Algorithms for Generation and Tracking of Fast and Agile Flight Trajectories

by

Ezra Tal

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author ....
Department of Aeronautics and Astronautics
October 31, 2021

Certified by ....
Sertac Karaman
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by ....
Russ Tedrake
Professor of Electrical Engineering and Computer Science
Thesis Committee Member

Certified by ....
Luca Carlone
Associate Professor of Aeronautics and Astronautics
Thesis Committee Member

Accepted by ....
Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Algorithms for Generation and Tracking of Fast and Agile Flight Trajectories

by

Ezra Tal

Submitted to the Department of Aeronautics and Astronautics
on October 31, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

High-speed flight through cluttered environments is essential to many time-sensitive robotics applications. It requires motion planning and flight control algorithms that enable highly accurate maneuvering at the edge of the vehicle's capability. These algorithms must overcome challenges particular to fast and agile flight, such as complex dynamics effects including significant unsteady aerodynamics and challenging conditions like post-stall and uncoordinated flight. We propose trajectory generation and tracking algorithms that address these challenges for a quadcopter aircraft and for a fixed-wing transitioning aircraft that combines vertical take-off and landing (VTOL) with efficient forward flight.

This thesis contains several contributions. First, we show that robust control based on incremental nonlinear dynamic inversion (INDI) enables fast and agile flight without depending on an accurate dynamics model. Based on the INDI technique, we design a comprehensive quadcopter flight control algorithm that achieves accurate trajectory tracking without relying on any vehicle aerodynamics model. Second, we show differential flatness of a global nonlinear six-degree-of-freedom (6DOF) flight dynamics model for a tailsitter flying wing transitioning aircraft. We leverage the flat transform to design an INDI flight control algorithm capable of tracking agile aerobatics maneuvers that exploit the entire flight envelope, including post-stall and sideways knife-edge flight. Third, we present a trajectory generation algorithm that aims to identify the actual dynamic feasibility boundary by efficiently combining analytical, numerical, and experimental evaluations in trajectory optimization. Finally, we demonstrate our contributions in fast and agile flight through elaborate experiments.

Thesis Supervisor: Sertac Karaman
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

Conducting the research presented in this thesis has been incredibly rewarding, exciting, and enjoyable. At the same time, it has also been a challenging experience that I could never have completed without the guidance, encouragement, and support of many.

Before anyone else, I would like to thank my advisor, Sertac Karaman, who provided me with the opportunity to come to MIT and work on this fascinating research. Throughout my time at MIT, Sertac has been a tremendous source of inspiration, advice, and encouragement. I want to thank the other members of my thesis committee as well, Russ Tedrake and Luca Carlone, whose advice has been impactful in both addressing important details and determining the high-level direction of the thesis project. I am also thankful to the thesis readers, Scott Van Broekhoven and Murat Bronz, who have been very generous in sharing their time and knowledge. Conversations with Murat helped shape some of the fundamental ideas in this thesis, and his aircraft designs enabled many of the flight experiments.

The mentorship I received from Nhan Nguyen and Tal Shima before coming to MIT has helped shape me as a researcher and motivated me to pursue doctoral studies. I am thankful for their continuing support. I also thankfully acknowledge the guidance from Qiping Chu, who, among other things, introduced me to incremental flight control.

The Autonomy and Embedded Robotics Accelerated (AERA) group has been a great source of suggestions, feedback, support, friendship, and community. I am thankful to all students and others with whom I have interacted over the years. There are too many to list, but I do want to mention a few specifically. Gilhyun Ryou has been a wonderful collaborator from whom I have learned a lot, and part of our joint work is included in this thesis. Dave McCoy contributed electronics designs that were essential towards enabling the flight experiments and helped me deal with the intricacies of real-time embedded systems programming. The indispensable support by John Aleman came in many forms, including assembly of the flight vehicles and countless repairs to them.

I am also thankful to many friends and colleagues in the broader LIDS, AeroAstro, and MIT community. The friendly environment of 32-D740 always provided plenty of opportunity to talk about research, and (moreso) about anything that had absolutely nothing to do with research. My house mates, with whom I escaped Tang Hall, and my team mates in intramural soccer (and all the other sports I attempted), who provided a welcome reprieve from the stress and busyness of graduate school. Skiing proved another great source of relaxation, so thanks to all who weathered the icy New England slopes with me. I also thank my friends in other parts of the world, who kept in touch, took an interest in my work, and provided welcome distractions.

Ultimately, none of this would have been possible without the support of my family, to whom I owe so much. I am thankful for the encouragement from my sister, brother, sister-in-law, and parents. My father, who taught me how to be patient and persistent and to not give up if things do not work on the first attempt, and my mother, who taught me the importance of taking a step back and keeping the big picture in mind.

Finally, and most importantly, I want to thank Claire. Coming to Boston, meeting you, and having you by my side throughout this adventure have made everything so much more meaningful. You have been a tremendous source of joy, encouragement, and support, and I cannot wait to see what the future holds for us.

Funding support from the Office of Naval Research through grant N000141712670 and from the Army Research Office through grant W911NF1910322 is thankfully acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Autonomous high-speed navigation through cluttered environments has been a focus of academic robotics research in recent decades. The problem is challenging and broad as it incorporates diverse facets of robotics, such as vehicle and electronics design, motion planning and control, and perception and estimation. The development of comprehensive approaches for fast and agile flight that are capable of striking demonstrations has been inspired and popularized through both research programs and challenges, such as AlphaPilot and the autonomous drone racing competition at IROS [35, 84, 91].

In addition to being of academic interest, the problem also connects to many real-world applications in which robotic vehicles incur similar challenges. In search and rescue operations, autonomous aircraft have the potential to quickly cover large, remote areas. The vehicles must be able to reach, search, and exit unknown environments quickly and safely [128]. These environments may be increasingly adversarial, e.g., due to strong winds or dust clouds, and may include indoor locations where global positioning is not available. This requires planning and control systems that are suitable for high-speed and agile movements, but also robust against external disturbances and estimation errors. Safety is an important aspect, because humans are present in the operational area and because the risk of damage to the vehicle is often considered in deployment decisions [86]. Another compelling example is autonomous collision avoidance, e.g., in automotive applications. Once perception systems detect that a conflict may occur, preventative control action, e.g., braking and swerving, must be applied accurately and instantaneously. This requires control systems that respond quickly and maneuver the vehicle outside normal operating conditions, at the boundary of the safe configuration space [131].

The maturation of technology that enables unmanned vehicles to reliably fly fast and agile trajectories also has the potential to offer novel commercial opportunities. Partially autonomous rotorcraft capable of performing tasks such as videography and infrastructure inspection are already marketed towards both consumers and public and private sector entities[1]. In the near future, this technology may become more widely applied, e.g., towards time-sensitive deliveries and fast response to medical emergencies [109, 138].

A large portion of academic research on fast and agile flight focuses on rotorcraft and, in particular, on multicopters, like the quadcopter shown in Fig. 1-1. Their relatively simple and compact design, ease of use, and capability to take-off and land vertically make

[1]See, e.g., `https://www.skydio.com`.

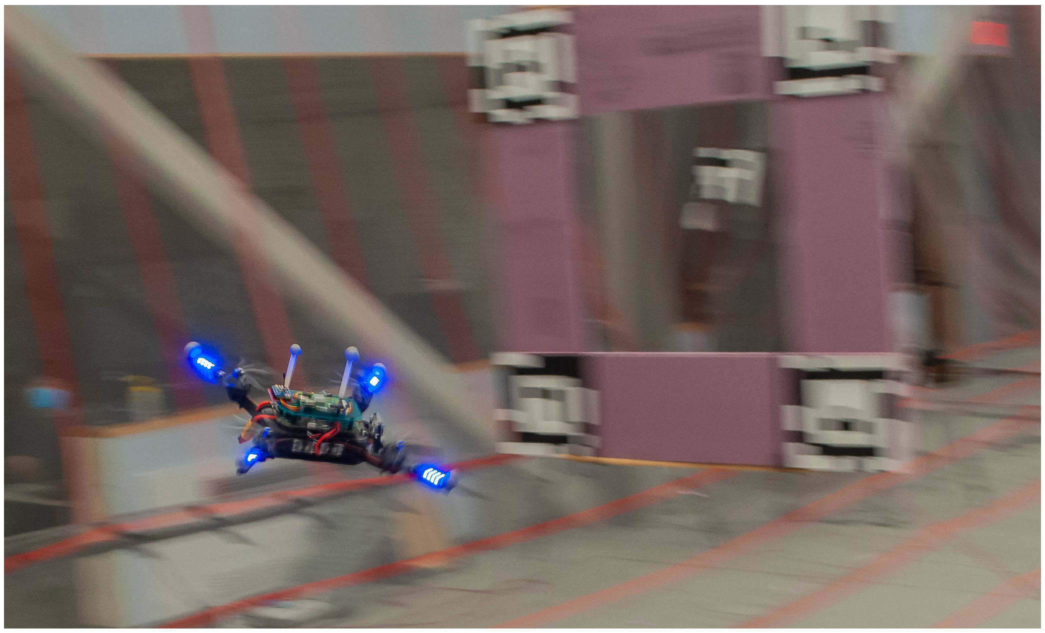


Figure 1-1: Quadcoper aircraft in agile flight.

multicopter aircraft an ideal research platform. When equipped with sufficient thrust-to-weight ratio, multicopters are highly capable of fast and agile flight. This capability is convincingly exhibited in drone racing, where remotely-operated quadcopter aircraft are piloted through challenging, obstacle-rich courses at very high speeds[2]. The use of multicopters also has disadvantages. Most prominently, multicopters have severely limited range and endurance when compared to fixed-wing aircraft, making them less suitable for many real-world applications.

Transitioning aircraft have the potential to combine the vertical take-off and landing (VTOL) and hover capabilities of rotorcraft with the speed and endurance of fixed-wing aircraft. Specifically, tailsitter aircraft take off in vertical configuration with their rotors pointed upward and change their attitude to transition to horizontal flight where the rotors provide propulsion and lift is provided by the wing. Tailsitter aircraft can exceed the range and endurance limitations typical of multicopters, without sacrificing the capability to take-off, hover, and land in confined spaces. This combination is relevant to many applications. For example, in search and rescue, unmanned tailsitter aircraft could quickly reach remote locations using horizontal flight, and inspect structures or enter buildings in hovering flight.

When considering manned flight, it is well known that fixed-wing aircraft with large control surfaces and sufficient thrust are capable of performing high-speed and agile maneuvers, e.g., in air shows, air racing, and close-range aerial battle. On the other hand, unmanned fixed-wing aircraft are not necessarily known for their maneuverability, and research in this direction is not nearly as expansive as for unmanned rotorcraft. However, when typically impermissible flight conditions, such as uncoordinated and post-stall flight, are considered, increasingly agile maneuvers become feasible. This is convincingly demonstrated in 3D aerobatics, where remotely-piloted fixed-wing aircraft put on dazzling performances that fully exploit the flight envelope, including the aforementioned unconventional flight conditions [3].

[2]See, e.g., `https://www.youtube.com/thedroneracingleague` for impressive demonstrations.
[3]An inspiring example can be found at `https://youtu.be/MNZPiDCmV9g`.

1.2 Problem Statement

In this thesis, we focus on generation and tracking of fast and agile flight trajectories for both quadcopter and fixed-wing VTOL aircraft. We aim to achieve accurate and reliable flight at the dynamic feasibility boundary, i.e., at the boundary of what the vehicle is physically capable of. Specifically, we are interested in the following time-optimal **trajectory generation** problem:

$$\begin{aligned} \underset{\boldsymbol{\sigma},\, T}{\text{minimize}} \quad & T \\ \text{subject to} \quad & \boldsymbol{\sigma}(0) = \boldsymbol{\sigma}^{\text{start}}, \\ & \boldsymbol{\sigma}(T) = \boldsymbol{\sigma}^{\text{end}}, \\ & \boldsymbol{\sigma} \in \mathcal{F}, \\ & \boldsymbol{\sigma} \in \Sigma_T. \end{aligned} \tag{1.1}$$

The trajectory

$$\boldsymbol{\sigma}(t) = [\mathbf{x}(t)^\top \; \psi(t)]^\top, \tag{1.2}$$

maps time to the three-dimensional position and yaw angle of the vehicle with regard to the world-fixed reference frame. This four-element trajectory (along with a finite number of its derivatives) is known to fully define the state and control input trajectories of a quadcopter. In Chapter 3, we will show that this is also the case for the tailsitter flying wing aircraft. The constraints of (1.1) prescribe the initial and final position and yaw angle. The set $\mathcal{F}$ represents geometrical constraints, e.g., the free space between obstacles or a sequence of waypoints that must be attained. Finally, the function space Σ_T is the set of all dynamically feasible trajectories, i.e., all trajectory functions that the aircraft can accurately track over $[0, T]$. The **trajectory tracking** problem entails following $\boldsymbol{\sigma}(t)$ as closely as possible, i.e., maintaining minimal position and yaw error with regard to this reference trajectory.

The high-speed, agile flight regime presents significant challenges that need to be addressed in both trajectory generation and tracking. For example, significant aerodynamic forces and momenta act on the vehicle, leading to both steady-state effects, such as drag, and unsteady effects, such as wake turbulence. In contrast, quadrotor aerodynamics can typically be neglected completely in hover and low-speed flight. Similarly, challenging conditions like stall and uncoordinated flight are typically not visited by fixed-wing aircraft, but may be essential towards fast and agile maneuvering.

Furthermore, actuation delay and bandwidth limitations, such as the mechanical time constant of the motors, may prove to be a limiting factor in the application of large and fast-changing control action required for agile maneuvering. In extreme cases, battery thermochemical dynamics may also play a role, as internal resistance causes the voltage to drop under large currents. Consequently, very high motor speeds can only be maintained for limited consecutive time periods. In addition to purely physical limits, we need to account for the software and hardware limitations of other subsystems, such as state estimation. At high speeds and accelerations, estimation performance may degrade due to phenomena like motion blur and increased vibrations. Communication and computation delays also become more significant, e.g., a 30 ms delay at 2 m/s corresponds to 6 cm traveled, while the same delay at 15 m/s corresponds to 45 cm.

The resulting flight dynamics are highly complex, and their feasibility boundary cannot conveniently be expressed, e.g., as state or input limits. Consequently, a trajectory generation algorithm that operates at the feasibility boundary must also be able to identify and enforce this boundary. When it comes to trajectory tracking, an effective flight control algorithm must either incorporate an accurate dynamics model—which may be challenging to obtain—or be robust against model discrepancies.

In this thesis, we address these challenges pertaining to generation and tracking of fast and agile flight trajectories through the following three research questions:

1. How can we find time-optimal trajectories on the boundary of the feasibility set?
2. How can we address the complex dynamics of fast and agile flight without relying on extensive flight dynamics modeling?
3. How can we generate and track fast and agile maneuvers that exploit the entire flight envelope of a VTOL fixed-wing aircraft?

1.3 Related Work

There exists a large body of academic work related to the subjects of this thesis. In this section, we give a brief, unified overview of main areas of related work to enable the reader to place the contributions listed in Section 1.4 into context. This overview also serves to direct the reader to the relevant chapters where a more comprehensive overview of related works is given.

Trajectory-tracking control for quadcopter aircraft in fast and agile flight has seen increasing interest in recent years. Existing works consider various approaches to address the aerodynamic effects that heavily influence the vehicle dynamics in this flight regime. Control systems may employ an aerodynamics model with parameters based on data obtained in bench tests [48], or in flight experiments [10, 102, 123]. Nonparametric modeling can also be used, e.g., to obtain an estimate of unmodeled forces [120], or to directly obtain feedforward control inputs that counteract these unmodeled forces [108]. Many existing methods have in common that they ultimately rely on data-driven modeling of aerodynamics. This approach may be effective, but has several downsides. Gathering flight data can be laborious, and the resulting controller may not generalize to other vehicles or trajectories. Additionally, it is challenging to accurately predict the effects of complex but significant unsteady aerodynamics and of external disturbances, such as wind and gusts. An alternative to model-based flight control is provided by incremental, or sensor-based, nonlinear dynamic inversion (INDI) [2, 117]. This control method directly incorporates inertial measurements to only rely on local accuracy of the dynamics model, leading to improved robustness against modeling errors and external disturbances [111]. In existing literature, INDI has been applied for robust hovering of a quadcopter [116], but not towards tracking of fast and agile flight trajectories. Further background on quadcopter flight control and INDI is given in Chapter 2.

Differential flatness plays an important role in quadcopter trajectory generation and tracking [80]. This property of a dynamics system implies the existence of a transformation between the system state and control inputs, and a flat output and its derivatives [24, 73]. In practice, this enables quadcopter trajectory planning in the flat output space consisting of the position and yaw angle [81, 101]. It also allows computation of feedforward control inputs corresponding to output derivatives that improve tracking on agile trajectories [20, 22, 23, 104]. Chapter 2 includes a more formal definition of differential flatness, and additional related work on its application in quadcopter trajectory-tracking control. Trajectory generation using differential flatness is described in Chapter 4 and Chapter 5.

When considering transitioning aircraft, flight control is complicated by the change of dynamics between hover and forward flight. Existing flight control designs for tailsitter aircraft address this difficulty in various ways, such as blending of separate controllers [54], gain scheduling [50,68], or pre-planned transition maneuvers [11]. However, a global control design is preferable when performing agile maneuvers at large angle of attack through the transition regime [103]. Flight or wind tunnel data can be used to improve the accuracy of the dynamics model [68, 69, 134]. However, similarly to quadcopter applications, the performance of the resulting controller will depend on the quality and extent of data gathered, and its applicability under different circumstances may be limited. This has prompted recent research into more robust control designs, e.g., based on incremental control [114] or model-free control [4]. More details and relevant work on control of transitioning aircraft are discussed in Chapter 3.

Trajectory generation for multicopter and fixed-wing aircraft is a broad subject, and many approaches exist. Broadly speaking, popular optimization-based algorithms can be divided into two categories. Methods that use differential flatness to transform the optimization problem to the low-dimensional flat output space [8,15,80,101]. These algorithms are typically based on the premise that sufficiently smooth trajectories are dynamically feasible and assess feasibility of candidate solutions decoupled from the actual trajectory optimization. On the other hand, there are methods that incorporate high-fidelity dynamics and feasibility constraints and perform trajectory optimization in the high-dimensional state space [3, 92]. While these methods are capable of obtaining high-quality solutions, they do impose large computational cost. We provide additional background and references on trajectory generation for quadcopters in Chapter 4 and for fixed-wing aircraft in Chapter 5.

Ultimately, the majority of trajectory generation algorithms depend on dynamics models to determine feasibility. The resulting solutions thus depend on the accuracy of these models, which is particularly challenging to assess in fast and agile flight, as described in Section 1.2. Bayesian optimization with Gaussian processes provides a model-free approach to optimization of a black-box objective function [100]. An extended version of the algorithm, called multi-fidelity Bayesian optimization, combines function evaluations from various sources [57, 60]. It has the potential to reduce the required number of expensive high-fidelity function evaluations, e.g., by combining simulation and physical experiments to optimize tuning parameters of robotics controllers [72, 98]. Bayesian optimization and relevant references are further discussed in Chapter 4.

1.4 Contributions

This thesis contains the following main contributions.

(i) We show that robust control based on incremental nonlinear dynamic inversion enables accurate tracking of fast and agile trajectories without relying on extensive flight dynamics modeling.

(ii) Based on (i), we present a comprehensive quadcopter flight control design that achieves accurate tracking of fast and agile maneuvers without depending on a vehicle aerodynamics model.

(iii) We show differential flatness of a global 6DOF flight dynamics model with aerodynamics equations for a tailsitter flying wing transitioning aircraft.

(iv) Based on (i) and (iii), we present a comprehensive algorithm to control a tailsitter flying wing aircraft in agile maneuvers that exploit the entire flight envelope, including post-stall, uncoordinated, and other challenging flight conditions.

(v) We present an algorithm for trajectory generation at the true dynamic feasibility boundary by combining analytical, numerical, and experimental evaluations using multi-fidelity Bayesian optimization.

(vi) Based on (iii), we present a tailsitter flying wing trajectory generation algorithm capable of generating aerobatics maneuvers that exploit the entire flight envelope, including post-stall, uncoordinated, and other challenging flight conditions.

(vii) We present elaborate experimental results to demonstrate our algorithmic contributions in fast and agile flight.

In addition, we provide more detailed statements of contributions for Chapter 2 through Chapter 5 in their respective introduction sections.

1.5 Outline of This Thesis

In this thesis, we address two problems: trajectory generation and trajectory tracking, and we consider two types of vehicles: quadcopter and flying wing tailsitter aircraft. Chapter 2 through Chapter 5 follow this division by problem and vehicle. Our approaches for trajectory-tracking flight control of quadcopter and flying wing aircraft are detailed in Chapter 2 and Chapter 3, respectively. We introduce main nomenclature for each vehicle at the beginning of these chapters. Chapter 2 also contains background on two concepts that are central to the work presented in this thesis, namely differential flatness and incremental nonlinear dynamic inversion. Chapter 4 and Chapter 5 present our work on trajectory generation for quadcopter and flying wing aircraft, respectively. Finally, Chapter 6 presents conclusions and avenues for future work.

Chapter 2

Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness

Autonomous unmanned aerial vehicles (UAVs) that can execute aggressive (i.e., high-speed and high-acceleration) maneuvers have attracted significant attention in the past few years. This chapter focuses on accurate tracking of aggressive quadcopter trajectories. We propose a novel control law for tracking of position and yaw angle and their derivatives of up to fourth order, specifically, velocity, acceleration, jerk, and snap along with yaw rate and yaw acceleration. Jerk and snap are tracked using feedforward inputs for angular rate and angular acceleration based on the differential flatness of the quadcopter dynamics. Snap tracking requires direct control of body torque, which we achieve using closed-loop motor speed control based on measurements from optical encoders attached to the motors. The controller utilizes incremental nonlinear dynamic inversion (INDI) for robust tracking of linear and angular accelerations despite external disturbances, such as aerodynamic drag forces. Hence, prior modeling of aerodynamic effects is not required. We rigorously analyze the proposed control law through response analysis, and we demonstrate it in experiments. The controller enables a quadcopter UAV to track complex 3D trajectories, reaching speeds up to 12.9 m/s and accelerations up to 2.1g, while keeping the root-mean-square tracking error down to 6.6 cm, in a flight volume that is roughly 18 m by 7 m and 3 m tall. We also demonstrate the robustness of the controller by attaching a drag plate to the UAV in flight tests and by pulling on the UAV with a rope during hover.

This chapter is based on [125]. A video of the experiments can be found at `https://youtu.be/K15lNBAKDCs`.

2.1 Introduction

High-speed aerial navigation through complex environments has been a focus of control theory and robotics research for decades. More recently, drone racing, in which remotely-operated rotary-wing aircraft are piloted through challenging, obstacle-rich courses at very high speeds, has further inspired and popularized this research direction. Development of fully-autonomous drone racers requires accurate control of aircraft during aggressive, i.e.,

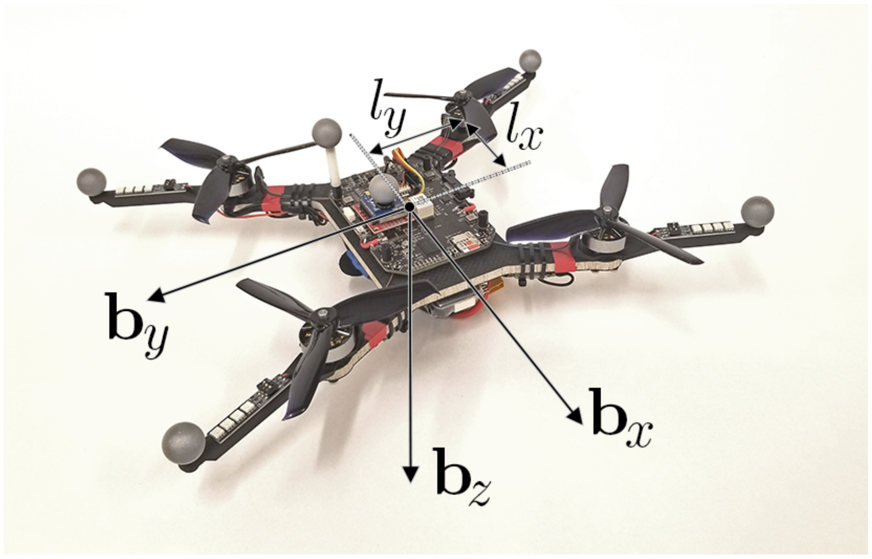


Figure 2-1: Quadrotor with body-fixed reference system and moment arm definitions.

high-speed and agile, maneuvers. At high speeds, aerodynamic drag, which is hard to model, becomes a dominant factor. This poses an important challenge in control design. Additionally, accurate tracking of a reference trajectory with fast-changing acceleration requires considering its higher-order time derivatives, i.e., jerk and snap. In contrast, control design for rotary-wing, vertical take-off and landing (VTOL) aircraft at low speeds typically neglects both aerodynamics and higher-order derivatives.

In this chapter, we propose a novel control design for accurate tracking of aggressive trajectories using a quadcopter aircraft, such as the one shown in Fig. 2-1. The proposed controller generates feedforward control inputs based on differential flatness of the quadcopter dynamics and uses incremental nonlinear dynamic inversion (INDI) to handle external disturbances, such as aerodynamic drag.

Nonlinear dynamic inversion (NDI), also called feedback linearization, enables the use of a linear control law by transforming the nonlinear dynamics into a linear input-output map [49, 106, 113]. Although variants of NDI were quickly developed for flight control [9,21,39,58,118], it is well known that exact dynamic inversion inherently suffers from lack of robustness [61]. As a result, other nonlinear control methods, e.g., adaptive sliding mode [61, 70, 139] and backstepping designs [27], have been considered in order to achieve robustness in flight control. More recently, an incremental version of nonlinear dynamic inversion has been developed [111, 112], based on earlier derivations [2, 117], which incrementally apply control inputs based on inertial measurements. The calculation of these incremental control updates relies only on local accuracy of the dynamics model, resulting in robustness against modeling errors and external disturbances with proof of stability [135]. In existing literature, the INDI technique has been applied to quadcopters for stabilization, e.g., for robust hovering [115, 116], but not for trajectory tracking.

Differential flatness, or feedback linearizability, of a dynamics system allows expressing all state and input variables in terms of a set of flat outputs and its derivatives [24,26,73,77, 133]. In the context of flight control, this property enables reformulation of the trajectory tracking problem as a state tracking problem [58, 74]. Specifically, it enables consideration of higher-order derivatives of the reference trajectory through feedforward state and input references, which has also been applied to control [20, 23, 104]. The quadcopter dynamics have been shown to be differentially flat, and this property has been leveraged for both trajectory generation and tracking [22, 80, 101].

Quadcopter aircraft are relatively easy to maneuver and experiment with. Arguably, these qualities make them ideal for drone racing events. For the same reasons, they have been heavily used as experimental platforms in robotics and control theory research since the start of this century [37, 44, 59, 96]. Complex trajectory tracking control systems have been designed and demonstrated for aircraft in motion capture rooms, where the position and the orientation of the aircraft can be obtained with high accuracy [6, 19, 46, 82, 89, 132, 142]. Agile maneuvers for quadcopter aircraft have also been demonstrated [81, 85]. Despite being impressive, these demonstrations have showcased complex trajectories only at relatively slow speeds, e.g., less than 2 m/s, so that aerodynamic forces and moments may be neglected.

Increasing effort has been devoted towards tracking of agile maneuvers at higher speeds where aerodynamic effects heavily influence the vehicle dynamics and need to be accounted for in control design to achieve accurate trajectory tracking. Parametric models can be used to incorporate aerodynamic force parameters, e.g., based on data obtained from bench tests [48] or flight tests [10, 22, 51, 102, 123]. Alternatively, nonparametric methods can be used, e.g., to experimentally learn a feedforward signal that augments a model-based nominal control input [108], or to predict the unmodeled force based on the vehicle state [120]. We note that approaches that ultimately rely on (parametric or nonparametric) modeling of the aerodynamics may suffer from several disadvantages. Methods based on flight data require data gathering experiments and may produce controllers that do not generalize to vehicles or trajectories other than those used for data gathering. Additionally, external disturbances (e.g., wind and gusts) and unknown changes to the vehicle (e.g., affecting the aerodynamic shape) may lead to degradation in model accuracy. Finally, complex but significant aerodynamic effects, such as aerodynamic interaction between rotors, may be highly challenging to model due to their unsteady behavior [110].

Our proposed control design takes a fundamentally different approach and relies on incremental control to address the aerodynamic force and moment based on inertial measurements. The authors of a recent study [122] compare our controller (with slight modifications) to a state-of-the-art nonlinear model predictive control (NMPC) design [130]. They find that our control design outperforms NMPC with a standard PID inner-loop controller and conclude that robustification using INDI is more effective than drag modeling at reducing tracking error on aggressive trajectories. When NMPC is combined with INDI inner-loop control, it achieves similar tracking performance on dynamically feasible trajectories as our control design. However, NMPC requires several orders of magnitude more computation power, making it impractical for size, weight, power, and cost constrained applications, and does not come with the convergence and stability guarantees provided by differential flatness and INDI [122]. When considering dynamically infeasible trajectories, NMPC is at an advantage due to its capability to address control saturation using online optimization with a receding horizon. This scenario can potentially be addressed by combining our proposed controller with an online trajectory (re)planning algorithm as well.

The main contribution of this chapter is a trajectory-tracking control design that achieves accurate tracking during high-speed and high-acceleration maneuvers without depending on modeling or estimation of aerodynamic drag parameters. The design exploits differential flatness of the quadcopter dynamics to generate feedforward control terms based on the reference trajectory and its derivatives up to fourth order, i.e., velocity, acceleration, jerk, and snap. Modeling inaccuracies and disturbances due to aerodynamic drag are compensated for using incremental control based on the INDI technique. This control design is novel in the following ways. Firstly, the design incorporates direct tracking of reference snap through accurate control of the motor speeds using optical encoders attached to each motor. We recognize that snap is directly related to vehicle angular acceleration and thus to the control torque acting on the quadcopter. Accurate application of torque commands is achieved by precise closed-loop control of the motor speeds using measurements from the optical encoders. To the best of our knowledge, the direct control over snap using motor speed measurements is novel. In contrast, trajectory-tracking control based on body rate inputs, e.g., using a typical inner-loop flight controller, is incapable of truly considering reference snap. Secondly, we develop a novel INDI control design for quadcopter trajectory tracking. Thrust and torque commands are applied incrementally for robustness against significant external disturbances, such as aerodynamic drag, without the need to model or estimate said disturbances. As far as we are aware, the proposed controller is the first design that is tailored for trajectory tracking, as existing INDI flight control designs focus on state regulation, e.g., for maintaining hover under external disturbances. Thirdly, we provide and evaluate a novel implementation of INDI angular acceleration control that includes nonlinear computation of the control increments, as opposed to the existing implementations that use inversion of linearized control effectiveness equations. Finally, we demonstrate the proposed controller in experiments, and we rigorously analyze the benefits of several key aspects through response analysis. In our experiments, the proposed control law enables a unmanned aerial vehicle (UAV) to track complex 3D trajectories, reaching speeds up to 12.9 m/s and accelerations up to 2.1g, while keeping the root-mean-square (RMS) tracking error down to 6.6 cm, in a flight volume that is roughly 18 m long, 7 m wide, and 3 m tall. We also demonstrate the robustness of the controller by attaching a drag plate to the UAV in flight tests and by pulling on the UAV using a tensioned wire during hover. The improved performance due to the tracking of reference jerk and snap through feedforward angular velocity and angular acceleration inputs is also demonstrated, both in theoretical analysis and in experiments.

The chapter is structured as follows: Nomenclature is presented in Table 2.1. In Section 2.2, the quadrotor dynamics model is specified. Section 2.3 gives an introduction on differential flatness and shows how it is used to formulate feedforward control inputs in terms of the reference trajectory. Introductory background on INDI is provided in Section 2.4. In Section 2.5, we describe the architecture of the trajectory-tracking controller, and its individual components. Section 2.6 analyzes the robustness of INDI and the effect of the feedforward control inputs. Finally, we present experimental results from real-life flights in Section 2.7.

2.2 Flight Dynamics Model

We consider a 6 degree-of-freedom (DOF) quadrotor, as shown in Fig. 2-1. The unit vectors depicted in the figure are the basis of the body-fixed reference frame and form the rotation matrix $\mathbf{R} = [\mathbf{b}_x \ \mathbf{b}_y \ \mathbf{b}_z] \in SO(3)$, which gives the transformation from the body-fixed reference frame to the inertial reference frame. The basis of the north-east-down (NED) inertial reference frame consists of the columns of the identity matrix $[\mathbf{i}_x \ \mathbf{i}_y \ \mathbf{i}_z]$.

Table 2.1: Main nomenclature. The subscript ref is used to indicate elements of the reference trajectory function and its time derivatives, as well as feedforward variables directly obtained from the reference trajectory function. The subscript c is used for commanded values that are obtained from a feedback control loop. Low-pass filtered measurements and signals obtained from such measurements are indicated by the subscript lpf.

$\circ$	Hamilton quaternion product
$[\bullet]_\times$	cross-product matrix
$\mathbf{a}$	linear acceleration in inertial frame, m/s^2
$\mathbf{a}^b$	linear acceleration including gravitational acceleration in body-fixed frame, i.e., as measured by IMU, m/s^2
$\mathbf{b}_x$, $\mathbf{b}_y$, $\mathbf{b}_z$	basis vectors of body-fixed frame
$\mathbf{f}_{\text{ext}}$	external disturbance force vector in inertial frame, N
g	gravitational acceleration, m/s^2
$\mathbf{G}_1$	propeller speed control effectiveness matrix
$\mathbf{G}_2$	propeller acceleration control effectiveness matrix
$H(s)$	low-pass filter transfer function
$\mathbf{i}_x$, $\mathbf{i}_y$, $\mathbf{i}_z$	standard basis vectors
$\mathbf{j}$	jerk in inertial frame, m/s^3
$\mathbf{J}$	vehicle moment of inertia matrix, kg·m^2
J_{yy}	vehicle moment of inertia around $\mathbf{b}_y$-axis, kg·m^2
J_{r_z}	motor rotor and propeller moment of inertia, kg·m^2
k_θ, k_q	scalar control gains
k_G	linearized pitch control effectiveness, kg·m^2/(rad·s)
k_{μ_z}	propeller torque coefficient, kg·m^2/rad^2
k_τ	propeller thrust coefficient, kg·m/rad^2
$\mathbf{K}$	diagonal control gain matrices
l_x	moment arm component parallel to $\mathbf{b}_x$-axis, m
l_y	moment arm component parallel to $\mathbf{b}_y$-axis, m
m	vehicle mass, kg
$\mathbf{m}$	control moment vector, N·m
$\mathbf{m}_{\text{ext}}$	external disturbance moment vector, N·m
$M(s)$	motor (control) dynamics transfer function
NI	transfer function corresponding to non-incremental controller
p	polynomial relating motor speeds to throttle inputs
q	vehicle pitch rate around $\mathbf{b}_y$-axis, rad/s
$\mathbf{r}_\psi$	yaw direction vector in inertial frame
$\mathbf{R}$	body-fixed to inertial frame rotation matrix
$\mathbf{s}$	snap in inertial frame, m/s^4
$\mathbf{S}$	angular rate to yaw rate transformation
T	total thrust, N
$\mathbf{v}$	velocity in inertial frame, m/s
$\mathbf{x}$	position in inertial frame, m
α	vehicle pitch acceleration around $\mathbf{b}_y$-axis, rad/s^2
Δ	modeling error parameter
$\boldsymbol{\zeta}$	throttle command vector
θ	vehicle pitch angle, rad
$\boldsymbol{\xi}$	normed quaternion attitude vector
$\boldsymbol{\xi}_c$	incremental command relative to current attitude
$\boldsymbol{\xi}_e$	vector of error angles in body-fixed frame
$\boldsymbol{\sigma}_{\text{ref}}(t)$	reference trajectory function, m, rad
τ	specific thrust, m/s^2
τ_m	motor dynamics time constant, s
ψ	vehicle yaw angle, rad
ω	deviation from hover state motor rotation speed, rad/s
ω_0	hover state motor rotation speed, rad/s
$\boldsymbol{\omega}$	vector of four motor rotation speeds, rad/s
$\boldsymbol{\Omega}$	vehicle angular velocity in body-fixed frame, rad/s

The vehicle translational dynamics are given by

$$\dot{\mathbf{x}} = \mathbf{v}, \tag{2.1}$$

$$\dot{\mathbf{v}} = g\mathbf{i}_z + \tau\mathbf{b}_z + m^{-1}\mathbf{f}_{\text{ext}}, \tag{2.2}$$

where $\mathbf{x}$ and $\mathbf{v}$ are the position and velocity in the inertial reference frame, respectively. Equation (2.2) includes three contributions to the linear acceleration. Firstly, the gravitational acceleration g in downward direction. Secondly, the specific thrust τ, which is the ratio of the total thrust T and the vehicle mass m. Note that the thrust vector is always aligned with the $\mathbf{b}_z$-axis, so that the quadrotor must pitch or roll to accelerate forward, backward or sideways. Finally, the external disturbance force vector $\mathbf{f}_{\text{ext}}$ accounts for all other forces acting on the vehicle, such as aerodynamic drag.

The rotational dynamics are given by

$$\dot{\boldsymbol{\xi}} = \frac{1}{2}\boldsymbol{\xi} \circ \boldsymbol{\Omega}, \tag{2.3}$$

$$\dot{\boldsymbol{\Omega}} = \mathbf{J}^{-1}(\mathbf{m} + \mathbf{m}_{\text{ext}} - \boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega}), \tag{2.4}$$

where $\boldsymbol{\Omega}$ is the angular velocity in the body-fixed reference frame, and $\boldsymbol{\xi} = [\xi^w\ \xi^x\ \xi^y\ \xi^z]^\top$ is the normed quaternion attitude vector, so that $\mathbf{R}\mathbf{x} = \boldsymbol{\xi} \circ \mathbf{x} \circ \boldsymbol{\xi}^{-1}$ with $\circ$ the Hamilton product. Note that a zero magnitude element is implied when multiplying three-element vectors with quaternions. The matrix $\mathbf{J}$ is the vehicle moment of inertia tensor. The control moment vector is indicated by $\mathbf{m}$, and the external disturbance moment vector by $\mathbf{m}_{\text{ext}}$. The third term of (2.4) accounts for the conservation of angular momentum.

Each propeller axis is assumed to be aligned perfectly with the $\mathbf{b}_z$-axis, so that all motor speeds are described by the four-element vector $\boldsymbol{\omega} > 0$. The total thrust T and control moment vector in body-reference frame $\mathbf{m}$ are given by

$$\begin{bmatrix} \mathbf{m} \\ T \end{bmatrix} = \mathbf{G}_1\boldsymbol{\omega}^{\circ 2} + \mathbf{G}_2\dot{\boldsymbol{\omega}}, \tag{2.5}$$

where $^\circ$ indicates the Hadamard power;

$$\mathbf{G}_1 = \begin{bmatrix} l_y k_\tau & -l_y k_\tau & -l_y k_\tau & l_y k_\tau \\ l_x k_\tau & l_x k_\tau & -l_x k_\tau & -l_x k_\tau \\ -k_{\mu_z} & k_{\mu_z} & -k_{\mu_z} & k_{\mu_z} \\ -k_\tau & -k_\tau & -k_\tau & -k_\tau \end{bmatrix}, \tag{2.6}$$

with l_x and l_y the moment arms indicated in Fig. 2-1, k_τ the propeller thrust coefficient, and k_{μ_z} the propeller torque coefficient; and

$$\mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -J_{r_z} & J_{r_z} & -J_{r_z} & J_{r_z} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.7}$$

with J_{r_z} the rotor and propeller moment of inertia. The second term in (2.5) represents the control torque directly due to motor torques. Due to their relatively small moment of inertia, the contribution of the motors to the total vehicle angular momentum may be neglected.

2.3 Differential Flatness

A differentially flat system admits expression of the state and input variables in terms of a, possibly fictitious, output and a finite number of its derivatives [24, 73]. Practically, this implies the existence of a transformation that maps any sufficiently smooth trajectory in the output space to a state and input trajectory that satisfies the system dynamics.

2.3.1 Basics on Differentially Flat Systems

We consider a system of general nonlinear dynamics

$$\dot{\mathbf{X}} = \mathbf{f}\left(\mathbf{X}, \mathbf{u}\right), \tag{2.8}$$

where $\mathbf{X} \in \mathbb{R}^n$ is the state, and $\mathbf{u} \in \mathbb{R}^m$ is the control input. For the system (2.8) to be differentially flat, there must exist an output $\boldsymbol{\sigma}$ of the same dimension as $\mathbf{u}$, such that we can determine all elements of $\mathbf{X}$ and $\mathbf{u}$ from $\boldsymbol{\sigma}$ and a finite number of its derivatives. In other words, if the system (2.8) with state $\mathbf{X} \in \mathbb{R}^n$ and input $\mathbf{u} \in \mathbb{R}^m$ is differentially flat, there exists an output $\boldsymbol{\sigma} \in \mathbb{R}^m$ of the form

$$\boldsymbol{\sigma} = \mathbf{h}\left(\mathbf{X}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \mathbf{u}^{(p)}\right) \tag{2.9}$$

such that

$$\mathbf{X} = \boldsymbol{\alpha}\left(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}, \ldots, \boldsymbol{\sigma}^{(q)}\right), \tag{2.10}$$

$$\mathbf{u} = \boldsymbol{\beta}\left(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}, \ldots, \boldsymbol{\sigma}^{(q+1)}\right). \tag{2.11}$$

The, possibly fictitious, output $\boldsymbol{\sigma}$ is called a flat or linearizing output. There exists a endogeneous (i.e., only depending on the original system state and control input and their derivatives) dynamic feedback, such that the closed-loop system is diffeomorphic to a linear system that may be transformed to Brunovsky canonical form, as

$$\begin{aligned} \sigma_1^{(q_1+1)} &= \nu_1, \\ &\vdots \\ \sigma_m^{(q_m+1)} &= \nu_m, \end{aligned} \tag{2.12}$$

where the subscript indicates individual elements of $\boldsymbol{\sigma}$ and of the linear system input $\boldsymbol{\nu} \in \mathbb{R}^m$. Note that the derivative order may differ between output variables. For convenience of notation we do not write each derivative order specifically, i.e., we denote $\boldsymbol{\sigma}^{(q+1)} = \boldsymbol{\nu}$.

2.3.2 Application to Trajectory Generation and Tracking

The existence of an equivalent linear system can be very useful when considering generation and tracking of trajectories. In general, trajectory generation or motion planning, deals with the task of finding an input trajectory $\mathbf{u}(t)$, such that there exists a state trajectory $\mathbf{X}(t)$

that is a solution of the boundary value problem

$$\mathbf{X}(0) = \mathbf{X}_0, \tag{2.13}$$

$$\mathbf{X}(T) = \mathbf{X}_T, \tag{2.14}$$

subject to (2.8). For flat systems, this problem can be simplified by considering a trajectory $\boldsymbol{\sigma}(t)$ in the output space. The boundary constraints then impose

$$\mathbf{X}_0 = \boldsymbol{\alpha}\left(\boldsymbol{\sigma}(0), \dot{\boldsymbol{\sigma}}(0), \ldots, \boldsymbol{\sigma}^{(q)}(0)\right), \tag{2.15}$$

$$\mathbf{X}_T = \boldsymbol{\alpha}\left(\boldsymbol{\sigma}(T), \dot{\boldsymbol{\sigma}}(T), \ldots, \boldsymbol{\sigma}^{(q)}(T)\right) \tag{2.16}$$

on the initial and final output values. From any trajectory $\boldsymbol{\sigma}(t)$ that satisfies these constraints and is of sufficient smoothness, corresponding state and input trajectories ($\mathbf{X}(t)$ and $\mathbf{u}(t)$, respectively) that provide a solution to the original boundary value problem can be obtained using (2.10) and (2.11). Indeed, flatness can be seen as a nonlinear generalization of the notion of controllability from linear system theory.

In many applications, including typical flight control systems, the control input is constrained, i.e., $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$, leading to complicated state-dependent constraints on $\boldsymbol{\nu}$. A practical way to address these constraints is by searching for a trajectory $\boldsymbol{\sigma}(t)$ that minimizes the magnitude of $\boldsymbol{\nu}$. This approach is the basis of minimum-snap trajectory generation, described in Section 4.2.

In reality, the open-loop control input reference trajectory $\mathbf{u}_{\mathrm{ref}}(t)$ is not sufficient to accurately track the output trajectory reference $\boldsymbol{\sigma}_{\mathrm{ref}}(t)$. Feedback control is needed to counteract disturbances and other deviations. The existence of a linear equivalent system simplifies the design of a stabilizing controller. We define the tracking error

$$\mathbf{e} = \boldsymbol{\sigma} - \boldsymbol{\sigma}_{\mathrm{ref}}, \tag{2.17}$$

and set

$$\mathbf{e}^{(q+1)} = \boldsymbol{\nu} - \boldsymbol{\nu}_{\mathrm{ref}} = -\sum_{i=0}^{q} k_i \mathbf{e}^{(i)} \tag{2.18}$$

with k_i such that the polynomial

$$s^{q+1} + k_q s^q + \cdots + k_1 s + k_0 = 0 \tag{2.19}$$

is Hurwitz stable. The tracking error then obeys exponentially stable dynamics, i.e.,

$$\|\mathbf{e}(t)\| \leq C e^{-\mathbf{A}t}, \tag{2.20}$$

and, by continuity, the state $\mathbf{X}$ converges to $\mathbf{X}_{\mathrm{ref}}$.

2.3.3 Flatness of Quadcopter Dynamics

The quadcopter dynamics model described in Section 2.2 admits the following flat output

$$\boldsymbol{\sigma}_{\mathrm{ref}}(t) = [\mathbf{x}_{\mathrm{ref}}(t)^\top \; \psi_{\mathrm{ref}}(t)]^\top, \tag{2.21}$$

which consists of four output variables: the quadrotor position in the inertial reference frame $\mathbf{x}_{\text{ref}}(t) \in \mathbb{R}^3$, and the vehicle yaw angle $\psi_{\text{ref}}(t) \in \mathbb{T}$, where $\mathbb{T}$ denotes the circle group [80]. The quadcopter trajectory-tracking control problem constitutes accurate tracking of the reference trajectory given by (2.21). Henceforward, we do not explicitly write the time argument t everywhere.

For (2.21) to be dynamically feasible, it is required that $\mathbf{x}_{\text{ref}}$ is of differentiability class C^4, i.e., its first four derivatives exist and are continuous, and that ψ_{ref} is of class C^2. The temporal derivatives of $\mathbf{x}_{\text{ref}}$ are successively the reference velocity $\mathbf{v}_{\text{ref}}$, the reference acceleration $\mathbf{a}_{\text{ref}}$, the reference jerk $\mathbf{j}_{\text{ref}}$, and the reference snap $\mathbf{s}_{\text{ref}}$, all in the inertial reference frame. Similarly, temporal differentiation of ψ_{ref} gives the yaw rate $\dot{\psi}_{\text{ref}}$, and the yaw acceleration $\ddot{\psi}_{\text{ref}}$.

Since the quadcopter dynamics are differentially flat, we can express its state and input as a function of $\boldsymbol{\sigma}_{\text{ref}}(t)$ and its derivatives. This enables reformulation of the trajectory tracking problem as a state tracking problem. In this section, we derive expressions for the angular rate reference $\boldsymbol{\Omega}_{\text{ref}}$, and the angular acceleration reference $\dot{\boldsymbol{\Omega}}_{\text{ref}}$ in terms of trajectory jerk, snap, yaw rate, and yaw acceleration. These references will be applied as feedforward inputs in the trajectory-tracking control design. Our proposed control design also uses flatness to obtain the attitude and motor speed commands, as described in Section 2.5.

Taking the derivative of (2.2) yields the following expression for jerk:

$$\mathbf{j} = \tau \mathbf{R} \left[\mathbf{i}_z\right]_\times^\top \boldsymbol{\Omega} + \dot{\tau}\mathbf{b}_z, \tag{2.22}$$

where $[\bullet]_\times$ indicates the cross-product matrix, and variations in the unmodeled external force $\mathbf{f}_{\text{ext}}$ are neglected. This external force consists chiefly of body drag and rotor drag [45, 76]. Both contributions can be included in the differential flatness transform [22], but the resulting controller will depend on a vehicle-specific aerodynamics model. Instead, we forgo modeling of the external force and use sensor-based control to directly compensate for it. Therefore, our controller is able to handle external disturbances without depending on a vehicle-specific model, as described in the next section.

By taking the derivative once more, the following expression for snap is found:

$$\mathbf{s} = \mathbf{R}\left(\ddot{\tau}\mathbf{i}_z + \left(2\dot{\tau} + \tau\left[\boldsymbol{\Omega}\right]_\times\right)\left[\mathbf{i}_z\right]_\times^\top \boldsymbol{\Omega} + \tau\left[\mathbf{i}_z\right]_\times^\top \dot{\boldsymbol{\Omega}}\right). \tag{2.23}$$

According to typical aerospace convention, we define yaw as the angle between $\mathbf{i}_x$ and the vector

$$\mathbf{r}_\psi = \begin{bmatrix} b_x^1 & b_x^2 & 0 \end{bmatrix}^\top \tag{2.24}$$

with superscripts indicating individual elements of $\mathbf{b}_x$. Taking the derivative of (2.24) using $\dot{\mathbf{R}} = \mathbf{R}[\boldsymbol{\Omega}]_\times$, we obtain the following expression for the yaw rate:

$$\dot{\psi} = \frac{\mathbf{r}_\psi \times \dot{\mathbf{r}}_\psi}{\mathbf{r}_\psi^\top \mathbf{r}_\psi} = \underbrace{\frac{\begin{bmatrix} -b_x^2 & b_x^1 \end{bmatrix}}{\mathbf{r}_\psi^\top \mathbf{r}_\psi}\begin{bmatrix} 0 & -b_z^1 & b_y^1 \\ 0 & -b_z^2 & b_y^2 \end{bmatrix}}_{\mathbf{S}} \boldsymbol{\Omega} = \mathbf{S}\boldsymbol{\Omega}, \tag{2.25}$$

and, by the product rule, the following expression for the yaw acceleration:

$$\ddot{\psi} = \mathbf{S}\dot{\boldsymbol{\Omega}} + \dot{\mathbf{S}}\boldsymbol{\Omega}. \tag{2.26}$$

An expression for the derivative $\dot{\mathbf{S}}$ is omitted here for brevity but can be obtained by applying the product rule to the expression for $\mathbf{S}$ given in (2.25). From (2.22) and (2.25), we obtain the angular rate reference

$$\begin{bmatrix} \mathbf{\Omega}_{\text{ref}} \\ \dot{\tau}_{\text{ref}} \end{bmatrix} = \begin{bmatrix} \tau \mathbf{R}[\mathbf{i}_z]_\times^\top & \mathbf{b}_z \\ \mathbf{S} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{j}_{\text{ref}} \\ \dot{\psi}_{\text{ref}} \end{bmatrix}, \tag{2.27}$$

and from (2.23) and (2.26) the angular acceleration reference

$$\begin{bmatrix} \dot{\mathbf{\Omega}}_{\text{ref}} \\ \ddot{\tau}_{\text{ref}} \end{bmatrix} = \begin{bmatrix} \tau \mathbf{R}[\mathbf{i}_z]_\times^\top & \mathbf{b}_z \\ \mathbf{S} & 0 \end{bmatrix}^{-1} \left(\begin{bmatrix} \mathbf{s}_{\text{ref}} \\ \ddot{\psi}_{\text{ref}} \end{bmatrix} - \begin{bmatrix} \mathbf{R}(2\dot{\tau} + \tau[\mathbf{\Omega}]_\times)[\mathbf{i}_z]_\times^\top \mathbf{\Omega} \\ \dot{\mathbf{S}}\mathbf{\Omega} \end{bmatrix} \right). \tag{2.28}$$

Note that these expressions also contain reference signals for the first and second derivatives of specific thrust. However, as we are unable to command the corresponding first and second derivatives of the motor speed, these references remain unused by the controller.

2.4 Incremental Nonlinear Dynamic Inversion

INDI is a nonlinear control method that reduces dependency on an accurate dynamics model by incrementally updating the control input. In this section, we will derive the INDI control update and compare it to the control law obtained using regular NDI. We also propose a modification of INDI specifically for differentially flat systems.

2.4.1 Nonlinear Dynamic Inversion

Extensive background on NDI can be found in various references, such as [49,106,113]. We show a basic derivation of an NDI control law for a nonlinear input-affine system of the form

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}) + \mathbf{g}(\mathbf{X})\mathbf{u}, \tag{2.29}$$

where $\mathbf{X} \in \mathbb{R}^n$ denotes the state and $\mathbf{u} \in \mathbb{R}^m$ denotes the control input, with the output given by

$$\boldsymbol{\sigma} = \mathbf{h}(\mathbf{X}). \tag{2.30}$$

The relative degree of the system is said to be equal to d if

$$L_{\mathbf{g}} L_{\mathbf{f}}^n \mathbf{h}(\mathbf{X}) = 0 \quad \text{for}\, n \leq d-2, \tag{2.31}$$

$$L_{\mathbf{g}} L_{\mathbf{f}}^{d-1} \mathbf{h}(\mathbf{X}) \neq 0, \tag{2.32}$$

where L denotes the Lie derivative defined as

$$L_{\mathbf{f}} \mathbf{h}(\mathbf{X}) = \frac{\partial \mathbf{h}(\mathbf{X})}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X}), \tag{2.33}$$

$$L_{\mathbf{f}}^n \mathbf{h}(\mathbf{X}) = L_{\mathbf{f}} L_{\mathbf{f}}^{n-1} \mathbf{h}(\mathbf{X}). \tag{2.34}$$

Successive differentiation of (2.30) gives

$$\boldsymbol{\sigma}^{(d)} = L_{\mathbf{f}}^d \mathbf{h}(\mathbf{X}) + L_{\mathbf{g}} L^{d-1} \mathbf{h}(x)\mathbf{u} = \boldsymbol{\nu}, \tag{2.35}$$

akin to (2.12). A stabilizing feedback control can now be obtained by inversion of (2.35), where $\boldsymbol{\nu}$ can be found using the method from Section 2.3.2.

2.4.2 Incremental Control

While NDI may be an effective method for control of nonlinear systems, it can suffer from sensitivity to model uncertainty. When applied to flight control, the model inversion depends on the full dynamics model, including the aerodynamics model. At each control update, the inputs are computed using the inverted model, and discrepancies such as unmodeled external forces or inaccurate parameters will lead to miscalculation. Incremental NDI addresses this lack of robustness by computing updates relative to the currently applied control inputs. The method relies only on the gradient of the dynamics model around the current state and control input and directly incorporates sensor measurements to account for the effect of the currently applied control inputs.

The system is written in incremental form using the first-order Taylor series approximation

$$\begin{aligned}\dot{\mathbf{X}} \approx\, & \underbrace{\mathbf{f}\left(\mathbf{X}_0\right) + \mathbf{g}\left(\mathbf{X}_0, \mathbf{u}_0\right)}_{\dot{\mathbf{X}}_0} + \\ & \underbrace{\frac{\partial}{\partial \mathbf{X}}\left(\mathbf{f}\left(\mathbf{X}\right) + \mathbf{g}\left(\mathbf{X}, \mathbf{u}\right)\right)_{\mathbf{X}=\mathbf{X}_0, \mathbf{u}=\mathbf{u}_0}}_{\mathbf{A}_0}\left(\mathbf{X} - \mathbf{X}_0\right) + \underbrace{\frac{\partial}{\partial \mathbf{u}}\left(\mathbf{g}\left(\mathbf{X}, \mathbf{u}\right)\right)_{\mathbf{X}=\mathbf{X}_0, \mathbf{u}=\mathbf{u}_0}}_{\mathbf{B}_0}\left(\mathbf{u} - \mathbf{u}_0\right), \end{aligned} \tag{2.36}$$

where the subscript 0 denotes the current or recent state and control input. Note that this approximation does not rely on input-affineness. As we will show in Section 2.5 and in Chapter 3, we will apply INDI to control the linear and angular velocity of the aircraft. Due to inertia, these quantities do not instantaneously change in response to changes in respectively the applied force and moment. Hence, we assume $\mathbf{X} = \mathbf{X}_0$ and neglect the corresponding term in (2.36). Since the state is to be controlled, we set

$$\boldsymbol{\sigma} = \mathbf{h}(\mathbf{X}) = \mathbf{X}, \tag{2.37}$$

resulting in

$$\dot{\boldsymbol{\sigma}} = \dot{\mathbf{X}}_0 + \mathbf{B}_0\left(\mathbf{u} - \mathbf{u}_0\right) = \boldsymbol{\nu}. \tag{2.38}$$

The obtained incremental control law

$$\mathbf{u} = \mathbf{u}_0 + \mathbf{B}_0^{-1}\left(\boldsymbol{\nu} - \dot{\mathbf{X}}_0\right) \tag{2.39}$$

does not depend on inversion of the global dynamics model, but instead only relies on the local gradient $\mathbf{B}_0$. Any discrepancy in the model of $\mathbf{f}$ is counteracted, as is any unknown state-dependent disturbance. We rewrite the system dynamics as

$$\begin{aligned}\dot{\mathbf{X}} &= \mathbf{f}(\mathbf{X}) + \mathbf{g}(\mathbf{X}, \mathbf{u}) + \mathbf{d}(\mathbf{X}) \\ &= \tilde{\mathbf{f}}(\mathbf{X}) + \mathbf{g}(\mathbf{X}, \mathbf{u}),\end{aligned} \tag{2.40}$$

where $\mathbf{d}(\mathbf{X})$ represents the modeling discrepancy, consisting of unmodeled dynamics and external disturbances. Since (2.35) depends on the assumed dynamics $\mathbf{f}$, the NDI control law may suffer from decreased performance due to the discrepancy with regard to the true dynamics $\tilde{\mathbf{f}}$. In contrast, (2.39) is not affected as it does not incorporate $\mathbf{f}$.

2.4.3 INDI with Nonlinear Inversion

In particular when tracking fast and agile trajectories, quick and large changes in the control input $\mathbf{u}$ may be required. Unless the system is input-affine, the resulting large $\mathbf{u} - \mathbf{u}_0$ may cause a large discrepancy in the linearized dynamics equation (2.36). In this thesis, we propose incremental controllers for quadcopter and tailsitter flying wing vehicles that avoid any linearization and instead employ nonlinear inversion. As far as we are aware, employing differential flatness to avoid the need to compute any Jacobian in a comprehensive INDI flight control design is novel.

Again assuming $\mathbf{X} = \mathbf{X}_0$, we have

$$\boldsymbol{\nu} - \dot{\mathbf{X}}_0 = \mathbf{f}(\mathbf{X}) + \mathbf{g}(\mathbf{X}, \mathbf{u}) - \mathbf{f}(\mathbf{X}_0) - \mathbf{g}(\mathbf{X}_0, \mathbf{u}_0) = \mathbf{g}(\mathbf{X}_0, \mathbf{u}) - \mathbf{g}(\mathbf{X}_0, \mathbf{u}_0), \tag{2.41}$$

from which we obtain the incremental control law

$$\mathbf{g}(\mathbf{X}_0, \mathbf{u}) = \mathbf{g}(\mathbf{X}_0, \mathbf{u}_0) + \boldsymbol{\nu} - \dot{\mathbf{X}}_0. \tag{2.42}$$

The quantities on the righthand side of (2.42) are known, so what remains is the computation of $\mathbf{u}$ by inversion of $\mathbf{g}$. In general, this function may not be invertible. However, for a differentially flat system we can employ (2.11) to obtain the required inverse as

$$\begin{aligned} \mathbf{u} = \boldsymbol{\beta}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}) &= \boldsymbol{\beta}(\mathbf{X}_0, \mathbf{f}(\mathbf{X}_0) + \mathbf{g}(\mathbf{X}_0, \mathbf{u})) \\ &= \boldsymbol{\beta}\left(\mathbf{X}_0, \mathbf{f}(\mathbf{X}_0) + \mathbf{g}(\mathbf{X}_0, \mathbf{u}_0) + \boldsymbol{\nu} - \dot{\mathbf{X}}_0\right). \end{aligned} \tag{2.43}$$

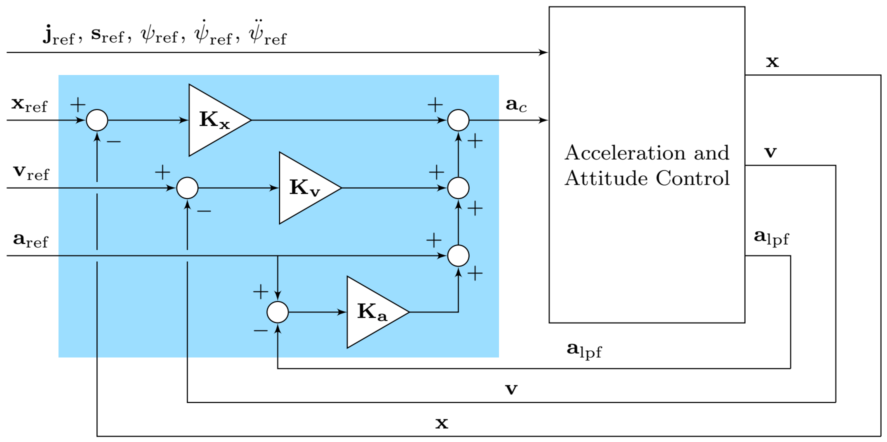
Hence, we obtain an incremental control law with the robustness properties of INDI, but without discrepancies due to linearization.

In this chapter, we apply the incremental control law consisting of (2.42) and (2.43) in a cascaded control design. Specifically, INDI is used for control of both the linear and angular accelerations, where the dynamics (2.40) are given by (2.2) and (2.4), respectively. For the linear acceleration controller, $\mathbf{g}(\mathbf{X}, \mathbf{u})$ represents the specific thrust vector with $\mathbf{u}$ consisting of the thrust magnitude and the vehicle tilt angle (i.e., the direction of the thrust vector); for the angular acceleration controller, $\mathbf{g}(\mathbf{X}, \mathbf{u})$ represents the specific control moment with $\mathbf{u}$ consisting of the motor speeds (which are also in part determined by the thrust magnitude). The corresponding incremental control laws, in the form of (2.42), are given by (2.48) and (2.59). The incremental linear and angular acceleration controllers for the tailsitter aircraft have a similar form and are presented in Chapter 3. In the next section, we present an intuitive derivation of the quadcopter INDI control laws based on their practical working as implicitly estimating and counteracting the disturbance $\mathbf{d}$, i.e., the external force or moment. While this derivation provides a complementary perspective on INDI, the resulting control equations are equivalent to those shown above.

Application of INDI requires measurement or estimation of the (angular) acceleration $\dot{\mathbf{X}}_0$ and the control input $\mathbf{u}_0$, which may require specific attention during the design of the control hardware and software, as described in Section 2.5 and Chapter 3. Finally, we note that the input $\boldsymbol{\nu}$ may contain both feedback and feedforward terms, as described in Section 2.3.2.

Table 2.2: Overview of trajectory tracking controller components.

Component	Methodology	Reference	Output	Description
Position and Velocity Control	PD	$\mathbf{x}_{\text{ref}}$, $\mathbf{v}_{\text{ref}}$, $\mathbf{a}_{\text{ref}}$	$\mathbf{a}_c$	Section 2.5.1
Linear Acceleration and Yaw Control	INDI	$\mathbf{a}_c$, ψ_{ref}	$\boldsymbol{\xi}_c$, T_c	Section 2.5.2
Jerk and Snap Tracking	Diff. Flatness	$\mathbf{j}_{\text{ref}}$, $\mathbf{s}_{\text{ref}}$, $\dot{\psi}_{\text{ref}}$, $\ddot{\psi}_{\text{ref}}$	$\boldsymbol{\Omega}_{\text{ref}}$, $\dot{\boldsymbol{\Omega}}_{\text{ref}}$	Section 2.3.3
Attitude and Angular Rate Control	PD	$\boldsymbol{\xi}_c$, $\boldsymbol{\Omega}_{\text{ref}}$, $\dot{\boldsymbol{\Omega}}_{\text{ref}}$	$\dot{\boldsymbol{\Omega}}_c$	Section 2.5.3
Angular Acceleration Control	INDI	$\dot{\boldsymbol{\Omega}}_c$	$\mathbf{m}_c$	Section 2.5.4
Moment and Thrust Control	Inversion	$\mathbf{m}_c$, T_c	$\boldsymbol{\omega}_c$	Section 2.5.5
Motor Speed Control	Integrative	$\boldsymbol{\omega}_c$	$\boldsymbol{\zeta}$	Section 2.5.5



$\mathbf{j}_{\text{ref}}$, $\mathbf{s}_{\text{ref}}$, ψ_{ref}, $\dot{\psi}_{\text{ref}}$, $\ddot{\psi}_{\text{ref}}$
$\mathbf{x}_{\text{ref}}$
$\mathbf{v}_{\text{ref}}$
$\mathbf{a}_{\text{ref}}$
$\mathbf{K_x}$
$\mathbf{K_v}$
$\mathbf{K_a}$
$\mathbf{a}_c$
Acceleration and Attitude Control
$\mathbf{x}$
$\mathbf{v}$
$\mathbf{a}_{\text{lpf}}$

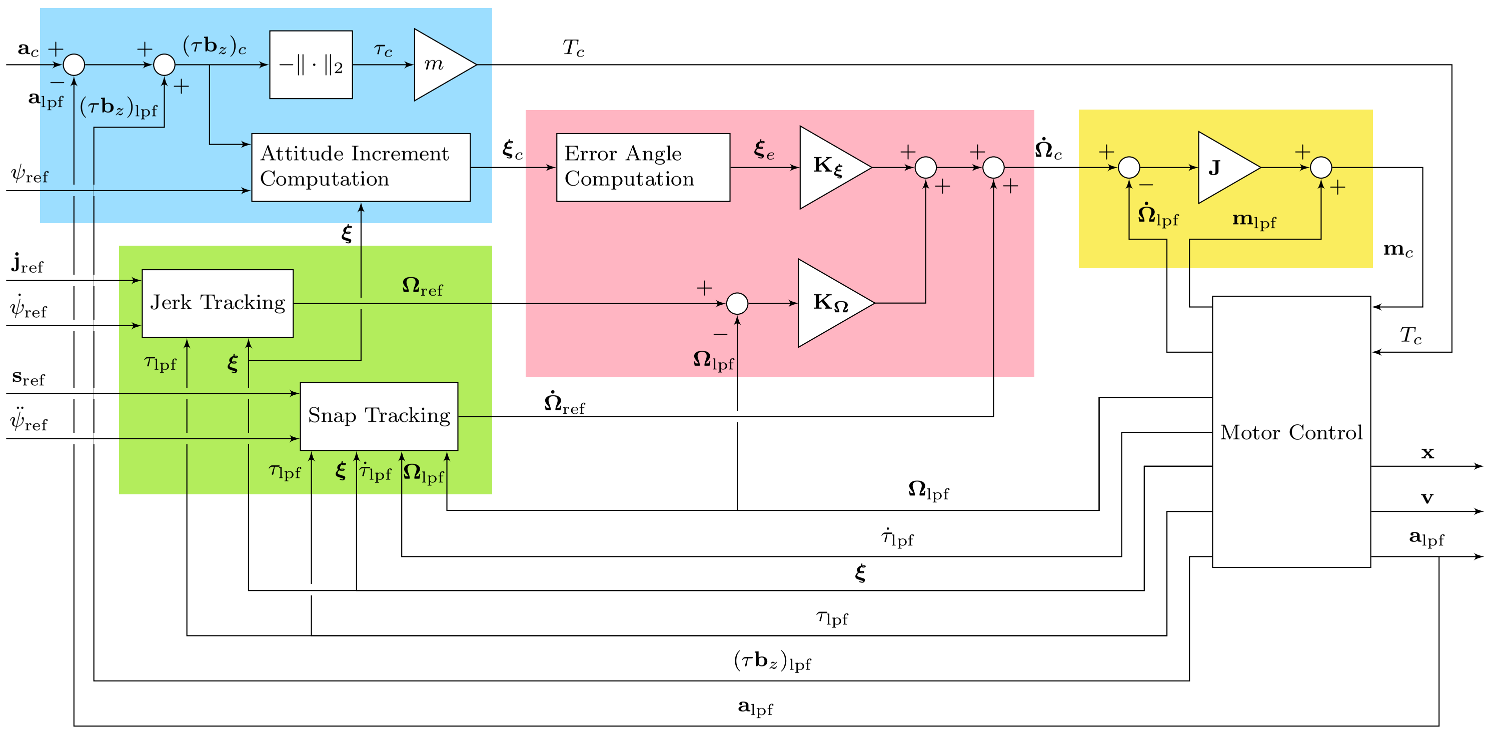
Figure 2-2: Position and velocity control. The blue area contains the PD control design as described in Section 2.5.1.

2.5 Trajectory-Tracking Control

We propose a control design to accurately track the trajectory reference (2.21). Our controller consists of several components based on various methods. Table 2.2 gives an overview of the components with their respective methodology, references, and control outputs. The control architecture is visualized in three block diagrams. Figure 2-2 shows the outer-loop position and velocity controller as described in Section 2.5.1. The intermediate control loop shown in Fig. 2-3 controls linear acceleration, attitude and angular rate, and angular acceleration as described in Section 2.5.2, 2.5.3, and 2.5.4, respectively. Finally, vehicle moment and thrust are directly controlled through closed-loop motor speed control in the inner loop, shown in Fig. 2-4 and described in Section 2.5.5.

The controller utilizes a vehicle state estimate consisting of position, velocity, and attitude. Additionally, motor speed measurements are obtained from optical encoders, and linear acceleration and angular rate measurements are obtained from the inertial measurement unit (IMU). For the application of incremental angular acceleration control, angular acceleration measurements are obtained by numerical differentiation of the measured angular rate. A low-pass filter (LPF) is required to alleviate the effects of noise, e.g., airframe vibrations, on measurements obtained directly from the IMU. We denote the LPF outputs using the subscript lpf, e.g., by $\boldsymbol{\Omega}_{\text{lpf}}$ and $\dot{\boldsymbol{\Omega}}_{\text{lpf}}$ for the angular rate output and its derivative, respectively. The gravity-corrected LPF acceleration output in the inertial reference frame is obtained as follows:

$$\mathbf{a}_{\text{lpf}} = (\mathbf{R}\mathbf{a}^b + g\mathbf{i}_z)_{\text{lpf}}. \tag{2.44}$$



$\mathbf{a}_c$
$\mathbf{a}_{\text{lpf}}$
$(\tau\mathbf{b}_z)_{\text{lpf}}$
$(\tau\mathbf{b}_z)_c$
$-\|\cdot\|_2$
τ_c
m
T_c
Attitude Increment Computation
ψ_{ref}
$\boldsymbol{\xi}_c$
Error Angle Computation
$\boldsymbol{\xi}_e$
$\mathbf{K}_\xi$
$\dot{\boldsymbol{\Omega}}_c$
$\mathbf{J}$
$\dot{\boldsymbol{\Omega}}_{\text{lpf}}$
$\mathbf{m}_{\text{lpf}}$
$\mathbf{m}_c$
$\mathbf{j}_{\text{ref}}$
$\dot{\psi}_{\text{ref}}$
Jerk Tracking
$\boldsymbol{\Omega}_{\text{ref}}$
$\mathbf{K}_\Omega$
$\boldsymbol{\Omega}_{\text{lpf}}$
τ_{lpf}
$\boldsymbol{\xi}$
$\mathbf{s}_{\text{ref}}$
$\ddot{\psi}_{\text{ref}}$
Snap Tracking
$\dot{\boldsymbol{\Omega}}_{\text{ref}}$
$\dot{\tau}_{\text{lpf}}$
Motor Control
$\mathbf{x}$
$\mathbf{v}$

Figure 2-3: Acceleration and attitude control. The blue area contains the INDI linear acceleration and yaw control as described in Section 2.5.2. The green area contains the computation of angular rate and angular acceleration references based on differential flatness as described in Section 2.3.3. The red area contains the attitude and angular rate control as described in Section 2.5.3. The yellow area contains the INDI angular acceleration control as described in Section 2.5.4.

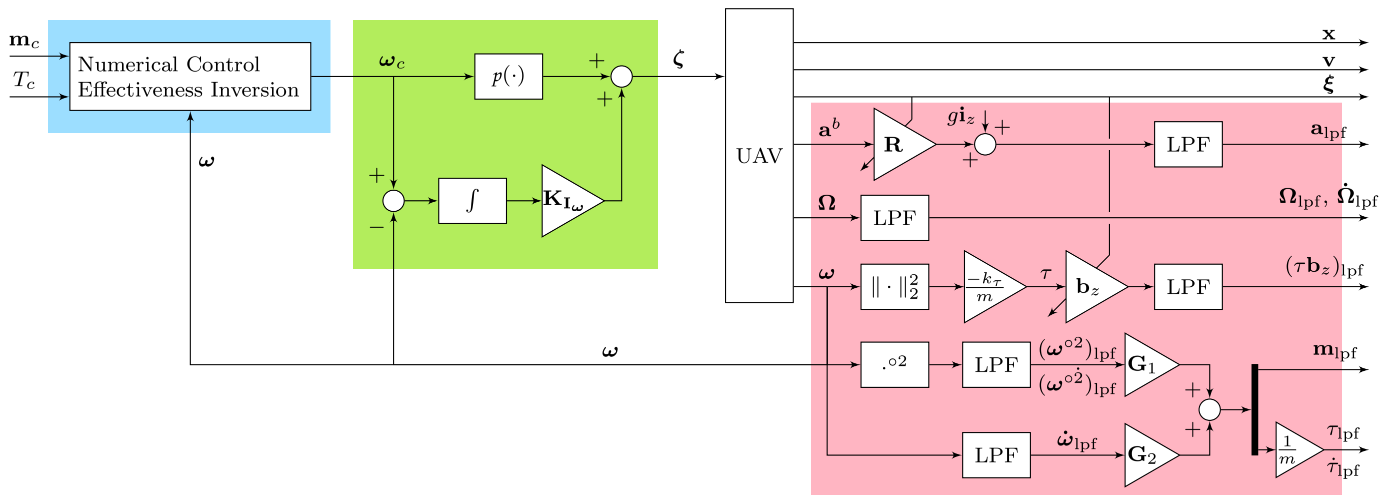

$\mathbf{m}_c$
T_c
Numerical Control Effectiveness Inversion
$\boldsymbol{\omega}$
$\boldsymbol{\omega}_c$
$p(\cdot)$
$+$
$+$
$-$
$\int$
$\mathbf{K}_{\mathbf{I}_\omega}$
$\boldsymbol{\zeta}$
UAV
$\mathbf{x}$
$\mathbf{v}$
$\boldsymbol{\xi}$
$\mathbf{a}^b$
$\mathbf{R}$
$g\mathbf{i}_z$
LPF
$\mathbf{a}_{\text{lpf}}$
$\boldsymbol{\Omega}$
$\boldsymbol{\Omega}_{\text{lpf}}, \dot{\boldsymbol{\Omega}}_{\text{lpf}}$
$\|\cdot\|_2^2$
$\frac{-k_\tau}{m}$
τ
$\mathbf{b}_z$
$(\tau\mathbf{b}_z)_{\text{lpf}}$
$\cdot^{\circ 2}$
$(\boldsymbol{\omega}^{\circ 2})_{\text{lpf}}$
$(\dot{\boldsymbol{\omega}^{\circ 2}})_{\text{lpf}}$
$\mathbf{G}_1$
$\dot{\boldsymbol{\omega}}_{\text{lpf}}$
$\mathbf{G}_2$
$\mathbf{m}_{\text{lpf}}$
$\frac{1}{m}$
τ_{lpf}
$\dot{\tau}_{\text{lpf}}$

Figure 2-4: Motor control and computation of filtered signals. The blue and green areas contain the moment and thrust control (including motor speed command saturation resolution), and the motor speed control, respectively. Both are described in Section 2.5.5. The *UAV* block represents the UAV hardware, including ESCs, motors, and sensors. The red area contains the computation of filtered signals based on IMU and optical encoder measurements.

2.5.1 PD Position and Velocity Control

Position and velocity control is based on two cascaded proportional-derivative (PD) controllers. The resulting controller is mathematically equivalent to the following single expression:

$$\mathbf{a}_c = \mathbf{K}_\mathbf{x} \left(\mathbf{x}_\text{ref} - \mathbf{x}\right) + \mathbf{K}_\mathbf{v} \left(\mathbf{v}_\text{ref} - \mathbf{v}\right) + \mathbf{K}_\mathbf{a} \left(\mathbf{a}_\text{ref} - \mathbf{a}_\text{lpf}\right) + \mathbf{a}_\text{ref} \tag{2.45}$$

with $\mathbf{K}_\bullet$ indicating diagonal gain matrices. The subscript ref is used to indicate values obtained directly from the reference trajectory. In contrast, the subscript c indicates commanded values that are computed in one of the control loops. For example, $\mathbf{a}_\text{ref}$ is obtained directly from the reference trajectory as the second derivative of $\mathbf{x}_\text{ref}$, while $\mathbf{a}_c$ is computed based on (2.45) and includes terms based on the position, velocity, and acceleration deviations. The first three terms in (2.45) ensure tracking of position and velocity references, while the final term serves as a feedforward input to ensure tracking of the reference acceleration. The control utilizes the inertial reference frame with — in our implementation — identical gains for the horizontal $\mathbf{i}_x$- and $\mathbf{i}_y$-directions, but separately tuned gains for the vertical $\mathbf{i}_z$-direction. The commanded acceleration is used to calculate thrust and attitude commands, as will be shown in the next section.

2.5.2 INDI Linear Acceleration and Yaw Control

In this section, we present an intuitive derivation of the INDI linear acceleration control law that follows the practical working of the INDI notion based on estimation of the external force acting on the quadrotor. We arrive at control equations equivalent to those given in Section 2.4.3.

An expression for the external force in terms of measured acceleration and specific thrust is obtained by rewriting (2.2), as follows:

$$\mathbf{f}_\text{ext} = m \left(\mathbf{a}_\text{lpf} - (\tau \mathbf{b}_z)_\text{lpf} - g\mathbf{i}_z\right), \tag{2.46}$$

where τ is the specific thrust calculated according to (2.5) using motor speed measurements. Identical LPFs must be used to ensure that equal phase lag is incurred by acceleration and thrust measurements [115]. Note that the specific thrust vector and the linear acceleration (cf. (2.44)) are both transformed to the inertial reference frame prior to filtering. This order is appropriate because the external force in the inertial reference frame $\mathbf{f}_\text{ext}$ is assumed to be slow-changing relative to the LPF dynamics, as described in Section 2.3.3. Substitution of (2.46) into (2.2) gives the following expression for the current acceleration:

$$\begin{aligned}\mathbf{a} &= \tau\mathbf{b}_z + g\mathbf{i}_z + m^{-1}\mathbf{f}_\text{ext} \\ &= \tau\mathbf{b}_z + g\mathbf{i}_z + m^{-1}\left(m\left(\mathbf{a}_\text{lpf} - (\tau\mathbf{b}_z)_\text{lpf} - g\mathbf{i}_z\right)\right) \\ &= \tau\mathbf{b}_z - (\tau\mathbf{b}_z)_\text{lpf} + \mathbf{a}_\text{lpf}.\end{aligned} \tag{2.47}$$

The specific thrust vector command that results in the commanded acceleration prescribed by (2.45) can be computed using the following incremental relation based on (2.47):

$$(\tau\mathbf{b}_z)_c = (\tau\mathbf{b}_z)_\text{lpf} + \mathbf{a}_c - \mathbf{a}_\text{lpf}. \tag{2.48}$$

The incremental nature of (2.48) enables the controller to achieve the commanded acceleration despite possible disturbances or modeling errors. If the commanded value is not obtained immediately, the thrust and attitude commands will be incremented further in subsequent control updates. This principle eliminates the need for integral action anywhere in the control design.

The thrust magnitude command is obtained as

$$T_c = -m\|(\tau\mathbf{b}_z)_c\|_2 \tag{2.49}$$

with the negative sign following from the definition that thrust is positive in $\mathbf{b}_z$-direction. The incremental attitude command $\boldsymbol{\xi}_\mathbf{c}$ represents the rotation from the current attitude to the commanded attitude and is obtained in two steps: first, the minimum rotation to align $-\mathbf{b}_z$ with the thrust vector command $(\tau\mathbf{b}_z)_c$ is obtained; second, a rotation around $\mathbf{b}_z$ is added to satisfy the yaw reference ψ_ref. For the first step, we transform the normalized thrust vector command to the current body-fixed reference frame, as follows:

$$(-\mathbf{b}_z)_c^b = \boldsymbol{\xi}^{-1} \circ (-\mathbf{b}_z)_c \circ \boldsymbol{\xi}. \tag{2.50}$$

The appropriate rotation to align the current $-\mathbf{b}_z$ with $(\tau\mathbf{b}_z)_c$ is then given by

$$\bar{\boldsymbol{\xi}}_\mathbf{c} = \widehat{\begin{bmatrix} 1 - \mathbf{i}_z^\top (-\mathbf{b}_z)_c^b \\ -\mathbf{i}_z \times (-\mathbf{b}_z)_c^b \end{bmatrix}}, \tag{2.51}$$

where hat refers to quaternion normalization, i.e., $\hat{\boldsymbol{\xi}} = \boldsymbol{\xi}/\|\boldsymbol{\xi}\|_2$. For the second step, the yaw reference normal vector is first transformed to the intermediate attitude command frame, as follows:

$$\bar{\mathbf{n}}_{\psi_\text{ref}} = (\boldsymbol{\xi} \circ \bar{\boldsymbol{\xi}}_\mathbf{c})^{-1} \circ \begin{bmatrix} \sin\psi_\text{ref} & -\cos\psi_\text{ref} & 0 \end{bmatrix}^\top \circ (\boldsymbol{\xi} \circ \bar{\boldsymbol{\xi}}_\mathbf{c}). \tag{2.52}$$

Next, we obtain the following rotation that makes $\mathbf{b}_x$ coincide with the plane defined by normal vector $\bar{\mathbf{n}}_{\psi_\text{ref}}$:

$$\boldsymbol{\xi}_\psi = \widehat{\begin{bmatrix} 1 & 0 & 0 & -\frac{\bar{n}^1_{\psi_\text{ref}}}{\bar{n}^2_{\psi_\text{ref}}} \end{bmatrix}}^\top. \tag{2.53}$$

Equation (2.53) implicitly selects between tracking of ψ_ref and $\psi_\text{ref}+\pi$ rad based on minimizing the magnitude of rotation. Due to continuity of ψ_ref this does not cause any unwanted switching, but it does prevent unwanted discontinuities such as a π rad rotation around $\mathbf{b}_z$ to maintain yaw tracking when pitching through $\pm\pi/2$ rad. Note that (2.51) and (2.53) incur singularities if $\mathbf{i}_z = (-\mathbf{b}_z)_c^b$ and $\bar{n}^2_{\psi_\text{ref}} = 0$, respectively. However, by computing the attitude command relative to the current attitude, we move these singularities far away from the nominal trajectory. Moreover, they are straightforwardly detected and resolved by selecting any direction of rotation. Finally, the incremental attitude command is obtained as

$$\boldsymbol{\xi}_c = \bar{\boldsymbol{\xi}}_c \circ \boldsymbol{\xi}_\psi. \tag{2.54}$$

2.5.3 PD Attitude and Angular Rate Control

In this section, we describe the attitude and angular rate controller. This controller specifies the angular rate command and is thus solely based on angular kinematics. This has two major advantages compared to incorporating control torque or motor speeds. Firstly, the attitude controller does not take into account any model-specific parameters, such as the vehicle inertia matrix $\mathbf{J}$. Therefore the control design avoids discrepancies due to model mismatches and has vehicle-independent gains. Secondly, accurate torque control cognizant of the external moment $\mathbf{m}_{\text{ext}}$ can be performed separately using sensor-based INDI, as described in Section 2.5.4. This eliminates the need to incorporate a complicated disturbance model in the attitude controller, which further improves controller robustness and simplicity.

The three-element angle vector $\boldsymbol{\xi}_e$ associated with the incremental attitude command $\boldsymbol{\xi}_c$ is computed as follows:

$$\boldsymbol{\xi}_e = \frac{2 \arccos \xi_c^w}{\sqrt{1 - \xi_c^w \xi_c^w}} \begin{bmatrix} \xi_c^x & \xi_c^y & \xi_c^z \end{bmatrix}^\top . \tag{2.55}$$

Using these error angles, the angular acceleration command is obtained as

$$\dot{\boldsymbol{\Omega}}_c = \mathbf{K}_{\boldsymbol{\xi}} \boldsymbol{\xi}_e + \mathbf{K}_{\boldsymbol{\Omega}} \left(\boldsymbol{\Omega}_{\text{ref}} - \boldsymbol{\Omega}_{\text{lpf}} \right) + \dot{\boldsymbol{\Omega}}_{\text{ref}}, \tag{2.56}$$

where $\boldsymbol{\Omega}_{\text{ref}}$ and $\dot{\boldsymbol{\Omega}}_{\text{ref}}$ are the angular velocity and angular acceleration feedforward terms defined in (2.27) and (2.28), respectively. The resulting attitude controller not only tracks the attitude command but also tracks angular rate and acceleration. This enables tracking of trajectory jerk and snap, which is essential for accurate tracking of aggressive trajectories, as will be shown analytically in Section 2.6 and experimentally in Section 2.7. In contrast, trajectory tracking control based on body rate inputs, e.g., using an off-the-shelf flight controller, is incapable of truly considering reference snap, because snap corresponds to the vehicle angular acceleration, as shown in (2.28).

2.5.4 INDI Angular Acceleration Control

Robust tracking of the angular acceleration command $\dot{\boldsymbol{\Omega}}_c$ is achieved through INDI control. We rewrite (2.4) into the following expression for the external moment based on the measured angular rate, angular acceleration, and control moment:

$$\mathbf{m}_{\text{ext}} = \mathbf{J} \dot{\boldsymbol{\Omega}}_{\text{lpf}} - \mathbf{m}_{\text{lpf}} + \boldsymbol{\Omega}_{\text{lpf}} \times \mathbf{J} \boldsymbol{\Omega}_{\text{lpf}} \tag{2.57}$$

with $\mathbf{m}_{\text{lpf}}$ the control moment in the body-fixed reference frame, obtained from the measured motor speeds by (2.5) and low-pass filtering. Analogous to the external force in Section 2.5.2, the external moment $\mathbf{m}_{\text{ext}}$ is assumed slow-changing with regard to the LPF dynamics. Substitution of (2.57) into (2.4) then gives:

$$\begin{aligned} \dot{\boldsymbol{\Omega}} &= \mathbf{J}^{-1} (\mathbf{m} + \mathbf{m}_{\text{ext}} - \boldsymbol{\Omega} \times \mathbf{J} \boldsymbol{\Omega}) \\ &= \mathbf{J}^{-1} (\mathbf{m} + (\mathbf{J} \dot{\boldsymbol{\Omega}}_{\text{lpf}} - \mathbf{m}_{\text{lpf}} + \boldsymbol{\Omega}_{\text{lpf}} \times \mathbf{J} \boldsymbol{\Omega}_{\text{lpf}}) - \boldsymbol{\Omega} \times \mathbf{J} \boldsymbol{\Omega}) \\ &= \dot{\boldsymbol{\Omega}}_{\text{lpf}} + \mathbf{J}^{-1} (\mathbf{m} - \mathbf{m}_{\text{lpf}}). \end{aligned} \tag{2.58}$$

In (2.58), it is assumed that the difference between the gyroscopic angular momentum term and its filtered counterpart is sufficiently small to be neglected, because the term is relatively slow-changing compared to the angular acceleration and control moment, and moreover is second-order. By inversion of the final line, we obtain the following incremental expression for the commanded control moment:

$$\mathbf{m}_c = \mathbf{m}_{\text{lpf}} + \mathbf{J} \left(\dot{\boldsymbol{\Omega}}_c - \dot{\boldsymbol{\Omega}}_{\text{lpf}} \right). \tag{2.59}$$

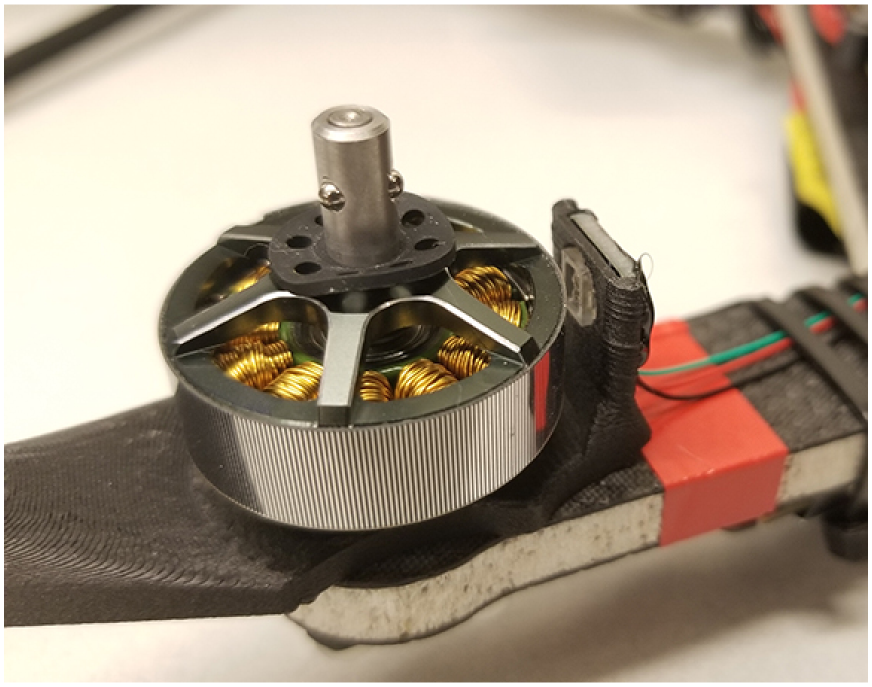


Figure 2-5: Motor (propeller removed) with optical encoder for rotational speed measurement. Note the optical encoder lens on the right, and the accompanying reflective strip on the motor hub.

2.5.5 Inversion-Based Moment and Thrust Control, and Integrative Motor Speed Control

In Section 2.5.2 and Section 2.5.4, we have found expressions for the commanded thrust T_c and control moment $\mathbf{m}_c$, respectively. Tracking of these commands requires control of the motor speeds, as evidenced by the direct relation given in (2.5). State-of-the-art INDI implementations for quadrotors are based on linearization of this relation and do not accurately model transient behavior [115, 116]. Our proposed implementation is based on a nonlinear inversion of the control effectiveness and explicitly incorporates the motor response time constant; as such, it provides a more accurate computation of control inputs.

In order to achieve fast and accurate closed-loop motor speed control, we employ optical encoders that measure the motor speeds. The availability of motor speed measurements furthermore enables accurate calculation of the thrust and control moment, as required by the INDI controller in (2.48) and (2.59). In practice, the optical encoder, shown in Fig. 2-5, measures the motor rotational speed by detecting the passage of stripes on a reflective strip attached to the motor hub. As such, the optical encoder provides a high-rate, accurate, lightweight, and unintrusive manner to obtain the motor speed.

The motor speed corresponding to the commanded thrust and control moment is found by inverting the nonlinear control effectiveness equation (2.5). In order to do so, we estimate the effect of the motor speed command on the motor speed derivative using the following first-order model:

$$\dot{\boldsymbol{\omega}} = \tau_m^{-1}(\boldsymbol{\omega}_c - \boldsymbol{\omega}) \tag{2.60}$$

with τ_m the motor dynamics time constant. After equating to the control moment and thrust commands, the resulting equation,

$$\begin{bmatrix} \mathbf{m}_c \\ T_c \end{bmatrix} = \mathbf{G}_1 \boldsymbol{\omega}_c^{\circ 2} + \tau_m^{-1} \mathbf{G}_2 (\boldsymbol{\omega}_c - \boldsymbol{\omega}), \tag{2.61}$$

can be solved numerically, e.g., using Newton's method. Inversion of this nonlinear control effectiveness relation improves the accuracy of thrust and control moment tracking, when compared to the linearized inversion that does not consider the motor transient response as given by (2.80).

Inversion of (2.61) may lead to infeasible, i.e., saturated, motor speed commands. We address this first by altering the control moment around $\mathbf{b}_z$. Since the control effectiveness is relatively much smaller around this axis, this is most likely to resolve the command saturation. Moreover, it typically least affects vehicle stability and position tracking, since rotation purely around the $\mathbf{b}_z$-axis does not alter the thrust vector. Let $\underline{\omega}$ and $\bar{\omega}$ be respectively the minimum and maximum feasible motor speeds, then the set of $\mathbf{b}_z$ control momenta — excluding J_{r_z} contributions — that result in feasible motor speed commands is

$$\begin{aligned}
\max\Bigg\{ \frac{k_{\mu_z}}{k_\tau} \left(4k_\tau \underline{\omega}^2 + T_c \pm \left(\frac{\mu_c^y}{l_x} - \frac{\mu_c^x}{l_y} \right) \right), \\
- \frac{k_{\mu_z}}{k_\tau} \left(4k_\tau \bar{\omega}^2 + T_c \pm \left(\frac{\mu_c^y}{l_x} + \frac{\mu_c^x}{l_y} \right) \right) \Bigg\} \leq \mu_c^z \\
\leq \min\Bigg\{ \frac{k_{\mu_z}}{k_\tau} \left(4k_\tau \bar{\omega}^2 + T_c \pm \left(\frac{\mu_c^y}{l_x} - \frac{\mu_c^x}{l_y} \right) \right), \\
- \frac{k_{\mu_z}}{k_\tau} \left(4k_\tau \underline{\omega}^2 + T_c \pm \left(\frac{\mu_c^y}{l_x} + \frac{\mu_c^x}{l_y} \right) \right) \Bigg\}.
\end{aligned} \tag{2.62}$$

If this set is non-empty, we set μ_c^z to equal the boundary closest to the original moment command. The motor speed command is then obtained as

$$\boldsymbol{\omega}_c = \left(\mathbf{G}_1^{-1} \begin{bmatrix} \mathbf{m}_c \\ T_c \end{bmatrix} \right)^{\circ \frac{1}{2}}. \tag{2.63}$$

Note that due to J_{r_z} contributions, the actual $\mathbf{b}_z$ control moment will not exactly be equal to μ_c^z. However, we still obtain the feasible control moment that is closest to the original commanded moment, because k_{μ_z} and J_{r_z} have identical signs in (2.6) and (2.7), respectively. If there exists no μ_c^z that results in feasible motor commands, we consider a reduction or limited increase in the thrust magnitude command T_c based on the reasoning that application of thrust is only effective in the correct direction, i.e., at the correct vehicle pitch and roll. Since adjustment of T_c results in equal magnitude shift of the constraints, it is straightforward to verify whether there exists an acceptable value of T_c such that the lower and upper boundaries in (2.62) coincide. If so, T_c is set to this value and μ_c^z to the feasible point, after which (2.63) is used to compute the motor speed commands. If not, μ_c^z is set to the average of the lower and upper boundaries in (2.62), and any infeasible motor speed commands resulting from (2.63) are clipped.

Finally, the throttle vector $\boldsymbol{\zeta}$ that contains the motor electronic speed control (ESC) commands is obtained as follows:

$$\boldsymbol{\zeta} = p(\boldsymbol{\omega_c}) + \mathbf{K}_{\mathbf{I}_\omega} \int \boldsymbol{\omega}_c - \boldsymbol{\omega} dt \tag{2.64}$$

with p a vector-valued polynomial function relating motor speeds to throttle inputs. This function was obtained by regression analysis of static test data. Integral action is added to account for changes in this relation due to decreasing battery voltage. The measured motor speed signal $\boldsymbol{\omega}$ remains unfiltered here to minimize phase lag.

2.6 Response Analysis

Incremental control and the tracking of high-order reference derivatives are two key aspects of our control design. In this section, we theoretically verify the advantages of these features. Namely, the improved robustness of incremental control in comparison to non-incremental control, and the improved trajectory tracking accuracy due to the consideration of high-order reference trajectory derivatives, i.e., jerk and snap. The purpose of this section is to provide an intuitive understanding of how these aspects improve tracking performance. In order to analyze the behavior of the closed-loop system, we use linearized dynamics and control equations, as the resulting simplifications allow for easier qualitative interpretation. However, the observations in this section also apply to the full, nonlinear dynamics and control equations. Our findings are validated and quantitatively assessed using real-life flights in Section 2.7.

We consider forward and pitch movement around the hover state. The subscript x indicates the forward component, e.g., $a_{x,\mathrm{ref}} = \mathbf{a}_{\mathrm{ref}}^\top \mathbf{i}_x$, and the subscript y the pitch component, e.g., $\mu_y = \mathbf{m}^\top \mathbf{i}_y$. In hover condition, $\tau = -g$, $\theta = 0$, and $\boldsymbol{\Omega} = \mathbf{0}_{3\times1}$, so that (2.2) and (2.4) can be linearized to obtain

$$a_x = -g\theta + m^{-1} f_{x,\mathrm{ext}}, \tag{2.65}$$

$$J_{yy}\dot{q} = \mu_y + \mu_{y,\mathrm{ext}}, \tag{2.66}$$

where θ is the pitch angle, and J_{yy} is the vehicle moment of inertia about the $\mathbf{b}_y$-axis. Similarly, the INDI linear acceleration control law (2.48) is linearized to obtain the error angle

$$-g\theta_e = -g\theta_{\mathrm{lpf}} + a_{x,\mathrm{ref}} - (a_x)_{\mathrm{lpf}} + g\theta, \tag{2.67}$$

where $-g\theta_{\mathrm{lpf}}$ represents the forward component of the specific thrust vector, and $(a_x)_{\mathrm{lpf}}$ represents the filtered forward acceleration as obtained by (2.44). The commanded pitch acceleration α_c is obtained by taking the pitch component of (2.56), as follows:

$$\alpha_c = k_\theta \theta_e + k_q \left(q_{\mathrm{ref}} - q_{\mathrm{lpf}}\right) + \alpha_{\mathrm{ref}}, \tag{2.68}$$

where $q_{\mathrm{ref}} = -\frac{j_{x,\mathrm{ref}}}{g}$ and $\alpha_{\mathrm{ref}} = -\frac{s_{x,\mathrm{ref}}}{g}$ by linearization of (2.27) and (2.28). The scalar control gains k_θ and k_q are obtained by selecting the pitch elements from the corresponding control gain matrices described in Section 2.5.

Next, we linearize the angular acceleration and moment control laws. The four motors can be modeled collectively, as the system is linearized around the hover state where all motors have identical angular speeds. The scalar value ω refers to the deviation from the hover state motor speed ω_0, or, equivalently, to half of the angular speed difference between

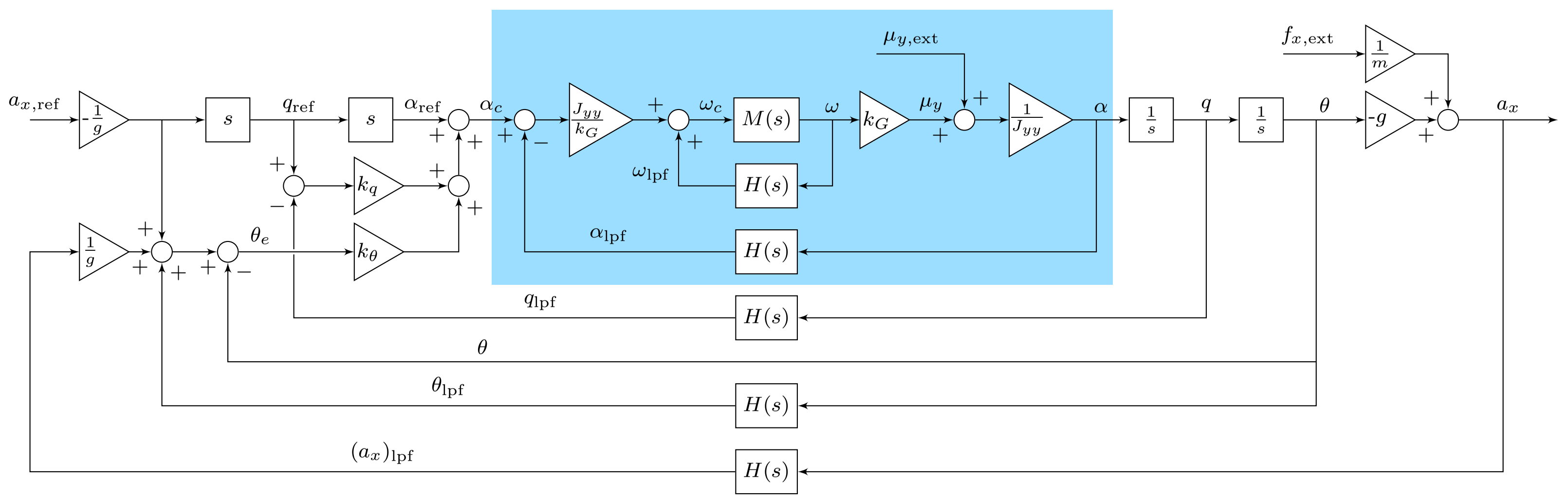


Figure 2-6: Linearized closed-loop forward acceleration dynamics, with pitch acceleration dynamics in blue area.

the front and rear motor pairs. Equating (2.59) and (2.61), and isolating the pitch channel gives

$$\omega_c = \sqrt{((\omega_0 + \omega)^2)_{\text{lpf}} + J_{yy}(4l_x k_\tau)^{-1}(\alpha_c - \alpha_{\text{lpf}})} - \omega_0. \tag{2.69}$$

with the factor 4 due to the number of motors. Linearization around the hover state gives

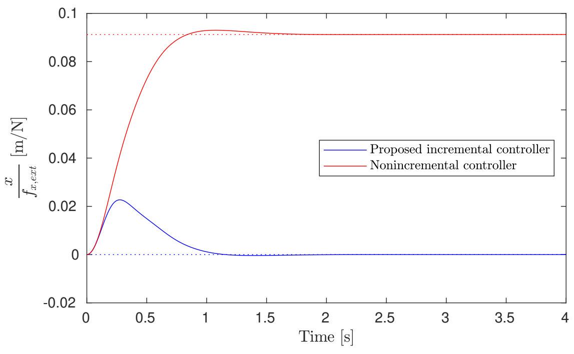
$$\omega_c = \omega_{\text{lpf}} + J_{yy}k_G^{-1}(\alpha_c - \alpha_{\text{lpf}}), \tag{2.70}$$

with the linearized control effectiveness gain $k_G = 8\omega_0 l_x k_\tau$, so that $\mu_y = k_G\omega$.

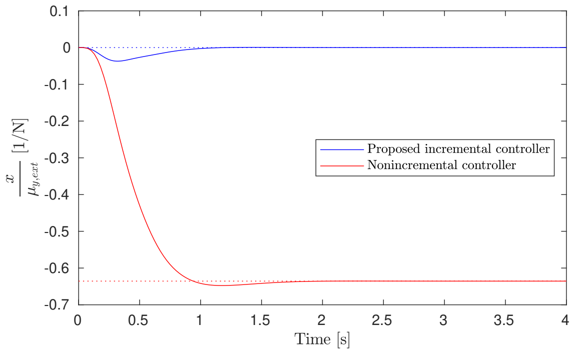
In order to analyze the robustness properties provided by the proposed incremental controller, it is compared to a regular, i.e., non-incremental, controller with linearized equations (cf. (2.67) and (2.70))

$$\theta_{c,NI} = -\frac{a_{x,\text{ref}}}{g}, \qquad \omega_{c,NI} = \frac{J_{yy}}{k_G}\alpha_c, \tag{2.71}$$

where α_c is still given by (2.68) using $\theta_e = \theta_c - \theta$, and the subscript NI is used to indicate the non-incremental controller.



(a) Position response to $f_{x,\text{ext}}$ step input.



(b) Position response to $\mu_{y,\text{ext}}$ step input.

Figure 2-7: Simulated disturbance response using the proposed incremental controller, and a non-incremental controller.

2.6.1 Robustness against Disturbance Forces and Moments

An overview of the resulting linearized closed-loop acceleration dynamics is given in Fig. 2-6. From the blue area, we obtain the following pitch acceleration dynamics:

$$\frac{\alpha}{\alpha_c}(s) = \frac{J_{yy}k_G^{-1}\frac{M(s)}{1-M(s)H(s)}k_G J_{yy}^{-1}}{1 + J_{yy}k_G^{-1}\frac{M(s)}{1-M(s)H(s)}k_G J_{yy}^{-1}H(s)} = M(s), \tag{2.72}$$

$$\frac{\alpha}{\mu_{y,\text{ext}}}(s) = \frac{J_{yy}^{-1}}{1 + \frac{H(s)M(s)}{1-H(s)M(s)}} = J_{yy}^{-1}\left(1 - H(s)M(s)\right) \tag{2.73}$$

with $\alpha(s)$ the pitch acceleration, i.e., $\alpha(s) = sq(s) = s^2\theta(s)$. The LPF transfer function is denoted by $H(s)$, e.g., $\frac{\alpha_{\text{lpf}}}{\alpha}(s) = H(s)$, and the motor (control) dynamics are denoted by $M(s)$, i.e., $\frac{\omega}{\omega_c}(s) = M(s)$. In (2.72), we observe that the closed-loop angular acceleration dynamics are solely determined by the motor dynamics [115]. Hence, the aggressiveness of

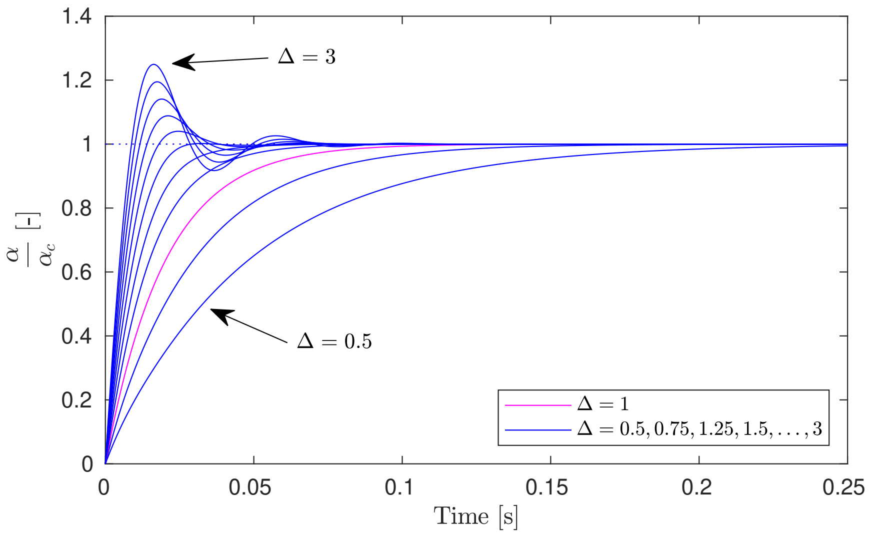


Figure 2-8: Simulated angular acceleration step response for various modeling errors using the proposed incremental controller.

trajectories that can be tracked is theoretically limited by only the bandwidth of the motor response. This is also the case for a non-incremental version of the controller.

The disturbance moment $\mu_{y,\text{ext}}$ is fully counteracted using incremental control based on the two feedback loops in the blue shaded area of Fig. 2-6: the expected angular acceleration from the motor speeds, i.e., $\frac{k_G}{J_{yy}}\omega_{\text{lpf}}$, and the measured angular acceleration α_{lpf}, which includes the effects of the disturbance moment. As shown in (2.73), the counteraction depends on $H(s)$ and $M(s)$ so that the ability to reject disturbances is limited by the bandwidth of both the LPFs and the motors. To the contrary, in a non-incremental controller the α_{lpf} and ω_{lpf} feedback loops are not present, so that $\alpha = J_{yy}^{-1}\mu_{y,\text{ext}}$ (cf. (2.73)). The disturbance moment now propagates undamped to the attitude and position control loops, as there is no closed-loop angular acceleration control that directly evaluates the moments acting on the vehicle.

We obtain similar results for the disturbance force $f_{x,\text{ext}}$, which is corrected for incrementally using the difference between the acceleration due to thrust, i.e., $-g\theta_{\text{lpf}}$, and the true acceleration including the disturbance force, i.e., $(a_x)_{\text{lpf}}$. All in all, the proposed incremental controller maintains identical nominal reference tracking performance for both angular and linear accelerations, while achieving superior disturbance rejection of external moments and forces when compared to the non-incremental controller.

In order to evaluate the effect of the disturbance force and moment on the position tracking error, we close the loop around Fig. 2-6 using the position and velocity controller given by (2.45). Figure 2-7 shows the resulting step responses for both incremental and non-incremental control. The response was simulated using the platform-independent control gains given in Table 2.3, a second-order Butterworth filter with cut-off frequency equal to 188.5 rad/s (30 Hz), and the first-order motor model given by (2.60) with τ_m set to 20 ms. It can be seen that the proposed incremental controller is able to counteract the disturbances and reaches zero steady-state error, while the non-incremental controller is unable to do so. In order to null the steady-state errors due to force and moment disturbances, integral action must be added to the non-incremental controller. This is not necessary in the case of INDI, so that our proposed control design is able to quickly and wholly counteract disturbance

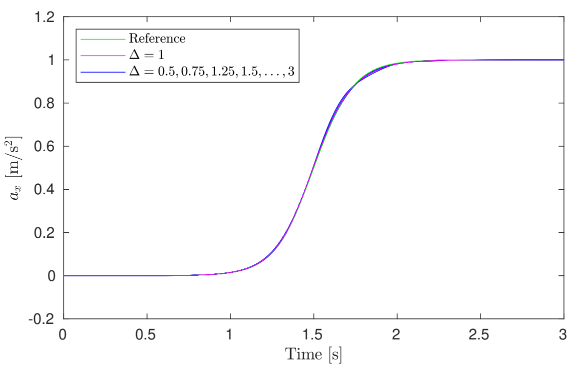
Table 2.3: Trajectory tracking controller gains.

Gain	Value
$\mathbf{K_x}$	diag ([18 18 13.5])
$\mathbf{K_v}$	diag ([7.8 7.8 5.9])
$\mathbf{K_a}$	diag ([0.5 0.5 0.3])
$\mathbf{K}_{\boldsymbol{\xi}}$	diag ([175 175 82])
$\mathbf{K}_{\dot{\boldsymbol{\xi}}}$	diag ([19.5 19.5 19.2])

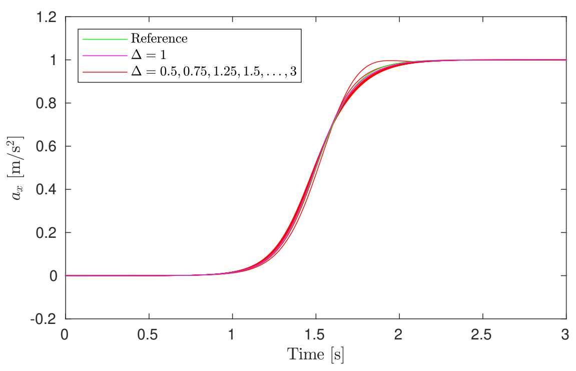
Table 2.4: 3D trajectory tracking performance for experiments with forward yaw and constant yaw.

	Forward yaw	Constant yaw
RMS $\lVert \mathbf{x} - \mathbf{x}_{\text{ref}} \rVert_2$ [cm]	6.6	6.1
max $\lVert \mathbf{x} - \mathbf{x}_{\text{ref}} \rVert_2$ [cm]	10.8	11.9
RMS $\lvert \psi - \psi_{\text{ref}} \rvert$ [deg]	5.1	1.9
max $\lvert \psi - \psi_{\text{ref}} \rvert$ [deg]	12.8	6.4
RMS $\lVert \mathbf{v} \rVert_2$ [m/s]	6.8	5.6
max $\lVert \mathbf{v} \rVert_2$ [m/s]	12.9	11.3
RMS $\lVert \mathbf{a} - g\mathbf{i}_z \rVert_2$ [m/s^2]	14.4	12.5
max $\lVert \mathbf{a} - g\mathbf{i}_z \rVert_2$ [m/s^2]	20.8	20.0

forces and moments, while avoiding the negative effects that integral action typically has on the tracking performance, e.g., degraded stability, and increased overshoot and settling time.



(a) Using the proposed incremental controller.



(b) Using a non-incremental controller.

Figure 2-9: Simulated linear acceleration tracking response for various modeling errors.

2.6.2 Robustness against Modeling Errors

The proposed control design requires only a few vehicle-specific parameters. Nonetheless, it is desirable that tracking performance is maintained if inaccurate parameters are used, e.g., because control effectiveness data obtained from static tests may not be representative for the entire flight envelope. The linearized control equations described above incorporate the ratio of the moment of inertia J_{yy} and the linearized control effectiveness k_G. We denote

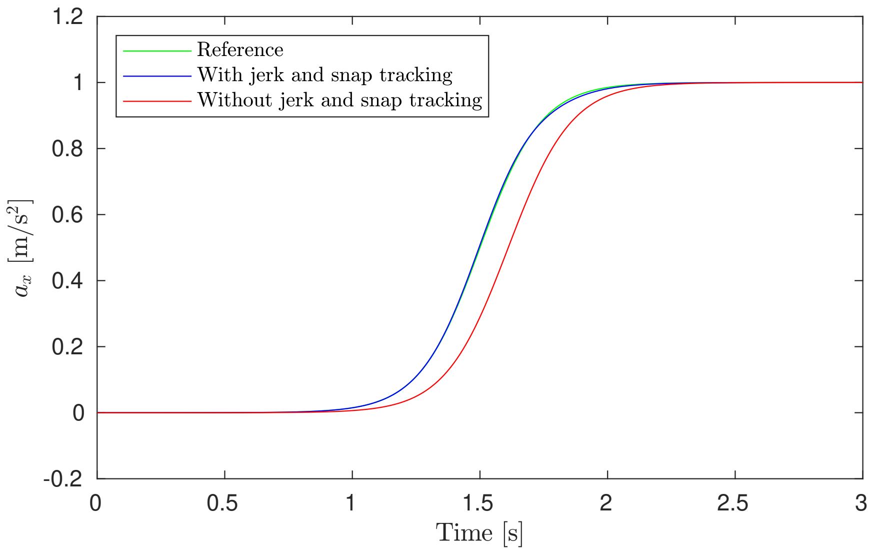


Figure 2-10: Simulated linear acceleration tracking response using the proposed controller with and without jerk and snap tracking.

Table 2.5: Roulette curve trajectory tracking performance for: **(i)** the proposed controller; **(ii)** jerk and snap tracking disabled; and **(iii)** drag plate attached.

	(i)	(ii)	(iii)
RMS $\|\mathbf{x}-\mathbf{x}_{\text{ref}}\|_2$ [cm]	9.0	16.8	7.6
max $\|\mathbf{x}-\mathbf{x}_{\text{ref}}\|_2$ [cm]	14.3	28.9	14.2
RMS ψ [deg]	1.8	5.3	12.6
max $\lvert\psi\rvert$ [deg]	5.3	14.9	51.7
RMS $\|\mathbf{v}\|_2$ [m/s]	3.7	4.3	3.8
max $\|\mathbf{v}\|_2$ [m/s]	7.3	8.2	7.7
RMS $\|\mathbf{a}-g\mathbf{i}_z\|_2$ [m/s^2]	14.0	15.2	14.2
max $\|\mathbf{a}-g\mathbf{i}_z\|_2$ [m/s^2]	19.1	21.3	20.4

the values used in the controller $\bar{J}_{yy}$ and $\bar{k}_G$, and define the modeling error Δ such that

$$\frac{\bar{J}_{yy}}{\bar{k}_G} = \Delta \frac{J_{yy}}{k_G}. \tag{2.74}$$

This leads to the following pitch acceleration dynamics for the proposed incremental NDI controller, and the non-incremental controller described above:

$$\frac{\alpha}{\alpha_c}(s) = \frac{\Delta M(s)}{(\Delta - 1)H(s)M(s) + 1}, \tag{2.75}$$

$$\frac{\alpha}{\alpha_{c\,NI}}(s) = \Delta M(s). \tag{2.76}$$

It can be seen that the error acts as a simple gain in the non-incremental controller, leading to an incorrect angular acceleration. On the contrary, the proposed incremental controller compares the expected angular acceleration from the motor speeds, i.e., $\frac{\bar{k}_G}{\bar{J}_{yy}}\omega_{\text{lpf}}$, to the measured angular acceleration, i.e., α_{lpf}, to implicitly correct for the modeling error. The corresponding angular acceleration responses for several values of Δ are shown in Fig. 2-8. The figure shows that the modeling error affects the transient response, but that the incremental controller is able to correct for it and quickly reaches the commanded acceleration value even for very large model discrepancies.

In order to assess the effect of modeling errors on acceleration tracking, we simulate the time response to the following acceleration reference:

$$a_{x,\text{ref}}(t) = \frac{1}{2}\tanh\left(\frac{4}{3}\pi t - 2\pi\right) + \frac{1}{2}, \tag{2.77}$$

which is C^2, i.e., the corresponding jerk and snap signals are continuous, and has boundary conditions $a_{x,\text{ref}}(0) = j_{x,\text{ref}}(0) = s_{x,\text{ref}}(0) = j_{x,\text{ref}}(3) = s_{x,\text{ref}}(3) = 0$ and $a_{x,\text{ref}}(3) = 1$ m/s^2. The responses for various values of Δ are shown in Fig. 2-9. It can be seen that the incremental controller is able to accurately track the reference signal even when large modeling errors are present. When non-incremental control is used, the tracking performance declines more severely with growing modeling error.

2.6.3 Jerk and Snap Tracking

Jerk and snap tracking is a crucial aspect of the proposed controller design that enables tracking of fast-changing acceleration references. It is embodied by the feedforward terms $k_q s$ and s^2 in the nominator of the acceleration response transfer function

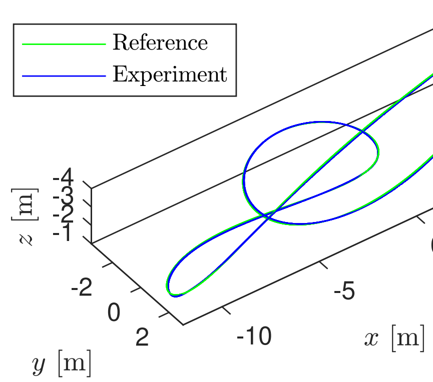
$$\frac{a_x}{a_{x,\text{ref}}}(s) = \frac{M(s)\left(s^2 + k_q s + k_\theta\right)}{s^2 + k_q H(s)M(s)s + k_\theta M(s)}. \tag{2.78}$$

These feedforward terms add two zeros to the closed-loop transfer function. These zeros — in combination with the LPF — act essentially as a lead compensator and help improve the transient response of the system. Effective placement of the zeros through tuning of k_q leads to improved tracking of a rapidly changing acceleration input signal, e.g., during aggressive flight maneuvers.

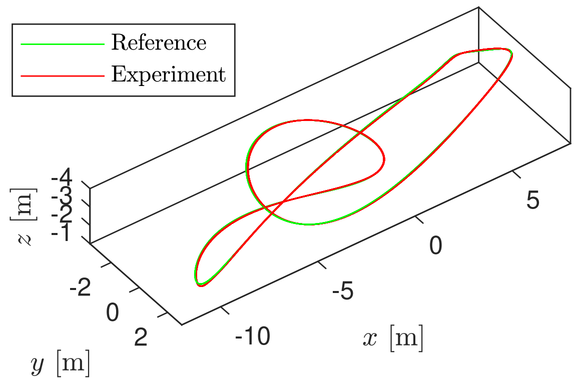
Figure 2-10 shows the simulated acceleration responses with and without jerk and snap tracking to the reference signal defined in (2.77). It can be seen that the inclusion of jerk and snap tracking causes a faster response, resulting in more accurate acceleration tracking. In the next section, we show that the improvement is also achieved in practice.

2.7 Experimental Results

In this section, experimental results for high-speed, high-acceleration flight are presented. A video of the experiments is available at `https://youtu.be/K15lNBAKDCs`. We evaluate the performance of the trajectory tracking controller on two trajectories that include yawing, tight turns with acceleration up to over 2g, and high-speed straights at up to 12.9 m/s. Furthermore, we examine the effect of the feedforward inputs based on the reference trajectory jerk and snap. We establish the independence of any model-based drag estimate by attaching a drag-inducing cardboard plate that more than triples the frontal area of the vehicle. Robustness against external disturbance forces is further displayed by pulling on a string attached to the quadcopter in hover. Finally, we compare the proposed nonlinear

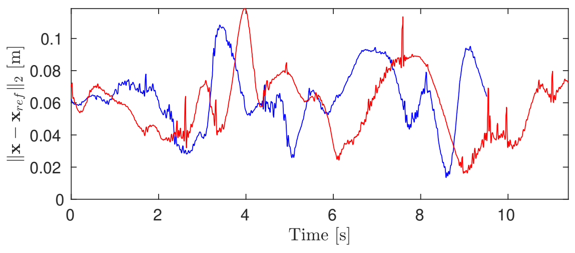


(a) Forward yaw.

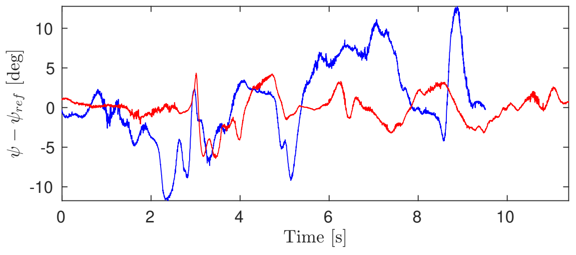


(b) Constant yaw.

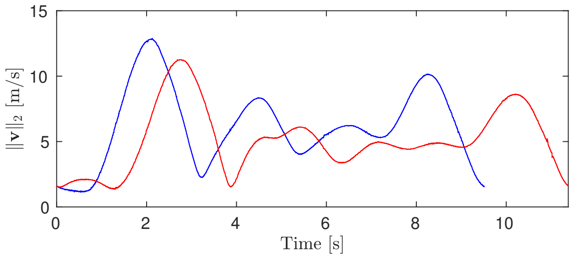
Figure 2-11: Experimental flight results for 3D trajectory.



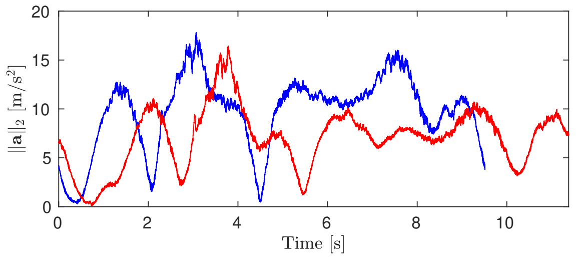
(a) Euclidean norm of position error.



(b) Yaw error.

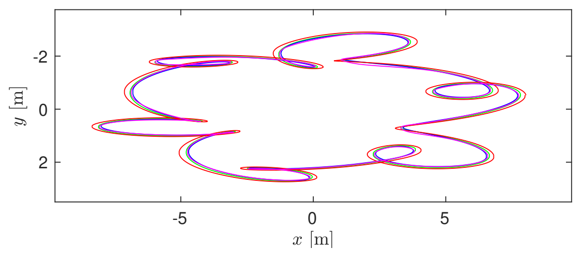


(c) Euclidean norm of velocity.

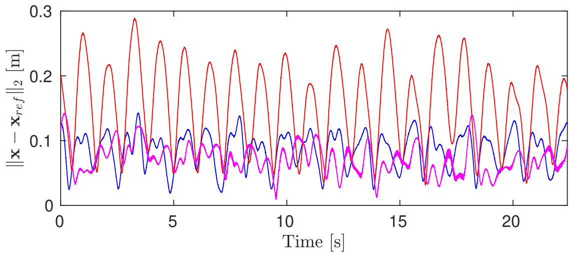


(d) Euclidean norm of acceleration.

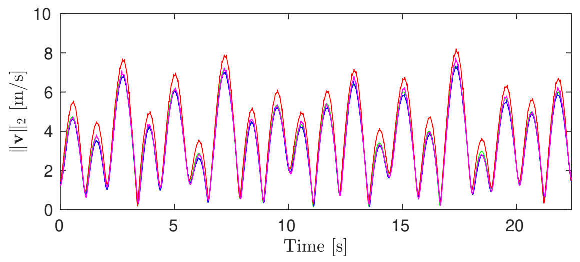
Figure 2-12: Experimental flight results for 3D trajectory: forward yaw (blue), and constant yaw (red).



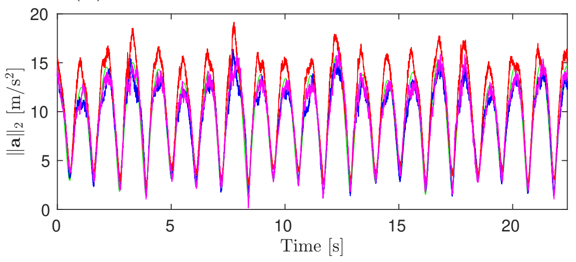
(a) Position.



(b) Euclidean norm of position error.



(c) Euclidean norm of velocity.



(d) Euclidean norm of acceleration.

Figure 2-13: Experimental flight results for roulette curve trajectory: reference trajectory (green), proposed controller (blue), without jerk and snap tracking (red), and with drag plate attached (magenta).

INDI angular acceleration control to its linearized counterpart.

2.7.1 Experimental Setup

Experiments were performed in an indoor flight room using the quadcopter shown in Fig. 2-1. The quadrotor body is machined out of carbon fiber composite with balsa wood core. The propulsion system consists of T-Motor F35A ESCs and F40 Pro II Kv 2400KV motors with Gemfan Hulkie 5055 propellers. Adjacent motors are mounted 18 cm apart. The quadcopter is powered by a single 4S LiPo battery. Its total flying mass is 609 g.

Control computations are performed at 2000 Hz using an onboard STM32H7 400 MHz microcontroller running custom firmware. On this platform, the total computation time of a control update at 32-bit floating point precision is 16 μs. Linear acceleration and angular rate measurements are obtained from an onboard Analog Devices ADIS16477-3 IMU at 2000 Hz, while position, velocity, and orientation measurements are obtained from an OptiTrack motion capture system at 360 Hz with an average latency of 18 ms. The latency is corrected for by propagating motion capture data using integrated IMU measurements. Motor speed measurements are obtained from the optical encoders at approximately 5000 Hz. The motor speed and IMU measurements are low-pass filtered using a software second-order Butterworth filter with cutoff frequency 188.5 rad/s (30 Hz).

The platform-independent controller gains listed in Table 2.3 were used. Additionally, the controller requires several platform-specific parameters, namely: vehicle mass m, moment of inertia $\mathbf{J}$, motor time constant τ_m, control effectiveness matrices $\mathbf{G}_1$ and $\mathbf{G}_2$, and the gain and polynomial fit used by the motor speed controller. We obtained control effectiveness data from static tests. In experiments, it was found that using the controller on a different quadcopter (with different dynamic properties, inertial sensors, and propulsion system) required no changes to controller algorithms or gains. After updating only the aforementioned platform-specific parameters, the controller performed without loss of tracking accuracy.

2.7.2 Evaluation of Proposed Controller

In this section, we evaluate the performance of the trajectory tracking controller on two trajectories: a 3D trajectory that includes a high-speed straight and fast turns, and a roulette curve trajectory consisting of fast successive turns resulting in high jerk and snap. The 3D trajectory is generated from a set of waypoints using the method described in [105]. The trajectory is flown with two yaw references: forward yaw, i.e., with the $\mathbf{b}_x$-axis in the velocity direction, and constant yaw set to zero. Figure 2-11 shows the corresponding reference trajectories, along with experimental results. The forward yaw trajectory is flown in a slightly shorter time. Performance data for both trajectories are given in Table 2.4 and shown in Fig. 2-12. Over the forward yaw trajectory, a maximum speed of 12.9 m/s is achieved, while the RMS tracking error is limited to 6.6 cm. The vehicle attains a maximum proper acceleration of 20.8 m/s^2 (2.12g). Similar values can be observed for the constant yaw trajectory. The most significant difference is a reduction in yaw tracking error from an RMS value of 5.1 deg to 1.9 deg and from a maximum value of 13 deg to 6.4 deg.

The second, roulette curve trajectory is defined as

$$\boldsymbol{\sigma}_{\text{ref}}(t) = \begin{bmatrix} r_1 \cos k_1 t + r_2 \cos k_2 t + r_3 \sin k_3 t \\ r_4 \sin k_1 t + r_3 \sin k_2 t + r_5 \cos k_3 t \\ r_z \\ 0 \end{bmatrix} \tag{2.79}$$

with r_1 = 6 m, r_2 = 1.8 m, r_3 = 0.6 m, r_4 = -2.25 m, r_5 = -0.3 m, r_6 = -0.45 m, k_1 = 0.28 rad/s, k_2 = 2.8 rad/s, k_3 = 1.4 rad/s, and r_z a constant offset. The trajectory, shown in Fig. 2-13(a), contains fast, successive turns. Accurate tracking is particularly demanding as it requires fast changes in acceleration, i.e., large jerk and snap, requiring high angular rates and angular accelerations. A single lap is traversed in 22.4 s. The position tracking error is shown in blue in Fig. 2-13(b), and tracking performance metrics are given in the first column of Table 2.5. Comparison of the position tracking error to the values in Table 2.4 confirms that the controller achieves consistent performance across trajectories. Due to its arduous nature, the roulette curve trajectory is particularly suitable to expose differences in tracking performance. Therefore, we use the trajectory defined by (2.79) to examine several modifications in subsequent sections. In all cases, the trajectory parameters are identical to those given above.

2.7.3 Jerk and Snap Tracking

The red curves in Fig. 2-13 correspond to our proposed control design, but with jerk and snap tracking disabled, i.e., $\boldsymbol{\Omega}_{\text{ref}} = \dot{\boldsymbol{\Omega}}_{\text{ref}} = \mathbf{0}_{3\times1}$. Examination of the figures shows the significant improvement in trajectory tracking performance obtained through the tracking of the jerk and snap feedforward terms. This observation is confirmed by comparing the first two columns of Table 2.5. It can be seen that the RMS position tracking error increases from 9.0 cm to 16.8 cm when jerk and snap tracking are disabled. In Section 2.6, it was shown that lead compensation provided by jerk and snap tracking results in improved performance when tracking fast-changing acceleration commands. This effect can also be observed in

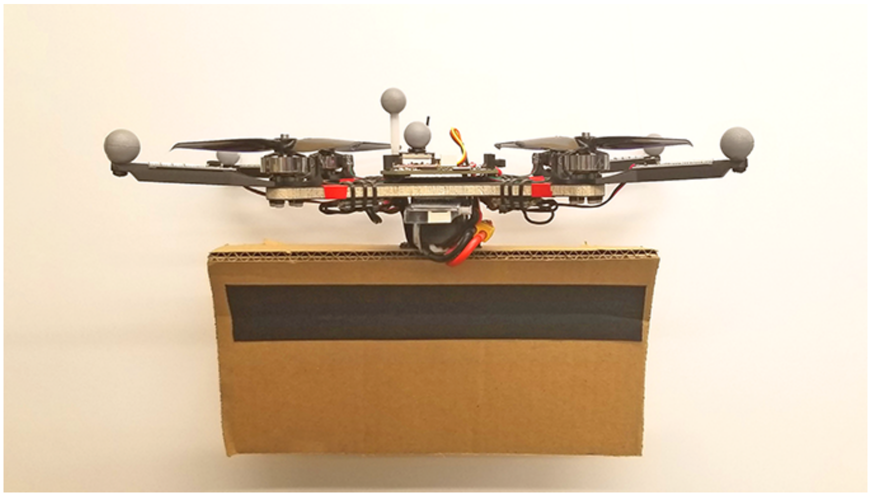


Figure 2-14: Quadrotor with 16 cm × 32 cm cardboard drag plate.

Fig. 2-13. It can be seen that the system response has less overshoot when jerk and snap tracking are enabled, conform the analytical response of the linearized system.

2.7.4 Increased Aerodynamic Drag

The magenta curves in Fig. 2-13 correspond to the trajectory tracking controller as described in this chapter, but using the quadcopter with attached drag plate. The drag plate is a 16 cm × 32 cm cardboard plate that is attached to the bottom of the quadrotor, as shown in Fig. 2-14. The plate more than triples the frontal surface area of the quadrotor, and as such has a significant effect on the aerodynamic force and moment that act on the vehicle, especially during high-speed flight, and fast pitch and yaw motion. The flight controller is not adapted in any way to account for either these aerodynamic effects, or the changes in mass and moment of inertia.

Comparison of columns (i) and (iii) in Table 2.5 shows that the drag plate does not significantly affect position tracking performance. Yaw tracking performance is also consistent, except when the drag plate generates an external yaw moment that causes motor speed saturation and very large momentary yaw tracking error. The consistent tracking performance demonstrates the robustness property of INDI. Controllers that depend on the estimation of drag forces based on velocity, such as [22] and [123], may suffer from much larger loss of tracking performance when the aerodynamic properties of the vehicle are modified. Instead of depending on a model-based drag estimate, INDI counteracts the disturbance force and moment by sensor-based incremental control. The controller implicitly estimates the external force by (2.46). In Fig. 2-15, it can be seen that the drag plate has a significant effect on the external disturbance force: its estimated magnitude is approximately tripled. In order to counteract the greater external force, commanded thrust and vehicle pitch increase when the drag plate is attached.

2.7.5 Nonlinear Control Effectiveness Inversion

We also compare our proposed nonlinear inversion of the control effectiveness (2.61), with linearized INDI as presented in [116]. In the latter case, control moment and thrust com-

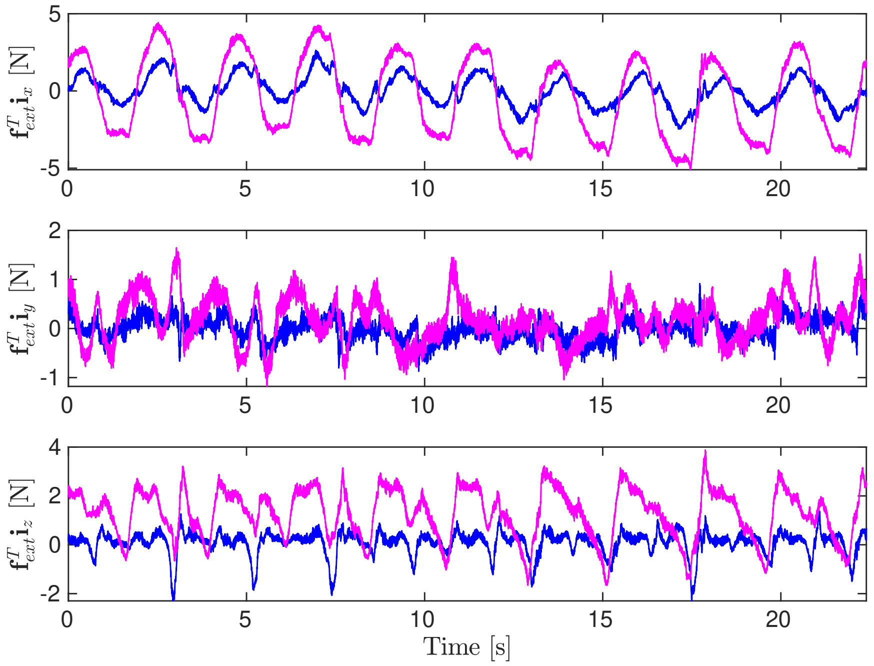


Figure 2-15: Estimated external disturbance force for roulette curve trajectory: proposed controller (blue), and with drag plate attached (magenta).

mands are tracked using linearized inversion of (2.5), as follows:

$$\boldsymbol{\omega}_c = \boldsymbol{\omega}_{\text{lpf}} + \left(2\mathbf{G}_1 + \Delta t^{-1}\mathbf{G_2}\right)\left(\begin{bmatrix} \mathbf{m}_c - \mathbf{m}_{\text{lpf}} \\ T_c - T_{\text{lpf}} \end{bmatrix} + \Delta t^{-1}\mathbf{G}_2 B(\boldsymbol{\omega}_c - \boldsymbol{\omega}_{\text{lpf}})\right), \tag{2.80}$$

where B is the one-sample backshift operator and Δt is the controller update interval. This linearized inversion does not take into account local nonlinearity of (2.5), nor does it consider the transient response of the motors. Therefore, nonlinear inversion of (2.61) — as described in Section 2.5.5 — theoretically results in improved tracking of the angular acceleration command and thereby in improved trajectory tracking performance.

In experimental flights, we found that the difference between nonlinear and linearized inversion does not lead to significant differences in tracking performance for our quadrotor system. However, we found that the failure to properly consider the transient response of the motors in (2.80) can be detrimental for controller performance. In particular, if the motor time constant τ_m and the controller interval Δt differ greatly, this may result in fast yaw oscillations. Consideration of the motor time constant τ_m, as in (2.61), resolves this issue.

2.7.6 Hover with Disturbance Force

For a constant $\boldsymbol{\sigma}_{\text{ref}}$ input, i.e., hover, the controller consistently achieves sub-centimeter position tracking error if no external disturbance is purposely applied. In this section, we present results for hover with an external disturbance force through a tensioned wire. One end of the wire is attached to the bottom plate of the quadrotor. We pull on the other end of the wire to drag the vehicle away from its hover position.

In Fig. 2-16, it can be seen that the quadrotor maintains its position to within at most 4 cm, while a changing disturbance force is applied through the wire. The largest position error occurs around 10 s when an external force of approximately 3.7 N is applied. Figure 2-17 shows the estimated external disturbance force, computed according to (2.46). The force component in the $\mathbf{i}_z$-direction has a small steady-state value due to discrepancy between true and estimated thrust. Comparison to Fig. 2-18 shows that the direction of the estimated external disturbance force vector corresponds to the direction of the wire. For example, at 22 s, Fig. 2-17 shows that the external force has a negative component in the $\mathbf{i}_x$-direction and a positive component in the $\mathbf{i}_y$-direction, and in Fig. 2-18(a) the wire is indeed tensioned in negative $\mathbf{i}_x$- and positive $\mathbf{i}_y$-direction.

2.8 Summary

In this chapter, we proposed a novel control system for the tracking of aggressive, i.e., fast and agile, trajectories for quadrotor vehicles. Our controller tracks reference position and yaw angle with their derivatives of up to fourth order, specifically, the position, velocity, acceleration, jerk, and snap along with the yaw angle, yaw rate, and yaw acceleration using incremental nonlinear dynamic inversion and differential flatness. The tracking of snap was enabled by closed-loop control of the propeller speeds using optical encoders attached to each motor hub. The resulting control system achieves 6.6 cm RMS position tracking error in agile and fast flight, reaching a top speed of 12.9 m/s and acceleration of 2.1g, in an 18 m long, 7 m wide, and 3 m tall flight volume. Our analysis and experiments demonstrated the robustness of the control design against external disturbances, making it particularly suitable for high-speed flight where significant aerodynamic effects occur. The proposed controller does not require any modeling or estimation of aerodynamic drag parameters.

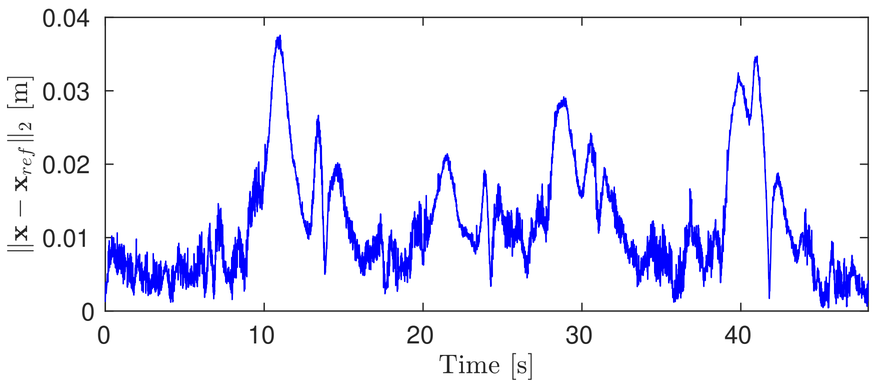


Figure 2-16: Euclidean norm of position error for hover with disturbance force through tensioned wire.

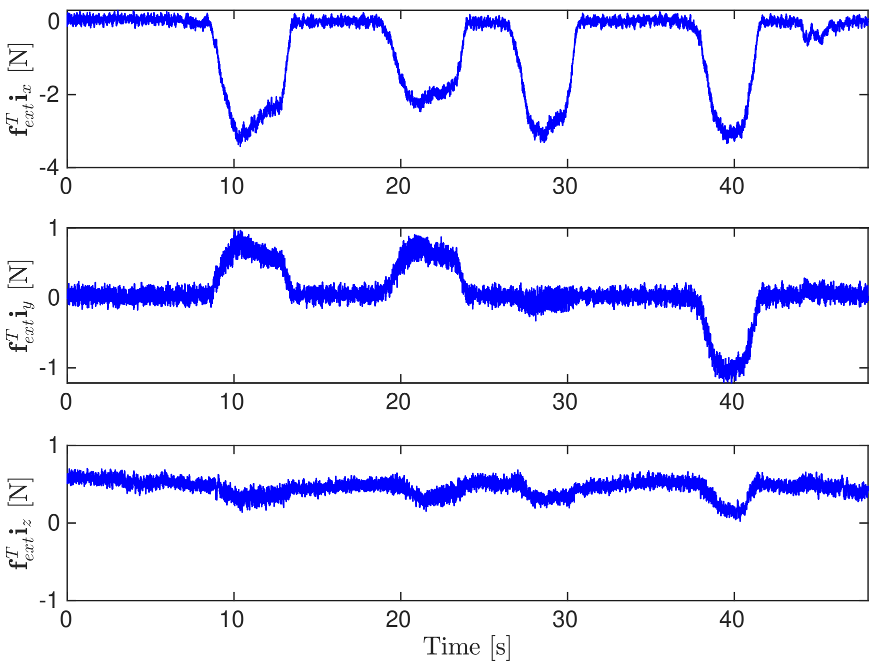
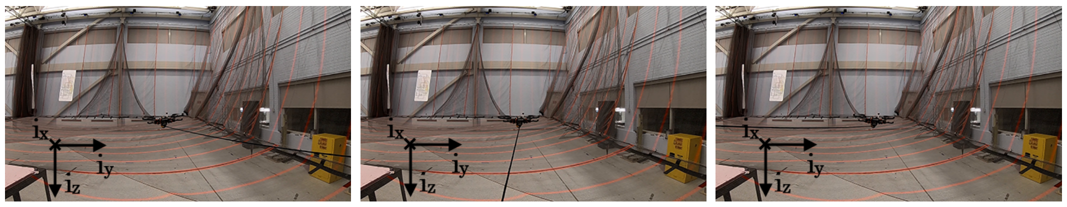


Figure 2-17: Estimated external disturbance force for hover with disturbance force through tensioned wire.



(a) Time is 22 s. (b) Time is 28 s. (c) Time is 40 s.

Figure 2-18: Quadrotor in hover with disturbance force through tensioned wire.

Chapter 3

Global Trajectory-Tracking Control for a Tailsitter Flying Wing in Agile Uncoordinated Flight

We propose a novel control law for accurate tracking of agile trajectories using a tailsitter flying wing unmanned aerial vehicle (UAV) that transitions between vertical take-off and landing (VTOL) and forward flight. Our global control formulation enables maneuvering throughout the flight envelope, including uncoordinated flight with sideslip. We show differential flatness of the nonlinear tailsitter dynamics with a simplified aerodynamics model. Using the flatness transform, the proposed controller incorporates tracking of the position reference along with its derivatives velocity, acceleration and jerk, as well as the yaw reference and yaw rate. The inclusion of jerk and yaw rate references through an angular velocity feedforward term improves tracking of trajectories with fast-changing accelerations. The control design is based on a simplified aerodynamics model that does not require extensive modeling of the aircraft dynamics. By applying incremental nonlinear dynamic inversion (INDI), the controller only depends on a local input-output relation to incrementally update control inputs, resulting in robustness against modeling inaccuracies. We achieve INDI with nonlinear dynamics inversion based on the flatness transform. The resulting control algorithm is extensively evaluated in flight tests, where it demonstrates accurate trajectory tracking and challenging agile maneuvers, such as sideways flight and aggressive transitions while turning.

This chapter is based on [126]. A video of the experiments can be found at `https://youtu.be/tGQO-6DPT1M`.

3.1 Introduction

Transitioning powered-lift aircraft combine the vertical take-off and landing (VTOL) and hover capabilities of rotorcraft with the speed and endurance of fixed-wing aircraft. Lift is generated by a powered rotor during take-off, landing, and hover flight, while a non-rotating wing generates lift during horizontal flight. There exist various design configurations that achieve powered lift. An aircraft may be equipped with dedicated lift rotors that are stopped once sufficient lift is generated by the wing. Alternatively, the orientation of the rotors may be changed from horizontally spinning to propeller configuration after take-off, like on tiltrotor and tiltwing aircraft. Tailsitter aircraft, on the other hand, rotate in their entirety during transition, so that their rotors transition between lift generation and propulsion based on the attitude of the vehicle.

While the large attitude envelope of tailsitter aircraft may render them less suitable for manned flight, their relative mechanical simplicity makes them an appealing option for unmanned aerial vehicle (UAV) applications. Tailsitter aircraft can exceed the range and endurance limitations typical of multicopters without sacrificing the capability to take-off, hover, and land in confined spaces. This combination is relevant to many applications. For example, in search and rescue, unmanned tailsitter aircraft could quickly reach remote locations using horizontal flight, and inspect structures or enter buildings in hovering flight.

A tailsitter flying wing is a tailsitter aircraft without fuselage, tail, and vertical stabilizers or control surfaces. Forgoing these structures simplifies the aerodynamic and mechanical design of the aircraft and potentially improves performance by lowering mass and aerodynamic drag. Due to the lack of vertical aerodynamic surfaces, flying wing aircraft often require active directional stabilization. The fast and relatively powerful brushless motors found on many UAVs are particularly suitable to fulfill this task through differential thrust. By placing flaps that act as elevator and aileron, i.e., *elevons*, in the rotor wash, the aircraft remains controllable throughout its flight envelope, including static hover conditions. The reduced stability of flying wing aircraft may also result in increased maneuverability. Specifically, the lack of vertical surfaces enables maneuvers such as fast skidding turns and knife edge flight where the wing points in the direction of travel. In general, it permits uncoordinated flight, where the vehicle incurs nonzero lateral velocity.

In this chapter, we propose a novel flight control algorithm that is specifically designed for tracking of agile trajectories using the tailsitter flying wing aircraft shown in Fig. 3-1. The proposed controller uses differential flatness to track the reference position, velocity, acceleration, and jerk (the third derivative of position), as well as yaw angle and yaw rate. It is based on a global formulation, without mode switching or blending, and able to exploit the entire flight envelope, including uncoordinated flight conditions, for agile maneuvering. We derive the controller based on a simplified aerodynamics model and apply incremental nonlinear dynamic inversion (INDI) to achieve accurate trajectory tracking despite model discrepancies.

We use φ-theory to model the aerodynamic force and moment [67]. The method captures dominant contributions over the entire flight envelope, including post-stall and uncoordinated flight conditions. The φ-theory model does not suffer from singularities that methods based on the angle of attack and sideslip angle may incur around hover, where these angles are not defined.

Differential flatness is a property of nonlinear dynamics systems that guarantees the existence of an equivalent controllable linear system [24–26]. The state variables and control inputs of a flat system can be expressed as the function of a (ficticious) flat output and a finite number of its time-derivatives. Using this function, trajectories can be generated in the flat output space and transformed to the state space for tracking control [74, 77]. This enables tracking higher-order derivatives of the output, which has been demonstrated to improve trajectory-tracking performance in fast and agile flight [22, 23, 104, 125]. The differential flatness property has been shown to hold for idealized aircraft dynamics [73], and for aggressive fighter maneuvers in coordinated flight [38]. For an introductory description of differential flatness, the reader is referred to Section 2.3.

Incremental, or sensor-based, nonlinear dynamic inversion is a version of nonlinear dynamic inversion (NDI) control that alleviates the lack of robustness associated with NDI [61] by incrementally updating control inputs based on inertial measurements [2, 117]. Instead of directly computing control inputs from the inverted dynamics model, it only considers the input-output relation around the current operating point and computes the required control increment relative to this point [111]. As such, it only relies on local accuracy of the dynamics model and can correct for discrepancies by further incrementing control inputs in subsequent updates. A general derivation of the INDI control law is given in Section 2.4.

Existing flight control designs for tailsitter aircraft are based on various approaches. Blending of separate controllers [54], gain scheduling [50, 68], or pre-planned transition maneuvers [11] can be used to handle the change of dynamics between hover and forward flight. However, when performing agile maneuvering at large angle of attack, the aircraft continuously enters and exits the transition regime, and it is preferable to utilize a controller without blending or switching. A global formulation for trajectory tracking in coordinated flight is proposed by [103]. The controller is based on numerical inversion of a global first-principles model, but does not account for model discrepancies, leading to a systematic pitch tracking error. Wind tunnel testing can be used to improve accuracy of the dynamics model [69, 134]. However, building an accurate model from measurements can be a time-consuming process that may need to be repeated if the controller is transferred to a different vehicle.

Robustification can be used to design a performant controller that does not rely on an accurate model of the vehicle dynamics, e.g., by using model-free control [4]. INDI has also been leveraged for robust control of various types of transitioning aircraft, such as tiltrotor aircraft [16, 97], and aircraft with dedicated lift rotors [65, 66, 95]. The additional control inputs of these configurations may lead to over-actuation, requiring specific attention to control allocation that is typically not necessary for tailsitter aircraft. On the other hand, tailsitter aircraft operate over a more extensive attitude range and at increased angle of attack, leading to distinct challenges in control design. Robustness of an INDI longitudinal flight controller for tracking pre-designed transition maneuvers is shown through analysis and simulation by [141], and an INDI attitude control design for a tailsitter with varying fuselage shape is proposed by [137]. An exhaustive framework for INDI-based linear and angular acceleration control of a tailsitter is presented by [114]. This work includes flight test results, focusing on coordinated flight at small flight path angles. Our proposed control design differs from existing INDI controllers for tailsitter aircraft in several fundamental ways. Firstly, we present a complete INDI flight control design for agile trajectory tracking, including aggressive maneuvers and uncoordinated flight. Secondly, we employ differential flatness to avoid reliance on pre-designed transition maneuvers and to accomplish feedforward jerk tracking. Thirdly, flatness enables nonlinear inversion of our dynamics model, whereas existing INDI implementations rely on local linearization to obtain inversion.

The trajectory generation algorithm by [79] utilizes differential flatness of a simplified longitudinal dynamics model to design transitions for a quadrotor biplane. The resulting trajectories are limited to forward motion and consider acceleration but no higher-order derivatives. The controller by [134] employs a pre-designed constant angular velocity feedforward input to improve transition. Theoretically, this feedforward signal corresponds to the acceleration rate of change, i.e., jerk. However, it is not applied beyond the pre-designed transition maneuver.

The main contribution of this chapter is a global control design for tracking agile trajectories using a flying wing tailsitter. Our proposed control design is novel in several ways. Firstly, we derive a differential flatness transform for the tailsitter flight dynamics with

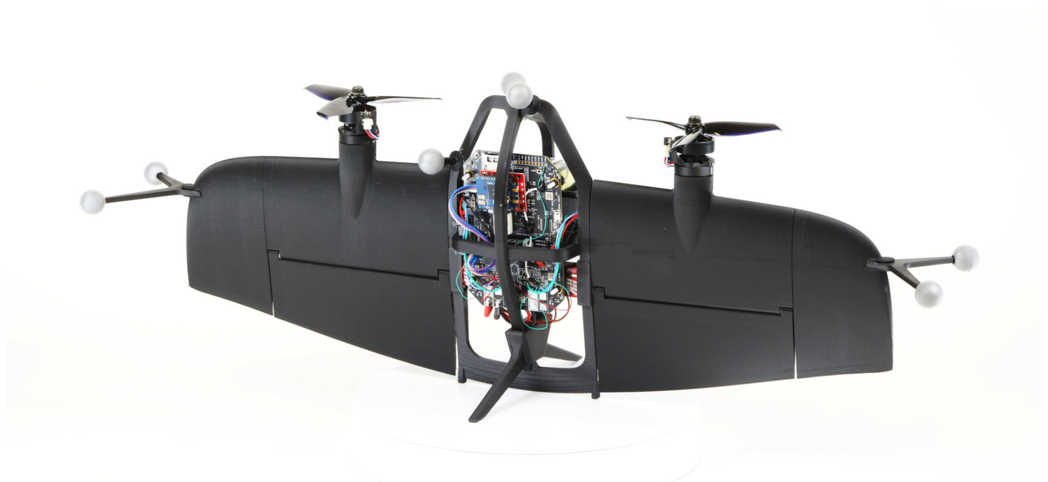


Figure 3-1: Tailsitter flying wing aircraft.

simplified φ-theory aerodynamics model. Secondly, we present a method to incorporate jerk tracking as an angular velocity feedforward input in tailsitter control design. As far as we are aware, this is the first tailsitter controller that achieves jerk tracking, making it suitable to fly agile trajectories with fast-changing acceleration references. Thirdly, we apply INDI to control a tailsitter aircraft in agile maneuvers that include large flight path angles and uncoordinated flight conditions. Our INDI control design is based on direct nonlinear inversion and, contrary to existing implementations, does not rely on linearization of the dynamics for inversion. Fourthly, we detail our methodology for analytical and experimental estimation of the φ-theory aerodynamic parameters used by the controller. Finally, we demonstrate the proposed controller in extensive flight experiments reaching up to 8 m/s in an indoor flight space measuring 18 m $\times$ 8 m. The flight experiments include agile maneuvers, such as aggressive transitions while turning, differential thrust turning, and uncoordinated flight. In order to explicitly show the advantages of incremental control and feedforward references, we also present an experimental comparison to a baseline version of our proposed control design.

The chapter is structured as follows: In Section 3.2, we provide an overview of the flight dynamics and aerodynamics model. We derive the corresponding differential flatness transform in Section 3.3. The design of the trajectory-tracking controller is presented in Section 3.4. Section 3.5 details our methodology for analytical and experimental estimation of the aerodynamic parameters used by the controller. Extensive experimental flight results are presented in Section 3.6. Table 3.1 contains the main nomenclature used in the chapter.

3.2 Flight Dynamics Model

This section provides a detailed overview of the flight dynamics model employed in our proposed control algorithm. The algorithm, described in Section 3.4, is based on the notion of incremental control action and therefore utilizes the dynamics model solely as a local approximation of the flight dynamics. Unlike conventional inversion-based controllers, it does not require a globally accurate dynamics model.

The model is employed by the incremental controller to predict (i) the change in linear

Table 3.1: Main nomenclature.

$\mathbf{a}$	linear coordinate acceleration (in world-fixed frame, unless noted otherwise), m/s^2
$\mathbf{b}_x$, $\mathbf{b}_y$, $\mathbf{b}_z$	basis vectors of body-fixed frame
c	propulsion and φ-theory aerodynamic coefficients
$\bar{c}$	Buckingham π aerodynamic coefficients
$\mathbf{f}$	force vector (in world-fixed frame, unless noted otherwise), N
g	gravitational acceleration, m/s^2
$\mathbf{i}_x$, $\mathbf{i}_y$, $\mathbf{i}_z$	standard basis vectors
$\mathbf{j}$	jerk in world-fixed frame, m/s^3
$\mathbf{J}$	vehicle moment of inertia tensor, kg m^2
$\mathbf{K}$	diagonal control gain matrix
l	flap/rotor location
m	vehicle mass, kg
$\mathbf{m}$	moment vector in body-fixed frame, Nm
$\mathbf{q}$	throttle input
$\mathbf{R}_b^i$	rotation matrix from frame b to frame i
T	total thrust, N
T_i	thrust by rotor i, N
$\mathbf{v}$	velocity (in world-fixed frame, unless noted otherwise), m/s
$\mathbf{x}$	position in world-fixed frame, m
$\bar{\alpha}$	sum of zero-lift angle of attack and thrust angle, rad
α_0	zero-lift angle of attack, rad
α_T	thrust angle, rad
$\boldsymbol{\alpha}_x$, $\boldsymbol{\alpha}_y$, $\boldsymbol{\alpha}_z$	basis vectors of zero-lift frame
δ	sum of flap deflections, rad
δ_i	deflection of flap i, rad
ΔT	differential thrust, N
θ	vehicle pitch angle, rad
$\bar{\theta}$	zero-lift reference frame pitch angle, rad
μ_i	torque by rotor i, Nm
$\boldsymbol{\xi}$	normed quaternion attitude vector
$\boldsymbol{\sigma}_{\text{ref}}$	reference trajectory
ϕ	vehicle roll angle, rad
ψ	vehicle yaw angle, rad
ω_i	angular speed of rotor i, rad/s
$\boldsymbol{\Omega}$	vehicle angular velocity in body-fixed frame, rad/s
Subscript and superscript	
b	body-fixed reference frame
c	control command
i	world-fixed reference frame, or flap/rotor index
T	thrust force or moment
w	wing force or moment
α	zero-lift reference frame
δ	flap force or moment
$\bar{\theta}$	intermediate control reference frame after pitch rotation
ϕ	intermediate control reference frame after roll rotation
ψ	intermediate control reference frame after yaw rotation
ext	unmodeled force or moment
hpf	high-pass filtered signal
lpf	low-pass filtered signal
ref	(derived from) reference trajectory

acceleration due to increments in attitude and collective thrust, and (ii) the change in angular acceleration due to increments in differential thrust and flap deflections. By inversion of these relationships, the control algorithm computes the increments required to attain the commanded changes in linear and angular acceleration. In order to maintain analytical invertibility and avoid undue complexity, the dynamics model omits any contributions that do not directly affect the aforementioned incremental relationships. For example, the velocity of the aircraft relative to the atmosphere may result in a significant aerodynamic moment. However, when compared to the fast dynamics of the motors and servos controlling the propellers and flaps, this moment contribution is relatively slow-changing, as it relates to the orientation and velocity of the entire vehicle. Consequently, it is assumed to be constant between control updates and does not need to be included in the incremental dynamics model. It is nonetheless accounted for in the control algorithm together with other unmodeled contributions to linear and angular acceleration through inertial measurement feedback, as described in Section 3.4.

3.2.1 Reference Frame Conventions

Figure 3-2 depicts the world and body-fixed reference frames used in the dynamics model and flight controller. The basis of the world-fixed north-east-down (NED) reference frame consists of the columns of the identity matrix $[\mathbf{i}_x \ \mathbf{i}_y \ \mathbf{i}_z]$. We define the basis of the body-fixed reference frame as the vectors $\mathbf{b}_x$, which coincides with the chord line and the wing symmetry plane; $\mathbf{b}_y$, which is perpendicular to this symmetry plane; and $\mathbf{b}_z$, which is defined to satisfy the right-hand rule. These vectors form the rotation matrix $\mathbf{R}_b^i = [\mathbf{b}_x \ \mathbf{b}_y \ \mathbf{b}_z] \in SO(3)$, which gives the transformation from the body-fixed reference frame (indicated by the subscript b) to the world-fixed reference frame (indicated by the superscript i). To avoid confusion when referring to rotations in the body-fixed reference frame, we use the terms in Fig. 3-2a irrespective of vehicle orientation. We note that the term *yaw* is also used to refer to rotation around the world-fixed $\mathbf{i}_z$-axis in the context of the reference trajectory defined in Section 3.3.

The zero-lift axis system, depicted in Fig. 3-2b, is obtained by rotating the body-fixed axis system around its negative $\mathbf{b}_y$-axis by the zero-lift angle of attack α_0, which is defined as the angle of attack for which the aircraft produces zero lift. For symmetric airfoils $\alpha_0 = 0$, while most cambered airfoils have $\alpha_0 < 0$. Finally, the thrust angle α_T is defined as the angle of the thrust line with regard to the $\mathbf{b}_x$-$\mathbf{b}_y$ plane. Typically, motors are slightly tilted down, leading to $\alpha_T < 0$.

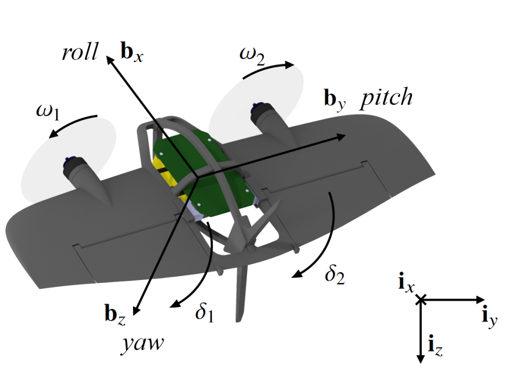
3.2.2 Vehicle Equations of Motion

The vehicle translational dynamics are given by

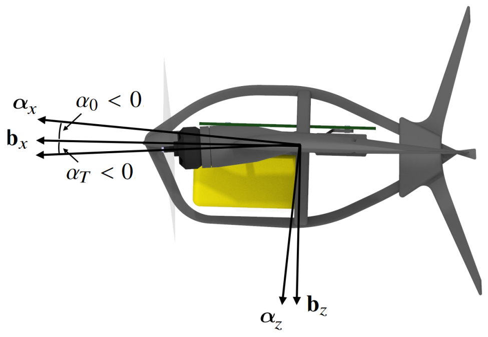
$$\dot{\mathbf{x}} = \mathbf{v}, \tag{3.1}$$

$$\dot{\mathbf{v}} = g\mathbf{i}_z + m^{-1}\left(\mathbf{R}_\alpha^i \mathbf{f}^\alpha + \mathbf{f}_{\text{ext}}\right), \tag{3.2}$$

where $\mathbf{x}$ and $\mathbf{v}$ are respectively the vehicle position and velocity in the world-fixed reference frame, g is the gravitational acceleration, and m is the vehicle mass. The vector $\mathbf{f}^\alpha$ represents the modeled aerodynamic and thrust force in the zero-lift reference frame. Any unmodeled forces are represented by the external force vector $\mathbf{f}_{\text{ext}}$, which is defined in the world-fixed reference frame.



(a) Body-fixed reference frame, control inputs, and angular motion terminology.



(b) Zero-lift reference frame, zero-lift angle of attack, and thrust angle.

Figure 3-2: Reference frame and control input conventions.

The rotational dynamics are given by

$$\dot{\boldsymbol{\xi}} = \frac{1}{2}\boldsymbol{\xi} \circ \boldsymbol{\Omega}, \tag{3.3}$$

$$\dot{\boldsymbol{\Omega}} = \mathbf{J}^{-1}(\mathbf{m} + \mathbf{m}_{\text{ext}} - \boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega}), \tag{3.4}$$

where $\boldsymbol{\Omega}$ is the angular velocity in the body-fixed reference frame, and $\boldsymbol{\xi}$ is the normed quaternion attitude vector. The Hamilton quaternion product is denoted by $\circ$, such that $\mathbf{v}^b = \mathbf{R}_i^b \mathbf{v} = \boldsymbol{\xi}^{-1} \circ \mathbf{v} \circ \boldsymbol{\xi}$. The matrix $\mathbf{J}$ is the vehicle moment of inertia tensor, and $\mathbf{m}$ represents the aerodynamic and thrust moment in the body-fixed reference frame. The external moment vector $\mathbf{m}_{\text{ext}}$ represents unmodeled moment contributions, similar to the force vector $\mathbf{f}_{\text{ext}}$. The final term of (3.4) accounts for the conservation of angular momentum. We note that the term *external* with regard to $\mathbf{f}_{\text{ext}}$ and $\mathbf{m}_{\text{ext}}$ refers to unmodeled force and moment contributions, i.e., external to the model, but not necessarily due to physically external influences, such as gusts.

3.2.3 Aerodynamic and Thrust Force and Moment

We employ φ-theory parametrization to model the aerodynamic force and moment [67]. This parametrization has several advantages over standard expressions for aerodynamic coefficients. Firstly, it provides a simple global model that includes dominant contributions over the entire flight envelope, including post-stall conditions. Our simplified model relies on only nine scalar aerodynamic coefficients: two for the wing, two for the flaps, two for the propellers, and three for propeller-wing interaction. Secondly, φ-theory parametrization avoids the singularities that methods based on angle of attack and sideslip angle incur near hover conditions, where these angles are undefined.

We obtain the force in the zero-lift axis system by summing contributions due to thrust, flaps, and wings. The thrust force is given by

$$\mathbf{f}_T^\alpha = \sum_{i=1}^{2} \underbrace{\begin{bmatrix} \cos\bar{\alpha}\,(1 - c_{D_T}) \\ 0 \\ \sin\bar{\alpha}\,(c_{L_T} - 1) \end{bmatrix} T_i}_{\mathbf{f}_{T_i}^\alpha}, \tag{3.5}$$

where $\bar{\alpha} = \alpha_0 + \alpha_T$, T_i is the thrust due to motor i, and the coefficients c_{D_T} and c_{L_T} represent drag and lift due to thrust vector components in the zero-lift axis system, respectively. The motor thrust is computed as follows:

$$T_i = c_T \omega_i^2 \quad \text{with} \;\; i = 1, 2, \tag{3.6}$$

where $\omega_i \geq 0$ is the speed of motor i. The thrust coefficient c_T is a function of propeller geometry and can be obtained from bench tests using a force balance. Intuitively, c_{D_T} mostly represents the loss of propeller efficiency due to the presence of the wing in the propwash, while c_{L_T} represents the propwash-induced lift. Note that the lift component vanishes if the thrust line coincides with the zero-lift axis, i.e., $\alpha_0 + \alpha_T = 0$. For convenience, all aerodynamic coefficients incorporate the effects of air density. If the coefficients are applied for flight in significantly varying conditions, their values may be recomputed using a simple scaling with air density, as shown in Section 3.5. The force contribution by the flaps is given by

$$\mathbf{f}_\delta^\alpha = \sum_{i=1}^{2} \underbrace{- \begin{bmatrix} 0 \\ 0 \\ c_{L_T}^\delta \cos\bar{\alpha}\; T_i + c_{L_V}^\delta \,\|\mathbf{v}\| \mathbf{i}_x^\top \mathbf{v}^\alpha \end{bmatrix} \delta_i}_{\mathbf{f}_{\delta_i}^\alpha}, \tag{3.7}$$

where δ_i is the deflection angle of flap i. The first term of (3.7), scaled with the coefficient $c_{L_T}^\delta$, is the flap lift due to the prop-wash induced airspeed. The second term, scaled with $c_{L_V}^\delta$, is the flap lift due to the airspeed along the zero-lift line. Finally, the wing force contribution is obtained as

$$\mathbf{f}_w^\alpha = - \begin{bmatrix} c_{D_V} \mathbf{i}_x^\top \mathbf{v}^\alpha \\ 0 \\ c_{L_V} \mathbf{i}_z^\top \mathbf{v}^\alpha \end{bmatrix} \|\mathbf{v}\|, \tag{3.8}$$

where c_{D_V} and c_{L_V} are the wing drag and lift coefficients, respectively. The total force in the zero-lift axis system is now obtained as

$$\mathbf{f}^\alpha = \mathbf{f}_T^\alpha + \mathbf{f}_\delta^\alpha + \mathbf{f}_w^\alpha. \tag{3.9}$$

We note that (3.9) does not contain any lateral force component. Due to the lack of a fuselage and vertical tail surface, the lateral force is relatively much smaller than the lift and drag components. Any incurred lateral force is captured in the unmodeled force $\mathbf{f}_{\text{ext}}$, and accounted for by the controller through accelerometer feedback, as described in Section 3.4.2.

The moment is obtained by summation of contributions due to motor thrust and torque, and flap deflections. We ignore the wing moment due to velocity, attitude, and rotation rates, as these state variables are relatively slow-changing compared to the motor speeds and flap deflections. The corresponding contributions are incorporated in the unmodeled moment $\mathbf{m}_{\text{ext}}$ and accounted for through angular acceleration feedback. The moment due to motor thrust is given by

$$\mathbf{m}_T = \begin{bmatrix} l_{T_y}\mathbf{i}_z^\top \mathbf{R}_\alpha^b(\mathbf{f}_{T_2}^\alpha - \mathbf{f}_{T_1}^\alpha) \\ c_{\mu_T}(T_1 + T_2) \\ l_{T_y}\mathbf{i}_x^\top \mathbf{R}_\alpha^b(\mathbf{f}_{T_1}^\alpha - \mathbf{f}_{T_2}^\alpha) \end{bmatrix}, \tag{3.10}$$

where l_{T_y} is the absolute distance along $\mathbf{b}_y$ between the vehicle center of gravity and each motor, and c_{μ_T} is the pitch moment coefficient due to thrust. The moment due to motor torque is obtained as follows:

$$\mathbf{m}_\mu = \begin{bmatrix} \cos\alpha_T \\ 0 \\ -\sin\alpha_T \end{bmatrix} \sum_{i=1}^{2} \mu_i, \tag{3.11}$$

where μ_i is the motor torque around the thrust-fixed x-axis given by

$$\mu_i = -(-1)^i c_\mu \omega_i^2 \tag{3.12}$$

with c_μ the propeller torque coefficient. The signs in (3.12) correspond to the rotation directions as defined in Fig. 3-2b. The flap contribution to the aerodynamic moment is given by

$$\mathbf{m}_\delta = \begin{bmatrix} l_{\delta_y}\cos\alpha_0\, \mathbf{i}_z^\top(\mathbf{f}_{\delta_2}^\alpha - \mathbf{f}_{\delta_1}^\alpha) \\ l_{\delta_x}\mathbf{i}_z^\top(\mathbf{f}_{\delta_1}^\alpha + \mathbf{f}_{\delta_2}^\alpha) \\ l_{\delta_y}\sin\alpha_0\, \mathbf{i}_z^\top(\mathbf{f}_{\delta_2}^\alpha - \mathbf{f}_{\delta_1}^\alpha) \end{bmatrix}, \tag{3.13}$$

where l_{δ_y} is the absolute distance between the vehicle center of gravity and each flap center along the $\mathbf{b}_y$ axis, and l_{δ_x} is the distance from this axis to the aerodynamic center of both flaps. The total aerodynamic moment can now be obtained by summing the contributions, as follows:

$$\mathbf{m} = \mathbf{m}_T + \mathbf{m}_\mu + \mathbf{m}_\delta. \tag{3.14}$$

3.3 Differential Flatness

The purpose of our control design is to accurately track the trajectory reference

$$\boldsymbol{\sigma}_{\text{ref}}(t) = [\mathbf{x}_{\text{ref}}(t)^\top\ \psi_{\text{ref}}(t)]^\top, \tag{3.15}$$

which consists of four elements: The vehicle position in the world-fixed reference frame $\mathbf{x}_{\text{ref}}(t) \in \mathbb{R}^3$, and the yaw angle $\psi_{\text{ref}}(t) \in \mathbb{T}$, where $\mathbb{T}$ denotes the circle group. The reference $\boldsymbol{\sigma}_{\text{ref}}(t)$ may be provided by a pre-planned trajectory or by an online motion planning algorithm. Henceforward, we do not explicitly write the time argument t everywhere. For dynamic feasibility, it is required that the position reference $\mathbf{x}_{\text{ref}}$ is at least fourth-order continuous, and the yaw reference ψ_{ref} is at least second-order continuous, as shown in Section 3.3.4. By taking the derivative of $\mathbf{x}_{\text{ref}}$, we obtain continuous references for velocity $\mathbf{v}_{\text{ref}}$, acceleration $\mathbf{a}_{\text{ref}}$, and jerk $\mathbf{j}_{\text{ref}}$. Similarly, we obtain a continuous yaw rate reference $\dot{\psi}_{\text{ref}}$ from the yaw reference ψ_{ref}.

Differential flatness of a nonlinear dynamics system entails the existence of an equivalent controllable linear system via a specific type of feedback linearization. For further details on differential flatness and its applications in general, we refer the reader to Section 2.3

and [24–26]. An important property of flat systems is that their state and input variables can be directly expressed as a function of the flat output and a finite number of its derivatives. This property is of major importance when developing trajectory generation and tracking algorithms, as it allows one to readily obtain state and input trajectories corresponding to an output trajectory, effectively transforming the output tracking problem into a state tracking problem. In practice, the state trajectory can serve as a feedforward control input that enables tracking of higher-order derivatives of the flat output. Inclusion of these feedforward inputs improves trajectory tracking performance by reducing the phase lag in response to rapid changes in the flat output.

In this section, we show differential flatness of the dynamics system described in Section 3.2—with some simplifications—by deriving expressions of the state and control inputs as a function of the flat output defined by (3.15). The expression for angular velocity is used in our trajectory-tracking controller to obtain a feedforward input based on the reference jerk and yaw rate. Expressions for attitude and the control inputs are used for linear and angular acceleration control, respectively.

3.3.1 Attitude and Collective Thrust

The position and velocity states are trivially obtained from (3.15). We arrive at expressions for the attitude and collective thrust by rewriting (3.2) as

$$\mathbf{f}^i = m\left(\mathbf{a} - g\mathbf{i}_z\right) - \mathbf{f}_{\text{ext}}, \tag{3.16}$$

where we assume that $\mathbf{f}_{\text{ext}}$ is constant. In practice $\mathbf{f}_{\text{ext}}$ may not be constant, but it is implicitly estimated and corrected for by incremental control, as described in Section 3.4.2. Given (3.16), the vehicle attitude and collective thrust are uniquely defined by three major constraints:

(i) the yaw angle reference ψ,

(ii) the fact that $\mathbf{i}_y^\top \mathbf{f}^\alpha = 0$ according to (3.5), and

(iii) the forces in the vehicle symmetry plane, i.e., $\mathbf{i}_x^\top \mathbf{f}^\alpha$ and $\mathbf{i}_z^\top \mathbf{f}^\alpha$.

Additionally, we exploit the continuity of yaw and the fact that the collective thrust must be non-negative.

In this section, we express the attitude using Euler angles ψ, ϕ, and θ in ZXY rotation sequence. The angle symbols are also used to refer to rotation matrices between intermediate frames, e.g., the rotation matrix $\mathbf{R}_i^\phi$ represents the rotations by ψ and ϕ. The ZXY Euler angles form a valid and universal attitude representation that is suitable for the flat transform because each of the angles is uniquely defined by one of the constraints, as shown in Fig. 3-3. In order to avoid the well-known issues with Euler angles, we convert the obtained attitude to quaternion format before it is used by the flight controller.

We define the yaw angle ψ as the angle between the world-fixed $\mathbf{i}_y$-axis and the projection of the body-fixed $\mathbf{b}_y$-axis onto the horizontal plane, i.e., the plane perpendicular to $\mathbf{i}_z$. While this angle is undefined if the wingtips are pointing straight up/down (i.e., $\mathbf{i}_z^\top \mathbf{b}_y = \pm 1$), we avoid ambiguity by performing the yaw rotation $\psi \mathbf{i}_z$ from the identity rotation (i.e., from $\mathbf{R}_i^b = \mathbf{I}$).

Next, we satisfy constraint (ii) by rotation around the yawed x-axis $\mathbf{R}^i_\psi \mathbf{i}_x$ by

$$\phi = -\operatorname{atan2}\left(\mathbf{i}_y^\top \mathbf{R}_i^\psi \mathbf{f}^i, \mathbf{i}_z^\top \mathbf{f}^i\right) + k\pi, \tag{3.17}$$

where atan2 is the four-quadrant inverse tangent function. In the second term, $k \in \{0, 1\}$ is set such that $\mathbf{b}_y \bullet \mathbf{R}^i_\phi \mathbf{i}_y > 0$, i.e., such that the obtained y-axis corresponds as closely as possible to the current $\mathbf{b}_y$-axis. This results in the equivalence $\psi \equiv \psi + \pi$, which enables continuous yaw tracking through discontinuities, such as a roll maneuver where the yaw angle instantly switches to $\psi + \pi$. Unwanted switching does not occur due to continuity of the yaw reference. If the commanded force is entirely in the horizontal plane and perpendicular to the yaw direction, both arguments of the tangent function are zero, and any ϕ satisfies the constraint. This condition is highly unlikely to occur in actual flight, but can be resolved in practice by setting ϕ to match the current direction of $\mathbf{b}_y$ as closely as possible.

To satisfy constraint (iii), we solve (3.9) for the collective thrust $T = T_1 + T_2$ and for the rotation angle $\bar{\theta}$ around the vehicle y-axis. In order to find these expressions, we assume that the flap angles are constant and known. We can make this assumption without consequence because of a limitation of the INDI acceleration controller. As described in Section 3.4, we only consider the low-frequency component of the flap deflection when controlling the linear acceleration. This slow-changing component is virtually constant between control updates.

Since the individual thrust values are still undetermined, we assume that the difference between the steady-state flap deflections is negligible so that

$$\delta_1 T_1 + \delta_2 T_2 \approx \frac{T}{2}\delta, \tag{3.18}$$

where $\delta = \delta_1 + \delta_2$. This assumption may be violated during sustained maneuvers or flight with sideslip, but we have found that it typically does not lead to large discrepancies. We substitute into (3.9) $\mathbf{f}^\alpha = \mathbf{R}_\phi^{\bar{\theta}} \mathbf{f}^\phi$ and $\mathbf{v}^\alpha = \mathbf{R}_\phi^{\bar{\theta}} \mathbf{v}^\phi$ with $\mathbf{f}^\phi = \mathbf{R}_i^\phi \mathbf{f}^i$, $\mathbf{v}^\phi = \mathbf{R}_i^\phi \mathbf{v}^i$. Note that $\bar{\theta}$ refers to the rotation from ϕ to the zero-lift reference frame, while θ is the rotation to the body-fixed reference frame. We obtain the following two equalities:

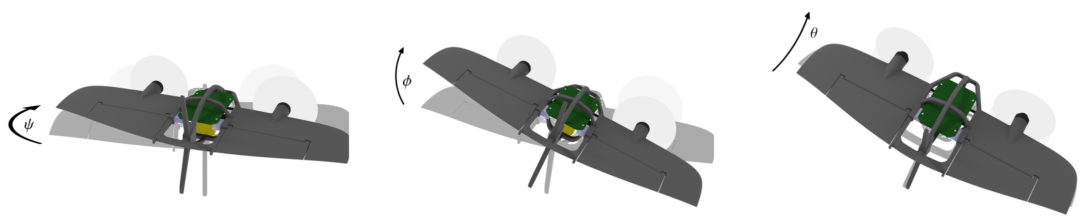
$$\mathrm{c}\bar{\alpha}\ (1 - c_{D_T})\, T - c_{D_V} \|\mathbf{v}\| \left(\mathrm{c}\bar{\theta}\ \mathbf{i}_x^\top \mathbf{v}^\phi - \mathrm{s}\bar{\theta}\ \mathbf{i}_z^\top \mathbf{v}^\phi\right) = \mathrm{c}\bar{\theta}\ \mathbf{i}_x^\top \mathbf{f}^\phi - \mathrm{s}\bar{\theta}\ \mathbf{i}_z^\top \mathbf{f}^\phi, \tag{3.19}$$

$$\begin{aligned}\left(\mathrm{s}\bar{\alpha}\ (c_{L_T} - 1) - \mathrm{c}\bar{\alpha}\ c_{L_T}^\delta \delta/2\right) T - c_{L_V}^\delta \delta \|\mathbf{v}\| \left(\mathrm{c}\bar{\theta}\ \mathbf{i}_x^\top \mathbf{v}^\phi - \mathrm{s}\bar{\theta}\ \mathbf{i}_z^\top \mathbf{v}^\phi\right) - \\ c_{L_V} \|\mathbf{v}\| \left(\mathrm{s}\bar{\theta}\ \mathbf{i}_x^\top \mathbf{v}^\phi + \mathrm{c}\bar{\theta}\ \mathbf{i}_z^\top \mathbf{v}^\phi\right) = \mathrm{s}\bar{\theta}\ \mathbf{i}_x^\top \mathbf{f}^\phi + \mathrm{c}\bar{\theta}\ \mathbf{i}_z^\top \mathbf{f}^\phi,\end{aligned} \tag{3.20}$$

where c and s represent respectively cosine and sine, and $\bar{\alpha} = \alpha_0 + \alpha_T$. Solving (3.19) and (3.20) for $\bar{\theta}$ and T gives

$$\begin{aligned}\bar{\theta} = \operatorname{atan2}\Big(&\eta\left(\mathbf{i}_x^\top \mathbf{f}^\phi + c_{D_V}\|\mathbf{v}\|\mathbf{i}_x^\top \mathbf{v}^\phi\right) - c_{L_V}^\delta \delta \|\mathbf{v}\|\mathbf{i}_x^\top \mathbf{v}^\phi - c_{L_V}\|\mathbf{v}\|\mathbf{i}_z^\top \mathbf{v}^\phi - \mathbf{i}_z^\top \mathbf{f}^\phi, \\ &\eta\left(\mathbf{i}_z^\top \mathbf{f}^\phi + c_{D_V}\|\mathbf{v}\|\mathbf{i}_z^\top \mathbf{v}^\phi\right) - c_{L_V}^\delta \delta \|\mathbf{v}\|\mathbf{i}_z^\top \mathbf{v}^\phi + c_{L_V}\|\mathbf{v}\|\mathbf{i}_x^\top \mathbf{v}^\phi + \mathbf{i}_x^\top \mathbf{f}^\phi\Big) + k\pi,\end{aligned} \tag{3.21}$$

$$T = \frac{1}{\mathrm{c}\bar{\alpha}\ (1 - c_{D_T})} \left(\mathrm{c}\bar{\theta}\ \mathbf{i}_x^\top \mathbf{f}^\phi - \mathrm{s}\bar{\theta}\ \mathbf{i}_z^\top \mathbf{f}^\phi + c_{D_V} \|\mathbf{v}\| \left(\mathrm{c}\bar{\theta}\ \mathbf{i}_x^\top \mathbf{v}^\phi - \mathrm{s}\bar{\theta}\ \mathbf{i}_z^\top \mathbf{v}^\phi\right)\right), \tag{3.22}$$



(a) Yaw rotation to satisfy the yaw reference.

(b) Roll rotation to satisfy $\mathbf{i}_y^\top \mathbf{f}^\alpha = 0$.

(c) Pitch rotation to attain $\mathbf{i}_x^\top \mathbf{f}^\alpha$ and $\mathbf{i}_z^\top \mathbf{f}^\alpha$.

Figure 3-3: Rotation sequence for attitude flatness transform.

where

$$\eta = \frac{\mathrm{s}\bar{\alpha}\ (c_{L_T} - 1) - \mathrm{c}\bar{\alpha}\ c_{L_T}^{\delta} \delta/2}{\mathrm{c}\bar{\alpha}\ (1 - c_{D_T})}. \tag{3.23}$$

is the ratio of lift and forward force due to thrust. We set $k \in \{0, 1\}$ such that $T \geq 0$. In the unlikely event that constraint (iii) is satisfied for any $\bar{\theta}$, both arguments of the atan2 function equal zero and, in practice, we can set $\bar{\theta}$ to match the current attitude as closely as possible. Since the angle $\bar{\theta}$ is the rotation to the zero-lift axis system, we use $\theta = \bar{\theta} + \alpha_0$ to obtain the corresponding rotation to the body-fixed reference frame.

Note that we purposely selected the ZXY rotation sequence and the definition of yaw, such that ϕ and θ do not affect the satisfaction of constraint (i), and θ does not affect the satisfaction of constraint (ii). Given that the Euler angles are uniquely defined (up to addition of π) by the yaw reference, (3.17), and (3.21), this implies that these expressions give the attitude as a function of $\boldsymbol{\sigma}_{\mathrm{ref}}$.

3.3.2 Angular Velocity

By taking the derivative of (3.17), we obtain

$$\dot{\phi} = -\frac{\dot{\beta}_x \beta_z - \beta_x \dot{\beta}_z}{\beta_x^2 + \beta_z^2}, \tag{3.24}$$

where β_x and β_z are respectively the first and second arguments of the atan2 function in (3.17), and

$$\dot{\beta}_x = -\mathrm{c}\psi\ \dot{\psi}\mathbf{i}_x^\top \mathbf{f}^i - \mathrm{s}\psi\ \mathbf{i}_x^\top \dot{\mathbf{f}}^i - \mathrm{s}\psi\ \dot{\psi}\mathbf{i}_y^\top \mathbf{f}^i + \mathrm{c}\psi\ \mathbf{i}_y^\top \dot{\mathbf{f}}^i, \tag{3.25}$$

$$\dot{\beta}_z = \mathbf{i}_z^\top \dot{\mathbf{f}}^i. \tag{3.26}$$

The force derivative is obtained as the derivative of (3.16), as follows:

$$\dot{\mathbf{f}}^i = m\mathbf{j}. \tag{3.27}$$

We take the derivative of (3.21) to obtain

$$\dot{\theta} = \frac{\dot{\sigma}_x \sigma_z - \sigma_x \dot{\sigma}_z}{\sigma_x^2 + \sigma_z^2}, \tag{3.28}$$

where σ_x and σ_z are respectively the first and second arguments of the atan2 function in (3.21), and

$$\dot{\sigma}_x = \eta\left(\mathbf{i}_x^\top \dot{\mathbf{f}}^\phi + c_{D_V}\tau_x\right) - c_{L_V}^\delta \delta\tau_x - c_{L_V}\tau_z - \mathbf{i}_z^\top \dot{\mathbf{f}}^\phi, \tag{3.29}$$

$$\dot{\sigma}_z = \eta\left(\mathbf{i}_z^\top \dot{\mathbf{f}}^\phi + c_{D_V}\tau_z\right) - c_{L_V}^\delta \delta\tau_z + c_{L_V}\tau_x + \mathbf{i}_x^\top \dot{\mathbf{f}}^\phi \tag{3.30}$$

with

$$\tau_x = \dot{\|\mathbf{v}\|}\mathbf{i}_x^\top \mathbf{v}^\phi + \|\mathbf{v}\|\mathbf{i}_x^\top \dot{\mathbf{v}}^\phi, \tag{3.31}$$

$$\tau_z = \dot{\|\mathbf{v}\|}\mathbf{i}_z^\top \mathbf{v}^\phi + \|\mathbf{v}\|\mathbf{i}_z^\top \dot{\mathbf{v}}^\phi \tag{3.32}$$

and

$$\dot{\|\mathbf{v}\|} = \frac{\mathbf{v}^\top \mathbf{a}}{\|\mathbf{v}\|}, \tag{3.33}$$

$$\dot{\mathbf{v}}^\phi = \dot{\mathbf{R}}_i^\phi \mathbf{v} + \mathbf{R}_i^\phi \mathbf{a}. \tag{3.34}$$

The expression for the force derivative $\dot{\mathbf{f}}^\phi$ is similar to (3.34). In the force equations, we assume that the flap deflection is known and that its temporal derivatives are negligible, as described in Section 3.3.1. Finally, we obtain the angular velocity in the body-fixed reference frame, as follows:

$$\mathbf{\Omega} = \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}_\phi^\theta \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_\psi^\theta \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}. \tag{3.35}$$

3.3.3 Control Inputs

At this point, we have expressed the state variables as a function of the flat output (3.15). To obtain an expression for the control inputs, an expression for angular acceleration is obtained as the derivative of (3.35) and substituted into (3.4) to obtain an expression for $\mathbf{m}$. The angular acceleration can be utilized as a feedforward input corresponding to snap, the fourth derivative of position, and yaw acceleration, as we showed for the quadcopter in Chapter 2. However, calculation of this feedforward input significantly complicates the controller expressions, and its benefit may be marginal given how challenging it is to perform accurate feedforward control of the angular acceleration of a fixed-wing aircraft. Hence, we do not incorporate the angular acceleration feedforward input in our control design. For the sake of completeness, we still include the corresponding expressions in Section 3.3.4.

As described in Section 3.4.4, our control design obtains a moment command using INDI. To find the corresponding control inputs, i.e., flap deflections and differential thrust $\Delta T = T_1 - T_2$, we solve (3.14) for these inputs. We find an expression for the differential thrust ΔT by equating

$$\mathbf{i}_z^\top (\mathbf{m}_T + \mathbf{m}_\mu) = \mathbf{i}_z^\top \mathbf{m}, \tag{3.36}$$

which assumes that the contribution by $\mathbf{i}_z^\top \mathbf{m}_\delta$ is negligible. Due to the multiplication with $\sin\alpha_0$, this assumption typically does not result in significant discrepancies. Using

$\mu_1 + \mu_2 = {}^{c_\mu}\!/_{c_T} \Delta T$, we obtain

$$\Delta T = \frac{\mathbf{i}_z^\top \mathbf{m}}{l_{T_y}\left(\mathrm{c}\alpha_0\ \mathrm{c}\bar{\alpha}\left(1 - c_{D_T}\right) - \mathrm{s}\alpha_0\ \mathrm{s}\bar{\alpha}\left(c_{L_T} - 1\right)\right) - \mathrm{s}\alpha_T\ \frac{c_\mu}{c_T}}. \tag{3.37}$$

The individual thrust values are then given by

$$T_1 = \frac{T + \Delta T}{2}, \qquad T_2 = \frac{T - \Delta T}{2}. \tag{3.38}$$

After obtaining the motor speeds from (3.6), we can deduct $\mathbf{m}_T$ and $\mathbf{m}_\mu$ from $\mathbf{m}$ to obtain $\mathbf{m}_\delta$. Finally, the flap deflections are computed by inversion of (3.13), as follows:

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} -l_{\delta_y}\,\mathrm{c}\alpha_0\ \nu_1 & l_{\delta_y}\,\mathrm{c}\alpha_0\ \nu_2 \\ l_{\delta_x}\nu_1 & l_{\delta_x}\nu_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{i}_x^\top \mathbf{m}_\delta \\ \mathbf{i}_y^\top \mathbf{m}_\delta \end{bmatrix} \tag{3.39}$$

with

$$\nu_i = -c_{L_T}^\delta \cos\bar{\alpha}\ T_i - c_{L_V}^\delta \|\mathbf{v}\| \mathbf{i}_x^\top \mathbf{v}^\alpha. \tag{3.40}$$

3.3.4 Angular Acceleration

In this section, we derive an expression for the angular acceleration as a function of reference snap, i.e., the fourth temporal derivative of position, and yaw acceleration. The transform can be used to obtain an angular acceleration feedforward input. Along with the expressions given in Section 3.3, it can also be used to obtain the control inputs, i.e., the motor speeds and flap deflections, required to track a given trajectory $\boldsymbol{\sigma}_{\mathrm{ref}}(t)$. This second property will be utilized in Chapter 5.

By taking the derivative of (3.24), we obtain the following expression for the roll acceleration

$$\ddot{\phi} = -\frac{\left(\ddot{\beta}_x\beta_z - \beta_x\ddot{\beta}_z\right)\left(\beta_x^2 + \beta_z^2\right) - \left(\dot{\beta}_x\beta_z - \beta_x\dot{\beta}_z\right)\left(2\beta_x\dot{\beta}_x + 2\beta_z\dot{\beta}_z\right)}{\left(\beta_x^2 + \beta_z^2\right)^2}, \tag{3.41}$$

where β_x and β_z are respectively the first and second arguments of the atan2 function in (3.17). Their first derivatives are given by (3.25) and (3.26). We obtain the second derivatives from these expressions as

$$\begin{aligned} \ddot{\beta}_x = {} & \left(\mathrm{s}\psi\ \dot{\psi}^2 - \mathrm{c}\psi\ \ddot{\psi}\right)\mathbf{i}_x^\top \mathbf{f}^i - 2\,\mathrm{c}\psi\ \dot{\psi}\mathbf{i}_x^\top \dot{\mathbf{f}}^i - \mathrm{s}\psi\ \mathbf{i}_x^\top \ddot{\mathbf{f}}^i \\ & - \left(\mathrm{c}\psi\ \dot{\psi}^2 + \mathrm{s}\psi\ \ddot{\psi}\right)\mathbf{i}_y^\top \mathbf{f}^i - 2\,\mathrm{s}\psi\ \dot{\psi}\mathbf{i}_y^\top \dot{\mathbf{f}}^i + \mathrm{c}\psi\ \mathbf{i}_y^\top \ddot{\mathbf{f}}^i, \end{aligned} \tag{3.42}$$

$$\ddot{\beta}_z = \mathbf{i}_z^\top \ddot{\mathbf{f}}^i, \tag{3.43}$$

where the second force derivative is a function of snap

$$\ddot{\mathbf{f}}^i = m\mathbf{s}. \tag{3.44}$$

Similarly, by taking the derivative of (3.28) we obtain the pitch acceleration

$$\ddot{\theta} = \frac{\left(\ddot{\sigma}_x\sigma_z - \sigma_x\ddot{\sigma}_z\right)\left(\sigma_x^2 + \sigma_z^2\right) - \left(\dot{\sigma}_x\sigma_z - \sigma_x\dot{\sigma}_z\right)\left(2\sigma_x\dot{\sigma}_x + 2\sigma_z\dot{\sigma}_z\right)}{\left(\sigma_x^2 + \sigma_z^2\right)^2}, \tag{3.45}$$

where σ_x and σ_z are respectively the first and second arguments of the atan2 function in (3.21). Their first derivatives are given by (3.29) and (3.30), and their second derivatives are

$$\ddot{\sigma}_x = \eta \left(\mathbf{i}_x^\top \ddot{\mathbf{f}}^\phi + c_{D_V} \dot{\tau}_x \right) - c_{L_V}^\delta \delta \dot{\tau}_x - c_{L_V} \dot{\tau}_z - \mathbf{i}_z^\top \ddot{\mathbf{f}}^\phi, \tag{3.46}$$

$$\ddot{\sigma}_z = \eta \left(\mathbf{i}_z^\top \ddot{\mathbf{f}}^\phi + c_{D_V} \dot{\tau}_z \right) - c_{L_V}^\delta \delta \dot{\tau}_z + c_{L_V} \dot{\tau}_x + \mathbf{i}_x^\top \ddot{\mathbf{f}}^\phi \tag{3.47}$$

with

$$\dot{\tau}_x = \ddot{\|\mathbf{v}\|} \mathbf{i}_x^\top \mathbf{v}^\phi + 2 \dot{\|\mathbf{v}\|} \mathbf{i}_x^\top \dot{\mathbf{v}}^\phi + \|\mathbf{v}\| \mathbf{i}_x^\top \ddot{\mathbf{v}}^\phi, \tag{3.48}$$

$$\dot{\tau}_z = \ddot{\|\mathbf{v}\|} \mathbf{i}_z^\top \mathbf{v}^\phi + 2 \dot{\|\mathbf{v}\|} \mathbf{i}_z^\top \dot{\mathbf{v}}^\phi + \|\mathbf{v}\| \mathbf{i}_z^\top \ddot{\mathbf{v}}^\phi \tag{3.49}$$

and

$$\ddot{\|\mathbf{v}\|} = \frac{\mathbf{a}^\top \mathbf{a} + \mathbf{v}^\top \mathbf{j}}{\|\mathbf{v}\|} - \frac{\mathbf{v}^\top \mathbf{a} \dot{\|\mathbf{v}\|}}{\|\mathbf{v}\|^2}, \tag{3.50}$$

$$\ddot{\mathbf{v}}^\phi = \ddot{\mathbf{R}}_i^\phi \mathbf{v} + 2 \dot{\mathbf{R}}_i^\phi \mathbf{a} + \mathbf{R}_i^\phi \mathbf{j}. \tag{3.51}$$

The expression for the force second derivative $\ddot{\mathbf{f}}^\phi$ is similar to (3.51). In the force equations, we assume that the flap deflection is known and that its temporal derivatives are negligible, as described in Section 3.3.1. We combine the roll acceleration and pitch acceleration obtained from respectively (3.41) and (3.45) with the yaw acceleration $\ddot{\psi}$ to obtain the angular acceleration in the body-fixed reference frame. We take the derivative of (3.35) to obtain the following expression:

$$\dot{\boldsymbol{\Omega}} = \begin{bmatrix} 0 \\ \ddot{\theta} \\ 0 \end{bmatrix} + \dot{\mathbf{R}}_\phi^\theta \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_\phi^\theta \begin{bmatrix} \ddot{\phi} \\ 0 \\ 0 \end{bmatrix} + \dot{\mathbf{R}}_\psi^\theta \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \mathbf{R}_\psi^\theta \begin{bmatrix} 0 \\ 0 \\ \ddot{\psi} \end{bmatrix}. \tag{3.52}$$

We can now find the aerodynamic and thrust moment in the body-fixed reference frame by rewriting (3.4), as follows:

$$\mathbf{m} = \mathbf{J} \dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times \mathbf{J} \boldsymbol{\Omega} - \mathbf{m}_{\text{ext}}. \tag{3.53}$$

In practice, the unmodeled moment $\mathbf{m}_{\text{ext}}$ is implicitly estimated and corrected for by incremental control, as described in Section 3.4.4. Based on the moment obtained from (3.53) and the total thrust obtained from (3.22), the required control inputs can be calculated using the expressions given in Section 3.3.3. Note that—since the aerodynamic and thrust moment cannot instantaneously change—dynamic feasibility of $\boldsymbol{\sigma}_{\text{ref}}$ requires continuity of (3.53), and thereby at least fourth-order continuity of the position reference $\mathbf{x}_{\text{ref}}$ and at least second-order continuity of the yaw reference ψ_{ref}.

3.4 Trajectory-Tracking Control

Our proposed controller is designed to accurately track the dynamic position reference $\boldsymbol{\sigma}_{\text{ref}}$. It consists of several components based on various control methodologies, as shown in Fig. 3-4. Each component employs a global formulation that enables seamless maneuvering throughout the flight envelope. By separating kinematics and dynamics, we are able to

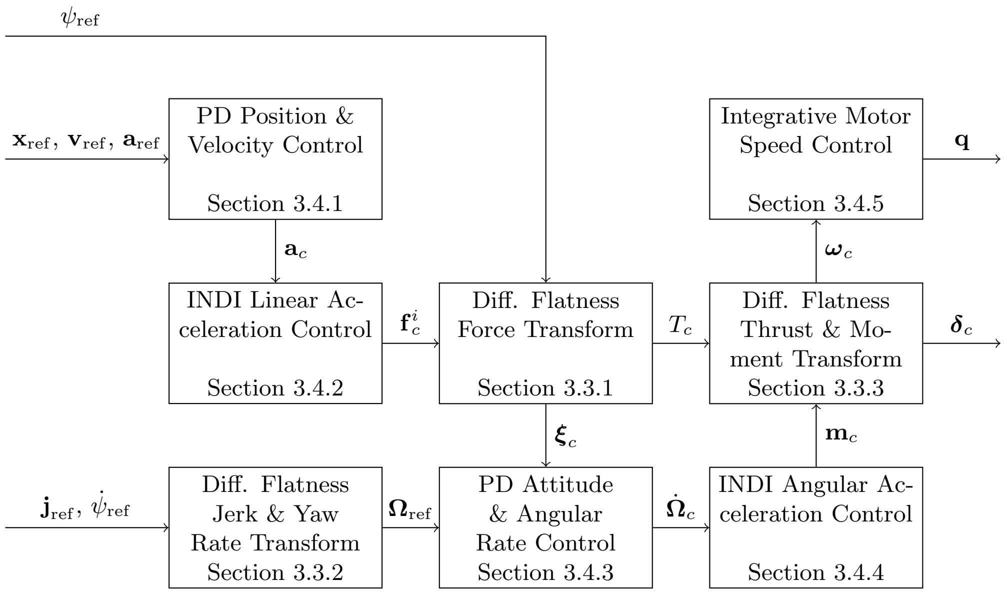

ψ_{ref}
$\mathbf{x}_{\text{ref}}, \mathbf{v}_{\text{ref}}, \mathbf{a}_{\text{ref}}$
PD Position & Velocity Control
Section 3.4.1
$\mathbf{a}_c$
INDI Linear Acceleration Control
Section 3.4.2
$\mathbf{f}_c^i$
Diff. Flatness Force Transform
Section 3.3.1
T_c
Diff. Flatness Thrust & Moment Transform
Section 3.3.3
$\boldsymbol{\delta}_c$
Integrative Motor Speed Control
Section 3.4.5
$\mathbf{q}$
$\boldsymbol{\omega}_c$
$\boldsymbol{\xi}_c$
$\mathbf{m}_c$
$\mathbf{j}_{\text{ref}}, \dot{\psi}_{\text{ref}}$
Diff. Flatness Jerk & Yaw Rate Transform
Section 3.3.2
$\boldsymbol{\Omega}_{\text{ref}}$
PD Attitude & Angular Rate Control
Section 3.4.3
$\dot{\boldsymbol{\Omega}}_c$
INDI Angular Acceleration Control
Section 3.4.4

Figure 3-4: Overview of trajectory-tracking control architecture.

employ proportional-derivative (PD) control on the translational and rotational kinematics. Application of the resulting linear and angular acceleration commands is performed using INDI control. INDI enables accurate control by incremental adjustment of control inputs, based on the inverted dynamics model derived in Section 3.3. Due to its incremental formulation, the controller only depends on local accuracy of the input-output relation, resulting in favorable robustness against modeling errors and external disturbances. As we will detail in this section, these errors and disturbances (i.e., $\mathbf{f}_{\text{ext}}$ and $\mathbf{m}_{\text{ext}}$) are implicitly estimated and corrected for based on the difference between sensor-based and model-based force and moment estimates. By directly incorporating linear and angular acceleration measurements to obtain the sensor-based estimates, the controller is able to quickly and wholly counteract errors and disturbances, without relying on integral action.

Our proposed control design uses a state estimate consisting of position $\mathbf{x}$, velocity $\mathbf{v}$, and attitude $\boldsymbol{\xi}$. Additionally, linear acceleration $\mathbf{a}$ and angular velocity $\boldsymbol{\Omega}$ measurements in the body-fixed reference frame are obtained from the inertial measurement unit (IMU). Motor speeds $\boldsymbol{\omega}$ and flap deflections $\boldsymbol{\delta}$ are measured and utilized as well.

3.4.1 PD Position and Velocity Control

We use cascaded proportional-derivative (PD) controllers for position and velocity control, resulting in the following expression for the acceleration command:

$$\mathbf{a}_c = \mathbf{R}_b^i \left(\mathbf{K}_{\mathbf{x}} \mathbf{R}_i^b (\mathbf{x}_{\text{ref}} - \mathbf{x}) + \mathbf{K}_{\mathbf{v}} \mathbf{R}_i^b (\mathbf{v}_{\text{ref}} - \mathbf{v}) + \mathbf{K}_{\mathbf{a}} \mathbf{R}_i^b (\mathbf{a}_{\text{ref}} - \tilde{\mathbf{a}}_{\text{lpf}}) \right) + \mathbf{a}_{\text{ref}} \tag{3.54}$$

with $\mathbf{K}_\bullet$ indicating diagonal gain matrices. The first term of (3.54) aims to null the position and velocity errors, while the second term is a feedforward input that ensures the acceleration reference is accurately tracked. Since the vehicle has different acceleration capabilities along its body-fixed axes, we define the control gains in the body-fixed reference frame and transform them to the world-fixed frame for each control update.

The gravity-corrected linear acceleration in the world-fixed reference frame is obtained as

$$\mathbf{a}_{\text{lpf}} = (\mathbf{R}_b^i \mathbf{a}^b + g\mathbf{i}_z)_{\text{lpf}}, \tag{3.55}$$

where lpf indicates low-pass filtering that is applied to IMU measurements to alleviate measurement noise, e.g., due to vibrations. We follow the method by [114] and deduct acceleration contributions due to the transient flap movements. This correction helps eliminate pitch oscillations that may result from the non-minimum phase response of acceleration to flap deflections. To isolate transient movement, we first filter the measured flap deflections using the low-pass filter and then using a high-pass filter, resulting in a band-pass filtered signal. The low-pass filter helps to match the phase delay between accelerometer and flap deflection measurements, and ensures that we do not (re-)introduce high-frequency noise in the resulting acceleration signal

$$\tilde{\mathbf{a}}_{\text{lpf}} = \mathbf{a}_{\text{lpf}} - m^{-1}\mathbf{R}_\alpha^i \mathbf{f}^\alpha_{\delta_{\text{hpf}}}. \tag{3.56}$$

3.4.2 INDI Linear Acceleration Control

INDI control incrementally updates the attitude and collective thrust to track the acceleration command $\mathbf{a}_c$. The controller estimates the unmodeled force $\mathbf{f}_{\text{ext}}$ by comparing the measured acceleration to the expected acceleration according to the vehicle aerodynamics model and motor speed measurements. By rewriting (3.2), we obtain

$$\mathbf{f}_{\text{ext}} = m\left(\tilde{\mathbf{a}}_{\text{lpf}} - g\mathbf{i}_z\right) - \mathbf{R}_\alpha^i \mathbf{f}^\alpha_{\text{lpf}}, \tag{3.57}$$

where, for consistency with $\tilde{\mathbf{a}}_{\text{lpf}}$, low-pass filtered motor speeds and the filtered flap deflections without transient component are used in the computation of $\mathbf{f}^\alpha_{\text{lpf}}$ according to (3.9). Substitution of (3.57) into (3.2) gives

$$\begin{aligned} \mathbf{a} &= g\mathbf{i}_z + m^{-1}\left(\mathbf{R}_\alpha^i \mathbf{f}^\alpha + \mathbf{f}_{\text{ext}}\right) \\ &= g\mathbf{i}_z + m^{-1}\left(\mathbf{R}_\alpha^i \mathbf{f}^\alpha + \left(m\left(\tilde{\mathbf{a}}_{\text{lpf}} - g\mathbf{i}_z\right) - \mathbf{R}_\alpha^i \mathbf{f}^\alpha_{\text{lpf}}\right)\right) \\ &= \tilde{\mathbf{a}}_{\text{lpf}} + m^{-1}\left(\mathbf{f}^i - \mathbf{f}^i_{\text{lpf}}\right). \end{aligned} \tag{3.58}$$

Solving (3.58) for $\mathbf{f}^i$ gives an incremental expression for the force command that corresponds to the commanded acceleration, as follows:

$$\mathbf{f}^i_c = m(\mathbf{a}_c - \tilde{\mathbf{a}}_{\text{lpf}}) + \mathbf{f}^i_{\text{lpf}}. \tag{3.59}$$

This incremental control law enables the controller to achieve the commanded acceleration despite potential modeling discrepancies and external forces, without depending on integral action. If the commanded acceleration is not yet attained, the force command will be adjusted further in subsequent control updates until the first term in (3.59) vanishes. Based on the force command $\mathbf{f}^i_c$, the commanded attitude $\boldsymbol{\xi}_c$ is obtained from ψ_{ref}, (3.17) and (3.21), and the collective thrust command T_c is obtained from (3.22). Note that it is through the flatness transform described in Section 3.3 that our INDI algorithm can perform fully nonlinear inversion, without linearization of the dynamics in (3.58) (see Section 2.4.3). The nonlinear inversion provides more accurate control commands when large acceleration deviations occur, such as may happen during aggressive maneuvers with quickly changing acceleration references.

3.4.3 PD Attitude and Angular Rate Control

Given the extensive attitude envelope of the tailsitter vehicle, our attitude controller employs quaternion representation to avoid kinematic singularities. The attitude error quaternion is obtained as

$$\boldsymbol{\xi}_e = \boldsymbol{\xi}^{-1} \circ \boldsymbol{\xi}_c, \tag{3.60}$$

and the corresponding three-element error angle vector is given by

$$\boldsymbol{\zeta}_e = \frac{2 \arccos \xi_e^w}{\sqrt{1 - \xi_e^w \xi_e^w}} \begin{bmatrix} \xi_e^x & \xi_e^y & \xi_e^z \end{bmatrix}^\top, \tag{3.61}$$

where the superscript refers to individual components of $\boldsymbol{\xi}_e$. The angular acceleration command is obtained using the PD controller

$$\dot{\boldsymbol{\Omega}}_c = \mathbf{K}_{\boldsymbol{\xi}} \boldsymbol{\zeta}_e + \mathbf{K}_{\boldsymbol{\Omega}} \left(\boldsymbol{\Omega}_{\text{ref}} - \boldsymbol{\Omega}_{\text{lpf}} \right), \tag{3.62}$$

where $\boldsymbol{\Omega}_{\text{lpf}}$ is the low-pass filtered angular velocity measurement from the IMU, and $\boldsymbol{\Omega}_{\text{ref}}$ is the feedforward angular velocity reference obtained by (3.35) based on $\dot{\psi}_{\text{ref}}$ and $\mathbf{j}_{\text{ref}}$. By including this feedforward jerk term, the controller improves trajectory-tracking accuracy, especially on agile trajectories with fast-changing acceleration references.

3.4.4 INDI Angular Acceleration Control

The angular acceleration controller has a similar construction as its linear acceleration counterpart described in Section 3.4.2. By rewriting (3.4), we obtain the following expression for the unmodeled moment:

$$\mathbf{m}_{\text{ext}} = \mathbf{J} \dot{\boldsymbol{\Omega}}_{\text{lpf}} - \mathbf{m}_{\text{lpf}} + \boldsymbol{\Omega}_{\text{lpf}} \times \mathbf{J} \boldsymbol{\Omega}_{\text{lpf}}, \tag{3.63}$$

where $\dot{\boldsymbol{\Omega}}_{\text{lpf}}$ is obtained by numerical differentiation of $\boldsymbol{\Omega}_{\text{lpf}}$, and $\mathbf{m}_{\text{lpf}}$ is calculated using (3.14) and based on the low-pass filtered flap deflection and motor speed measurements. Substitution of (3.63) into (3.4) gives

$$\begin{aligned} \dot{\boldsymbol{\Omega}} &= \mathbf{J}^{-1} \left(\mathbf{m} + \mathbf{m}_{\text{ext}} - \boldsymbol{\Omega} \times \mathbf{J} \boldsymbol{\Omega} \right) \\ &= \mathbf{J}^{-1} \Big(\mathbf{m} + \Big(\mathbf{J} \dot{\boldsymbol{\Omega}}_{\text{lpf}} - \mathbf{m}_{\text{lpf}} + \boldsymbol{\Omega}_{\text{lpf}} \times \mathbf{J} \boldsymbol{\Omega}_{\text{lpf}} \Big) - \boldsymbol{\Omega} \times \mathbf{J} \boldsymbol{\Omega} \Big) \\ &= \dot{\boldsymbol{\Omega}}_{\text{lpf}} + \mathbf{J}^{-1} \left(\mathbf{m} - \mathbf{m}_{\text{lpf}} \right), \end{aligned} \tag{3.64}$$

where it is assumed that the angular momentum term is relatively slow changing, so that the difference with its filtered version may be neglected. Solving (3.64) for $\mathbf{m}$ gives the incremental control law

$$\mathbf{m}_c = \mathbf{J} (\dot{\boldsymbol{\Omega}}_c - \dot{\boldsymbol{\Omega}}_{\text{lpf}}) + \mathbf{m}_{\text{lpf}}. \tag{3.65}$$

Based on the commanded moment $\mathbf{m}_c$, the thrust and flap deflection commands can now be calculated by (3.37) and (3.38), and (3.39), respectively. Finally, the commanded motor speeds $\boldsymbol{\omega}_c$ are calculated by inversion of (3.6).

3.4.5 Integrative Motor Speed Control

While the flaps are controlled by servos equipped with closed-loop position control, the propellers are driven by brushless motors that cannot directly track motor speed commands. Instead, we use the second-order polynomial p to find the corresponding throttle input. This function was obtained from regression analysis of static test data relating motor speed to throttle input. We add integral action to account for changes due to the fluctuating battery voltage, so that the throttle command that is sent to the motor electronic speed controller (ESC) is obtained as

$$q_i = p(\omega_{i,c}) + k_{I_\omega} \int \omega_{i,c} - \omega_i \;\; dt, \tag{3.66}$$

where k_{I_ω} is the integrator gain.

3.5 Estimation of Aerodynamic Parameters

The controller utilizes several mass, geometric, and aerodynamic properties of the vehicle. The vehicle mass m, moment of inertia $\mathbf{J}$, motor position l_{T_y}, motor incidence angle α_T, and flap position l_{δ_y} can be measured using standard methods or determined based on the design. The propeller thrust coefficient c_T, torque coefficient c_μ, and throttle response curve p are obtained using static bench tests. For a simple wing without twist, the zero-lift angle of attack α_0 is determined by the airfoil and can thus be obtained from literature or two-dimensional analysis.

Table 3.2: Tailsitter aircraft properties.

Property	Symbol	Value	
Airfoil lift slope	$\bar{c}_{l_\alpha}$	5.73	rad^{-1}
Wing surface area	S	0.070	m^2
Aspect ratio	AR	4.3	
Taper ratio	λ	0.59	
Flap chord ratio	c_f/c	0.5	
Propeller diameter	D	0.13	m
Thrust angle	α_T	$-5\frac{\pi}{180}$	rad
Monoplane circulation coefficient	τ	0.14	
Span efficiency factor	e	0.97	

What remains are the φ-theory aerodynamic coefficients c_{L_V}, c_{D_V}, c_{L_T}, c_{D_T}, $c^\delta_{L_V}$, $c^\delta_{L_T}$, and c_{μ_T}, as well as the position of the aerodynamic center of the flaps l_{δ_x}. Instead of relying on extensive analysis (e.g., through computational fluid dynamics (CFD) or wind tunnel tests), we initially estimate these parameters using back-of-the-envelope calculations and then refine them using flight test data.

3.5.1 Analytical Model

We approximate the φ-theory coefficients based on conventional Buckingham π dimensionless aerodynamic coefficients obtained using Prandtl's classical lifting-line theory. In order to avoid confusion, we use $\bar{c}$ to denote the conventional Buckingham π coefficients, whereas the φ-theory and propulsion coefficients used by the model described in Section 3.2 are written without an overbar. The lift slope of a finite wing without twist or sweep is given by

$$\bar{c}_{L_\alpha} = \frac{\bar{c}_{l_\alpha}}{1 + \frac{\bar{c}_{l_\alpha}}{\pi AR}(1+\tau)}, \tag{3.67}$$

where τ is a function of the Fourier coefficients used to represent the circulation distribution [34]. In symmetric flight, small angles of attack can be approximated as

$$\alpha - \alpha_0 = \frac{\mathbf{i}_z^\top \mathbf{v}^\alpha}{\|\mathbf{v}\|}. \tag{3.68}$$

By substituting (3.68) into (3.67) and equating to the expression in (3.8), we obtain

$$\frac{1}{2}\rho\|\mathbf{v}\|^2 S\bar{c}_{L_\alpha}(\alpha - \alpha_0) = \frac{1}{2}\rho S \frac{\bar{c}_{l_\alpha}}{1 + \frac{\bar{c}_{l_\alpha}}{\pi AR}(1+\tau)} \mathbf{i}_z^\top \mathbf{v}^\alpha \|\mathbf{v}\| = c_{L_V} \mathbf{i}_z^\top \mathbf{v}^\alpha \|\mathbf{v}\| \Rightarrow$$

$$c_{L_V} = \frac{1}{2}\rho S \frac{\bar{c}_{l_\alpha}}{1 + \frac{\bar{c}_{l_\alpha}}{\pi AR}(1+\tau)}, \tag{3.69}$$

where ρ is the air density. We note that the coefficient c_{L_V} is not dimensionless, as described in Section 3.2.3. The drag coefficient c_{D_V} represents the force along the zero-lift axis. Since we assume inviscid flow, this force is fully lift-induced and due to the tilting of the lift vector with regard to the zero-lift line by

$$\alpha_{\text{eff}} = \alpha - \alpha_0 - \alpha_i, \tag{3.70}$$

where α_{eff} is the effective angle of attack and α_i is the induced angle of attack due to downwash. The lift vector is perpendicular to the local velocity, so that its dimensionless component along the zero-lift axis (in positive α_x direction) is given by

$$\bar{c}_A = \bar{c}_L \alpha_{\text{eff}} = \bar{c}_L(\alpha - \alpha_0 - \alpha_i) = \bar{c}_L(\alpha - \alpha_0) - \bar{c}_{D_i} = \left(\bar{c}_{L_\alpha} - \frac{\bar{c}_{L_\alpha}^2}{\pi ARe}\right)(\alpha - \alpha_0)^2, \tag{3.71}$$

where $\bar{c}_{D_i}$ is the induced drag coefficient and we again use small angle assumptions. The span efficiency factor $e \leq 1$ is a function of the taper ratio and aspect ratio and represents the deviation from a perfectly elliptical lift distribution [78]. For anything but very small aspect ratios, the factor on the righthand side of (3.71) is positive, resulting in $\bar{c}_A > 0$. This means that we can expect a forward force along the zero-lift axis, especially at low speeds where drag is dominated by the lift-induced contribution. The format of (3.71) does not correspond to the c_{D_V} component of (3.8), since the latter is meant to model viscous drag contributions. At the relatively low speeds that can be achieved in our indoor flight area, viscous drag is likely small, so we leave $c_{D_V} = 0$ kg/m for now.

From momentum disc theory, we obtain

$$T_i = \frac{\pi D^2}{4}\frac{\rho}{2}v_{e,i}^2, \tag{3.72}$$

where D is the propeller diameter, $v_{e,i}$ is the downstream velocity of propeller i, and we assume negligible upstream velocity. For simplicity, we assume that the wake of each propeller results in a uniform velocity over a third of the corresponding semi-wing at a geometric angle of attack of $-\alpha_T$. In reality, the flow is not uniform and significant interaction between the wing and propeller may occur [62]. In order to keep the control algorithm tractable, we neglect these effects in our dynamics model. Substituting (3.72) into the lefthand side of (3.69) and equating to the corresponding component of (3.5) gives

$$-\sum_i T_i \frac{2}{3}\frac{S}{\pi D^2}\bar{c}_{L_\alpha}\bar{\alpha} = -\sum_i T_i c_{L_T}\bar{\alpha} \Rightarrow c_{L_T} = \frac{2}{3}\frac{S}{\pi D^2}\bar{c}_{L_\alpha}, \tag{3.73}$$

where we assume small $\bar{\alpha}$. We set $c_{D_T} = 0$, similar to c_{D_V}.

Flap deflection changes the lift coefficient in two major ways: change in angle of attack and change in airfoil camber [1]. For simplicity, we only consider the change in angle of attack due to flap deflection, so that for small angles

$$\frac{1}{2}\rho\|\mathbf{v}\|^2 S\bar{c}_{L_\alpha}\frac{c_f}{2c}\sum_i \delta_i = c_{L_V}^\delta \|\mathbf{v}\|\mathbf{i}_x^\top \mathbf{v}^\alpha \sum_i \delta_i \Rightarrow c_{L_V}^\delta = \frac{1}{2}\rho S\bar{c}_{L_\alpha}\frac{c_f}{2c} = \frac{c_f}{2c}c_{L_V} \tag{3.74}$$

where c is the wing chord, c_f is the flap chord, and the division by 2 in the righthand side is due to the fact that each flap is attached to half of the wing. Similarly, we obtain $c_{L_T}^\delta$ based on c_{L_T}, as follows

$$c_{L_T}^\delta = \frac{c_f}{c}c_{L_T}. \tag{3.75}$$

Wing properties corresponding to the tailsitter aircraft shown in Fig. 3-1 are given in Table 3.2. Using these values and $\rho = 1.225$ kg/m^3 for standard sea-level conditions, we obtain the aerodynamic parameters given in the middle column of Table 3.3. We set the thrust pitch moment coefficient c_{μ_T} to zero for now, because its analytical estimation is complicated by the propeller-wing interaction. The flap pitch moment effectiveness l_{δ_x} is set to 0.075 m based on the location of the flap quarter-chord point.

Table 3.3: Aerodynamic force parameters.

Parameter	Analytical	Experimental	
c_{L_V}	0.17	0.29	kg/m
c_{D_V}	0	0	kg/m
c_{L_T}	3.4	2.23	
c_{D_T}	0	0	
$c_{L_V}^\delta$	0.041	0.18	kg/m
$c_{L_T}^\delta$	1.7	1.25	

3.5.2 Experimental Data

We found that our control algorithm is able to stabilize the tailsitter in flight when using the analytically obtained aerodynamic parameters from Table 3.3. The incremental nature of the algorithm enables the controller to compensate for discrepancies in the dynamics model, such as inaccurate aerodynamics parameters. Despite the fact that the parameters were obtained using small angle of attack assumptions, we found that they also enable stable flight at large angles of attack and in static hover. A description of the experimental setup is given in Section 3.6.1.

We use experimental data to improve our estimate of the aerodynamic parameters. Specifically, we use the fact that the force contributions in the zero-lift reference frame

$\mathbf{f}_T^\alpha$, $\mathbf{f}_\delta^\alpha$, and $\mathbf{f}_w^\alpha$ are linear in the parameters. This allows us to formulate the parameter estimation problem as a multiple linear regression. For the drag coefficients we have

$$\mathbf{i}_x^\top \mathbf{f}^\alpha - \cos\bar{\alpha} \sum_i T_i = \left[\begin{array}{cc} c_{D_V} & c_{D_T} \end{array} \right] \left[\begin{array}{c} -\mathbf{i}_x^\top \mathbf{v}^\alpha \|\mathbf{v}\| \\ -\cos\bar{\alpha} \sum_i T_i \end{array} \right], \tag{3.76}$$

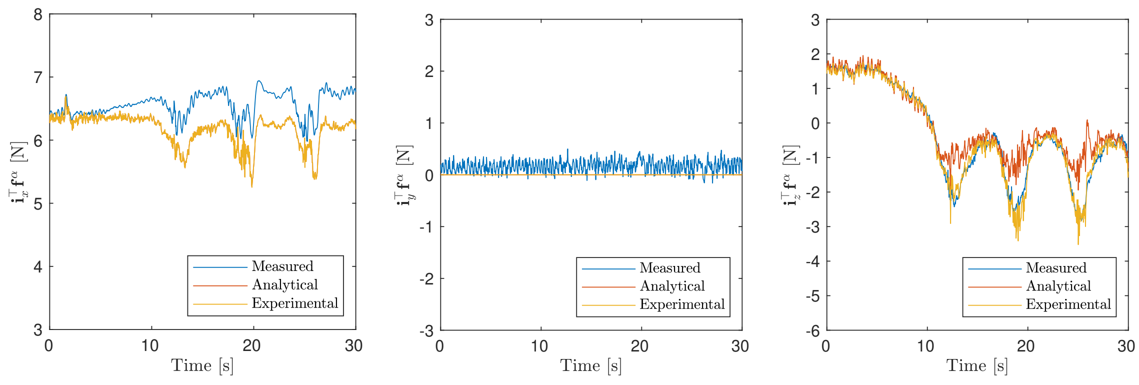
and for the lift coefficients we have

$$\mathbf{i}_z^\top \mathbf{f}^\alpha + \sin\bar{\alpha} \sum_i T_i = \left[\begin{array}{cccc} c_{L_V} & c_{L_T} & c_{L_V}^\delta & c_{L_T}^\delta \end{array} \right] \left[\begin{array}{c} -\mathbf{i}_z^\top \mathbf{v}^\alpha \|\mathbf{v}\| \\ \sin\bar{\alpha} \sum_i T_i \\ -\mathbf{i}_x^\top \mathbf{v}^\alpha \|\mathbf{v}\| \sum_i \delta_i \\ -\cos\bar{\alpha} \sum_i \delta_i T_i \end{array} \right]. \tag{3.77}$$

We perform a flight with coordinated turns in both directions. The flight starts from static hover, after which the vehicle reaches speeds up to 4.5 m/s in between turns. We estimate the forces in the zero-lift reference frame based on gravity-corrected acceleration measurements.

The force measurements are shown in Fig. 3-5, along with estimates based on analytical and experimental estimates of the aerodynamic parameters. Since we set both c_{D_V} and c_{D_T} to zero, the analytical force estimate along $\boldsymbol{\alpha}_x$, shown in Fig. 3-5a, consists solely of the direct thrust contribution. At the beginning of the trajectory, where the vehicle is in static hover, this analytical estimate closely matches the measured force, meaning that the presence of the wing has no significant effect on the thrust magnitude and c_{D_T} is indeed close to zero. As the speed increases, the measured forward force increases beyond the analytical thrust force estimate. This may be due to (a combination of) various reasons. The actual thrust may be underestimated because of increasing efficiency of the high-pitch propellers as the blade angle of attack is reduced in forward flight. Another reason may be the lift-induced forward force given by (3.71). Possibly, this forward force eclipses the parasitic drag force acting in the opposite direction at the speeds we achieve in the indoor flight space. Regardless of its exact cause, the increasing forward force drives us to set c_{D_V} to zero. When flying at higher speeds, it may be possible to obtain an accurate nonzero estimate of c_{D_V} using the regression equation. We found that, in practice, the discrepancy in the forward force estimate has little influence on controller performance. It is closely aligned with the thrust force and can thus be counteracted quickly and accurately by the incremental controller. The lateral force component, shown in Fig. 3-5b, is close to zero, as expected. The small bias in its measurement may be due to an imbalance or misalignment. The measured and estimated force component along $\boldsymbol{\alpha}_z$ is shown in Fig. 3-5c. It can be seen that the estimate based on the analytical parameters has a significant deviation at increased speeds. The estimate based on the experimental regression parameters closely matches the measurements throughout the trajectory with an R^2 value of 0.97. The resulting parameters are given in the final column of Table 3.3 and have similar order of magnitude as the corresponding analytical parameters. The analytical underestimation of $c_{L_V}^\delta$ may be because the change of camber due to flap deflection is not considered.

We also attempted to use multiple linear regression on the pitch moment $\mathbf{i}_y^\top \mathbf{m}$ to obtain experimental estimates for c_{μ_T} and l_{δ_x}. This approach did not result in consistent parameters and good moment predictions; most likely due to significant moment contributions that are not captured by the simplified aerodynamic model, e.g., due to airspeed, angle of attack, and angular rates. Modeling these contributions is not required for our control



(a) Component along $\boldsymbol{\alpha}_x$. (b) Component along $\boldsymbol{\alpha}_y$. (c) Component along $\boldsymbol{\alpha}_z$.

Figure 3-5: Forces in zero-lift reference frame based on measurements, and on analytical and experimental estimates of aerodynamic parameters. The analytical and experimental lines coincide in (a) and (b).

design, which relies only on an incremental expression that relates the change in moment to changes in differential thrust and flap deflections. However, their absence in the aerodynamic model makes accurate estimation of c_{μ_T} and l_{δ_x} more difficult. We found that the initial estimate of $l_{\delta_x} = 0.075$ m results in satisfactory pitch control, so we leave this value unchanged. Finally, we set c_{μ_T} based on the trim flap deflections in static hover, as follows:

$$c_{\mu_T} = \frac{\delta}{2} l_{\delta_x} c_{L_T}^{\delta} \cos \bar{\alpha}. \tag{3.78}$$

For our vehicle, we found $\delta/2 = -0.27$ rad, resulting in $c_{\mu_T} = -0.025$ m.

The flight test results presented in Section 3.6 were obtained using the experimental parameter estimates. We found that these parameters result in improved trajectory-tracking performance when compared to the analytical parameters. As expected, the difference is most significant at increased speeds where the discrepancy between the force estimates is largest. Specifically, we found that altitude oscillations may occur due to the underestimated c_{L_V} and $c_{L_V}^{\delta}$ analytical parameters.

3.6 Experimental Results

In this section, we evaluate controller performance on various trajectories that include challenging flight conditions, such as large accelerations, transition on curved trajectories, and uncoordinated flight. We also present a comparison to a baseline version of the controller that illustrates the advantage and necessity of feedforward jerk and yaw rate tracking and of robustification using incremental control. A video of the experiments can be found at `https://youtu.be/tGQO-6DPT1M`.

3.6.1 Experimental Setup

Experiments were performed in an 18 m × 8 m indoor flight space using the tailsitter vehicle shown in Fig. 3-1 and described in [7]. The vehicle is 3D-printed using Onyx filament with carbon fiber reinforcement. It weighs 0.7 kg and has a wingspan of 55 cm from tip to tip. It is equipped with two T-Motor F40 2400 KV motors with Gemfan Hulkie 5055 propellers.

The motor speeds are measured using optical encoders at one measurement per rotation, i.e., at about 200 Hz in hover. MKS HV93i micro servos are used to control the flaps. We obtain the flap deflection measurement as an analog signal by connecting a wire to the potentiometer in the servo.

The flight control algorithm runs onboard on an STM32 microcontroller using custom firmware. The microcontroller has a clock speed of 400 MHz and takes 25 μs to compute a control update at 32-bit floating point precision. Accelerometer and gyroscope measurements are obtained from an Analog Devices ADIS16475-3 IMU at 2000 Hz, and control updates are performed at the same frequency. Position and attitude measurements are provided by a motion capture tracking system at 360 Hz and sent to the vehicle over Wi-Fi with an average latency of 18 ms.

The IMU, motor speed, and flap deflection measurements are filtered using a second-order Butterworth low-pass filter with cutoff frequency of 15 Hz. The transient flap deflections are obtained by subsequent filtering using a second-order Butterworth high-pass filter with cutoff frequency of 1 Hz. The latency of motion caption data is corrected for by forward propagation of IMU measurements.

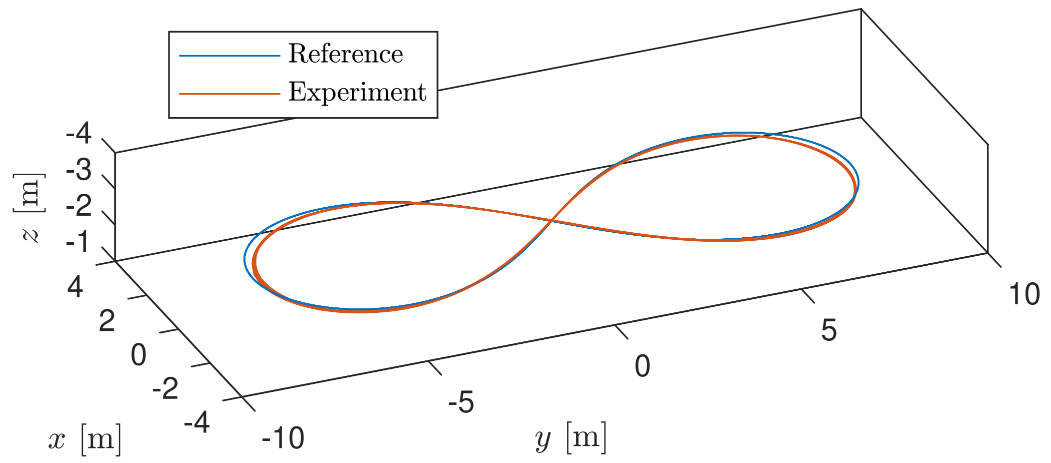
3.6.2 Lemniscate Trajectory

Figure 3-6 shows experimental results for tracking of a Lemniscate of Bernouilli with a constant speed of 6 m/s. Throughout the trajectory, ψ_{ref} is set perpendicular to the velocity, leading to coordinated flight. The reference and flown trajectories over eight consecutive laps (of 7 s each) starting at the y extreme are shown in Fig. 3-6a. It can be seen that the tracking performance is very consistent between laps. Figure 3-6b shows that the largest position deviation occurs at the end of the circular parts where the vehicle does not fully maintain the reference acceleration of almost 2 g, as can be seen in Fig. 3-6d. Over the middle part of the trajectory, the vehicle increases speed to catch up (see Fig. 3-6c), and the position error is reduced again. Overall, the controller achieves a position tracking error of 17 cm root mean square (RMS) with a maximum error of 33 cm.

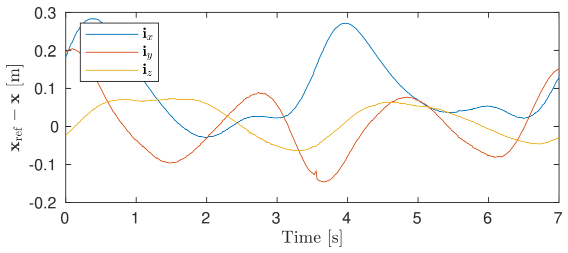
Figure 3-6e shows the commanded and flown attitude. The attitude controller uses quaternion representation, but to ease interpretation the figure uses the ZXY Euler angles described in Section 3.3. For this trajectory, ϕ corresponds to the bank angle and reaches almost 1 rad, which matches the acceleration nearing 2 g in Fig. 3-6d. The maximum attitude error occurs during a small period in the turn, where the vehicle incurs a pitch error of 6 deg. Controlling the pitch angle of a flying wing during aggressive maneuvers is generally challenging due to the lack of an elevator, and the pitch deviation is likely the cause of the relatively large position deviation at the exit of the turn. Overall, the controller is able to track the dynamic attitude command well, and it maintains an attitude error of less than 2 deg on each axis during the rest of the trajectory.

3.6.3 Knife Edge Transitioning Flight

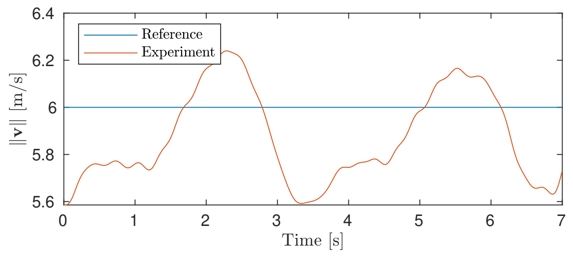
Our proposed algorithm is able to control the vehicle in uncoordinated flight conditions where it has significant lateral velocity. In knife edge flight, the wingtip is pointing in the velocity direction, leading to roll instability due to the location of the center of gravity behind the quarter span point [114]. Figure 3-7 shows experimental results for a trajectory where ψ_{ref} is set to enforce knife edge flight on one side and coordinated flight on the other side. The results show that our controller is able to stabilize the knife edge flight condition,



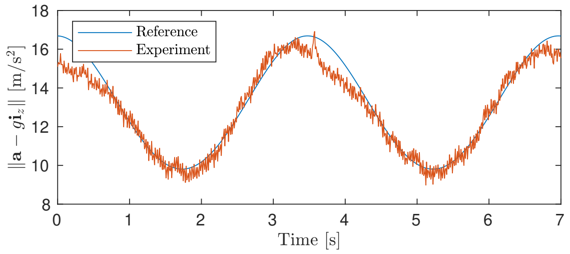
(a) Position.



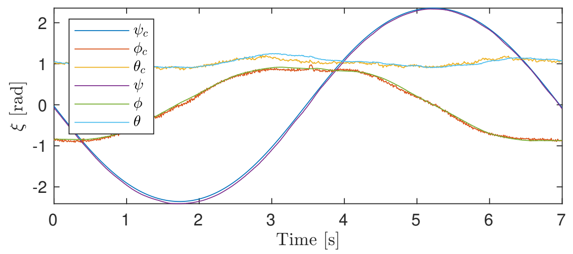
(b) Position tracking error.



(c) Speed.



(d) Acceleration.



(e) Attitude.

Figure 3-6: Experimental results for lemniscate trajectory at 6 m/s.

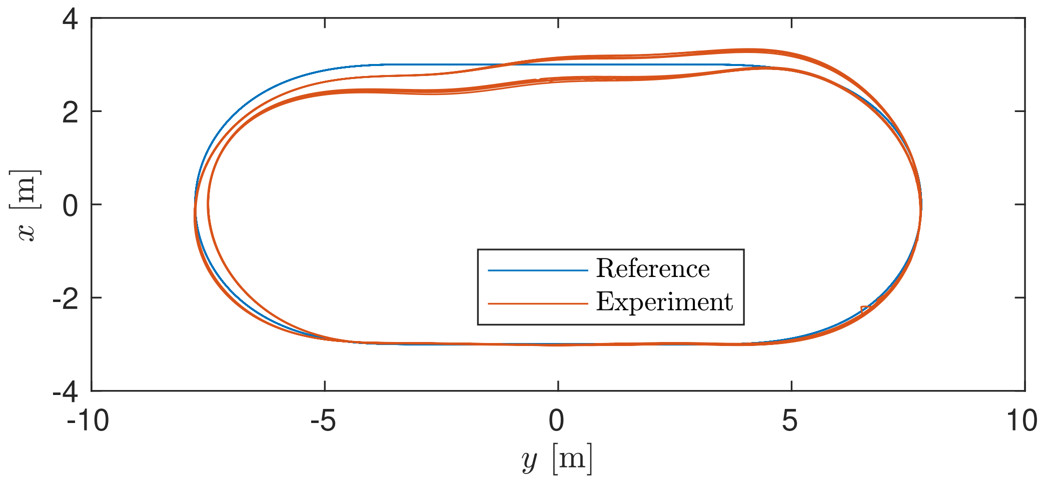
and that it is able to transition between knife edge and coordinated flight while at the same time performing a 1.6 g turn.

One lap of the oval trajectory takes 6.25 s to complete at a constant speed of 6 m/s. Referring to the view from above in Fig. 3-7a, the trajectory is flown in clockwise direction with the straight at the top in knife edge configuration and the straight at the bottom in coordinated flight. During each turn, the yaw reference is rotated by $\pi/2$ rad to enforce the switch between configurations. Consequently, the coordinated flight segment is flown in inverted orientation every other lap. Figure 3-7a shows the reference and flown position over eight successive laps, and Fig. 3-7b and Fig. 3-7c show data corresponding to two laps starting and ending at the transition from knife edge flight to coordinated flight in inverted orientation. It can be seen that the reference is followed accurately during both regular and inverted coordinated flight. Even while performing the transition from knife edge to coordinated flight, the controller is able to track the turning trajectory. At the transition from coordinated flight to knife edge, we see that the trajectory is shifted during transitions from inverted orientation. This leads to a position tracking error of at most 60 cm at the start of the knife edge segment. During transition from regular coordinated flight to knife edge a position error of at most 25 cm is incurred.

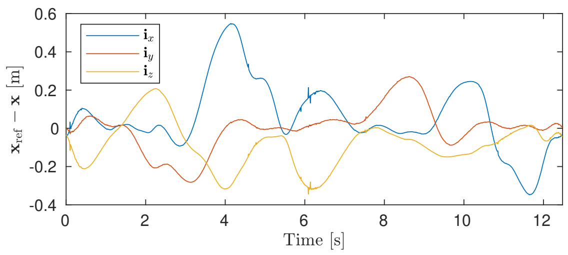
Flight during the knife edge segment is consistent and stable, and the yaw reference is tracked within 3 deg. In knife edge flight, the vehicle-fixed $\mathbf{b}_z$-axis coincides with the x-axis of Fig. 3-7a. The position deviation along this axis is due to the fact that the acceleration due to transient movement of the flaps is not considered in the position controller, as described in Section 3.4.1. The changing acceleration and jerk references result in opposite movement by the flaps at the start and end of the knife edge segment, which is why the turn preceding the knife edge segment is too tight while the turn following it is started not tight enough. Despite this, the controller achieves tracking of the position reference within 26 cm RMS and the yaw reference within 1.7 deg RMS.

3.6.4 Circular Trajectory

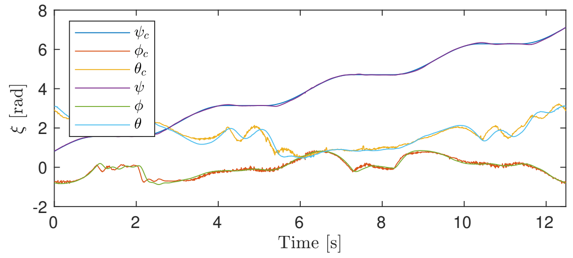
Figure 3-8 shows experimental results for tracking of a circular trajectory reference with a 3.5 m radius at a speed of 8.1 m/s. The left column of figures corresponds to coordinated flight where the $\mathbf{b}_y$-axis is perpendicular to the circle tangent, and the right column corresponds to knife edge flight where the wing tip points along the tangent of the circle. Position tracking performance is very similar between both flights. In both cases, the flown trajectory has a slightly smaller radius than the reference, reducing the flight speed to 7.8 m/s. The RMS position tracking error is 15 cm for both coordinated and knife edge flight. The aircraft reaches a continuous acceleration of 2.1 g. In coordinated flight, this requires a bank angle of 63 deg. In knife edge flight, the aircraft is rolled 14 deg towards the direction of travel to compensate for drag and pitched over by 152 deg to provide thrust towards the circle center. Maintaining this state requires flap deflections of 20 deg and rotor speeds of over 2000 rad/s in knife edge flight. In coordinated flight, the aircraft exploits the lift force to achieve circular flight more efficiently and requires rotor speeds of 1800 rad/s with flap deflections of 38 deg. In contrast to the trajectory described in Section 3.6.3, the flap deflections are almost constant during the circular trajectory. Hence, there are no transient accelerations caused by the flaps that are not considered in the position controller, and very accurate trajectory tracking is achieved in knife edge flight. This shows that our controller is not only able to stabilize the knife edge flight condition, but also attains accurate trajectory tracking in knife edge flight at speeds close to 8 m/s.



(a) Position.

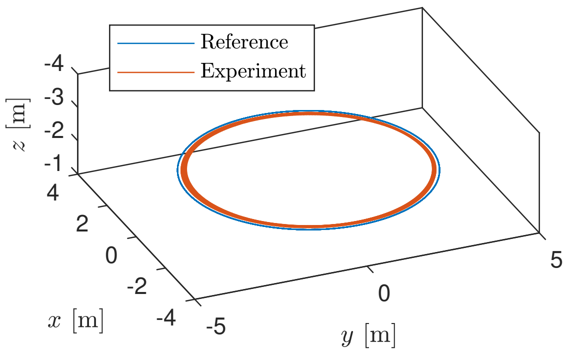


(b) Position tracking error.

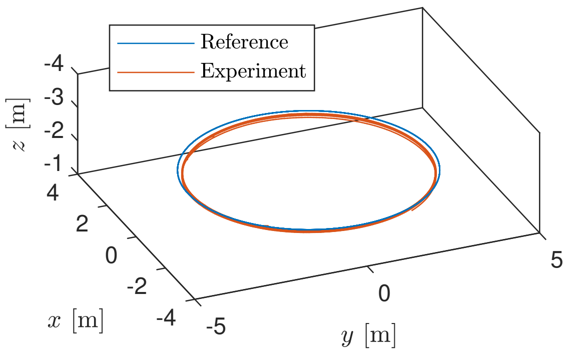


(c) Attitude.

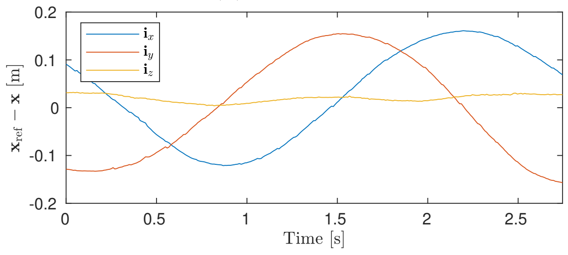
Figure 3-7: Experimental results for knife edge transitioning trajectory at 6 m/s.



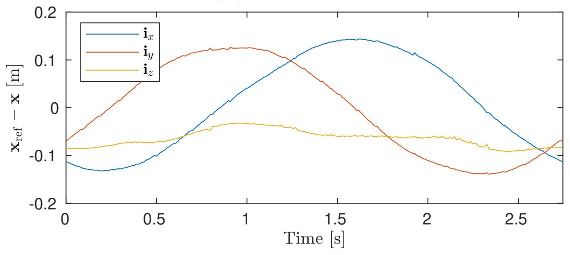
(a) Position.



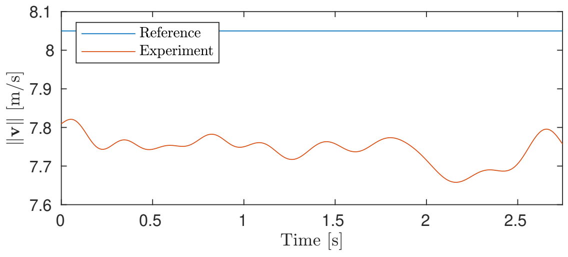
(b) Position.



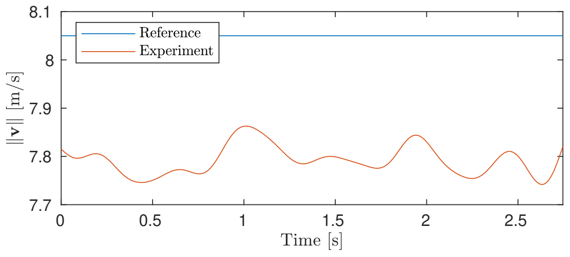
(c) Position tracking error.



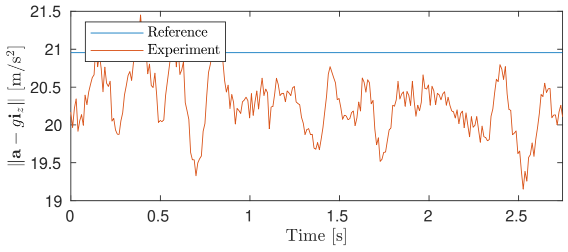
(d) Position tracking error.



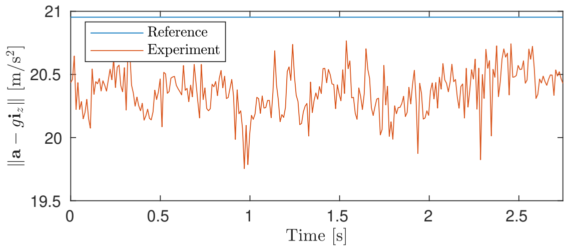
(e) Speed.



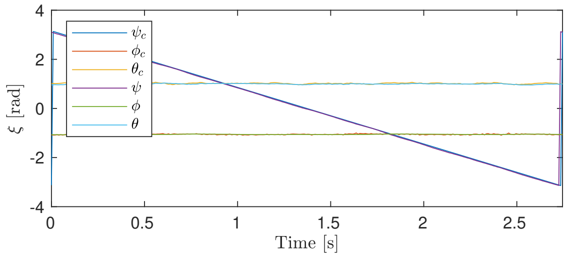
(f) Speed.



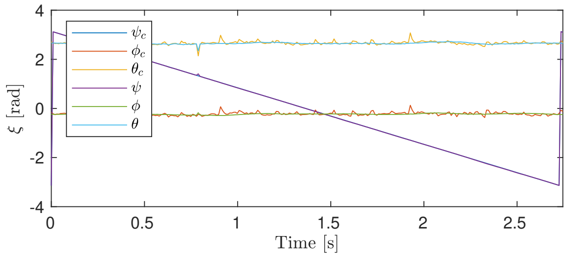
(g) Acceleration.



(h) Acceleration.



(i) Attitude.



(j) Attitude.

Figure 3-8: Experimental results for circle trajectory at 7.8 m/s for coordinated flight in (a), (c), (e), (g), and (i); and for knife edge flight (b), (d), (f), (h), and (j).

3.6.5 Transitions

Figure 3-9 shows experimental results for transitions between static hover and coordinated flight at 8 m/s on a circular trajectory with 3.5 m radius. Each transition takes 3 s at a constant tangential acceleration of 2.7 m/s^2 and is completed in 12 m. The pitch angle varies over a range of 64 deg. While transitioning from and to hover, the controller tracks the circle trajectory with respectively 10 cm and 15 cm RMS, and 15 cm and 24 cm maximum position error. These maneuvers demonstrate that the controller is capable of performing aggressive transitions while simultaneously tracking turns with significant acceleration.

To evaluate the benefits of the feedforward input based on jerk and yaw rate, we also attempted to fly the same transitions without the angular velocity reference, i.e., with $\boldsymbol{\Omega}_{\text{ref}} = \mathbf{0}$ in (3.62). We found that the controller is still able to perform 3 s transitions to speeds up to 3 m/s. However, if the target speed is increased and the corresponding tangential acceleration exceeds 1 m/s^2, the absence of the feedforward term causes large deviations from the reference trajectory to the point where the vehicle cannot be stabilized anymore. This confirms the benefit of jerk and yaw rate tracking when flying aggressive maneuvers. Intuitively, the feedforward input enables the control to anticipate future accelerations by regulating not only the forces acting on the vehicle but also their temporal derivative. Additional experiments that show the effect of feedforward control are described in Section 3.6.7.

3.6.6 Differential Thrust Turning

Since the controller is not restricted to coordinated flight, it can perform turns without banking. The tailsitter aircraft is particularly suitable for quick turns using yaw, because of the absence of any vertical surfaces and the availability of relatively powerful motors. Figure 3-10 shows a fast turn that is executed using differential thrust. The reference trajectory, entered in coordinated flight at 7 m/s, changes direction without deviating from a straight line. The controller responds with large differential thrust and flap deflections. At the onset of the turn, both flaps are almost fully deflected in opposite directions and the motors produce a differential thrust of 6.1 N. This causes the aircraft to turn at a maximum rate of 650 deg/s and point in the opposite direction within half a second, while remaining within 1 m of the position reference.

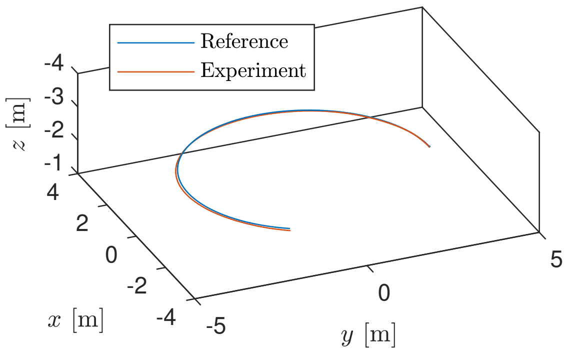
3.6.7 Comparison to Baseline Controller

In order to experimentally demonstrate the advantages offered by two key aspects of our flight control design, we compare our proposed controller to a baseline version. The baseline controller is identical to the proposed controller, except that (i) it does not incorporate feedforward jerk and yaw rate tracking, and (ii) it does not incrementally update the control inputs. More concretely, we set the feedforward angular rate reference $\boldsymbol{\Omega}_{\text{ref}}$ to zero in (3.62), and we update the force and moment commands by direct inversion of (3.2) and (3.4), respectively. These commands are thus obtained as

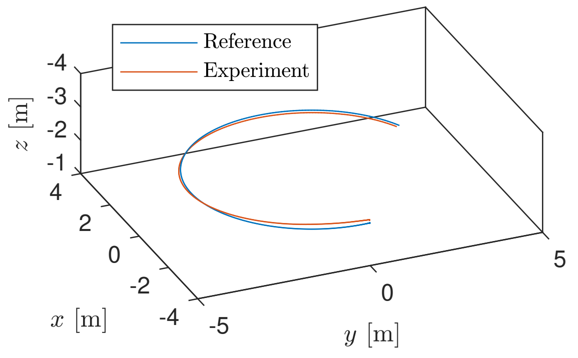
$$\mathbf{f}_c^i = m\left(\mathbf{a}_c - g\mathbf{i}_z\right), \tag{3.79}$$

$$\mathbf{m}_c = \mathbf{J}\dot{\boldsymbol{\Omega}}_c + \boldsymbol{\Omega}_{\text{lpf}} \times \mathbf{J}\boldsymbol{\Omega}_{\text{lpf}}, \tag{3.80}$$

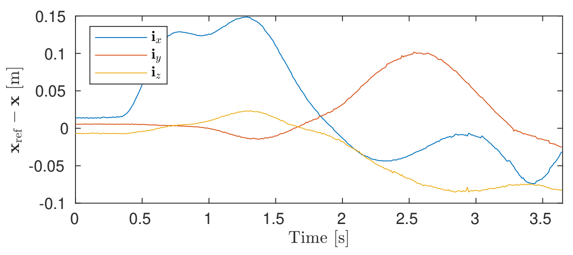
in contrast to the incremental updates (3.59) and (3.65) used in our proposed controller. We add integral action to the attitude controller (3.62) to improve the rejection of modeling



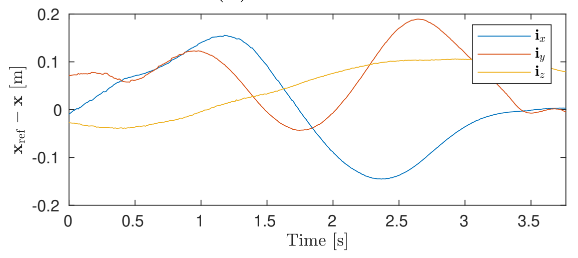
(a) Position.



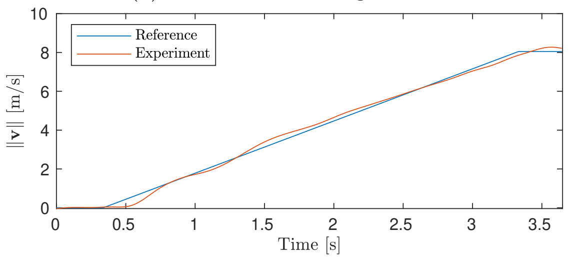
(b) Position.



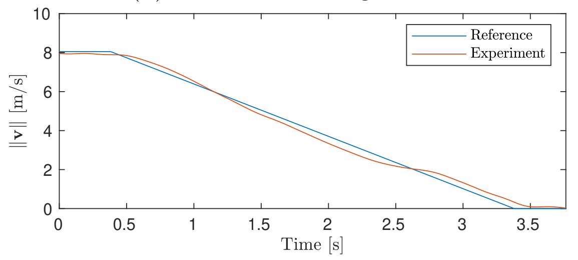
(c) Position tracking error.



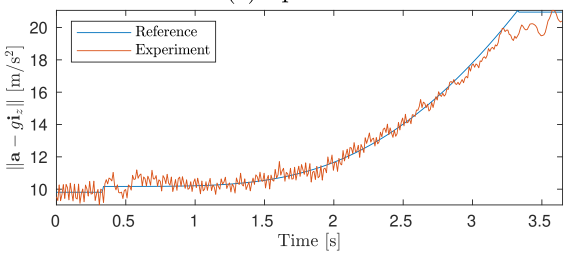
(d) Position tracking error.



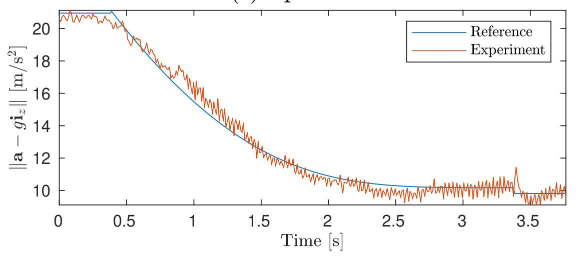
(e) Speed.



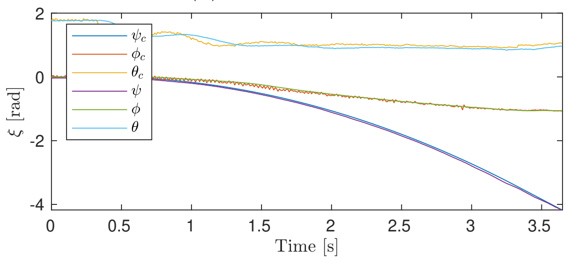
(f) Speed.



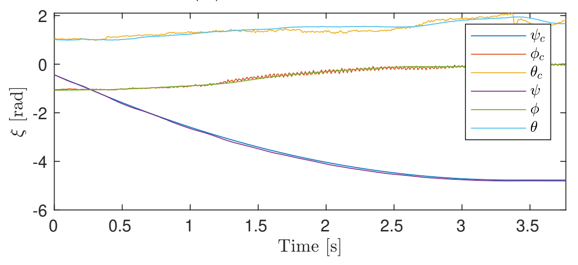
(g) Acceleration.



(h) Acceleration.

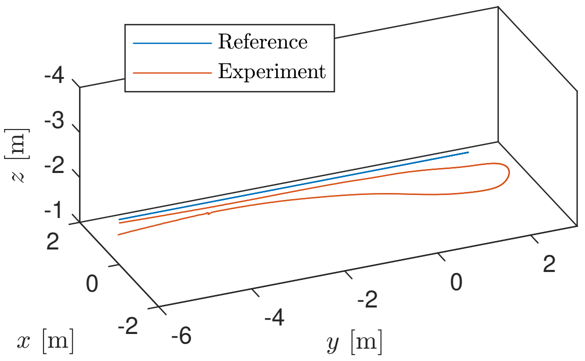


(i) Attitude.

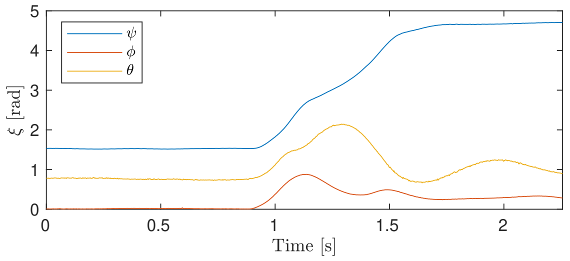


(j) Attitude.

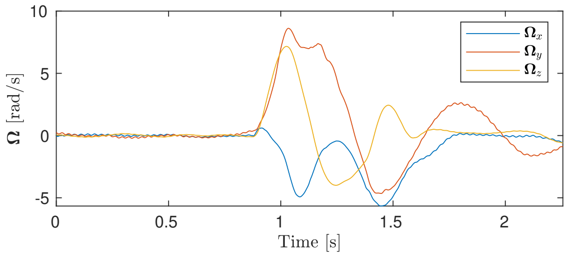
Figure 3-9: Experimental results for transition on a circle with radius 3.5 m from static hover to coordinated flight at 8 m/s (a), (c), (e), (g), and (i); and vice versa (b), (d), (f), (h), and (j).



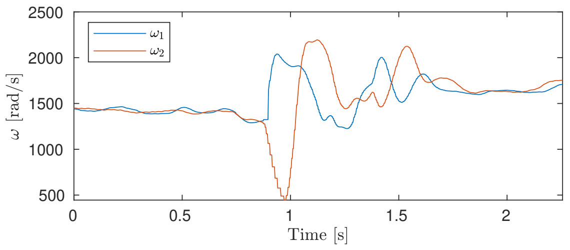
(a) Position.



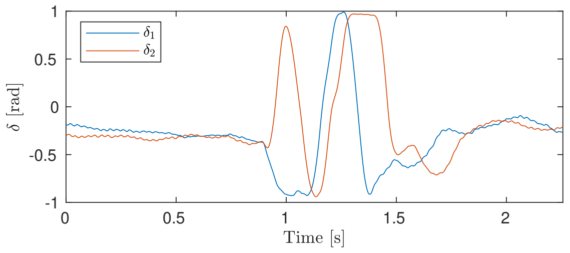
(b) Attitude.



(c) Angular velocity.



(d) Rotor speed.



(e) Flap deflection.

Figure 3-10: Experimental results for differential thrust turn.

errors and external disturbances in the absence of incremental control updates. The baseline attitude controller is thus given by

$$\dot{\boldsymbol{\Omega}}_c = \mathbf{K}_{\boldsymbol{\xi}}\boldsymbol{\zeta}_e - \mathbf{K}_{\boldsymbol{\Omega}}\boldsymbol{\Omega}_{\text{lpf}} + \mathbf{K}_{I_\xi}\int \boldsymbol{\zeta}_e \; dt. \tag{3.81}$$

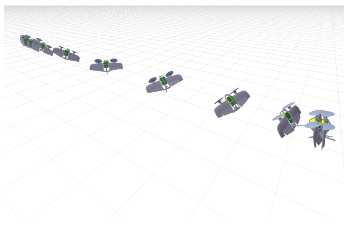
Table 3.4 gives key properties of the baseline controller, our proposed controller, and several tailsitter control designs presented in recent literature. In order to enable comparison, we only include guidance and control designs that fully govern both translational and rotational motion, i.e., we omit works that consider only hover, only longitudinal control, only rotational control etc. Also not included are control designs for (over-actuated) aircraft that possess actuators beyond the configuration we consider (i.e., two flaps and two rotors), since these aircraft typically present fundamentally different challenges in control design.

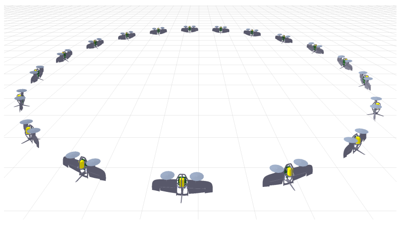
Table 3.4: Comparison of tailsitter flight control algorithms.

	Methodology	Aerodynamic model	Robustification	Feedforward
Proposed	INDI, and differential flatness	Global φ-theory	Incremental	Acceleration, jerk, and yaw rate
Baseline	Dynamic inversion	Global φ-theory	-	Acceleration
Lustosa, 2017 [68]	Scheduled LQR	Linearized φ-theory	-	Attitude, yaw rate (at trim points)
Ritz, 2017 [103]	Dynamic inversion (coordinated flight)	Global first principles	-	Acceleration
Chiappinelli, 2018 [11]	PD	Control effectiveness (moment only)	-	Attitude
Smeur, 2020 [114]	INDI	Control effectiveness (coordinated flight at small flight path angle)	Incremental	-
Barth, 2020 [4]	Model-free control (coordinated flight)	-	Model free	-

While the baseline controller lacks key aspects of our proposed control design, it is still a state-of-the-art dynamic inversion controller based on a global aerodynamics model that incorporates the acceleration feedforward signal. We note that none of the existing algorithms listed in the table combine robust control with feedforward control inputs to the same extent as our proposed controller. Controllers that do not incorporate any robustification require an accurate aerodynamics model that covers the entire flight envelope or will incur significant systematic tracking errors. Several algorithms include attitude or (equivalently) acceleration feedforward inputs, but none consider reference jerk, which is essential for accurate tracking of fast-changing accelerations. Without the angular rate feedforward input corresponding to jerk, the derivative term of the attitude controller will counteract the nonzero angular rate required to track a dynamic attitude reference. Finally, we note that existing controllers are often limited to coordinated flight because of limitations of the aerodynamic model or the control algorithm itself. In typical cruise conditions, this limitation is not prohibitive, but when performing agile maneuvers, it restricts the usable flight envelope, e.g., excluding knife edge flight.

In order to examine the combined and individual effects of incremental control and feedforward tracking, we employ two additional versions of the baseline controller: baseline+FF includes feedforward tracking but no incremental control, while baseline+INDI includes in-


(a) Hover-to-hover trajectory with $\pi/2$ rad yaw.


(b) Circle trajectory in knife edge orientation.

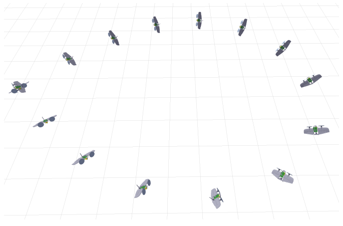

(c) Coordinated flight on lemniscate trajectory (partially shown).

Figure 3-11: Reference trajectories with attitude and flap deflections obtained from differential flatness transform.

cremental control but no feedforward tracking. Table 3.5 presents tracking performance data for the proposed and baseline controllers on the three trajectories shown in Fig. 3-11.

The hover-to-hover trajectory includes transitions and sideways flight. When flown slowly (i.e., 5 s), the maximum acceleration of 2.3 $\mathrm{m/s^2}$ is similar to transition maneuvers found in literature. All controllers track this slow trajectory with reasonable error. At low speeds, incremental control already significantly reduces the error by accounting for unmodeled dynamics, while feedforward makes less difference as the angular rates remain low. As expected, the importance of feedforward increases as the maneuver speeds up, evidenced by a dramatic increase in tracking error for the baseline and baseline+INDI controllers. The error growth is much smaller for the proposed and baseline+FF controllers, confirming that jerk and yaw rate tracking is essential for accurate tracking of aggressive maneuvers. As far as we are aware, our proposed control design is the first tailsitter controller to incorporate these feedforward inputs to enable agile maneuvers.

The importance of feedforward is further demonstrated on the knife edge circle trajectory. We found that the baseline and baseline+INDI controllers are unable to accurately maintain knife edge orientation due to absence of the 76 deg/s yaw rate feedforward reference. As knife edge flight is inherently unstable, failing yaw control leads to an overall failure to stabilize the vehicle. The baseline+FF controller performs much better, but—due to the lack of incremental control—still incurs a significantly larger systematic tracking error than the proposed controller.

The lemniscate trajectory is flown at higher speed, where the controller increasingly relies on the aerodynamics model for accurate application of force and moment. We found that the baseline controllers without INDI are unable to avoid crashing before the trajectory speed is reached. This may be due to inaccurate aerodynamics parameters or due to significant aerodynamic effects that are not captured by the simplified dynamics model. The baseline+INDI controller corrects for these modeling inaccuracies, and is able to avoid crashes. However, it incurs large tracking error—due to the lack of jerk and yaw rate tracking—to the point that we were unable to obtain an error measurement because of flight space size limitations.

3.7 Summary

In this chapter, we presented a control design aimed at tracking agile trajectories using a tailsitter flying wing. We derived a flatness transform for the nonlinear tailsitter flight dy-

Table 3.5: Position and yaw tracking error for proposed and baseline controllers. The top three rows (corresponding to hover-to-hover trajectories) contain maximum values, while the bottom two rows (corresponding to periodic trajectories) contain RMS values.

			$\|\mathbf{x}-\mathbf{x}_{\text{ref}}\|_2$ [cm]				$\|\psi-\psi_{\text{ref}}\|$ [deg]			
	$\|\mathbf{v}_{\text{ref}}\|_2$ [m/s]	$\|\mathbf{a}_{\text{ref}}\|_2$ [m/s^2]	Proposed	Baseline	Baseline+FF	Baseline+INDI	Proposed	Baseline	Baseline+FF	Baseline+INDI
Hover-to-hover (6 m, $\pi/2$ rad, 5 s)	3.0	2.3	**7.4**	41.2	31.9	17.4	**1.3**	21.7	13.6	8.4
Hover-to-hover (6 m, $\pi/2$ rad, 4 s)	3.7	3.5	**15.5**	63.0	33.8	34.9	**2.0**	21.5	19.8	10.1
Hover-to-hover (6 m, $\pi/2$ rad, 3 s)	4.9	6.2	**23.3**	>400	40.4	64.7	**10.4**	>25	17.4	20.6
Knife edge circle (3 m radius)	4.0	5.3	**2.8**	-	8.2	-	**0.6**	-	1.8	-
Lemniscate	6.0	9.1	**16.6**	-	-	>200	**2.8**	-	-	>25

namics with φ-theory aerodynamics model and formulated an angular velocity feedforward input that enables the controller to track the reference position along with its derivatives up to jerk. By applying INDI control, we obtain accurate trajectory tracking without relying on a globally accurate dynamics model. The aerodynamic parameters used by the controller were estimated based on flight data using a simple regression method, and the controller was evaluated in extensive experiments where it achieved accurate tracking of challenging trajectories. The advantage of robust control using INDI and the necessity of jerk and yaw rate feedforward inputs for tracking agile maneuvers were explicitly demonstrated through experimental comparison with a baseline version of the proposed controller.

Chapter 4

Multi-Fidelity Black-Box Optimization for Time-Optimal Quadrotor Maneuvers

We consider the problem of generating a time-optimal quadrotor trajectory for highly maneuverable vehicles, such as quadrotor aircraft. The problem is challenging since the optimal trajectory is located on the boundary of the set of dynamically feasible trajectories. This boundary is hard to model as it involves limitations of the entire system, including complex aerodynamic and electromechanical phenomena, in agile high-speed flight. In this chapter, we propose a multi-fidelity Bayesian optimization framework that models the feasibility constraints based on analytical approximation, numerical simulation, and real-world flight experiments. By combining evaluations at different fidelities, trajectory time is optimized while the number of costly flight experiments is kept to a minimum. The algorithm is thoroughly evaluated for the trajectory generation problem in two different scenarios: 1) connecting predetermined waypoints, 2) planning in obstacle-rich environments. For each scenario, we conduct both simulation and real-world flight experiments at speeds up to 11 m/s. Resulting trajectories were found to be significantly faster than those obtained through minimum-snap trajectory planning.

This chapter is based on joint work with Gilhyun Ryou [105]. A video of the experiments can be found at `https://youtu.be/FjCIZ5lSRg0`.

4.1 Introduction

In recent years, fast navigation of autonomous vehicles has received increasing interest. For several years an autonomous drone racing competition has been organized at IROS [84], and last year the AlphaPilot challenge introduced a similar competition to a more general audience [35]. Currently and in the near future, state-of-the-art algorithms in estimation and control are reaching a level of maturity, such that the bounds of what is physically possible with a given vehicle are approached. This presents the need for trajectory planning algorithms that fully exploit the capabilities of the vehicle and take into account the intricate limitations of vehicle dynamics, instead of relying on simplified models.

In this chapter, we consider the problem of generating, i.e., planning, a dynamically feasible, time-optimal quadrotor trajectory. By definition, such an optimal trajectory is found at the boundary of the set of feasible trajectories. Hence, precise knowledge of the dynamic

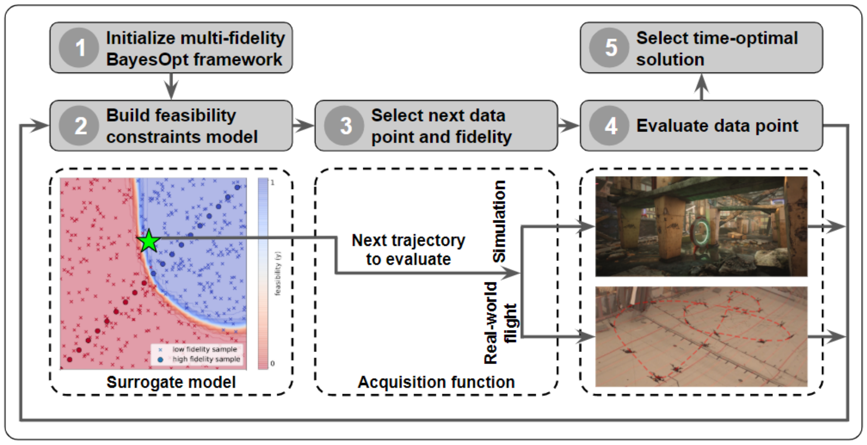

1 Initialize multi-fidelity BayesOpt framework
5 Select time-optimal solution
2 Build feasibility constraints model
3 Select next data point and fidelity
4 Evaluate data point
Next trajectory to evaluate
Simulation
Real-world flight
low fidelity sample
high fidelity sample
feasibility (y)
Surrogate model
Acquisition function

Figure 4-1: Overview of the proposed algorithm that models dynamic feasibility constraints based on simulation and flight data to efficiently find the time-optimal trajectory.

feasibility constraints is required to find the time optimum. This complicates the problem, as these feasibility constraints can become highly complex in light of high-acceleration flight and aggressive attitude changes, as required to achieve time optimality. The demanding maneuvers are affected by flight dynamics, but also by hardware and software for control and state estimation. The resulting set of non-convex feasibility constraints with memory (i.e., depending on present and past states and control inputs) cannot readily be incorporated in a typical trajectory planner for two main reasons. Firstly, the feasibility constraints are not easily expressed in a convenient way, e.g., as constraints on an admissible set of control inputs and states. Instead, the feasibility of the trajectory must be considered in a holistic manner. Secondly, in most scenarios, precise modeling of these constraints is only possible through real-world experiments. As these experiments aim to seek the boundary of the feasible set, they are risky and potentially costly and should thus be kept to a minimum. Contrarily to this objective, many trajectory optimization schemes rely on a large number of evaluations. These two issues form the main motivation for our algorithm, which uses a multi-fidelity optimization technique that can approximate the system feasibility constraints based on a limited number of experiments. It uses a Gaussian process black-box model to classify candidate trajectories as feasible or infeasible and is thereby able to plan increasingly fast trajectories as the model improves. An overview of the algorithm is shown in Fig. 4-1.

This chapter contains several contributions. Firstly, we propose an algorithm for modeling of quadrotor feasibility constraints and generation of time-optimal trajectories based on Gaussian process classification. Secondly, we extend the applicability of the multi-fidelity deep Gaussian process kernel from the regression problem to the classification problem in order to obtain a multi-fidelity Gaussian process classification algorithm that can incorporate evaluations from analytical approximation, numerical simulation, and real-world flight experiments. Thirdly, we design an acquisition process specifically tailored to experimental robotics. The acquisition function takes into account the additional cost of infeasible evaluations, as these may pose a threat to the vehicle and its surroundings. Candidate data points are generated using minimum-snap perturbations in order to maintain feasibility. Finally, we present an extensive evaluation of the proposed algorithm in both simulation and real-life flight experiments at speeds up to 11 m/s. It is found that optimized trajectories are significantly faster than those obtained through minimum-snap trajectory planning.

The outline of the chapter is as follows. In Section 4.2 and Section 4.3, we present the problem definition and preliminaries on trajectory planning, and preliminaries on Bayesian optimization, respectively. We detail our algorithm for the generation of time-optimal trajectories using iterative experiments in Section 4.4. In Section 4.5, we present experimental results from both simulation and real-world flights. Finally, we provide a summary in Section 4.6. In this chapter, we mostly use the prevailing notation from literature on Bayesian optimization. Consequently, the notation partly deviates from the nomenclature that was introduced in preceding chapters.

4.2 Quadcopter Trajectory Generation

We consider the problem of planning a time-optimal quadrotor trajectory. A trajectory $p : \mathbb{R}_{\geq 0} \to \mathbb{R}^3 \times \mathbb{T}$—with $\mathbb{T}$ the circle group—is a continuous function that maps time to position and yaw, i.e., $p(t) = \begin{bmatrix} p_r(t)^\top & p_\psi(t) \end{bmatrix}^\top$. We define the following time-optimal planning problem:

$$\begin{aligned} \underset{p,\,T}{\text{minimize}} \quad & T \\ \text{subject to} \quad & p(0) = \tilde{p}^{\text{start}}, \\ & p\,(T) = \tilde{p}^{\text{end}}, \\ & p \in \mathcal{F}, \\ & p \in \mathcal{P}_T, \end{aligned} \tag{4.1}$$

where $\tilde{p}^{\text{start}}$ and $\tilde{p}^{\text{end}}$ are respectively the start and end points, each consisting of a prescribed position and yaw angle, i.e., $\tilde{p} = \begin{bmatrix} \tilde{p}_r{}^\top & \tilde{p}_\psi \end{bmatrix}^\top \in \mathbb{R}^3 \times \mathbb{T}$, and $\mathcal{F}$ denotes the set of trajectories that satisfy the relevant geometric constraints, e.g., for obstacle avoidance. The function space $\mathcal{P}_T$ is the set of all feasible trajectories, i.e., all trajectory functions that the quadrotor can successfully track over $[0, T]$. Using differential flatness of the idealized quadcopter dynamics, we know that for any feasible trajectory, $p_r \in \mathcal{C}^4$ and $p_\psi \in \mathcal{C}^2$ over the interval $[0, T]$ [80]. However, more constraints should be incorporated when considering fast and agile trajectories. Critically, the complete system including aerodynamics, sensor and actuation hardware, and estimation and control software must be considered, instead of solely the idealized vehicle dynamics. This is necessary because all aspects of the system affect tracking performance during demanding maneuvers, e.g., as follows:

- Significant aerodynamic forces and momenta act on the vehicle and need to be accounted for in control inputs. In contrast, vehicle aerodynamics can typically be neglected in low-speed flight.
- State estimation errors due to delay and phase lag are exacerbated by the fast-changing vehicle state. Moreover, sensors measurements may incur additional noise due to large accelerations.
- Actuation delay and bandwidth limitations, such as the mechanical time constant of the motors, inhibit tracking of fast-changing control inputs.
- Battery internal resistance causes the voltage to drop under large currents. Consequently, very high motor speeds can only be maintained for limited consecutive time periods.

What results is a set of non-convex feasibility constraints with memory (i.e., depending on present and past states and control inputs). These characteristics of $\mathcal{P}_T$ make the planning problem hard to solve, even when using nonlinear optimization techniques such as direct collocation or shooting methods. Additionally, it is important to note that in practice no trajectory can be tracked perfectly due to stochastic sensing and actuation imperfections, so the definition of $\mathcal{P}_T$ must incorporate an error bound.

Popular methods for trajectory planning avoid the complicated feasibility constraints by reformulating the optimization problem such that feasibility is the objective instead of a constraint [80, 101]. In practice, this is achieved by minimizing high-order derivatives of the trajectory function. Particularly, minimizing the fourth-order derivative of position, i.e., snap, and the second-order derivative of yaw results in a minimization of the required angular accelerations. This reduces trajectory agility and may thereby prevent activation of the aforementioned feasibility constraints. Before we detail the resulting minimum-snap optimization problem, we describe two formulations to incorporate geometric constraints imposed by $\mathcal{F}$. For certain applications, like mapping and surveillance, one may opt to define a sequence with $m-1$ intermediate waypoints, i.e., $\tilde{\mathbf{p}} = \begin{bmatrix} \tilde{p}^{\text{start}} & \tilde{p}^1 & \cdots & \tilde{p}^{m-1} & \tilde{p}^{\text{end}} \end{bmatrix}$, to be attained in order. The corresponding constraints take the form

$$p\left(\sum_{j=1}^{i} x_j\right) = \tilde{p}^i,\ i = 1,\ \ldots,\ m-1, \tag{4.2}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_m \end{bmatrix}$ denotes the time allocation over trajectory segments, e.g., x_i is the time allocated for the segment between waypoints $\tilde{p}^{i-1}$ and $\tilde{p}^i$. When planning trajectories through obstacle-rich environments, it may be preferable to incorporate collision avoidance constraints instead of waypoints, e.g., by utilizing a series of convex boxes to describe the obstacle-free space [28, 127], or decomposing the obstacle-free space into a set of convex polytopes through semi-definite programming [15]. A sequence of m obstacle-free convex polytopes to pass through can be incorporated in the trajectory optimization as follows:

$$A_i p\left(t\right) \leq b_i,\ \forall t \in \left[\sum_{j=1}^{i-1} x_j, \sum_{j=1}^{i} x_j\right],\ i = 1,\ \ldots,\ m, \tag{4.3}$$

where matrix $A_i \in \mathbb{R}^{d_i \times 3}$ and vector $b_i \in \mathbb{R}^{d_i}$ constrain the trajectory segment to be within a polytope of d_i faces. The methods for construction and sequencing of these obstacle avoidance polytopes are described in Section 4.4.4.

The minimum-snap trajectory optimization problem is given by:

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & \sigma(p, \sum_{i=1}^{m} x_i) \\ \text{subject to} \quad & p(0) = \tilde{p}^{\text{start}}, \\ & p\left(\sum_{j=1}^{m} x_j\right) = \tilde{p}^{\text{end}}, \\ & p \in \mathcal{F}, \end{aligned} \tag{4.4}$$

where

$$\sigma(p, T) = \int_0^T \mu_r \left\| \frac{d^4 p_r}{dt^4} \right\|^2 + \mu_\psi \left(\frac{d^2 p_\psi}{dt^2} \right)^2 dt \tag{4.5}$$

with μ_r and μ_ψ weighing parameters. The final constraint in (4.4) is given by either (4.2) or (4.3). By utilizing a piecewise polynomial function to describe the trajectory, differentiability constraints can be guaranteed by appropriately selecting the order of continuity between segments, and the optimization problem (4.4) can be formulated as a quadratic program [80]. In case of waypoint constraints, efficient solvers for unconstrained quadratic programming can be used [101]. For convenience, we define the function that gives the minimizer trajectory of (4.4) for the time allocation $\mathbf{x}$ as follows:

$$p = \chi(\mathbf{x}, \tilde{\mathcal{F}}), \tag{4.6}$$

where $\tilde{\mathcal{F}}$ represents either $\tilde{\mathbf{p}}$ (in case of waypoint constraints), or $(\tilde{p}^{\text{start}}, \tilde{p}^{\text{end}}, \mathbf{A}, \mathbf{b})$ with $\mathbf{A}$ and $\mathbf{b}$ containing respectively all A_i and b_i (in case of polytope constraints).

Minimum-snap trajectory generation algorithms employ a two-step process based on (4.6). First, the minimum-snap trajectory for a (very large) initial guess of the total trajectory time is found, as follows:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^m_{\geq 0}}{\text{minimize}} \quad & \sigma\left(\chi(\mathbf{x}, \tilde{\mathcal{F}}), T\right) \\ \text{subject to} \quad & T = \sum\nolimits_{i=1}^{m} x_i. \end{aligned} \tag{4.7}$$

Next, the obtained time allocation is scaled down, i.e.,

$$\begin{aligned} \underset{\eta\in\mathbb{R}_{>0}}{\text{minimize}} \quad & T \\ \text{subject to} \quad & T = \sum\nolimits_{i=1}^{m} \eta x_i, \\ & \chi(\eta\mathbf{x}, \tilde{\mathcal{F}}) \in \mathcal{P}_T. \end{aligned} \tag{4.8}$$

The feasibility constraint is typically evaluated based on control inputs obtained from differential flatness of the idealized quadrotor dynamics [80]. While this method can generate fast, feasible trajectories, it does not attain time optimality for several reasons. Firstly, the time allocation ratio obtained in (4.7) may not be optimal as the total trajectory time is decreased, i.e., there may exist an alternative allocation ratio that will enable lower feasible total trajectory time. Secondly, the method uses a simple feasibility model that fails to consider the trajectory as a whole, as is required for a realistic feasibility check.

Our proposed algorithm searches for a time-optimal trajectory by directly addressing the two major shortcomings of minimum-snap trajectory generation. It builds a realistic, probabilistic feasibility constraint model that considers the trajectory as a whole, and uses this model to find the optimal time allocation over trajectory segments.

4.3 Bayesian Optimization

Bayesian optimization, or *BayesOpt*, is a class of algorithms that use machine learning techniques for solving optimization problems with black-box objective or constraint functions that are expensive to evaluate. Evaluation points are selected to model the unknown functions and approach the optimum with maximum efficiency, so that the total number of evaluations is kept to a minimum. Within the BayesOpt framework, Gaussian process (GP) modeling [100] is widely used to build a *surrogate model* that approximates the unknown objective or constraint functions based on noisy measurements. Given data points $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ with corresponding evaluations $\mathbf{y} = \{y_1, \cdots, y_N\}$, GP modeling predicts the probability $P(y_*|\mathbf{y}, \mathbf{x}_*, \mathbf{X})$ for a test point $\mathbf{x}_*$. It assumes the joint distribution over the evaluations is a joint Gaussian

$$P(\mathbf{y}, y_*|\mathbf{x}_*, \mathbf{X}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}\\ y_*\end{bmatrix} \middle| 0, \begin{bmatrix}K(\mathbf{X}, \mathbf{X}) \, K(\mathbf{X}, \mathbf{x}_*)\\ K(\mathbf{x}_*, \mathbf{X}) K(\mathbf{x}_*, \mathbf{x}_*)\end{bmatrix}\right). \tag{4.9}$$

$P(y_*|\mathbf{y}, \mathbf{x}_*, \mathbf{X})$ can then be obtained by marginalizing $\mathbf{y}$ in (4.9). Each element of the covariance $K(\mathbf{X}, \mathbf{X})$ is nonparametrically estimated by the corresponding data points and a covariance kernel $K_{ij}(\mathbf{X}, \mathbf{X}) = k(\mathbf{x}_i, \mathbf{x}_j)$ with hyperparameters θ. The hyperparameters are trained by maximizing the marginal likelihood $P(\mathbf{y}|\mathbf{X})$. As a nonparametric method, GP can flexibly model a functional distribution, even with a small number of data points. This functional approximation using GP is often called Kriging or Gaussian process regression.

When modeling inequality constraints, Gaussian process classification (GPC) is often used instead of direct GP modeling. Rather than assuming a joint distribution over the evaluations, GPC assumes a joint Gaussian distribution over the evaluations and the latent variables $\mathbf{f} = \begin{bmatrix}f_1, \cdots, f_N\end{bmatrix}$. The latent variables encode label probabilities for the evaluations, which can be obtained through a mapping onto the probability domain $[0, 1]$, e.g.,

$$\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(s|0, 1)ds. \tag{4.10}$$

The latent variables and the hyperparameters of the kernel function are trained by maximizing the marginal likelihood function

$$\begin{aligned} P(\mathbf{y}, \mathbf{f}|\mathbf{X}) &= \Pi_{i=n}^{N} P(y_n|f_n)P(\mathbf{f}|\mathbf{X}) \\ &= \Pi_{n=1}^{N} \mathcal{B}(y_n|\Phi(f_n))\mathcal{N}(\mathbf{f}|0, K(\mathbf{X}, \mathbf{X})), \end{aligned} \tag{4.11}$$

where $\mathcal{B}(x)$ is the Bernoulli likelihood used to formulate $\Phi(f_n)$ as a probability distribution. The joint GP prior assumption given by (4.9) can now be used to estimate the distribution of f_*, which is the latent variable corresponding to the class probability y_*. The covariance between $\mathbf{X}$ and a test point $\mathbf{x}_*$ is modeled with the same covariance kernel, as follows

$$P(\mathbf{f}, f_*|\mathbf{x}_*, \mathbf{X}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix} \middle| 0, \begin{bmatrix}K(\mathbf{X}, \mathbf{X}) \, K(\mathbf{X}, \mathbf{x}_*)\\ K(\mathbf{x}_*, \mathbf{X}) K(\mathbf{x}_*, \mathbf{x}_*)\end{bmatrix}\right), \tag{4.12}$$

so that the distribution of the latent variable f_* can be estimated as

$$P(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int P(f_*|\mathbf{f}, \mathbf{x}_*, \mathbf{X})P(\mathbf{f}|\mathbf{X}, \mathbf{y})d\mathbf{f}. \tag{4.13}$$

The resulting class probability is obtained by

$$P(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int P(y_*|f_*)P(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})d\mathbf{f}_*. \tag{4.14}$$

For more details on GPC and its implementation, the reader is referred to [88].

The BayesOpt *acquisition function* is designed to select the next evaluation point by considering both reducing the uncertainty of the surrogate model and finding the precise optimum of the objective function. Based on the data $\mathcal{D}$ obtained in previous evaluations, the surrogate model can be trained to approximate the unknown function. Next, the approximate optimal solution can be estimated using the trained surrogate model. Each next evaluation point is obtained by solving the following optimization problem:

$$\mathbf{x}_{next} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}|\mathcal{D}) \tag{4.15}$$

where $\alpha(\mathbf{x}|\mathcal{D})$ is the acquisition function that represents the value of an evaluation point $\mathbf{x}$ given the data $\mathcal{D}$. If the surrogate model approximates an unknown objective function, the acquisition function can, for example, be based on the expected improvement of the objective function [83], the expected entropy reduction of the distribution over the location of the solution [43,136], or the upper bound of the optimum [121]. If constraint functions are also modeled, the acquisition function must consider the uncertainty of both objective and constraint function models, e.g., using the product of the expected objective improvement and the probability of constraint satisfaction [29,32]. This approach has been extended to select data points more efficiently utilizing the asymmetric design of the acquisition function [64] and to incorporate constraint satisfaction into the estimation of the expected entropy reduction [42, 71]. In our proposed algorithm, Bayesian optimization is applied to model the feasibility constraints of the time-optimal trajectory planning problem. Related applications can be found in literature, where the acquisition function is designed considering the uncertainty of model feasibility functions and dynamics boundaries of control systems [5,98].

In multi-fidelity Bayesian optimization, function evaluations of different fidelities can be combined. For instance, a rough simulation or an expert's opinion may serve as a low-fidelity model, while a high-accuracy simulation or real-world experiment serves as a high-fidelity model. The key idea is that combining cheap low-fidelity evaluations with expensive high-fidelity measurements improves overall efficiency. To incorporate information from multiple sources, the surrogate model must be modified to combine multi-fidelity evaluations, e.g., by using a linear transformation to describe the relationship between different fidelities [57,60]. Suppose that we have L fidelity levels, and each level is denoted by $l \in \{l^1, l^2, \ldots, l^L\}$, where l^1 is the level of the lowest-fidelity experiment and l^L is the level of the highest-fidelity experiment. The relationship between adjacent fidelity levels l^j and l^{j-1} can be modeled as

$$f_{l^j}(\mathbf{x}) = \rho_{l^{j-1}} f_{l^{j-1}}(\mathbf{x}) + \delta_{l^j}(\mathbf{x}), \tag{4.16}$$

where $f_{l^j}(\mathbf{x})$ and $f_{l^{j-1}}(\mathbf{x})$ are the output distributions of $\mathbf{x}$ for l^j and for l^{j-1}, respectively. The bias distribution δ_{l^j} is independent of $f_{l^{j-1}}, \ldots, f_{l^1}$, and the constant $\rho_{l^{j-1}}$ represents the correlation between the output distributions for the two adjacent fidelity levels. This method can be extended to include a more advanced nonlinear space-dependent transformation [14, 93]. Besides auto-regressive models, other approaches for multi-fidelity optimization exist, e.g., utilizing the decision boundary of a support vector machine (SVM) to reduce the search space of high-fidelity data points [18], or utilizing pairwise comparison of low-fidelity evaluations to determine the adversarial boundary of the high-fidelity model [140].

Similar to the surrogate model, the acquisition function has to be modified to incorporate multi-fidelity evaluations. In the multi-fidelity Bayesian optimization framework, the acquisition function not only selects the next evaluation point, but also its fidelity level, as follows:

$$\mathbf{x}_{next}, l_{next} = \arg\max_{\mathbf{x}, l \in \{l^1, \ldots, l^L\}} \alpha(\mathbf{x}, l|\mathcal{D}). \tag{4.17}$$

The acquisition function is modified by introducing weights based on the evaluation cost at the different fidelity levels. In practice, high-fidelity evaluations will have smaller weights compared to low-fidelity evaluations. This makes the algorithm less likely to select high-fidelity evaluations, so that the overall cost of the experiments is minimized. Common metrics used in acquisition functions are modified in this manner, e.g., weighted expected improvements [47], weighted expected entropy reductions [124], and weighted upper bounds of the optimum [53]. The multi-fidelity classification problem can also be approached by differently weighting the uncertainty reduction of the classifiers at each fidelity level [13]. Since multi-fidelity Bayesian optimization has the potential to reduce the number of expensive high-fidelity experiments, it has been applied in various fields, including analog circuit design [143], aircraft wing design [99], and control synthesis [72, 98].

Although Bayesian optimization aims to contain the number of required function evaluations, it may still suffer from large computation cost as the number of data points increases. This is mainly due to inversion of a covariance matrix including all data points for uncertainty quantification in the surrogate function, leading to a computational cost proportional to the number of data points cubed. Particularly multi-fidelity Bayesian optimization may suffer from this issue, as the number of data points can quickly increase by adding low-fidelity evaluations. The problem is addressed by the inducing points method, which uses a set of pseudo-data points for uncertainty quantification [40, 119]. The pseudo-data points are selected to minimize the Kullback Leibler (KL) divergence between the function posterior given all data points (including the pseudo-data points), and the function posterior given only the pseudo-data points. Since the number of pseudo-data points is much smaller than the number of actual data points, the inducing points method can greatly improve algorithm performance as the size of the data set increases. The method has been extended to the classification problem [41] and applied to multi-fidelity Bayesian optimization [14], and GPU acceleration can be used to further increase its efficiency [30].

4.4 Algorithm

Our proposed algorithm uses Bayesian optimization to efficiently minimize the total trajectory time T by approximating the feasibility constraints of $\mathcal{P}_T$ using multi-fidelity evaluations from various sources, such as simulation and real-world flight experiments. Unlike typical minimum-snap trajectory planning, the algorithm maintains the ideal planning formulation of (4.1) with minimum time as the objective and feasibility as a constraint. We exploit the fact that any time allocation over the trajectory segments can be mapped to a smooth trajectory that attains the given geometric constraints using (4.6). This enables us to transform the problem of finding the time-optimal trajectory to the problem of finding the optimal time allocation over segments. Therefore, we can formulate the following optimization problem on the finite-dimensional space $\mathbb{R}^m_{\geq 0}$ that is the set of all possible time allocations:

$$\begin{aligned} \underset{T,\, \mathbf{x}\in\mathbb{R}^m_{\geq 0}}{\text{minimize}} \quad & T \\ \text{subject to} \quad & \chi(\mathbf{x}, \tilde{\mathcal{F}}) \in \mathcal{P}_T, \\ & T = \textstyle\sum_{i=1}^{m} x_i. \end{aligned} \tag{4.18}$$

This formulation is based on the assumption that if there exists a feasible trajectory, i.e., a member of the set $\mathcal{P}_T$, that attains geometric constraints $\tilde{\mathcal{F}}$ with time allocation $\mathbf{x}$, then $\chi(\mathbf{x}, \tilde{\mathcal{F}}) \in \mathcal{P}_T$. This assumption is reasonable, since (4.4) optimizes for feasibility (by minimizing snap), so it is unlikely that its optimum is infeasible while there exists a feasible trajectory subject to the same time allocation and waypoints.

We use Gaussian process classification to learn a surrogate model of the feasibility constraint in (4.18). The surrogate model combines results from sequential experiments at L fidelity levels. An experiment $f_l(\mathbf{x})$ evaluates the feasibility of the trajectory $\chi(\mathbf{x}, \tilde{\mathcal{F}})$ at fidelity level l, resulting in a classification $y \in \{\text{feasible}, \text{infeasible}\}$. Evaluations at fidelity level l are gathered in dataset $\mathcal{D}_l = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,N_l}$, where N_l is the number of evaluations at fidelity level l. The GPC probability model at fidelity level l, i.e., $P_l(y = \text{feasible}\,|\mathbf{x})$, is based on data points from fidelity levels l and lower. The overall objective is to find the optimal time allocation that satisfies the feasibility constraint at the highest fidelity level with sufficient confidence, i.e., $P_{l^L}(y = \text{feasible}\,|\mathbf{x}^*) \geq h$.

Each evaluation is selected based on the acquisition function, which considers *exploration* and *exploitation* with the goal of maximizing the efficiency of the overall optimization process. Based on the current trained surrogate model, it estimates the expected improvements in objective function value and model accuracy. By iteratively improving the feasibility model and minimizing the objective function, our algorithm searches for the time allocation that minimizes total trajectory time. An overview of the complete algorithm is given in Algorithm 1, and its major components are detailed in ensuing sections.

Algorithm 1: Multi-fidelity black-box optimization for time-optimal trajectory planning

```
   Input:  Multi-fidelity experiments $f_l$, $l = l^1, \dots, l^L$;
           acquisition function $\alpha$
 1 Find initial solution by solving (4.8)
 2 Initialize multi-fidelity dataset $\mathcal{D}_l$, $\forall l \in \{l^1, \dots, l^L\}$
 3 for $i = 1, \cdots$ do
 4     Build the surrogate model $\mathcal{M}_l$, $\forall l \in \{l^1, \dots, l^L\}$
 5     Generate candidate solutions $\mathcal{X}$
 6     $\mathbf{x}_i, l_i \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}, l \in \{l^1, \dots, l^L\}} \alpha(\mathbf{x}, l|\mathcal{D})$
 7     $y_i \leftarrow f_{l_i}(\mathbf{x}_i)$
 8     $\mathcal{D}_{l_i} \leftarrow \mathcal{D}_{l_i} \cup \{(\mathbf{x}_i, y_i)\}$
 9 end
10 $T^* \leftarrow \sum_{i=1}^{m} x_i^*$
11 $p^* \leftarrow \chi(\mathbf{x}^*, \tilde{\mathcal{F}})$
   Output: $T^*$, $p^*$
```

4.4.1 Multi-Fidelity Constraint Model

We define the GPC-based surrogate model $\mathcal{M}$ as the set of latent variables $\mathbf{f} = \left[f_1, \cdots, f_N\right]$, the hyperparameters of covariance matrix θ, and the hyperparameters of the inducing points $\mathbf{m}$ and $\mathbf{S}$, such that $\mathcal{M} = (\mathbf{f}, \theta, \mathbf{m}, \mathbf{S})$. The inducing points method is used to accelerate the optimization process [41]. This enables us to avoid the $\mathcal{O}(N^3)$—with N the number of data points—computational complexity otherwise associated with the calculation of the inverse covariance matrix, which is used for estimating the distribution of the latent variables, i.e., $P(\mathbf{f}|\mathbf{X}, \mathbf{y})$. By introducing an additional variational distribution $q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$, the inducing point method approximates

$$P(\mathbf{f}|\mathbf{X}, \mathbf{y}) \sim q(f) := \int p(f|\mathbf{u})q(\mathbf{u})d\mathbf{u}, \tag{4.19}$$

where the hyperparameters $\mathbf{m}$ and $\mathbf{S}$ represent the mean and covariance of the inducing points, respectively. The method uses the following variational lower bound as loss function to determine the latent variables and the hyperparameters:

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(f_n)} \left[\log p(y_n | f_n)\right] - D_{KL} \left[q(\mathbf{u}) \| p(\mathbf{u})\right]. \tag{4.20}$$

In order to utilize the multi-fidelity optimization technique, we update the definition of the surrogate model and define the l^j-fidelity surrogate model ($j = 1, \ldots, L$) as

$$\mathcal{M}_{l^j} = \{f^{l^j}, [\theta^{l^1}, \cdots, \theta^{l^j}], \mathbf{m}^{l^j}, \mathbf{S}^{l^j}\}. \tag{4.21}$$

We use the multi-fidelity deep Gaussian process (MFDGP) [14] as covariance kernel function to estimate the GP prior from multiple evaluation sources. The multi-fidelity Gaussian process has separate latent variables and inducing points $\mathbf{f}^{l^j}$, $\mathbf{u}^{l^j} \sim \mathcal{N}(\mathbf{m}^{l^j}, \mathbf{S}^{l^j})$ for each fidelity l^j. Thus, we define the surrogate model separately for each fidelity model. However, fidelity models share the hyperparameters of the covariance kernels. Specifically, the MFDGP models the relationship between adjacent fidelities with another Gaussian process as

$$\begin{aligned} k_{l^j}(\mathbf{x}, \mathbf{x}') = k_{l^j,corr}(\mathbf{x}, \mathbf{x}')(\sigma^2_{l^j,linear}\, f_{l^{j-1}}(\mathbf{x})^\top f_{l^{j-1}}(\mathbf{x}') \\ + k_{l^j,prev}(f_{l^{j-1}}(\mathbf{x}), f_{l^{j-1}}(x'))) + k_{l^j,bias}(\mathbf{x}, \mathbf{x}'), \end{aligned} \tag{4.22}$$

where $f_{l^{j-1}}$ is the Gaussian process estimation from the preceding fidelity level, $\sigma_{l^j,linear}$ is a constant scaling the linear covariance kernel, and $k_{l^j,prev}$, $k_{l^j,corr}$ and $k_{l^j,bias}$ represent the covariance with the preceding fidelity, the space-dependent correlation function and the bias function, respectively. Radial basis functions are used for these kernels. Their hyperparameters are trained by maximizing the following variational lower bound:

$$\begin{aligned} \mathcal{L} = \sum_{j=1}^{L} \Big(\sum_{n=1}^{N_{l^j}} \mathbb{E}_{q_{l^j}(f_n^{l^j})} \left[\log p(y_n^{l^j} | f_n^{l^j})\right] \\ - D_{KL} \left[q_{l^j}(\mathbf{u}^{l^j}) \| p(\mathbf{u}^{l^j})\right] \Big), \end{aligned} \tag{4.23}$$

where N_{l^j} denotes the number of data points in the l^j-fidelity dataset D_{l^j}, and $q_{l^j}(f_n^{l^j}) := \int p(f_n^{l^j} | \mathbf{u}^{l^j}) q(\mathbf{u}^{l^j}) d\mathbf{u}^{l^j}$, which is the same formulation as used in (4.20) for the hyperparameters of the l^j-fidelity model. Note that inferencing of data points at l^j-fidelity requires estimation of the covariance kernel at l^j and lower fidelities (cf. (4.21)), while the l^j-fidelity covariance kernel is trained based on the all data points at l^j and higher fidelities. We follow the implementation of the inducing points proposed in [14], but utilize the likelihood approximation from [41] to extend the multi-fidelity kernel to the classification problem. The overall GPC modeling is developed in the GPytorch framework with GPU acceleration [30].

4.4.2 Acquisition Function

We design the acquisition function considering both *exploration* to improve the surrogate model, and *exploitation* to find the optimal solution. In exploration, we aim to maximize the effectiveness of improving the surrogate model and select the most uncertain sample near the decision boundary of the classifier [13]. Since the latent function mean approaches zero at the decision boundary, this sample is found as the maximizer of

$$\alpha_{explore}(\mathbf{x}, l) = -\frac{|\mu_l(\mathbf{x})|}{\sigma_l(\mathbf{x})} C_l, \tag{4.24}$$

where C_l reflects the cost of an evaluation at fidelity level l. We note that in practice the feasible set is connected — feasibility of a time allocation implies feasibility of any time allocation that is element-wise larger — so that exploration on the boundary suffices. In exploitation, we utilize expected improvement with constraints (EIC) to quantify the expected effectiveness of a candidate data point based on the product of expected objective improvement and the probability of feasibility [29]. We modify EIC for use in an experimental robotics application, where the cost of evaluation depends on the outcome, as attempting infeasible experiments may pose a danger to the vehicle and its surroundings. We consider not only the probability of success, but also the corresponding variance, as follows:

$$\tilde{P}_l(y = 1|\mathbf{x}) = P_l(\mu_l(\mathbf{x}) - \beta\sigma_l(\mathbf{x}) \geq 0|\mathbf{x}), \tag{4.25}$$

where β is a penalty on variance, and μ_l and σ_l are respectively the mean and standard deviation of the posterior distribution $P(f|\mathbf{x}, \mathcal{D})$ obtained from (4.13). The resulting acquisition function is given by

$$\alpha_{exploit}(\mathbf{x}, l) = \begin{cases} \alpha_{EI}(\mathbf{x})\tilde{P}_l(y = 1|\mathbf{x}), & \text{if } \tilde{P}_l(y = 1|\mathbf{x}) \geq h_l \\ 0, & \text{otherwise} \end{cases} \tag{4.26}$$

where h_l is based on the cost of an infeasible evaluation. Since the objective function is deterministic, so is the expected improvement, i.e., $\alpha_{EI}(\mathbf{x}) = \sum_i \bar{x}_i - \sum_i x_i$ with $\bar{\mathbf{x}}$ the current best solution.

Finally, we combine (4.24) and (4.26) to obtain

$$\alpha(\mathbf{x}, l) = \begin{cases} \alpha_{exploit}(\mathbf{x}, l), & \text{if } \exists x \in \mathcal{X} \text{ s.t. } \alpha_{exploit}(x, l) > 0 \\ \alpha_{explore}(\mathbf{x}, l), & \text{otherwise} \end{cases} \tag{4.27}$$

Note that exploration only occurs if exploitation is ineffective. We found that this greedy approach works well in practice, as overly explorative searching leads to a large number of infeasible evaluations.

In case of a discontinuous acquisition function like (4.27), Latin hypercube sampling (LHS) is often used to generate candidate solutions of the aquisition function [121]. However, we observe that this method may fail to generate an optimal solution because it does not consider the correlation between adjacent time allocations. We address this by devising a sampling-based algorithm that uses a smooth perturbation of the current best solution. For this purpose, we extend the notion of goal-convergent exploration from [52]. This method generates smooth trajectories by minimizing the expected sum of acceleration at each collocation point. Instead of acceleration, we minimize the expected jerk. As time allocation is inversely related to flight speed, this approximately minimizes the jerk of the speed profile, and thereby the snap of the perturbation, which is favorable for maintaining feasibility. The perturbation vector ϵ is found by solving the following semi-definite program:

$$\begin{aligned} \underset{\Sigma \in \mathbb{R}^{m\times m}}{\text{minimize}} \quad & \operatorname{trace}(A^\top A \Sigma)\big(= \mathbb{E}[\epsilon A^\top A \epsilon]\big) \\ \text{subject to} \quad & \epsilon \sim \mathcal{N}(0, \Sigma), \\ & \Sigma \succeq 0,\ \Sigma_{ii} = \gamma, \end{aligned} \tag{4.28}$$

where $\gamma > 0$ scales the variance of perturbations and A is the third-order finite differencing matrix. Finally, the set of N_c candidate solutions is generated as $\mathcal{X} = \{\bar{\mathbf{x}} \odot (1 + \epsilon_i)\}_{i=1,\ldots,N_c}$ with $\epsilon_i \sim \mathcal{N}(0, \Sigma)$ and $\odot$ the element-wise product. Candidates with negative elements are rejected.

4.4.3 Initialization

Initial data points to build the surrogate model are generated around the trajectory found by (4.8), which may differ between fidelity levels, as feasibility constraints differ. At fidelity levels with low evaluation cost, data point generation is done using LHS. At fidelity levels where this method imposes prohibitive evaluation cost, we use the fact that the initial trajectory is on the feasibility boundary. As such, time allocations with the same ratio but shorter total time are infeasible, while time allocations with the same ratio but larger total time are feasible. This enables generation of data points without any additional evaluation cost by scaling the initial trajectory.

The initial trajectory is also used for the normalization of data points. This is required because feasibility constraints differ between the different levels of fidelity, leading to bias in the dataset. Moreover, time allocations between trajectory segments may be at different scales of magnitude, which may make the training process numerically unstable. To resolve these issues, time allocation for each trajectory segment is scaled with the time allocation of the corresponding segment in the initial trajectory at the same fidelity level.

4.4.4 Geometric Constraints

We consider two formulations to incorporate geometric constraints in the trajectory optimization problem: waypoint constraints and polytope constraints. In many applications, candidate trajectories are constrained to attain a pre-defined sequence of waypoints. By using a piecewise polynomial trajectory representation, the optimization can be formulated as an unconstrained quadratic program based on (4.4) and (4.2). In other scenarios, particularly when planning in obstacle-rich environments, waypoints may become an ineffective method to avoid collisions. Convex polytopes can instead be used to constrain the trajectory to the obstacle-free space without imposing waypoint restrictions. Each polynomial trajectory segment is now constrained to a polytope, leading to the constrained quadratic program (4.4) subject to (4.3). As described in Section 4.2, the function (4.6) that maps time allocation and geometric constraints to the corresponding minimum-snap trajectory can be formulated based on either waypoint constraints or polytope constraints. Hence, our proposed algorithm for multi-fidelity Bayesian optimization can be applied to either case without any modification.

In order to fully demonstrate the compatibility of our algorithm with polytope constraints, the remainder of this section details a procedure to construct such constraints. We focus on planning in 2-D obstacle-rich environments, and use convex decomposition [56] in combination with Dijkstra's algorithm [17]. Major steps of the resulting algorithm to generate the geometric constraints are shown in Fig. 4-3.

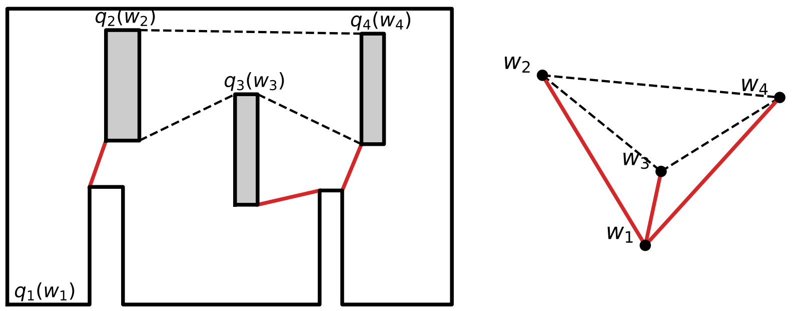
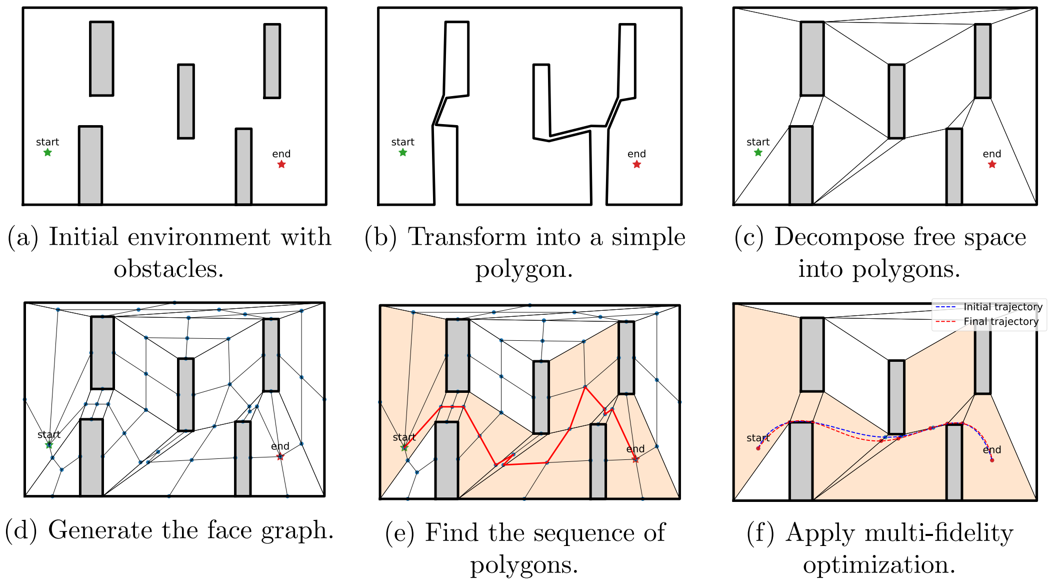


Figure 4-2: Polygon graph to convert free space into a simple polygon.



(a) Initial environment with obstacles.

(b) Transform into a simple polygon.

(c) Decompose free space into polygons.

(d) Generate the face graph.

(e) Find the sequence of polygons.

(f) Apply multi-fidelity optimization.

Figure 4-3: Algorithm to generate collision-free trajectories.

The free space of an environment, e.g., as shown in Fig. 4-3a, can be modeled as a polygon with inner holes. However, its decomposition into a set of convex polygons would be NP-hard [56]. Thus, we add bridges to transform the free space into *a simple polygon*, i.e., a polygon without intersections or inner holes. To place these bridges, we model the border of the environment and the obstacles as polygons $q_1, \cdots, q_n$, and we generate a complete graph with corresponding vertices $w_1, \cdots, w_n$. Each edge in the graph is weighted according to the minimum Euclidean distance between the corresponding obstacles. If an edge is included in the minimum spanning tree, a bridge is placed on the shortest path that connects the corresponding polygons. Since the bridges are added on the shortest line between the polygons, they will not intersect polygon faces or other bridges. Figure 4-3b shows the resulting simple polygon.

We obtain a convex decomposition of the resulting simple polygon using the method from [56]. The pseudocode in Alg. 2 represents the procedure, and Fig. 4-3c shows the resulting set of convex polygons. Given trajectory start and end positions, a connecting sequence of polygons can be found by various methods. In some scenarios, such as drone racing on a known track, the sequence of polygons can be selected manually or based on a guiding trajectory. If this is not possible, one may find the shortest path from start to end using a planning algorithm in geometrical space and select the polygon sequence through which the resulting path passes. For our proposed algorithm, we first generate a graph connecting the center point of each polygon and its faces, as shown in Fig. 4-3d. The shortest path that connects the start point and end point can then be found efficiently through combinatorial planning using Dijkstra's algorithm [17]. Finally, the sequence of polygons on the shortest path is selected to construct the geometric constraints, as shown in Fig. 4-3e.

Algorithm 2: Convex polygonal decomposition

```
   Input: A simple polygon $\mathbf{P}$ with vertices $v_1, \ldots, v_n$
   Output: $\mathcal{E}_d$
1  Definition:
2      Notch vertex: vertex with a reflex interior angle
3      Visibility pair $(v_i, v_j)$: set of vertices $(v_i, v_j)$ such
        that $edge(v_i, v_j)$ does not intersect with other faces
4  Function Decomp($\mathbf{P}$):
5      $\mathcal{E}_d = \emptyset$
6      $num\_edges = \infty$
7      for Notch vertex $v_i$ do
8          for $v_j \in \mathbf{P}$ where $(v_i, v_j)$ is visibility pair do
9              $\mathbf{P}_{i,j}$ = subpolygon with $v_i, v_{i+1}, \cdots, v_j$
10             $\mathbf{P}_{j,i}$ = subpolygon with $v_j, v_{j+1}, \cdots, v_i$
11             $\mathcal{E}_t$ = Decomp($\mathbf{P}_{i,j}$) $\cup$ Decomp($\mathbf{P}_{j,i}$)
12             if $|\mathcal{E}_t| < num\_edges$ then
13                 $num\_edges = |\mathcal{E}_t|$
14                 $|\mathcal{E}_d| = \mathcal{E}_t \cup \{edge(v_i, v_j)\}$
15             end
16         end
17     end
18     return $\mathcal{E}_d$
```

Let $\mathcal{Q}_i = \{\mathbf{p}_{i,0}, \mathbf{n}_{i,0}, \cdots, \mathbf{p}_{i,d_i-1}, \mathbf{n}_{i,d_i-1}\}$ denote the i-th polygon, where $\mathbf{p}_{i,j}$ and $\mathbf{n}_{i,j}$ are respectively the adjacent vertex and the normal vector of the j-th face, and d_i is the number of faces. The corresponding polynomial trajectory segment is constrained to be within the convex polygon by the following linear constraints:

$$\begin{aligned} A_i &= \left[\mathbf{n}_{i,0} \ \cdots \ \mathbf{n}_{i,d_i-1}\right]^\top, \\ b_i &= \left[\mathbf{n}_{i,0}^\top \mathbf{p}_{i,0} \ \cdots \ \mathbf{n}_{i,d_i-1}^\top \mathbf{p}_{i,0}\right]^\top, \\ A_i p\left(t\right) &\le b_i, \ \forall t \in \left[\sum\nolimits_{j=1}^{i-1} x_j, \sum\nolimits_{j=1}^{i} x_j\right]. \end{aligned} \tag{4.29}$$

For each polygonal cell, such a set of corresponding linear constraints is incorporated into (4.4).

4.4.5 Levels of Fidelity

Evaluations at three different fidelity levels are used for quadrotor trajectory planning. Low-fidelity evaluations are based on differential flatness of the quadrotor dynamics, which enables us to transform a trajectory and its time derivatives from the output space, i.e., position and yaw angle with derivatives, to the state and control input space, i.e., position, velocity, attitude, angular rate, and motor speeds [80]. The resulting reference control input trajectory $u(t) = \zeta(p, t)$ would enable a hypothetical perfect quadcopter to track the trajectory p. Feasibility of the control input values can be evaluated at relatively small computational cost, and as such can serve as a cheap, cursory evaluation of trajectory feasibility. The set of feasible trajectories at fidelity level l^1 is defined as

$$\mathcal{P}_T^{l^1} = \left\{ p \middle| \zeta(p,t) \in \left[\underline{u}, \bar{u}\right]^4 \quad \forall t \in [0, T] \right\}, \tag{4.30}$$

where $\underline{u}$ is the minimum and $\bar{u}$ is the maximum motor speed.

Medium-fidelity evaluations are obtained using the open-source multicopter dynamics and inertial measurement simulation from [35] with the trajectory tracking controller from [125]. The feasible set is defined as

$$\mathcal{P}_T^{l^2} = \left\{ p \middle| \, \|p_r(t) - r(t)\|_2 \leq \bar{r} \wedge |p_\psi(t) - \psi(t)| \leq \bar{\psi} \quad \forall t \in [0, T] \right\}, \tag{4.31}$$

where $\bar{r}$ is the maximum allowable Euclidean position tracking error, and $\bar{\psi}$ is the maximum allowable yaw tracking error. When planning in close proximity to obstacles, the vehicle may collide with an obstacle, even when tracking the reference trajectory within the bounds defined by (4.31). Therefore, we also exclude any trajectories that result in a collision from the feasible set. At this fidelity level, stochastic measurement and actuation noise, motor dynamics, and simplified aerodynamic effects are incorporated. We note that these factors are typically unfavorable, leading to the reduction of the feasible set compared to low fidelity. At the same time, the controller may be able to perform adequate tracking even if reference control inputs are infeasible, so that neither feasible set is a subset of the other. In order to account for stochastic effects, each evaluation consists of multiple simulations that all need to succeed for a trajectory to be deemed feasible.

High-fidelity evaluations are obtained from real-world experiments using a quadcopter and motion capture system. At this fidelity level, each evaluation incorporates dynamics of the full system, including actuation and sensor systems, vehicle vibrations, unsteady aerodynamics, battery performance, and estimation and control algorithms. This provides a highly accurate assessment of feasibility, but comes at great cost as it involves actual flying hardware and cannot be executed any faster than real-time. We use the same controller as in simulation, and again perform multiple flights to account for stochastic effects. The feasible set is identical to (4.31), and any trajectories resulting in a collision are again deemed infeasible. The overall objective of the algorithm is to find the time-optimal trajectory, subject to feasibility at the highest-fidelity model included.

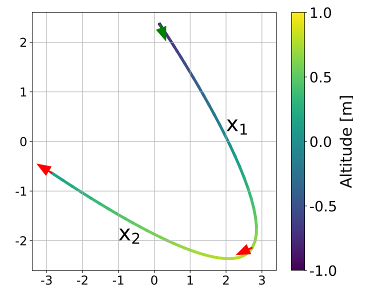
4.5 Experimental Results

We present experimental results that demonstrate the operation and performance of our proposed algorithm. First, we focus on a simple two-segment trajectory. Through visualization of the corresponding two-dimensional solution space, insights in the algorithm operation, particularly its acquisition function, are provided. Next, we present an extensive experimental evaluation of the proposed algorithm in two scenarios. For the first scenario, we apply the algorithm to obtain time-optimal trajectories based on a predetermined sequence of waypoints through an environment without any obstacles. For the second scenario, we define only start and end points and consider environments that include obstacles. We apply the proposed algorithm including polygonal decomposition to find a time-optimal trajectory that avoids the obstacles. In both scenarios, we evaluate the proposed algorithm in a *simulation environment*, and in a *hybrid environment* including simulation and real-world experiments.

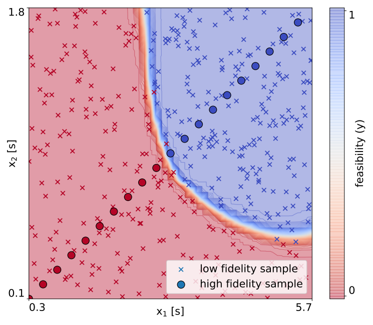
4.5.1 Two-Segment Trajectory

In order to illustrate the operation of our proposed algorithm, we present results for the simple two-segment trajectory shown in Fig. 4-4a. The yaw rate, velocity, acceleration, and jerk are constrained to zero at the first waypoint. Results were obtained in the simulation environment, where we incorporate low-fidelity evaluations using reference control input feasibility, and medium-fidelity evaluations using the multicopter simulation. For the medium-fidelity evaluations, we set the maximum Euclidean position tracking error to 20 cm and the maximum yaw tracking error to 15 deg. To reflect the difference in evaluation cost between evaluations, we set the parameters of the acquisition functions (4.24) and (4.26) as $C_{l^1} = 1$ and $h_{l^1} = 0.1$ corresponding to the low-fidelity evaluations, and $C_{l^2} = 10$ and $h_{l^2} = 0.4$ corresponding to the medium-fidelity evaluations, and $\beta = 3.0$.

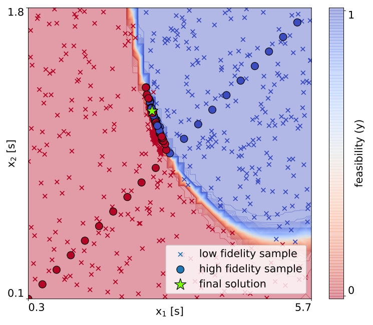
As shown in Fig. 4-4b, the low-fidelity dataset is initialized using LHS of 400 data points, and the medium-fidelity dataset is initialized using 20 evenly scaled data points. Given the low dimension of the two-segment time allocation, third-order finite differencing as required by (4.28) cannot be applied. Instead, we resort to LHS for generation of candidate solutions for the acquisition function. We run the algorithm 20 times for 50 iterations using different random seeds. Each medium-fidelity evaluation along with preceding low-fidelity evaluations is considered a single iteration, and the number of low-fidelity evaluations per iteration is limited to 20. This limit is imposed to prevent the acquisition function from selecting too many successive low-fidelity evaluations, which may otherwise occur at iterations where the selected acquisition parameters do not behave well with the state of the medium-fidelity surrogate model.



(a) Trajectory with waypoints.



(b) Initial medium-fidelity feasibility.



(c) Final medium-fidelity feasibility.

Figure 4-4: Waypoints and feasibility maps for two-segment trajectory optimization in the simulation environment.

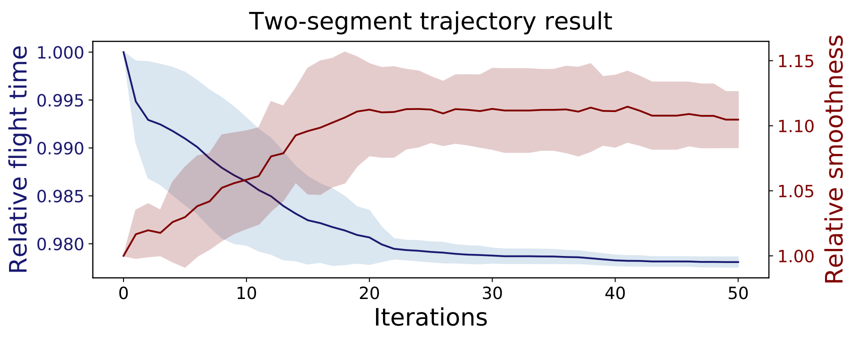


Figure 4-5: Mean and standard deviation of relative trajectory time and smoothness for two-segment trajectory, obtained over 20 random seeds in the simulation environment.

Figure 4-4b and Fig. 4-4c show respectively the initial and the final medium-fidelity feasibility maps for one of the runs. Note that all medium-fidelity evaluations are in proximity of the final solution, as this promising region was first found by low-fidelity evaluations. To evaluate our algorithm, we compare the total trajectory time of the best solution at each iteration with the trajectory time obtained by (4.8) using the same feasibility evaluation constraints. Figure 4-5 shows a flight time improvement of approximately 2% compared to the initial trajectory. Additionally, we compare the trajectory smoothness given by (4.5). For this comparison, we scale the total trajectory time to the value obtained by (4.8), while maintaining the allocation ratio found by our algorithm. Thereby, the comparison is based solely on the difference in allocation ratio, and not affected by total trajectory time. Figure 4-5 shows that our algorithm selects time allocation ratios that result in increased snap and yaw acceleration at the same total trajectory time. Despite this, the corresponding trajectories are still feasible at total trajectory times smaller than the initial trajectories. This shows that our algorithm is able to find an allocation ratio that compares favorably to the one found by (4.7) and (4.8), and is thereby able to find feasible trajectories with smaller total trajectory time. Indeed, we can conclude that plain snap minimization does not result in the time-optimal trajectory. Our approach based on modeling of realistic feasibility constraints converges to a quicker feasible trajectory with increased snap. While the trajectory duration is only improved by a modest 2%, this does illustrate and validate the rationale behind our proposed multi-fidelity trajectory optimization algorithm. In Section 4.5.2 and Section 4.5.3, we show that larger improvements can be obtained for more complicated multi-segment trajectories.

4.5.2 Waypoint Trajectories

Next, we apply our algorithm to the more complicated multi-segment trajectories shown in Fig. 4-6. For each trajectory, yaw rate, velocity, acceleration, and jerk are constrained to zero at the first and final waypoints. The trajectories range from 9 to 19 segments. Due to the corresponding increase in dimensionality of the solution space, the minimum-jerk perturbation given by (4.28) can now be used to generate the candidate solution set. The scaling factor γ is set to 0.2.

We present results for all eight multi-segment trajectories using the simulation environment, which incorporates low-fidelity evaluations using reference control input feasibility and medium-fidelity evaluations using the multicopter simulation. Additionally, we apply

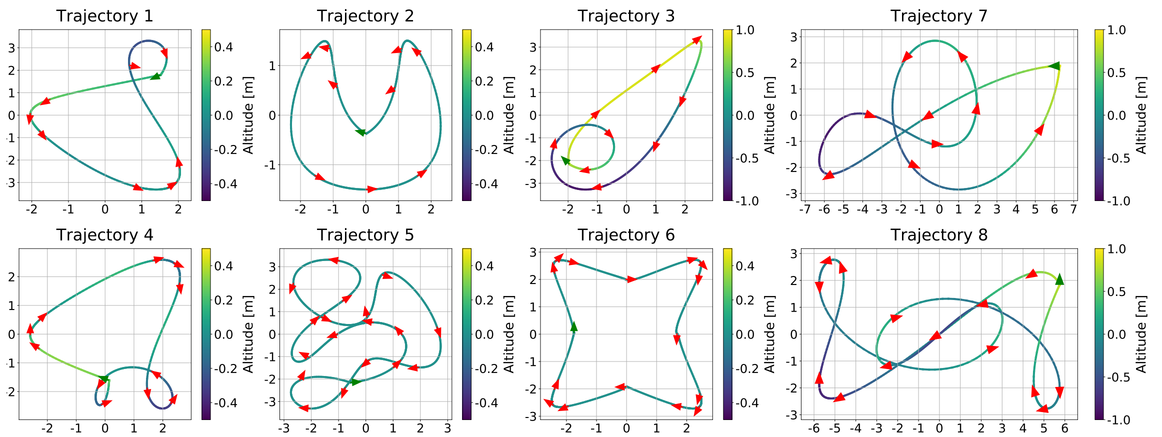


Figure 4-6: Multi-segment trajectories with starting point and subsequent waypoints indicated by green and red arrows, respectively.

our algorithm to a subset of three trajectories in the hybrid environment, which incorporates medium-fidelity evaluations using the multicopter simulation and high-fidelity evaluations using real-world flight experiments. We chose not to incorporate low-fidelity evaluations in the hybrid environment, because their evaluation time is relatively very similar to medium-fidelity evaluations when compared to the time of flight experiments. Consequently, they provide less valuable data points at similar cost. In both environments, we set the parameters of the acquisition functions (4.24) and (4.26) as $C_{l^j} = 1$, $C_{l^{j+1}} = 10$, $h_{l^j} = 0.1$, $h_{l^{j+1}} = 0.4$, and $\beta = 3.0$ where l^j is the lowest fidelity in the environment. We set the maximum Euclidean position tracking error to 20 cm and the maximum yaw tracking error to 15 deg for all medium and high-fidelity evaluations.

For the experiments in the simulation environment, we initialize the surrogate model using 1000 low-fidelity samples. The maximum amount of low-fidelity evaluations per iteration is set to 50. The results in Fig. 4-7 show that the algorithm is able to significantly reduce the trajectory time for each of the eight evaluated trajectories. The largest improvement is obtained for *Trajectory 3*, where the flight time is reduced by 22% on average. The optimized trajectory is feasible despite having almost four times larger snap and yaw acceleration, which shows our algorithm's capability to find faster feasible trajectories. This is achieved by changing the time allocation ratio between trajectory segments, as shown in Fig. 4-8. Figure 4-9 shows that the resulting trajectories have an increased speed on most segments, but also slow down on some segments. Speed decreases often occur during or just prior to tight turns and help maintain tracking feasibility as the speed over preceding and subsequent segments is increased. This interaction between trajectory segments is not captured by the low-fidelity control input evaluations, and demonstrates how using multi-fidelity modeling towards a holistic approach to optimization results in faster feasible trajectories.

In the hybrid environment, we optimize three of the waypoint trajectories shown in Fig. 4-6 based on medium-fidelity evaluations using the multicopter simulation and high-fidelity evaluations using real-world quadrotor flight experiments. The medium-fidelity dataset is initialized using LHS and the high-fidelity dataset using the scaling method. At each optimization step, the acquisition function selects the type of evaluation (i.e., simulation or real-world experiment) and updates the feasibility model based on the evaluation result.

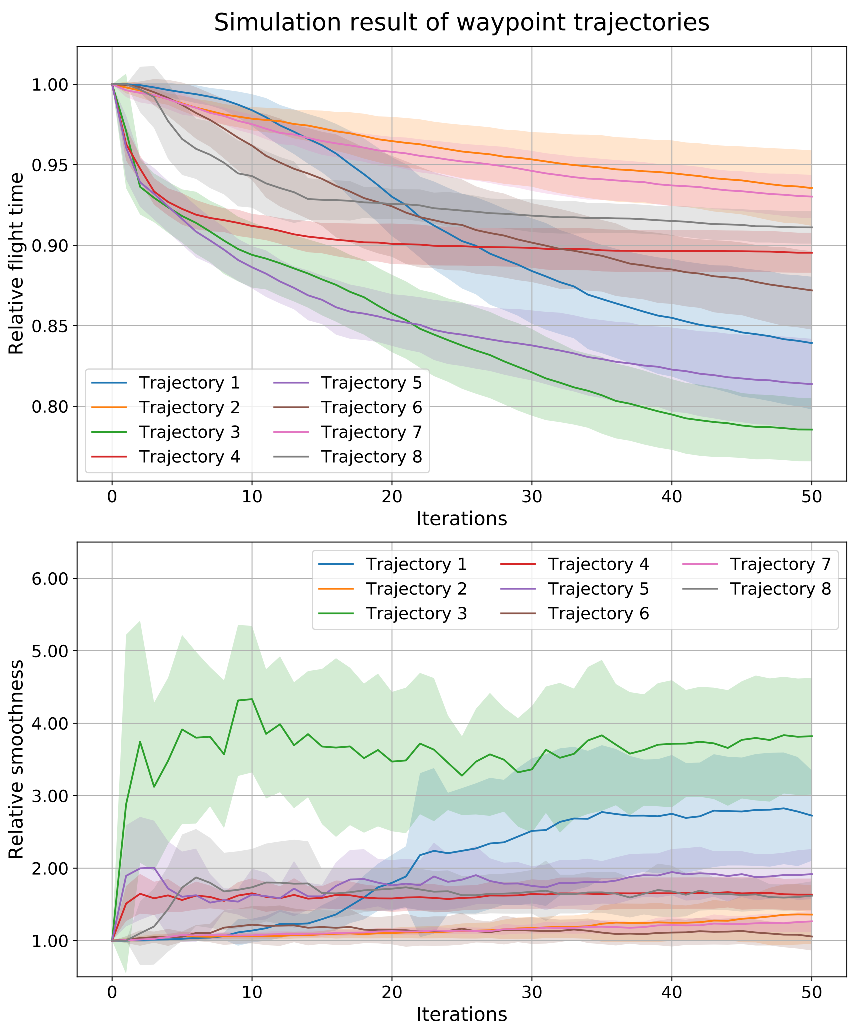


Figure 4-7: Mean and standard deviation of relative trajectory time and smoothness for multi-segment trajectories, obtained over 20 random seeds in the simulation environment.

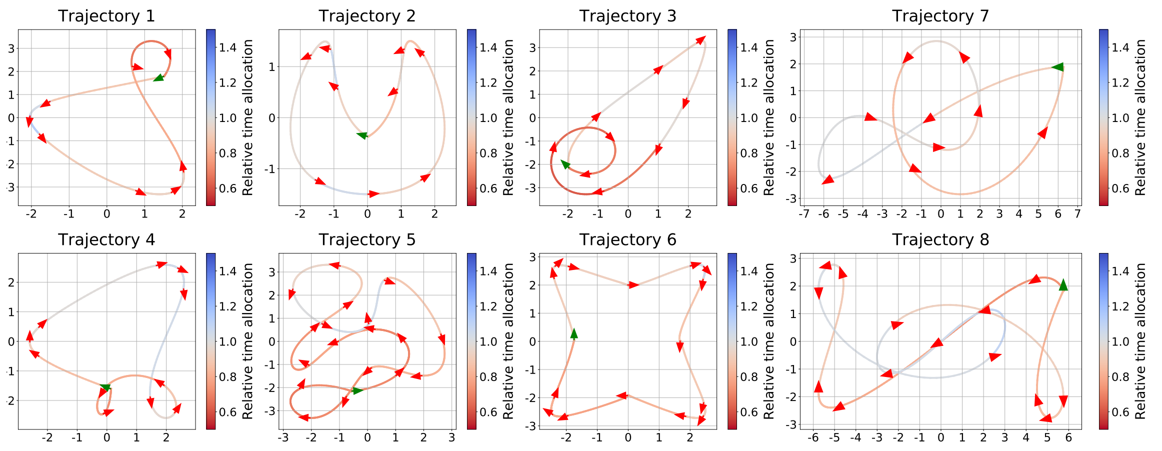


Figure 4-8: Average relative time allocation of initial and optimized multi-segment trajectories, obtained over 20 random seeds in the simulation environment.

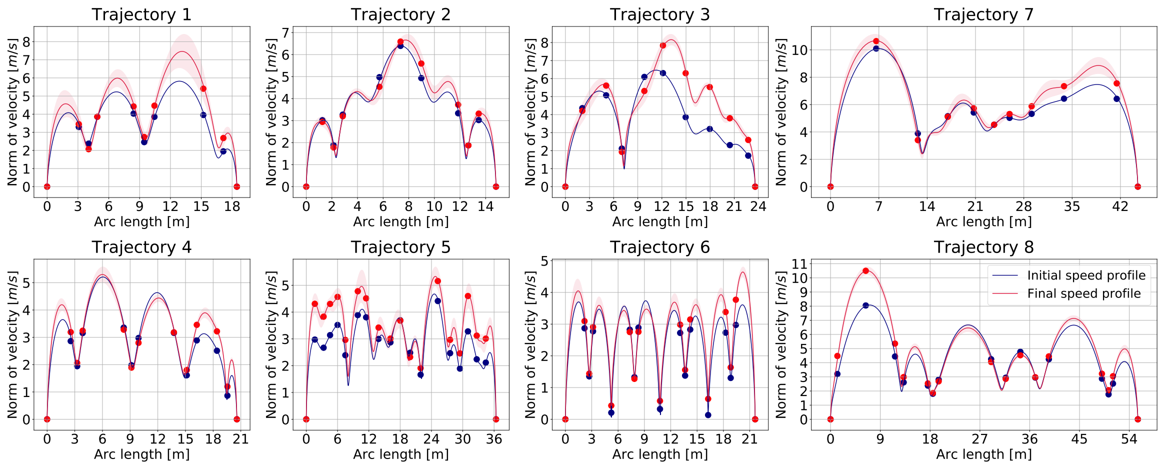


Figure 4-9: Speed profiles of initial and optimized multi-segment trajectories, obtained over 20 random seeds in the simulation environment. Shading indicates standard deviation.

The corresponding computation time varies between 5 and 10 seconds, depending on the convergence time of the Gaussian process feasibility model. The time spent on computation is relatively insignificant in the hybrid environment, since real-world flight experiments take several minutes on average when including steps like battery replacement. Comparison of Fig. 4-7 to Fig. 4-10 shows that the obtained flight time improvements are similar to those from the simulation environment. However, Fig. 4-11 and Fig. 4-12 show time allocations different from those obtained in simulation (cf. Fig. 4-8 and Fig. 4-9). In fact, the time allocation ratios obtained in the simulation environment may not be feasible in flight with the real-world vehicle, as it may behave differently than the medium-fidelity simulation. This underlines the importance of incorporating real-world flight experiments in trajectory optimization. At the same time, the inclusion of medium-fidelity simulation evaluations is essential to find effective samples for high-fidelity evaluations, which enables the algorithm to lower the number of required real-world flights. Without this feature, optimization of the trajectories in Fig. 4-11 would be infeasible, as it amounts to search over a 9 or 10-dimensional solution space.

The optimized trajectories in the hybrid environment show similar characteristics as in the simulated environment. The time allocation is reduced for most segments, but also increased for some in order to maintain feasibility. It can be seen that the vehicle decelerates before the turns and accelerates through them. During the experiments, we observed that entering turns with reduced speed stabilizes tracking on the remainder of the trajectory, which can eventually reduce the overall flight time. We also found that our vehicle is generally able to stabilize to static hover very quickly, allowing it to finish the trajectory quite aggressively.

4.5.3 Polytope Trajectories

In this section, our proposed algorithm including convex decomposition is applied to the planning of trajectories between start and end points in environments with obstacles, as shown in Fig. 4-13. Velocity, acceleration, and jerk are constrained to zero at the start and end points, and yaw and yaw rate are constrained to zero for the whole trajectory. We focus on planning in environments with two-dimensional polygonal obstacles, and constrain the

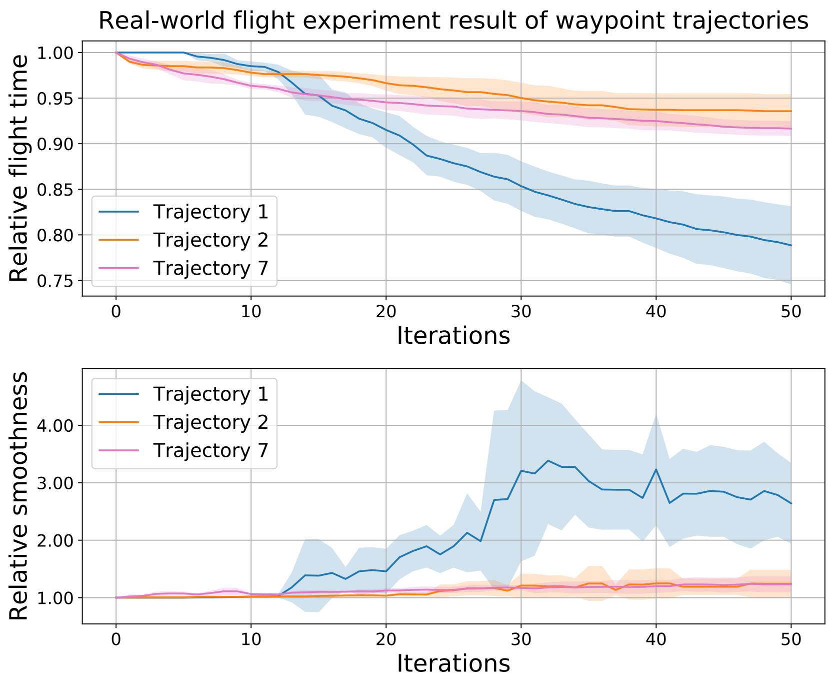


Figure 4-10: Mean and standard deviation of relative trajectory time and smoothness for multi-segment trajectories, obtained over 5 random seeds in the hybrid environment using simulation and real-world flights.

height to be constant throughout each trajectory. The acquisition function hyperparameters and the method for generating the initial candidate solution set are identical to those described in Section 4.5.2.

We first verify the algorithm for constructing polytope constraints by comparing the minimum-snap segment time allocations obtained from (4.7) for waypoint and polytope constraints. The waypoint sequences are obtained using straight-line RRT*, as proposed in [101]. For both types of constraints, the total trajectory times are scaled down according to (4.8) based on the medium-fidelity simulation with the maximum Euclidean position tracking error set to 20 cm and the maximum yaw tracking error set to 15 deg. The resulting waypoint and polytope trajectories are shown in blue in Fig. 4-14 and Fig. 4-13, respectively. The corresponding flight times are given in the first two columns of Table 4.1. It can be seen that the polytope trajectories are all faster than the waypoint trajectories. This is not surprising, as the waypoints represent stricter constraints on the trajectory geometry. As they are set without considering any vehicle model, they may inhibit finding time-optimal trajectories when considering a dynamics model. Hence, polytope constraints are particularly advantageous to our application. Additionally, we note that obstacle avoidance is imposed throughout the reference trajectory by using polytope constraints, whereas this is only the case at the actual waypoints for waypoint constraints. Consequently, it may be necessary to add additional waypoints for collision avoidance, which further inhibits trajectory optimization.

Our proposed algorithm for multi-fidelity Bayesian optimization is used to optimize the time allocation ratio for each trajectory in Fig. 4-13 using the simulation environment, which incorporates low-fidelity evaluations based on reference control input feasibility and

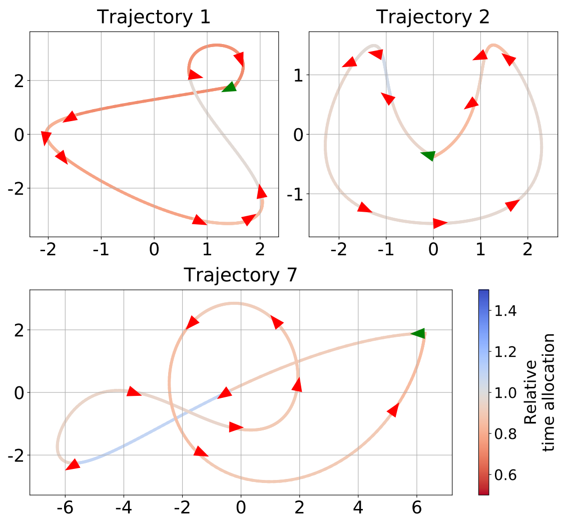


Figure 4-11: Average relative time allocation of initial and optimized multi-segment trajectories, obtained over 5 random seeds in the hybrid environment using simulation and real-world flights.

Table 4.1: Comparison of flight times for trajectories obtained through minimum-snap planning with waypoint and polytope constraints, and our proposed algorithm for multi-fidelity Bayesian optimization (MFBO). Final column lists mean and standard deviation obtained over 20 random seeds in the simulation environment.

	Min-snap (waypoint)	Min-snap (polytope)	MFBO (**Ours**)
Traj. 9	3.302 s	3.116 s	2.717 $\pm$ 0.0954 s
Traj. 10	4.490 s	3.743 s	3.538 $\pm$ 0.104 s
Traj. 11	5.637 s	5.110 s	4.704 $\pm$ 0.0706 s
Traj. 12	5.637 s	3.364 s	3.028 $\pm$ 0.0760 s

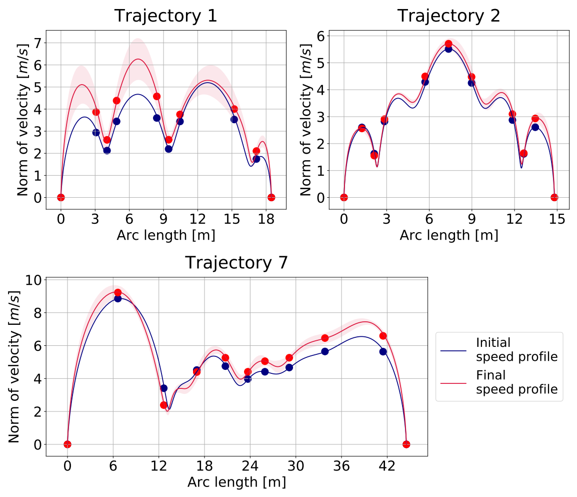


Figure 4-12: Speed profiles of initial and optimized multi-segment trajectories, obtained over 5 random seeds in the hybrid environment using simulation and real-world flights. Shading indicates standard deviation.

medium-fidelity evaluations based on the multicopter simulation. By comparing the blue and red lines, it can be seen that the leeway provided by the polytope constraints is exploited to subtly adapt the trajectory geometry. The resulting decreased flight times are given in the final column of Table 4.1. Figure 4-15 and Fig. 4-16 show the corresponding change in speed profile and time allocation. Our proposed algorithm is able to reduce flight times by adjusting the time allocation such that the reference trajectory is further away from the obstacles. In practice, this means that its feasibility is less sensitive to tracking error. Hence, the tracking error bounds can be exploited to a larger degree, before the trajectory becomes infeasible due to an obstacle violation, resulting in faster trajectories.

In the hybrid environment, we optimize one of the polytope trajectories by incorporating medium-fidelity evaluations based on the multicopter simulation and high-fidelity evaluations based on real-world flight experiments. The real-world flights were performed in the space shown in Fig. 4-17. Figure 4-18 shows that the trajectory geometry was significantly adapted during the Bayesian optimization process. The resulting trajectory is further away from the first two obstacles. Additionally, the speed is decreased in proximity of the first obstacle, as can be seen in Fig. 4-19. Subsequent trajectory segments can now be performed at significantly increased speed, resulting in an average flight time reduction of over 20%, as shown in Fig. 4-20.

4.6 Summary

We presented an algorithm for the generation of dynamically-feasible time-optimal quadrotor trajectories based on multi-fidelity Bayesian optimization. The vehicle feasibility constraints are efficiently modeled by incorporating evaluations from analytical approximation, numerical simulation, and real-world experiments in multi-fidelity Gaussian process classification. We conducted extensive evaluations through simulation and real-life experiments for both waypoint-constrained and polytope-constrained trajectories. It was found that the algorithm is able to generate feasible trajectories that are significantly faster than those ob-

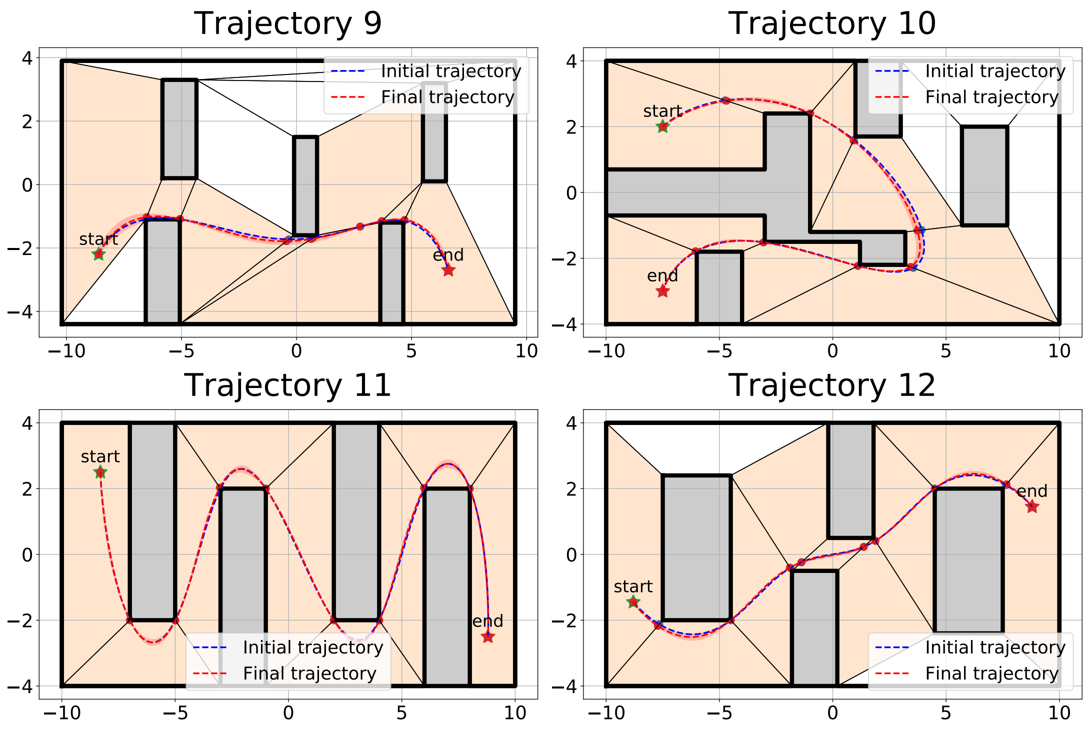


Figure 4-13: Initial and average optimized polytope trajectories, obtained over 20 random seeds in the simulation environment. Shading indicates standard deviation.

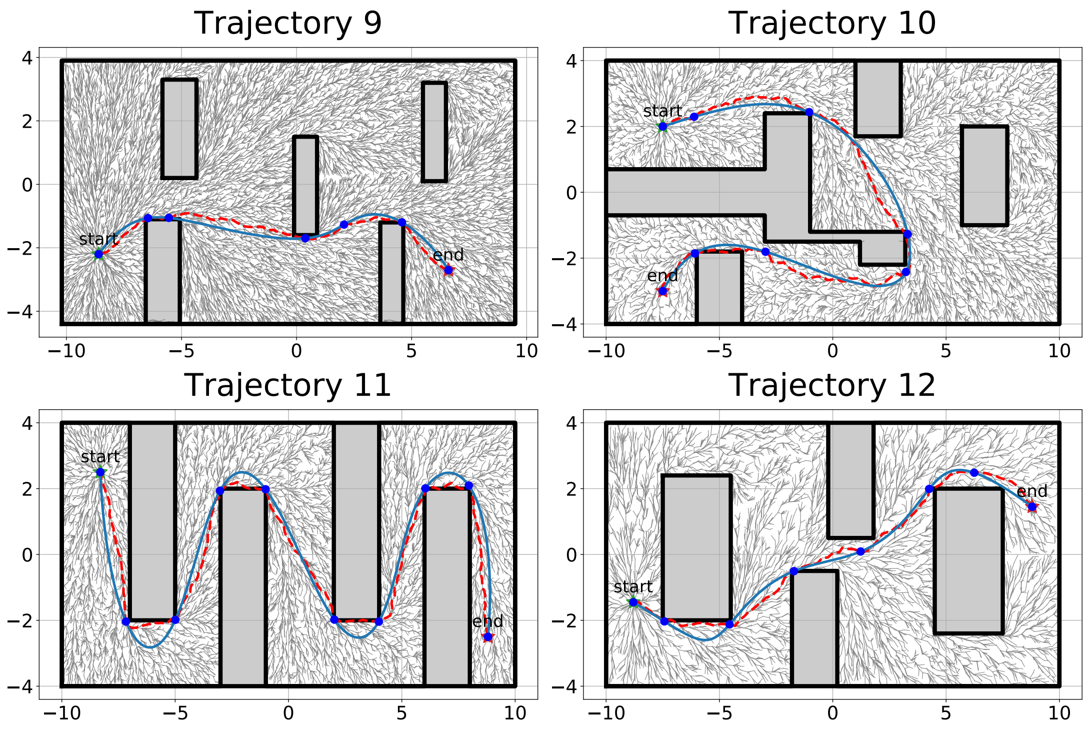


Figure 4-14: Trajectories based on waypoints from straight-line RRT*.

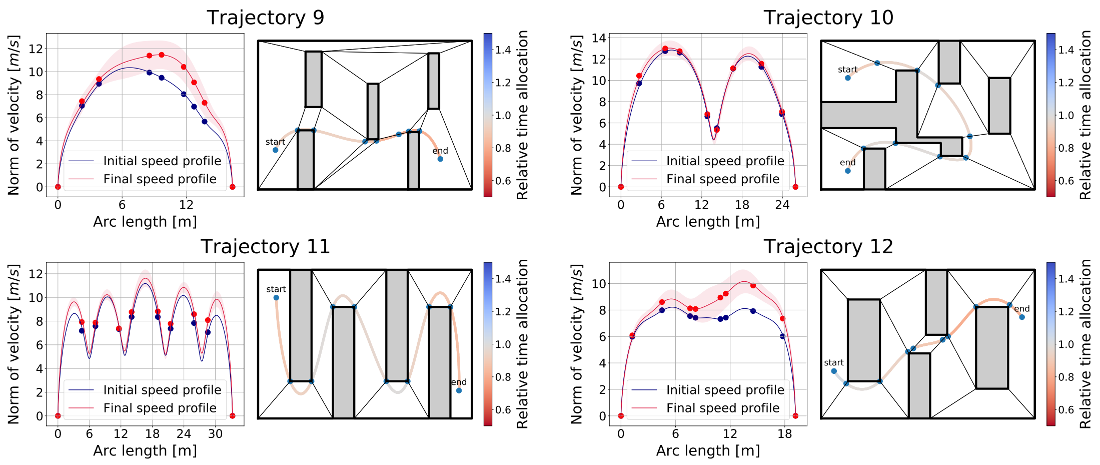


Figure 4-15: Speed profiles and average relative time allocation of initial and optimized polytope trajectories, obtained over 20 random seeds in the simulation environment. Shading indicates standard deviation.

tained from existing minimum-snap trajectory optimization algorithms. This demonstrates the capability of our algorithm to efficiently model the feasibility boundary and ultimately plan faster trajectories by incorporating multi-fidelity data, including flight experiments.

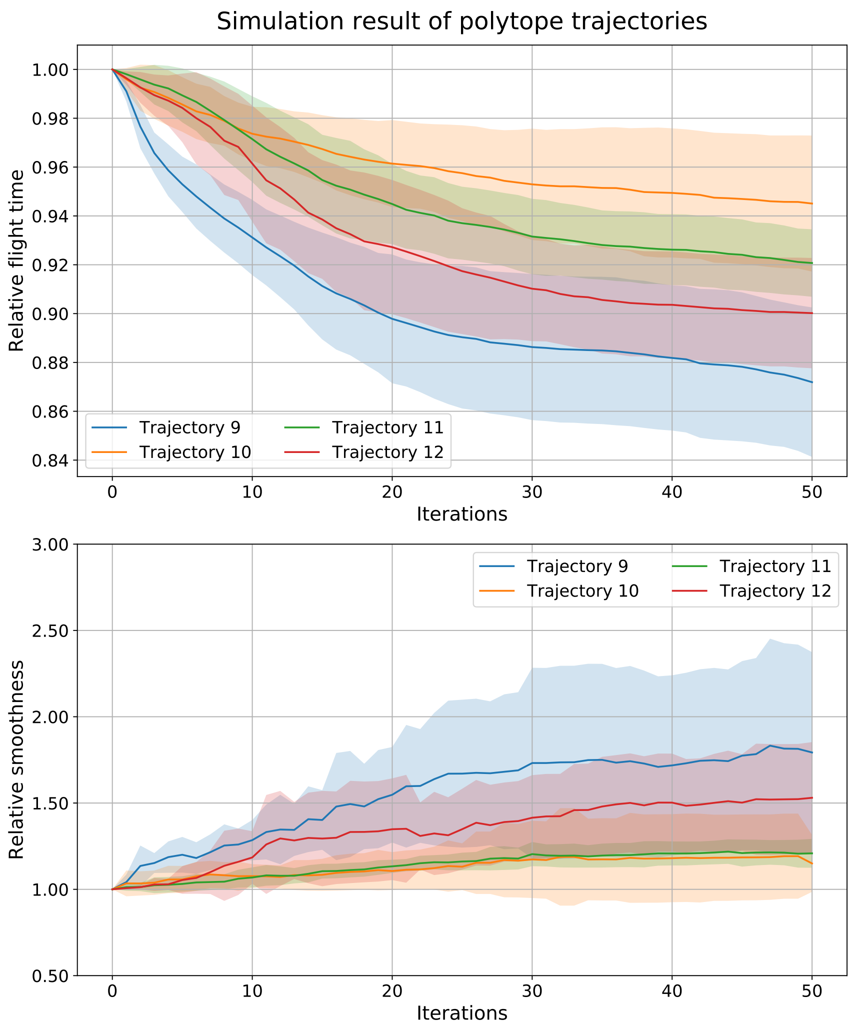


Figure 4-16: Mean and standard deviation of relative trajectory time and smoothness for polytope trajectories, obtained over 20 random seeds in the simulation environment.

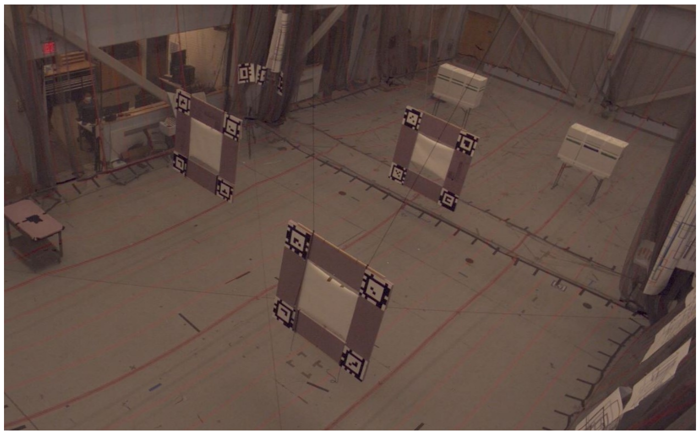


Figure 4-17: Environment for the real-world flight experiments for *Trajectory 9* (cf. Fig. 4-18).

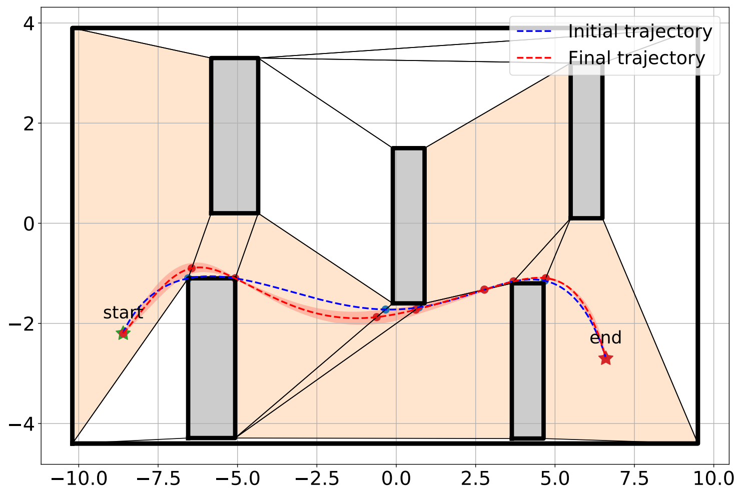


Figure 4-18: Initial and average optimized polytope trajectories, obtained over 5 random seeds in the hybrid environment using simulation and real-world flights. Shading indicates standard deviation.

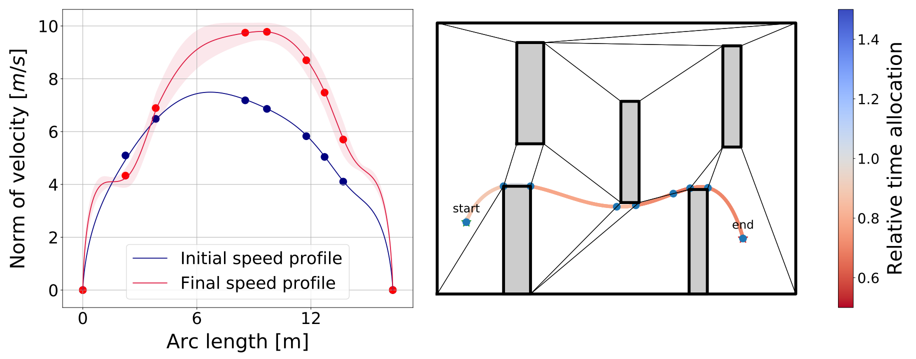


Figure 4-19: Speed profiles and average relative time allocation of initial and optimized polytope trajectory, obtained over 5 random seeds in the hybrid environment using simulation and real-world flights. Shading indicates standard deviation.

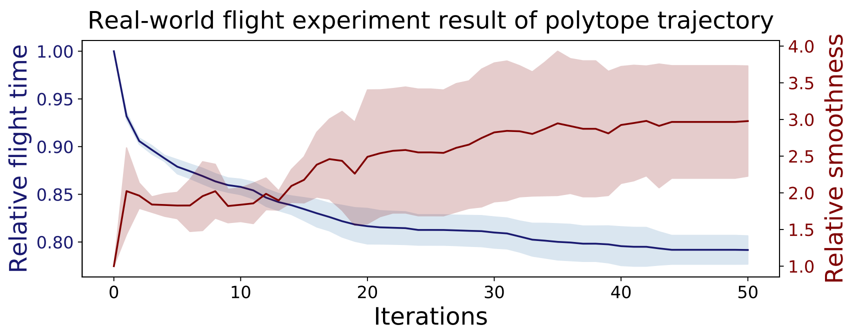


Figure 4-20: Mean and standard deviation of relative trajectory time and smoothness for polytope trajectory, obtained over 5 random seeds in the hybrid environment using simulation and real-world flights.

Chapter 5

Aerobatic Trajectory Generation for a VTOL Fixed-Wing Aircraft Using Differential Flatness

In this chapter, we propose an algorithm for aerobatic trajectory generation in the flat output space of a vertical take-off and landing (VTOL) fixed-wing aircraft. Existing methods for fixed-wing aircraft trajectory generation are often based on simplified models that are unsuitable for aerobatic trajectories. By exploiting differential flatness of the global 6DOF tailsitter flying wing flight dynamics, our algorithm efficiently generates trajectories that exploit the entire flight envelope, including uncoordinated and inverted flight conditions. We present extensive analysis and experimental results to establish the suitability of the flight dynamics model to determine the feasibility of candidate trajectories. Using our proposed algorithm, we present reference trajectories and experimental results for six fast and agile aerobatic maneuvers and a racing trajectory through a sequence of gates.

5.1 Introduction

The generation and tracking of fast and agile aircraft maneuvers have been topics of interest in academic research. In recent years, we have seen increasingly impressive demonstrations that keep pushing the boundary in terms of speed, acceleration etc. We observe that a large majority of this work focuses on rotorcraft, in particular, multicopters (and to a lesser degree unmanned helicopters) [31, 33, 81, 85]. Their simple design, lack of stability, and typically high thrust-to-weight ratio make multicopters an ideal platform for research into agile maneuvering. At the same time, it is well known that multicopters have several disadvantages compared to small unmanned fixed-wing aircraft. Most notably, they are often inefficient at increased speeds, leading to limited range and endurance.

Vertical take-off and landing (VTOL) fixed-wing aircraft can combine many of the advantages traditionally associated with either fixed-wing aircraft or rotorcraft, like multicopters. As described in Chapter 3, the unmanned tailsitter flying wing is capable of efficient cruise flight where lift is produced by its wing, and at the same time possesses many favorable qualities for fast and agile flight. The aircraft does not rely on any tilting components but instead changes attitude in its entirety when transitioning between hover and forward flight. Its relatively simple mechanical design—involving only a wing, two rotors, and two flaps—enables lightweight construction, resulting in a high thrust-to-weight ratio, especially when combined with powerful brushless motors. In addition, the absence of a vertical tail surface reduces directional stability, which makes the aircraft more maneuverable and makes it easier to enter and maintain uncoordinated flight. The combination of efficiency, speed, and agility enables aircraft with increased range and endurance to maintain the capability to quickly navigate cluttered environments. This versatility is relevant to many real-world applications. For example, transitioning search and rescue aircraft can cover large areas efficiently and closely inspect areas of particular interest. Similarly, delivery drones with an extended range can make time-critical deliveries without requiring a dedicated landing area.

In this chapter, we present an algorithm for generating fast and agile trajectories for a tailsitter flying wing aircraft. Our proposed algorithm is capable of generating aerobatics maneuvers that exploit the entire flight envelope of the vehicle, including challenging conditions, such as sideways knife edge flight and inverted flight. We make extensive use of the differential flatness property of the tailsitter dynamics that was shown in Chapter 3.

Trajectory generation algorithms for fixed-wing aircraft often neglect the flight dynamics and instead use kinematics models. An extension of Dubins paths can be used to find the time-optimal trajectory with curvature constraints [12]. While accurate tracking of the resulting paths is not dynamically feasible due to the instantaneous acceleration changes needed to transition between straight lines and circular arcs, feedback control can be used to maintain a tracking error that is acceptable in calm flight [90]. When considering fast and agile flight, the aircraft dynamics and control input constraints must be considered in trajectory generation so that the resulting trajectory is dynamically feasible. Trajectory optimization subject to the 6DOF nonlinear flight dynamics model is computationally costly, e.g., optimization of the 4.5 m knife edge maneuver presented by [3] takes 3–5 minutes of computation time (using direct collocation with twelve states and five control inputs), according to [8]. This can be partially addressed by offline computation, e.g., by using a sampling-based planner with pre-computed maneuvers [63].

In the context of trajectory generation, differential flatness enables transformation of trajectories from the flat output space to the state and control input space [24, 73]. This property is widely applied towards computationally efficient trajectory generation for quadcopters by initially generating the trajectory in the flat output space consisting of the three-dimensional position and the yaw angle [80, 101]. Differential flatness of fixed-wing aircraft dynamics has also been considered [73]. However, as far as we are aware, differential flatness of a global dynamics model for a transitioning aircraft had not been shown prior to our derivation in Chapter 3. The application of differential flatness towards trajectory generation for fixed-wing aircraft has mostly been limited to kinematics or simplified dynamics models. Existing works consider path generation and tracking using a differentially flat coordinated flight model [38] and aerobatics maneuvers using an aircraft kinematics model that does not incorporate angle of attack or sideslip angle [36]. The algorithm presented in [8] is based on the differentially flat coordinated flight model given in [38] and combines Dubins paths with a transverse polynomial offset to obtain smooth trajectories.

Our method differs from existing flatness-based approaches for fixed-wing trajectory generation, as it considers a global 6DOF flight dynamics model, including aerodynamics equations. By using a global dynamics model, our method is able to generate aerobatics maneuvers that exploit the entire flight envelope, enabling agile maneuvering through the stall regime, sideways uncoordinated flight, inverted flight etc. We observe that the corresponding tailsitter flatness transform that was derived in Section 3.3 resembles the well-known quadcopter flat transform given in Section 2.3.3, in that snap and yaw acceleration roughly correspond to the control inputs. Hence, reduction of snap and yaw acceleration may also increase feasibility of tailsitter trajectories, similarly to minimum-snap trajectory generation of quadcopter trajectories [80,101]. Potentially, this enables the application of efficient algorithms for minimum-snap trajectory generation in the flat output space towards generation of tailsitter aerobatics trajectories.

In this chapter, we show that minimum-snap trajectory generation is indeed suitable for generation of fast and agile aerobatics trajectories for the tailsitter flying wing aircraft, and we demonstrate tracking of these trajectories in flight experiments. The chapter contains several contributions. Firstly, we propose an approach for trajectory generation for a VTOL fixed-wing aircraft using differential flatness. As far as we are aware, this is the first algorithm that uses differential flatness of a realistic flight dynamics model to generate fast and agile flight trajectories for a fixed-wing aircraft. Secondly, we analyze the flatness transform derived in Chapter 3 to illustrate how differential flatness is used to generate dynamically consistent maneuvers through the transition region, without depending on heuristics or predesigned references. Thirdly, we provide elaborate experimental results that establish the suitability of the flat model dynamics to determine feasibility of candidate trajectories. Fourthly, we present trajectories and experimental results for six well-known aerobatics maneuvers, and we apply the algorithm for multi-fidelity Bayesian optimization that was proposed in Chapter 4 to generate a time-optimal racing trajectory at the limit of the vehicle's capability.

The outline of this chapter is as follows. Section 5.2 presents the objective function for minimization of snap and yaw acceleration. In Section 5.3 and Section 5.4, we analyze the flatness transform to illustrate the flight dynamics of transition, and we present experimental results to validate the capability of the flat dynamics model to predict feasibility of candidate trajectories, respectively. Experimental results for aerobatics maneuvers and for multi-fidelity Bayesian trajectory optimization are presented in Section 5.5 and Section 5.6, respectively.

5.2 Minimization of Snap and Yaw Acceleration

We exploit differential flatness of the global tailsitter dynamics to generate dynamically feasible aerobatics trajectories without resorting to computationally expensive state-space methods. Since dynamic feasibility (i.e., whether a trajectory can be accurately tracked by the real vehicle) is determined in large part by the control input constraints, we aim to find maneuvers for which accurate tracking requires only permissible control inputs. The corresponding control input constraints cannot readily be enforced in the flat output space, so we separate trajectory generation from feasibility checking. We first generate the trajectory in the flat output space, and then check feasibility based on the corresponding control input trajectory that is obtained through the flat transform.

In Section 3.3, we showed differential flatness of the tailsitter dynamics with the flat output

$$\boldsymbol{\sigma}(t) = [\mathbf{x}(t)^\top \ \psi(t)]^\top, \tag{5.1}$$

consisting of the vehicle position $\mathbf{x}$ and its yaw angle ψ. Upon examination of the derived flat transform, we find that the required control inputs are directly related to the control moment, which in turn is a function of the flat output and its derivatives up to snap and yaw acceleration. The relation between the trajectory snap and yaw acceleration and the required control inputs is reminiscent of the flat transform for the quadcopter dynamics, making snap minimization a natural approach for trajectory generation in the flat output space [80]. In practice, reducing the snap and yaw acceleration typically corresponds to reducing the required control inputs, and thus to increasing the likelihood that the trajectory is feasible.

Minimum-snap trajectory generation with waypoint constraints is formulated as follows:

$$\begin{aligned} \underset{\boldsymbol{\sigma},\mathbf{t}}{\text{minimize}} \quad & \int_0^T \left\| \frac{d^4\mathbf{x}}{dt^4} \right\|^2 + \mu_\psi \left(\frac{d^2\psi}{dt^2} \right)^2 dt \\ \text{subject to} \quad & \boldsymbol{\sigma}\left(\sum\nolimits_{j=1}^{i} t_j \right) = \tilde{\boldsymbol{\sigma}}_i,\ i = 0,\ \ldots,\ m, \\ & \sum\nolimits_{j=1}^{m} t_j = T, \end{aligned} \tag{5.2}$$

where μ_ψ is a weighing parameter and T is the total trajectory time. The nonnegative vector $\mathbf{t}$ represents the time allocation over the trajectory segments between the $m+1$ waypoints $\tilde{\boldsymbol{\sigma}}$ that must be attained in order. We utilize piecewise polynomial functions to describe the trajectory, so that differentiability constraints can straightforwardly be enforced and the optimization can be solved efficiently [101]. In order to obtain aggressive trajectories that exploit the tailsitter's capability to perform agile maneuvers, we first solve (5.2) with an estimate for T based on the distance between waypoints, and then scale the resulting time allocation $\mathbf{t}$ to obtain the quickest minimum-snap trajectory that is in the feasible set given by

$$\Sigma_T = \left\{ \boldsymbol{\sigma} \middle| \mathbf{u}(t) \in \mathcal{U} \quad \forall t \in [0, T] \right\}, \tag{5.3}$$

where $\mathbf{u}$ is the control input trajectory corresponding to $\boldsymbol{\sigma}$ and $\mathcal{U}$ is the set of permissible control inputs, i.e., a bounded set defined by the minimum and maximum allowed rotor speeds and flap deflections.

As it is based on the widely used minimum-snap objective function, our algorithm can be extended in many directions based on existing methods, e.g. joint minimization of snap and total trajectory time [101], and various formulations for obstacle avoidance [15,28,127]. In Section 5.6, we combine the trajectory generation algorithm with the method presented in Chapter 4 to optimize the time allocation $\mathbf{t}$ using experimental evaluations.

5.3 Flatness Transform

We use the differentially flat flight dynamics model presented in Chapter 3 to enable efficient trajectory generation in the output space. Crucially, the flat transform (see Section 3.3) provides a mapping from the flat output space to the state and control input space. Since it is a bijective mapping, any agile maneuver or flight condition defined in the state space of the 6DOF flight dynamics model can be obtained efficiently from a corresponding trajectory in the flat output space. Consequently, trajectory generation in the flat output space by no means limits the state-space aerobatics trajectories that can be generated.

In this section, we examine how the flatness transform provides natural solutions for maneuvers through the transition region between hover and horizontal flight. We examine straight-and-level flight conditions varying from hover to forward flight at the maximum speed, forward and sideways transitions, and the transition from thrust vector tilting for lateral acceleration in hover to banking for lateral acceleration in forward flight.

5.3.1 Straight-and-Level Flight

Before examining transition maneuvers, we consider straight-and-level flight without accelerations, i.e., $\mathbf{v} = \begin{bmatrix} v & 0 & 0 \end{bmatrix}^\top$ and constant, and $\psi = 0$ and constant. For this symmetric flight condition, we can simplify (3.21) and (3.22) to obtain the pitch angle with regard to the zero-lift frame and the total thrust, as follows:

$$\bar{\theta} = \operatorname{atan2}\left(\eta c_{D_V} v^2 + mg, c_{L_V} v^2 - \eta mg\right) + k\pi, \tag{5.4}$$

$$T = \frac{1}{\mathrm{c}\bar{\alpha}\ (1 - c_{D_T})}\left(\mathrm{s}\bar{\theta}\ mg + c_{D_V}\ \mathrm{c}\bar{\theta}\ v^2\right), \tag{5.5}$$

where

$$\eta = \frac{\mathrm{s}\bar{\alpha}\ (c_{L_T} - 1)}{\mathrm{c}\bar{\alpha}\ (1 - c_{D_T})}. \tag{5.6}$$

The moment equilibrium reduces to

$$\mathbf{i}_y^\top \mathbf{m}_T + \mathbf{i}_y^\top \mathbf{m}_\delta = c_{\mu_T} T + l_{\delta_x} \mathbf{i}_z^\top \mathbf{f}_\delta^\alpha = 0, \tag{5.7}$$

so that we can obtain the flap deflections as

$$\frac{\delta}{2} = \frac{c_{\mu_T} T}{l_{\delta_x}\left(c_{L_T}^\delta\ \mathrm{c}\bar{\alpha}\ T + 2c_{L_V}^\delta\ \mathrm{c}\bar{\theta}\ v^2\right)}. \tag{5.8}$$

Figure 5-1a shows the vehicle pitches down as the speed increases. As expected, the pitch asymptotically approaches α_0.

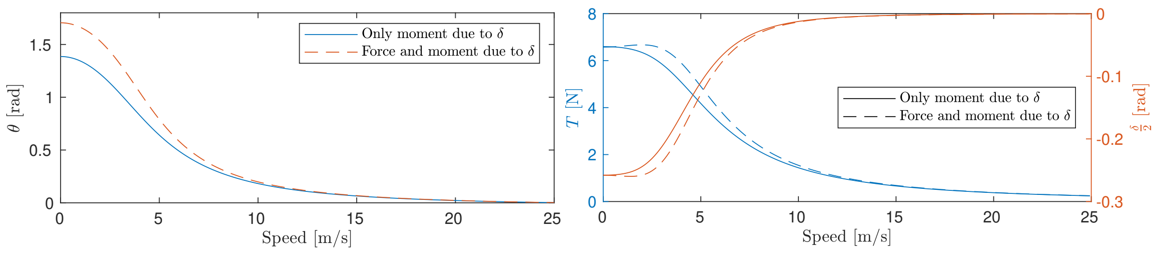
As described in Section 3.3, we utilize flatness of a simplified system where nonminimum phase dynamics due to the direct force contribution of the flaps are neglected. When combined with feedback control, this approach achieves good performance for slightly nonminimum phase systems [39]. The method is simple and avoids the large and quickly changing control actions that exact feedback linearization of the nonminimum phase system may result in [129]. We note that potentially a flat output of the unsimplified dynamics could be used to guarantee stable tracking [75]. However, this approach requires defining the trajectory in terms of the center of oscillation instead of the vehicle center of mass, leading to difficulty with the relatively complicated 6DOF tailsitter dynamics model.

The dashed lines in Fig. 5-1a do include the flap force contribution and were obtained by using

$$\eta' = \frac{\mathrm{s}\bar{\alpha}\ (c_{L_T} - 1) - \frac{c_{\mu_T}}{l_{\delta_x}}}{\mathrm{c}\bar{\alpha}\ (1 - c_{D_T})} \tag{5.9}$$

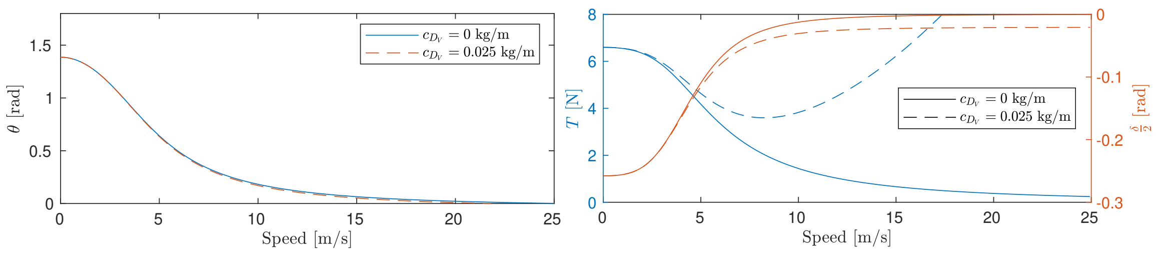
instead of (5.6). It can be seen that this leads to increased pitch angle. In fact, we can also observe this increase to $\theta > \pi/2$ rad in hover by comparing the hover poses obtained from the flat transform and from experiments at the start of the trajectories in the figures in Section 5.5. More important from the perspective of trajectory planning is the effect on the required control inputs. In Fig. 5-1b, we can see that the solid and dashed lines are relatively close, meaning that the flap force does not have a significant impact in these flight conditions.

The figure also shows that the required thrust decreases monotonously with increasing speed, reflecting the efficiency of cruise flight. An initial decrease is expected due to the reduction in lift-induced drag, but the thrust requirement should increase again at higher



(a) Pitch angle. (b) Control inputs.

Figure 5-1: Trim for straight-and-level flight without/with force due to flap deflection.



(a) Pitch angle. (b) Control inputs.

Figure 5-2: Trim for straight-and-level flight without/with parasitic drag.

speeds due to parasitic drag. This contribution is represented in the flight dynamics model by the parameter c_{D_V}, which we set to zero (see Section 3.5). In order to visualize the effect of parasitic drag, we also compute the pitch angle and control inputs using a realistic value of $c_{D_V} = 0.025$ kg/m. The corresponding dashed curve in Fig. 5-2 shows an increase in required thrust after the initial decrease, resembling a typical drag curve. Based on the required thrust, we now obtain a maximum flight speed of 28 m/s.

The flying wing vehicle is also capable of straight-and-level knife edge flight, i.e., at identical constant velocity but with $\psi = \pm{}^{\pi}/_{2}$ rad. For this condition, the flatness transform gives attitude and control inputs equal to hover, as the lateral aerodynamic force on the tailless body without vertical surfaces is assumed negligible. This is a reasonably accurate approximation of the physical behavior of the aircraft, as evidenced by the small ϕ angle during the knife edge sections of the trajectory shown in Fig. 3-7. Moreover, in Section 5.4 we show that the vehicle is able to approach the theoretical maximum speed for knife edge flight on a circular trajectory, which affirms that the lateral aerodynamic force is indeed quite small, even at increased speeds.

5.3.2 Forward and Sideways Transitions

Figure 5-3 shows the vehicle attitude during transitions varying from forward flight to sideways flight at a constant acceleration of 3 m/s^2, which is an aggressive transition maneuver that can still be accurately tracked by our control algorithm, as shown in Section 3.6. In the first column, which shows forward transition, we see decreasing pitch angle with increasing speed. Compared to Fig. 5-1, the pitch angle is further reduced to attain the forward acceleration required for transition. The bottom row of the figure corresponds to transition to knife edge flight. As described in Section 5.3.1, the model does not include aerodynamic forces in the $\mathbf{b}_x$ and $\mathbf{b}_y$ directions. Hence the constant acceleration results in virtually constant attitude throughout the transition. The remaining vehicle poses in the figure show that differential flatness enables the generation of transition maneuvers in arbitrary directions. The corresponding state trajectories can be obtained from the trajectory reference in real time through the analytical flatness transform, obviating the need for pre-planned transition maneuvers.

5.3.3 Lateral Acceleration in Hover and Forward Flight

A major challenge in control design for transitioning vehicles is the difference between flight phases when it comes to the mechanism for applying lateral acceleration. In hover and knife edge flight, horizontal accelerations are mainly obtained by thrust vectoring, i.e., by changing the vehicle attitude to direct the rotor thrust in the appropriate horizontal direction. Lateral acceleration is thus obtained by rotation around approximately $\mathbf{b}_z$. In forward flight, lateral force is generated by tilting of the aerodynamic lift vector through a rolling motion, chiefly around $\mathbf{b}_x$. This mechanism is referred to as *banking.* Existing controllers may employ various strategies for the transition between rotation axes, e.g., gain scheduling [50] or pre-computed trim points [68].

The ZXY rotation sequence employed by the flatness transform (described in Section 3.3.1) results in a continuous transition trajectory that can be tracked without relying on any gains or pre-computed trim points. In this rotation sequence, the angle ϕ determines the orientation of the pitch axis. Simplifying its expression 3.17 for $\mathbf{v} = \begin{bmatrix} v & 0 & 0 \end{bmatrix}^\top$ and $\mathbf{a} = \begin{bmatrix} 0 & a & 0 \end{bmatrix}^\top$ shows how this angle only depends on the ratio between the lateral and vertical forces

$$\phi = \operatorname{atan2}(a, g). \tag{5.10}$$

Hence, the resulting pitch plane of rotation is independent of the speed v, and it is oriented such that in hover the pitch rotation $\theta \approx \pi/2$ results in sideways tilting of the thrust vector, and in forward flight at higher speeds the small pitch angle θ results in banking. Both conditions, as well as the intermediate, are depicted in Fig. 5-4. Note that the pitch plane of rotation is identical for all three conditions.

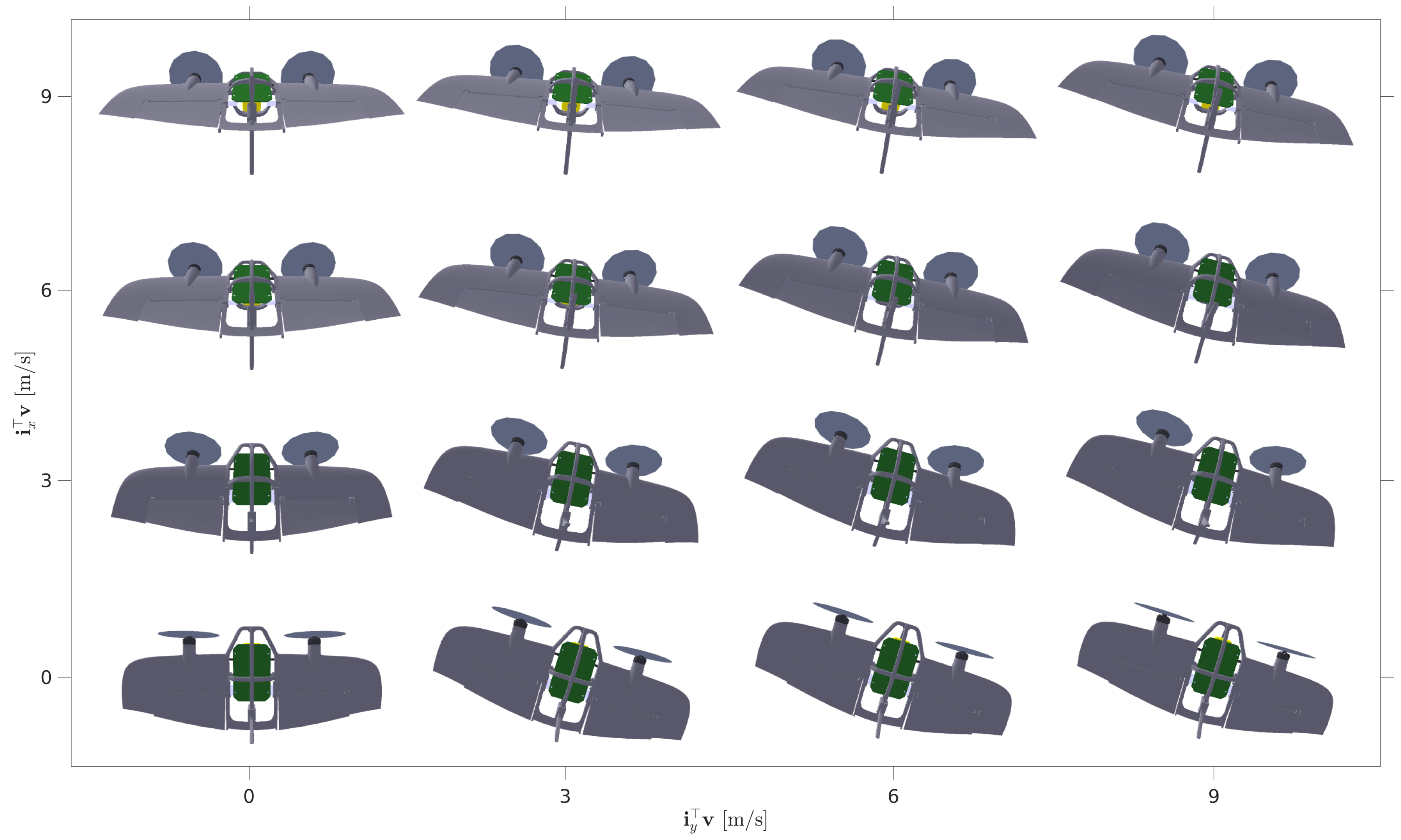
5.4 Dynamic Feasibility

The differential flatness transform allows computation of the vehicle state and control inputs based on the trajectory reference and its derivatives. Feasibility of the trajectory can then be determined based on the obtained control inputs. In this section, we evaluate the suitability of the differentially flat dynamics model presented in Section 3.3 to determine feasibility of a candidate trajectory.

We consider two types of representative trajectories. A fast transition between static hover at positions that are 6 m apart. This trajectory requires fast acceleration and deceleration of the vehicle and, at the same time, includes a yawing motion through the transition regime. The second trajectory combines high-speed flight with high accelerations on a circular trajectory with a yaw reference for coordinated, knife-edge, or rolling flight.

5.4.1 Hover-to-Hover Trajectory

The propulsion and aerodynamic forces acting on the tailsitter are highly dependent on the yaw angle, so that the maximum feasible acceleration may be highly dependent on



$\mathbf{i}_x^\top \mathbf{v}$ [m/s]
9
6
3
0
0
3
6
9
$\mathbf{i}_y^\top \mathbf{v}$ [m/s]

Figure 5-3: Attitude and flap deflections for various horizontal velocities and accelerations. All figures correspond to $\|\mathbf{a}\| = 3$ m/s^2 with acceleration and velocity in the same direction.

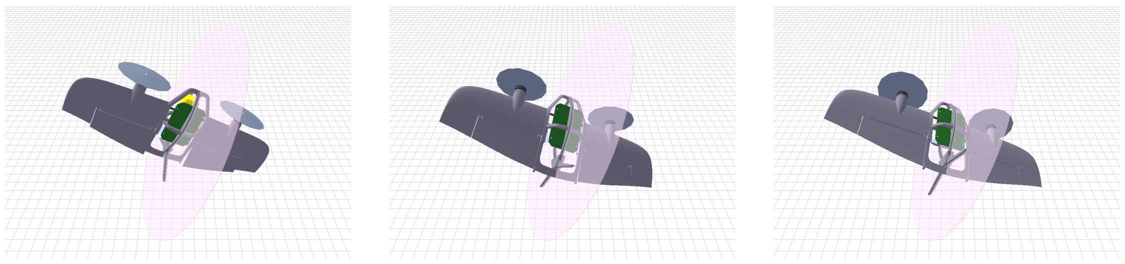

(a) $v = 0$ m/s. (b) $v = 7.5$ m/s. (c) $v = 15$ m/s.

Figure 5-4: Hover and forward flight with $g/2$ lateral acceleration. The pitch plane of rotation is indicated by the pink disk.

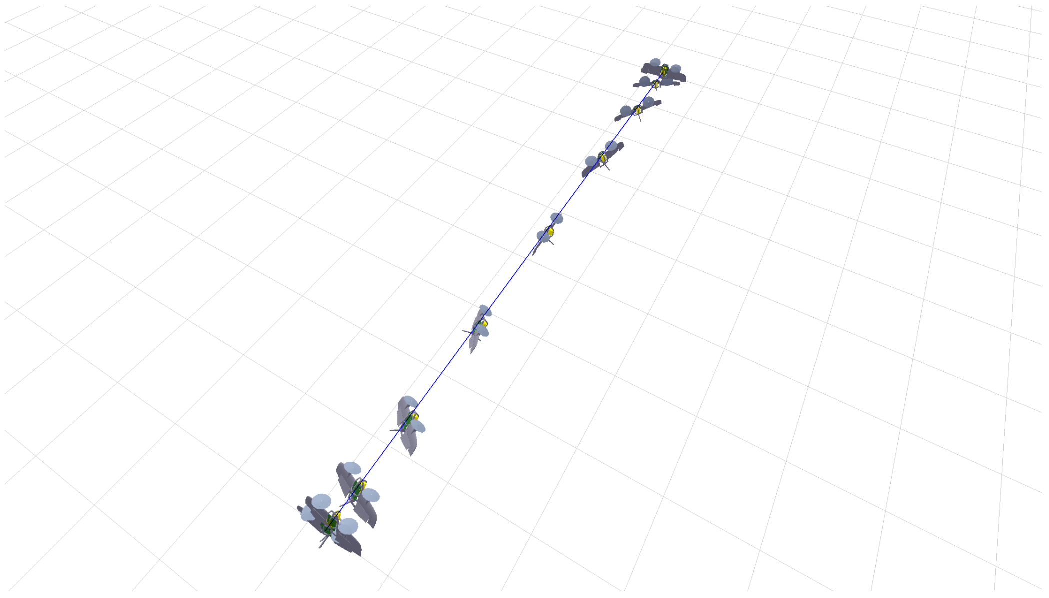
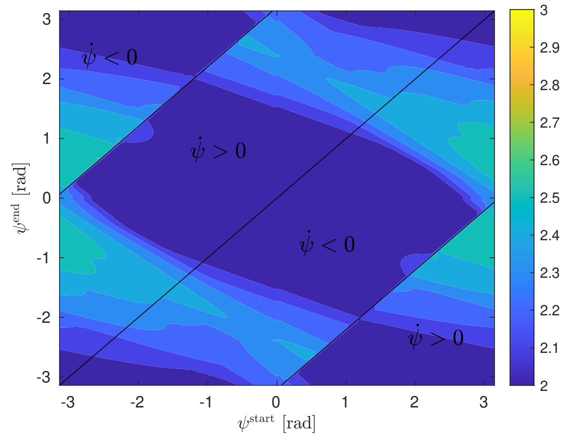
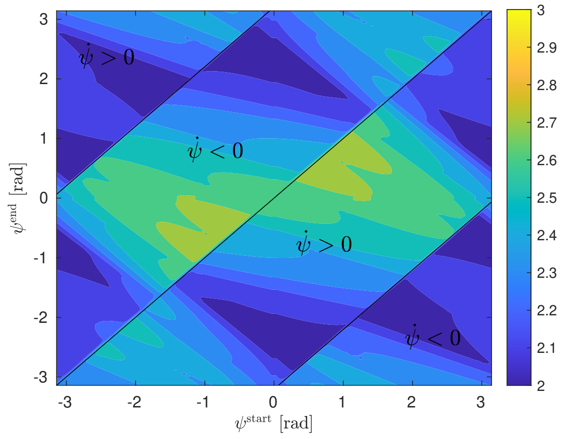


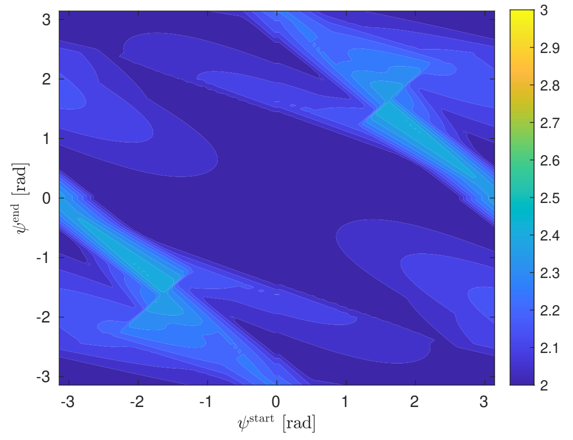
Figure 5-5: 6 m hover-to-hover trajectory with $\psi^{\text{start}} = 0$ rad, $\psi^{\text{end}} = \pi$ rad. Trajectory time is 3 s, interval between poses is 0.25 s.



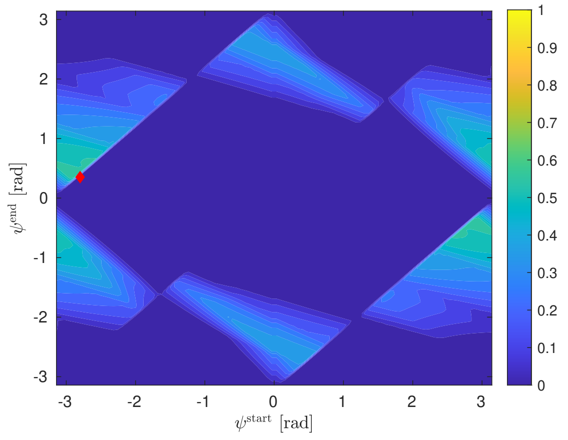
(a) Using minimal yaw rotation.



(b) Using opposite yaw rotation.

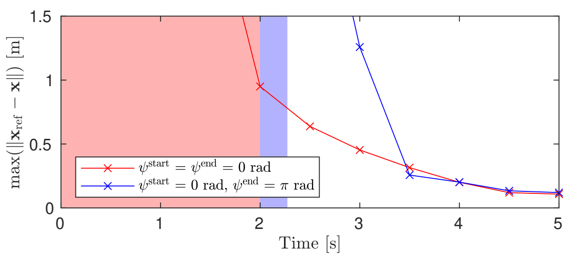


(c) Minimum (a) and (b).

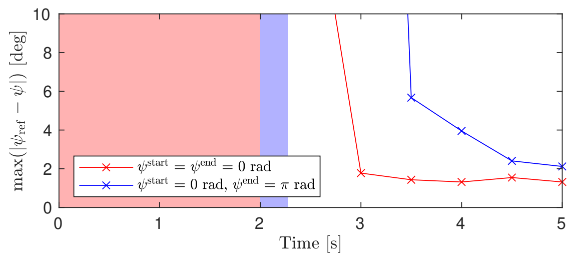


(d) Difference (a) and (c).

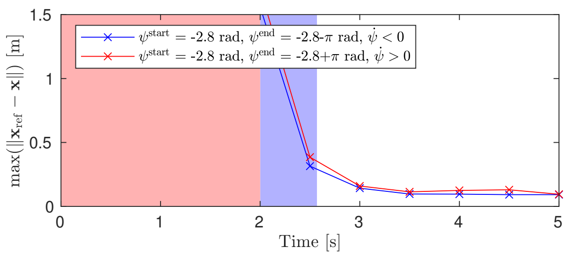
Figure 5-6: Minimum feasible time for 6 m hover-to-hover trajectory.



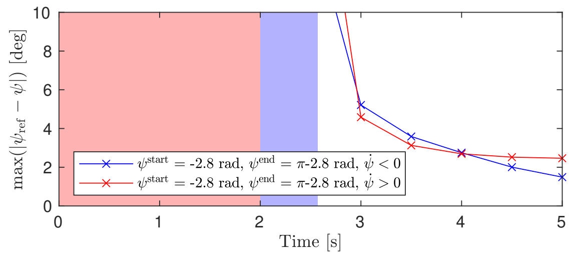
(a) Maximum position tracking error with and without yaw rotation.



(b) Maximum yaw tracking error with and without yaw rotation.



(c) Maximum position tracking error for opposite yaw rotations.



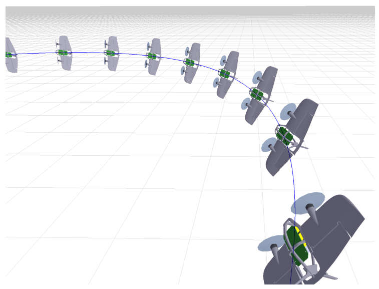
(d) Maximum yaw tracking error for opposite yaw rotations.

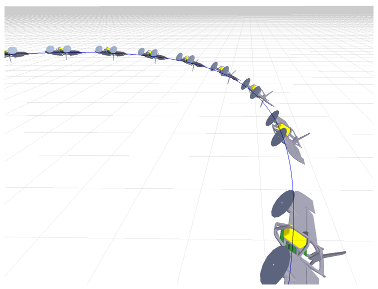
Figure 5-7: Tracking error in flight experiments 6 m hover-to-hover trajectory. Shaded area indicates infeasibility according to differential flatness transform.

the yaw motion. To explore how this dependency is reflected in the dynamic feasibility of trajectories, we compute the minimal feasible trajectory time for the 6 m hover-to-hover trajectory with varying start and end yaw. An example trajectory is shown in Fig. 5-5.

Figure 5-6a shows results for the trajectory with yawing motion from ψ^{start} to ψ^{end} using the minimal rotation. It can be seen that the fastest times are achieved around $\psi^{\mathrm{start}} = \psi^{\mathrm{end}} = 0$ rad, which corresponds to forward coordinated flight. We observe discontinuity along the yaw direction switching lines in Fig. 5-6a, which indicates that, in some cases, it may be beneficial to yaw in the opposite direction. Results corresponding to the opposite yaw direction are shown in Fig. 5-6b, and the minimum time between the two yaw directions is shown in Fig. 5-6c. Slight discontinuities persist along the line $\psi^{\mathrm{start}} = \psi^{\mathrm{end}}$, indicating that, in some cases, it may be beneficial to yaw slightly more than a full rotation. In practice, this advantage does not materialize because of the challenges posed by flight with large angular velocities that are not captured by the dynamics model so that trajectories with superfluous yaw rotations can be disregarded. The benefit of using the non-minimal rotation over the minimal yaw rotation (cf. Fig. 5-6b and Fig. 5-6a) is shown in Fig. 5-6d. For most yaw references the difference is small, meaning that the minimal rotation that is obtained from optimization in the flat output space is (nearly) optimal. As expected, we observe that the largest benefit occurs around $|\psi^{\mathrm{start}} - \psi^{\mathrm{end}}| = \pi$ rad, where the difference between the magnitude of the two yaw rotations is smallest.

We conduct experiments for four different yaw references to compare the feasibility boundary from Fig. 5-6 to the tracking error of the actual vehicle. Figure 5-7a and Fig. 5-7b show the tracking error for the hover-to-hover trajectory in coordinated flight without yaw and for the same trajectory but with a π rad yaw rotation. Each point on the curves corresponds to a flight experiment. As the trajectory time on the horizontal axis increases, the maneuvers become less aggressive, and the tracking error decreases. The corresponding


(a) Coordinated.


(b) Knife edge.

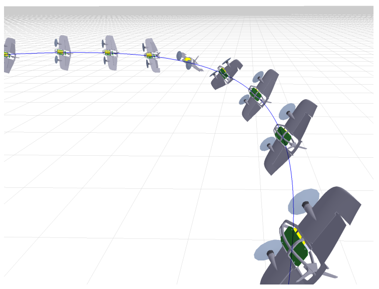
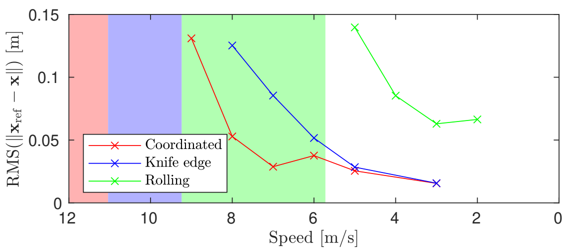
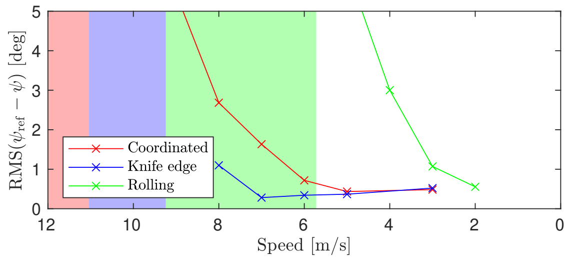

(c) Rolling.

Figure 5-8: Circular trajectory with various yaw references.



(a) RMS position tracking error.



(b) RMS yaw tracking error.

Figure 5-9: Tracking error in flight experiments for circular trajectory with various yaw references. Shaded area indicates infeasibility according to differential flatness transform.

feasibility boundaries predicted in Fig. 5-6 are indicated by the colored shading, i.e., the shaded areas in the left of the figure correspond to infeasible trajectory times. While only a single color is shown at a time, the infeasibility areas continue from their boundary all the way to the vertical axis on the left. The tracking error increases at lower speeds for the yawing trajectory than for the coordinated flight trajectory, as predicted by the predicted feasibility boundaries. We note that these boundaries correspond to the most aggressive trajectories that theoretically can be tracked by the given vehicle dynamics model, neglecting practical factors such as modeling errors and imperfect state estimation and control, so that it is expected that significant tracking error occurs before they are reached. The coordinated flight trajectory at the feasibility boundary (2.0 s) attains a maximum speed of 7.6 m/s within 1 s and attains a maximum acceleration of 3.1 g. It is tracked with less than 1 m position tracking error.

To evaluate the effect of opposing yaw rotation direction, we performed flight experiments for both rotation directions corresponding to the maximum of Fig. 5-6d (indicated by the red diamond). Figure 5-7a and Fig. 5-7b show that—even for this yaw reference—the effect of the rotation direction on the tracking error is negligible. We conjecture that, in practice, it may be best to always use the minimum yaw rotation, which conveniently corresponds to the minimization of the yaw derivatives of the flat output trajectory.

5.4.2 Circular Trajectory

In order to evaluate the tailsitter capability to maintain a large acceleration in high-speed flight, we use the differential flatness transform to determine the maximum speed on a circular trajectory with a 3 m radius. As shown in Fig. 5-8, we consider two trimmed

Velocity			Acceleration			Jerk			Snap		
$\mathbf{i}_x$	$\mathbf{i}_y$	$\mathbf{i}_z$	$\mathbf{i}_x$	$\mathbf{i}_y$	$\mathbf{i}_z$	$\mathbf{i}_x$	$\mathbf{i}_y$	$\mathbf{i}_z$	$\mathbf{i}_x$	$\mathbf{i}_y$	$\mathbf{i}_z$
v	0	0	0	$-\Omega v$	0	$-\Omega^2 v$	0	0	0	$\Omega^3 v$	0

(a) Position derivatives.

Coordinated			Knife edge			Rolling		
ψ	$\dot{\psi}$	$\ddot{\psi}$	ψ	$\dot{\psi}$	$\ddot{\psi}$	ψ	$\dot{\psi}$	$\ddot{\psi}$
0	$-\Omega$	0	$\pi/2$	$-\Omega$	0	$[0, 2\pi]$	Ω	0

(b) Yaw (derivatives).

Table 5.1: Flat output (derivatives) for various circular trajectories with $\Omega = v/r$.

conditions, coordinated and knife edge flight, as well as a rolling/yawing motion where ψ_{ref} changes at the same rate but in the opposite direction as the circular motion. The position and yaw derivatives for analytical evaluation of the feasibility are given in Table 5.1.

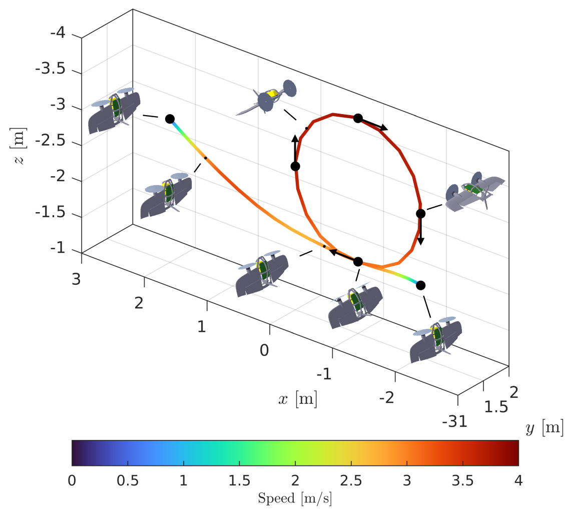
We perform experiments for all three circular trajectories at various speeds. The results are shown in Fig. 5-9, where each point on the curves corresponds to a flight experiment. The figure is oriented similarly to Fig. 5-7 with the most aggressive, i.e., the highest speed, trajectories towards the left. The figure shows that the flat dynamics model predicts that coordinated flight can be performed up to the highest speed, followed by knife edge flight, and finally the rolling circle, which has a relatively low maximum speed. The position tracking errors obtained from flight experiments agree with this prediction. Figure 5-9a shows the expected increase in each position tracking error before the corresponding shaded area is reached. In Fig. 5-9b, we see that the yaw tracking error in knife edge flight remains very small, even at high speeds. The controller achieves this small yaw error because the thrust vector is pointing inward the circle, almost horizontally, so that differential thrust can be used to very effectively control yaw.

Since the aerodynamics model does not consider lateral forces on the tailless aircraft, the speed in circular knife edge flight is mostly limited by the maximum thrust. In fact, completely neglecting the aerodynamics and solving for the maximum speed as

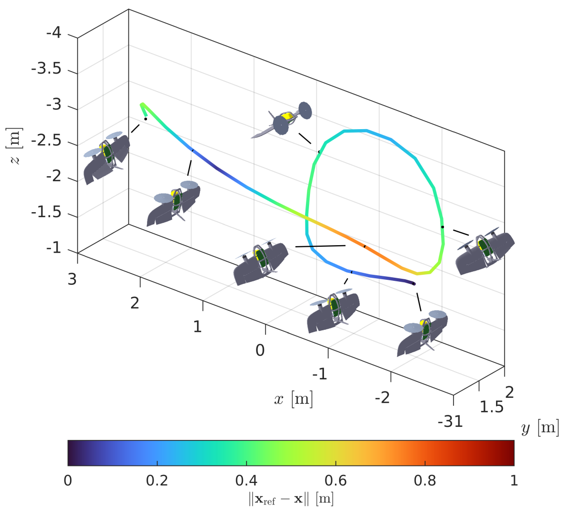
$$v_{\max} = \sqrt{2 c_T \bar{\omega}^2 \frac{r}{m}} \tag{5.11}$$

with $\bar{\omega}$ the maximum motor speed, results in only a small overestimation when compared to the maximum speed obtained from the flat transform (9.5 m/s versus 9.2 m/s). In flight experiments, the vehicle achieves RMS position and yaw tracking errors of respectively 12.5 cm and 1.1 deg at 8 m/s. The fact that it approaches the theoretical maximum speed with relatively small tracking error, affirms that the lateral aerodynamic force must indeed be quite small. We also note that at least some control input margin must be maintained to enable stabilization of the unstable knife edge condition, so that attaining the full theoretical maximum speed is certainly impossible in practice.

Considering the comparitive results for both trajectories, we can conclude that the differential flatness transform gives a useful qualitative prediction of the critical trajectory time or speed where we can expect to observe a stark increase in tracking error on the real vehicle.



(a) Reference with waypoints. Start and end points are static hover, and arrows indicate velocity direction constraints.



(b) Experiment.

Figure 5-10: Loop. Interval between poses is 0.7 s.

5.5 Aerobatic Maneuvers

In this section, we demonstrate how the flatness transform can generate aerobatic maneuvers using relatively simple constraints, i.e., position and yaw waypoints, and—where needed—derivative constraints at these waypoints. In particular, we show that planning in the flat output space generates aerobatic maneuvers that exploit the full flight envelope of the tailsitter aircraft, including post-stall and uncoordinated flight conditions. In contrast, existing methods for fixed-wing planning in the output space rely on restrictive assumptions such as coordinated flight and curvature limitations. State-space approaches that do consider a more sophisticated dynamics model are often computationally expensive, preventing their use in real-time applications.

In this section, we show that the generated reference trajectories for complex aerobatic maneuvers are indeed realistic and can be tracked by the actual vehicle in flight experiments. We are mainly interested in generating fast and agile trajectories and fly each trajectory in proximity of the feasibility boundary, often approaching maximum motor speeds and flap deflections. As observed in Section 5.4, this may lead to increased tracking errors. While the tracking error in each aerobatic maneuver can be reduced by slowing down, we chose to accept some tracking error in favor of increased aggressiveness. For experimental analysis focusing on the tracking performance of the flight control algorithm, the reader is referred to Section 3.6.

5.5.1 Loop

The loop trajectory shown in Fig. 5-10 consists of five waypoints (of which two coincide) on a vertical circle with 1 m radius, and start and end points constrained to static hover. We add tangential velocity constraints to enforce a circular path. As shown in Fig. 5-11, the

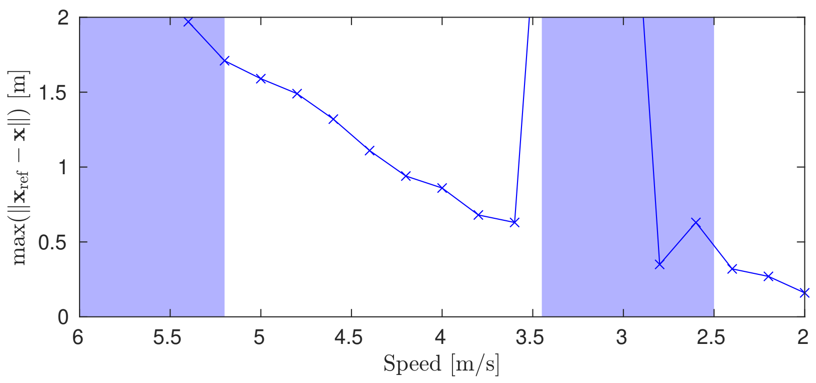
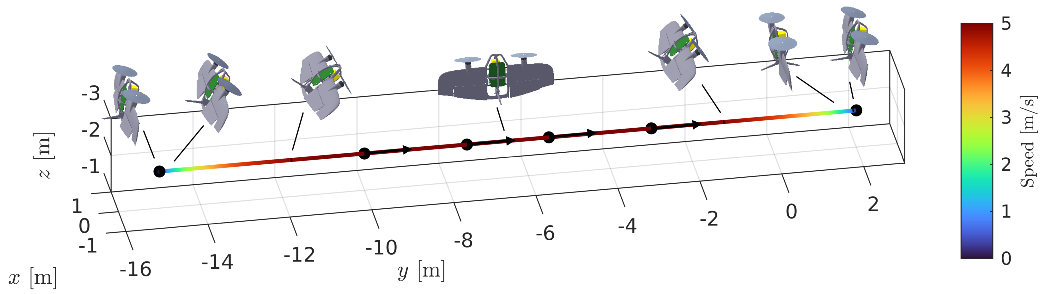


Figure 5-11: Maximum position tracking error in flight experiments for loop trajectory at various speeds. Shaded area indicates infeasibility according to differential flatness transform.

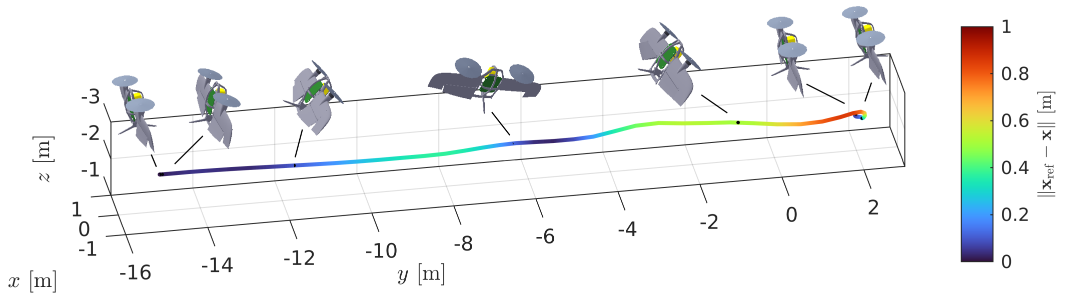
loop trajectory has several feasibility boundaries. When flown slowly (i.e., below 2.5 m/s), the trajectory is feasible and flown in hover attitude with $\theta \approx {}^{\pi}/{}_{2}$ rad. When flown faster (i.e., around 4.5 m/s), the vehicle performs a loop, making a full upward pitch rotation. Intermediate speeds (i.e., around 3 m/s) are too slow to perform a loop and require the vehicle to quickly pitch back down at the top of the circular segment, rendering the trajectory infeasible due to flap deflection limits. The maximum position tracking error obtained from flight experiments shows a stark increase in this region of infeasibility and also increases as the infeasibility boundary at very high speed (i.e., 5.2 m/s) is approached. The trajectory with a maximum speed of 3.8 m/s is shown in Fig. 5-10. The loop maneuver is successfully performed in the flight experiment. The maximum position error of 71 cm is incurred when exiting the final circular segment.

5.5.2 Knife Edge Flight

Figure 5-12 shows a straight trajectory between static hover start and end points. The intermediate waypoints enforce a constant speed of 5 m/s and serve to transition between flight attitudes through the yaw reference ψ_{ref}. In the first of the three middle segments, the vehicle transitions from coordinated to knife edge fight; in the second, it maintains constant knife edge orientation; and in the third, it transitions back to coordinated flight. Performing the transitions while maintaining straight flight at 5 m/s is challenging due to the aerodynamic interactions between vehicle attitude, flap deflections, and rotor speeds. As expected, the position tracking error in the flight experiment increases at the transitions. Once knife edge orientation is reached, the position tracking error quickly reduces again. The vehicle attitude during knife edge flight differs somewhat between the reference and experiment trajectories. The increased pitch angle in the experiment compensates for the neglected flap force contribution, and the small rotation towards the direction of travel compensates for the nonzero lateral force. Finally, we note that the largest position tracking error is incurred close to the end point. This error is mainly along the trajectory, and is caused by delayed deceleration. The maximum path error, i.e., position error with regard to the closest point on the trajectory line, occurs during the second transition and amounts to 0.47 m.



(a) Reference with waypoints. Start and end points are static hover, and arrows indicate 5 m/s velocity constraints.



(b) Experiment.

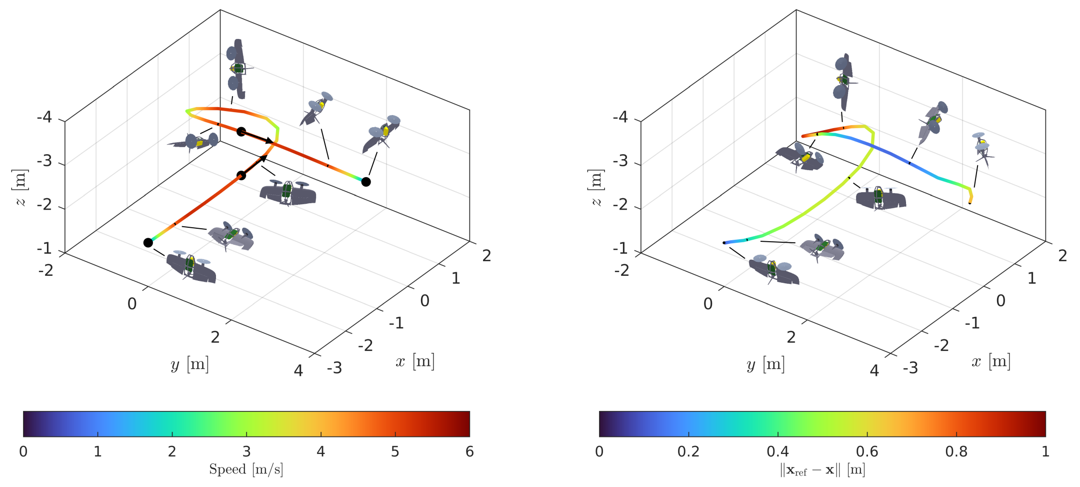
Figure 5-12: Coordinated-Knife Edge-Coordinated Flight. Interval between poses is 0.6 s.

5.5.3 Climbing Turn

We plan a climbing turn trajectory using four waypoints, as shown in Fig. 5-13. The start and end points are constrained to static hover, and the two intermediate waypoints are positioned with only a height difference. Using velocity constraints, we enforce straight and coordinated flight at the intermediate waypoints. Hence, the entire 270 deg turn and 1 m climb occur between these two waypoints. During the turn, the reference trajectory reaches about 90 banking angle, requires nearly the maximum motor speeds of 2500 rad/s, and reaches a peak angular velocity of 11.3 rad/s (650 deg/s). A peak acceleration of 3 g is required during the turn, while the acceleration and deceleration close to respectively the start and end points reach up to 2 g. Consequently, the vehicle quickly completes the 11.3 m trajectory in 3.1 s, despite starting and ending in static hover. In the flight experiment, we observe that, during the turn, indeed a maximum acceleration of 3.4 g and a peak angular velocity of 10.9 rad/s (625 deg/s) are attained. The motors briefly saturate, resulting in some loss of altitude. Once the saturation is resolved, the vehicle quickly catches up and reduces the position tracking error to below 20 cm before the turn is exited.

5.5.4 Immelmann Turn

The Immelmann turn is a well-known aerobatics and aerial combat maneuver that turns the aircraft by performing a half loop followed by a half roll. We generate the trajectory using static hover start and end points, and four intermediate waypoints to enforce constant speed coordinated flight prior to the half loop and constant speed transition from inverted to regular coordinated flight afterward. Similar to the loop and knife edge maneuvers described above, we observe increased error when exiting the loop segment, increased error during transition through uncoordinated flight orientation, and delayed deceleration towards



(a) Reference with waypoints. Start and end points are static hover, and arrows indicate 5 m/s velocity constraints.

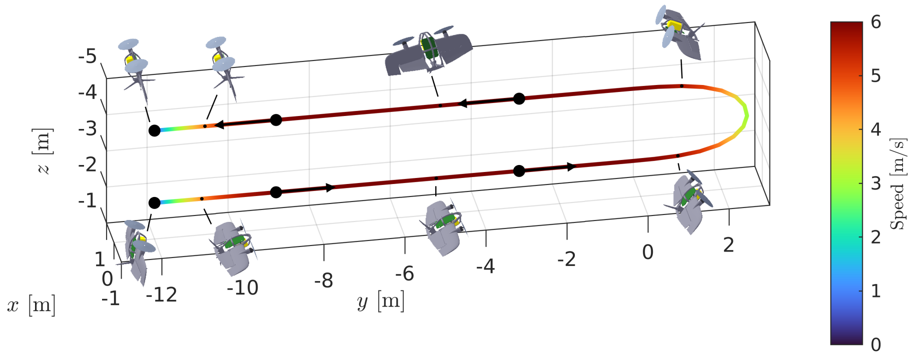
(b) Experiment.

Figure 5-13: Climbing turn with 1 m height difference. Interval between poses is 0.5 s.

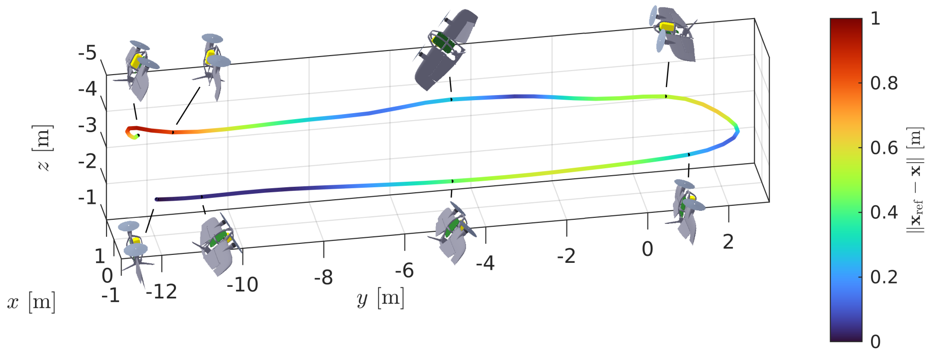
the end point. Comparison of the vehicle poses also leads to similar observations of small differences: increased pitch to account for flap force and increased yaw in uncoordinated flight to compensate for the nonzero lateral force. The Immelmann turn combines several challenging aspects to exploit the expansive flight envelope of the tailsitter vehicle to a large degree. The maneuver contains large accelerations, inverted flight, and a transition through the entire yaw range, i.e., from $\psi_{\text{ref}} = 0$ to $\psi_{\text{ref}} = \pm\pi$ rad, at a peak angular rate of 10 rad/s (573 deg/s) while maintaining a linear speed of 6 m/s. Based on snap minimization and differential flatness, the state-space trajectory and corresponding control inputs were generated efficiently and based on only four waypoints. The flight experiment shows that the resulting maneuver can be tracked with acceptable position error (< 0.6 m during the maneuver itself) while approaching the feasibility boundary, as over 90% of the maximum flap deflection is reached during the half roll.

5.5.5 Split S

The Split S maneuver, shown in Fig. 5-15, is similar to the Immelmann but performed in opposite order. The maneuver starts the top leg in coordinated flight, then transitions to inverted coordinated flight using the yaw reference ψ_{ref}, and ends with a half loop which is exited at the bottom in regular coordinated flight condition. The trajectory is generated using similar waypoints as the Immelmann maneuver, albeit with opposite order and velocity direction. The flight experiment is performed at a slightly lower speed than the Immelmann turn (5 m/s versus 6 m/s), reducing the tracking error. The final part of the Split S trajectory consists of a relatively long stretch of coordinated flight so that the vehicle starts its deceleration from a more stable condition and incurs less position tracking error as it approaches the end point. We observe a downward pitch motion during the half loop in both the reference and experiment trajectories. By increasing the speed, we can obtain

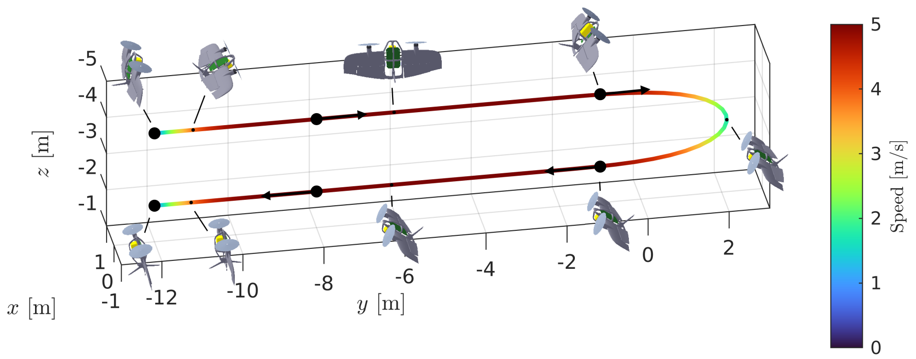


(a) Reference with waypoints. Start and end points are static hover, and arrows indicate 6 m/s velocity constraints.

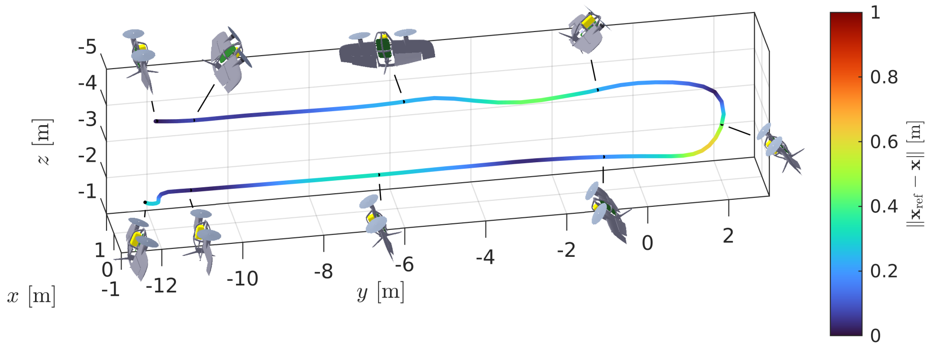


(b) Experiment.

Figure 5-14: Immelmann turn. Interval between poses is 1.0 s.



(a) Reference with waypoints. Start and end points are static hover, and arrows indicate 5 m/s velocity constraints.



(b) Experiment.

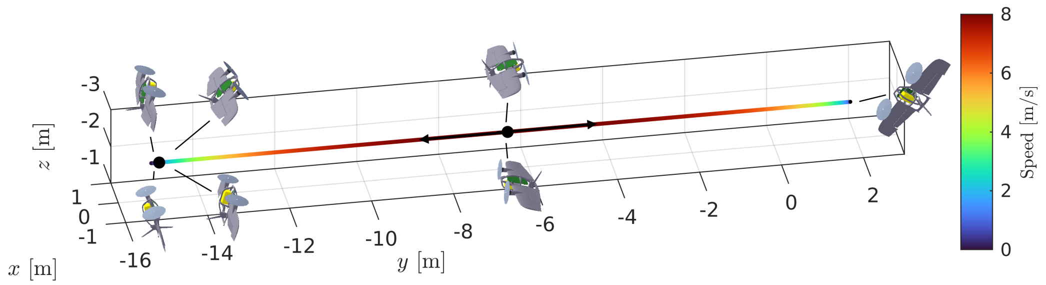
Figure 5-15: Split S maneuver. Interval between poses is 1.0 s.

a more traditional Split S maneuver with a positive pitch rate. However, this maneuver requires a significantly larger flight volume.

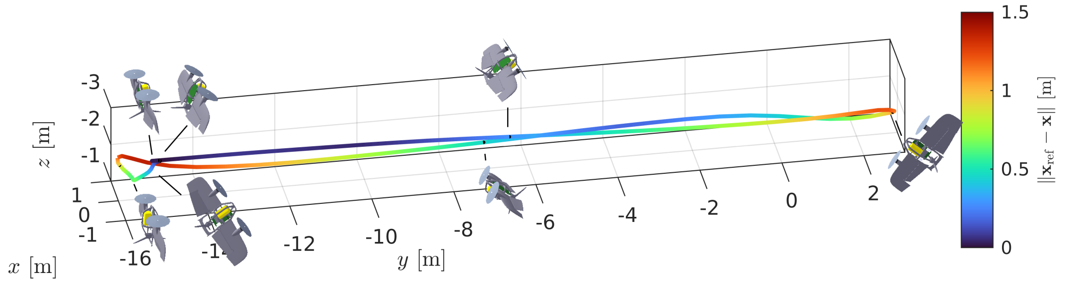
5.5.6 Differential Thrust Turn

The differential thrust turn is an agile flight maneuver in which the vehicle transitions to coordinated flight in the opposite direction without deviating from a straight-line trajectory. Unlike more traditional turns, which involve turning on a circular trajectory segment, the differential thrust turn is performed by rotating the vehicle itself using differential thrust and flap deflections, and then applying a large collective thrust to accelerate in the opposite direction. The behavior and performance of the trajectory-tracking flight control algorithm during a differential thrust turn is analyzed in Section 3.6.6.

Figure 5-16 shows a differential thrust turn trajectory. The turn itself follows directly from snap minimization based on two coinciding waypoints with opposite velocity and yaw constraints. In the flight experiment, a peak angular rate of 8.6 rad/s (493 deg/s) is reached during the turn. As shown in the figure, the differential flatness transform is able to accurately predict the vehicle attitude at the midpoint of the turn.



(a) Reference with waypoints. Start and end points are static hover, and arrows indicate 8 m/s velocity constraints.



(b) Experiment.

Figure 5-16: Differential thrust turn. Interval between poses is 1.5 s.

5.6 Multi-Fidelity Trajectory Optimization

In this section, we apply the algorithm for multi-fidelity Bayesian optimization that was proposed in Chapter 4 to a tailsitter racing trajectory. The algorithm adjusts the time allocation $\mathbf{t}$ over the trajectory segments based on data obtained from flight experiments. In order to efficiently leverage flight experiments, we also incorporate feasibility evaluations based on control inputs obtained from the flatness transform, i.e., using (5.3). These evaluations serve as a low-fidelity data source and help select the most valuable candidate trajectories to evaluate in flight experiments. In the high-fidelity experimental evaluations, we classify a candidate trajectory as feasible if the maximum Euclidean position tracking error is within 0.5 m, i.e.,

$$\Sigma_T = \left\{ \boldsymbol{\sigma}_{\mathrm{ref}} \middle| \|\mathbf{x}_{\mathrm{ref}}(t) - \mathbf{x}(t)\| \leq 0.5 \text{ m} \quad \forall t \in [0, T] \right\}. \tag{5.12}$$

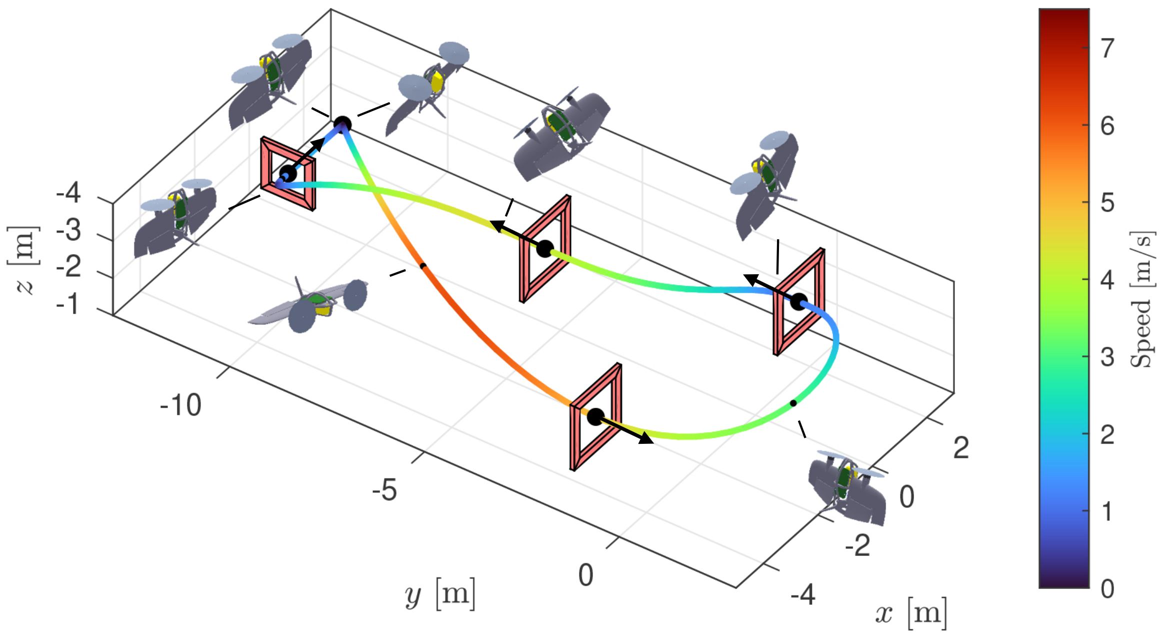
We also constrain the trajectory to remain within the finite dimensions of the flight space.

The trajectory, shown in Fig. 5-17, consists of six waypoints. The start and end points coincide and are constrained to static hover. The four intermediate waypoints are placed at the center of four gates. Each of these waypoints is equipped with a directional velocity constraint that enforces flight perpendicular to the gate window. Yaw is constrained so that the first three gates are passed in coordinated flight. The final gate is smaller and must be passed in knife edge flight.

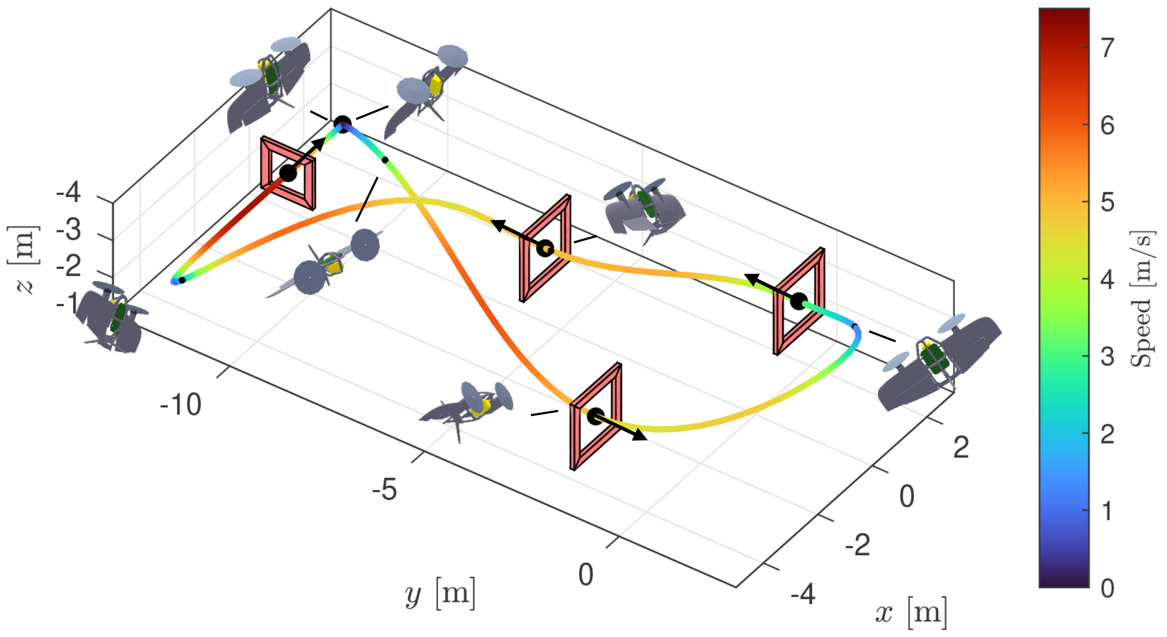
Figure 5-17a shows the initial trajectory obtained from standard minimum-snap trajectory optimization. The trajectory is on the feasibility boundary of the high-fidelity flight experiment. Its largest position tracking error occurs right after the first gate when the vehicle quickly decelerates. Note that we also observed relatively large tracking errors during deceleration for the trajectories in Section 5.5. The optimized trajectory, shown in Fig. 5-17b, shows that the Bayesian optimization algorithm has addressed the critical point at the first gate by reducing speed on the first trajectory segment. This enables an increase in speed on the remaining segments so that less deceleration is needed and the tracking error is reduced. Especially the final two segments are significantly faster. During these final segments, the vehicle attitude is such that the large acceleration changes in $\mathbf{i}_x$ direction are attained by rotation around the $\mathbf{b}_z$ axis. Attitude control around this axis relies mainly on differential thrust of the powerful, high-bandwidth motors, so that high-snap maneuvers can be tracked accurately. Figure 5-18 shows that the optimized trajectory has over ten times more snap, confirming that the generated trajectory exploits the vehicle dynamics to enable accurate tracking of more aggressive maneuvers. The figure also shows that the optimized trajectory requires 19% less flight time, reducing the flight time from 13.7 s to 11.1 s despite traveling a significantly greater distance of 40.6 m versus 34.2 m.

5.7 Summary

In this chapter, we proposed a trajectory generation algorithm for the global 6DOF tailsitter flying wing dynamics based on minimization of snap and yaw acceleration in the flat output space. We presented theoretical and experimental analyses of the differential flatness transform, showing how it provides natural solutions for challenging maneuvers through the transition regime and gives a qualitative prediction of the critical trajectory time or speed where a stark increase in tracking error can be expected on the real vehicle. Using our proposed algorithm, we presented reference trajectories and experimental results for six fast and agile aerobatic maneuvers. Finally, we showed that the framework for multi-fidelity Bayesian trajectory optimization proposed in Chapter 4 can also be leveraged effectively to incorporate experimental evaluations towards trajectory optimization for the tailsitter vehicle.



(a) Initial trajectory obtained from minimum-snap trajectory generation. Interval between poses is 2.3 s.



(b) Final trajectory obtained from multi-fidelity Bayesian optimization. Interval between poses is 1.9 s.

Figure 5-17: Multi-fidelity Bayesian optimization of trajectory through gates. Start and end points are static hover, and arrows indicate velocity direction constraints.

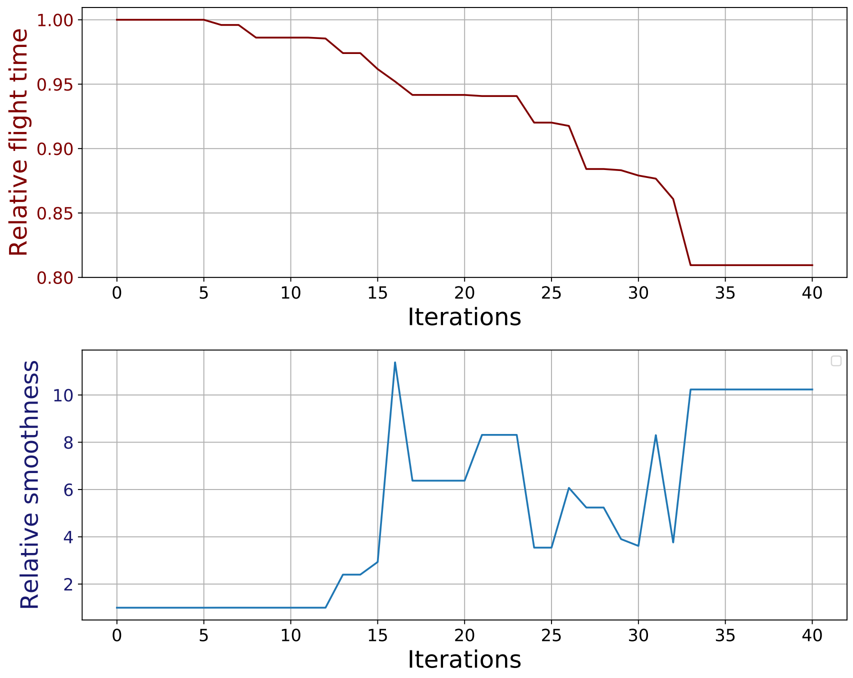


Figure 5-18: Relative trajectory time and smoothness for trajectory through gates.

Chapter 6

Conclusions

6.1 Summary of Contributions

In this section, we summarize our contributions by revisiting the research questions posed in Section 1.2.

1. How can we find time-optimal trajectories on the boundary of the feasibility set?

In Chapter 4, we presented an algorithm for trajectory generation at the true dynamic feasibility boundary, i.e., fully exploiting the capabilities of the actual vehicle. The algorithm, based on multi-fidelity Bayesian optimization, combines analytical, numerical, and experimental evaluations to efficiently optimize the trajectory using a limited number flight experiments. As far as we are aware, this is the first trajectory generation algorithm that explicitly uses experimental evaluations to push the boundary of fast and agile flight.

2. How can we address the complex dynamics of fast and agile flight without relying on extensive flight dynamics modeling?

In Chapter 2 and Chapter 3, we showed that robust control based on incremental nonlinear dynamic inversion enables accurate tracking of fast and agile trajectories without relying on extensive flight dynamics modeling. Our proposed control designs for quadcopter and tailsitter flying wing aircraft rely only on local accuracy of the dynamics model, making them robust against modeling errors and external disturbances. Despite not incorporating any vehicle aerodynamics model, the proposed incremental quadcopter controller outperforms nonincremental methods that include vehicle-specific aerodynamics parameters.

3. How can we generate and track fast and agile maneuvers that exploit the entire flight envelope of a VTOL fixed-wing aircraft?

In Chapter 3, we showed differential flatness of the flying wing tailsitter global nonlinear 6DOF flight dynamics with aerodynamics equations. We utilized the corresponding flatness transform to derive a feedforward angular velocity input that enables our proposed control algorithm to attain accurate tracking of agile maneuvers. In Chapter 5, we showed that the flatness transform enables trajectory generation and tracking of aerobatics maneuvers. As far as we are aware, this is the first time differential flatness of a global transitioning aircraft dynamics model has been shown. Our trajectory generation and tracking algorithms for the tailsitter flying wing are also unique in the sense that they are capable of exploiting the entire flight envelope, including post-stall, uncoordinated, and other challenging flight conditions.

6.2 Recommendations for Future Work

This section provides various avenues for future work.

6.2.1 Autonomous Flight

Our experimental results throughout this thesis are limited to the indoor motion capture facility. It would be of interest to evaluate our flight control algorithms in outdoor flights. Removing space constraints may enable flight at much higher speed and acceleration, so that the proposed controllers can be evaluated over a larger portion of the flight envelope, potentially up to the maximum airspeed. For the tailsitter, flight at increased airspeed also reduces the angle of attack, fully transitioning to steady forward flight, which is not possible in the constrained indoor flight space. The INDI control architecture may need to be extended to include wind compensation by accounting for a difference in the world-fixed velocity and the local freestream velocity, e.g., using the method proposed in [94].

Flight experiments outside the motion capture space require an alternative means of state estimation, such as a coupled satellite and inertial navigation system. A potentially more interesting future study would be to employ our proposed control algorithms on an autonomous vehicle that uses on-board sensors to enable fully autonomous fast and agile flight. In initial experiments, we have successfully flown our proposed quadcopter controller with visual-inertial state estimation using computer-generated camera images [35,107]. Future work may focus on elaborate autonomous flight experiments at high speeds and in various environments, potentially using real camera imagery.

6.2.2 Incremental Control

A practical requirement of INDI is the estimation or measurement of the vehicle angular acceleration. Implementations typically rely on differentiation and low-pass filtering of the angular rate measurements obtained from the IMU, but more sophisticated methods exist, e.g., predictive filtering [111] or using dedicated angular accelerometer hardware [55]. Since large and fast-changing angular accelerations are often necessary during agile maneuvering, research into measurement and estimation methods that reduce phase lag and transport delay may result in improved trajectory tracking performance.

The robustness of incremental control against unmodeled external disturbances may also be applied towards flight in tight formations. When aircraft fly close to each other, significant aerodynamic interactions may occur. Conventional control methods may not be able to account for the resulting forces and moments, leading to decreased performance and risk of collisions. In preliminary tests, we found that INDI control significantly improves hover stability in ground effect for both the quadcopter and flying wing aircraft, suggesting that it may also provide robustness against wake effects from other aircraft.

6.2.3 Differential Flatness

The proposed trajectory-tracking control methods for quadcopter and tailsitter aircraft are based on a similar architecture, leveraging INDI and flatness. Potentially, this architecture could also be used to develop robust controllers for tracking fast and agile maneuvers on other systems, such as conventional aircraft configurations or other mechanical systems that have flat dynamics (see, e.g., [87] for a catalog of flat systems).

As described in Chapter 2, differential flatness enables a modification of INDI control that avoids any linearization of the dynamics model. Additional insights into the advantages of this modification may be obtained from further analysis and rigorous comparison to conventional INDI on various flat systems.

6.2.4 Trajectory Generation

We note that our framework for multi-fidelity Bayesian trajectory optimization is agnostic to the actual trajectory generation. In principle, the only requirement is the existence of a map from a finite-dimensional parameter space to the function space of candidate trajectories. In Chapter 4, we use minimum-snap trajectory generation because it represents a widely-used baseline to improve upon and because the aforementioned map is conveniently provided in terms of the time allocation vector (i.e., as (4.6)). However, a different trajectory generation algorithm that is not constrained to polynomials may ultimately result in faster and more agile trajectories. For example, it may be possible to use a general-purpose optimal control solver, such as [92], and use Bayesian optimization to incorporate real-world experiments, e.g., in order to optimize the cost function by varying the input cost between trajectory segments.

Our current framework for Bayesian trajectory optimization considers a given set of fixed waypoints or obstacles and thus must be retrained using new flight experiments for a different set of trajectory constraints. Future work could focus on a more general application of multi-fidelity Bayesian optimization. For example, an extensive set of experiments on various trajectories and at multiple fidelities could be performed to learn a more general transformation between surrogate models at different fidelities. Next, this general transformation could be used to estimate feasibility constraints of the real vehicle on different trajectories using only low-fidelity evaluations, e.g., using only simulation. We note that learning such a general transformation may be challenging, and several obstacles must be resolved, e.g., trajectories with a different number of waypoints may originate from parameter spaces with different dimensionality.

Bibliography

[1] John D. Anderson. Aircraft performance and design. WCB/McGraw-Hill, Boston, 1999.

[2] Barton Bacon and Aaron Ostroff. Reconfigurable flight control using nonlinear dynamic inversion with a special accelerometer implementation. In AIAA Guidance, Navigation, and Control Conference and Exhibit, pages 4565–4579, 2000.

[3] Andrew J Barry, Tim Jenks, Anirudha Majumdar, Huai-Ti Lin, Ivo G Ros, Andrew A Biewener, and Russ Tedrake. Flying between obstacles with an autonomous knife-edge maneuver. In IEEE International Conference on Robotics and Automation (ICRA), pages 2559–2559, 2014.

[4] Jacson MO Barth, Jean-Philippe Condomines, Murat Bronz, Jean-Marc Moschetta, Cédric Join, and Michel Fliess. Model-free control algorithms for micro air vehicles with transitioning flight capabilities. International Journal of Micro Air Vehicles, 12:1756829320914264, 2020.

[5] Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with Gaussian processes. In IEEE International Conference on Robotics and Automation (ICRA), pages 491–496, 2016.

[6] Stefan Bieniawski, David Halaas, and John Vian. Micro-aerial vehicle flight in turbulent environments: Use of an indoor flight facility for rapid design and evaluation. In AIAA Guidance, Navigation and Control Conference and Exhibit, pages 6512–6522, 2008.

[7] Murat Bronz, Ezra Tal, Federico Favalli, and Sertac Karaman. Mission-oriented additive manufacturing of modular mini-UAVs. In AIAA Scitech 2020 Forum, page 0064, 2020.

[8] Adam Bry, Charles Richter, Abraham Bachrach, and Nicholas Roy. Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. The International Journal of Robotics Research, 34(7):969–1002, 2015.

[9] Daniel J Bugajski and Dale F Enns. Nonlinear control law with application to high angle-of-attack flight. AIAA Journal of Guidance, Control, and Dynamics, 15(3):761–767, 1992.

[10] Michael Burri, Janosch Nikolic, Helen Oleynikova, Markus W Achtelik, and Roland Siegwart. Maximum likelihood parameter identification for MAVs. In IEEE International Conference on Robotics and Automation (ICRA), pages 4297–4303, 2016.

[11] Romain Chiappinelli and Meyer Nahon. Modeling and control of a tailsitter UAV. In International Conference on Unmanned Aircraft Systems (ICUAS), pages 400–409, 2018.

[12] Hamidreza Chitsaz and Steven M LaValle. Time-optimal paths for a Dubins airplane. In IEEE Conference on Decision and Control (CDC), pages 2379–2384, 2007.

[13] Francisco Sahli Costabal, Paris Perdikaris, Ellen Kuhl, and Daniel E. Hurtado. Multi-fidelity classification using Gaussian processes: Accelerating the prediction of large-scale computational models. arXiv preprint arXiv:1905.03406, 2019.

[14] Kurt Cutajar, Mark Pullin, Andreas Damianou, Neil Lawrence, and Javier González. Deep Gaussian processes for multi-fidelity modeling. arXiv preprint arXiv:1903.07320, 2019.

[15] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for UAVs in cluttered environments. In IEEE International Conference on Robotics and Automation (ICRA), pages 42–49, 2015.

[16] Gabriele Di Francesco and Massimiliano Mattei. Modeling and incremental nonlinear dynamic inversion control of a novel unmanned tiltrotor. AIAA Journal of Aircraft, 53(1):73–86, 2015.

[17] Edsger W Dijkstra. A note on two problems in connexion with graphs. Numerische mathematik, 1(1):269–271, 1959.

[18] Christoph Dribusch, Samy Missoum, and Philip Beran. A multifidelity approach for the construction of explicit decision boundaries: Application to aeroelasticity. Structural and Multidisciplinary Optimization, 42(5):693–705, 2010.

[19] Guillaume Ducard and Raffaello D'Andrea. Autonomous quadrotor flight using a vision system and accommodating frames misalignment. In IEEE International Symposium on Industrial Embedded Systems (SIES), pages 261–264, 2009.

[20] Thomas Engelhardt, Thomas Konrad, Björn Schäfer, and Dirk Abel. Flatness-based control for a quadrotor camera helicopter using model predictive control trajectory generation. In Mediterranean Conference on Control and Automation (MED), pages 852–859, 2016.

[21] Dale F Enns, Daniel J Bugajski, Russ Hendrick, and Gunter Stein. Dynamic inversion: An evolving methodology for flight control design. International Journal of Control, 59(1):71–91, 1994.

[22] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. IEEE Robotics and Automation Letters, 3(2):620–626, 2018.

[23] Jeff Ferrin, Robert Leishman, Randy Beard, and Tim McLain. Differential flatness based control of a rotorcraft for aggressive maneuvers. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2688–2693, 2011.

[24] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. Sur les systèmes non linéaires différentiellement plats. CR Acad. Sci. Paris, pages 619–624, 1992.

[25] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. Linéarisation par bouclage dynamique et transformations de lie-bäcklund. Comptes rendus de l'Académie des sciences. Série 1, Mathématique, 317(10):981–986, 1993.

[26] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. Flatness and defect of non-linear systems: Introductory theory and examples. International Journal of Control, 61(6):1327–1361, 1995.

[27] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Trajectory tracking control design for autonomous helicopters using a backstepping algorithm. In American Control Conference (ACC), pages 4102–4107, 2000.

[28] Fei Gao, William Wu, Yi Lin, and Shaojie Shen. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In IEEE International Conference on Robotics and Automation (ICRA), pages 344–351, 2018.

[29] Jacob R. Gardner, Matt J. Kusner, Zhixiang Eddie Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian optimization with inequality constraints. In International Conference on Machine Learning (ICML), pages 937–945, 2014.

[30] Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In Conference on Neural Information Processing Systems (NeurIPS), pages 7576–7586, 2018.

[31] Vladislav Gavrilets, Emilio Frazzoli, Bernard Mettler, Michael Piedmonte, and Eric Feron. Aggressive maneuvering of small autonomous helicopters: A human-centered approach. The International Journal of Robotics Research, 20(10):795–807, 2001.

[32] Michael Adam Gelbart. Constrained Bayesian optimization and applications. PhD thesis, Harvard University, 2015.

[33] Marco B Gerig. Modeling, guidance, and control of aerobatic maneuvers of an autonomous helicopter. PhD thesis, ETH Zurich, 2008.

[34] H. Glauert. The Elements of Aerofoil and Airscrew Theory. Cambridge Science Classics. Cambridge University Press, 1983.

[35] Winter Guerra, Ezra Tal, Varun Murali, Gilhyun Ryou, and Sertac Karaman. FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality. arXiv preprint arXiv:1905.11377, 2019.

[36] James Hall and Timothy McLain. Aerobatic maneuvering of miniature air vehicles using attitude trajectories. In AIAA Guidance, Navigation and Control Conference and Exhibit, page 7257, 2008.

[37] Tarek Hamel, Robert Mahony, Rogelio Lozano, and James Ostrowski. Dynamic modelling and configuration stabilization for an X4-flyer. In 15th IFAC World Congress, pages 217–222, 2002.

[38] John Hauser and Rick Hindman. Aggressive flight maneuvers. In IEEE Conference on Decision and Control (CDC), pages 4186–4191, 1997.

[39] John Hauser, Shankar Sastry, and George Meyer. Nonlinear control design for slightly non-minimum phase systems: Application to V/STOL aircraft. Automatica, 28(4):665–679, 1992.

[40] James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian processes for big data. In Conference on Uncertainty in Artificial Intelligence, 2013.

[41] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In International Conference on Artificial Intelligence and Statistics, pages 351–360, 2015.

[42] José Miguel Hernández-Lobato, Michael A Gelbart, Matthew W Hoffman, Ryan P Adams, and Zoubin Ghahramani. Predictive entropy search for Bayesian optimization with unknown constraints. In International Conference on Machine Learning (ICML), pages 1699–1707, 2015.

[43] José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In Conference on Neural Information Processing Systems (NeurIPS), pages 918–926, 2014.

[44] Gabe Hoffmann, Dev Gorur Rajnarayan, Steven L Waslander, David Dostal, Jung Soon Jang, and Claire J Tomlin. The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC). In Digital Avionics Systems Conference (DASC), pages 12.E.4.1–10, 2004.

[45] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In AIAA Guidance, Navigation and Control Conference and Exhibit, pages 6461–6480, 2007.

[46] Jonathan P How, Brett Behihke, Adrian Frank, Daniel Dale, and John Vian. Real-time indoor autonomous vehicle test environment. IEEE Control Systems, 28(2):51–64, 2008.

[47] Deng Huang, Theodore T. Allen, William I. Notz, and R. Allen Miller. Sequential kriging optimization using multiple-fidelity evaluations. Structural and Multidisciplinary Optimization, 32(5):369–382, 2006.

[48] Haomiao Huang, Gabriel M Hoffmann, Steven L Waslander, and Claire J Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In IEEE International Conference on Robotics and Automation (ICRA), pages 3277–3282, 2009.

[49] Alberto Isidori. Nonlinear control systems. Springer, New York, 1995.

[50] Yeunduk Jung and David Hyunchul Shim. Development and application of controller for transition flight of tail-sitter UAV. Journal of Intelligent & Robotic Systems, 65(1):137–152, 2012.

[51] Jean-Marie Kai, Guillaume Allibert, Minh-Duc Hua, and Tarek Hamel. Nonlinear feedback control of quadrotors exploiting first-order drag effects. In 20th IFAC World Congress, pages 8189–8195, 2017.

[52] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. In IEEE International Conference on Robotics and Automation (ICRA), pages 4569–4574, 2011.

[53] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B. Oliva, Jeff Schneider, and Barnabas Poczos. Multi-fidelity Gaussian process bandit optimisation. arXiv preprint arXiv:1603.06288v4, 2016.

[54] Yijie Ke, Kangli Wang, and Ben M Chen. Design and implementation of a hybrid UAV with model-based flight capabilities. IEEE/ASME Transactions on Mechatronics, 23(3):1114–1125, 2018.

[55] Twan Keijzer, Gertjan Looye, Q Ping Chu, and Erik-Jan Van Kampen. Design and flight testing of incremental backstepping based control laws with angular accelerometer feedback. In AIAA Scitech 2019 Forum, page 0129, 2019.

[56] J Mark Keil. Decomposing a polygon into simpler components. SIAM Journal on Computing, 14(4):799–817, 1985.

[57] Marc C. Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. Biometrika, 87(1):1–13, 2000.

[58] TKJ Koo and SS Sastry. Output tracking control design of a helicopter model based on approximate linearization. In IEEE Conference on Decision and Control (CDC), pages 3635–3640, 1998.

[59] Ilan Kroo, Fritz Prinz, Michael Shantz, Peter Kunz, Gary Fay, Shelley Cheng, Tibor Fabian, and Chad Partridge. The Mesicopter: A miniature rotorcraft concept. Phase II interim report. Stanford University, 2000.

[60] Loic Le Gratiet and Josselin Garnier. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. International Journal for Uncertainty Quantification, 4(5):365–386, 2014.

[61] Daewon Lee, H Jin Kim, and Shankar Sastry. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. International Journal of Control, Automation and Systems, 7(3):419–428, 2009.

[62] Yuchen Leng, Thierry Jardin, Murat Bronz, and Jean-Marc Moschetta. Experimental analysis of a blown-wing configuration during transition flight. In AIAA Scitech 2020 Forum, page 1983, 2020.

[63] Joshua M Levin, Meyer Nahon, and Aditya A Paranjape. Real-time motion planning with a fixed-wing UAV using an agile maneuver space. Autonomous Robots, 43(8):2111–2130, 2019.

[64] David V Lindberg and Herbert KH Lee. Optimization under constraints by applying an asymmetric entropy measure. Journal of Computational and Graphical Statistics, 24(2):379–393, 2015.

[65] Zhenchang Liu, Jie Guo, Mengting Li, Shengjing Tang, and Xiao Wang. VTOL UAV transition maneuver using incremental nonlinear dynamic inversion. International Journal of Aerospace Engineering, 2018:1–19, 2018.

[66] Thomas Lombaerts, John Kaneshige, Stefan Schuet, Bimal L Aponso, Kimberlee H Shish, and Gordon Hardy. Dynamic inversion based full envelope flight control for an eVTOL vehicle using a unified framework. In AIAA Scitech 2020 Forum, page 1619, 2020.

[67] Leandro R Lustosa, François Defaÿ, and Jean-Marc Moschetta. Global singularity-free aerodynamic model for algorithmic flight control of tail sitters. AIAA Journal of Guidance, Control, and Dynamics, 42(2):303–316, 2019.

[68] Leandro Ribeiro Lustosa. The φ-theory approach to flight control design of hybrid vehicles. PhD thesis, ISAE-SUPAERO, 2017.

[69] Ximin Lyu, Haowei Gu, Jinni Zhou, Zexiang Li, Shaojie Shen, and Fu Zhang. Simulation and flight experiments of a quadrotor tail-sitter vertical take-off and landing unmanned aerial vehicle with wide flight envelope. International Journal of Micro Air Vehicles, 10(4):303–317, 2018.

[70] Tarek Madani and Abdelaziz Benallegue. Backstepping sliding mode control applied to a miniature quadrotor flying robot. In IEEE Conference on Industrial Electronics (IECON), pages 700–705, 2006.

[71] Alonso Marco, Dominik Baumann, Philipp Hennig, and Sebastian Trimpe. Classified regression for Bayesian optimization: Robot learning with unknown penalties. arXiv preprint arXiv:1907.10383, 2019.

[72] Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P. Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization. In IEEE International Conference on Robotics and Automation (ICRA), pages 1557–1563, 2017.

[73] Philippe Martin. Contribution à l'étude des systèmes différentiellement plats. PhD thesis, École Nationale Supérieure des Mines de Paris, 1992.

[74] Philippe Martin. Aircraft control using flatness. In IMACS/IEEE-SMC Multiconference and CESA Symposium on Control, Optimization and Supervision, pages 194–199, 1996.

[75] Philippe Martin, Santosh Devasia, and Brad Paden. A different look at output tracking: Control of a VTOL aircraft. Automatica, 32(1):101–107, 1996.

[76] Philippe Martin and Erwan Salaün. The true role of accelerometer feedback in quadrotor control. In IEEE International Conference on Robotics and Automation (ICRA), pages 1623–1629, 2010.

[77] Phillipe Martin, Richard M Murray, and Pierre Rouchon. Flat systems, equivalence and trajectory generation. Technical Report 2003.008, California Institute of Technology, 2003.

[78] Barnes W. McCormick. Aerodynamics, Aeronautics, and Flight Mechanics. John Wiley & Sons, 1995.

[79] K McIntosh, JP Reddinger, D Zhao, and S Mishra. Optimal trajectory generation for transitioning quadrotor biplane tailsitter using differential flatness. In Vertical Flight Society Annual Forum, pages 386–389, 2021.

[80] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In IEEE International Conference on Robotics and Automation (ICRA), pages 2520–2525, 2011.

[81] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. The International Journal of Robotics Research, 31(5):664–674, 2012.

[82] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP multiple micro-UAV testbed. IEEE Robotics & Automation Magazine, 17(3):56–65, 2010.

[83] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. Towards Global Optimisation, 2:117–129, 1978.

[84] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, et al. Challenges and implemented technologies used in autonomous drone racing. Intelligent Service Robotics, 12(2):137–148, 2019.

[85] Mark Müller, Sergei Lupashin, and Raffaello D'Andrea. Quadrocopter ball juggling. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5113–5120, 2011.

[86] Robin Murphy. Robots for unstructured and extreme environments, 2018.

[87] Richard M Murray, Muruhan Rathinam, and Willem Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. In ASME International Mechanical Engineering Congress and Exposition, 1995.

[88] Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary Gaussian process classification. Journal of Machine Learning Research, 9(Oct):2035–2078, 2008.

[89] Michael Ol, Greg Parker, Gregg Abate, and Johnny Evers. Flight controls and performance challenges for MAVs in complex environments. In AIAA Guidance, Navigation and Control Conference and Exhibit, pages 6508–6529, 2008.

[90] Mark Owen, Randal W. Beard, and Timothy W. McLain. Implementing Dubins airplane paths on fixed-wing UAVs. In Kimon P. Valavanis and George J. Vachtsevanos, editors, Handbook of Unmanned Aerial Vehicles, pages 1677–1701. Springer, 2015.

[91] Steve Paschall and Julius Rose. Fast, lightweight autonomy through an unknown cluttered environment. In IEEE Aerospace Conference, pages 1–8, 2017.

[92] Michael A Patterson and Anil V Rao. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. ACM Transactions on Mathematical Software (TOMS), 41(1):1–37, 2014.

[93] Paris Perdikaris, Maziar Raissi, Andreas Damianou, Neil D. Lawrence, and George E. Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 473(2198), 2017.

[94] Ole Pfeifle and Walter Fichter. Cascaded incremental nonlinear dynamic inversion for three-dimensional spline-tracking with wind compensation. AIAA Journal of Guidance, Control, and Dynamics, 44(8):1559–1571, 2021.

[95] Ole Pfeifle and Walter Fichter. Energy optimal control allocation for INDI controlled transition aircraft. In AIAA Scitech 2021 Forum, 2021.

[96] Paul Pounds, Robert Mahony, Peter Hynes, and Jonathan M Roberts. Design of a four-rotor aerial robot. In Australasian Conference on Robotics and Automation (ACRA), pages 145–150, 2002.

[97] Stefan A Raab, Jiannan Zhang, Pranav Bhardwaj, and Florian Holzapfel. Proposal of a unified control strategy for vertical take-off and landing transition aircraft configurations. In AIAA Applied Aerodynamics Conference, page 3478, 2018.

[98] Akshara Rai, Rika Antonova, Franziska Meier, and Christopher G. Atkeson. Using simulation to improve sample-efficiency of Bayesian optimization for bipedal robots. Journal of Machine Learning Research, 20(49):1–24, 2019.

[99] Dev G. Rajnarayan. Trading risk and performance for engineering design optimization using multifidelity analyses. PhD thesis, Stanford University, 2009.

[100] Carl Edward Rasmussen and Christopher K.I. Williams. Gaussian processes for machine learning. The MIT Press, 2006.

[101] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In International Symposium on Robotics Research (ISRR), pages 649–666. Springer, 2016.

[102] Marc Rigter, Benjamin Morrell, Robert G Reid, Gene B Merewether, Theodore Tzanetos, Vinay Rajur, KC Wong, and Larry H Matthies. An autonomous quadrotor system for robust high-speed flight through cluttered environments without GPS. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5227–5234, 2019.

[103] Robin Ritz and Raffaello D'Andrea. A global controller for flying wing tailsitter vehicles. In IEEE International Conference on Robotics and Automation (ICRA), pages 2731–2738, 2017.

[104] G Rivera and O Sawodny. Flatness-based tracking control and nonlinear observer for a micro aerial quadcopter. In AIP International Conference on Numerical Analysis and Applied Mathematics (ICNAAM), pages 386–389, 2010.

[105] Gilhyun Ryou, Ezra Tal, and Sertac Karaman. Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers. The International Journal of Robotics Research, 2021.

[106] Shankar Sastry. Nonlinear systems: Analysis, stability, and control. Springer-Verlag, New York, 1999.

[107] Thomas Sayre-McCord, Winter Guerra, Amado Antonini, Jasper Arneberg, Austin Brown, Guilherme Cavalheiro, Yajun Fang, Alex Gorodetsky, Dave McCoy, Sebastian Quilter, et al. Visual-inertial navigation algorithm development using photorealistic camera simulation in the loop. In IEEE International Conference on Robotics and Automation (ICRA), pages 2566–2573, 2018.

[108] Angela P Schoellig, Fabian L Mueller, and Raffaello D'Andrea. Optimization-based iterative learning for precise quadrocopter trajectory tracking. Autonomous Robots, 33(1):103–127, 2012.

[109] Judy E. Scott and Carlton H. Scott. Drone delivery models for medical emergencies. In Nilmini Wickramasinghe and Freimut Bodendorf, editors, Delivering Superior Health and Wellness Management with IoT and Analytics, pages 69–85. Springer, 2020.

[110] Dhwanil Shukla and Narayanan Komerath. Multirotor drone aerodynamic interaction investigation. Drones, 2(4):43, 2018.

[111] S Sieberling, QP Chu, and JA Mulder. Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction. AIAA Journal of Guidance, Control, and Dynamics, 33(6):1732–1742, 2010.

[112] P Simplício, MD Pavel, E Van Kampen, and QP Chu. An acceleration measurements-based approach for helicopter nonlinear flight control using incremental nonlinear dynamic inversion. Control Engineering Practice, 21(8):1065–1077, 2013.

[113] J.-J. E. Slotine and Weiping Li. Applied nonlinear control. Prentice Hall, Englewood Cliffs, 1991.

[114] Ewoud JJ Smeur, Murat Bronz, and Guido CHE de Croon. Incremental control and guidance of hybrid aircraft applied to a tailsitter unmanned air vehicle. AIAA Journal of Guidance, Control, and Dynamics, 43(2):274–287, 2020.

[115] Ewoud JJ Smeur, Qiping P Chu, and Guido CHE de Croon. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. AIAA Journal of Guidance, Control, and Dynamics, 38(12):450–461, 2015.

[116] Ewoud JJ Smeur, Guido CHE de Croon, and Qiping P Chu. Cascaded incremental nonlinear dynamic inversion control for MAV disturbance rejection. Control Engineering Practice, 73:79–90, 2018.

[117] P Smith. A simplified approach to nonlinear dynamic inversion based flight control. In AIAA Atmospheric Flight Mechanics Conference, pages 4461–4469, 1998.

[118] S Antony Snell, Dale F Enns, and William L Garrard. Nonlinear inversion flight control for a supermaneuverable aircraft. AIAA Journal of Guidance, Control, and Dynamics, 15(4):976–984, 1992.

[119] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Conference on Neural Information Processing Systems (NeurIPS), pages 1257–1264, 2006.

[120] Alexander Spitzer and Nathan Michael. Inverting learned dynamics models for aggressive multirotor control. In Robotics: Science and Systems (RSS), 2019.

[121] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. IEEE Transactions on Information Theory, 58(5):3250–3265, 2012.

[122] Sihao Sun, Angel Romero, Philipp Foehn, Elia Kaufmann, and Davide Scaramuzza. A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight. arXiv preprint arXiv:2109.01365, 2021.

[123] James Svacha, Kartik Mohta, and Vijay Kumar. Improving quadrotor trajectory tracking by compensating for aerodynamic effects. In International Conference on Unmanned Aircraft Systems (ICUAS), pages 860–866, 2017.

[124] Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity Bayesian optimization with max-value entropy search. arXiv preprint arXiv:1901.08275, 2019.

[125] Ezra Tal and Sertac Karaman. Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness. IEEE Transactions on Control Systems Technology, 29(3):1203–1218, 2021.

[126] Ezra Tal and Sertac Karaman. Global trajectory-tracking control for a tailsitter flying wing in agile uncoordinated flight. In AIAA Aviation 2021 Forum, 2021.

[127] Gao Tang, Weidong Sun, and Kris Hauser. Enhancing bilevel optimization for UAV time-optimal trajectory using a duality gap approach. In IEEE International Conference on Robotics and Automation (ICRA), pages 2515–2521, 2020.

[128] Teodor Tomic, Korbinian Schmid, Philipp Lutz, Andreas Domel, Michael Kassecker, Elmar Mair, Iris Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. IEEE Robotics & Automation Magazine, 19(3):46–56, 2012.

[129] Claire Tomlin, John Lygeros, Luca Benvenuti, and Shankar Sastry. Output tracking for a non-minimum phase dynamic CTOL aircraft model. In IEEE Conference on Decision and Control (CDC), pages 1867–1872, 1995.

[130] Guillem Torrente, Elia Kaufmann, Philipp Foehn, and Davide Scaramuzza. Data-driven MPC for quadrotors. IEEE Robotics and Automation Letters, 2021.

[131] Panagiotis Tsiotras and Ricardo Sanz Diaz. Real-time near-optimal feedback control of aggressive vehicle maneuvers. In Harald Waschl, Ilya Kolmanovsky, Maarten Steinbuch, and Luigi del Re, editors, Optimization and Optimal Control in Automotive Systems, pages 109–129. Springer, 2014.

[132] Mario Valenti, Brett Bethke, Gaston Fiore, Jonathan How, and Eric Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. In AIAA Guidance, Navigation, and Control Conference and Exhibit, pages 6200–6217, 2006.

[133] M Van Nieuwstadt, M Rathinam, and RM Murray. Differential flatness and absolute equivalence of nonlinear control systems. SIAM Journal on Control and Optimization, 36(4):1225–1239, 1998.

[134] Sebastian Verling, Basil Weibel, Maximilian Boosfeld, Kostas Alexis, Michael Burri, and Roland Siegwart. Full attitude control of a VTOL tailsitter UAV. In IEEE International Conference on Robotics and Automation (ICRA), pages 3006–3012, 2016.

[135] Xuerui Wang, Erik-Jan Van Kampen, Qiping Chu, and Peng Lu. Stability analysis for incremental nonlinear dynamic inversion control. AIAA Journal of Guidance, Control, and Dynamics, 42(5):1116–1129, 2019.

[136] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In International Conference on Machine Learning (ICML), pages 3627–3635, 2017.

[137] Xinglu Xia, Muqing Yang, Gang Chen, Liang Zhang, and Jiajia Hou. Transition flight control and simulation of a novel tail-sitter UAV with varying fuselage shape. IEEE Access, 9:65574–65587, 2021.

[138] Jia Xu. Design perspectives on delivery drones. Report RR-1718/2-RC, RAND Corporation, 2017.

[139] Rong Xu and Umit Ozguner. Sliding mode control of a quadrotor helicopter. In IEEE Conference on Decision and Control (CDC), pages 4957–4962, 2006.

[140] Yichong Xu, Hongyang Zhang, Kyle Miller, Aarti Singh, and Artur Dubrawski. Noise-tolerant interactive learning using pairwise comparisons. In Conference on Neural Information Processing Systems (NeurIPS), pages 2431–2440, 2017.

[141] Yunjie Yang, Jihong Zhu, and Jiali Yang. INDI-based transitional flight control and stability analysis of a tail-sitter UAV. In IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 1420–1426, 2020.

[142] Sam Zarovy, Mark Costello, Ankur Mehta, Greg Gremillion, Derek Miller, Badri Ranganathan, J Sean Humbert, and Paul Samuel. Experimental study of gust effects on micro air vehicles. In AIAA Atmospheric Flight Mechanics Conference, pages 7818–7844, 2010.

[143] Shuhan Zhang, Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, Xuan Zeng, and Xiangdong Hu. An efficient multi-fidelity Bayesian optimization approach for analog circuit synthesis. In ACM/IEEE Design Automation Conference (DAC), 2019.