A qLMPC Framework for Path-Following Control of Fixed-Wing UAVs

Vom Promotionsausschuss der
Technischen Universität Hamburg
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing)

genehmigte Dissertation (Monografie)

von
Ahmed Samir Said Metwalli Rezk

aus
Kairo, Ägypten

2025

Erster Gutachter:
Prof. Dr. Herbert Werner
Institut für Regelungstechnik
Technische Universität Hamburg

Zweiter Gutachter:
Prof. Dr. Frank Thielecke
Institut für Flugzeug-Systemtechnik
Technische Universität Hamburg

Vorsitzender des Prüfungsauschusses:
Prof. Dr.-Ing. Robert Seifried
Institut für Mechanik und Meerestechnik
Technische Universität Hamburg

Tag der mündlichen Prüfung: 17. June 2024

ORCID: https://orcid.org/my-orcid?orcid=0000-0003-4073-217X

More than research, a reshaping of self.

Summary

This thesis presents a novel predictive path-following control scheme based on quasi-Linear-Parameter-Varying (qLPV) modeling of fixed-wing UAVs' 3D position dynamics. It addresses limitations in prevailing literature by offering robustness against variable winds as well as computational efficiency based on solving Quadratic Programs (QPs). This approach extends from a simplified kinematic model to a high-fidelity dynamic model of the ULTRA-Extra UAV, enabling real-time solutions.

Deriving a qLPV representation of nonlinear dynamics is not unique, yet employing velocity-based linearization emerges as a fitting approach. This method not only ensures straightforward implementation but also yields an accurate representation of nonlinear dynamics. Additionally, its applicability is not dependent on the designer's expertise or intuition, making it well-suited for control design. Another advantage of utilizing velocity-based linearization models is their compatibility with offset-free MPC, showcasing efficiency in handling potential disturbances.

The proposed scheme is further extended to address the obstacle avoidance problem, which presents a notable challenge due to the nonlinear nature of the obstacles. This challenge is met by representing the obstacles as qLPV constraints. Consequently, these constraints can be integrated into the optimization problem within the proposed approach.

The stability of the proposed algorithm is examined using two approaches: the first employs terminal ingredients to ensure output boundedness but introduces complexity and high computational cost, particularly with velocity models. The second approach mitigates these issues by leveraging the velocity model, requiring only two additional constraints to ensure steady-state convergence and hence system stability.

The proposed control scheme is evaluated in simulation studies for two different scenarios previously used in flight experiments with the same aircraft employing a geometric path-following algorithm. The simulations demonstrate the effectiveness of the proposed algorithm in both path-following and obstacle avoidance tasks.

Abstract

This thesis presents a novel predictive path-following control scheme for fixed-wing UAVs, addressing limitations in the existing literature. While geometric methods struggle with variable winds and predictive strategies tend to be computationally intensive, the qLPV approach proposed here strikes a balance—accurately capturing the system dynamics while maintaining computational efficiency through the use of Quadratic Programs (QPs). The practicability of the approach is demonstrated by applying it to a high-fidelity model of the ULTRA-Extra UAV. Constructing qLPV representations is not unique; in this work, velocity-based linearization is employed, which is known to be well-suited for controller design. It is also compatible with offset-free Model Predictive Control (MPC) and enables efficient disturbance rejection. The proposed framework is further extended to obstacle avoidance by representing obstacles as qLPV constraints.

Kurze Zusammenfassung

In dieser Arbeit wird ein neuartiger prädiktiver Pfadfolgeregler für Starrflügler-UAVs vorgestellt, der Einschränkungen in der bestehenden Literatur überwindet. Während geometrische Methoden mit variablen Windstärken zu kämpfen haben und prädiktive Strategien rechenintensiv sind, bietet der hier vorgestellte Ansatz sowohl eine präzise Darstellung der nichtlinearen Dynamik als qLPV-Modell als auch eine effiziente Implementierung einer auf quadratischer Programmierung basierenden Regelung. Dieser Ansatz wurde auf ein realitätsgetreues Modell des ULTRA-Extra-UAVs angewendet. Die Konstruktion von qLPV-Darstellungen ist nicht eindeutig, aber die geschwindigkeitsbasierte Linearisierung ist bekannt für ihre guten Eigenschaften im Hinblick auf den Reglerentwurf. Sie ist auch mit der offset-freien prädiktiven Regelung kompatibel und kann Störungen effizient ausregeln. Das vorgeschlagene Framework wurde zudem auf die Hindernisvermeidung ausgeweitet, wobei die Herausforderungen der Linearisierung und der kontinuierlichen Aktualisierung berücksichtigt wurden. Simulationsstudien in zwei Szenarien bestätigen die Effektivität bei der Bahnverfolgung und Hindernisvermeidung.

Acknowledgments

I express my sincere gratitude to **Prof. Dr. Herbert Werner** for his patience, kindness, and adept problem-solving skills. His mentorship and insightful guidance have been invaluable to my research journey since its inception. Our engaging discussions and seminars have been especially enriching. Beyond the academic sphere, the enjoyable moments we shared during summer and winter trips fostered a family-like bond among the institute members.

I also extend my appreciation to **Prof. Dr. Frank Thielecke** for graciously serving as my second examiner and for providing the aircraft model that was crucial to my research. This model significantly supported the practical application of my theoretical work. My gratitude also goes to **Prof. Dr. Robert Seifried** for chairing my PhD defense.

Special thanks to **Benjamin Herrmann** and **Leif Rieck** from the FST group for our productive meetings that bridged the gap between control engineering and flight control engineering, helping me address key challenges in my research. I am also grateful to **Dr. Julian Theis** for his excellent course on aircraft control parameter design, which greatly accelerated my progress in that area.

I am deeply thankful to the **Egyptian government** for its financial support during my stay in Hamburg. I also appreciate the continuous assistance of the **Egyptian Defence Office in Berlin**, especially **Mr. Ahmed Abd El-Hakim** and **Mr. Mohammed Mohey**.

My time in Hamburg has been an enriching chapter of my life, and I am fortunate to have met so many wonderful colleagues and friends. Special thanks to **Ms. Christina Kopf** and **Ms. Kirsten Johanson**, the ICS secretaries, for their kind and consistent support, both before my arrival and throughout my stay.

I would also like to thank my friends **Adwait Datar**, **Patrick Göttsch**, **Philipp Hastedt**, **Christian Hespe**, **Jannis Lübsen**, **Bindu Sharan**, **Furugh Mirali**, and **Antonio Mendez Gonzalez** for the many enjoyable conversations and moments we shared over coffee breaks, weekly breakfasts, and group outings. I'm also thankful to **Dirk Bacck** for his technical support and his creative ideas during colleagues' graduation celebrations.

Special thanks to **Shuyuan Fan**, **Yongyuan Xu**, and **Prima Aditya** — with

a special mention of Prima. I truly cherished our time traveling and hanging out together; I can't imagine how this journey would have been without our regular gatherings.

I would also like to express my heartfelt gratitude to **Ahmed Hamdy**, my closest friend in Egypt, whose regular calls and thoughtful conversations provided comfort and helped ease the sense of distance from home. I am equally thankful to my long-time school friend **Mostafa Fathy** for his unwavering support during my time in Germany—particularly in the early days—when his help was instrumental in navigating the challenges of settling in a new country.

Lastly, I am forever grateful to my **parents** for their unwavering love and support, even across great distances. Their encouragement has been constant throughout my life, not just during my PhD years. I thank my **sisters — Marwa, Mai, and Riham** — for always standing by me. Most of all, I am deeply thankful to my wife, **Donia**, and my children, **Ziad** and **Gamila**, for their patience, love, and support throughout this journey. Words cannot fully express how much they mean to me.

Hamburg, July 8, 2025
Ahmed Samir

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent decades, Unmanned Aerial Vehicles (UAVs) have found diverse applications, spanning both civilian and military domains. In military contexts, UAVs are used for surveillance and reconnaissance, whereas in civilian applications, they play roles in mapping, rescue operations [Luo et al., 2019], and agriculture [Christiansen et al., 2017]. Consequently, extensive research has been conducted to enhance UAV automation. This automation encompasses tasks such as takeoff, executing planned missions, and landing at specified locations. One common requirement across all UAV applications is the ability of the aircraft to follow a predetermined path, which is crucial for mission success. Therefore, the path-following problem has emerged as a prominent research topic, with numerous researchers striving to develop efficient and accurate solutions to ensure high path-following performance. In this study, we aim to address the path-following problem for fixed-wing UAVs by leveraging predictive control techniques, considering the nonlinear position dynamics [Beard, 2012], and prioritizing solution efficiency to enable real-time implementation with minimal hardware resources. However, before delving into the path-following problem, it is crucial to differentiate between two similar yet distinct problems: trajectory tracking and path-following. In the trajectory tracking problem, the path to be tracked is time-parameterized, meaning that the vehicle following the path should reach specific reference points at specific times. On the other hand, in the path-following problem, the objective is to minimize the error between the vehicle and the path without considering time parameterization [Faulwasser et al. 2009]. In essence, the solution to the trajectory tracking problem addresses both the "where and when" questions, while the solution to the path-following problem only addresses the "where" question. In this research, the focus is on the path-following problem.

1.1 Path-Following Problem

The capability of path-following stands as a cornerstone for numerous UAV applications, representing a fundamental requirement for their successful operation.

Virtually all UAV applications necessitate precise path-following capabilities, highlighting its paramount importance. These applications vary widely, each with its unique demands in terms of accuracy and adherence to timing constraints. Adhering to timing constraints transforms the problem from a path-following task to a trajectory tracking challenge, introducing time as an additional parameter to be followed.

Various types of unmanned vehicles, including UAVs, autonomous surface vehicles (ASVs), autonomous underwater vehicles (AUVs), and unmanned ground vehicles (UGVs), exhibit distinct characteristics. While the principles of the path-following problem share similarities across these vehicle types, their dynamics and kinematics differ significantly. Consequently, the constraints imposed on path-following algorithms vary accordingly. This underscores the existence of diverse path-following algorithms tailored to the specific requirements of each vehicle type. In this study, we present a model-based predictive control (MPC) algorithm for path-following, which can be customized for various vehicles by simply adjusting the model provided to the MPC controller and incorporating the relevant constraints based on the specific vehicle requirements. Our research primarily concentrates on path-following for UAVs.

The essence of the path-following problem for UAVs is to minimize the spatial distance between the aircraft and the designated path. Given a path represented by $P(\theta)$, along with the initial position of the aerial vehicle denoted by $p(x, y, z)$, as well as course angle χ and flight-path angle γ, the objective of the path-following controller is to devise a control strategy that generates the required commands. These commands are aimed at minimizing the discrepancy between the path and the aircraft. Essentially, the controller guides the aircraft towards the path by generating commands for both yaw (χ) and pitch (γ) to control the aircraft's yaw and pitch motion, respectively. However, this approach assumes that the entire path is predefined. This becomes problematic when the path evolves dynamically, as it requires controlling the aircraft's speed in addition to its direction. If the aircraft speed is controlled separately and the path evolves at a different rate than the aircraft, it may affect tracking performance and potentially lead to tracking failure. In this thesis, we address this issue by considering aircraft speed as an additional degree of freedom, alongside the required commands for pitch and yaw movements, all of which are determined by the path-following controller. A depiction of the path-following problem can be found in Fig. 1.1.

Definition 1.1. *(Path-following problem) Consider a UAV located at a position* $p(t) \in \mathbb{R}^3$ *tasked with following a path* $P(\theta) \in \mathbb{R}^3$*, which is characterized by a parameter* $\theta \in \mathbb{R}$*. We assume that* $P(\theta)$ *is suitably smooth with bounded derivatives. The objective here is to devise a feedback control policy in such a manner that the distance* $\|p(t) - P(\theta)\|$ *tends towards the vicinity of the origin. Let* d *denote the cross-track error, defined as the norm distance between the aircraft and the path. The goal of path-following is to minimize this distance, aiming for* $d \to 0$ *as* $t \to \infty$*.*

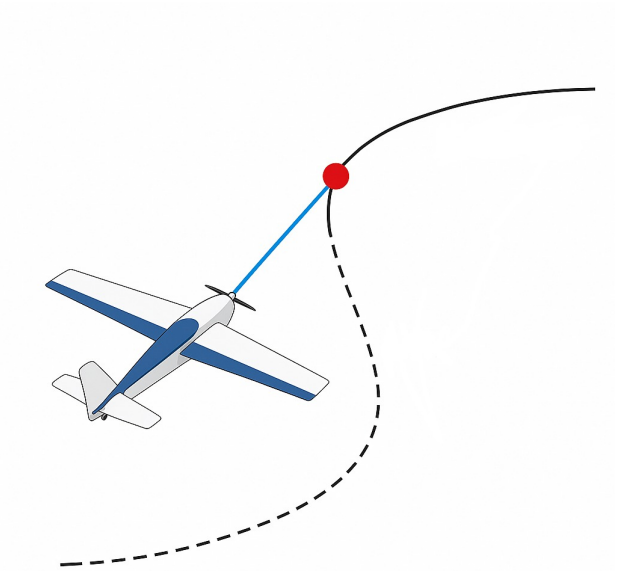

Figure 1.1: Path-following problem.

1.2 Path-Following Algorithms

As previously mentioned, path-following is essential across diverse unmanned vehicle domains, including ground, underwater, and aerial applications, with particular significance in aerospace contexts. The differences in kinematics and dynamics among various vehicles justify the existence of several path-following algorithms. A basic objective of path-following is to guide the vehicle toward a predefined path. With this definition in mind, path-following algorithms can be broadly categorized into geometric and control algorithms, based on the approach used to derive the guidance laws required to direct the vehicle.

1.2.1 Geometric-Based Algorithms

Geometric techniques, as their name implies, involve steering the aircraft by adjusting the heading and flight-path angle based on the cross-track error between the aircraft and the path, which is influenced by changes in the path geometry. Examples of geometric techniques include pure pursuit [Conte et al., 2004] and line-of-sight (LOS) [Rolf Rysdyk, 2012]. Path-following algorithms based on pure

pursuit and LOS guidance laws rely on a virtual target point (VTP) on the path. The guidance law is designed to generate the necessary commands for the vehicle to track this VTP. The distance between the VTP and the UAV position projected onto the path is often referred to as the "virtual distance". The role of the guidance law is simply to generate the required commands for the vehicle to pursue the VTP effectively. The stability of LOS guidance algorithms is heavily influenced by the selection of the virtual distance parameter. Additionally, a fusion of pure pursuit and LOS guidance laws yields a novel guidance approach for path-following [Kothari et al., 2009].

Another geometric path-following control method is the Nonlinear Guidance Law (NLGL) [Park et al., 2007], which also involves pursuing a VTP and generating the necessary lateral and longitudinal acceleration commands. This concept is further discussed in [Sedlmair et al., 2019]. While NLGL is often categorized as a geometric path-following method in the literature [Sujit et al., 2014], it can also be viewed as a control-based approach because it relies on feedback control techniques in its design.

Another alternative to guidance laws relies on vector fields (VFs) [Nelson et al., 2007], offering an intuitive and straightforward implementation.

All the aforementioned approaches are geometric, making them simple, easy to implement, and quick. However, they lack robustness against wind disturbances. Therefore, alternative path-following strategies are required to handle disturbances, particularly wind disturbances, for UAV path-following.

1.2.2 Control-Based Algorithms

Path-following control-based techniques have become popular for deriving guidance laws that rely on feedback from the aircraft's sensors. They provide a more robust solution against wind disturbances for the path-following problem compared to geometric techniques.

One common approach in path-following, relying on control algorithms, is proportional-integral-derivative (PID) control, although it does not perform as effectively as NLGL, as demonstrated in [Sun et al., 2008]. To enhance performance compared to NLGL, a PID controller with feedforward capability was developed in [Rhee et al., 2010]. Other control methodologies encompass linear quadratic regulator (LQR) [Wang et al., 2002], sliding mode control [Healey and Lienard, 1993], backstepping control [P. Encarnacao and A. Pascoal, 2001], and dynamic programming [da Silva and Estrela, 2010].

Notably, Model-based Predictive Control (MPC) has also gained prominence in addressing both path-following and path-planning challenges, optimizing trajectories as discussed in [Li et al., 2010] and [Jackson et al., 2008]. Stability and performance assurances are crucial requirements for nonlinear control laws to ensure accurate path following under various path and environmental conditions.

Local performance and stability results exist for a category of path-following controllers derived from the concept of trimming trajectories [Kaminer et al., 1998]. The analytical performance of a path-following controller compared to a trajectory tracking controller is discussed in [Aguiar et al., 2008].

The algorithms mentioned above have also found applications in the industry. For instance, the VF path-following has been evaluated on a Kestrel autopilot, and the NLGL has been assessed on a Piccolo autopilot [Piccolo, 2024]. Additionally, ArduPilot utilizes the VF implementation, while the Paparazzi autopilot [Paparazzi, 2025] employs the carrot-chasing algorithm. A comprehensive survey covering path-following algorithms can be accessed in [Sujit et al., 2014], which delves into 2D path-following for both straight-line and circular orbit paths. Additionally, [Bartomeu et al., 2020] offers an in-depth exploration of path-following algorithms tailored specifically for quadcopters. However, it is worth noting that these algorithms can be extended and adapted for use with various other types of vehicles.

1.2.3 Predictive Algorithms

In the following, we consider MPC path-following, a control-based algorithm renowned for its effectiveness. MPC, also known as receding horizon control (RHC), transforms the control task into an online optimization control problem (OCP). At each sampling interval, a sequence of future control inputs is computed by solving a finite horizon optimization problem. Subsequently, only the first element of this computed control sequence is applied, and the process repeats at the subsequent sampling interval.

MPC offers several advantages for path-following problem, notably its capability to predict future path dynamics and its adeptness in handling constraints within the optimization framework. However, a primary drawback of MPC lies in its computational complexity, particularly evident when solving OPs for nonlinear systems. Furthermore, accurately achieving path-following across diverse paths and environmental conditions presents an additional challenge owing to the nonlinearities inherent in the system, particularly concerning stability and performance guarantees. Simply deriving linear control laws by linearizing aircraft position dynamics is insufficient for precise path-following.

With the growing adoption of predictive algorithms for path-following problems, significant research has focused on developing efficient solvers for the associated optimization problems. Prominent examples include `acados` [Verschueren et al., 2020], `GRAMPC` [Englert et al., 2018], and `OpEn` [Sopasakis et al., 2020], which employ methods such as Sequential Quadratic Programming (SQP) and real-time iteration [Moritz et al., 2005] to enhance computational performance.

In this thesis, we aim to find a middle ground between accurately representing the nonlinear position dynamics of fixed-wing aircraft and devising an efficient

solution with minimal computational complexity feasible for real-time implementation, while also ensuring an acceptable level of stability assurance.

1.3 Obstacle Avoidance Problem

In the previous section, we discussed the importance of enhancing the autonomy of UAVs across various applications. We emphasized that one crucial aspect contributing to increased autonomy is the ability to follow predefined paths, which is essential across a wide range of applications. Additionally, obstacle avoidance capability serves as another functionality to further augment the autonomy of aircraft. This feature proves particularly beneficial in scenarios such as fire detection in urban areas, where numerous obstructive buildings may be present, or in mapping applications to navigate around obstacles like electricity towers.

Integrating obstacle avoidance functionality is also advantageous in other applications, such as agricultural and wildlife monitoring [Linchant et al., 2015]. This functionality enables the UAV to adhere closely to its intended trajectory while maintaining a safe distance from any potential obstacles. Additionally, to comply with safety regulations set forth by the Federal Aviation Administration (FAA) in US civil aviation for small UAVs weighing under 55 pounds, it is imperative to equip these aircraft with Sense and Avoid (SAA) capabilities. The primary objective of implementing SAA is to mitigate the risk of collisions with natural flying objects, other air traffic, and ground populations, thereby minimizing the potential for damage to buildings and other structures, as shown in [Lin et al., 2019] where the authors devised an obstacle avoidance algorithm rooted in geometry. This algorithm was designed for navigation scenarios where self-separation was unattainable or prevented due to emergent events, necessitating obstacle avoidance measures.

Reactive methods also represent a category of real-time obstacle avoidance techniques that rely on sensor data to generate motion directives. These methods determine the required set of commands using navigation strategies like velocity or steering angle calculations. An example of a reactive method is the Nearness Diagram [Minguez and Montano, 2004], which leverages proximity to obstacles and areas of free space for navigation in cluttered environments. Other reactive methods include boundary following algorithms, which integrate local planning with global information to ensure convergence [Kamon and Rivlin, 1997].

To tackle the obstacle avoidance problem, two approaches can be employed. The first involves replanning the path to circumvent the obstacle [Zhang et al., 2020], while the second involves maintaining the original path but deviating from it in the vicinity of the obstacle. Once the obstacle is cleared, the aircraft resumes its trajectory [Adhikari et al., 2020]. The first approach escalates the computational demands within the path-planner module, negating the advantage of simplicity. In this study, we demonstrate how to manage obstacle avoidance using the second

approach, thereby reducing computational complexity and conserving hardware resources.

1.4 Stability Analysis

While stability analyses for MPC were relatively scarce in the 1980s, this had not hindered its widespread adoption in industrial applications. The practical effectiveness of MPC has led industries to implement it extensively, even in the absence of formal stability proofs.

In their survey [Mayne et al., 2000], the authors provided a comprehensive overview of constrained MPC, with a particular emphasis on stability analysis techniques that exploit the benefits of infinite-horizon optimal control formulations. In [Löfberg, 2001], the stability analysis for linear model predictive control (LMPC) is examined, wherein additional constraints are introduced to ensure stability through the solution of offline linear matrix inequalities (LMIs).

Additionally, [Cisneros et al., 2016] extended these stability conditions to quasi-Linear Parameter Varying (qLPV) systems, thereby generalizing the analysis to encompass both linear systems and systems with time-varying parameters.

A key insight from the existing literature is that terminal ingredients are predominantly applied within state-space model formulations. In contrast, for velocity-based models, it is possible to circumvent the need for solving LMIs by exploiting the structural properties of these models. Specifically, stability and solution feasibility can be ensured by introducing only two additional constraints into the OCP, thereby guaranteeing convergence to a steady-state without incurring significant computational overhead, as demonstrated in [Cisneros and Werner, 2021].

In this thesis, we employ two distinct methodologies for stability analysis. The first relies on terminal ingredients, where LMIs are solved to obtain terminal constraints that ensure bounded system states and, consequently, system stability. We also discuss the limitations of this approach, particularly when applied to velocity-based models. As an alternative, we exploit the structure of linearized velocity models to ensure stability through a simpler method—by introducing just two additional constraints that guarantee both feasibility of the OCP and convergence to a steady-state output.

1.5 Scope and Contribution

To address the aforementioned challenges, we propose a novel predictive path-following framework. This framework presents a control-based approach to address the path-following problem, enhancing robustness against wind disturbances within the solution. This is achieved by representing the nonlinear position dynamics of the aircraft using a qLPV representation derived from velocity-based

linearization [Cisneros et al., 2016].

This approach enables the accurate capture of the nonlinear position dynamics in a linearized form, which is updated at each time instant. Subsequently, it facilitates the formulation of a quadratic program (QP) involving a quadratic cost function with linear constraints. This led to the term qLMPC, which reflects the solution of a QP problem within an MPC strategy based on a qLPV system representation.

The qLMPC algorithm was initially introduced in [Cisneros et al., 2016] and has more recently been applied in [Rieck et al., 2023] alongside Laguerre functions for the stabilization and reference tracking of a flexible aircraft.

With the availability of efficient solvers for such QPs, the proposed scheme ensures accurate path-following with manageable computational times. In addition, the proposed framework is extended to handle obstacle avoidance by linearizing the nonlinear obstacle constraints and incorporating them into the OCP. These constraints are updated online based on the aircraft's position, effectively making them qLPV constraints.

To ensure closed-loop stability, two methods from the literature are employed: the first utilizes terminal ingredients, while the second is a more recent approach that leverages velocity-based models to ensure the solution converges to a steady-state. The thesis provides a detailed comparison of these two methods, highlighting their respective limitations and practical challenges.

We deploy our algorithm within a cascaded scheme, a common approach in path-following problems, as demonstrated in [Alcalá et al., 2019] for autonomous vehicles. This scheme consists of three cascaded modules forming two loops. The outermost module is the path-planner, responsible for generating the necessary reference position components $[x_\mathrm{r}, y_\mathrm{r}, z_\mathrm{r}]^\top$. These components are then forwarded to the path-following controller, which generates the necessary commands for the low-level controller $[V_{a_c}, \phi_c, n_{z_c}]^\top$.

The low-level controller, in turn, is responsible for producing commands for the elevators, ailerons, rudder, and throttle, which are transmitted to the aircraft to ensure successful path-following. The path-planner module and the path-following module constitute the high-level loop, while the autopilot constitutes the low-level loop. Figure 1.2 illustrates the implementation of path-following within the cascaded scheme.

The contributions of this thesis are summarized as follows:

1. **A novel predictive path-following framework**, termed qLMPC path-following, has been introduced, which integrates qLPV modeling and MPC control within a cascaded structure for predictive path-following. This framework operates on two main fronts: Firstly, it accurately captures the nonlinearities of the aircraft's position by representing them in a linear form that is continuously updated at each time instant through velocity-based

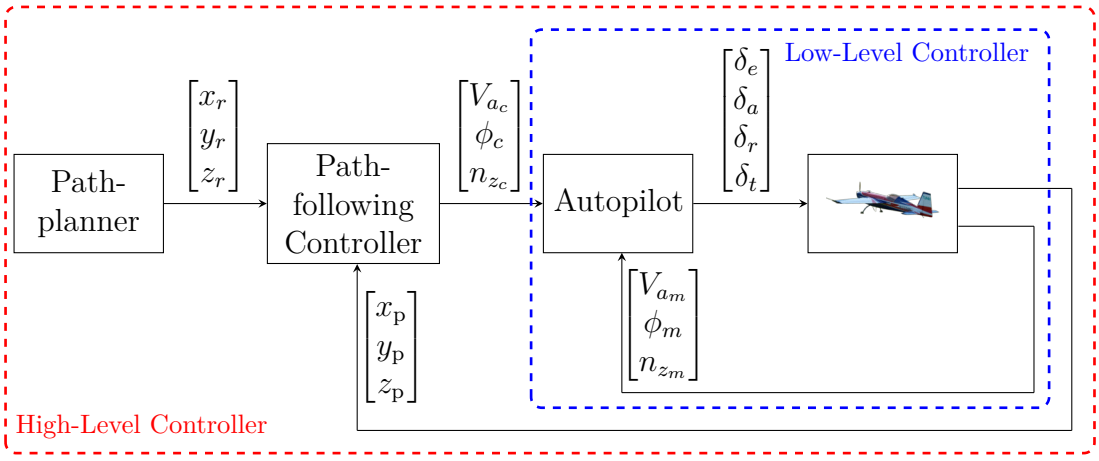

Figure 1.2: Path-following problem setup where the position's reference is $[x_r, y_r, z_r]^\top$, the aircraft's position is $[x_p, y_p, z_p]^\top$, and the actuation signals for the elevators, ailerons, rudder and throttle are $[\delta_e, \delta_a, \delta_r, \delta_t]^\top$.

linearization. Secondly, it employs an offset-free MPC approach to solve the path-following problem, thereby achieving accurate path-following while maintaining low computational times. Moreover, this approach facilitates the implementation of stability guarantee conditions, drawing on extensive literature studies and leveraging the qLPV modeling to formulate an LTI system at each time instant. This framework has been demonstrated through a case study involving the ULTRA-Extra UAV, an unmanned aerial vehicle developed at the Hamburg University of Technology. Details of this contribution are presented in [Samir et al., 2024a] and elaborated upon in Chapter 3.

2. **Incorporating obstacle avoidance** into the proposed framework alongside path-following. This introduces challenges due to the nonlinear nature of obstacle constraints. To address this, the constraints were linearized and updated at each time step, as they depend on the relative distance between the aircraft and the obstacles. As a result, these constraints are treated as qLPV constraints. This contribution is documented in [Samir et al., 2024b] and is featured in this thesis in Chapter 4.

3. **Stability analysis** of the proposed scheme was performed by augmenting the optimization problem with additional constraints. These constraints can be derived either by solving offline LMIs to obtain terminal ingredients, or by leveraging the velocity-based linearized model and introducing two additional constraints that ensure feasibility and steady-state convergence of the optimal solution. The details of this contribution are documented in Chapter 5.

In this thesis, we use a 3D kinematic model for the fixed-wing aircraft as the foundation of the MPC controller. Initially, we apply our framework to this kinematic model to validate the implementation. Subsequently, we extend the implemen-

tation to a high-fidelity model of a research aircraft, known as ***ULTRA-Extra*** [Krings et al., 2013]. This model encompasses the aircraft dynamics, including actuator dynamics and associated time delays. However, it is noteworthy that this high-fidelity model does not incorporate sensor dynamics.

1.6 Thesis Structure

Chapter 2 reviews the path-generation methodologies employed in this work. We employ two distinct approaches to generate paths, beginning with a set of waypoints and ending in the creation of continuous, sufficiently differentiable, and trackable paths. The first approach is a hybrid path-generator that combines arc length parameterization with cubic splines. The second approach introduces a path generator based on synthetic waypoints, a concept previously employed alongside pure pursuit guidance. In this work, we adapt the idea to design a path generator driven by synthetic waypoint tracking.

Chapter 3 formulates the path-following problem and its adaptation within the qLMPC framework. It details the process of constructing the problem within this framework, beginning with the derivation of a qLPV representation for the aircraft's kinematics through velocity-based linearization. Subsequently, the chapter discusses the utilization of predictive control, incorporating linear constraints to attain a solution for the problem within reasonable computational time, while ensuring efficient performance.

Chapter 4 presents the extension of the proposed framework to include obstacle avoidance while maintaining the efficiency. This chapter demonstrates the conversion of nonlinear constraints imposed by obstacles into qLPV constraints, allowing to still solve a QP.

Chapter 5 analyzes the stability of the proposed algorithm, assuming convergence to an optimal rather than a suboptimal solution, as discussed in [Hespe and Werner, 2021]. The analysis involves augmenting the optimization problem with additional constraints, which can be introduced either through the use of terminal ingredients or by exploiting the structure of the velocity-based linearized model.

Chapter 6 summarizes our research findings and outlines potential future research directions.

The structure of this thesis is intended to be followed sequentially, with the exception of Chapters 4 and 5, which can be read independently after Chapter 3. To aid readers in understanding the organization of the research, a flowchart outlining the thesis structure is provided in Figure 1.3.

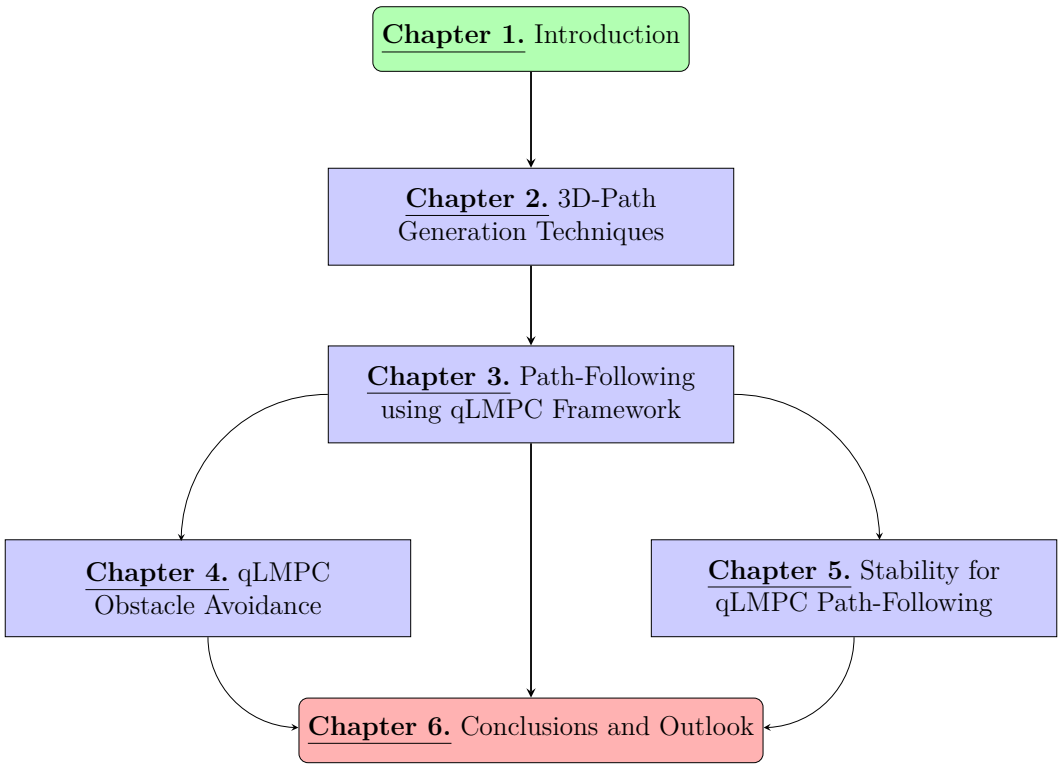

Figure 1.3: Flowchart illustrating recommended reading sequences and the organization of the chapters.

Chapter 2

From Waypoints to Trajectories: UAV Path-Generation

This chapter focuses on generating feasible trajectories that are continuous and sufficiently smooth to be tracked by fixed-wing UAVs. Ensuring the generated path respects system dynamics is essential for enabling safe and efficient autonomous flight, particularly in complex or constrained environments such as urban areas, mountainous terrain, or restricted airspace.

The choice and quality of the generated path play a pivotal role not only in avoiding static and dynamic obstacles, but also in optimizing energy consumption, mission time, and control performance. Therefore, generating a path that satisfies both geometric feasibility and dynamic trackability is a critical step between high-level mission planning and low-level control.

Before presenting the specific path-generation methods used in this thesis, it is essential to distinguish between path planning and path generation. Path planners operate at a high level, computing globally feasible, collision-free paths from a start point to a goal based on environmental representations such as occupancy grids or graphs. These paths are typically represented as discrete waypoints or piecewise-linear segments and are generated using classical algorithms such as A*, Dijkstra's algorithm, or sampling-based methods like Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM) [LaValle, 2006; Karaman and Frazzoli, 2011].

Various classes of planners exist to suit different requirements: grid-based planners discretize the environment into fixed cells for search [Thrun et al., 2005], sampling-based planners randomly explore the configuration space [Kothari et al., 2009; LaValle et al., 2001], and graph-based planners rely on connectivity representations [Hart et al., 1968]. More recent advances have introduced learning-based planners that leverage machine learning to generalize planning strategies from data [Bojarski et al., 2016], and optimization-based planners that formulate the task as a constrained minimization problem [Li and Tong, 2024]. Additionally, curvature-

constrained planners, such as Dubins paths, are designed for vehicles with turning constraints and seek the shortest feasible path between two configurations [Dai and Xie, 2015].

While path planners define *where* the vehicle should go, they do not typically account for vehicle dynamics, actuation limits, or the smoothness needed for real-time tracking. This is where path generators come into play. A path generator refines the discrete output of the planner into a smooth, continuous, and dynamically feasible trajectory. It considers constraints on velocity, acceleration, curvature, and sometimes higher-order derivatives such as jerk, ensuring that the resulting path can be reliably followed by a real vehicle [Richter et al., 2013; Oleynikova et al., 2016].

Popular path generation techniques include spline-based methods like B-splines and quintic polynomials, which offer continuity in position and derivatives [Whitney, 1987; Egerstedt and Hu, 2001], as well as differential flatness-based approaches, which transform complex nonlinear systems into simplified planning spaces [Fliess et. al, 1995]. Optimization-based generators also integrate seamlessly with control frameworks like MPC, enabling real-time trajectory tracking and online re-planning [Kamel et al., 2017]. While some architectures treat planning and generation as separate stages, more integrated approaches allow for feedback-informed adjustments to improve robustness in uncertain or dynamic environments.

In the following, we present two distinct path-generation strategies used in this thesis. The first is a *Hybrid Approach* that combines arc-length parameterization with cubic splines to ensure smoothness and flexibility. The second is a *Synthetic Waypoint Guidance* method, where a synthetic waypoint (SW) acts as a virtual tracking point (VTP), dynamically guiding the UAV along the reference path with adaptive behavior.

2.1 Arc-length Parameterized Spline Curves

The concept of applying this technique originates from [Wang et al., 2002], where the authors proposed arc-length parameterization with spline curves within the domain of computer animation and virtual environments to define motion paths for controlling the movements of synthetic objects in real-time applications [Willemsen et al., 2003].

Parametric cubic splines are commonly chosen curves for various applications. The process of approximating a smooth curve can be broken down into three steps. Firstly, the arc lengths of all the cubic segments in the input spline curve, denoted as $S(t)$, are computed and summed to determine the total arc length, denoted as L, of $S(t)$. Secondly, $m+1$ points are evenly distributed along $S(t)$. Lastly, a new spline curve is computed using these equally spaced points as knots. The outcome

is an approximately arc-length parameterized piecewise spline curve, segmented into m cubic sections. The arc length of each spline segment on the input curve is

$$l_i = \int_{t_i}^{t_{i+1}} \sqrt{(x(t))^2 + (y(t))^2 + (z(t))^2}\, dt. \tag{2.1}$$

The parameter i ranges from 0 to $n-1$, where n denotes the number of spline segments in the original curve. Consequently, the total arc length of the curve is given by $L = \sum_{i=0}^{n-1} l_i$. Employing the bisection method, we compute $m+1$ evenly distributed points along $S(t)$, situated at intervals of $0, \tilde{l}, 2\tilde{l}, \cdots, m\tilde{l}$ from the curve's starting point. Here, $\tilde{l} = L/m$ represents the length of each segment in the resulting curve. These points are determined by parameter values $\tilde{t}_0, \tilde{t}_1, \cdots, \tilde{t}_m$, which satisfy the following integration

$$\int_{t_0}^{\tilde{t}_i} \frac{d\theta}{dt}\, dt = i.\tilde{l}, \tag{2.2}$$

where $i = 0, 1, \cdots, m$, θ represents the arc length parameter, and t denotes the parameter of the spline functions. The value of $\tilde{t}_i$ can be computed in two steps. The first step involves finding a spline segment indexed by j that satisfies $\sum_{p=0}^{j-1} l_p \preceq i \cdot \tilde{l} \prec \sum_{p=0}^{j} l_p$. This condition ensures that $t_j \preceq \tilde{t}_i \prec t_{j+1}$. Equation (2.2) can be expressed as

$$\int_{t_0}^{\tilde{t}_i} \frac{d\theta}{dt}\, dt = \int_{t_0}^{t_j} \frac{d\theta}{dt}\, dt + \int_{t_j}^{\tilde{t}_i} \frac{d\theta}{dt}\, dt = \sum_{p=0}^{j-1} l_p + \int_{t_j}^{\tilde{t}_i} \frac{d\theta}{dt}\, dt = i.\tilde{l}. \tag{2.3}$$

The second step is to compute $\tilde{t}_i$ such that

$$\int_{t_j}^{\tilde{t}_i} \frac{d\theta}{dt}\, dt = i.\tilde{l} - \sum_{p=0}^{j-1} l_p, \tag{2.4}$$

where $\tilde{t}_i$ corresponds to the cubic spline segment beginning with the parameter value t_j. The second step employs the bisection method. We initialize $t_{\text{left}} = t_j$ and $t_{\text{right}} = t_{j+1}$. The interval $[t_{\text{left}}, t_{\text{right}}]$ contains the solution $\tilde{t}_i$. This interval is divided into two subintervals, namely $[t_{\text{left}}, t_{\text{middle}}]$ and $[t_{\text{middle}}, t_{\text{right}}]$, where $t_{\text{middle}} = \frac{t_{\text{left}} + t_{\text{right}}}{2}$. The arc length between $[t_j, t_{\text{middle}}]$ can be calculated as $\Delta\theta = \int_{t_j}^{t_{\text{middle}}} \frac{d\theta}{dt}\, dt$. If $\Delta\theta \prec i \cdot (\tilde{l} - \sum_{p=0}^{j-1} l_p)$, the solution lies in the upper subinterval $[t_{\text{middle}}, t_{\text{right}}]$; otherwise, it lies in the lower subinterval $[t_{\text{left}}, t_{\text{middle}}]$. The process of bisecting continues until the desired error tolerance for the arc length is reached.

Using the bisection method described above, we obtain $\tilde{t}_0, \tilde{t}_1, \cdots, \tilde{t}_m$, which divide the original curve into equal arc-length segments. Then, utilizing the original cubic spline function, we compute evenly spaced points $(\tilde{x}_0, \tilde{y}_0, \tilde{z}_0)$, $(\tilde{x}_1, \tilde{y}_1, \tilde{z}_1), \cdots, (\tilde{x}_m, \tilde{y}_m, \tilde{z}_m)$ at arc-lengths $\theta_0 = 0, \theta_1 = \tilde{l}, \theta_2 = 2\tilde{l}, \cdots, \theta_m = m\tilde{l}$.

Next, we reparameterize the spline curve by interpolating the points $(\theta_0, \tilde{x}_0)$, $(\theta_1, \tilde{x}_1)$, $\cdots$, $(\theta_m, \tilde{x}_m)$ and $(\theta_0, \tilde{z}_0)$, $(\theta_1, \tilde{z}_1)$, $\cdots$, $(\theta_m, \tilde{z}_m)$. In this interpolation, we aim to interpolate x, y, and z relative to the arc-length θ. This results in cubic spline functions, as expressed in (2.6). Our goal is to obtain the following outcome for $\theta \in [0, L]$

$$\sqrt{(\tilde{x}(\theta))^2 + (\tilde{y}(\theta))^2 + (\tilde{z}(\theta))^2} = 1. \tag{2.5}$$

Note that the notation $\tilde{*}$ is utilized here to denote the equally spaced new position points. Hence, the magnitude of the initial tangent vector and the magnitude of the final tangent vector should both be 1. The resulting curve will take the form

$$\begin{aligned}
\tilde{x}(\theta) &= \tilde{a}_{x,i}(\theta - \theta_i)^3 + \tilde{b}_{x,i}(\theta - \theta_i)^2 + \tilde{c}_{x,i}(\theta - \theta_i) + \tilde{d}_{x,i} \\
\tilde{y}(\theta) &= \tilde{a}_{y,i}(\theta - \theta_i)^3 + \tilde{b}_{y,i}(\theta - \theta_i)^2 + \tilde{c}_{y,i}(\theta - \theta_i) + \tilde{d}_{y,i} \\
\tilde{z}(\theta) &= \tilde{a}_{z,i}(\theta - \theta_i)^3 + \tilde{b}_{z,i}(\theta - \theta_i)^2 + \tilde{c}_{z,i}(\theta - \theta_i) + \tilde{d}_{z,i},
\end{aligned} \tag{2.6}$$

where $\theta \in [\theta_i, \theta_{i+1}]$, $i = 0, 1, 2, \cdots, m-1$, and the values for $\tilde{x}$, $\tilde{y}$, and $\tilde{z}$ are of class C^2 on $[0, L]$. The tangent vectors at the initial and final points of the derived curve are adjusted to match the normalized tangent vectors of the original curve at those points, respectively.

Thus far, we have outlined the process of parameterizing a spline curve $S(t)$ using the arc length parameter θ. Now, the pertinent question arises: how can we integrate this method to dynamically evolve a trackable path for a UAV?

To address this question, we employ the approach outlined in [Faulwasser et al. 2009]. We can simply add an additional state to the system to dynamically describe the path evolution. While this state is virtual, its inclusion is crucial for designing the path-following controller. By controlling this state with a virtual control input, we can determine the speed of the path evolution.

This speed can be adjusted relative to the velocity of the aircraft, and the path-following controller can determine it along with the aircraft speed. Another benefit of this approach within the predictive framework is the ability to impose constraints on the arc length evolution. This ensures the correct evolution of the path in the desired direction, alleviating the path-following controller's workload to focus solely on adhering to the predetermined path. Here we choose the path dynamics

$$\dot{\theta} = \lambda\theta + v. \tag{2.7}$$

In our application, we chose λ as a scalar set to -0.001, in order to maintain stability of the arc length parameter. Additionally, v serves as the virtual control input responsible for regulating the path evolution. As previously mentioned, it is the responsibility of the path-following controller to determine v, in conjunction with the aircraft's control inputs, to ensure accurate path-following. We illustrate

the arc-length dynamics described in (2.7) under the condition where the velocity, v, is zero in Fig. 2.1 which depicts the linear relationship between the arc-length parameter, θ, and its derivative, $\dot{\theta}$. In the subsequent chapter, we will demonstrate how imposing constraints on this relationship ensures the path evolves in the correct direction. Furthermore, we will show how the path evolution can be controlled by the velocity, v, as determined by the path-following controller.

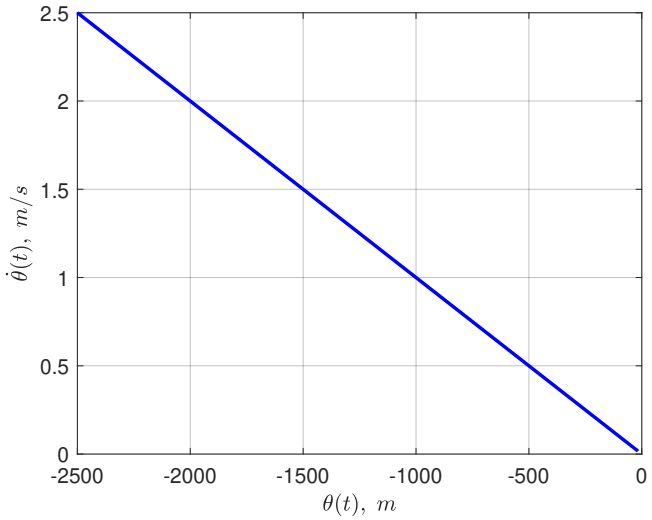

Figure 2.1: Arc length dynamics when $v = 0$.

We now have a spline-based curve parameterized by arc length and composed of m equal-length segments. This method mitigates the high computational burden of traditional arc-length parameterization by generating a spline that closely approximates the original curve while inherently preserving arc-length parametrization.

This approach offers a significant advantage by transferring the heavy computational workload to offline preprocessing stages. Consequently, online computations can be performed swiftly. Moreover, the precision of the online computations can be predetermined during the preprocessing phase by adjusting the resolution of the approximation curve.

The primary online cost incurred when enhancing the accuracy of the approximation lies in the number of coefficients of the spline functions that need to be stored. However, there is a drawback concerning paths that may require online alterations, which is not feasible with this method. Essentially, the path must be predetermined as the coefficients are dependent on the trajectory. Therefore, a new approach to path-generation addressing this limitation is warranted.

In the subsequent section, we will introduce the SW path-planner as an example of such methodologies. The choice between these methodologies depends on the nature of the path-following application. For scenarios where path alterations are unlikely, the hybrid path-planner is suitable, whereas for applications where online path replanning is more probable, the SW path-planner is preferable.

2.1.1 Illustrative Example: Hybrid 3D Path Generation

Next, we evaluate both methodologies by testing them on a set of waypoints to create a feasible path for the fixed-wing UAV. We selected two paths previously used in flight experiments with the same aircraft [Sedlmair et al., 2019], employing cubic splines alongside an NLGL path-following controller. Each path comprises nine waypoints, with the first and last waypoints being identical to form a closed loop. The first path resembles an "Aerodrome circuit," while the second is more challenging, resembling a "Roller coaster" ride. Altitude variations in the first path range from 200 m to 270 m, while those in the second path range from 180 m to 280 m. Table 2.1 displays the waypoints for both paths, in the ECEF coordinate system.

In Fig. 2.2, we demonstrate the formulation of a smooth path for the 'Aerodrome-circuit' using the set of the determined waypoints (blue circles). By employing cubic splines with equal intervals ($m = 11$), the resulting path (shown in red) is smooth and effectively passes through all waypoints (in black). Additionally, in Fig. 2.3, we visualize the individual components of the path in terms of x, y, z, and their time derivatives to verify continuity and suitability for tracking by the fixed-wing aircraft. We applied the same approach to the second scenario

Table 2.1: Waypoints, $[x, y, z]$ used for path definition.

No.	Position, m	No.	Position, m	No.	Position, m
'Aerodrome-circuit' Path					
1	[110,-40,220]	4	[551,122,270]	7	[-157,272,200]
2	[333,-177.7,220]	5	[203.5,480,250]	8	[-57,72,220]
3	[511,-100,250]	6	[-97,422.3,200]	9	[110,-40,220]
'Roller-coaster' Path					
1	[155,315,220]	4	[400,420,250]	7	[-190,490,280]
2	[260,50,250]	5	[155,315,180]	8	[40,600,250]
3	[500,150,280]	6	[-80,220,250]	9	[155,315,220]

(the Roller-coaster), as depicted in Fig. 2.4. The resulting path is smooth with equal intervals, similar to the previous scenario ($m = 11$). Additionally, Fig. 2.5 illustrates the continuity of the individual components and their time derivatives, ensuring trackability by our aircraft. We denote the resulting path as $\mathcal{P}(\theta)$, where $\forall\theta \in [0, \hat{\theta}] : \theta \rightarrow \mathcal{P}(\theta) \in \mathcal{X} \subseteq \mathbb{R}^n$, and $\mathcal{P}(0) = 0$. Hence, the path $\mathcal{P}(\theta)$ to be followed is defined as $\mathcal{P}(\theta) \in \mathbb{R}^n : [0, \hat{\theta}] \rightarrow \mathcal{P}(\theta)$. The reference path is now

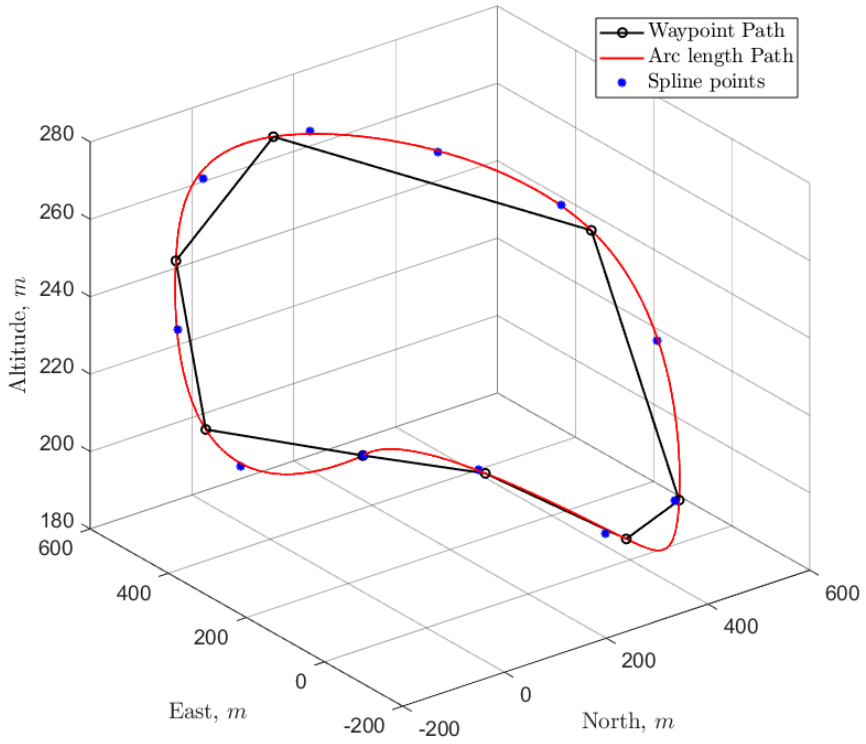

Figure 2.2: 'Aerodrome-circuit' path generation using the hybrid approach.

denoted by $[x_r(\theta)\ \ y_r(\theta)\ \ z_r(\theta)\ \ \psi_r(\theta)\ \ \gamma_r(\theta)]^\top$, defined as

$$\mathcal{P}(\theta) = \begin{bmatrix} a\theta^3 + b\theta^2 + c\theta + d \\ e\theta^3 + f\theta^2 + g\theta + h \\ i\theta^3 + j\theta^2 + k\theta + l \\ \tan^{-1} \dfrac{dy(\theta)}{dx(\theta)} \\ \sin^{-1} \dfrac{dz(\theta)}{\sqrt{dx^2(\theta) + dy^2(\theta)}} \end{bmatrix}.$$

The constant coefficients $a, b, c, d, e, f, g, h, i, j, k$, and l are determined based on the path waypoints and computed beforehand. These coefficients are stored in the aircraft's memory. Subsequently, the path is dynamically updated online to accommodate changes in θ.

2.2 Synthetic Waypoint Path-Planner

In this section, we explore an alternative path-generation method to address the limitations of the hybrid approach discussed in Section 2.1. As shown earlier, the hybrid approach requires preprocessing steps to parameterize the path based on arc length according to changes in the waypoints. Here, we present the SW path-planner, inspired by [Medagoda and Gibbens, 2010], where the authors employed it along with pure pursuit as a guidance approach.

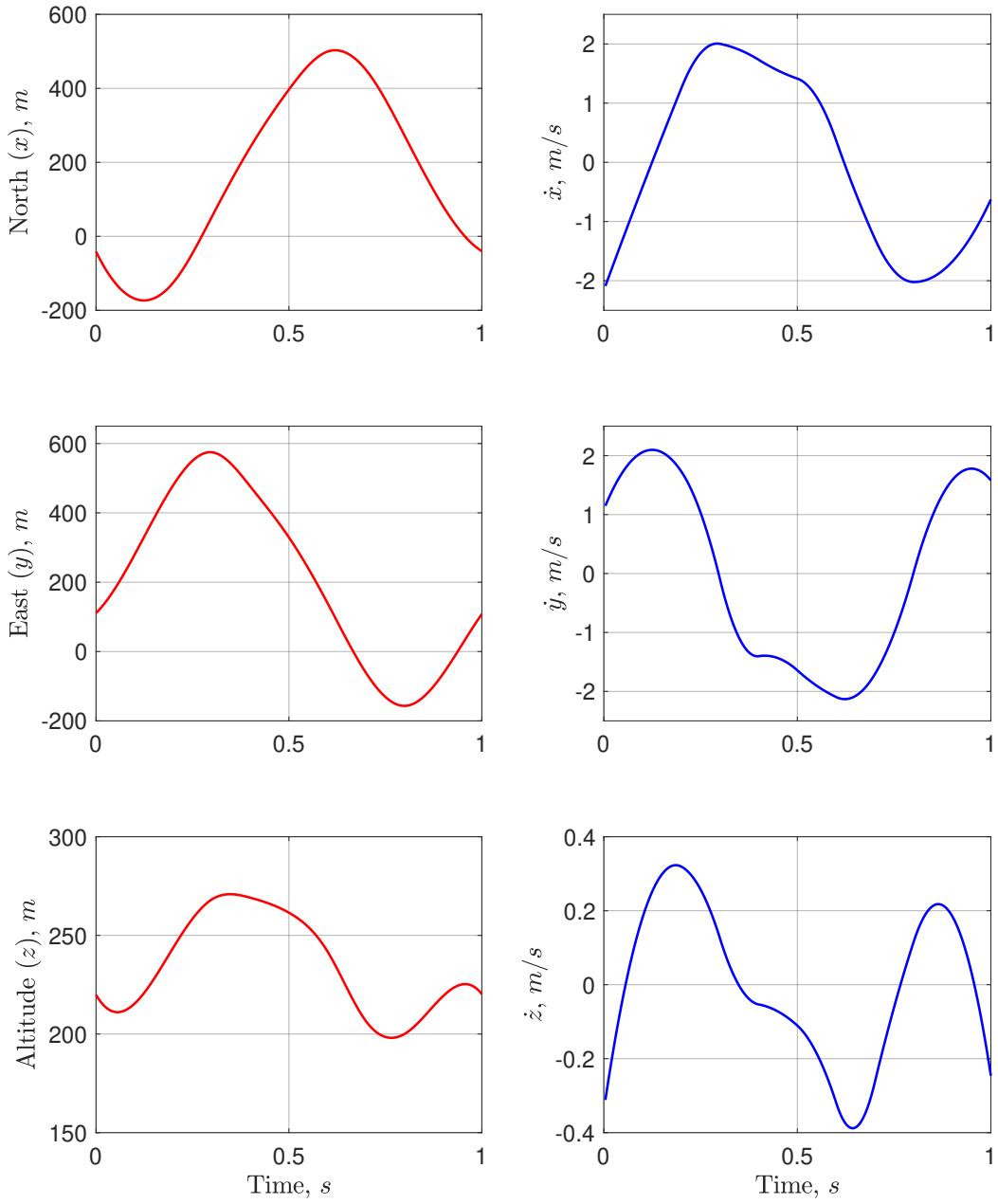

Figure 2.3: 'Aerodrome-circuit' path position components and its time derivatives.

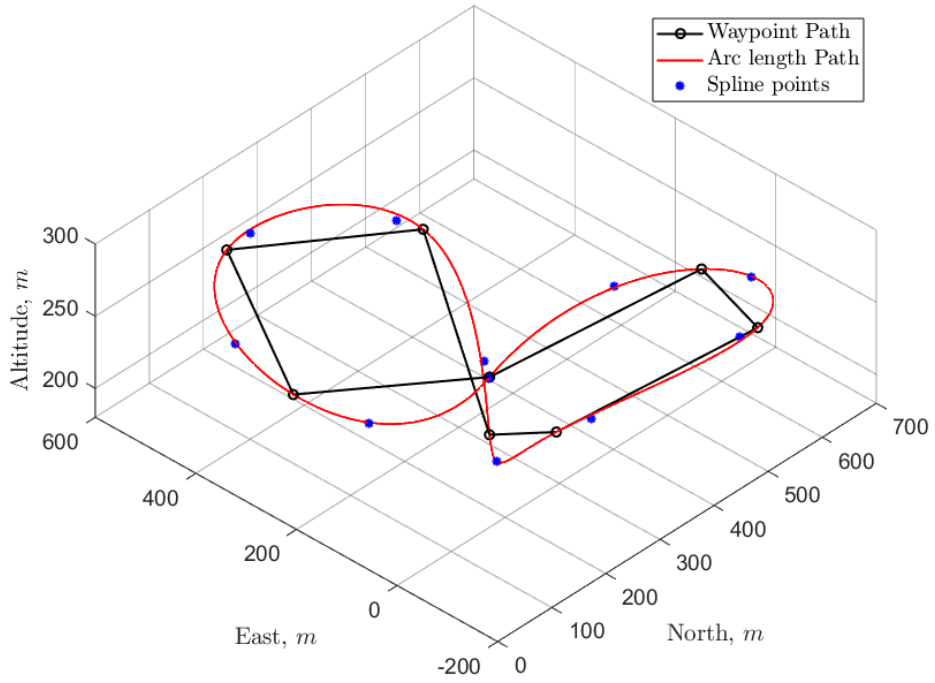

Figure 2.4: 'Roller-coaster' path generation using the hybrid approach.

This path-planner serves as an online path-planner and only requires prior knowledge of the initial waypoint, which functions as the starting location for the SW. In the SW algorithm, a flight path is defined with waypoints specified in inertial space relative to a fixed local frame. Subsequently, a SW is placed at the location of the first waypoint. The SW algorithm operates by tracking this SW as it moves along the designated flight path. The synthetic waypoint is termed as such because its position is derived from projecting the anticipated position of the aircraft within a predefined time horizon. This time horizon, determined by the user, dictates the interval over which the aircraft will follow the SW. As the aircraft nears the SW along the flight path, the algorithm dynamically adjusts its position, thereby prompting the aircraft to adjust its heading in response to the changes in the SW's position.

2.2.1 Synthetic Waypoint Law

Here, we illustrate the operation of the SW path-planner. Initially, the aircraft maintains a distance R from the flight path, which is determined by the norm distance between the aircraft and the SW positioned at the first waypoint to be followed. Subsequently, ψ_c represents the commanded heading angle for the aircraft to steer towards the SW from its current location. As the distance R diminishes to a specified threshold R^* determined by the user-defined time horizon T_H (see Equaiton (2.10)), the SW begins traversing the path between the first and second waypoints, guided by a heading reference denoted as ψ_{ref}, which is determined by the flight path geometry.

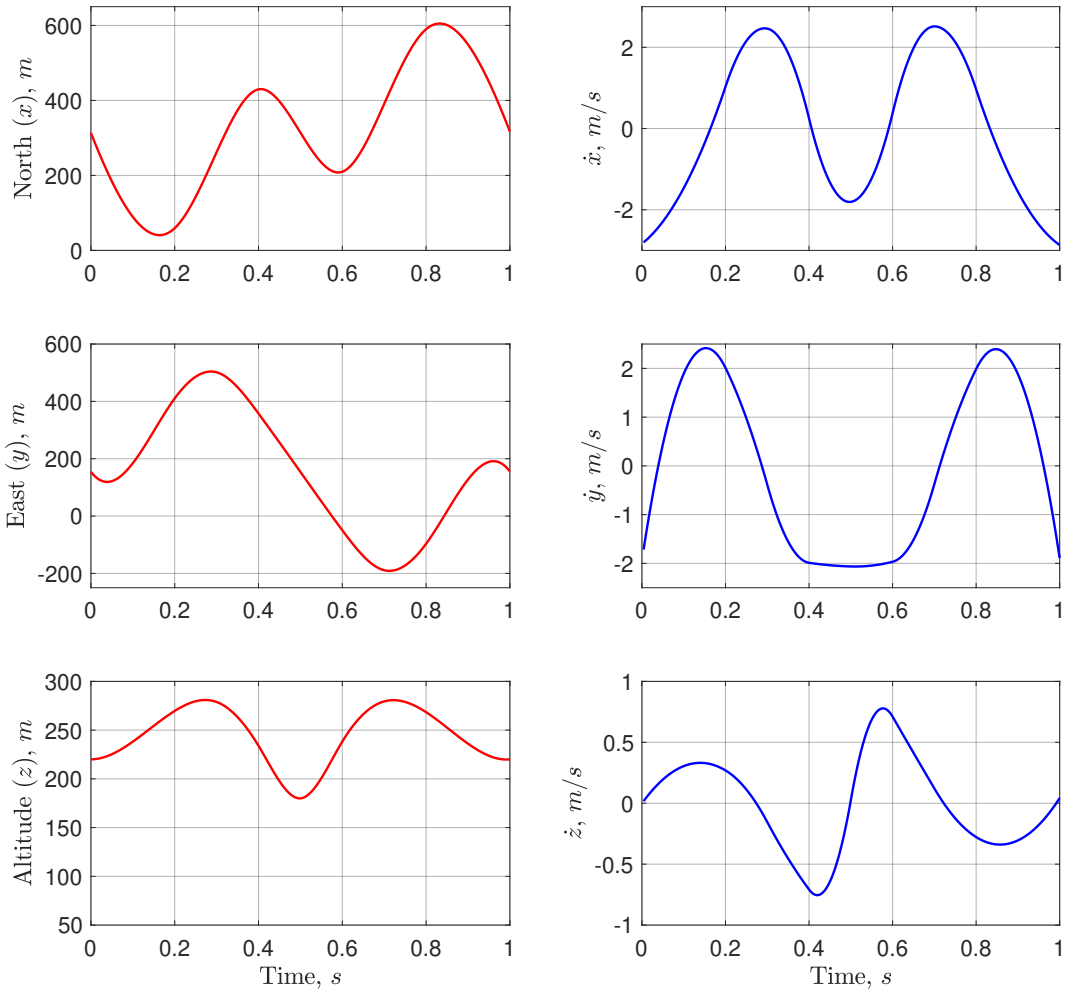

Figure 2.5: 'Roller-coaster' path position components and its time derivatives.

The key question arising from the previous illustration of the SW algorithm's operation is how to ascertain the distance by which the aircraft trails the SW and how to select the time horizon used in this calculation. This inquiry will be addressed in the subsequent discussion. To enable successful tracking of the SW by the aircraft, certain constraints must be applied to the dynamics of the SW. Firstly, the SW is constrained to travel solely along the flight path delimited by the two active waypoints. Here, "active" refers to the two waypoints between which the SW is presently positioned. This constraint is crucial to ensure that the SW's movement remains aligned with the designated path, thereby preventing the aircraft from deviating onto an incorrect trajectory. Tracking an erroneous point could result in the aircraft straying from the intended path. Mathematically, this constraint can be formulated as

$$\begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix}_i = \begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix}_{i-1} + V_s \begin{bmatrix} \cos\gamma_{ref} \cos\psi_{ref} \\ \cos\gamma_{ref} \sin\psi_{ref} \\ -\sin\gamma_{ref} \end{bmatrix} \Delta T. \tag{2.8}$$

The three-dimensional position of the SW, denoted by X_s, Y_s, and Z_s, evolves along the path within inertial space. Its velocity is represented by V_s. Additionally, ψ_{ref} and γ_{ref} specify the reference heading angle and climb angle of the path segment between the two currently active waypoints. The time interval ΔT determines the frequency at which the SW's position is updated as per the designer's specifications (sampling time). The second constraint to be implemented in the algorithm concerns the minimum distance between the aircraft and the SW. This distance must be maintained within a certain range to ensure a successful and seamless transition between waypoints. The constraint dictates that the aircraft cannot approach the SW closer than this minimum distance, denoted here as R^*. Mathematically, this condition can be expressed as

$$R \geq R^*. \tag{2.9}$$

where R denotes the closing range between the aircraft and the current position of the SW. This virtual distance can be represented as

$$R^* = V_a T_H, \tag{2.10}$$

where V_a represents the current airspeed of the aircraft and T_H denotes the desired time horizon for the vehicle to respond to the flight path changes. It is evident that this distance R^* is not directly set by the designer but can be indirectly influenced by adjusting T_H to determine the level of performance achieved by the aircraft based on the intended goal. We can liken this time horizon to the prediction horizon that will be discussed in Chapter 3 within the context of MPC, where the prediction horizon specifies a duration over which control actions are determined to achieve a future objective. The velocity of the SW is computed as the quotient of the virtual closing distance R^* and the current closing distance R, multiplied

by the current airspeed of the aircraft. This criterion guarantees that the aircraft is continuously advancing toward the SW. Mathematically, this condition can be expressed as

$$V_s = V_a \frac{R^*}{R}. \tag{2.11}$$

Algorithm 2.1 outlines the functioning of the SW path-planner.

Algorithm 2.1 Synthetic Waypoint Path-Planner

Initialization: Initial position (start), destination position (goal), max distance
1: Current position ← start
2: Create an empty list synthetic waypoints
3: **while** distance(current position, goal) ≻ max distance **do**
4: Heading vector ← normalize(goal - current position)
5: Next waypoint ← current position + max distance × heading vector
6: Add next waypoint to synthetic waypoints
7: Current position ← next waypoint
8: **end while**

2.2.2 Path-following Control Action

The objective of the path-planner is to generate the essential reference components for the path-following controller. In this study, we define the reference (r) such that $r \in \mathbb{R}^5$. Up to this point, we have computed the initial three reference components, delineating the position components in (2.8). Subsequently, the remaining two reference components, which specify the path heading ψ_c and climb changes γ_c, can be determined using the components of R as follows

$$\begin{aligned} \psi_c &= \tan^{-1}\left(\frac{R_y}{R_x}\right), \\ \gamma_c &= \sin^{-1}\left(\frac{R_z}{R}\right). \end{aligned} \tag{2.12}$$

The components R_x, R_y, and R_z are defined as the differences between the SW's position (X_s, Y_s, Z_s) and the aircraft's position (X_a, Y_a, Z_a) in the inertial frame. Figure 2.6 illustrates the three phases of path tracking using the SW methodology. It begins with the initial phase where the aircraft is distant from the SW (a), followed by the mid-course phase after both the SW and the aircraft have moved (b), and concludes with the aircraft achieving tracking of the SW (c). The stability analysis of this algorithm is discussed in [Medagoda, 2010].

2.3 Summary

This chapter presented two distinct methodologies for path generation, both aimed at producing continuous and differentiable trajectories from a predefined set of

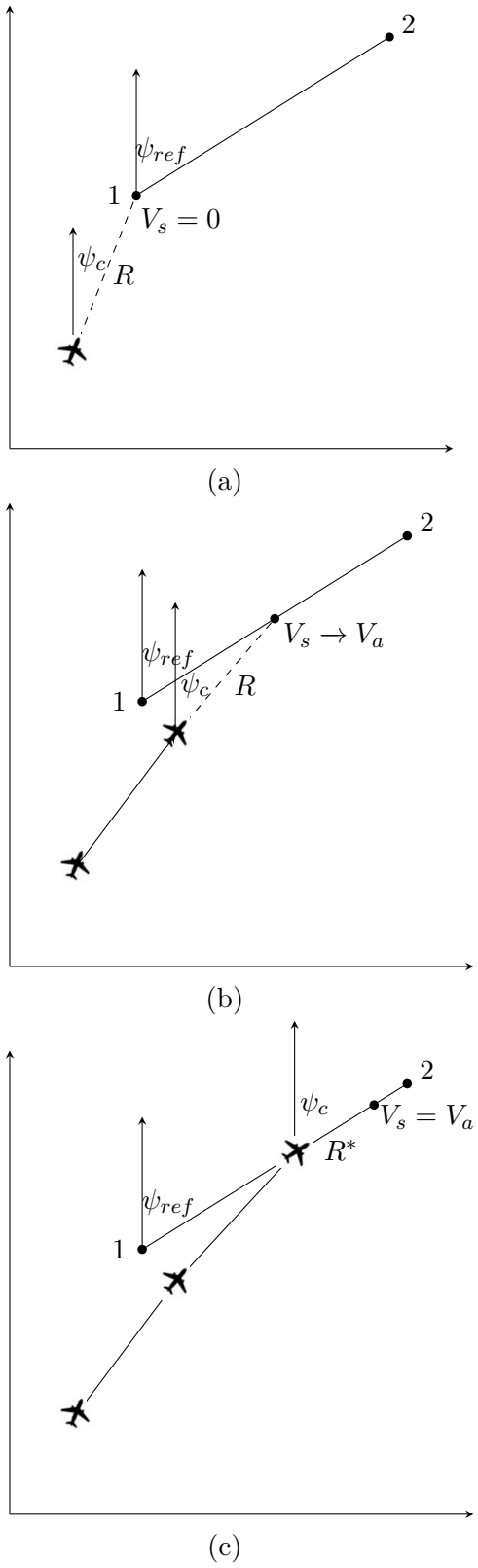

Figure 2.6: Synthetic waypoint dynamics: a) initial phase, b) midphase, c) path-acquisition.

waypoints, ensuring compatibility with the nonholonomic constraints of fixed-wing UAVs.

The first method adopts a hybrid approach, combining arc-length parameterization with cubic spline interpolation to generate smooth trajectories. While this technique provides high path smoothness, it requires offline preprocessing and a predefined set of waypoints. In contrast, the second method leverages synthetic waypoint tracking, which utilizes dynamically generated virtual waypoints that can be updated online in response to changes during flight. This approach enables real-time adaptability. And, it results in less smooth paths by connecting waypoints with straight segments and applying smoothing only at transitions between segments.

In Chapter 3, we introduce the qLMPC path-following problem and present simulation results based on the scenarios outlined in this chapter.

Chapter 3

qLMPC Framework for UAV Path-Following

This chapter details the main contribution of this work, documented in [Samir et al., 2024a] focusing on formulating the path-following problem within the quasi-linear parameter-varying model predictive control (qLMPC) framework. The fundamental concept involves utilizing a model that captures the nonlinear dynamics of a fixed-wing UAV's position, subsequently representing this model within a qLPV form. This representation allows for embedding model nonlinearities within a semi-linear framework. The optimal control problem is solved as a QP at every time step, a computationally efficient method. This OCP is solved iteratively to ensure convergence to an (sub)optimal solution, thereby achieving high path-following performance. The qLMPC method was recently introduced in [Cisneros and Werner, 2020].

We use a 3D kinematic model to represent the aircraft's position dynamics. By leveraging velocity-based linearization, we obtain a qLPV representation, providing an accurate depiction of the model's nonlinearities. This representation facilitates the use of offset-free model predictive control (MPC), known for its robustness against disturbances. Moreover, in the next chapter, we will illustrate how this framework can be augmented to tackle the obstacle avoidance problem by incorporating additional constraints. Additionally, we will discuss how stability can be guaranteed using the same approach by adding a terminal constraint. Moreover, in this chapter, we highlight the computational efficiency of this algorithm, which facilitates its real-time implementation on hardware, potentially enabling future flight experiments.

This chapter is structured as follows: In Section 3.1, we discuss the velocity-based linearization strategy building upon the concepts of velocity algorithms. Within this section, we represent the mathematical derivation, advantages, and limitations of velocity-based linearization. Additionally, we demonstrate how to derive a velocity-based linearized model from the kinematic model of the fixed-

wing aircraft, serving as the foundational model for the predictive path-following controller. Following that, in Section 3.3, we introduce the fixed-wing aircraft utilized in our simulations, named ULTRA-Extra [Krings et al., 2013]. We illustrate the process of designing its attitude and stabilization controller using the Successive-Loop-Closure (SLC) strategy, as outlined in [Sedlmair et al., 2019; Beard, 2012]. Subsequently, we analyze the simulation results obtained through the proposed methodology for both scenarios introduced in Chapter 2, employing both path-generators. This analysis demonstrates efficient path-following performance coupled with expedient computational times. Finally, in Section 3.5, we summarize the outcomes of the chapter and outline preparations for the subsequent steps.

3.1 Velocity-based Linearization

In the following, we demonstrate how to leverage velocity algorithms within the MPC strategy. Moreover, we present the velocity-based linearization approach and its integration into the predictive framework. The need for adopting velocity-based linearization [Leith and Leithead, 1998a] arises due to constraints encountered when employing Jacobian Linearization. Its primary drawback lies in its reliance on an equilibrium point, restricting its validity to the vicinity of this equilibrium. Additionally, its affine representation of dynamics can complicate linear control design. However, the primary drawback of employing velocity-based linearization is that it leads to higher-dimensional models (n_x+n_y), which can result in increased computational complexity. Additionally, this approach requires measurements of all states or additional observer design. Consider a nonlinear continuous time system

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

Definition 3.1. *(qLPV model from Velocity-based linearization) [Pablo Cisneros, 2021], Assume $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}^l$ in (3.1) are differentiable, then a velocity-based linearization is obtained by differentiating the equations of motion w.r.t. time, i.e.*

$$\begin{aligned} \ddot{x} &= \nabla_x f(x,u)\dot{x} + \nabla_u f(x,u)\dot{u} \\ \dot{y} &= \nabla_x h(x)\dot{x}. \end{aligned} \tag{3.2}$$

The model represented in (3.2) is linear in the state and input derivatives, and it features state-space matrices dependent on states and inputs. It is crucial to select suitable initial conditions $x(0), u(0), y(0) = h(x(0))$, and $\dot{x}(0) = f(x(0), u(0))$ when using velocity-based linearization. By defining the augmented state vector

as $[y^\top \ \dot{x}^\top]^\top$ in order to preserve absolute information, then a velocity-form is given by

$$\begin{aligned} \begin{bmatrix} \dot{y} \\ \ddot{x} \end{bmatrix} &= \begin{bmatrix} 0 & \nabla_x h(x) \\ 0 & \nabla_x f(x,u) \end{bmatrix} \begin{bmatrix} y \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \nabla_u f(x,u) \end{bmatrix} \dot{u} \\ y &= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{x} \end{bmatrix}. \end{aligned} \tag{3.3}$$

By defining $\rho_i = e_j \begin{bmatrix} x \\ u \end{bmatrix}$, $i = 1, \cdots, n_\rho$, where e_j is the j^{th} row of an identity matrix, this state-dependent linear system can be expressed as a qLPV system. The augmented model (3.3) is then discretized using Runge-Kutta4 (RK4) as

$$\begin{aligned} x_{k+1} = (\frac{1}{24}A_\rho^4 T_s^4 + \frac{1}{6}A_\rho^3 T_s^3 + \frac{1}{2}A_\rho^2 T_s^2 + A_\rho T_s + I)x_k \\ + (\frac{1}{24}A_\rho^3 B_\rho T_s^4 + \frac{1}{6}A_\rho^2 B_\rho T_s^3 + \frac{1}{2}A_\rho B_\rho T_s^2 + B_\rho T_s)u_k. \end{aligned} \tag{3.4}$$

After discretization, we can write the discretized velocity-model as

$$\begin{aligned} \begin{bmatrix} y_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} &= A(\rho_k) \begin{bmatrix} y_k \\ \dot{x}_k \end{bmatrix} + B(\rho_k)\Delta u_k \\ y_k &= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} y_k \\ \dot{x}_k \end{bmatrix}. \end{aligned} \tag{3.5}$$

Velocity-Based qLPV Kinematics for Fixed-Wing UAVs

Prior to implementing the predictive path-following strategy on the high-fidelity model, it is initially applied to a simplified kinematic model of the ULTRA-Extra, known as the 'point mass model'. This model characterizes the aircraft's motion solely based on its position, velocity, and orientation in space and time, without considering any external forces or moments. We utilize this model to derive a velocity-based qLPV model, which is subsequently incorporated into the predictive framework. This model is represented as

$$\begin{aligned} \dot{x}_\mathrm{p} &= V_a \cos\gamma \cos\chi \\ \dot{y}_\mathrm{p} &= V_a \cos\gamma \sin\chi \\ \dot{z}_\mathrm{p} &= -V_a \sin\gamma \\ \dot{\chi} &= \frac{g}{V_g} \tan\phi \\ \dot{\gamma} &= \frac{g}{V_g}(n_z \cos\phi - \cos\gamma). \end{aligned} \tag{3.6}$$

By applying velocity-based linearization, presented in (3.3), to the model described in (3.6), we derive the velocity kinematic model as follows:

$$\begin{bmatrix} \dot{x}_\mathrm{p} \\ \dot{y}_\mathrm{p} \\ \dot{z}_\mathrm{p} \\ \dot{\chi} \\ \dot{\gamma} \\ \ddot{x}_\mathrm{p} \\ \ddot{y}_\mathrm{p} \\ \ddot{z}_\mathrm{p} \\ \ddot{\chi} \\ \ddot{\gamma} \end{bmatrix} = \left[\begin{array}{c|ccc} 0_l & \multicolumn{3}{c}{I_{l\times n}} \\ \hline 0_{n\times l} & 0_{n\times 3} & \begin{matrix} -V_a\cos\gamma\sin\chi \\ V_a\cos\chi\cos\gamma \\ 0 \\ 0 \\ 0 \end{matrix} & \begin{matrix} -V_a\cos\chi\sin\gamma \\ -V_a\sin\chi\sin\gamma \\ V_a\cos\gamma \\ 0 \\ \frac{g}{Va}\sin\gamma \end{matrix} \end{array}\right] \begin{bmatrix} x_\mathrm{p} \\ y_\mathrm{p} \\ z_\mathrm{p} \\ \chi \\ \gamma \\ \dot{x}_\mathrm{p} \\ \dot{y}_\mathrm{p} \\ \dot{z}_\mathrm{p} \\ \dot{\chi} \\ \dot{\gamma} \end{bmatrix} + \left[\begin{array}{ccc} \multicolumn{3}{c}{0_{l\times m}} \\ \hline \cos\chi\cos\gamma & 0 & 0 \\ \cos\gamma\sin\chi & 0 & 0 \\ \sin\gamma & 0 & 0 \\ -\frac{g}{V_a^2}\tan\phi & \frac{g\tan\phi^2+1}{V_a} & 0 \\ \frac{g\cos\gamma - n_z\cos\phi}{V_a^2} & -\frac{gn_z\sin\phi}{V_a} & \frac{g\cos\phi}{V_a} \end{array}\right] \begin{bmatrix} \dot{V}_a \\ \dot{\phi} \\ \dot{n}_z \end{bmatrix}. \tag{3.7}$$

Note that this model now comprises ten states instead of the original five, highlighting a primary drawback of velocity-based models as previously discussed. In our research, this expanded state-space is not problematic when extended to the high-fidelity model, given that the utilized model does not incorporate sensor dynamics. Nevertheless, incorporating sensor dynamics could pose a challenge. To address such complexity, one potential solution, proposed in [Rieck et al., 2023], involves utilizing Laguerre functions to simplify the design of a stabilizing controller for fixed-wing UAVs based on a dynamic velocity model.

3.2 Predictive Path-following Control

The objective is to address the nonlinear position dynamics of the fixed-wing aircraft in a (semi)-linear format, enabling the solution of the optimal control problem as a QP. As detailed in Section 3.1, we obtain a qLPV for these position dynamics using velocity-based linearization. In the following discussion, we outline formulating an appropriate cost function and constraints to tackle the path-following problem effectively. Additionally, we present how to frame the OCP as a QP, enabling the use of efficient solvers for resolution. A general cost function, as

depicted in (B.8), can be expressed as follows

$$J = \sum_{i=1}^{N} x_{k+i}^{\top} Q x_{k+i} + u_{k+i-1}^{\top} R u_{k+i-1}. \tag{3.8}$$

Alternatively, in a more concise format utilizing the stacked vectors outlined in B.9, it can be expressed as

$$J_k = X_k^{\top} \tilde{Q} X_k + U_k^{\top} \tilde{R} U_k, \tag{3.9}$$

where

$$\tilde{Q} = \begin{bmatrix} Q & & & \\ & \ddots & & \\ & & Q & \\ & & & P \end{bmatrix}, \qquad \tilde{R} = \begin{bmatrix} R & & & \\ & \ddots & & \\ & & \ddots & \\ & & & R \end{bmatrix}. \tag{3.10}$$

Note that (3.9) depends on the stacked vectors X_k and U_k. One can get rid of this dependency by substituting the prediction equation

$$X_k = H x_k + S U_k, \tag{3.11}$$

in (3.9). The resulting cost function from this substitution is as follows

$$J_k = U_k^{\top}(S^{\top}\tilde{Q}S + \tilde{R})U_k + x_k^{\top} H^{\top} \tilde{Q} S U_k + U_k^{\top} S^{\top} \tilde{Q} H x_k + x_k^{\top} H^{\top} \tilde{Q} H x_k. \tag{3.12}$$

A QP is formed as follows

$$\min_{u} \ \frac{1}{2} u^{\top} H_s u + g^{\top} u \qquad s.t. \qquad Au \preceq b. \tag{3.13}$$

In the optimization problem represented by (3.13), $u \in \mathbb{R}^p$ denotes the vector of decision variables, $H_s \in \mathbb{R}^{p \times p}$ is the Hessian matrix, $g \in \mathbb{R}^p$ is the gradient vector, $A \in \mathbb{R}^{q \times p}$ is a matrix multiplied by the decision variables, and $b \in \mathbb{R}^q$ is a vector of constraints. Here, p represents the number of decision variables, and q indicates the number of constraints. The objective of solving this OCP is to determine the vector u that minimizes the given quadratic cost function. This problem can be solved using MATLAB with the *quadprog* command from the Optimization Toolbox.

The objective now is to reshape the cost function in (3.12) to align with the cost function in the OCP in (3.13). This entails multiplying the first term on the right-hand side by 2, merging the second and third terms because they are identical, and disregarding the last term since it is a constant and does not influence the

optimization. Consequently, the cost function can be reformulated as

$$J_k = U_k^\top \underbrace{2(S^\top \tilde{Q} S + \tilde{R})}_{H_s} U_k + \underbrace{2x_k^\top H^\top \tilde{Q}}_{g^\top} S U_k. \tag{3.14}$$

Now, to impose constraints, the matrix A and the vector b need to be constructed. Let's consider a system with both input and output constraints. Suppose the input constraint is represented by the following inequality

$$\begin{aligned} U_k &\preceq \quad \bar{U} \\ -U_k &\preceq -\underline{U}. \end{aligned} \tag{3.15}$$

Here, $\bar{U} = 1_N \otimes \bar{u}$ represents the Kronecker product between 1_N and $\bar{u}$, where 1_N denotes a vector of ones $[1, 1, \cdots, 1]^\top$ with a length of N, and $\bar{u}$ signifies the upper limit for the control input. Similarly, $\underline{U} = 1_N \otimes \underline{u}$ can be defined, where $\underline{u}$ denotes the lower limit for the control input. In a similar manner the output constraints will have the form

$$\begin{aligned} Y_k &\preceq \quad \bar{Y} \\ -Y_k &\preceq -\underline{Y}. \end{aligned} \tag{3.16}$$

Similarly, we define $\bar{Y} = 1_N \otimes \bar{y}$, where $\bar{y}$ represents the upper limit for the output, and $\underline{Y} = 1_N \otimes \underline{y}$, where $\underline{y}$ denotes the lower limit for the output. The constraints currently do not directly depend on the control input, which is the focal point of solving the OCP. Therefore, it is necessary to alter these constraints to incorporate a dependency on U_k. This can be accomplished by substituting them with the prediction equation (3.11) as outlined below

$$\begin{aligned} \tilde{C} S U_k &\preceq \quad \bar{Y} - \tilde{C} H x_k \\ -\tilde{C} S U_k &\preceq -\underline{Y} + \tilde{C} H x_k. \end{aligned} \tag{3.17}$$

The input and output constraints can be merged to create the A matrix and b vector in (3.13) as follows

$$A = \begin{bmatrix} I_{Nm} \\ -I_{Nm} \\ \tilde{C} S \\ -\tilde{C} S \end{bmatrix}, \quad b = \begin{bmatrix} \bar{U} \\ -\underline{U} \\ \bar{Y} - \tilde{C} H x_k \\ -\underline{Y} + \tilde{C} H x_k \end{bmatrix}. \tag{3.18}$$

Now, with the cost function specified in (3.14) and both input and output constraints in (3.18), we are ready to solve a QP. However, two additional steps are required to achieve our objective of efficient path-following. The first step involves incorporating the dependency of (3.14) on the scheduled variables vectors to align with the qLPV modeling. As the matrices H and S are no longer constant, they need to be constructed at each time instant. The second step entails enhancing

the robustness of the proposed framework by employing offset-free MPC, which can be readily derived using velocity-based linearization, as will be illustrated.

Before explaining these steps, we can make a slight modification to the cost function to better match our problem. This modification involves constructing the cost function for reference tracking, which entails punishing the tracking error. This requires knowledge of the future reference values, which can be obtained using the path-generators discussed in Chapter 2. This modification simplifies the control problem and make it more desirable for controller designers, as the goal is to minimize the tracking error. The modified cost function for this context can be expressed as follows

$$J_k = \sum_{i=1}^{N} (r_{k+i} - y_{k+i})^\top T (r_{k+i} - y_{k+i}) + u_{k+i-1}^\top R u_{k+i-1}. \tag{3.19}$$

Here, $T \in \mathbb{R}^{l \times l}$ represents a symmetric positive definite matrix. This matrix penalizes the tracking error $e_{k+i} = r_{k+i} - y_{k+i}$, while $R \in \mathbb{R}^{m \times m}$ penalizes the control effort. Similar to (3.9), we can express the cost function in (3.19) more concisely as follows

$$J_k = (\mathbf{R}_k - Y_k)^\top \tilde{T} (\mathbf{R}_k - Y_k) + U_k^\top \tilde{R} U_k. \tag{3.20}$$

Here, $\mathbf{R}_k = [r_{k+1}^\top \; r_{k+2}^\top \; \cdots \; r_{k+2}^\top]^\top$ represents the future values of the reference. Please note the distinction between $\mathbf{R}$, which denotes the future reference values and is bold, and R, which denotes the control effort punishing matrix. Now, we can express the modified cost function to match a QP as follows

$$J_k = U_k^\top \underbrace{2(S^\top \tilde{C}^\top \tilde{T} \tilde{C} S + \tilde{R})}_{H_s} U_k + \underbrace{2(x_k^\top H^\top \tilde{C}^\top \tilde{T} \tilde{C} S - \mathbf{R}_k^\top \tilde{T} \tilde{C} S)}_{g^\top} U_k. \tag{3.21}$$

As previously mentioned, the cost function in (3.21) cannot be directly applied to our problem since X_k depends not only on the future control inputs U_k but also on the future scheduling parameters, denoted as $P_k = [\rho_k \; \rho_{k+1} \; \cdots \; \rho_{k+N-1}]^\top$, where ρ represents the scheduling vector. While this would typically pose a challenge for general LPV systems due to the need for future scheduling variable measurements, it does not present an issue with qLPV, as qLPV depends on x_k and u_k, which can be predicted. Expanding on this, the matrices H and S should be updated to $H(P_k)$ and $S(P_k)$ respectively. This modification transforms the cost function in (3.21) into the following form

$$J_k = U_k^\top \underbrace{2(S(P_k)^\top \tilde{C}^\top \tilde{T} \tilde{C} S(P_k) + \tilde{R})}_{H_s} U_k + \underbrace{2(x_k^\top H(P_k)^\top \tilde{C}^\top \tilde{T} \tilde{C} S(P_k) - \mathbf{R}_k^\top \tilde{T} \tilde{C} S(P_k))}_{g^\top} U_k. \tag{3.22}$$

As planned, the final step is to implement offset-free MPC, which integrates an integral action to enhance the robustness of the path-following algorithm against

potential disturbances such as wind or uncertainties in the model. This is crucial for MPC, given its reliance on the model. By introducing integral action into the MPC formulation, the controller can continually adapt the control signal to eliminate any steady-state errors, thereby ensuring accurate tracking of the desired reference trajectory over time. Here, we illustrate a simple approach to achieve this. Let's consider the control input increment Δu, defined as

$$\Delta u_k = u_k - u_{k-1}. \tag{3.23}$$

Therefore, the vector U_k can be represented as follows

$$U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+N-1} \end{bmatrix} = \begin{bmatrix} \Delta u_k + u_{k-1} \\ \Delta u_k + \Delta u_{k+1} + u_{k-1} \\ \Delta u_k + \Delta u_{k+1} + \Delta u_{k+2} + u_{k-1} \\ \vdots \\ \Delta u_k + \Delta u_{k+1} + \cdots + \Delta u_{k+N-1} + u_{k-1} \end{bmatrix}. \tag{3.24}$$

This equation can be reformulated as follows

$$U_k = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}}_{C_\Delta} \underbrace{\begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \vdots \\ \Delta u_{k+N-1} \end{bmatrix}}_{\hat{U}_k} + \underbrace{\begin{bmatrix} u_{k-1} \\ u_{k-1} \\ u_{k-1} \\ \vdots \\ u_{k-1} \end{bmatrix}}_{1_N \otimes u_{k-1}} = C_\Delta \hat{U}_k + 1_N \otimes u_{k-1}. \tag{3.25}$$

The constraints in (3.15) should be reformulated to account for C_Δ, a lower triangular matrix, multiplied by the vector of future control input increments $\hat{U}_k$ as follows

$$\begin{aligned} C_\Delta \hat{U}_k &\preceq \bar{U} - 1_N \otimes u_{k-1} \\ -C_\Delta \hat{U}_k &\preceq -\underline{U} + 1_N \otimes u_{k-1}. \end{aligned} \tag{3.26}$$

Fortunately, we do not have to perform this task manually. The velocity-based model in (3.7) provides us directly with a linear model that is linear in states and control input derivatives, which can be directly used in the modeling. This is evident from the fact that the obtained system matrix has eigenvalues at zero, indicating that the obtained system has integral action. The cost function will now undergo one final modification to obtain $\hat{U}$, which represents the vector of future control input increments, instead of U_k, which signifies the vector of control inputs. This adjustment yields

$$J_k = \hat{U}_k^\top \underbrace{2(S(P_k)^\top \tilde{C}^\top \tilde{T} \tilde{C} S(P_k) + \tilde{R})}_{H_s} \hat{U}_k + \underbrace{2(x_k^\top H(P_k)^\top \tilde{C}^\top \tilde{T} \tilde{C} S(P_k) - \mathbf{R}_k^\top \tilde{T} \tilde{C} S(P_k))}_{g^\top} \hat{U}_k. \tag{3.27}$$

With the cost function in (3.27) and the constraints formulated as shown in (3.26), we are prepared to solve a QP to obtain $\hat{U}^*$, satisfying

$$\hat{U}_k^* = \min_{\hat{U}_k} J_k \qquad s.t. \quad U_k \in \mathcal{U}, \quad x \in \mathcal{X}. \tag{3.28}$$

This OCP addresses both system dynamics and state and input constraints. In Chapter 5, we enhance this OCP with two additional constraints to ensure algorithm stability and guarantee that the solution obtained by the end of the horizon in the final iteration is a steady-state. Now, we have reached at the last step of the implementation of the iterative qLMPC algorithm which is summarized in Algorithm 3.1.

Algorithm 3.1 qLMPC Algorithm [Cisneros and Werner, 2020]

Initialization: Model, $\tilde{Q}$, $\tilde{R}$, $\tilde{T}$, N

1: $k \leftarrow 0$
2: Define $P^0 = 1_N \otimes \rho(x(0), u(0))$
3: **repeat**
4: $\quad l \leftarrow 0$
5: $\quad$ **repeat**
6: $\quad\quad$ Solve QP (3.13) with P_k^l for ΔU_k^l
7: $\quad\quad$ Predict state $X_k^l = H(P_k^l)x_k + S(P_k^l)\Delta U_k^l$
8: $\quad\quad$ Define $P_k^{l+1} = \rho(X_k^l, U_k^l)$
9: $\quad\quad$ $l \leftarrow l + 1$
10: $\quad$ **until** stop criterion
11: $\quad$ Apply $u_k = u_{k-1} + \Delta u_k$
12: $\quad$ Define $P_{k+1}^0 = \rho(X_k^l, U_k^l)$
13: $\quad$ $k \leftarrow k + 1$
14: **until** end

This iterative algorithm is necessitated by the fact that the qLMPC problem can be solved given that the parameter trajectory depends on the state, and thus can be predicted based on previous states and inputs. However, the nonlinear nature of the problem may pose implementation challenges. Therefore, to address the complexity stemming from its nonlinear nature, an iterative algorithm is employed. This algorithm relies on solving a sequence of QPs to find the solution to the underlying nonlinear OCP. Initially, problem (3.13) is solved with the qLPV model in (3.7) replaced by the LTI model obtained when the vector P_k is replaced by $P_k^0 = 1 \otimes \rho(x(0), u(0))$. This yields P_k^1, which is then iteratively adjusted towards a potential suboptimal sequence $P_k^* = \rho(X_k^* U_k^*)$, where X_k^* and U_k^* represent the state and input trajectories corresponding to the obtained (sub)optimal solution. Subsequently, the new scheduling vector is utilized to solve (3.13) again, resulting in P_k^2, and this process repeats till the stop criterion is met. Following the last iteration, X_k^* and U_k^* are employed in the subsequent time step to initialize P_{k+1}^0, and so forth. This iterative approach involves solving a sequence of OPs where the qLPV model is substituted by an LTV model, which is then updated at each

iteration by updating the scheduling variables vector. This process continues iteratively until a stopping criterion is met, such as a tolerance error or a specified number of iterations.

3.3 ULTRA-Extra

In this section, we present the fixed-wing aircraft utilized in our simulations, known as the *ULTRA-Extra* [Krings et al., 2013]. This aircraft, depicted in Fig. 3.1, is an unmanned scale replica of the aerobatic aircraft Extra 330 ML, scaled at 1:2.5. Propelled by an electric motor, it boasts a maximum power output of 7.2 kW, facilitating flight test experiments of up to 20 minutes duration.

With a total mass of 24.6 kg and a wingspan measuring 3.1 m, the aircraft offers versatility in experimentation. Control can be exercised either by a safety pilot using a remote controller or via a flight control computer. For data distribution purposes, Ethernet and computer area network (CAN) interfaces are integrated, while data recording functionality is available on a separate computer. Moreover, the aircraft features a telemetry link enabling communication between the aircraft and the flight test engineer for flight experiment control and monitoring.

We also present the details of designing the attitude and stabilization control laws for this aircraft following [Sedlmair et al., 2019]. These laws are designed using a Successive Loop Closure (SLC) methodology [Beard, 2012]. This approach facilitates the development of simple single-input, single-output control laws. Moreover, it exploits the inherent structural properties of fixed-wing UAVs, taking advantage of the chain of integrators they exhibit.

Figure 3.1: ULTRA-Extra aircraft.

In general, an autopilot serves as a system designed to navigate an aircraft without requiring direct intervention from a pilot. In the realm of manned aircraft, autopilots can vary in complexity, ranging from basic single-axis wing leveling systems to sophisticated flight control systems capable of managing both position (north, east, altitude) and attitude (roll, pitch, yaw) across different flight phases

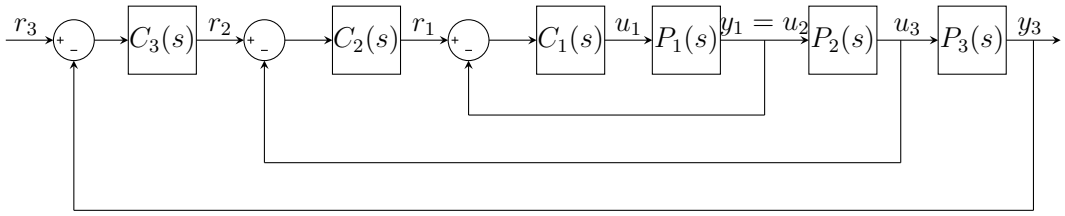

Figure 3.2: Successive-loop-closure design.

(take-off, landing, level-flight, approach, ascent, descent). However, for UAVs, the autopilot must assume full control over the aircraft throughout all stages of flight. While certain control functions may still be managed via a ground control station (GCS), the primary responsibilities of the autopilot are executed onboard the UAV itself. Returning to the SLC methodology for autopilot design, we will now provide a concise overview of this approach before detailing its application to the UAV.

3.3.1 Successive Loop Closure

As previously stated, the primary objective of autopilot design is to effectively control the inertial position (p_n, p_e, h) and attitude (ϕ, θ, χ) of the UAV. In the case of the ULTRA-Extra, given its aerobatic nature, it is possible to decouple the longitudinal dynamics (forward speed, climbing/descending motions, and pitching) from the lateral dynamics (yawing and rolling motions).

This division offers a significant advantage by simplifying the autopilot development process. In the SLC approach, the fundamental concept revolves around closing multiple straightforward feedback loops sequentially around the open-loop plant dynamics. The schematic representation of the SLC methodology is illustrated in Fig. 3.2.

The closed-loop dynamics consist of three consecutive loops, each typically of relatively low order. Each loop produces an output that is available for measurement and used for feedback. In this configuration, the objective is to regulate the outermost output y_3. Instead of closing a single feedback loop with y_3, the approach involves closing three feedback loops sequentially around y_1, y_2, and y_3. To achieve this, three controllers $C_1(s)$, $C_2(s)$, and $C_3(s)$ are designed sequentially. However, a crucial condition in this design process is that the inner loop should exhibit the highest bandwidth, with each successive loop having a bandwidth 5-10 times smaller in frequency.

When examining the inner loop, the objective is to devise a closed-loop system from r_1 to y_1 with a designated bandwidth ω_{BW1}. Leveraging the aforementioned assumption, this closed-loop system can be approximated as a unity gain. Subsequently, designing the second loop becomes more straightforward, and this process continues until reaching the outermost loop, which is the target loop for control. Given that our systems typically exhibit first or second-order dynamics,

conventional PID or lead-lag compensators can be employed. Additionally, transfer function-based design methods such as loop shaping [Horowitz, 1963] or root locus approaches [Walter R. Evans, 1950] can be employed effectively.

3.3.2 ULTRA-Extra Attitude and Stabilization Controller Design

Here, we illustrate the application of the SLC methodology for designing the low-level controller tailored for the ULTRA-Extra. It is crucial to emphasize that to achieve precise control of the aircraft along its flight path, we must linearize its dynamics across various airspeeds, encompassing the potential speed range of the aircraft during flight. Subsequently, we will implement the SLC methodology for each linearized model, addressing both longitudinal and lateral dynamics. Additionally, we will employ gain-scheduling to seamlessly transition between controllers in response to changes in the aircraft's velocity. For our specific aircraft, we opt to linearize its dynamics at seven distinct operating points, corresponding to airspeeds ranging from 20 to 50 m/s.

Since the ULTRA-Extra is an aerobatic aircraft, it allows for the decoupling of its longitudinal and lateral dynamics. This enables the application of the SLC methodology within a cascaded structure comprising two separate control loops: one designated for longitudinal motion and the other for lateral motion. Each loop operates within distinct frequency ranges, ensuring effective control over both longitudinal and lateral dynamics while maintaining independence between the two.

To design appropriate control laws for a fixed-wing UAV, it is crucial to determine the objectives of these control mechanisms. In this study, the aim is to formulate control laws primarily focused on managing the aircraft's attitude, ensuring stabilization, and effectively implementing commands derived from the higher-level path-following controller.

We need to select suitable control signals for each type of motion. For instance, in longitudinal motion, we might choose to control the pitch angle (θ), which would necessitate the innermost loop to manage the pitch rate (q). This methodology has been explored in [Beard, 2012]. Alternatively, we can opt to control the climb speed (w), and in this case controlling the normal load factor (n_{zb}) within the inner loop can be accomplished by leveraging the inherent physical integrators present in the aircraft structure [Sedlmair et al., 2019].

Applying similar procedures to address lateral motion, two channels are set up within a cascaded scheme, around the aircraft's longitudinal axis. For roll motion control, mirroring the approach used in longitudinal motion, an integrator chain facilitates lateral aircraft motion control. The innermost control loop, operating with the fastest dynamics, controlling the roll rate (p). The output signal from the roll rate controller determines the aileron deflection (ξ). To achieve this, the

measured roll rate of the aircraft from the navigation platform is fed back into the control mechanism.

Longitudinal Controller Design

In the following, we outline the development process of the longitudinal controller. Before turning to the controller structure, we provide insights into the modeling details. The innermost longitudinal controller uses the elevator, δ_e, to follow the

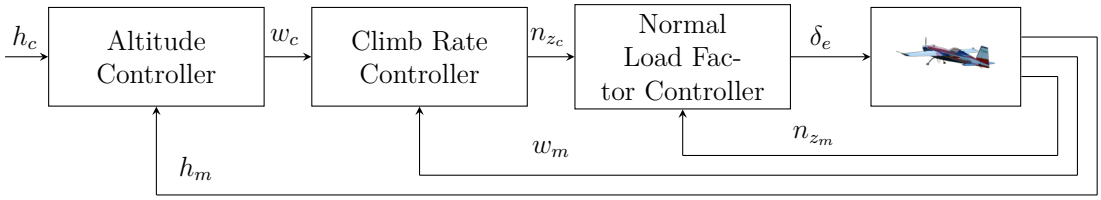

Figure 3.3: Longitudinal controller: altitude channel.

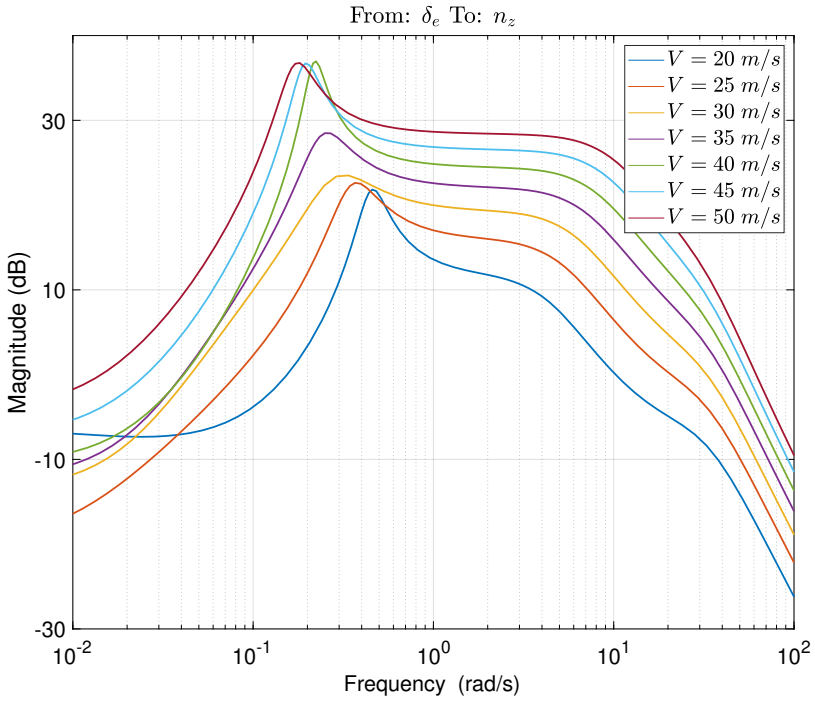

Figure 3.4: Bode plots of longitudinal dynamics at different airspeeds.

normal acceleration signal (n_z). By tracking the normal acceleration, an integrator chain for vertical speed (w) can be established, simplifying the design of the outer control loop, which can also be extended for altitude hold.

Figure 3.4 displays the Bode plot representing the linearized longitudinal dynamics for different airspeeds, mapping the elevator signal to the normal acceleration signal. The linearization process incorporates seven different operating points, covering airspeeds within the range of $20-50$ m/s. The plot notably demonstrates

substantial variations in the longitudinal dynamics corresponding to changes in airspeed, providing a rationale for implementing gain-scheduling.

This Bode plot reveals both phugoid and short-period dynamics. Observations indicate that the short-period frequency tends to rise proportionally with increasing airspeed. Additionally, the transfer function's magnitude exhibits quadratic growth relative to airspeed changes. Within the considered airspeed range, analysis indicates that the ULTRA-Extra showcases a well-damped short-period mode, boasting a damping ratio ($\zeta = 0.7$). Consequently, in this case, a pitch damper becomes unnecessary. The aircraft's servo dynamics operate within a range of ($\omega_{act} = 34\ rad/s$).

To design a controller for the longitudinal dynamics, we focus on the short-period dynamics. By reducing the model to the short-period dynamics, loop-shaping technique can be used to craft a proportional-integral controller to the normal acceleration loop (C_{nz}). The objective is to optimize the bandwidth of the inner loop while maintaining a phase margin constraint ($PM_{nz} \succ 60°$). This approach involves increasing the integral gain approximately and diminishing the proportional gain roughly quadratically with airspeed. In (3.29), we find the resulting model after linearization at $V_a = 20\ m/s$ as an illustration.

$$\begin{bmatrix} \dot{u} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \\ \dot{\eta} \\ \ddot{\eta} \end{bmatrix} = \begin{bmatrix} -0.09 & 11.54 & -2.87 & -9.73 & 0.29 & -0.00014 \\ -0.03 & -3.05 & 0.99 & -0.07 & -0.09 & 0 \\ 0.05 & -7.32 & -2.99 & 0 & -5.38 & 0.003 \\ 0 & 0.004 & 1.001 & 0 & 0.003 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1161 & -43.48 \end{bmatrix} \begin{bmatrix} u \\ \alpha \\ q \\ \theta \\ \eta \\ \dot{\eta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -0.001 \\ 0 \\ 0 \\ 2363 \end{bmatrix} \delta_e. \tag{3.29}$$

For further insights, please refer to Table 3.1 for a detailed analysis of the longitudinal modes based on this linearized model. The phugoid mode, governed by

Pole	Damping	Frequency(rad/s)	Time Constant(s)
-0.0614+0.108i	0.494	0.124	16.3
-0.0614-0.108i	0.494	0.124	16.3
-7.44+3.9i	0.886	8.4	0.134
-7.44-3.9i	0.886	8.4	0.134
-21.7+26.2i	0.638	34.1	0.046
-21.7-26.2i	0.638	34.1	0.046

Table 3.1: Matlab *damp* output for ULTRA-Extra.

a pole pair near the origin at $\lambda_p = -0.0614 - 0.108i$, mainly involves forward speed and pitch angle, depicting slow longitudinal aircraft motion. Speed increase induces a pitch-up moment ($M_u \succ 0$), decreasing pitch angle. When pitch angle returns to zero, speed surpasses trim speed, causing the aircraft to pitch up and climb, reducing speed. As airspeed returns to trim value, pitch angle becomes

excessive, leading to continued climb and speed loss, resulting in a pitch-down. This oscillation gradually decays with each cycle, characterized by low frequency ($\omega_n = 0.124$ rad/s) and low damping ratio ($\zeta = 0.494$). The oscillation settles after approximately 16 s.

The second mode known as short-period mode corresponds to the eigenvalues, $\lambda_p = -7.44 + 3.9i$. The principal influence on this mode is the pitch rate, which precedes the angle of attack. Named for its relatively high frequency, this mode reflects brief intervals where the aircraft's longitudinal motion is primarily defined by this dynamic.

By closing the n_z control loop, the selection of the vertical speed controller C_w involves choosing a proportional gain. Maximizing the control loop bandwidth while ensuring a phase margin $PM_w \succ 50°$ generates a gain-schedule that linearly increases with airspeed. The controller output, $n_{z,ref}$, is limited based on the current airspeed to prevent stalling. This limit is adapted online from the model-based relation

$$n_{z,max} = \frac{\rho V_a^2 S}{2G} C_{L\alpha} \alpha_{max}.$$

Using the current airspeed of the aircraft denoted as V_a, air density ρ, the specified wing area S, weight G, and the lift effectiveness concerning angle of attack $C_{L\alpha}$, we adhered to the approach outlined in [Sedlmair et al., 2019], adopting $\alpha_{max} = 12°$.

The second channel within the longitudinal control loop pertains to regulating the aircraft's airspeed. We have devised a proportional-integral controller aimed at attaining a closed-loop bandwidth of 3.1 rad/s while maintaining a phase margin of 75°. Refer to Fig. 3.5 for an illustration of the airspeed control loop. The closed loop response obtained using the designed longitudinal controller for the entire envelope of the airspeed can be shown in Fig. 3.6.

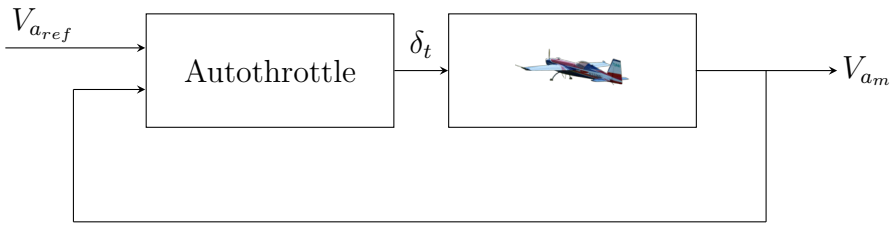

Figure 3.5: Longitudinal controller: autothrottle channel.

Lateral Controller Design

Employing the identical methodology as utilized in the design of the longitudinal controller using the SLC approach, we replicate the process to design the lateral controller. The lateral dynamics, encompassing roll rate (p), yaw rate (r), lateral velocity (v), and sideslip angle (β), are included within the linearized model. This ensures an accurate depiction of the lateral dynamics, enabling effective control of

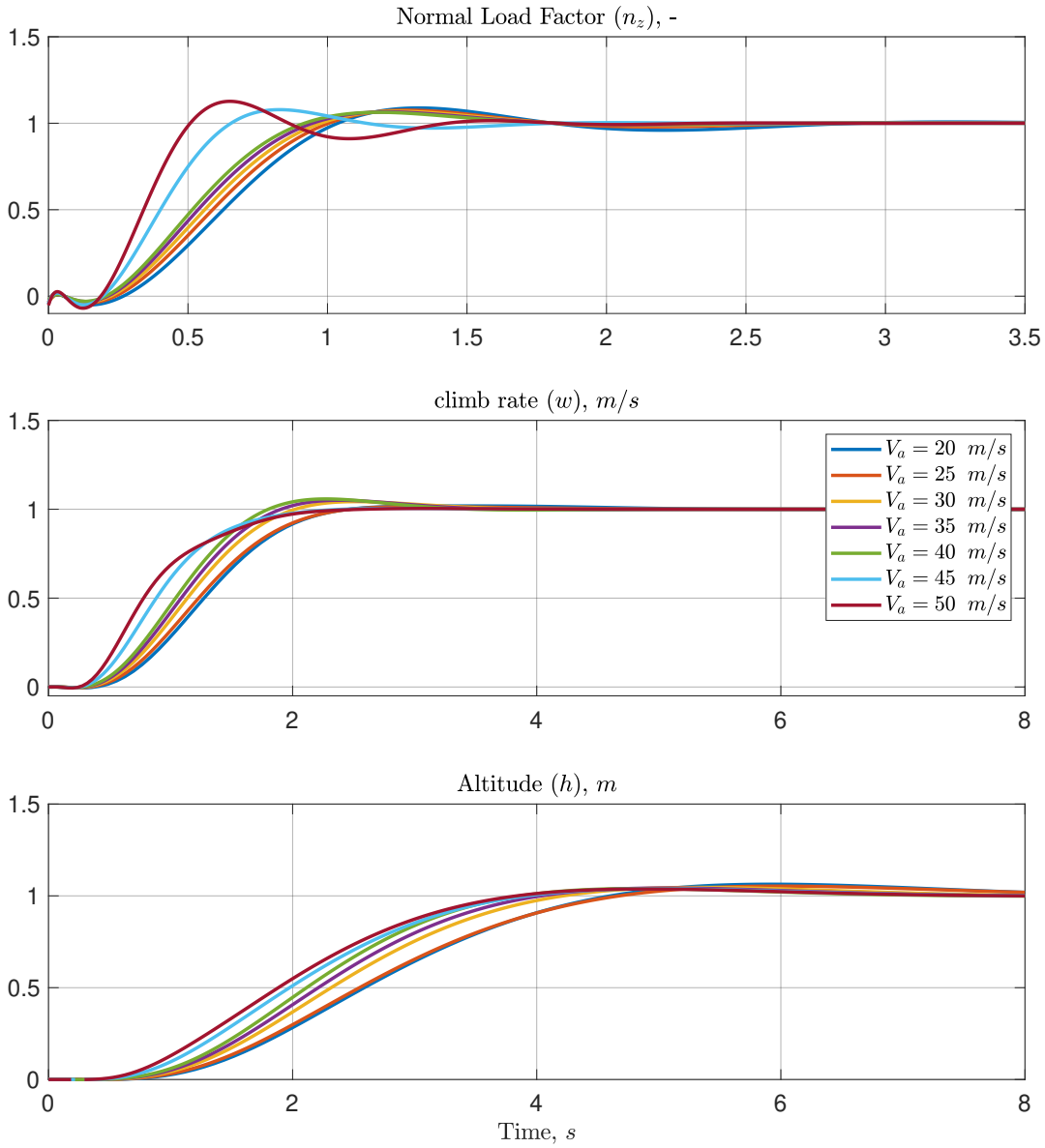

Figure 3.6: Closed-loop response for the longitudinal controllers.

the bank angle (ϕ) of the aircraft through aileron deflection (δ_a). Utilizing loop shaping techniques, we fine-tune the lateral controller to achieve desired performance outcomes.

Before embarking on the design of the lateral controller, it is imperative to analyze the lateral dynamics, as depicted in the Bode plot in Figure 3.7. This plot reveals minimal roll-yaw coupling. Moreover, notable changes in lateral dynamics are observed with variations in airspeed. Following the integrator chain strategy akin

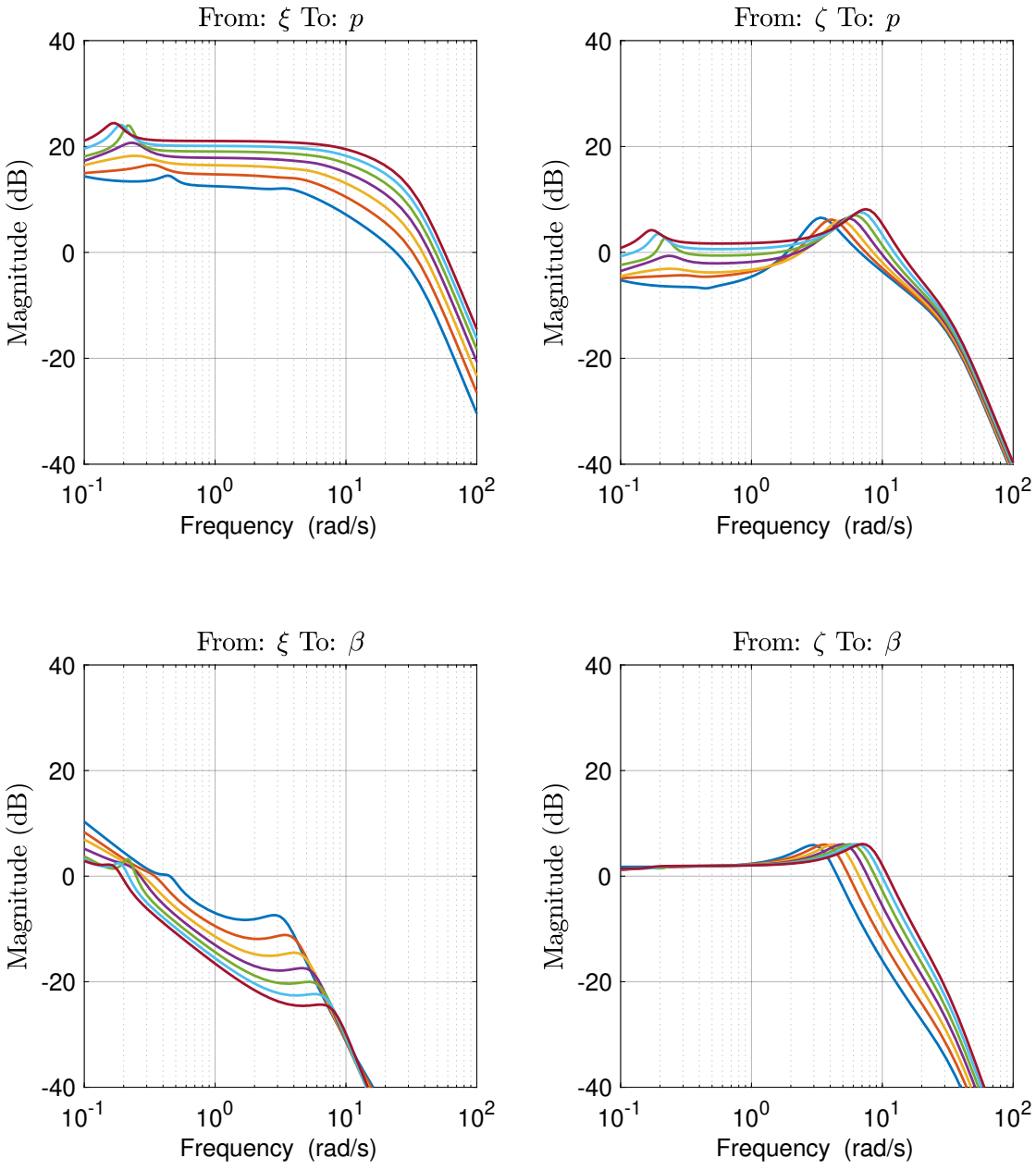

Figure 3.7: Bode plots of lateral dynamics.

to the longitudinal dynamics, we formulate an inner loop for roll rate (p) tracking. Proportional-integral control is applied in the inner loop, while a proportional outer loop incorporates feedback for the bank angle (ϕ). This configuration differs

slightly from the typical proportional-integral bank angle control loop, featuring proportional inner loop roll rate feedback.

The roll rate controller is designed to optimize bandwidth while adhering to a phase margin constraint of 60°. The gain for the bank angle controller is determined to meet a 60° phase margin constraint, leading to a slight increase in gain with airspeed. Recognizing that the body-fixed roll rate utilized in the inner feedback loop is not precisely the derivative of the bank angle ϕ, a minor steady-state error is deemed acceptable.

For the lateral dynamics, two channels are necessary, as illustrated in Fig. 3.8 and Fig. 3.9. The first channel is dedicated to controlling the bank angle (ϕ) or the course angle (χ), while the second channel ensures clean flight and coordinated turns by maintaining a side-slip angle ($\beta = 0$). Initially, yaw damping is augmented by incorporating feedback from the washed-out yaw rate (r) to enhance Dutch roll damping to $\zeta_{DR} \succ 0.4$. Subsequently, a proportional-integral controller is devised to track the sideslip angle (β), ensuring aerodynamically clean flight, particularly in crosswind conditions. Furthermore, this controller facilitates the execution of decrab maneuvers and specific aerobatic flight maneuvers. The design of this controller emphasizes maximizing the closed-loop bandwidth while adhering to a 60° phase margin constraint. To prevent a loss of altitude due to the banking of the lift vector, the maximum bank angle is constrained based on the current airspeed, as outlined in Table 3.2. Figure 3.10 illustrates the closed-loop response for the lateral controllers.

So far, we have successfully designed controllers for both the longitudinal and lateral dynamics of the ULTRA-Extra across its entire airspeed envelope, and we have verified the closed-loop responses of these individual controllers. The subsequent step involves testing both controllers simultaneously in a dynamic scenario with varying reference signals to assess the performance of the designed autopilot. To accomplish this, we initiated a random scenario, and the autopilot's performance is deemed satisfactory, as illustrated in Figure 3.11. This indicates that the autopilot is now prepared to execute commands received from the path-following controller (the high-level controller), as outlined in the framework depicted in Figure 1.2.

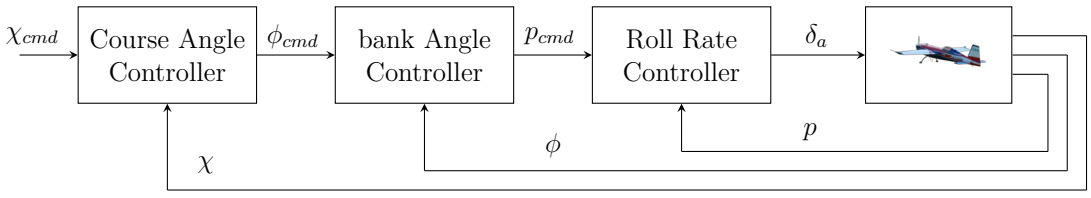

Figure 3.8: Lateral controller: course angle channel.

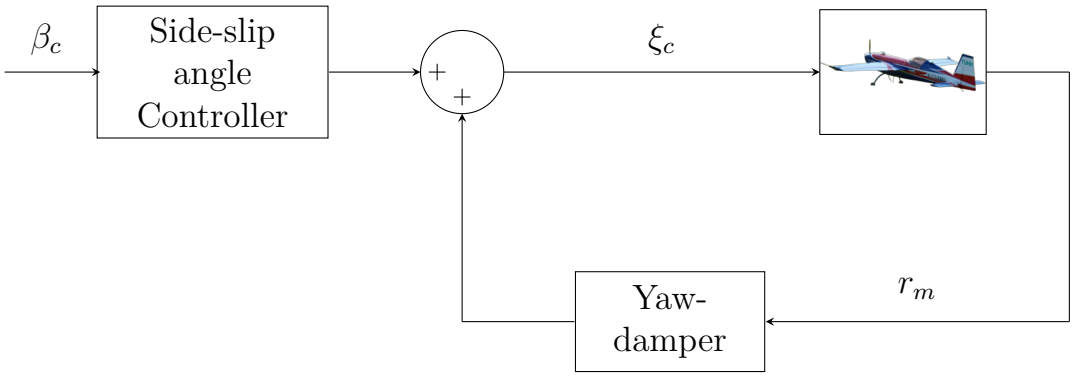

Figure 3.9: Lateral controller: side-slip angle channel.

Airspeed (V_a), m/s	20	25	30	35	40	45	50
ϕ_{max}, $^\circ$	30	45	45	50	50	50	50

Table 3.2: Maximum bank angle command ϕ_{cmd} according to the change in airspeed.

3.4 Tracking Scenarios-

In this section, we discuss the simulation results acquired through the implementation of the introduced methodology. In these simulations, we employ both the hybrid approach and the synthetic waypoint path-generators for the two scenarios introduced in Chapter 2. We assess our methodology from two perspectives. Firstly, we examine the tracking performance and conduct error analysis across the two scenarios. Secondly, we evaluate the computational efficiency of the algorithm.

3.4.1 Tracking - Hybrid Path Generator

In this section, we assess the tracking performance using the hybrid path-generator, evaluating the ability to adhere to the desired path and analyzing the tracking error. We also present a histogram illustrating the error distribution along the path. It is worth noting that in our approach, the aircraft airspeed is a controlled variable determined by the path-following controller. This differs from previous works, such as [Sedlmair et al., 2019] or [Sedlmair et al., 2020], where the aircraft speed was commanded independently of the path-following controller. However, those studies focused on employing a nonlinear guidance law (NLGL) and nonlinear differential geometric path-following guidance law (NDGPFG), respectively.

Furthermore, in the first scenario, we simulate a constant wind with a mean wind speed of $V_w = 4$ m/s and an orientation of $\chi_w = 89^\circ$. In the second scenario, we simulate time-varying wind with a wind speed V_w ranging from 1 m/s to 5 m/s and an orientation of $\chi_w = 335^\circ$. These wind parameters were specifically chosen to emulate real flight environments encountered in previous flight experiments with the same aircraft, as documented in [Sedlmair et al., 2019].

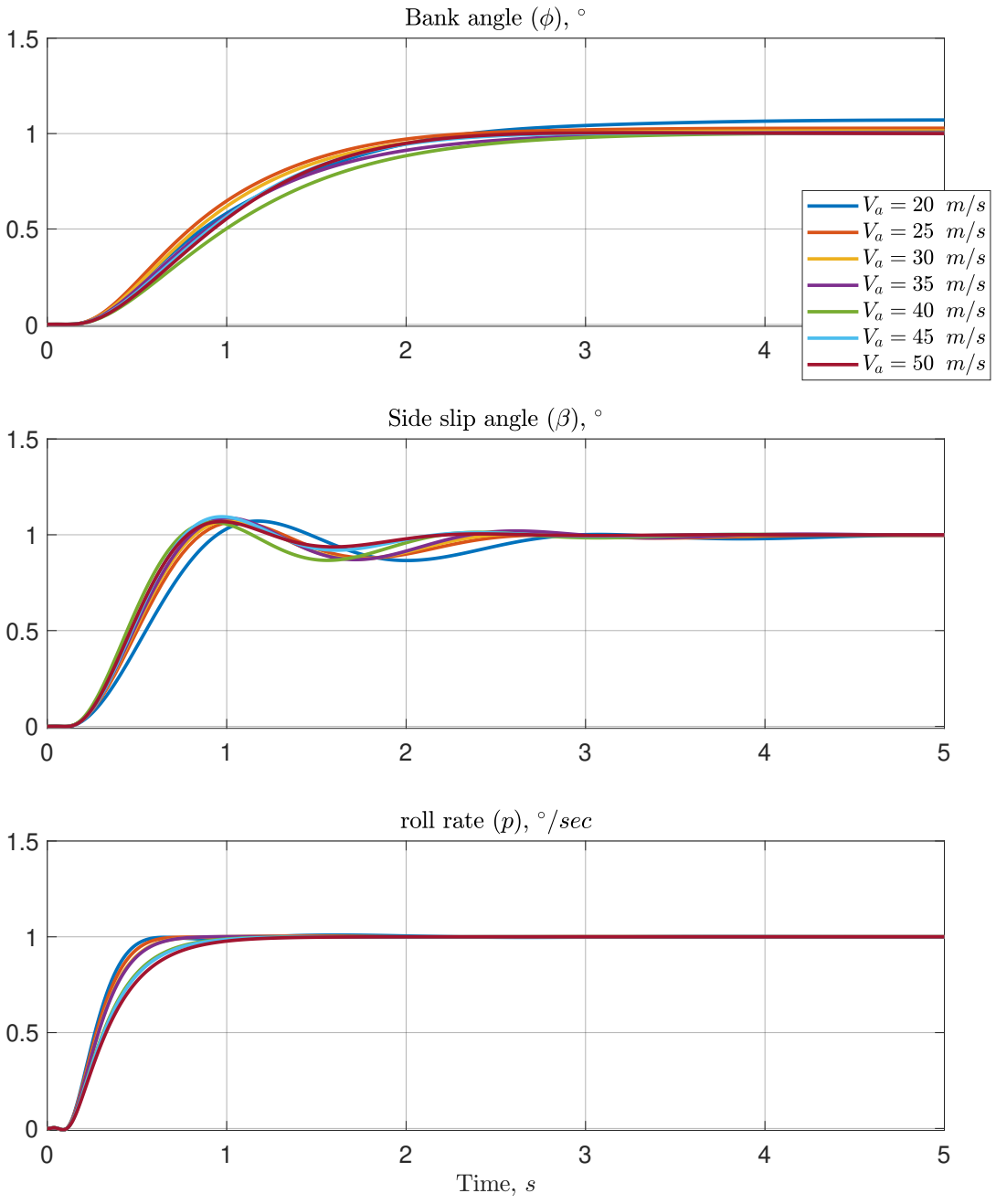

Figure 3.10: Closed-loop response for the lateral controllers.

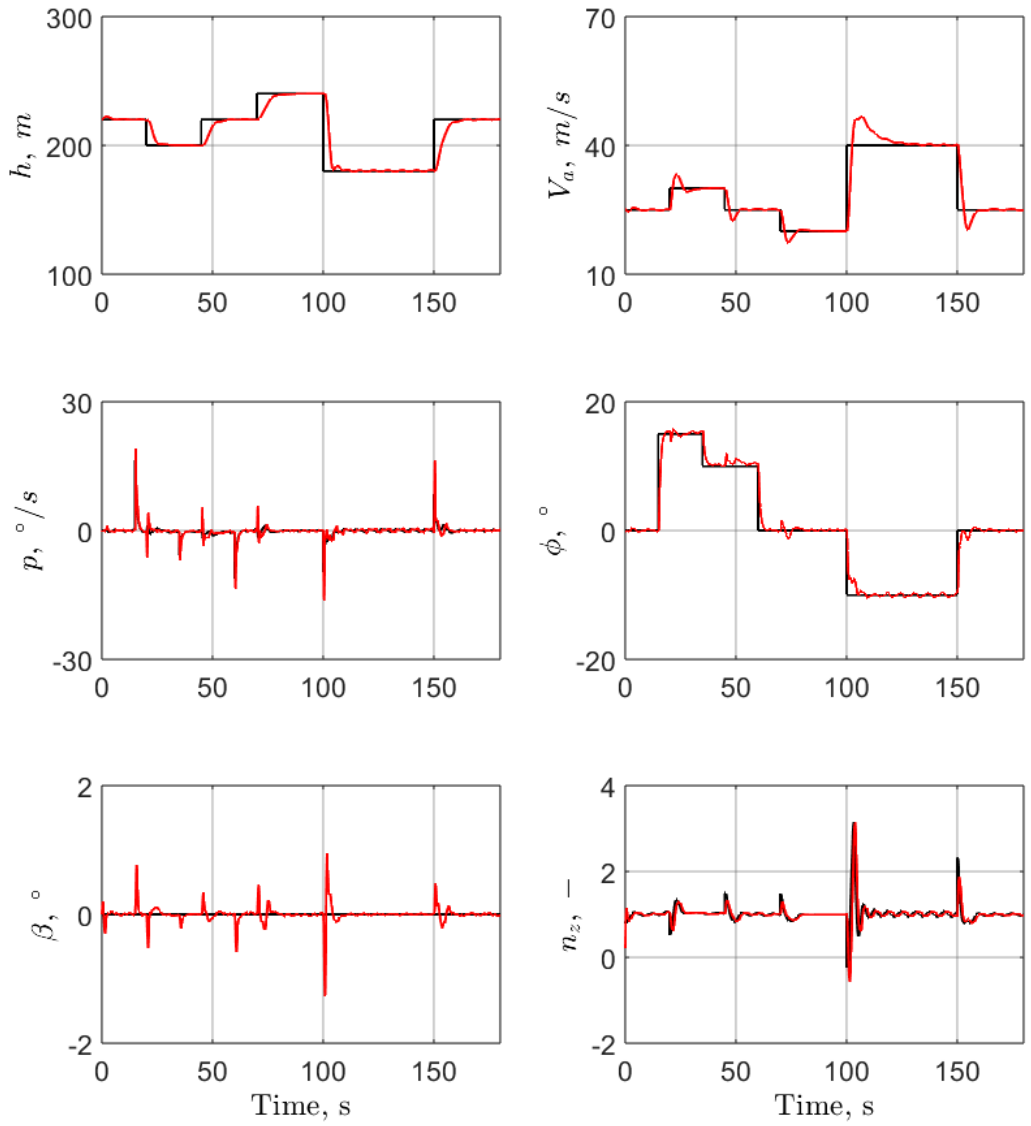

Figure 3.11: Stabilization and attitude controller evaluation: command (▬), measurement (▬).

We assume that the pilot guides the aircraft near the first waypoint before transitioning to path-following mode, where the control system will guide the aircraft along the designated path. The simulations were performed on a computer equipped with an 8-core processor operating at 3.6 GHz. We used the *quadprog* solver from the MATLAB optimization toolbox to solve the QP problem within the qLMPC algorithm.

Figure 3.12 and Figure 3.14 show satisfactory tracking performance, demonstrating the aircraft's ability to follow the desired path without deviating significantly. Additionally, Figure 3.13 depicts the histogram for tracking errors in the first scenario. It reveals that over 60% of the path-time, the tracking error remained below 1.5 m, with a maximum error of approximately 2.1 m. The average path-time (APT) for this scenario is approximately 56 s.

Similarly, in the second scenario, as shown in Figure 3.15, the tracking error is below 3 m for around 85% of the path-time, with a maximum error of approximately 4.5 m. Although slightly higher than the first scenario, this level of error is still acceptable for path-following using a fixed-wing aircraft, especially considering the increased complexity of the path. The APT for this scenario is around 60 s.

The error values achieved in the simulations are comparable to those reported in [Sedlmair et al., 2019] and [Sedlmair et al., 2020]. Although the scenarios yielded slightly lower errors, it is essential to note that the referenced errors were obtained from flight experiments rather than simulations.

As highlighted earlier, we initially implemented our framework on the kinematic model of the fixed-wing aircraft, then extended it to the full-state model of the ULTRA-Extra after validating the implementation. We provide a table comparing the errors obtained using the kinematic model and the full-state model. Additionally, the table includes the Average Path Time (APT), indicating the duration required for the aircraft to complete one full loop of the path. We used the following equation [Sedlmair et al., 2020] as a reference to evaluate the average error along the path

$$E = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} ||e||\ dt. \tag{3.30}$$

	'Aerodrome-circuit' Path		'Roller-coaster' Path	
	Full Model	Kinematic Model	Full Model	Kinematic Model
Min Error, m	0.112	0.02	0.45	0.04
Max Error, m	2.0643	1.61	4.41	1.93
Average Error, m	1.09	0.66	1.94	0.33
APT, s	56		60	

Table 3.3: Average error comparison - hybrid approach path-generator.

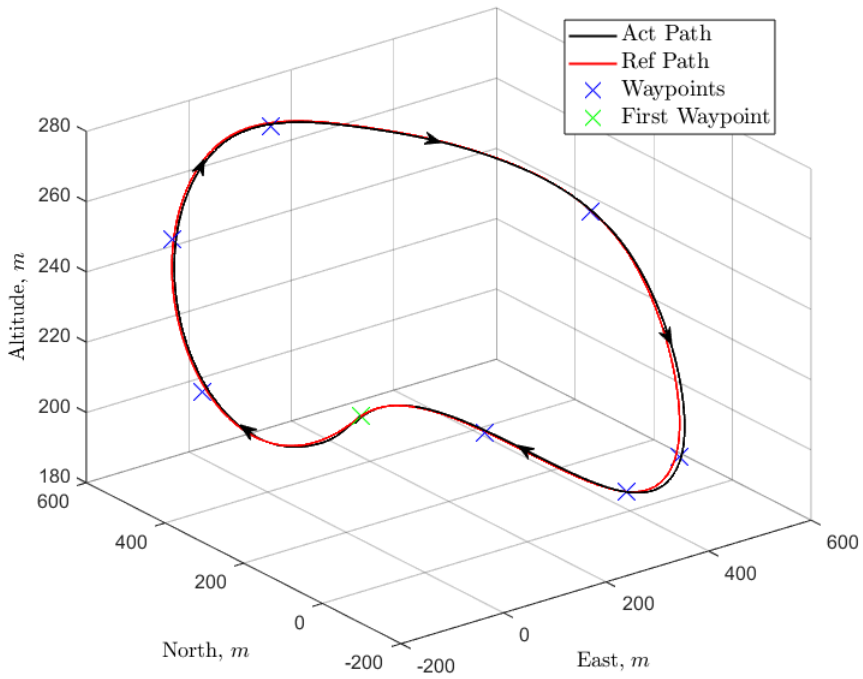

Figure 3.12: 'Aerodrome-circuit' - tracking - hybrid path-generator.

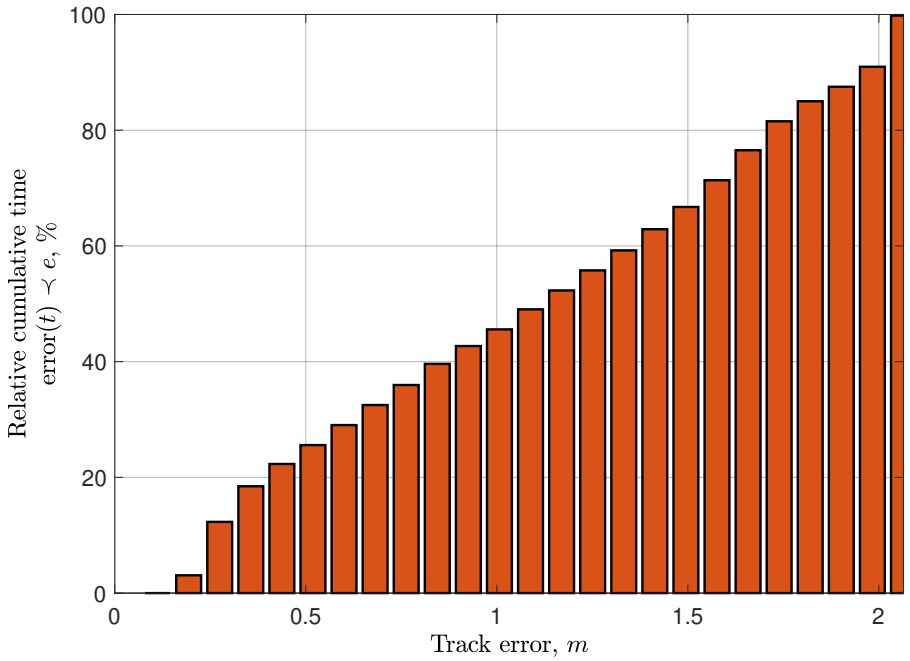

Figure 3.13: 'Aerodrome-circuit' - absolute track error - hybrid path-generator.

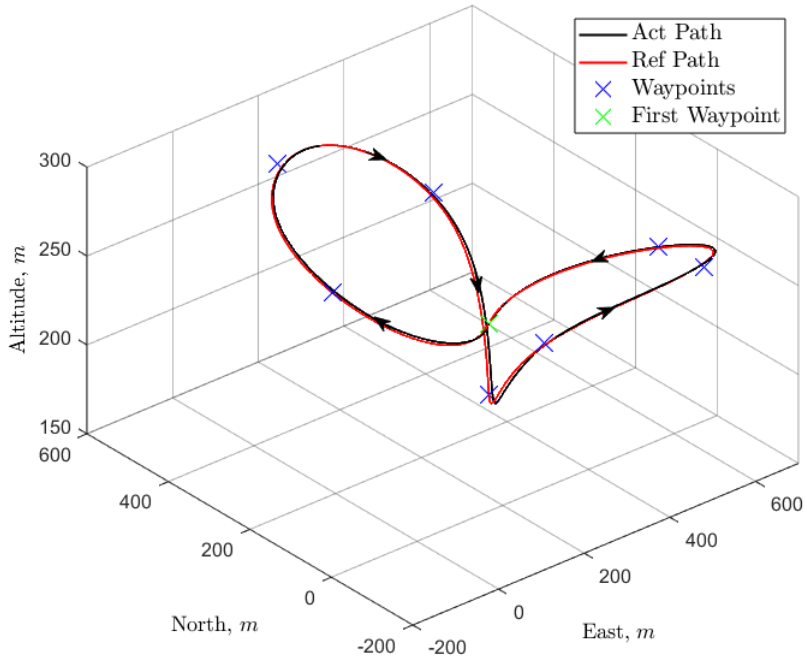

Figure 3.14: 'Roller-coaster' - tracking - hybrid path-generator.

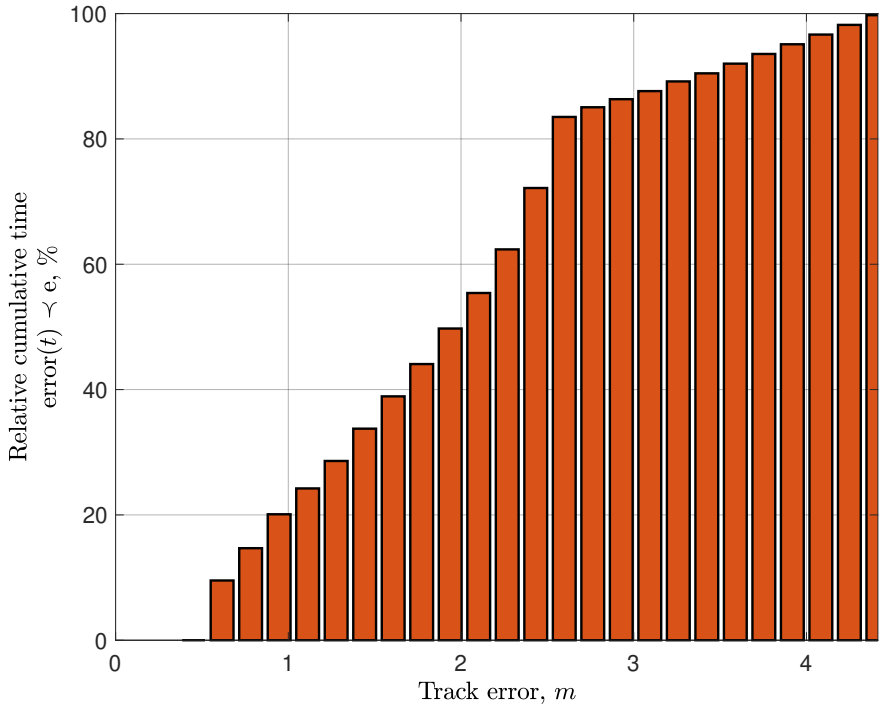

Figure 3.15: 'Roller-coaster' - absolute track error - hybrid path-generator.

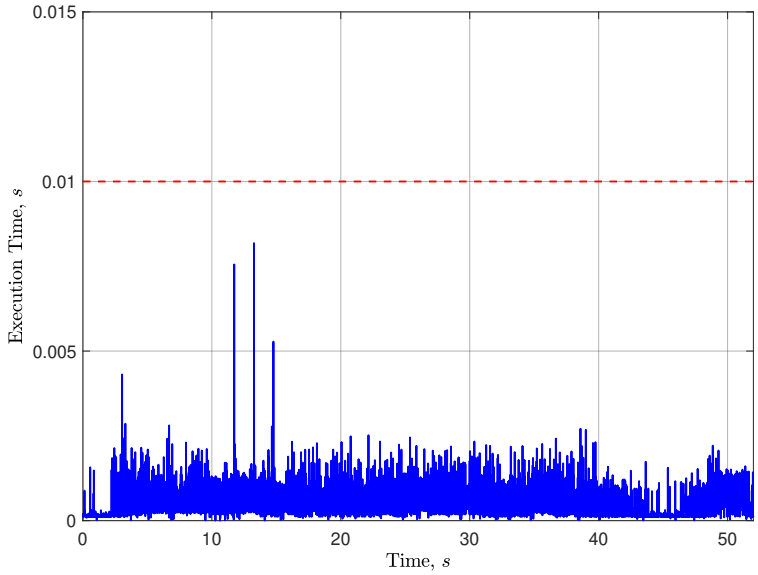

Figure 3.16: 'Aerodrome-circuit' - execution time - hybrid path-generator.

3.4.2 Computational Times - Hybrid Path Generator

In the following, we provide a concise overview of the execution times associated with the implementation of the qLMPC path-following algorithm using the hybrid path-generator. Our sampling time is set to 10 milliseconds, meaning that the execution time of the algorithm, including qLPV matrices construction and QP solving, must be well below 10 milliseconds. Otherwise, the algorithm will not be able to operate in real-time.

Figures 3.16 and 3.17 illustrate that the execution times of the algorithm in both scenarios consistently adhered to the sampling time limitations (indicated by the dashed red lines). Although there are occasional peaks, possibly due to maneuvers in the path or transitions between waypoint locations, these peaks remain within the boundaries of the sampling time.

3.4.3 Tracking - Synthetic Waypoint Path Generator

Previously, we presented the simulation results of the proposed algorithm using the hybrid path-generator, which combined arc length parameterization with cubic splines. In the following, we provide a similar evaluation, but this time employing a synthetic waypoint as the reference.

From Fig. 3.18 and Fig. 3.20, it is evident that the aircraft can still successfully follow the path defined by the synthetic waypoint, although the tracking errors appear slightly larger compared to those obtained using the hybrid path-generator. This observation is consistent with the histogram results depicted in Fig. 3.19 and Fig. 3.21, where the error in the first scenario reached approximately 3 m for nearly 20% of the flight path time and around 5 m for approximately 30% of the flight path time in the second scenario.

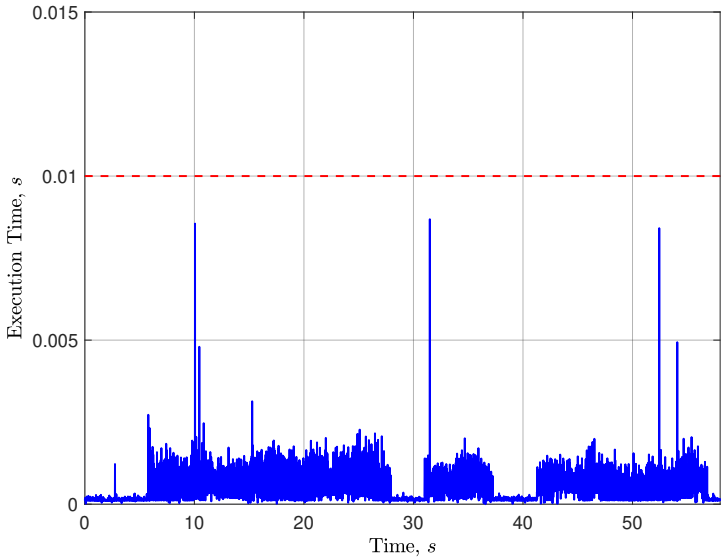

Figure 3.17: 'Roller-coaster' - execution time - hybrid path-generator.

Moreover, the maximum tracking error in the first scenario increased to 4.8 m from 2.1 m when using the hybrid approach and increased to 8.5 m from 4.5 in the hybrid approach for the second scenario. This increase in error can be attributed to the change in path-planners. While the hybrid approach provides a very smooth path owing to the use of cubic splines, the synthetic waypoint path-generator selects transition decision values between waypoints. If the distance between the aircraft and the following waypoint is less than this decision value, the waypoint is considered visited, and the next one is activated. In these scenarios, we selected a transition value of 10 m. Despite these relatively high tracking errors, all tracking errors remained within acceptable limits for a fixed-wing aircraft.

	'Aerodrome-circuit' Path		'Roller-coaster' Path	
	Full Model	Kinematic Model	Full Model	Kinematic Model
Min Error, m	0.15	0.10	1.36	1.24
Max Error, m	4.89	1.724	8.75	6.79
Average Error, m	3.15	0.81	4.98	1.65
APT, s	57		61	

Table 3.4: Average error comparison - synthetic waypoint path-generator.

3.4.4 Computational Times - Synthetic Waypoint Path Generator

Here we discuss the execution times associated with applying the qLMPC path-following with a synthetic waypoint path-generator. We find from Fig. 3.22 and Fig. 3.23 that the algorithm execution times are quite similar to those obtained using the hybrid path-generator. We still observe some peaks due to path maneuvers or waypoint transitions, but they remain within the sampling time of 0.01 sec.

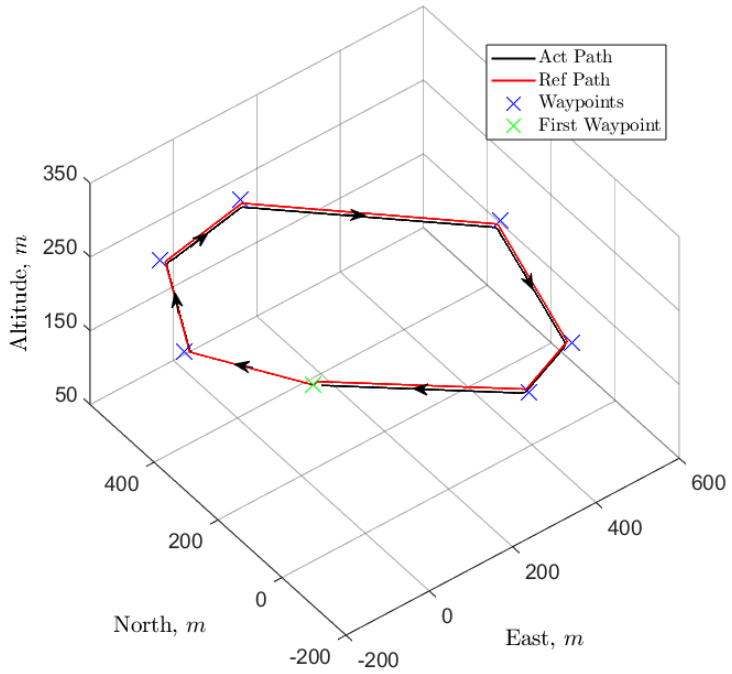

Figure 3.18: 'Aerodrome-circuit' - tracking - synthetic waypoint path-generator.

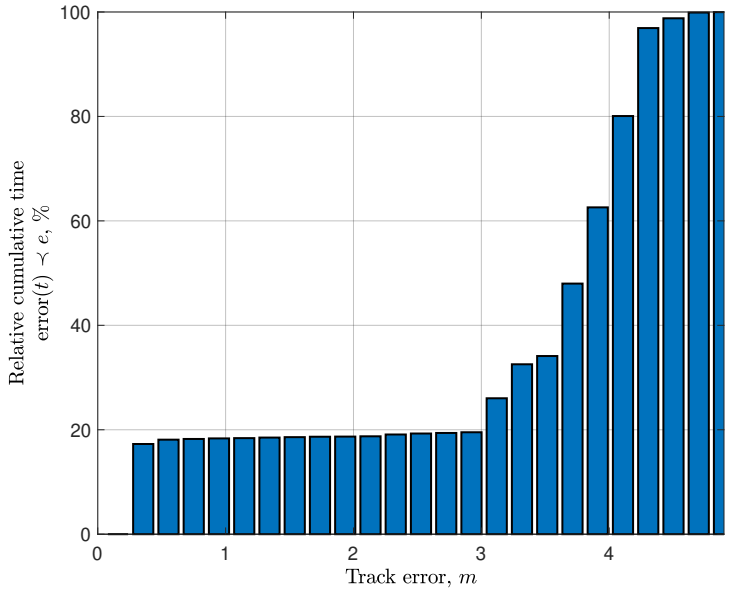

Figure 3.19: 'Aerodrome-circuit' - absolute track error - synthetic waypoint path-generator.

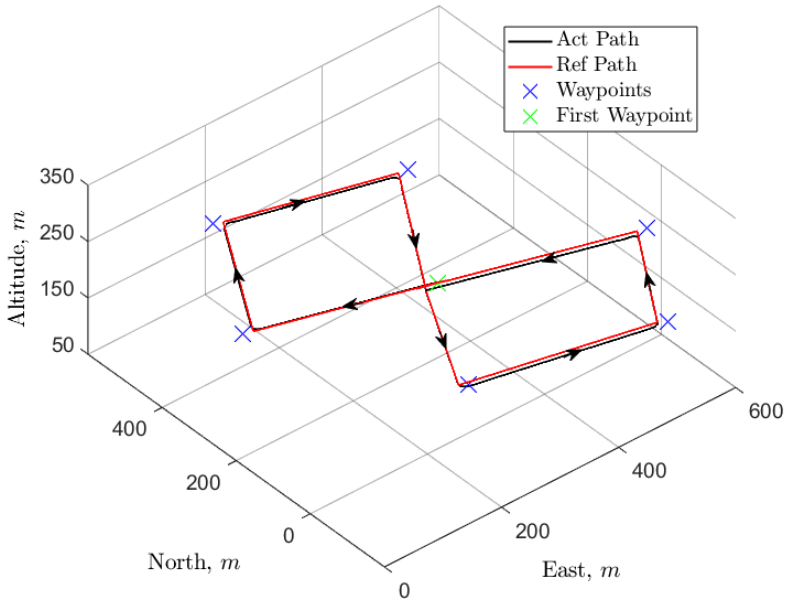

Figure 3.20: 'Roller-coaster' - tracking - synthetic waypoint path-generator.

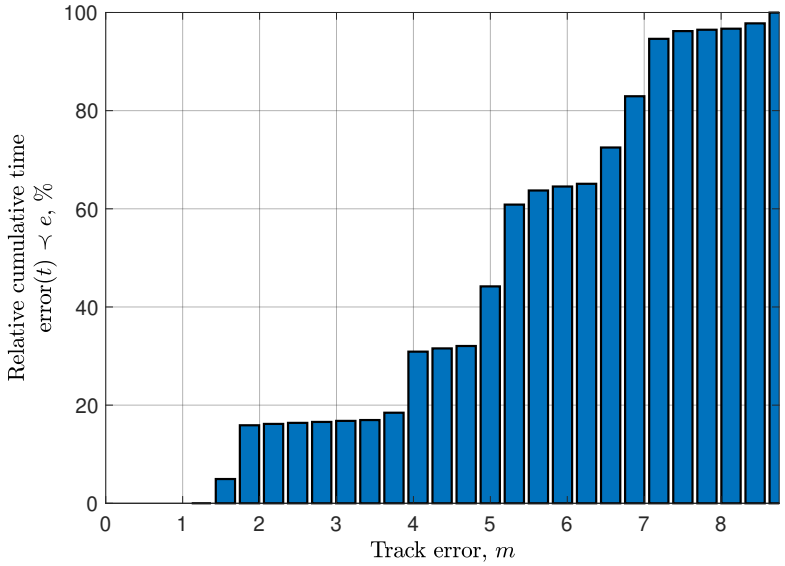

Figure 3.21: 'Roller-coaster' - absolute track error - synthetic waypoint path-generator.

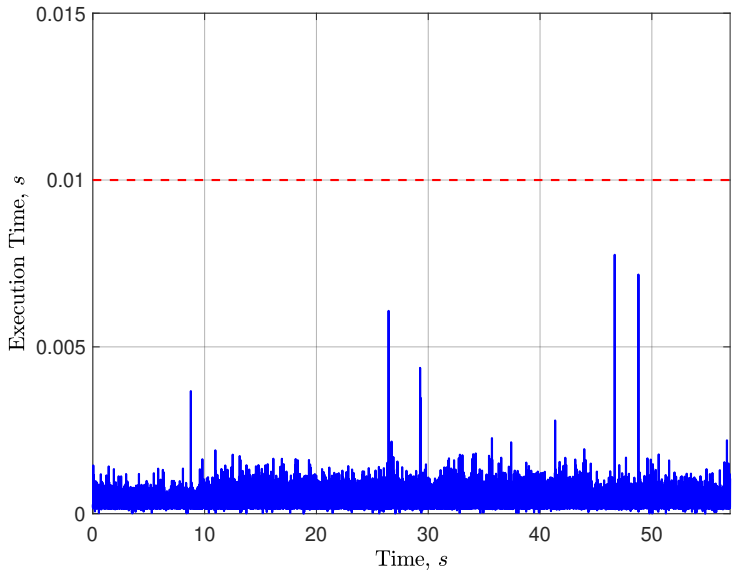

Figure 3.22: 'Aerodrome-circuit' - execution time - synthetic waypoint path-generator.

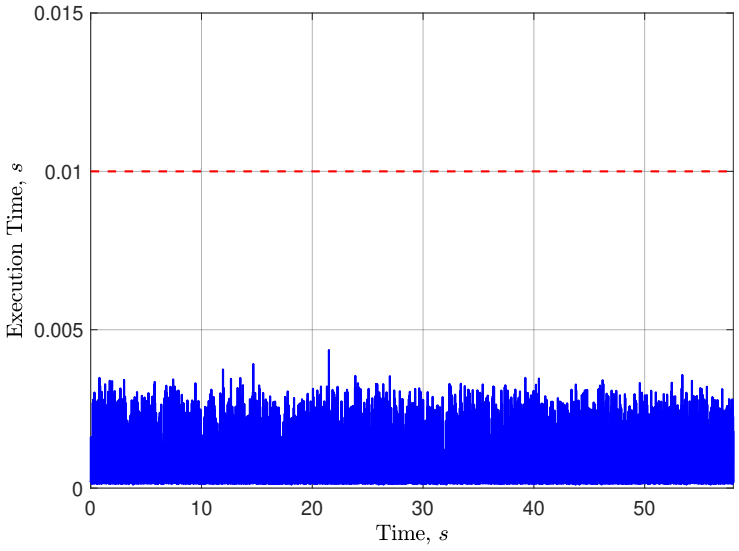

Figure 3.23: 'Roller-coaster' - execution time - synthetic waypoint path-generator.

3.5 Summary

In this chapter, we presented a predictive path-following approach that requires only to solve a QP in each sampling interval, based on a qLPV representation of the fixed-wing aircraft kinematics. This representation allows us to accurately capture the aircraft's position while maintaining computational efficiency in solving the OCP. Through simulations across two distinct scenarios, we demonstrate effective tracking performance within computational timeframes that do not exceed the sampling time of the high-level controller.

In the subsequent chapter, we will expand upon this framework by incorporating obstacle constraints to address obstacle avoidance within our methodology. Furthermore, we will explore strategies for handling the nonlinearity inherent in obstacle constraints, ensuring their compatibility with our framework, which excels in solving QPs with linear constraints.

Chapter 4

qLMPC Obstacle Avoidance

In this chapter, we extend the proposed framework from Chapter 3 to address the obstacle avoidance problem. Handling nonlinear obstacles presents complexities. One solution is solving a nonlinear problem with nonlinear constraints, as discussed in [Liu et al., 2017], but this can be computationally intensive. Another option is designing a separate predictive path-planner to handle obstacle constraints, as suggested in [Zhang et al., 2020], though this may increase computational demands. Our framework offers a compromise by processing constraints within the path-following problem, avoiding the need for a separate predictive path-planner.

The chapter commences with an introduction to the nature of the constraints, followed by the incorporation of these constraints into our framework. Finally, simulation results are presented, and commentary on these results is provided. The outcomes of this chapter have been documented in [Samir et al., 2024b].

4.1 Introduction

Most literature addressing the obstacle avoidance problem typically treats it as a distinct issue from the path-following problem. While this approach often yields good performance, it also tends to be computationally intensive, requiring separate controllers for path-following and obstacle avoidance. An intriguing alternative for addressing 3D obstacle avoidance is presented in [Lin et al., 2019], where the authors introduce the Fast Geometric Avoidance algorithm (FGA).

The primary objective of this algorithm is to achieve efficient obstacle avoidance with fast computational times. The research outcomes have been evaluated through Monte Carlo simulations and flight simulators, demonstrating the algorithm's efficiency. However, one limitation is its reliance on obstacle geometry to determine the necessary heading and pitch angles for avoidance. Similar to geometric path-following methods, this dependency compromises robustness against wind disturbances.

Another attempt to mitigate the robustness issue involves employing predictive control to tackle obstacle avoidance, as demonstrated in [Taherian et al., 2021] for a vehicle model. In this study, the authors focused on optimizing the predictive controller's tuning parameters. While this approach yields efficient performance with a degree of robustness attributed to its control-based methodology, it is based on a linear vehicle model.

Based on the preceding discussion, we can deduce the primary challenge in the obstacle avoidance problem: striking a balance between accurately representing the nonlinearities of obstacles and devising a computationally efficient solution with an acceptable degree of robustness.

4.2 qLMPC Obstacle Avoidance

In this section, we illustrate how we integrate the obstacle avoidance problem into our framework. We formulate our problem based on a common scenario encountered in path-following tasks: a UAV aims to adhere to a specific path but encounters an obstacle along the way, necessitating avoidance while continuing its intended trajectory.

As outlined in the introduction to this chapter, one potential approach involves replanning the path and subsequently feeding the updated reference commands to the path-following controller for execution. However, we find this method to be computationally demanding, as it entails solving two distinct problems.

Here, we propose an alternative solution: embedding the obstacle constraints directly into our framework. When the aircraft encounters an obstacle—a stringent constraint—the optimization control problem seeks the most viable solution without straying significantly from the planned path. Once the obstacle is circumvented, the algorithm facilitates the aircraft in reducing tracking error and returning to the intended path. Figure 4.1 depicts the obstacle avoidance scenario with spherical obstacles, where (x_a, y_a, z_a) denotes the aircraft's position, and (x_o, y_o, z_o) represents the center of the obstacle. We assume that the obstacle

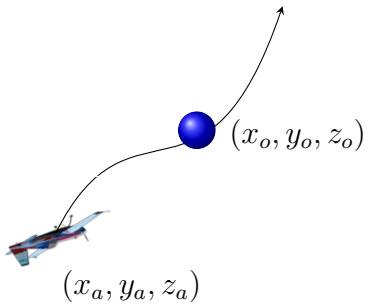

Figure 4.1: Obstacle avoidance problem.

location is constant.

Mathematical Formulation of the Obstacle Constraint

The objective is to navigate around the obstacle depicted in Fig. 4.1. This involves calculating the distance between the aircraft and the obstacle's center, denoted as d, and ensuring that it remains greater than the obstacle's radius, denoted by r (assuming spherical obstacles). The distance d can be computed in Euclidean space using the formula

$$d = \sqrt{(x_a - x_o)^2 + (y_a - y_o)^2 + (z_a - z_o)^2}. \tag{4.1}$$

This formula can be used to establish a constraint relating the distance d to the radius r, ensuring the aircraft maintains a safe distance from the obstacle. It is important to consider that the ULTRA-Extra's wingspan is approximately 3.1 m, and its autopilot covers airspeeds between 20-50 m/s. Hence, the choice of distance d should account for these limitations for the framework to be effective. We set the constraint as

$$d^2 \succ r^2. \tag{4.2}$$

By substituting (4.1) in (4.2), we get

$$(x_a - x_o)^2 + (y_a - y_o)^2 + (z_a - z_o)^2 \succ r^2. \tag{4.3}$$

With further expansion, we obtain

$$x_a^2 - 2x_o x_a + x_o^2 + y_a^2 - 2y_o y_a + y_o^2 + z_a^2 - 2z_o z_a + z_o^2 \succ r^2. \tag{4.4}$$

Multiplying both sides by -1 to reverse the inequality and bring it in line with the QP form, we obtain

$$\underbrace{-x_a^2 + 2x_o x_a - y_a^2 + 2y_o y_a - z_a^2 + 2z_o z_a}_{l(x_a, y_a, z_a)} \prec -r^2 + x_o^2 + y_o^2 + z_o^2. \tag{4.5}$$

Linearization Using Taylor Expansion

Note that the right-hand side in (4.5) is constant. The left-hand side is nonlinear and can be expressed in the form of $l(x_a, y_a, z_a)$ which can be linearized using the following Taylor series expansion

$$l(x_a, y_a, z_a) \approx l(\rho_k^a) + \nabla_x l(\rho_k^a)(x - x(\rho_k^a)). \tag{4.6}$$

The term $\nabla_x l(\rho_k^a)$ represents the gradient of $l(x_a, y_a, z_a)$ with respect to x. Upon substituting (4.6) into inequality (4.5) and rearranging the inequality terms, we obtain

$$\begin{aligned} &\begin{bmatrix} 2x_o - 2\rho_{1,k}^a & 2y_o - 2\rho_{2,k}^a & 2z_o - 2\rho_{3,k}^a & 0_{1\times 7} \end{bmatrix} x \\ &\prec -r^2 + x_o^2 + y_o^2 + z_o^2 - (\rho_{1,k}^a)^2 - (\rho_{2,k}^a)^2 - (\rho_{3,k}^a)^2, \end{aligned} \tag{4.7}$$

Note that this constraint requires periodic updates based on the changing aircraft position. Consequently, we refer to it as a qLPV constraint [Cisneros et al., 2018], where $\rho^a_{1,k} = x_a$ is the first value of the scheduling vector $\rho^a_{i,k} = [x_a, y_a, z_a]^\top$ at time k. It is important to note that this vector does not necessarily need to align with the scheduling vector for the qLPV representation of the system. Additionally, this constraint is only applicable at time k. Also, note that the row vector on the left side of (4.7) has been expanded with seven zeros to align with the augmented state vector in (3.7), which comprises ten states following velocity-based linearization. To extend the constraint for the entire horizon N, the vector of future scheduling parameters for constraints would be

$$P^a_k = \begin{bmatrix} \rho^a_k \\ \rho^a_{k+1} \\ \rho^a_{k+2} \\ \vdots \\ \rho^a_{k+N-1} \end{bmatrix}. \tag{4.8}$$

Now the obstacle avoidance inequality can be expressed in the form

$$\tilde{C}(P^a_k)X_k \preceq 1_N \otimes q_a + q(P^a_k), \tag{4.9}$$

where $\tilde{C}(P^a_k)$ has the form

$$\tilde{C}(P^a_k) = \begin{bmatrix} C(\rho^a_k) & 0 & 0 & \cdots & 0 \\ 0 & C(\rho^a_{k+1}) & 0 & \ddots & 0 \\ 0 & 0 & C(\rho^a_{k+2}) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & C(\rho^a_{k+N-1}) \end{bmatrix}, \quad q(P^a_k) = \begin{bmatrix} q(\rho^a_k) \\ q(\rho^a_{k+1}) \\ q(\rho^a_{k+2}) \\ \vdots \\ q(\rho^a_{k+N-1}) \end{bmatrix} \tag{4.10}$$

One further processing that should be done to replace the dependency on X_k with the dependency on U_k is to substitute with the prediction equation, so the inequality will be

$$\tilde{C}(P^a_k)\hat{S}\hat{U} \preceq 1_N \otimes q_a + q(P^a_k) - \tilde{C}(P^a_k)\hat{H}\hat{x_k}. \tag{4.11}$$

Here, $\hat{H}$ and $\hat{S}$ are dependent on the scheduling vector of the system. This inequality format is now primed for integration into the constraints vector in the QP formulation of the qLMPC algorithm (3.28).

4.3 Simulation Results

In this section, we illustrate how the qLMPC algorithm effectively addresses the obstacle avoidance problem within its framework. We replicate the scenarios used for the path-following problem, but this time we introduce two obstructing obsta-

cles in each scenario. This allows us to evaluate the aircraft's ability to maintain accurate path-following while successfully avoiding obstacles. We are examining the simulation results from two perspectives: successful path-following with obstacle avoidance, and the computational times required to execute this task.

4.3.1 Tracking

Here, we examine the tracking outcomes obtained by implementing the setup described in the preceding section, with adjustments made to include obstacle constraints. We assess the tracking performance across two scenarios, mirroring those used for path-following without obstacles' constraints in Chapter 3. However, in this case, we introduce two hindering obstacles to evaluate the algorithm's response to obstacles. Additionally, it is important to note that in our simulations, we depict the obstacles as cylinders for illustrative purposes to demonstrate the intended scenarios of obstacle avoidance. However, in the mathematical representation within the problem setup, we consider them as spheres.

In the first scenario in Fig. 4.2, we simulate two hindering obstacles with radii of 15 m and 25 m, respectively. The first obstacle is positioned approximately 30 m after the first waypoint, while the second obstacle is situated around 7 m after the second waypoint. We select a prediction horizon (N) of 20, which translates to a prediction distance of $(20 \times 0.01 \times 20 - 50) =$ 4-10 m, still suitable for a 3.1 m wing-span aircraft.

In this scenario, the aircraft effectively follows the planned path, fulfilling the primary objective. Upon encountering each obstacle, it navigates around them while maintaining its course along the planned path. Once past the obstacles, the aircraft smoothly converges back to the intended trajectory, minimizing tracking errors. This process repeats for the second obstacle as well.

In the second scenario, in Fig. 4.3, we introduce two hindering obstacles, each with a radius of 10 m and 30 m, respectively. The first obstacle is positioned 10 m beyond the first waypoint, while the second obstacle is situated 12 m before the fourth waypoint. We maintain the same prediction horizon (N) as in the first scenario. Through simulations, we observe successful path-following with effective obstacle avoidance for both hindering obstacles.

This approach shares similarities with the one presented in [Adhikari et al., 2020], where a customized nonlinear predictive controller was employed. However, the primary distinction lies in our adoption of a qLPV model for the problem, contrasting with the utilization of Taylor series linearization to transform the OCP into a quadratic form.

Initially, during controller tuning, our algorithm operated on the simplified 3D kinematic model of the aircraft, as introduced in (3.6). This served as an initial phase before transitioning to the full-state model.

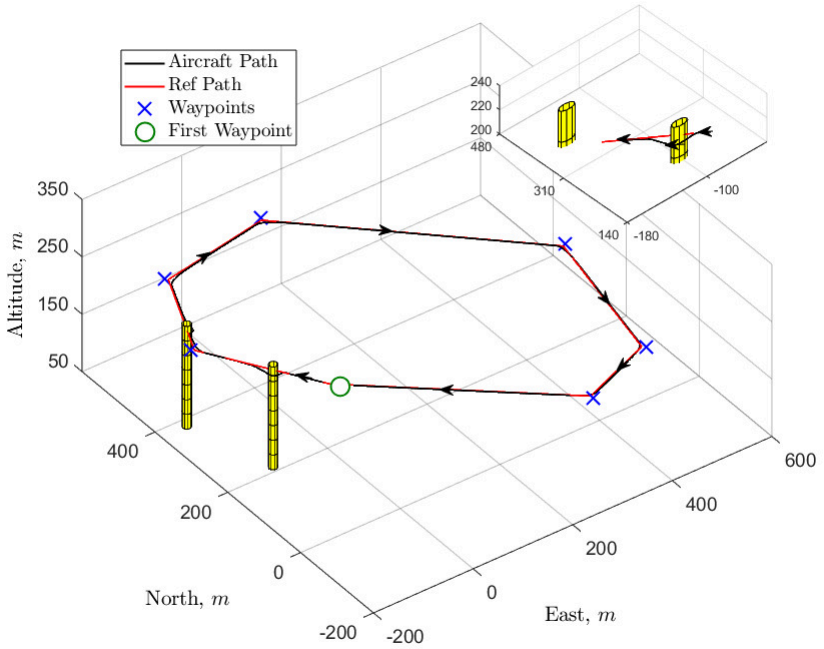

Figure 4.2: Obstacle avoidance: Aerodrome-circuit.

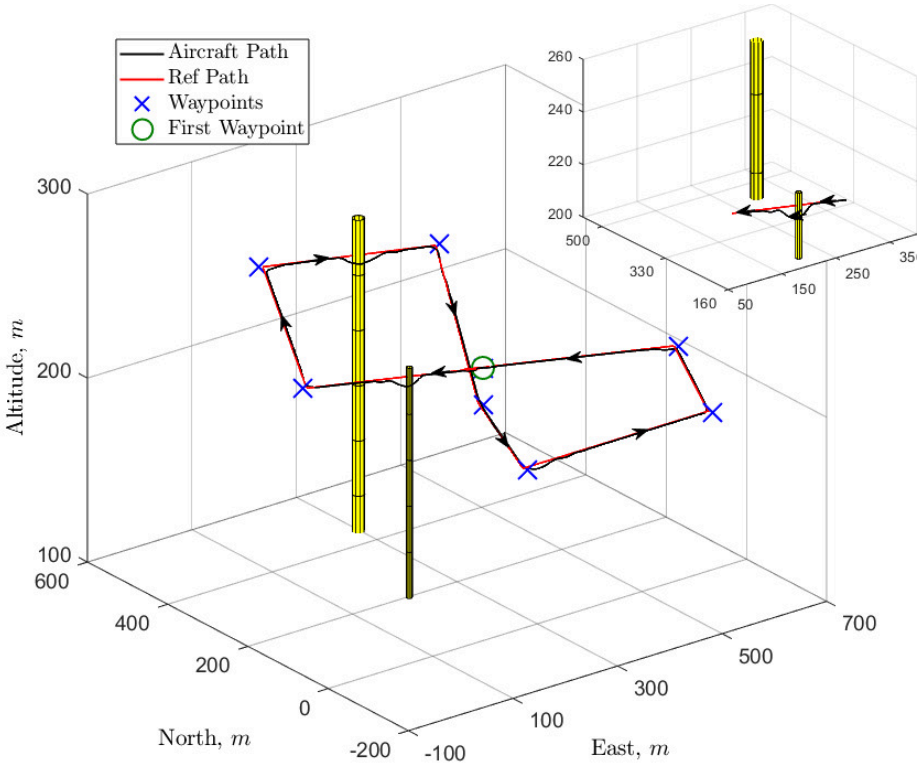

Figure 4.3: Obstacle avoidance: Roller-coaster.

4.3.2 Computational Times

Table 4.1 offers a summary of the computational times linked with implementing the qLMPC algorithm to incorporate obstacle avoidance. 'Matrix' denotes the time required to construct the system and constraints matrices at each sampling instant, 'QP' represents the time needed to solve the QP, and 'qLMPC' indicates the total execution time for the entire algorithm, with only a single iteration ($l = 1$ in Algorithm 3.1). The OCP is solved using an 8-core processor operating at 3.6 GHz, employing the *quadprog* solver from the MATLAB Optimization Toolbox. In our setup, a sampling time of 0.01 s is utilized for the path-following loop, 0.001 s for the low-level controller.

	Aerodrome-circuit path			Roller-coaster path		
	Min	Max	Mean	Min	Max	Mean
Matrix	0.452	3.643	1.294	0.431	3.501	1.147
QP	0.448	3.534	1.286	0.427	3.472	1.138
qLMPC	1.231	5.678	2.234	1.133	5.541	2.122

Table 4.1: Computational times overview in milli-seconds.

4.4 Summary and Discussion

In this chapter, we demonstrated how to integrate obstacle avoidance alongside path-following within a single controller. By treating obstacle constraints as qLPV constraints and updating them dynamically with the aircraft's position, we achieved low complexity and efficient computational times, enhancing suitability for real-time applications and flight experiments. However, there are some crucial points regarding the qLMPC obstacle avoidance problem that we believe are essential to highlight.

Initially, the simulations successfully integrated obstacle avoidance with the SW path-planner. However, when attempting to extend this to the hybrid path-planner, challenges emerged. The hybrid planner's additional state of arc length, crucial for path evolution, posed a conflict. While the path-following controller determined the path evolution velocity, it also conflicted with obstacle avoidance goals, potentially leading to infeasible solutions and unsafe aircraft trajectories. We explored adaptive tuning as a solution, adjusting the penalty weights to prioritize obstacle avoidance near obstacles and later aiding path convergence. While effective in some scenarios, this approach proved scenario-dependent and unreliable as a general solution.

Another challenge lies in analyzing the types of obstacles the aircraft can evade, a concern not exclusive to the qLMPC algorithm. Given the test aircraft's wingspan of 3.1 meters and velocity range of 20-50 m/s, it may struggle to avoid obstacles

smaller than roughly 4 meters. To accommodate this, extending the prediction horizon, particularly with shorter sampling times, may be necessary. However, this poses a potential computational burden on available hardware. In the simulated scenarios, this observation holds true. An alternative approach to address this issue is to increase r, thereby expanding the buffer zone around the obstacle. This enlargement provides the aircraft with additional space to maneuver and effectively avoid the obstacle.

In the simulations, we operated under the assumption of static obstacles with predetermined positions, presuming a known navigation map. However, we envision extending this approach to accommodate dynamic obstacles. Our methodology hinges on dynamically updating obstacle constraints at each time step, assuming constant obstacle positions. Yet, this adaptation necessitates additional sensors such as laser range finders (LRFs) for frequent updates on aircraft-obstacle distances. Furthermore, the effectiveness of this adaptation hinges on the dynamics and speed of the moving obstacles.

In the next chapter, we consider another crucial aspect concerning the qLMPC path-following problem: the analysis of its stability and the strategies employed to ensure the framework's stability while maintaining efficient performance.

Chapter 5

Stability of qLMPC Path-Following

Thus far, we have demonstrated how to formulate the UAV model using a qLPV representation based on velocity-based linearization. We then structured the path-following problem as an OCP within an iterative predictive framework to achieve optimal convergence while adhering to the imposed constraints.

Notably, this was achieved without addressing the stability guarantees of the framework. In this chapter, we examine the stability of the proposed framework using two different approaches. The first approach employs what are known as *terminal ingredients* to ensure that the vehicle's states remain bounded at the end of the prediction horizon, thereby enhancing overall system stability. However, applying terminal constraints to velocity models is challenging due to their higher order, which complicates the solution of the associated LMIs. Moreover, incorporating terminal constraints increases the computational burden of the qLMPC algorithm, potentially limiting its real-time applicability.

To address these challenges, we adopt a second approach tailored to velocity models. This method introduces two additional constraints to the OCP, eliminating the need for complex LMI solutions and enhancing real-time feasibility.

5.1 Stability Guarantees via Terminal Ingredients

In this section, we analyze the stability of the qLMPC path-following framework by incorporating terminal ingredients. This methodology entails solving offline LMIs to determine the requisite terminal components. These precomputed elements are subsequently integrated into the online solution of the OCP, with the corresponding constraints embedded within the OCP.

These terminal ingredients include a terminal cost function (Ψ_N), basically a Lya-

punov function for the closed-loop system; a terminal constraint set ($\mathcal{X}_N$), a designated set that the system's state must enter by the end of the prediction horizon, encompassing the origin of the state space and remaining invariant under a stabilizing control law; and a terminal state feedback control law ($u_k = Fx_k$), a hypothetical control law applied when the system's state reaches the terminal constraint set, ensuring that the state remains within this set. The process of calculating these components is typically conducted offline. While the terminal controller is virtual, the terminal constraint and terminal cost must be implemented online. By doing this, we can write the cost function as follows

$$J_k = \sum_{i=1}^{N} x_{k+i}^\top Q x_{k+i} + \underbrace{x_{k+N}^\top P x_{k+N}}_{\Psi_{k+N}} + u_{k+i-1}^\top R u_{k+i-1}. \tag{5.1}$$

Note that the cost function in (5.1) is different from the cost function in (3.19) as this one contains an extra term representing the terminal cost ($\Psi_{k+N} = x_{k+N}^\top P x_{k+N}$). Now we consider the control law that solves the OCP iteratively

$$\begin{aligned} &\min_{U_k} J_k(x_k, U_k, X_k) \\ &\text{s.t.} \\ &x_{k+i} = Ax_{k+i} + Bu_{k+i} \\ &x_{k+i} \in \mathcal{X} \\ &u_{k+i} \in \mathcal{U} \\ &x_{k+N} \in \mathcal{X}_N \end{aligned} \tag{5.2}$$

The terminal components are integrated into the optimization as follows: the terminal constraint is applied to the terminal state x_{k+N}; the terminal cost $\Psi(x_{k+N})$ is incorporated into the cost function; and a nominal state feedback controller $F(x_k)$ maps the state x_k to the control input u_k. This feedback law is designed to stabilize the system to the origin once the states are within $\mathcal{X}_N$, ensuring that the states remain in $\mathcal{X}_N$ thereafter, thereby maintaining its invariance under $u_k = F(x_k)$. To stabilize the qLMPC path-following framework, we adapt the stability results presented in [Löfberg, 2001] for LTI systems and their extension to qLPV systems, detailed in [Cisneros et al., 2016] as follows

Theorem 5.1

[Cisneros et al., 2016] Assume a nominal controller $F(x) = Fx$, a terminal state domain $\mathcal{X}_N$ and a terminal state weight $\Psi(x)$ exist such that

1. $0 \in \mathcal{X}$
2. $(A(\rho(x, Fx)) + B(\rho(x, Fx))F)x \in \mathcal{X}, \quad \forall x \in \mathcal{X}$
3. $\Psi((A(\rho(x, Fx)) + B(\rho(x, Fx))F)x) - \Psi(x) \leq -x^\top Qx - x^\top F^\top RFx, \quad \forall x \in \mathcal{X}$
4. $Fx \in \mathcal{U}, \quad \forall x \in \mathcal{X}$

hold. Then, assuming feasibility of the initial state, an MPC controller solving the OCP (5.2) guarantees asymptotic stability.

The assumptions outlined in Theorem 5.1 are enforced through the offline solution of a set of LMIs. These LMIs are solved over a discretized grid spanning the admissible parameter set, under the assumption that the unknown matrix variables are independent of the parameters. The control constraint set is mapped into the state space $\mathbb{R}^n$ via the nominal control law $u = Fx$, resulting in a polyhedral representation. Within this polyhedron, we identify the largest ellipsoid defined by $\mathcal{X} = \{x \mid x^\top Wx \leq 1\}$, compatible with control constraints which is then designated as the terminal set. To determine appropriate selections for F, $\mathcal{X}$, and $\Psi(x)$, we consider the following LMIs.

Feasibility Problem

Define $P = Y^{-1}$ and $F = XY^{-1}$ such that

$$\begin{bmatrix} Y & (A(\rho)Y + B(\rho)X)^\top & Y & X^\top \\ A(\rho)Y + B(\rho)X & Y & 0 & 0 \\ Y & 0 & Q^{-1} & 0 \\ X & 0 & 0 & R^{-1} \end{bmatrix} \succeq 0, \quad \forall \rho \in \mathcal{P}. \tag{5.3}$$

Terminal Region

Solve for $W = Z^{-1}$ such that

$$\begin{aligned} &\max_{Z} \quad det(Z) \\ \text{s.t.} \quad &\begin{bmatrix} -Z & Z(A(\rho) + B(\rho)F)^\top \\ (A(\rho) + B(\rho)F)Z & -Z \end{bmatrix} \prec 0, \\ &F_i Z F_i^\top \leq \bar{u}_i^\top, \\ &\forall \rho \in \mathcal{P}. \end{aligned} \tag{5.4}$$

Here, F_i represents the i-th row of the matrix F, and correspondingly, $\bar{u}_i$ denotes the i-th element of the vector $\bar{u}$, which comprises the permissible values of u. To achieve a linear relationship with respect to the unknown matrix variables X and Y, we introduce a change of variables defined by $X = FY$. The matrix $W \succ 0$,

along with the terminal state feedback gain matrix F, are to be determined. In the process of identifying the largest ellipsoid $\mathcal{E}_W$ that satisfies the given constraints, it is important to note that the volume of this ellipsoid increases monotonically with $\det(W^{-1})$. This property can be leveraged to maximize the ellipsoid's volume within the feasible region.

The objective is to determine a terminal constraint set $\mathcal{X}_N$ that is as large as possible, thereby reducing the necessity for an extended prediction horizon to ensure the terminal set is reachable. Since the size of the terminal ellipsoid is constrained by input limitations, a terminal controller F with smaller gains can facilitate a larger terminal set. To address this, an iterative approach can be developed to optimize the terminal set size while adhering to input constraints.

Selecting the terminal controller by solving the associated LQR problem can result in a relatively small terminal state domain, $\mathcal{X}_N$, determined by the Riccati solution matrix P. This limitation implies that for states distant from the origin, the problem may become infeasible or require an excessively long prediction horizon. To address this issue, instead of using the same matrix P for both the terminal cost and the terminal constraint, one can independently determine the largest ellipsoid $\mathcal{X}_N$ based on the terminal controller F and a terminal cost function defined by P. This approach allows for a more expansive terminal state domain, enhancing feasibility and reducing the required prediction horizon length. To satisfy Condition (3) in Theorem 5.1, we can select P such that $P \succ 0$ and fulfills

$$A^\top P A - P \prec Q. \tag{5.5}$$

We assume that the terminal constraint set $\mathcal{X}_N$ is represented as an ellipsoid, the virtual controller $u = Fx$ is a linear state feedback law, and the terminal cost $\Psi(x) = x^\top P x$ is quadratic. Consequently, it is necessary to select matrices F and P that fulfill the conditions outlined in Assumption (3), leading to

$$(A + BF)^\top P (A + BF) - P \preceq -Q - F^\top R F \tag{5.6}$$

Then, we use the following result [Cisneros and Werner, 2021]

Theorem 5.2

Assume P and F satisfy 5.3 and W satisfies 5.4. Then $\Psi(x) = x^\top P x$, $\mathcal{X} = \{x | x^\top W x \leq 1\}$ and F satisfy the assumptions of Theorem 5.1.

Accordingly, we specify the terminal constraint set as

$$\mathcal{X}_N = \mathcal{E}_W = \{x \in \mathbb{R}^n : x^\top W x \leq 1\} \tag{5.7}$$

Under linear constraints on u, this leads to an admissible polyhedron in state space. We then need to search for the largest ellipsoid $\mathcal{E}_W$ that is contained in this polyhedron. To find the largest ellipsoid that satisfies the constraints, we use the

following lemma

Lemma 5.1. *[Löfberg, 2001]: The maximum of $c^\top x$ in the ellipsoidal set $x^\top W x \leq 1$ is $\sqrt{c^\top W^{-1} c}$.*

Note that the LMIs (5.3) and (5.4) are solved iteratively. Additionally, we solve these LMIs on a parameter-independent grid, which introduces conservatism into the solution. To mitigate this conservatism, one could consider solving the LMIs with dependence on the scheduling parameters. Another limitation of applying this approach to velocity models is their relatively higher order compared to state-space models, which can increase computational complexity.

5.2 Stability Analysis for Velocity-Based Systems

In Section 5.1, we discussed qLMPC path-following stability guarantee through the incorporation of terminal ingredients. While effective, this approach presents certain challenges, particularly when applied to velocity models. One significant issue is the relatively high order of velocity models, which complicates the solution of the corresponding LMIs. Additionally, incorporating terminal constraints increases the computational load of the qLMPC algorithm, potentially hindering real-time implementation and making practical application more demanding.

In 3.3, we defined the continuous-time version of these models, where linearity naturally arises from applying the chain rule to the time derivatives of the state and output equations. However, this property does not directly carry over to the discrete-time case, as the difference operator does not inherently preserve linearity. To address this, we rely on a result from differential calculus—the multivariable extension of the Mean Value Theorem—outlined in [Zemouche et al., 2005], which facilitates the derivation of a discrete-time linearized model.

Lemma 5.2. *[Zemouche et al., 2005]: Let $g(x) : \mathbb{R}^n \to \mathbb{R}^q$, assume that g is differentiable on $Co(a,b)^a$, then there exist constant vectors $c_i \in Co(a,b)$, $c_i \neq a, c_i \neq b$, $i = 1, \cdots, q$ such that: $g(a) - g(b) = \left(\sum_{i,j=1}^{q,n} e_q(i) e_n(j)^\top \frac{\partial g_i}{\partial x_j}(c_i) \right) (a - b)$.*

where $e_s(i)$ denotes the i^{th} column of I_s, and $\text{Co}(a,b)$ represents the convex hull of vectors a and b. Based on Lemma 5.2, the velocity form of system 3.3 is expressed as

$$\begin{aligned} \begin{bmatrix} y - k \\ \Delta x_{k+1} \end{bmatrix} &= \begin{bmatrix} I & \nabla_x h(x_k) \\ 0 & \nabla_x f(x_k, u_k) \end{bmatrix} \begin{bmatrix} y_{k-1} \\ \Delta x_k \end{bmatrix} + \begin{bmatrix} 0 \\ \nabla_u f(x_k, u_k) \end{bmatrix} \Delta u_k \\ y_k &= \begin{bmatrix} I & \nabla_x h(x_k) \end{bmatrix} \begin{bmatrix} y_{k-1} \\ \Delta x_k \end{bmatrix} \end{aligned} \tag{5.8}$$

The system in 5.8 is a discrete-time qLPV representation. In the following, we

analyze stability by incorporating two additional constraints into the OCP. This methodology is inspired by the work of [Cisneros and Werner, 2021], where the authors applied this approach to velocity models, thereby circumventing the need to solve complex LMIs. Although their method was demonstrated for systems stabilized around a fixed point, which differs from our scenario involving a moving reference, we address this challenge by utilizing the error dynamics as the system output, which remains centered around zero.

This approach can be simply described as achieving a steady-state output, characterized by the error dynamics between the desired path and the aircraft. The primary aim is to guarantee that, at the end of the prediction horizon, the variation in this output approaches zero, signifying a stable state. To attain this objective, two additional constraints need to be integrated into the OCP (3.28).

This approach relies on the advantages of velocity models over conventional state-space models (e.g., as discussed in [Ferramosca et al., 2009)], wherein all equilibria are mapped to the origin ($\Delta x = 0, \Delta u = 0$). This characteristic eliminates the need for parameterizing all equilibria, a process that can be particularly complex for nonlinear systems. This simplification is especially beneficial for tracking problems. Although the OCP does not necessitate the computation of steady-state values, such values remain essential to satisfy set point constraints. In the following, we illustrate how we adapt the OCP to incorporate stability guarantee conditions.

Consider a nonlinear system (3.6) and the augmented velocity model (3.7). Suppose both the state and input are subject to constraints $x \in \mathcal{X}$ and $u \in \mathcal{U}$, respectively. Our objective now is to devise an optimal feedback control law to follow a given reference set by the path-planner module, while adhering to the specified constraints and guaranteeing stability of the algorithm. In the subsequent implementation, the following assumptions are made:

Assumption 1: Model (3.6) is stabilizable $\forall \rho \in \mathcal{P}$.

Assumption 2: The number of tracked outputs is less than or equal to the number of inputs ($l \preceq m$).

Assumption 3: The system possesses a continuous locus of forced equilibria, i.e. the system does not have an isolated forced equilibria.

Assumption 1 denotes a condition for local stabilizability, whereas Assumption 3 may seem restrictive; nonetheless, it is implicitly assumed in numerous conventional tracking MPC algorithms, especially when addressing unreachable reference set points. We then employ the following result to ensure the stability of velocity-based models.

Theorem 5.3

[Cisneros and Werner, 2021] Let Assumptions 1,2, 3 hold, and let the terminal offset penalty $P = \alpha Q_1$ for some $\alpha \succ 0$. Given a set point y_{sp}, the control law $\kappa(y_k, \Delta x_k, y_{sp})$ derived from the solution of OCP starting from a feasible state $[y^\top \ \Delta x^\top]^\top$ stabilizes the system, is recursively feasible and makes the output converge to one of the following

1. y_{sp} is $y_{sp} \in \mathfrak{R}_y$
2. $\tilde{y}$ if $y_{sp} \notin \mathfrak{R}_y$ where

$$\tilde{y} = \arg\min_{y \in \mathfrak{R}_y} ||y - y_{sp}||_P^2$$

The proof of Theorem 5.3 is structured into three main steps: Step I establishes recursive feasibility, Step II demonstrates the monotonic decrease of the cost function, thereby ensuring stability, and Step III shows that the only equilibria permitted by the optimization problem are those stated in items (1)–(2) of Theorem 5.3. Steps I and II follow standard MPC methodology, adapted here for the velocity-based algorithm. Step III builds on the approach in [Limón et al., 2008], where convergence to an optimal steady state is established through convexity arguments. For a detailed proof, refer to [Cisneros and Werner, 2021], such that $\mathfrak{R}_y$ denotes the permissible output set. Taking into account the aforementioned theorem, we can write the OCP as follows

$$\Delta U_k^* = \min_{\Delta U_k} J_k \tag{5.9a}$$

s.t.

$$\dot{x}_{k+1} = A(\rho)\dot{x}_k + B(\rho)\dot{u}_k \tag{5.9b}$$

$$x_k \in \mathcal{X} \tag{5.9c}$$

$$u_{k+i-1} = u_{k-1} + \sum_{j=0}^{i} \Delta u_{k+j} \in \mathcal{U}, \quad i \in [1, N] \tag{5.9d}$$

$$\Delta u_{k+i-1} \in \tilde{\mathcal{U}} \tag{5.9e}$$

$$y_{k+N} = y_{sp} \tag{5.9f}$$

$$\Delta x_{k+N} = 0. \tag{5.9g}$$

To provide deeper insight into the OCP formulation in (5.9), additional constraints have been introduced. The dynamic constraints from (5.9b) remain unchanged, as do the state constraints in (5.9c), the control input constraints in (5.9d), and the control input rate constraints in (5.9e). Furthermore, two stability-related constraints, given in (5.9f), have been added to enforce convergence to the reference, defined by the set-point output y_{sp}.

It is important to note that the system has been augmented with path error dynamics, and the output is represented by this error. Consequently, y_{sp} corresponds to the desired error value, which is zero. The constraint in (5.9g) ensures that the

obtained solution corresponds to a steady-state condition.

Remark 5.1: The constraint (5.9f) can be incorporated into the cost function to better accommodate our path-following problem. This adaptation converts it into a soft constraint rather than a hard one. This adjustment recognizes that achieving exact zero error may be impractical due to the dynamics of the aircraft and possible wind disturbances encountered during UAV flight.

Remark 5.2: For similar reasons, it is necessary to relax the constraint (5.9g) to a small value ε, where $\underline{\varepsilon} \preceq \Delta x_{k+N} \preceq \bar{\varepsilon}$. This relaxation is justified by the requirements of path-following. At the end of the prediction horizon, the set position point should be very close to the reference position point, and the change rate of this position should ideally be as minimal as possible to guarantee stability and ensure high-performance path-following. However, achieving an exact zero change rate is not feasible because the path is continuous, and the aircraft's airspeed cannot be reduced to zero during path-following. Such a condition would result in a crash and could render the OCP infeasible from a mathematical standpoint. In this work, we impose heuristic constraints by selecting very small bounds on the state derivatives. Alternatively, a simplified formulation of terminal ingredients, introduced in Section 5.1, could be employed to explicitly ensure the boundedness of the system's derivatives, potentially offering a more robust and systematic stability guarantee.

Remark 5.3: Concerning the convergence properties of the qLMPC algorithm, our study assumes that the algorithm consistently converges to the optimal solution. In [Hespe and Werner, 2021], the authors pointed out that the qLPMC algorithm typically converges to a suboptimal solution, and proposed a way of correcting this and make it converge to the optimal solution.

5.3 Simulation Results Using Terminal Ingredients

In this section, we examine the stability of the qLMPC path-following algorithm using the ULTRA-Extra UAV case study using the scenario introduced in Chapter 3. Unlike previous analyses, we modify the cost function to incorporate a terminal cost ($x^\top P x$) and introduce a terminal constraint, as discussed in the preceding section.

Recall that within the path-following loop, a 3D kinematic model of the UAV is employed, featuring a state dimension of $n = 18$ and a control input dimension of $u = 4$. At the low level, a high-fidelity model of the aircraft is used. By solving the offline LMIs, we have determined the matrices F, W, and P. Figure 5.1 illustrates the singular values of the system, which have been found sufficient for solving the corresponding LMIs. Lower singular values indicate a larger ellipsoid for the terminal set, as evident in Figure 5.2, where the ellipsoid after two iterations is

adequately expansive. It is important to note that Figure 5.2 depicts the ellipsoid in only two dimensions, specifically the x-y plane. The overall ellipsoid is relatively large due to the high order of the model, encompassing 18 dimensions, including the aircraft's position.

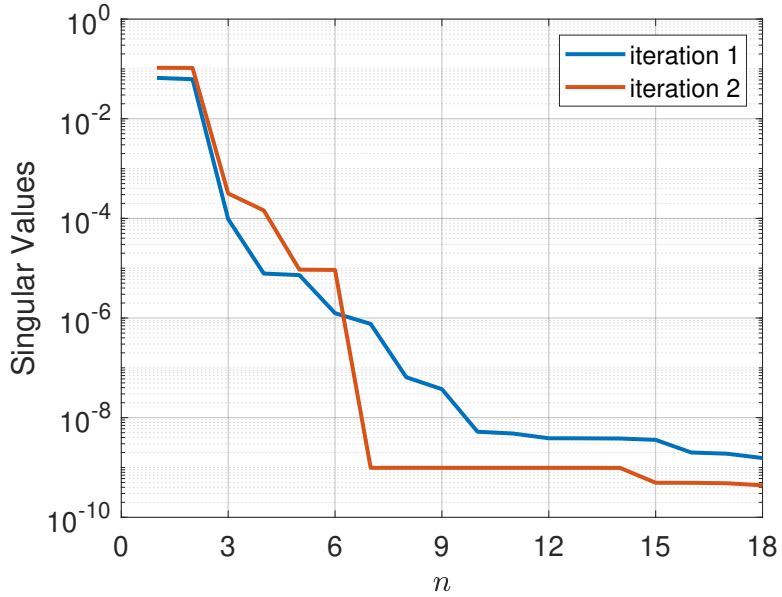

Figure 5.1: System singular values plotted over iterations.

We solved the LMIs on a grid, selecting grid points ranging from 1 to 4. After solving these LMIs, we determined the matrices W, P, and F. Subsequently, we incorporated the terminal cost $x^\top Px$ into the objective function and enforced the terminal constraint defined by $x^\top Wx \leq 1$ within the OCP.

Figure 5.3 presents a comparison of the fulfillment of the terminal constraint when applying the obtained solutions from solving the LMIs at different grid points. In Fig. 5.4, we illustrate the aircraft's path-following performance, which is affected, as expected, by the application of the terminal cost and terminal constraint. Notably, deviations from the designed path can reach up to 10 m at certain points; however, such deviations are still considered acceptable for UAV path-following tasks.

Figure 5.5 provides a detailed overview of the tracking data for various states, including position components, heading angle (ψ), flight path angle (γ), and the arc length extra state (θ). An additional consideration is the computational burden introduced by the use of terminal ingredients.

As illustrated in Fig. 5.6, the execution time of the qLMPC algorithm increases by a factor of approximately 2.5 compared to Fig. 3.16, primarily due to the application of terminal ingredients to a high-order model. It should be noted that the framework, including the stability constraints, was implemented in MATLAB/Simulink using both the `quadprog` and Lemke solvers. The Lemke solver demonstrated superior computational efficiency, achieving an average execution

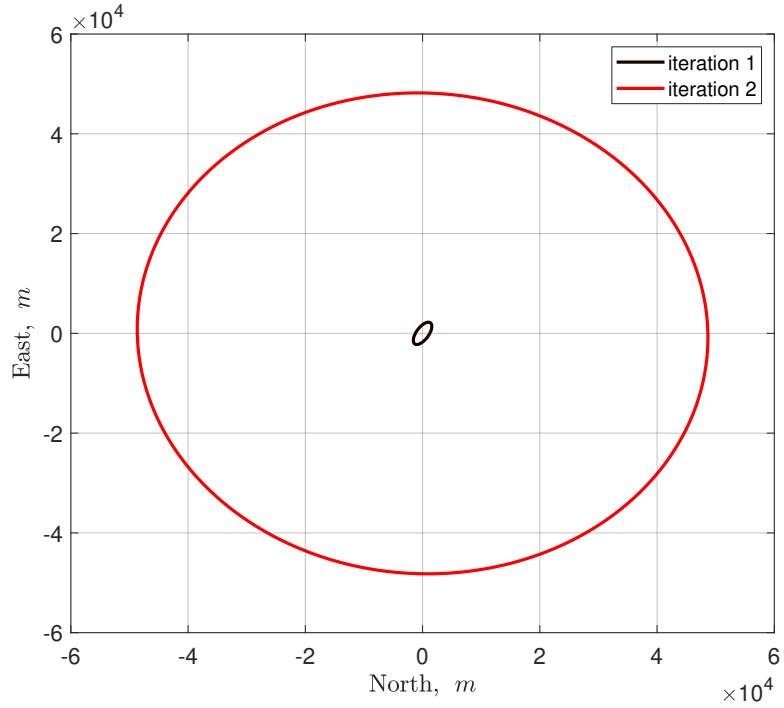

Figure 5.2: $x - y$ cross section of the initial solution and after one iteration.

time of approximately 5.5 ms, outperforming `quadprog`, which averaged around 8.1 ms. In practical real-time applications, it is also worth mentioning that practitioners may choose to relax strict stability guarantees if they are not critical for the specific mission objectives.

5.4 Simulation Results for Velocity-Based Models

In this section, we present the simulation outcomes of the enhanced algorithm. We utilize the same scenarios as those introduced in Chapter 2 and employed in Chapter 3 for demonstrating path-following.

The objective is to attain efficient path-following performance while ensuring algorithm stability. As outlined in (5.9f) and (5.9g), we introduced two additional constraints to the OCP to guarantee stability. These constraints ensure that the solution obtained at the end of the prediction horizon maintains a steady-state. Our simulations demonstrate the fulfillment of these constraints alongside the algorithm's tracking performance.

Figure 5.7 demonstrates efficient path-following in the 'Aerodrome circuit' path. The green circle ($\circ$) depicts the starting waypoint, and the blue ($\times$) marks the waypoints the aircraft must traverse. The red line represents the reference path generated by the hybrid path-generator, while the black line indicates the actual path of the aircraft. The figure exhibits satisfactory performance as the aircraft successfully navigated through all waypoints.

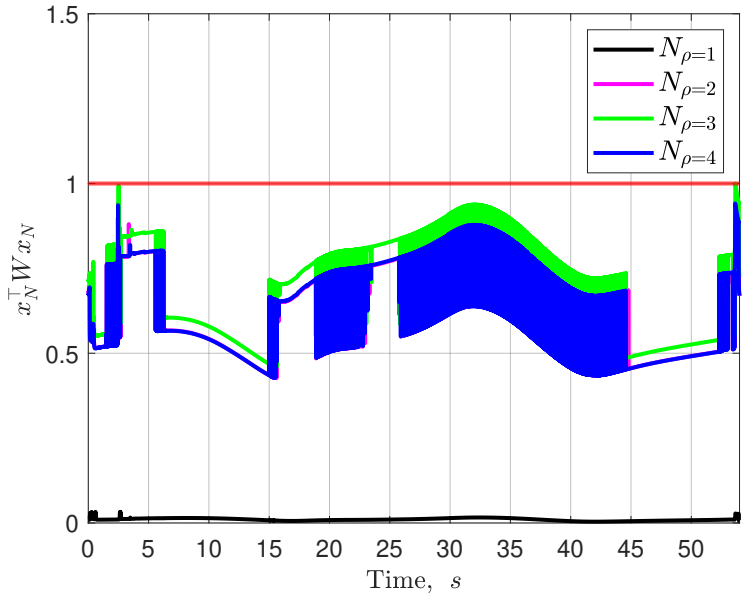

Figure 5.3: Fulfillment of the terminal constraint during online execution.

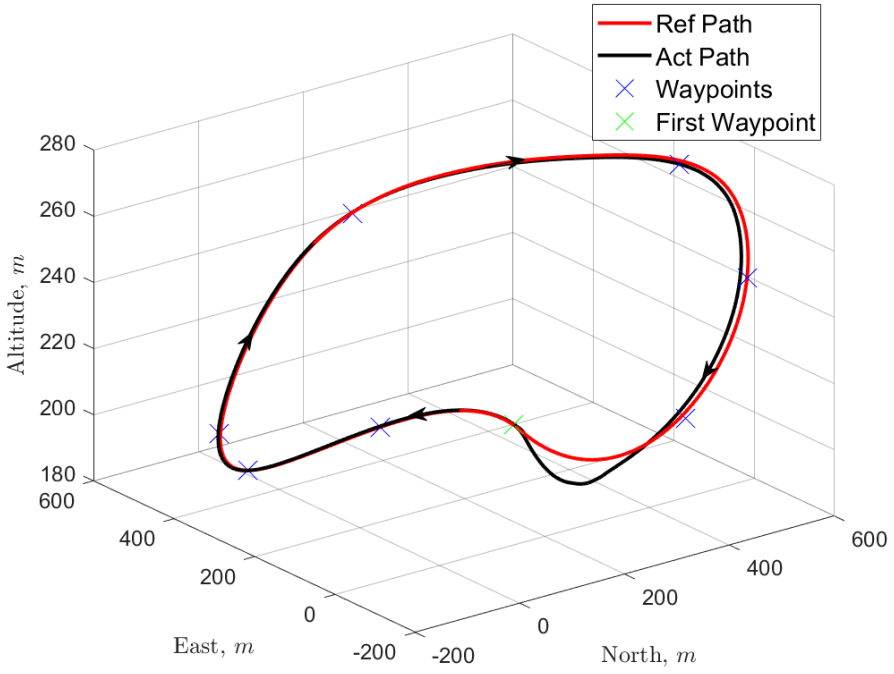

Figure 5.4: Path-following scenario with terminal ingredients.

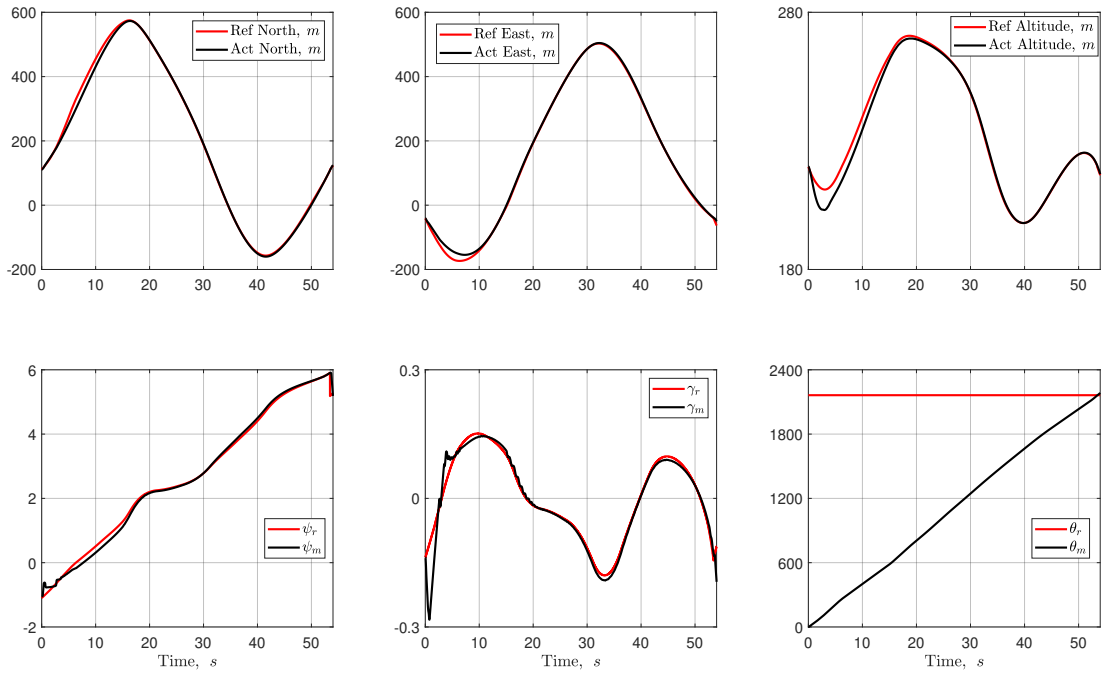

Figure 5.5: State tracking performance with terminal ingredients.

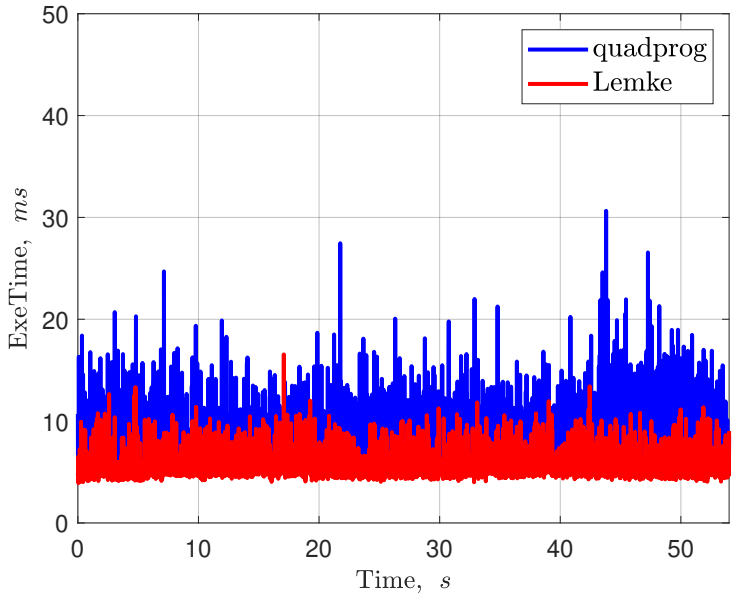

Figure 5.6: Online computation time with terminal ingredients applied.

Figure 5.8 illustrates the accurate tracking of individual state variables, including the position components x, y, and z, as well as the heading angle ψ and the flight-path angle γ. This behavior validates the satisfaction of the first stability condition in (5.9f), demonstrating that the spatial error between the aircraft and the reference path remains close to zero by the end of the prediction horizon ($y_{k+N} = y_{sp}$). It is important to note that achieving exactly zero error is impractical in real-world scenarios; hence, this requirement is not enforced as a hard constraint, as discussed in the previous section.

Figure 5.9 showcases the satisfaction of the second stability constraint in (5.9g), ensuring that the solution obtained from the OCP achieves a steady state. It is worth noting, as mentioned in Remark 5.2, that exact zero changes in state or output derivatives at the end of the horizon are unattainable in the context of the path-following problem. Instead, we relaxed the constraint to a small tolerance, typically around $\pm$ 0.5 m for position components and $\pm$ 0.01 rad/s for the heading and flight path angles. This tolerance ensures the feasibility of the OCP without compromising its efficiency.

Another significant point to note in Figure 5.9 (bottom-right) is that the arc length parameter evolution remains unconstrained. This is essential to guarantee the accurate progression of the path, as this parameter dictates path evolution and is controlled by the path-following controller. Constraining it could result in inaccurate path evolution. Additionally, it is specific to the hybrid path-generator; if an alternative path-generator is utilized, this parameter can be omitted from the entire framework.

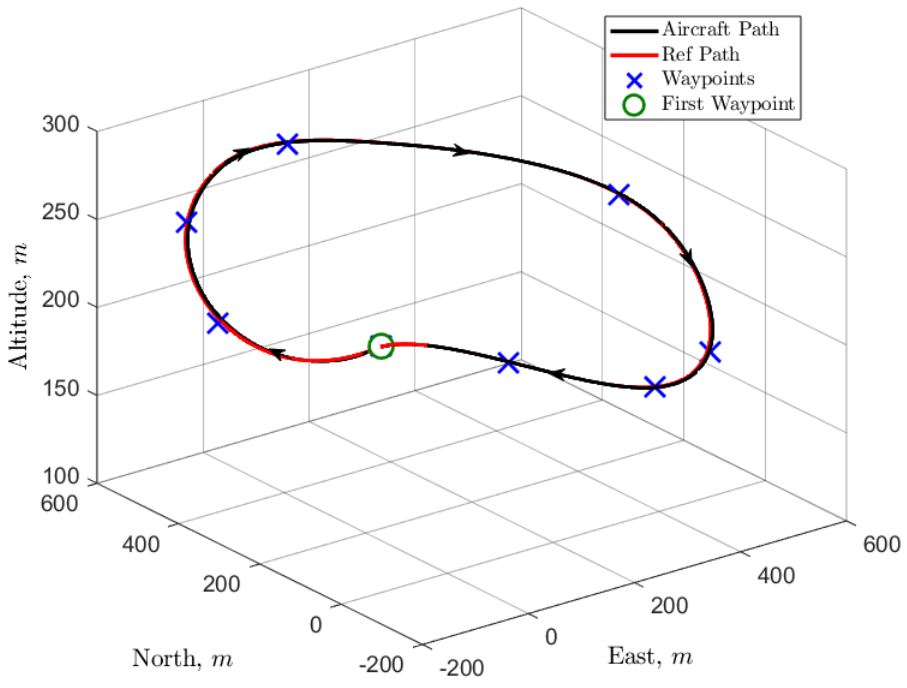

Figure 5.7: Aerodrome-circuit - tracking.

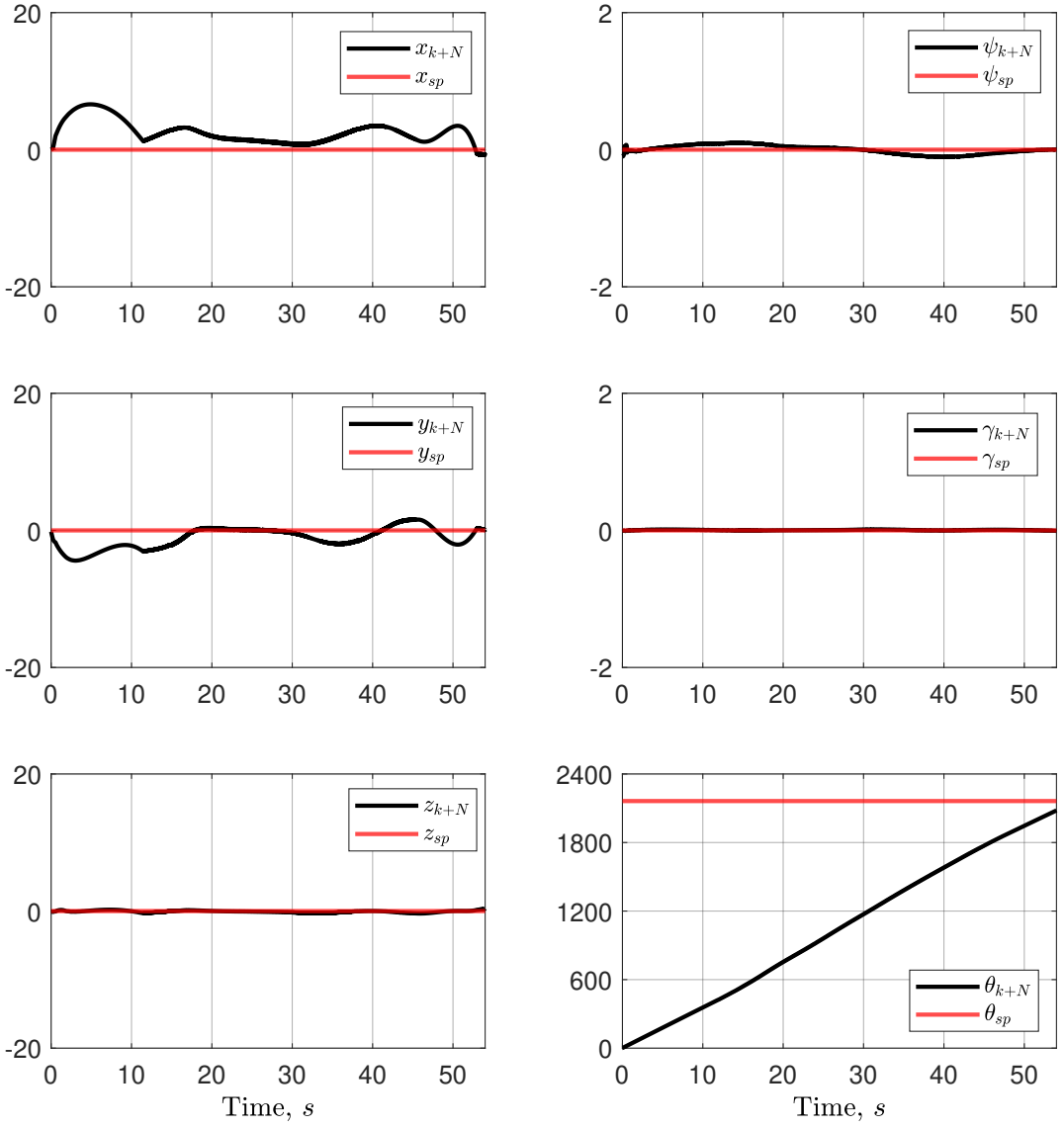

Figure 5.8: Aerodrome-circuit - fulfillment of first stability constraint ($y_{k+N} = y_{sp}$).

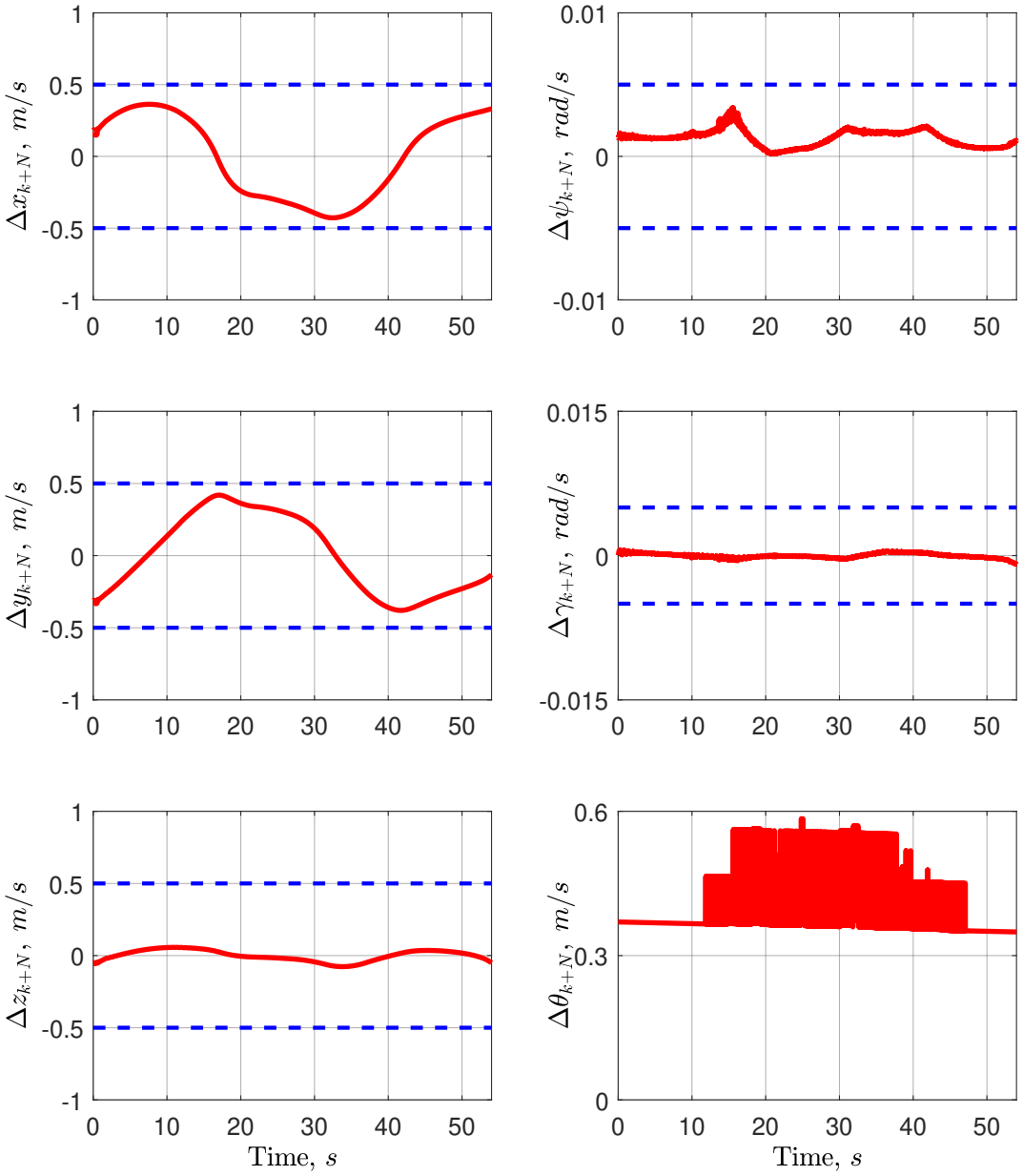

Figure 5.9: Aerodrome-circuit - fulfillment of second stability constraint($\Delta x_{k+N} = 0$).

Similarly, we applied identical constraints to the 'Roller-coaster' scenario, maintaining the previously mentioned tolerances of $\pm$ 0.5 m for the position components and $\pm$ 0.01 rad for the angles. Figure 5.10 demonstrates efficient path-following performance under the new problem formulation, while Fig. 5.11 confirms the fulfillment of the first stability guarantee constraint in problem (5.9). Additionally, Fig. 5.12 illustrates the satisfaction of the second stability guarantee constraint in the same problem.

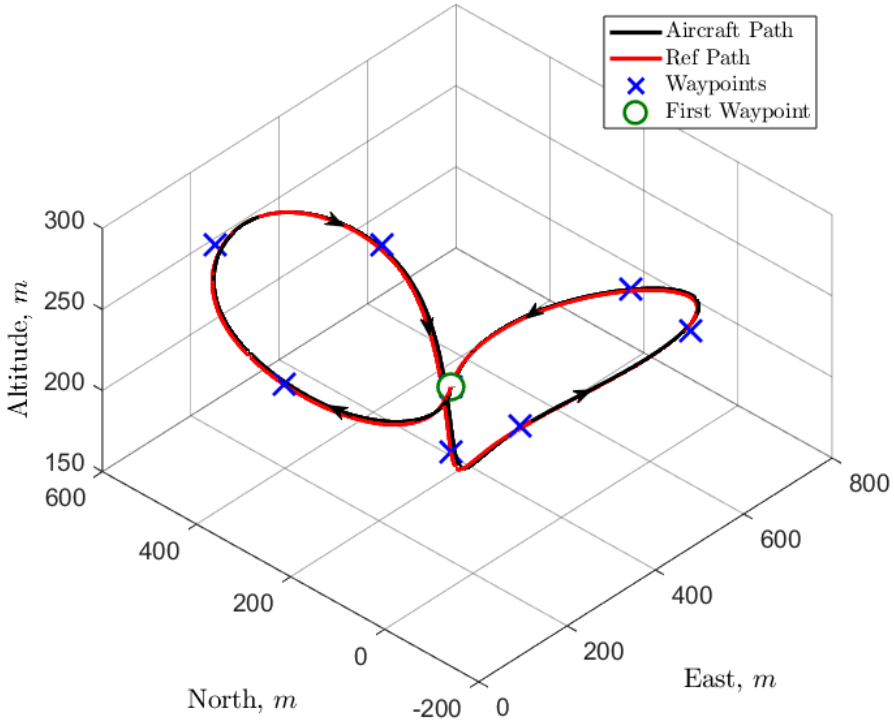

Figure 5.10: Roller-coaster - tracking.

5.4.1 Computational Performance Under Stability Constraints

In the following, we present an overview of the computational performance when stability constraints are applied to velocity-based models. This analysis is essential for comparing the two stability approaches introduced in this work. As shown in Section 5.3, incorporating terminal ingredients into the OCP significantly increases the computational load, with an average execution time of approximately 6 ms.

In contrast, Fig. 5.13 illustrates that for the first simulation scenario (Aerodrome circuit), the average computation time is reduced to approximately 1.5 ms. While some peaks are observed—primarily due to the high maneuverability required along certain path segments—the average remains consistently low. Similarly, Fig. 5.14 shows that in the second scenario (roller coaster path), the average computation time remains around 2 ms.

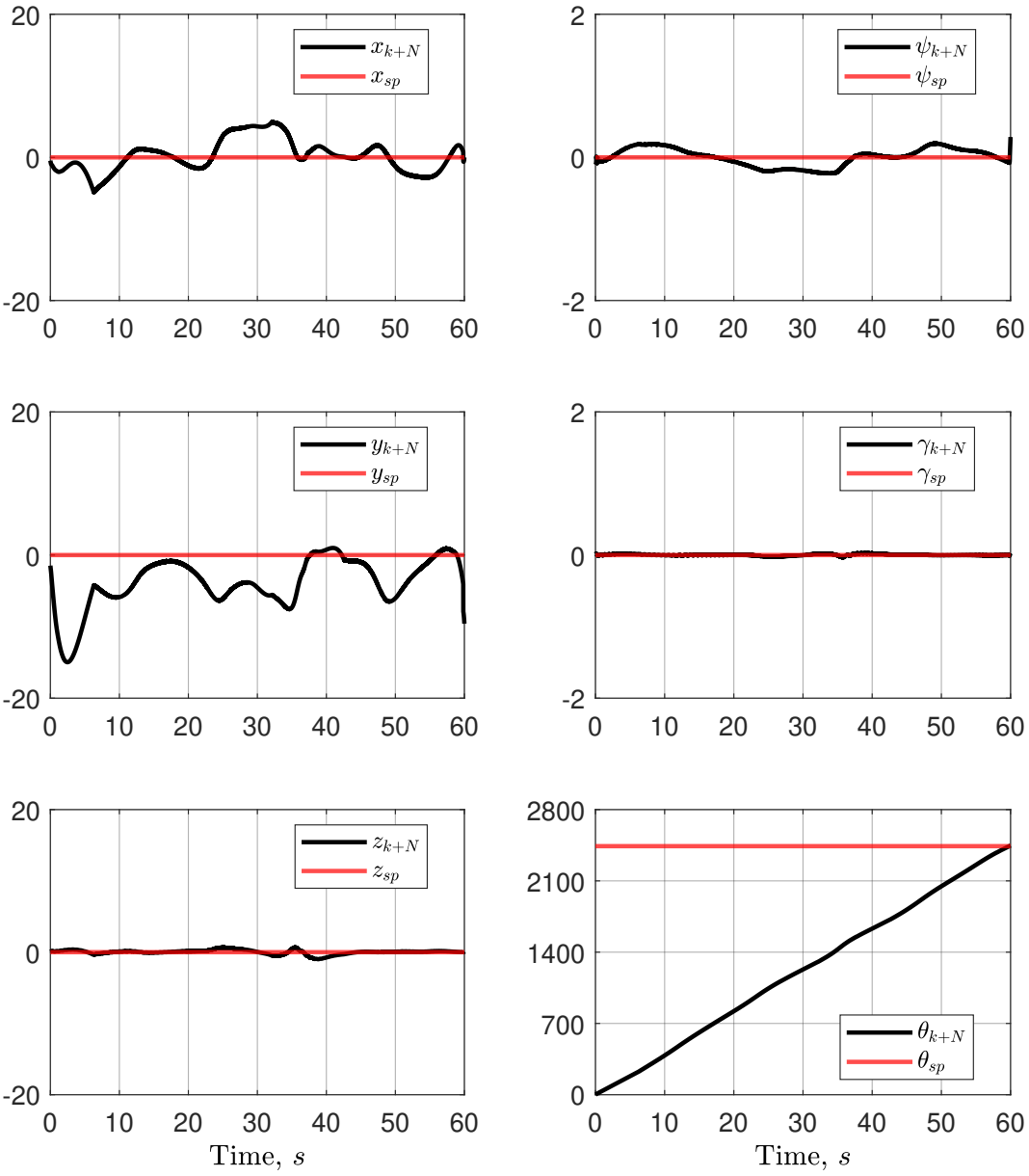

Figure 5.11: Roller-coaster - fulfillment of first stability constraint($y_{k+N} = y_{sp}$).

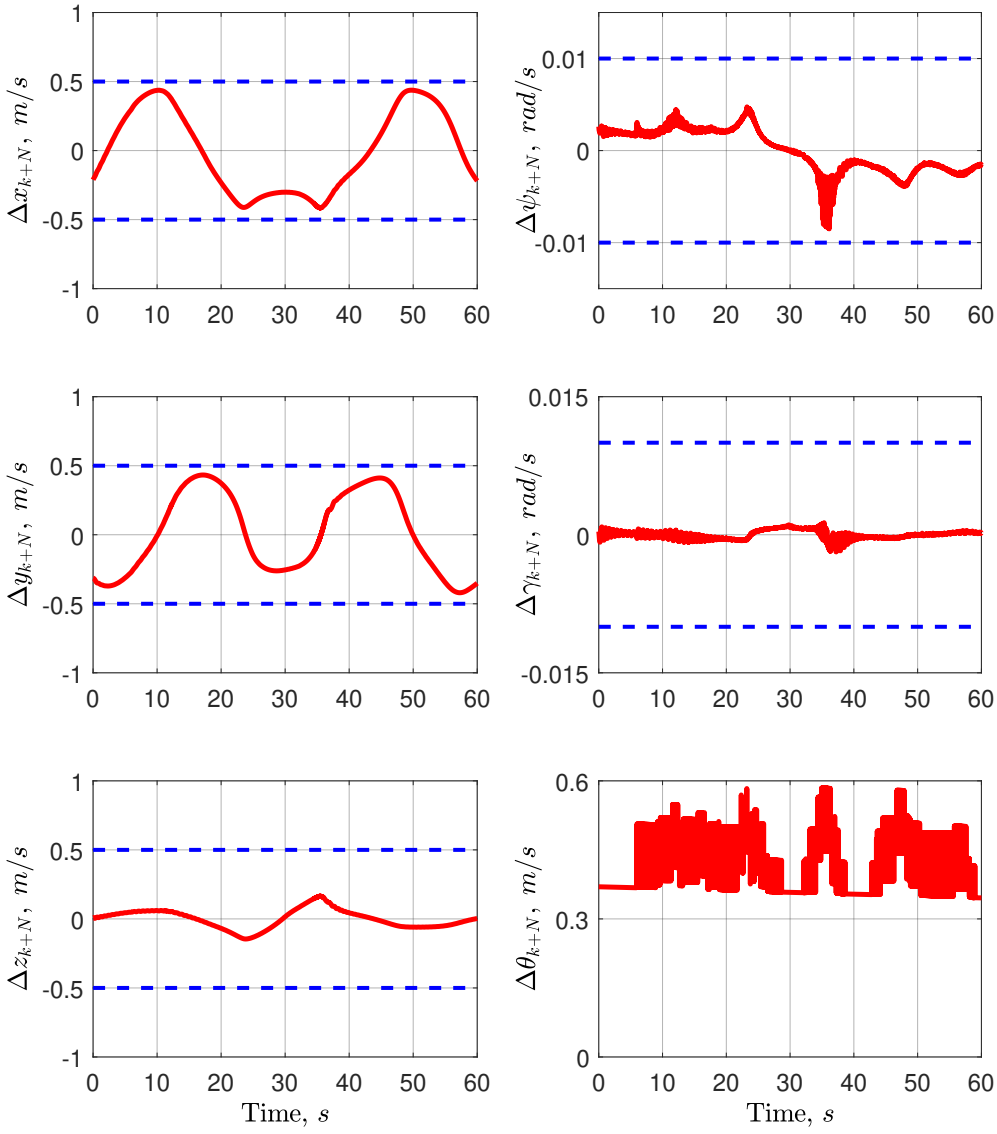

Figure 5.12: Roller-coaster - fulfillment of second stability constraint($\Delta x_{k+N} = 0$).

These results indicate that the second stability approach, which avoids terminal ingredients, leads to a substantial reduction in computational effort. This improvement is particularly reasonable given the higher-order nature of velocity-based models.

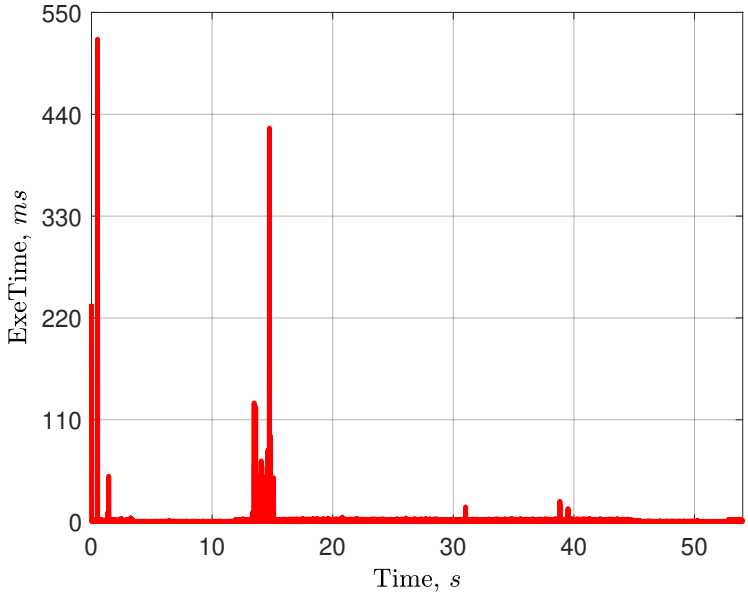

Figure 5.13: Aerodrome circuit - Computational Time in ms.

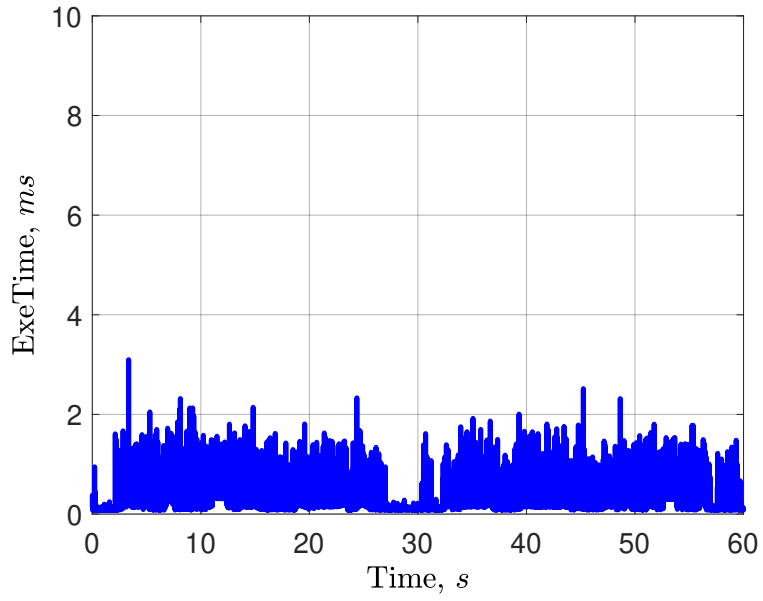

Figure 5.14: Roller coaster - Computational Time in ms.

5.5 Summary

This chapter examined the stability of the qLMPC path-following framework through two distinct approaches. The first approach incorporated terminal in-

gredients to guarantee stability. However, it presented two main drawbacks: the complexity arising from solving LMIs to derive the terminal components, and the increased computational load, which could limit real-time applicability. To address these limitations, a second approach is adopted, leveraging the benefits of velocity-based linearization. This method introduces only two additional constraints into the OCP, significantly simplifying the overall formulation. The simplicity and effectiveness of the proposed method are demonstrated, highlighting its clear advantages over the terminal-ingredient-based approach. Simulation results validate the successful path-following performance while ensuring compliance with the prescribed stability conditions.

Chapter 6

Conclusions and Outlook

6.1 Conclusions

This thesis presented a predictive path-following control framework based on a quasi-Linear Parameter-Varying (qLPV) modeling of the position dynamics of fixed-wing aircraft. The proposed approach aims to strike a balance between nonlinear Model Predictive Control (MPC) strategies—known for high prediction accuracy at the expense of computational demand—and linear MPC strategies, which offer computational efficiency but often lack modeling fidelity and performance.

The methodology leverages velocity-based linearization to derive a parameter-dependent linear representation of the inherently nonlinear position dynamics. This qLPV model is then embedded within an MPC framework, resulting in a constrained Quadratic Program (QP), which is solved iteratively to ensure convergence.

The proposed control framework was validated using the ULTRA-Extra aerobatic aircraft, developed at the Institute of Aircraft Engineering, Hamburg University of Technology. The framework explicitly incorporates aircraft constraints into the Optimal Control Problem (OCP), ensuring its relevance to real-world experimental conditions. Furthermore, the use of velocity-based linearization enables offset-free MPC, allowing the aircraft to reject wind disturbances commonly encountered in practical flight scenarios.

The versatility of the qLMPC framework extends beyond the path-following problem, as demonstrated by its application to obstacle avoidance. The framework maintains adherence to the designated path while dynamically generating control commands to safely navigate around encountered obstacles. Once past the obstacle, the controller minimizes tracking errors and realigns the aircraft with the original path.

This task is particularly challenging due to the nonlinear nature of obstacle con-

straints. To address this, the constraints were linearized. However, this linearization alone was insufficient for reliable obstacle avoidance, as the resulting constraints depend not only on obstacle positions but also on the current position of the aircraft. Even for static obstacles, the aircraft's position changes over time, necessitating the update of obstacle constraints at each sampling time. Thus, these constraints are more appropriately treated as qLPV constraints rather than purely linear ones. Encouragingly, they can still be incorporated within the QP optimization framework.

Another major contribution of this thesis is the stability analysis of the proposed control scheme. The objective of integrating stability considerations into predictive control is to ensure boundedness of the system states at the end of the prediction horizon. Two different strategies were employed to achieve this. The first involved using terminal ingredients, but it presented two main challenges: (1) the complexity of solving Linear Matrix Inequalities (LMIs) to compute the terminal set, and (2) the increased computational burden, particularly for velocity-based models due to their higher system order.

To address these limitations, a method from the literature tailored for velocity-based models was adopted and adapted to the proposed framework. This method introduces only two additional constraints into the OCP, guaranteeing the existence of a feasible steady-state solution at the prediction horizon's end.

It is worth noting that this work assumed convergence of the algorithm to an optimal solution. However, recent research by [Hespe and Werner, 2021] has shown that the qLMPC algorithm may converge to suboptimal solutions in some cases. Their proposed solution addresses this issue for unconstrained problems, but it cannot be directly applied to path-following scenarios due to essential system constraints. Nevertheless, this issue warrants consideration in future research.

To support the controller's evaluation, two path-generation techniques were implemented. One, previously used in computer animation, and the other, adapted from a pure pursuit guidance method. The goal was to generate continuous and sufficiently differentiable paths based on a set of predefined waypoints, suitable for tracking by a fixed-wing aircraft. Both methods successfully produced continuous and trackable paths.

Simulations were conducted in stages. Initially, the framework was tested on a 3D kinematic UAV model to verify its conceptual validity. It was then extended to a high-fidelity ULTRA-Extra UAV model, which required further controller tuning. This model captures full aircraft dynamics, including wind disturbances and actuator behaviors, but excludes sensor dynamics. Finally, obstacle avoidance scenarios were introduced to evaluate the framework's performance under more realistic operational constraints.

6.2 Outlook

This research has identified several areas for further exploration, which can be grouped into three main categories: modeling enhancements, control design improvements, and preparation for real-world experimentation.

1. **Modeling approaches:** This work utilized a qLPV representation of the kinematic model based on velocity-based linearization of the position dynamics. An alternative research direction could involve deriving a qLPV state-space representation directly from the aircraft kinematics. This might lead to a more computationally efficient incorporation of terminal ingredients for stability, as it avoids the increased complexity associated with the higher-order structure of velocity-based models.

2. **Control approaches:** A successive linearization control (SLC)-based low-level controller was used, relying on linearizing aircraft dynamics around multiple operating points. Future work could explore using velocity-based linearization in the inner control loop, which may offer more efficiency than conventional gain scheduling. Additionally, further investigation into the convergence behavior of the qLMPC algorithm is necessary to ensure convergence to an optimal, rather than suboptimal, solution.

   For obstacle avoidance, integrating dynamic obstacles into the framework is a promising direction. This could be achieved by incorporating onboard sensors that continuously monitor obstacle proximity. Based on this data, the path-following controller could compute necessary commands for safe navigation. Such capabilities would also benefit multi-agent systems, enabling efficient collision avoidance.

   Furthermore, in this thesis, system derivative bounds for stability using velocity-based models were selected heuristically. Future research could involve deriving these bounds by solving a simplified LMI formulation, which would provide a more rigorous theoretical foundation. This may result in less conservative constraints and improve optimization flexibility and system performance.

3. **Flight experiments:** A logical next step is to conduct Hardware-in-the-Loop (HIL) simulations to validate the consistency of performance between the simulation environment and actual hardware. These trials would lay the groundwork for planning and conducting real flight experiments.

Beyond these avenues, the qLMPC framework has the potential to be extended to multi-agent systems (MAS), where it could be applied to problems such as formation control and flocking by incorporating appropriate inter-agent constraints into the optimization problem.

Appendix A

Nomenclature

Abbreviations

ASV	Autonomous Surface Vehicle
AUV	Autonomous Underwater Vehicle
CMG	Control Moment Gyroscope
ECEF	earth-centered, earth-fixed
ET	Execution Time
GNC	Guidance, Navigation and Control
GPS	Global Positioning System
LMI	Linear Matrix Inequality
LQR	Linear Quadratic Regulator
LOS	Line-of-Sight
LPV	Linear-Parameter-Varying
LTI	Linear-time Invariant
MPC	Model-based Predictive Control
NGDPFG	Nonlinear geometric differential path following guidance
NLGL	Nonlinear guidance law
NMPC	Nonlinear MPC
OCP	Optimal Control Problem
PID	Proportional-Integral-Derivative
qLMPC	quasi-Linear-Parameter-Varying-Model Predictive Control
qLPV	quasi-Linear-Parameter-Varying
QP	Quadratic Program
RHC	Receding Horizon Control
SISO	Single-input-single-output
SLC	Successive-loop-closure
SQP	Sequential Quadratic Programming
SW	Synthetic waypoint
SWG	Synthetic waypoint guidance

UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
ULTRA	Unmanned low-cost testing and research aircraft
VF	Vector field
VTP	Virtual Tracking Point

Symbols

A, B, C, D	state space matrices
g	gravitational acceleration, m/s^2, gradient
h	altitude, m
H_s	Hessian matrix
J	cost function
n_z	normal load factor [dimensionless]
p	roll rate, rad/s
Q, R, T	weighting matrices
u	control input
V_a	air speed, m/s
V_g	ground speed, m/s
V_s	synthetic waypoint speed, m/s
V_w	wind speed, m/s
w	climb rate, m/s
x	system states
x, y, z	position components in ECEF coordinate system, m
X_s, Y_s, Z_s	moving synthetic waypoint position components, m
δ_a	aileron command, rad
δ_e	elevator command, rad
δ_r	rudder command, rad
δ_t	throttle command
β	side slip angle, rad
χ	course angle, rad
ϕ	bank angle, rad
γ	flight-path angle, rad
ψ	heading angle, rad
θ	pitch angle, rad - arc length parameter, m
α	angle of attack, rad
ρ	scheduling parameters vector

Appendix B

Additional Material

Linear-Parameter Varying Systems

Commonly in the literature [Slotine, 2002a; Khalil, 2002b], a class of nonlinear time-invariant systems is described in the following form:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= g(x(t), u(t)), \end{aligned} \tag{B.1}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$ and $y \in \mathbb{R}^{n_y}$. The functions $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ are continuously differentiable functions and satisfy a Lipschitz condition. If the system described by (B.1) can be represented as:

$$\begin{aligned} \dot{x}(t) &= A(\rho(t))x(t) + B(\rho(t))u(t) \\ y(t) &= C(\rho(t))x(t) + D(\rho(t))u(t), \end{aligned} \tag{B.2}$$

where $\rho(t)$ is a time-varying vector of scheduling variables that may depend on $x(t)$ or $u(t)$, and $A(\cdot), B(\cdot), C(\cdot)$, and $D(\cdot)$ are continuous functions of ρ, then the system (B.2) is referred to as a quasi-linear-parameter-varying system (qLPV) [Shamma and Athans, 1990; Hoffmann and Werner, 2015]. This modeling approach is particularly advantageous for representing the nonlinear dynamics of systems in an LPV format, enabling the utilization of modern and straightforward linear synthesis techniques for nonlinear design purposes. The scheduling vector ρ is typically considered unknown beforehand but should be measurable online, with its values falling within a known compact set $\mathcal{P} \subset \mathbb{R}^{n_\rho}$. To represent the nonlinear system in a qLPV form, it is essential to allow the scheduling variables to depend on the system states and/or input variables. Consequently, the set $\mathcal{P}$ cannot be presumed known in advance but must be constructed during the analysis of system performance and stability. We define a set:

$$\mathcal{F}_{\mathcal{P}} = \{\rho(t) \in C^1(\mathbb{R}^+, \mathbb{R}^{n_\rho}) | \rho(t) \in \mathcal{P} \ \forall t \succeq 0\}. \tag{B.3}$$

Here, $\mathcal{F}_{\mathcal{P}}$ represents a subset of the signal space comprising admissible continuously differentiable scheduling trajectories. An LPV model is subsequently derived from an initial nonlinear state space model using various techniques. Commonly employed methods for converting a nonlinear system into the LPV form are outlined in e.g. [Kwiatkowski et al., 2006; Lovera et al., 2013; Hossam Abbas, 2010].

Local Modeling Method : Jacobian Linearization

The Jacobian linearization method is commonly used in research literature, *e.g.* in [Reberga et al., 2005], to generate a collection of LTI models by linearizing (B.1) around various equilibrium points using first-order Jacobian linearization. Interpolation between these points yields an LPV model, which is linear with respect to the inputs and states.

$$\begin{aligned} \delta\dot{x} &= A(\rho)\delta x + B(\rho)\delta u \\ \delta y &= C(\rho)\delta x + D(\rho)\delta u. \end{aligned} \tag{B.4}$$

Here, $\delta x = x - x_0$, $\delta u = u - u_0$, and $\delta y = y - y_0$, where x_0, u_0, y_0 denote the steady-state equilibrium points. The resultant model provides a local approximation of the nonlinear plant's dynamics around these equilibrium points. Consequently, the LPV trajectories are constrained to a subset of the actual trajectories of the nonlinear system precisely at equilibrium.

Global Modeling Method

Velocity based linearization: A different approach, proposed by [Leith and Leithead, 1998; Leith and Leithead, 1998b], is velocity-based linearization, offering a direct transformation of systems into a qLPV form. Velocity algorithms find application in various fields, including turbofan engine control [Jonathan Decastro, 2007], unmanned vehicles [Leith and Leithead, 2001], and robotic arms [Cisneros and Werner, 2021], relying on nonlinear MPC.

Compared to Jacobian linearization, velocity-based models offer several advantages. They accurately capture the nonlinear dynamics across all operating points, not just equilibrium points [Kaminer et al., 1995; Halás et al., 2003]. Additionally, they present system modeling in a linear form, dependent on state and input derivatives rather than solely on states and inputs, as in regular state space models. Despite these advantages, velocity-based linearization has limitations, notably yielding higher-dimensional models ($n_x + n_y$) that may lead to increased computational complexity, particularly in systems with numerous outputs. However, as demonstrated by [Rieck et al., 2023], this challenge can be addressed by employing Laguerre functions to reduce system complexity, facilitating the stabilization and control of fixed-wing aircraft attitude. In this method, the LPV model is derived by differentiating (B.4) with respect to time. Consequently, there's no need for perturbation quantities to be constrained near the interpolation point. As a result, the LPV states perfectly match the derivatives of the actual system states as

follows

$$\begin{aligned} \ddot{x} &= A(\rho)\dot{x} + B(\rho)\dot{u} \\ \dot{y} &= C(\rho)\dot{x} + D(\rho)\dot{u}. \end{aligned} \tag{B.5}$$

For this new system we consider an augmented state vector $\xi = [\dot{x}^\top \ y^\top]^\top$, in order to preserve absolute information. A drawback of velocity-based models is the requirement of observers for obtaining the state derivatives.

Another aspect to consider when employing velocity algorithms is the selection of appropriate initial conditions. Failure to do so could lead to errors, as the steady-state information is lost during the differentiation process.

Another approach involves directly using the continuous-time nonlinear model in (B.1) and transforming it into a qLPV form as shown in (B.2) by assigning virtual scheduling variables to each nonlinear function element of the resulting matrix functions $A(\cdot), B(\cdot), C(\cdot)$, and $D(\cdot)$ in (B.2). While this method is ad-hoc in nature, it yields a qLPV representation that precisely captures the nonlinear dynamics of the system. Its effectiveness has also been discussed in the literature, for example, in [Abbas et al., 2009]. However, this method has drawbacks, including non-uniqueness, overbounding, and tendency for the number of scheduling parameters to increase.

Mathematical Preliminaries

Before discussing the details of MPC and the qLMPC algorithm, we first introduce some mathematical tools.

Optimization Problems

Definition B.1. *(Quadratic Program): A quadratic program (QP) is an optimization problem that takes the form*

$$\begin{aligned} \min_x \ & \frac{1}{2}x^\top H x + g^\top x + r \\ \textit{subject to} \quad & Ax \preceq b \\ & A_{eq}x = b_{eq} \end{aligned}$$

This class of optimization problems is extensively used with MPC, particularly in linear MPC, as it naturally arises when solving a quadratic cost function with linear constraints. In this work, we solve QPs in the context of nonlinear systems. Among the most commonly used QP solvers are *quadprog*, *OSQP* [Stellato et al., 2020], *Gurobi* [Gurobi, 2023], *Mosek*, and *CVX* [Grant and Boyd, 2008; CVX, 2012].

Definition B.2. *(Linear Matrix Inequality): A Linear Matrix Inequality (LMI) is an inequality of the form*

$$F(x) = F_0 + \sum_{i=1}^{m} F_i x_i \succ 0,$$

where $x \in \mathbb{R}^m$ is a variable and $F_i = F_i^\top$ are given real matrices.

Velocity-form MPC

Velocity algorithms have found application within the scope of MPC for linear time invariant (LTI) systems [Liuping Wang, 2004]. In the LTI context, deriving a discrete-time velocity model is straightforward. We have

$$x_{k+1} - x_k = Ax_k + Bu_k - (Ax_{k-1} + Bu_{k-1}).$$

Thus we obtain an incremental model that mirrors the identical dynamic model of the non-incremental version as

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k.$$

To preserve absolute information, we again augment the state vector as

$$\begin{aligned} \begin{bmatrix} y_{k+1} \\ \Delta x_{k+1} \end{bmatrix} &= \begin{bmatrix} I & CA \\ 0 & A \end{bmatrix} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix} + \begin{bmatrix} CB \\ B \end{bmatrix} \Delta u_k \\ y_k &= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix}. \end{aligned} \tag{B.6}$$

Model-Based Predictive Control

MPC is a control strategy that relies on a model to predict the future behavior of a system over a given time horizon, called the prediction horizon. Consequently, the control input can be determined by minimising a performance criteria. A general tracking loop is illustrated in Figure B.1. Typically, we consider a nonlinear discrete-time plant model in the following structure:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= h(x_k). \end{aligned} \tag{B.7}$$

Here, $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}^l$, f and h are continuous, and f is continuously differentiable, while satisfying a Lipschitz condition. The core concept of predictive control involves utilizing a predictive model of system dynamics to anticipate future behavior and make control decisions based on predicted future states. An optimization problem is solved to determine the optimal control inputs that minimize a cost function while adhering to system dynamics and constraints. We

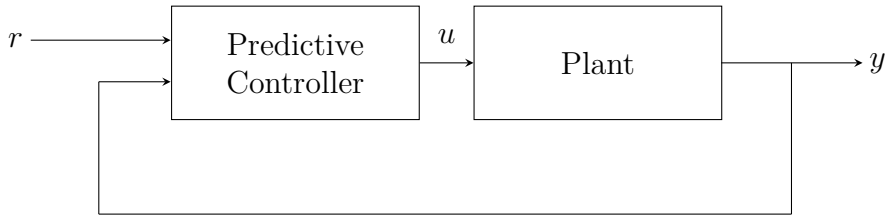

Figure B.1: Predictive control loop.

consider the cost function

$$J_k = \sum_{l=1}^{N-1} (r_{k+l} - y_{k+l})^\top Q(r_{k+l} - y_{k+l}) + (r_{k+N} - y_{k+N})^\top P(r_{k+N} - y_{k+N}) + \sum_{l=0}^{N-1} u_{k+l}^\top R u_{k+l}, \tag{B.8}$$

where P, Q, and R are positive definite weighting matrices, and N represents the prediction horizon. Over the prediction horizon, the first and third terms address the tracking error and control effort, respectively. Meanwhile, the second term introduces an additional penalty, using a distinct weighting matrix P, on the tracking error at the end of the horizon. We introduce the stacked vectors:

$$U_k = \begin{bmatrix} u_k \\ \vdots \\ u_{k+N-1} \end{bmatrix}, \qquad X_k = \begin{bmatrix} x_{k+1} \\ \vdots \\ x_{k+N} \end{bmatrix}, \qquad Y_k = \begin{bmatrix} y_{k+1} \\ \vdots \\ y_{k+N} \end{bmatrix}, \tag{B.9}$$

where U_k, X_k and Y_k are the vectors of future control inputs, states and outputs, respectively.

Constraints

One appealing feature of MPC is its capability to handle constraints on both inuts and states, a common occurence in practical applications. These constraints can be expressed as $u_{k+l} \in \mathcal{U}$ where $\mathcal{U} \subset \mathbb{R}^m$ and $x_{k+l} \in \mathcal{X}$ where $\mathcal{X} \subset \mathbb{R}^n$, such that $\mathcal{U}$ and $\mathcal{X}$ are the sets of admissible inputs and states, respectively. For instance, constraints on the control inputs can be formulated as

$$\underline{u}_i \leq u_i \leq \bar{u}_i.$$

where the inequality is taken elementwise. It is also possible to impose constraints on the rate of change of the control input, states, and outputs, as we will illustrate

in Chapter 3. The optimization problem then takes the form

$$\begin{aligned} &\min_{U_k, X_k} J_k(x_k, R_k, U_k, Y_k) \\ &\text{s.t.} \\ &x_{k+l+1} = f(x_{k+l}, u_{k+l}), \qquad y_k = h(x_k) \\ &x_{k+l+1} \in \mathcal{X} \\ &u_{k+l} \in \mathcal{U}, \qquad l = 0, 1, \cdots, N-1 \end{aligned} \tag{B.10}$$

which is then solved at each time step. The solution is implemented in a receding horizon manner i.e. the first control input is applied, and at the next sampling instant the procedure is repeated.

Coordinate Frames

When studying UAVs, it is essential to comprehend the alignment of bodies in relation to each other to obtain precise measurements and, consequently, accurate control laws. For instance, it is necessary to understand the orientation of the aircraft concerning the Earth. If sensors are present, it is important to know their orientation relative to both the aircraft and the Earth. Similarly, if there are antennas onboard, their orientation relative to the signal source on the ground needs to be understood.

Multiple coordinate systems are necessary because Newton's equations of motion are derived with respect to a fixed-inertial reference frame. However, motion is more effectively described in a body-fixed frame. Additionally, aerodynamic forces and torques act on the aircraft body, making it most convenient to describe them in a body-fixed reference frame. Onboard systems like gyros and accelerometers measure data relative to the body frame, while others measure relative to the inertial frame, such as course angle and ground speed. Regarding our primary research focus, we find that most mission planners and map information are specified in the inertial frame. To understand the dynamic behavior of the UAV, it is essential to define the following coordinate frames: the inertial frame, the vehicle frame, the vehicle-1 frame, the vehicle-2 frame, the body frame, and the wind frame.

The Inertial Frame $\mathcal{F}^i$

The inertial coordinate system typically denotes a reference frame fixed to the Earth's surface and aligned with its inertial space. It is used to describe the UAV's motion relative to the Earth independently of external forces like wind. The origin of this system is often set to coincide with a specific Earth point, like the UAV's takeoff location or a designated reference point, while its axes align with the Earth's inertial surface. Unlike the UAV's motion, the orientation of this system remains fixed relative to the Earth's surface. It serves as a consistent reference frame for detailing the UAV's position and orientation and is vital for

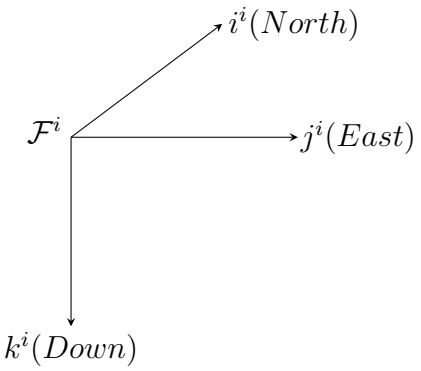

Figure B.2: The inertial coordinate frame.

navigation control and mission planning. It allows operators to designate waypoints, track the UAV's position, and plan trajectories based on Earth's inertial space. Moreover, it aids in integrating sensor data (e.g., GPS, inertial measurement units) and navigation algorithms to precisely determine the UAV's position and velocity relative to the Earth's surface. This coordinate system is often known as the North-East-Down (NED) frame. In this system, north typically corresponds to the inertial x direction, east to the inertial y direction, and down to the inertial z direction.

The Vehicle Frame $\mathcal{F}^v$

The vehicle frame originates from the UAV's center of mass. However, its axes align with those of the inertial frame. Therefore, the unit vector i^v points north, j^v points east, and k^v points towards the center of the Earth.

The Vehicle-1 Frame $\mathcal{F}^{v1}$

The vehicle-1 frame shares its origin with the vehicle frame, located at the aircraft's center of mass. However, $\mathcal{F}^{v1}$ undergoes a rotation in the positive right-handed direction around k^v by the heading angle ψ. Without further rotations, i^{v1} indicates the airframe's nose, j^{v1} indicates the right wing, and k^{v1} aligns with k^v and points towards the Earth. The transformation from $\mathcal{F}^v$ to $\mathcal{F}^{v1}$ is given by

$$p^{v1} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} p^v$$

The Vehicle-2 Frame $\mathcal{F}^{v2}$

The vehicle-2 frame originates once more at the aircraft's center of mass, resulting from a right-handed rotation of the vehicle-1 frame around the j^{v1} axis by the pitch angle θ. Within this frame, i^{v2} refers to the aircraft's nose, j^{v2} represents the right wing direction, and k^{v2} indicates the belly. The transition from $\mathcal{F}^{v1}$ to

$\mathcal{F}^{v2}$ is described by the following transformation:

$$p^{v2} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} p^{v1}$$

The Body Frame $\mathcal{F}^b$

The body frame is achieved through a right-handed rotation of the vehicle-2 frame around i^{v2} by the roll angle ϕ. Thus, the origin remains fixed at the aircraft's center of mass. In this frame, i^b denotes the nose of the airframe, j^b signifies the right wing, and k^b indicates the belly. These directions, represented by the unit vectors i^b, j^b, and k^b, are commonly referred to as the body x, body y, and body z directions, respectively. The transition from $\mathcal{F}^{v2}$ to $\mathcal{F}^b$ is described as follows

$$p^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} p^{v2}$$

Now we can express the transformation from the vehicle frame to the body frame as

$$p^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} p^v$$

The angles ϕ, θ, and ψ are known as Euler angles, a widely used method for describing the orientation of a body in three dimensions. The sequence of rotations $\psi - \theta - \phi$ is particularly common in aviation, although various other Euler angle systems exist. Euler angles offer a straightforward interpretation of spatial orientation, hence their popularity. However, they are susceptible to a mathematical singularity known as *gimbal lock*, occurring when the pitch angle $\theta = \pm 90°$, rendering the yaw angle undefined. To address this issue, quaternion representation is often employed as an alternative. Although less intuitive than Euler angles, quaternions are devoid of mathematical singularities and are more computationally efficient.

The Stability Frame $\mathcal{F}^s$

This frame results from a left-handed rotation of the angle of attack around j^b, causing i^s to align with the projection of V_a onto the plane formed by i^b and k^b. The rationale behind the necessity of a stability frame stems from the generation of aerodynamic forces as the airframe moves through its surrounding air. The requirement for a left-handed rotation arises due to the definition of a positive angle of attack, which is positive for a right-handed rotation from the stability

frame's i^s axis to the body frame's i^b axis. As α involves a left-handed rotation, the transformation from $\mathcal{F}^b$ to $\mathcal{F}^s$ can be expressed as:

$$p^s = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} p^b$$

The Wind Frame $\mathcal{F}^w$

The angle formed between the airspeed vector and the $i^b - k^b$ plane is termed the side-slip angle (β). The wind frame is derived by rotating the stability frame through a right-handed rotation of β around k^s. Consequently, the unit vector i^w aligns with the airspeed vector V_a. The transformation matrix from $\mathcal{F}^s$ to $\mathcal{F}^w$ can be expressed as follows

$$p^w = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} p^s$$

and the total transformation from the body frame to the wind frame can be expressed as

$$p^w = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} p^b$$

Ground speed, Airspeed, and Wind speed

When formulating the dynamic equations of motion for a UAV, it is crucial to recognize that the inertial forces experienced by the UAV depend on velocities and accelerations measured relative to the inertial (fixed) reference frame. However, aerodynamic forces are dependent on the airframe's velocity relative to the surrounding air. In the absence of wind, these velocities align. However, in real-world applications, wind is typically present and must be considered in any relevant controller design, particularly for small UAVs. Thus, it is necessary to distinguish between three velocities: ground speed V_g, airspeed V_a, and wind speed V_w. Ground speed represents velocity relative to the inertial frame, airspeed denotes velocity measured relative to the surrounding air, and wind speed indicates wind velocity relative to the inertial frame. The relationship between these velocities can be expressed as follows

$$V_a = V_g - V_w$$

The UAV velocity V_g can be expressed in the body frame as follows

$$V_g^b = \begin{bmatrix} u_g^b \\ v_g^b \\ w_g^b \end{bmatrix},$$

If we define the north, east, and down components of the wind as w_n, w_e, and w_d respectively, the wind velocity in the body frame can be expressed as

$$V_w^b = \begin{bmatrix} u_w \\ v_w \\ w_w \end{bmatrix} = \mathcal{R}_v^b(\phi, \theta, \psi) \begin{bmatrix} w_n \\ w_e \\ w_d \end{bmatrix}$$

where $\mathcal{R}_v^b(\phi, \theta, \psi)$ is the rotation matrix from the vehicle frame to the body frame. Keep in mind that the airspeed vector V_a represents the UAV velocity relative to the wind, thus it can be formulated in the wind frame as follows

$$V_a^w = \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix}$$

If u_b, v_b, and w_b represent the body-frame components of the airspeed vector, we can express the airspeed vector as follows

$$V_a^b = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} = \begin{bmatrix} u_g^b - u_w \\ v_g^b - v_w \\ w_g^b - w_w \end{bmatrix}$$

When constructing a simulation model for a UAV, we use u_b, v_b, and w_b to denote the aerodynamic forces and moments exerted on the aircraft. The velocity components u_g^b, v_g^b, and w_g^b in the body frame serve as the states of the UAV system and are readily obtained from solving the equations of motion. On the other hand, the wind velocity components u_w, v_w, and w_w are derived from a wind model and serve as inputs to the equations of motion. To sum up our discussion and consolidate the expressions, we can represent the airspeed vector's body-frame components in terms of the airspeed magnitude, side-slip angle (β), and angle-of-attack (α) as follows

$$V_a^b = \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} = \mathcal{R}_w^b \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\beta\cos\alpha & -\sin\beta\cos\alpha & -\sin\alpha \\ \sin\beta & \cos\beta & -\sin\beta\sin\alpha \\ \cos\beta\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix} \quad \text{(B.11)}$$

this suggests that

$$\begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix} = V_a \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}.$$

From the above equation and (B.11), we can conclude that

$$\alpha = \tan^{-1}(\frac{w_b}{u_b}) \qquad \text{and} \qquad \beta = \sin^{-1}(\frac{v_b}{\sqrt{u_b^2 + v_b^2 + w_b^2}}).$$

This is beneficial because aerodynamic forces and moments are typically described in relation to V_a, α, and β, which define the boundaries of the aircraft's flight envelope. This flight envelope encompasses the spectrum of flight conditions within which the aircraft can safely operate.

Bibliography

[Abbas et al., 2009] Abbas, H., S. M. Hashemi, and H. Werner (Oct. 2009). "Decentralized LPV Gain-Scheduled PD Control of a Robotic Manipulator". In: vol. ASME 2009 Dynamic Systems and Control Conference, Volume 2. Dynamic Systems and Control Conference, pp. 801–808. doi: https://doi.org/10.1115/DSCC2009-2651.

[Hossam Abbas, 2010] Abbas, Hossameldin (Jan. 2010). "LPV Modeling, Identification and Low-Complexity Controller Synthesis". PhD thesis.

[Adhikari et al., 2020] Adhikari, Min Prasad and Anton H. J. de Ruiter (2020). "Real-Time Autonomous Obstacle Avoidance for Fixed-Wing UAVs Using a Dynamic Model". In: *Journal of Aerospace Engineering* 33.4, p. 04020027. doi: https://doi.org/10.1061/(ASCE)AS.1943-5525.0001143.

[Aguiar et al., 2008] Aguiar, A. Pedro, João P. Hespanha, and Petar V. Kokotović (2008). "Performance limitations in reference tracking and path following for nonlinear systems". In: *Automatica* 44.3, pp. 598–610. ISSN: 0005-1098. doi: https://doi.org/https://doi.org/10.1016/j.automatica.2007.06.030.

[Alcalá et al., 2019] Alcalá, Eugenio, Vicenç Puig, and Joseba Quevedo (2019). "LPV-MPC Control for Autonomous Vehicles". In: *IFAC-PapersOnLine* 52.28. 3rd IFAC Workshop on Linear Parameter Varying Systems LPVS 2019, pp. 106–113. ISSN: 2405-8963. doi: https://doi.org/https://doi.org/10.1016/j.ifacol.2019.12.356.

[Beard, 2012] Beard, Randal W. (2012). *Small unmanned aircraft. theory and practice*. Princeton University Press, p. 300. ISBN: 9780691149219.

[Bojarski et al., 2016] Bojarski, Mariusz et al. (2016). "End to End Learning for Self-Driving Cars". In: *CoRR* abs/1604.07316. arXiv: 1604.07316. URL: http://arxiv.org/abs/1604.07316.

[Christiansen et al., 2017] Christiansen, Martin Peter, Morten Stigaard Laursen, Rasmus Nyholm Jørgensen, Søren Skovsen, and René Gislum (2017). "Designing and Testing a UAV Mapping System for Agricultural Field Surveying". In: *Sensors* 17.12. ISSN: 1424-8220. doi: https://doi.org/10.3390/s17122703.

[Cisneros et al., 2016] Cisneros, Pablo S. G., Sophia Voss, and Herbert Werner (2016). "Efficient Nonlinear Model Predictive Control via quasi-LPV representation". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. doi: https://doi.org/10.1109/cdc.2016.7798752.

[Cisneros and Werner, 2021] Cisneros, Pablo S. G. and Herbert Werner (2021). "A Velocity Algorithm for Nonlinear Model Predictive Control". In: *IEEE Transactions on Control Systems Technology* 29, pp. 1310–1315.

[Cisneros et al., 2018] Cisneros, Pablo S.G., Aadithyan Sridharan, and Herbert Werner (2018). "Constrained Predictive Control of a Robotic Manipulator using quasi-LPV Representations". In: *IFAC-PapersOnLine* 51.26. 2nd IFAC Workshop on Linear Parameter Varying Systems LPVS 2018, pp. 118–123. ISSN: 2405-8963. doi: https://doi.org/https://doi.org/10.1016/j.ifacol.2018.11.158.

[Pablo Cisneros, 2021] Cisneros, Pablo Sebastian Gonzalez (2021). "Quasi-Linear Model Predictive Control: Stability, Modelling and Implementation". doctoralThesis. Technische Universität Hamburg. ISBN: 978-3-8439-4797-8. doi: https://doi.org/10.15480/882.3574.

[Conte et al., 2004] Conte, Gianpaolo, Simone Duranti, and Torsten Merz (2004). "Dynamic 3D path following for an autonomous helicopter". In: *IFAC Proceedings Volumes* 37.8. IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, 5-7 July 2004, pp. 472–477. ISSN: 1474-6670. doi: https://doi.org/https://doi.org/10.1016/S1474-6670(17)32021-9.

[CVX, 2012] CVX Research, Inc. (Aug. 2012). *CVX: Matlab Software for Disciplined Convex Programming, version 2.0*. https://cvxr.com/cvx.

[Dai and Xie, 2015] Dai, Li and Zheng Xie (2015). "On the Length of Dubins Path with Any Initial and Terminal Configurations". In: *Pure and Applied Mathematics Journal* 4.6, pp. 248–254. doi: https://doi.org/10.11648/j.pamj.20150406.14.

[Jonathan Decastro, 2007] DeCastro, Jonathan A. (2007). "Rate-Based Model Predictive Control of Turbofan Engine Clearance". In: *Journal of Propulsion and Power* 23.4, pp. 804–813. doi: https://doi.org/10.2514/1.25846.

[Moritz et al., 2005] Diehl, Moritz, Hans Georg Bock, and Johannes P. Schlöder (2005). "A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control". In: *SIAM Journal on Control and Optimization* 43.5, pp. 1714–1736. doi: https://doi.org/10.1137/S0363012902400713.

[Egerstedt and Hu, 2001] Egerstedt, Michael and Xiaoming Hu (2001). "Formation constrained multi-agent control". In: *IEEE Transactions on Robotics and Automation* 17.6, pp. 947–951.

[P. Encarnacao and A. Pascoal, 2001] Encarnacao, P. and A. Pascoal (2001). "Combined trajectory tracking and path following: an application to the coordinated control of autonomous marine craft". In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*. Vol. 1, 964–969 vol.1. doi: https://doi.org/10.1109/CDC.2001.980234.

[Englert et al., 2018] Englert, Tobias, Andreas Völz, Felix Mesmer, Sönke Rhein, and Knut Graichen (2018). "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)". In: *Optimization and Engineering*, pp. 1–41.

[Walter R. Evans, 1950] Evans, Walter R. (1950). "Control System Synthesis by Root Locus Method". In: *Transactions of the American Institute of Electrical Engineers* 69.1, pp. 66–69. doi: https://doi.org/10.1109/T-AIEE.1950.5060121.

[Faulwasser et al. 2009] Faulwasser, T., B. Kern, and R. Findeisen (2009). "Model predictive path-following for constrained nonlinear systems". In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE. doi: https://doi.org/10.1109/cdc.2009.5399744.

[Ferramosca et al., 2009] Ferramosca, A., D. Limon, I. Alvarado, T. Alamo, and E.F. Camacho (2009). "MPC for tracking of constrained nonlinear systems". In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 7978–7983. doi: https://doi.org/10.1109/CDC.2009.5400618.

[Fliess et. al, 1995] Fliess, M., J. Levine, P. Martin, and P. Rouchon (1995). "Flatness and defect of nonlinear systems: introductory theory and examples". In: *International Journal of Control* 61.6, pp. 1327–1361.

[Cisneros and Werner, 2020] González Cisneros, Pablo S. and Herbert Werner (2020). "Nonlinear model predictive control for models in quasi-linear parameter varying form". In: *International Journal of Robust and Nonlinear Control* 30.10, pp. 3945–3959. doi: https://doi.org/https://doi.org/10.1002/rnc.4973.

[Grant and Boyd, 2008] Grant, M. and S. Boyd (2008). "Graph implementations for nonsmooth convex programs". In: *Recent Advances in Learning and Control*. Lecture Notes in Control and Information Sciences. http://stanford.edu/~boyd/graph_dcp.html. Springer-Verlag Limited, pp. 95–110.

[Gurobi, 2023] Gurobi Optimization, LLC (2023). *Gurobi Optimizer Reference Manual*. URL: https://www.gurobi.com.

[Halás et al., 2003] Halás, Miroslav, Mikuláš Huba, and Katarína Žáková (2003). "The Exact Velocity Linearization Method". In: *IFAC Proceedings Volumes*

36.18. 2nd IFAC Conference on Control Systems Design (CSD '03), Bratislava, Slovak Republic, 7-10 September 2003, pp. 259–264. ISSN: 1474-6670. doi: https://doi.org/https://doi.org/10.1016/S1474-6670(17)34678-5.

[Hart et al., 1968] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107. doi: https://doi.org/10.1109/TSSC.1968.300136.

[Healey and Lienard, 1993] Healey, A.J. and D. Lienard (1993). "Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles". In: *IEEE Journal of Oceanic Engineering* 18.3, pp. 327–339. doi: https://doi.org/10.1109/JOE.1993.236372.

[Hespe and Werner, 2021] Hespe, Christian and Herbert Werner (2021). "Convergence Properties of Fast quasi-LPV Model Predictive Control". In: *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 3869–3874. doi: https://doi.org/10.1109/CDC45484.2021.9683612.

[Hoffmann and Werner, 2015] Hoffmann, Christian Herzog né and Herbert Werner (2015). "A Survey of Linear Parameter-Varying Control Applications Validated by Experiments or High-Fidelity Simulations". In: *IEEE Transactions on Control Systems Technology* 23, pp. 416–433. URL: https://api.semanticscholar.org/CorpusID:2333805.

[Horowitz, 1963] Horowitz, Isaac M. (1963). "Chapter 7 - Fundamental Properties and Limitations of the Loop Transmission Function". In: *Synthesis of Feedback Systems*. Academic Press, pp. 299–361. ISBN: 978-1-4832-3282-9. doi: https://doi.org/https://doi.org/10.1016/B978-1-4832-3282-9.50010-7.

[Jackson et al., 2008] Jackson, Stephen, John Tisdale, Maryam Kamgarpour, Brandon Basso, and J. Karl Hedrick (2008). "Tracking controllers for small UAVs with wind disturbances: Theory and flight results". In: *2008 47th IEEE Conference on Decision and Control*, pp. 564–569. doi: https://doi.org/10.1109/CDC.2008.4739415.

[Kamel et al., 2017] Kamel, Mahmoud, Timo Stastny, Kostas Alexis, and Roland Siegwart (2017). "Model predictive control for trajectory tracking of unmanned aerial vehicles using ROS". In: *Robot Operating System (ROS), Springer*, pp. 3–39.

[Kaminer et al., 1998] Kaminer, Isaac, Antonio Pascoal, Eric Hallberg, and Carlos Silvestre (1998). "Trajectory Tracking for Autonomous Vehicles: An Integrated Approach to Guidance and Control". In: *Journal of Guidance, Control, and Dynamics* 21.1, pp. 29–38. doi: https://doi.org/10.2514/2.4229. eprint: https://doi.org/10.2514/2.4229.

[Kaminer et al., 1995] Kaminer, Isaac, Antonio M. Pascoal, Pramod P. Khargonekar, and Edward E. Coleman (1995). "A velocity algorithm for the implementation of gain-scheduled controllers". In: *Automatica* 31.8, pp. 1185–1191. ISSN: 0005-1098. doi: https://doi.org/https://doi.org/10.1016/0005-1098(95)00026-S.

[Kamon and Rivlin, 1997] Kamon, I. and E. Rivlin (1997). "Sensory-based motion planning with global proofs". In: *IEEE Transactions on Robotics and Automation* 13.6, pp. 814–822. doi: https://doi.org/10.1109/70.650160.

[Karaman and Frazzoli, 2011] Karaman, Sertac and Emilio Frazzoli (2011). "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7, pp. 846–894.

[Khalil, 2002b] Khalil, H.K. (2002). *Nonlinear Systems*. Pearson Education. Prentice Hall. ISBN: 9780130673893. URL: https://books.google.de/books?id=t_d1QgAACAAJ.

[Kothari et al., 2009] Kothari, Mangal, Da-Wei Gu, and Ian Postlethwaite (2009). "An intelligent suboptimal path planning algorithm using Rapidly-exploring Random Trees". In: *2009 European Control Conference (ECC)*, pp. 677–682. doi: https://doi.org/10.23919/ECC.2009.7074481.

[Krings et al., 2013] Krings, M., B. Annighöfer, and F. Thielecke (2013). "ULTRA - Unmanned Low-cost Testing Research Aircraft". In: *American Control Conference*, pp. 1472–1477.

[Kwiatkowski et al., 2006] Kwiatkowski, Andreas, Marie-Theres Boll, and Herbert Werner (2006). "Automated Generation and Assessment of Affine LPV Models". In: *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 6690–6695. URL: https://api.semanticscholar.org/CorpusID:20876661.

[LaValle, 2006] LaValle, Steven M. (2006). *Planning Algorithms*. Cambridge University Press.

[LaValle et al., 2001] LaValle, Steven M. and Jr. James J. Kuffner (2001). "Randomized Kinodynamic Planning". In: *The International Journal of Robotics Research* 20.5, pp. 378–400. doi: https://doi.org/10.1177/02783640122067453.

[Leith and Leithead, 2001] Leith, D.J. and W.E. Leithead (June 2001). "Gain-scheduled control of a skid-to-turn missile: relaxing slow variation requirements by velocity-based design". English. In: American Control Conference ; Conference date: 25-06-2001 Through 27-06-2001, pp. 500–505. doi: https://doi.org/10.1109/ACC.2001.945594.

[Leith and Leithead, 1998] Leith, Douglas J. and W.E. Leithead (1998). "Comments on Gain Scheduling Dynamic Linear Controllers for a Nonlinear Plant?" In: *Automatica* 34, pp. 1041–1043. URL: https://mural.maynoothuniversity.ie/1836/.

[Leith and Leithead, 1998a] Leith, Douglas J. and William E. Leithead (1998a). "Gain-scheduled and nonlinear systems : dynamic analysis by velocity-based linearization families". In: *International Journal of Control* 70, pp. 289–317.

[Leith and Leithead, 1998b] – (1998b). "Gain-scheduled controller design: An analytic framework directly incorporating non-equilibrium plant dynamics". In: *International Journal of Control*, pp. 249–269. URL: https://api.semanticscholar.org/CorpusID:14721983.

[Li and Tong, 2024] Li, Xiangjie and Yala Tong (2024). "Path Planning of a Mobile Robot Based on the Improved RRT Algorithm". In: *Applied Sciences* 14.1. ISSN: 2076-3417. doi: https://doi.org/10.3390/app14010025.

[Li et al., 2010] Li, Zhen, Jing Sun, and Soryeok Oh (2010). "Handling roll constraints for path following of marine surface vessels using coordinated rudder and propulsion control". In: *Proceedings of the 2010 American Control Conference*, pp. 6010–6015. doi: https://doi.org/10.1109/ACC.2010.5531275.

[Limón et al., 2008] Limón, Daniel, Ignacio Alvarado, Teodoro Alamo, and Eduardo F. Camacho (2008). "MPC for tracking piecewise constant references for constrained linear systems". In: *Automatica* 44.9, pp. 2382–2387. doi: https://doi.org/10.1016/j.automatica.2008.01.023.

[Lin et al., 2019] Lin, Zijie, Lina Castano, Edward Mortimer, and Huan Xu (June 2019). "Fast 3D Collision Avoidance Algorithm for Fixed Wing UAS". In: *Journal of Intelligent Robotic Systems* 97.3-4, pp. 577–604. doi: https://doi.org/10.1007/s10846-019-01037-7.

[Linchant et al., 2015] Linchant, Julie, Jonathan Lisein, Jean Semeki, Philippe Lejeune, and Cédric Vermeulen (2015). "Are unmanned aircraft systems (UASs) the future of wildlife monitoring? A review of accomplishments and challenges". In: *Mammal Review* 45.4, pp. 239–252. doi: https://doi.org/https://doi.org/10.1111/mam.12046.

[Liu et al., 2017] Liu, Chang, Seungho Lee, Scott Varnhagen, and H. Eric Tseng (2017). "Path planning for autonomous vehicles using model predictive control". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 174–179. URL: https://api.semanticscholar.org/CorpusID:11835385.

[Löfberg, 2001] Löfberg, Johan (2001). "Linear Model Predictive Control Stability and Robustness". PhD thesis.

[Lovera et al., 2013] Lovera, Marco, Marco Bergamasco, and Francesco Casella (2013). "LPV Modelling and Identification: An Overview". In: *Robust Control and Linear Parameter Varying Approaches: Application to Vehicle Dynamics*. Springer Berlin Heidelberg, pp. 3–24. ISBN: 978-3-642-36110-4. doi: https://doi.org/10.1007/978-3-642-36110-4_1.

[Luo et al., 2019] Luo, Chunbo, Wang Miao, Hanif Ullah, Sally McClean, Gerard Parr, and Geyong Min (2019). "Unmanned Aerial Vehicles for Disaster Management". In: *Geological Disaster Monitoring Based on Sensor Networks*. Singapore: Springer Singapore, pp. 83–107. ISBN: 978-981-13-0992-2. doi: https://doi.org/10.1007/978-981-13-0992-2_7.

[Mayne et al., 2000] Mayne, D. Q., J. B. Rawlings, C. V. Rao, and P. O.M. Scokaert (June 2000). "Constrained model predictive control: Stability and optimality". English (US). In: *Automatica* 36.6, pp. 789–814. ISSN: 0005-1098. doi: https://doi.org/10.1016/S0005-1098(99)00214-9.

[Medagoda and Gibbens, 2010] Medagoda, Eran D. B. and Peter W. Gibbens (2010). "Synthetic-Waypoint Guidance Algorithm for Following a Desired Flight Trajectory". In: *Journal of Guidance, Control, and Dynamics* 33.2, pp. 601–606. doi: https://doi.org/10.2514/1.46204.

[Medagoda, 2010] Medagoda, Eran D.B. (2010). "Closed Loop Stability of Synthetic Waypoint Guidance Algorithm". In: *IFAC Proceedings Volumes* 43.15, pp. 75–80. ISSN: 1474-6670. doi: https://doi.org/https://doi.org/10.3182/20100906-5-JP-2022.00014.

[Minguez and Montano, 2004] Minguez, Javier and L. Montano (2004). "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios". In: *IEEE Transactions on Robotics and Automation* 20.1, pp. 45–59. doi: https://doi.org/10.1109/TRA.2003.820849.

[Nelson et al., 2007] Nelson, Derek R., D. Blake Barber, Timothy W. McLain, and Randal W. Beard (2007). "Vector Field Path Following for Miniature Air Vehicles". In: *IEEE Transactions on Robotics* 23.3, pp. 519–529. doi: https://doi.org/10.1109/TRO.2007.898976.

[Oleynikova et al., 2016] Oleynikova, Helen, Zachary Taylor, Roland Siegwart, and Juan Nieto (2016). "Safe Local Exploration for Replanning in Cluttered Unknown Environments for Micro-Aerial Vehicles". In: *IEEE Robotics and Automation Letters* 1.2, pp. 812–819.

[Paparazzi, 2025] Paparazzi UAV (2025). *Paparazzi Autopilot: Open-Source UAV System*. Accessed: March 13, 2025. URL: https://paparazziuav.org.

[Park et al., 2007] Park, Sanghyuk, John Deyst, and Jonathan P. How (2007). "Performance and Lyapunov Stability of a Nonlinear Path Following Guidance Method". In: *Journal of Guidance, Control, and Dynamics* 30.6, pp. 1718–1728. doi: https://doi.org/10.2514/1.28957. eprint: https://doi.org/10.2514/1.28957.

[Piccolo, 2024] Piccolo Autopilot Development Team (2024). *Piccolo Autopilot*. URL: https://github.com/ArduPilot/Piccolo.

[Reberga et al., 2005] Reberga, Luc, Didier Henrion, Jacques Bernussou, and Florian Vary (2005). "LPV Modeling of a Turbofan Engine". In: *IFAC Proceedings Volumes* 38.1. 16th IFAC World Congress, pp. 526–531. ISSN: 1474-6670. doi: https://doi.org/https://doi.org/10.3182/20050703-6-CZ-1902.00488.

[Rhee et al., 2010] Rhee, Ihnseok, Sanghyuk Park, and Chang-Kyung Ryoo (2010). "A tight path following algorithm of an UAS based on PID control". In: *Proceedings of SICE Annual Conference 2010*, pp. 1270–1273.

[Richter et al., 2013] Richter, Charles, Adam Bry, and Nicholas Roy (2013). "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments". In: *ISRR*.

[Rieck et al., 2023] Rieck, Leif, Benjamin Herrmann, Frank Thielecke, and Herbert Werner (2023). "Efficient Quasi-Linear Model Predictive Control of a Flexible Aircraft Based on Laguerre Functions". In: *2023 American Control Conference (ACC)*, pp. 2855–2860. doi: https://doi.org/10.23919/ACC55779.2023.10156329.

[Bartomeu et al., 2020] Rubí, Bartomeu, Ramon Pérez, and Bernardo Morcego (May 2020). "A Survey of Path Following Control Strategies for UAVs Focused on Quadrotors". In: *Journal of Intelligent & Robotic Systems* 98.2, pp. 241–265.

[Rolf Rysdyk, 2012] Rysdyk, Rolf (2012). "UAV Path Following for Constant Line-of-Sight". In: *2nd AIAA "Unmanned Unlimited" Conf. and Workshop & Exhibit*. doi: https://doi.org/10.2514/6.2003-6626.

[Samir et al., 2024b] Samir, Ahmed, Horacio M. Calderón, Herbert Werner, Benjamin Herrmann, Leif Rieck, and Frank Thielecke (2024). "A Velocity qLMPC Algorithm for Path-Following with Obstacle Avoidance for Fixed-Wing UAVs". In: *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 107–112. doi: https://doi.org/10.1109/ICUAS60882.2024.10557102.

[Samir et al., 2024a] Samir, Ahmed, Horacio M. Calderón, Herbert Werner, Benjamin Herrmann, and Frank Thielecke (2024). "Predictive Path Following Control for Fixed Wing UAVs Using the qLMPC Framework in the Presence of Wind Disturbances". In: *AIAA SciTech 2024 Forum*. doi: https://doi.org/10.2514/6.2024-1594.

[Sedlmair et al., 2019] Sedlmair, N., J. Theis, and F. Thielecke (2019). "Design and experimental validation of UAV control laws - 3D spline-path-following and easy-handling remote control". In: *Proceedings of the 5th CEAS Conf. Guid., Navigation, Control*.

[Sedlmair et al., 2020] Sedlmair, Nicolas, Julian Theis, and Frank Thielecke (2020). "Experimental Comparison of Nonlinear Guidance Laws for Unmanned Aircraft". In: *IFAC-PapersOnLine* 53.2. 21st IFAC World Congress, pp. 14805–14810. ISSN: 2405-8963. doi: https://doi.org/https://doi.org/10.1016/j.ifacol.2020.12.1922.

[Shamma and Athans, 1990] Shamma, Jeff S. and Michael Athans (Aug. 1990). "Analysis of Gain Scheduled Control for Nonlinear Plants". English (US). In: *IEEE Transactions on Automatic Control* 35.8. Copyright: Copyright 2015 Elsevier B.V., All rights reserved., pp. 898–907. ISSN: 0018-9286. doi: https://doi.org/10.1109/9.58498.

[da Silva and Estrela, 2010] Silva, Jorge Estrela da and João Borges de Sousa (2010). "A dynamic programming approach for the motion control of autonomous vehicles". In: *49th IEEE Conference on Decision and Control (CDC)*, pp. 6660–6665. doi: https://doi.org/10.1109/CDC.2010.5717937.

[Slotine, 2002a] Slotine, J.J.E. and W. Li (1991). *Applied Nonlinear Control*. Prentice Hall. ISBN: 9780130408907. URL: https://books.google.de/books?id=cwpRAAAAMAAJ.

[Sopasakis et al., 2020] Sopasakis, Pantelis, Emil Fresk, and Panagiotis Patrinos (2020). *OpEn: Code Generation for Embedded Nonconvex Optimization*. arXiv: 2003.00292 [math.OC].

[Stellato et al., 2020] Stellato, Bartolomeo, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd (2020). "OSQP: An Operator Splitting Solver for Quadratic Programs". In: *Mathematical Programming Computation* 12.4, pp. 637–672. doi: https://doi.org/10.1007/s12532-020-00179-2.

[Sujit et al., 2014] Sujit, P.B., Srikanth Saripalli, and Joao Borges Sousa (2014). "Unmanned Aerial Vehicle Path Following: A Survey and Analysis of Algorithms for Fixed-Wing Unmanned Aerial Vehicless". In: *IEEE Control Systems Magazine* 34.1, pp. 42–59. doi: https://doi.org/10.1109/MCS.2013.2287568.

[Sun et al., 2008] Sun, Meirui, Rongming Zhu, and Xueguang Yang (2008). "UAV Path Generation, Path Following and Gimbal Control". In: *2008 IEEE International Conference on Networking, Sensing and Control*, pp. 870–873. doi: https://doi.org/10.1109/ICNSC.2008.4525338.

[Taherian et al., 2021] Taherian, Shayan, Kaushik Halder, Shilp Dixit, and Saber Fallah (2021). "Autonomous Collision Avoidance Using MPC with LQR-Based Weight Transformation". In: *Sensors* 21.13. ISSN: 1424-8220. doi: https://doi.org/10.3390/s21134296.

[Thrun et al., 2005] Thrun, Sebastian, Wolfram Burgard, and Dieter Fox (2005). *Probabilistic robotics*. Cambridge, Mass.: MIT Press.

[Verschueren et al., 2020] Verschueren, Robin, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl (2020). *acados: a modular open-source framework for fast embedded optimal control*. arXiv: 1910.13753 [math.OC].

[Wang et al., 2002] Wang, Biao, Xiangxu Dong, and Ben M. Chen (2010). "Cascaded control of 3D path following for an unmanned helicopter". In: *2010 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 70–75. doi: https://doi.org/10.1109/ICCIS.2010.5518579.

[Wang et al., 2002] Wang, Hongling, Joseph Kearney, and Kendall Atkinson (2002). "Arc-length parameterized spline curves for real-time simulation". In: *Proc. 5th International Conference on Curves and Surfaces*. Vol. 387396.

[Liuping Wang, 2004] Wang, Liuping (Jan. 2004). "A Tutorial on Model Predictive Control: Using a Linear Velocity-Form Model". In: *Developments in Chemical Engineering and Mineral Processing* 12.5–6, pp. 573–614. ISSN: 0969-1855. doi: https://doi.org/10.1002/apj.5500120511.

[Whitney, 1987] Whitney, Donald E. (1987). "Path planning for robots via Bézier curves". In: *IEEE Journal of Robotics and Automation* 3.1, pp. 29–37.

[Willemsen et al., 2003] Willemsen, P., J.K. Kearney, and Hongling Wang (2003). "Ribbon networks for modeling navigable paths of autonomous agents in virtual urban environments". In: *IEEE Virtual Reality, 2003. Proceedings.* Pp. 79–86. doi: https://doi.org/10.1109/VR.2003.1191124.

[Zemouche et al., 2005] Zemouche, A., M. Boutayeb, and G.I. Bara (2005). "Observer Design for Nonlinear Systems: An Approach Based on the Differential Mean Value Theorem." In: *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 6353–6358. doi: https://doi.org/10.1109/CDC.2005.1583180.

[Zhang et al., 2020] Zhang, Shuiqing, Tianye Xu, Hui Cheng, and Fan Liang (2020). "Collision Avoidance of Fixed-Wing UAVs in Dynamic Environments Based on Spline-RRT and Velocity Obstacle". In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 48–58. doi: https://doi.org/10.1109/ICUAS48674.2020.9213934.

List of Publications

1. Ahmed Samir, Horacio M. Calderón, Herbert Werner, Benjamin Herrmann, Frank Thielecke “Predictive Path Following Control for Fixed Wing UAVs Using the qLMPC Framework in the Presence of Wind Disturbances”. In: AIAA SciTech 2024 Forum.
   doi: 10.2514/6.2024-1594

2. A. Samir, H. M. Calderón, H. Werner, B. Herrmann, L. Rieck and F. Thielecke, "A Velocity qLMPC Algorithm for Path-Following with Obstacle Avoidance for Fixed-Wing UAVs," 2024 International Conference on Unmanned Aircraft Systems (ICUAS), Chania - Crete, Greece, 2024, pp. 107-112.
   https://ieeexplore.ieee.org/document/10557102