the design space. Using the rule of thumb derived from the maximum clique problem by Gotshall [155], a minimum population size of 77 was chosen for optimizing four parameters. Increasing population size improves design space exploration but also increases the number of simulations per generation. The termination criteria, ideally based on proof of convergence, can be determined by tracking the average relative change in the best fitness value over a given number of generations. Additional termination flags, such as maximum runtime or number of generations, should also be set. Design variable properties, including bounds and resolution, help define and constrain the design space. Evolutionary properties determine mutation, crossover, migration, and survival rules, which must be balanced between exploiting the current best individuals and exploring for a global minimum [156].

   Two sets of hyperparameters are recommended: one for the actual optimization and one for tuning the cost function. The latter should be optimized for speed to evaluate the cost function's behavior quickly, which may involve reducing variable resolution, increasing function tolerance, reducing population size, and narrowing the design space.

5. **Tune the Cost Function**

   The final step before running the optimization script is to tune the cost function weights. These weights should not only scale each component to similar magnitudes but also prioritize the desired performance aspects. Running individual simulations with different tuning parameters provides diverse responses to verify each component's effect on the cost function. Then, fast optimization runs with various weight combinations help determine appropriate weight magnitudes. If a weight's contribution is negligible, it may indicate an issue with the cost function design. This process can be iteratively refined by gradually

reducing the resolution of weight adjustments or manually fine-tuning weights. The weights used for the optimization in this study are $Q = 1.2$, $R = 2$ and $C = 0.002$. Cost function tuning often requires multiple optimization runs, making the use of a faster set of hyperparameters advantageous.

6. **Run the Optimization**

   With the optimized and tuned script, the final optimization can be executed using the main set of hyperparameters. For gain scheduling, separate optimization runs should be conducted for each set of flight conditions. Visual inspection of the time-series plot of the optimized response is recommended to ensure the algorithm does not exploit flaws in the flight dynamics model or cost function to minimize the cost value. It is also essential to verify that optimized parameters do not reach user-imposed upper or lower limits. If they do, relaxing these constraints might lead to improved solutions.

4.2.5 Validation

Each fundamental component of the tuning optimization framework undergoes a validation process to ensure its effectiveness. Although validation needs to be performed only once, it is recommended to repeat the process when applying the method to different vehicle designs or implementing new control laws. Similar to other stages of this optimization framework, the validation process is automated, utilizing inputs such as the FDM, optimized gains, and, if available, real flight test logs featuring the optimized gains.

The validation process is structured into three parts:

1. *Flight Dynamic Model Matching*

   This step involves confirming the fidelity of the Flight Dynamic Model (FDM) with respect to the actual dynamics of the vehicle. While validating a physics-based FDM can be time-consuming, an FDM constructed through system identification inherently undergoes a basic validation process. The validation algorithm extends this by calculating additional fitness metrics, such as Mean Squared Error (MSE), Final Prediction Error (FPE), and R-Square against both training and validation data. Time-series plots of the best-fitting models, along with additional plots that analyze artificially introduced noise versus real recorded noise, and comparisons between the command signal generated by

the autopilot during simulation and real flight data, provide a comprehensive validation.

2. *Control Matching*

   Control matching ensures that the output of the control law implemented in the simulation environment aligns with the output from an actual autopilot. An additional validation step is conducted in this study to verify the correct isolation of the control law from the estimated transfer function $G(s)$, which contains both vehicle dynamics ($A(s)$) and the control law ($C(s)$). Subsequently, the controller itself is directly compared with the real controller output, using flight data from the system identification (SID) flights that contain autopilot inputs and outputs. It is important to note that matching during specific flight phases does not guarantee universal matching across all flight conditions, as autopilots often employ logic switches for different control laws in various phases. Moreover, the matching process becomes more intricate if the control law is encoded in the simulation environment rather than wrapping the compiled controller used in the autopilot.

3. *Optimization Process Validation*

   The validation of the optimization process assesses whether the newly optimized parameters genuinely enhance performance compared to the initial tuning parameters. The validation script conducts a series of simulations that directly compare the two tuning sets. Assuming the cost function was appropriately designed and the optimization converged to a solution, the new optimized response should meet the performance requirements that guided the design of the optimization process. This evaluation is carried out both graphically, using time-series plots, and analytically, by calculating performance metrics such as rise time, damping factor, overshoot, and steady-state error. The script also plots several responses from the last generation, demonstrating convergence toward the optimal response with a few diverging traces (under and over-tuned) to ensure a balance between exploitation and exploration of the evolutionary properties. Additionally, the validation process checks for convergence by examining the cost function value of the best individual in each generation. Finally, the optimization's predicted response can be validated against real flight test data by loading the optimized gains into the autopilot configuration, flying the

vehicle, and comparing the flight data with the predicted optimized response. This test, like the SID flights, is preferably conducted in a controlled environment with minimal disturbances and similar inputs to those used during optimization.

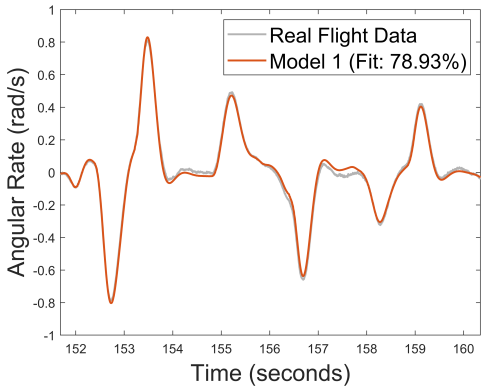
The following section presents plots and analyses generated during the validation process, demonstrating the effectiveness of the architecture and the performance improvements achieved through optimization.

4.3 Results

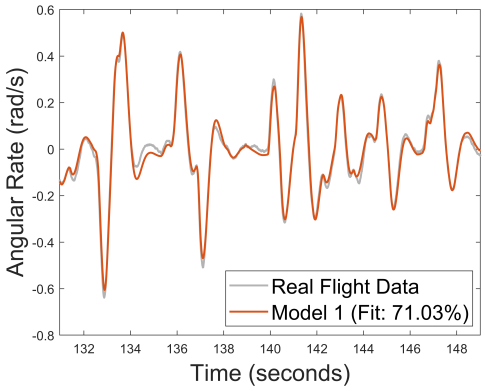
The results section presents the outcomes of applying the tuning optimization framework to the PX4 multi-copter rate controller governing the Mini-E 5050A VTOL vehicle. This section is structured similarly to the Validation section described previously, with three subsections: Flight Dynamic Model Matching, Control Matching, and Optimization Results.

4.3.1 Flight Dynamic Model Matching

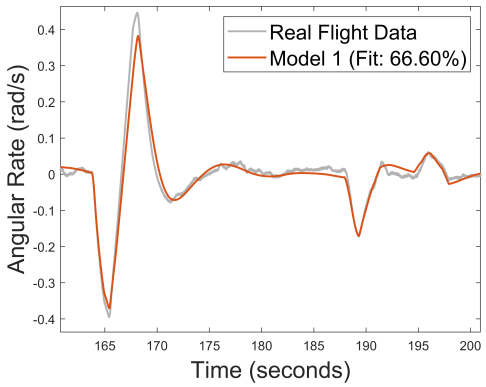
The system identification process includes optimizing the transfer function estimation using out-of-sample flight data, which serves as an initial validation step. This validation uses the NRMSE metric, as described in Equation (4.3). By simulating the estimated transfer functions with the angular setpoint inputs from real flight data, we can visualize the identified model's response against the actual flight data, as shown in Figure 4.8. The figure provides a close-up view of a maneuver for each axis. It is important to note that the fit metric is computed using the entire validation flight, which includes maneuvers across all axes and hovering segments. If only maneuver sections were used, the fit would be higher.



(a) Roll rate



(b) Pitch rate



(c) Yaw rate

Figure 4.8: Comparison of best-fitting transfer functions against the real vehicle response

In this case, the models for roll and pitch were derived exclusively from chirp maneuvers, while the yaw models were obtained from doublet maneuvers. The yaw chirp maneuvers failed to sufficiently excite the vehicle dynamics because the chirp frequency range did not extend to low enough frequencies. This is evident in Figure 4.9, which shows the setpoint and response of the yaw angular rate during the SID flight containing yaw maneuvers. The chirp maneuvers did not reach sufficiently low frequencies to bring the response in phase with the command, necessitating the use of doublet maneuvers for system identification.

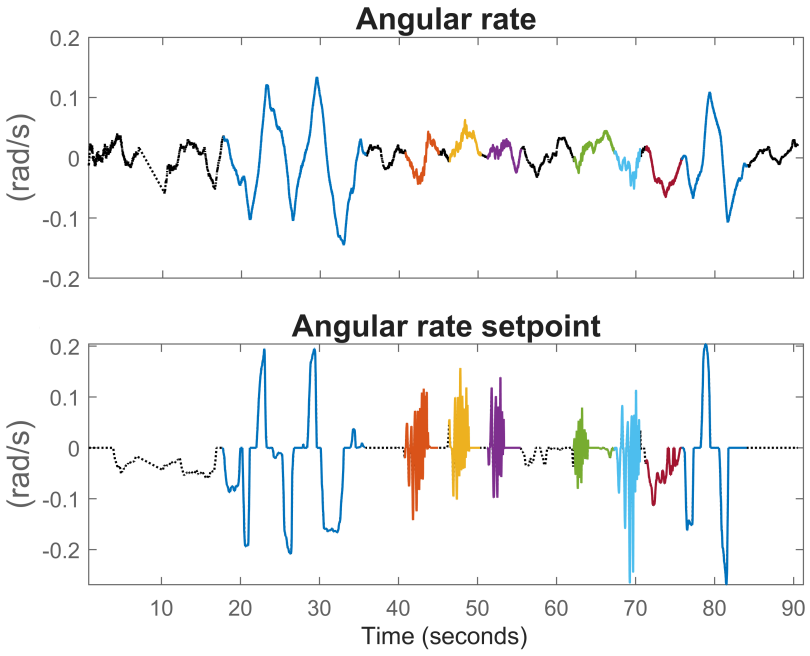


Figure 4.9: Command and response of the yaw angular rate during doublet and chirp maneuvers

As outlined in the methodology, relying only on the fit metric could lead to the selection of a transfer function that poorly represents the vehicle dynamics. Therefore, it is recommended to visually inspect the transfer function's time-series responses before proceeding with any optimization. In this study, the selected transfer functions (G) for tuning optimization are presented for each axis in Equation (4.7).

$$\begin{cases} G_{roll} = e^{-0.0362s} \cdot \frac{0.08011s^2+2.324s+168.6}{s^2+15.43s+169.6} \\ G_{pitch} = e^{-0.0646s} \cdot \frac{-0.0418s^2+7.602s+18.16}{s^2+8.174s+19.47} \\ G_{yaw} = e^{-0.0547s} \cdot \frac{0.07029s^2+0.8329s+0.2963}{s^2+0.9628s+0.4283} \end{cases} \tag{4.7}$$

Using Equation (4.5), the corresponding open-loop transfer functions (A) are calculated and shown in Equation (4.8).

$$\begin{cases} A_{roll} = e^{-0.0362s} \cdot \frac{8.011e08s^7+3.56e10s^6+2.181e12s^5+2.996e13s^4+2.859e14s^3+2.859e04s^2}{2.76e09s^7+8.377e10s^6+1.133e12s^5+7.433e12s^4+4.963e12s^3+3.249e11s^2+32.49s} \\ A_{pitch} = e^{-0.0646s} \cdot \frac{-4.18e08s^7+7.26e10s^6+7.948e11s^5+2.965e12s^4+3.536e12s^3+353.6s^2}{2.396e09s^7+2.351e10s^6+8.315e10s^5+1.159e11s^4+1.132e11s^3+6.373e10s^2+6.373s} \\ A_{yaw} = e^{-0.0547s} \cdot \frac{0.07029s^6+0.9005s^5+1.128s^4+0.642s^3+0.1269s^2}{0.5578s^6+0.6336s^5+0.4137s^4+0.1228s^3+0.03758s^2+0.001131s} \end{cases} \tag{4.8}$$

Note that $e^{-x \cdot s}$ represents the time delay term, introducing x seconds of delay between the command and response. While $G(s)$ uses setpoints as inputs, $A(s)$ uses normalized torque commands, and both return angular rates as outputs.

Table 4.3 presents the fitness metrics of the identified models against both the training data, calculated as NRMSE (Equation (4.3)), MSE, FPE, and the out-of-sample data, calculated as NRMSE and R-squared.

	Fit on training data (in-sample)			Fit on validation flight (out-of-sample)	
Axis	NRMSE	MSE	FPE	NRMSE	R-square
Roll	92.41%	2.178e-4	2.204e-4	78.93%	95.26%
Pitch	84.26%	6.288e-4	6.359e-4	71.03%	90.48%
Yaw	95.21%	7.078e-6	7.122e-06	66.60%	88.85%

Table 4.3: Fitness metrics of SID model calculated on training and validation flight data

The observed discrepancies in vehicle response, as reflected in the various fitness levels, can likely be attributed to using a SISO model instead of a MISO or MIMO model, as well as unaccounted dynamics of the motor/propeller, structural elasticity, and coupling effects during flight. The vehicle exhibited high deflections in the fuselage's torsional mode and the wing and canard's bending mode. Additionally, coupling between roll command and yaw response is likely due to the large aerodynamic surfaces. Despite potential improvements in fitness metrics achievable through further refinement of the SID process, the results from this simplified model still provide a sufficiently representative response to enable controller optimization without requiring additional iterations or model refinement.

Figure 4.10 validates the identified model's response, driven by real flight command setpoints. It illustrates how the designed noise injected into the sensor blocks matches both the amplitude and frequency of the real flight data, minimally impacting model fitness during both a maneuver, as shown in Figure 4.10.A, and while hovering, as shown in Figure 4.10.B.

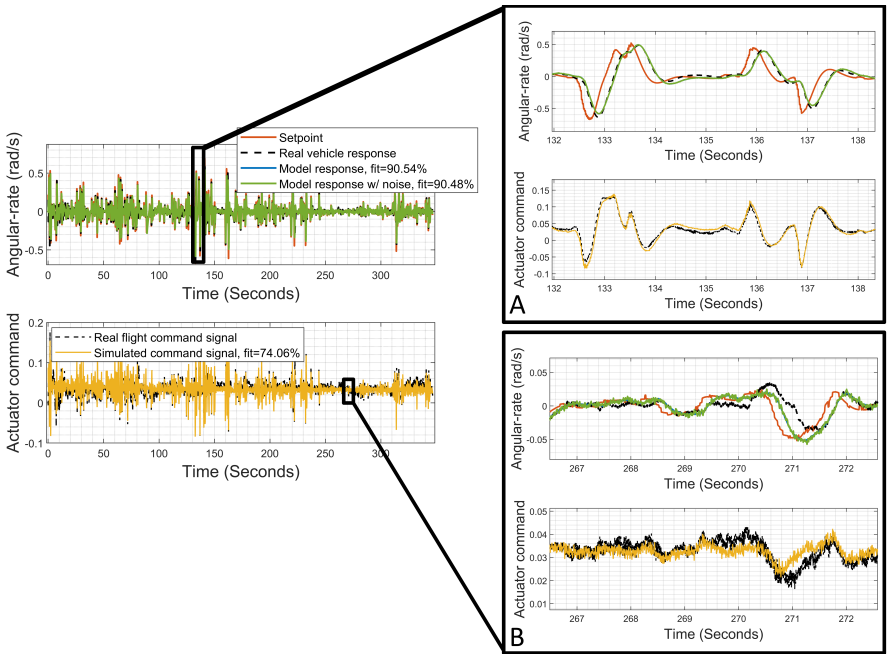
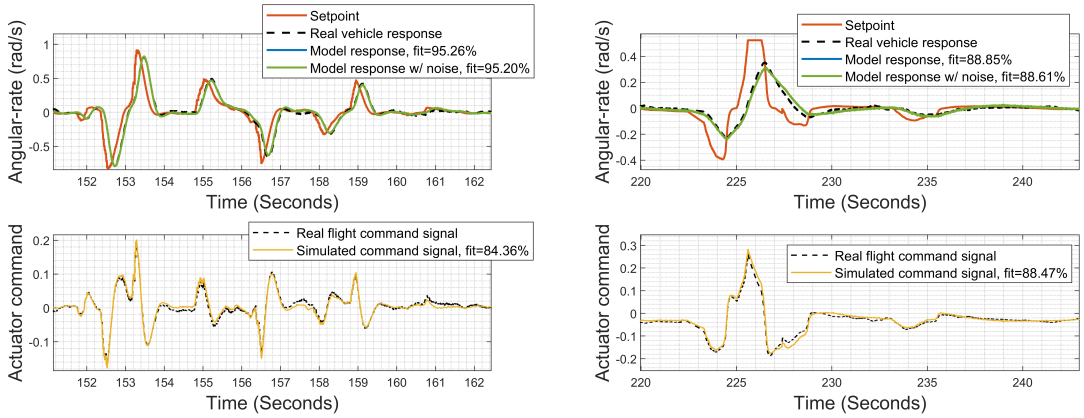


Figure 4.10: Pitch rate model compared to real vehicle response during a maneuver and hovering

For completeness, Figure 4.11 shows similar data for the roll and yaw models during a maneuver.



(a) Roll (b) Yaw

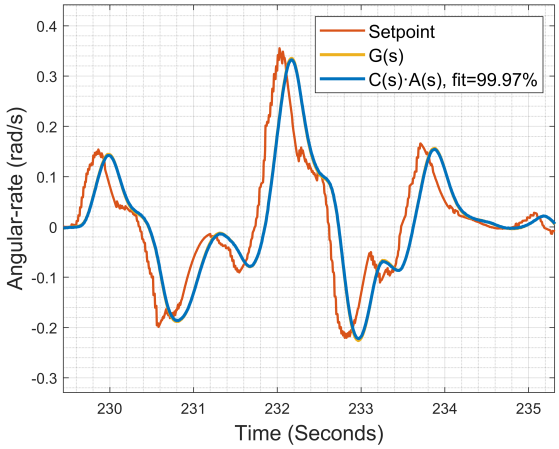
Figure 4.11: Roll and yaw rate models compared to real vehicle response during a maneuver

The SID flights presented here were conducted indoors to minimize atmospheric disturbances, whereas the validation flights were conducted outdoors. Nonetheless, successful outdoor SID flights were achieved under stable atmospheric conditions

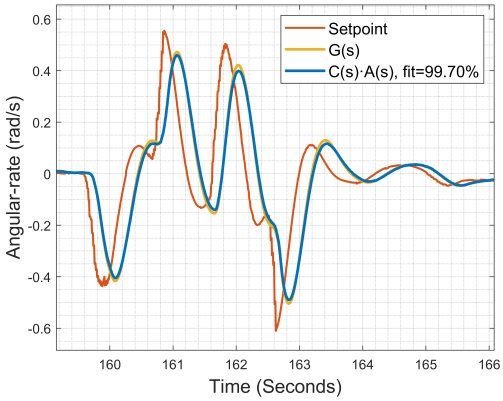
(low wind and minimal turbulence), generating acceptable results. As outlined in the SID methodology, parameter adjustments, including enabling delay estimation and modifying maneuver combinations, were explored. The best results consistently involved enabling delay estimation and limiting transfer function order to 2 or 3 to prevent overfitting on the training data.

4.3.2 Control Matching

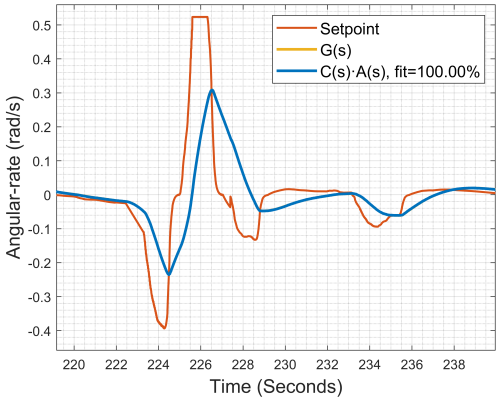
To validate the process of isolating the plant ($A(s)$) from the controller ($C(s)$) using Equation (4.5), a comparison was made between the closed-loop transfer function ($G(s)$) and the plant response paired with the simulated controller. Figure 4.12 visually shows the signal matching and the R-squared metric (reported in the figure's legend) numerically expresses a goodness of fit exceeding 99% for all axes. The minor differences at the peaks can likely be attributed to the different differentiation methods between the controller and the simulation. This aligns with the fact that the yaw axis has a fit of 100% since the differential gain of the controller for that axis is set to zero.



(a) Roll



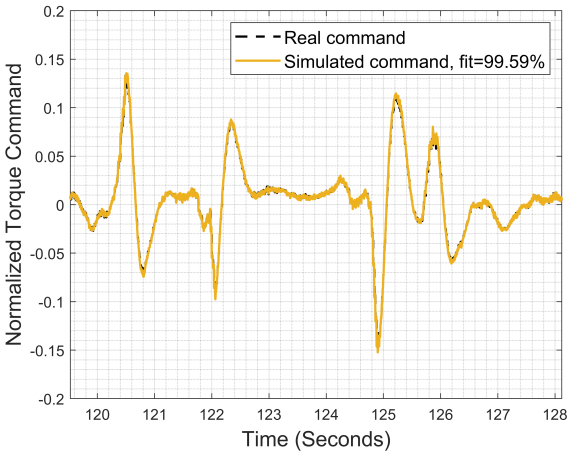
(b) Pitch



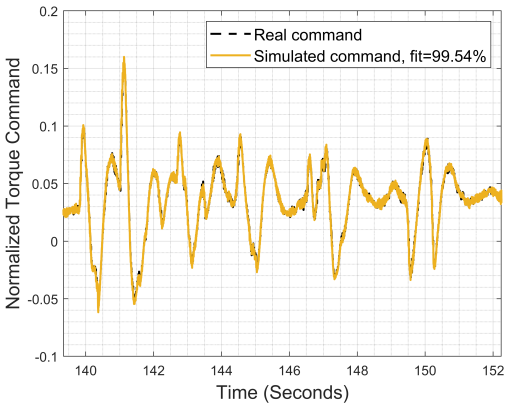
(c) Yaw

Figure 4.12: Validation of controller isolation and integration process

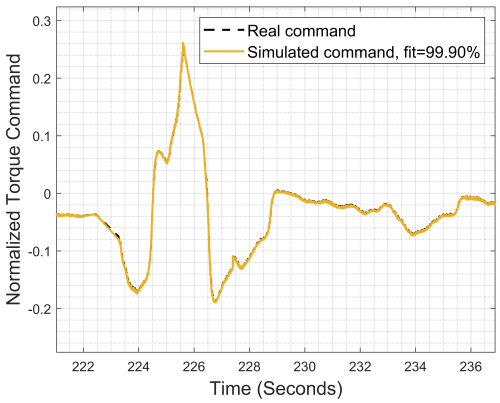
The controller integrated into the simulation environment can be further verified by its self by feeding as input real flight data and comparing the output of the simulated controller versus the real one. Figure 4.13 and the R-squared value in the legend also prove the match of the two controllers during the validation flights.



(a) Roll



(b) Pitch



(c) Yaw

Figure 4.13: Simulated controller output compared with real controller output from flight data

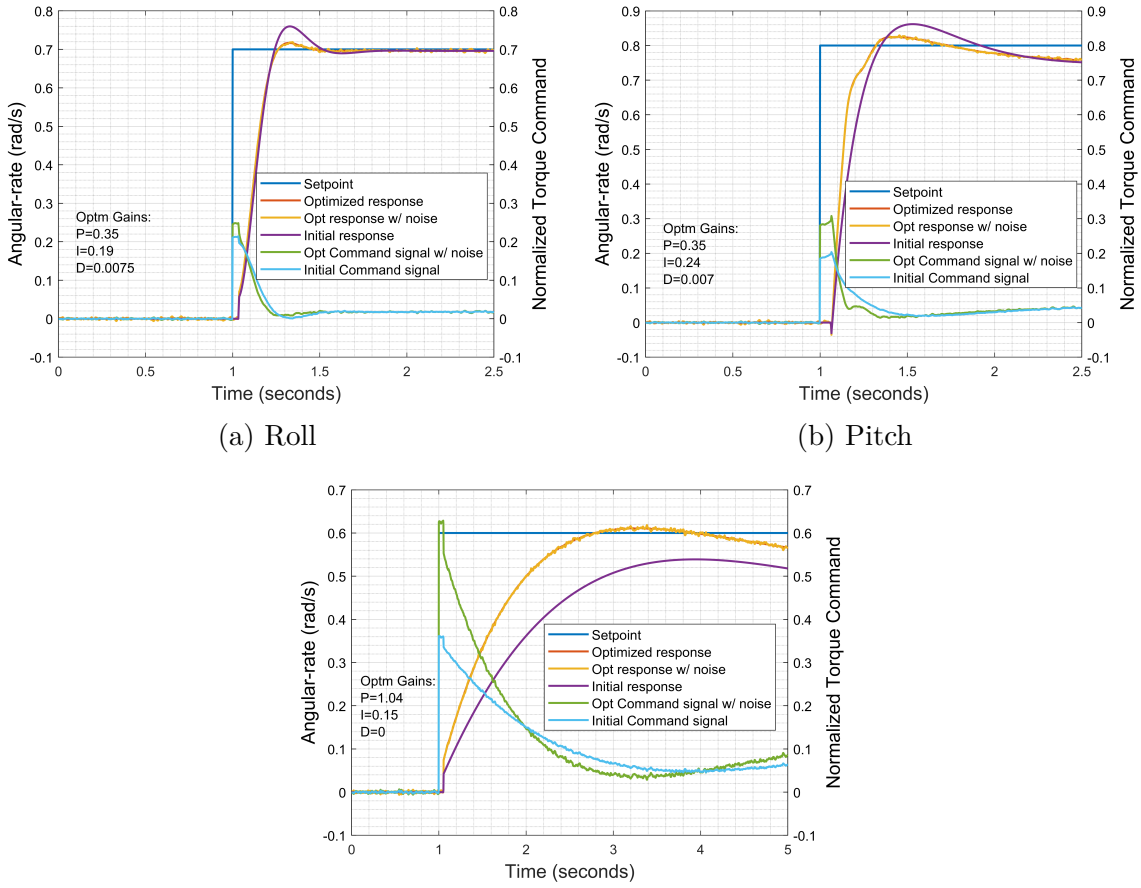
4.3.3 Optimization Results

Table 4.4 presents the optimized controller parameters obtained from the optimization process alongside the initial hand-tuned values. The differential gyro low-pass filter ($DGLPF$) was optimized during the first iteration and then fixed (using the average value) for all axes in a second optimization round. This was necessary due to the autopilot architecture, which requires the $DGLPF$ to have a single value for all axes.

	Roll		Pitch		Yaw	
	Hand-tuned	Optimized	Hand-tuned	Optimized	Hand-tuned	Optimized
P	0.3	0.35 (+16.6%)	0.23	0.35 (+52.2%)	0.6	1.04 (+73.3%)
I	0.2	0.19 (-5%)	0.25	0.24 (-4%)	0.02	0.15 (+650%)
D	0.001	0.0075 (+650%)	0.003	0.007 (+133%)	0	0
DGLPF	30	12 (-60%)	30	12 (-60%)	30	12 (-60%)

Table 4.4: Optimized versus Hand-tuned controller parameters

The simulated step responses with the optimized parameters compared to the initial hand-tuned parameters, shown in Figure 4.14, demonstrate improvements in vehicle response across all axes. Additionally, these graphs reveal differences in the torque command signal between the two sets of controller tunings.



(a) Roll

(b) Pitch

(c) Yaw

Figure 4.14: Step response of the optimized controller compared to the hand-tuned one

Table 4.5 quantifies these changes using performance metrics, highlighting reductions in settling time and overshoot for the roll axis, improvements in rise time and overshoot for the pitch axis, and significant enhancements in all metrics for the yaw axis.

	Roll		Pitch		Yaw	
	Hand-tuned	Optimized	Hand-tuned	Optimized	Hand-tuned	Optimized
Steady-state Error [rad/s]	0.0047	0.0047	0.0550	0.0545	0.0822	0.0334
Rise Time [s]	0.162	0.167	0.218	0.144	∞	1.150
Settling Time [s]	0.42	0.22	1.09	0.89	∞	3.05
Overshoot [%]	7.35	2.35	7.08	3.16	-10.08	1.90

Table 4.5: Optimized versus Hand-tuned controller performance metrics

Most improvements occurred in the yaw axis due to the vehicle's severe underactuation.

The balance between exploration and exploitation within the optimization algorithm can be illustrated by plotting the population's step responses. For instance, Figure 4.15 shows the step response for the last generation of the roll-rate optimization. The presence of both divergent and convergent responses indicates a well-balanced approach. Moreover, these plots can be used to evaluate the tuning of the cost function for the desired performance. The presence of responses that better track performance requirements might indicate a need to redesign or re-tune the cost function.

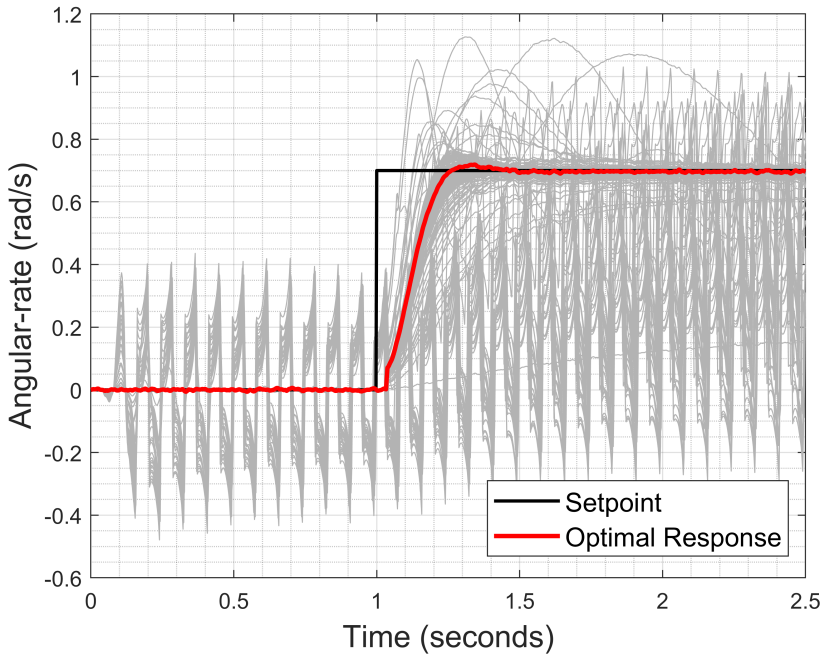
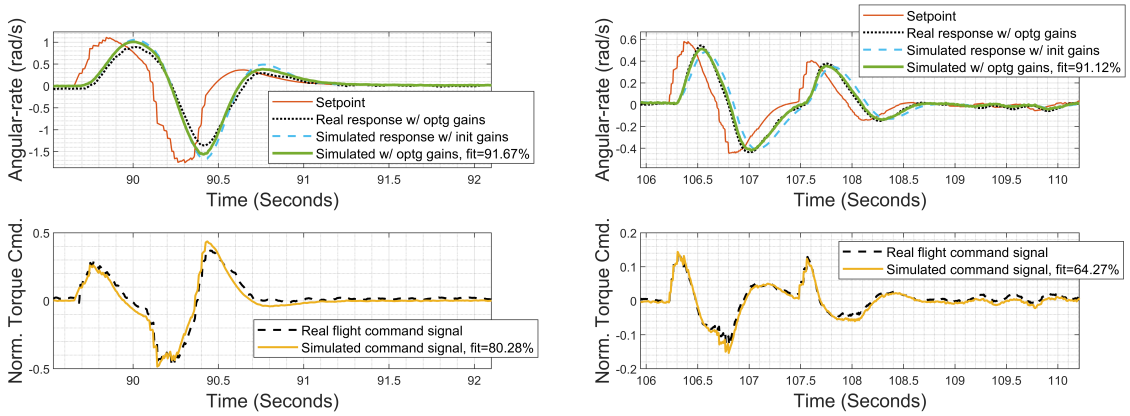


Figure 4.15: Step response of the last generation for roll-rate

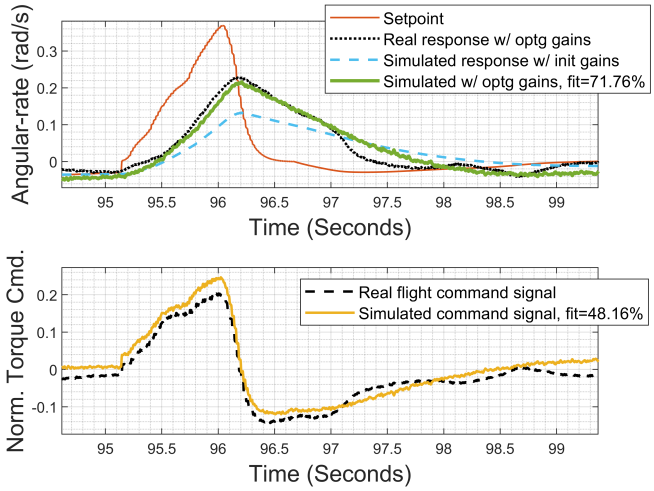
The convergence of the genetic algorithm employed in this study was quantified based on the stability of the best fitness function value over successive generations. Specifically, convergence was deemed achieved when the best fitness value remained within a defined tolerance (1e-6 in this case) for 50 consecutive generations. This criterion ensures that no significant improvement occurred despite the ongoing exploration mechanisms during these generations. In the example presented here, with a population size of 77 individuals, the GA typically converged around generation 25, requiring a total of 75 generations to satisfy the convergence criterion. This corresponds to a total of 5775 simulations for the optimization process. These results demonstrate the effectiveness of the optimization algorithm in reaching the optimal solution for the controller parameters.

After loading the optimized gains into the autopilot and conducting another set of flight tests, it is possible to validate the predicted optimization results against real flight data. The results are presented in Figure 4.16. In these plots, the setpoint, the real flight response with optimized gains, the predicted response with optimized gains, and the simulated response with initial gains are compared. Despite varying fit values (91% for pitch and roll, 71% for yaw), the performance of the real vehicle response improved as predicted by the optimization analysis.



(a) Roll

(b) Pitch



(c) Yaw

Figure 4.16: Validation of optimized tuning against real flight data

4.4 Discussion

This paper introduces an automatic autopilot tuning framework that utilizes Genetic Algorithms (GA) and System Identification (SID) to optimize the parameters of Unmanned Aerial Vehicle (UAV) flight controllers. The framework follows a structured, multi-stage process, involving flight data processing, control law implementation, flight dynamics model (FDM) development through SID, and subsequent controller optimization.

The SID process successfully identified transfer functions for the roll, pitch, and yaw axes, providing valuable insights into the dynamic response of the vehicle. Dis-

crepancies in the model, mainly due to unmodeled structural flexibility and coupling effects, highlight the inherent challenges of simplified modeling approaches. Nevertheless, SID process was considerably less resource-intensive compared to the development of a full physics-based flight dynamics model and proved sufficient for the proposed tuning optimization. However, for unconventional vehicle configurations, additional data sources, such as wind tunnel testing or computational fluid dynamics analyses, may be required to establish a more accurate baseline model before initial flights.

The application of GA resulted in notable improvements in autopilot performance across all control axes. Although GA may not be the most computationally efficient method for this particular application, it offers scalability for controllers with a larger number of tuning parameters. Important model features, such as noise and time delays, played a crucial role in shaping the optimization process. The proposed framework, therefore, offers an effective tool for rapid parameter adjustment in response to vehicle modifications.

While the example in this paper focused on the rate controller of a VTOL aircraft, the proposed framework demonstrated effectiveness across various control laws, autopilot systems, and vehicle types. Only minimal adjustments, such as changes to setpoint step values and simulation duration, were necessary to maintain cost function balance for similarly sized vehicles, highlighting the framework's versatility and broad applicability.

In conclusion, the automatic autopilot tuning framework presented here provides a valuable tool for optimizing UAV control systems, contributing to the advancement of autonomous flight capabilities. Future work could focus on refining the optimization process, including the implementation of parallelization techniques and the exploration of hybrid methods. Additionally, incorporating stability analysis during both the validation and optimization phases would help ensure stability requirements are met, further enhancing the robustness of the framework.

Chapter 5

In-Flight Nonlinear System Identification for Adaptive Control

In the previous chapters, we explored the development of the aircraft, its flight dynamic model, and an automated offline autopilot tuning algorithm designed to accommodate the continuous changes during the aircraft's development and deployment. Chapter 5 marks a significant shift in approach, addressing the challenge of adapting to changing dynamics in real time.

This chapter presents the initial steps and findings in developing online system identification techniques using a novel data-driven algorithm for nonlinear dynamics. The work detailed here was conducted entirely in simulation, utilizing the flight dynamic model of the Mini-E 5050B Eusphyra aircraft. This approach was essential to evaluate the feasibility of online system identification given the high noise levels typically encountered during flight tests. Additionally, it examined the training time required for effective implementation, providing foundational insights into the approach's practicality.

The research in this chapter also laid the groundwork for two critical components of the Model Identification Adaptive Control (MIAC) technique discussed in later: control perturbation and rapid unsupervised system identification. These advancements represent key milestones in enabling real-time adaptability in dynamic environments.

The content of this chapter is derived from the following published manuscripts:

- Bazzocchi, S., Suleman, A. (2023). *In-Flight Nonlinear System Identification for UAS Adaptive Control.* In: Karakoc, T.H., Yilmaz, N., Dalkiran, A., Ercan,

A.H. (eds) *New Achievements in Unmanned Systems.* ISUDEF 2021. Sustainable Aviation. Springer, Cham. [https://doi.org/10.1007/978-3-031-29933-9_19]

- Bazzocchi, S., Suleman, A. (2024). *Non-linear System Identification for UAS Adaptive Control.* In: Karakoc, T.H., Özbek, E. (eds) *Unmanned Aerial Vehicle Design and Technology.* Sustainable Aviation. Springer, Cham. [https://doi.org/10.1007/978-3-031-45321-2_10]

Abstract

Fast aircraft prototyping, fault detection, morphing surfaces and real-time generation of dynamic models are just some of the advantages of a model identification adaptive controller. The research presented in this paper investigates a proposed control architecture and validates a novel data-driven algorithm to be used for online system identification of a UAS. The reported simulation results explore the effects and the limits of short training time and sensor noise on the identified model.

Nomenclature

UAS	: Unmanned Aerial System
CFD	: Computational Fluid Dynamics
SID	: System Identification
MIAC	: Model Identification Adaptive Controller
SINDy	: Sparse Identification of Nonlinear Dynamics
u	: Linear rate on x body axis, m/s
w	: Linear rate on z body axis, m/s
q	: Pitch angular rate, rad/s
u	: Command
X	: States
T	: Training time, s
k	: Harmonic multiplier
t	: Time, s
R^2	: R-Squared fitness value
θ	: Pitch angle, rad

Θ : Library of candidate nonlinear functions
Ξ : Sparse coefficient matrix
ξ : Sparse coefficient vector
η : Noise magnitude
c : Canard
t : Throttle
$\hat{}$: Estimated noisy state

5.1 Introduction

The conventional design and tuning method of an autopilot system for UAS is a highly time-consuming process. This procedure, shown in figure 5.1, usually starts with ground-based analysis, such as CFD or wind tunnel sessions, to generate the data needed to develop a flight dynamic model (FDM) of a vehicle. After designing or selecting appropriate control laws, the model and controller can then be tested in software-in-the-loop simulations with the goal of tuning control parameters for the desired performance. The controller is then deployed into hardware and the vehicle flight tested while recording all observable states from the sensors. This data is then usually processed to correct the initial FDM allowing for greater precision in a second tuning iteration. The described process needs to be repeated for all new prototypes and when a system goes through any modification that alters the dynamics of the vehicle.

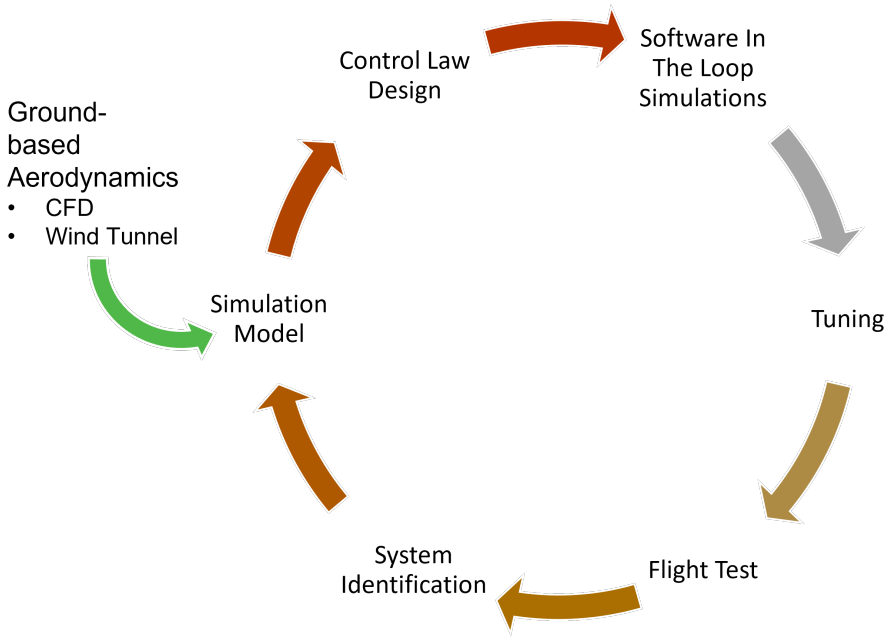


Figure 5.1: Conventional control design and tuning process.

One of the benefits of adaptive controllers is the elimination of this lengthy procedure and automation of the tuning process. But rapid prototyping and the ability to easily overcome changes in system dynamics are not the only motivations that are driving this research field. As stated by Morelli in [146] there are several other advantages including:

- Detection and mitigation of failures
- Ability to cope with morphing aerodynamic properties
- Dynamic flight envelope correction
- Exploration of non-linear dynamics for complex aircraft configuration
- Fast generation of a time varying system model

Some of the well-known agencies and companies in the aerospace industries are pursuing these objectives, including the US Air Force with project RESTORE, Boeing with project RACE and NASA with the L2F project [50].
This study firstly presents an effective architecture to achieve the stated adaptive control benefits and secondly validates a system identification method in the context of a UAS platform developed at the University of Victoria Centre for Aerospace Research (UVic CfAR).

5.2 Method

There are many techniques to achieve adaptive control capabilities, including extremum-seeking control, adaptive pole placement, model reference adaptive control, reinforcement learning, and self-designing control. Among these, model identification adaptive control (MIAC) is unique in its ability to generate a human-readable, time-varying system model online.

5.2.1 Model Identification Adaptive Control

Figure 5.2 shows the high-level structure of a model identification adaptive controller. It is possible to identify several blocks part of a conventional autopilot system such as the estimator, guidance, and controller. The main difference is the presence of the System Identification Process which collects data from the controller output and state estimator with the goal of generating a near real-time model of the vehicle. This information is then fed to the model-based controller which becomes "adaptive" by virtue of the updating dynamic model. It is also possible to extrapolate model performance parameters from the identified system which can be used to tune the guidance and navigation controller.

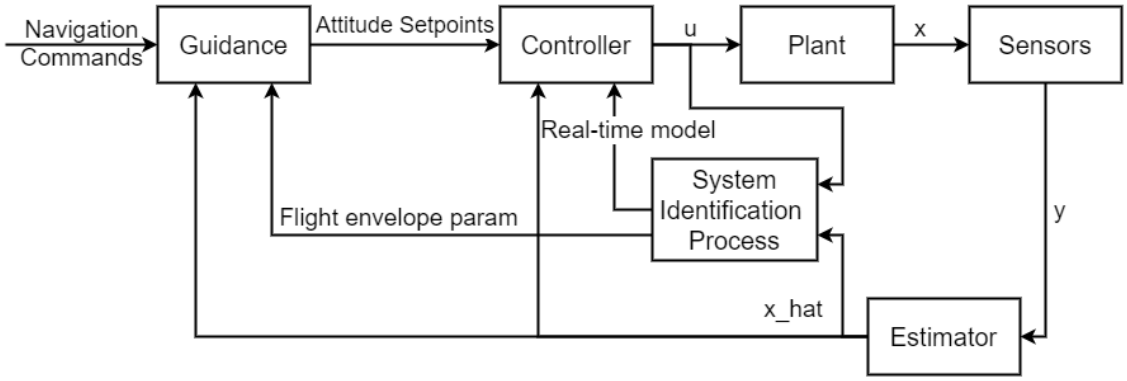


Figure 5.2: Model identification adaptive controller architecture.

One of the main advantages of this type of architecture compared to other data-driven controllers is the human readability. This property allows for formal validation of the controller. In other words, it is possible to validate the control behaviour without testing every single condition, something that is not possible with a Neural Network type of architecture for example [68]. Furthermore, the generation of a real-time SID model allows for rapid investigation of the system performance and analysis

of any changes to the dynamics during flight.

In a real-world application of this controller other three components would need to be present to complete this architecture: an online simulation of the identified model, a model supervisor, and a control perturbator. The online simulation is needed to generate the state prediction form the predicted model. This can be intelligently obtained from the prediction of a MPC reducing the computational power required. The model supervisor compares the predicted state of the identified model with the real states being computed by the estimator. If a discrepancy of a certain amplitude is recorded for a set number of samples this would trigger the control perturbator and the SID program to start a system identification process. The control perturbator generates and injects the desired perturbation signal on top of the actuator commands coming from the controller.

5.2.2 System Identification Method

The selection and development of the system identification method is crucial since the MIAC is highly dependent on the capability of the SID process to effectively identify the vehicle dynamics.

NARMAX, Eigensystem Realization Algorithm (ERA), Genetic Programming (GP), Neural Newtwork (NN) and Koopman with control, are some of the multiple data-driven techniques for SID and control discussed in [33], a recommended reference for an overview on machine learning for control. The method selected and employed in this project is the Sparse Identification of Nonlinear Dynamics or SINDy for short [157]. This theory combines sparsity-promoting techniques and machine learning with nonlinear dynamical systems to discover governing equations from noisy measurement data. SINDy was published in 2016 and rapidly evolved in SINDYc to include external inputs and feedback control to the system identification process [35]. In 2018 Kaiser demonstrated the capability of SINDy algorithm to perform system identification with limited and noisy data, proving the capability of this technique to be applied on-line for a control optimization problem [37]. Quade investigated the capability of SINDy by generating a structure for fault detection based on model prediction to trigger the system identification process [38]. The results from the tests performed on nonlinear dynamic system with discrete and noisy data highlighted the interesting

potential of this new theory to be applied in the ongoing aerospace research of an adaptive controller.

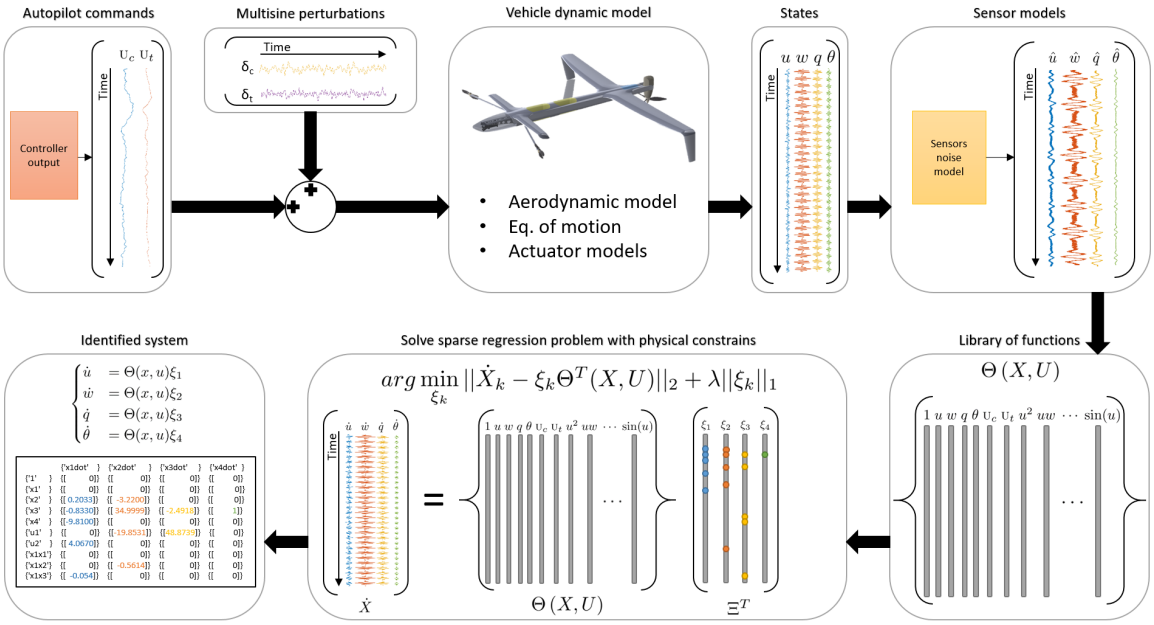


Figure 5.3: System Identification Process.

The scheme shown in figure 5.3 illustrates the high-level system identification process for the longitudinal dynamics (u, w, q, θ) of the Eusphyra aircraft using SINDy. Note that the same study can be performed on the latero-directional dynamics or both combined.

During forward flight, the longitudinal actuator commands (u_c and u_t) generated by the controller can be perturbed by unique, zero-mean, multi-frequency signals (δ_c and δ_t). This practice ensures that the vehicle dynamics are effectively excited to allow for an effective system identification. In this study, the nonlinear flight dynamic model was generated using CFD analysis and coded in Simulink to calculate the system response (X). The calculated true states are then fed into the sensors and estimator models, which apply noise to the response depending on the signals' standard deviation. The estimated or noisy states are used to generate the matrix Θ which holds the pool of candidate nonlinear equations that may describe the system. It is important to properly select a list of functions that is capable of describing the dynamic system. Ideally, this pool of functions contains all the terms necessary to represent the system but at the same time is small enough to guarantee a convergence to the correct solution. In this case, the library was formed by constants, polynomial

terms up to 3rd order, and trigonometric terms. Because only a few terms of the proposed library of functions will be active in each differential equation, a sparse regression problem is formulated to determine the vector of coefficients ξ forming the matrix Ξ:

$$\xi_i = \arg\min_{\hat{\xi}_i} \left(\frac{1}{2} \|X_i' - \hat{\xi}_i \Theta^T(X, U)\|_2^2 + \lambda \|\hat{\xi}_i\|_1 \right) \tag{5.1}$$

Each column ξ_i is a sparse vector of coefficients determining which terms of Θ are active in the corresponding differential equation:

$$\dot{x}_i = f_i(x) = \Theta(x^T)\xi_i \tag{5.2}$$

The sparsity of the resulting equations can be tuned by the parameter λ, which defines the threshold for the elimination of the small entries [158].

The execution speed of this algorithm depends on the following factors: number of states, number of functions in the library, length of training data, and number of iterations required to obtain a solution to the sparse regression problem that does not present any small entries. In the simulations conducted in this research, the time required to process the solution was negligible compared to the time needed to acquire the training data. This is because the solution is calculated only once per buffer of finite-history data set (near real-time) instead of recalculating the solution for every new data point by continually shifting the buffer (real-time).

The convergence of this method is not guaranteed. If the parameter λ is not set correctly or the pool of equations is too vast, the algorithm may terminate with an incorrect non-sparse solution. It is recommended to set as many constraints as possible to contain the dimensions of Θ and reduce the number of differential equations to discover. In the presented study, it was possible to constrain one of the differential equations (Eq. (5.3)) and reduce the size of the pool of candidate functions by computing the angle of attack α and therefore eliminating the need for trigonometric terms (Eq. (5.4)):

$$\dot{\theta} = q \tag{5.3}$$

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right) \tag{5.4}$$

5.2.3 Command Perturbation

One of the most important components in the system identification is the correct perturbation of the available actuators. This not only serves the purpose of identifying control effectiveness but also excites the system dynamics across a defined spectrum of frequencies. The amplitude of the perturbations must also be properly set to ensure that the signal-to-noise ratio remains above the usable threshold without endangering the state of the vehicle.

There are several types of perturbation signals that may be used for this purpose, such as singlets, 3-2-1-1, doublets, sinusoids, and chirps. The signal type used in this project is orthogonal, phase-optimized, multisine inputs [69]. The primary benefit of this type of signal is that the orthogonal nature allows simultaneous perturbations of different actuators without correlating responses. These inputs are also effective in generating the steady-state response data needed to estimate frequency responses.

Defining the duration of the excitation T is the first step towards the design of a multisine perturbation. This characterizes the fundamental frequency $1/T$ and, by extension, the harmonic multiples k/T. Each actuator u_j has a unique subset of multipliers K_j which define the frequency range spanned during excitation. The multisine signal is then generated as a sum of sinusoids as shown in equation (5.5):

$$u_j(t) = \sum_{k \in K_j} a_k \sin\left(\frac{2\pi k}{T} t + \phi_k\right) \tag{5.5}$$

The amplitudes a_k are calculated according to the desired power spectra for each input, and the phase angles ϕ_k are optimized to reduce the peak value of the multisine signal to keep the response near the desired flight condition.

An example of the multisine signals generated to perturb the thrust and canard controls is shown in figure 5.4. It is possible to observe the multiple overlaid frequencies and the fact that both signals average to zero, which maintains the vehicle states close to the desired setpoints.

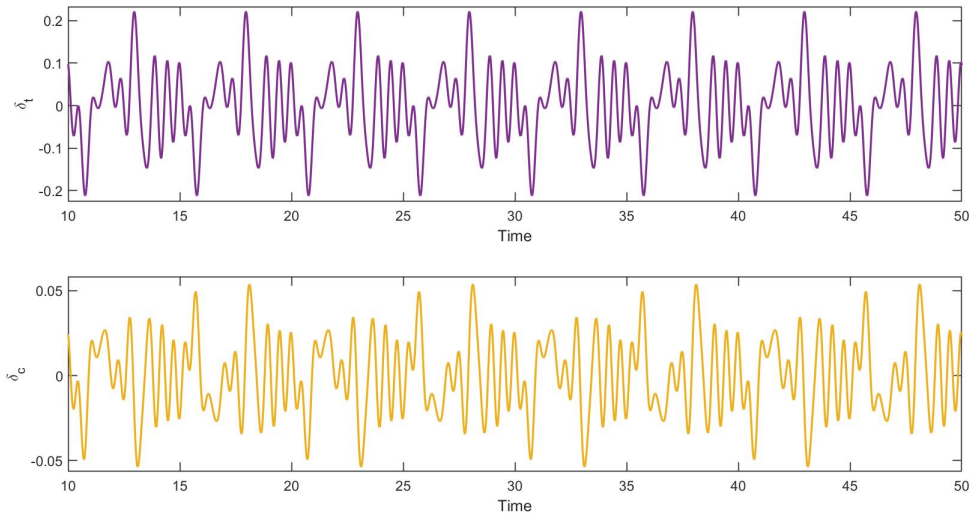


Figure 5.4: Multisine perturbation.

5.3 Results and Discussion

After finding the correct order of magnitude of the tuning parameter λ, it was possible to repeatedly identify the differential equations coded in the FDM with small deviations from the original coefficients. This error can be eliminated by reducing the noise injected through the sensor models under a certain threshold.

To better understand the results obtained from the SID process, the response of the true dynamic system was compared with the simulated response of the discovered model generated from the same control inputs and initial conditions. The graph in figure 5.5 displays the simulation results of a model trained with only 5 seconds of dynamic data and a noise-to-signal ratio of 0.01. The vertical dashed line separates the training data (on the left) from the validation data (on the right). Once the training time elapsed, the SID algorithm processes the data and generates a model. The model can then be initialized with the current flight conditions and fed with the same control inputs as the true system.

The goodness-of-fit of the identified model can then be numerically evaluated with the R-Square method.

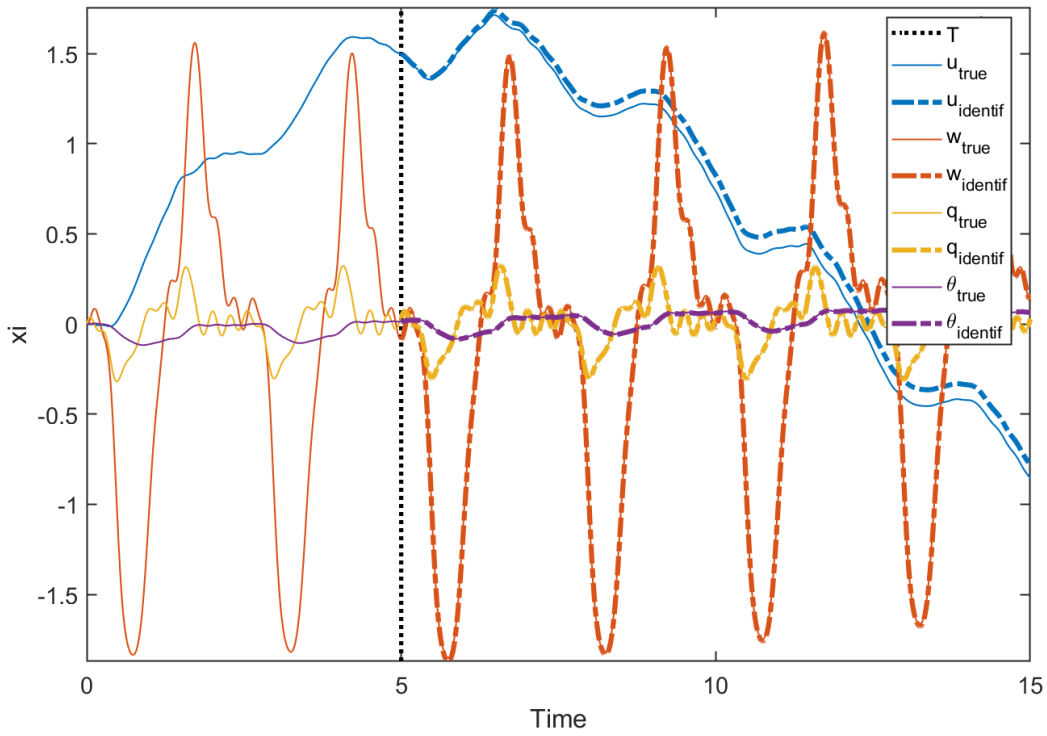


Figure 5.5: Validation of the identified system.

Even with this short training time, the SID process was capable of generating a model with a minimum of 88% goodness of fit across the four states (u, w, q, θ). It is worth noting that the main deviation of the identified model from the true dynamics in this example is observed in the response of the linear body rate u. The main contributing factor to this error is the low-frequency limit of the perturbation input, which is set to be equal to or greater than 2 times the training time T to allow at least two full cycles for each frequency. Increasing the training time T lowers the minimum frequency of the command perturbation, improving the identification of the slower dynamics associated with the linear body speed u.

Once the identification problem is set, the main variables that can influence the results are training time T and noise levels. The surface in figure 5.6 reports the minimum fitness level of the identified model as a function of the noise level η and training time T. As expected, increasing training time improves the fitness of the identified model. In this study, the vehicles' system dynamics can be correctly identified with training times greater than 9 seconds. On the other hand, increasing the noise-to-signal ratio above 0.5 while holding training time constant significantly reduces the fitness level of the model.

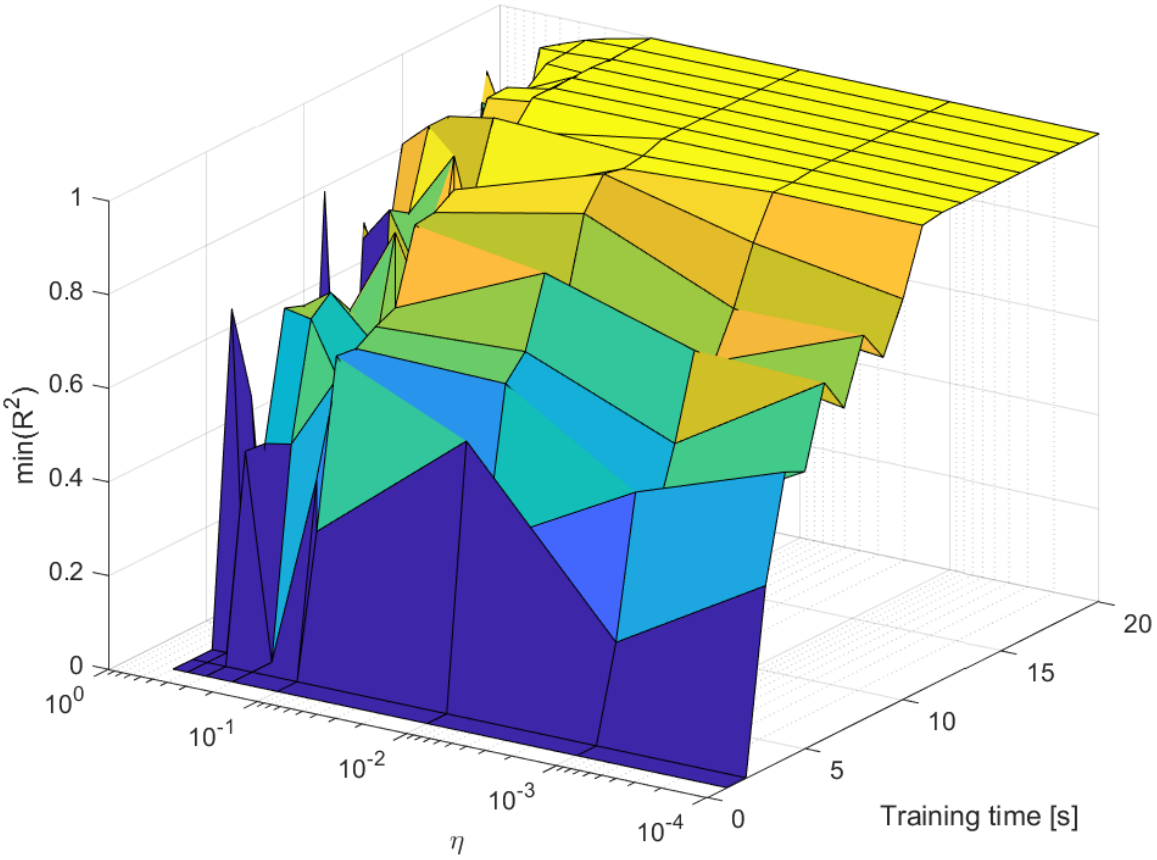


Figure 5.6: Fitness level as a function of noise and training time.

Table 5.1 displays the maximum noise-to-signal ratio, in relation to specific training times, for the SID algorithm to obtain a model with an R-Square greater than 80%.

η	T [s]	R^2 [%]
0.01	5	88
0.22	10	99
0.35	20	98
0.42	30	98
0.53	50	94
0.61	80	86
0.63	120	83
0.63	160	81

Table 5.1: Required training time for extreme noise-to-signal ratios.

Even with large increases in training data, the maximum tolerable noise to obtain a fitness level of 80% averages around $\eta = 0.64$. This asymptotic trend can also

be observed in figure 5.7. The yellow-colored area defines the region in which it is possible to correctly identify the studied system. The curve that outlines this region flattens for $T > 60$ seconds; therefore, for higher levels of noise, increasing the training time does not show any substantial improvement in the model fitness.

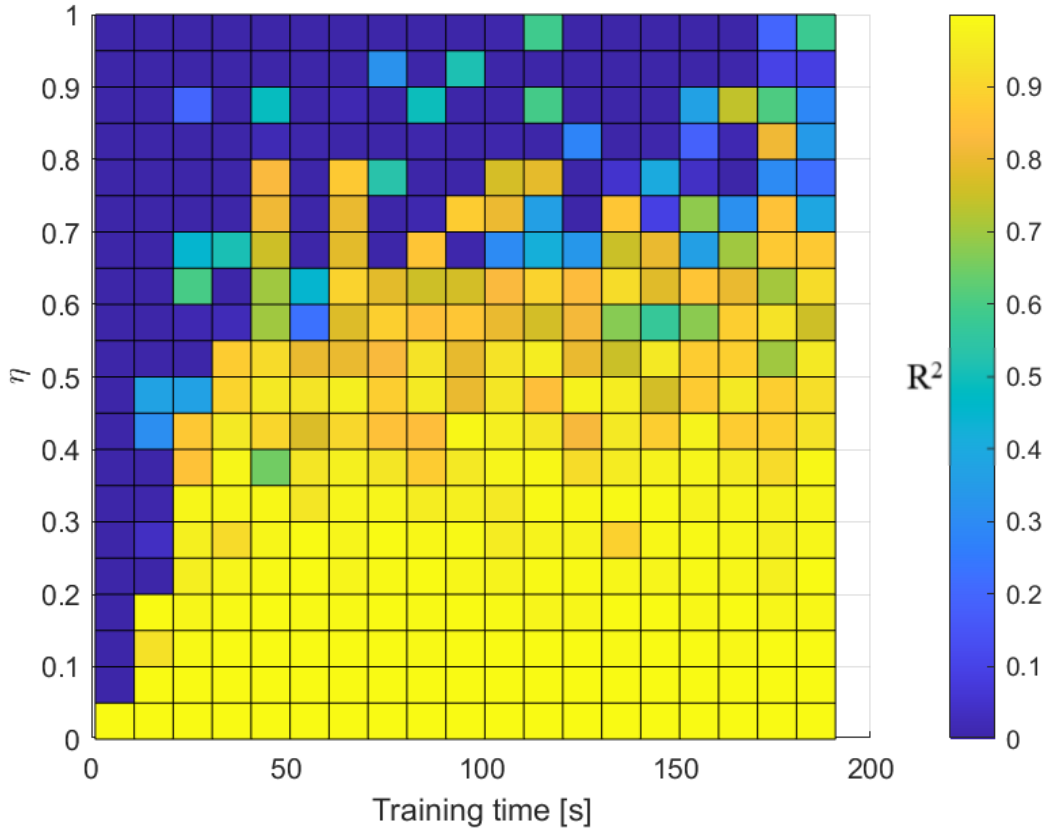


Figure 5.7: Fitness level distribution for high noise and increased training time.

It is important to mention that if the training time is increased, the harmonic multipliers K_j should also increase in value to ensure that the higher frequencies are still being excited. Failing to do so results in a degradation of the fitness level.

5.4 Conclusion

The innovation in vehicle configurations and the increase of onboard computational power are contributing to the growing interest towards adaptive controllers and data-drive models.

Model identification adaptive controller (MIAC) stands as the preeminent architecture to meet the objectives set in this project, allowing for exploration of a machine learning technique while maintaining a certain level of verification for the safety of flight. In this paper the high-level MIAC structure was presented, highlighting the benefits of having a human readable controller. This method heavily relies on the SID process to correctly identify the vehicle dynamics, therefore a significant part of this research focused on the study of an effective technique for online system identification processing. SINDy method was proven to successfully identify the nonlinear dynamics of the flight test vehicle. The results demonstrate the fitness of the identified model in several condition of noise and training time. Lastly, the analysis of the simulation data established the limit conditions within which the SID process can produce an accurate model.

To conclude, it is worth pointing out the conflicting nature of this type of control theory. The controller's main objective is to suppress perturbations and track the set point in the most actuator-efficient way, while the SID process requires some degree of vehicle perturbation and control actuation in order to generate a dynamic model. The outcomes of this study indicate that it is possible to generate a suitable model from the system identification process described with minimal perturbation. The next steps of this research will be firstly to run the SID process online using real flight data fed from an autopilot then secondly pair the validated SID algorithm with a model-based controller to close the loop and control the vehicle.

Chapter 6

Model Identification Adaptive Control

In the previous chapter, we introduced the concept of a Model Identification Adaptive Controller (MIAC) and focused on developing an online system identification (SID) method suited for this application. The SID method was designed, tested in simulation, and analyzed for its response to noise and training time using the high-fidelity flight dynamic model (FDM) of the Mini-E 5050B.

Building upon these findings, this chapter presents the full development of the MIAC system. The method was refined to operate unsupervised onboard a UAV, transitioning from Matlab to Python to facilitate development and integration within a unified environment. Several enhancements were incorporated into the SID, including multiple regression optimization methods, system delay identification, actuator modeling, and a comprehensive suite of preprocessing tools to prepare sensor data for system identification, including state estimation.

A key advancement in this work is the integration of Model Predictive Control (MPC) as a model-based controller, completing the MIAC framework. Additionally, a model supervisor was introduced to manage the SID-generated models, evaluate their quality, and assign the most suitable one for control. The perturbation injection process was also improved, allowing for more flexible perturbation types and easier design modifications.

The objective of this research was to develop a fully functional MIAC system ready for deployment on real hardware for flight testing. While the Mini-E 5050B high-fidelity FDM was utilized for initial development, a more cost-effective and eas-

ily deployable UAV was chosen for early flight tests, given the risks associated with testing an experimental controller. Ultimately, the MIAC framework enhances rapid prototyping capabilities, supporting the research and development of novel UAV configurations.

The content of this chapter is based on the following manuscript, currently in preparation for journal submission:

- Bazzocchi, S., Grant, H., Suleman, A. (2025). *Model Identification Adaptive Control for UAVs: Development, Implementation and Flight Testing.* [Manuscript in preparation]

Abstract

This paper presents a Model Identification Adaptive Controller (MIAC) architecture for Unmanned Aerial Vehicles (UAVs), integrating online system identification and adaptive model-based control to handle unknown nonlinear dynamics. The proposed approach combines control perturbation, Sparse Identification of Nonlinear Dynamics (SINDy) for real-time model identification, Model Predictive Control (MPC) for setpoint tracking, and adaptive model selection and validation to ensure the most accurate model is used for control. The MIAC framework is first developed in simulation and then validated on a real UAV using a dual-autopilot setup, where a reliable off-the-shelf autopilot serves as the primary controller while the experimental MIAC operates in parallel. Flight tests demonstrate that MIAC successfully identifies the vehicle's governing equations, maintains stable control, and improves setpoint tracking performance. These results highlight the feasibility of real-time system identification for adaptive flight control, offering a robust architecture for rapid UAV prototyping to support research and development.

6.1 Introduction

Unmanned Aerial Vehicles (UAVs) have emerged as pivotal tools in both research and commercial aviation, enabling rapid prototyping and experimental validation of innovative design concepts. However, UAVs often operate in dynamic environments where variations in payload, aerodynamic configurations, and external disturbances significantly alter vehicle behavior. Conventional control methods, such as gain scheduling

and dynamic inversion, rely on accurate models that are derived from extensive aerodynamic analysis, system identification campaigns, and flight tests. This modeling process is not only time-consuming but also lacks the flexibility required for rapidly evolving systems.

Adaptive control strategies that incorporate online system identification offer a compelling alternative. By updating control laws based on real-time flight data, these methods can accommodate shifts in UAV dynamics and generate time-varying models useful for simulation, performance evaluation, and fault diagnosis. As Morelli [9] points out, integrating an online adaptive dynamic model with full-envelope flight control can accelerate prototyping while enhancing capabilities in fault detection, self-learning systems, morphing wing control, and flight envelope protection.

Over the past decades, a variety of adaptive control techniques have been proposed to address the inherent uncertainties in UAV operations. For example, Sayyaadi and Sharifi [159] introduced a nonlinear adaptive impedance controller, while Chowdhary et al. [160] developed model reference adaptive controllers applicable to both fixed-wing and rotary-wing platforms. Hybrid methods that incorporate fuzzy logic [161] and artificial neural networks (ANN) [162] have improved adaptive control robustness. More recently, reinforcement learning approaches have gained traction as adaptive control methods; however, these typically yield "black box" solutions that do not provide an explicit vehicle model during adaptation, complicating validation and troubleshooting, especially in safety-critical aerospace applications [68].

Direct adaptive schemes, such as Model Reference Adaptive Control (MRAC) and $\mathcal{L}_1$ adaptive control, have demonstrated improved robustness to payload changes and environmental disturbances by forcing the plant to track a desired reference model [53]. Yet, these methods depend critically on the designer's choice of reference model and adaptation law, and their performance can degrade when actual system dynamics deviate from the underlying assumptions or when unmodeled phenomena are present. In contrast, indirect adaptive control methods (those that perform online system identification to update the control law) offer increased flexibility and provide an explicit representation of the system dynamics.

The Model Identification Adaptive Control (MIAC) framework is an example of such an indirect approach. MIAC decouples the system identification from the controller design, allowing independent validation of the model identification component. In [163], an adaptive predictor-based controller based on MIAC was successfully demonstrated on a high-fidelity aircraft simulation, showing that these schemes can

handle large variations across the flight envelope. Despite these advances, applying MIAC to smaller UAVs and ensuring real-time performance remains challenging due to practical constraints such as short data windows and sensor noise.

Recent advances in data-driven identification techniques offer a promising solution. Among these, the *Sparse Identification of Nonlinear Dynamics* (SINDy) algorithm has demonstrated impressive capabilities in extracting compact, interpretable models of nonlinear systems from limited data [115, 164]. By employing sparse regression to identify the most significant terms governing the system's behavior, SINDy yields simplified yet accurate representations of complex UAV aerodynamics,an essential feature for onboard implementation where computational resources are limited.

One notable example of adaptive control using model identification is the *Learn-to-Fly* (L2F) project led by NASA's Langley and Ames research centers [50]. Their approach, based on nonlinear dynamic inversion supplemented by an adaptive disturbance rejection module [60], used orthogonal phase-optimized multi-sine inputs to excite the aircraft dynamics [31]. In contrast, the framework presented in this paper adopts a more flexible nonlinear model structure via SINDy and integrates a Model Predictive Controller (MPC) that updates its model during flight. Whereas existing MPC frameworks for UAVs [165, 166] typically require accurate precomputed models, recent extensions have introduced some fault tolerance [102]. The proposed MIAC system leverages the adaptability of MPC while reducing the dependency on fixed models.

In this work, we develop a comprehensive MIAC architecture that combines a flight data pre-processing pipeline, a Model Supervisor module to detect inconsistencies between the current model and incoming flight data, an MPC-based controller that uses the most accurate, currently identified model to generate control commands, and a Command Perturbation Injector designed to excite the system when necessary, thereby improving identification quality. This architecture is first validated in simulation and then integrated into an onboard target computer for real-world flight testing. Unlike many studies that stop at simulation, our work bridges the gap between theory and practice by demonstrating performance differences between simulation and actual flight conditions.

The remainder of the paper is organized as follows. Section 2 details the proposed system architecture for MIAC, providing an overview and an in-depth discussion of each component. Section 3 describes the hardware integration, deployment process, modifications made for flight testing, and the test plan. Section 4 presents the ex-

perimental results, and Section 5 discusses the challenges encountered, limitations of the current approach, and potential directions for future research. Appendices A and B contain the model library and the reference model used in this study.

6.2 System Architecture

Figure 6.1 illustrates the high-level concept of the proposed Model Identification Adaptive Controller (MIAC) architecture.

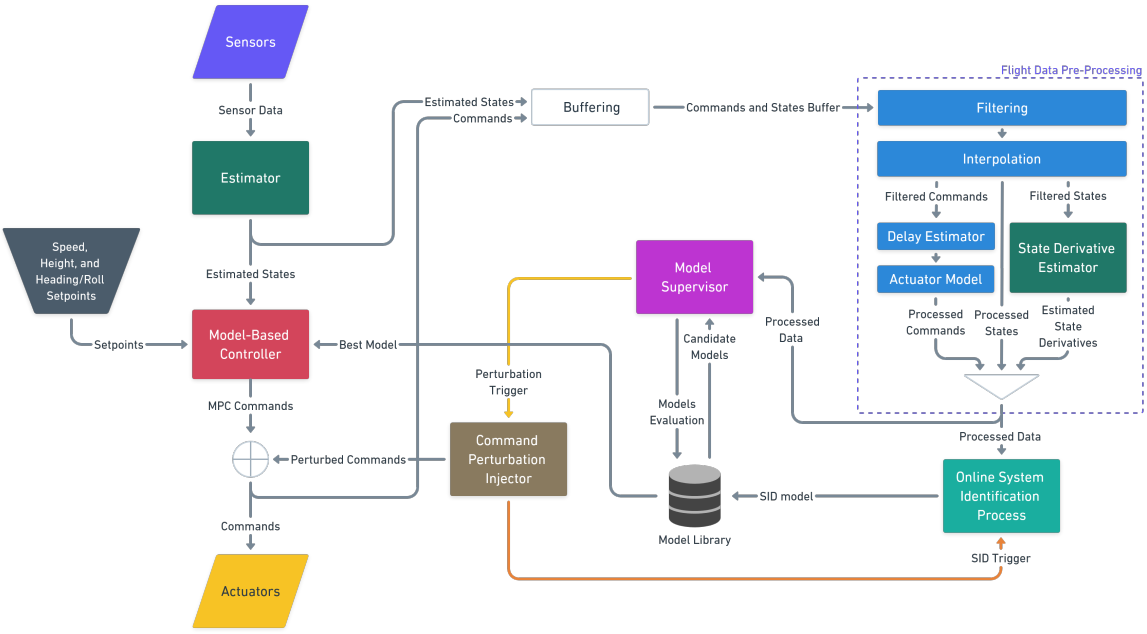


Figure 6.1: Block diagram of the MIAC architecture.

The overall architecture comprises several interconnected modules that collaboratively enable adaptive control of UAVs. Sensor data is initially processed by an estimator (employing a Kalman Filter) to yield accurate estimates of the vehicle states. These estimates are used in two parallel streams:

1. They serve as input to a model-based controller, which computes the actuator commands necessary for the UAV to track given setpoints.

2. They are stored in data buffers, along with the corresponding actuator commands, to form discrete chunks of flight data.

After buffering, the collected flight data is processed by a pre-processing algorithm that organizes the data into neatly time-synchronized matrices. This pre-processed

data is then evaluated by the model supervisor. The supervisor compares the predictions of candidate models, stored in a model library, with the actual flight data. Should the model predictions prove unsatisfactory, the supervisor triggers a command perturbation.

The command perturbation injector is responsible for executing a pre-planned maneuver that excites the UAV's dynamics by actuating all control surfaces. Following the perturbation, the system identification process is activated. This process loads the buffered flight data corresponding to the perturbation segment and identifies a new model for the system. The newly identified model is added to the model library and its performance is verified and scored against existing models.

Each execution of a perturbation prompts the model supervisor to update the fit values for all stored models. The best-performing model is then forwarded to the model-based controller to be used for active vehicle control. In normal operations (when no perturbation is executed) the controller continuously relies on the best-fitting model to generate actuator commands that ensure the estimated states track the desired setpoints.

In the following subsections, the individual components of the MIAC architecture are described in greater detail.

6.2.1 Estimator and Buffering

As with all UAV control systems, accurate state estimation is critical. In this project, an Extended Kalman Filter (EKF) algorithm developed by Riseborough [167] and implemented in the PX4 firmware [168] is employed. This estimator fuses various sensor measurements to provide estimates of the following states:

- **Attitude:** A quaternion representing the rotation from the North-East-Down (NED) local navigation frame to the vehicle body frame.
- **Velocity:** The linear velocity at the Inertial Measurement Unit (IMU) in the NED frame (m/s).
- **Position:** The geographic position at the IMU, given as latitude (rad), longitude (rad), and altitude (m).
- **IMU Biases:** Gyroscope bias estimates (X, Y, Z in rad/s) and accelerometer bias estimates (X, Y, Z in m/s^2).

- **Magnetic Field:** Earth magnetic field components in the NED frame (gauss) along with magnetic field bias estimates in the body frame (X, Y, Z in gauss).
- **Wind Velocity:** Wind speed components in the North and East directions (m/s).
- **Relative Altitude:** The altitude of the vehicle relative to the takeoff point (m).

Each of these state topics is estimated at a frequency appropriate to its nature and is individually timestamped. The estimated states are published as topics on the uORB data bus. These topics are used both for real-time control (by the model-based controller) and for offline analysis. To facilitate the latter, dedicated data buffers accumulate both state topics and the corresponding actuator commands.

The lengths of these buffers are user-configurable through two parameters:

- T_{train}: the duration (in seconds) of the training or perturbation phase.
- T_{dead}: the duration (in seconds) of the dead band before and after the perturbation.

The total number of samples stored in a buffer, L_{buf}, is defined by

$$L_{\text{buf}} = (T_{\text{train}} + 2\,T_{\text{dead}}) \cdot f_{\text{topic}}, \tag{6.1}$$

where f_{topic} is the frequency (in Hz) at which the topic is published.

The buffer is implemented as a circular (FIFO) buffer. That is, when a new message is received, the oldest message is discarded to maintain a constant buffer size. This ensures that the buffers always contain the most recent flight data segments, which are critical for subsequent model identification and performance evaluation.

6.2.2 Flight Data Pre-Processing

This subsection details the data pre-processing pipeline designed to convert raw flight data buffers into synchronized matrices suitable for further analysis, such as state-derivative estimation and system identification. The pipeline consists of several key stages:

1. **Signal Filtering:** Attenuates high-frequency noise in sensor measurements using a digital Butterworth filter.

2. **Data Resampling via Interpolation:** Aligns data from different sensors onto a uniform time grid, ensuring precise time synchronization.

3. **State and Derivative Estimation:** Constructs the state vector and computes its time derivative using both sensor measurements and EKF outputs.

4. **Command Delay Estimation and Shifting:** Detects and compensates for temporal misalignments between actuator commands and measured states.

5. **Actuator Modeling:** Optionally incorporates actuator dynamics by filtering control commands to better reflect the physical behavior of the actuators.

Each of these stages is described in detail in the subsequent subsubsections.

Filtering

Although the Extended Kalman Filter (EKF) already applies a degree of filtering to the raw flight data, certain signals benefit from additional noise attenuation, especially when differentiation is required for system identification. High-frequency noise can mask the true dynamics of the UAV and negatively affect parameter estimation. To address this, a digital Butterworth low-pass filter is applied to the measured signals.

The Butterworth filter is chosen for its maximally flat passband response, which minimizes amplitude distortion up to the cutoff frequency. This property ensures that the signal dynamics are preserved within the passband. The filter parameters, such as the cutoff frequency, are configurable via the system's configuration file, allowing targeted filtering for signals with significant sensor noise (e.g., angular rates, linear accelerations, and airspeed).

A Butterworth filter of order N, with cutoff frequency ω_c, has a magnitude-squared frequency response given by:

$$|H(\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_c}\right)^{2N}}, \tag{6.2}$$

ensuring a flat response in the passband and a steep attenuation beyond ω_c. In discrete time, the filter is implemented using standard infinite impulse response (IIR) techniques. The filter coefficients b_k and a_k are computed such that the filtered output

$y[n]$ satisfies the difference equation:

$$y[n] = \frac{1}{a_0}\left(\sum_{k=0}^{M} b_k\, x[n-k] - \sum_{k=1}^{N} a_k\, y[n-k]\right), \tag{6.3}$$

where $x[n]$ is the discrete input signal, $y[n]$ is the filtered output, and a_0 is typically normalized to unity. The digital cutoff frequency is obtained by normalizing the desired cutoff frequency to the Nyquist frequency (half the sampling rate).

Data Resampling via Interpolation

Flight data collected from various estimator processes often feature non-uniform timestamps and different sampling rates, complicating data fusion. To resolve this, each data stream is resampled onto a common, uniformly spaced time vector with a frequency of 100 Hz.

Uniform Timestamp Generation and Data Cropping: Let $\{t_k\}_{k=1}^{K}$ denote the original timestamps for a given data topic, where $t_k < t_{k+1}$. Since different topics may begin and end at slightly different times, the first step is to crop the data to a common interval. Define:

- $t_{\text{start}} = \max\{t_{\text{start}}^{(j)}\}$, the latest start time among all topics (indexed by j),
- $t_{\text{end}} = \min\{t_{\text{end}}^{(j)}\}$, the earliest end time among all topics.

Only data within the interval $[t_{\text{start}},\, t_{\text{end}}]$ is retained, ensuring that no extrapolation is required during interpolation.

Interpolation onto a Uniform Time Grid: A common sampling period is defined as $\Delta t = 0.01$ seconds (corresponding to 100 Hz), and a uniform time vector is generated:

$$\{\tau_m\}_{m=1}^{M} = \{t_{\text{start}},\, t_{\text{start}} + \Delta t,\, \ldots,\, t_{\text{end}}\}. \tag{6.4}$$

This vector $\{\tau_m\}$ serves as the reference time base for all data topics. For a given topic with measurement vector

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_d(t) \end{bmatrix}, \tag{6.5}$$

where d is the number of measured variables, linear interpolation is applied componentwise. Given the original timestamps $t_1, t_2, \ldots, t_K$, the interpolated value for component $x_i(t)$ at a new timestamp τ (with $t_k \leq \tau \leq t_{k+1}$) is computed as:

$$x_i(\tau) = x_i(t_k) + \frac{\tau - t_k}{t_{k+1} - t_k}\Big(x_i(t_{k+1}) - x_i(t_k)\Big). \tag{6.6}$$

This process produces the resampled values $\tilde{x}_i(\tau_m)$ for each timestamp τ_m. Arranging these values for all components forms the resampled topic matrix:

$$\widetilde{\mathbf{X}} = \begin{bmatrix} \mathbf{x}(\tau_1)^\mathsf{T} \\ \mathbf{x}(\tau_2)^\mathsf{T} \\ \vdots \\ \mathbf{x}(\tau_M)^\mathsf{T} \end{bmatrix}. \tag{6.7}$$

Yaw Angle Unwrapping and Translation: Yaw measurements, being inherently circular, often exhibit discontinuities (e.g., abrupt jumps at $\pm\pi$) that can throw off simple interpolation schemes. To address this, the yaw angle is first unwrapped prior to interpolation. This unwrapping is critical, it leverages the very discontinuities it later removes to correctly reconstruct a continuous angular profile.

After unwrapping, the yaw values are translated so that 0 no longer corresponds to geographic north, but rather to the heading at which the maneuver was initiated. This translation simplifies the interpretation of analysis tool plots by aligning the reference with the maneuver's start direction. Importantly, while the translated yaw is used for visualization, the original unwrapped yaw is preserved separately. This unaltered data is essential for generating the rotational matrix needed to accurately compute the velocity components u, v, and w.

Merging State and Command Data: After interpolation, state-related topics are concatenated column-wise to form a single, time-synchronized *state matrix*. Similarly, command signals are interpolated onto the same time grid, resulting in a *command matrix*. These uniformly synchronized matrices facilitate subsequent processes, including state-derivative estimation and system identification.

This resampling methodology guarantees strict synchronization across data topics while avoiding the inaccuracies associated with extrapolation. Although the cropping step slightly reduces the overall data length, it ensures that every resampled data point lies within the valid range of the original measurements. Additionally, incorporating a deadband in the buffer helps to fully capture the desired perturbation or maneuver within the final resampled data arrays.

State and Derivative Estimation

This is a core process that computes the state vector $\mathbf{x}$ and its time derivative $\dot{\mathbf{x}}$ using data from the Extended Kalman Filter (EKF) and raw sensor measurements. The following steps detail the necessary mathematical operations:

1. **Input and Sensor Selection.**
   From the merged state data matrix, the following signals are extracted:

$\phi,\ \theta,\ \psi$	(Euler angles: roll, pitch, yaw),
$V_{x_e},\ V_{y_e},\ V_{z_e}$	(Velocities in the Earth frame),
$p,\ q,\ r$	(Angular rates),
$\dot{p},\ \dot{q},\ \dot{r}$	(Angular accelerations derived from the EKF),
$a_x,\ a_y,\ a_z$	(Linear accelerations),
TAS, WindSpeed_n, WindSpeed_e	(True airspeed and wind estimates),
$h,\ \dot{h}$	(Altitude and altitude rate).

2. **Rotation to the Center-of-Gravity (CG) Frame and Angular Acceleration.**
   The IMU measurements are initially reported in the sensor's frame. To obtain values at the aircraft's center of gravity, a fixed rotation is applied using the matrix $\mathbf{R}_{\text{IMU}\to\text{CG}}$. This transformation converts both the linear accelerations

and angular rates:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \mathbf{R}_{\text{IMU}\to\text{CG}} \begin{bmatrix} a_{x_{\text{IMU}}} \\ a_{y_{\text{IMU}}} \\ a_{z_{\text{IMU}}} \end{bmatrix}, \quad \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{R}_{\text{IMU}\to\text{CG}} \begin{bmatrix} p_{\text{IMU}} \\ q_{\text{IMU}} \\ r_{\text{IMU}} \end{bmatrix}. \tag{6.8}$$

3. **IMU Translation Correction.**
   If the IMU is not located at the aircraft CG, the measured accelerations must be corrected for the effects of rotation about the CG. Let $\mathbf{r}_{\text{IMU}\to\text{CG}}$ denote the position vector from the IMU to the CG (expressed in the body frame). The corrected acceleration in the CG frame is given by:

$$\mathbf{a}_{\text{CG}}^{(\text{body})} = \mathbf{a}_{\text{IMU}}^{(\text{body})} - \dot{\boldsymbol{\omega}} \times \mathbf{r}_{\text{IMU}\to\text{CG}} - \Big(\boldsymbol{\omega} \times \big(\boldsymbol{\omega} \times \mathbf{r}_{\text{IMU}\to\text{CG}}\big)\Big), \tag{6.9}$$

   where $\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^{\mathsf{T}}$ and $\dot{\boldsymbol{\omega}} = \begin{bmatrix} \dot{p} & \dot{q} & \dot{r} \end{bmatrix}^{\mathsf{T}}$. This correction accounts for the apparent acceleration effects induced by the sensor's offset from the CG.

4. **Reconstruction of Body Velocities** $(u,\, v,\, w)$**.**
   The corrected linear accelerations $(a_x,\, a_y,\, a_z)$ in the body frame are numerically integrated to obtain preliminary estimates of the body-frame velocities:

$$\begin{aligned} u_{\text{temp}}(k) &= \sum_{\tau=0}^{k} a_x(\tau)\, \Delta t, \\ v_{\text{temp}}(k) &= \sum_{\tau=0}^{k} a_y(\tau)\, \Delta t, \\ w_{\text{temp}}(k) &= \sum_{\tau=0}^{k} a_z(\tau)\, \Delta t. \end{aligned} \tag{6.10}$$

   Because numerical integration is susceptible to drift, these temporary velocities are corrected using the estimated airspeed. First, the relative airspeed in Earth coordinates is computed as:

$$\mathbf{V}_{\text{earth}} = \Big(V_{x_e} - \text{WindSpeed}_n,\, V_{y_e} - \text{WindSpeed}_e,\, V_{z_e}\Big). \tag{6.11}$$

   Then, by applying the standard direction-cosine matrix $\mathbf{R}_{\text{body}\to\text{earth}}(\phi, \theta, \psi)$, the

Earth-frame velocity is transformed into the body frame:

$$\mathbf{V}_{\text{body}} = \mathbf{R}^{\mathsf{T}}_{\text{body}\to\text{earth}}\,\mathbf{V}_{\text{earth}}. \tag{6.12}$$

The final body-frame velocities are corrected as follows:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u_{\text{temp}} \\ v_{\text{temp}} \\ w_{\text{temp}} \end{bmatrix} + \left(\mathbf{V}_{\text{body}} - \begin{bmatrix} u_{\text{avg}} \\ v_{\text{avg}} \\ w_{\text{avg}} \end{bmatrix}\right), \tag{6.13}$$

where u_{avg}, v_{avg}, and w_{avg} represent averaged values over a suitable window to mitigate drift. An alternative correction using the True Airspeed (TAS) was evaluated but found less effective due to its lower sampling rate and higher noise.

5. **Euler Angles and Final State Vector Formation.**
   For simplicity in subsequent analyses, the yaw angle ψ is adjusted by subtracting its initial value, effectively zeroing the heading at the start of the maneuver. The final state vector is then defined as:

$$\mathbf{x} = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta & \psi & h \end{bmatrix}^{\mathsf{T}}. \tag{6.14}$$

6. **Time Derivatives of the State Vector.**
   The time derivatives of the state vector components are computed using the vehicle's translational and rotational dynamics. For the translational dynamics in the body frame, the following equations are used:

$$\begin{aligned} \dot{u} &= a_x + r\,v - q\,w - g\,\sin(\theta), \\ \dot{v} &= a_y + p\,w - r\,u + g\,\cos(\theta)\,\sin(\phi), \\ \dot{w} &= a_z + q\,u - p\,v + g\,\cos(\theta)\,\cos(\phi). \end{aligned} \tag{6.15}$$

The evolution of the angular rates (p, q, r) is given by:

$$\begin{aligned} \dot{p} &= \dot{p}_{\text{body}}, \\ \dot{q} &= \dot{q}_{\text{body}}, \\ \dot{r} &= \dot{r}_{\text{body}}, \end{aligned} \tag{6.16}$$

where $\dot{p}_{\text{body}}$, $\dot{q}_{\text{body}}$, and $\dot{r}_{\text{body}}$ are obtained either from the EKF or through numerical differentiation of the IMU data. The Euler-angle kinematics are expressed as:

$$\begin{aligned} \dot{\phi} &= p + q\,\sin(\phi)\,\tan(\theta) + r\,\cos(\phi)\,\tan(\theta), \\ \dot{\theta} &= q\,\cos(\phi) - r\,\sin(\phi), \\ \dot{\psi} &= q\,\frac{\sin(\phi)}{\cos(\theta)} + r\,\frac{\cos(\phi)}{\cos(\theta)}. \end{aligned} \tag{6.17}$$

Additionally, the height rate is computed as per [169]:

$$\dot{h} = u\,\sin(\theta) - v\,\cos(\theta)\,\sin(\phi) - w\,\cos(\theta)\,\cos(\phi). \tag{6.18}$$

Thus, the state derivative vector is assembled as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{u} & \dot{v} & \dot{w} & \dot{p} & \dot{q} & \dot{r} & \dot{\phi} & \dot{\theta} & \dot{\psi} & \dot{h} \end{bmatrix}^{\mathsf{T}}. \tag{6.19}$$

In summary, the state vector $\mathbf{x}$ and its derivative $\dot{\mathbf{x}}$ are returned as two $\mathrm{N} \times 10$ arrays that serve as the basis for subsequent system identification or validation steps. Although $\dot{\mathbf{x}}$ is primarily reconstructed from the available IMU data, an alternative approach involving direct differentiation of the state vector $\mathbf{x}$ is also implemented. Comparative testing, particularly under higher noise levels, indicated that the reconstruction approach yields better results. Figure 6.2 compares the two methods for obtaining $\dot{\mathbf{x}}$.

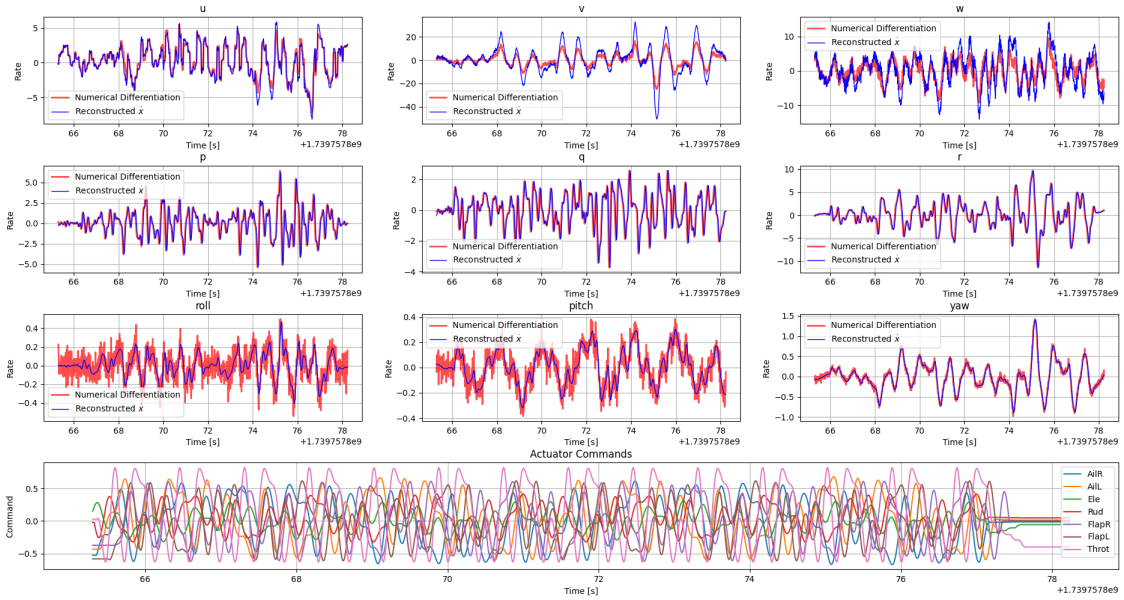


Figure 6.2: Comparison of $\dot{\mathbf{x}}$ derived via direct reconstruction versus numerical differentiation.

Command Delay Estimation and Shifting

When actuator commands and measured states are not synchronized, it is necessary to estimate and compensate for inherent time delays (such as communication lags, actuator dynamics, and sensor response times) to facilitate accurate system identification. The following procedure outlines our in-flight method for detecting and correcting these delays.

Let $u_i[n]$ denote the command signal for actuator i, and let $x_j[n]$ represent a measured state that is strongly influenced by that actuator (for example, roll rate or pitch rate). The delay estimation process proceeds as follows:

1. Signal Normalization: Both the command and state signals are normalized to zero mean and unit variance:

$$\tilde{u}_i[n] = \frac{u_i[n] - \mu_{u_i}}{\sigma_{u_i}}, \quad \tilde{x}_j[n] = \frac{x_j[n] - \mu_{x_j}}{\sigma_{x_j}}, \tag{6.20}$$

where μ_{u_i} and σ_{u_i} (respectively, μ_{x_j} and σ_{x_j}) denote the sample mean and standard deviation of the command (state) signal.

2. Cross-Correlation Calculation: To quantify the delay between $\tilde{u}_i[n]$ and $\tilde{x}_j[n]$, we compute the discrete cross-correlation for nonnegative lags:

$$R_{\tilde{u}_i,\tilde{x}_j}[\tau] = \sum_n \tilde{u}_i[n]\,\tilde{x}_j[n-\tau], \tag{6.21}$$

with $\tau \geq 0$ to reflect the physical constraint that the system response cannot precede the command.

3. Delay Identification: Within the interval $[0, \tau_{\max}]$, where $\tau_{\max}$ is set based on prior knowledge of plausible system latencies (e.g., 500 ms in our case), the optimal delay $\widehat{\tau}$ is determined by:

$$\widehat{\tau} = \underset{0\leq\tau\leq\tau_{\max}}{\arg\max} \left| R_{\tilde{u}_i,\tilde{x}_j}[\tau] \right|. \tag{6.22}$$

Because an actuator command may influence multiple states, the cross-correlation is computed for each candidate state, and the state yielding the largest absolute correlation identifies both the effective delay and the most directly influenced state.

Figure 6.3 illustrates a representative cross-correlation between an aileron command and the roll rate measured by the gyroscope. The prominent peak at a positive delay indicates the effective time offset between the control signal and the aircraft response.

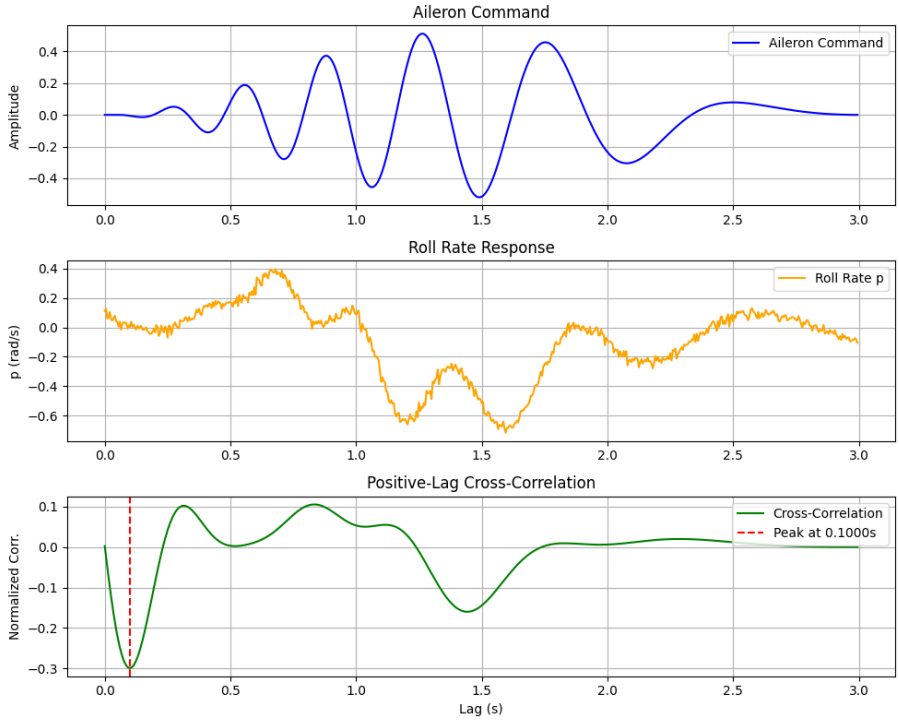


Figure 6.3: Example cross-correlation between an aileron command and the roll rate during a SID maneuver in turbulence.

4. Command Shifting: Once the delay $\widehat{\tau}_i$ for each actuator command is identified, the delay is converted to an integer number of samples. Given a common sampling period dt, the sample delay is

$$\Delta n_i = \left\lfloor \frac{\widehat{\tau}_i}{dt} \right\rceil . \tag{6.23}$$

The time-aligned (shifted) command signal $\hat{u}_i[n]$ is then constructed as:

$$\hat{u}_i[n] = \begin{cases} u_i[n - \Delta n_i], & \text{for } n \geq \Delta n_i, \\ u_i[0], & \text{for } 0 \leq n < \Delta n_i. \end{cases} \tag{6.24}$$

Filling the initial segment with $u_i[0]$ ensures continuity. This procedure is repeated for every actuator command signal to guarantee that the control inputs are temporally aligned with the corresponding system responses.

Actuator Model

In practice, the command sent to an actuator does not directly reflect the physical input due to the inherent filtering characteristics of the actuator. Sudden or rapid changes in the command signal are not executed instantaneously; rather, the actuators behave like low-pass filters. Ideally, direct measurements of control surface deflections (in degrees) or motor RPMs would be available for system identification. However, in many UAVs—including the one used in this work—such feedback signals are not accessible.

There are several options for addressing this limitation:

- **In-Flight Identification:** Include the actuator dynamics within the online system identification (SID) problem.
- **Bench Testing:** Identify actuator dynamics in a controlled test bench environment (as demonstrated in [79]) and use fixed models for each actuator.
- **Neglect Actuator Dynamics:** Omit actuator modeling at the cost of a reduced fit in the SID model.

In our controller implementation, the option to account for actuator dynamics is configurable. When enabled, each control input is passed through a first-order filter that approximates the actuator behavior. The filter is defined as

$$\begin{aligned} \alpha &= \frac{\Delta t}{\tau_a + \Delta t}, \\ u_{\text{deflected}}[k] &= (1-\alpha)\, u_{\text{deflected}}[k-1] + \alpha\, K\, u[k], \end{aligned} \tag{6.25}$$

where:

- Δt is the sampling period,
- τ_a is the actuator time constant (we use $\tau_a = 0.015$ seconds, based on bench tests of the servo under cruise load conditions),
- K is the actuator gain (set to $K = 1$ in our implementation, with any effectiveness discrepancies compensated by the aerodynamic coefficients identified during SID), and
- k is the discrete time index.

The output $u_{\text{deflected}}[k]$ represents the effective command that mimics the physical response of the actuator.

Finally, the processed commands, states, and state derivatives are concatenated into a time-synchronized matrix. This matrix serves as the input for the Model Supervisor and the Online System Identification Process, ensuring that all components of the system are aligned and accurately reflect the underlying dynamics.

6.2.3 Model Supervisor and Library

The Model Supervisor plays a critical role in maintaining an accurate representation of the vehicle dynamics by continuously evaluating and updating the model used for control. Its responsibilities can be summarized in two primary tasks:

1. **Real-Time Model Fit Evaluation:** The supervisor continuously compares the predictions of the active vehicle model against the actual flight data. If the discrepancy between the model's predicted states and the measured states exceeds a predefined threshold, the supervisor initiates a new command perturbation, thereby triggering the system identification process. This mechanism is designed to detect changes in vehicle dynamics and prompt an investigation of any new or evolving dynamics. For experimental flexibility, this triggering can be manually overridden from the ground station.

2. **Model Library Management and Selection:** After a command perturbation, newly identified models are archived in a library. The supervisor then evaluates these models against the currently active model, as well as previously stored models, to determine which best represents the vehicle's dynamics. The evaluation is based on a fitness metric—specifically, the coefficient of determination R^2—which quantifies the agreement between the predicted state derivatives and the measured (or numerically differentiated) state derivatives. Only a model that improves upon the current fit is selected for subsequent use by the model-based controller.

To quantify model performance, let $\mathbf{x}(t)$ denote the vector of vehicle states, $\dot{\mathbf{x}}_{\text{model}}(t)$ the time derivatives predicted by a candidate model, and $\dot{\mathbf{x}}_{\text{meas}}(t)$ the measured or numerically differentiated time derivatives. For each state i, the R^2 score is computed as:

$$R_i^2 = 1 - \frac{\sum_t \left(\dot{x}_{\text{meas},i}(t) - \dot{x}_{\text{model},i}(t)\right)^2}{\sum_t \left(\dot{x}_{\text{meas},i}(t) - \bar{\dot{x}}_{\text{meas},i}\right)^2}, \tag{6.26}$$

where $\bar{\dot{x}}_{\text{meas},i}$ is the mean of $\dot{x}_{\text{meas},i}(t)$ over the evaluation window. For multi-output systems, an aggregate R^2 value may be obtained by averaging or weighting the individual R_i^2 scores. In our implementation, a single scalar R^2 is used to reflect the overall goodness-of-fit across all states [170].

The R^2 metric is advantageous because it compresses the model's performance into a single interpretable value within the interval $(-\infty, 1]$. An R^2 value close to 1 (or 100%) indicates that the model explains most of the variance in the measured data, whereas a value near or below 0 suggests that the model performs poorly relative to a simple constant baseline. This single numeric score thus simplifies model comparison and selection within the library.

Additional parameters, such as model complexity, are monitored as well. Excessive model complexity may indicate issues with the convergence or tuning of the system identification (SID) process, leading to overfitting or poorly converged models. In such cases, overly complex models are flagged and discarded in favor of sparser representations.

Furthermore, the MIAC architecture offers the option for the supervisor to search the model library for a better-fitting model under current flight conditions before triggering a new command perturbation. This capability is particularly useful when the vehicle configuration changes (e.g., landing gear deployment), allowing a seamless model switch without the delay of a new SID process.

The fact that the model library maintains a historical archive of models over time means that can be used to parameterize certain model coefficients as operating conditions change or, in some cases, averaging model coefficients from multiple stored models can enhance robustness.

At the maiden flight, the model library is initialized with a reference model. This reference model incorporates estimated aerodynamic properties, control surface effectiveness, and mass/inertia characteristics, and in our case, it was derived from a CFD analysis of a similarly sized UAV. Details of this reference model are provided in Appendix A.

6.2.4 Command Perturbation

When the command perturbation process is triggered, the reference commands generated by the model-based controller are temporarily replaced with predefined perturbation signals. This intentional injection of external commands serves several purposes:

- **Dynamic Excitation:** It excites the vehicle's dynamics to achieve a high signal-to-noise ratio (SNR) for system identification.
- **Actuator Characterization:** It activates all control surfaces and propulsion units so that both their individual and collective contributions to the vehicle's dynamics can be observed.
- **Flight Envelope Exploration:** It allows the vehicle to safely explore its flight envelope by inducing controlled changes in state, thereby revealing how variations in airspeed and attitude affect overall dynamics.

To ensure safety, the amplitude and frequency content of the perturbation signals are carefully designed to avoid excessive loads, high angles of attack, extreme attitudes, or speeds outside the allowable range.

Data collection efficiency is also a key consideration. Perturbations that maximize the information content in the shortest possible training time reduce risk. For this reason, open-loop perturbations are chosen to simplify the system identification process by removing the confounding effects of feedback compensation. For vehicles that require continuous active control (such as inherently unstable platforms) a closed-loop perturbation approach may be necessary.

In this work, four different perturbation signals are implemented: *multisine*, *sequences of chirps*, *3-2-1-1*, and *doublets*. The multisine and chirp inputs are particularly well-suited for system identification due to their rich frequency content, while the 3-2-1-1 and doublet inputs serve as convenient validation signals. The following subsubsection provides a detailed description of the multisine perturbation.

Multisine

A key perturbation strategy adopted in this study is the multisine control input design, as proposed by Morelli [117] and validated in subsequent flight testing for system

identification [171, 172]. The multisine approach superimposes multiple sine waves—each with distinct frequencies, amplitudes, and phases—into the control command of each actuator. By exciting each control surface with a unique set of frequency components, the contribution of each actuator to the overall vehicle response can be identified simultaneously, enabling robust dynamic identification over a broad frequency range.

For the jth control actuator, the multisine input signal is defined as:

$$u_j(t) = \sum_{k \in K_j} A\sqrt{P_k} \cos\left(\frac{2\pi k}{T} \left(t + t_{\text{shift}}\right) + \phi_k \right), \tag{6.27}$$

where:

- A represents the maximum allowed amplitude of the perturbation (relative to the trim condition),
- K_j is the set of integer frequency indices allocated to the jth actuator,
- T is the total training (or excitation) time,
- t_{shift} is a time shift ensuring that the signal starts and ends at zero,
- ϕ_k is the phase lag for the kth frequency component, and
- P_k characterizes the power spectral distribution among the frequency components, satisfying $\sum_k P_k = 1$.

While a uniform power distribution (i.e., $P_k = 1/n$ for n frequencies) is commonly used, it is sometimes beneficial to bias or dampen certain frequency bands according to known or anticipated vehicle dynamics. For instance, for the elevator control input, an exponentially decaying power spectrum is applied to avoid excessive amplitudes at lower frequencies. This modification ensures sufficient excitation of the pitch dynamics without over- or under-speeding the vehicle.

The phase angles ϕ_k are optimized to minimize the Relative Peak Factor (RPF) of the control signal, which is defined as:

$$RPF\big(u_j\big) = \frac{\max\big(u_j(t)\big) - \min\big(u_j(t)\big)}{2\sqrt{2}\,\text{rms}\big(u_j(t)\big)}, \tag{6.28}$$

where $\mathrm{rms}\big(u_j(t)\big)$ is the root-mean-square value of the jth control input over one period of excitation. A lower RPF is preferable, as it indicates that the signal provides high excitation energy without causing large deviations from the trim condition.

To compute the phase angles that minimize the RPF, a non-convex optimization is performed offline for each actuator. In this study, a variant of the Nelder-Mead simplex method is used, terminating the optimization either when the RPF improvement falls below a predefined threshold (e.g., 0.05) or after a maximum number of iterations (e.g., 300). Although this approach does not guarantee a global optimum, it substantially reduces computational cost and typically yields acceptable RPF values (around 1.25) within the constraints of flight safety.

After optimizing the phase angles, a time shift t_{shift} is applied to ensure that the multisine signal begins and ends at zero. This is accomplished by locating the first zero crossing of the generated signal and adjusting the phase accordingly, thereby ensuring a smooth transition between the baseline and perturbed flight states.

Additional considerations are made when mapping the frequency indices k to a user-defined frequency range $[f_{\min}, f_{\max}]$. The minimum spacing between consecutive frequencies and the total number of frequencies are determined based on actuator-specific constraints and the training duration. A new parameter is introduced to enforce a higher minimum frequency spacing between actuators, ensuring adequate separation for reliable identification of each actuator's contribution. Moreover, if $f_{\min}$ is set too low relative to T, it is adjusted so that the corresponding low-frequency sine completes a full cycle.

Figures 6.4 and 6.5 show a representative multisine control input and its corresponding frequency spectrum, respectively. In the FFT plot, distinct peaks represent each frequency component, with the amplitude at each frequency determined by the chosen power distribution P_k. (In this example, only actuator 3 (elevator), actuator 4 (rudder), and actuator 7 (motor) employ a non-uniform P_k distribution.)

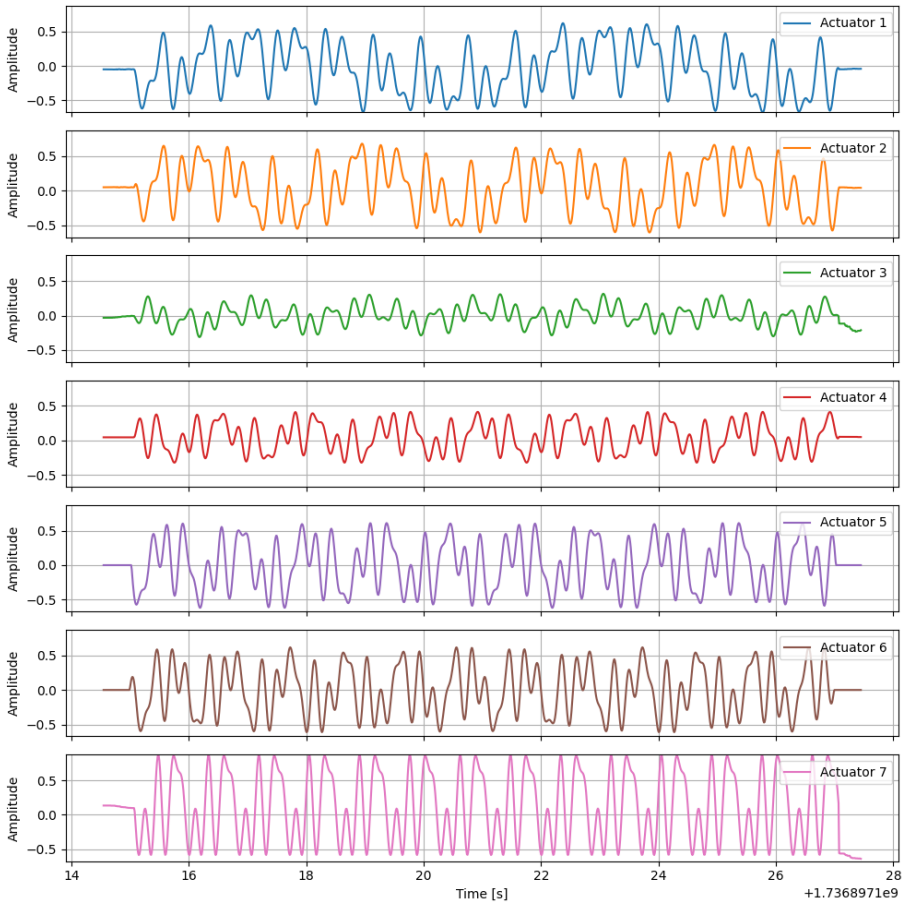


Figure 6.4: Example of a multisine control input.

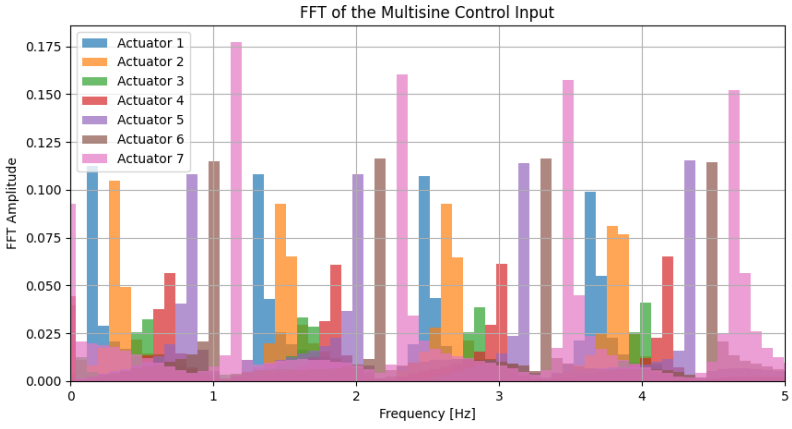


Figure 6.5: FFT of the example multisine input, showing the distinct frequency components that collectively excite the system.

Overall, the multisine approach is highly effective for online flight system identification. It provides a broad excitation spectrum while minimizing abrupt control deflections and reducing the training time required. This ensures that the vehicle's response remains within a safe operating envelope while yielding rich data for accurate model identification.

Chirp Sequence

For comparative testing and to explore a more classical perturbation approach, chirp signals (or swept-sine signals) are also implemented for control excitation. Chirps systematically excite the system across a specified frequency range by gradually sweeping from a lower bound frequency f_1 to an upper bound frequency f_2. This frequency sweep yields valuable insight into the dynamics over that range.

The chirp design adopted in this work is an *exponential swept-sine signal* as proposed by Novak and Burrascano [173, 174]. The signal is defined by a time-varying amplitude $a(t)$ and an instantaneous phase $\phi(t)$:

$$u(t) = a(t)\,\sin\big(\phi(t)\big). \tag{6.29}$$

In this formulation:

- The amplitude $a(t)$ is designed to decay exponentially over time, mitigating the risk of large deviations from trim at low frequencies where system gains may be high. Typically, the amplitude is given by

  $$a(t) = A_0\,\exp\big(-\alpha\,t\big), \tag{6.30}$$

  where A_0 is the initial amplitude and $\alpha > 0$ is a constant controlling the decay rate.

- The instantaneous phase $\phi(t)$ ensures that the frequency sweeps exponentially from f_1 at $t = 0$ to f_2 at $t = T$. Following [173], the phase is defined as

  $$\phi(t) = 2\pi\,f_1\,L\left[\mathrm{e}^{t/L} - 1\right] + \phi_0, \tag{6.31}$$

  with

  $$L = \frac{T}{\ln\left(\frac{f_2}{f_1}\right)}, \tag{6.32}$$

where T is the total duration of the chirp and ϕ_0 is an optional phase offset. The instantaneous frequency is then given by

$$f(t) = \frac{1}{2\pi} \frac{d\phi(t)}{dt}, \tag{6.33}$$

which produces a logarithmic distribution of frequencies over the interval $[f_1, f_2]$.

In practice, the total training time T_{train} may be divided equally among $N_{\text{actuators}}$, assigning each actuator a time slice $T_{\text{slice}} = T_{\text{train}}/N_{\text{actuators}}$. During its allocated slice, an actuator performs the exponential chirp while the others maintain trim or zero perturbation. This serial approach minimizes the risk of overlapping frequency content between actuators, thereby simplifying parameter estimation.

Figure 6.6 illustrates an example of a chirp excitation sequence, showing the amplitude decay over time and the exponential frequency sweep from f_1 to f_2. Compared to the multisine input, the chirp approach generally requires a longer training period to acquire equivalent frequency-response data. However, by isolating the frequency content in separate time windows, actuator-to-actuator interference is reduced, an advantage when dynamic responses at closely spaced frequencies are challenging to distinguish or when cross-coupling is a concern.

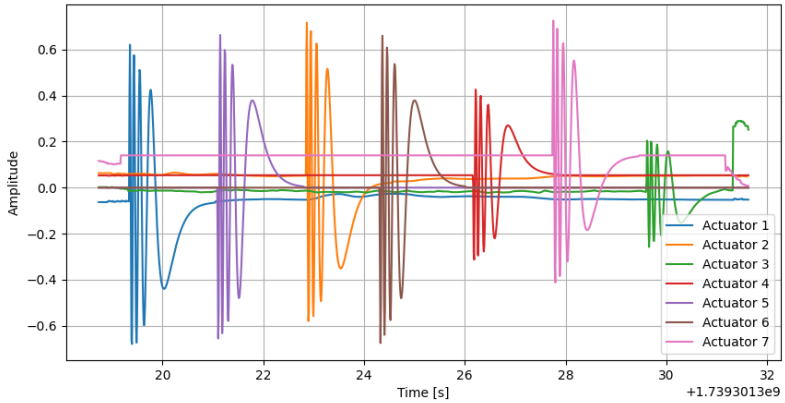


Figure 6.6: Example of a chirp sequence (exponential swept-sine).

3-2-1-1 Sequence

A classical step-based excitation used in flight control system identification is the *3-2-1-1* sequence. This method applies a series of step inputs with alternating signs

following a prescribed ratio, typically denoted as "3," "2," "1," and "1" in terms of relative magnitude or duration. Like the chirp signal, these excitations are executed sequentially across different actuators within the available training time.

In its simplest implementation, each 3-2-1-1 maneuver divides the total allocated time for an actuator into four distinct segments:

- The total duration of the maneuver matches the time slice allocated to that actuator.
- Each segment applies a step of either $+A$ or $-A$, where A is a design parameter.
- The sign of A alternates to ensure the vehicle experiences both positive and negative deflections.

The amplitude pattern is scaled according to the ratios (3, 2, 1, 1), ensuring that the excitation remains within safe flight conditions.

In our experimental setup, the total training time T_{train} is divided among $N_{\text{actuators}}$, with each actuator assigned a time window $T_{\text{slice}} = T_{\text{train}}/N_{\text{actuators}}$. Within this window, the actuator steps through four sub-windows that correspond to the 3-2-1-1 pattern plus a deadband. The durations of these sub-windows are proportional to the specified timing ratios (i.e., $3 + 2 + 1 + 1 + 1 = 8$ parts in total), and the deflection amplitude alternates in sign accordingly.

Figure 6.7 shows a representative 3-2-1-1 sequence. This pattern produces clear, well-separated step responses, which simplifies the interpretation of transient dynamics during system identification. However, since the steps are executed sequentially, the approach can be time-consuming for vehicles with many actuators. Moreover, large step amplitudes (especially in the initial or intermediate phases) may risk driving the aircraft outside its normal flight envelope. Therefore, careful selection of the amplitude A is essential to balance sufficient excitation energy with safety constraints.

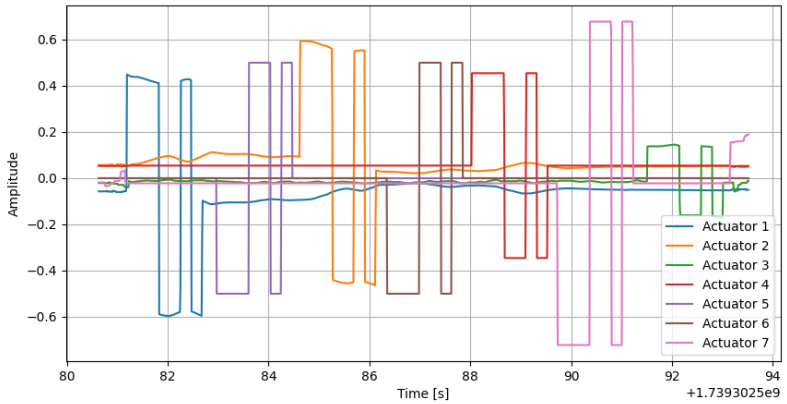


Figure 6.7: Example of a 3-2-1-1 control input sequence.

Doublet Sequence

Another well-known step-based maneuver is the *doublet* input. A doublet consists of two consecutive step commands of equal magnitude and opposite sign, causing the control surface to move quickly from a positive deflection to a negative deflection (or vice versa). This approach is widely used for simpler excitation and quick checks of the dynamic response.

In practice, each actuator is assigned its own time window T_{slice}, which can be subdivided to accommodate the doublet maneuver. The simplest doublet sequence for a single actuator is defined as:

$$u(t) = \begin{cases} +A, & 0 \leq t < \Delta t, \\ -A, & \Delta t \leq t < 2\Delta t, \\ 0, & t \geq 2\Delta t, \end{cases} \tag{6.34}$$

where A is the chosen amplitude and Δt is the step duration. In our implementation, each actuator executes its doublet in turns and then returns to zero or its trim value, allowing the remaining actuators to perform their excitations during their allocated time windows.

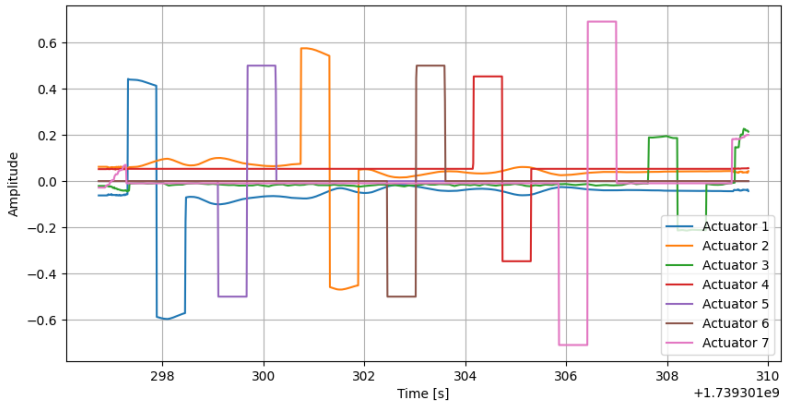


Figure 6.8: Example of a doublet control input sequence.

Similar to the 3-2-1-1 and chirp approaches, doublets are applied sequentially across actuators. In our scheduling, actuators that induce minimal disturbances are tested first, followed by those with more significant effects. This strategy helps keep the aircraft closer to trim during the initial portions of the test and minimizes abrupt deviations from safe flight conditions.

Compared to complex signals like multisine or chirps, doublet inputs provide fewer frequency components and thus yield less rich information for high-fidelity system identification. However, their simplicity, ease of implementation, and distinct transient responses make them valuable for rapid checks or to supplement more sophisticated perturbation methods.

Note on Sequencing. The order in which actuators execute these perturbation sequences can significantly affect data quality and flight safety. Actuators with minimal impact on overall flight dynamics are typically perturbed first to ensure stability, while actuators with opposing effects (e.g., ailerons on opposite wings) are excited in succession to maintain symmetry in the vehicle's response. This sequencing concern is less critical for multisine inputs, which are generally injected concurrently over the entire training period.

Trim

To prevent the aircraft from departing excessively from trim conditions during perturbation sequences, a trim offset is added to the command signal. This offset ensures

that the vehicle's nominal state remains close to a safe, steady flight condition, even during strong or prolonged excitations.

Static Trim Estimation. A static trim value, u_{trim_j}, is computed before the perturbation procedure begins by averaging the autopilot commands over a recent time window. Formally, if $u_{\text{AP}_{j,t}}$ denotes the jth channel command from the onboard autopilot at time t, then:

$$u_{\text{trim}_j} = \frac{1}{n} \sum_{t \in T_{\text{window}}} u_{\text{AP}_{j,t}}, \tag{6.35}$$

where T_{window} is the interval covering the last n seconds (e.g., 3 seconds in our tests). This value captures the nominal control positions required to maintain a stable flight state. Once established, u_{trim_j} is added as a baseline offset to all subsequent perturbation commands.

Active (Dynamic) Trimming. In addition to the static trim, a slow, low-gain *dynamic trim* adjustment is optionally applied during excitation. Its purpose is to gently nudge the aircraft back toward nominal flight if it drifts too far from desired states such as roll angle, airspeed, or altitude. The dynamic trim can be expressed as a feedback term:

$$u_{\text{dyn\_trim}_j}(t) = f\Big(\mathbf{e}(t)\Big), \tag{6.36}$$

where $\mathbf{e}(t)$ represents the error between the current flight state and the initial reference, and f is a slow-reacting controller that filters and limits the rate of change in the trim correction.

An exponential smoothing (low-pass) filter is applied to the raw error $\mathbf{e}(t)$ to avoid reacting to the rapid variations induced by the perturbation signal:

$$\mathbf{e}_f(t_k) = \alpha\, \mathbf{e}(t_k) + \big(1 - \alpha\big)\, \mathbf{e}_f(t_{k-1}), \tag{6.37}$$

where $0 < \alpha < 1$ is the smoothing factor, t_k denotes the kth discrete time step, and $\mathbf{e}_f(t_{k-1})$ is the previously filtered error.

The filtered error is then scaled by a small proportional gain to produce a preliminary correction command $\mathbf{c}_0(t_k)$. For example, for roll and speed channels:

$$c_{\phi_1}(t_k) = k_\phi\, e_{\phi_f}(t_k), \qquad c_{\phi_2}(t_k) = -\,k_\phi\, e_{\phi_f}(t_k), \qquad c_v(t_k) = -\,k_v\, e_{v_f}(t_k), \tag{6.38}$$

or in vector form,

$$\mathbf{c}_0(t_k) = K\,\mathbf{e}_f(t_k), \tag{6.39}$$

where K is a gain matrix mapping each filtered error component to the relevant control channel.

To avoid abrupt or large changes in the trim command, the magnitude of the change per unit time is capped. Let $\Delta t = t_k - t_{k-1}$ and denote the dynamic trim command at time t_{k-1} by $\mathbf{c}(t_{k-1})$. Then:

$$\Delta\mathbf{c}(t_k) = \mathbf{c}_0(t_k) - \mathbf{c}(t_{k-1}), \tag{6.40}$$

$$\Delta\mathbf{c}(t_k) = \mathrm{clip}\Big(\Delta\mathbf{c}(t_k), -\dot{\mathbf{c}}_{\max}\,\Delta t,\, \dot{\mathbf{c}}_{\max}\,\Delta t\Big), \tag{6.41}$$

and the updated dynamic trim command is

$$\mathbf{c}(t_k) = \mathbf{c}(t_{k-1}) + \Delta\mathbf{c}(t_k). \tag{6.42}$$

Here, the clip function limits each component of $\Delta\mathbf{c}(t_k)$ to the range $\pm\dot{\mathbf{c}}_{\max}\,\Delta t$.

Finally, the total command sent to each actuator is given by the sum of the static trim, the dynamic trim, and the perturbation command, all constrained by the actuator limits:

$$u_{\mathrm{final}_j}(t) = \mathrm{clip}\Big(u_{\mathrm{trim}_j} + c_j(t) + u_{\mathrm{pert}_j}(t),\, -1,\, 1\Big). \tag{6.43}$$

In practice, parameters such as α, k_ϕ, k_v, and $\dot{\mathbf{c}}_{\max}$ are tuned to ensure that the dynamic trim gently counteracts low-frequency drifts or disturbances without negating the higher-frequency excitation required for system identification. This approach keeps the aircraft near its desired flight condition while preserving the integrity of the perturbation input.

Considerations for System Identification. For short-duration excitations, open-loop tests using only a static trim are generally preferred because any active feedback may distort the measured vehicle responses or interfere with the intended input frequencies. However, for longer test campaigns or in environments with significant disturbances (e.g., gusty winds), a slow dynamic trim can help maintain safety and prevent large deviations. Care must be taken (especially in multisine perturbations) since low-frequency components might be counteracted by the feedback trim, compro-

mising the orthogonality of the actuator signals. Future work may consider dynamic inversion or other feedforward strategies to maintain altitude or speed without sacrificing identification fidelity.

Transition to System Identification. At the conclusion of the command perturbation phase—and after a brief waiting interval to allow transients to settle—the collected input-output data are passed to the identification routine. The *System Identification* process then fits a new dynamic model or updates the existing model library based on the most recent vehicle responses, setting the stage for adaptive control updates as described in subsequent sections.

6.2.5 Online System Identification

The online system identification (SID) process is the core component of the MIAC architecture. Its ability to accurately and robustly capture the vehicle dynamics is critical for effective adaptive control. The conceptual basis for the online SID is outlined in [115] and builds upon the Sparse Identification of Nonlinear Dynamical Systems (SINDy) framework developed by Brunton et al. [35, 157, 170, 175, 176]. By leveraging sparse regression techniques, the SID method strikes a balance between performance and flexibility, enabling real-time estimation of the governing equations of motion for the UAV from measured states and control inputs.

When triggered, the online SID process receives a flight data matrix containing the pre-processed, time-synchronized measurements of the vehicle states, their time derivatives, and the control inputs. These data have been carefully filtered, resampled, and adjusted for command delays and actuator dynamics, as described in previous sections. This rigorous data preparation is essential for successful system identification.

State Representation and Control Input

An important early decision in the SID process is the selection of the state variables. In our work, we use the standard fixed-wing states. Note that certain variables, such as air density, are omitted from the state vector; consequently, the identified models are not explicitly parameterized with respect to air density. For the vehicles used in this research, this limitation is acceptable given the limited range of operating conditions.

The state vector is defined as:

$$\mathbf{x} = \begin{bmatrix} u & v & w & p & q & r & \phi & \theta & \psi \end{bmatrix}^{\mathsf{T}}, \tag{6.44}$$

where u, v, w represent the body-frame linear velocities, p, q, r denote the angular velocities, and ϕ, θ, ψ are the roll, pitch, and yaw Euler angles, respectively. The altitude h is omitted from the SID process because its dynamics are well defined for fixed-wing aircraft:

$$\dot{h} = u \sin(\theta) - v \cos(\theta) \sin(\phi) - w \cos(\theta) \cos(\phi). \tag{6.45}$$

The control input vector for the test vehicle is defined as:

$$\mathbf{u} = \begin{bmatrix} u_{ailR} & u_{ailL} & u_{ele} & u_{rud} & u_{flapR} & u_{flapL} & u_{thr} \end{bmatrix}^{\mathsf{T}}. \tag{6.46}$$

The time derivatives of the states, $\dot{\mathbf{x}}$, are computed as described in Section 6.2.2.

Function Library

The underlying assumption in our model is that the system dynamics follow a general nonlinear form:

$$\dot{\mathbf{x}} = \boldsymbol{\Theta}(\mathbf{x}, \mathbf{u})\, \boldsymbol{\Xi}, \tag{6.47}$$

where $\boldsymbol{\Theta}(\mathbf{x}, \mathbf{u})$ is a library of candidate functions, and $\boldsymbol{\Xi}$ is the coefficient matrix that selects the active terms in the model. The candidate function library is constructed to capture a wide range of dynamic behaviors, including:

- **Bias Term:** Constant offset for baseline effects.
- **Linear Functions:** Identity mappings of states and inputs for proportional dynamics.
- **Polynomial Functions:** Higher-order terms (squares, cross-products) to capture nonlinear interactions.
- **Trigonometric Functions:** Sine and cosine terms to model periodic and oscillatory behavior, particularly in angular motions.
- **Kinematic Equations:** Predefined relations that enforce the constraints of rotational motion.

- **Gravity and Force Terms:** Functions incorporating gravitational acceleration and force-based effects.
- **Moment Equations:** Terms representing rotational interactions involving inertia and angular rates.
- **Aerodynamic Angle Terms:** Angle of attack (α) and sideslip angle (β), which are critical for accurate aerodynamic modeling.
- **True Airspeed:** Functions capturing the magnitude of velocity components to reflect aerodynamic influences.

A careful selection of the library is crucial. A library that is too large increases the complexity of the regression problem, whereas one that is too small may fail to capture the vehicle dynamics. The complete library of functions used in our simulation and flight tests is available in Appendix B.

Sparse Regression Problem

The identification of $\boldsymbol{\Xi}$ is achieved by solving a sparse regression problem with sparsity-promoting constraints. Several methods can be employed:

- **Constrained Sparse Relaxed Regression (Constrained SR3):** This method solves a relaxed regression problem with constraints that enforce physical consistency. The objective function is formulated as:

$$\begin{aligned} &\min_{\boldsymbol{\Xi},\mathbf{W}} \frac{1}{2}\|\dot{\mathbf{x}} - \boldsymbol{\Theta}(\mathbf{x},\mathbf{u})\,\boldsymbol{\Xi}\|_2^2 + \lambda\, R(\mathbf{W}) + \frac{1}{2}\nu\|\boldsymbol{\Xi} - \mathbf{W}\|_2^2, \\ &\text{subject to } C\,\mathbf{W} = d, \end{aligned} \tag{6.48}$$

  where $\mathbf{W}$ is an auxiliary variable, λ and ν are hyperparameters, and C and d encode physical constraints.

  In our formulation, the regularization function $R(\mathbf{W})$ is treated as a hyperparameter that can be set to different functions:

  - ℓ_0: Enforces strict sparsity by directly penalizing the number of non-zero coefficients.
  - ℓ_1: Provides a convex relaxation of the ℓ_0 norm, promoting sparsity while keeping the optimization computationally efficient.

  - ℓ_2: Applies ridge regularization, which smoothly shrinks coefficients prioritizing robustness over aggressive sparsity.

  More information about this optimization and the implementation may be found in [177, 178].

- **Mixed Integer Optimization for Sparse Regression (MIOSR):** This approach enforces sparsity through combinatorial constraints:

$$\min_{\boldsymbol{\xi},\mathbf{z}} \|\dot{\mathbf{x}}_j - \boldsymbol{\Theta}(\mathbf{x},\mathbf{u})\,\boldsymbol{\xi}\|_2^2 + \lambda\|\boldsymbol{\xi}\|_2^2, \tag{6.49}$$

$$\text{subject to} \quad M_i^L z_i \le \xi_i \le M_i^U z_i, \quad i = 1,\dots,D, \tag{6.50}$$

$$\sum_{i=1}^{D} z_i \le k_j, \tag{6.51}$$

$$\xi_i \in \mathbb{R}, \quad z_i \in \{0,1\}, \quad i = 1,\dots,D. \tag{6.52}$$

  The parameters M_i^L and M_i^U are bounds on the coefficients, and k_j specifies the target sparsity. More information about this optimization method might be found in [179].

- **Sequential Threshold Least Squares (STLSQ):** This iterative method solves a least-squares problem while thresholding small coefficients to zero:

$$\min_{\boldsymbol{\Xi}} \|\dot{\mathbf{x}} - \boldsymbol{\Theta}(\mathbf{x},\mathbf{u})\,\boldsymbol{\Xi}\|_2^2 + \lambda\|\boldsymbol{\Xi}\|_2^2. \tag{6.53}$$

  More information about the application of this optimization method may be found in [157].

Initialization Strategy: For all methods, the optimization's convergence is highly sensitive to the initial guess for the coefficients. Instead of starting with random values, we initialize the coefficients using prior models. Specifically, the previous current best model is used as the initial guess, while a reference model is employed for the first flight. Remarkably, even an initial guess from a completely different vehicle has been observed to substantially improve convergence.

For the SR3 and MIOSR methods, the identification can be constrained using

prior physical knowledge, such as enforcing the kinematic equations for angular rates:

$$\dot{\phi} = p + q\ \sin(\phi)\ \tan(\theta) + r\ \cos(\phi)\ \tan(\theta), \tag{6.54}$$

$$\dot{\theta} = q\ \cos(\phi) - r\ \sin(\phi), \tag{6.55}$$

$$\dot{\psi} = \frac{q\ \sin(\phi) + r\ \cos(\phi)}{\cos(\theta)}. \tag{6.56}$$

These constraints improve convergence by ensuring that the estimated model adheres to fundamental flight dynamics. In our experiments, only the equations for $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ were constrained, as unconstrained trigonometric functions often degraded convergence in the optimizer.

Hyperparameter Selection

Hyperparameter optimization plays a critical role in balancing model accuracy and complexity. The optimization process searches over key hyperparameters such as:

- Regularization thresholds (λ) for sparsity enforcement.
- Penalization terms for the SR3 and STLSQ methods.
- Target sparsity levels for MIOSR.
- Maximum iteration limits to ensure convergence.

Each hyperparameter configuration is evaluated based on:

1. The model's ability to reconstruct the training dataset ($\dot{\mathbf{x}}$) from $\mathbf{x}$ and $\mathbf{u}$.
2. The validation score computed on an independent dataset (out-of-sample).
3. The model complexity, measured by the number of nonzero coefficients in $\boldsymbol{\Xi}$.

Rather than selecting the hyperparameters that yield the absolute highest out-of-sample fit, we choose those that achieve a fit within a specified threshold of the maximum while minimizing model complexity. Figure 6.9 shows an example hyperparameter search outcome for the SR3 method.

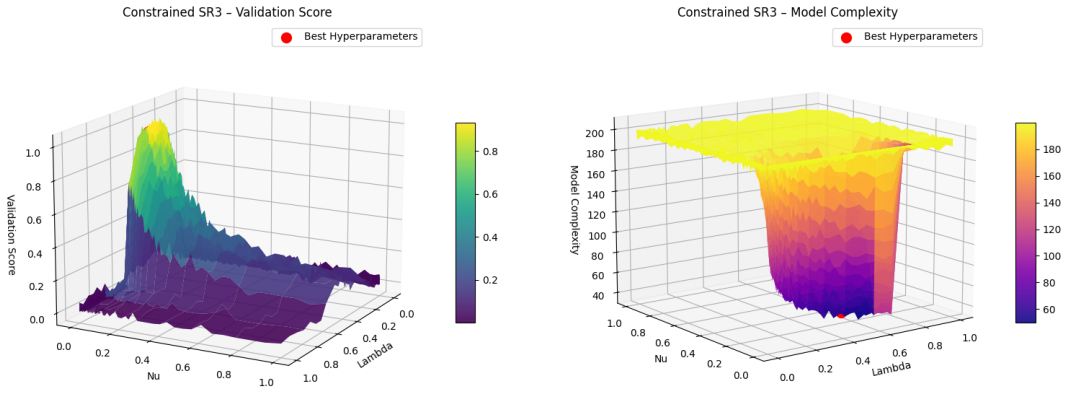


Figure 6.9: Model Score and Complexity of the SID model as a function of the optimization hyperparameters.

A careful analysis of the sparsification parameter is crucial to ensure that the regression converges reliably over a broad range of operating conditions.

Real-Time Identification Process

The real-time SID process proceeds as follows:

1. Pre-process sensor data to obtain the state $\mathbf{x}$, its time derivative $\dot{\mathbf{x}}$, and control input $\mathbf{u}$.
2. Generate the function library $\boldsymbol{\Theta}(\mathbf{x}, \mathbf{u})$ based on the selected candidate functions.
3. Load a reference model as an initial guess for the optimization.
4. Solve the sparse regression problem to estimate the coefficient matrix $\boldsymbol{\Xi}$.
5. Evaluate model accuracy using a scoring function (e.g., R^2).
6. Save the identified model for use within the adaptive control framework.

Additional options for enhancing the SID process include:

- **Using Multiple Trajectories:** If training data is limited (e.g., due to constrained flight space or significant deviations from setpoints), perturbations can be performed multiple times. The resulting data can be concatenated, and the sparse regression can be run with multiple initial conditions.

- **Ensembling:** The training data may be segmented (e.g., into 20 segments) and the SID process applied separately to each. The resulting models can be averaged to yield an "ensemble" model with statistical measures (such as coefficient deviation) to improve robustness.
- **Library Ensembling:** Alternatively, the function library itself can be segmented into subsets, and identification performed on each subset. This approach is generally less effective if the library is small.
- **Incorporating Prior Physical Knowledge:** Adding inequality constraints (rather than strict equality constraints) to the differential equations provides some leeway while still enforcing fundamental physical relationships.

Once the SID model is identified, extracting aerodynamic coefficients is straightforward if the vehicle's mass, inertia, and geometric properties are known. This capability is useful not only for simulating the test vehicle but also for exploring flight scenarios with modified physical properties.

6.2.6 Model-Based Controller

This section describes the model-based controller used in our research. The controller leverages the best available model, obtained from the online system identification (SID) process and selected by the model supervisor, to generate optimal control inputs that drive the vehicle toward desired setpoints. In a model-based controller, a mathematical representation of the system dynamics is used to predict future behavior and plan control actions accordingly. For our UAV application, a Model Predictive Control (MPC) strategy was chosen due to its proven performance in similar applications [180].

The controller is implemented using the state-of-the-art do-mpc framework [181], which formulates and solves a finite-horizon optimal control problem at each time step. The identified model from the SID process is used as the internal prediction model. In the following sections, we detail how this MPC model is constructed, how delays and actuator dynamics are incorporated, and how the cost function and constraints are designed.

MPC Model Formulation

Before the MPC can be applied, the SID model must be transformed into a form suitable for real-time optimization. The SID model, originally obtained as a sparse regression solution with many candidate functions, is first "cleaned" to retain only the active terms. It is then translated into CasADi notation [182] to facilitate efficient nonlinear optimization.

After conversion, additional dynamics that were not directly captured in the SID process are re-introduced:

- **Height Dynamics:** Although the SID process omits altitude h (given its well-defined behavior in fixed-wing aircraft), the height dynamics are incorporated by adding the differential equation (6.18).
- **Control Delay and Actuator Dynamics:** To account for the identified delays in the system, a first-order lag filter approximates the time delay. For each control input $u(t)$, a corresponding delayed state $u_{\text{delayed}}(t)$ is introduced:

$$\frac{d}{dt}\big(u_{\text{delayed}}(t)\big) = \frac{1}{\tau}\Big(u(t) - u_{\text{delayed}}(t)\Big), \tag{6.57}$$

  where τ is the identified delay. In the model equations, every occurrence of u is replaced by u_{delayed} to accurately represent the lagged effect. More accurate delay representations (e.g., cascading multiple first-order lags or using higher-order Padé approximations) can be used if higher fidelity is required.

MPC Cost Function Design

A receding horizon strategy is adopted, where at each time step the following finite-horizon optimal control problem is solved:

$$\begin{aligned}
\min_{\{x_k,\,u_k\}_{k=0}^{N}} \quad & \sum_{k=0}^{N-1}\Big(\ell(x_k,u_k) + \Delta u_k^{\mathsf{T}} R\,\Delta u_k\Big) + m(x_N,u_N) \\
\text{subject to} \quad & x_{k+1} = f(x_k,u_k), \quad k = 0,\dots,N-1, \\
& x_0 = x_{\text{measured}}, \\
& x_k \in \mathcal{X}, \quad u_k \in \mathcal{U}, \quad k = 0,\dots,N-1.
\end{aligned} \tag{6.58}$$

In our implementation:

- The **stage cost** $\ell(x_k, u_k)$ penalizes deviations from reference setpoints. For example, a typical stage cost is defined as

$$\ell(x_k, u_k) = \Big(\frac{V - V_{\text{sp}}}{C_{\text{Speed}}}\Big)^2 + \Big(\frac{v}{C_v}\Big)^2 + \Big(\frac{\phi - \phi_{\text{sp}}}{C_{\text{Roll}}}\Big)^2 + \begin{cases} \Big(\frac{h - h_{\text{sp}}}{C_{\text{Altitude}}}\Big)^2, & \text{(TECS mode)} \\ \Big(\frac{\theta - \theta_{\text{sp}}}{C_{\text{Pitch}}}\Big)^2, & \text{(Attitude mode)} \end{cases} \tag{6.59}$$

  where $V = \sqrt{u^2 + v^2 + w^2}$ is the total airspeed, v is the side-slip speed, and C_{Speed}, C_v, C_{Roll}, C_{Altitude}, C_{Pitch} are user-defined weights.

- The term $\Delta u_k^{\mathsf{T}} R\, \Delta u_k$ penalizes the rate of change in the control inputs ($\Delta u_k = u_k - u_{k-1}$) to ensure smooth control actions, where R is a weighting matrix.

- The **terminal cost** $m(x_N, u_N)$ includes the stage cost at the terminal state and an additional penalty on the angular rates (p, q, r) to ensure that rotational dynamics settle:

$$m(x_N, u_N) = \ell(x_N, u_N) + \Delta u_N^{\mathsf{T}} R\, \Delta u_N + (p^2 + q^2 + r^2). \tag{6.60}$$

Optional Constraints and Setpoint Management

Although our core MPC formulation is unconstrained for computational efficiency, the framework permits the imposition of state and input constraints when necessary. For instance, one might specify:

$$9.0\,\text{m/s} \le V \le 20.0\,\text{m/s},$$

and

$$-1.0 \le u_{\text{ailR}},\, u_{\text{ailL}},\, u_{\text{ele}},\, u_{\text{rud}},\, u_{\text{flapR}},\, u_{\text{flapL}},\, u_{\text{thr}} \le 1.0.$$

These constraints guarantee that the controller operates within safe and hardware-prescribed limits. However, they add computational overhead, so in our nominal tests the cost function was designed conservatively enough to naturally keep states and inputs within acceptable ranges.

The MPC framework also supports *time-varying setpoints*. Two modes are available for handling setpoints over the prediction horizon:

- **Future-Aware Setpoints:** The MPC is provided with the expected evolution of the setpoints over the prediction horizon, allowing it to anticipate changes.
- **Blind Setpoints:** The setpoint is treated as constant over the horizon and is updated only at each re-optimization. This allows for an easier performance comparison with traditional PID controllers.

Figure 6.10 illustrates the difference in vehicle response between these approaches. When the controller is supplied with future setpoints, it effectively anticipates upcoming maneuvers, thereby reducing the overall error between the state and the setpoint. This advantage is particularly significant for missions that follow a predetermined plan. In contrast, when the setpoints are provided in real time, such anticipatory control is not possible, resulting in a delayed and more reactive response.

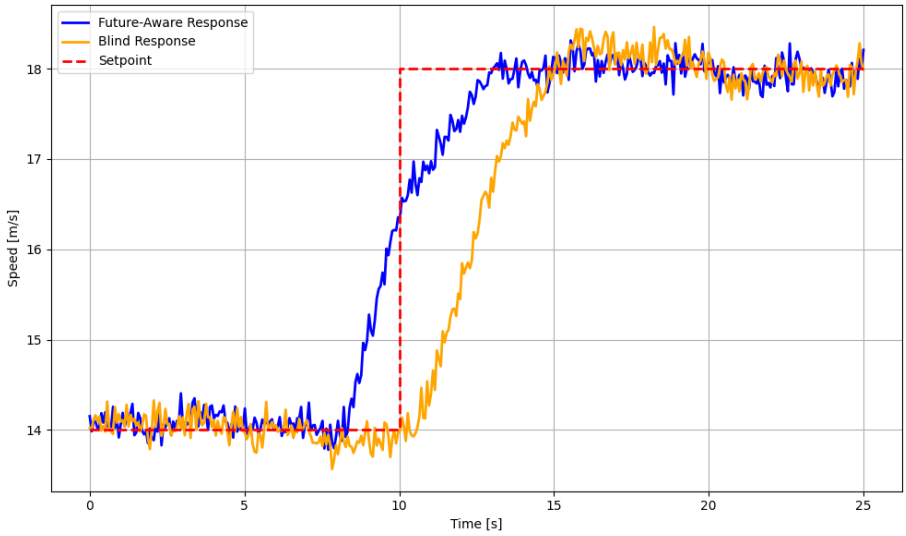


Figure 6.10: Comparison of MPC setpoint strategies: future-aware versus blind setpoint modes.

Closed-Loop Implementation and Computational Considerations

After solving the finite-horizon optimal control problem, the first control input u_0^* is applied to the system. The MPC then operates in a receding horizon fashion, re-solving the optimization at each time step using updated measurements.

The computational burden of the MPC is influenced by several factors, including the prediction horizon N, time step dt, model complexity, number of constraints, and cost function formulation. For our fixed-wing UAV application, we maintain

a command frequency between 20 and 50 Hz (using $N = 20$ and $dt = 0.1$ s). In contrast, more agile platforms such as multirotors may require a significantly higher command frequency (e.g., 100–400 Hz).

Every additional element in the model (e.g., extra dynamics, additional constraints) increases computational demand. Attractive options to reduce complexity is to reduce model order or to adopt alternative state representations. For example, replacing the body-frame velocities u, v, w with total airspeed, angle of attack (α), and sideslip angle (β) could lower the computational cost required by the MPC.

6.3 MIAC Integration and Testing

In this section, we describe the integration of the Model Identification Adaptive Controller (MIAC) into hardware. We detail the modifications made for flight testing, the test vehicle setup, and the validation plan for the proposed architecture.

6.3.1 Hardware Architecture

The MIAC system, described in Section 6.2, was initially developed and validated in a simulation environment. In simulation, the core adaptive control and model identification algorithms were implemented in Python, while a high-fidelity Simulink flight dynamics model emulated vehicle aerodynamics, actuator response, sensor behavior, and communication delays. This setup enabled rapid prototyping and refinement with realistic flight variable behavior.

Transitioning from simulation to real flight required addressing several hardware integration and safety considerations. A key design choice was to adopt a dual-autopilot configuration (see Figure 6.11). In this configuration, a robust, proven autopilot (the *primary autopilot*) handles essential flight tasks such as takeoff, landing, guidance, and navigation. It also provides a full command-and-control (C2) interface to Ground Station 1 and the pilot console, while managing low-level sensor drivers and actuator outputs. This architecture minimizes duplication of critical functions in the experimental MIAC controller, reducing development overhead and increasing safety.

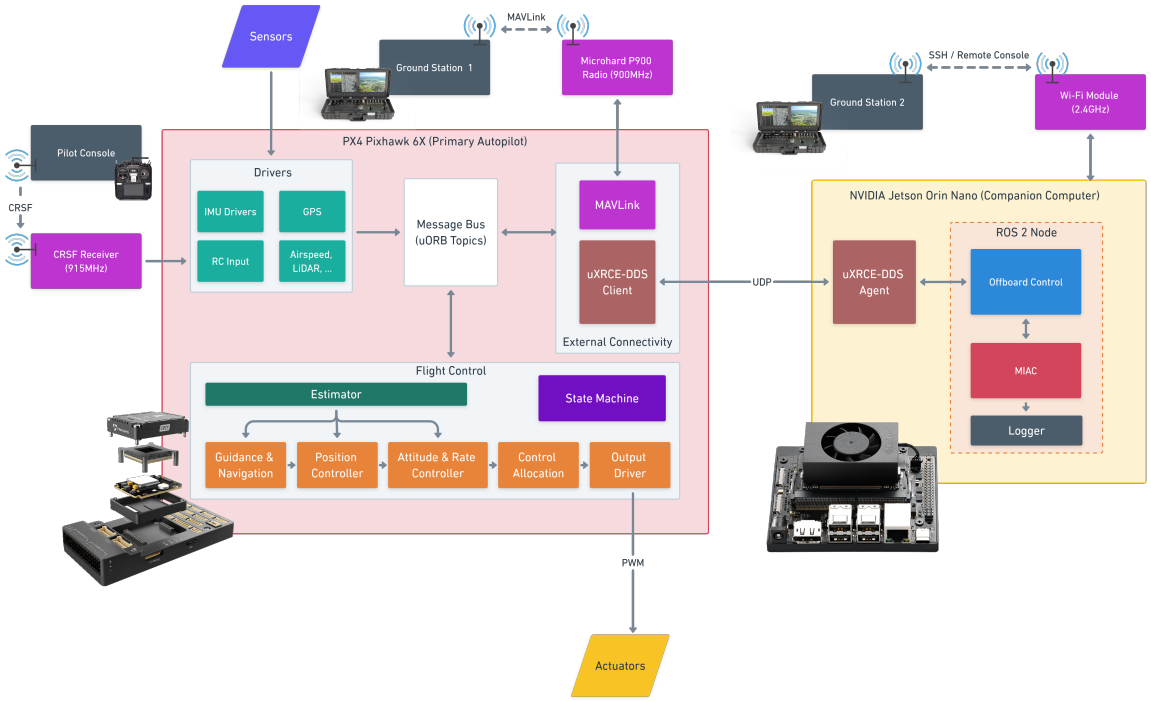


Figure 6.11: High-level hardware and software block diagram of the MIAC integrated with PX4.

Primary Autopilot. The primary autopilot is implemented using `PX4 v1.15.0-beta2` running on Pixhawk 6X hardware. PX4 provides a comprehensive flight control stack that includes:

- An on-board estimator that fuses data from IMU, GPS, airspeed sensors, LiDAR, etc., to compute position, velocity, and attitude.
- Guidance and navigation algorithms for autonomous stabilization, waypoint tracking, and automated takeoffs/landings.
- A full suite of sensor drivers and multiple flight modes (manual, assisted, and autonomous).
- A state machine that assigns control authority to the pilot, the internal PX4 controllers, or the MIAC.

PX4's built-in safety and fallback mechanisms ensure that it can promptly reclaim control if the MIAC exhibits unexpected behavior, thereby maintaining flight safety during tests.

Experimental Autopilot (MIAC). The MIAC is deployed on a Jetson Orin Nano companion computer running Ubuntu 22.04, which provides sufficient computational power for real-time model identification, adaptive control, and flight logging. Communication with the primary autopilot is established via a wired Ethernet link using UDP and the `micro XRCE-DDS` protocol. On the MIAC side, 3 processes are managed within a *ROS 2* node

- **Offboard Control Node**: Operating at 100 Hz, this node initiates MIAC functions and relays commands and data to the PX4 message bus (*uORB*) through the DDS bridge.
- **Model Identification and Adaptive Control (MIAC) Logic**: This module executes the perturbation, SID, and MPC-based adaptive control strategies.
- **Logging System**: It records detailed event logs, flight data (including perturbation signals), SID process outputs, MPC data, and archives the Model Library.

This configuration leverages PX4's full *uORB* interface, allowing the MIAC to both monitor sensor/estimator outputs and (when authorized) directly publish actuator commands, effectively bypassing PX4's internal controllers. Such direct access is essential for validating inner-loop control algorithms and would be difficult to achieve with more restrictive communication protocols.

Operator Interfaces and Ground Stations. Two ground stations facilitate mission management and development:

- **Ground Station 1**: Utilizes PX4's native C2 link for standard flight operations including takeoff, landing, waypoint navigation, and failsafe procedures.
- **Ground Station 2**: Connects to the Jetson Orin Nano over a 2.4 GHz WiFi link. This station provides an SSH-based console for rapid code updates, real-time monitoring of MIAC logs, and troubleshooting. Although intended for short-range development and debugging, it is critical for iterative testing.

Fallback Control Mechanism. Safety is further ensured by PX4's flight *State Machine*, which supervises active control authority. An operator at Ground Station 1 or a pilot using an RC transmitter can grant or revoke control from the MIAC.

The MIAC may voluntarily relinquish control if internal faults are detected, but it *cannot* seize control without explicit handover. In the event of a lost heartbeat from the Companion Computer, PX4 automatically reverts to its internal controllers or manual pilot control. This robust fallback mechanism is crucial for testing novel control logic in real-world scenarios.

Overall, the dual-autopilot architecture allows PX4 to manage vehicle-level safety, communication, and baseline functions, while the MIAC focuses on advanced adaptive control, real-time model identification, and comprehensive data collection. For further discussion on ROS 2 integration, the reader is referred to [183].

6.3.2 MIAC Logic Implementation for Testing

The MIAC logic underwent several modifications to support real-world flight tests, enhance safety, and fully exploit the dual-autopilot configuration. These modifications primarily focused on:

- Leveraging PX4's existing estimation and sensor fusion capabilities,
- Introducing new operational modes (e.g., an *idle* state), and
- Integrating robust safety checks.

Partial State Estimation in MIAC. PX4 already provides an Extended Kalman Filter (EKF2) that estimates key vehicle states. MIAC reuses these state estimates whenever possible. However, EKF2 does not supply certain information, such as body-axis velocity components (u, v, w) or state derivatives required for our SID and adaptive control algorithms. To bridge this gap, MIAC incorporates a lightweight estimation routine that supplements PX4's outputs with the missing parameters.

Idle State for Secondary Control. Since MIAC is not continuously the active controller, an *idle* operational state has been implemented. In the idle state, MIAC continues to:

- Listen to PX4 updates,
- Buffer sensor data,
- Execute internal logic (e.g., model identification and data logging),

but it does *not* publish actuator commands. In particular, the MPC module pauses its output during idle. This design enables a seamless transition from idle to active control as soon as the primary autopilot hands over authority, without requiring additional warm-up or initialization.

Manual Perturbation Injection. Originally, the MIAC *supervisor* automatically triggered flight perturbations for the SID process. For flight testing, these perturbations are now manually triggered by the pilot or ground station operator. This manual control allows the test team to precisely time perturbations, gather more meaningful data, and repeat the SID and validation cycles multiple times during a single flight.

Safety Limit Monitor. A critical new feature is the *Safety Limit Monitor*, which continuously checks flight conditions at 100 Hz whenever MIAC has control. By comparing real-time measurements of roll, pitch, and airspeed against predefined thresholds, the monitor ensures that the vehicle remains within safe operational limits. If any threshold is exceeded, the Safety Limit Monitor immediately:

- Relinquishes control back to PX4's internal controller,
- Halts the MIAC command outputs.

During initial test flights, an additional safeguard capped MIAC's control authority to 20 s intervals. Table 6.1 summarizes the safety limits used during these tests.

Table 6.1: Safety Limits Used During Initial Flight Tests

Parameter	**Value**
MAX_SPEED	21 m/s
MIN_SPEED	10 m/s
MAX_PITCH	+20°
MIN_PITCH	−25°
MAX_ROLL	40°

These stringent limits are essential for early "maiden" flights but are expected to be relaxed or removed for advanced performance tests once confidence in the MIAC algorithms increases.

Controller Handover Considerations. Given that MIAC and PX4 may exchange control authority multiple times during a flight, special procedures are implemented to ensure smooth transitions:

- **Controller Initialization:** When control is handed over, the newly active controller (either PX4 or MIAC) initializes its control loops using the most recent actuator commands and state measurements from the outgoing controller.
- **Filter Reset:** Integrators and filter states in the MPC and SID modules are re-initialized to minimize abrupt changes or "bumps" in the command signals.

These measures help prevent transient disturbances during control handovers, ensuring that the aircraft remains stable and safe at all times.

6.3.3 Flight Test Vehicle

For the initial flight trials of the MIAC system, we selected a cost-effective, off-the-shelf platform called *BAT* (Barebone Autopilot Tester). This subscale glider, depicted in Figure 6.12, offers several advantages for experimental controller development:

- **Low Kinetic Energy:** Its slow stall speed and foam construction minimize the risk of severe damage in the event of a crash.
- **Ease of Repair:** The use of off-the-shelf airframe components allows for rapid repairs and quick part replacements, reducing downtime.
- **Predictable Flight Characteristics:** The airframe's predictable flight characteristics facilitate tuning and streamline data collection during testing.

Airframe Specifications. The BAT is based on a commercially available ASW-28 subscale glider featuring a 2.6 m wingspan and a typical takeoff weight (TOW) of approximately 2.1 kg. In addition to the main wing, the fuselage is equipped with a puller folding propeller system for propulsion. Its off-the-shelf design enables rapid turnaround for repairs or replacements, making it an ideal platform for iterative control law development prior to testing on more complex UAVs.

Actuation and Sensors. The BAT features two independent ailerons, two independent flaps, one elevator, one rudder, and a folding propeller. These actuators are managed by the primary autopilot (Pixhawk 6X), which can hand over authority to the MIAC as needed. A comprehensive sensor suite provides essential flight data:

- **Air Data:** A pitot tube (MS5525DSO digital sensor) for airspeed measurement.
- **Altitude:** A Garmin LiDAR-Lite v3HP optical rangefinder for accurate altitude sensing.
- **Navigation:** A u-blox M9N GPS module, an external IST8310 compass, and a secondary internal BMM150 magnetometer.
- **Inertial Measurements:** Three ICM-45686 IMU chips (accelerometer and gyroscope) and two barometers (ICP20100 and BMP388) for redundant altitude measurements.

Additionally, an onboard first-person view (FPV) camera mounted on the tail provides live video feedback to a manual safety pilot.

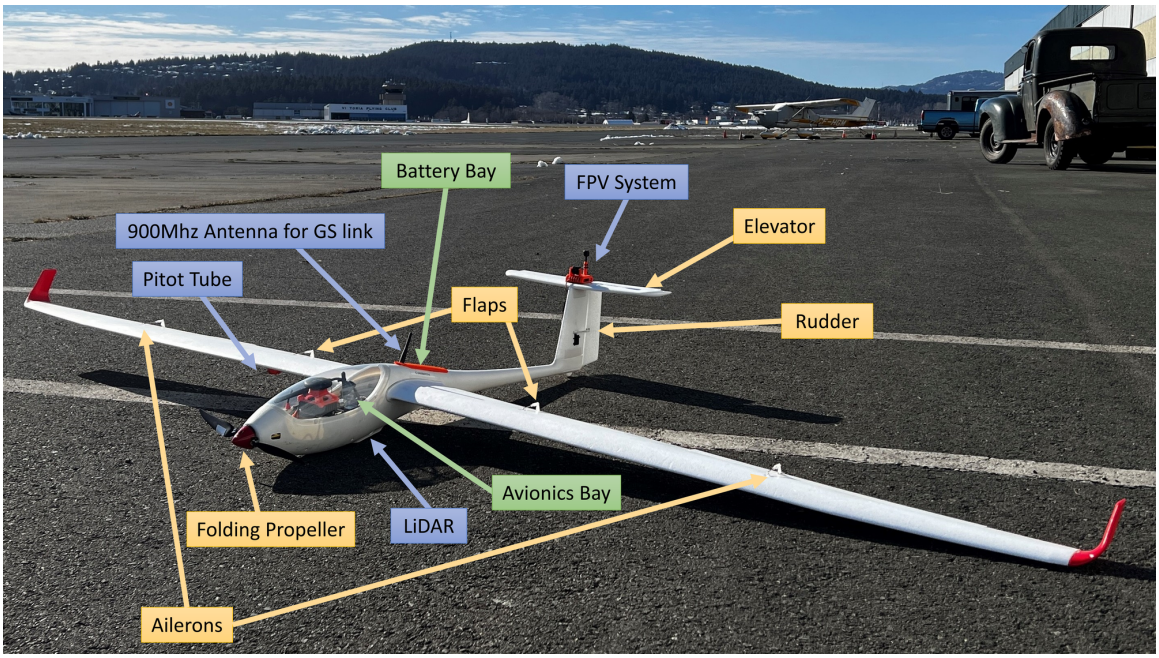


Figure 6.12: BAT UAV used for MIAC flight testing.

Avionics Architecture. Figure 6.13 shows a close-up of the avionics installed onboard. The Pixhawk 6X autopilot (running PX4 v1.15) and the NVIDIA Jetson Orin

Nano companion computer (hosting the MIAC software) reside in the main electronics bay. The Jetson and Pixhawk communicate via Ethernet using UDP. A Microhard P900 radio provides a long-range command-and-control (C2) link from Ground Station 1 to the Pixhawk, while dual 2.4 GHz Wi-Fi antennas on the Jetson enable a secondary link for MIAC monitoring and control from Ground Station 2. A Crossfire receiver connects to a manual pilot console to allow override control in emergencies. The forward bay houses the electronic speed controller (ESC) and the motor for the propeller.

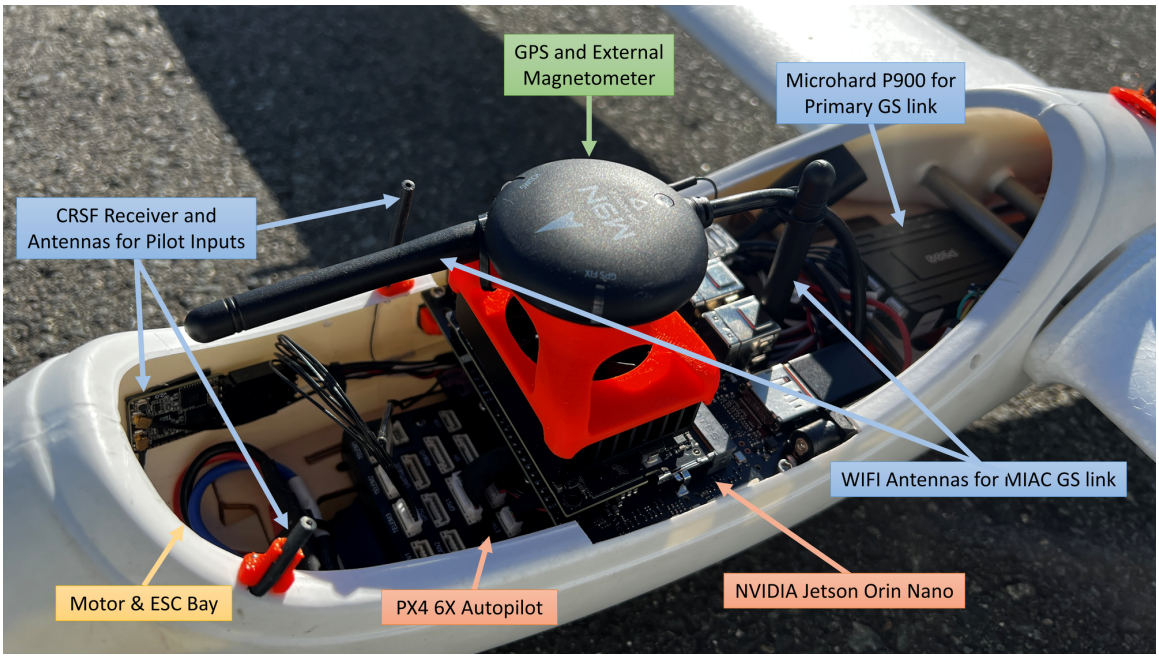


Figure 6.13: Avionics installation in the BAT UAV's main electronics bay.

Power System. All onboard systems are powered by a single 3S 2.2 Ah lithium-polymer (LiPo) battery, nominally rated at 11.1 V. Despite its compact size, this battery provides sufficient power to the autopilot, companion computer, and peripheral sensors for typical test flights lasting approximately 10 minutes.

6.3.4 Test Plan

The validation of the proposed MIAC architecture was performed in three distinct phases: *Simulation*, *Hardware-in-the-Loop* (HIL), and *Flight Testing*. The primary objective was to verify that the key functionalities of the MIAC rather than to exhaustively optimize performance.

Specifically, two major aspects were targeted:

- **Online System Identification (SID):** To demonstrate that MIAC can accurately and robustly identify the vehicle's dynamics in real time and update its internal model accordingly.
- **Adaptive MPC Control:** To validate that the model, once identified, is sufficiently accurate to enable the MPC to generate control commands that effectively track the desired setpoints.

Simulation Setup

Prior to hardware deployment, the MIAC algorithms were developed and tested in a simulation environment. In this stage:

- **Vehicles Simulated:** Two distinct platforms were simulated:

  1. The *BAT* platform.
  2. A novel VTOL UAV (Eusphyra) exhibiting more complex, highly nonlinear dynamics [79].

  Testing on these two airframes enhanced our confidence in the robustness and adaptability of the MIAC algorithms.

- **Model Sources and Validation:** The high-fidelity flight-dynamics model (FDM) in Simulink was developed using a traditional offline system identification procedure as described in [79]. This FDM captured aerodynamic behavior, actuator response, sensor noise, and communication delays, thereby providing a realistic simulation environment.
- **SID Tuning and Replay:** The simulation framework enabled the replay of logged flight data from previous experiments. This replay capability was used to tune the SID hyperparameters for improved noise robustness, faster convergence, and higher estimation accuracy.

Hardware-in-the-Loop

After verifying the algorithms in pure software simulation, the next phase involved hardware-in-the-loop (HIL) testing. In HIL:

- **Target Code Execution:** Both the PX4 autopilot stack (on Pixhawk 6X) and the MIAC software (on the Jetson Orin Nano) were deployed on their respective hardware platforms. The vehicle's avionics remained connected on a test bench while the actual flight dynamics were still simulated in Simulink.
- **Real-Time Performance Verification:** HIL tests confirmed that timing jitter, communication latencies, and CPU load on the Jetson did not compromise the control-loop performance.
- **Fallback Mechanisms Validation:** Full handover routines were exercised by switching between the primary autopilot, MIAC, and manual pilot override. Safety logic, including early termination of MIAC control and actuator reversion, was validated to ensure robust fallback in case of anomalies.

Flight Testing

Following successful simulation and HIL evaluations, flight testing was conducted using the BAT UAV. Flight tests were conducted incrementally, starting with simple checks and gradually progressing to more advanced maneuvers as confidence in each subsystem grew.

Figure 6.14 shows an example first-person view (FPV) captured during a flight test.

Figure 6.14: First-person view of the BAT during flight tests.

Flight Test Progression. Each flight session followed a structured procedure to minimize risk and ensure smooth transitions:

1. **Takeoff with the Primary Autopilot (PX4):** The vehicle climbs out using the proven PX4 autopilot to establish a stable initial flight.
2. **Holding Pattern:** The UAV enters a holding circuit (typically a rectangular or racetrack pattern with each leg lasting about 30 s) to provide a stable window for experimental maneuvers.
3. **Controller Handover to MIAC:** After a period of steady, level flight, control authority is transferred from PX4 to MIAC. During this handover, the pilot continuously monitors the flight data and retains immediate override capability.
4. **Execution of Perturbation Maneuvers:** MIAC injects predetermined perturbation signals (e.g., Multisine, Chirps, 3-2-1-1, and Doublets) to excite the system for SID. Data is collected during these maneuvers, and in subsequent passes, the models are validated using out-of-sample data.
5. **MPC Engagement:** Following successful SID, MIAC transitions to MPC-based control for tasks such as trim flight or executing specific maneuvers (e.g., altitude steps, speed changes, roll/pitch doublets).
6. **Handover Back to PX4 or Pilot:** After completing each test, control is handed back to PX4 or the manual pilot. Typical flight sessions lasted between 7 to 10 minutes or until approximately 80% of the battery capacity was used.
7. **Landing:** The primary autopilot regains control to safely land the UAV.

Additional Considerations. Several key factors were addressed during flight testing:

- **Safety Margins:** Early test flights employed conservative safety limits (e.g., tight roll/pitch/airspeed envelopes and a time cap on MIAC control segments) to minimize risk.
- **Data Logging:** Both PX4 and MIAC maintained detailed logs of state estimates, actuator commands, and internal model parameters. Synchronization of these logs was critical for post-flight analysis and validation of SID and MPC performance.

- **Progressive Complexity:** As confidence in the system increased, more aggressive perturbations and longer MIAC control segments were executed. Later flights also relaxed certain safety constraints and incorporated more dynamic maneuvers.

The next section summarizes the experimental results, highlighting the performance of the online system identification and the closed-loop adaptive MPC control achieved by the MIAC.

6.4 Results

In this section, we present data and plots from flight testing that demonstrate the functionality of the proposed MIAC architecture. The results are organized into three main categories: control perturbation, system identification, and model predictive control.

6.4.1 Control Perturbation

Figures 6.15, 6.16, 6.17, and 6.18 illustrate the state responses elicited by different perturbation signals.

Figure 6.15 shows the response to a multisine perturbation. Notably, the multisine signal, with its broad low-frequency content, excites the airspeed channel over a wider range. This enhanced excitation allows the system identification process to better capture the influence of airspeed on the aerodynamic coefficients.

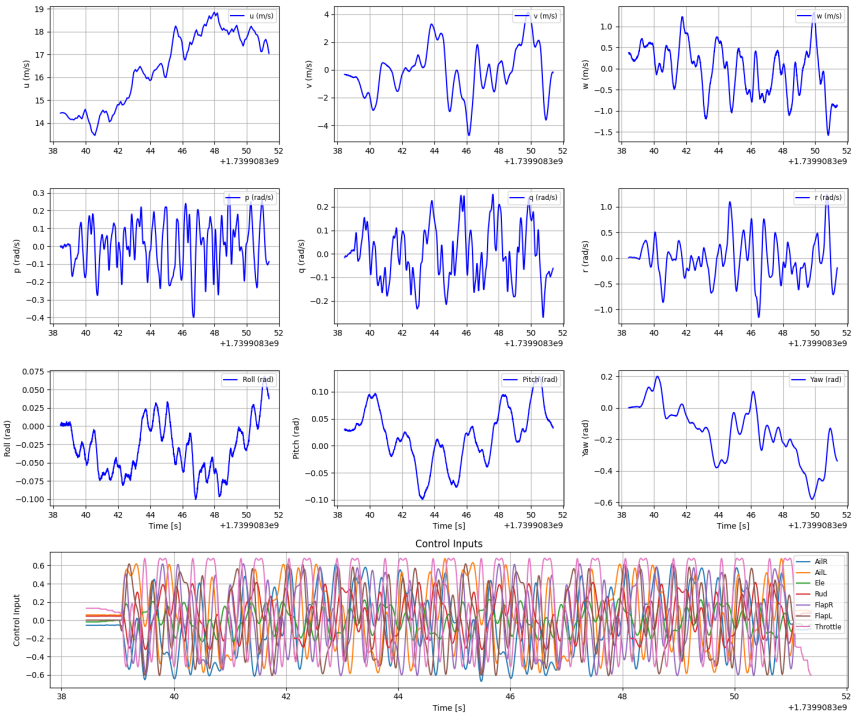


Figure 6.15: State response to a multisine perturbation.

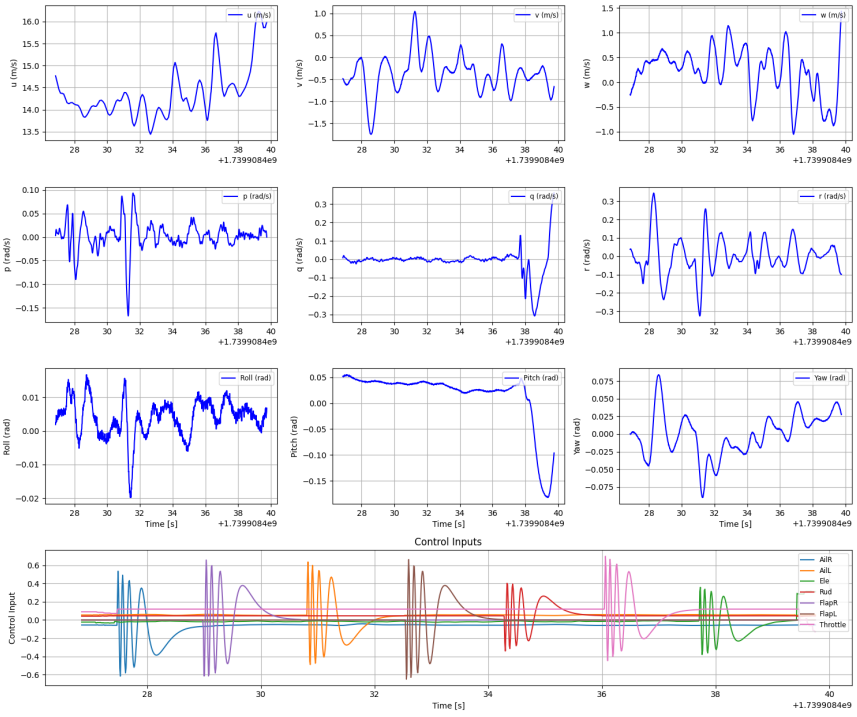


Figure 6.16: State response to a chirp sequence perturbation.

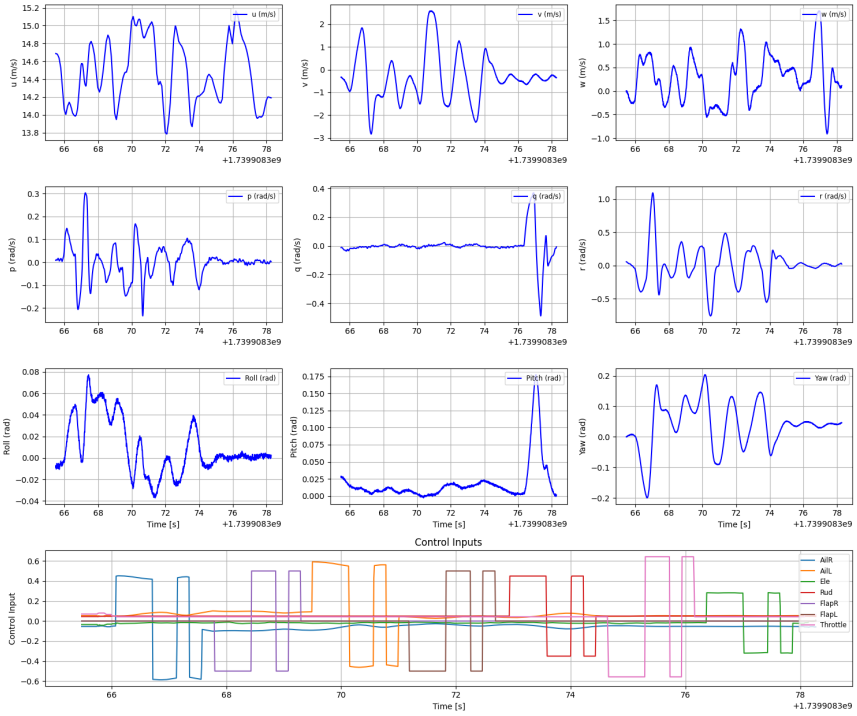


Figure 6.17: State response to a 3-2-1-1 sequence perturbation.

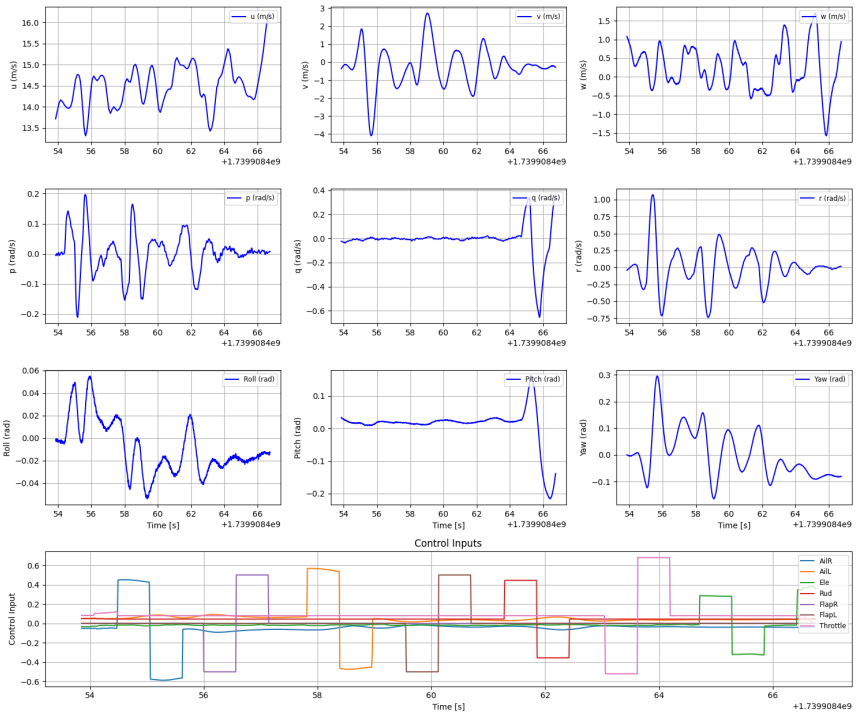


Figure 6.18: State response to a doublet perturbation.

Figure 6.16 presents the state response to a chirp sequence perturbation, while Figures 6.17 and 6.18 depict the responses to 3-2-1-1 and doublet perturbations, respectively.

It is important to note that, due to the sequential nature of the chirp, 3-2-1-1, and doublet perturbations, many flight segments do not continuously excite all states. While offline system identification can easily discard non-informative segments, an online SID process that must operate without prior knowledge of the actuator contributions cannot rely on selective data extraction.

We evaluated the excitation performance of the adaptive controller using two primary perturbations: multisine and chirp sequences. Tables 6.2 and 6.3 summarize the vehicle state and control command excitation metrics under multisine perturbations. The results indicate a significant increase in the perturbed standard deviations compared to baseline conditions, leading to high signal-to-noise ratios (SNR) across most channels.

Table 6.2: Vehicle State Excitation Analysis (Multisine)

Channel	Baseline STD	Perturbed STD	Signal STD	SNR (dB)	Baseline PTP	Perturbed PTP
u	0.4383	1.6176	1.5570	11.01	2.2110	5.3970
v	0.2320	1.7834	1.7682	17.64	1.2061	8.8486
w	0.4069	0.5838	0.4187	0.25	2.1815	2.9202
p	0.0096	0.1309	0.1306	22.70	0.0575	0.6906
q	0.0110	0.1155	0.1150	20.39	0.1348	0.5234
r	0.0426	0.4011	0.3988	19.43	0.2051	2.4019
roll	0.0043	0.0359	0.0356	18.38	0.0193	0.1684
pitch	0.0045	0.0525	0.0523	21.38	0.0327	0.2281
yaw	0.0268	0.1779	0.1759	16.36	0.1051	0.7803

Table 6.3: Control Command Perturbation Analysis (Multisine)

Channel	Baseline STD	Perturbed STD	Signal STD	SNR (dB)	Baseline PTP	Perturbed PTP
AilR	0.0028	0.3441	0.3441	41.65	0.0208	1.2967
AilL	0.0028	0.3418	0.3418	41.59	0.0208	1.2620
Ele	0.0156	0.1229	0.1219	17.86	0.1167	0.4717
Rud	0.0012	0.2043	0.2043	44.30	0.0069	0.7372
FlapR	0.0000	0.3404	0.3404	NaN	0.0000	1.2285
FlapL	0.0000	0.3404	0.3404	NaN	0.0000	1.2274
Throttle	0.0739	0.4163	0.4097	14.87	0.4191	1.2828

Tables 6.4 and 6.5 summarize the analysis for chirp excitation. In comparison to

multisine, chirp signals result in a more modest increase in standard deviations, with several channels exhibiting lower or even negative SNR values. Such results could adversely affect the quality of system identification.

Table 6.4: Vehicle State Excitation Analysis (Chirps)

Channel	Baseline STD	Perturbed STD	Signal STD	SNR (dB)	Baseline PTP	Perturbed PTP
u	0.4383	0.5759	0.3735	-1.39	2.2110	2.7900
v	0.2320	0.4369	0.3702	4.06	1.2061	2.7917
w	0.4069	0.4750	0.2450	-4.41	2.1815	2.4848
p	0.0096	0.0305	0.0289	9.62	0.0575	0.2604
q	0.0110	0.0738	0.0729	16.43	0.1348	0.6684
r	0.0426	0.0998	0.0902	6.52	0.2051	0.6692
roll	0.0043	0.0054	0.0033	-2.28	0.0193	0.0366
pitch	0.0045	0.0548	0.0546	21.74	0.0327	0.2373
yaw	0.0268	0.0305	0.0147	-5.17	0.1051	0.1742

Table 6.5: Control Command Perturbation Analysis (Chirps)

Channel	Baseline STD	Perturbed STD	Signal STD	SNR (dB)	Baseline PTP	Perturbed PTP
AilR	0.0028	0.0838	0.0837	29.37	0.0208	1.1521
AilL	0.0028	0.0846	0.0845	29.45	0.0208	1.1261
Ele	0.0156	0.0755	0.0739	13.50	0.1167	0.7384
Rud	0.0012	0.0557	0.0557	33.01	0.0069	0.7434
FlapR	0.0000	0.0973	0.0973	NaN	0.0000	1.2734
FlapL	0.0000	0.0973	0.0973	NaN	0.0000	1.3134
Throttle	0.0739	0.0868	0.0454	-4.24	0.4191	1.1461

Overall, the multisine excitation produced a more pronounced increase in both the signal magnitude and SNR compared to the chirp excitation, indicating its superiority for enhancing the excitation necessary for robust system identification.

6.4.2 SID Process

The SID process runs as a secondary thread within the MIAC framework. Once the data is collected, the SID computation is carried out whenever computational resources allow for it, therefore its execution may be delayed. On our desktop computer, the SID process takes an average of 0.463 seconds, while on the Jetson Orin Nano it requires approximately 2.485 seconds when the MPC is idle and up to 15

seconds when the MPC is running (this variability depends on the MPC settings and could most likely be avoided with proper resource management).

Table 6.6 reports the average fit scores (measured as R^2) and model complexity for four different command perturbation types: multisine, chirps, 3-2-1-1, and doublets. For each perturbation, we list the in-sample (training) fit, the out-of-sample (validation) fit, and the model complexity, defined as the number of active functions in the differential equation (i.e., the number of nonzero elements in the coefficient matrix $\boldsymbol{\Xi}$). In these tests, a validation maneuver consisting of a doublet (with 12 seconds of training data) was used. Note that variations in flight conditions may cause deviations in these numbers, and a total of 14 perturbations does not necessarily constitute a statistically significant sample. All models were identified using the same Constrained SR3 method with an l_0 regularization function.

Table 6.6: Average Fit Scores (R^2) and Model Complexity for Different Command Perturbations

Perturbation Type	**Training R^2**	**Out-of-Sample R^2**	**Model Complexity**
Multisine	0.8053	0.7831	63
Chirps	0.6757	0.7090	62
3211	0.8205	0.7508	59
Doublets	0.8518	0.7682	59

Among the perturbations, the multisine excitation consistently produces the highest out-of-sample fit. Although its training fit is slightly lower than some alternatives, the model derived from multisine data is more representative when it comes to prediction. In contrast, the chirp excitation yields the poorest overall performance, likely because the compressed chirp signal does not allow sufficient excitation, suggesting that a longer training duration might be necessary. The 3-2-1-1 and doublet maneuvers, while yielding high training fits due to their simplicity, show slightly lower out-of-sample performance. These high out-of-sample fit values are possibly because the validation maneuver (a doublet) does not explore as broad a range of frequencies or speeds as the multisine input does. It is also worth noting that the model complexity remains similar across perturbation types; this parameter is primarily influenced by the regularization hyperparameters and serves as an indicator of convergence quality (values that are too low or too high suggest poor convergence).

Equation (6.61) shows the differential equations of one SID model obtained from

a multisine perturbation.

$$
\begin{aligned}
\dot{u} &= r\,v - q\,w - g\,\sin(\theta) \\
&\quad - 0.100\,V + 0.006\,V^2 + 0.091\,\alpha\,V^2 + 0.188\,\alpha^2\,V^2 \\
&\quad - 0.027\,V^2\,\beta^2 - 0.202\,q\,V - 0.009\,q\,V^2 \\
&\quad + 0.008\,u_{\text{ele}}\,V^2 - 0.009\,u_{\text{flapR}}^2\,V^2 + 0.004\,u_{\text{thr}}\,V^2 \\
&\quad - 0.009\,u_{\text{flapL}}^2\,V^2 + 0.008\,u_{\text{thr}}^2\,V^2, \\
\dot{v} &= p\,w - r\,u + g\,\cos(\theta)\,\sin(\phi) \\
&\quad + 0.022\,\alpha\,V^2 - 0.025\,V^2\,\beta - 0.091\,p\,V \\
&\quad + 0.020\,p\,V^2 - 0.003\,q\,V^2 + 0.012\,r\,V - 0.056\,r\,V^2 \\
&\quad - 0.005\,u_{\text{rud}}\,V^2, \\
\dot{w} &= -0.075 + q\,u - p\,v + g\,\cos(\theta)\,\cos(\phi) \\
&\quad - 0.356\,V - 0.011\,V^2 - 0.308\,\alpha\,V^2 - 0.053\,\alpha^2\,V^2 \\
&\quad - 0.181\,q\,V + 0.074\,q\,V^2 - 0.003\,u_{\text{ele}}\,V \\
&\quad + 0.026\,u_{\text{ele}}\,V^2 - 0.044\,u_{\text{ele}}^2\,V^2 - 0.068\,u_{\text{thr}}^2\,V, \\
\dot{p} &= -0.200\,q\,r - 0.889\,p\,V + 0.055\,p\,V^2 + 0.161\,r\,V \\
&\quad + 0.002\,r\,V^2 + 0.005\,u_{\text{ailR}}\,V^2 - 0.004\,u_{\text{ailL}}\,V^2 \\
&\quad - 0.003\,u_{\text{rud}}\,V^2 - 0.001\,u_{\text{flapR}}\,V^2 + 0.002\,u_{\text{flapL}}\,V^2, \\
\dot{q} &= 0.201\,p\,r - 0.234\,q\,V + 0.005\,q\,V^2 + 0.021\,u_{\text{ele}}\,V^2 \\
&\quad - 0.022\,u_{\text{ele}}^2\,V^2, \\
\dot{r} &= -0.020\,p\,q + 0.026\,V^2\,\beta - 0.066\,V^2\,\beta^2 - 1.157\,p\,V \\
&\quad - 0.051\,r\,V + 0.007\,r\,V^2 + 0.002\,u_{\text{ailR}}\,V^2 \\
&\quad - 0.002\,u_{\text{ailL}}\,V^2 + 0.011\,u_{\text{rud}}\,V^2 + 0.001\,u_{\text{ailR}}^2\,V^2, \\
\dot{\phi} &= p + q\,\sin(\phi)\,\tan(\theta) + r\,\cos(\phi)\,\tan(\theta), \\
\dot{\theta} &= q\,\cos(\phi) - r\,\sin(\phi), \\
\dot{\psi} &= q\,\sin(\phi)\frac{1}{\cos(\theta)} + r\,\cos(\phi)\frac{1}{\cos(\theta)}.
\end{aligned}
\tag{6.61}
$$

Several observations can be made from this model. For example, the roll rate is influenced not only by the ailerons but also strongly by the rudder, indicating a pronounced yaw-roll coupling typical of gliders. Similarly, the presence of multiple actuator terms in the $\dot{u}$ equation reflects the contributions of throttle, flaps (which contribute drag, as indicated by their negative coefficients), and the coupling between elevator input and forward acceleration. The rudder appears in the $\dot{v}$ equation as expected, since rudder deflections produce side forces on the vertical tail that drive lateral motion. Although throttle is nominally aligned with the fuselage, its influence in the $\dot{w}$ equation suggests that misalignment or induced flows (e.g., prop wash) generate vertical force components. The multiplication of many coefficients by V or V^2 aligns with the aerodynamic force scaling with dynamic pressure. The presence of control inputs in seemingly unexpected equations (e.g., elevator in $\dot{u}$ and ailerons in $\dot{r}$) underscores the inherently coupled nature of real flight dynamics.

From these coefficients, key vehicle parameters such as the inertias can be estimated. For instance, the coefficients in front of qr, pr, and pq relate to the inertia differences according to:

$$\frac{I_{zz} - I_{yy}}{I_{xx}}, \quad \frac{I_{xx} - I_{zz}}{I_{yy}}, \quad \frac{I_{yy} - I_{xx}}{I_{zz}},$$

respectively. Moreover, knowing the vehicle mass m, wing span b, mean aerodynamic chord (MAC), wing area $S = b \times \text{MAC}$, and air density ρ, one can compute the aerodynamic coefficients by appropriately scaling the identified coefficients. For the parameters in $\dot{u}$, $\dot{v}$, and $\dot{w}$, the scaling factor is $\frac{\rho S}{2m}$; for $\dot{p}$ it is $\frac{\rho S b}{2I_{xx}}$; for $\dot{q}$ it is $\frac{\rho S\,\text{MAC}}{2I_{yy}}$; and for $\dot{r}$ it is $\frac{\rho S b}{2I_{zz}}$.

The delay estimation process applied to this model yielded the delays summarized in Table 6.7.

Table 6.7: Estimated Command Delays and Corresponding States for Actuators

Actuator	**Delay (s)**	**State Used**
AilR	0.13	p
AilL	0.13	p
Ele	0.09	q
Rud	0.11	$\dot{v}$
FlapR	0.15	r
FlapL	0.15	r
Throttle	0.13	$\dot{u}$

It is worth noting that in some cases the elevator shows higher correlation with the vertical rate ($\dot{w}$) and the rudder with the yaw rate (r). Although the delay identification process works well for most actuators, the delays for the flaps are more irregular. This irregularity likely arises because the dynamic effect of the flaps is lower, resulting in weaker correlations that are more sensitive to noise. Furthermore, the current delay resolution is limited by the step size of the resampled data (0.01 s, corresponding to 100 Hz); increasing this resolution could further improve delay estimation accuracy.

Figure 6.19 illustrates the online SID model's prediction of the state rates ($\dot{x}$) on validation data (out-of-sample) for a model identified using a multisine perturbation. This prediction is used by the model supervisor to compute the model fitness.

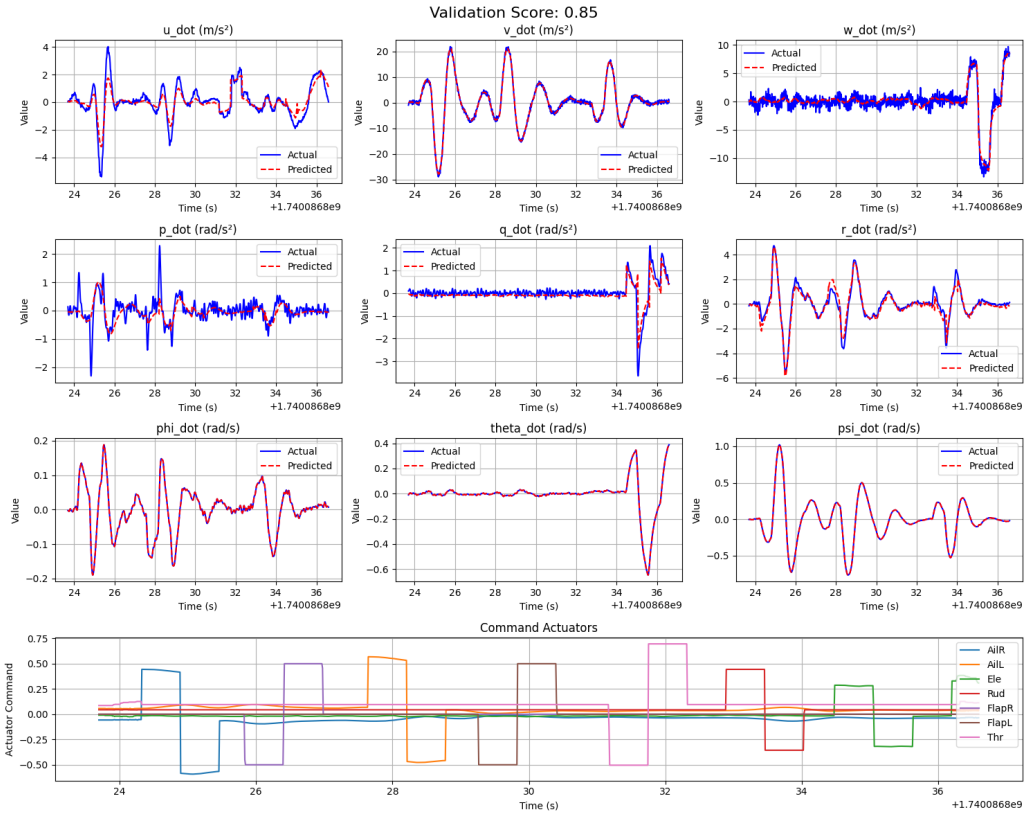


Figure 6.19: Prediction of the state rates $\dot{x}$ generated online by a SID model during validation.

Figure 6.20 shows the vehicle state prediction obtained by integrating the SID model forward in time from a given initial state, compared with the actual flight data. While the short-term predictions closely follow the measured states, disturbances such as turbulence and other unmodeled effects cause the prediction to drift over longer time horizons. Since the MPC uses a simulation time horizon of only 1 to 2 seconds, the short-term predictive capability of the SID model is sufficient for our application.

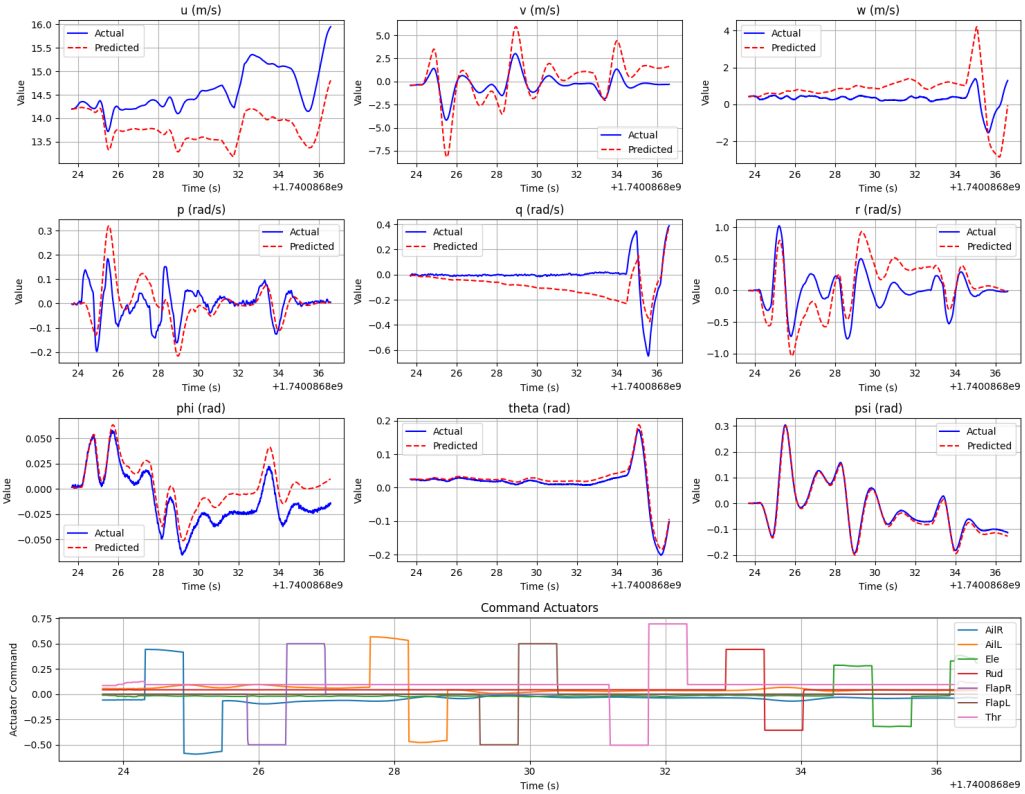


Figure 6.20: Forward integration of the SID model from an initial state, compared to actual flight data.

Figure 6.21 presents the evolution of selected aerodynamic coefficients over successive SID iterations starting from the reference model. The reference model, though it initially exhibited a relatively low average fit (approximately 0.28), had coefficients within the correct ballpark. A good initial guess significantly aids convergence. Some coefficients that were initially inaccurate or absent (such as the αV^2 term in $\dot{w}$) required several SID iterations to stabilize. Overall, most model coefficients remained stable across multiple SID updates, providing confidence in the repeatability of the SID process. An ideal future test would involve an abrupt change in the vehicle properties to verify that the SID process can reliably detect and adapt to such changes. One limitation, however, is that when the identified model structure changes between SID runs, direct one-to-one comparisons of aerodynamic coefficients may not be feasible.

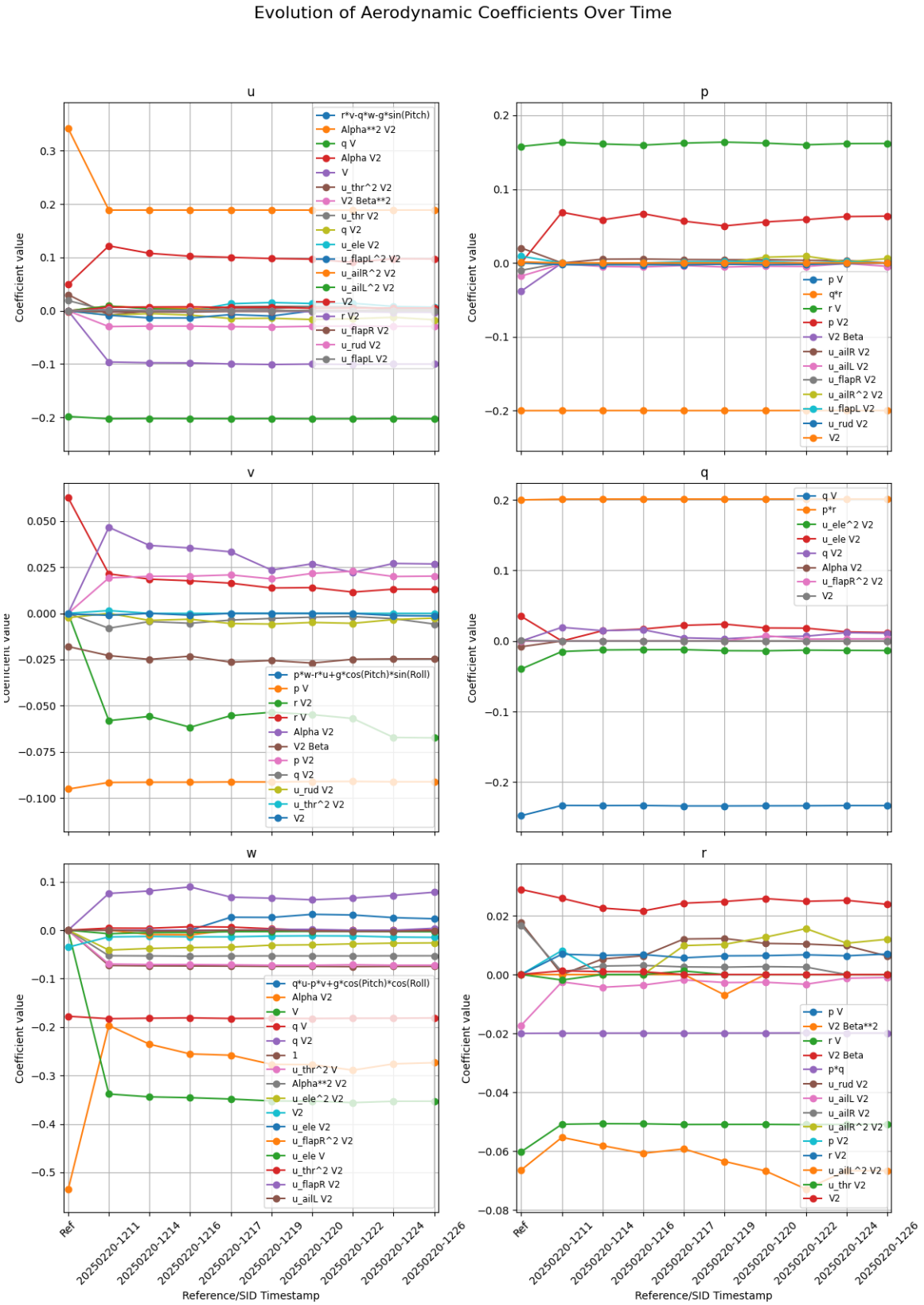


Figure 6.21: Model coefficient evolution over time.

Finally, onboard video footage revealed that the BAT UAV's wings exhibited noticeable flexing during aileron commands, likely due to the wing spar not extending along the full wing span. The current function library does not capture this behavior. To model wing flex, additional sensors (such as strain gauges, fiber optic sensors, or accelerometers in the wing) would be required to measure wing deflection. Similar approaches have been applied by Fonzi et al. [184], representing a promising direction for future work.

6.4.3 MPC

For the flight tests, the MPC frequency was fixed at 20 Hz to ensure that it would not consume excessive computational resources and disrupt other critical processes. However, monitoring of the MPC step time (which includes filtering, state estimation, simulation, and optimization) revealed an average execution time of 26 ± 2 ms on bench test and higher execution time of 47 ± 5 ms during later flight tests. This means that with the flight testing setup the MPC would sometime lag behind the expected frequency rate expected causing the test to be interrupted. Below are reported only cases in which this phenomena did not occur but this limitation should be addressed in future work.

Figure 6.22 shows the vehicle states and control commands during a trim flight segment when the MPC is in control. In the displayed flight segment, control transitions occur as follows: the primary autopilot (PX4) is initially in control; control is then transferred to the experimental autopilot to perform a doublet excitation for model validation; control reverts briefly to PX4; after about 10 s, control is again handed over to MIAC running MPC; and finally, control is returned to PX4. In the figures, the blue background indicates periods when PX4 is in control, while a green background denotes when the experimental autopilot is active (either during perturbation or while running MPC).

Figure 6.22: MPC test for trim flight with doublet validation and PX4 handovers.

Focusing on the second green segment (when MPC is in control) it is evident from the roll and pitch plots that the MPC achieves superior trim performance compared to the primary autopilot. The pitch remains nearly constant, and the roll quickly returns to its zero setpoint. Although some noise and minor perturbations are present, the overall state tracking is notably more steady. The estimation of the body-axis speed u via wind measurements is not highly accurate and the small UAV's airspeed sensor is inherently noisy; however, the MPC rapidly drives the airspeed toward the 14 m/s target, maintaining this value with relative stability. Additionally, the lateral speed v quickly converges to zero under MPC control, effectively eliminating sideslip. This effect is also visible in the reconstructed sideslip angle β plot. In contrast, the primary autopilot does not actively control the sideslip angle—the rudder control loop is open-loop for roll coordination—while the MPC trims the rudder command much more effectively. Moreover, analysis of the control command signals reveals that MPC outputs are smoother and less reactive to noise, with the rudder command rapidly converging to a new trim point to reduce sideslip.

Figure 6.23 shows the MPC performance during a roll doublet test, using future-

aware setpoints. The roll command steps to $+5°$ then $-5°$ (approximately ± 0.087 rad). The high penalty weights on the control command in the MPC cost function yield a smooth, relaxed response.

Figure 6.23: MPC test for roll attitude doublet with doublet validation and PX4 handovers.

Figure 6.24 displays the MPC response during a speed doublet, where the speed setpoint is commanded to change from 14 m/s to 15.5 m/s and then to 12.5 m/s. To maintain a fixed pitch during this test, the MPC optimizer relies solely on throttle adjustments. Interestingly, the flaps are also actuated to help reduce speed during the deceleration phase, and the angle of attack α correspondingly follows the changes in speed.

Figure 6.24: MPC test of speed doublet with doublet validation and PX4 handovers.

Finally, Figure 6.25 illustrates the MPC performance during a pitch doublet test. Here, the pitch is commanded to step $\pm 3^\circ$ (approximately ± 0.052 rad) from its trim state of $+0.025$ rad. Due to the unmodified cost function weights, the pitch doublet competes with the airspeed component of the cost function. Consequently, the vehicle is required to change pitch while maintaining steady airspeed, and the throttle is adjusted accordingly. The throttle response is less aggressive than that of the elevator, so slight variations in airspeed are observed. Additionally, the MPC slightly adjusts the pitch to prevent excessive deviations in airspeed. The lateral speed v is effectively regulated to zero, and the reconstructed sideslip angle β confirms minimal sideslip.

Figure 6.25: MPC test of pitch attitude with doublet validation and PX4 handovers.

6.5 Discussion

In this manuscript, we proposed an architecture for model identification adaptive control (MIAC) that integrates sparse identification of nonlinear dynamics with model predictive control (MPC). The entire system was developed initially in simulation, then deployed on hardware, and finally flight tested, thereby demonstrating its practical viability under real-world conditions.

Our experimental results indicate that the online system identification process can capture the vehicle's dynamics in real time with sufficient accuracy. Moreover, the adaptive MPC controller successfully leverages the identified model to generate control inputs that maintain the desired flight performance. Nonetheless, several challenges emerged during development. One major challenge was the reconstruction of the body-axis velocities u, v, and w. The approach of integrating accelerometer measurements and correcting biases using GPS data and wind estimates proved to be sensitive to noise, which limits the fidelity of the reconstructed velocities. While

the obtained results were acceptable, the accuracy is fundamentally constrained by the quality of the wind estimation. An alternative that employs true airspeed (TAS) along with aerodynamic angles (α and β) may provide improvements; however, such an approach would require additional sensors that are not typically available on small-scale UAVs.

Another challenge involved the estimation of angular accelerations. Deriving these accelerations from gyroscope data introduces considerable noise, necessitating heavy filtering. The lack of dedicated angular accelerometers in most autopilot IMU packages further complicates this task. Despite these difficulties, our simulation tests demonstrated that the system could achieve acceptable performance even when state estimation noise was increased.

The system's robustness is notably compromised under high-turbulence conditions. Although steady wind effects can be effectively filtered out, high levels of turbulence often prevent the SID process from generating a model with a sufficiently good fit to maintain control. In such cases, it may be necessary to develop methods to isolate or subtract turbulence effects to avoid degrading model updates.

The multisine perturbation approach, while effective for the seven actuators tested, may encounter issues when scaled to platforms with a larger number of actuators, due to insufficient frequency spacing. Mitigation strategies such as grouping actuators and employing ensemble methods could be considered. In addition, the requirement for a minimum training duration to capture slow dynamics (e.g., longitudinal body speed) limits how rapidly new models can be generated. The current architecture also presupposes that the vehicle can sustain flight for several seconds to accommodate perturbation and model identification, an assumption that might not hold for highly unstable platforms; in such cases, a shift toward closed-loop perturbation strategies would be warranted.

Computational demands, particularly for the MPC controller, further constrain performance. The current implementation in Python, although well-suited for rapid prototyping, is already constraining some of the test performed with a simpler model. Reimplementation in a compiled language such as C or C++ could be necessary to meet higher controller rate requirements.

Looking forward, several avenues for future work emerge. Testing the supervisor block in actual flight scenarios by deliberately altering aerodynamic properties in-flight (such as shifting the center of gravity or simulating an actuator fault) would provide valuable insights into the system's ability to detect and adapt to dynamic

changes. Enhancing the perturbation strategy so that the amplitude and frequency parameters adjust automatically based on the current model could further optimize system excitation. Incorporating advanced SID techniques, such as ensemble methods for both training data and parameter estimation, may enhance robustness and accuracy. Additional sensors, such as position encoders and RPM sensors, could help refine delay estimation and actuator model identification. Moreover, exploring alternative state representations (e.g., using TAS, α, and β instead of u, v, and w) might reduce estimation errors and improve wind rejection. Finally, advanced MPC techniques, including model order reduction and more refined constraint formulations, may enable longer prediction horizons without compromising computational performance.

In conclusion, the MIAC architecture demonstrates substantial promise as a tool for rapid prototyping of adaptive control systems. Although it is not yet at the stage of completely replacing conventional autopilots, its ability to serve as a secondary control system for rapid model updates and detailed flight data collection is invaluable. The integration of online system identification with MPC provides a flexible and adaptive framework that can be further improved with future developments in sensor technology, computational hardware, and advanced control methodologies.

Chapter 7

Conclusions and Future Work

This dissertation presented a systematic exploration of development, modeling, tuning, and control strategies for novel UAV configurations, ultimately establishing a rapid prototyping environment for innovative VTOL designs. Adopting a thesis-by-publication approach, the work has contributed in key areas such as vehicle design, system identification, controller tuning, and adaptive control methodologies. This chapter synthesizes the research findings by summarizing the contributions, discussing overarching challenges and limitations, outlining potential avenues for future work, and providing final remarks on the overall impact of this study.

7.1 Summary of Contributions

The research comprises a series of interconnected studies that collectively address the challenges inherent in developing advanced UAV systems. The initial investigation examined the airworthiness and configuration of a tilt tri-rotor system for fixed-wing VTOL applications. Through rigorous design, simulation, ground testing, and flight experiments, a robust understanding of vehicle dynamics during both hovering and transition to forward flight was achieved. This study established the feasibility of a smooth transition mechanism via rotor tilting, characterizing the vehicle's performance under varying wind conditions.

Building on this, a high-fidelity flight dynamics model was developed for the primary vehicle tasked with testing a magnetometer in a mock-up mission. Utilizing limited onboard sensor data and state-of-the-art system identification techniques, the research successfully captured the complex interactions between propulsive and

aerodynamic forces. This model not only served as a reliable baseline for controller development and tuning but also demonstrated that the vehicle could safely accommodate sensitive payloads.

Subsequently, an automatic framework for offline autopilot tuning was introduced. By combining genetic algorithms with system identification methods, the framework reliably identified transfer functions for roll, pitch, and yaw dynamics, optimizing the controller tuning and leading to notable enhancements in vehicle performance. The method proved versatile and adaptable across different vehicles and control laws, significantly reducing the workload on control engineers.

Advancing into adaptive control, the concept of a Model Identification Adaptive Controller (MIAC) for UAVs was proposed. This architecture employed an online system identification process based on sparse identification of nonlinear dynamics, enabling real-time updates of the dynamic model. Insights into model fitness as influenced by training time and noise levels were obtained, paving the way for integrating model-based control with adaptive strategies.

Finally, the integration of the online identification method with a Model Predictive Control (MPC) scheme was validated through simulations, hardware-in-the-loop tests, and flight experiments. Despite encountering several challenges, the MIAC framework demonstrated its capability to capture essential vehicle dynamics in real time, underscoring its potential as a platform for rapid prototyping and adaptive control for future UAV applications.

7.2 Limitations and Challenges

Each component of this research presented its own set of challenges and limitations. A central issue was the difficulty of executing online or fully autonomous system identification (SID) without human intervention. This necessitated additional robustness checks and validation logic to ensure convergence, which, while effective, sometimes resulted in suboptimal performance compared to manual, ad hoc tuning approaches.

The SID method developed in controlled, offline scenarios yielded superior results but required intensive supervision and was specifically tailored to the test vehicle. Applying the same approach to different configurations would demand significant additional data and manpower. In both cases, turbulence and gust-induced disturbances posed major challenges, often necessitating the rejection of compromised data. Future implementations may benefit from enhanced sensor suites (e.g., aeroprobes for

measuring flow angles) to better isolate environmental effects.

Additional limitations were observed in state reconstruction, where inaccurate estimates of body velocities and noise in angular acceleration measurements constrained the accuracy of the SID. Transitioning to alternative state representations (e.g., true airspeed and flow angles) and incorporating dedicated sensors could mitigate these issues. Moreover, many experiments were conducted in open-loop, which is feasible for stable fixed-wing platforms but would require redesigning perturbation strategies for more unstable vehicles (closed-loop).

Computational limitations also restricted the MPC controller's execution speed and time horizon. While acceptable for the test platform, more unstable configurations may demand higher update rates. Finally, the MIAC framework—by dynamically adjusting control laws online—introduces an inherent risk, as convergence of the adaptive process cannot always be guaranteed. This risk, along with the challenge of ensuring safe fallback strategies, remains an area for further refinement.

7.3 Future Research Directions

Several promising avenues for future research emerge from this work. The ongoing Eusphyra project will focus on integrating the tested tri-rotor VTOL strategies into a full-scale vehicle and exploring hybrid propulsion techniques to enhance power and endurance.

Refinements to the automatic autopilot tuning architecture could involve parallelization to expedite the optimization process and broaden the design space, as well as integrating classical stability analysis directly into the tuning process. Further improvements in SID robustness might be achieved through ensemble methods, adaptive perturbation strategies, and alternative state representations that mitigate the impact of noise.

Additional experimental validation of the MIAC framework (particularly through fault-injection scenarios such as actuator failures or aerodynamic shifts) would provide critical insights into the system's resilience. Moreover, incorporating actuator feedback sensors could enhance the accuracy of actuator modeling and enable rapid fault detection.

Transitioning the MIAC implementation from Python to a compiled language (such as C or C++) may also improve its real-time performance, making it more suitable for embedded flight controllers. Finally, expanding the application of MIAC

to a broader range of UAV configurations, including unstable or multi-actuator systems, will help establish its scalability and potential as an augmentation layer within conventional autopilot architectures.

7.4 Final Remarks

In closing, this dissertation has significantly advanced the state of the art in control development and tuning for UAVs in a rapid prototyping environment. The contributions outlined in this work not only validate the innovative methodologies developed and tested in real flight conditions but also establish a robust foundation for future research in UAV technologies. While challenges remain (particularly in the domains of autonomous system identification and robust control) the insights gained here provide clear directions for overcoming these challenges. Ultimately, the integration of advanced modeling, control tuning, and adaptive strategies promises to enhance the performance, reliability, and safety of next-generation UAVs, driving further innovation in the field.

Appendix A

Reference Model

The reference model is used to prime the MIAC architecture with an initial guess of the aerodynamic, mass/inertia, and control surface effectiveness coefficients before any online system identification (SID) is performed. Even a rough initial model—obtained via a CFD analysis or based on a flight dynamics model (FDM) of a similarly sized UAV [79]—can significantly improve the convergence of the subsequent SID regression.

In the following, we present the vehicle dynamics in a form that isolates inertial contributions from aerodynamic (and, where applicable, thrust) effects. First, the total airspeed V and the flow angles are defined as

$$V = \sqrt{u^2 + v^2 + w^2}, \quad \alpha = \arctan\left(\frac{w}{u}\right), \quad \beta = \arcsin\left(\frac{v}{V}\right). \tag{A.1}$$

The explicit vehicle dynamics of the FDM are given by

$$\begin{cases} \dot{u} &= r\,v - q\,w + \frac{X}{m} - g\sin\theta, \\ \dot{v} &= p\,w - r\,u + \frac{Y}{m} + g\cos\theta\sin\phi, \\ \dot{w} &= q\,u - p\,v + \frac{Z}{m} + g\cos\theta\cos\phi, \\ \dot{p} &= \frac{L}{I_{xx}} - \frac{(I_{zz}-I_{yy})}{I_{xx}}\,q\,r - \frac{I_p}{I_{xx}}\,\dot{\Omega}_p, \\ \dot{q} &= \frac{M}{I_{yy}} - \frac{(I_{xx}-I_{zz})}{I_{yy}}\,p\,r - \frac{I_p}{I_{yy}}\,\Omega_p\,r, \\ \dot{r} &= \frac{N}{I_{zz}} - \frac{(I_{yy}-I_{xx})}{I_{zz}}\,p\,q + \frac{I_p}{I_{zz}}\,\Omega_p\,q, \\ \dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta, \\ \dot{\theta} &= q\cos\phi - r\sin\phi, \\ \dot{\psi} &= \frac{q\sin\phi + r\cos\phi}{\cos\theta}. \end{cases} \tag{A.2}$$

The aerodynamic forces and moments acting on the vehicle are modeled in non-dimensional form. The aerodynamic force vector $\boldsymbol{F}_{aero}$ and moment vector $\boldsymbol{M}_{aero}$ are expressed as

$$\boldsymbol{F}_{aero} = \bar{q}S \begin{bmatrix} C_X\big(\boldsymbol{V},\boldsymbol{\omega},\dot{\boldsymbol{V}},\boldsymbol{\delta}\big) \\ C_Y\big(\boldsymbol{V},\boldsymbol{\omega},\dot{\boldsymbol{V}},\boldsymbol{\delta}\big) \\ C_Z\big(\boldsymbol{V},\boldsymbol{\omega},\dot{\boldsymbol{V}},\boldsymbol{\delta}\big) \end{bmatrix}, \quad \boldsymbol{M}_{aero} = \bar{q}S \begin{bmatrix} b\,C_l\big(\boldsymbol{V},\dot{\boldsymbol{V}},\boldsymbol{\omega},\boldsymbol{\delta}\big) \\ \bar{c}\,C_m\big(\boldsymbol{V},\boldsymbol{\omega},\dot{\boldsymbol{V}},\boldsymbol{\delta}\big) \\ b\,C_n\big(\boldsymbol{V},\boldsymbol{\omega},\dot{\boldsymbol{V}},\boldsymbol{\delta}\big) \end{bmatrix}, \tag{A.3}$$

where $\bar{q} = \frac{1}{2}\rho V^2$ is the dynamic pressure, S is the wing area, b is the wing span, and $\bar{c}$ is the mean aerodynamic chord.

The non-dimensional aerodynamic coefficients are expressed as follows:

$$\begin{aligned} C_X &= C_{X_\alpha}\alpha + C_{X_{\alpha^2}}\alpha^2 + C_{X_q}\hat{q} + C_{X_{\delta_c}}\delta_c + C_{X_{\mathcal{J}_c}}\mathcal{J}_c + C_{X_{\mathcal{J}_c^2}}\mathcal{J}_c^2 + C_{X_0}, \\ C_Y &= C_{Y_\beta}\beta + C_{Y_{\delta_r}}\delta_r + C_{Y_0}, \\ C_Z &= C_{Z_\alpha}\alpha + C_{Z_q}\hat{q} + C_{Z_{\mathcal{J}_c}}\mathcal{J}_c + C_{Z_{\mathcal{J}_c^2}}\mathcal{J}_c^2 + C_{Z_0}, \\ C_l &= C_{l_\beta}\beta + C_{l_p}\hat{p} + C_{l_r}\hat{r} + C_{l_{\delta_a}}\delta_a + C_{l_{r\,\delta_a^2}}\,\hat{r}\,\delta_a^2 + C_{l_0}, \\ C_m &= C_{m_\alpha}\alpha + C_{m_q}\hat{q} + C_{m_{\delta_c}}\delta_c + C_{m_{\alpha\,\delta_c}}\,\alpha\,\delta_c + C_{m_0}, \\ C_n &= C_{n_\beta}\beta + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r + C_{n_{\delta_r^2\,\mathcal{J}_c}}\,\delta_r^2\mathcal{J}_c + C_{n_0}. \end{aligned} \tag{A.4}$$

Here, the terms $\hat{p}$, $\hat{q}$, and $\hat{r}$ are the non-dimensional angular velocities defined as

$$\hat{p} = \frac{p\,b}{2V}, \quad \hat{q} = \frac{q\,\bar{c}}{2V}, \quad \hat{r} = \frac{r\,b}{2V}. \tag{A.5}$$

The governing equations for the vehicle dynamics can then be expressed explicitly by substituting the aerodynamic forces and moments into the equations of motion:

$$
\begin{cases}
\dot{u} = & r\,v - q\,w + \frac{\rho S}{2m}\Big(C_{X_\alpha}\,\alpha\,V^2 + C_{X_{\alpha^2}}\,\alpha^2\,V^2 + C_{X_q}\,\frac{\bar{c}}{2}\,q\,V + C_{X_{\delta_c}}\,\delta_c\,V^2 \\
& \quad + C_{X_{\mathcal{J}_c}}\,\mathcal{J}_c\,V^2 + C_{X_{\mathcal{J}_c^2}}\,\mathcal{J}_c^2\,V^2 + C_{X_0}\,V^2\Big) - g\,\sin\theta, \\
\dot{v} = & p\,w - r\,u + \frac{\rho S}{2m}\Big(C_{Y_\beta}\,\beta\,V^2 + C_{Y_{\delta_r}}\,\delta_r\,V^2 + C_{Y_0}\,V^2\Big) + g\,\cos\theta\,\sin\phi, \\
\dot{w} = & q\,u - p\,v + \frac{\rho S}{2m}\Big(C_{Z_\alpha}\,\alpha\,V^2 + C_{Z_q}\,\frac{\bar{c}}{2}\,q\,V + C_{Z_{\mathcal{J}_c}}\,\mathcal{J}_c\,V^2 \\
& \quad + C_{Z_{\mathcal{J}_c^2}}\,\mathcal{J}_c^2\,V^2 + C_{Z_0}\,V^2\Big) + g\,\cos\theta\,\cos\phi, \\
\dot{p} = & \frac{\rho S b}{2I_{xx}}\Big(C_{l_\beta}\,\beta\,V^2 + C_{l_p}\,\frac{b}{2}\,p\,V + C_{l_r}\,\frac{b}{2}\,r\,V + C_{l_{\delta_a}}\,\delta_a\,V^2 \\
& \quad + C_{l_{r\delta_a^2}}\,\frac{b}{2}\,r\,\delta_a^2\,V + C_{l_0}\,V^2\Big) - \frac{I_{zz}-I_{yy}}{I_{xx}}\,q\,r - \frac{I_p}{I_{xx}}\,\dot{\Omega}_p, \\
\dot{q} = & \frac{\rho S \bar{c}}{2I_{yy}}\Big(C_{m_\alpha}\,\alpha\,V^2 + C_{m_q}\,\frac{\bar{c}}{2}\,q\,V + C_{m_{\delta_c}}\,\delta_c\,V^2 + C_{m_{\alpha\delta_c}}\,\alpha\,\delta_c\,V^2 \\
& \quad + C_{m_0}\,V^2\Big) - \frac{I_{xx}-I_{zz}}{I_{yy}}\,p\,r - \frac{I_p}{I_{yy}}\,\Omega_p\,r, \\
\dot{r} = & \frac{\rho S b}{2I_{zz}}\Big(C_{n_\beta}\,\beta\,V^2 + C_{n_{\delta_a}}\,\delta_a\,V^2 + C_{n_{\delta_r}}\,\delta_r\,V^2 + C_{n_{\delta_r^2\mathcal{J}_c}}\,\delta_r^2\,\mathcal{J}_c\,V^2 \\
& \quad + C_{n_0}\,V^2\Big) - \frac{I_{yy}-I_{xx}}{I_{zz}}\,p\,q + \frac{I_p}{I_{zz}}\,\Omega_p\,q, \\
\dot{\phi} = & p + q\,\sin\phi\,\tan\theta + r\,\cos\phi\,\tan\theta, \\
\dot{\theta} = & q\,\cos\phi - r\,\sin\phi, \\
\dot{\psi} = & q\,\sin\phi\,\sec\theta + r\,\cos\phi\,\sec\theta.
\end{cases}
\tag{A.6}
$$

From the coefficients in these equations, the reference model is defined. Table A.1 lists the numerical values of the non-dimensional aerodynamic coefficients as derived from the FDM process. In our analysis, the following constants were used:

$$
g = 9.80665\ \text{m/s}^2, \quad m = 2.106\ \text{kg}, \quad b = 2.555\ \text{m}, \quad \bar{c} = 0.160\ \text{m},
$$

$$I_{xx} = 1.159 \text{ kgm}^2, \quad I_{yy} = 1.2301 \text{ kgm}^2, \quad I_{zz} = 2.3864 \text{ kgm}^2, \quad ,$$

$$S = 0.4088 \text{ m}^2, \quad \rho = 1.225 \text{ kg/m}^3,$$

and we define the scaling factors:

$$rS = \frac{\rho S}{2m}, \quad rSb = \frac{\rho S\, b}{2}, \quad rSCIy = \frac{\rho S\, \bar{c}}{2I_{yy}}.$$

Table A.1: Coefficients in the Reference Model (approximate numerical values)

State	**Term**	**Coefficient**	**Value (approx.)**
u	V^2	$rS\, C_{X0}$	$0.2504 \times (-0.00499) \approx -0.00125$
	$\alpha\, V^2$	$rS\, C_{Xa}$	$0.2504 \times 0.19635 \approx 0.04916$
	$\alpha^2\, V^2$	$rS\, C_{Xa2}$	$0.2504 \times 1.36622 \approx 0.34167$
	$q\, V$	$rS\, \frac{\bar{c}}{2}\, C_{X_{dc}}$	$0.2504 \times 0.08 \times (-9.89184) \approx -0.198$
	$u_{thr}\, V^2$	$rS\, C_{u,thr}$	$0.2504 \times 0.07633 \approx 0.01912$
	$u_{thr}^2\, V^2$	$rS\, C_{u,thr^2}$	$0.2504 \times 0.12124 \approx 0.03030$
v	V^2	$rS\, C_{Y0}$	$0.2504 \times (-0.00145) \approx -0.00036$
	$\beta\, V^2$	$rS\, C_{Y\beta}$	$0.2504 \times (-0.07180) \approx -0.01795$
	$p\, V$	$rS\, \frac{b}{2}\, C_{Yp}$	$0.2504 \times 1.2775 \times (-0.29753) \approx -0.09517$
	$r\, V$	$rS\, \frac{b}{2}\, C_{Yr}$	$0.2504 \times 1.2775 \times 0.19579 \approx 0.06253$
	$u_{rud}\, V^2$	$rS\, C_{Ydr}$	$0.2504 \times (-0.00937) \approx -0.00234$
w	V^2	$rS\, C_{Z0}$	$0.2504 \times (-0.14010) \approx -0.03503$
	$\alpha\, V^2$	$rS\, C_{Za}$	$0.2504 \times (-2.13810) \approx -0.53497$
	$q\, V$	$rS\, \frac{\bar{c}}{2}\, C_{Zq}$	$0.2504 \times 0.08 \times (-8.87196) \approx -0.17777$

Continued on next page

Table A.1 – Continued

State	Term	Coefficient Symbol	Value (approx.)
p	$q\,r$	$-\frac{I_{zz}-I_{yy}}{I_{xx}}$	-0.9977
	V^2	$rSb\,\frac{C_{l0}}{I_{xx}}$	$0.6396 \times 0.19309 \approx 0.1235$
	$\beta\,V^2$	$rSb\,\frac{C_{l\beta}}{I_{xx}}$	$0.6396 \times 0.03176 \approx 0.0203$
	$p\,V$	$rSb\,\frac{b}{2}\,\frac{C_{lp}}{I_{xx}}$	$0.6396 \times 1.2775 \times (-0.02776) \approx -0.02266$
	$r\,V$	$rSb\,\frac{b}{2}\,\frac{C_{lr}}{I_{xx}}$	$0.6396 \times 1.2775 \times (-0.01554) \approx -0.01269$
	$u_{ailR}\,V^2$	$rSb\,\frac{C_{ldaR}}{I_{xx}}$	$0.6396 \times 0.01425 \approx 0.00911$
	$u_{ailL}\,V^2$	$rSb\,\frac{C_{ldaL}}{I_{xx}}$	$0.6396 \times 9.4447 \times 10^{-5} \approx 0.00006$
	$u_{flapR}\,V^2$	$rSb\,\frac{C_{ldflR}}{I_{xx}}$	$0.6396 \times 0.04518 \approx 0.02887$
	$u_{flapL}\,V^2$	$rSb\,\frac{C_{ldflL}}{I_{xx}}$	$0.6396 \times (-0.10382) \approx -0.06642$
q	$p\,r$	$-\frac{I_{xx}-I_{zz}}{I_{yy}}$	$+0.9978$
	V^2	$rSCIy\,C_{m0}$	$0.04006 \times 0.01975 \approx 0.00079$
	$\alpha\,V^2$	$rSCIy\,C_{ma}$	$0.04006 \times (-0.20172) \approx -0.00807$
	$q\,V$	$rSCIy\,\frac{\bar{c}}{2}\,C_{mq}$	$0.04006 \times 0.08 \times (-77.23817) \approx -0.2475$
	$u_{ele}\,V^2$	$rSCIy\,C_{m,de}$	$0.04006 \times 0.87292 \approx 0.03490$
	$u_{ele}^2\,V^2$	$rSCIy\,C_{m,de^2}$	$0.04006 \times (-0.98853) \approx -0.03960$
r	$p\,q$	$-\frac{I_{yy}-I_{xx}}{I_{zz}}$	-0.0298
	V^2	$rSb\,\frac{C_{n0}}{I_{zz}}$	$0.6396 \times 9.4447 \times 10^{-5} \approx 0.00006$
	$\beta\,V^2$	$rSb\,\frac{C_{n\beta}}{I_{zz}}$	$0.6396 \times 0.04518 \approx 0.02890$
	$\beta^2\,V^2$	$rSb\,\frac{C_{n\beta^2}}{I_{zz}}$	$0.6396 \times (-0.10382) \approx -0.06641$
	$p\,V$	$rSb\,\frac{b}{2}\,\frac{C_{np}}{I_{zz}}$	$0.6396 \times 1.2775 \times (-1.41586) \approx -1.1570$
	$r\,V$	$rSb\,\frac{b}{2}\,\frac{C_{nr}}{I_{zz}}$	$0.6396 \times 1.2775 \times (-0.07370) \approx -0.06030$
	$u_{rud}\,V^2$	$rSb\,\frac{C_{ndr}}{I_{zz}}$	$0.6396 \times 0.02756 \approx 0.01762$
	$u_{ailR}\,V^2$	$rSb\,\frac{C_{ndaR}}{I_{zz}}$	$0.6396 \times 0.02608 \approx 0.01666$
	$u_{ailL}\,V^2$	$rSb\,\frac{C_{ndaL}}{I_{zz}}$	$0.6396 \times (-0.02700) \approx -0.01727$

From the coefficients listed in Table A.1, the sparse coefficient matrix $\boldsymbol{\Xi}$ is generated. This matrix is then used by the MIAC architecture to initialize the subsequent system identification regression.

Appendix B

SID Library of Functions

The library of functions $\boldsymbol{\Theta}(\mathbf{x}, \mathbf{u})$ comprises all user-defined functions used in the system identification process to capture the vehicle dynamics. In our framework, combinations of these functions form the basis for the governing equations identified by the optimization routine.

States and Control Inputs The available states are:

$$u, \quad v, \quad w, \quad p, \quad q, \quad r, \quad \phi, \quad \theta, \quad \psi \tag{B.1}$$

The control inputs are:

$$\text{AilR}, \quad \text{AilL}, \quad \text{Elev}, \quad \text{Rud}, \quad \text{FlapR}, \quad \text{FlapL}, \quad \text{Thr} \tag{B.2}$$

Derived States Additional algebraic equations are incorporated to introduce derived states:

$$\begin{aligned} V &= \sqrt{u^2 + v^2 + w^2}, \\ \alpha &= \arctan\left(\frac{w}{u}\right), \\ \beta &= \arcsin\left(\frac{v}{V}\right). \end{aligned} \tag{B.3}$$

Base Library Components In schematic terms, the base library for the SID is organized as follows:

$$
\begin{aligned}
\textbf{Bias term:}\quad & 1, \\
\textbf{Linear Velocities:}\quad & u,\ v,\ w, \\
\textbf{Angular Rates:}\quad & p,\ q,\ r, \\
\textbf{Control Inputs:}\quad & u_{ailR},\ u_{ailL},\ u_{ele},\ u_{rud},\ u_{flapR},\ u_{flapL},\ u_{thr}, \\
& u_{ailR}^2,\ u_{ailL}^2,\ u_{ele}^2,\ u_{rud}^2,\ u_{flapR}^2,\ u_{flapL}^2,\ u_{thr}^2, \\
\textbf{Kinematic Terms:}\quad & \sin(\phi)\tan(\theta),\ \cos(\phi)\tan(\theta),\ \cos(\phi),\ \sin(\phi), \\
& \frac{q\sin(\phi)}{\cos(\theta)},\ \frac{r\cos(\phi)}{\cos(\theta)}, \\
\textbf{Force Equation Terms:}\quad & rv - qw - g\sin(\theta), pw - ru + g\cos(\theta)\sin(\phi), \\
& qu - pv + g\cos(\theta)\cos(\phi), \\
\textbf{Moment Equations:}\quad & pq,\ pr,\ qr,\ \left(p^2 - r^2\right), \\
\textbf{Angle of Attack:}\quad & \alpha,\ \alpha^2, \\
\textbf{Airspeed:}\quad & V,\ V^2, \\
\textbf{Angle of Side-Slip:}\quad & \beta,\ \beta^2.
\end{aligned}
\tag{B.4}
$$

Tensoring (Cross Terms) The following base components are allowed to be tensored to form cross terms:

$$
\begin{aligned}
&\text{(Angle of Attack)} \times \text{(Airspeed)}, \\
&\text{(Angular Rates)} \times \text{(Airspeed)}, \\
&\text{(Control inputs)} \times \text{(Airspeed)}, \\
&\text{(Airspeed)} \times \text{(Beta)}.
\end{aligned}
\tag{B.5}
$$

Complete Library of Functions The resulting 79 base functions, after tensoring to include cross terms, were concatenated and formatted as a single library for system identification. The full library is given in Table B.1.

In total, the library $\boldsymbol{\Theta}(\mathbf{x}, \mathbf{u})$ comprises 711 elements (79×9).

Table B.1: Library of Functions used in the SID Process

Functions (1-40)	**Functions (41-79)**
1	u
v	w
p	q
r	u_{ailR}
u_{ailL}	u_{ele}
u_{rud}	u_{flapR}
u_{flapL}	u_{thr}
u_{ailR}^2	u_{ailL}^2
u_{ele}^2	u_{rud}^2
u_{flapR}^2	u_{flapL}^2
u_{thr}^2	$p + q\sin(\text{Roll})\tan(\text{Pitch}) + r\cos(\text{Roll})\tan(\text{Pitch})$
$q\cos(\text{Roll}) - r\sin(\text{Roll})$	$\dfrac{q\sin(\text{Roll})}{\cos(\text{Pitch})} + \dfrac{r\cos(\text{Roll})}{\cos(\text{Pitch})}$
$r\,v - q\,w - g\sin(\text{Pitch})$	$p\,w - r\,u + g\cos(\text{Pitch})\sin(\text{Roll})$
$q\,u - p\,v + g\cos(\text{Pitch})\cos(\text{Roll})$	$p\,q$
$p\,r$	$q\,r$
$(p^2 - r^2)$	α
α^2	V
V^2	β
β^2	$\alpha\,V$
$\alpha\,V^2$	$\alpha^2\,V$
$\alpha^2\,V^2$	$V\,\beta$
$V\,\beta^2$	$V^2\,\beta$
$V^2\,\beta^2$	$p\,V$
$p\,V^2$	$q\,V$
$q\,V^2$	$r\,V$
$r\,V^2$	$u_{\text{ailR}}\,V$
$u_{\text{ailR}}\,V^2$	$u_{\text{ailL}}\,V$
$u_{\text{ailL}}\,V^2$	$u_{\text{ele}}\,V$
$u_{\text{ele}}\,V^2$	$u_{\text{rud}}\,V$
$u_{\text{rud}}\,V^2$	$u_{\text{flapR}}\,V$
$u_{\text{flapR}}\,V^2$	$u_{\text{flapL}}\,V$
$u_{\text{flapL}}\,V^2$	$u_{\text{thr}}\,V$
$u_{\text{thr}}\,V^2$	$u_{\text{ailR}}^2\,V$
$u_{\text{ailR}}^2\,V^2$	$u_{\text{ailL}}^2\,V$
$u_{\text{ailL}}^2\,V^2$	$u_{\text{ele}}^2\,V$
$u_{\text{ele}}^2\,V^2$	$u_{\text{rud}}^2\,V$
$u_{\text{rud}}^2\,V^2$	$u_{\text{flapR}}^2\,V$
$u_{\text{flapR}}^2\,V^2$	$u_{\text{flapL}}^2\,V$
$u_{\text{flapL}}^2\,V^2$	$u_{\text{thr}}^2\,V$
$u_{\text{thr}}^2\,V^2$	

Bibliography

[1] Amina Khan, Sumeet Gupta, and Sachin Kumar Gupta. "Emerging UAV Technology for Disaster Detection, Mitigation, Response, and Preparedness". In: *Journal of Field Robotics* (2022). DOI: 10.1002/rob.22075.

[2] Fabio Andrade et al. "Autonomous Unmanned Aerial Vehicles in Search and Rescue Missions Using Real-Time Cooperative Model Predictive Control". In: *Sensors* (2019). DOI: 10.3390/s19194067.

[3] F.L. Lewis. *Applied Optimal Control and Estimation*. Pearson Technology Group, 1992, Chapter 1: Introduction to Modern Control Theory.

[4] R.V. Jategaonkar et al. "Identification of C-160 Simulator Data Base from Flight Data". In: *IFAC Proceedings Volumes* 27 (8 1994), pp. 1031–1038. DOI: 10.1016/s1474-6670(17)47844-x.

[5] B. David McNally. "Full Envelope Aerodynamic Modeling of the Harrier Aircraft." In: *Proceedings, Annual Symposium - Society of Flight Test Engineers* (1986).

[6] Kris Ellis et al. "Beechjet flight test data gathering and Level-D simulator aerodynamic mathematical model development". In: *AIAA Atmospheric Flight Mechanics Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2001. DOI: doi:10.2514/6.2001-4012. URL: https://doi.org/10.2514/6.2001-4012.

[7] Susanne Seher-Weiss. "Identification of nonlinear aerodynamic derivatives using classical and extended local model networks". In: *Aerospace Science and Technology* 15 (1 2011), pp. 33–44. DOI: 10.1016/j.ast.2010.06.002. URL: http://dx.doi.org/10.1016/j.ast.2010.06.002.

[8] Eric L Tobias et al. “Army Aviation Development Directorate-AFDD”. In: *Flight Control Technology, Member AIAA. † Flight Control Technology Group Leader, Senior Scientist* (), pp. 1–38. URL: https://pdfs.semanticscholar.org/debb/5882b9bea5f580f1f775232709b0a9d82e98.pdf.

[9] Eugene A. Morelli. “Practical aspects of real-time modeling for the learn-to-fly concept”. In: *2018 Atmospheric Flight Mechanics Conference* (December 2018). DOI: 10.2514/6.2018-3309.

[10] Bruce Owens, David Cox, and Eugene Morelli. “Development of a Low-Cost Sub-Scale Aircraft for Flight Research: The FASER Project”. In: *25th AIAA Aerodynamic Measurement Technology and Ground Testing Conference*. American Institute of Aeronautics and Astronautics, June 2006. DOI: 10.2514/6.2006-3306. URL: https://arc.aiaa.org/doi/10.2514/6.2006-3306.

[11] Hairol N M Shah et al. “Develop and Design Small Scale UAV”. In: *Modern Applied Science* 17 (2 2023), p. 49. DOI: 10.5539/mas.v17n2p49.

[12] Guowei Cai, Jorge Dias, and Lakmal Seneviratne. “A Survey of Small-Scale Unmanned Aerial Vehicles: Recent Advances and Future Development Trends”. In: *Unmanned Systems* 02 (02 2014), pp. 175–199. DOI: 10.1142/s2301385014300017.

[13] Or D. Dantsker, Mirco Theile, and Marco Caccamo. “A Cyber-Physical Prototyping and Testing Framework to Enable the Rapid Development of UAVs”. In: *Aerospace* 9 (5 May 2022), p. 270. DOI: 10.3390/aerospace9050270. URL: https://www.mdpi.com/2226-4310/9/5/270.

[14] Nino Krznar, Petar Piljek, and Zdenka Keran. “Multirotor UAV Design and Development – Case Study”. In: *Tehnicki Glasnik* 17 (4 2023), pp. 588–593. DOI: 10.31803/tg-20220909092242.

[15] Andy J Keane et al. “Maritime Flight Trials of the Southampton University Laser Sintered Aircraft–Project Albatross”. In: *The Aeronautical Journal* 121 (1244 2017), pp. 1502–1529. DOI: 10.1017/aer.2017.71.

[16] Zahari Taha, Yuan Tang, and Kok-Leong Yap. “Development of an Onboard System for Flight Data Collection of a Small-Scale UAV Helicopter”. In: *Mechatronics* 21 (1 2011), pp. 132–144. DOI: 10.1016/j.mechatronics.2010.09.008.

[17] Fadjar R Triputra et al. “Nonlinear Dynamic Modeling of a Fixed-Wing Unmanned Aerial Vehicle: A Case Study of Wulung”. In: *Mechatronics Electrical Power and Vehicular Technology* 6 (1 2015), pp. 19–30. DOI: 10.14203/j.mev.2015.v6.19-30.

[18] Chunyang Wang, Zhou Zhou, and Rui Wang. “Research on Dynamic Modeling and Transition Flight Strategy of VTOL UAV”. In: *Applied Sciences* 9 (22 2019), p. 4937. DOI: 10.3390/app9224937.

[19] Bogdan Løw-Hansen et al. “Time-Domain System Identification and Validation of Small Fixed-Wing UAV Dynamics”. In: *AIAA AVIATION FORUM AND ASCEND 2024*. American Institute of Aeronautics and Astronautics, July 2024. DOI: 10.2514/6.2024-4653. URL: https://arc.aiaa.org/doi/10.2514/6.2024-4653.

[20] Benjamin M Simmons et al. *System Identification of a Nonlinear Flight Dynamics Model for a Small, Fixed-Wing UAV*. Tech. rep. 2018.

[21] Andrei Dorobantu et al. “Frequency Domain System Identification for a Small, Low-Cost, Fixed-Wing UAV”. In: (2011). DOI: 10.2514/6.2011-6719.

[22] Eugene A. Morelli and Jared A. Grauer. “Advances in Aircraft System Identification at NASA Langley Research Center”. In: *Journal of Aircraft* 60 (5 Sept. 2023), pp. 1354–1370. DOI: 10.2514/1.c037274.

[23] Eugene Morelli. “System Identification Programs for AirCraft (SIDPAC)”. In: *AIAA Atmospheric Flight Mechanics Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2002. DOI: 10.2514/6.2002-4704.

[24] Mark B. Tischler and Robert K. Remple. *Aircraft and Rotorcraft System Identification, Second Edition*. American Institute of Aeronautics and Astronautics, Inc., Aug. 2012. DOI: 10.2514/4.868207. URL: https://arc.aiaa.org/doi/book/10.2514/4.868207.

[25] Chingiz Hajiyev, Demet Cilden-Guler, and Ulviye Hacizade. “Two-Stage Kalman Filter for Fault Tolerant Estimation of Wind Speed and UAV Flight Parameters”. In: *Measurement Science Review* 20 (1 Feb. 2020), pp. 35–42. DOI: 10.2478/msr-2020-0005.

[26] Jared Grauer, Jennifer Heeg, and Eugene Morelli. "Real-Time Frequency Response Estimation Using Joined-Wing SensorCraft Aeroelastic Wind-Tunnel Data". In: *AIAA Atmospheric Flight Mechanics Conference*. American Institute of Aeronautics and Astronautics, Aug. 2012. DOI: 10.2514/6.2012-4641.

[27] Matthew Holzel and Eugene Morelli. "Real-Time Frequency Response Estimation from Flight Data". In: *AIAA Atmospheric Flight Mechanics Conference*. American Institute of Aeronautics and Astronautics, Aug. 2011. DOI: 10.2514/6.2011-6358.

[28] Jared Grauer and Eugene Morelli. "Method for Real-Time Frequency Response and Uncertainty Estimation". In: *Journal of Guidance, Control, and Dynamics* 37 (1 Jan. 2014), pp. 336–344. DOI: 10.2514/1.60795.

[29] Jared A. Grauer. "Frequency Response Estimation for Multiple Aircraft Control Loops Using Orthogonal Phase-Optimized Multisine Inputs". In: *Processes* 10 (4 Mar. 2022), p. 619. DOI: 10.3390/pr10040619.

[30] Jared A. Grauer and Matthew Boucher. "System Identification of Flexible Aircraft: Lessons Learned from the X-56A Phase 1 Flight Tests". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-1017.

[31] Jared A Grauer and Matthew J. Boucher. "Aircraft System Identification from Multisine Inputs and Frequency Responses". In: (January 2020). DOI: 10.2514/6.2020-0287.

[32] Jared A. Grauer, Eugene A. Morelli, and Daniel G. Murri. "Flight-Test Techniques for Quantifying Pitch Rate and Angle-of-Attack Rate Dependencies". In: *Journal of Aircraft* 54 (6 Nov. 2017), pp. 2367–2377. DOI: 10.2514/1.C034407.

[33] Steven L Brunton and J Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. DOI: DOI:10.1017/9781108380690. URL: https://www.cambridge.org/core/books/datadriven-science-and-engineering/77D52B171B60A496EAFE4DB662ADC36E.

[34] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. "Dynamic mode decomposition with control". In: *SIAM Journal on Applied Dynamical Systems* 15 (1 2016), pp. 142–161. DOI: 10.1137/15M1013857.

[35] Steven L Brunton et al. "of of of of Dynamics Control Dynamics Control Control Control (SINDYc)". In: *IFAC-PapersOnLine* 49 (18 2016), pp. 710–715. DOI: 10.1016/j.ifacol.2016.10.249. URL: http://dx.doi.org/10.1016/j.ifacol.2016.10.249.

[36] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Sparse Identification of Nonlinear Dynamics with Control (SINDYc)". In: *IFAC-PapersOnLine* 49 (18 2016), pp. 710–715. DOI: 10.1016/j.ifacol.2016.10.249. URL: http://dx.doi.org/10.1016/j.ifacol.2016.10.249.

[37] E. Kaiser, J. N. Kutz, and S. L. Brunton. "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474 (2219 2018). DOI: 10.1098/rspa.2018.0335.

[38] Markus Quade et al. "Sparse identification of nonlinear dynamics for rapid model recovery". In: *Chaos* 28 (6 2018), pp. 1–10. DOI: 10.1063/1.5027470.

[39] U. Fasel et al. "Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 478 (2260 Apr. 2022). DOI: 10.1098/rspa.2021.0904.

[40] Jared A. Grauer. "A learn-to-fly approach for adaptively tuning flight control systems". In: *2018 Atmospheric Flight Mechanics Conference* (December 2018). DOI: 10.2514/6.2018-3312.

[41] Karl Åström. "History of Adaptive Control BT - Encyclopedia of Systems and Control". In: ed. by John Baillieul and Tariq Samad. Springer London, 2015, pp. 526–533. DOI: 10.1007/978-1-4471-5058-9_120. URL: https://doi.org/10.1007/978-1-4471-5058-9_120.

[42] D G Ward and Roger Barron. *Self-designing receding horizon optimal flight controller*. Vol. 5. July 1995, 3490–3494 vol.5. DOI: 10.1109/ACC.1995.533785.

[43] David G. Ward, Jeffrey F. Monaco, and Marc Bodson. "Development and flight testing of a parameter identification algorithm for reconfigurable control". In: *Journal of Guidance, Control, and Dynamics* 21 (6 1998), pp. 948–956. DOI: 10.2514/2.4329.

[44] Wayne C Durham and Kenneth A Bordignon. "Multiple control effector rate limiting". In: *Journal of Guidance, Control, and Dynamics* 19 (1 Jan. 1996), pp. 30–37. DOI: 10.2514/3.21576. URL: https://doi.org/10.2514/3.21576.

[45] John Bolling et al. "Control allocation with adaptive failure control". In: *22nd Atmospheric Flight Mechanics Conference*. American Institute of Aeronautics and Astronautics, Aug. 1997. DOI: doi:10.2514/6.1997-3775. URL: https://doi.org/10.2514/6.1997-3775.

[46] James Buffington. "Tailless aircraft control allocation". In: *Guidance, Navigation, and Control Conference* (Aug. 1997). DOI: doi:10.2514/6.1997-3605. URL: https://doi.org/10.2514/6.1997-3605.

[47] Joseph Brinker and Kevin Wise. "Reconfigurable flight control for a tailless advanced fighter aircraft". In: *Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 1998. DOI: doi:10.2514/6.1998-4107. URL: https://doi.org/10.2514/6.1998-4107.

[48] Byoung S Kim and Anthony J Calise. "Nonlinear Flight Control Using Neural Networks". In: *Journal of Guidance, Control, and Dynamics* 20 (1 Jan. 1997), pp. 26–33. DOI: 10.2514/2.4029. URL: https://doi.org/10.2514/2.4029.

[49] Youmin Zhang and Jin Jiang. "Bibliographical review on reconfigurable fault-tolerant control systems". In: *Annual Reviews in Control* 32 (2 2008), pp. 229–252. DOI: 10.1016/j.arcontrol.2008.03.008.

[50] Eugene Heim et al. "NASA's Learn-to-Fly Project Overview". In: (2020). URL: https://ntrs.nasa.gov/search.jsp?R=2019002721.

[51] Chengyu Cao and N Hovakimyan. "Design and Analysis of a Novel L1 Adaptive Controller, Part I: Control Signal and Asymptotic Stability". In: *2006 American Control Conference*. 2006, pp. 3397–3402. DOI: 10.1109/ACC.2006.1657243.

[52] Chengyu Cao and N Hovakimyan. "Design and Analysis of a Novel L1 Adaptive Controller, Part II: Guaranteed Transient Performance". In: *2006 American Control Conference*. 2006, pp. 3403–3408. DOI: 10.1109/ACC.2006.1657244.

[53] Steven Snyder. "L1 ADAPTIVE CONTROL WITHIN LEARN-TO-FLY". PhD thesis. University of Illinois at Urbana-Champaign, 2019.

[54] Kasey A Ackerman et al. "Evaluation of an L1 Adaptive Flight Control Law on Calspan's Variable-Stability Learjet". In: *Journal of Guidance, Control, and Dynamics* 40 (4 Apr. 2017), pp. 1051–1060. DOI: 10.2514/1.G001730. URL: https://doi.org/10.2514/1.G001730.

[55] Irene Gregory et al. "Flight Test of an L1 Adaptive Controller on the NASA AirSTAR Flight Test Vehicle". In: *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, Aug. 2010. DOI: doi:10.2514/6.2010-8015. URL: https://doi.org/10.2514/6.2010-8015.

[56] Romeo Ortega and Elena Panteley. "Adaptation is Unnecessary in L1-"Adaptive" Control". In: (September 2014), pp. 1–5. URL: http://arxiv.org/abs/1409.2389.

[57] Romeo Ortega and Elena Panteley. *L1-"Adaptive" control always converges to a linear pi control and does not perform better than the PI*. Vol. 19. IFAC, 2014, pp. 6926–6928. DOI: 10.3182/20140824-6-za-1003.00680. URL: http://dx.doi.org/10.3182/20140824-6-ZA-1003.00680.

[58] Eugene A Morelli. "Practical Aspects of Real-Time Modeling for the Learn-To-Fly Concept Practical Aspects of Real-Time Modeling for the Learn-to-Fly Concept". In: (December 2019). DOI: 10.2514/6.2018-3309.

[59] Jay M. Brandon and Eugene A. Morelli. *Real-time onboard global nonlinear aerodynamic modeling from flight data*. 2014, pp. 1–55. DOI: 10.2514/6.2014-2554.

[60] Steven Snyder et al. "Online control design for learn-to-fly". In: *2018 Atmospheric Flight Mechanics Conference* (2018), pp. 1–27. DOI: 10.2514/6.2018-3311.

[61] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An introduction*. Ed. by A Bradford Book. Second edi. Vol. 258. 2017, pp. 675–676. DOI: 10.1016/S0140-6736(51)92942-X.

[62] Adrian Carrio et al. "A review of deep learning methods and applications for unmanned aerial vehicles". In: *Journal of Sensors* 2017 (2017). DOI: 10.1155/2017/3296874.

[63] Tianhao Zhang et al. “Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search”. In: *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June (2016), pp. 528–535. DOI: 10.1109/ICRA.2016.7487175.

[64] Klaas Kelchtermans and Tinne Tuytelaars. “How hard is it to cross the room? – Training (Recurrent) Neural Networks to steer a UAV”. In: (2017). URL: http://arxiv.org/abs/1702.07600.

[65] Matteo Hessel et al. “Rainbow: Combining improvements in deep reinforcement learning”. In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018* (2018), pp. 3215–3222.

[66] Ajay Chander and Ramya Srinivasan. “Machine Learning and Knowledge Extraction”. In: *CD-MAKE: International Cross-Domain Conference for Machine Learning and Knowledge Extraction* 11015 (2018), pp. 314–328. DOI: 10.1007/978-3-319-99740-7. URL: http://dx.doi.org/10.1007/978-3-319-99740-7_23%0Ahttp://link.springer.com/10.1007/978-3-319-99740-7.

[67] Markus Quade, Thomas Isele, and Markus Abel. “Explainable Machine Learning Control – robust control and stability analysis”. In: (January 2020). URL: http://arxiv.org/abs/2001.10056.

[68] Perry Van Wesel and Alwyn E Goodloe. “Challenges in the Verification of Reinforcement Learning Algorithms NASA STI Program . . . in Profile”. In: (June 2017 2017), pp. 2017–219628. URL: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20170007190.pdf.

[69] Jared A. Grauer. “A learn-to-fly approach for adaptively tuning flight control systems”. In: *2018 Atmospheric Flight Mechanics Conference* (October 2018). DOI: 10.2514/6.2018-3312.

[70] Cody Daniel Robert Hansen. “Magnetic Signature Characterization of a Fixed-Wing Vertical Take-off and Landing (VTOL) Unmanned Aircraft Vehicle (UAV)”. PhD thesis. University of Victoria, Dec. 2018. URL: http://hdl.handle.net/1828/10413.

[71] Goncalo Ferreira Gameiro. “Design Modifications of a UAV Wing for Optimal Integration of a Magnetic Anomaly Detection Sensor”. PhD thesis. Instituto Superior Tecnico, Universidade de Lisboa, Oct. 2018. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/844820067126160/GGameiro_THESIS.pdf.

[72] Bruno Miguel Wong Luis. “Design and Development of a Magnetic Anomaly Detection Unmanned Aerial Vehicle”. PhD thesis. Instituto Superior Tecnico, Universidade de Lisboa, Nov. 2019. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997259488/Thesis_Bruno_Luis_80891.pdf.

[73] Sara de Almeida Pedro. “Sizing and Integration of an Electric Propulsion System for a VTOL UAV”. PhD thesis. Instituto Superior Tecnico, Universidade de Lisboa, Dec. 2020. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997262370/Resumo_alargado_83730.pdf.

[74] Diogo Beirão Valente Henriques Tomas. “Design, Development and Manufacture of a New VTOL, Canard UAV”. PhD thesis. Instituto Superior Tecnico, Universidade de Lisboa, Dec. 2020. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/1126295043838730/Thesis_83675.pdf.

[75] Tiago Alexandre Salreta de Jesus. “Surrogate-based Multidisciplinary Design Optimization of a Canard Configuration VTOL UAV”. PhD thesis. Instituto Superior Tecnico, Universidade de Lisboa, Dec. 2020. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997262753/Tese%20Mestrado_Tiago%20Jesus_83733.pdf.

[76] Pedro Miguel Micaelo Fernandes. “Development, Computational and Experimental Characterization of a Canard Aircraft Configuration”. PhD thesis. Instituto Superior Tecnico, Universidade de Lisboa, Nov. 2021. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/844820067127397/Extended Abstract_86675.pdf.

[77] António Augusto Lopes do Arco. “Development and testing of a novel tri-rotor configuration for application in a fixed wing VTOL aircraft”. PhD thesis. Instituto Superior Tecnico, Universidade de Lisboa, Dec. 2022. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/1970719973969904/Thesis_89645.pdf.

[78] Daniele Obertino. "Hybrid-Electric Propulsion System: Design and Simulation Study of a Hybrid-Electric Propulsion System for a VTOL Tilt-Rotor UAV". PhD thesis. Politecnico di Torino, Dec. 2021. URL: http://webthesis.biblio.polito.it/id/eprint/20900.

[79] João C. Figueira et al. "Nonlinear Aero-Propulsive Modeling for Fixed-Wing eVTOL UAV from Flight Test Data". In: *Journal of Aircraft* (Nov. 2024), pp. 1–13. DOI: 10.2514/1.C037964. URL: https://arc.aiaa.org/doi/10.2514/1.C037964.

[80] Nicholas Castellani et al. "Development of a Series Hybrid Multirotor". In: *Journal of Physics: Conference Series* 2716 (1 Mar. 2024), p. 012021. DOI: 10.1088/1742-6596/2716/1/012021. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2716/1/012021.

[81] James Sease. "Autonomous Air Vehicle Flight Formation Coordination Using Onboard Real-Time Control". PhD thesis. University of Victoria, 2021. URL: http://hdl.handle.net/1828/13648.

[82] S. Pedro et al. "Design and performance quantification of VTOL systems for a canard aircraft". In: *The Aeronautical Journal* 125 (1292 Oct. 2021), pp. 1768–1791. DOI: 10.1017/aer.2021.63. URL: https://www.cambridge.org/core/product/identifier/S0001924021000634/type/journal_article.

[83] Tiago Jesus et al. "Surrogate based MDO of a canard configuration aircraft". In: *Structural and Multidisciplinary Optimization* 64 (6 Dec. 2021), pp. 3747–3771. DOI: 10.1007/s00158-021-03051-6. URL: https://link.springer.com/10.1007/s00158-021-03051-6.

[84] James Sease, Stephen Warwick, and Afzal Suleman. "Onboard Trajectory Coordination of Multiple Unmanned Air Vehicles". In: *Unmanned Aerial Vehicle Design and Technology*. Ed. by Emre Karakoc T. Hikmet and Özbek. Springer International Publishing, 2024, pp. 53–67. DOI: 10.1007/978-3-031-45321-2_4. URL: https://link.springer.com/10.1007/978-3-031-45321-2_4.

[85] CAE Inc. *Magnetic Anomaly Detection-Extended Role*. Tech. rep. Defense Security, 2023. URL: https://www.cae.com/content/docs/defense-security/DS_CAEMADXR_March2023.pdf.

[86] UKR SPEC Systems. *PD-1 Unmanned Aerial System*. Tech. rep. UKR SPEC Systems, 2016. URL: https://www.unmannedsystemstechnology.com/wp-content/uploads/2016/06/PD-1-Fixed-Wing-UAV.pdf.

[87] B. Eggleston et al. "DEVELOPMENT OF THE BRICAN TD100 SMALL UAS AND PAYLOAD TRIALS". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL-1/W4 (1W4 Aug. 2015), pp. 143–149. DOI: 10.5194/isprsarchives-XL-1-W4-143-2015. URL: https://isprs-archives.copernicus.org/articles/XL-1-W4/143/2015/.

[88] David Hambling. "Unmanned Systems in Anti-Submarine Warfare". In: *BASIC* Briefing No.1 (Parliamentary Briefings on Trident Renewal Feb. 2016). URL: https://basicint.org/wp-content/uploads/2018/06/BASIC_Hambling_ASW_Feb2016_final_0.pdf.

[89] Staff Writer. *Northrop Grumman MQ-8 Fire Scout UAVs*. May 2024. URL: https://www.militaryfactory.com/aircraft/detail.php?aircraft_id=889#history.

[90] N.M. Babashkin, S.A. Kadnichanskiy, and S.S. Nekhin. "Research tests of the Geoscan 101, the Geoscan 201 hardware-and-software complexes". In: *Geodesy and Cartography* 955 (1 Feb. 2020), pp. 19–25. DOI: 10.22389/0016-7126-2020-955-1-19-25.

[91] Matthew Wells et al. "Reducing magnetic interference in a geomagnetic survey UAS through modelling and design optimization". In: *2012 IEEE International Conference on Mechatronics and Automation*. IEEE, Aug. 2012, pp. 144–149. DOI: 10.1109/ICMA.2012.6282822.

[92] Mohammad H. Sadraey. *Design of Unmanned Aerial Systems Aerospace*. Wiley, 2020.

[93] Thomas C.. Corke. *Design of aircraft*. Prentice Hall, 2003, p. 391.

[94] QuSpin. *Total-Field Magnetometer (QTFM)*. URL: https://quspin.com/qtfm/.

[95] Billingsley Aerospace and Defense. *TFM100-G2 Ultra-miniature triaxial fluxgate magnetometer*. 2008. URL: https://magnetometer.com/wp-content/uploads/TFM100-G2-Spec-Sheet-February-2008.pdf.

[96] António Arco et al. “Investigation on the Airworthiness of a Novel Tri-Rotor Configuration for a Fixed Wing VTOL Aircraft”. In: *International Journal of Aviation Science and Technology* vm04 (is02 Dec. 2023), pp. 53–62. DOI: 10.23890/IJAST.vm04is02.0201. URL: https://www.ijast.org/issues/vm04is02/article68.pdf.

[97] Jay Michael Todd Matlock. “Evaluation of Hybrid-Electric Propulsion Systems for Unmanned Aerial Vehicles”. PhD thesis. University of Victoria, 2019. URL: http://hdl.handle.net/1828/11484.

[98] Jamal Wilson. “Modelling, Experimental Validation and Evaluation of a Parallel Hybrid-Electric Propulsion System for Small-Scale UAVs”. PhD thesis. University of Victoria, Dec. 2024. URL: https://hdl.handle.net/1828/20894.

[99] João Farinha et al. “Hydrogen fuel cell integration and testing in a hybrid-electric propulsion rig”. In: *International Journal of Hydrogen Energy* 48 (97 Dec. 2023), pp. 38473–38483. DOI: 10.1016/j.ijhydene.2023.06.090. URL: https://linkinghub.elsevier.com/retrieve/pii/S036031992302949X.

[100] Binyan Xu et al. “Adaptive Distributed Lyapunov-Based Model Predictive Control for Multi-UAV Formation Tracking with Weighted Directed Graphs”. In: *2024 IEEE 18th International Conference on Control amp; Automation (ICCA)*. IEEE, June 2024, pp. 418–423. DOI: 10.1109/ICCA62789.2024.10591925. URL: https://ieeexplore.ieee.org/document/10591925/.

[101] Binyan Xu et al. “Adaptive Distributed Observer-based Model Predictive Control for Multi-agent Formation with Resilience to Communication Link Faults”. In: (Oct. 2024). DOI: 10.48550/arXiv.2410.23592. URL: http://arxiv.org/abs/2410.23592.

[102] Binyan Xu. “An integrated fault-tolerant model predictive control framework for UAV systems”. PhD thesis. University of Victoria, 2024. URL: https://hdl.handle.net/1828/16753.

[103] Sergio Salazar-Cruz, Rogelio Lozano, and Juan Escareño. “Stabilization and nonlinear control for a novel trirotor mini-aircraft”. In: *Control Engineering Practice* 17 (8 Aug. 2009), pp. 886–894. DOI: 10.1016/j.conengprac.2009.02.013.

[104] Xun Gu, Bin Xian, and Jieqi Li. "Model free adaptive control design for a tilt trirotor unmanned aerial vehicle with quaternion feedback: Theory and implementation". In: *International Journal of Adaptive Control and Signal Processing* 36 (1 Jan. 2022), pp. 122–137. DOI: 10.1002/acs.3344.

[105] Mohamed Kara Mohamed and Alexander Lanzon. "Design and control of novel tri-rotor UAV". In: *Proceedings of 2012 UKACC International Conference on Control*. IEEE, Sept. 2012, pp. 304–309. DOI: 10.1109/CONTROL.2012.6334647.

[106] C. Papachristos and A. Tzes. "Modeling and control simulation of an unmanned Tilt Tri-Rotor Aerial vehicle". In: *2012 IEEE International Conference on Industrial Technology*. IEEE, Mar. 2012, pp. 840–845. DOI: 10.1109/ICIT.2012.6210043.

[107] Warren F. Phillips. *Mechanics of Flight*. Second edition. Wiley, Dec. 2009.

[108] Bogdan Clavac and Zoltan Korka. "Efficiency Investigation on a Helical Gear Transmission". In: *Analele Universităţii "Eftimie Murgu" Reşiţa, Fascicola Inginerie* XXIV (Jan. 2017), pp. 55–66.

[109] R. Prabhu Sekar. "Determination of load dependent gear loss factor on asymmetric spur gear". In: *Mechanism and Machine Theory* 135 (May 2019), pp. 322–335. DOI: 10.1016/j.mechmachtheory.2019.02.011.

[110] P. Zhou et al. "Temperature-Dependent Demagnetization Model of Permanent Magnets for Finite Element Analysis". In: *IEEE Transactions on Magnetics* 48 (2 Feb. 2012), pp. 1031–1034. DOI: 10.1109/TMAG.2011.2172395.

[111] Sami Ruoho et al. "Interdependence of Demagnetization, Loading, and Temperature Rise in a Permanent-Magnet Synchronous Motor". In: *IEEE Transactions on Magnetics* 46 (3 Mar. 2010), pp. 949–953. DOI: 10.1109/TMAG.2009.2033592.

[112] Eugene A. Morelli and Vladislav Klein. *Aircraft System Identification: Theory and Practice*. 2nd ed. Lulu.com, 2016. URL: https://books.google.ca/books?id=HsW-AQAACAAJ.

[113] Ravindra V. Jategaonkar. *Flight Vehicle System Identification: A Time-Domain Methodology, Second Edition*. American Institute of Aeronautics and Astronautics, Inc., Feb. 2015. DOI: 10.2514/4.102790.

[114] Stephen E. Riddick. "An Overview of NASA's Learn-to-Fly Technology Development". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-0760.

[115] Sean Bazzocchi and Afzal Suleman. "Non-linear System Identification for UAS Adaptive Control". In: Feb. 2023, pp. 167–177. DOI: 10.1007/978-3-031-45321-2_10.

[116] Benjamin M. Simmons, James L. Gresham, and Craig A. Woolsey. "Aero-Propulsive Modeling for Propeller Aircraft Using Flight Data". In: *Journal of Aircraft* 60 (1 Jan. 2023), pp. 81–96. DOI: 10.2514/1.C036773.

[117] Eugene A. Morelli. "Determining aircraft moments of inertia from flight test data". In: *AIAA Scitech 2021 Forum* (2021), pp. 1–17. DOI: 10.2514/6.2021-1642.

[118] Piotr Lichota, Krzysztof Sibilski, and Per Ohme. "D-Optimal Simultaneous Multistep Excitations for Aircraft Parameter Estimation". In: *Journal of Aircraft* 54 (2 Mar. 2017), pp. 747–758. DOI: 10.2514/1.C033794.

[119] *System IDentification Programs for AirCraft (SIDPAC)*. URL: https://software.nasa.gov/software/LAR-16100-1.

[120] Benjamin M. Simmons, Hunter G. McClelland, and Craig A. Woolsey. "Non-linear model identification methodology for small, fixed-wing, unmanned aircraft". In: *Journal of Aircraft* 56 (3 2019), pp. 1056–1067. DOI: 10.2514/1.C035160.

[121] Eugene Morelli. "Practical Aspects of the Equation-Error Method for Aircraft Parameter Estimation". In: *AIAA Atmospheric Flight Mechanics Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2006. DOI: 10.2514/6.2006-6144.

[122] Benjamin M. Simmons. "System Identification for Propellers at High Incidence Angles". In: *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2021. DOI: 10.2514/6.2021-1190.

[123] Robert W. Deters, Gavin Kumar Ananda Krishnan, and Michael S. Selig. "Reynolds Number Effects on the Performance of Small-Scale Propellers". In: *32nd AIAA Applied Aerodynamics Conference*. American Institute of Aeronautics and Astronautics, June 2014. DOI: 10.2514/6.2014-2151.

[124] Gautier Hattenberger, Antoine Drouin, and Murat Bronz. "Electric Propulsion System Characterization through Experiments". In: *International Micro Air Vehicle Conference and Competition* (IMAV Oct. 2016). URL: https://enac.hal.science/hal-01396980.

[125] Darren Lance Gabriel, Johan Meyer, and Francois du Plessis. "Brushless DC motor characterisation and selection for a fixed wing UAV". In: *IEEE Africon '11*. IEEE, Sept. 2011, pp. 1–6. DOI: 10.1109/AFRCON.2011.6072087.

[126] Erlend M. Coates et al. "Propulsion System Modeling for Small Fixed-Wing UAVs". In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, June 2019, pp. 748–757. DOI: 10.1109/ICUAS.2019.8798082.

[127] Nicolas Michel et al. "Modeling and validation of electric multirotor unmanned aerial vehicle system energy dynamics". In: *eTransportation* 12 (May 2022), p. 100173. DOI: 10.1016/j.etran.2022.100173.

[128] Eugene A. Morelli and Vladislav Klein. "Accuracy of Aerodynamic Model Parameters Estimated from Flight Test Data". In: *Journal of Guidance, Control, and Dynamics* 20 (1 Jan. 1997), pp. 74–80. DOI: 10.2514/2.3997.

[129] Eugene A. Morelli. "Global nonlinear aerodynamic modeling using multivariate orthogonal functions". In: *Journal of Aircraft* 32 (2 Mar. 1995), pp. 270–277. DOI: 10.2514/3.46712.

[130] Jared A. Grauer and Eugene A. Morelli. "A Generic Nonlinear Aerodynamic Model for Aircraft". In: *AIAA Atmospheric Flight Mechanics Conference*. American Institute of Aeronautics and Astronautics, Jan. 2014. DOI: 10.2514/6.2014-0542.

[131] Dan Liao and Richard Valliant. "Condition indexes and variance decompositions for diagnosing collinearity in linear model analysis of survey data". In: *Survey Methodology* 38 (2 Dec. 2012), pp. 189–202. URL: https://nces.ed.gov/FCSM/pdf/Liao_2012FCSM_VI-A.pdf.

[132] David A. Belsley. "Demeaning Conditioning Diagnostics through Centering". In: *The American Statistician* 38 (2 May 1984), pp. 73–77. DOI: 10.1080/00031305.1984.10483169.

[133] Jong Hae Kim. "Multicollinearity and misleading statistical results". In: *Korean Journal of Anesthesiology* 72 (6 Dec. 2019), pp. 558–569. DOI: 10.4097/kja.19087.

[134] Henri Theil. *Economic Forecasts and Policy*. 2nd ed. North-Holland Publishing Company, 1970, pp. 31–42.

[135] Andrei Dorobantu et al. "System Identification for Small, Low-Cost, Fixed-Wing Unmanned Aircraft". In: *Journal of Aircraft* 50 (4 July 2013), pp. 1117–1130. DOI: 10.2514/1.C032065.

[136] David J. Grymin and Mazen Farhood. "Two-Step System Identification and Trajectory Tracking Control of a Small Fixed-Wing UAV". In: *Journal of Intelligent Robotic Systems* 83 (1 July 2016), pp. 105–131. DOI: 10.1007/s10846-015-0298-8.

[137] Bruno Miguel Wong Luis. "Design and Development of a Magnetic Anomaly Detection Unmanned Aerial Vehicle". PhD thesis. Instituto Superior Tecnico, Nov. 2019. URL: https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997259488/Thesis_Bruno_Luis_80891.pdf.

[138] Daniele Obertino et al. "Design Considerations for Hybrid-Electric Propulsion Systems for FW-VTOL Aircraft". In: *Advances in Electric Aviation*. Ed. by Öznur et al. Springer International Publishing, 2023, pp. 9–16. DOI: 10.1007/978-3-031-32639-4_2. URL: https://link.springer.com/10.1007/978-3-031-32639-4_2.

[139] F. Jia and D. Lichti. "A COMPARISON of SIMULATED ANNEALING, GENETIC ALGORITHM and PARTICLE SWARM OPTIMIZATION in OPTIMAL FIRST-ORDER DESIGN of INDOOR TLS NETWORKS". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. 4. Copernicus GmbH, Sept. 2017, pp. 75–82. DOI: 10.5194/isprs-annals-IV-2-W4-75-2017.

[140] Mayank Joshi, Manasi Gyanchandani, and Dr Rajesh Wadhvani. "Analysis of Genetic Algorithm, Particle Swarm Optimization and Simulated Annealing on Benchmark Functions". In: *Proceedings - 5th International Conference on Computing Methodologies and Communication, ICCMC 2021*. Institute of Electrical and Electronics Engineers Inc., Apr. 2021, pp. 1152–1157. DOI: 10.1109/ICCMC51019.2021.9418458.

[141] Dionisio S. Pereira and João O.P. Pinto. "Genetic Algorithm based system identification and PID tuning for optimum adaptive control". In: *IEEE/ASME*

International Conference on Advanced Intelligent Mechatronics, AIM 1 (67 2005), pp. 801–806. DOI: 10.1109/aim.2005.1511081.

[142] Komal Khuwaja et al. "PID Controller Tuning Optimization with Genetic Algorithms for a Quadcopter". In: *Recent Innovations in Mechatronics* 5 (1. Apr. 2018). DOI: 10.17667/riim.2018.1/11. URL: https://ojs.lib.unideb.hu/rIim/article/view/3840.

[143] Huu Khoa Tran and Thanh Nam Nguyen. "Flight Motion Controller Design using Genetic Algorithm for a Quadcopter". In: *Measurement and Control (United Kingdom)* 51 (3-4 Apr. 2018), pp. 59–64. DOI: 10.1177/0020294018768744. URL: https://doi.org/10.1177/0020294018768744.

[144] Mark B. Tischler et al. *Practical Methods for Aircraft and Rotorcraft Flight Control Design: An Optimization-Based Approach*. American Institute of Aeronautics and Astronautics, Inc., Apr. 2017. DOI: 10.2514/4.104435.

[145] Sean Bazzocchi. *Automatic Autopilot Tuning Optimization with SID and GA for PX4 Multirotor*. 2019. URL: https://github.com/SeanBaz/Automatic-Autopilot-Tuning-Optimization-PX4-MC-RateController.git.

[146] Eugene A. Morelli and Jared A. Grauer. "Practical aspects of the frequency domain approach for aircraft system identification". In: *2018 Atmospheric Flight Mechanics Conference* (December 2018). DOI: 10.2514/6.2018-3477.

[147] MathWorks. *Estimate output-error polynomial model using time-domain or frequency-domain data*. 2022. URL: https://www.mathworks.com/help/ident/ref/oe.html.

[148] MathWorks. *Estimate parameters of ARX, ARIX, AR, or ARI model*. 2022. URL: https://www.mathworks.com/help/ident/ref/arx.html.

[149] Ahmet Arda Ozdemir and Suat Gumussoy. "Transfer Function Estimation in System Identification Toolbox via Vector Fitting". In: *IFAC-PapersOnLine* 50 (1 July 2017), pp. 6232–6237. DOI: 10.1016/j.ifacol.2017.08.1026.

[150] Dronecode. *PX4 Development: Controller Diagrams*. 2023. URL: https://docs.px4.io/main/en/flight_stack/controller_diagrams.html.

[151] Pedro Sanchez-Cuevas, Guillermo Heredia, and Anibal Ollero. "Characterization of the Aerodynamic Ground Effect and Its Influence in Multirotor Control". In: *International Journal of Aerospace Engineering* 2017 (2017), pp. 1–17. DOI: 10.1155/2017/1823056.

[152] Emre Çelik and Nihat Öztürk. "A hybrid symbiotic organisms search and simulated annealing technique applied to efficient design of PID controller for automatic voltage regulator". In: *Soft Computing* 22 (23 Dec. 2018), pp. 8011–8024. DOI: 10.1007/s00500-018-3432-2. URL: http://link.springer.com/10.1007/s00500-018-3432-2.

[153] Saurabh Srivastava and V. S. Pandit. "Studies on PI/PID controllers in the proportional integral plane via different performance indices". In: *2016 IEEE First International Conference on Control, Measurement and Instrumentation (CMI)*. IEEE, Jan. 2016, pp. 151–155. DOI: 10.1109/CMI.2016.7413729. URL: http://ieeexplore.ieee.org/document/7413729/.

[154] Salam Ibrahim Khather, Mohammed Almaged, and Abdullah I Abdullah. "Fractional order based on genetic algorithm PID controller for controlling the speed of DC motors". In: *International Journal of Engineering and Technology* 7 (4 2018), pp. 5386–5392. DOI: 10.14419/ijet.v7i4.25601. URL: www.sciencepubco.com/index.php/IJET.

[155] Stanley Gotshall and Bart Rylander. *Optimal Population Size and the Genetic Algorithm*. Tech. rep. 2002. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0a7b63cfe4b40741452692366e2047db43b467c7.

[156] J.C. Potts, T.D. Giddens, and S.B. Yadav. "The development and evaluation of an improved genetic algorithm based on migration and artificial selection". In: *IEEE Transactions on Systems, Man, and Cybernetics* 24 (1 1994), pp. 73–86. DOI: 10.1109/21.259687.

[157] Steven L. Brunton et al. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences of the United States of America* 113 (15 2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[158] N. M. Mangan et al. "Model selection for dynamical systems via sparse regression and information criteria". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473 (2204 2017). DOI: 10.1098/rspa.2017.0009.

[159] Hassan Sayyaadi and Mojtaba Sharifi. "Adaptive Impedance Control of UAVs Interacting With Environment Using a Robot Manipulator". In: (2014). DOI: 10.1109/icrom.2014.6990974.

[160] Girish Chowdhary et al. "Experimental Results of Concurrent Learning Adaptive Controllers". In: (2012). DOI: 10.2514/6.2012-4551.

[161] Zain Anwar Ali et al. "Attitude and Altitude Control of Trirotor UAV by Using Adaptive Hybrid Controller". In: *Journal of Control Science and Engineering* (2016). DOI: 10.1155/2016/6459891.

[162] Egemen BELGE et al. "Estimation of Small Unmanned Aerial Vehicle Lateral Dynamic Model With System Identification Approaches". In: *Balkan Journal of Electrical and Computer Engineering* (2020). DOI: 10.17694/bajece.654499.

[163] Zsombor Oreg, Hyo Sang Shin, and Antonios Tsourdos. "Model identification adaptive control - Implementation case studies for a high manoeuvrability aircraft". In: *27th Mediterranean Conference on Control and Automation, MED 2019 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., July 2019, pp. 559–564. DOI: 10.1109/MED.2019.8798513.

[164] Bryan S. Guevara et al. "PD-Based and SINDy Nonlinear Dynamics Identification of UAVs for MPC Design". In: (Oct. 2024). DOI: 10.3390/drones9010071. URL: http://arxiv.org/abs/2410.11791.

[165] Halit Firat Erdogan, Ayhan Kural, and Can Özsoy. "Model Predictive Control of an Unmanned Aerial Vehicle". In: *Aircraft Engineering and Aerospace Technology* (2017). DOI: 10.1108/aeat-03-2015-0074.

[166] Hakan Ülker, Cemal Baykara, and Can Özsoy. "Design of MPCs for a Fixed Wing UAV". In: *Aircraft Engineering and Aerospace Technology* (2017). DOI: 10.1108/aeat-08-2015-0198.

[167] Paul Riseborough. "State Estimation Overview". In: 2019. URL: https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html.

[168] Paul Riseborough. *PX4's Extended Kalman Filter for Navigation (EKF2)*. 2024. URL: https://github.com/PX4/PX4-Autopilot/tree/main/src/modules/ekf2.

[169] Vladislav Klein and Eugene A. Morelli. *Aircraft System Identification: Theory and Practice*. American Institute of Aeronautics and Astronautics, Jan. 2006. DOI: 10.2514/4.861505.

[170] Brian M. de Silva et al. "PySINDy: A Python package for the Sparse Identification of Nonlinear Dynamics from Data". In: (Apr. 2020). URL: http://arxiv.org/abs/2004.08424.

[171] Eugene A. Morelli. "Multiple input design for real-time parameter estimation in the frequency domain". In: *IFAC Proceedings Volumes* 36 (16 Sept. 2003), pp. 639–644. DOI: 10.1016/S1474-6670(17)34833-4.

[172] Eugene A. Morelli. "Flight-Test Experiment Design for Characterizing Stability and Control of Hypersonic Vehicles". In: *Journal of Guidance, Control, and Dynamics* 32 (3 May 2009), pp. 949–959. DOI: 10.2514/1.37092. URL: https://arc.aiaa.org/doi/10.2514/1.37092.

[173] Antonín Novák et al. "Nonlinear System Identification Using Exponential Swept-Sine Signal". In: *IEEE Transactions on Instrumentation and Measurement* 59 (8 Aug. 2010), pp. 2220–2229. DOI: 10.1109/TIM.2009.2031836. URL: http://ieeexplore.ieee.org/document/5299278/.

[174] Pietro Burrascano et al. "Chirp design in a pulse compression procedure for the identification of non-linear systems". In: *2017 14th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, June 2017, pp. 1–4. DOI: 10.1109/SMACD.2017.7981565. URL: http://ieeexplore.ieee.org/document/7981565/.

[175] Alan Kaptanoglu et al. "PySINDy: A comprehensive Python package for robust sparse system identification". In: *Journal of Open Source Software* 7 (69 Jan. 2022), p. 3994. DOI: 10.21105/joss.03994. URL: https://joss.theoj.org/papers/10.21105/joss.03994.

[176] Alan A. Kaptanoglu et al. "Promoting global stability in data-driven models of quadratic nonlinear dynamics". In: *Physical Review Fluids* 6 (9 Sept. 2021), p. 094401. DOI: 10.1103/PhysRevFluids.6.094401. URL: https://link.aps.org/doi/10.1103/PhysRevFluids.6.094401.

[177] Kathleen Champion et al. “A Unified Sparse Optimization Framework to Learn Parsimonious Physics-Informed Models From Data”. In: *IEEE Access* 8 (2020), pp. 169259–169271. DOI: 10.1109/ACCESS.2020.3023625. URL: https://ieeexplore.ieee.org/document/9194760/.

[178] Peng Zheng et al. “A Unified Framework for Sparse Relaxed Regularized Regression: SR3”. In: *IEEE Access* 7 (2019), pp. 1404–1423. DOI: 10.1109/ACCESS.2018.2886528.

[179] Dimitris Bertsimas and Wes Gurnee. “Learning sparse nonlinear dynamics via mixed-integer optimization”. In: *Nonlinear Dynamics* 111 (7 Apr. 2023), pp. 6585–6604. DOI: 10.1007/s11071-022-08178-9.

[180] Martina Mammarella and Elisa Capello. “A Robust MPC-based autopilot for mini UAVs”. In: *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, June 2018, pp. 1227–1235. DOI: 10.1109/ICUAS.2018.8453290. URL: https://ieeexplore.ieee.org/document/8453290/.

[181] Felix Fiedler et al. “do-mpc: Towards FAIR nonlinear and robust model predictive control”. In: *Control Engineering Practice* 140 (Nov. 2023), p. 105676. DOI: 10.1016/j.conengprac.2023.105676.

[182] Joel A. E. Andersson et al. “CasADi: a software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11 (1 Mar. 2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[183] Steven Macenski et al. “Robot Operating System 2: Design, architecture, and uses in the wild”. In: *Science Robotics* 7 (66 May 2022). DOI: 10.1126/scirobotics.abm6074.

[184] N. Fonzi, S. L. Brunton, and U. Fasel. “Data-driven nonlinear aeroelastic models of morphing wings for control”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476 (2239 July 2020), p. 20200079. DOI: 10.1098/rspa.2020.0079. URL: https://royalsocietypublishing.org/doi/10.1098/rspa.2020.0079.