Doctoral theses at NTNU, 2022:197

Dirk Reinhardt

On Nonlinear and Optimization-based Control of Fixed-Wing Unmanned Aerial Vehicles

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Dirk Reinhardt

On Nonlinear and Optimization-based Control of Fixed-Wing Unmanned Aerial Vehicles

Thesis for the Degree of Philosophiae Doctor

Trondheim, June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

© Dirk Reinhardt

ISBN 978-82-326-6666-9 (printed ver.)
ISBN 978-82-326-6479-5 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2022:197

Printed by NTNU Grafisk senter

To Astrid.
For your love and kind ways of explaining the flaws in my ideas to me.
And your patient looks when I pursue them anyway.

Summary

Methods from linear control theory have a considerable share in the state-of-the-art toolbox for control systems engineers when designing autopilots for conventional passenger aircraft. Linear controllers have well-established safety certificates leaning on gain and phase margins of the closed-loop dynamics. Passenger comfort further restricts the operational regime to regions where the aircraft's dynamics are approximately linear. Similar controller designs are part of the flight-control algorithms onboard small unmanned aerial vehicles (UAVs), for which we have seen increasing use over the past two decades. However, operators often can accept higher risks for low-cost UAVs. Initial experiments have shown that industry-standard linear controllers do not match the agile flight and performance in comparison to manual control of a skilled human pilot. The hypothesis that suitably designed nonlinear controllers can close the performance gap and therefore widen the flight envelope has been the driving motivation for the work described in this thesis.

The text briefly revisits preliminary concepts concerning kinetics and actuation of standard UAVs before continuing with the main contributions in nonlinear flight control. The work results can be summarized as:

A Geometric Attitude Controller with hybrid proportional feedback to track roll and pitch angle references is the subject of the first contribution. We avoid the gimbal lock problem of Euler angles and the unwinding phenomenon of quaternions by designing the control laws directly on the unit two-sphere. The resulting control law steers the attitude dynamics in the direction of the shortest path on the two-sphere towards the reference. With dynamic model inversion, the designed control law achieves almost semi-global exponential stability for time-varying attitude reference signals and almost global asymptotic stability of constant references. A hybrid extension to the proportional feedback results in stronger stability properties with global asymptotic stability regardless of the angular rates and global exponential stability if an additional angular rate condition is satisfied. Simulation studies and experimental results verify the efficacy of this approach.

In a second contribution, we exploit the designed geometric controller and use nonlinear model predictive controller (NMPC) to track roll, pitch and airspeed references. The NMPC generates suitable references for the geometric controller and adds constraint satisfaction and optimal performance.

In a third contribution, different NMPCs are designed with direct access to the actuators of the UAV. As for the geometric controller, the dynamic model of the NMPC uses the Special Orthogonal Group of order Three as attitude representation to avoid the outlined issues of alternative attitude representation and to decrease the nonlinearity of kinematic equations. In contrast to most other aerospace applications using NMPC, the designed controllers are not designed for guidance control with kinematic models and control-augmentation through a low-level proportional-integral-derivative (PID) controller, but include the full nonlinear kinematic and dynamic equations to control the low-level dynamics of the UAV. To the best of my knowledge the results in this thesis are the first go set the focus on attitude and speed control using NMPC. Simulation studies and experimental verification demonstrate improved performance to industry-standard PID controllers despite harsh weather conditions and significant model inaccuracies.

Considering that the focus of the thesis is on model-based control, we discuss improvements to the dynamic model of the UAV that was part of the experiments. The model identification procedure is based on wind tunnel measurements, including a novel calibration routine for planar symmetry of the airframe. The model from the wind tunnel is augmented with a damping model based on data collected in flight experiments. Cross-validation based on flight data shows significant model improvements compared to the baseline model that was used in the experiments.

As a final result, this thesis includes the design of a NMPC for the path-following problem with direct actuator access. It can be seen as an integrated approach that does not rely on a low-level motion controller, but directly controls actuator set-points and takes a parameterized path as input. Numerical results show significant performance improvements compared to the other cascaded controllers in the thesis, where an advanced guidance controller sends commands to low-level controllers.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the Norwegian University of Science and Technology (NTNU). I carried out the presented work at the Department of Engineering Cybernetics (ITK) under main supervision of Professor Tor Arne Johansen and Professor Thor Inge Fossen as my co-supervisor. My work has been supported by the Center for Autonomous Marine Operations and Systems (AMOS) through grant number 223254 and the Norwegian Research through project Autofly (grant number 261791/O70).

Acknowledgements

I got vital support from friends, colleagues, and family during my work as a Ph.D. candidate, and I appreciate having had a good time over the past four years thanks to many of them. My supervisors, Tor Arne Johansen and Thor Inge Fossen, gave me the chance to dive into a fascinating engineering field despite a notable lack of experience in guidance, navigation, and control of autonomous vehicles. In particular, Tor Arne's advice and feedback have been a reliable and essential guardrail throughout my academic endeavors. Erlend Coates, effectively my supervisor from day to day, even if not listed as one on paper, never lost interest in discussing a wide range of topics in control theory and practical applications related to our collaborations within the Autofly project. One indicator for continuous learning is in a room where you feel stupid at least once a day. Our office in D447 was a productive working environment in that sense. I want to give a shoutout to the people of the UAV-Lab, be it previous Ph.D. candidates who contributed work that I relied on or my colleagues who have been around when Erlend and I spent late nights at the lab to get our X8 into the air. In particular, to Kristoffer Gryte and Martin Sollie who directly supported us during the experiments and were always open to sharing their experience and discussing the problems at hand. Among the people at the lab, I want to extend my thanks to our skilled pilot Pål Kvåløy whose patience I stretched to its limits on several occasions. Thanks to Stefano Bertelli, Terje Haugen, and Glenn Angel for their technical assistance. The silent stars behind the scenes were Tove Kristin Blomset Johnsen and the staff around here. There was not one instance during my time at the department where I had to worry about administrative procedures of any kind.

Thanks to many colleagues and friends who gave me a joyful experience at the department, even considering the global pandemic.

The list is long, but I appreciated our mental resets in the skiing tracks, on the bike, during cabin trips, Aufguss sessions, the dance floor, a good-tempered trash talk along the hallway, and many other activities. There are so many of you, and you know who you are. I had a great time and hope this will continue after our periods as Ph.D. candidates.

Finally, thank you, Astrid for continuous review of my work-life-balance and bringing order and sunshine to our days. And thanks to my family in the south for their never-ending support. My grandmother Erika makes sure to let me know where my roots are. As parents, Sigrid and Hille have always been helping me out and trusted me to get things done, even though my physics grades were a hotter topic in the public Sauna than at the lunch table. The last but not least shoutout goes to my brother Lutz for not losing patience on whatever stupid question I throw at his programming expertise. If not lifting much else (grandfather Hermann's words), you for sure played an essential part in raising all my technical skills.

Contents

List of figures

List of tables

Glossary

ANN artificial neural network.
AUV autonomous underwater vehicle.

CFD computational fluid dynamics.

DARE discrete-time algebraic ricatti equation.
DRL deep reinforcement learning.
DUNE dynamic unified navigation environment.

EKF extended kalman filter.
ESC electronic speed control.

FPGA field programmable gate array.

GAC geometric attitude control.
GC geometric controller.
GNC guidance navigation and control.
GNSS global navigation satellite system.

HPIPM high-performance interior point method.

I2C inter-integrated circuit.
IMC inter module communication.
IMU inertial measurement unit.
INS inertial navigation system.
Ipopt interior point optimizer.

LAN local area network.
LMI linear matrix inequality.
LOS line-of-sight.
LPV linear parameter varying.
LQG linear quadratic gaussian.

LQR	linear quadratic regulator.
MIMO	multiple-input-multiple-output.
MPC	model predictive controller.
MSE	mean squared error.
MUX	multiplexer.
NDGPFG	nonlinear differential geometric path-following guidance.
NLP	nonlinear program.
NMPC	nonlinear model predictive controller.
OCP	optimal control problem.
ODE	ordinary differential equation.
PD	proportional-derivative.
PI	proportional-integral.
PID	proportional-integral-derivative.
PWM	pulse width modulation.
QP	quadratic problem.
RL	reinforcement learning.
RTI SQP	real-time iteration sequential quadratic programming.
RTL	return to launch.
SBC	single board computer.
SINDy	sparse identification of nonlinear dynamics.
SISO	single-input-single-output.
SITL	software-in-the-loop.
SQP	sequential quadratic programming.
STLSQ	sequentially thresholded least squares algorithm.
TECS	total energy control system.
UAV	unmanned aerial vehicle.
USV	unmanned surface vessel.
VTOL	vertical take-off and landing.

Chapter 1

Introduction

1.1 Motivation

The autopilot is one of the most critical components in any system designed for guidance navigation and control (GNC). On the lowest level in the control hierarchy, the task of the autopilot is to track reference orientation and speed[1] to enable the aircraft to follow a defined path or fly towards a specific location. In earlier applications, its primary purpose was stability augmentation to assist a human pilot in passenger aircraft, but engineers have since implemented them in a range of autonomous systems ranging from robotics platforms such as autonomous underwater vehicles (AUVs), unmanned surface vessels (USVs), UAVs to rockets and more recently, cars.

The necessary hardware for an onboard GNC system has been initially costly and heavyweight, but during the past 20 years, weight and prices have reduced to such a degree that low-cost vehicles find an increasing amount of applications and open-source projects. A good indicator for this trend is the rapid increase of available software and hardware solution for autopilots from 2007 [26] to 2018 [49], which opens up for a broad range of use cases. In the context of marine operations, the cheap autopilot technology facilitates the use of UAVs instead of surface vessels or ground vehicles. This can lead to reduced emissions [52], lower cost of operation, and faster completion of the mission.

One case is *search and rescue*, where a fixed-wing UAV is deployed from a surface vessel, as depicted in Fig. 1.1. The UAV is responsible for monitoring and object detection across the sea [110] to find the survivors of a boat accident. Once the UAV locates the survivors, the surface vessel can adapt its course to rescue them. A notable example is the searchwing project [162] where a hobby-grade UAV that costs less than 700 Euros assists surface vessels on their missions in the Mediterranean Sea. However, there are days when severe wind conditions do not allow to start the UAV. More challenging weather, such as at the coasts of the Atlantic Ocean, can be even more constraining to UAV operations that rely on industry-standard autopilot designs.

[1]Controller designs for this purpose will often be referred to as "low-level controller" in this thesis.

Throughout the thesis, we will refer to "challenging conditions" when we mean that the environmental disturbances due to wind are such that the wind speed makes up a significant portion of the vehicle's cruise speed with moreover rapidly changing magnitude and direction of the wind velocity vector due to gust winds. An example that we will encounter in experiments in a later chapter is the operation of the Skywalker X8 with a nominal cruise speed of 18 m/s in weather with 12 m/s average wind speed and 6 m/s gust winds. More advanced autopilots that take root in nonlinear control methods may reduce the number of days in which these environmental conditions are prohibitive to a safe operation.

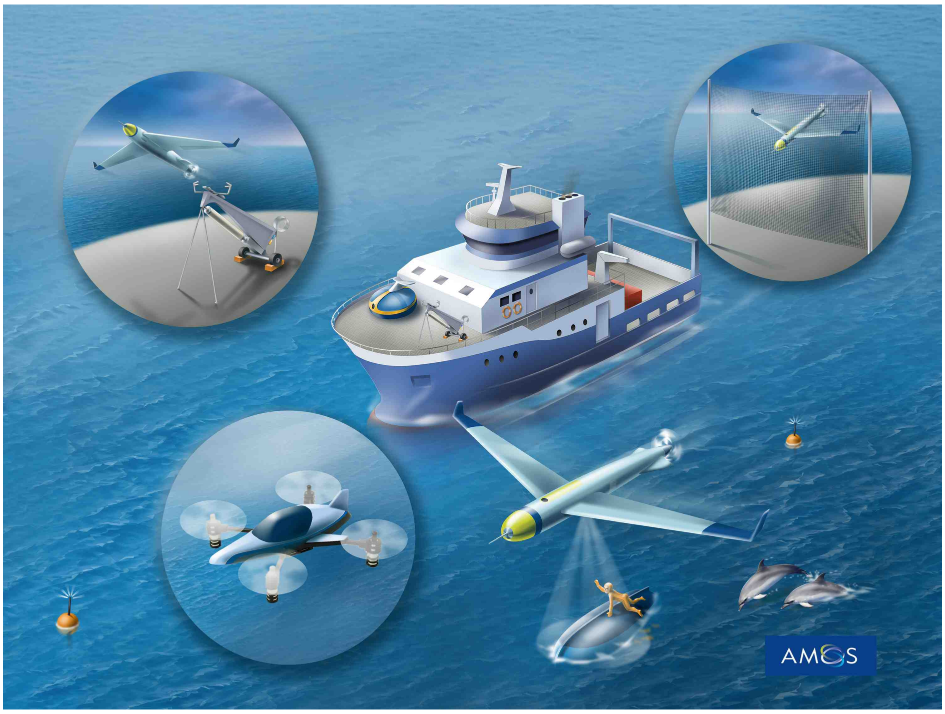

Figure 1.1: Deployment of a UAV to locate people in need for help in a search and rescue mission. Copyright: Bjarne Stenberg, NTNU.

UAVs can take a vital role in disaster management [78]. They provide mobile sensing and a communication network [135] to compensate for the loss of critical infrastructure resulting from natural catastrophes such as floods, forest fires, or earthquakes. The UAV can moreover be helpful in other surveillance missions similar to ocean monitoring, including applications within *traffic monitoring* in urban areas [94] or *precision agriculture* and *crop monitoring* [115, 149, 198]. The latter enables more efficient use of fertilizer or pesticides and thus has a positive environmental impact and increases productivity of the farm in general. Other applications include data acquisition and transfer in remote sensing operations, with the UAV serving as a data relay [148], and carrying of camera equipment in the *mapping* of terrain and urban areas [141].

They can also be included in logistics as part of the package delivery system. One example would be the transport of medical goods or devices to remote areas where multi-rotor UAVs would be too energy intensive [1].

State-of-the-art autopilots may be sufficient to be used in the outlined use cases under nominal conditions in a limited range of flight conditions, referred to as flight envelope. A careful operation of the UAV with mild maneuvers and conservative tuning of the control algorithms ensures that the UAV does not exceed the limits imposed by safety constraints. However, for agile flight, where the UAV tracks a high curvature path with the simultaneously extensive course- and climb-rates, nonlinear controllers are better equipped to deal with the nonlinearities that arise from the inherently nonlinear kinematic and dynamic equations of the problem. An example is loitering at a minimum turn radius [129].

As previously mentioned, adverse weather conditions may induce severe turbulence and wind gusts that may push the UAV to a roll angle as far as 140 degrees from its level flight reference. A skilled human pilot in manual control can recover the UAV [199]. However, the increasingly dominant nonlinearities make it difficult to do this with a standard autopilot based on linear control design. Other challenging maneuvers include deep stall landing [121]. A model of increased drag at higher angles of attack allows the controller to actively "brake" and thus hit the landing net at reduced kinetic energy and consequently with minor potential damage to the airframe or landing equipment.

Finally, stabilization of the payload such as a camera may be done in a more efficient way using aileron and rudder instead of only the aileron in a banked-to-turn maneuver. All these scenarios illustrate that the control algorithm can be a bottleneck, and suitable nonlinear controllers with a global region of attraction may enable flight in a broader envelope. Conservatism in operation can further be reduced by explicitly including the operational limits in the controller design, and we will discuss the types of controllers for which this is possible in the following section.

The objective of this thesis is to develop control algorithms to partially replace the autopilot in fixed-wing UAVs. When discussing the *autopilot*, we usually refer to the guidance controllers and low-level motion controllers as depicted in Fig. 1.2.

The focus will be on nonlinear controllers as an alternative to the widely adopted PID controllers presented in most textbooks [9, 177] and with slight adaptation often implemented in open-source autopilots [7, 128]. The focus of this thesis is on low-level motion control[2]. The controllers will be compared in simulations and experiments to evaluate their performance. We will look at geometric control algorithms for the attitude control problem in fixed-wing UAVs. The geometric controllers have a small computational footprint and engineers can implement them on embedded platforms with limited computing power. Another control methodology in this thesis will be numerical optimal control, and we use NMPC [68, 151] to achieve at least comparable performance to the baseline controller.

Additional desirable features include the enforcement of the constraints arising from physical limits of the UAV, e.g., constraints due to actuator saturation and

[2]A brief presentation of an extension of the developed NMPC to the guidance control problem is included in the end

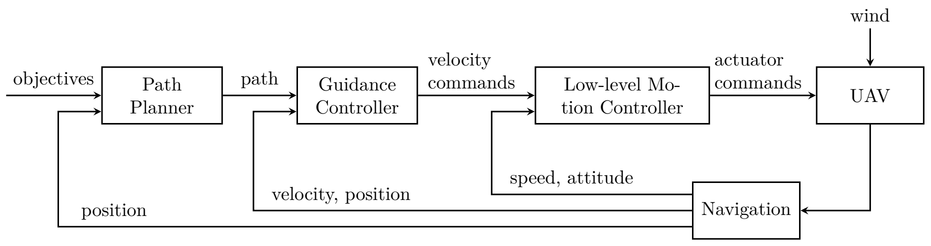

Figure 1.2: Example of a control hierarchy that can be treated via successive loop closure.

aerodynamic phenomena such as stall at high angle of attack. The resulting controller provides low-level controller that addresses safety considerations. Classical PID loops would require the outer-level guidance logic to be cautious and restrict the references passed to the low-level controller [33].

1.2 Background

To provide the context of the contributions, this section includes an overview of state-of-the-art control algorithm designs for fixed-wing aircraft and UAVs. It starts with classical methods of linear control theory that are part of the control architecture of modern passenger aircraft that require strict safety certificates and are operated in a minimal flight envelope to ensure safety and passenger comfort. We discuss their limitations and provide some directions of more advanced controllers that still use linear models before giving an overview of nonlinear control methods and some example applications in the field of fixed-wing UAVs.

This section is by no means an exhaustive survey but rather a list of examples of how it is possible to achieve more agile flight by using nonlinear control methods. We hint to relevant survey papers for additional references. A detailed technical discussion is out of scope, but a treatment of the relevant concepts is given by How et al. [80] for linear flight control and Girish et al. [63] for nonlinear flight control.

1.2.1 Linear Controller Review

Aerospace engineers have widely adopted PID control based on single-input-single-output (SISO) feedback loops. The design is the subject of any textbook that discusses controller design for aircraft, such as [9, 40, 51, 138, 177], and we usually find modified implementations in open-source projects. See for example the implementation in *PX4* [128] or *Ardupilot* [7]. For the attitude controller on the lower level of the control hierarchy, Euler angles are often the go-to attitude representation with one individual control loop design for each angle as shown by Beard and McLain [9]. The implied assumption is that the motion of the aircraft can be decoupled into lateral and longitudinal direction. This is a fair approximation for mild maneuvers, but becomes increasingly invalid at more dynamic flight. However, there exist more general formulations based on quaternions as shown by Mayhew et al. [124] or rotation matrices, discussed by Chaturvedi et al. [28].

The resulting controllers do not need the decoupling assumption. In either attitude representation, there is a separate outer-level control loop that deals with speed control as outlined by Sobolic [171]. Even with careful designs and suitable feedforward terms it can be difficult to tune the parallel feedback loops to give satisfactory results in a wide range of conditions.

A more integrated approach in which multiple-input-multiple-output (MIMO) feedback loops can be considered, is the linear quadratic regulator (LQR). However, it is common practice to have an estimator for navigation in the feedback loop. In case this is an extended kalman filter (EKF), the combination of LQR and EKF is called linear quadratic gaussian (LQG) control. LQG control works well for MIMO problems where a linear state-space representation of the dynamics is available and examples can be found in the textbooks of Stengel [176] and Lavretsky and Wise [103].

Even though the UAV dynamics are inherently nonlinear, a linearization of the dynamic model at equilibrium conditions (trim points) is usually available. The linearized system matrices are then used in the controller design, and established linear control theory can be applied, see for example Dorf and Bishop [48]. The performance is specified based on weighting matrices that penalize tracking error of the states and actuator usage. The designer can prioritize tracking of specific states and specify how costly actuator usage is, facilitating energy-efficient controller design.

The LQG minimizes the sum of squares of these metrics for all frequencies by solving the discrete-time algebraic ricatti equation (DARE) that results from the specification of the objective function and the linearized dynamics. The performance is optimal but also sensitive to modeling errors. Notable examples for the successful application include the work of Cory and Tedrake [41] and Moore et al. [133], who achieved impressive results using a tree of time-varying LQRs as introduced by Tedrake et al. [180]. They approximate the NMPC but avoid its computational demands by continuously linearizing the system dynamics at the reference trajectory to compute the solutions to the DARE.

A different MIMO design methodology that addresses robustness issues, is H Infinity control, explained by McFarlane and Glover [127]. The design procedure starts with the definition of weighted sensitivities for tracking performance, actuator usage, and disturbance rejection. An iterative loop shaping procedure can then minimize the respective gains of the closed-loop transfer function for all frequencies as discussed in Skogestad and Postlethwaite [168].

A design choice is the compromise between good reference tracking at low frequencies and disturbance rejection at high frequencies. The method is well-studied for fixed-wing aircraft, and there are toolboxes tailored to aerospace applications for which Hyde et al. [81] is one example. An extension of this approach is μ-synthesis, allowing the designer to include parametric uncertainties. Uncertain mass, inertia, or aerodynamic effects, can then be part of the controller design, and Michailidis et al. [130] make use of this for attitude/speed control. Modeling errors again may lead to a degradation in both robustness and performance.

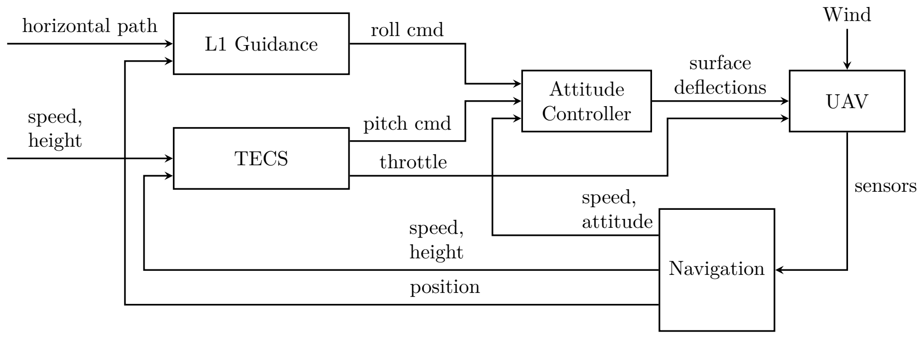

Figure 1.3: Example of a fixed-wing guidance, navigation and control scheme that is implemented in PX4 and Ardupilot.

1.2.2 Nonlinear Controller Review

Even though linear control theory has a long history and may be regarded as more mature, there is a rich set of nonlinear control methods which are described in, for example, Slotine et al. [169] and Khalil [95]. Many have been applied to aircraft and, in particular, to UAVs. The drawbacks of linear methods include the lack of global stability results for nonlinear systems and performance can quickly degrade in situations where nonlinear dynamics effects significantly impact the overall dynamics. Nonlinear effects may include time- or state-varying dynamics, such as different linear models at individual equilibrium conditions, nonlinear aerodynamics, and inherently nonlinear kinematics. Other nonlinearities are saturation limits of the actuators and safety-related constraints on the state. We will give a short overview of some established methods and how engineers applied them to control fixed-wing UAVs.

One nonlinear control methodology that bridges the gap to linear control, and is part of flight control designs, is *gain scheduling*, as discussed in, for example, Rugh and Shamma [160]. The procedure is such that the control system designer identifies a range of equilibrium conditions within the operational regime, the flight envelope in aerospace applications, and linearizes the dynamics at each equilibrium state. For each equilibrium condition, the parameters of the state-space matrices will be different, leading to a linear parameter varying (LPV) model. For a switched system of controllers, Lyapunov theory shows that if the gain matrix for each controller satisfies the Lyapunov equation, the system asymptotically converges to the equilibrium for each controller. Based on the partitioned state space, a linear matrix inequality (LMI) can be solved to establish the stability result for the gain-scheduling controller. Linear controllers such as the ones just mentioned can then be designed for each pair of state and linearized dynamics.

Slowly time-varying state variables, for example, airspeed, can be used to compute a scheduling variable that interpolates the controller parameters based on its current value and the value at the neighboring equilibrium conditions. Gain scheduling is therefore also referred to as parameter varying control. Biannic et al. [12] discuss longitudinal control based on gain scheduling with H Infinity control and μ synthesis, where they use speed and height as scheduling variables. An extended gain-scheduled longitudinal controller based on the solution of a LMI and a descriptor representation of the LPV system is presented by Masubuchi et al. [120]. Lee et al. [104] present a lateral gain scheduling controller as a solution to the bank-to-turn control problem based on Eigenvalue assignment for linear controller design. Angle of attack commands and speed determine the scheduling of the bank of linear controllers.

In *Dynamic Model Inversion*, the designer composes a control law such that the originally nonlinear system exhibits closed-loop dynamics of a reference model, which may have linear form. As the dynamic models of a UAV are often in control-affine form, this means that the control effectiveness matrix needs to be inverted to cancel modeled nonlinear terms via the control law. One can then use established linear control theory results to prove exponential stability. Publications from the 80s include design examples for aircraft control in Smith et al. [170] and Lane et al. [102]. The strong theoretical results are appealing, but the drawback of this approach is the potential high actuator usage to cancel the nonlinear terms. Moreover, in practice, some of them may be useful for a suitably designed controller.

Sliding Model Control is a more modern approach to the control of nonlinear systems, with theoretical contributions published by Shtessel et al. [165, 166], among other authors. It can be a tool for control problems that include measurement uncertainties, external disturbances, and modeling errors. Around 2000, Levant et al. [111] published their work on pitch control to compensate for unknown external disturbances. A second-order sliding surface is used in the controller design to avoid the chattering phenomenon known for first-order sliding mode control [164].

Backstepping control is another Lyapunov-based design methodology that provides a recursive approach for stabilizing systems with a dynamic representation in the strict feedback form. Details can be found in Khalil's text book [95]. Most aircraft control architectures that exploit timescale separation principles to increase bandwidth from the outer level to the low-level dynamics, allow for the strict feedback form. The idea is to use a virtual control variable to stabilize an internal subsystem and then design the outer loop to stabilize the virtual control variable. This way, a possibly complex control law that stabilizes the system can be designed by recursively stabilizing each subsystem. For fixed-wing aircraft, one would design a position controller by using course, flight-path angle, and speed as virtual control inputs. In another step, control laws to stabilize flight-path angle and speed can be designed based on the aerodynamic angles and the thrust input. Sonneveldt et al. [173] demonstrate the adaptive backstepping control design for fighter aircraft in this way.

An outstanding example of how a nonlinear control law can outperform its linear counterpart is the "New nonlinear guidance logic" proposed by Park et al. [147] in 2004 to provide lateral guidance based on acceleration commands. One can argue about how well the contribution title is aging, but the proposed guidance method is widely adopted and has since remained an essential part of the code-base for both PX4 and Ardupilot. Controlling specific force accelerations through to follow a path allows for separate roll control through the aileron. This can be exploited for aerobatics as demonstrated by Park [146]. Cho et al. extended the guidance logic to three dimensions in 2015 by Cho et al. [32]. However, guidance controller implementations in PX4 and Ardupilot use the original lateral guidance logic and the "Total Energy Control System" by Lambregts et al. [101] for longitudinal guidance. The combination of both guidance controllers can be found in the control architecture of PX4 and ArduPilot as depicted in Fig. 1.3.

Kai et al. [87] recently published a more unified approach for the path-following problem on model-scale fixed-wing UAVs. They do not construct a composite Lyapunov function for the total control system as in backstepping designs, but give a detailed derivation for each subsystem that relies on stabilization of its virtual control variable through an inner-loop controller. They prove ultimately exponential convergence for each subsystem and use cascade stability arguments to prove the stability of the total control architecture. Bulka et al. [17] discuss another unified approach to keep the controller design platform generic. They design the control laws assuming that the UAV can generate a force vector along a body-fixed direction and a moment vector about an arbitrary axis. A static mapping then transforms the control forces to a specific platform. Together with the use of dynamic mode inversion, this suggests that accurate models of the UAV need to be available. The propulsion system of most UAVs satisfies the assumption of a body-fixed force vector along the longitudinal axis. However, the moment assumption is more restrictive, and classic fixed-wing UAVs do not satisfy it. The type of agile fixed-wing UAVs that Bulka et al. use in their experiments can generate arbitrary moment vectors thanks to its low weight, which gives it a high thrust-to-weight ratio and the resulting maneuverability, as demonstrated in prior work by Khan and Nahon [96]. Cory and Tedrake [41] use a similar light-weight UAV in their perching experiments but significantly less actuation. See also the work of Moore et al. [133] and Basescu and Moore [8], who recently published results on post-stall motion planning based on NMPC with direct actuator access. The outlined results are impressive regarding the agile maneuverability thanks to model-based control algorithms. However, the experiments are conducted in controlled lab environments where wind disturbances are not a factor. This thesis follows the trend of model-based controllers. The distinction is that we target outdoor applications in potentially harsh weather conditions.

NMPC extend the abilities of LQR control with the short-term prediction of the system trajectories and inherent enforcement of system constraints. A linear state-space model is not required, and the full nonlinear dynamics can be part of the constraints as long as the formulation is computationally tractable. Depending on the dynamic model, employed solver, and embedded hardware, the optimal control problem can repeatedly be solved fast enough for the desired update rate of the controller. The computational demands of NMPC have been prohibitive for real-time applications in UAVs due to the need for high update rates to control the fast dynamics. This is the reason that we mainly see linear MPCs in aerospace control in the past two decades with local linear models in a small envelope where it can be hard to outperform a PID controller. To meet the computational capacities of available hardware, engineers have mostly looked at NMPC for kinematic control with a low-level autopilot in the loop, an approach called control-augmented MPC. Good examples for this are the publications by Yang et al. [196], Garcia et al. [61], Stastny et al. [175], Gavilan et al. [62], and Alessandretti et al. [3]. A comprehensive overview of MPC in aerospace systems is the work by Eren et al. [50].

In all the mentioned examples, aerodynamic effects and control surface limits remain hidden behind a layer of abstraction in the control-augmented approach, making them inaccessible to the MPC.

Except for iterative LQR and NMPC, the outlined nonlinear control methodologies extend linear control theory in the sense that they can handle nonlinear process models and add a level of adaption. Proofs of asymptotic or exponential convergence to a bounded set around the reference exist for Lyapunov-based designs. However, they often do not add much conceptually to regular PID designs. In contrast, NMPC, with its predictive capabilities and constraint enforcement, actually brings something new to the table. Therefore, it is subject to a significant part of this thesis, and we give it more focus in the next section.

1.2.3 Model Predictive Control

The key strengths of (nonlinear) MPC are the simple design of MIMO control loops, constraint enforcement, inherent robustness, and performance optimization, as discussed by Di Cairano and Kolmanovsky [45]. But there are challenges to a real-time controller that delivers satisfactory results. In addition to the mentioned high computational capacities, we need a process model that captures nonlinearities that are in particular significant during transients. Finding an accurate model usually requires a substantial system identification effort. Other obstacles can be the dependence on accurate state estimates and a lack of well-studied tuning guidelines. We address each desirable feature of MPC and questions on tackling the challenges separately in the following.

The multivariable control ability of MPC can already yield superior performance to SISO control loops that connect single pairs of state variables and actuators, and an increase of available actuators in the configuration adds to the gap. Together with reinforcement learning (RL), it is one of the go-to methods for complex tasks such as gait control of quadrupeds which is shown by Neunert et al. [140].

Another complex task is the control of the take-off and landing in vertical take-off and landing (VTOL) tilt-wing UAVs as Rohr et al. [159] demonstrate. One appeal of MPC is that it approximates LQR locally, including its guaranteed gain and phase margin, explaining its inherent local robustness. A global robustness result is hard to achieve but considering practical limitations of the aerodynamic model, which is usually the bottleneck, is not the most limiting factor in practice.

The explicit enforcement of constraints allows for a more aggressive controller design in the presence of safety constraints, which would otherwise be dealt with by careful tuning and protection logic within the control architecture. Another type of constraint, such as those naturally arising from collision avoidance objectives or cooperative missions, can be easily added. The designer of the NMPC can address performance through the objective function and conduct the tuning in the space of the performance criterion itself rather than tweaking proportional or derivative gains of a proportional-derivative (PD) controller. However, the other way of looking at it is that it is not intuitive to provide higher bandwidth and modify damping.

The inherent re-planning capabilities of MPCs can increase the autonomy of UAVs as it can re-compute trajectories in events of appearing obstacles and system failures, which is shown by Kufoalor and Johansen [98].

A feature that is generally nice to have and particularly important in the event of delays or loss of communication.

As mentioned, the repeated solution of the optimal control problem results in a high computational footprint that needs to be met by the embedded computing platform. The required computing capacity can be problematic on platforms restricted to robust and resource-limited microcontrollers. However, in the field of fixed-wing UAVs, it is often possible to extend the flight stack with a computationally powerful SBC. The cost of the added hardware is usually a fraction of a standard fixed-wing UAV's hardware configuration and well-developed open-source solvers are available. One software package for embedded optimization is *acados*[3], which is widely adopted by the research community and under active development by the group of Mortitz Diehl. For more details about acados and a list of alternative software packages see [185].

The dynamics of fixed-wing aircraft are well-studied and understood to the extent that it remains to identify a model of the generalized aerodynamic forces of a particular airframe either in wind tunnel experiments or based on flight data. The result is a nominal physics-based model. To further improve the model, data-driven approaches are available to capture the mismatch online, for example based on results from Gaussian Process regression published by Williams and Rasmussen [191]. Successful applications include the work by Hewing et al. [76] on racing cars, and by Torrente et al. [184] on multi-rotor UAVs. Similar results for fixed-wing UAVs have not been published.

Full-state feedback needs to be available to set the initial conditions of the controller model at each update. The estimator in the feedback loop is thus a critical component that can be a limiting factor to the overall control performance. In UAV applications, the use of inertial navigation systems (INSs) aided by global navigation satellite systems (GNSSs) is state-of-the-art and Farrel et al. [53] and Markley et al. [119], among others, document available algorithms. Recent results on wind velocity estimators include the work by Johansen et al. [85] and Wenz and Johansen [189], with available open-source implementations [7].

Finally, safety certificates for aerospace applications lean on phase and gain margins in the relevant frequency spectrum of the closed-loop. It is relatively straightforward to establish the properties for linear systems, but a translation from MPC is difficult. For UAV operations, higher risk can be accepted compared to passenger aircraft, but certification remains a topic of current development.

The attitude control problem has been addressed with NMPC in research on multi-rotor UAVs by Kamel et al. [91] and Zanelli et al. [197]. Gupta et al. [73] and Kalabic et al. [89, 90] solve the problem for constrained maneuvers with satellites. However, there are few applications to fixed-wing UAVs. Those that exist use linear perturbation models and thus linear MPC, as Oettershagen et al. [143] or Mammarella et al. [116]. The difference is arguably in the complexity of modeling the actuated rotational dynamics.

[3]`https://github.com/acados/acados`

Rotor speeds that can be statically mapped to the resulting torque vector (multi-rotors, see [184]) or reaction wheels to induce the control moment directly (satellites, see [119]) are more straightforward than the aerodynamic effects of deflected control surfaces (fixed-wing).

1.3 Research Objectives

Being part of the Autofly project team, which includes one post-doc and two Ph.D. candidates, with numerous colleagues at the UAV Lab opens for a range of intriguing pursuits that one can follow. The overall objective of the Autofly Project and this thesis was to address fundamental research questions related to the nonlinear control of UAVs.

Our research hypothesis was that nonlinear autopilots could significantly increase the flight envelopes of fixed-wing UAVs compared to conventional linear autopilot designs. An increased flight envelope opens for a broad range of possible new applications, such as the described maneuvers in the preceding sections. Exploiting the physical limitations by using advanced nonlinear control designs can increase the UAV's performance. Further, increasing the autopilot's capabilities to reject environmental disturbances opens for operation in more challenging weather conditions and therefore increase the number of days UAVs can be operated safely. The idea that nonlinear designs can recover control after severe failures and unforeseen events is a step towards increased autonomy of the UAVs.

We seek nonlinear controller designs that achieve high performance and global asymptotic closed-loop stability to tackle sudden disturbances that cast the UAV far from its nominal desired state. Recall the example of attitude stabilization with a 140-degree angle between the UAV roll angle and its reference. This scenario is a strong motivation for global closed-loop stability, for which we also wanted to achieve fault tolerance and robustness of the nonlinear autopilot system. Where possible, the stability properties of the designed control laws should be proven based on rigorous Lyapunov stability analysis.

The focus is on the development of low-level motion control algorithms with explicit consideration of nonlinear aerodynamic models and actuator limitations. The autopilot designs should be tested in numerical case studies that include nonlinear simulation models with realistic aerodynamic coefficients obtained in wind tunnel experiments.

The final essential factor for the development process is the experimental verification of the designed nonlinear autopilot algorithms, with successful demonstrations of their real-time implementation in challenging applications and case studies at NTNU's UAV Lab. Overall, the thesis focuses on the benefits of advanced control methods in implementable autopilot designs and related control tasks for fixed-wing UAVs. We are not trying to establish fundamentally novel contributions to nonlinear control theory, but instead use recent advanced control methods, such as Model Predictive Control, as a basis for designing the autopilot and overall control tasks. Practical demonstrations in experiments have always been a major goal in our work.

More specifically, the research objectives led us to consider the following tasks in this thesis:

- Design Lyapunov-based attitude control laws to meet desirable global stability and performance requirements in the unconstrained case. Use hybrid feedback control, motivated by [126], to overcome the topological obstruction of the attitude control problem and globally stabilize time-varying reference signals, which can not be done by smooth feedback alone.
- Design (nonlinear) MPCs for low-level motion control that satisfies system constraints and provides optimal performance. Due to the high computational footprint of MPC, implementing it on resource-limited embedded computing platforms can be challenging, which possibly limits the MPC to serve as a reference method for other developed controllers with a smaller computational footprint. Investigate the feasibility of onboard control in real-time and further pursue MPC designs for experimental testing if the employed nonlinear solver meets the closed-loop runtime requirements on the targeted embedded hardware.
- Benchmark the low-level motion control methods using developed nonlinear simulator models. The benchmarks will, in particular, focus on disturbance rejection in highly turbulent flight and recovery after the loss of control, leading to attitude angles far from the reference with up to 140 degrees.
- Develop an experimental platform that allows testing the developed controllers in flight experiments. The goal is to enable the rapid development and testing of highly experimental control algorithms. To this end, the architecture and procedures of the platform need to be designed in a way that does not interfere with the standard autopilot system such that a safe fall-back mechanism is always available.

1.4 Contributions and Outline

The thesis is organized into 8 chapters, including this introduction and a concluding chapter. In the following, we look at the topic and contributions of Chapter 2 to Chapter 8.

Chapter 2: Preliminaries

This chapter introduces preliminary concepts necessary for the following chapters, ranging from the notation, coordinate frames and the dynamic models used for the model-based controller designs to the experimental platform to validate the controllers. The feedback assumptions are outlined, and a motivational example concludes the chapter. The content of this chapter summarizes results that appeared in publications by other authors; among others Beard and McLain [9] and Stevens et al. [177].

Chapter 3: Experimental Platform

Publications:

[38] Erlend M. Coates, Dirk Reinhardt, Kristoffer Gryte, and Tor Arne Johansen. Toward Nonlinear Flight Control for Fixed-Wing UAVs: System Architecture, Field Experiments, and Lessons Learned. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, accepted

Topic: The intention of this chapter is to cover implementation aspects and test procedures in relation to our work in the Autofly project. It is not a contribution that covers research questions concerning autopilot designs of fixed-wing UAVs, but is meant to facilitate rapid testing of experimental low-level motion controllers. The focus is on the hardware components and communication architecture to allow implementing experimental algorithms. The outlined architecture does not interfere with the standard flight stack in order to keep it as a fall-back mechanism whenever necessary, which was one of the design requirements. We moreover describe development tools ranging from simulation to experiment protocols that are proven in the field. The publication moreover contains valuable lessons learned and experimental results that are not included in this chapter.

Contributions: The nature of this chapter's contributions is less academic and is intended as a description of our experimental platform and valuable experiences that we had to collect the hard way when working on the flight stack and implementations of the control algorithms. It moreover includes valuable practices than can be implemented by other researchers to facilitate rapid testing of experimental controllers that require direct access to the actuators with minimal impact on safety protocols.

Chapter 4: Geometric Attitude Control

Publications:

[37] Erlend M. Coates, Dirk Reinhardt, and Thor I. Fossen. Reduced-Attitude Control of Fixed-Wing Unmanned Aerial Vehicles Using Geometric Methods on the Two-Sphere. *IFAC-PapersOnLine*, 53(2):5749–5756, 2020. 21st IFAC World Congress

[153] Dirk Reinhardt, Erlend. M. Coates, and Tor Arne Johansen. Hybrid Control of Fixed-Wing UAVs for Large-Angle Attitude Maneuvers on the Two-Sphere. *IFAC-PapersOnLine*, 53(2):5717–5724, 2020. 21st IFAC World Congress.

Topic: This chapter includes the presentation of geometric attitude controllers for the tracking control of roll and pitch references. The control law leverages attitude representations on the two-sphere and thus avoids problems of conventional representations such as Euler angles or quaternions. Euler angles have the drawback of the singularity known as "Gimbal lock". A design of separate control loops for pitch and roll control can introduce disturbances that a coupled roll and pitch control can avoid. Quaternion-based control formulations can address a coupled controller formulation but usually control the entire attitude, which is unsuitable for fixed-wing UAVs where heading or course is controlled through bank-to-turn maneuvers. An additional drawback of attitude controller designs based on quaternions is the unwinding phenomenon that results from the double cover of the Special Orthogonal Group of order Three by the quaternion space.

Some authors address this with a switching logic based on the sign of the quaternion. By directly designing the feedback control law on the two-sphere, we avoid the singularities of Euler angles and the unwinding phenomenon of quaternions. The coupling between roll and pitch is a natural result of the design on the two-sphere, and we design proportional state feedback such that the controller is incentivized to steer the attitude in the direction of the shortest path. We establish almost semi-global exponential stability for attitude tracking of a time-varying reference signal and almost global asymptotic stability for attitude regulation to a constant reference.

A demonstration in a simulation study shows the efficacy of our approach before we provide the extension to a hybrid control law that achieves global exponential stability, which we show in detailed proofs based on results in hybrid Lyapunov stability theory published by Goebel et al. [64]. We draw on similar results that Mayhew et al. [126] and Casau et al. [24] published for general rigid bodies and work done by Lee [108] on multi-rotor UAVs. The chapter ends with an experimental demonstration showing the practical use of our results.

Contributions: The contributions include a solution to the low-level attitude control problem based on results in geometric control to which Bullo et al. made substantial contributions [18, 21]. We propose an alternative attitude representation that can replace the frequently used expressions based on Euler angles or quaternions. A Lyapunov-based stability analysis establishes almost global asymptotic stability in the case of regulating to constant reference and semi-global exponential stability when tracking time-varying references. The extension to hybrid proportional feedback is shown to result in global exponential stability. This contribution is the first to apply geometric attitude control to the low-level control problem of fixed-wing UAVs.

Chapter 5: Direct Nonlinear Model Predictive Control for Attitude and Speed Control

Publications:

[155] Dirk Reinhardt and Tor Arne Johansen. Nonlinear Model Predictive Attitude Control for Fixed-Wing Unmanned Aerial Vehicle based on a Wind Frame Formulation. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 503–512, June 2019

[156] Dirk Reinhardt and Tor Arne Johansen. Control of Fixed-Wing UAV Attitude and Speed based on Embedded Nonlinear Model Predictive Control. *IFAC-PapersOnLine*, 54(6):91–98, 2021. 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021

[152] Dirk Reinhardt, E. M. Coates, and Tor Arne Johansen. Low-level Nonlinear Model Predictive Attitude and Speed Control of Fixed-Wing Unmanned Aerial Vehicles. *Control Engineering Practice*, submitted.

Topic: We exploit the attitude representation introduced in the preceding chapters to design different control strategies based on NMPC with direct access to the control surfaces to solve the problem of controlling the attitude and speed of a fixed-wing UAV. As we discussed previously, it is the state of the art to use NMPC to design kinematic controllers for fixed-wing UAVs that rely on off-the-shelf low-level autopilots to stabilize the UAV to given attitude and speed commands. There exist linear MPC designs to control the low-level dynamics, but we could not find publications where the full nonlinear dynamics model of a fixed-wing UAV is part of the MPC to solve the attitude and speed control problem.

We present a NMPC that uses the Special Orthogonal Group of order Three for attitude parameterization and a dynamic model expressed in the stability and wind frame coordinates. The attitude kinematics are bilinear, and flight envelope constraints are formulated as box constraints of the state vector to limit the consequences of the non-convexity of the problem. We present a discussion on how the devised NMPC can be incorporated into a standard control architecture and evaluate the closed-loop solver runtime on a suitable embedded SBC before demonstrating superior performance compared to industry-standard PID control in flight experiments.

Contributions: The contribution of this chapter is a NMPC with direct access to the actuators that tackles the low-level control problem of tracking attitude and speed references from a higher-level guidance controller. The developed NMPC can work as a low-level controller that enforces safe operation within the flight envelope such that engineers can design the outer level guidance controllers without having to carefully consider safety constraints. Experiments show the robustness of the controller to significant inaccuracies of the employed model for the type of UAV that was used in the experiments, which speaks to the robustness of the controller and the possibility to use aerodynamic models from limited system identification efforts. This contribution is the first to include a full nonlinear dynamic model in a NMPC implemented on an onboard embedded computing platform to control the low-level dynamics of a fixed-wing UAV with direct access to the actuators instead of sending reference signals to an onboard controller, and demonstrates a successful real-time application in experiments.

Chapter 6: Coupled Nonlinear Model Predictive Control and Geometric Attitude Control

Publications:

[157] Dirk Reinhardt and Tor Arne Johansen. Nonlinear Model Predictive Control combined with Geometric Attitude and Speed Control for Fixed-Wing UAVs. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 465–475, 2021.

Topic: The geometric control law in the previous chapter includes dynamic model inversion. The assumed reference signals on angular rate and acceleration are also not provided by many outer level guidance controllers.

In this chapter, we propose the design of a NMPC that can be used to overcome the topological obstruction on the two-sphere for which we previously designed the hybrid control law, and generate the required reference signals. The NMPC includes the closed-loop dynamics of the geometric attitude controller in its process model. It uses angular acceleration as a decision variable to generate the reference signals for the geometric controller and extends the low-level controller with speed control in which the throttle set-point is set directly by the NMPC. A combination of the predictive capabilities of the NMPC with the more reactive geometric controller allows for lower update frequencies of the NMPC. We discuss the results of this approach in simulation examples.

Contributions: The contribution of this chapter is a cascaded controller that consists of the geometric attitude controller, presented in Chapter 4, and a NMPC. The NMPC exploits the tracking capabilities of the geometric attitude controller by setting suitable reference signals for angular rate and acceleration. Results from a simulation study show that the cascade with NMPC significantly improves the performance in comparison to the nominal geometric attitude controller and that is a viable alternative to the hybrid extension. However, the cost is the increased computational complexity of the NMPC which limits this controller to more powerful embedded computing platforms.

Chapter 7: Extended Aerodynamic Modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle

Publications:

[158] Dirk Reinhardt, Morten D. Pedersen, Kristoffer Gryte, and Tor Arne Johansen. A Symmetry Calibration Procedure for Sensor-to-Airframe Misalignments in Wind Tunnel Data. In *2022 Conference on Control Technologies and Applications (CCTA)*, accepted

[154] Dirk Reinhardt, Kristoffer Gryte, and Tor Arne Johansen. Modeling of the Skywalker X8 Fixed-Wing UAV: Flight Tests and System Identification. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, accepted

Topic: The experiments conducted in Chapter 5 showed that the dynamic model used in the NMPC has some significant shortcomings which can be attributed to the aerodynamic model. Improving the aerodynamic model, by using the available wind tunnel data that is based on, in combination with flight test data, is the main motivation for this chapter.

One contribution of this chapter is a calibration routine to transform datasets obtained in wind tunnel experiments to minimize asymmetries that take root in misalignments of the coordinate frame of the force and moment sensors and the body-fixed coordinate frame that describes the airframe.

The chapter is moreover a contribution that provides an improved aerodynamic model than was previously published by Gryte et al. [70]. We achieve this by modifying the model structure, using the data from the wind tunnel more cautiously, and finally, by identifying the dynamic parameters by using sequentially thresholded least squares algorithm (STLSQ) optimization proposed by Brunton et al. [16]. The result is a model that augments the identifiable model from the wind tunnel with a parsimonious damping model based on flight data. Flight experiments show that the new model is a significant improvement to the baseline model.

Contributions: The contributions of this chapter include a calibration routine that can be used to increase the symmetry of measurements of the generalized aerodynamic forces for airframes that have a plane of symmetry by design. The method is demonstrated using the available wind-tunnel data recorded for the Skywalker X8 airframe. With a slight modification to the model structure, the static model already improves the baseline model that was used at the UAV Lab. Augmentation of the static model with a damping model found in flight experiments using STLSQ optimization further improves the model quality. The resulting models are made publicly available for the research community.

Chapter 8: Direct Nonlinear Model Predictive Control for the Path-Following Control Problem

This chapter includes preliminary work and its content has not been published by the time of writing the thesis.

In contrast to the preceding chapters that are primarily concerned with low-level motion control, this chapter discusses a NMPC design for the path-following problem. The research results have not appeared at a journal or conference when submitting the thesis. The NMPC design in this chapter can track parametric curves in three dimensions. We use the framework by Faulwasser et al. [54] and design the NMPC with a timing law to simultaneously optimize actuator signals and the reference position on the path. Our controller is similar to the one that Yang et al. [194] recently published, but with the difference that we consider the full dynamic model instead of a kinematic guidance model for controller design. The approach is evaluated in numerical examples and the chapter concludes with a discussion on future work toward real-time experiments.

Other publications

I contributed to the development and experimental testing in a publication where the attitude control problem was solved using deep reinforcement learning (DRL). The results are intriguing, but are not included in this thesis. They can be found in the following paper:

[14] Eivind Bøhn, Erlend M Coates, Dirk Reinhardt, and Tor Arne Johansen. Data-Efficient Deep Reinforcement Learning for Attitude Control of Fixed-Wing UAVs: Field Experiments. *arXiv preprint arXiv:2111.04153, submitted to IEEE Transactions on Neural Networks and Learning Systems*, 2021

Chapter 2

Preliminaries

2.1 Nomenclature and Definitions

Bold font indicates a vector entity, i.e. $\mathbf{x}$, and its elements are denoted by x_i. Capital letters are used in addition to bold font to denote matrices, $\mathbf{A}$. Its element in row i and column j is denoted by $[\mathbf{A}]_{i,j}$. The set of all natural numbers is denoted by $\mathbb{N}$. The real numbers are denoted by $\mathbb{R}$. All positive real numbers and all positive real numbers including zero are denoted by $\mathbb{R}_{>0}$ and $\mathbb{R}_{\geq 0}$, respectively. The set of real-valued vectors of dimension n is denoted by $\mathbb{R}^n$ and similarly the set of real-valued matrices is denoted by $\mathbb{R}^{m \times n}$.

To make the text easier to follow, we will regularly use n_x when we refer to dimension of a vector $\mathbf{x} \in \mathbb{R}^{n_x}$. We are usually looking at column vectors when discussing elements in $\mathbb{R}^n$. The transpose of either a matrix or a vector is denoted by $\cdot^\top$. The concatenation of vectors $\mathbf{x} \in \mathbb{R}^{n_x}$ and $\mathbf{y} \in \mathbb{R}^{n_y}$ is denoted by $(\mathbf{x}; \mathbf{y}) = [\mathbf{x}^\top, \mathbf{y}^\top]^\top$. An ordered collection of objects $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ is denoted by $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$. The identity matrix of dimension n is denoted by $\mathbf{I}_n$, and the subscript is dropped the dimension should be clear from context. Similarly, the matrix of zeros with dimension m and n is denoted by $\mathbf{0}_{m \times n}$.

The Euclidean norm of a vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}$. The first time derivative of a signal $x(t)$ is denoted by $\frac{d}{dt}x = \dot{x}$. The second time-derivative is denoted by $\frac{d}{dt}\dot{x} = \ddot{x}$. The same notation applies for vector quantities. For ease of notation, we drop the time argument whenever possible. An Eigenvalue of a matrix $\mathbf{A}$ is denoted by $\lambda(\mathbf{A})$ and the minimum and maximum Eigenvalue are denoted by $\lambda_{\min}(\mathbf{A})$ and $\lambda_{\max}(\mathbf{A})$, respectively.

When discussing the proximity of points, we sometimes use the unit ball defined as $\mathbb{B} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| < 1\}$. The quadratic norm with respect to a positive definite weighting matrix $\mathbf{Q} = \mathbf{Q}^\top$ is denoted by $\|\mathbf{x}\|^2_{\mathbf{Q}} = \mathbf{x}^\top \mathbf{Q} \mathbf{x}$. Let $\mathbf{e}^n_i \in \mathbb{R}^{n \times 1}$ be a column vector with the i-th element equal to 1, but 0 anywhere else. In the special case of $n = 3$, we drop the dimension, i.e. use $\mathbf{e}_i = \mathbf{e}^n_i$. Let $\mathbf{1}_n \in \mathbb{R}^{n \times 1}$ be a column vector with 1 at every element. Matrices with m rows and n columns with all elements being zero is denoted by $\mathbf{0}_{m \times n}$. For any bounded variable, $\underline{\cdot}$ and $\bar{\cdot}$ denote lower and upper bounds, respectively.

2.1.1 Matrix and Vector Identities

Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$, their cross product can be represented as a matrix multiplication $\mathbf{x} \times \mathbf{y} = \mathbf{S}(\mathbf{x})\mathbf{y}$ where the skew-symmetric operator $\mathbf{S} : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ is defined as

$$\mathbf{S}(\mathbf{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{2.1}$$

The inner product is denoted by $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^\top \mathbf{y} \in \mathbb{R}$.

A few practical properties of the skew-symmetric operator, cross product and inner product for any $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z} \in \mathbb{R}^3$ include

$$\mathbf{S}(\mathbf{x})^\top = -\mathbf{S}(\mathbf{x}) \tag{2.2}$$

$$\mathbf{x} \cdot (\mathbf{y} \times \mathbf{z}) = \mathbf{y} \cdot (\mathbf{z} \times \mathbf{x}) = \mathbf{z} \cdot (\mathbf{x} \times \mathbf{y}) \tag{2.3}$$

$$\|\mathbf{x} \times \mathbf{y}\|^2 = \|\mathbf{x}\|\|\mathbf{y}\| - (\mathbf{x}^\top \mathbf{y})^2 \tag{2.4}$$

$$\mathbf{S}(\mathbf{x})^3 = -\|\mathbf{x}\|^2 \mathbf{S}(\mathbf{x}) \tag{2.5}$$

$$\mathbf{S}(\mathbf{x} \times \mathbf{y}) = \mathbf{S}(\mathbf{y})\mathbf{S}(\mathbf{x}) - \mathbf{S}(\mathbf{x})\mathbf{S}(\mathbf{y}), \tag{2.6}$$

and will be used in the context of geometric control in Chapter 4.

Let $\mathbf{n}$ be the unit vector that is orthogonal to the plane spanned by $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ and $\theta \in [-\pi, \pi]$ be the angle between $\mathbf{x}$ and $\mathbf{y}$ on that plane, then the following relationships are useful

$$\mathbf{x} \times \mathbf{y} = \|x\|\|y\| \sin(\theta)\mathbf{n} \tag{2.7}$$

$$\mathbf{x} \cdot \mathbf{y} = \cos(\theta). \tag{2.8}$$

2.1.2 Derivative Functions

Suppose that $f : \mathcal{X} \to \mathbb{R}$ is a scalar-valued function that is differentiable on the open set $\mathcal{X} \subset \mathbb{R}^n$. Then its gradient is denoted by $\nabla f = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^n$. The gradient at $\mathbf{x} \in \mathbb{R}^n$ in the direction of $\mathbf{y} \in \mathbb{R}^n$ is given by $\nabla f \cdot \mathbf{y} \in \mathbb{R}$, and is often referred to as the directional derivative in the literature. In cases where we are working with a scalar-valued function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that is differentiable on the open set $\mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^n \times \mathbb{R}^n$, the gradient with respect to $\mathbf{x}$ will be denoted by $\nabla_{\mathbf{x}} f = \frac{\partial f(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}} \in \mathbb{R}$.

Suppose now that $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$ is a vector-valued function instead, and it is differentiable on the open set $\mathcal{X} \subset \mathbb{R}^n$. Its derivative function is defined as

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \nabla f_1 \\ \vdots \\ \nabla f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m\times n}, \tag{2.9}$$

and is usually referred to as Jacobian matrix.

2.1.3 Manifold Concepts

In parts of this thesis we will look at control algorithms that draw upon the field of *Geometric Control.* The topic of Geometric Control itself is quite rich [19, 28, 109], but one can say that the pervasive paradigm is the goal to design control algorithms directly on the configuration manifold instead of using *local coordinates.* This allows for the design of global controllers that can handle wide range of rotational motion and rapid attitude changes. This is in contrast to controllers that are reflected in the classical PID control used for conventional aircraft autopilots which usually deal with local set-point stabilization of a given attitude reference.

In general, a differentiable manifold embedded in $\mathbb{R}^n$ is described by

$$M = \{\mathbf{x} \in \mathbb{R}^n : f_i(\mathbf{x}) = 0,\, i = 1, \ldots, l\}, \tag{2.10}$$

where the scalar-valued and differentiable functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are such that their gradients ∇f_i are linearly independent vectors in $\mathbb{R}^n$ for each $\mathbf{x} \in M$. Manifolds for which the functions f_i are globally differentiable with respect to the manifold are often referred to as smooth manifolds.

The global configuration manifold that describes the attitude of a rigid-body is the *Special Orthogonal Group of Order Three* which is defined as

$$\mathrm{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} : \mathbf{R}^\top \mathbf{R} = \mathbf{I},\, \det(R) = 1\}. \tag{2.11}$$

The elements $\mathbf{R} \in \mathrm{SO}(3)$ are referred to as a rotation matrix, and its columns describe axes of a frame of interest relative to a reference frame. The Lie Group SO(3) has an associated Lie algebra $\mathfrak{so}(3)$ defined as

$$\mathfrak{so}(3) = \{\mathbf{L}_1\omega_1 + \mathbf{L}_2\omega_2 + \mathbf{L}_3\omega_3 | \omega \in \mathbb{R}^3\}, \tag{2.12}$$

with the basis

$$\mathbf{L}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{L}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad \mathbf{L}_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.13}$$

Note the equivalence to the skew-symmetric operator which can be defined by using the basis $\mathbf{L}_i$.

Another important manifold for this thesis is the *n-Sphere* which is defined as

$$\mathbb{S}^n = \{\mathbf{x} \in \mathbb{R}^{n+1} : \|\mathbf{x}\| = 1\}. \tag{2.14}$$

In particular the *Two-Sphere* which is embedded in $\mathbb{R}^3$ and denoted by $\mathbb{S}^2$ will be used throughout this text to describe the direction in which a given axis is pointing.

The *Tangent Space* at a point $\mathbf{x} \in \mathbb{S}^2$ is defined as the set of three-dimensional vectors that are orthogonal to $\mathbf{x}$

$$\mathrm{T}_{\mathbf{x}}\mathbb{S}^2 = \{\mathbf{y} \in \mathbb{R}^3 : \mathbf{x} \cdot \mathbf{y} = 0\}. \tag{2.15}$$

It will be useful to decompose a vector $\mathbf{v} \in \mathbb{R}^3$ into an element that is in $\mathrm{T}_{\mathbf{x}}\mathbb{S}^2$ and component normal to it. For this purpose consider the *orthogonal projection operator* $\mathbf{\Pi}_{\mathbf{x}}^{\perp} : \mathbb{R}^3 \to \mathrm{T}_{\mathbf{x}}\mathbb{S}^2$ defined as [109]

$$\mathbf{\Pi}_{\mathbf{x}}^{\perp} = \mathbf{I} - \mathbf{x}\mathbf{x}^\top = -\mathbf{S}(\mathbf{x})^2, \tag{2.16}$$

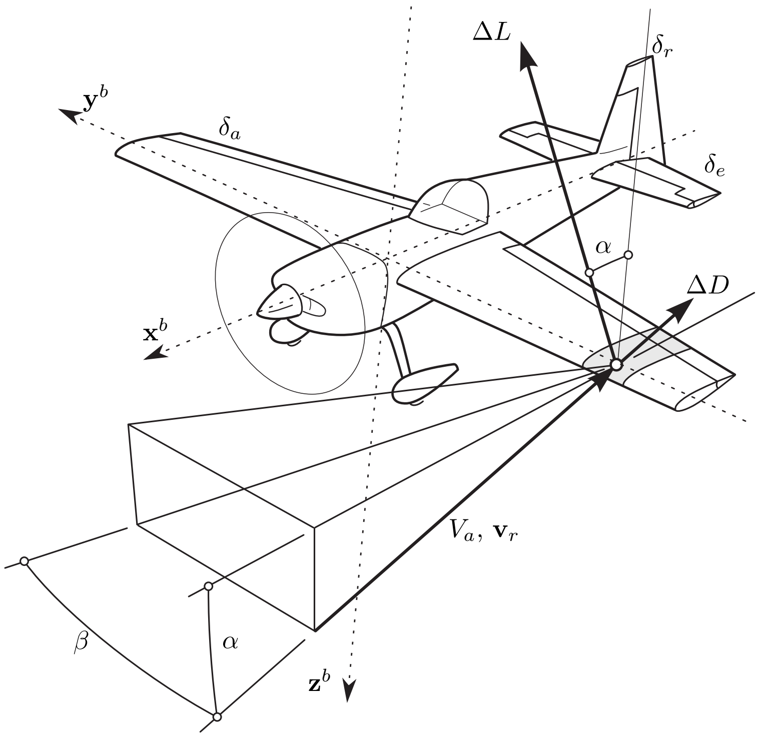

Figure 2.1: Illustration of the local flow for a strip of the wing generating lift ΔL and drag ΔD, which act along the axes $\mathbf{z}^w$ and $\mathbf{x}^w$, respectively. The body-fixed axes $(\mathbf{x}^b, \mathbf{y}^b, \mathbf{z}^b)$ are also shown. Modified based on [163].

and the decomposition

$$\mathbf{v} = \mathbf{\Pi}_{\mathbf{x}}^{\perp}\mathbf{v} + (\mathbf{I} - \mathbf{\Pi}_{\mathbf{x}}^{\perp})\mathbf{v}, \tag{2.17}$$

where the first term on the right-hand side is the part of $\mathbf{v}$ is the projection onto $\mathrm{T}_{\mathbf{x}}\mathbb{S}^2$ and the second term is the part that is orthogonal to $\mathrm{T}_{\mathbf{x}}\mathbb{S}^2$. We will sometimes refer to this space as the *normal space* and denote it as $\mathrm{N}_{\mathbf{x}}\mathbb{S}^2$. In this context we will also use the *parallel projection operator* $\mathbf{\Pi}_{\mathbf{x}}^{\parallel} : \mathbb{R}^3 \to \mathrm{N}_{\mathbf{x}}\mathbb{S}^2$ defined as

$$\mathbf{\Pi}_{\mathbf{x}}^{\parallel} = \mathbf{I} - \mathbf{\Pi}_{\mathbf{x}}^{\perp} = \mathbf{x}\mathbf{x}^{\top}. \tag{2.18}$$

2.2 Kinematics, Dynamics and Actuation

2.2.1 Coordinate Frames

Throughout the text it will often be necessary to use different coordinate frames in kinematic and dynamic equations and the associated vector or matrix variables. Let $\{a\}$, $\{b\}$, $\{c\}$ be different frames that may be moving relative to each other. Then a vector $\mathbf{x}_{bc}^{a} \in \mathbb{R}^3$ denotes a vector that is decomposed along the axes of $\{a\}$ and describes a quantity, e.g. orientation or velocity, of $\{c\}$ relative to a reference frame $\{b\}$. In some cases we leave the coordinate axes unspecified and drop the superscript, only assuming that all variables within an equation are described by the same coordinates. The exception to this rule are variables that describe a rotation. In that case there is no superscript and the axes of $\{c\}$ are described in coordinates of $\{b\}$, e.g. $\mathbf{\Theta}_{bc}$ for Euler angles.

The following coordinate frames are useful to understand and describe kinematics and dynamics of aircraft in general, but particular for small UAVs that are subject to a more pronounced impact of environmental disturbances such as wind. We give a brief presentation here, but a more elaborate discussion can be found in [9, 176, 177].

Relative velocity For an understanding of part of coordinate frame, it is important to first introduce the relative velocity vector, which describes the velocity of the UAV relative to the surrounding air mass. The surrounding air mass is moving with the local wind field. Independent of the coordinates, let the vector describing the linear velocity of the UAV with respect to the ground be denoted by $\mathbf{v}_{nb} \in \mathbb{R}^3$, and the linear velocity vector of the surrounding air mass with respect to the ground, i.e. wind in layman terms, be denoted by $\mathbf{v}_{nw} \in \mathbb{R}^3$, then the relative velocity vector is given by

$$\mathbf{v}_r = \mathbf{v}_{nb} - \mathbf{v}_{nw}. \tag{2.19}$$

This relationship is sometimes referred to as wind triangle in textbooks that explicitly deal with the flight mechanics of small UAVs (see [9]). Based on $\mathbf{v}_r$ the aerodynamic forces and moments can be defined. Other textbooks that consider more heavy passenger aircraft do not take wind into account, given that forces and moments due to the surrounding air mass are negligible and $\mathbf{v}_r \approx \mathbf{v}_{nb}$ is often a good approximation at the speed that these aircraft operate. Nonetheless, modern passenger aircraft normally come with equipment to measure the relative velocity vector. For small UAVs in contrast, an estimation algorithm usually needs to be employed (for examples see [15, 85, 189]).

NED The North-East-Down frame is a local reference frame that is practical to use for applications with UAVs limited to a small area such that the Earth can be approximated by a flat surface and a local tangent plane may be used to describe the position and orientation of the UAV. The coordinate frame is denoted by $\{n\}$ and usually has its origin and orientation defined relative to a local base-station from which the UAV is operated. Its axis $\mathbf{x}^n$ points towards the North, $\mathbf{z}^n$ points in the direction of gravity, and $\mathbf{y}^n = \mathbf{z}^n \times \mathbf{x}^n$ completes a right-handed coordinate system. For small UAVs, considering the NED frame to be the inertial frame is usually a good approximation, and we make use of it throughout this text.

BODY The body-fixed frame is denoted by $\{b\}$ and is described by the axes $(\mathbf{x}^b, \mathbf{y}^b, \mathbf{z}^b)$. It is fixed to the airframe of the vehicle. Position and attitude of the body-fixed frame are expressed relative to the NED frame, but its linear and angular velocities are often expressed along the axes of b. The axis $\mathbf{x}^b$ points forward, and we define $\mathbf{z}^b$ as a vertical axis pointing downward. The axis $\mathbf{y}^b = \mathbf{z}^b \times \mathbf{x}^b$ again completes a right-handed coordinate system.

STABILITY To generate lift, the airfoil of the UAV needs to be oriented at a positive angle with respect to the relative velocity vector that describes the linear velocity of the UAV with respect to the velocity of the surrounding air mass. This angle is referred to as angle of attack and denoted by $\alpha \in \mathbb{R}$. The angle of attack is described by a left-handed rotation along $\mathbf{y}^b$ such that the relative velocity vector projected onto the $\mathbf{x}^b - \mathbf{z}^b$ plane is aligned with $\mathbf{x}^b$. The resulting coordinate frame is denoted by $\{s\}$.

WIND Similar to the angle of attack, there is an angle that describes the rotation between the relative velocity vector and the $\mathbf{x}^b - \mathbf{z}^b$ plane. It is called the sideslip angle and denoted by β. The wind frame is rotated relative to the stability frame by a right-handed rotation along $\mathbf{z}^s$ with rotation angle $\beta \in \mathbb{R}$. The wind frame is denoted by $\{w\}$.

2.2.2 Attitude Representation and Kinematics

Rotation matrix

As discussed previously, a rotation matrix is an element of SO(3) and denoted by $\mathbf{R} \in \mathrm{SO}(3)$. We use it in this text develop the control laws, given that this eliminates the need for handling the unwinding problem of quaternions and the singularity problem of Euler angles, which we will briefly discuss in the next. Suppose the angular velocity vector with which frame $\{a\}$ rotates relative to frame $\{b\}$ is given along the axes of $\{b\}$, and it is thus denoted by $\boldsymbol{\omega}^b_{ab}$. The continuous kinematic equation of $\mathbf{R}_{ab}$ is then given as

$$\dot{\mathbf{R}}_{ab} = \mathbf{R}_{ab}\mathbf{S}(\boldsymbol{\omega}^b_{ab}). \tag{2.20}$$

Euler Angles

Representing the attitude of a rigid body in terms of *Euler Angles* is often done because they give a minimum parameterization of SO(3), and allow for an intuitive interpretation of trajectories. However, one drawback is the problem of singular configurations when two of the intermediate rotation axes coincide, commonly known as *gimbal lock*. For local control laws, the singularity can be avoided, but it can become an issue when designing global control laws that target a wide range of rotational motion.

In aerospace applications it is common practice to use the *intrinsic ZYX-convention*. This means that orientation of the body-fixed frame relative to the reference frame is described by the rotation along $\mathbf{z}^n$ by the yaw angle $\psi \in [-\pi, \pi]$. Thereafter, a rotation along the intermediate axis $\mathbf{y}'$ by the pitch angle $\theta \in [-\pi/2, \pi/2]$. And the final rotation is about the intermediate axis $\mathbf{x}''$ by the roll angle $\phi \in [-\pi, \pi]$.

A more thorough discussion of the rotation sequences, including illustrations of the intermediate frames, can be found in any textbook on rigid-body motion such as [9, 56, 119, 176, 177]. Some authors summarize the Euler sequence in one variable denoted by $\boldsymbol{\Theta} = [\phi, \theta, \psi]^\top \in [\pi, \pi] \times [-\pi/2, \pi/2] \times [-\pi, \pi]$.

This sequence of Euler angles parameterizes elements of SO(3) by

$$\begin{aligned}\mathbf{R}(\mathbf{\Theta}) &= \begin{bmatrix}\cos(\psi) & -\sin(\psi) & 0\\ \sin(\psi) & \cos(\psi) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos(\theta) & 0 & \sin(\theta)\\ 0 & 1 & 0\\ -\sin(\theta) & 0 & \cos(\theta)\end{bmatrix}\begin{bmatrix}\cos(\phi) & \sin(\phi) & 0\\ -\sin(\phi) & \cos(\phi) & 0\\ 0 & 0 & 1\end{bmatrix}\\ &= \begin{bmatrix}\mathrm{c}\psi\mathrm{c}\theta & -\mathrm{s}\psi\mathrm{c}\phi+\mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi+\mathrm{c}\psi\mathrm{c}\phi\mathrm{s}\theta\\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi+\mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{c}\psi\mathrm{c}\phi+\mathrm{s}\theta\mathrm{s}\psi\mathrm{c}\phi\\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi\end{bmatrix}, \end{aligned} \tag{2.21}$$

where we used the shortened notation s := sin and c := cos. We will normally use $\mathbf{\Theta}$ or its associated rotation matrix to describe the orientation of the axes of the body-fixed frame $\{b\}$ with respect to the NED frame $\{n\}$. Thus, the Euler angles will be denoted by $\mathbf{\Theta}_{nb}$ and a vector $\mathbf{v}^b$ can be transformed to $\mathbf{v}^n$ and vice versa by

$$\mathbf{v}^n = \mathbf{R}(\mathbf{\Theta}_{nb})\mathbf{v}^b, \qquad \mathbf{v}^b = \mathbf{R}(\mathbf{\Theta}_{nb})^\top\mathbf{v}^n. \tag{2.22}$$

A useful property of $\mathbf{R}(\mathbf{\Theta})$ is that it gives a direct relationship between the axes of the body-fixed frame and the axes of the NED frame by means of Euler angles. Upon inspection of Eq. (2.21), note that this can be useful for designing a low-level controller for roll and pitch that operates directly on $\mathbb{S}^2$ by extracting the vector $\mathbf{R}(\mathbf{\Theta}_{nb})^\top\mathbf{e}_3$.[1]

The kinematic equation for the Euler angles is

$$\dot{\mathbf{\Theta}}_{nb} = \mathbf{T}(\mathbf{\Theta}_{nb})\boldsymbol{\omega}^b_{nb} \tag{2.23}$$

with

$$\mathbf{T}(\mathbf{\Theta}_{nb}) = \begin{bmatrix}1 & \sin(\varphi)\tan(\theta) & \cos(\varphi)\tan(\theta)\\ 0 & \cos(\varphi) & -\sin(\varphi)\\ 0 & \sin(\varphi)/\cos(\theta) & \cos(\varphi)/\cos(\theta)\end{bmatrix}. \tag{2.24}$$

Details can be found in standard textbooks [53, 56, 119, 177]. Note that using Euler angles comes with the drawback of singular attitudes (gimbal lock).

Quaternions

Unit quaternions, commonly denoted by $\mathbf{q} \in \mathbb{S}^3$, can be decomposed into $\mathbf{q} = [s, \mathbf{r}]^\top$ where s denotes as scalar part and $\mathbf{r} \in \mathbb{R}^3$ denotes a vector part. The interpretation of unit quaternions is less intuitive compared to Euler angles, but they are numerically more robust when used in integration schemes, which is the reason why they are often used in numerical simulations. They also allow for attitude interpolation.

An issue that needs to be considered when using unit quaternions for attitude control is that $\mathbb{S}^3$ is a double cover of SO(3) which means that the same attitude can be described by two equivalent quaternions[2]. When this is not taken into account in the design of the control law, a rigid body that is initially close to the reference attitude can be caused to perform a large rotational maneuver before eventually

[1]This will be central in the context of geometric attitude control in Chapter 4.

[2]The original unit quaternion and its conjugate [119] describe the same attitude

being stabilized to the desired values. This problem is known as the *Unwinding Phenomenon* [28] and some authors tackle this by stabilizing both the original reference quaternion and its conjugate, given that unit quaternions allow for an otherwise more intuitive design of *Control Lyapunov Functions*.

In this text we only use unit quaternions for simulation purposes so that it is enough to be familiar with the kinematic equations

$$\dot{\mathbf{q}}_{nb} = \frac{1}{2}\mathbf{T}(\mathbf{q}_{nb})\boldsymbol{\omega}^b_{nb}, \qquad \text{with } \mathbf{T}(\mathbf{q}_{nb}) = \begin{bmatrix} -\mathbf{r} \\ s\mathbf{I} + \mathbf{S}(\mathbf{r}) \end{bmatrix}, \tag{2.25}$$

For an exponential map from unit quaternions to the corresponding rotation matrix one can use the *quaternion to rotation matrix formula* [172]

$$\mathbf{R}(\mathbf{q}) = (s^2 - \mathbf{r}^\top\mathbf{r})\mathbf{I} + 2\mathbf{r}\mathbf{r}^\top + 2s\mathbf{S}(\mathbf{r}). \tag{2.26}$$

Rotations between BODY, STABILITY and WIND

The angle of attack $\alpha \in [-\pi/2,\, \pi/2]$ and sideslip angle $\beta \in [-\pi,\, \pi]$ can be obtained from the relative velocity vector $\mathbf{v}_r = [u_r\, v_r\, w_r]^\top$ by

$$\alpha = \arctan\left(\frac{w_r}{u_r}\right), \qquad \beta = \arcsin\left(\frac{v_r}{V_a}\right). \tag{2.27}$$

Vectors in the frames $\{b\}$, $\{s\}$, $\{w\}$ can then be transformed via the rotation matrices $\mathbf{R}_{bs}, \mathbf{R}_{sw} \in \mathrm{SO}(3)$ defined as

$$\mathbf{R}_{bs}(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}, \quad \mathbf{R}_{sw}(\beta) = \begin{bmatrix} \cos(\beta) & sin(\beta) & 0 \\ -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.28}$$

The magnitude of the relative velocity vector $\mathbf{v}_r$ is often referred to as *airspeed* and denoted by $V_a = \|\mathbf{v}_r\| \in \mathbb{R}_{\geq 0}$.

Velocities

The velocity relative to the surrounding air is

$$\mathbf{v}^b_r = \mathbf{R}_{bs}\mathbf{R}_{sw}\mathbf{e}_1 V_a = \begin{bmatrix} \cos(\alpha)\cos(\beta) \\ \sin(\beta) \\ \sin(\alpha)\cos(\beta) \end{bmatrix} V_a \tag{2.29}$$

The direction of travel of an aircraft is often expressed in terms of course angle $\chi \in [-\pi,\, \pi]$ and flight-path angle $\gamma \in [-\pi/2, \pi/2]$, a representation which is independent of speed. The course angle describes the angle between the axis $\mathbf{x}^n$ and the ground velocity vector of the aircraft projected onto the North-East plane. It can thus be determined from a velocity estimate $\mathbf{v}^n_{nb} = \begin{bmatrix} v^n_{\text{north}} & v^n_{\text{east}} & v^n_{\text{down}} \end{bmatrix}$ as

$$\chi = \arctan\left(\frac{v^n_{\text{north}}}{v^n_{\text{east}}}\right) \tag{2.30}$$

The flight-path angle characterizes the climb rate of the aircraft and can be determined as [9]

$$\dot{h} = -v^n_{\text{down}} = \|\mathbf{v}^n\| \sin(\gamma) \Leftrightarrow \gamma = \arcsin\left(\frac{-v^n_{\text{down}}}{\|\mathbf{v}^n\|}\right). \tag{2.31}$$

2.2.3 Dynamics of a fixed-wing Unmanned Aerial Vehicle

The kinematic and dynamic equations that are commonly used to describe the state of a single fixed-wing UAV can be derived by applying Newton's second law [9] and are given as:

$$\begin{aligned}
\dot{\mathbf{p}}^n_{nb} &= \mathbf{R}_{nb}\mathbf{v}^b_{nb} && (2.32\text{a})\\
\dot{\mathbf{v}}^b_{nb} &= \frac{1}{\mathrm{m}}(\mathbf{R}^\top_{wb}\mathbf{f}^w_a + \mathbf{f}^b_t) + \mathbf{R}^\top_{nb}\mathbf{g}^n - \boldsymbol{\omega}^b_{nb} \times \mathbf{v}^b_{nb} && (2.32\text{b})\\
\dot{\mathbf{R}}_{nb} &= \mathbf{R}_{nb}\mathbf{S}(\boldsymbol{\omega}^b_{nb}) && (2.32\text{c})\\
\mathbf{J}\dot{\boldsymbol{\omega}}^b_{nb} &= \mathbf{S}(\mathbf{J}\boldsymbol{\omega}^b_{nb})\boldsymbol{\omega}^b_{nb} + \mathbf{m}^b, && (2.32\text{d})
\end{aligned}$$

where $\mathbf{p}^n_{nb} \in \mathbb{R}^3$ denotes the position of the UAV. The angular velocity is denoted by $\boldsymbol{\omega}^b_{nb} \in \mathbb{R}^3$, $\mathbf{J} = \mathbf{J}^\top \in \mathbb{R}^{3\times3}$ denotes the inertia matrix and $\mathbf{g}^n = [0 \;\; 0 \;\; g]^\top$ denotes the acceleration due to gravity. The mass of the UAV is denoted by $\mathrm{m} \in \mathbb{R}$. The external forces due to aerodynamics and thrust and the external torque are denoted by $\mathbf{f}^w_a, \mathbf{f}^b_t, \mathbf{m}^b \in \mathbb{R}^3$. They are functions of the relative velocities and actuators, which will be discussed in a later section. The inertia matrix is given as

$$\mathbf{J} = \begin{bmatrix} J_{xx} & 0 & -J_{xz} \\ 0 & J_{yy} & 0 \\ -J_{xz} & 0 & J_{zz} \end{bmatrix}. \tag{2.33}$$

An alternative dynamic model that is often used for stability analysis with respect to the aerodynamic quantities (V_a, β, α) can be found in [177]. We will make use of it in the context of MPC, so it is worth to present it here. Let $\boldsymbol{\omega}^s_{nb}$ denote the angular velocity vector decomposed in $\{s\}$. The dynamic equations are then given as:

$$\begin{bmatrix} \dot{V}_a \\ \dot{\beta} V_a \\ \dot{\alpha} V_a \cos\beta \end{bmatrix} = \frac{1}{\mathrm{m}}(\mathbf{f}^w_a + \mathbf{R}_{wb}\mathbf{f}^b_t) + \mathbf{R}_{wb}\mathbf{R}^\top_{nb}\mathbf{g}^n - \boldsymbol{\omega}^w_{nb} \times \mathbf{v}^w_r \tag{2.34}$$

$$\mathbf{J}^s\dot{\boldsymbol{\omega}}^s_{nb} = \mathbf{S}(\mathbf{J}^s\boldsymbol{\omega}^s_{nb})\boldsymbol{\omega}^s_{nb} + \mathbf{R}_{sb}\mathbf{m}^b - \mathbf{J}^s(\boldsymbol{\omega}^s_{bs} \times \boldsymbol{\omega}^s_{nb}), \tag{2.35}$$

where $\mathbf{J}^s$ is given by the similarity transformation $\mathbf{J}^s = \mathbf{R}_{sb}\mathbf{J}^b\mathbf{R}^\top_{sb}$. The angular velocity $\boldsymbol{\omega}^s_{bs}$ and the relative velocity vector $\mathbf{v}^w_r$ are given as:

$$\boldsymbol{\omega}^s_{bs} = \begin{bmatrix} 0 & \dot{\alpha} & 0 \end{bmatrix}^\top, \quad \mathbf{v}^w_r = \begin{bmatrix} V_a & 0 & 0 \end{bmatrix}^\top. \tag{2.36}$$

For a detailed derivation of the model in a slightly different form, see [177].

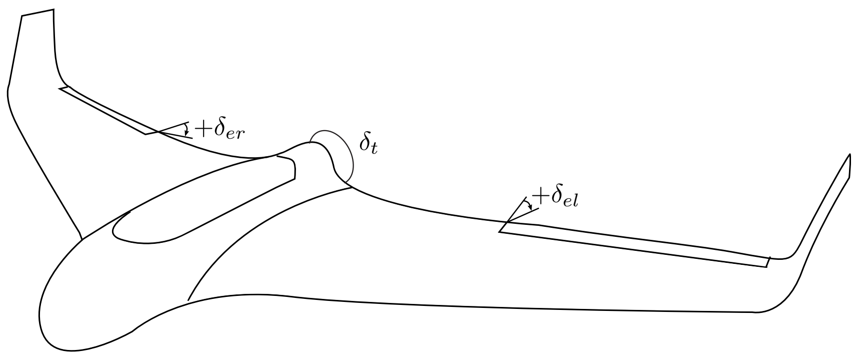

Figure 2.2: The Skywalker X8 as one example of a flying-wing UAV. Moving the elevons together or differentially corresponds to a virtual elevator or aileron deflection, respectively. The throttle set-point determines the propeller speed.

2.2.4 Actuation

A standard set of actuators includes the control surfaces referred to as aileron, elevator and rudder, with their deflections denoted by δ_a, δ_e, $\delta_r \in \mathbb{R}$. They are illustrated in Fig. 2.1 and actuated to generate a moment around their respective body-fixed axes. The aileron is a composite of two control surfaces on each side of the aircraft and can be collected in one variable as

$$\delta_a = \frac{1}{2}(\delta_{a,\text{left}} - \delta_{a,\text{right}}).$$

Its primary use is to induce a roll moment around $\mathbf{x}^b$. The elevator and rudder usually consist of one single control surface, and they primarily act around $\mathbf{y}^b$ and $\mathbf{z}^b$, respectively.

Remark 2.1. In addition to their primary axes, the aileron and elevator induce a minor moment around the respective axes of the other actuator. This cross effect is considered as a disturbance in classical control design, but may be exploited when using more advanced control methods such as model predictive control. We will have more to say about this in Chapter 5.

One type of UAV that is more compact due to a missing tail and often comes with a less expensive airframe is the *flying wing*, which does have a throttle and two control surfaces on each side of the UAV, see Fig. 2.2. These surfaces are referred to as *elevons*, and they are moved differentially to induce a roll moment and in the same direction to induce a pitch moment. Thus, there is a linear map between left and right elevons denoted by $\delta_{el}, \delta_{er} \in \mathbb{R}$ and δ_a, δ_a which we define as:

$$\begin{bmatrix}\delta_{el}\\ \delta_{er}\end{bmatrix} = \begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}\begin{bmatrix}\delta_a\\ \delta_e\end{bmatrix}. \tag{2.37}$$

In classic flight mechanics, the control surfaces deflect an air-stream which induces a dynamic pressure that is determined by the air density and speed to

$$\bar{q} = \frac{1}{2}\rho V_a^2 \in \mathbb{R}_{\geq 0}.$$

The product of dynamic pressure and the surface area then results in a force vector that has a component orthogonal to the primary axis around which the control surface actuates. The control moment is then the result of the product of the force component and the distance to the respective body-fixed axis. A large enough airspeed is thus critical to ensure sufficient control authority of the surface actuators. This fact can be taken into consideration when designing control schemes to desaturate the control surfaces by increasing airspeed, thus making them more effective [145]. However, for our control design purposes, we use the following assumption

Assumption 1. The airspeed is greater or equal to a minimum airspeed denoted by $\underline{V}_a$, i.e. $V_a \geq \underline{V}_a$.

Note that this assumption excludes take-off maneuvers during which the airspeed may theoretically be zero.

The propeller is the primary actuator to add energy to the system. It is controlled through the throttle $\delta_t \in [0, 1]$ which is usually mapped to a PWM signal that is within the engine limits. The resulting thrust force $\mathbf{f}_t^b$ that can be generated by the spinning propeller depends on the airspeed. We assume that the propeller is mounted such that the force acts along the longitudinal axis

Assumption 2. The thrust force vector $\mathbf{f}_t^b$ is aligned with the longitudinal axis $\mathbf{x}^b$.

And we impose an assumption on the wind conditions:

Assumption 3. The local wind field has a negligible angular velocity component, i.e. $\boldsymbol{\omega}_{nb}^b \approx \boldsymbol{\omega}_r^b$.

2.2.5 Model of the Generalized Forces

As it is often done in the literature we model the aerodynamic force vector in wind frame coordinates and express it as a function of the relative velocities and control inputs, resulting in the forces referred to as drag, crosswind and lift, which are denoted by D, C and L, respectively. They are obtained as a product of the aerodynamic pressure with the wing surface area, denoted by $S \in \mathbb{R}$ and a dimensionless aerodynamic coefficient. The result can be expressed as

$$\begin{bmatrix} D \\ C \\ L \end{bmatrix} = \mathbf{f}_a^w(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) = \bar{q} S_{\text{wing}} \begin{bmatrix} C_D(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) \\ C_C(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) \\ C_L(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) \end{bmatrix} \tag{2.38}$$

with the structure of the aerodynamic coefficients as identified in [71]

$$C_D = C_{D_0} + C_{D_{\alpha 1}} \alpha + C_{D_{\alpha 2}} \alpha^2 + C_{D_\beta} \beta + C_{D_{\beta 2}} \beta^2 + C_{D_q} \frac{c}{2V_a} q + C_{D_{\delta_e}} \delta_e^2, \tag{2.39}$$

$$C_C = C_{Y_0} + C_{Y_\beta} \beta + C_{Y_p} \frac{b}{2V_a} p + C_{Y_r} \frac{b}{2V_a} r + C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_r}} \delta_r, \tag{2.40}$$

$$C_L = C_{L_0} + C_{L_\alpha} \alpha + C_{L_q} \frac{c}{2V_a} q + C_{L_{\delta_e}} \delta_e. \tag{2.41}$$

The moment vector $\mathbf{m}^b$ is a function of the same variables as $\mathbf{f}_a^w$, but is modelled in the body-fixed frame

$$\begin{bmatrix} l \\ m \\ n \end{bmatrix} = \mathbf{m}^b(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) = \bar{q} S_{\text{wing}} \begin{bmatrix} bC_l(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) \\ cC_m(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) \\ bC_n(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) \end{bmatrix}, \tag{2.42}$$

and its aerodynamic coefficients have a similar structure compared to the force coefficients

$$C_l = C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r, \tag{2.43}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e, \tag{2.44}$$

$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r. \tag{2.45}$$

The structure of the aerodynamic coefficients is for the most part linear in the state variables, which is the established way of modelling the aerodynamics in the aerospace literature [9, 176, 177] where models are commonly expressed as linear perturbation models at trim conditions around state vectors in the regime that the aircraft operates. A model that consists of higher order polynomials to capture the aerodynamics globally, i.e. for all possible relative velocities in the flight envelope, is presented in [65] for passenger aircraft. Pioneering work for global models of agile fixed-wing UAVs with high thrust-to-weight ratio is done in [163]. Our motivation to use the model structure as outlined here is that wind-tunnel data for the Skywalker X8 and the identified model is available at our lab thanks to the work of Gryte et al. [71]. We do however discuss preliminary work based on a modern machine-learning approach [16] that allows to identify the dynamic model based on flight data, which requires less effort than classical approaches [82]. The results of the machine-learning approach and improvements to the model of Gryte et al. are the subject of Chapter 7.

The generalized forces can be decomposed into an aerodynamic flow term $\mathbf{f}(\mathbf{v}_r)$, a damping term $\mathbf{D}(\mathbf{v}_r, \boldsymbol{\omega}_r)$ and a control-effectiveness term $\mathbf{G}(\mathbf{v}_r)$. The resulting moment can then be expressed as

$$\mathbf{m}^b(\mathbf{v}_r, \boldsymbol{\omega}_r, \boldsymbol{\delta}) = \mathbf{f}(\mathbf{v}_r) + \mathbf{D}(\mathbf{v}_r, \boldsymbol{\omega}_r) + \mathbf{G}(\mathbf{v}_r)\begin{bmatrix}\delta_a & \delta_e & \delta_r\end{bmatrix}^\top \tag{2.46}$$

with the aerodynamic flow term as

$$\mathbf{f}(\mathbf{v}_r) = \bar{q} S_{\text{wing}} \begin{bmatrix} b(C_{l_0} + C_{l_\beta}\beta) \\ c(C_{m_0} + C_{m_\alpha}\alpha) \\ b(C_{n_0} + C_{n_\beta}\beta) \end{bmatrix}, \tag{2.47}$$

the damping matrix

$$\mathbf{D}(\mathbf{v}_r, \boldsymbol{\omega}_r) = \bar{q} S_{\text{wing}} \begin{bmatrix} \frac{b^2}{2V_a}C_{l_p} & 0 & \frac{b^2}{2V_a}C_{l_r} \\ 0 & \frac{c^2}{2V_a}C_{m_q} & 0 \\ \frac{c^2}{2V_a}C_{n_p} & 0 & \frac{c^2}{2V_a}C_{n_r} \end{bmatrix}, \tag{2.48}$$

and the control-effectiveness matrix

$$\mathbf{G}(\mathbf{v}_r) = \bar{q}S_{\text{wing}} \begin{bmatrix} bC_{l_{\delta_a}} & 0 & bC_{n_{\delta_r}} \\ 0 & cC_{m_{\delta_e}} & 0 \\ bC_{n_{\delta_a}} & 0 & bC_{n_{\delta_r}} \end{bmatrix}. \tag{2.49}$$

The matrix $\mathbf{G}(\mathbf{v}_r)$ has full rank for all $V_a > 0$ and $C_{m_{\delta_e}}(C_{l_{\delta_a}}C_{l_{\delta_a}} - C_{l_{\delta_a}}C_{l_{\delta_a}}) \neq 0$ as noted in [145].

The angular acceleration can now be expressed as

$$\mathbf{J}\dot{\boldsymbol{\omega}}^b_{nb} = \mathbf{S}(\mathbf{J}\boldsymbol{\omega}^b_{nb})\boldsymbol{\omega}^b_{nb} + \mathbf{f}(\mathbf{v}_r) + \mathbf{D}(\mathbf{v_r})\boldsymbol{\omega}^b_{nb} + \mathbf{G}(\mathbf{v}_r)\begin{bmatrix}\delta_a & \delta_e & \delta_r\end{bmatrix}^\top. \tag{2.50}$$

This representation is useful to design controllers that include feedback linearization/dynamic inversion, as it allows us to explicitly remove the aerodynamic flow term, the damping term or parts of it from the dynamics. In the case of a flying-wing configuration that that does include a rudder, the third column is removed from $\mathbf{G}(\mathbf{v}_r)$, making it rank-deficient, which shows that this type of UAV is underactuated [179].

Another representation of the rotational dynamics is in the control-affine form in which the right-hand side of the dynamic equation is split into a drift-term, independent of the control variables, and a control-affine term

$$\mathbf{J}\dot{\boldsymbol{\omega}}^b_{nb} = \mathbf{f}(\mathbf{v}_r, \boldsymbol{\omega}^b_{nb}) + \mathbf{G}(\mathbf{v}_r)\begin{bmatrix}\delta_a & \delta_e & \delta_r\end{bmatrix}^\top, \tag{2.51}$$

with $\mathbf{f}(\mathbf{v}_r, \boldsymbol{\omega}^b_{nb}) = \mathbf{S}(\mathbf{J}\boldsymbol{\omega}^b_{nb})\boldsymbol{\omega}^b_{nb} + \mathbf{f}(\mathbf{v}_r) + \mathbf{D}(\mathbf{v_r})\boldsymbol{\omega}^b_{nb}$.

Note that the moment vector is purely based on the relative velocities and control surface deflections. Depending on the type and mounting of the propeller, the UAV may actually experience a moment coming from the propulsion system, suggesting that the throttle set-point δ_t should be included in the moment vector. However, we consider this effect to be difficult to model and also small enough to be compensated for through integral action in the controller design [9].

2.3 Assumptions on Feedback Signals

We assume that the UAV comes with a standard sensor suite that consists of a GNSS receiver to measure position and at least one inertial measurement unit (IMU) to measure linear acceleration and angular velocities. This allows for state-of-the art *Inertial Navigation Systems* (INS) to be employed, either in the form of nonlinear observers [75, 114] or the more prominent EKF [53, 74, 113]. In addition to the sensors for the INS, assume that airspeed measurements are obtained via a *pitot tube* to facilitate the estimation of the local wind field, for example by using the estimators presented in [85, 190]. Based on the available sensor suite, we make the following assumptions regarding the feedback signals:

Assumption 4. The set of available feedback signals includes position estimates, $\mathbf{p}^n_{nb}(t) = \hat{\mathbf{p}}^n_{nb}(t)$

Assumption 5. The set of available feedback signals includes attitude estimates, $\mathbf{R}_{nb}(t) = \hat{\mathbf{R}}_{nb}(t)$

Assumption 6. The set of available feedback signals includes linear velocity estimates, $\mathbf{v}^b_{nb}(t) = \hat{\mathbf{v}}^b_{nb}(t)$

Assumption 7. The set of available feedback signals includes angular velocity estimates, $\boldsymbol{\omega}^b_{nb}(t) = \hat{\boldsymbol{\omega}}^b_{nb}(t)$

Assumption 8. The set of available feedback signals includes linear wind velocity estimates, $\mathbf{v}^n_{nw}(t) = \hat{\mathbf{v}}^n_{nw}(t)$

The controller designs in this thesis are therefore based on the assumption of full state feedback. However, there are still uncertainties that need to be dealt with, such as estimation errors and model mismatch.

2.4 Motivational Example for Nonlinear Control

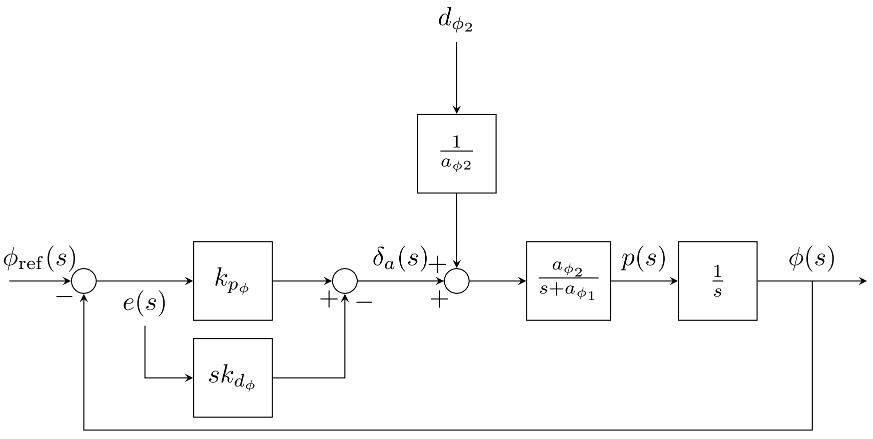

Figure 2.3: Roll attitude control loops.

In this section we briefly discuss the involved dynamics that lead to the design of standard PID controllers for the low-level dynamics based on the example of the roll attitude loop as presented by Beard et al. [9]. The aim is to motivate the use of more advanced nonlinear MIMO controllers.

Consider the kinematic equation Eq. (2.23) and Eq. (2.24), which result in the first and second time-derivative of the roll angle as

$$\dot{\phi} = p + q \sin\phi \tan\theta + r \cos\phi \tan\theta = p + d_{\phi_1} \tag{2.52}$$

and

$$\ddot{\phi} = \dot{p} + \dot{d}_{\phi_1}. \tag{2.53}$$

The angular acceleration $\dot{p}$ can be determined from Eq. (2.32d) as

$$\dot{p} = c_1 pq + c_2 qr + \frac{1}{2}\rho V_a^2 Sb(c_3 C_l(\mathbf{v}_r, \boldsymbol{\omega}, \boldsymbol{\delta}) + c_4 C_n(\mathbf{v}_r, \boldsymbol{\omega}, \boldsymbol{\delta})) \tag{2.54}$$

with

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \frac{1}{J_{xx}J_{zz} - J_{xz}^2} \begin{bmatrix} J_{xz}(J_{xx} - J_{yy} + J_{zz}) \\ J_{zz}(J_{zz} - J_{yy}) + J_{xz}^2 \\ J_{zz} \\ J_{xz} \end{bmatrix}. \tag{2.55}$$

For the design of a SISO controller that steers the aileron δ_a based on roll feedback, Eq. (2.52) - Eq. (2.55) can be summarized in terms of state, control input and disturbance to

$$\ddot{\phi} = a_{\phi 1}\dot{\phi} + a_{\phi 2}\delta_a + d_{\phi 2} \tag{2.56}$$

with

$$a_{\phi 1} = -\frac{1}{2}\rho V_a S b^2 (c_3 C_{l_p} + c_4 C_{n_p}) \tag{2.57}$$

$$a_{\phi 2} = \frac{1}{2}\rho V_a^2 S b (c_3 C_{l_{\delta_a}} + c_4 C_{n_{\delta_a}}) \tag{2.58}$$

$$d_{\phi 2} = c_1 pq + c_2 qr + \frac{1}{2}\rho V_a^2 S b (c_3 C_l(\mathbf{v}_r, \boldsymbol{\omega}, \boldsymbol{\delta}) + c_4 C_n(\mathbf{v}_r, \boldsymbol{\omega}, \boldsymbol{\delta}) + \frac{b}{2V_a} d_{\phi 1}) + \dot{d}_{\phi 1}. \tag{2.59}$$

A PD controller can then be designed such that the closed-loop dynamics of the roll angle look as in Fig. 2.3. It may be desirable to use an additional integral term in the controller to compensate for a steady-state offset caused by the disturbance [9]. This would however decrease the bandwidth of the closed-loop roll dynamics and is also not the focus of our current discussion. The closed-loop transfer function from $\phi_{\text{ref}}(s)$ to $\phi(s)$ can be determined from Fig. 2.3 as

$$\frac{\phi(s)}{\phi_{\text{ref}}(s)} = \frac{k_{p_\phi} a_{\phi_2}}{s^2 + (a_{\phi_1} + a_{\phi_2} k_{d_\phi})s + k_{p_\phi} a_{\phi_2}} \tag{2.60}$$

and can be represented as a canonical second-order transfer function

$$\frac{\phi(s)}{\phi_{\text{ref}}(s)} = \frac{\omega_{n_\phi}^2}{s^2 + 2\zeta\omega_{n_\phi} s + \omega_{n_\phi}^2} \tag{2.61}$$

with $2\zeta\omega_{n_\phi} = (a_{\phi_1} + a_{\phi_2} k_{d_\phi})$ and $\omega_{n_\phi}^2 = k_{p_\phi} a_{\phi_2}$. In a situation where a parameterization of the outlined model of the vehicle is known, the parameters a_{ϕ_i} can be computed and additional airspeed scaling in the control law can be used such that the controller parameters k_{p_ϕ}, k_{d_ϕ} can be determined based on the desired natural frequency ω_{n_ϕ} and damping ratio ζ_ϕ. This approach is widely known as *pole placement* in linear control theory. In practical controller design of UAVs, where the parameters are often unknown or identified with a high degree of uncertainty, it is common to tune the controller parameters directly.

The aim of this discussion is to illustrate, based on the example of the roll channel, how state variables that are not controlled by the roll controller enter the closed loop as shown in the expression in Eq. (2.52) and Eq. (2.59).

Longitudinal variables such as pitch angle θ and pitch rate q, controlled in a separate SISO control loop, are part of the disturbances d_{ϕ_1}, d_{ϕ_2}, and rudder deflections are implicit in Eq. (2.59). Following similar manipulations of the kinematic and dynamic equations in the longitudinal direction, it can be shown that the roll control loop introduces disturbances to the closed-loop pitch dynamics. This coupling between lateral and longitudinal plane can be assumed small in a limited flight envelope when non-aggressive maneuvers are considered. For example in a maneuver where the UAV is following an orbital path with negligible longitudinal motion, we can approximate $\theta \approx q \approx \alpha \approx \beta \approx 0$ which leads to $d_{\phi_1} \approx d_{\phi_2} \approx 0$, and the closed-loop dynamics are well approximated by Eq. (2.60) and Eq. (2.61). However, in maneuvers where both lateral and longitudinal dynamics are significant, this approximation becomes increasingly invalid and the SISO control loops introduce disturbances to each other.

Remark 2.2. The example given here includes the independent PD control of each Euler angle. This is to have a clear presentation of the problem of independent control loops running in parallel to control low-level dynamics. Note however that more general attitude control approaches exist where quaternion representation or rotation matrices are used as attitude representation. We will look at one such approach in Chapter 4

The aim of this thesis is to explore how more advanced control methods can be applied to design the low-level autopilot to extend the flight envelope to more aggressive maneuvers in more extreme conditions, in which the terms included in the disturbance here have increasing impact.

In Chapter 4 we will look at geometric attitude control (GAC) where roll and pitch dynamics are controlled in one MIMO controller such that disturbances introduced through the outlined cross-coupling are avoided. A suitable feedforward term based on the model of the UAV will compensate for other disturbing factors.

A more integrated approach that considers simultaneous control of the airspeed dynamics in addition to roll and pitch will be the focus of Chapter 5, where all available actuators will be directly set based on the output of a NMPC.

2.5 Benchmark Scenario

Throughout the thesis, we examine the developed controllers with respect to a benchmark that consists of individual step responses and a path-following scenario where the reference signals to the low-level controllers are generated by a guidance controller. The UAV for the benchmark is the Skywalker X8, which also serves as our test vehicle in experiments.

2.5.1 Benchmark Controllers for Low-Level Motion Control

We examine the performance of the controllers that are the results of the following chapters in comparison to a simple set of SISO PID controllers and the ArduPlane controller. The simple PID controllers follow the design from the discussion in Section 2.4, with additional integral terms for offset-free control. The reference signals for roll, pitch and airspeed, are denoted by $\phi_{\mathrm{ref}}, \theta_{\mathrm{ref}}$ and $V_{a,\mathrm{ref}}$, respectively.

For the simple PID design, the actuator signals are then determined based on the control laws

$$\delta_a = k_{p,\phi} \left(\phi_{\text{ref}} - \phi\right) + k_{i,\phi} \int_0^t \left(\phi_{\text{ref}} - \phi\right) d\tau - k_{d,\phi} p, \tag{2.62a}$$

$$\delta_e = -k_{p,\theta} \left(\theta_{\text{ref}} - \theta\right) - k_{i,\theta} \int_0^t \left(\phi_{\text{ref}} - \phi\right) d\tau - k_{d,\theta} q, \tag{2.62b}$$

$$\delta_t = k_{p,V_a} (V_{a,\text{ref}} - V_a) + k_{i,V_a} \int_0^t (V_{a,\text{ref}} - V_a) d\tau. \tag{2.62c}$$

Whenever we compare the controllers in the benchmark, we will refer to this controller as PID.

A slightly more sophisticated controller design for low-level motion control is the ArduPlane controller. The ArduPlane attitude controller implements separate, cascaded SISO feedback loops for the roll and pitch channels to control aileron and elevator, respectively. The control laws are based on Release 4.0.9, which is the most recent stable release (as of August 2021). The outer loop consists of proportional controllers, where desired roll and pitch rates $p_r, q_r \in \mathbb{R}$ are calculated according to

$$p_{\text{ref}} = k_\phi \left(\phi_{\text{ref}} - \phi\right) \tag{2.63a}$$

$$q_{\text{ref}} = k_\theta \left(\theta_{\text{ref}} - \theta\right) + q_{ct}, \tag{2.63b}$$

where $k_\phi, k_\theta > 0$ and q_{ct} is the pitch rate offset needed to maintain height in a coordinated turn, given by

$$q_{ct} = \sin(\phi)\cos(\theta)\frac{g}{V_a}\tan(\phi). \tag{2.64}$$

The rate set-points are inputs to the inner loop, which consists of proportional-integral (PI) controllers with feedforward action:

$$\delta_a = k_{p,p}\nu^2 \left(p_{\text{ref}} - p\right) + k_{i,p}\nu^2 \int_0^t \left(p_{\text{ref}} - p\right) d\tau + k_{ff,p}\nu p_{\text{ref}} \tag{2.65a}$$

$$\delta_e = -k_{p,q}\nu^2 \left(q_{\text{ref}} - q\right) - k_{i,q}\nu^2 \int_0^t \left(q_{\text{ref}} - q\right) d\tau - k_{ff,q}\nu q_{\text{ref}}, \tag{2.65b}$$

where $k_{p,*}$, $k_{k_i,*}$ and $k_{ff,*}$ are proportional, integral and feedforward gains, respectively. The variable $\nu = V^*/V_a$, where V^* is some constant reference airspeed, provides airspeed scaling of the controller parameters, accounting for the fact that larger airspeeds give greater aerodynamic control authority. The negative sign in the control law for δ_e is introduced to account for the convention that positive elevator deflections yield a negative pitch moment [9].

For UAVs equipped with a rudder, additional control loops utilize the extra control surface for turn coordination. However, as the Skywalker X8 considered in this paper is rudderless, this part of the control algorithm is not relevant here.

The ArduPlane autopilot controls altitude and airspeed simultaneously using a total energy control system (TECS) [101], where throttle and desired pitch angle are the control variables. Since altitude control is not part of the low-level motion controller and thus out of scope for most chapters, we rather use the simple PI control law Eq. (2.62c) to control airspeed. Whenever we compare the controllers in the benchmark, we will refer to the ArduPlane controller as AP.

2.5.2 Benchmark Guidance Controller

As a guidance controller in the benchmark scenario, we use the nonlinear differential geometric path-following guidance (NDGPFG) by Cho et al. [32], which is an extension of the widely adopted guidance law published by Park et al. [147]. It is conceptually a line-of-sight (LOS) guidance method with a look-ahead vector that defines a reference position on the path and an approach angle that is subject to design parameters, but in general monotonically decreasing with decreasing cross-track error. The output of the guidance controller is a linear acceleration command in the inertial frame that is orthogonal to the velocity of the UAV, which makes it suitable for simultaneous path-following and airspeed stabilization. We repeat the essential ingredients to implement the algorithm for completeness but refer to [32] for more details.

Let $\hat{\mathbf{T}}_P, \hat{\mathbf{N}}_P \in \mathbb{R}^3$ denote the vectors that are tangent and normal to the path at the closest projection point $\mathbf{P}$, and let the curvature at $\mathbf{P}$ be denoted by κ_P. Let δ_{BL}, k, $\epsilon \in \mathbb{R}$ denote design parameters of the guidance algorithm. The shifted distance along the path, denoted by d_{shift}, is given by

$$d_{\mathrm{shift}} \triangleq \frac{|\kappa|}{k} \frac{\delta_{\mathrm{BL}}}{1-\epsilon} \tag{2.66}$$

such that the distance to the reference point is

$$\mathbf{d} \triangleq \mathbf{e} + d_{\mathrm{shift}} \mathrm{sign}(\kappa_P) \hat{\mathbf{N}}_P. \tag{2.67}$$

The approach angle is computed via

$$\theta_L \triangleq \arccos\left((1-\epsilon)\mathrm{sat}\left(\frac{\|d\|}{\delta_{\mathrm{BL}}}\right)\right) \tag{2.68}$$

which results in the look-ahead vector

$$\hat{\mathbf{L}} \triangleq \cos\theta_L \hat{\mathbf{d}} + \sin\theta_L \hat{\mathbf{T}}_P, \tag{2.69}$$

and the acceleration reference

$$\dot{\mathbf{v}}^n_{nb,\mathrm{ref}} = k(\mathbf{v}^n_{nb} \times \hat{\mathbf{L}}) \times \mathbf{v}^n_{nb}. \tag{2.70}$$

In the benchmark scenario, we use the design parameters $\delta_{\mathrm{BL}} = 100$, $k = 0.04$, and $\epsilon = 10^{-4}$.

We then transform $\dot{\mathbf{v}}^n_{nb,\mathrm{ref}}$ to the body-fixed frame

$$\dot{\mathbf{v}}^b_{nb,\mathrm{ref}} = \mathbf{R}_{nb}{}^{\top} \dot{\mathbf{v}}^n_{nb,\mathrm{ref}}, \tag{2.71}$$

and compute the roll and pitch references

$$\phi_{\text{ref}} = \phi_0 + \arctan(\mathbf{e}_2{}^\top \dot{\mathbf{v}}^b_{nb,\text{ref}}/g)\cos(\theta) \tag{2.72}$$

$$\theta_{\text{ref}} = \theta_0 + \arcsin(-\mathbf{e}_3{}^\top \dot{\mathbf{v}}^b_{nb,\text{ref}}/g) + k_{i,h}\int_0^t (\mathbf{e}_3{}^\top \mathbf{d})d\tau, \tag{2.73}$$

where ϕ_0 and θ_0 denote the attitude at trim state. Note that the lateral guidance controller as presented here is very similar to the L1 Guidance implemented in the ArduPlane Controller.

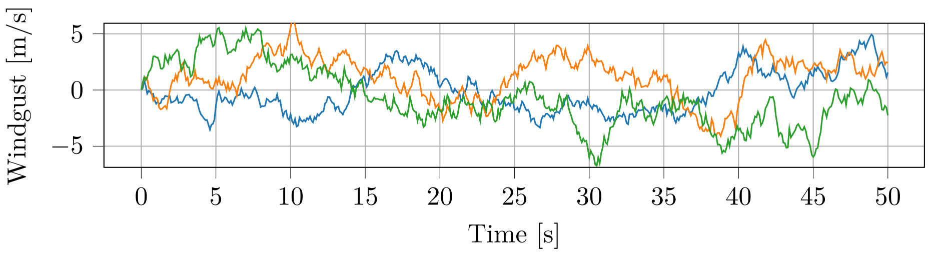

Figure 2.4: Gust wind profile for the benchmark simulations.

2.5.3 Performance Metrics

In all simulations, the reference signal for attitude and speed is assumed constant throughout the entire prediction horizon of the MPCs, which allows for a fair comparison to the baseline controllers. The tracking performance of the controllers and their actuator usage are evaluated based on the metrics

$$J_e = \frac{1}{n}\sum_{i=1}^{n}|e_i|, \quad J_u = \frac{1}{n}\sum_{i=1}^{n}|u_i|, \quad J_f = \frac{2}{n_f f_s}\sum_{i=1}^{n_f} M_i f_i, \tag{2.74}$$

where n denotes the amount of samples in the data series. The sampling frequency is denoted by f_s and f_i, M_i denote the respective frequencies and magnitudes of the control signal. The metric S_f jointly considers the amplitudes of the evaluated frequencies, effectively measuring the smoothness of the control signals [136].

2.5.4 Simulation Setup

The actuators in the simulation are modelled with first-order lag dynamics

$$\dot{\delta}_i = \frac{1}{T}(\delta_{i,\text{ref}} - \delta_i) \tag{2.75}$$

The wind is modelled as the combination of a static component in the inertial frame and a gust component in the body-fixed frame as $\mathbf{v}^n_{nw} = \mathbf{v}^n_{nw,s} + \mathbf{R}_{nb}\mathbf{v}^b_{nw,g}$. The gust component is generated by the Dryden wind model, which means essentially passing white noise through a low-pass filter. The resulting sample of the gust wind

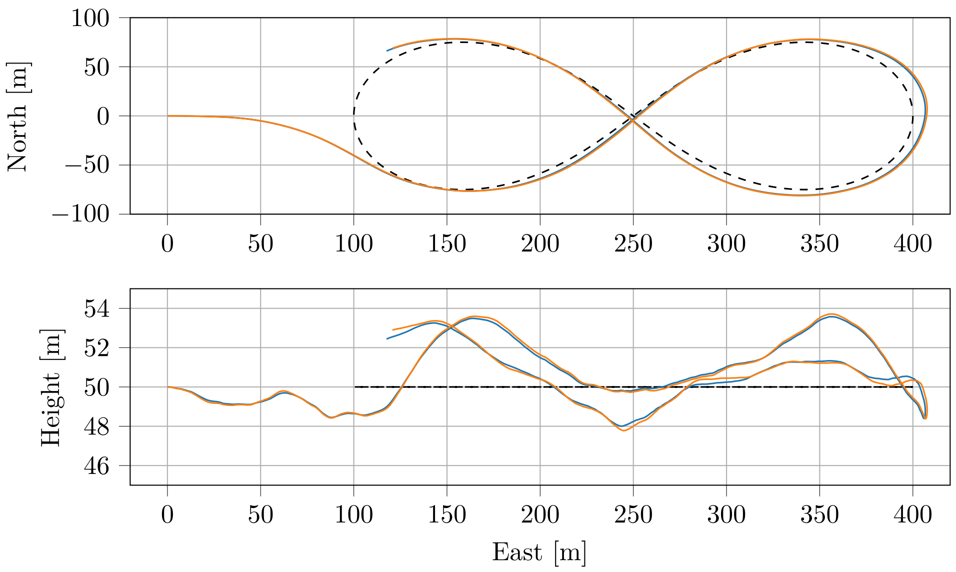

Figure 2.5: The position of the UAV in the benchmark simulation for the PID (blue) and AP controller (orange). Both controllers are tracking the attitude and speed reference coming from the NDGPFG which follows the reference path (black, dashed).

profile used in the benchmark simulation is depicted in Fig. 2.4. The static wind component in the inertial frame is set to $\mathbf{v}^n_{nw,s} = \begin{bmatrix}4 & 3 & 0\end{bmatrix}^\top$.

The initial conditions of the UAV are set to trimmed horizontal flight in east direction at 18 m/s airspeed given the static wind. The resulting initial state is given by

$$\mathbf{p}^n_{nb}(0) = \begin{bmatrix}0.0\\0.0\\-50.0\end{bmatrix}\text{m}, \quad \Theta_{nb}(0) = \begin{bmatrix}0.0\\1.76\\90.0\end{bmatrix}\text{deg}, \tag{2.76}$$

$$\mathbf{v}^b_{nb}(0) = \begin{bmatrix}17.99\\0.0\\0.55\end{bmatrix}\text{m/s}, \quad \boldsymbol{\omega}^b_{nb}(0) = \begin{bmatrix}0.0\\0.0\\0.0\end{bmatrix}\text{deg/s}, \quad \boldsymbol{\delta}(0) = \begin{bmatrix}\delta_a(0)\\\delta_e(0)\\\delta_t(0)\end{bmatrix} = \begin{bmatrix}0.0\text{ deg}\\2.10\text{ deg}\\0.12\end{bmatrix}.$$

The path to be followed by the UAV is a horizontal lemniscate defined by

$$\mathbf{p}(u) = \mathbf{r}_o + \mathbf{R}_{np}(\phi_p, \theta_p, \psi_p)\begin{bmatrix}\frac{l}{2}\cos(u)/(1+\sin(u)^2)\\\frac{w}{2}\sqrt{2}\sin(2u)/(1+\sin(u)^2)\\0\end{bmatrix} \tag{2.77}$$

where $u \in \mathbb{R}$ denotes a path variable, $l,\ w \in \mathbb{R}$ define the size of the lemniscate, and $\mathbf{r}_o \in \mathbb{R}^3$ denotes the origin. The rotation matrix $\mathbf{R}_{np}$ describes the orientation of the lemniscate in the NED frame. For the benchmark scenario, the path is parameterized with

$$\mathbf{r_0} = \begin{bmatrix} 0 \\ 250 \\ -50 \end{bmatrix}, \quad \mathbf{R}_{np}(0,0,\pi/2) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad l = 300, \quad w = 150. \tag{2.78}$$

For the simple PID design and the ArduPlane controller, the simulation results in which the UAV approaches the path and then follows it are depicted in Fig. 2.5 and Fig. 2.6. Both controllers are able to track the low-level motion commands sent by the guidance controller at almost no distinguishable actuator usage and performance. In the following chapters, we will come back to this benchmark scenario to evaluate the performance of each developed controller. A complete comparison for all controllers is given in Appendix A, which also includes the tuning parameters for each controller.

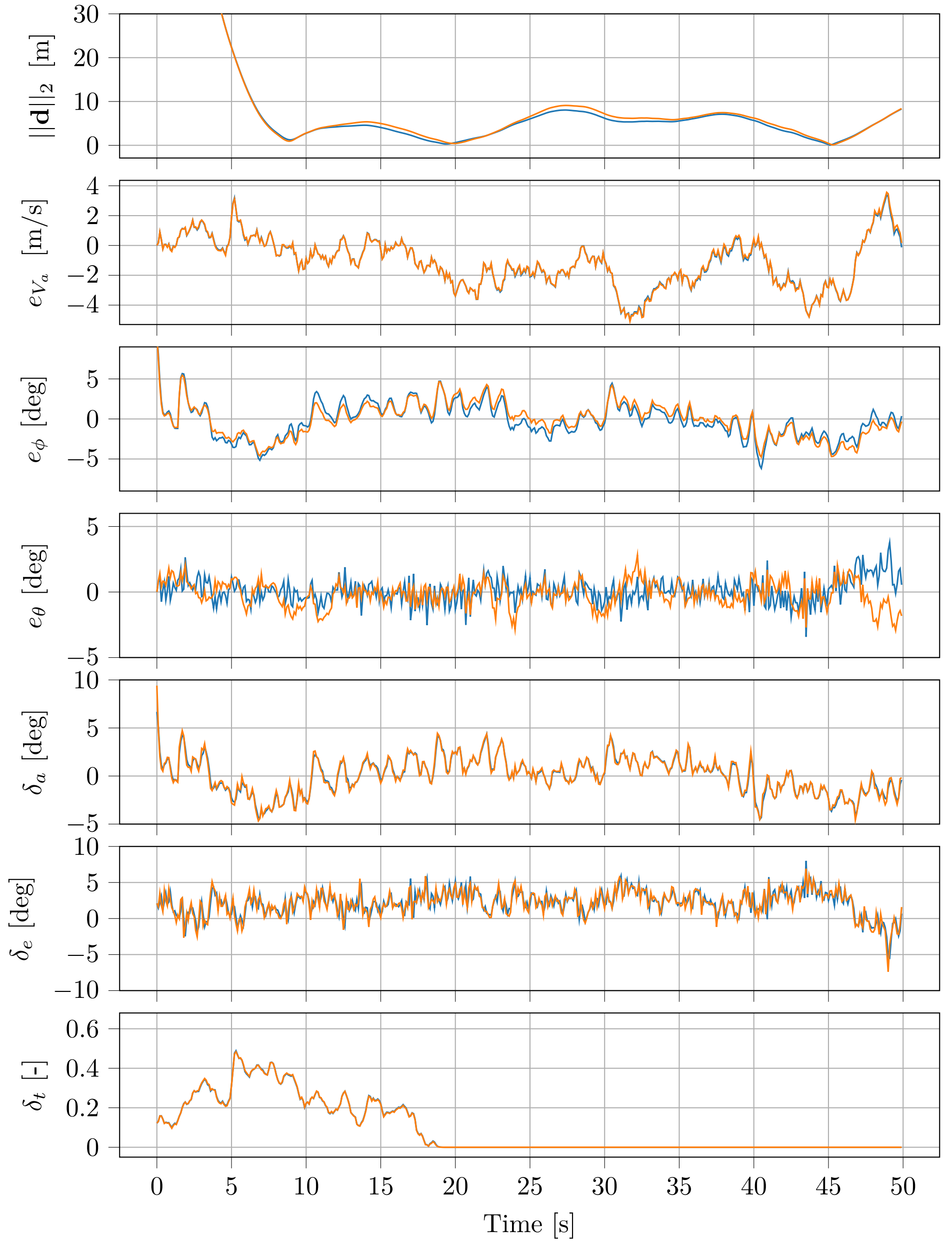

Figure 2.6: The position of the UAV in the benchmark simulation for the simple PID (PID, blue) and ArduPlane controller (AP, orange). The distance to the path $\|\mathbf{d}\|_2$ is stabilized by the NDGPFG. The airspeed error $\mathbf{e}_{V_a}$, roll error $\mathbf{e}_\phi$ and pitch error $\mathbf{e}_\theta$ are stabilized by the PID and AP controller. Both controllers result in very similar actuator usage and performance.

Chapter 3

Experimental Platform

This chapter includes a description of the platform architecture and test procedures that we used in the experiments to test the controllers. The work described was also intended to facilitate the testing of the DRL controller in [14]. The content of this chapter, in addition to experimental results and lessons learned, is submitted as a conference contribution:

[38] Erlend M. Coates, Dirk Reinhardt, Kristoffer Gryte, and Tor Arne Johansen. Toward Nonlinear Flight Control for Fixed-Wing UAVs: System Architecture, Field Experiments, and Lessons Learned. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, accepted.

3.1 Architecture

In this section, we describe our experimental platform, which has evolved over several years as a result of a wide range of research topics carried out at the NTNU UAV-lab. An alternative system architecture is described in [199], which provides a flexible architecture for system integration that is well suited for research on high-level planning, guidance and payload control, but less ideal for low-level control research. For our purpose, the goal was to extend the existing capabilities with the following:

1. An embedded platform powerful enough to run low-level NMPC online on-board the vehicle at a sufficiently high update frequency.
2. This platform should also have direct access to the actuators.
3. The system should be flexible enough to run a wide range of advanced control algorithms, also including DRL and MPC.
4. To lower the threshold for early testing of highly experimental low-level control algorithms (and to lower the risk of crashing), we needed some way to safely transition between the well-tested (and trusted) standard autopilot and our experimental algorithms.
5. For continued safe operation, the added functionality should not interfere with the existing fail-safe systems.

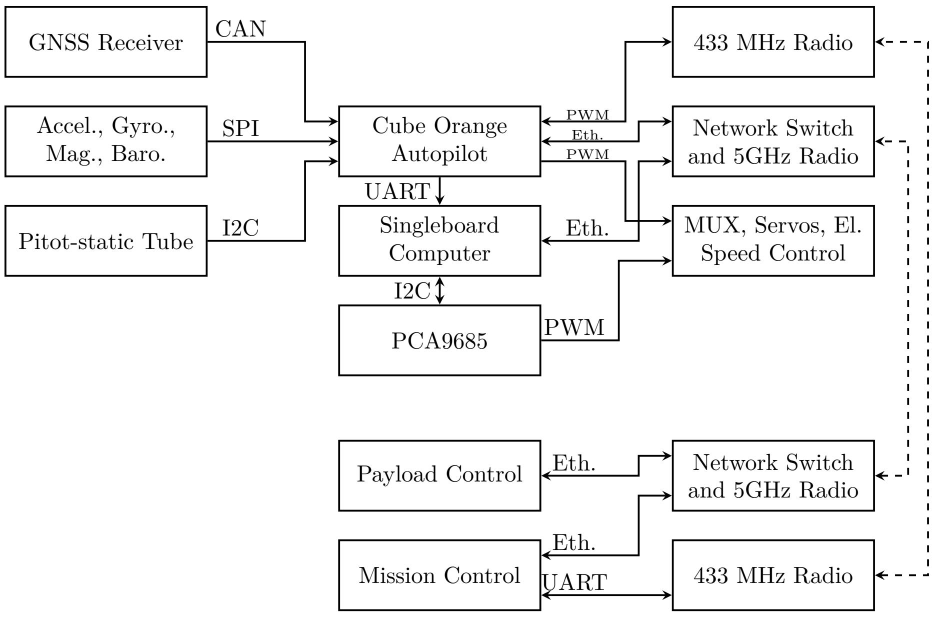

Figure 3.1: Hardware configuration of the experimental platform.

6. A software-in-the-loop (SITL) simulation environment to test the airworthiness of the low-level algorithms before conducting flight experiments.
7. Support for an automated reference generator to gather repeated sample trajectories for system identification, as well as evaluation and comparison of different algorithms.

An overview of the hardware configuration and communication architecture of the UAV and ground station is depicted in Fig. 3.1. We proceed by describing each main element of the system architecture.

3.1.1 UAV Platform

Our platform is built around a Skywalker X8 airframe, depicted in Fig. 3.2. The X8 is a tailless aircraft with two elevon control surfaces, one on each wing, which can be moved differentially to produce a rolling acceleration, or collectively, to produce a pitch acceleration. The control signals consists of pulse width modulation (PWM) signals to the two servo motors that actuate the elevons, and the throttle signal (also PWM) to a consumer-grade electronic speed control (ESC) that controls the motor and propeller in the back of the UAV.

The standard avionics flight stack is centered around a Cube Orange[1] that is running ArduPlane open-source autopilot, which is the fixed-wing build of the ArduPilot firmware [7]. The sensor suite consists of triple redundant IMUs with magnetometers, pressure sensors for altitude and airspeed (pitot-static tube), and a GNSS receiver.

[1]https://cubepilot.org/

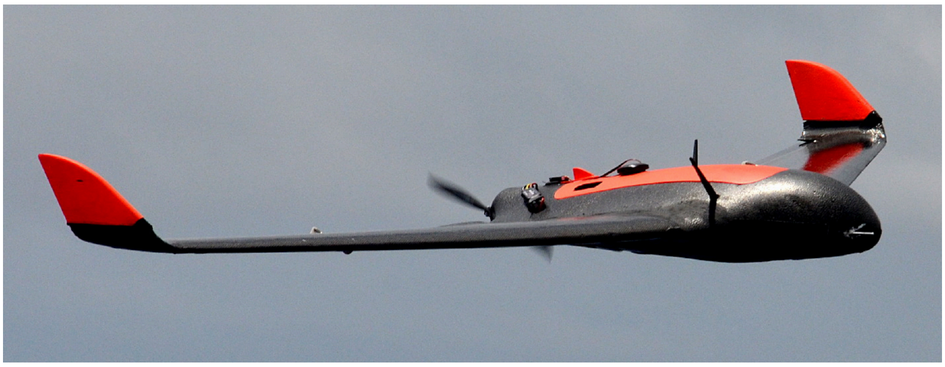

Figure 3.2: Skywalker X8 fixed-wing UAV.

3.1.2 Payload Computer

Alongside the Cube Orange, we use the Khadas Vim3[2] SBC that includes four 2.2Ghz Cortex-A73 cores and two 1.8Ghz Cortex-A53 cores. We initially started using other SBCs, including the Odroid-XU4 and a Raspberry Pi 4. After a series of benchmarking tests, we settled with the Khadas Vim3, mainly driven by the computational requirements of the NMPC. Details on the simulations and closed-loop runtime of the solver on the SBC can be found in Chapter 5.

On the SBC we run the DUNE Uniform Navigation Environment. DUNE is part of the LSTS tool-chain [148] developed at the Underwater Systems and Technology Laboratory (LSTS), University of Porto. DUNE allows us to write different tasks that run independently of each other on separate threads or processes, while exchanging data using a message bus mechanism (similarly to ROS).

The NMPC is implemented using acados [185], which we interfaced as a DUNE Task. The closed-loop runtime of the solver was benchmarked for each SBC based on simulations that reflect targeted maneuvers. Benchmarking results for the Khadas Vim3 are depicted in Fig. 3.3, which show the closed-loop runtime of the solver to find solutions to the optimal control problem (OCP) at each solver update for a Monte-Carlo study that includes a range of initial conditions and environmental disturbances. Approximately 96% of the simulations allow the solver to find a solution in less than 50 ms after two controller updates when warm-starting the solver based on the time-shifted previous solution. We therefore chose an update period of 50 ms in the experiments, which led to satisfactory performance. More details can be found in Chapter 5 or [156].

The DRL controller is implemented as a DUNE task in C++ with the artificial neural network (ANN) implemented in TensorFlow. Benchmarking tests show that the controller is able to run with an update rate of several thousand hertz. However, this is orders of magnitude faster than needed since state estimates are delivered at 50 Hz (see next section). Naturally, the computational demands of the DRL controller is not the bottleneck when selecting our hardware, but rather that of the NMPC.

[2]https://www.khadas.com/vim3

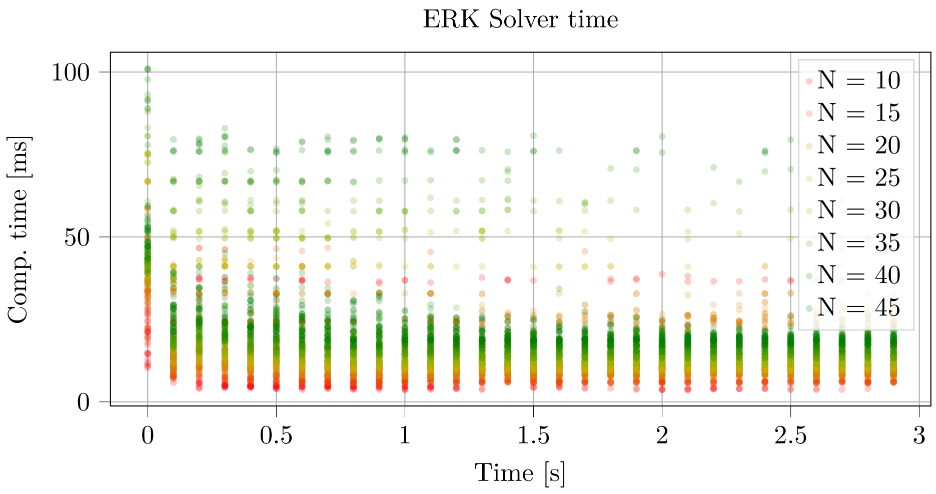

Figure 3.3: NMPC benchmarking results for the employed SBC for different prediction horizons N in a direct multiple-shooting scheme with 0.1 s shooting interval.

3.1.3 State Estimates

State estimates, including estimates of the local wind velocity, are provided by ArduPilot's EKF, and is propagated to the SBC together with attitude references (either originating from the pilot's radio transmitter or ArduPlane's guidance system) and other auxiliary signals via a serial communication link using the MAVLink protocol. This provides our controllers with all the necessary data. The MAVLink communication was configured to provide data at the highest possible rate, which in this case is 50 Hz, corresponding to the loop rate of the ArduPlane scheduler.

Communicating such large amount of data at a high rate turned out to be a demanding task for the ArduPilot system, which in turn made us select the Cube Orange among several candidate autopilots. Cube Orange is (as of February 2022) the most powerful of the CubePilot series of autopilots, with a 400MHz ARM Cortex M7 processor. Benchmarking tests showed that less powerful autopilot hardware platforms such as the Pixhawk 1 and Pixhawk 2.1/Cube Black was not powerful enough to handle the high data throughput over the serial link.

3.1.4 Actuators

Our controllers output desired throttle and control surface deflections that are converted to PWM duty cycle using static linear maps. For the elevons these were identified based on lab experiments using a camera.

Remark 3.1. For future work, an interesting extension to a static linear mapping would be to identify a second-order model of the actuator dynamics. This can be done with a series of step responses that can be recorded in a motion capture lab [100]. More advanced servos that provide position feedback and control of the surface deflection angles can also be considered for model-based control.

Most of the SBCs require additional hardware for PWM output. For instance, the Odroid-XU4 has no hardware PWM ports, and the Raspberry Pi 4 only has two (we need three). For the Khadas Vim3, we chose a solution based on a PCA9685 servo driver which is interfaced through inter-integrated circuit (I2C) communication.

3.1.5 Multiplexer Switch

The PWM signals to the actuators can be set by both computing platforms. A PWM multiplexer (MUX) is used to switch between the controller that runs on the SBC and the ArduPilot controllers. A switch on the pilot's radio transmitter is mapped to the MUX switch using ArduPilot's RC pass-through functionality, allowing the pilot to choose the output source at any given time, including a manual recovery if loss of control should occur. To achieve an additional layer of safety, the manual mode always overrides the SBC output.

This architecture allows for a redundant PID controller to run on the autopilot that may overwrite the commands from the experimental controller whenever necessary to ensure a safe operation, for example if instability occurs or when the required update rates of an optimization-based controller such as MPC can not be met by the employed solver. The switching mechanism enables us to safely engage the highly experimental low-level control code in flight, while takeoff and landing are performed by the pilot operating the standard ArduPlane autopilot.

For an alternative control selection method, based on a performance monitoring scheme, together with a thorough discussion of MPC employed on alternative computing platforms such as field programmable gate arrays (FPGAs), see [84].

3.1.6 Fail-Safe

The ArduPilot system includes standard fail-safe functionality, such as an automatic return to launch (RTL) mode that is triggered if the pilot's radio transmitter signal is out of range or otherwise lost. Since this includes the MUX switch signal, we had to augment the fail-safe functionality. Otherwise, if the signal is lost when the SBC is in control, we would have no way to recover the aircraft should our algorithms fail. To solve this, the fail-safe configuration of our RC receiver (FrSky) is set to move the controls to the following preset values in the case of a lost control signal for some period of time:

- The mode switch is set to RTL.
- The MUX switch is set such that the Cube Orange's PWM output is sent to the servos.
- The other controls are set such that they correspond to centered sticks on the transmitter.

This way, our fail-safe system works similarly to ArduPilot's. In addition, we are sure that ArduPilot will be in control should we lose the control signal. A potential downside of this is that the ArduPilot system will not be aware that the control signal is out of range, since the receiver channels are just set to some preset values, instead of the usual "no signal".

3.1.7 Ground Station

Communication with the ground station is handled through a redundant radio link using one 433MHz SiK Telemetry Radio and a 5GHz Ubiquiti Rocket M5, both providing MAVLink communication with the Cube autopilot. The 5GHz radio also enables us to communicate with the SBC on a local area network (LAN) through an onboard network router (see [199] for details).

We use the ArduPilot-compatible ground control software Mission Planner running on a dedicated lab computer. Both radio communication links are used for redundancy, and multiplexing of the two radio signals is handled by MAVProxy.

Control of the DUNE controller tasks is done through Neptus, which is the command and control framework of the LSTS toolchain, communicating with DUNE using the inter module communication (IMC) protocol. Neptus allows the operator to set configuration parameters, monitor telemetry data and execute commands on the SBC.

3.1.8 Reference Generation for Automated Testing

We can use pre-defined signals to overwrite references coming from the guidance controller to test our low-level motion controllers repeatably. This means that we can e.g. use ArduPlane to fly a square waypoint mission, where we run repeated custom maneuvers when on the long sides of the square. Step sequences and chirp signals with increasing frequency turned out to be a good way to test the closed-loop dynamics with different control algorithms that need to be compared.

We follow a similar approach to collect data for identifying dynamic models of a particular airframe. However, instead of manipulating reference signals for the low-level controller, the actuator signals of a particular actuator are overwritten by suitable step sequences or oscillating signals. A frequency analysis of the open-loop model dynamics or of the linearized closed-loop system around trim states can be used to guide the parametrization of the test signals such that their power spectral density covers the natural frequencies of the system. The aim is to sufficiently excite the dynamics such that the collected aerodynamic data can be used for system identification.

This section provides a description of our testing procedures. To assess the airworthiness of our algorithms, we use a three-stage ground-testing process before finally attempting field experiments: (a) initial verification of promising designs in our laptop simulators, (b) SITL simulation to verify the platform specific implementation of the algorithms, and (c) system integration testing at the lab.

All of our simulators are based on previous and ongoing modeling efforts, in particular [39, 70, 71]. For more recent work on how to improve these models, see Chapter 7 or [154, 158].

3.1.9 Python Simulator

As a first verification step, prototype implementations of promising designs are first tested in simulator environments implemented in Matlab or Python.

Model mismatch can be introduced in a controlled environment to assess the algorithm's robustness to modeling errors. Also, initial tuning guidelines are established during this stage.

3.1.10 Software-In-The-Loop (SITL) Simulations

The SITL simulator is based on a combination of a SITL configuration of our DUNE application, in combination with ArduPilot's SITL framework, using a JSBSim simulation model for the Skywalker X8 based on our previously mentioned models. The standard SITL framework is sufficient for systems where the SBC only sends commands to a low-level autopilot using the MAVLink interface, e.g. when testing high-level guidance controllers. However, since we need to simulate the case where the SBC has direct access to the actuators, we need to extend this functionality.

Our solution uses MAVLink's "RC override" functionality to emulate the behavior of our physical system. In DUNE, instead of sending actuator signals to the PWM driver, the controller output is transmitted to ArduPlane SITL using our MAVLink interface, using the RC override message. In the simulator, these values are interpreted as servo set-points, *as if the UAV was under manual control.* Therefore, for this to work, ArduPlane needs to be in "MANUAL" mode.

To achieve automated testing of different maneuvers, we implemented a DUNE Task that essentially provides scripting capabilities of a succession of different maneuvers and system commands, including automated arming, takeoff and loitering, mode switching, as well as switching between ArduPlane and out controllers.

3.1.11 Lab Testing

At the lab, we conduct system integration tests on the physical hardware, checking all communication channels, and that critical systems work as expected. This includes the MUX switch, data logging, and telemetry. In particular, we check edge cases concerning arming/disarming of the propeller, and confirm that the MUX switch does not interfere with the safety-critical features.

When preparing for field tests, we first communicate the expected behavior of our system to the pilot, and demonstrate safety critical features. An important tool we use when verifying and configuring our controller implementations, is the surface deflection test ("ground test"), where we check that the control surfaces move in the correct directions in response to manually tilting the vehicle, or moving the transmitter sticks.

3.1.12 Field Experiments

When performing field experiments, we typically use a team of three persons: (1) the pilot (first in command), operating the UAV in the manually controlled modes using an RC transmitter, (2) ground station operator (second in command) operating the automatically controlled modes and setting ArduPilot parameters through Mission Planner, and (3) one researcher controlling the payload computer through Neptus.

This is typically the researcher that designed the experiment or implemented the algorithm that we test. During the experiment, the team communicates over radio. Additional personnel, if any, is in charge of taking notes.

The flight testing procedure can be roughly broken down into the following steps:

1. After all pre-flights checks are passed, the pilot takes off manually and puts the UAV into loiter mode or a square waypoint pattern.
2. With the experimental algorithm running in the background, we monitor its outputs while comparing them with the PWM values set by ArduPlane.
3. If everything looks good, we switch to our controller using the MUX switch mapped to a switch on the pilot's radio transmitter. When testing controllers with dynamic elements such as integral action or disturbance observers, the dynamic elements are engaged (or their states reset) when we perform the switch. This is to avoid any wind-up or other potential issues.
4. We then observe the behavior of the experimental controller and test it with increasingly challenging maneuvers, starting with straight and level flight. If the UAV performs any sudden maneuvers, or if substantial oscillations or instability occurs, the pilot takes back control over the UAV by using the MUX switch. The pilot's visual eye contact with the vehicle is aided by the other operators, constantly monitoring telemetry data, and warning the pilot if needed.
5. After some initial tuning, we initiate the automated maneuver sequences for tuning and repeatability of the collected evaluation data. This is especially useful when comparing the performance of two controllers.
6. When data collection is complete, we switch the actuator control back to ArduPlane using the MUX switch before landing.

Chapter 4

Geometric Attitude Control

The chapter is based on

[37] Erlend M. Coates, Dirk Reinhardt, and Thor I. Fossen. Reduced-Attitude Control of Fixed-Wing Unmanned Aerial Vehicles Using Geometric Methods on the Two-Sphere. *IFAC-PapersOnLine*, 53(2):5749–5756, 2020. 21st IFAC World Congress.

[153] Dirk Reinhardt, Erlend. M. Coates, and Tor Arne Johansen. Hybrid Control of Fixed-Wing UAVs for Large-Angle Attitude Maneuvers on the Two-Sphere. *IFAC-PapersOnLine*, 53(2):5717–5724, 2020. 21st IFAC World Congress.

4.1 Introduction

The attitude control system provides the main stabilization function in autopilots for fixed-wing UAVs. It enables a UAV to follow commands originating from outer-loop guidance schemes, thus allowing fully automatic flight. Guidance controllers typically achieve path-following or waypoint-tracking capabilities by controlling climb and turn rates through roll and pitch commands to the inner-loop attitude controller [9]. Turning is not achieved by controlling yaw angle or turn rate directly, but rather through banked-turn maneuvers.

The orientation, or *attitude*, of a fixed-wing aircraft relative to an inertial reference frame is represented, both globally and uniquely, by an element of the special orthogonal group SO(3), which is the set of 3 by 3 rotation matrices. The Euler angles given by roll, pitch and yaw provide a minimal, local coordinate system on SO(3), but will suffer from "gimbal-lock" singularities [119].

In the last decades, coordinate-free geometric attitude controllers, designed directly on SO(3), have been proposed in the literature, without the need for attitude parameterizations, and with no singularities [29]. Another advantage of these approaches is that such controllers are often *geodesic* in the sense that proportional action is designed to steer the vehicle along the shortest path in the physical rotation space, whereas controllers based on Euler angles are not. These advantages are desirable when the controlled vehicle is subject to large angle rotations, e.g. a UAV recovering from large attitude errors resulting from severe wind gusts [86].

Controllers designed on SO(3), or using quaternions [188], control the full attitude, and therefore can not be directly applied to fixed-wing aircraft using banked turn maneuvers. Instead of studying the full attitude, some authors consider a reduced-attitude representation, evolving on the two-sphere, $\mathbb{S}^2 \subset \mathbb{R}^3$ [21]. In this space of reduced attitude, all rotations that are related by a rotation about some fixed axis, are considered the same [27]. Control systems with reduced attitude evolving on $\mathbb{S}^2$ have previously been studied in the context of spin-axis stabilization of satellites [21], pendulum stabilization [27], path-following control of underwater vehicles [193], control of multi-rotor UAVs [24] and for general rigid bodies [126].

In the beginning of this chapter, we present a smooth, nonlinear reduced-attitude controller for fixed-wing UAVs, in a coordinate-free manner, using a global, singularity-free attitude representation on $\mathbb{S}^2$. The chosen reduced-attitude representation is independent of the yaw angle and thus enables traditional banked-turn maneuvers. A consequence of this is that the presented approach fits directly into existing control architectures that rely on roll and pitch control in the inner loop. Also, no lateral/longitudinal decoupling assumptions are used in the design.

The reduced-attitude representation allows for a convenient decomposition of the dynamics and a natural corresponding decoupling of the control objective into two parts: 1. Reduced-attitude (roll/pitch) control, and 2. Control of the angular velocity about the inertial z-axis (turn rate control). Since only two control torques are needed to control the reduced attitude, there is one degree of freedom left to do turn rate control, which essentially performs turn coordination, providing damping about the inertial z-axis, and reducing the sideslip angle.

Almost semi-global exponential tracking of reduced attitude is established using Lyapunov methods. In the special case of regulation, a stronger almost global asymptotic stability result is established. Because of topological constraints when dealing with compact manifolds [11], the latter is the strongest possible stability result possible for continuous attitude control systems [29].

Using hybrid control however, the region of attraction can be made global as shown in e.g. [125, 126]. We use these results to further extend the smooth, nonlinear reduced-attitude controller to use hybrid proportional feedback and solve the problem of vanishing proportional action at the opposite direction of the reference point, which otherwise introduces significant performance limitations [30].

Recently, a general discussion of attitude tracking on $\mathbb{S}^n$ has been published in [24], with application to multi-rotor UAVs. However, despite their desirable stability properties, results for fixed-wing UAVs have not been published in the literature. We thus discuss a suitable adaptation in this chapter.

4.2 Preliminaries and Problem Statement

4.2.1 Notation

We briefly discuss notation that is specific to this chapter and the related proofs. The rotation matrix and velocities express the rotation and velocity of the body-fixed frame with respect to the NED frame in the coordinates of the body-fixed frame, i.e. $\mathbf{R} \triangleq \mathbf{R}_{nb}$ and $\boldsymbol{\omega} \triangleq \boldsymbol{\omega}_{nb}^{b}$. Relevant dynamic equations will however be recalled to make the reading easier.

Explicit time arguments will be used for state variables only when considering specific solutions, or for signals and functions in general when we want to highlight that time-varying exogenous signals are considered. At some points we will use a slight abuse of notation and write e.g. $V(t)$ for a Lyapunov function evaluated along system trajectories, when we really mean $V(\mathbf{x}(t))$. Lastly, the positive real numbers will be denoted $\mathbb{R}_{>0}$, the set of 3 by 3 symmetric positive definite matrices will be denoted $\mathcal{P}^3_+$.

4.2.2 UAV Attitude Dynamics

Throughout this chapter, we will consider the fully actuated attitude dynamics for a fixed-wing UAV in control-affine form. Recall the dynamics in Eq. (2.32c), Eq. (2.51) given as

$$\dot{\mathbf{R}} = \mathbf{R}\mathbf{S}(\omega) \tag{4.1}$$

$$\mathbf{J}\dot{\boldsymbol{\omega}} = \mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r) + \mathbf{G}(\mathbf{v}_r)\begin{bmatrix}\delta_a & \delta_e & \delta_r\end{bmatrix}^\top \tag{4.2}$$

where $\boldsymbol{\omega} \in \mathbb{R}^3$ denotes the angular velocity in the body-fixed frame, $\mathbf{J} = \mathbf{J}^\top \in \mathbb{R}^{3\times 3}$ denotes the inertia matrix. Given aileron, elevator and rudder deflections δ_a, δ_e, $\delta_r \in \mathbb{R}$ we define the control input vector $\mathbf{u} = \begin{bmatrix}\delta_a & \delta_e & \delta_r\end{bmatrix}^\top$.

For UAVs that are dependent on control surface deflections to deflect an airstream, a sufficiently large airspeed is needed to ensure controllability. We thus consider flight conditions for which the following assumption holds

Assumption 9. The control effectiveness matrix $\mathbf{G}(\mathbf{v}_r)$ has full rank.

From Eq. (2.49) and the related discussion, it is clear that a consequence of this assumption is that a strictly positive airspeed is required, i.e. $V_a \geq \underline{V}_a$ for some $\underline{V}_a \in \mathbb{R}_{>0}$.

Remark 4.1. For common parameterizations based on constant control effectiveness coefficients [9, 177], it can be shown that the full rank condition corresponds to *primary control* coefficients being larger than the coefficients associated with *secondary* roll-yaw coupling effects. The full rank assumption is therefore reasonable for most common fully actuated control surface configurations.

We also need an additional assumption to ensure that the moment vector is uniformly bounded:

Assumption 10. There exist constants $c_f, c_G > 0$ such that $\|\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r)\| \leq c_f$ and $\|\mathbf{G}(\boldsymbol{\omega}, \mathbf{v}_r)\| \leq c_G$.

The scope throughout this chapter is on the rotational subsystem of the dynamics, meaning that the throttle δ_t and relative velocity $\mathbf{v}_r$, and therefore also α, β and V_a (as functions of $\mathbf{v}_r$), will be treated as known time-varying input signals.

Remark 4.2. Note that since the translational subsystem (see [177]) depends on $\mathbf{R}$, $\boldsymbol{\omega}$ and $\mathbf{u}$, the relative velocity $\boldsymbol{v}_r$ is not truly an exogenous signal. Nevertheless, as a decoupling maneuver, we will assume that it is a known signal available for feedback. This should be considered when integrating the control system developed in this paper in UAV GNC systems, e.g. using bandwidth separation.

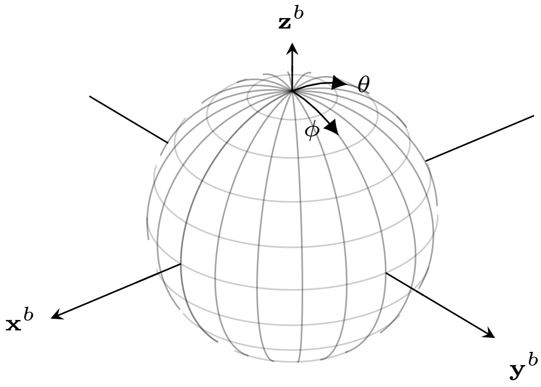

Figure 4.1: The two-sphere parameterized by the angles roll ϕ and pitch θ.

4.2.3 Reduced Attitude

We make use of the reduced-attitude vector $\boldsymbol{\Gamma} \in \mathbb{S}^2$ as presented by Chaturvedi et al. [29] and define it here as the representation of the vertical axis of the inertial frame $\mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$ expressed in the body-fixed frame

$$\boldsymbol{\Gamma} = \mathbf{R}^\top \mathbf{e}_3. \tag{4.3}$$

The same reduced-attitude parameterization has been applied to stabilization of the inverted 3D pendulum [30]. Note that the reduced-attitude vector is invariant to rotations about e_3 and therefore independent of yaw. In fact, given a roll angle $\phi \in [-\pi, \pi]$ and pitch angle $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, the reduced-attitude vector can be parameterized as

$$\boldsymbol{\Gamma}(\phi, \theta) = \begin{bmatrix} -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}^\top, \tag{4.4}$$

which can be seen by expanding Eq. (4.3) using Eq. (2.21). A graphical illustration how the two-sphere is parameterized by roll and pitch angles is given in Fig. 4.1. Using Eq. (2.17), we can obtain an orthogonal decomposition of the angular velocity $\boldsymbol{\omega}$ with respect to $\boldsymbol{\Gamma}$ such that $\boldsymbol{\omega} = \boldsymbol{\omega}^\perp + \boldsymbol{\omega}^\parallel$, where

$$\boldsymbol{\omega}^\perp \triangleq \Pi_\Gamma^\perp \boldsymbol{\omega} \in \mathrm{T}_\Gamma \mathbb{S}^2 \qquad \boldsymbol{\omega}^\parallel \triangleq \Pi_\Gamma^\parallel \boldsymbol{\omega} \in \mathrm{N}_\Gamma \mathbb{S}^2. \tag{4.5}$$

The kinematic equation follows by differentiating Eq. (4.3) using Eq. (4.1) and is given by

$$\dot{\boldsymbol{\Gamma}} = \boldsymbol{\Gamma} \times \boldsymbol{\omega} = \boldsymbol{\Gamma} \times \boldsymbol{\omega}^\perp \in \mathrm{T}_\Gamma \mathbb{S}^2. \tag{4.6}$$

The parallel component $\boldsymbol{\omega}^\parallel$ is the angular velocity about the inertial z-axis (expressed in the body-fixed frame), and clearly does not influence $\dot{\boldsymbol{\Gamma}}$. Differentiating Eq. (4.5) and using Eq. (4.6) gives

$$\dot{\boldsymbol{\omega}}^\perp = \boldsymbol{\Pi}_\Gamma^\perp \dot{\boldsymbol{\omega}} + \boldsymbol{\omega}^\perp \times \boldsymbol{\omega}^\parallel \tag{4.7}$$

$$\dot{\boldsymbol{\omega}}^\parallel = \boldsymbol{\Pi}_\Gamma^\parallel \dot{\boldsymbol{\omega}} + \boldsymbol{\omega}^\parallel \times \boldsymbol{\omega}^\perp, \tag{4.8}$$

where we have also applied the identity $\mathbf{x}^\top \mathbf{S} \mathbf{x} = \mathbf{0}$ for any $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{S} \in \mathfrak{so}(3)$.

4.2.4 Reference System

Let a time-varying reduced-attitude reference vector $\mathbf{\Gamma}_d(t) \in \mathbb{S}^2$ satisfy

$$\dot{\mathbf{\Gamma}}_d = \mathbf{\Gamma}_d \times \boldsymbol{\omega}_d, \tag{4.9}$$

where $\boldsymbol{\omega}_d \in \mathrm{T}_{\mathbf{\Gamma}_d}\mathbb{S}^2$. Consider the projection of $\boldsymbol{\omega}_d$ onto the tangent space $\mathrm{T}_\Gamma\mathbb{S}^2$, given by $\mathbf{\Pi}_\Gamma^\perp \boldsymbol{\omega}_d \in \mathrm{T}_\Gamma\mathbb{S}^2$. Using Eq. (2.3), Eq. (2.16), Eq. (2.18), Eq. (4.6), the derivative can be found to satisfy

$$\frac{d}{dt}(\mathbf{\Pi}_\Gamma^\perp \boldsymbol{\omega}_d) = \mathbf{\Pi}_\Gamma^\perp \dot{\boldsymbol{\omega}}_d + \boldsymbol{\omega}^\perp \times \mathbf{\Pi}_\Gamma^\parallel \boldsymbol{\omega}_d + \mathbf{\Pi}_\Gamma^\parallel (\boldsymbol{\omega}_d \times \boldsymbol{\omega}^\perp). \tag{4.10}$$

We assume that $\boldsymbol{\omega}_d$ is twice continuously differentiable and that its derivative is continuous and uniformly bounded, i.e. there exist constants $c_{\boldsymbol{\omega}_d}, c_{\dot{\boldsymbol{\omega}}_d}$ such that

$$\boldsymbol{\omega}_d(t) \in c_{\omega_d}\mathbb{B}, \qquad \dot{\boldsymbol{\omega}}_d(t) \in c_{\dot{\omega}_d}\mathbb{B}. \tag{4.11}$$

The reasons for this assumption are twofold. First, the reference trajectory is smooth such that it may be used in feedforward-terms of the control law. And second, the differential inclusion of $\dot{\boldsymbol{\omega}}_d(t)$ allows to formulate an autonomous closed-loop system such that hybrid invariance principles can be applied.

We briefly give an example for a reference filter that can be used to generate a reference trajectory $(\mathbf{\Gamma}_d(t), \boldsymbol{\omega}_d(t), \dot{\boldsymbol{\omega}}_d(t))$ based on trajectories parameterized by Euler angles. An expression for the reduced-attitude vector $\mathbf{\Gamma}$ in terms of the Euler angles roll and pitch is given by Eq. (4.4). Differentiating leads to

$$\dot{\mathbf{\Gamma}} = \begin{bmatrix} -\cos(\theta)\dot{\theta} \\ -\sin(\theta)\sin(\phi)\dot{\theta} + \cos(\theta)\cos(\phi)\dot{\phi} \\ -\sin(\theta)\cos(\phi)\dot{\theta} - \cos(\theta)\sin(\phi)\dot{\phi} \end{bmatrix}. \tag{4.12}$$

For $\boldsymbol{\omega}_{nb}^{b\perp} \in \mathrm{T}_\Gamma\mathbb{S}^2$ we can invert Eq. (4.6) using Eq. (4.4) and Eq. (4.12) to get

$$\boldsymbol{\omega}^\perp = \dot{\mathbf{\Gamma}} \times \mathbf{\Gamma} = \begin{bmatrix} \cos^2(\theta)\dot{\phi} \\ \cos(\phi)\dot{\theta} + \sin(\theta)\cos(\theta)\sin(\phi)\dot{\phi} \\ -\sin(\phi)\dot{\theta} + \sin(\theta)\cos(\theta)\cos(\phi)\dot{\phi} \end{bmatrix}, \tag{4.13}$$

with derivative

$$\dot{\boldsymbol{\omega}}^\perp = \ddot{\mathbf{\Gamma}} \times \mathbf{\Gamma}, \tag{4.14}$$

where $\ddot{\mathbf{\Gamma}} = [\ddot{\Gamma}_1 \;\; \ddot{\Gamma}_2 \;\; \ddot{\Gamma}_3]^\top$ can be found by differentiating Eq. (4.12), and its elements are given by

$$\ddot{\Gamma}_1 = \sin(\theta)\dot{\theta}^2 - \cos(\theta)\ddot{\theta} \tag{4.15a}$$

$$\ddot{\Gamma}_2 = -\cos(\theta)\sin(\phi)(\dot{\theta}^2 + \dot{\phi}^2) - 2\sin(\theta)\cos(\phi)\dot{\theta}\dot{\phi} - \sin(\theta)\sin(\phi)\ddot{\theta} + \cos(\theta)\cos(\phi)\ddot{\phi} \tag{4.15b}$$

$$\ddot{\Gamma}_3 = -\cos(\theta)\cos(\phi)(\dot{\theta}^2 + \dot{\phi}^2) + 2\sin(\theta)\sin(\phi)\dot{\theta}\dot{\phi} - \sin(\theta)\cos(\phi)\ddot{\theta} - \cos(\theta)\sin(\phi)\ddot{\phi}. \tag{4.15c}$$

Given twice continuously differentiable reference trajectories $\phi_d(t), \theta_d(t)$ and their first and second derivatives (e.g. using third order linear reference filters [56]), the relations Eq. (4.4) and Eq. (4.12) - Eq. (4.15c) can be used to generate continuous signals $\mathbf{\Gamma}_d(t)$, $\boldsymbol{\omega}_d(t)$, $\dot{\boldsymbol{\omega}}_d(t)$, which can be used to implement the tracking controller that we discuss in this chapter.

4.2.5 Potential Function and Error States

Let a smooth configuration error function $\Psi\colon \mathbb{S}^2 \times \mathbb{S}^2 \to \mathbb{R}$ be defined by

$$\Psi(\mathbf{\Gamma}, \mathbf{\Gamma}_d) = \frac{1}{2}\|\mathbf{\Gamma} - \mathbf{\Gamma}_d\|^2 = 1 - \mathbf{\Gamma}_d \cdot \mathbf{\Gamma}. \tag{4.16}$$

The function Ψ measures the “distance” between two points $\mathbf{\Gamma}$ and $\mathbf{\Gamma}_d$ on $\mathbb{S}^2$, and is positive definite with respect to $\mathbf{\Gamma} = \mathbf{\Gamma}_d$. There are two critical points, i.e. points where the gradient of the function vanishes: A minimum when $\mathbf{\Gamma} = \mathbf{\Gamma}_d$, and a maximum when $\mathbf{\Gamma} = -\mathbf{\Gamma}_d$. Our goal is to drive the potential to its minimum by finding a proportional feedback in the direction of the gradient. To this end, let a variation of a curve $\mathbf{R}(t) \in \mathrm{SO}(3)$ be given in terms of a perturbation parameter $\boldsymbol{\epsilon} \in \mathbb{R}^3$ as $\mathbf{R}_\epsilon(t, \boldsymbol{\epsilon}) = \mathbf{R}(t)e^{\boldsymbol{\epsilon}\mathbf{S}(\boldsymbol{\eta}(t))}$, where $\boldsymbol{\eta}(t) \in \mathrm{T}_{\Gamma(t)}\mathbb{S}^2\,\forall t$. The corresponding variation of $\mathbf{\Gamma}(t)$ on $\mathbb{S}^2$ is given by

$$\mathbf{\Gamma}_\epsilon(t, \boldsymbol{\epsilon}) = \mathbf{R}_\epsilon^\top(t, \boldsymbol{\epsilon})\mathbf{e}_3 = e^{-\boldsymbol{\epsilon}\mathbf{S}(\boldsymbol{\eta}(t))}\mathbf{\Gamma}(t), \tag{4.17}$$

and an infinitesimal variation can be found by

$$\delta\mathbf{\Gamma}(t) = \left.\frac{d}{d\boldsymbol{\epsilon}}\right|_{\epsilon=0} \Gamma_\epsilon(t, \boldsymbol{\epsilon}) = \mathbf{\Gamma}(t) \times \boldsymbol{\eta}(t). \tag{4.18}$$

The derivative of Ψ in the direction $\delta\mathbf{\Gamma}$ is given by

$$\nabla_\Gamma \Psi \cdot \delta\mathbf{\Gamma} = -\mathbf{\Gamma}_d \cdot (\mathbf{\Gamma} \times \boldsymbol{\eta}) = -\boldsymbol{\eta} \cdot (\mathbf{\Gamma}_d \times \mathbf{\Gamma}) = \boldsymbol{\eta} \cdot \mathbf{e}_\Gamma, \tag{4.19}$$

so with $\mathbf{\Gamma}_d$ fixed, the term $\mathbf{\Gamma}_d \times \mathbf{\Gamma}$ can be viewed as the gradient vector field on $\mathbb{S}^2$ induced by the potential function Ψ. The example in Fig. 4.2 illustrates this and also shows that the gradient vector field vanishes at both critical points. In subsequent Lyapunov analysis, Ψ will be used as pseudo-potential energy[1].

We can now apply proportional feedback on $\mathbb{S}^2$, by using the configuration error vector $\mathbf{e}_\Gamma\colon \mathbb{S}^2 \times \mathbb{S}^2 \to \mathrm{T}_\Gamma\mathbb{S}^2$ given by

$$\mathbf{e}_\Gamma = \mathbf{\Gamma} \times \mathbf{\Gamma}_d, \tag{4.20}$$

and define the angular velocity error as

$$\mathbf{e}_\omega = \boldsymbol{\omega}^\perp - \mathbf{\Pi}_\Gamma^\perp \boldsymbol{\omega}_d = \mathbf{\Pi}_\Gamma^\perp(\boldsymbol{\omega} - \boldsymbol{\omega}_d) \in \mathrm{T}_\Gamma\mathbb{S}^2. \tag{4.21}$$

From Eq. (4.7) and Eq. (4.10), the derivative of $\mathbf{e}_\omega$ can be written as

$$\dot{\mathbf{e}}_\omega = \mathbf{\Pi}_\Gamma^\perp \left(\dot{\boldsymbol{\omega}} - \dot{\boldsymbol{\omega}}_d + \boldsymbol{\omega}^\perp \times (\boldsymbol{\omega}^\parallel - \mathbf{\Pi}_\Gamma^\parallel \boldsymbol{\omega}_d)\right) - \mathbf{\Pi}_\Gamma^\parallel(\boldsymbol{\omega}_d \times \mathbf{e}_\omega). \tag{4.22}$$

[1]When $\mathbf{\Gamma}_d$ is constant, we write $\Psi(\mathbf{\Gamma})$ and also remove $\mathbf{\Gamma}_d$ as an argument of the corresponding Lyapunov function.

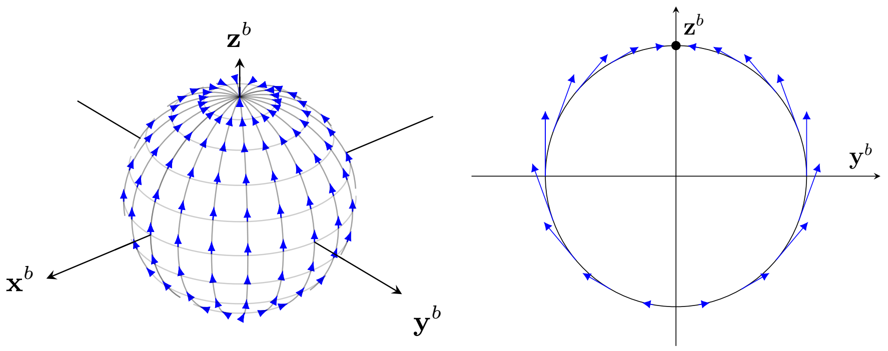

Figure 4.2: The left figure shows the gradient field defining $\mathbf{e}_\Gamma$ as induced by the potential Ψ for the reference example $\mathbf{\Gamma}_d = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$. Note the additional unstable equilibrium at the opposite direction of $\mathbf{\Gamma}_d$ in the circle section on the right figure.

The error terms $\mathbf{e}_\Gamma$ and $\mathbf{e}_\omega$ are also compatible in the sense that $\dot{\Psi} = \mathbf{e}_\omega^\top \mathbf{e}_\Gamma$, which will cancel with the proportional feedback term defined later when calculating the derivative of a Lyapunov function. The error vector $\mathbf{e}_\Gamma$ is geodesic in the sense that its direction defines an axis of rotation which connects Γ and $\mathbf{\Gamma}_d$ with the shortest possible curve on $\mathbb{S}^2$.

Remark 4.3. Other configuration error vectors (with corresponding potential functions) on $\mathbb{S}^2$ could be used in place of Eq. (4.20), without changing the general approach considered in this chapter. For instance, alternative error vectors that do not vanish when approaching $-\mathbf{\Gamma_d}$ (at the cost of being undefined at this point) can be found in [21], [30] and [150].

4.2.6 Control Objective

Let an orthogonal decomposition of the control input vector $\mathbf{u}$ be given by $\mathbf{u} = \mathbf{u}^\perp + \mathbf{u}^\parallel$, where $\mathbf{J}^{-1}\mathbf{G}(\boldsymbol{\omega}, \mathbf{v}_r)\mathbf{u}^\perp \in \mathrm{T}_\Gamma\mathbb{S}^2$, and $\mathbf{J}^{-1}\mathbf{G}(\boldsymbol{\omega}, \mathbf{v}_r)\mathbf{u}^\parallel \in \mathrm{N}_\Gamma\mathbb{S}^2$.

The control objective can be stated as follows:

Reduced-Attitude Tracking.

Design a state-feedback control law $\mathbf{u}^\perp$ such that $\mathbf{\Gamma}(t) \to \mathbf{\Gamma}_d(t)$ and $\mathbf{e}_\omega(t) \to \mathbf{0}$ as $t \to \infty$.

A special case of reduced-attitude tracking is the case when $\mathbf{\Gamma}_d$ is constant, so $\boldsymbol{\omega}_d = \mathbf{0}$ and $\mathbf{e}_\omega = \boldsymbol{\omega}^\perp$. We can formulate the following regulation problem:

Reduced-Attitude Regulation.

Design a state-feedback control law $\mathbf{u}^\perp$ such that $\mathbf{\Gamma}(t) \to \mathbf{\Gamma}_d$, with constant reference $\mathbf{\Gamma}_d$, and $\boldsymbol{\omega}^\perp(t) \to \mathbf{0}$ as $t \to \infty$.

The remaining degree of freedom provided by $\mathbf{u}^\parallel$ is in the nullspace of the orthogonal projection and as such does not interfere with the control of reduced attitude. It should be used for turn coordination and to stabilize the parallel component of angular velocity. We will discuss this after introducing the smooth geometric attitude controller and its stability properties.

4.3 Design of the smooth Geometric Attitude Controller

This section introduces the main result for reduced-attitude tracking control for fixed-wing UAVs, where a control law $\mathbf{u}^\perp$ is given. The design of a control law using the remaining degree of freedom, given by $\mathbf{u}^\parallel$ is treated in Section 4.3.1.

Proposition 4.1 (Reduced-Attitude Tracking)**:** Consider the reduced-attitude error dynamics defined by Eq. (4.2), Eq. (4.6), and Eq. (4.22), assuming $V_a \geq \underline{V}_a > 0$ and that the control effectiveness matrix $\mathbf{G}(\boldsymbol{\omega}, \mathbf{v}_r)$ has full rank. With $k_p \in \mathbb{R}_{>0}$ and $\mathbf{K}_d \in \mathcal{P}^3_+$, let a smooth tracking control law $\mathbf{u}^\perp$ satisfying $\mathbf{J}^{-1}\mathbf{G}(\boldsymbol{\omega}, \mathbf{v}_r)\mathbf{u}^\perp \in \mathrm{T}_\Gamma\mathbb{S}^2$ be given by

$$\begin{aligned}\mathbf{u}^\perp = \mathbf{G}^{-1}(\boldsymbol{\omega}, \mathbf{v}_r)\mathbf{J}\Big(&- k_p\mathbf{e}_\Gamma - \mathbf{\Pi}^\perp_\Gamma\mathbf{K}_d\mathbf{e}_\omega - \mathbf{\Pi}^\perp_\Gamma\mathbf{J}^{-1}\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r) \\ &-\boldsymbol{\omega}^\perp \times (\boldsymbol{\omega}^\parallel - \mathbf{\Pi}^\parallel_\Gamma\boldsymbol{\omega}_d) + \mathbf{\Pi}^\perp_\Gamma\dot{\boldsymbol{\omega}}_d\Big),\end{aligned} \tag{4.23}$$

with $\mathbf{v}_r$ treated as a bounded, time-varying exogenous signal. The closed-loop system then satisfies the following properties:

(i) There exist two equilibrium solutions given by $(\mathbf{\Gamma}, \mathbf{e}_\omega) = (\pm\mathbf{\Gamma}_d, 0)$.

(ii) The desired equilibrium $(\mathbf{\Gamma}_d, 0)$ is exponentially stable, with region of exponential convergence given by

$$\Psi(\mathbf{\Gamma}(0), \mathbf{\Gamma}_d(0)) \leq \overline{\Psi} \tag{4.24}$$

$$\|\mathbf{e}_\omega(0)\| \leq \sqrt{2k_p\left(\overline{\Psi} - \Psi(\mathbf{\Gamma}(0), \mathbf{\Gamma}_d(0))\right)}, \tag{4.25}$$

for some $\overline{\Psi} < 2$, where 2 is the maximum value of Ψ, attained at $\mathbf{\Gamma} = -\mathbf{\Gamma}_d$.

(iii) The additional undesired equilibrium $(-\mathbf{\Gamma}_d, 0)$ is unstable.

(iv) Additionally, if $\boldsymbol{\omega}_d = 0$, the desired equilibrium $(\mathbf{\Gamma}_d, 0)$ is almost globally asymptotically stable.

Proof: The proof can be split into three parts. First, we show that the equilibria of the closed-loop error system are $(\mathbf{\Gamma}, e_\omega) = (\pm\mathbf{\Gamma}_d, 0)$. Then we proof local exponential stability based on derived bounds of the potential function Ψ, which define the region of exponential convergence. Then we show that the antipodal of the desired equilibrium is unstable by studying its local properties.

Differentiating Eq. (4.20), applying the identity Eq. (2.6) and combining with Eq. (4.2), Eq. (4.22), Eq. (4.23) gives the non-autonomous closed-loop error system

$$\dot{\mathbf{e}}_\Gamma = -\mathbf{S}(\boldsymbol{\omega}_d)\mathbf{e}_\Gamma - \mathbf{S}(\mathbf{\Gamma}_d)\mathbf{S}(\Gamma)\mathbf{e}_\omega \tag{4.26}$$

$$\dot{\mathbf{e}}_\omega = -k_p\mathbf{e}_\Gamma - \mathbf{\Pi}^\perp_\Gamma\mathbf{K}_d\mathbf{e}_\omega - \mathbf{\Pi}^\parallel_\Gamma(\boldsymbol{\omega}_d \times \mathbf{e}_\omega), \tag{4.27}$$

which gives the equilibrium condition

$$\begin{bmatrix} -\mathbf{S}(\boldsymbol{\omega}_d) & -\mathbf{S}(\boldsymbol{\Gamma}_d)\mathbf{S}(\boldsymbol{\Gamma}) \\ -k_p\mathbf{I}_3 & -\mathbf{S}^2(\boldsymbol{\Gamma})\mathbf{K}_d\mathbf{S}^2(\boldsymbol{\Gamma}) - \boldsymbol{\Gamma}\boldsymbol{\Gamma}^\top\mathbf{S}(\boldsymbol{\omega}_d) \end{bmatrix} \begin{bmatrix} \mathbf{e}_\Gamma \\ \mathbf{e}_\omega \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{4.28}$$

where we have used the fact that $-\mathbf{S}^2(\boldsymbol{\Gamma})\mathbf{e}_\omega = \mathbf{e}_\omega$. For Eq. (4.28) to be satisfied for all t, where the time-dependence is implicit through $\boldsymbol{\Gamma}_d(t), \boldsymbol{\omega}_d(t)$, we get $\mathbf{e}_\Gamma = \mathbf{e}_\omega = \mathbf{0}$. Note that, when $\boldsymbol{\omega}_d = \mathbf{0}$, the matrix above is rank-deficient, with $\mathbf{v} = \left[\mathbf{0}^\top \;\; \mathbf{w}^\top\right]^\top$ as a basis of the nullspace, where w is parallel to $\boldsymbol{\Gamma}$. But $\mathbf{e}_\omega$ lies in $\mathrm{T}_\Gamma\mathbb{S}^2$. Thus, equilibrium solutions are given by $(\boldsymbol{\Gamma}, \mathbf{e}_\omega) = (\pm\boldsymbol{\Gamma}_d, 0)$, which shows (i).

For showing (ii), consider the Lyapunov-like function $V\colon \mathrm{T}\mathbb{S}^2 \times \mathbb{S}^2 \to \mathbb{R}_{\geq 0}$

$$V_1(\boldsymbol{\Gamma}, \mathbf{e}_\omega, \boldsymbol{\Gamma}_d) = k_p\Psi(\Gamma, \Gamma_d) + \frac{1}{2}\mathbf{e}_\omega^\top\mathbf{e}_\omega. \tag{4.29}$$

Differentiating along closed-loop trajectories of Eq. (4.27) gives

$$\begin{aligned} \dot{V}_1 &= k_p\mathbf{e}_\omega^\top\mathbf{e}_\Gamma + \mathbf{e}_\omega^\top\left(-k_p\mathbf{e}_\Gamma - \boldsymbol{\Pi}_\Gamma^\perp\mathbf{K}_d\mathbf{e}_\omega - \boldsymbol{\Pi}_\Gamma^\parallel(\boldsymbol{\omega}_d \times \mathbf{e}_\omega)\right) &(4.30)\\ &= -\mathbf{e}_\omega^\top\mathbf{K}_d\mathbf{e}_\omega \leq -\lambda_{\min}(\mathbf{K}_d)\|\mathbf{e}_\omega\|^2 \leq 0, &(4.31) \end{aligned}$$

where the last term in Eq. (4.30) disappears since $\mathbf{e}_\omega \in \mathrm{T}_\Gamma\mathbb{S}^2$, and we have used the property $\mathbf{e}_\omega^\top\boldsymbol{\Pi}_\Gamma^\perp = (\boldsymbol{\Pi}_\Gamma^\perp\mathbf{e}_\omega)^\top = \mathbf{e}_\omega$. For initial conditions satisfying Eq. (4.24), Eq. (4.25), we get $V_1(t_0) \leq k_p\overline{\Psi}$. Since $V_1(t)$ is non-increasing, we get:

$$k_p\Psi(\boldsymbol{\Gamma}(t), \boldsymbol{\Gamma}_d(t)) \leq V_1(t) \leq V_1(t_0) \leq k_p\overline{\Psi}, \tag{4.32}$$

which means that $\Psi(\boldsymbol{\Gamma}(t), \boldsymbol{\Gamma}_d(t)) \leq \overline{\Psi}$. Using the bound $\overline{\Psi}$, let a sublevel set for Ψ be $L_2 = \left\{\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d \in \mathbb{S}^2\colon \Psi(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d) \leq \overline{\Psi}\right\}$. In L_2 we can then bound Ψ by

$$\frac{1}{2}\|\mathbf{e}_\Gamma\|^2 \leq \Psi(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d) \leq \frac{1}{2 - \overline{\Psi}}\|\mathbf{e}_\Gamma\|^2. \tag{4.33}$$

Now, consider the Lyapunov function candidate

$$V_2(\boldsymbol{\Gamma}, \mathbf{e}_\omega, \boldsymbol{\Gamma}_d) = V_1 + c\mathbf{e}_\omega^\top\mathbf{e}_\Gamma. \tag{4.34}$$

Using Eq. (4.33), we can derive upper and lower bounds

$$\frac{1}{2}\mathbf{z}^\top\mathbf{M}_1\mathbf{z} \leq V_2 \leq \frac{1}{2}\mathbf{z}^\top\mathbf{M}_2\mathbf{z}, \tag{4.35}$$

where $\mathbf{z} = [\|\mathbf{e}_\Gamma\| \;\; \|\mathbf{e}_\omega\|]^\top$ and

$$\mathbf{M}_1 = \begin{bmatrix} k_p & -c \\ -c & 1 \end{bmatrix}, \qquad \mathbf{M}_2 = \begin{bmatrix} \frac{2k_p}{2-\overline{\Psi}} & c \\ c & 1 \end{bmatrix}. \tag{4.36}$$

If $c < \sqrt{k_p}$, both $\mathbf{M}_1$ and $\mathbf{M}_2$ are positive definite.

Differentiating V_2 along the closed-loop trajectories gives

$$\dot{V}_2 = -\mathbf{e}_\omega^\top\mathbf{K}_d\mathbf{e}_\omega + c\dot{\mathbf{e}}_\omega^\top\mathbf{e}_\Gamma + c\mathbf{e}_\omega^\top\dot{\mathbf{e}}_\Gamma. \tag{4.37}$$

The cross terms can be bounded as follows:

$$\|\mathbf{e}_\omega^\top \dot{\mathbf{e}}_\Gamma\| \leq \|\mathbf{e}_\omega\|\,\|\boldsymbol{\omega}_d\|\,\|\mathbf{e}_\Gamma\| + \|\mathbf{e}_\omega\|^2 \leq B\|\mathbf{e}_\omega\|\|\mathbf{e}_\Gamma\| + \|\mathbf{e}_\omega\|^2 \tag{4.38}$$

$$\|\mathbf{e}_\Gamma^\top \dot{\mathbf{e}}_\omega\| \leq -k_p\|\mathbf{e}_\Gamma\|^2 + \lambda_{\max}(\mathbf{K}_d)\|\mathbf{e}_\Gamma\|\|\mathbf{e}_\omega\|, \tag{4.39}$$

which leads to

$$\dot{V}_2 = -\mathbf{e}_\omega^\top \mathbf{K}_d \mathbf{e}_\omega + c\dot{\mathbf{e}}_\omega^\top \mathbf{e}_\Gamma + c\mathbf{e}_\omega^\top \dot{\mathbf{e}}_\Gamma \leq -z^\top \mathbf{M}_3 z, \tag{4.40}$$

where the matrix $\mathbf{M}_3$ is given by

$$\mathbf{M}_3 = \begin{bmatrix} ck_p & -\frac{c(B+\lambda_{\max}(\mathbf{K}_d)))}{2} \\ -\frac{c(B+\lambda_{\max}(\mathbf{K}_d)))}{2} & \lambda_{\min}(\mathbf{K}_d) - c \end{bmatrix} \tag{4.41}$$

If c is chosen to satisfy

$$c < \min\left\{\sqrt{k_p}, \frac{4k_p\lambda_{\min}(\mathbf{K}_d)}{4k_p + (B+\lambda_{\max}(\mathbf{K}_d))^2}\right\}, \tag{4.42}$$

then $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_3$ are positive definite.

By following similar arguments as in the proof of [95, Theorem 4.10], we get that $V_2(t)$ and $\|\mathbf{z}(t)\|$ converge exponentially to zero, which in turn means that $(\boldsymbol{\Gamma}(t), \mathbf{e}_\omega(t))$ converges exponentially to $(\boldsymbol{\Gamma}_d(t), \mathbf{0})$, with the region of exponential convergence given by Eq. (4.24) and Eq. (4.25). This shows (ii).

To show that the undesired equilibrium is unstable, define

$$W = 2k_p - V_2 \geq -\frac{1}{2}\|\mathbf{e}_\omega\|^2 - c\|\mathbf{e}_\omega\|\|\mathbf{e}_\Gamma\| + k_p(2 - \Psi(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d)). \tag{4.43}$$

At the undesired equilibrium $(-\boldsymbol{\Gamma}_d, \mathbf{0})$, we have $W = 0$, and $\dot{W} = -\dot{V}_2$ is positive definite from Eq. (4.40). Now consider $\boldsymbol{\Gamma}$ arbitrarily close to $-\boldsymbol{\Gamma}_d$. In this case, the term $2 - \Psi(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d)$ is positive, and we can choose $\|\mathbf{e}_\omega\|$ sufficiently small such that $W > 0$ and $\dot{W} > 0$. By [95, Theorem 4.3], the equilibrium $(-\boldsymbol{\Gamma}_d, 0)$ is unstable, which show (iii).

When $\boldsymbol{\omega}_d = \mathbf{0}$, Eq. (4.31) reduces to $\dot{V}_1 = -(\boldsymbol{\omega}^\perp)^\top \mathbf{K}_d \boldsymbol{\omega}^\perp \leq 0$, so the set given by

$$\Omega \triangleq \{(\boldsymbol{\Gamma}, \boldsymbol{\omega}^\perp) \in \mathbb{S}^2 \times \mathrm{T}_\Gamma\mathbb{S}^2 \colon V_1(\boldsymbol{\Gamma}(t), \boldsymbol{\omega}^\perp(t)) \leq V_1(\boldsymbol{\Gamma}(t_0), \boldsymbol{\omega}^\perp(t_0))\} \tag{4.44}$$

is positively invariant. Since $\mathbb{S}^2$ is compact, all sublevel sets of V_1 are compact, which means that the set Ω is compact. Let E be set of points in Ω where $\dot{V}_1 = 0$. In E, $\boldsymbol{\omega}^\perp = \mathbf{0}$, which when inserted into Eq. (4.27) and using Eq. (4.20) leads to $\boldsymbol{\Gamma} = \pm\boldsymbol{\Gamma}_d$. By LaSalle's invariance principle [95, Theorem 4.4], every solution starting in Ω then converges asymptotically to one of the equilibrium solutions $(\pm\boldsymbol{\Gamma}_d, \mathbf{0})$. Local asymptotic stability of the desired equilibrium point, as well as the instability of the undesired equilibrium follows from (ii) and (iii). To establish almost global asymptotic stability of the desired equilibrium we will study the local structure of the undesired equilibrium.

Similarly to Eq. (4.17), let a perturbation of the equilibrium solution $(\boldsymbol{\Gamma}(t), \boldsymbol{\omega}^{\perp}(t)) = (\boldsymbol{\Gamma}_e, \mathbf{0})$ be given in terms of a perturbation parameter $\boldsymbol{\epsilon}$ as $(\boldsymbol{\Gamma}_\epsilon(t,\boldsymbol{\epsilon}), \boldsymbol{\omega}_\epsilon^{\perp}(t,\boldsymbol{\epsilon})) = (e^{-\boldsymbol{\epsilon}\mathbf{S}(\boldsymbol{\eta}(t))}\boldsymbol{\Gamma}_e, \boldsymbol{\epsilon}\delta\boldsymbol{\omega}(t))$, which satisfies $\boldsymbol{\eta}(t)\cdot\boldsymbol{\Gamma}_e = \delta\boldsymbol{\omega}(t)\cdot\boldsymbol{\Gamma}_e = \mathbf{0}$ for all t. Now, consider the perturbed equations of motion Eq. (4.6), Eq. (4.27) given by

$$\dot{\boldsymbol{\Gamma}}_\epsilon(t,\boldsymbol{\epsilon}) = \boldsymbol{\Gamma}_\epsilon(t,\boldsymbol{\epsilon}) \times \boldsymbol{\omega}_\epsilon^{\perp}(t,\boldsymbol{\epsilon}) \tag{4.45}$$

$$\dot{\boldsymbol{\omega}}_\epsilon^{\perp}(t,\boldsymbol{\epsilon}) = -k_p\boldsymbol{\Gamma}_\epsilon(t,\boldsymbol{\epsilon}) \times \boldsymbol{\Gamma}_d - \mathbf{K}_d\boldsymbol{\omega}_\epsilon^{\perp}(t,\boldsymbol{\epsilon}) + \boldsymbol{\Gamma}_\epsilon(t,\boldsymbol{\epsilon})\boldsymbol{\Gamma}_\epsilon^\top(t,\boldsymbol{\epsilon})\mathbf{K}_d\boldsymbol{\omega}_\epsilon^{\perp}(t,\boldsymbol{\epsilon}).$$

Differentiating both sides with respect to $\boldsymbol{\epsilon}$ and inserting $\boldsymbol{\epsilon} = \mathbf{0}$ gives the linearized set of equations $\dot{\mathbf{x}} = \mathbf{A}(\boldsymbol{\Gamma}_e)x$, where $\mathbf{x} = [\boldsymbol{\eta}^\top \;\; \delta\boldsymbol{\omega}^\top]^\top$. For $\boldsymbol{\Gamma}_e = -\boldsymbol{\Gamma}_d$, we get

$$\mathbf{A}(-\boldsymbol{\Gamma}_d) \triangleq \mathbf{A} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 \\ -k_p\mathbf{S}^2(\boldsymbol{\Gamma}_d) & -\mathbf{S}^2(\boldsymbol{\Gamma}_d)\mathbf{K}_d\mathbf{S}^2(\boldsymbol{\Gamma}_d) \end{bmatrix}, \tag{4.46}$$

where the relation $-\mathbf{S}^2(\boldsymbol{\Gamma}_d)\delta\boldsymbol{\omega} = \delta\boldsymbol{\omega}$ has been used to add the last factor in the lower right element of the matrix.

The state space has dimension six, but in reality, the system evolves on a four-dimensional subspace according to the constraints

$$\mathbf{C}\mathbf{x} = \begin{bmatrix} \boldsymbol{\Gamma}_e^\top & \mathbf{0}_{1\times3} \\ \mathbf{0}_{1\times3} & \boldsymbol{\Gamma}_e^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta} \\ \delta\boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{4.47}$$

which is respected by the linearized dynamics, in the sense that $\mathbf{CA} = \mathbf{0}$.

If decomposing eigenvectors $\mathbf{v}_i$ of $\mathbf{A}$ into $\mathbf{v}_i = [\mathbf{v}_{i1}^\top \;\; \mathbf{v}_{i2}^\top]^\top$, it follows from the equation $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ that eigenvalue-eigenvector pairs of $\mathbf{A}$ need to satisfy

$$\mathbf{v}_{i2} = \lambda_i\mathbf{v}_{i1} \tag{4.48}$$

$$-k_p\mathbf{S}^2(\boldsymbol{\Gamma}_d)\mathbf{v}_{i1} - \mathbf{S}^2(\boldsymbol{\Gamma}_d)\mathbf{K}_d\mathbf{S}^2(\boldsymbol{\Gamma}_d)\mathbf{v}_{i2} = \lambda_i\mathbf{v}_{i2}. \tag{4.49}$$

Inserting Eq. (4.48) into Eq. (4.49) and pre-multiplying with the complex conjugate transpose $\bar{\mathbf{v}}_{i1}^\top$ of $\mathbf{v}_{i1}$ gives

$$a\lambda^2 + b\lambda - c = 0, \tag{4.50}$$

where

$$a = \bar{\mathbf{v}}_{i1}^\top\mathbf{v}_{i1} > 0 \tag{4.51}$$

$$b = \bar{\mathbf{v}}_{i1}^\top[-\mathbf{S}^2(\boldsymbol{\Gamma}_d)]\mathbf{K}_d[-\mathbf{S}^2(\boldsymbol{\Gamma}_d)]\mathbf{v}_{i1} \geq 0 \tag{4.52}$$

$$c = k_p\bar{\mathbf{v}}_{i1}^\top[-\mathbf{S}^2(\boldsymbol{\Gamma}_d)]\mathbf{v}_{i1} \geq 0, \tag{4.53}$$

since the matrix $-\mathbf{S}^2(\boldsymbol{\Gamma}_d)$ is positive semi-definite. The coefficients b and c are only (simultaneously) zero when $-\mathbf{S}^2(\boldsymbol{\Gamma}_d)\mathbf{v}_{i1} = \mathbf{0}$, i.e. when $\mathbf{v}_{i1}$ has the form $\mathbf{v}_{i1} = \mathbf{z}_1\boldsymbol{\Gamma}_d$ for some $z_1 \in \mathbb{C}$. In this case, Eq. (4.50) reduces to $a\lambda^2 = 0$, so $\lambda_1 = 0$ is an eigenvalue of $\mathbf{A}$ with algebraic multiplicity two, corresponding to the eigenvector $\mathbf{v}_1 = [z_1\boldsymbol{\Gamma}_d^\top \;\; 0^\top]^\top$. However, since $\mathbf{A}$ has rank five, the geometric multiplicity of λ_1 is one. To get a full Jordan basis, we choose the generalized eigenvector $\mathbf{v}_2 = [z_2\boldsymbol{\Gamma}_d^\top \;\; z_1\boldsymbol{\Gamma}_d^\top]^\top$, which satisfies $(\mathbf{A} - \lambda_1\mathbf{I})\mathbf{v}_2 = \mathbf{A}\mathbf{v}_2 = \mathbf{v}_1$, for $z_2 \in \mathbb{C}$.

The solutions of the linearized system defined by Eq. (4.46) can be written in terms of its Jordan form as

$$\mathbf{x}(t) = c_1 e^{\lambda_1 t}\mathbf{v}_1 + c_2 e^{\lambda_1 t}(\mathbf{v}_1 t + \mathbf{v}_2) + \sum_{i=3}^{6} c_i \mathbf{h}_i(t), \tag{4.54}$$

where the functions $\mathbf{h}_i(t)$ depend on the vectors $\mathbf{v}_j$, $j \in \{1, \dots, 6\}$, the eigenvalues λ_k, $k = \{2, 3, 4, 5\}$ and their multiplicities. The constants c_i depend on the initial condition $\mathbf{x}(0) = \sum_{i=1}^{6} c_i \mathbf{v}_i$, which satisfies $\mathbf{C}\mathbf{x}(0) = \mathbf{0}$. Since the vectors $\mathbf{v}_1, \mathbf{v}_2$ do not satisfy the constraints Eq. (4.47), $c_1 = c_2 = 0$, so the solution $\mathbf{x}(t)$ does not depend on λ_1.

Since $\mathbf{A} \in \mathbb{R}^{6\times 6}$ is a rank-five matrix, we know that no other eigenvectors are parallel to $\mathbf{v}_1$, so for all remaining eigenvector pairs, $a, b, c > 0$. Given that Eq. (4.50) has two solutions, we know from the quadratic formula that from the remaining eigenvalues λ_k, $k = \{2, 3, 4, 5\}$, two are positive, and two are negative. This confirms that the undesired equilibrium point $(\boldsymbol{\Gamma}, \boldsymbol{\omega}^{\perp}) = (-\boldsymbol{\Gamma}_d, \mathbf{0})$ is unstable. Moreover, the stable eigenspace corresponding to the two negative eigenvalues is the tangent space to a two-dimensional stable invariant manifold $\mathcal{M}$, where all trajectories starting in $\mathcal{M}$ converge to the undesired equilibrium point [72]. Since the zero eigenvalue has no influence on the solution, all trajectories converging to the undesired equilibrium lie in $\mathcal{M}$. We conclude that all trajectories except those starting in $\mathcal{M}$ converge to the desired equilibrium. Since the dimension of $\mathcal{M}$ is two, it has measure zero in the state space $\mathrm{T}\mathbb{S}^2$, and we say that the domain of attraction of the desired equilibrium point is almost global, which shows (iv). ■

For almost all $\Psi(\boldsymbol{\Gamma}(t_0), \boldsymbol{\Gamma}_d(t_0)), \mathbf{e}_\omega(t_0)$ (excluding $\boldsymbol{\Gamma}(t_0) = -\boldsymbol{\Gamma}_d(t_0)$), some k_p can be chosen such that Eq. (4.24), Eq. (4.25) is satisfied. The equilibrium $(\boldsymbol{\Gamma}_d, 0)$ is therefore said to be *almost semi-globally exponentially stable* [107]. By exploiting passivity-properties in the Lyapunov design, the term $-\boldsymbol{\omega}^{\perp} \times (\boldsymbol{\omega}^{\parallel} - \boldsymbol{\Pi}_{\Gamma}^{\parallel}\boldsymbol{\omega}_d)$ in Eq. (4.23) could be replaced by $-\boldsymbol{\Pi}_{\Gamma}^{\perp}\boldsymbol{\omega}_d \times (\boldsymbol{\omega}^{\parallel} - \boldsymbol{\Pi}_{\Gamma}^{\parallel}\boldsymbol{\omega}_d)$, without changing $\dot{V}$. This would also remove the seemingly unneeded cross product term in Eq. (4.55). However, this would give a closed-loop system that depends on $\boldsymbol{\omega}^{\parallel}$, and invalidate several arguments used in the proof.

A notable property of the control law Eq. (4.23) is that since the control effectiveness matrix $\mathbf{G}(\mathbf{v}_r)$ given by Eq. (2.49) contains a factor of V_a^2, the inverse matrix $\mathbf{G}^{-1}(\mathbf{v}_r)$ contains a factor of $1/V_a^2$. This means that the control law includes airspeed scaling. Also note that instead of compensating for the entire drift term $\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r)$, only the orthogonal projection is compensated for.

A part of the proof is inspired by Lee [105], where a tracking controller for a double integrator system on $\mathbb{S}^2$ is presented. However, Lee [105] considers an inertial frame representation of reduced attitude, as opposed to Eq. (4.3), which is defined in the body-fixed frame. Also, no dynamics, or parallel component of the angular velocity is considered. In addition to compensating for the dynamics, compared to [105], the controller Eq. (4.23) allows a matrix gain $\mathbf{K}_d$, projects the feedforward term $\dot{\boldsymbol{\omega}}_d$ to $\mathrm{T}_{\Gamma}\mathbb{S}^2$, and adds a term $-\boldsymbol{\omega}^{\perp} \times \boldsymbol{\omega}^{\parallel}$ to compensate for the "coriolis" term that appears when $\boldsymbol{\omega}^{\parallel}$ is nonzero.

In the special case of regulation, where $\boldsymbol{\omega}_d = \mathbf{0}$, and $\mathbf{e}_\omega = \boldsymbol{\omega}^{\perp}$, the control law Eq. (4.23) reduces to

$$\mathbf{u}^{\perp} = \mathbf{G}^{-1}(\boldsymbol{\omega}, \mathbf{v}_r)\mathbf{J}\bigg(- k_p\mathbf{e}_\Gamma - \boldsymbol{\Pi}_\Gamma^{\perp}\mathbf{K}_d\boldsymbol{\omega}^{\perp}\boldsymbol{\Pi}_\Gamma^{\perp}\mathbf{J}^{-1}\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r) - \boldsymbol{\omega}^{\perp} \times \boldsymbol{\omega}^{\parallel}\bigg). \tag{4.55}$$

The closed-loop system in this case is autonomous. This means that LaSalle's invariance theorem [95] can be applied. Inspired by the methodology presented for the 3-D pendulum in [30], this can be combined with local analysis of the linearized closed-loop dynamics at the equilibria to show almost global asymptotic stability. However, the linearized dynamics in [30] evolve on $\mathbb{R}^5$, since the state space includes the full angular velocity $\boldsymbol{\omega}$. In [106], a closed-loop 3-D pendulum system is analyzed, with angular velocity in $\mathrm{T}_\Gamma\mathbb{S}^2$ and with linearization evolving on $\mathbb{R}^4$, but only for very specific numerical values of the controller gains. The proof of almost global asymptotic stability follows [30], but is adjusted to use a linearization on $\mathbb{R}^4$ instead of $\mathbb{R}^5$, inspired by [106], but done in full generality. In addition, a matrix gain $\mathbf{K}_d$ is used instead of a scalar.

4.3.1 Turn Coordination

The control law defined by Eq. (4.23) does not inject damping about the axis defined by $\boldsymbol{\Gamma}$. The control $\mathbf{u}^{\parallel}$ can be utilized to do this. Let a reference for the angular velocity about the vertical axis of the inertial frame be given by the equation for a coordinated turn with zero sideslip [9]:

$$\boldsymbol{\omega}_d^{\parallel} = \dot{\psi}_d\boldsymbol{\Gamma}, \qquad \dot{\psi}_d = \frac{g}{V_a}\tan(\phi). \tag{4.56}$$

Note that care needs to be taken to avoid the singularity at $\Gamma_3 = 0$, corresponding to $\phi = \pm\pi/2$, either by constraining the value of ϕ used in Eq. (4.56), or by using the reference instead. A controller that adds damping about $\boldsymbol{\Gamma}$ without interfering with the banked turn maneuver is then given by

$$\mathbf{u}^{\parallel} = \mathbf{G}^{-1}(\boldsymbol{\omega}, \mathbf{v}_r)\mathbf{J}\left(-k_{tc}(\boldsymbol{\omega}^{\parallel} - \boldsymbol{\omega}_d^{\parallel}) - \boldsymbol{\Pi}_\Gamma^{\parallel}\mathbf{J}^{-1}\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r)\right), \tag{4.57}$$

where $k_{tc} \in \mathbb{R}_{>0}$ is a scalar design parameter.

As an alternative to Eq. (4.57), for some $k_\beta \in \mathbb{R}_{>0}$, consider the following control law, which is designed to drive the sideslip angle to zero:

$$\mathbf{u}^{\parallel} = \mathbf{G}^{-1}(\boldsymbol{\omega}, \mathbf{v}_r)\mathbf{J}\,\boldsymbol{\Pi}_\Gamma^{\perp}(k_\beta\beta\mathbf{e}_3). \tag{4.58}$$

4.3.2 Comparison With Controller Based on Euler Angles

In this section, the geometric controller presented in Section 4.3 will be compared to a controller based on Euler angles. For $\mathbf{K}_\omega \in \mathcal{P}_+^3$, consider the cascaded dynamic inversion based controller

$$\mathbf{u} = \mathbf{G}^{-1}(\boldsymbol{\omega}, \mathbf{v}_r)\mathbf{J}\bigg(- \mathbf{K}_\omega(\boldsymbol{\omega} - \bar{\boldsymbol{\omega}}_d) - \mathbf{J}^{-1}\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r, \delta_t)\bigg), \tag{4.59}$$

where the bar in $\bar{\boldsymbol{\omega}}_d$ is introduced to distinguish it from $\boldsymbol{\omega}_d$ in Eq. (4.9). The desired angular velocity is computed using Eq. (4.56) and linear state feedback from the roll and pitch regulation errors $\tilde{\phi} \triangleq \phi - \phi_d$, $\tilde{\theta} \triangleq \theta - \theta_d$ as follows:

$$\bar{\boldsymbol{\omega}}_d = \mathbf{T}^{-1}(\phi, \theta) \begin{bmatrix} -k_\phi \tilde{\phi} \\ -k_\theta \tilde{\theta} \\ \frac{g}{V_a} \tan(\phi) \end{bmatrix} \tag{4.60}$$

where $k_\phi, k_\theta \in \mathbb{R}_{>0}$, and $\mathbf{T}^{-1}(\phi, \theta)$ is the inverse of the Euler angle transformation matrix, given by [9]

$$\mathbf{T}^{-1}(\phi, \theta) = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{4.61}$$

for $\theta \neq \pm\pi/2$. Note that the third column is $\boldsymbol{\Gamma}$ (see Eq. (4.4)).

Remark 4.4. Apart from the dynamic inversion term, this controller has similar structure as the control architecture used in the PX4 open source autopilot [128].

To compare the geometric controller from Section 4.3 with the Euler angle controller Eq. (4.59)-Eq. (4.61), we consider the regulation case with $\boldsymbol{\omega}_d = 0$, and set $\mathbf{K}_d = \mathbf{K}_\omega = k_d \mathbf{I}_3$, $k_\theta = k_\phi = k_p/k_d$, and Eq. (4.57) is used for $\mathbf{u}^{\parallel}$ with $k_{tc} = k_d$.

For the geometric controller, the closed-loop angular velocity dynamics then become

$$\dot{\boldsymbol{\omega}} = -k_p \mathbf{e}_\Gamma - k_d \boldsymbol{\omega} + k_d \boldsymbol{\omega}_d^{\parallel}, \tag{4.62}$$

while for the controller based on Euler angles, the closed-loop dynamics are

$$\dot{\boldsymbol{\omega}} = -k_p \mathbf{e}_{\theta\phi} - k_d \boldsymbol{\omega} + k_d \boldsymbol{\omega}_d^{\parallel}, \tag{4.63}$$

where we have introduced the following error vector:

$$\mathbf{e}_{\theta\phi} \triangleq \begin{bmatrix} \tilde{\phi} & \frac{k_\theta}{k_\phi} \tilde{\theta} \cos(\phi) & -\frac{k_\theta}{k_\phi} \tilde{\theta} \sin(\phi) \end{bmatrix}^\top . \tag{4.64}$$

The only difference between Eq. (4.62) and Eq. (4.63) lies in the proportional error vectors. By calculating $\boldsymbol{\Gamma} \cdot \mathbf{e}_{\theta\phi} = -\tilde{\phi} \sin(\theta) \neq 0$, we see that $\mathbf{e}_{\theta\phi} \notin \mathrm{T}_\Gamma \mathbb{S}^2$, so the proportional action has a different direction than the geodesic direction defined by $\mathbf{e}_\Gamma$. The error vectors also have different magnitude, but this can be changed using a different potential function (see Remark 4.3). In the numerical simulation study of Section 4.4 we will normalize the magnitude of the error vectors by redefining $\mathbf{e}_\Gamma$ as $\mathbf{e}'_\Gamma = \|\mathbf{e}_{\theta\phi}\| \cdot \mathbf{e}_\Gamma / \|\mathbf{e}_\Gamma\|$, which enables us to compare the controllers on equal grounds.

4.4 Simulation Study of the smooth Geometric Attitude Controller

This section presents some simulation results using a model of the Aerosonde UAV [9] with a simple PI controller for airspeed and a constant reference of 35 m/s. In the tracking example, Eq. (4.58) is used for $\mathbf{u}^{\parallel}$, while Eq. (4.57) is used in the regulation case. The angular velocity is initialized to zero in both cases, while the controller parameters are set to $k_p = 9.5$, $\mathbf{K}_d = 8\mathbf{I}_3$ and $k_\beta = 10$.

4.4.1 Tracking

Consider a tracking scenario, where a trajectory $(\mathbf{\Gamma}_d(t),\ \boldsymbol{\omega}_d(t), \dot{\boldsymbol{\omega}}_d(t))$ has been generated using Eq. (4.4), $\phi_d(t) = 60\frac{\pi}{180}\cos(0.1 \cdot 2\pi t)$, $\theta_d(t) = 30\frac{\pi}{180}\cos(0.08 \cdot 2\pi t)$ and their analytical first and second order derivatives. Initial reduced attitude is set using $\phi(0) = -70°$ and $\theta(0) = -30°$. Fig. 4.3 shows that reduced attitude, visualized using roll and pitch angles, converge to the desired trajectory from large initial errors, while the angular velocity error goes to zero. Angle of attack, sideslip and control surface deflection angles, which attain reasonable values throughout the maneuver, are displayed in Fig. 4.3.

4.4.2 Regulation

Now consider a regulation case, where $\boldsymbol{\omega}_d = 0$. The constant reference is generated using $\phi_d = 60°$ and $\theta_d = 30°$. Initial roll and yaw angles are set to zero, while $\theta(0)$ is calculated using a trim routine. As explained in Section 4.3.2, the magnitude of the error vector Eq. (4.20) is scaled for comparison with the Euler angle controller. Fig. 4.4 shows that the UAV performs a banked turn maneuver with approximately constant turn rate, and roll and pitch angles converge in both cases. For this specific maneuver, the geometric controller seem to have a slightly faster response in pitch. The difference can be more clearly understood by looking at Fig. 4.5. The response of the geometric controller is shown to make the UAV take the shortest path between $\mathbf{\Gamma}$ and $\mathbf{\Gamma}_d$, while the controller based on Euler angles does not. Fig. 4.4 might indicate that this makes the geometric controller spend less control energy. However, the sideslip angle is smaller, which also reduces the magnitude of the compensated drift term $\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r)$. Further investigation should compare the two controllers when $\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r)$ is unknown, and integral action is implemented.

4.5 Design of the Hybrid Geometric Attitude Controller

This section extends the smooth proportional feedback of the controller design to hybrid feedback to render the reduced attitude reference globally exponentially stable. The essential ingredient is the use of synergistic potential functions [125] that enable switching of the proportional action in the vicinity of the undesired equilibrium where the gradient of the nominal potential function vanishes.

We begin by introducing the framework presented in [64] and let the hybrid system be defined as

$$(\dot{\boldsymbol{\xi}}, \dot{q}) \in \mathcal{F}(\boldsymbol{\xi}, q), \ \ \boldsymbol{\xi} \in \mathcal{C} \tag{4.65}$$

$$(\boldsymbol{\xi}^+, q^+) = \mathcal{G}(\boldsymbol{\xi}, q), \ \ \boldsymbol{\xi} \in \mathcal{D}, \tag{4.66}$$

with state $\boldsymbol{\xi} \in \mathbb{R}^n$. When the state is inside the flow set $\mathcal{C} \subset \mathbb{R}^n$, its continuous motion is governed by the set-valued flow map $\mathcal{F} : \mathbb{R}^n \times Q \rightrightarrows \mathbb{R}^n \times Q$. Complementary, when the state is inside the jump set $\mathcal{D} \subset \mathbb{R}^n$ it evolves in the form of discrete jumps with its dynamics governed by the jump map $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}^n$. The variable $q \in Q$ is a discrete logic state. The hybrid time domain (t, i) consists of continuous time $t \in \mathbb{R}_{\geq 0}$ and jump time $i \in \mathbb{N}$. In the stability analysis of this section we will

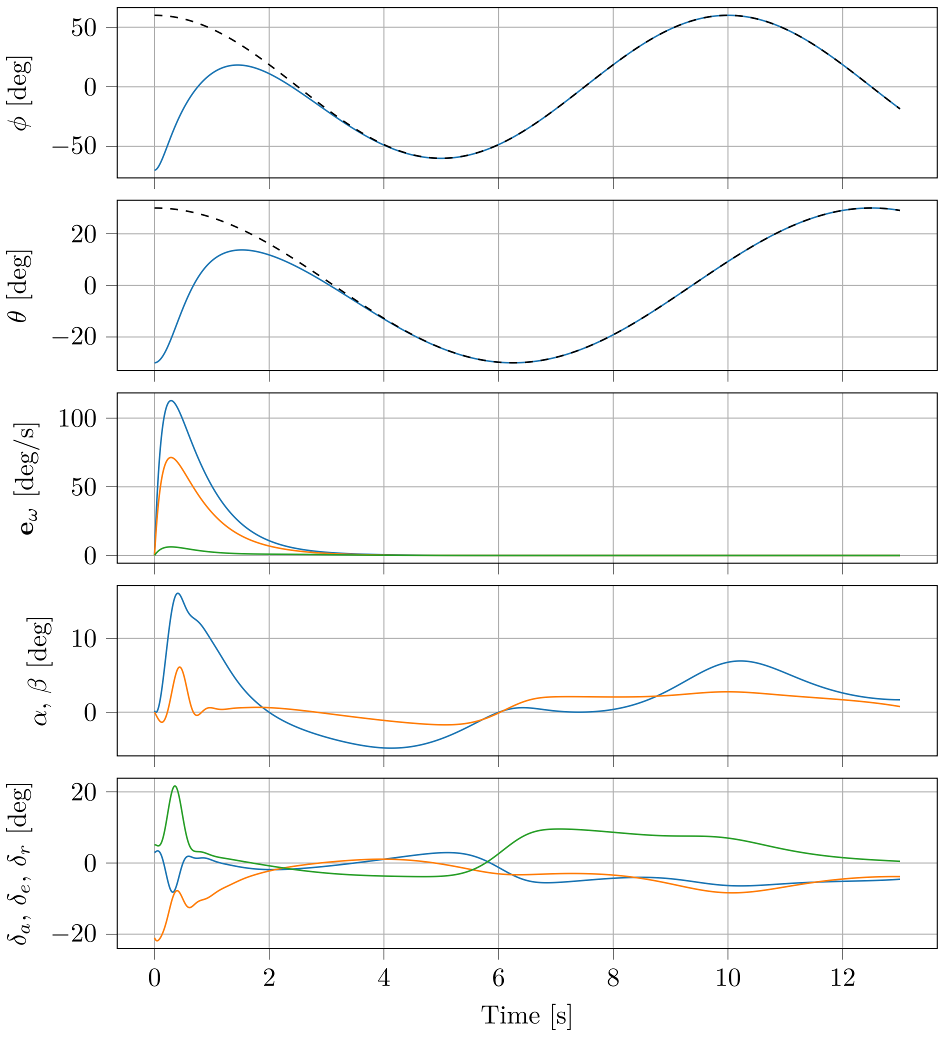

Figure 4.3: Tracking scenario: The attitude trajectories of the geometric controller (blue) converge exponentially to the reference values (dashed). The angular velocity error, $\mathbf{e}_\omega$ in the third subplot (blue, orange, green for the axes x, y, z, respectively), and the angle of attack, α (blue), and sideslip angle, β (orange), in the third subplot remain within reasonable bounds. The deflections for the aileron, δ_a (blue), elevator, δ_e (orange), and rudder, δ_r (green), are also within their respective limits.

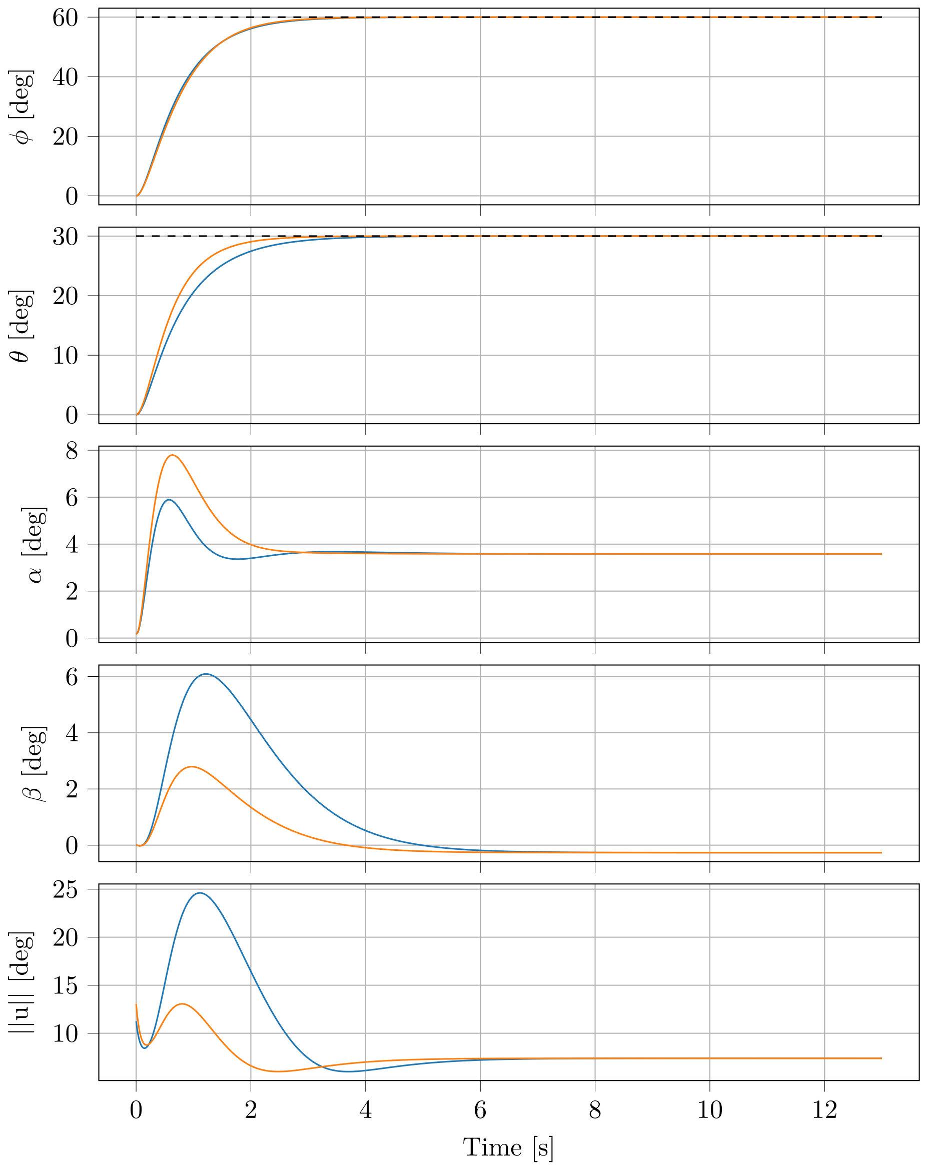

Figure 4.4: Regulation scenario: The baseline controller (blue) results in larger control surface deflections, as indicated by $\|\mathbf{u}\|$, and slower convergence to the reference (dashed) when compared to the geometric controller (orange). The angle of attack, α, and sideslip angle, β, are within reasonable bounds for both controllers.

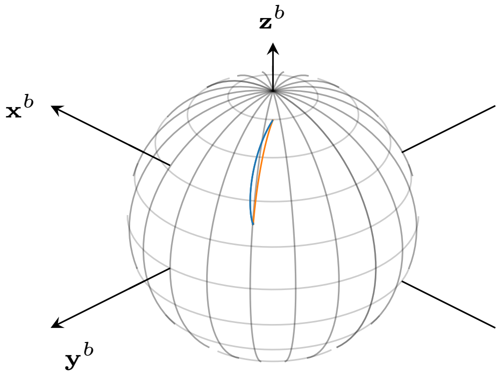

Figure 4.5: Regulation scenario: Trajectories on the two-sphere. The controller based on Euler angles (blue) takes a longer trajectory compared to the geometric controller (orange).

use the notation $V(t,i)$ instead of $V(\phi(t,i))$ where $\phi(t,i)$ denotes a solution to the system dynamics.

In the following we describe the design of a hybrid controller which employs proportional feedback based on a synergistic potential function as presented in [126], coordinated by a set of modes. The magnitude of the proportional feedback will depend on the gradient of the potential function in the active mode, which vanishes at the critical points of the potential function. The synergism property means that at all points other than the reference where this occurs, there is another mode in which the potential function has a significantly lower value. This solves the problem of vanishing proportional action at the opposite direction of the reduced-attitude reference and renders the desired equilibrium globally asymptotically or exponentially stable, depending on the design of the sets $\mathcal{C}$ and $\mathcal{D}$.

4.5.1 Synergistic Potential Function

The design of a synergistic potential function $\Psi : \mathbb{S}^2 \times \mathbb{S}^2 \times \mathbb{S}^2 \times Q \to \mathbb{R}_{\geq 0}$ follows the approach presented in [108] based on two modes which gives the set $Q = \{0, 1\}$. The nominal mode $q = 0$ drives the reduced attitude in the direction of the nominal reference $\mathbf{\Gamma}_d$. The expelling mode $q = 1$ will be designed such that the critical points of its potential function are at maximum distance to both the nominal reference and its antipodal point, i.e. they evolve on the unit circle on $\mathbb{S}^2$ orthogonal to $\mathbf{\Gamma}_d$. Let the reference in the expelling mode be $\mathbf{s}_d \in \mathbb{S}^2$ satisfying

$$\dot{\mathbf{s}}_d = \mathbf{s}_d \times \boldsymbol{\omega}_{\boldsymbol{d}}. \tag{4.67}$$

It follows from Eq. (4.6), Eq. (4.67) and the identity Eq. (2.3) that when $\mathbf{s}_d$ is initialized orthogonal to $\mathbf{\Gamma}_d$, i.e. satisfies $\mathbf{s}_d(0)\cdot\mathbf{\Gamma}_d(0) = 0$, it holds that $\mathbf{s}_d(t)\cdot\mathbf{\Gamma}_d(t) =$

0 for all time. This can be shown expending the orthogonality expression to

$$\frac{d}{dt}(\mathbf{s}_d(t) \cdot \mathbf{\Gamma}_d(t)) = \dot{\mathbf{\Gamma}}_d \cdot \mathbf{s}_d + \mathbf{\Gamma}_d \cdot \dot{\mathbf{s}}_d = (\mathbf{\Gamma}_d \times \boldsymbol{\omega}_d) \cdot \mathbf{s}_d + \mathbf{s}_d \cdot (\boldsymbol{\omega}_d \times \mathbf{\Gamma}_d) = 0, \quad (4.68)$$

where we used Eq. (4.6), Eq. (4.67) and the identity Eq. (2.3).

In the following let $\boldsymbol{\xi} = (\mathbf{\Gamma}, \boldsymbol{\omega}_{nb}^{b\perp}, \mathbf{\Gamma}_d, \mathbf{s}_d, \boldsymbol{\omega}_d) \in \Xi$ be the continuous state of the hybrid system which evolves in the space $\Xi \triangleq \mathbb{S}^2 \times \mathrm{T}_\Gamma\mathbb{S}^2 \times \mathbb{S}^2 \times \mathbb{S}^2 \times \mathrm{T}_{\mathbf{\Gamma}_d}\mathbb{S}^2$. As in [126] we define the synergistic potential function as

$$\Psi_q(\mathbf{\Gamma}, \mathbf{\Gamma}_d, \mathbf{s}_d) = \begin{cases} 1 - \mathbf{\Gamma} \cdot \mathbf{\Gamma}_d & \text{if } q = 0 \\ a + b(1 - \mathbf{\Gamma} \cdot \mathbf{s}_d) & \text{if } q = 1, \end{cases} \quad (4.69)$$

where the parameters $a, b \in \mathbb{R}_{>0}$ act as a bias and a scaling factor to the expelling potential which are designed such that Ψ_q is positive definite relative to $\mathbf{\Gamma}_d$. The gradient of Ψ_q with respect to $\mathbf{\Gamma}$ is given by

$$\nabla_\Gamma \Psi_q(\mathbf{\Gamma}, \mathbf{\Gamma}_d, \mathbf{s}_d) = \begin{cases} -\mathbf{\Gamma}_d & \text{if } q = 0 \\ -b\mathbf{s}_d & \text{if } q = 1. \end{cases} \quad (4.70)$$

Note that since $\mathbb{S}^2$ is a compact manifold and b is finite, $\|\nabla_\Gamma \Psi_q(\mathbf{\Gamma}, \mathbf{\Gamma}_d, \mathbf{s}_d)\|$ is bounded. The nominal mode may be interpreted as a special case with zero bias and unity scaling. The potential function is continuous in each mode and by choosing a, b strictly positive it is positive definite relative to $\mathbf{\Gamma}_d$.

4.5.2 Error States

The goal in either mode is to converge to the attitude with minimum potential along the shortest path on $\mathbb{S}^2$. This can be achieved by applying state feedback that is proportional to the gradient of the potential function with respect to Riemannian metric on $\mathbb{S}^2$. By [19, Lemma 11.6], the gradient vector field on $\mathbb{S}^2$ induced by Ψ_q is given by $\mathrm{d}\Psi_q(\mathbf{\Gamma}) = \mathbf{S}(\mathbf{\Gamma})^2 \nabla_\Gamma \Psi_q$. It can be driven to zero by proportional feedback in the direction of the error vector

$$\mathbf{e}_{\Gamma q} = -\mathbf{\Gamma} \times \nabla_\Gamma \Psi_q(\mathbf{\Gamma}, \mathbf{\Gamma}_d, \mathbf{s}_d) \in \mathrm{T}_\Gamma\mathbb{S}^2, \quad (4.71)$$

which can be interpreted as a rotation axis acting on $\mathbf{\Gamma}$ in the direction of $-\mathrm{d}\Psi_q(\Gamma)$ towards the reference. The illustration of both the nominal and the expelling gradient field is given in Fig. 4.6, which shows that the additional expelling potential solves the problem of the vanishing proportional action at the antipodal of the desired reduced attitude.

Let the angular velocity error $\mathbf{e}_\omega \in \mathrm{T}_\Gamma\mathbb{S}^2$ be defined by

$$\mathbf{e}_\omega = \boldsymbol{\omega}_{nb}^{b\perp} - \mathbf{\Pi}_\Gamma^\perp \boldsymbol{\omega}_d, \quad (4.72)$$

which, using Eq. (4.7) and Eq. (4.10), can be shown to satisfy

$$\dot{\mathbf{e}}_\omega = \mathbf{\Pi}_\Gamma^\perp \Big(\dot{\boldsymbol{\omega}} - \dot{\boldsymbol{\omega}}_d + \boldsymbol{\omega}_{nb}^{b\perp} \times (\boldsymbol{\omega}_{nb}^{b\|} - \mathbf{\Pi}_\Gamma^\| \boldsymbol{\omega}_d) \Big) - \mathbf{\Pi}_\Gamma^\| (\boldsymbol{\omega}_d \times \boldsymbol{\omega}_{nb}^{b\perp}). \quad (4.73)$$

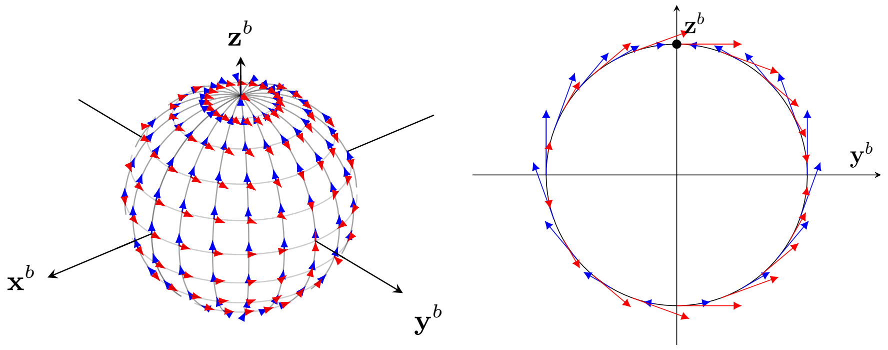

Figure 4.6: The left figure shows the gradient fields defining $\mathbf{e}_{\Gamma_q}$ as induced by the potential Ψ_q for the reference example $\boldsymbol{\Gamma}_d = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^\top$ and $\mathbf{s}_d = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^\top$. The proportional action of the hybrid controller is not vanishing at the antipodal of $\boldsymbol{\Gamma}_d$.

Note that, even though $\boldsymbol{\omega}_{nb}^{b\perp}$ and $\boldsymbol{\omega}_d$ belong to different tangent bundles, the angular velocity error in Eq. (4.72) provides a valid comparison. This can be verified by showing that the sufficiently smooth map $\boldsymbol{\Pi}_\Gamma^\perp$ acts as a transport map following the definition given in [20], which means that the equality

$$\nabla_r \Psi(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d)^\top \mathbf{S}(r) = -\nabla_\Gamma \Psi(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d)^\top \mathbf{S}(\boldsymbol{\Gamma}) \boldsymbol{\Pi}_\Gamma^\perp \tag{4.74}$$

needs to be satisfied for $\mathbf{r} \in \{\boldsymbol{\Gamma}_d, \mathbf{s}_d\}$. Since this is the case, $\boldsymbol{\Pi}_\Gamma^\perp$ is said to be compatible transport map to the potential function Ψ_q, which is essential in the stability proofs [19, Lemma 11.16].

4.5.3 Control Law

The control law is a combination of dynamic inversion, feedforward terms and PD-like feedback in terms of $\mathbf{e}_{\Gamma_q}$ and $\mathbf{e}_\omega$. It is defined as

$$\mathbf{u} = \mathbf{G}^{-1}(\mathbf{v}_r)\mathbf{J}\Big(- \boldsymbol{\Pi}_\Gamma^\perp \mathbf{J}^{-1}\big(\mathbf{f}(\boldsymbol{\omega}, \mathbf{v}_r)\big) + \boldsymbol{\kappa}(\boldsymbol{\xi}, q)\Big) \tag{4.75}$$

where $\boldsymbol{\kappa} : \Xi \times Q \to \mathrm{T}_\Gamma \mathbb{S}^2$ is defined as

$$\boldsymbol{\kappa}(\boldsymbol{\xi}, q) = -\boldsymbol{\omega}_{nb}^{b\perp} \times (\boldsymbol{\omega}_{nb}^{b\|} - \boldsymbol{\Pi}_\Gamma^\| \boldsymbol{\omega}_d) + \boldsymbol{\Pi}_\Gamma^\| \dot{\boldsymbol{\omega}}_d - k_p \mathbf{e}_{\Gamma q} - \boldsymbol{\Pi}_\Gamma^\perp \mathbf{K}_d \mathbf{e}_\omega, \tag{4.76}$$

with proportional gain $k_p \in \mathbb{R}_{>0}$ and positive definite gain matrix $\mathbf{K}_d = {\mathbf{K}_d}^\top \in \mathbb{R}^{3\times3}$. As in the case of the continuous controller, this control law only affects $\dot{\boldsymbol{\omega}}^\perp \in \mathrm{T}_\Gamma \mathbb{S}^2$. This leaves the possibility for an independent design of a control law $\mathbf{u}^\| \in \mathrm{N}_\Gamma \mathbb{S}^2$ for other objectives such as turn coordination, which we discussed in Section 4.3.1.

4.5.4 Jump Map

In the nominal mode, the error vector $\mathbf{e}_{\Gamma q}$ drives the reduced attitude in the direction on the sphere that lies on the path of minimal distance between $\boldsymbol{\Gamma}$ and $\boldsymbol{\Gamma}_d$, known as the minimal geodesic path [21]. To extend this path to the case where the expelling mode is active, $\mathbf{s}_d^+$ is chosen as

$$\mathbf{s}_d^+ \in \mathbf{g}_{\mathbf{s}_d}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d) \triangleq \begin{cases} \frac{\boldsymbol{\Gamma}_d \times \boldsymbol{\Gamma}}{\|\boldsymbol{\Gamma}_d \times \boldsymbol{\Gamma}\|} \times \boldsymbol{\Gamma}_d & \text{if } \boldsymbol{\Gamma} \neq \pm\boldsymbol{\Gamma}_d \\ \mathbf{s}_d & \text{otherwise} \end{cases} \tag{4.77}$$

which in the first case gives the point on $\mathbb{S}^2$ that lies on the geodesic path and satisfies the orthogonality constraint with respect to the nominal reference. The second case is included to ensure a well-defined solution.

4.5.5 Closed-loop System

We can now describe the closed-loop dynamics of the hybrid system. The continuous kinematics of $\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d$ are given by Eq. (4.6), Eq. (4.10) and Eq. (4.67), respectively. The closed-loop dynamics for $\boldsymbol{\omega}_{nb}^{b\perp}$ are given by Eq. (2.32d), Eq. (4.6), Eq. (4.7) and the control law Eq. (4.75). Both references are governed by the same kinematic equation as the reduced-attitude vector and the derivative of the angular velocity reference is included in $c_{\dot{\omega}_d}\mathbb{B}$ which allows for the formulation of an autonomous system [124]. The resulting continuous motion of the closed-loop system is governed by

$$\begin{bmatrix} \dot{\boldsymbol{\Gamma}} \\ \dot{\boldsymbol{\omega}}^{\perp} \\ \dot{\boldsymbol{\Gamma}}_d \\ \dot{\mathbf{s}}_d \\ \dot{\boldsymbol{\omega}}_d \\ \dot{q} \end{bmatrix} \in \mathcal{F}(\boldsymbol{\xi}, q) \triangleq \begin{bmatrix} \boldsymbol{\Gamma} \times \boldsymbol{\omega}_{nb}^{b\perp} \\ \boldsymbol{\kappa}(\boldsymbol{\xi}, q) + \boldsymbol{\omega}_{nb}^{b\perp} \times \boldsymbol{\omega}_{nb}^{b\|} \\ \boldsymbol{\Gamma}_d \times \boldsymbol{\omega}_d \\ \mathbf{s}_d \times \boldsymbol{\omega}_d \\ c_{\dot{\omega}_d}\mathbb{B} \\ 0 \end{bmatrix}. \tag{4.78}$$

The discrete motion is independent of the control law and only has an effect on the mode and the expelling reference which results in jumps as defined by

$$\begin{bmatrix} \boldsymbol{\Gamma}^+ \\ \boldsymbol{\omega}^{\perp+} \\ \boldsymbol{\Gamma}_d^+ \\ \mathbf{s}_d^+ \\ \boldsymbol{\omega}_d^+ \\ q^+ \end{bmatrix} = \mathcal{G}(\boldsymbol{\xi}, q) \triangleq \begin{bmatrix} \boldsymbol{\Gamma} \\ \boldsymbol{\omega}_{nb}^{b\perp} \\ \boldsymbol{\Gamma}_d \\ \mathbf{g}_{\mathbf{s}_d}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d) \\ \boldsymbol{\omega}_d \\ 1 - q \end{bmatrix}. \tag{4.79}$$

4.5.6 Hybrid Controller Sets

To coordinate the control laws, we use the difference between the potential of the current mode to the minimum potential, referred to as synergy gap by [126]. It is defined as

$$\mu(\boldsymbol{\Gamma}, q) = \Psi_q(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d) - \min_{\nu \in Q} \Psi_\nu(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d). \tag{4.80}$$

Let us further introduce the constant hysteresis parameter $\delta \in \mathbb{R}_{>0}$ to define the sets $\mathcal{C}, \mathcal{D} \subset \Xi \times Q$ as

$$\mathcal{C} = \{(\boldsymbol{\xi}, q) : \mu(\boldsymbol{\Gamma}, q) \leq \delta\}, \tag{4.81}$$

$$\mathcal{D} = \{(\boldsymbol{\xi}, q) : \mu(\boldsymbol{\Gamma}, q) \geq \delta\}. \tag{4.82}$$

The following proposition establishes conditions on the potential function such that it is synergistic and positive definite relative to $\mathbf{\Gamma}_d$:

Proposition 4.2: Let the sets $\mathcal{C}$, $\mathcal{D}$ be given by Eq. (4.81) and Eq. (4.82), with synergy gap μ defined in Eq. (4.80). Then the potential function Ψ_q in Eq. (4.69) is a synergistic potential function with gap exceeding δ satisfying

$$0 < \delta < \min\{2 - a - b, a - 1, a + 2b - 1\}. \tag{4.83}$$

Proof: We first show that Ψ_q describes a synergistic potential function with synergy gap exceeding δ. This is to say that at every critical point other than the nominal reference the difference to the other potential function is larger than some specified δ. To this end, denote the set of critical points of Ψ_q for a fixed $q \in Q$ as

$$\text{Crit}\Psi_q = \{(\mathbf{\Gamma}, \mathbf{\Gamma}_d, \mathbf{s}_d) \in (\mathbb{S}^2)^3 : \mathbf{e}_{\Gamma q} = 0\}. \tag{4.84}$$

From the definition of $\mathbf{e}_{\Gamma q}$ it follows that at all critical points, the reduced attitude $\mathbf{\Gamma}$ is parallel to the gradient of the potential function $\nabla_\Gamma \Psi_q$. The set of all critical points follows as $\cup_{q\in Q}\text{Crit}\,\Psi_q = \{(\pm\mathbf{\Gamma}_d), (\pm\mathbf{s}_d)\}$. For $\{\Psi_q\}_{q\in Q}$ to be centrally synergistic relative to $\mathbf{\Gamma}_d$ with gap exceeding δ, the condition $\cup_{q\in Q}\text{Crit}\,\Psi_q \setminus \{\mathbf{\Gamma}_d\} \subset \mathcal{D}$ needs to be satisfied. At $(\mathbf{\Gamma}, q) = (-\mathbf{\Gamma}_d, 0)$, the potential function evaluates to $\Psi_0(-\mathbf{\Gamma}_d) = 2$ and $\Psi_1(-\mathbf{\Gamma}_d) = a + b$, where we use the fact that $\mathbf{s}_d \cdot \mathbf{\Gamma}_d = 0$. To include that point in the jump set, the synergy gap needs to satisfy

$$\mu(-\mathbf{\Gamma}_d, 0) = 2 - a - b > \delta > 0. \tag{4.85}$$

At the critical points of the expelling mode, the nominal potential evaluates to $\Psi_0(\pm\mathbf{s}_d) = 1$ and for the expelling potential we have $\Psi_1(-\mathbf{s}_d) = a + 2b$ and $\Psi_1(\mathbf{s}_d) = a$. Therefore, the synergy gap also has to satisfy

$$\mu(-\mathbf{s}_d, 1) = a - 1 > \delta > 0 \tag{4.86}$$

$$\mu(+\mathbf{s}_d, 1) = a + 2b - 1 > \delta > 0. \tag{4.87}$$

Then by [126, Proposition 1], the potential function Ψ_q is centrally synergistic relative to $\mathbf{\Gamma}_d$ with synergy gap exceeding δ given by Eq. (4.83). ∎

The closed-loop hybrid system is designed such that it satisfies the hybrid basic conditions [64, Assumption 6.5] which makes it nominally robust to measurement noise. The hybrid basic conditions lean on outer semicontinuous set-valued maps, a concept which we briefly look at before our proposition and proof for the basic conditions to be satisfied by the hybrid control system. In [64] outer semicontinuity is defined as

Definition 4.1 (Outer semicontinuity). A set-valued mapping $M : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ is outer semicontinuous (osc) at $\mathbf{x} \in \mathbb{R}^m$ if for every sequence of points $\mathbf{x}_i$ convergent to $\mathbf{x}$ and any convergent sequence of points $\mathbf{y} \in M(\mathbf{x}_i)$, one has $\mathbf{y} \in M(\mathbf{x})$, where $\lim_{i\to\infty} \mathbf{y}_i = \mathbf{y}$. The mapping M is outer semicontinuous if it is outer semicontinuous at each $\mathbf{x} \in \mathbb{R}^m$. Given a set $S \subset \mathbb{R}^m$, $M : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ is outer semicontinuous relative to S if the set-valued mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$ defined by $M(\mathbf{x})$ for $\mathbf{x} \in S$ and $\emptyset$ for $\mathbf{x} \notin S$ is outer semicontinuous at each $\mathbf{x} \in S$.

Goebel adds that outer semicontinuity of $M : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ relative to $S \subset \mathbb{R}^m$ just means that for each $\mathbf{x} \in S$, each sequence of points $\mathbf{x}_i \in S_i$ convergent to $\mathbf{x}$, and each sequence of points $\mathbf{y}_i \in M(\mathbf{x}_i)$ convergent to $\mathbf{y}$, $\mathbf{y} \in M(\mathbf{x})$. More details including examples can be found in [64, Chap. 5]. For our goal of attitude control on $\mathbb{S}^2$, it suffices to know that a set-valued mapping $M : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ is outer semicontinuous if and only if its graph is closed, as given by [64, Lemma 5.10].

We are now ready to propose that the designed closed-loop control system satisfies the hybrid basic conditions:

Proposition 4.3: Consider the sets $\mathcal{C}$ in Eq. (4.81), $\mathcal{D}$ in Eq. (4.82) and the maps $\mathcal{F}$, $\mathcal{G}$ in Eq. (4.78),Eq. (4.79). Then, the following is satisfied:

1. The sets $\mathcal{C}$ and $\mathcal{D}$ are closed.
2. The map $\mathcal{F}$ is outer semicontinuous and locally bounded relative to $\mathcal{C}$ and $\mathcal{F}(\boldsymbol{\xi}, q)$ is convex for every $(\boldsymbol{\xi}, q) \in \mathcal{C}$.
3. The map $\mathcal{G}$ is outer semicontinuous and locally bounded relative to D.

Proof: To show that $\mathcal{C}$ and $\mathcal{D}$ are closed, note that Ψ_q is continuous for $q \in Q$ and that the minimum of two continuous functions is continuous. The synergy gap μ in Eq. (4.80) then is the difference of two continuous functions, which makes it continuous. Therefore, the sets $\mathcal{C}$ and $\mathcal{D}$ are closed. The unit ball $\mathbb{B}$ is compact and convex for any $(\boldsymbol{\xi}, q) \in \mathcal{C}$ such that $\boldsymbol{\omega}_d$, $\dot{\boldsymbol{\omega}}_d$ are bounded by assumption. All remaining components of $\mathcal{F}$ are continuous and single-valued functions on $\mathcal{C}$. Thus, the map $\mathcal{F}$ is convex and locally bounded relative to $\mathcal{C}$ and outer semi-continuity follows from [64, Lemma 5.10], which shows (ii). Further, note that $\mathbb{S}^2$ is compact and hence $\mathbf{s}_d^+ = g_{\mathbf{s}_d}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d) \in \mathbb{S}^2$ is locally bounded relative to $\mathcal{D}$ and the graph of $\mathbf{g}_{\mathbf{s}_d} : \mathbb{S}^2 \times \mathbb{S}^2 \to \mathbb{S}^2$ given by

$$\operatorname{gph} \mathbf{g}_{\mathbf{s}_d} = \{(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d) \in \mathbb{S}^2 \times \mathbb{S}^2 \times \mathbb{S}^2 : \mathbf{s}_d \in g_{\mathbf{s}_d}(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d\} \tag{4.88}$$

is closed. Since $\mathcal{D}$ is closed, outer semi-continuity of $g_{\mathbf{s}_d}$ relative to $\mathcal{D}$ follows from [64, Lemma 5.10]. ■

The stability results can be summarized in two propositions that need the following additional assumption:

Assumption 11. The parameters a, b, $\delta \in \mathbb{R}$ are such that Ψ_q in Eq. (4.69) satisfies Eq. (4.83) according to Proposition 4.2. Further, the time-varying reference trajectory $(\boldsymbol{\Gamma}_d(t), \mathbf{s}_d(t), \boldsymbol{\omega}_d(t), \dot{\boldsymbol{\omega}}_d(t))$ satisfies Eq. (4.10), Eq. (4.11) and Eq. (4.67).

Proposition 1: Let Assumption 9 and Assumption 11 hold. Consider the closed-loop hybrid system $\mathcal{H} = (\mathcal{C}, \mathcal{F}, \mathcal{D}, \mathcal{G})$ with $\mathcal{F}$, $\mathcal{G}$ defined in Eq. (4.78), Eq. (4.79) and the sets $\mathcal{C}$, $\mathcal{D}$ given by Eq. (4.81), Eq. (4.82). Then the set

$$\mathcal{A} = \{(\boldsymbol{\xi}, q) \in \Xi \times Q : \boldsymbol{\Gamma} = \boldsymbol{\Gamma}_d,\ \boldsymbol{\omega}_{nb}^{b\perp} = \boldsymbol{\omega}_d\} \tag{4.89}$$

is globally asymptotically stable for $\mathcal{H}$.

Proof: Given Eq. (4.69), Eq. (4.72) and Eq. (4.6) - Eq. (4.10), the closed-loop solution to Ψ_q can be shown to satisfy

$$\dot{\Psi}_q = \mathbf{e}_{\Gamma q} \cdot \mathbf{e}_\omega, \tag{4.90}$$

where we used the identity Eq. (2.5). The time-derivative of the error vectors are given by

$$\dot{\mathbf{e}}_{\Gamma q} = -\mathbf{S}(\boldsymbol{\omega}_d)\mathbf{e}_{\Gamma q} + \mathbf{S}(\nabla_\Gamma \Psi_q)\mathbf{S}(\boldsymbol{\Gamma})\mathbf{e}_\omega \tag{4.91}$$

$$\dot{\mathbf{e}}_\omega = -k_p \mathbf{e}_{\Gamma q} - \boldsymbol{\Pi}_\Gamma^\perp \mathbf{K}_d \mathbf{e}_\omega - \boldsymbol{\Pi}_\Gamma^\parallel(\boldsymbol{\omega}_d \times \mathbf{e}_\omega) \tag{4.92}$$

which can be found using Eq. (2.3), Eq. (4.5) and Eq. (4.10). Note that the error dynamics in either mode correspond to those in the smooth controller case Eq. (4.26), Eq. (4.27).

To show asymptotic stability, let a Lyapunov function candidate be defined as

$$V = k_p \Psi_q + \frac{1}{2}{\mathbf{e}_\omega}^\top \mathbf{e}_\omega. \tag{4.93}$$

It follows from Eq. (4.90) and Eq. (4.92) that for $(\boldsymbol{\xi}, q) \in \mathcal{C}$ along solutions of the closed-loop system, V satisfies

$$\dot{V}(t,i) = -{\mathbf{e}_\omega}^\top \boldsymbol{\Pi}_\Gamma^\perp \mathbf{K}_d \mathbf{e}_\omega \leq -\lambda_{\min}(\mathbf{K}_d)\|\mathbf{e}_\omega\|^2 \triangleq \mathbf{u}_c(\boldsymbol{\xi}). \tag{4.94}$$

It follows from $\mathbf{K}_d$ being positive definite that $\mathbf{u}_c(\boldsymbol{\xi}) \leq 0$ such that $V(t,i)$ is non-increasing along flows. In $\mathcal{D}$, the mode is switched to the lower potential which leads to the difference during jumps

$$V(t,i+1) - V(t,i) = -k_p\delta \triangleq \mathbf{u}_d. \tag{4.95}$$

This shows that the growth of $V(t,i)$ along solutions to $\mathcal{H}$ is bounded by $\mathbf{u}_c(\boldsymbol{\xi}) \leq 0$ and $\mathbf{u}_d < 0$. Note that by requiring the reference to be bounded, the dynamics of closed-loop system in *Eq.* (4.78) and *Eq.* (4.79) are autonomous and hybrid invariance principles can be applied. Then by [64, Theorem 8.8] we have that for an arbitrary $c \in V(\Xi, Q)$ each pre-compact solution to $\mathcal{H}$ converges to the nonempty set that is the largest weakly invariant subset of

$$\Omega = V(c)^{-1} \cap cl\,(\mathbf{u}_c(0)^{-1}), \tag{4.96}$$

where $V(c)^{-1}$ denotes the preimage of the Lyapunov function candidate at c and $\mathbf{u}_c(0)^{-1}$ denotes the preimage of $\mathbf{u}_c$ at 0 for which $cl\,(\mathbf{u}_c(0)^{-1})$ is the closure. Then from Eq. (4.94) we see that $cl\,(\mathbf{u}_c(0)^{-1})$ leads to $\mathbf{e}_\omega = 0$ which implies $\dot{\mathbf{e}}_\omega = 0$. We substitute this into Eq. (4.92) to see that $\mathbf{e}_{\Gamma q} = 0$ which gives $(\boldsymbol{\Gamma}, q) \in \operatorname{Crit}\Psi$. Since all critical points except $(\boldsymbol{\Gamma}_d, 0)$ are included in $\mathcal{D}$, it follows that $\mathcal{A}$ is the largest weakly invariant subset of Ω. Thus, all pre-compact solutions of $\mathcal{H}$ converge to $\mathcal{A}$. Further, note that $\mathcal{A}$ is compact and $cl\,(\mathcal{C}) \cup \mathcal{D} = \Xi \times Q$ and therefore $\mathcal{G}(\mathcal{D}) \subset cl\,(\mathcal{C}) \cup \mathcal{D}$. Since V is positive definite with respect to $\mathcal{A}$ it follows from [64, Theorem 8.8] and [64, Corollary 8.9 (iii)] that $\mathcal{A}$ is globally asymptotically stable.
■

It follows from Eq. (4.71) and Eq. (4.72) that inclusion in the set $\mathcal{A}$ implies $\mathbf{e}_{\Gamma q} = 0$ and $\mathbf{e}_\omega = 0$. An additional condition for inclusion in the jump set based on the angular velocity error can be shown to yield a stronger stability result. The next proposition summarizes the conditions for global exponential stability.

Proposition 2: Let Assumption 9 and Assumption 11 hold. Consider the closed-loop hybrid system $\mathcal{H} = (\mathcal{C}, \mathcal{F}, \mathcal{D}, \mathcal{G})$ with $\mathcal{F}$, $\mathcal{G}$ defined in Eq. (4.78), Eq. (4.79) and the sets $\mathcal{C}$, $\mathcal{D} \subset \Xi \times Q$ given by

$$\mathcal{C} = \{(\boldsymbol{\xi}, q) : \mu(\boldsymbol{\Gamma}, q) \leq \delta \text{ or } \|\mathbf{e}_\omega\| \geq B_{e_\omega}\}, \tag{4.97}$$

$$\mathcal{D} = \{(\boldsymbol{\xi}, q) : \mu(\boldsymbol{\Gamma}, q) \geq \delta \text{ and } \|\mathbf{e}_\omega\| \leq B_{e_\omega}\} \tag{4.98}$$

where $B_{e_\omega} \in \mathbb{R}_{>0}$ is constant. Then the set

$$\mathcal{A} = \{(\boldsymbol{\xi}, q) \in \Xi \times Q : \boldsymbol{\Gamma} = \boldsymbol{\Gamma}_d,\ \boldsymbol{\omega}_{nb}^{b\perp} = \boldsymbol{\omega}_d\} \tag{4.99}$$

is globally exponentially stable for $\mathcal{H}$.

Proof: The aim of the proof is to show global exponential stability as defined in [181]. Along continuous flows and for bounds, Ψ_q and $\mathbf{e}_{\Gamma q}$ are assumed for a fixed q, unless specified otherwise. We first show that the potential function is uniformly quadratic [20]. Since all critical points other than $\boldsymbol{\Gamma}_d$ are excluded from the flow set $\mathcal{C}$, there exists a constant γ such that the potential function can be bounded from above (see [107]) as

$$\Psi_q(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d) \leq \frac{1}{2}\gamma\|\mathbf{e}_{\Gamma q}\|^2. \tag{4.100}$$

To show a lower bound, we use scaling in each mode and define b_q as $b_0 = 1$ and $b_1 = b$. It then follows from Eq. (4.70) and Eq. (4.71), using Eq. (2.4) that the potential function is uniformly bounded by

$$\frac{1}{2b_q}\|\mathbf{e}_{\Gamma q}\|^2 \leq \Psi_q(\boldsymbol{\Gamma}, \boldsymbol{\Gamma}_d, \mathbf{s}_d) \leq \frac{1}{2}\gamma\|\mathbf{e}_{\Gamma q}\|^2. \tag{4.101}$$

Using Eq. (4.93), let a Lyapunov-function candidate be

$$V_\epsilon = V + \epsilon\mathbf{e}_\omega^\top\mathbf{e}_{\Gamma q} \tag{4.102}$$

for some $\epsilon \in \mathbb{R}_{>0}$. From Eq. (4.101) and defining $\mathbf{z} = \begin{bmatrix}\|\mathbf{e}_{\Gamma q}\| & \|\mathbf{e}_\omega\|\end{bmatrix}^\top$ we see that V_ϵ can be bounded by

$$\frac{1}{2}\mathbf{z}^\top\mathbf{M}_1\mathbf{z} \leq V_\epsilon \leq \frac{1}{2}z^\top\mathbf{M}_2\mathbf{z}, \tag{4.103}$$

where the matrices $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{R}^{2\times 2}$ are given by

$$\mathbf{M}_1 = \begin{bmatrix}\frac{k_p}{b_q} & -\epsilon \\ -\epsilon & 1\end{bmatrix}, \qquad \mathbf{M}_2 = \begin{bmatrix}k_p\gamma & \epsilon \\ \epsilon & 1\end{bmatrix}. \tag{4.104}$$

Next, we show that there exists a $\lambda > 0$ such that along solutions of the closed-loop dynamics, V_ϵ can be bounded by

$$V_\epsilon(t, i) \leq V_\epsilon(0, 0)\exp(-\lambda t). \tag{4.105}$$

During jumps, the difference in V_ϵ is given by

$$V_\epsilon(t, i+1) - V_\epsilon(t, i) = -k_p\delta + \epsilon\mathbf{e}_\omega{}^\top(\mathbf{e}_{\Gamma q^+} - \mathbf{e}_{\Gamma q}), \tag{4.106}$$

which can be bounded using Eq. (2.7) and the triangle inequality such that

$$\begin{aligned} &V_\epsilon(t, i+1) - V_\epsilon(t,i) \\ &\qquad \leq -k_p\delta + \epsilon \|\mathbf{e}_\omega\| \|\mathbf{\Gamma} \times (\mathbf{\Gamma}_d - b\mathbf{s}_d)\| \\ &\qquad \leq -k_p\delta + \epsilon \|\mathbf{e}_\omega\| (\|\mathbf{\Gamma}_d\| + |b| \|\mathbf{s}_d\|) \\ &\qquad \leq -k_p\delta + \epsilon B_{e_\omega}(1 + |b|). \end{aligned} \tag{4.107}$$

The right side of the last inequality is non-positive for

$$\epsilon \leq \frac{k_p \delta}{B_{e_\omega}(1+|b|)}, \tag{4.108}$$

which shows that V_ϵ is non-increasing during jumps. Next we show that there exists a positive definite matrix $\mathbf{M}_3 \in \mathbb{R}^{2\times 2}$ such that along flows, V_ϵ satisfies

$$\dot{V}_\epsilon(t,i) \leq -z^\top \mathbf{M}_3 z. \tag{4.109}$$

The upper bound of $\dot{V}$ is given by Eq. (4.94), and it remains to find a bound for the time-derivative of the cross-term in Eq. (4.102). From Eq. (4.91), Eq. (4.92) we see that

$$\begin{aligned} \frac{d}{dt}(\mathbf{e}_\omega^\top \mathbf{e}_{\Gamma q}) &= \mathbf{e}_\omega^\top (\mathbf{S}(\nabla_\Gamma \Psi_q)\mathbf{S}(\mathbf{\Gamma}))\mathbf{e}_\omega - k_p \|\mathbf{e}_{\Gamma q}\|^2 - \mathbf{e}_{\Gamma q}{}^\top (\mathbf{K}_d + \mathbf{S}(\boldsymbol{\omega}_d))\mathbf{e}_\omega \\ &\leq \|\nabla_\Gamma \Psi_q\| \|\mathbf{e}_\omega\|^2 - k_p \|\mathbf{e}_{\Gamma q}\|^2 + (\lambda_{\max}(\mathbf{K}_d) + B_{\omega_d}) \|\mathbf{e}_{\Gamma q}\| \|\mathbf{e}_\omega\|, \end{aligned}$$

where *Eq.* (2.7) is used. From Eq. (4.94) and Eq. (4.109) follows

$$\mathbf{M}_3 = \begin{bmatrix} \epsilon k_p & -\frac{\epsilon}{2}(\lambda_{\max}(\mathbf{K}_d) + B_{\omega_d}) \\ -\frac{\epsilon}{2}(\lambda_{\max}(\mathbf{K}_d) + B_{\omega_d}) & \lambda_{\min}(\mathbf{K}_d) - \epsilon \|\nabla_\Gamma \Psi_q\| \end{bmatrix}. \tag{4.110}$$

The matrices $\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3$ are positive definite for any ϵ satisfying

$$\epsilon < \min_{q \in Q} \left\{ \sqrt{\frac{k_p}{b_q}}, \frac{4\lambda_{\min}(\mathbf{K}_d)}{4k_p \|\nabla_\Gamma \Psi_q\| + (\lambda_{\max}(\mathbf{K}_d) + B_{\omega_d})^2} \right\}, \tag{4.111}$$

which shows that Eq. (4.105) is satisfied for all initial conditions with $\lambda = \lambda_{\min}(\mathbf{M}_3)$. We can then apply [181, Theorem 1] to conclude global exponential convergence of V_ϵ. Then $V_\epsilon = 0$ if and only if $\Psi_q = 0$ and $\mathbf{e}_\omega = 0$ and hence $\mathbf{\Gamma} \to \mathbf{\Gamma}_d$ and $\boldsymbol{\omega}_{nb}^{b\perp} \to \boldsymbol{\omega}_d$, which shows that $\mathcal{A}$ is globally exponentially stable. ■

Note in the proof to Proposition 2 that B_{e_ω} may be chosen arbitrarily large such that it does not necessarily impose practical limitations. Moreover, the sets $\mathcal{C}$ and $\mathcal{D}$ are closed and the maps $\mathcal{F}$ and $\mathcal{G}$ are not changed. The hybrid basic conditions are thus also satisfied for Proposition 2.

4.6 Simulation Study of the Hybrid Geometric Attitude Controller

We use the model of the Aerosonde UAV with the nonlinear aerodynamics as described in [9] and compare two controllers. First, a continuous controller that

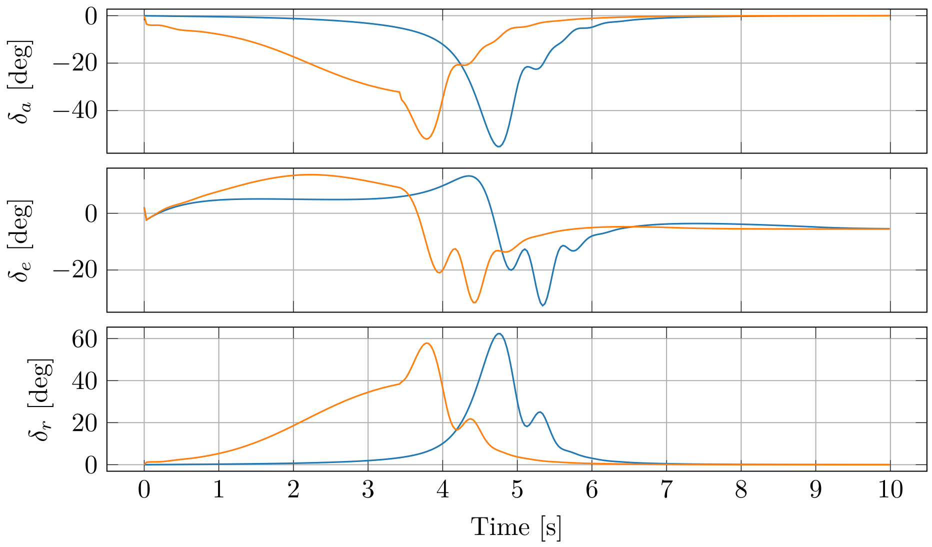

Figure 4.7: Control surface deflections of aileron δ_a, elevator δ_e and rudder δ_r for the continuous controller (blue) and the hybrid controller (orange).

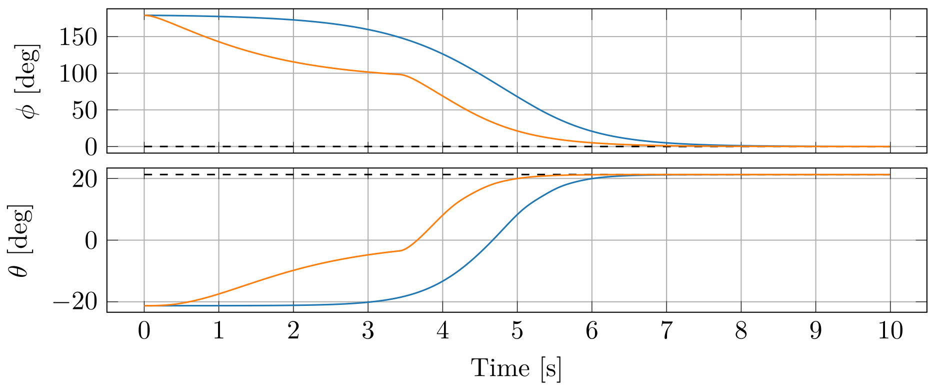

Figure 4.8: Attitude response represented by the angles roll ϕ and pitch θ for the continuous controller (blue) and the hybrid controller (orange) including the reference (dashed).

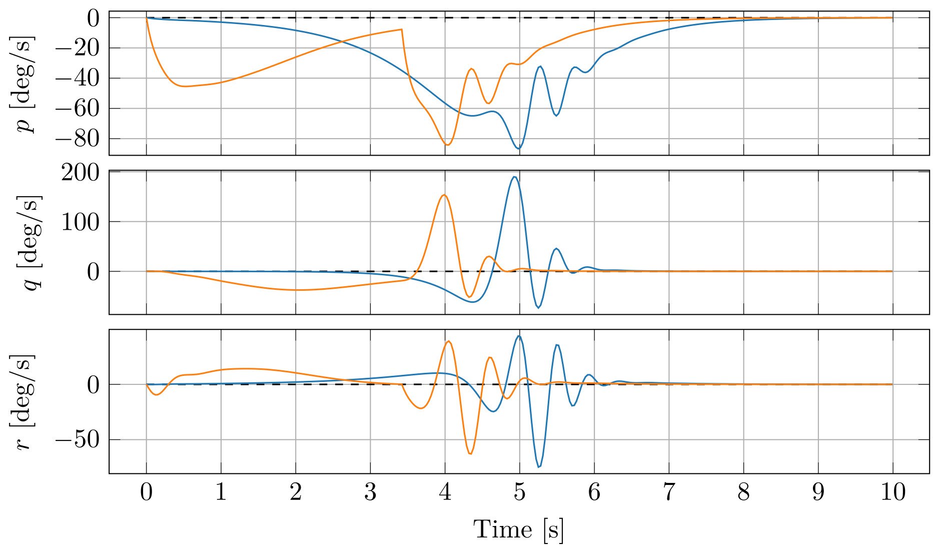

Figure 4.9: Angular rate response represented by roll rate p, pitch rate q, and yaw rate r, for the continuous controller (blue) and the hybrid controller (orange) including the reference (dashed).

employs the nominal mode throughout the simulation and second, the presented hybrid controller that may switch to the expelling mode in addition. A stability analysis showing semi-global exponential stability of the continuous controller is presented in [37]. The airspeed is treated as an exogenous signal and controlled via a PI-Controller through the propeller throttle.

The controller parameters are chosen as $k_p = 9.5$, $\mathbf{K}_d = 8I_3$, $a = 1.25$, $b = 0.6$. We simulate the recovery from a large initial attitude disturbance and set the initial state such that $\mathbf{\Gamma}(0) = -\exp(e_1\epsilon)\mathbf{\Gamma}_d$ with $\epsilon = \pi/180$ and $e_1 = [1\ 0\ 0]^\top$. In terms of Euler angles, this corresponds a roll angle of -179 degrees and pitch angle of -21.26 degrees. The yaw angle is set to zero. The reference is parameterized according to Eq. (4.4) with zero roll angle and 21.26 degrees pitch angle, which is the trim condition for wings-level ascending flight at 35 meters per second airspeed. Note that the initial attitude is thus far from the given reference.

As shown in Fig. 4.8, the continuous controller remains close to the initial attitude up to 3 seconds whereas the hybrid controller reacts instantly and uses the expelling potential with a larger gradient up to 3.5 seconds into the simulation before switching to the nominal potential (cf. Fig. 4.11). As a consequence, the hybrid controller recovers faster from the descending flight condition at a lower speed (cf. Fig. 4.10) and returns to ascending flight two seconds before the continuous controller, with similar actuator usage (cf. Fig. 4.7). A drawback of the hybrid controller however is the deceleration close to the expelling reference as shown in Fig. 4.9 and Fig. 4.11. This suggests using a dynamic extension in which the control action is given by a dynamic weighting of both configuration error vectors as done

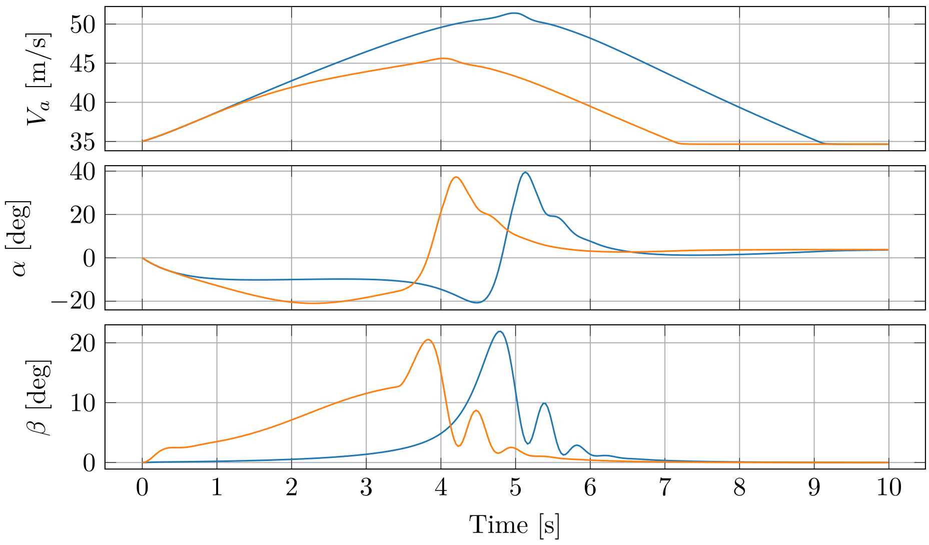

Figure 4.10: Results of the relative velocity represented by airspeed V_a, angle of attack α, and sideslip angle β, for the continuous controller (blue) and the hybrid controller (orange).

in [10] or [126]. Future work will also address performance of the hybrid controller in the face of non-vanishing disturbances and model perturbations. Another aspect is the extension to the optimal use of the actuators while respecting saturation constraints, potentially in a model predictive control scheme.

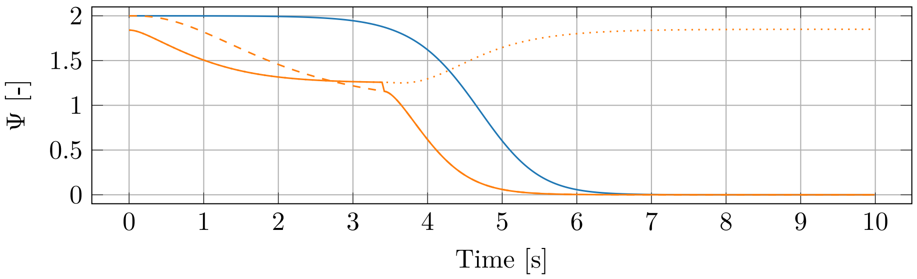

Figure 4.11: Trajectories of the potential functions for the continuous controller (blue) and the hybrid controller (orange). The values for the nominal potential function Ψ_0 (dashed), the expelling potential function Ψ_1 (dotted), and the activated potential function Ψ_q (solid) are shown.

4.7 Experimental Verification

For the experiments we used a more simple controller structure that is conceptually the same as outlined in the preceding sections, but does not depend on a dynamic model except for the control-effectiveness matrix $\mathbf{G}$. The controller does not include hybrid feedback and was tested in conditions where this would not have had an effect. The structure reads as

$$\boldsymbol{\omega}_d^{\perp} = k_{\omega,e_\Gamma}\mathbf{e}_\Gamma \tag{4.112}$$

$$\boldsymbol{\omega}_d^{\parallel} = \frac{g}{V_a}\tan(\phi)\boldsymbol{\Gamma} \tag{4.113}$$

$$\boldsymbol{\omega}_d = \boldsymbol{\omega}_d^{\perp} + \boldsymbol{\omega}_d^{\parallel} \tag{4.114}$$

$$\mathbf{z} = \boldsymbol{\omega} - \boldsymbol{\omega}_d \tag{4.115}$$

$$\mathbf{u} = \mathbf{G}(\mathbf{v}_r)^{\dagger}(-k_1\mathbf{e}_\Gamma - \mathbf{K}_2\mathbf{z} - \hat{\boldsymbol{\Delta}}) \tag{4.116}$$

$$\dot{\hat{\boldsymbol{\Delta}}} = \mathbf{K}_3\mathbf{z}, \tag{4.117}$$

where Eq. (4.113) is equivalent to Eq. (4.56). The controller structure is designed for practical implementations where sufficient knowledge of the aerodynamic model or matrix of inertia may not be available. We therefore drop the feedforward terms and instead introduce the disturbance estimate $\hat{\boldsymbol{\Delta}} \in \mathbb{R}^3$.

As we conducted the experiments with the X8 which is an underactuated UAV, with its rank-deficient matrix $\mathbf{G}$ as outlined in Chapter 2, it is necessary to use the Moore-Penrose inverse $\mathbf{G}^{\dagger} = (\mathbf{G}^\top\mathbf{G})^{-1}\mathbf{G}^\top$. To extend the stability proof for asymptotic stability of the resulting closed-loop trajectories to the scenario of the underactuated UAV requires an additional symmetry condition for the matrix of inertia [2]. The elements about two principal axes would need to be equal which would be fulfilled for the example of a homogenous cylinder, but not for a regular fixed-wing UAV. We therefore do not give a stability proof here, but show the practical efficacy in experiments. For an in-depth analysis of the controller structure used in the experiments, but with full actuation, see [35]. The controller is parameterized with $k_0 = 5.0$, $k_1 = 1.0$, $\mathbf{K}_2 = \operatorname{diag}(\begin{bmatrix}5.0 & 1.5 & 2.0\end{bmatrix})$, and $\mathbf{K}_3 = \operatorname{diag}(\begin{bmatrix}0.2 & 0.2 & 0.1\end{bmatrix})$. The update rate was set to 50 Hz.

The data collected in the experiments includes two modes of operation for which we present an example. The first mode is referred to as Fly-by-wire-A (FBWA), when it is active, the pilot that operates the UAV uses the RC transmitter to set reference values for the angles roll and pitch directly. The references are then tracked by the low-level controller, and the throttle set-point is controlled manually. The second mode sets the UAV into automatic control (AUTO) and the low-level controller receives reference values from a higher-level guidance module that is following a path defined via a pattern of waypoints.

The results are shown in Fig. 4.12. In the first half, until 930s, the UAV is operated in FBWA, before it is set to mode AUTO to follow a rectangular pattern of waypoints. During FBWA operation, we tested the response to series of manually set steps for pitch and then for roll, which are set to their respective limits. The controller is able to track the references with convergence that appears to be approximately exponential. The tracking performance is better in roll compared to pitch, where the controller seems to struggle to track the negative steps. During AUTO, the controller is able to follow the reference signals set by the guidance module (L1 and TECS implemented in ArduPlane), but again there are periods during which there is a static offset in pitch. The results are from an early stage of testing and further tuning of the integral gain may compensate for this problem. However, for control architectures as the one employed here, the timescale separation principle requires the low-level controller to have significantly higher bandwidth compared to the guidance controller, such that integral action on this level is usually not included.

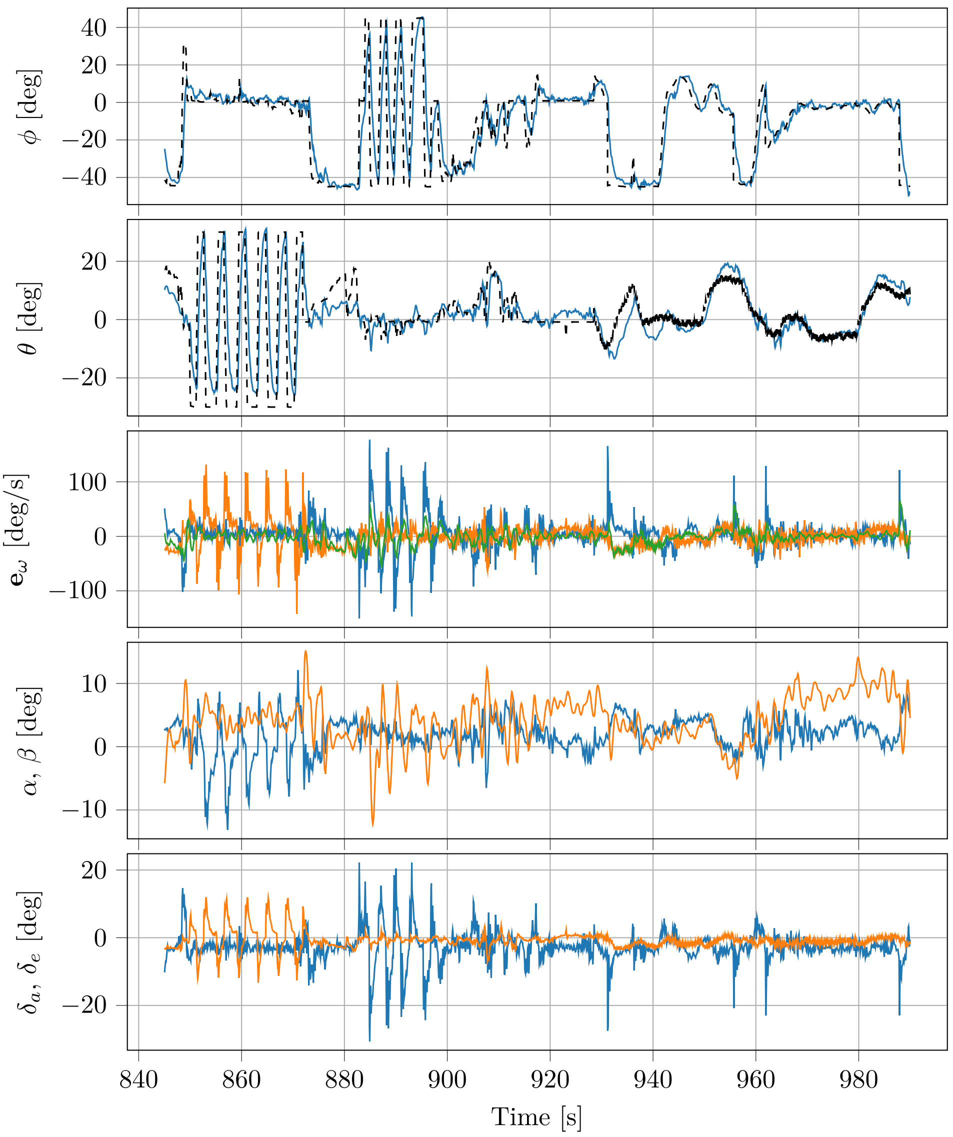

Figure 4.12: Experiment with the smooth controller. In the first half of the data series the pilot controlled the UAV by setting reference to roll and pitch via the RC Transmitter (mode FBWA). In the second half, the UAV received the reference signals (black, dashed) from a path following controller to follow a rectangular pattern of waypoints. Angular velocity errors for roll, pitch and yaw direction (in blue, orange, green, respectively) are shown in the third plot. The angle of attack (blue) and sideslip angle (orange), and the deflection of the aileron (blue) and elevator (orange) are shown in the last two plots.

4.8 Benchmark Scenario

The geometric attitude controller that we use in the benchmark scenario has a simpler structure than presented in the main contribution of this chapter. It does not use dynamic model inversion to the same extent, but rather includes integral action. The required knowledge of the model is therefore reduced to the control-effectiveness matrix $\mathbf{G}$. The resulting control law for the geometric attitude controller is given as

$$\mathbf{u} = \mathbf{G}(\mathbf{v}_r)^{\dagger}(-k_p \mathbf{e}_{\Gamma} - \mathbf{K}_d \mathbf{e}_{\omega} - \mathbf{K}_i \hat{\boldsymbol{\Delta}}) \tag{4.118}$$

$$\dot{\hat{\boldsymbol{\Delta}}} = \mathbf{e}_{\Gamma}. \tag{4.119}$$

The controller gains are tuned to $k_p = 20.0$ and $\mathbf{K}_i = \mathbf{K}_d = \mathrm{diag}(2.0, 2.0, 2.0)$, to give a similar step response to the baseline controllers as outlined in Appendix A, and the desired angular velocity $\boldsymbol{\omega}_d$ is set to zero. The airspeed controller is the same as for the baseline controllers introduced in Section 2.5. The path-following performance in cascade with the guidance controller is only marginally different to the baseline controllers, as can be seen from the plots in Fig. 4.13 and the mean squared distance $J_{e,d}$ given in Section A.2. From Fig. 4.14, it appears that the geometric attitude controller results in slightly slower changes in the control surfaces. To some extent this confirms the observation of more efficient actuator usage drawn in Section 4.4.2. However, the reduced actuator usage seems to cause an increase in the roll and pitch errors, suggesting that the difference in performance and actuator usage in this case can simply be reduced by selecting the gains differently.

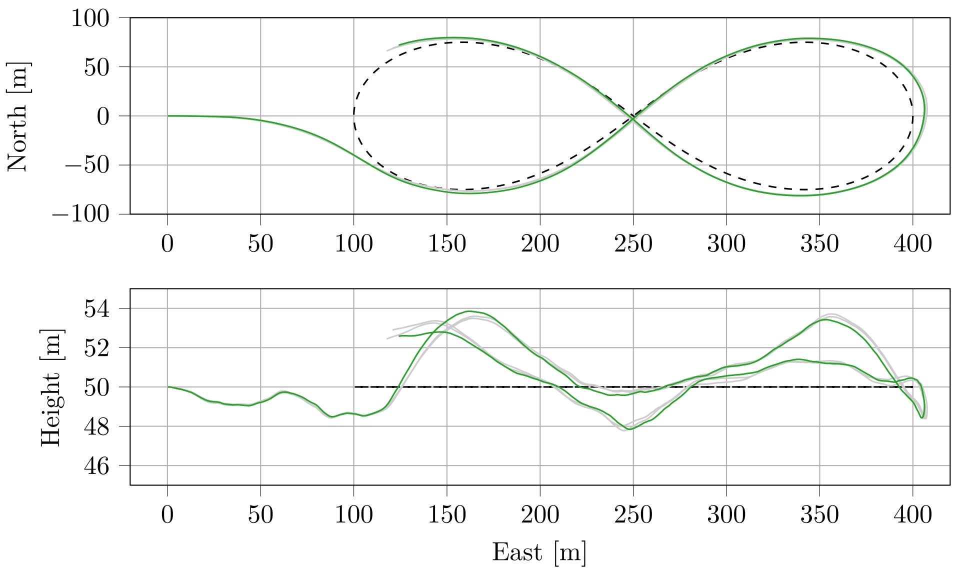

Figure 4.13: Benchmark simulation comparing the geometric controller (GC, plotted in green) introduced in this chapter to the previous controller designs (gray). The reference path is plotted in black, dashed.

4.9 Chapter Summary

We introduced a geometric controller for the control of roll and pitch angle based on and equivalent reduced-attitude vector. We highlighted the benefits of this alternative attitude representation and used it to derive a proportional-derivative control law with feedback linearization. Based on Lyapunov theory, the control law is shown to render the reduce-attitude reference almost semi-globally exponentially stable and almost globally asymptotically stable, with region of exponential convergence depending on the chosen controller gains. After comparing the geometric controller to a more conventional controller based on Euler angles, both controllers were evaluated in numerical results, showing that the geometric achieves set-point stabilization on the shortest path on the two sphere. The proportional feedback of the geometric controller was then extended to hybrid feedback to overcome the topological obstruction of the two-sphere and render the desired reduced attitude exponentially stable, which we showed in detailed stability proofs. The benefits of the hybrid controller over the smooth geometric controller was demonstrated in a simulation scenario where the performance of the smooth controller is negatively effected by the unstable equilibrium. We closed the chapter with experimental results that demonstrate the practical use of the smooth geometric controller.

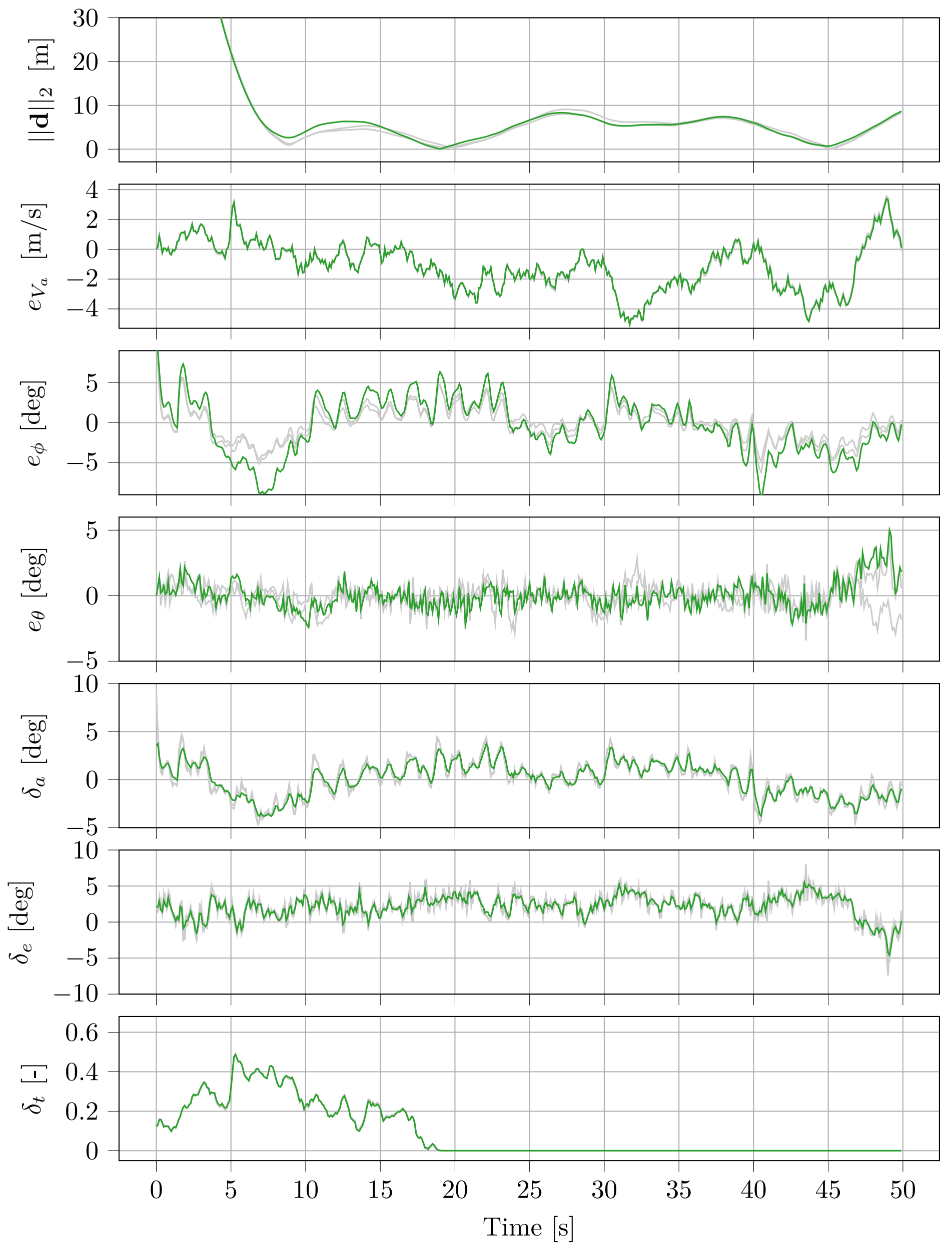

Figure 4.14: Benchmark simulation comparing the geometric controller (GC, plotted in red) introduced in this chapter to the previous controller designs (gray). The first subplot includes the distance to the reference path $\|\mathbf{d}\|_2$, and the following subplots include the error signals for airspeed, roll and pitch, as well as the actuator signals.

Chapter 5

Direct Nonlinear Model Predictive Control for Attitude and Speed Control

This chapter is based on

[155] Dirk Reinhardt and Tor Arne Johansen. Nonlinear Model Predictive Attitude Control for Fixed-Wing Unmanned Aerial Vehicle based on a Wind Frame Formulation. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 503–512, June 2019.

[156] Dirk Reinhardt and Tor Arne Johansen. Control of Fixed-Wing UAV Attitude and Speed based on Embedded Nonlinear Model Predictive Control. *IFAC-PapersOnLine*, 54(6):91–98, 2021. 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021.

[152] Dirk Reinhardt, E. M. Coates, and Tor Arne Johansen. Low-level Nonlinear Model Predictive Attitude and Speed Control of Fixed-Wing Unmanned Aerial Vehicles. *Control Engineering Practice*, submitted.

5.1 Introduction

The classic design for the low-level autopilot of modern Unmanned Aerial Vehicles (UAVs) is based on first-order Taylor approximations of the nonlinear dynamics around trim states in which the UAV is either in wings-level horizontal flight or performing a banked turn with a possible change in altitude [9, 177]. The resulting controllers show desirable performance in the vicinity of the trim conditions and the UAV is operated with set-points for speed and attitude such that the vehicle stays in the region where the local approximation is valid. This often results in a rather conservative usage of the UAVs physical capabilities in autonomous flight. In addition, it is difficult to handle actuator limits in the controller design, leading to controller gains to be tuned in a way to not saturate the control inputs.

Considering the full flight envelope, in particular, effects related to the detachment of the laminar flow from the airfoil in the transition to the angles of attack above the stall angle introduce more significant non-linearities into the dynamic equations. On the kinematic level, the natural configuration space of the UAV attitude, either be it the full attitude represented by elements of the Special Orthogonal Group SO(3) or vectors on the unit two-sphere $\mathbb{S}^2$ for roll-pitch or pitch-yaw control, are often captured by linear approximations in the local tangent space at each element, which implies only local performance and stability properties. This makes large-angle attitude maneuvers, with consistent performance throughout the transition, difficult to achieve by applying linear methods. Assuming that the model of the dynamics is sufficiently accurate, a nonlinear controller may not have these problems.

Nonlinear Model Predictive Control (NMPC) can be applied to include the nonlinear effects into the controller design and explicitly deal with the boundary of the flight envelope and the actuator limits by including them in the definition of the constraint set. Due to the rather fast low-level dynamics in roll and pitch motion, most NMPCs for UAVs are designed for the guidance level, such as [3, 174, 194], and rely on low-level PID-type controllers to track given attitude/speed set-points. However, the autopilot comes with a set of tuning parameters that have been obtained for the nominal trim conditions. When far away from them, the low-level controller might show degrading performance which will carry over to the higher-level NMPC. In this case, direct actuation may lead to improved performance, as shown for multi-rotors in [139, 197]. For fixed-wing UAVs however, few works use MPC to directly control surfaces and engine. One exception is [143] where linear perturbation models in lateral/longitudinal direction are identified from which explicit UAVs with low computational demands can be derived. Mammarella and Capello [116, 117] also use liner models to design a robust MPC in a tube-based approach. See also their work in [118] using sample-based stochastic MPC for tracking control.

For high-performance flight in a wide envelope, however, we believe that it is necessary to consider coupled dynamics in the nonlinear regime (see [92] for a comparison between linear and nonlinear MPC for multi-rotors). The present work is a contribution toward this goal. A limiting factor for the use of NMPC in robotic applications with fast dynamics has been the achievable update rate to counteract disturbances, which is limited by the closed-loop runtime of the employed numerical solver to update the predicted optimal trajectory. We show that due to recent advancements on both algorithmic and hardware levels, low-level control using NMPCs with up to 40Hz update rates is feasible, despite a rather rich dynamic model and moderate environmental disturbances due to wind. The controller design may be employed on standard embedded computing platforms and the proposed architecture of the flight stack allows for operation parallel to an established low-level autopilot.

5.2 Controller Design and Implementation

5.2.1 Dynamic Model

We use the rotation matrix for the attitude representation, in contrast to minimum parameterizations such as Euler angles or quaternions, which is motivated by the fact that they result in a globally unique and non-singular attitude representation. Quaternions evolve on $\mathbb{S}^3$, which is a double cover of the natural configuration space SO(3), meaning that two quaternions can be used to represent the same attitude. Euler angles are prone to singular attitude representations (gimbal lock), which would need to be handled in the implementation. Chaturvedi et al. [29] offers an excellent discussion on this topic.

We decompose the rotation matrix into the individual axes, i.e. $\mathbf{R}_{nb} = [\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z]$ with $\mathbf{r}_{\{x,y,z\}} \in \mathbb{S}^2$ representing the axes of the body-fixed frame expressed in the coordinates of the inertial frame. Then define the state vector $\mathbf{x} \in \mathbb{R}^{n_x}$ and input vector $\mathbf{u} \in \mathbb{R}^{n_u}$ as

$$\mathbf{x} = \left[V_a \ \beta \ \alpha \ \mathbf{r}_x{}^\top \ \mathbf{r}_y{}^\top \ \mathbf{r}_z{}^\top \ (\boldsymbol{\omega}^s_{nb})^\top \ \delta_a \ \delta_e \ \delta_r \ \delta_t\right]^\top, \tag{5.1}$$

$$\mathbf{u} = [\dot{\delta}_a \ \dot{\delta}_e \ \dot{\delta}_r \ \dot{\delta}_t], \tag{5.2}$$

with $n_x = 19$ and $n_u = 4$. The vector $\boldsymbol{\omega}^s_{nb}$ denotes the angular velocity vector decomposed in $\{s\}$. The dynamic and kinematic equations for the state variables are given by

$$\begin{bmatrix} \dot{V}_a \\ \dot{\beta} V_a \\ \dot{\alpha} V_a \cos\beta \end{bmatrix} = \frac{1}{\mathrm{m}}(\mathbf{F}^w_a + \mathbf{R}_{wb}\mathbf{F}^b_T) + \mathbf{R}_{wb}\mathbf{R}_{nb}{}^\top \mathbf{g}^n - \boldsymbol{\omega}^w_{nb} \times \mathbf{v}^w_r \tag{5.3a}$$

$$\dot{\mathbf{R}}_{nb} = \begin{bmatrix} \dot{\mathbf{r}}_x & \dot{\mathbf{r}}_y & \dot{\mathbf{r}}_z \end{bmatrix} = \mathbf{R}_{nb}\mathbf{S}(\mathbf{R}_{sb}{}^\top \boldsymbol{\omega}^s_{nb}) \tag{5.3b}$$

$$\dot{\boldsymbol{\omega}}^s_{nb} = -\boldsymbol{\omega}^s_{bs} \times \boldsymbol{\omega}^s_{nb} + (\mathbf{J}^s)^{-1}(\mathbf{R}_{sb}\boldsymbol{\tau}^b - \boldsymbol{\omega}^s_{nb} \times \mathbf{J}^s \boldsymbol{\omega}^s_{nb}), \tag{5.3c}$$

with $\mathbf{J}^s$ given by the similarity transformation $\mathbf{J}^s = \mathbf{R}_{sb}\mathbf{J}^b\mathbf{R}_{sb}{}^\top$. Other new quantities are the angular velocity of $\{s\}$ with relative to $\{b\}$ and the relative velocity vector in $\{w\}$ denoted given by

$$\boldsymbol{\omega}^s_{bs} = \begin{bmatrix} 0 & \dot{\alpha} & 0 \end{bmatrix}^\top, \quad \mathbf{v}^w_r = \begin{bmatrix} V_a & 0 & 0 \end{bmatrix}^\top. \tag{5.4}$$

For a detailed derivation of the model in a slightly different form, see [177]. Throughout the rest of the paper, we will use the continuous vector ODE $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ to denote the state dynamics described in Eq. (5.3a) - Eq. (5.3c). Where applicable, the discretized version based on an explicit Runge-Kutta method will be used, in that case denoted as the difference equation $\mathbf{x}(k+1) = \mathbf{f}_{\mathrm{RK4}}(\mathbf{x}(k), \mathbf{u}(k))$.

5.2.2 Target Generation

The full attitude on SO(3), or it's parameterization based Euler angle or quaternion, is not controllable for fixed-wing aircraft. This fact is caused by the coupling between roll angle and yaw rate in banked turns, implying that they can not be controlled independently. The consequence of this is that a projection to a controllable subspace has to be applied first. The two natural alternatives that exist are roll-pitch control or pitch-yaw control. For roll-pitch control, we effectively steer the direction of the gravity axis of the inertial frame with respect to the body-fixed frame, as discussed in detail in Chapter 4. Pitch-yaw control steers the direction of the longitudinal axis of the body-fixed frame, i.e. $\mathbf{x}^b$, with respect to the inertial frame. We continue the notation introduced in Chapter 4 and let the reduced attitude vector to represent both projections be denoted by $\mathbf{\Gamma} : \mathrm{SO}(3) \to \mathbb{S}^2$.

Roll-Pitch Projection

The reduced-attitude vector $\mathbf{\Gamma} \in \mathbb{S}^2$ in a roll-pitch projection is the same as we used for geometric attitude control in Chapter 4, which was defined as the direction of the vertical axis of the inertial frame $\mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$ expressed in the body-fixed frame

$$\mathbf{\Gamma} = \mathbf{R}_{nb}{}^\top \mathbf{e}_3. \tag{5.5}$$

The same reduced-attitude parameterization has been applied to stabilization of the inverted 3D pendulum [30]. Note that the reduced-attitude vector is invariant to rotations about $\mathbf{e}_3$ and therefore independent of yaw. In fact, given a roll angle $\phi \in [-\pi, \pi]$ and pitch angle $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, the reduced-attitude vector can be parameterized as

$$\mathbf{\Gamma}(\phi, \theta) = \begin{bmatrix} -\sin\theta & \cos\theta \sin\phi & \cos\theta\cos\phi \end{bmatrix}^\top. \tag{5.6}$$

It is rather simple to use roll angle and pitch angle directly to generate the output target. In practice, however, references on climb rate $\dot{h}_{\mathrm{ref}}$ and course turn-rate $\dot{\chi}_{\mathrm{ref}}$ can be used to set up an optimization problem to find the desired reduced-attitude vector.

As input to the target generation routine, consider the triplet $(\dot{h}_{\mathrm{ref}}, \dot{\chi}_{\mathrm{ref}}, V_{a,\mathrm{ref}})$. The output target for the controller is then given by the optimization problem

$$\begin{aligned} J^*(\mathbf{v}^n_{nw}) := \min_{\mathbf{x}} \dot{\mathbf{x}}^\top \mathbf{C}_{\phi\theta}{}^\top \mathbf{C}_{\phi\theta} \dot{x} && \text{(5.7a)} \\ \text{s.t. } \dot{\mathbf{x}} - \mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0} && \text{(5.7b)} \\ \dot{\chi}_{\mathrm{ref}} - \frac{V_a}{g} \frac{\mathbf{e}_2 \mathbf{R}_{nb}{}^\top \mathbf{e}_3}{\mathbf{e}_3 \mathbf{R}_{nb}{}^\top \mathbf{e}_3} = 0 && \text{(5.7c)} \\ \dot{h}_{\mathrm{ref}} - (\mathbf{R}_{nb}\mathbf{R}_{bw} V_a \mathbf{e}_1 + \mathbf{v}^n_w)^\top \mathbf{e}_3 = 0 && \text{(5.7d)} \\ V_{a,\mathrm{ref}} - V_a = 0 && \text{(5.7e)} \\ \mathbf{u} = \mathbf{0} && \text{(5.7f)} \\ \mathbf{x} \in \mathcal{X}, && \text{(5.7g)} \end{aligned}$$

where the wind velocity vector $\mathbf{v}^n_{nw}$ is treated as a fixed parameter and the state-selection matrix is defined as $\mathbf{C}_{\phi\theta} \triangleq \mathrm{diag}(\mathbf{1}_3, \mathbf{e}_3, \mathbf{e}_3, \mathbf{e}_3, \mathbf{1}_7) \in \mathbb{R}^{n_x \times n_x}$ The constraints are included to enforce the turn rate in a coordinated turn Eq. (5.7c), the desired climb rate Eq. (5.7d) and the airspeed Eq. (5.7e). Let the optimizer of Eq. (5.7) be denoted by $\mathbf{x}^*$. The resulting output target is then extracted via

$$\begin{bmatrix} V_{a,\mathrm{ref}} & \mathbf{\Gamma}_{\mathrm{ref}}{}^\top \end{bmatrix}^\top_{\phi\theta} \triangleq \begin{bmatrix} \mathbf{e}^{n_x}_1 & \mathbf{e}^{n_x}_6 & \mathbf{e}^{n_x}_9 & \mathbf{e}^{n_x}_{12} \end{bmatrix}^\top \mathbf{x}^* \in \mathbb{R}^{n_y \times n_x}. \tag{5.8}$$

Pitch-Yaw Projection

As an alternative to references on rates of turn and climb, consider the objective of primarily controlling the direction of travel, i.e. the direction of the velocity vector of the vehicle. For this purpose, the axis to be controlled is the longitudinal axis of the UAV, which is obtained by the projection

$$\boldsymbol{\Gamma} = \mathbf{R}_{nb}\mathbf{e}_1 \tag{5.9}$$

Replace the roll angle with the yaw angle $\psi \in [-\pi, \pi]$, then the reduced-attitude vector for pitch-yaw control can be parameterized as

$$\boldsymbol{\Gamma}(\theta, \psi) = \begin{bmatrix} -\cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \end{bmatrix}^\top. \tag{5.10}$$

Again, one can use the parameterization by means of a subset of Euler angles given in Eq. (5.10). But in most cases a higher-level objective can be defined based on desired course angle χ_{ref} which is different from ψ when considering nonzero wind disturbances. A modified version of the optimization problem for output target generation for pitch-yaw control can be posed as

$$\begin{aligned}
J(\mathbf{v}_{nw}^n) := \min_{\mathbf{x}}\ & \dot{\mathbf{x}}^\top \mathbf{C}_{\theta\psi}{}^\top \mathbf{C}_{\theta\psi}\dot{\mathbf{x}} && \text{(5.11a)}\\
\text{s.t. } & \dot{\mathbf{x}} - \mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0} && \text{(5.11b)}\\
\chi_{\text{ref}} - \arctan \frac{\mathbf{e}_1{}^\top(\mathbf{R}_{nb}\mathbf{R}_{bw}\mathbf{e}_1 V_{a,\text{ref}} + \mathbf{v}_{nw}^n)}{\mathbf{e}_2{}^\top(\mathbf{R}_{nb}\mathbf{R}_{bw}\mathbf{e}_1 V_{a,\text{ref}} + \mathbf{v}_{w}^n)} & = 0 && \text{(5.11c)}\\
\dot{h}_{\text{ref}} - (\mathbf{R}_{nb}\mathbf{R}_{bw}V_a\mathbf{e}_1 + \mathbf{v}_w^n)^\top \mathbf{e}_3 & = 0 && \text{(5.11d)}\\
V_{a,\text{ref}} - V_a & = 0 && \text{(5.11e)}\\
\mathbf{u} & = \mathbf{0} && \text{(5.11f)}\\
\mathbf{x} & \in \mathcal{X}, && \text{(5.11g)}
\end{aligned}$$

with $\mathbf{C}_{\theta\psi} = \text{blkdiag}(\mathbf{I}_{6\times6}, \mathbf{0}_{13\times13}) \in \mathbb{R}^{n_x \times n_x}$. Now, instead of the turn-rate constraint Eq. (5.7c), the velocity vector is constrained to result in the course angle reference Eq. (5.11c). Let the optimizer of Eq. (5.11) be denoted by $\mathbf{x}^*$. The resulting output target then given by

$$\begin{bmatrix} V_{a,\text{ref}} & \boldsymbol{\Gamma}_{\text{ref}}{}^\top \end{bmatrix}_{\theta\psi}^\top \triangleq \begin{bmatrix} \mathbf{e}_1^{n_x} & \mathbf{e}_4^{n_x} & \mathbf{e}_4^{n_x} & \mathbf{e}_4^{n_x} \end{bmatrix}^\top \mathbf{x}^* \tag{5.12}$$

Remark 5.1. Let the set $\mathcal{X}_{\text{target}}$ denote the set of all states $\mathbf{x}$ in the preimage of either Eq. (5.8) or Eq. (5.12). And suppose that $\mathbf{F}$ denotes an integrator function of the continuous dynamics. Then it holds that $\mathbf{F} : \mathcal{X}_{\text{target}} \to \mathcal{X}_{\text{target}}$ for all states in $\mathcal{X}_{\text{target}}$. This means that $\mathcal{X}_{\text{target}}$ is invariant under the state dynamics at the target, which is a requirement towards a well-posed output for the NMPC problem.

Remark 5.2. The optimization problems defined in Eq. (5.7) - Eq. (5.7g) and Eq. (5.11) - Eq. (5.11g) are sensitive to changes in the wind velocity vector, that can either be caused by gust winds or dynamics of the estimator. Solving a target optimization problem adds to the complexity of the control architecture, and solutions may drastically change due to the non-convexity of the problems. Regularizing the problem with an additional cost term that penalizes differences from preceding solutions, similar to the presentation in Chapter 6 help to alleviate this problem. However, in practice it is preferable to apply the maps Eq. (5.6) and Eq. (5.10) directly for given reference signals. The roll reference signal for example may be found by inverting the coordinated turn equation [161], and analytical solutions for a mapping between the angles for course and flight path to yaw and pitch are discussed in [9, 161].

5.2.3 Constraints

At each time instant, the state and input are subject to polytopic constraints that arise from safety considerations and physical limits of the UAV. The state constraint set, denoted by $\mathcal{X}$, represents safety-related constraints due to the admissible flight envelope and actuator limits. We define it as

$$\mathcal{X} \triangleq \{\mathbf{x} \in \mathbb{R}^{n_x} | \mathbf{h}(\mathbf{x}, \mathbf{s}) \geq \mathbf{0}\}, \tag{5.13}$$

where $\mathbf{h}$ is a vector-valued function that is linear in the state variables, which we now discuss based on the example of the Skywalker X8. However, it is straight forward to extend the actuator suite. The Skywalker X8, as used in the experiments, is a flying wing with no tail. The set of actuators includes a throttle and two control surfaces on each side of the UAV.

Actuator limits can be expressed in terms of the true surface deflection limits of the elevons by substituting Eq. (2.37). The constraints include limits to the angle of attack and airspeed to avoid stalling and structural damage to the vehicle. Further, including the deflection limits, we get the set of constraints state constraints $\mathcal{X}$ defined by aerodynamic-related constraints

$$V_a - (1 + \epsilon_{V_a})\underline{V}_a + \underline{s}_{V_a} \geq 0 \tag{5.14a}$$
$$-V_a + (1 - \epsilon_{V_a})\overline{V}_a + \overline{s}_{V_a} \geq 0 \tag{5.14b}$$
$$\alpha - (1 - \epsilon_{\alpha})\underline{\alpha} + \underline{s}_{\alpha} \geq 0 \tag{5.14c}$$
$$-\alpha + (1 - \epsilon_{\alpha})\overline{\alpha} + \overline{s}_{\alpha} \geq 0 \tag{5.14d}$$
$$\beta - (1 - \epsilon_{\beta})\underline{\beta} + \underline{s}_{\beta} \geq 0 \tag{5.14e}$$
$$-\beta + (1 - \epsilon_{\beta})\overline{\beta} + \overline{s}_{\beta} \geq 0, \tag{5.14f}$$
$$\tag{5.14g}$$

constraints to limit the angular rates

$$p_s - (1 - \epsilon_{p_s})\underline{p}_s + \underline{s}_{p_s} \geq 0 \tag{5.15a}$$
$$-p_s + (1 - \epsilon_{p_s})\overline{p}_s + \overline{s}_{p_s} \geq 0 \tag{5.15b}$$
$$q_s - (1 - \epsilon_{q_s})\underline{q}_s + \underline{s}_{q_s} \geq 0 \tag{5.15c}$$
$$-q_s + (1 - \epsilon_{q_s})\overline{q}_s + \overline{s}_{q_s} \geq 0 \tag{5.15d}$$
$$r_s - (1 - \epsilon_{r_s})\underline{r}_s + \underline{s}_{r_s} \geq 0 \tag{5.15e}$$
$$-r_s + (1 - \epsilon_{r_s})\overline{r}_s + \overline{s}_{r_s} \geq 0, \tag{5.15f}$$
$$\tag{5.15g}$$

and actuator constraints

$$\delta_{\mathrm{a}} + \delta_e - \underline{\delta}_{\mathrm{el}} \geq 0 \tag{5.16a}$$

$$-\delta_{\mathrm{a}} - \delta_e + \overline{\delta}_{\mathrm{el}} \geq 0 \tag{5.16b}$$

$$-\delta_{\mathrm{a}} + \delta_e - \underline{\delta}_{\mathrm{er}} \geq 0 \tag{5.16c}$$

$$\delta_{\mathrm{a}} - \delta_e + \overline{\delta}_{\mathrm{er}} \geq 0 \tag{5.16d}$$

$$\delta_{\mathrm{t}} - \underline{\delta}_{\mathrm{t}} \geq 0 \tag{5.16e}$$

$$-\delta_{\mathrm{t}} + \overline{\delta}_{\mathrm{t}} \geq 0, \tag{5.16f}$$

which include slack variables for constraint relaxation to guarantee the feasibility of the quadratic problem (QP). The slack variables are part of the NLP formulation as a concatenated vector $\mathbf{s} \in \mathbb{R}^{n_s}$ with the elements in the order that they appear in Eq. (5.14a) - Eq. (5.15f). The slack variables are constrained non-negative real numbers, i.e.

$$\mathbf{s} \geq 0. \tag{5.17}$$

The tightening parameters ϵ_* allow for tightening of the original constraints for a robust MPC formulation. The tightening parameter can be chosen as a static back-off parameter [66] or based on the exponential contraction rate to the reference [99]. The concatenation of Eq. (5.14a) - Eq. (5.16f) and Eq. (5.17) gives $\mathbf{h}(\mathbf{x}, \mathbf{s}) \geq \mathbf{0}$, where the inequality is to be interpreted element-wise.

Remark 5.3. Since the control surface deflections δ_{el}, δ_{er} and the throttle set-pint δ_t are determined by the controller, it is not necessary to use slack variables in the associated constraints. However, there are practical scenarios when another controller is active while the NMPC may run in the background with initial conditions set by state estimates and control signals. In a case where it can not be ensured that the surface deflections of the alternative controller do not exceed the limits of the NMPC, slack variables would be necessary.

The control variables $\mathbf{u}$ do not require slack variables, considering that the rates of the actuators are virtual variables used in the NMPC. The output of the controller will be the actual actuator set-points after one integration step, which we will discuss later. For now, let the set of input constraints $\mathcal{U}$ be defined by

$$\mathcal{U} \triangleq \{\mathbf{u} \in \mathbb{R}^{n_u} | \mathbf{u} - \underline{\mathbf{u}} \geq \mathbf{0} \wedge -\mathbf{u} + \overline{\mathbf{u}} \geq \mathbf{0}\} \tag{5.18}$$

for some bounds $\underline{\mathbf{u}}$, $\overline{\mathbf{u}} \in \mathbb{R}^{n_u}$. Practical rate limits of the control surfaces deflections can, for example, be found in a motion capture lab based on a step change from minimum to maximum deflection and vice versa. Similarly test for the engine can be conducted if measurement equipment for the power output is available. We did not see actuator rate limits to be a problem in practice. However, the Skywalker X8 comes with an electric engine, and the situation may be different in case of combustion engines that only allow for significantly slower changes of the speed set-point.

For a more compact notation, let the constraint in Eq. (5.14a) - Eq. (5.16f), Eq. (5.17) and Eq. (5.18) be concatenated by $\mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{s}) \geq \mathbf{0}$, where the inequality again denotes an element-wise operation.

Remark 5.4. We did not discuss constraints on the roll and pitch angle, which are common elements in open-source flight controllers. They can be included as nonlinear constraints of the state vector through the mapping between elements of SO(3) and Euler angles in the yaw-pitch-roll convention. A box constraint that couples roll and pitch limits can be formulated using Eq. (5.5) as

$$\mathbf{\Gamma}^\top \mathbf{e}_3 = \mathbf{e}_3^\top \mathbf{R}_{nb} \mathbf{e}_3 = {\mathbf{e}_{12}^{n_x}}^\top \mathbf{x} \leq \cos(\Theta) \tag{5.19}$$

with Θ denoting the maximum spherical angle between the $\mathbf{\Gamma}$ and the vertical axis of the inertial frame.

5.2.4 Disturbance Observer

For offset-free attitude stabilization in the presence of e.g. unmodelled dynamics, parametric disturbances or estimation errors, we include an observer for moment and force disturbances that affect angular rate and airspeed. This follows the general discussion on offset-free NMPC in [134]. Consider the observed airspeed V_a and angular rate $\boldsymbol{\omega}_{nb}^b$ and the difference to their predictions in the controller at each time instant

$$\Delta V_a(t) = V_a(t) - (\mathbf{e}_1^{n_x})^\top \mathbf{x}^*(1|t-1) \tag{5.20}$$

$$\Delta \boldsymbol{\omega}_{nb}^b(t) = \boldsymbol{\omega}_{nb}^b(t) - \mathbf{R}_{bs} \begin{bmatrix} \mathbf{0}_{3\times 12} & \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} \end{bmatrix} \mathbf{x}^*(1|t-1) \tag{5.21}$$

The disturbance estimates are initialized as $\mathbf{d}_f(0) = \mathbf{d}_m(0) = \mathbf{0}_{3\times 1}$ and can simply be updated continuously together with the NMPC as

$$\mathbf{d}_f(t) \leftarrow \mathbf{d}_f(t) + l_f \begin{bmatrix} \Delta V_a(t) & 0 & 0 \end{bmatrix}^\top \tag{5.22}$$

$$\mathbf{d}_m(t) \leftarrow \mathbf{d}_m(t) + \mathbf{L}_m \Delta \boldsymbol{\omega}_{nb}^b(t) \tag{5.23}$$

with $l_f \in \mathbb{R}$ as the learning gain for the force disturbance and $\mathbf{L}_m \in \mathbb{R}^{3\times 3}$ defined as

$$\mathbf{L}_m = \operatorname{diag}(l_p, l_q, l_r) \tag{5.24}$$

acting as the learning gain for the moment disturbance which is the diagonal composition of the individual gains for the angular rates $l_p, l_q, l_r \in \mathbb{R}$. In the following, we will concatenate the disturbances to

$$\mathbf{d} = \begin{bmatrix} {\mathbf{d}_f}^\top & {\mathbf{d}_m}^\top \end{bmatrix}^\top \tag{5.25}$$

and augment $\mathbf{f}$ defined in Section 5.2.1 with the disturbance as follows

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}) \triangleq \mathbf{f}(\mathbf{x}, \mathbf{u}) + \begin{bmatrix} {\mathbf{d}_f}^\top & \mathbf{0}_{1\times 3} & {\mathbf{d}_m}^\top \end{bmatrix}^\top . \tag{5.26}$$

It is also possible to integrate the dynamic model in a separate simulator that runs in parallel to the onboard estimators at the same update rate. The integration of the dynamic model in an explicit integration scheme is computationally cheap compared to solving the NLP online in the controller. At each controller update, the state of the parallel simulator is reset to the initial conditions of the NMPC, which ensures that the disturbance estimates are not diverging. The dynamic of the disturbances in the controller and the simulator are modeled as slowly time-varying, which we express as

$$\dot{\mathbf{d}} = \mathbf{0} \tag{5.27}$$

This is approach works well in practice, as we will show in the experimental validation.

Remark 5.5. We also experimented with an EKF implementation to provide the disturbance estimates. The EKF runs in parallel to the onboard state estimator, which is another EKF in our experimental platform. However, there were no significant performance improvements in simulations that would justify the more complex tuning and potential instabilities. Further work may include the integration of the disturbance estimates in the existing EKF.

5.2.5 Cost and Nonlinear Program

In the following we assume that a suitable reference for the controller has been found. This can be done either through the outlined methods in the preceding chapter or using the nominal cruise speed and Euler angle references that are mapped to the reduced attitude using Eq. (5.6) or Eq. (5.10). Let the resulting reference vector be defined as

$$\mathbf{r} \triangleq \begin{bmatrix} V_{a,\mathrm{ref}} & {\boldsymbol{\Gamma}_{\mathrm{ref}}}^\top \end{bmatrix}^\top \tag{5.28}$$

To formulate the NMPC scheme for output tracking, we define the stabilizing stage cost as a sum of quadratic terms

$$l(\mathbf{x}, \mathbf{u}, \mathbf{r}) = q_{V_a}(V_a - V_{a,\mathrm{ref}})^2 + \|\boldsymbol{\Gamma} - \boldsymbol{\Gamma}_{\mathrm{ref}}\|^2_{\mathbf{Q}_{\boldsymbol{\Gamma}}} + \|\mathbf{u}\|^2_{\mathbf{R}}, \tag{5.29}$$

which includes the positive definite and symmetric weighting matrices $\mathbf{Q} \in \mathbb{R}^{3\times 3}$, $\mathbf{R} \in \mathbb{R}^{n_u \times n_u}$ and the positive scalar q_{V_a}. To shed some light on the geometric interpretation of the least-squares penalty term regarding the attitude, note that

$$\frac{1}{2}\|\boldsymbol{\Gamma} - \boldsymbol{\Gamma}_{\mathrm{ref}}\|^2 = 1 - \boldsymbol{\Gamma}^\top \boldsymbol{\Gamma}_{\mathrm{ref}} = 1 - \cos(\Theta) \tag{5.30}$$

where $\Theta \in [0, \pi]$ denotes the spherical angle between $\boldsymbol{\Gamma}$ and $\boldsymbol{\Gamma}_{\mathrm{ref}}$. This implies that there exists a unique minimum where the angle between both axes is zero. A more in-depth discussion of this transformation on the manifold can be found in Chapter 4 or more general in [19]. We will use this intuition to derive tuning guidelines in the coming section.

Using the disturbance-augmented dynamics in Eq. (5.26) and a prediction horizon T, the OCP reads as

$$\begin{aligned}
&\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \int_0^T l(\mathbf{x}(\tau), \mathbf{u}(\tau), \mathbf{r}(\tau)) d\tau + \frac{1}{2}\mathbf{s}^\top \mathbf{P} \mathbf{s} && t \in [0, T) && (5.31\text{a}) \\
&\quad \text{s. t.} \quad \mathbf{x}(0) = \mathbf{x}_0 && && (5.31\text{b}) \\
&\qquad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(0) && t \in [0, T) && (5.31\text{c}) \\
&\qquad \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{s}) \geq \mathbf{0} && t \in [0, T), && (5.31\text{d})
\end{aligned}$$

which includes an additional cost term to penalize the slack variables with a symmetric and positive-definite weighting matrix $\mathbf{P} \in \mathbb{R}^{4\times 4}$. The magnitude of the elements in $\mathbf{P}$ should be significantly higher in comparison to $\mathbf{Q}$ and $\mathbf{R}$ to approximate hard constraints in nominal operation, but allow for relaxed constraints when necessary.

We use direct multiple-shooting [13] with an explicit Runge-Kutta integration scheme of order four to integrate $\mathbf{f}$ and let $\mathbf{f}_{\text{RK4}}$ denote the resulting integrator function. The system is discretized into N steps with the resulting shooting interval $\Delta t = T/N$. The MPC scheme is then based on solving the NLP at time t for the predictions at $k \in [0, ..., N]$. Let a predicted state and input sequence be denoted by $\mathbf{x}(\cdot|t) \in \mathbb{R}^{n_x \times (N+1)}$, $\mathbf{u}(\cdot|t) \in \mathbb{R}^{n_u \times N}$. The resulting then NLP reads

$$\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \sum_{k=0}^{N-1} l(\mathbf{x}(k|t),\mathbf{u}(k|t),\mathbf{r}(k|t)) + \frac{1}{2}\mathbf{s}^\top \mathbf{P}\mathbf{s} \qquad k \in [0, ..., N] \tag{5.32a}$$

$$\text{s. t.} \quad \mathbf{x}(0|t) = \mathbf{x}(t) \tag{5.32b}$$

$$\mathbf{x}(k+1|t) = \mathbf{f}_{\text{RK4}}(\mathbf{x}(k|t),\mathbf{u}(k|t),\mathbf{d}(0|t)) \qquad k \in [0, ..., N] \tag{5.32c}$$

$$\mathbf{h}(\mathbf{x}(k|t),\mathbf{u}(k|t),\mathbf{s}) \geq 0 \qquad k \in [0, ..., N], \tag{5.32d}$$

with the estimated state $\mathbf{x}(t)$ as initial condition. Note that we do not employ a terminal cost or terminal constraint set, which is common practice in MPC for aerospace applications that may be subject to severely turbulent and uncertain dynamics [45]. We also observed that simulations show already sufficiently large region of attraction for rather low prediction horizons, suggesting that terminal conditions may not be needed [68]. Given the computation of the state target however, the presented scheme can be readily extended following the discussion in [31] to not only improve stability, but also increase closed-loop performance [151]. We will discuss some intuitive tuning guidelines and evaluation of the closed-loop runtime on the targeted hardware is given in the following sections.

The control signals that are propagated to the actuators are extracted from the optimal state after one shooting interval denoted by $\mathbf{x}^*(1|t)$. The control commands to the vehicle, denoted by $\mathbf{u}_{\text{uav}}$ are thus given by

$$\mathbf{u}_{\text{uav}}(t) = \begin{bmatrix}\mathbf{0}_{n_u\times(n_x-n_u)} & \mathbf{I}_{n_u\times n_u}\end{bmatrix} \mathbf{x}^*(1|t). \tag{5.33}$$

Using the control signal after the one shooting interval worked well in simulations and in practice for a discretization interval $\Delta t = 0.1\,\text{s}$. We came to a similar observation when we targeted the attitude control problem with the same platform (Skywalker X8) using DRL in [14]. Successful experiments required the DRL controller to be trained with a significant actuator delay, and an actuator delay of $0.1\,\text{s}$ gave satisfactory results. Note however that this might be that the true accumulated delays due to communication and actuators might be shorter and that more elaborate schemes can be implemented based on integration of the optimal actuator rate $\mathbf{u}^*(0|t)$. This can be useful when accurate knowledge of communication delays is available. If this is not the case, the integration step can still be used as a tuning parameter. Note that our MPC formulation allows for time-varying reference signals to be used in the controller. However, future references for the

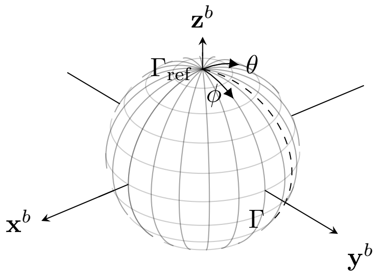

Figure 5.1: Illustration of how the reduced-attitude vector is parameterized by the angles roll ϕ and pitch θ. In the tuning example, the attitude Γ is to be steered to its reference Γ_{ref} along the dashed curve.

low-level controller are usually not available in most UAV control architectures, except for particular applications such as aerobatic maneuvering [146]. Assuming a constant reference signal over the prediction horizon is therefore the best approach in a conventional operation.

5.2.6 Tuning

The geometric formulation of the control objective allows for some constructive tuning guidelines which we briefly outline in this subsection. We follow standard arguments in the proof of stability and performance properties for NMPC without terminal ingredients [68] in which the closed-loop stage cost is upper bounded by a class $\mathcal{KL}_0$ function. For better performance and stability at small N, it is necessary to shape the cost in such a way as to allow for a faster decay of the bounding function. In essence, a class $\mathcal{KL}$ is a function that maps two arguments to $\mathbb{R}$. A special property is that the output of the function is monotonically increasing with increasing its first argument, and monotonically decreasing with increasing second argument. Class $\mathcal{KL}$ are essential in nonlinear control theory and more details on class $\mathcal{KL}_0$ functions can be found in [95].

We discuss this with an illustrative example for the roll-pitch controller, but the conceptual idea can be readily carried over to the pitch-yaw controller case. Consider the parts of the output vector that correspond to the attitude evolve on the two-sphere depicted in Fig. 5.1 and keep other state and input variables constant. Tuning the part of the weighting matrix that corresponds to the attitude can be thought of as a scaling of the sphere along its three axes. One can see that in nominal flight conditions, the attitude vector evolves on the top part of the sphere at a low roll and pitch angle. The lower hemisphere, in contrast, corresponds to a situation in which the UAV is far from a nominal flight and either conducts an acrobatic maneuver or needs to be recovered to nominal conditions.

In the constructed example, the current attitude $\mathbf{\Gamma}$ is on the lower hemisphere with the reference $\mathbf{\Gamma}_{\text{ref}}$ at zero roll and pitch angle. In this case, the first and second elements of $\mathbf{\Gamma}$ would increase in magnitude before decreasing again when the upper hemisphere is reached. A higher weight on these elements would potentially lead to a temporary increase of the cost function (overshoot). In that case, the horizon

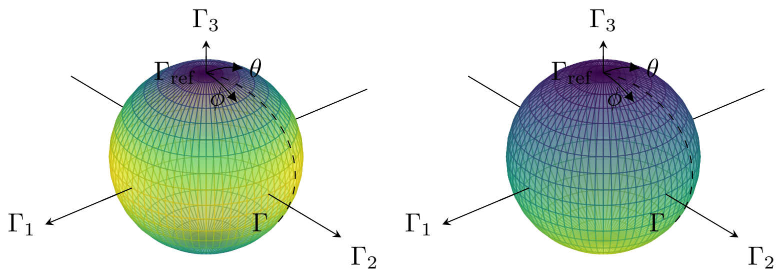

Figure 5.2: Illustration of how the reduced-attitude vector is parameterized by the angles roll ϕ and pitch θ. In the tuning example, the attitude Γ is to be steered to its reference Γ_{ref} along the dashed curve.

N has to be long enough to include a decrease of the stage cost in the predicted trajectory. Increasing the weight on the third element however helps to flatten the overshoot and results in monotonic decrease of the stage cost, which allows for shorter control horizons. This is illustrated in Fig. 5.3 for a scaled version of Eq. (5.30), i.e. $l_{\mathbf{\Gamma}} = 0.5\|\mathbf{\Gamma} - \mathbf{\Gamma}_{\text{ref}}\|^2_{\mathbf{Q}}$.

To tune the controller for the recovery case, a heuristic tuning procedure would be to increase the horizon length to a level at which the available hardware can update the controller at the desired rate and then increase the weight on Γ_3 in a trial-and-error procedure on numerical simulations.

When nominal flight conditions are considered, a higher weight on Γ_1 or Γ_2 will result in tighter tracking of pitch or roll angle, respectively. When a set of suitable entries of the weighting matrix for the attitude is found, one can then tune the element corresponding to the airspeed. A general guideline here is to increase the weighting relative to the attitude for the expected level of turbulence to keep the UAV in stable conditions. The weights on the actuator rates should in general be tuned to be low relative to the weights that correspond to parts of the state vector, as is often done in practice. Too low weights however may lead to chattering, which may be readily observed in numerical simulations.

5.2.7 Implementation and Hardware

We implement the NMPC and the simulator for the disturbance observer using the open-source software package acados [185] and employ the Realtime-Iteration sequential quadratic programming (SQP) solver based on [46] with the high-performance interior point method (HPIPM) presented in [57] for the solutions of the underlying QPs. The auto-generated C/C++ code is then interfaced in dynamic unified navigation environment (DUNE) [148], which is part of the tool chain that runs on the SBC during operation to handle inter-process communication and compute the optimal control inputs onboard.

It will request estimates of the vehicle state and the wind velocity vector from the onboard navigation system that is part of the standard avionics flight stack centered around a Pixhawk/Cube Orange[1] that is running Ardupilot, as outlined

[1]https://cubepilot.org/

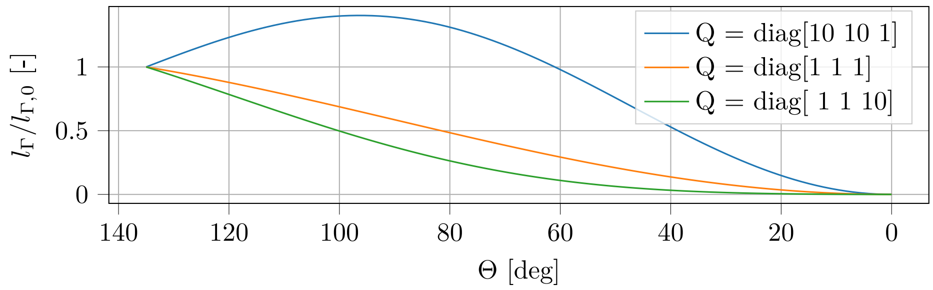

Figure 5.3: Different cost shaping for the constructed example. The tuning of the blue trajectory requires a longer horizon N due to the overshoot, and a less desirable performance compared to the tuning for the green trajectory.

in Chapter 3. The companion SBC to run the MPC onboard is a Khadas Vim3[2] which includes four 2.2Ghz Cortex-A73 cores and two 1.8Ghz Cortex-A53 cores. The code to run the NMPC is cross-compiled on a laptop computer with four 2.7GHz i7-7500 CPUs which takes about two seconds.

5.3 Simulation Study

The initial evaluation of the closed-loop runtime of the controller is designed as a lab experiment. We conduct a simulation study in which the problem data is generated on a lab computer. The initial conditions, output target values, and parameters in each controller update are then sent to the single-board computer embedded in the aircraft where the successive controller updates are repeated to obtain measurements of the achievable update rates. For this test, the DUNE task that would retrieve current initial conditions from the communication with ArduPilot instead reads them from local data files and sends them as messages over the IMC protocol.

5.3.1 Simulation Setup

To assess closed-loop performance, stability, and the real-time applicability of the proposed NMPC, we conduct a Monte-Carlo simulation study in with varying initial conditions and environmental disturbances. The length of the shooting interval of the controller is set to $\Delta t = 0.1s$. We ran a series of simulations for varying horizon length given by $N \in \{10, 15, 20, 25, 30, 35, 40, 45\}$. The update rate of the simulation is set to $f_{\text{sim}} = 100\text{Hz}$ with controller updates at $f_{\text{mpc}} = 10\text{Hz}$. The dynamic model of the aircraft is based on the Skywalker X8 as identified in [71] and discussed in Chapter 2.

The actuators in the simulation are modelled with first-order lag dynamics, i.e.

$$\dot{\delta}_i = \frac{1}{T}(u_{\text{uav},i} - \delta_i) \tag{5.34}$$

[2]https://www.khadas.com/vim3

Table 5.1: Inequality constraints used in the controller.

Variable	*unit*	*min*	*max*
V_a	[m/s]	15.0	25.0
β	[deg]	-90.0	90.0
α	[deg]	-15.0	27.0
p^s	[deg]	-180.0	180.0
q^s	[deg]	-180.0	180.0
r^s	[deg]	-180.0	180.0
δ_a	[deg]	-35.0	35.0
δ_e	[deg]	-35.0	35.0
δ_r	[deg]	0.0	0.0
δ_t	[-]	0.0	1.0

with $T = 0.01$ for the control surfaces and $T = 1$ for the throttle. The wind is modelled as the combination of a static component in the inertial frame and a gust component in the body-fixed frame $\mathbf{v}^n_{nw} = \mathbf{v}^n_{nw,s} + \mathbf{R}_{nb}\mathbf{v}^b_{nw,g}$. The gust component is generated by the Dryden wind model, which means essentially passing white noise through a low-pass filter as discussed by Beard and McLain [9]. We selected the parameters of the Dryden process (length scale, intensity) to simulate moderate turbulence. Both the propagation of the state and the controller updates are based on the same model to simulate ideal conditions. The effect of model mismatch will be the subject of a following subsection when we look at the performance in experiments.

5.3.2 Monte-Carlo Simulations

To establish the robustness results of the controller, we conduct a brief Monte-Carlo simulation study. The aerodynamic quantities, attitude, angular rate vector, and the wind velocity vector in NED were sampled from a uniform distribution with bounds as summarized in Table 5.3. For each prediction horizon N, we ran 30 simulations with the initial state and wind velocity vector drawn from the distribution. The constraints for the controller are given in Table 5.1. The motivations for the constraints are twofold. First, restrict the aerodynamic quantities to ensure a stable flight regime, i.e. stay within the flight envelope that is bounded by stall angle and stall speed, and at the same time limit the load factor acting on the airframe through the angular rate constraints. Note that initial conditions outside the constraint set of the controller are part of the simulation study. In these cases, the high weighting on the slack variables will cause the controller to first try to reach the constraint set and then stabilize the given reference.

We follow the tuning procedure outlined in Section 5.2.6 and use the cost ma-

Table 5.2: Initial condition for the edge cases.

Variable	***unit***	0	1	2	3
V_a	[m/s]	10.0	15.0	25.0	30.0
β	[deg]	-10.0	-10.0	10.0	10.0
α	[deg]	33.0	27.0	-15.0	-20.0
ϕ	[deg]	-100.0	-100.0	100.0	100.0
θ	[deg]	20.0	20.0	20.0	20.0

trices

$$\mathbf{Q} = \operatorname{diag}(0.1, 1, 1, 50), \tag{5.35a}$$

$$\mathbf{R} = 10^{-3}\mathbf{I}_{n_u \times n_u}, \tag{5.35b}$$

$$\mathbf{P} = \operatorname{diag}(1, 1, 10^3, 1, 1, 10^3). \tag{5.35c}$$

This is to penalize deviations from the reference significantly higher than actuator rates while still maintaining a high enough penalty to avoid chattering. A different factor that prevents fast oscillations of the actuators is the rather low update rate of the controller itself. It is also notable that violations of the airspeed and sideslip constraint are tolerated a lot more than violations of the angle of attack constraint.

The Monte-Carlo simulation has two objectives. The first objective is to illustrate that the NMPC can recover the aircraft from a wide range of initial conditions into a wings-level horizontal flight at nominal cruise speed. We use the proposed roll-pitch controller to execute these maneuvers with input triplet $(\dot{h}_{\text{ref}}, \dot{\chi}_{\text{ref}}, V_{a,\text{ref}}) = (0, 0, 18)$ to generate the target outputs for the current wind conditions. The second objective is to record simulation data to gauge the achievable update rates.

5.3.3 Edge Cases

To illustrate the closed-loop behavior for $N = 10$ and $N = 45$, we simulate different initial conditions on the edge of the constraint set and outside it. The actuators and angular rates are initialized to zero. The state variables that correspond to the initial attitude and velocity for each case are summarized in Table 5.2. The reference for each case is set to $(\phi, \theta, V_a) = (0, 0, 18)$, given that the primary goal is a stable flight condition. The wind-velocity vector is set to be constant at $v_w^n = [-4, -3, 0]$.

5.3.4 Discussion of the Simulation Results

The edge cases are illustrated based on their aerodynamic quantities in Fig. 5.4, output error trajectories in Fig. 5.5 and the applied control signals in Fig. 5.6. For each initial condition, $N = 10$ is plotted in dashed and $N = 45$ with solid lines. For all scenarios, the controller can recover the UAV from the initial condition. It is clear from Fig. 5.4 that when driving the state trajectory into the constraint set, the angle of attack constraint is prioritized as encoded in the weighting matrix $\mathbf{P}$.

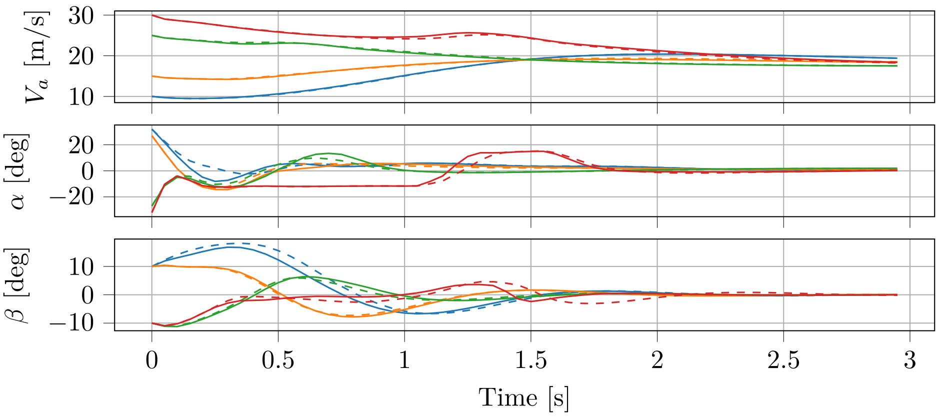

Figure 5.4: Aerodynamic quantities including airspeed V_a, angle of attack α, and sideslip angle β.

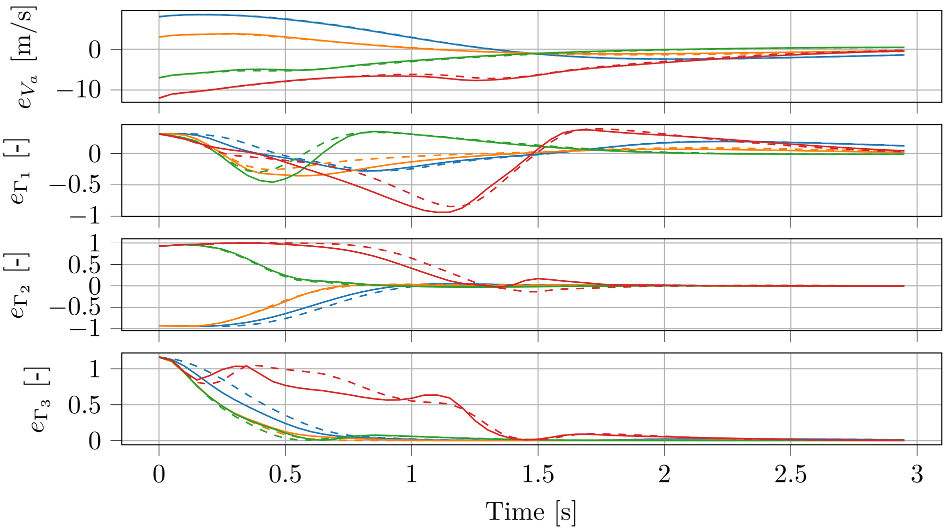

Figure 5.5: Output error including the airspeed error in the first subplot and the reduced attitude error components in the bottom three subplots.

Table 5.3: Bounds for the uniform distribution of initial state and wind conditions.

Variable	*unit*	*min*	*max*
V_a	[m/s]	10.0	30.0
β	[deg]	-45.0	45.0
α	[deg]	-15.0	40.0
ϕ	[deg]	-100.0	100.0
θ	[deg]	-20.0	20.0
p	[deg]	-50.0	50.0
q	[deg]	-50.0	50.0
r	[deg]	-50.0	50.0
v_{w_x}	[m/s]	-5.0	5.0
v_{w_y}	[m/s]	-5.0	5.0
v_{w_z}	[m/s]	-1.0	1.0

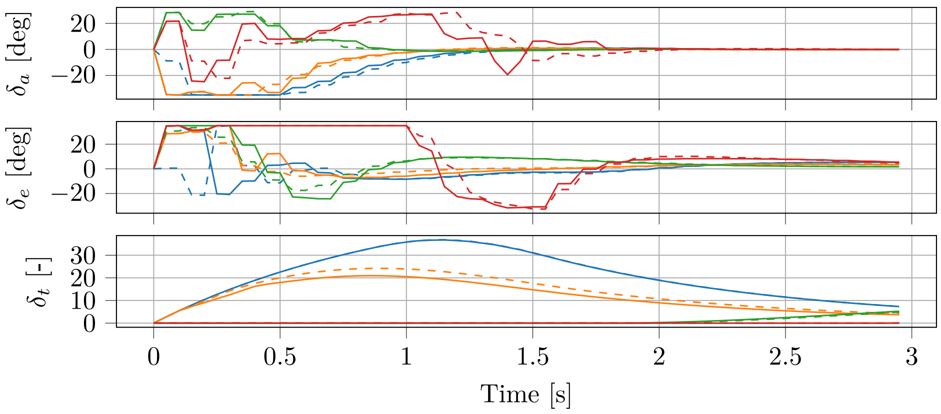

Figure 5.6: Control signals from the controller for the edge cases.

Both prediction horizons, $N = 10$ and $N = 45$, result in a very similar closed-loop trajectory, which suggests that stability can be achieved at shorter prediction horizons with only a minor loss of performance.

However, it is worth noting that in all cases of the simulation, the time needed by the controller to update the solution does not exceed 100 ms, even at the start of the maneuver (c.f. Fig. 5.8) where the longest closed-loop runtime can be expected due to the initialization of the numerical solver.

Further, note that except for a low amount of outliers, along almost all trajectories the closed-loop cost of the NMPC converges within one second as does the time the solver needs to update the solutions to the new problem, with 4.6% of the simulation runs showing closed-loop solver runtimes exceeding 25 ms at 1 s into the simulation. This is despite moderate wind turbulence being present which has the effect of pushing the UAV continuously away from the predicted closed-loop trajectory.

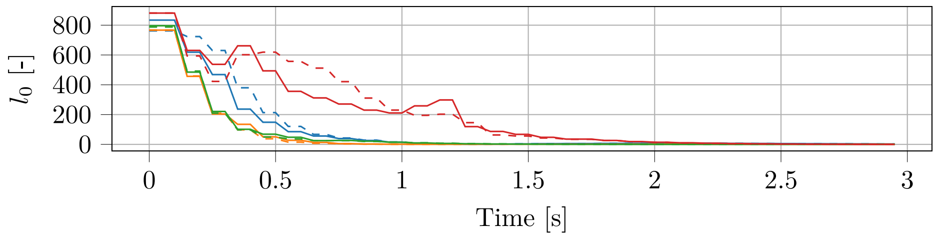

Figure 5.7: Cost for the edge cases at the first interval.

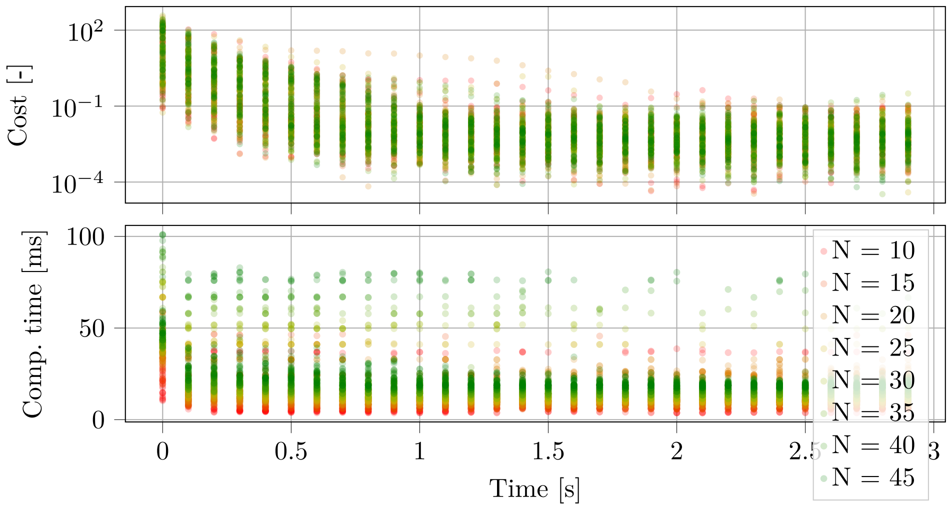

Figure 5.8: Cost evaluated at each controller update.

These simulations imply that even though the UAV is initialized in an extreme condition with large attitude errors, the controller is still able to update the actuator signals reasonably fast to recover the UAV into a nominal flight condition at the reference attitude. In the worst-case employing update rates of 10 Hz can be expected to increase to 40 Hz for $N = 45$. In the case of $N = 10$, even 100Hz update rates may be achieved, which is in the range of established low-level autopilots using PID-type controllers. This implies a desirable length of update periods for the low-level controller onboard modern UAV in flight operations. However, this observation hinges on the assumption of perfect model knowledge, which can not be satisfied in practice.

So far, we showed that the presented NMPC can incorporate higher-level objectives such as turn-rate control or course control with simultaneous control of climb rate and airspeed. Monte-Carlo simulations demonstrated that the UAV can be recovered from rather large attitude errors and stabilize the aircraft in nominal flight conditions at the reference attitude. The most crucial part in our discussion is the fact that the presented controller can be updated at high rates even for complex models and that the closed-loop runtime of the nonlinear solver is not a bottleneck for a successful application in practice, even with the rather fast dynamics of the UAV. We are now prepared to test the controller in practical experiments, which will be the subject of the next section.

5.4 Experimental Verification

In this section we will first look at the implementation of the controller in the experiments and the parameterization of the baseline controller. The results of the experiments follow after a brief discussion on how to assess the model mismatch and experiment descriptions. Finally, the comparative benefits of the MPC in contrast to the baseline controller will be discussed based on the different test scenarios.

5.4.1 NMPC Implementation

The update frequency of the MPC is set to 20 Hz. We chose a time horizon $T = 3\,\mathrm{s}$ split into $N = 30$ shooting nodes, which corresponds to $\Delta t = 0.1$. The cost matrices to parameterize the NLP are

$$\mathbf{Q} = \mathrm{diag}([0.001, 10, 10, 10]) \tag{5.36a}$$
$$\mathbf{R} = \mathrm{diag}([1, 1, 0.01]) \tag{5.36b}$$
$$\mathbf{P} = \mathrm{diag}([100, 100, 1000, 1000]). \tag{5.36c}$$

The weight for airspeed tracking is significantly lower compared to the attitude weights. We noticed in initial experiments that the MPC was struggling with negative pitch references in which the UAV would necessarily increase speed, such that tracking of the airspeed and pitch reference were conflicting objectives. The outlined tuning prioritizes attitude tracking until the upper airspeed constraint is reached, but still gives satisfactory airspeed tracking in cases when the throttle is not saturated to the lower limit. The weights in the slack penalty matrix **P** are significantly larger to allow for constraint relaxation when this is necessary for the QP solver to converge, but nominally approximate hard constraints.

The gains for the disturbance observer were set to $l_{V_a} = l_r = 0.1$ and $l_p = l_q = 0.5$ The observer is updated at the same rate as the MPC. The gains for the moments in roll and pitch direction were slightly increased based on the observation of the model mismatch. We will discuss inaccuracies of the model together with the performance in the next section. The given disturbance observer gains gave satisfactory results after a short tuning procedure based on a sequence of step responses. The parameters for the constraint set are given in Table 5.4. Note that we do not constrain the sideslip angle or angular rates.

5.4.2 Observed and modeled Accelerations

This work is the first to use the models presented in [39, 71] for flight control in experiments. We therefore include an assessment of the model quality for a subset of the test sequences based on the difference between observed and modeled accelerations. The observed linear acceleration is indicated by a subscript z and its

Table 5.4: NMPC parameters.

Parameter	Value	Unit
$\underline{V}_a$, $\overline{V}_a$	12.0, 25.0	[m/s]
$\underline{\alpha}$, $\overline{\alpha}$	-10.0, 12.0	[deg]
$\underline{\delta}_{el}$, $\underline{\delta}_{er}$, $\overline{\delta}_{el}$, $\overline{\delta}_{er}$	-30, -30, 20, 20	[deg]
$\underline{\delta}_t$, $\overline{\delta}_t$	0.0, 1.0	[-]

computation follows standard literature [9]

$$\dot{\mathbf{v}}^b_{nb,z} = \mathbf{a}^b_{nb} + {\mathbf{R}_{nb}}^\top \mathbf{g}^n - \boldsymbol{\omega}^b_{nb} \times \mathbf{v}^b_{nb}, \tag{5.37}$$

where $\mathbf{a}^b_{nb} \in \mathbb{R}^3$ denotes the acceleration measured by the IMU. The observed angular acceleration $\dot{\boldsymbol{\omega}}^b_{nb} \in \mathbb{R}^3$ is computed using the centered difference formula based on the estimated angular velocity. The accelerations obtained through the model, indicated by a subscript y, are given by

$$\dot{\mathbf{v}}^b_{nb,y} = \frac{1}{m}({\mathbf{R}_{wb}}^\top \mathbf{f}^w_{\mathrm{a},y} + \mathbf{f}^b_{\mathrm{t},y}) + {\mathbf{R}_{nb}}^\top \mathbf{g}^n - \boldsymbol{\omega}^b_{nb} \times \mathbf{v}^b_{nb} \tag{5.38}$$

$$\dot{\boldsymbol{\omega}}^b_{nb,y} = \mathbf{J}^{-1}(\mathbf{J}\boldsymbol{\omega}^b_{nb} \times \boldsymbol{\omega}^b_{nb} + \mathbf{m}^b_a). \tag{5.39}$$

5.4.3 Experiment Description

We examine the performance of the MPC in comparison to the ArduPilot Controller based on a series of tests for roll and pitch angle references. The ArduPilot attitude controller implements separate, cascaded SISO feedback loops for the roll and pitch channels to control aileron and elevator, respectively. The control laws are based on Release 4.0.9, which is the most recent stable release (as of August 2021). The outer loop consists of proportional controllers, where desired roll and pitch rates are calculated based on the error in the respective angles. Details of the structure of the baseline controllers are given in Section 2.5.

The parameters of the baseline controller were tuned in a manual procedure based on visual inspection of the response to reference tracking in a rectangular pattern and individual step responses in roll and pitch. The resulting parameters that were used in the experiments are summarized in Table 5.5.

Table 5.5: AP parameters.

Parameter	Value
k_ϕ, $k_{ff,p}$, $k_{p,p}$, $k_{i,p}$	2.222, 0.312, 0.041, 0.015
k_θ, $k_{ff,q}$, $k_{p,q}$, $k_{i,q}$	2.222, 0.197, 0.023, 0.0113
k_{p,V_a}, k_{i,V_a}	0.25, 0.1

The integral terms in the baseline control laws are implemented using a first order forward Euler method with a discretization step of 0.02 s, corresponding to the control frequency of 50 Hz. To avoid integral windup issues, we use conditional integration. The baseline controller is implemented on the SBC to use the same hardware and communication pipeline as the MPC, which ensures a fair comparison. Through the rest of the paper we will follow the convention $e = y_{\text{ref}} - y$ to report tracking errors.

In all tests, unless explicitly stated, the commanded airspeed reference is set to the nominal cruise speed of the Skywalker X8, which is 18 m/s

We initially test the tracking performance to smooth and continuously changing reference signals coming from a higher-level guidance controller.

During a step sequence for the roll angle reference, the roll angle is commanded to zero for two seconds, then to −50 deg for three seconds and to 50 deg for three seconds before finishing with another two seconds at zero. The reference of the pitch angle is kept constant at the last value before initiating the step sequence.

During a step sequence for the pitch angle reference, the roll angle is kept at zero throughout the entire sequence. Similar to the roll angle sequence, the pitch angle reference is set to zero for two seconds at the beginning and end of the sequence with −20 deg and 20 deg for three seconds in between.

To gauge how both controllers compare in maneuvers where the roll and pitch dynamics are excited simultaneously, we use step sequences, oscillating references and a simultaneous constant reference for 30 s. All of these sequences use the same magnitude of the individual step sequences.

The oscillating references have a frequency $f = 0.3\,\text{Hz}$ and are defined as

$$\phi_{\text{ref}}(t) = 50\frac{\pi}{180}\cos(2\pi f t), \quad \theta_{\text{ref}}(t) = 2\frac{\pi}{180}\sin(2\pi f t), \tag{5.40}$$

which makes for a moderately varying reference that can be followed by both controllers.

In all experiments, the reference signal for attitude and speed is assumed constant throughout the entire prediction horizon of the MPC, which allows for a fair comparison to the baseline controller. The tracking performance of the controllers and their actuator usage are evaluated based on the metrics

$$S_e = \frac{1}{n}\sum_{i=1}^{n}|e_i|, \quad S_u = \frac{1}{n}\sum_{i=1}^{n}|u_i|, \quad S_f = \frac{2}{n_f f_s}\sum_{i=1}^{n_f} M_i f_i, \tag{5.41}$$

where n denotes the amount of samples in the data series. The sampling frequency is denoted by f_s and f_i, M_i denote the respective frequencies and magnitudes of the control signal. The metric S_f jointly considers the amplitudes of the evaluated frequencies, effectively measuring the smoothness of the control signals [136].

5.4.4 Results

We begin by presenting the experimental results based on a discussion of each test sequence before drawing conclusions and discussing common features. The experiments include several runs of each test sequence for both controllers. Each run is encoded in a different color in the plots. All tests were conducted during two days in autumn 2021 at a local airfield at Breivika, Norway. The wind conditions on the first day when we tested the controller's response to continuously changing references coming from a guidance controller were rather harsh, with an average wind speed of 12.5 m/s. This is a significant portion of the nominal cruise speed of 18.0 m/s, which makes control of the UAV increasingly challenging in comparison to passenger aircraft where the wind velocity can be safely neglected. The conditions on the second day when we tested individual step responses and oscillating references were more moderate with estimated wind speeds of approximately 6 m/s. Gust winds were significant on both days. Due to the size of the plots of the results, the related figures are given in Appendix B.

Reference from Guidance Module

In a first scenario, both controllers were given commands by a guidance module to follow a pattern that consists of two loiters and two rectangles. We conducted the tests at severe wind conditions with average wind speed estimates of approximately 12.5 m/s. The resulting trajectories are shown in Fig. B.1.

The reference trajectories in this case are slowly varying such that while both tracking results look similar, the PID achieves a slightly tighter attitude tracking compared to the MPC. Regarding airspeed on the other hand, the MPC is far outperforming the PID, suggesting a trade-off between attitude and airspeed tracking performance. It is notable that the MPC in general appears to have a more efficient actuator usage with S_u for the elevons at 60 % of the PID with less frequent set-point changes, resulting in S_f at 61 %. During the test, the airspeed command to the AP was 18 m/s and to the MPC it was 20 m/s. This was a request from our pilot who is responsible for a safe operation. The MPC is therefore running the UAV at a higher throttle. In this case where the controller is purely reacting to guidance set-points and operating well within the operational limits in the linear regime, the PID is sufficient.

Roll Steps

The sequence with only step changes in roll were run five times for each controller and are shown in Fig. B.2. The transient response in the roll angle are different for both controllers in the sense that the AP controller results in an approximately linear convergence, verified by roughly constant angular rates, towards the reference, whereas the MPC gives an approximately exponential convergence.

A closer look at usage of the control surfaces shows that the MPC achieves this by using high deflections of both elevons in the beginning but then quickly reduces the deflections as the attitude converges towards its reference. The AP controller on the other hand commands moderate elevon deflections that are approximately constant throughout the entire transient. Moreover, the AP controller shows oscillations in both roll and pitch angle. The MPC oscillates less and appears to actively use the pitch angle to react to step changes in the roll angle.

Moreover, the throttle is actively used at each step, which is most prominent at the change from -50 to 50 deg. In effect, the airspeed is kept within 1 m/s around the reference. The AP controller does not use this coupling and has a notable airspeed offset of approximately 1 m/s in all tests. The TECS controller would be handling the offset in the original control architecture of Ardupilot. However, in the experiments it is replaced by the PI airspeed controller as detailed in Appendix A.

Pitch Steps

The results of the pitch angle step sequence are shown in Fig. B.3. The transients for both controllers look similar in the beginning, but the AP controller induces pitch oscillations in a negative pitch change, whereas the MPC settles more quickly to the reference value. Moreover, the AP does not settle at -20 deg, but this may be handled by a different tuning, e.g. a higher integral gain.

The error in roll angle is similar for both controllers, but it is again notable how the oscillations in roll and pitch angle have an adverse effect on each other for the AP controller. The airspeed error looks similar for both controllers up to 5 s into the test, given that the only control strategy when following a negative pitch angle reference is to shut down the throttle. Note here that simultaneously commanding pitch angle references and airspeed references lead to conflicting objectives when the throttle is saturated. The tuning in Eq. (5.36a) - Eq. (5.36c) is such that the attitude tracking is prioritized as long as the airspeed is kept within the constraints and the respective slack variables are zero.

Shortly after the pitch angle reference changes from -20 to 20 deg, the MPC engages the throttle, even though the airspeed is already 5 m/s above its reference value. This behavior shows that MPC predicts a declining airspeed towards the end of the prediction horizon and by counteracting by using the throttle, the airspeed is kept closer to its reference compared to the AP controller. Not considering the airspeed limits, the AP controller comes dangerously close to the stall speed of 12 m/s.

Simultaneous Steps

The results for a simultaneous step change in both roll and pitch are shown Fig. B.4. Here, the MPC gives a response that is very close to that of the scenarios where each channel was excited separately. The AP controller does not show the oscillations at negative pitch step changes, but its overall pitch response is again worse compared to that of the MPC, with the AP unable to settle to either -20 or +20 deg. This time, the AP controller also violates the limits of the right elevon at the first step change which happens at 3 s into the test. Again, the MPC appears to actively use the throttle to counteract anticipated declines in the airspeed.

To study the quality of the model and its effect on the controller performance, we plot the trajectories of the norm of the difference of the linear acceleration $\|\Delta\dot{v}^b_{nb}\| = \|\dot{v}^b_{nb,z} - \dot{v}^b_{nb,y}\|$ and the difference of the angular accelerations $\Delta\dot{p} = \dot{p}_z - \dot{p}_y$, $\Delta\dot{p} = \dot{p}_z - \dot{p}_y$ in Fig. 5.9. There is a notable correlation between the errors in linear and pitch acceleration and the tracking performance concerning pitch angle and airspeed. It is not possible to distinguish between estimation errors of the wind velocity observer and inaccuracies of the aerodynamic model, given that equipment to measure the relative wind velocity vector was not part of the sensor suite during the experiments. However, the analysis of a dataset that includes flow measurements of a precision air data instrument suggests that the aerodynamic

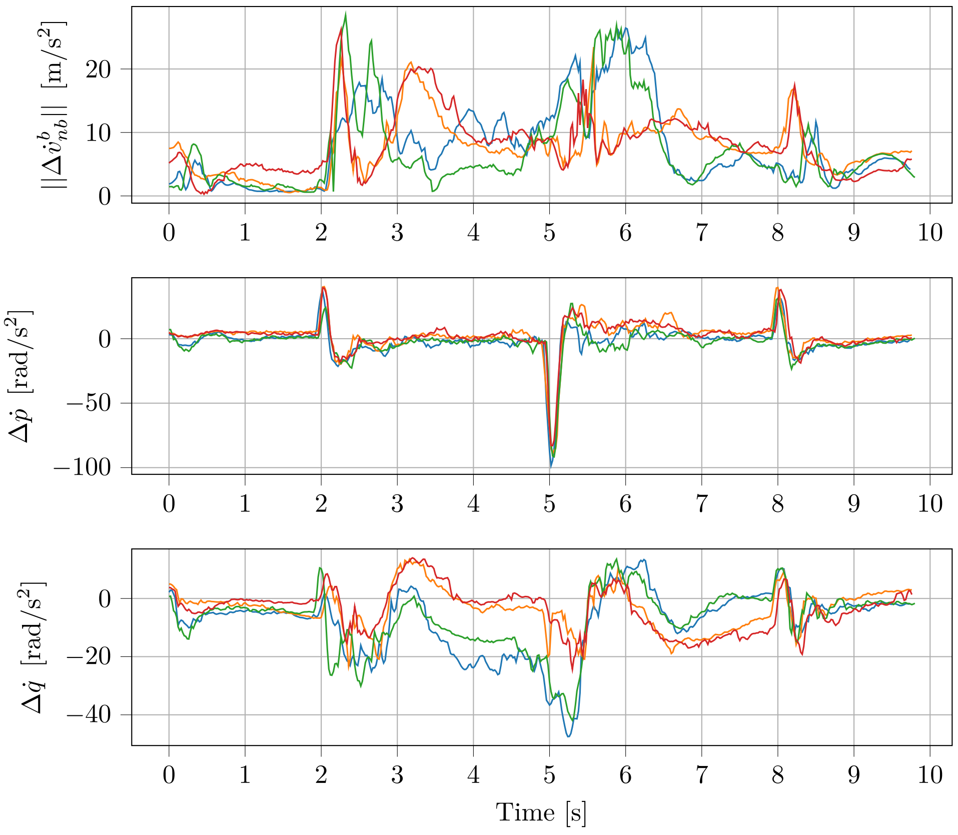

Figure 5.9: Difference between observed and modeled accelerations for the simultaneous step maneuvers.

model is the main cause for the difference between modeled and observed accelerations. In any case, the performance of the MPC despite the significant acceleration errors, in particular in the roll acceleration error with up to 100 rad/s^2, speaks to its robustness.

Oscillation References

The response for the scenario where the UAV was given oscillations references in roll and pitch are shown in Fig. B.5. This case demonstrates again the cross-coupling effects for the AP controller that do not appear when the MPC is in control. It is however notable that the MPC is not able to track both sinusoidal signals and the airspeed reference simultaneously. We have not checked if the given output trajectories consisting of roll, pitch and airspeed may be tracked at the same time and the obtained results are the trade-off implemented through the cost function in the given weather conditions. This explanation is most intuitive regarding the pitch angle and the airspeed. In the phase where the pitch reference is decreasing, the UAV follows, and the airspeed is necessarily increasing. With the exception for reference signals where thrust and drag forces can completely compensate for this effect, tracking both references at the same time becomes infeasible. Other reasons for the lack of performance in this case can be attributed to model mismatch, as discussed in the following section, and tuning. Both the control surfaces and the throttle could be tuned more aggressive for the MPC when compared to the AP controller.

Constant References and Disturbance Discussion

In the last scenario we look at the response to a constant reference command in roll and pitch that results in the UAV following a path that can be described as an upward spiral. This could be practical in a scenario where the UAV needs to quickly increase its height with a limited change in its horizontal position. The AP controller is again inducing oscillations until 5 s into the test, which are avoided by the MPC as it quickly converging to both references.

However, the performance of the MPC degrades notably, leading to oscillating errors in speed and pitch tracking, most visible at 10 s and 20 s into the test. The phenomenon is consistent over two separate runs, indicating that this is a systematic error which is likely due to errors in the dynamic model. The fact that the model is not accurately capturing the dynamics of the vehicle in this flight condition, is revealed by the correlating difference between observed and modeled accelerations depicted in Fig. 5.10. Ongoing work concerning the aerodynamic model suggests that our model has two shortcomings. The first is the decoupling assumption in the wind tunnel experiments that lead to purely lateral and longitudinal models, whereas this scenario leads to simultaneously significant angles of attack and sideslip.

The other flaw is that the damping model has been identified using an independent Vortex-Lattice method which leads to significantly different static aerodynamic coefficients. The selected damping coefficients are therefore not consistent with the static coefficients from the wind tunnel tests and leads to a poor damping model which we confirmed by inspection of the difference to a static aerodynamic model without damping.

Thus, handling a state that resembles a consistent simultaneous turn and climb, exposing the UAV to non-zero angular rates, is a weakness of the MPC, which is partially what we see in this test sequence. We conclude the discussion considering rejection of random disturbances such as sudden wind gusts or turbulences. At high roll angles, the wing surface area is more exposed to horizontal gust winds, which results in an increased sensitivity to disturbances. In this case, the MPC appears to reject wind gusts as good as the baseline controller, suggesting sufficient reactivity of the MPC.

Another concern is the quality of the wind estimate, which is likely to degrade at a large angle of attack and sideslip angle. Rapid oscillation roll and pitch movements such as in the previous test will therefore introduce disturbances. These can either be compensated for by the disturbance observer employed with the controller or avoided by using sensor equipment that directly measures the relative velocity vector. The wind estimate only enters the model-free baseline controller through

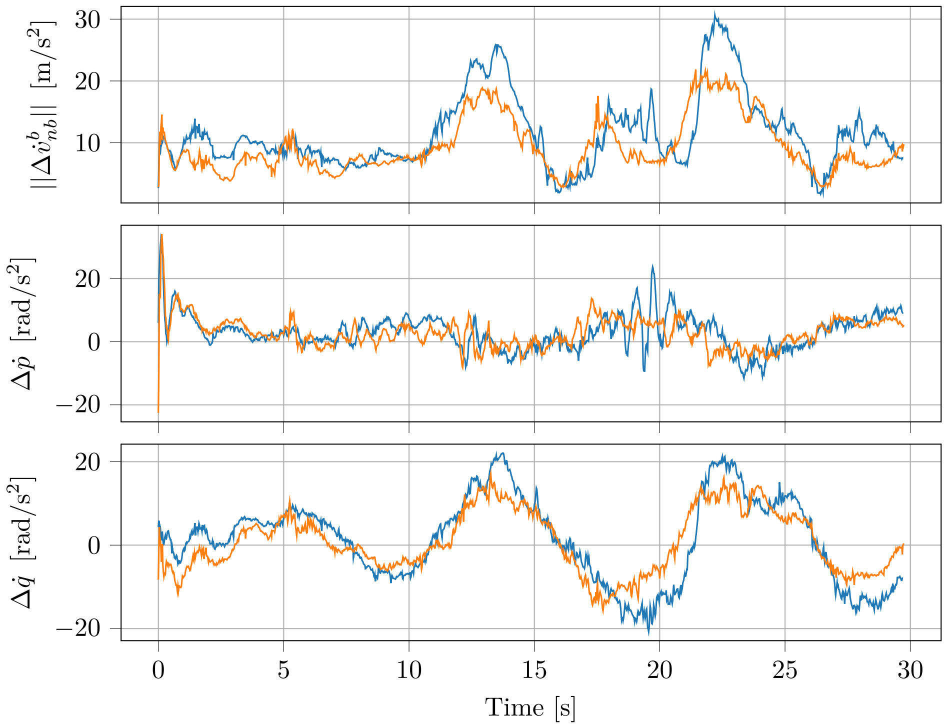

Figure 5.10: Difference between observed and modeled accelerations for the 30 s constant reference.

airspeed scaling in the control law. We can expect that degrading wind estimates will be less problematic here.

We have seen in the experiments that the MPC is better equipped to keep the airspeed error small, and it further includes the angle of attack limits in the constraint function. The wind disturbances become more significant at low-speed flights where the UAV is also at increased risk of stalling. Assuming the model in this flight condition is sufficiently accurate and that disturbances are well-estimated, the MPC provides safety benefits that the model-free baseline controller can not provide.

5.5 Benchmark Scenario

For the benchmark scenario simulation in this chapter, the controller structure is as presented in the preceding sections. The cost matrices are again tuned for similar performance to the preceding benchmarked controllers in the step sequence shown in Fig. A.1, which results in the cost matrices

$$\mathbf{Q} = \operatorname{diag}(q_{V_a},\ q_{\Gamma,x},\ q_{\Gamma,y},\ q_{\Gamma,z}) = \operatorname{diag}(0.01,\ 30,\ 30,\ 30) \tag{5.42a}$$

$$\mathbf{R} = \operatorname{diag}(r_{\dot{\delta}_a},\ r_{\dot{\delta}_e},\ r_{\dot{\delta}_t}) = \operatorname{diag}(10^{-1},\ 10^{-1},\ 10^{-1}). \tag{5.42b}$$

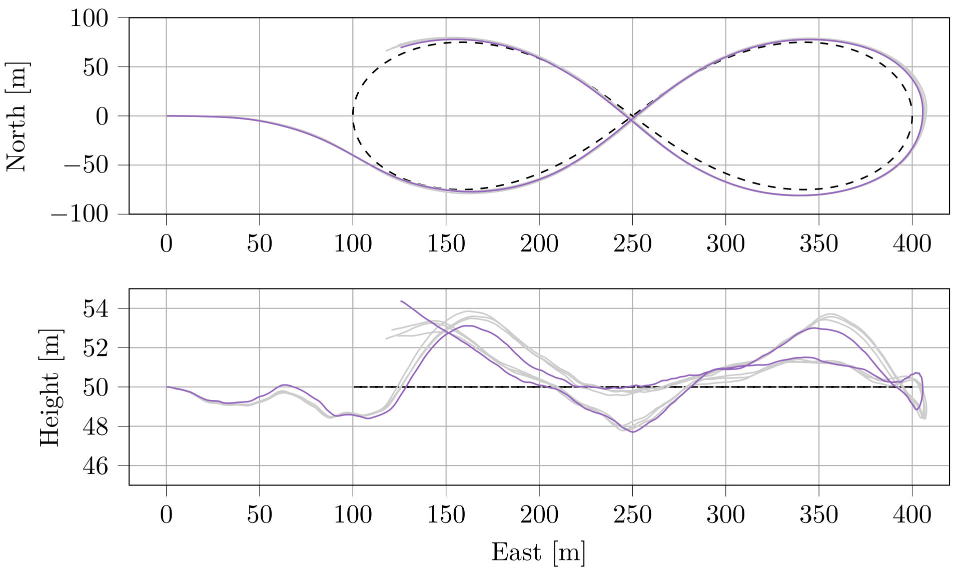

Figure 5.11: Benchmark simulation comparing the low-level MPC (LLMPC, plotted in purple) introduced in this chapter to the previous controller designs (gray). The reference path is plotted in black, dashed.

The prediction horizon is set to $N = 30$ and the length of the shooting intervals is set to $\Delta t = 0.1\,\mathrm{s}$.

The results of the simulation in comparison to the previously introduced controllers are plotted for the position in Fig. 5.11 and for the distance to the path, error signals and actuator signals in Fig. 5.12. Again, the scores for the performance metrics and colored plots for all controllers are given in Appendix A. We refer to the controller presented in this chapter as low-level MPC (LLMPC). From Table A.2 it can be seen that the LLMPC has the best roll-tracking performance and almost equal airspeed-tracking performance to the MPCGC. Again, the inferior performance in pitch-tracking when compared to the reactive controllers (PID, AP, GC), is likely a result of the trade-off with the airspeed-tracking performance.

It is moreover clear that the LLMPC commands the fastest changes in the control surface set-points as indicated by the metric J_f. This is due to the design decision to prioritize tracking performance over actuator usage and it is straightforward to improve the score in J_f by modifying the elements of $\mathbf{R}$.

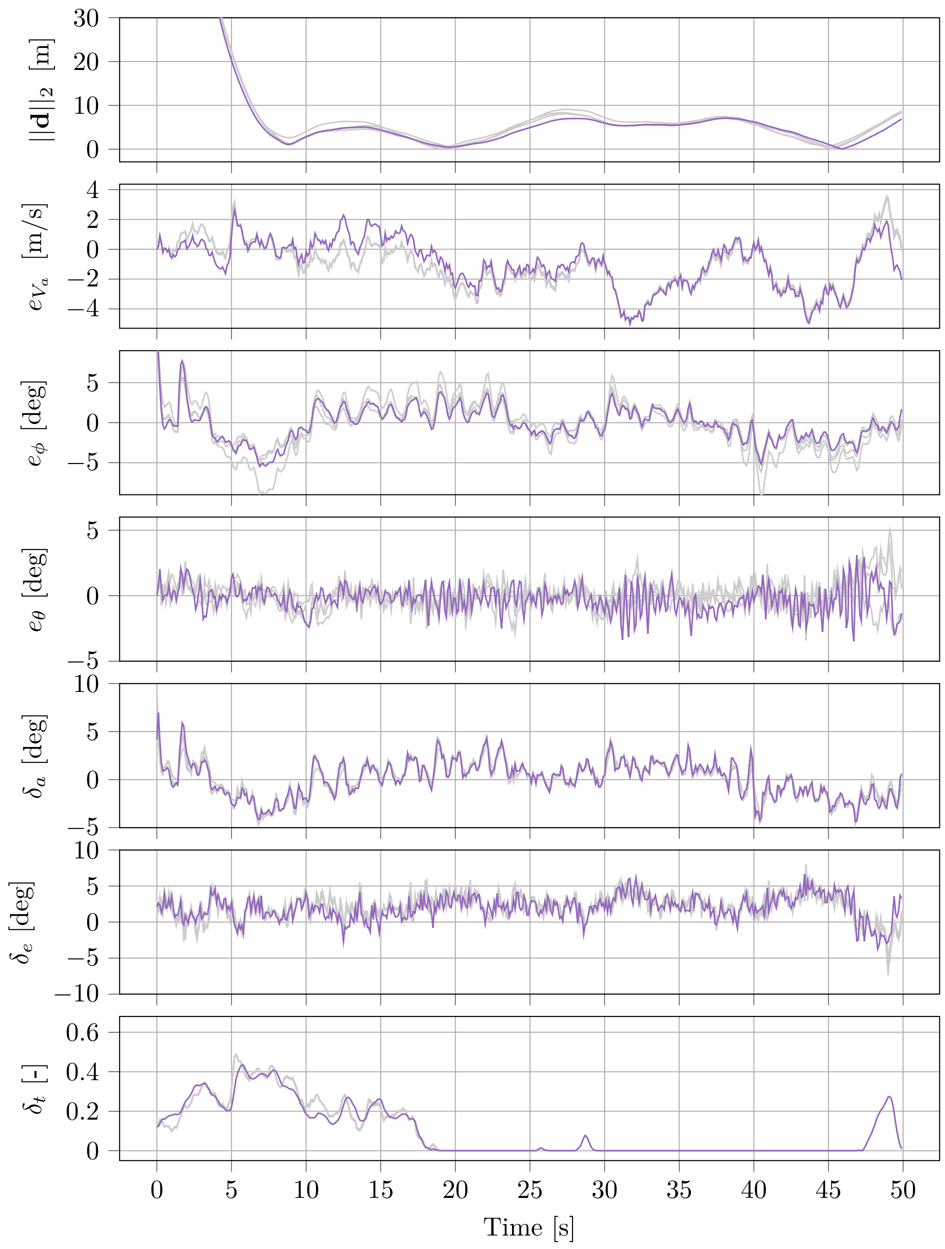

Figure 5.12: Benchmark simulation comparing the low-level MPC (LLMPC, plotted in purple) introduced in this chapter to the previous controller designs (gray). The first subplot includes the distance to the reference path $\|\mathbf{d}\|_2$, and the following subplots include the error signals for airspeed, roll and pitch, as well as the actuator signals.

5.6 Chapter Summary

In this chapter we looked at NMPC design for the control problem of tracking attitude and speed references, using the previously introduced notion of reduced-attitude representations to use dynamics in the controller where the attitude evolves on the Special Orthogonal Group of order Three. The actuators of the UAV are directly accessed by the NMPC. A feature that is distinct to most publications in the literature that often use control-augmented approaches to solve a higher-level guidance problem.

The introduced controllers can be used to track reduced-attitude projections in either roll-pitch or pitch-yaw direction, and we discussed possible extensions to include objectives such as turn-rate control. The focus was on the control of the roll, pitch and airspeed, given that this formulation can readily be integrated into existing GNC architectures to enable experimental testing. Tuning guidelines were given and possible values for the closed-loop solver runtime on a suitable SBC were assessed through a Monte-Carlo simulation study.

We then tested the controller in experiments in different scenarios and compared its performance to the attitude controller implementation of ArduPilot, which showed favorable results for the presented NMPC. The performance of the NMPC in the experiments despite significant inaccuracies of the dynamic model speaks to the robustness of the controller.

We have seen in the experiments that under nominal flight conditions, the NMPC and the baseline AP have comparable performance when following attitude references from a guidance controller, and that the NMPC has in almost all test cases at least equal or superior performance to the AP controller. The guidance example has shown the ability of the NMPC to keep the UAV significantly closer to the airspeed reference when following standard flight patterns such as loitering maneuvers or rectangular paths. Together with the coupling between pitch and airspeed, this can be exploited in scenarios where the UAV is to be operated at low airspeed. In addition to the ability of the NMPC to keep the UAV close to the airspeed reference, it is explicitly considering the increased angle of attack during this flight condition and through its constraint definition readily includes a mechanism to avoid stalling.

Chapter 6

Coupled Nonlinear Model Predictive Control and Geometric Attitude Control

This chapter is based on

[157] Dirk Reinhardt and Tor Arne Johansen. Nonlinear Model Predictive Control combined with Geometric Attitude and Speed Control for Fixed-Wing UAVs. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 465–475, 2021.

6.1 Introduction

6.1.1 Motivation and Background

In this chapter, we discuss a combination of NMPC and Geometric Attitude Control for the attitude and speed control problem of fixed-wing UAVs. The problem that we aim at with this controller design is to give a viable alternative to local controller designs based on PID loops as is common in modern low-level autopilots. The design based on a perturbation model around a local trim state is effective in nominal flight conditions, with degrading performance at more distant regions of the state space. An example for this is where the aircraft is in agile flight or has to be recovered from severe wind disturbances [86], motivating the use of globally stabilizing controllers. Toward this, a geometric controller in its basic form has been proposed in [37] showing almost semi-global exponential stability of the closed-loop dynamic system. A natural consequence of the controller design on the two-sphere $\mathbb{S}^2$ is the existence of an additional unstable equilibrium, due to the fact that it is a compact manifold. This precludes the global attitude stabilization via continuous feedback [11] or discontinuous feedback as shown by Mayhew et al. [125]. Using hybrid control theory as a remedy to this problem for the spherical orientation is thoroughly discussed in [126] and the theory has recently been developed to the case of general smooth manifolds by Casau et al. [24]. Drawing from these ideas we designed a hybrid controller with an additional expelling potential as discussed in [153] where we showed global exponential stability.

The motivation of this work is to use NMPC in a cascade with the geometric controller [37] instead of the hybrid control scheme [153]. This way we solve the problem of performance loss and discontinuous actuator commands that may occur at instances where the potential function is switched, as an alternative to a hybrid controller with an additional dynamic of the logic state [25]. Moreover, except for constructed examples it is not a trivial task to design suitable angular acceleration references for the geometric controller, such that it solely relies on attitude or possibly rate set-points. This can be readily dealt with by a suitable NMPC design, together with a relaxation of decoupling assumption of throttle input and slowly time-varying angle-of-attack and airspeed. Finally, constraint satisfaction and optimal performance is readily achieved and the NMPC may be allowed to run at slower update rates by exploiting the disturbance rejection capabilities of the more reactive geometric controller.

6.1.2 Related work

Up until recently, employing NMPC has been prohibitive in robotic systems with fast dynamics, but due to advancements on both algorithmic and hardware level, finds more applications also in UAVs. Trajectory tracking for fixed-wing aircraft is considered in [66] where it is demonstrated that efficient algorithms allow for NMPC that directly actuate control surfaces and throttle. However, most applications focus on the guidance level and rely on off-the-shelf autopilots to handle low-level dynamics. For example, the approach [174] in avoids the use of a complex aerodynamic model and instead finds a model of the response to attitude/speed set-points with the autopilot in closed-loop. The resulting model is then used in a path-following NMPC. In [4], path-following MPCs for constrained under-actuated vehicles is proposed based on kinematic guidance models, where the resulting angular rate and forward velocity set-points may be tracked by a sufficiently fast low-level controller [3]. See also [194], where atmospheric disturbances are considered. A more recent approach where the complete model including actuators is used in a trajectory optimization problem for motion planning, with low-level LQG feedback between updates is [8], resulting in extended maneuverability at high angles of attack. For attitude control with direct actuation of the control surfaces, the work in [143] employs small perturbations to apply linear models that are decoupled in lateral/longitudinal direction from which explicit MPCs can be derived. The low computational demands allow for implementation on a standard avionic system with limited resources. For high performance flight in a wide envelope however, we believe that it is necessary to consider coupled dynamics in the nonlinear regime.

In fact, there are several examples for multi-rotors which support this. For example in [139] designs a NMPC scheme with an iterative LQG for direct control of the actuators on the prospect of significant performance improvement. [5] assumes sufficiently fast low-level controller and sets references for roll/pitch angles. In later work, some authors propose NMPC in [91] to replace the low-level autopilot, potentially in a more extensive control architecture [93]. Similarly, in [197], with a stronger focus on the efficient numerical solvers. And recently, [92] showed for path-following on a multi-rotor that considering the full system dynamics in NMPC can improve disturbance rejection, tracking performance and computational effort compared to linear MPC.

6.2 Geometric Controller Design

We first recapitulate the necessary ingredients to implement the geometric controller. Then explain the model predictive controller and outline the control architecture. A simulation study of the cascaded controller concludes this chapter.

For the geometric attitude controller, we again assume the standard kinematic equation for the rotation matrix and the moment equation as in Section 4.2.2, but a more careful notation with respect to the coordinates is needed. Recall the dynamics as

$$\dot{\mathbf{R}}_{nb} = \mathbf{R}_{nb}\mathbf{S}(\boldsymbol{\omega}^b_{nb}) \tag{6.1}$$

$$\mathbf{J}\dot{\boldsymbol{\omega}}^b_{nb} = \mathbf{f}(\boldsymbol{\omega}^b_{nb}, \mathbf{v}_r) + \mathbf{G}(\mathbf{v}_r)\begin{bmatrix}\delta_a & \delta_e & \delta_r\end{bmatrix}^\top \tag{6.2}$$

with drift term $\mathbf{f}(\mathbf{v}^b_r, \boldsymbol{\omega}^b_{nb})$, control effectiveness matrix $\mathbf{G}(\mathbf{v}^b_r, \boldsymbol{\omega}^b_{nb})$ and control input vector $\mathbf{u} = \begin{bmatrix}\delta_a & \delta_e & \delta_r\end{bmatrix}^\top$. The inertia matrix is denoted by $\mathbf{J} \in \mathbb{R}^{3\times 3}$.

6.2.1 Reduced Attitude

An orthogonal decomposition of the angular velocity vector $\boldsymbol{\omega}^b_{nb} = \boldsymbol{\omega}^{b\perp}_{nb} + \boldsymbol{\omega}^{b\|}_{nb}$ with respect to $\boldsymbol{\Gamma}$ can be obtained using Eq. (2.17) as

$$\boldsymbol{\omega}^{b\perp}_{nb} = \boldsymbol{\Pi}^{\perp}_{\Gamma}\boldsymbol{\omega}^b_{nb} \in \mathrm{T}_{\Gamma}\mathbb{S}^2, \;\; \boldsymbol{\omega}^{b\|}_{nb} = \boldsymbol{\Pi}^{\|}_{\Gamma}\boldsymbol{\omega}^b_{nb} \in \mathrm{N}_{\Gamma}\mathbb{S}^2. \tag{6.3}$$

The kinematic equation for $\boldsymbol{\Gamma}$ follows from Eq. (6.1) and Eq. (4.3) as

$$\dot{\boldsymbol{\Gamma}} = \boldsymbol{\Gamma} \times \boldsymbol{\omega}^b_{nb} = \boldsymbol{\Gamma} \times \boldsymbol{\omega}^{b\perp}_{nb}, \tag{6.4}$$

where the second equality results from $\boldsymbol{\omega}^b_{nb} = \boldsymbol{\omega}^{b\perp}_{nb} + \boldsymbol{\omega}^{b\|}_{nb}$ and $\boldsymbol{\Gamma} \times \boldsymbol{\omega}^{b\|}_{nb} = \mathbf{0}$. The derivative of Eq. (6.3) is given by

$$\dot{\boldsymbol{\omega}}^{b\perp}_{nb} = \boldsymbol{\Pi}^{\perp}_{\Gamma}\dot{\boldsymbol{\omega}}^b_{nb} + \boldsymbol{\omega}^{b\perp}_{nb} \times \boldsymbol{\omega}^{b\|}_{nb} \tag{6.5}$$

$$\dot{\boldsymbol{\omega}}^{b\|}_{nb} = \boldsymbol{\Pi}^{\|}_{\Gamma}\dot{\boldsymbol{\omega}}^b_{nb} - \boldsymbol{\omega}^{b\perp}_{nb} \times \boldsymbol{\omega}^{b\|}_{nb}. \tag{6.6}$$

6.2.2 Reference System

Consider a time-varying reference trajectory for the reduced attitude $\boldsymbol{\Gamma}_d \in \mathbb{S}^2$, satisfying

$$\dot{\boldsymbol{\Gamma}}_d = \boldsymbol{\Gamma}_d \times \boldsymbol{\omega}^b_{nb,d} \tag{6.7}$$

for some time-varying desired angular velocity $\boldsymbol{\omega}^b_{nb,d} \in \mathrm{T}_{\boldsymbol{\Gamma}_d}\mathbb{S}^2$. The objective of the higher-level NMPC will be to drive the reference through its acceleration $\dot{\boldsymbol{\omega}}^b_{nb,d} \in \mathrm{N}_{\Gamma}\mathbb{S}^2$, effectively running the exogenous reference system to which we imposed a boundedness assumption in Section 4.2.4.

6.2.3 Error States

Recall the potential function used in the geometric attitude controller Eq. (4.16), given as

$$\Psi(\mathbf{\Gamma}, \mathbf{\Gamma}_d) = \frac{1}{2}\|\mathbf{\Gamma} - \mathbf{\Gamma}_d\|^2 = 1 - \mathbf{\Gamma} \cdot \mathbf{\Gamma}_d \tag{6.8}$$

which leads to the definition of the error vector

$$\mathbf{e}_\Gamma = -\mathbf{\Gamma} \times \nabla_\Gamma \Psi(\mathbf{\Gamma}, \mathbf{\Gamma}_d) = \mathbf{\Gamma}_d \times \mathbf{\Gamma} \in \mathrm{T}_\Gamma \mathbb{S}^2. \tag{6.9}$$

Let the angular velocity error $\mathbf{e}_\omega \in \mathrm{T}_\Gamma \mathbb{S}^2$ be defined as

$$\mathbf{e}_\omega = \mathbf{\Pi}_\Gamma^\perp (\boldsymbol{\omega}_{nb}^b - \boldsymbol{\omega}_{nb,d}^b). \tag{6.10}$$

The potential function and the error states are precisely as introduced in the preceding chapter and more details can be found in Section 4.2.5.

6.2.4 Control Law

In its original form as discussed in Chapter 4, we used dynamic inversion to formulate the control law for geometric attitude control as a combination of feed-forward terms and PD-like feedback in terms of $\mathbf{e}_\Gamma$ and $\mathbf{e}_\omega$. However, the dynamic inversion effectively eliminates all - and thus also the potentially useful - nonlinearities that are present in the dynamic equations. This was necessary to prove the almost semi-global exponential stability property of a Lyapunov-based controller design. Here we take a different approach and leave it up to the NMPC to deal with the modelled nonlinearities. Let the state-feedback control law $\mathbf{u}_{\mathrm{GC}}$ for the geometric controller be

$$\begin{bmatrix}\delta_a & \delta_e & \delta_r\end{bmatrix}^\top = \mathbf{u}_{\mathrm{GC}}(\mathbf{x}, \dot{\boldsymbol{\omega}}_{nb,d}^b) = \mathbf{G}^{-1}(\boldsymbol{\omega}_{nb}^b, \mathbf{v}_r^b)\mathbf{J}\boldsymbol{\kappa}(\mathbf{x}, t) \tag{6.11}$$

with

$$\boldsymbol{\kappa}(\mathbf{x}, t) = -\boldsymbol{\omega}_{nb}^{b\perp} \times (\boldsymbol{\omega}_{nb}^{b\|} - \mathbf{\Pi}_\Gamma^\| \boldsymbol{\omega}_{nb,d}^b) + \mathbf{\Pi}_\Gamma^\| \dot{\boldsymbol{\omega}}_{nb,d}^b - k_p \mathbf{e}_\Gamma - \mathbf{\Pi}_\Gamma^\perp \mathbf{K}_d \mathbf{e}_\omega. \tag{6.12}$$

The state vector $\mathbf{x}$ will be defined in the next section.

6.3 Nonlinear Model Predictive Controller Design

6.3.1 State and Dynamic Model

To have globally unique attitude representations without singularities, we use the rotation matrix for the attitude representation, instead of the common minimum parameterizations such as Euler angles or quaternions. Quaternions evolve on $\mathbb{S}^3$, which is a double cover of the natural configuration space SO(3), meaning that two quaternions can be used to represent the same attitude. Euler angles are prone to singular attitude representations (gimbal lock), which would need to be handled in the implementation. Chaturvedi et al. [29] offer an excellent discussion on this topic. Another benefit is that the attitude kinematics of the rotation matrix only include bilinear expressions instead of trigonometric terms.

Let a decomposition of the rotation matrix into the individual axes be $\mathbf{R}_{nb} = [\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z]$ with $\mathbf{r}_{\{x,y,z\}} \in \mathbb{S}^2$ representing the axes of the body-fixed frame expressed in the coordinates the inertial frame. Then define the state vector $x \in \mathbb{R}^{n_x}$ and input vector $\mathbf{u} \in \mathbb{R}^{n_u}$ as

$$\mathbf{x} = \left[V_a \; \beta \; \alpha \; \mathbf{r}_x{}^\top \; \mathbf{r}_y{}^\top \; \mathbf{r}_z{}^\top \; (\boldsymbol{\omega}^s_{nb})^\top \; \boldsymbol{\Gamma}_d{}^\top \; \boldsymbol{\omega}^b_{nb,d}{}^\top \; \delta_t\right]^\top, \tag{6.13}$$

$$\mathbf{u} = \left[\dot{\boldsymbol{\omega}}^b_{nb,d} \quad \dot{\delta}_t\right]^\top \tag{6.14}$$

with $n_x = 22$ and $n_u = 4$. With $\boldsymbol{\omega}^s_{nb} \in \mathbb{R}^3$ denoting the angular velocity vector decomposed in the axes components of $\{s\}$. The dynamic equations for the translational motion are given by

$$\begin{bmatrix} \dot{V}_a \\ \dot{\beta} V_a \\ \dot{\alpha} V_a \cos\beta \end{bmatrix} = \frac{1}{\mathrm{m}} \mathbf{f}^w(\mathbf{x}, \mathbf{u}_{\mathrm{GC}}(\mathbf{x}, \dot{\boldsymbol{\omega}}^b_{nb,d})) + \mathbf{R}_{wb} \mathbf{R}_{nb}{}^\top \mathbf{g}^n - \boldsymbol{\omega}^w_{nb} \times \mathbf{v}^w_r + \mathbf{d}_f \tag{6.15}$$

where $\mathbf{f}^w(\mathbf{x}, \mathbf{u}_{\mathrm{GC}}(\mathbf{x}, \dot{\boldsymbol{\omega}}^b_{nb,d})) = \mathbf{f}^w_a(\mathbf{x}, \mathbf{u}_{\mathrm{GC}}(\mathbf{x}, \dot{\boldsymbol{\omega}}^b_{nb,d})) + \mathbf{R}_{wb} \mathbf{f}^b_t(V_a, \delta_t)$ is the force vector resulting from aerodynamics $\mathbf{f}^w_a$ and thrust $\mathbf{f}^b_t$. The mass of the UAV is given by $\mathrm{m} \in \mathbb{R}$ and $\mathbf{g}^n$ denotes the gravity vector in $\{n\}$. For the rotational motion we have

$$\dot{\mathbf{R}}_{nb} = \begin{bmatrix} \dot{\mathbf{r}}_x & \dot{\mathbf{r}}_y & \dot{\mathbf{r}}_z \end{bmatrix} = \mathbf{R}_{nb} \mathbf{S}(\mathbf{R}^\top_{sb} \boldsymbol{\omega}^s_{nb}) \tag{6.16}$$

$$\mathbf{J}^s \dot{\boldsymbol{\omega}}^s_{nb} = \mathbf{S}(\mathbf{J}^s \boldsymbol{\omega}^s_{nb}) \boldsymbol{\omega}^s_{nb} + \mathbf{R}_{sb} \mathbf{m}^b(\mathbf{x}, \mathbf{u}_{\mathrm{GC}}(\mathbf{x}, \dot{\boldsymbol{\omega}}^b_{nb,d})) - \mathbf{J}^s(\boldsymbol{\omega}^s_{bs} \times \boldsymbol{\omega}^s_{nb}) + \mathbf{J}^s \mathbf{d}_m, \tag{6.17}$$

The difference to the nominal dynamic equation assumed by the geometric controller is the added moment disturbance term $\mathbf{J}^s \mathbf{d}_m$ in Eq. (6.17) which is not present in Eq. (6.2). Another disturbance is acting as a force disturbance $\mathbf{d}_f$ in Eq. (6.15). We consider both disturbances as slowly varying and provide details on how they are updated in Section 6.3.5. For now, let the disturbances be concatenated by

$$\mathbf{d} = \begin{bmatrix} \mathbf{d}_f{}^\top & \mathbf{d}_m{}^\top \end{bmatrix}^\top. \tag{6.18}$$

The remaining equations for the evolution of the NMPC state $\mathbf{x}$ are given by Eq. (6.7) for $\dot{\boldsymbol{\Gamma}}_d$. The rest is directly controlled through the input vector $\mathbf{u}$ as defined in Eq. (6.14). The control algorithm design, in particular the interconnection between the geometric controller and the NMPC is illustrated Fig. 6.1.

Throughout the rest of this chapter, we will use the continuous vector ODE $\dot{\mathbf{x}} = \mathbf{f}_c(\mathbf{x}, \mathbf{u}, \mathbf{d})$ to denote the state dynamics described in Eq. (6.7) and Eq. (6.15) - Eq. (6.17). Where applicable, the discretized version based on an explicit Runge-Kutta method of order four will be used, with $\mathbf{x}(k+1) = \mathbf{f}_{\mathrm{RK4}}(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k))$ denoting the resulting difference equation.

6.3.2 Constraints

The constraint set includes box constraints and mixed nonlinear constraints. To ensure safe flight conditions within the admissible envelope, the aerodynamic angles

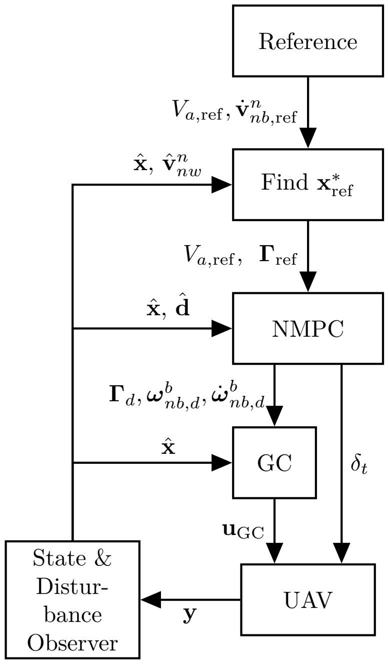
Reference
$V_{a,\mathrm{ref}}, \dot{\mathbf{v}}^n_{nb,\mathrm{ref}}$
$\hat{\mathbf{x}}, \hat{\mathbf{v}}^n_{nw}$
Find $\mathbf{x}^*_{\mathrm{ref}}$
$V_{a,\mathrm{ref}}, \mathbf{\Gamma}_{\mathrm{ref}}$
$\hat{\mathbf{x}}, \hat{\mathbf{d}}$
NMPC
$\mathbf{\Gamma}_d, \boldsymbol{\omega}^b_{nb,d}, \dot{\boldsymbol{\omega}}^b_{nb,d}$
$\hat{\mathbf{x}}$
GC
δ_t
$\mathbf{u}_{\mathrm{GC}}$
State & Distur-bance Observer
$\mathbf{y}$
UAV

Figure 6.1: The control algorithm design. Reference signals for airspeed $V_{a,\mathrm{ref}}$ and linear acceleration $\dot{\mathbf{v}}^n_{nb,\mathrm{ref}}(t)$ are used to find suitable attitude and airspeed references for the NMPC through $\mathbf{x}^*_{\mathrm{ref}}$ by solving Eq. (6.23). Control surface deflections are set by the GC, based on the control-law $\mathbf{u}_{\mathrm{GC}}$ and the reference signals $(\mathbf{\Gamma}_d, \boldsymbol{\omega}^b_{nb,d}, \dot{\boldsymbol{\omega}}^b_{nb,d})$.

and airspeed are subject to the relaxed box constraints with the possibility to tighten the constraints. The related constraint inequalities are given by

$$V_a - (1+\epsilon_{V_a})\underline{V}_a + \underline{s}_{V_a} \geq 0 \tag{6.19a}$$
$$-V_a + (1-\epsilon_{V_a})\overline{V}_a + \overline{s}_{V_a} \geq 0 \tag{6.19b}$$
$$\alpha - (1-\epsilon_\alpha)\underline{\alpha} + \underline{s}_\alpha \geq 0 \tag{6.19c}$$
$$-\alpha + (1-\epsilon_\alpha)\overline{\alpha} + \overline{s}_\alpha \geq 0 \tag{6.19d}$$
$$\beta - (1-\epsilon_\beta)\underline{\beta} + \underline{s}_\beta \geq 0 \tag{6.19e}$$
$$-\beta + (1-\epsilon_\beta)\overline{\beta} + \overline{s}_\beta \geq 0. \tag{6.19f}$$

The tightening parameters ϵ_* allow for further tightening the original constraints. This can be useful to react to turbulent weather conditions and uncertainty of the dynamic model. The slack variables denoted by $s_* \in \mathbb{R}_{\geq 0}$ in the constraints are included to guarantee the feasibility of the underlying QPs.

Limits of the control surface deflections are included as mixed nonlinear constraints, given that they are not being set directly, but are the output of the geometric controller function $\mathbf{u}_{\mathrm{GC}}$. The throttle set-point δ_t is subject to a box constraint. The constraints related to the actuator limits are thus given by

$$\mathbf{u}_{\mathrm{GC}}(\mathbf{x}, \dot{\boldsymbol{\omega}}^b_{nb,d}) - \begin{bmatrix}\underline{\delta}_a & \underline{\delta}_e & \underline{\delta}_r\end{bmatrix} + \underline{\mathbf{s}}_{\mathbf{u}_{\mathrm{GC}}} \geq \mathbf{0} \tag{6.20a}$$

$$-\mathbf{u}_{\mathrm{GC}}(\mathbf{x}, \dot{\boldsymbol{\omega}}^b_{nb,d}) + \begin{bmatrix}\overline{\delta}_a & \overline{\delta}_e & \overline{\delta}_r\end{bmatrix} + \overline{\mathbf{s}}_{\mathbf{u}_{\mathrm{GC}}} \geq \mathbf{0} \tag{6.20b}$$

$$\delta_t - \underline{\delta}_t \geq 0 \tag{6.20c}$$

$$-\delta_t + \overline{\delta}_t \geq 0. \tag{6.20d}$$

Finally, constraints for the control input of the NMPC and constraints that ensure positive slack variables are included through the inequalities

$$\mathbf{u} - \underline{\mathbf{u}} \geq \mathbf{0}, \quad -\mathbf{u} + \overline{\mathbf{u}} \geq \mathbf{0}, \quad \mathbf{s} \geq \mathbf{0}, \tag{6.21}$$

where the vector $\mathbf{s} \in \mathbb{R}^{12}$ is defined as the concatenation of all slack variables in the order they appear in Eq. (6.19a) - Eq. (6.20b). All vector inequalities are interpreted as element-wise. For the remainder of this chapter, let the inequalities Eq. (6.19a) - Eq. (6.21) be concatenated in

$$\mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{s}) \geq \mathbf{0}. \tag{6.22}$$

6.3.3 Reference Generation for the Nonlinear Model Predictive Controller

As a reference signal, we consider linear acceleration commands in the inertial frame, denoted by $\dot{\mathbf{v}}^n_{nb,\mathrm{ref}}$, which can be generated by employing a nonlinear path-following controller, e.g. the guidance law proposed in [32]. The acceleration command then enters an optimization problem to find a reference state $\mathbf{x}_{\mathrm{ref}}$ that is close to the current state $\mathbf{x}$ with airspeed close to cruise speed $V_{a,\mathrm{ref}}$. The optimization problem is there formulated as

$$\min_{\mathbf{x}_{\mathrm{ref}}} \left(\|\dot{\mathbf{v}}^n_{nb,\mathrm{ref}}(t) - \dot{\mathbf{v}}^n_{nb}\|^2_{\mathbf{Q}_{\dot{v}}} + q_{Va}(V_{a,\mathrm{ref}}(t) - V_a)^2 + \|\mathbf{x}(t) - \mathbf{x}_{\mathrm{ref}}\|^2_{\mathbf{Q}_x}\right) \tag{6.23a}$$

$$\text{s. t.} \quad \dot{\mathbf{v}}^n_{nb} = \frac{1}{m}\mathbf{R}_{nb}{\mathbf{R}_{wb}}^\top \mathbf{f}^w(\mathbf{x}_{\mathrm{ref}}, \boldsymbol{\delta}) + \mathbf{g}^n \tag{6.23b}$$

$$V_a = {\mathbf{e}_1}^\top \mathbf{x}_{\mathrm{ref}} \tag{6.23c}$$

$$\mathbf{h}(\mathbf{x}_{\mathrm{ref}}, \mathbf{0}, \mathbf{0}) \geq \mathbf{0}. \tag{6.23d}$$

The first two terms in Eq. (6.23a) constitute the deviation from the reference signals which we seek to minimize whereas the last term is a regularization term to keep the solution close to the given estimated state of the UAV. The dynamic model enters through the equality constraint Eq. (6.23b). The airspeed is extracted from the state vector through Eq. (6.23c), and we impose the previously defined constraints on the reference through Eq. (6.23d).

The positive definite weightings q_{V_a}, $\mathbf{Q}_{\dot{v}}$ and $\mathbf{Q}_x$ are included to allow for a weighting between the reference signals and the regularization term. Defining $\mathbf{x}^*_{\mathrm{ref}}$ as the argument that minimizes Eq. (6.23), the output reference signal for the NMPC is given by

$$\mathbf{r} \triangleq \begin{bmatrix}V_{a,\mathrm{ref}} & {\boldsymbol{\Gamma}_{\mathrm{ref}}}^\top\end{bmatrix}^\top = \begin{bmatrix}\mathbf{e}^{n_x}_1 & \mathbf{e}^{n_x}_6 & \mathbf{e}^{n_x}_9 & \mathbf{e}^{n_x}_{12}\end{bmatrix}^\top \mathbf{x}^*_{\mathrm{ref}}. \tag{6.24}$$

Note that we distinguish between $\mathbf{\Gamma}_{\text{ref}}$ and $\mathbf{\Gamma}_d$. The reference obtained through via the optimization problem is $\mathbf{\Gamma}_{\text{ref}}$ and is included in the output reference to the NMPC, and $\mathbf{\Gamma}_d$ is part of the state vector $\mathbf{x}$ as defined in Eq. (6.13).

Remark 6.1. Let the set $\mathcal{X}_{\text{target}}$ be defined by all states $\mathbf{x}$ in the pre-image of Eq. (6.24). We see that for all states in $\mathcal{X}_{\text{target}}$, for either the continuous or the discrete mapping, we have $\mathbf{f} : \mathcal{X}_{\text{target}} \to \mathcal{X}_{\text{target}}$, i.e. $\mathcal{X}_{\text{target}}$ is invariant under the state dynamics at $\mathbf{r}$, which is a necessary requirement for a well-posed output in the NMPC.

6.3.4 Cost and Nonlinear Program

To formulate the NMPC for output stabilization, let the stabilizing quadratic stage cost be

$$l(\mathbf{x}, \mathbf{u}, \mathbf{r}) = q_{V_a}(V_a - V_{a,\text{ref}})^2 + \|\mathbf{\Gamma} - \mathbf{\Gamma}_{\text{ref}}\|^2_{\mathbf{Q}_\Gamma} + \|\mathbf{u}\|^2_{\mathbf{R}}, \tag{6.25}$$

which includes $q_{V_a} > 0$ and the positive definite and symmetric weighting matrices $\mathbf{Q}_\Gamma \in \mathbb{R}^{3\times3}$ and $\mathbf{R} \in \mathbb{R}^{n_u\times n_u}$.

We use direct multiple shooting to discretize the OCP into N control intervals and integrate the continuous dynamics for each interval with an explicit Runge Kutta scheme of order four. Let a predicted state and input sequence be denoted by $\mathbf{x}(\cdot|t) \in \mathbb{R}^{n_x\times(N+1)}$, $\mathbf{u}(\cdot|t) \in \mathbb{R}^{n_u\times N}$. The resulting NMPC scheme is based on the solving the NLP at time t given by

$$\begin{aligned}
\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \sum_{k=0}^{N-1} l(\mathbf{x}(k|t),&\mathbf{u}(k|t), \mathbf{r}(k|t)) + \frac{1}{2}\mathbf{s}^\top\mathbf{P}\mathbf{s} && k \in [0, ..., N] && \text{(6.26a)}\\
\text{s.t.}\quad \mathbf{x}(0|t) &= \mathbf{x}(t) && && \text{(6.26b)}\\
\mathbf{x}(k+1|t) &= \mathbf{f}_{\text{ERK4}}(\mathbf{x}(k|t), \mathbf{u}(k|t), \mathbf{d}(0|t)) && k \in [0, ..., N] && \text{(6.26c)}\\
\mathbf{h}(\mathbf{x}(k|t), \mathbf{u}(k|t), \mathbf{s}) &\geq \mathbf{0} && k \in [0, ..., N], && \text{(6.26d)}
\end{aligned}$$

with the current state $\mathbf{x}(t)$ and a positive definite and symmetric weighting matrix $\mathbf{P} \in \mathbb{R}^{n_s\times n_s}$. The disturbance terms are modelled as constant over the prediction horizon. A possible variant is the use of exponentially decaying disturbance terms along the prediction horizon, i.e.

$$\mathbf{d}(k|t) = (1 - e^{-(N-k)T_s/\tau})\mathbf{d}(t), \tag{6.27}$$

for disturbance which can be expected to be zero-mean. Here T_s denotes the length of the shooting intervals. The parameter $\tau \in \mathbb{R}$ is a tuning parameter that can be used to change the rate of decay of the disturbance terms.

After each update of the NMPC, the predicted optimal state sequence $\mathbf{x}^*(\cdot|t)$ is used to extract the reference signals for the low-level geometric controller, given by $(\mathbf{\Gamma}(t)$, $\boldsymbol{\omega}^b_{nb,d}(t)$, $\dot{\boldsymbol{\omega}}^b_{nb,d}(t))$. This is done after the first shooting interval as

$$\begin{bmatrix} \mathbf{\Gamma}_d(t) \\ \boldsymbol{\omega}^b_{nb,d}(t) \\ \delta_t(t) \end{bmatrix} = \text{blkdiag}(\mathbf{0}_{15}, \mathbf{I}_7)\mathbf{x}^*(1|t) \tag{6.28}$$

and

$$\dot{\boldsymbol{\omega}}_{nb,d}^{b}(t) = \text{blkdiag}(\mathbf{I}_3, 0)\mathbf{u}^*(1|t). \tag{6.29}$$

The throttle command $\delta_t(t)$ is sent to directly the UAV as shown in Fig. 6.1.

6.3.5 Disturbance Observer

For offset-free attitude stabilization in the presence of e.g. unmodelled dynamics, parametric disturbances or estimation errors, we include an observer for moment and force disturbances that affect angular rate and airspeed. This follows the general discussion on offset-free NMPC in [134]. Consider the observed airspeed $\hat{V}_a$ and angular rate $\hat{\boldsymbol{\omega}}_{nb}^{b}$ and the difference to their predictions in the controller at each time instant

$$\Delta V_a(t) = \hat{V}_a(t) - (\mathbf{e}_1^{n_x})^\top \mathbf{x}^*(1|t-1) \tag{6.30}$$

$$\Delta\boldsymbol{\omega}_{nb}^{b}(t) = \hat{\boldsymbol{\omega}}_{nb}^{b}(t) - \begin{bmatrix}\mathbf{0}_{3\times 12} & \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 7})\end{bmatrix} \mathbf{x}^*(1|t-1). \tag{6.31}$$

For simplicity, we discuss the case in which the update period and prediction interval of the controller have equal length.[1] For different update rates of the controller and the disturbance observer, a practical approach is to separately integrate the dynamics of the NMPC at the desired rate of the observer. Another alternative is to map the update period of the disturbance observer to the shooting intervals of the predicted state trajectory of the controller and then apply interpolation based on the neighboring points of $\mathbf{x}(\cdot|t)$.

The disturbance estimates are initialized as $\mathbf{d}_f(0) = \mathbf{d}_m(0) = \mathbf{0}_{3\times 1}$ and then updated continuously together with the NMPC

$$\mathbf{d}_f(t) \leftarrow \mathbf{d}_f(t) + l_f \begin{bmatrix}\Delta V_a(t) & 0 & 0\end{bmatrix}^\top \tag{6.32}$$

$$\mathbf{d}_m(t) \leftarrow \mathbf{d}_m(t) + \mathbf{L}_m \Delta\boldsymbol{\omega}_{nb}^{b}(t) \tag{6.33}$$

with $l_f \in \mathbb{R}$ as the learning gain for the force disturbance and $\mathbf{L}_m \in \mathbb{R}^{3\times 3}$ defined as

$$\mathbf{L}_m = \text{diag}(l_p, l_q, l_r) \tag{6.34}$$

acting as the learning gain for the moment disturbance which is the diagonal composition of the individual gains for the angular rates $l_p, l_q, l_r \in \mathbb{R}$.

6.3.6 Implementation

We implement the NMPC using the open-source software package Acados [185] using a fourth-order explicit Runge-Kutta integrator for discretization. To solve the NLP, we use real-time iteration sequential quadratic programming (RTI SQP) [46] with the high-performance interior point method presented in [57] for the solutions of the underlying QPs. The QP solver is relying on the numerical subroutines of BLASFEO [58].

[1] An alternative approach with different update rates is presented in Chapter 5.

6.4 Numerical Results and Discussion

6.4.1 Simulation Study

UAV & Wind Model

The UAV model used for all simulation is the Aerosonde with dynamic model and parameter set as given in [9]. It includes the control surfaces aileron, elevator rudder and a throttle. The control effectiveness matrix of the rotational dynamics has thus full rank. This is needed to be able to employ the version of the geometric controller presented in Chapter 4 against which we compare the MPCs. The nominal cruise speed of the Aerosonde is 35 m/s with a 250 m turn radius[2]. The actuators in the simulation are modelled with first-order lag dynamics

$$\dot{\delta}_i = \frac{1}{T}(\delta_{i,d} - \delta_i) \tag{6.35}$$

and $T = 0.01$ for the control surfaces, i.e. $i \in \{a, e, r\}$ and $T = 1$ for the throttle δ_t. The wind is modelled as the combination of a static component in the inertial frame and a gust component in the body-fixed frame as $\mathbf{v}^n_{nw} = \mathbf{v}^n_{nw,s} + \mathbf{R}_{nb}\mathbf{v}^b_{nw,g}$. The gust component is generated by the Dryden wind model, which means essentially passing white noise through a low-pass filter. The static wind component in the inertial frame is set to $\mathbf{v}^n_{nw,s} = \begin{bmatrix} 4 & 3 & 0 \end{bmatrix}^\top$. The design of an observer for the estimation of the wind-velocity vector is out of scope of this paper, and we assume the wind to be known at all times. To take the changing wind conditions into account, the output target that determines reference values for airspeed and the reference $\mathbf{r}$ is updated at 10 Hz in all simulations. Both the propagation of the state and the controller updates are based on the same model parameters. Seeing as all controllers are model-based, this is still a fair comparison.

Geometric Controller Setup

The geometric controller is updated at 100 Hz and parameterized based on the gains $k_p = 9.5$ and $\mathbf{K}_d = 8\mathbf{I}_3$. The same parameterization is used in the pure geometric attitude controller and in the combined controller of NMPC and geometric controller.

Nonlinear Model Predictive Controllers

Both MPCs are updated at 10 Hz. The weighting matrices of the pure NMPC are chosen as

$$\mathbf{Q}_{\text{NMPC}} = \operatorname{diag}(10^{-2}, 10, 1, 1), \tag{6.36a}$$

$$\mathbf{R}_{\text{NMPC}} = \operatorname{diag}(10^{-1}, 10^{-1}, 10^{-1}, 10^{-1}), \tag{6.36b}$$

$$\mathbf{P}_{\text{NMPC}} = \operatorname{diag}(1, 10^{3}, 1, 10^{3}), \tag{6.36c}$$

[2]http://uavbook.byu.edu/doku.php

and the weighting matrices of the combined controller are

$$\mathbf{Q}_{\text{NMPC+GC}} = \text{diag}(10^{-2}, 10, 1, 1), \tag{6.37a}$$

$$\mathbf{R}_{\text{NMPC+GC}} = \text{diag}(5 \cdot 10^{-3}, 5 \cdot 10^{-3}, 5 \cdot 10^{-3}, 10^{-1}), \tag{6.37b}$$

$$\mathbf{P}_{\text{NMPC+GC}} = \text{diag}(1, 10^3, 1, 10^3). \tag{6.37c}$$

Both parameterizations differ only in the first three elements of the input weight matrix $\mathbf{R}$, given that the pure NMPC sets the rates of the actuators directly, whereas the combined controller sets the angular acceleration commands of the low-level control in addition to the throttle rate command. Regarding reference following, the pitch reference is prioritized. There is a lower penalty on airspeed deviations, note however that violations of the constraint are given a significantly higher penalty due to the weighting matrix of the slack variables $\mathbf{P}$. Finally, violation of the angle-of-attack constraint receives the highest cost. This is to ensure that the UAV remains within bounds to avoid stall.

For evaluation of reference following performance for a variable a and actuator usage $\mathbf{u}$, we use the performance indices

$$J_{e_a} = \frac{1}{T}\int_0^T \|\mathbf{e}_a\| ds, \qquad J_u = \frac{1}{T}\int_0^T \|\mathbf{u}\| ds \tag{6.38}$$

6.4.2 Disturbance Rejection

To illustrate the disturbance rejection capabilities of the different controllers, we use constant reference signals for roll and pitch angle to command a loitering maneuver with a 250 m radius in the horizontal plane. The constant wind velocity in this scenario is set to $\mathbf{v}^n_{nw} = 0_{3\times 1}$, and the gust winds are simulated for low, moderate and high intensity.

The results for 300 s of simulations with 0.01 s step size are summarized for each controller and gust intensity in Table 6.1. Attitude disturbances are significantly better rejected by the combined controller, when compared to the pure NMPC for all levels of turbulence, which is indicated by J_{e_ϕ}, J_{e_θ}. Also, actuator usage as indicated by J_u remains approximately at the same level in contrast to the pure NMPC. On the other hand, the pure geometric controller appears to have the best performance regarding reference following in both airspeed and pitch as indicated by $J_{e_{V_a}}$ and J_{e_θ}. This is likely to be caused by the reference signals for angular rate and acceleration set by the NMPC, which are then tracked by the low-level geometric controller (GC) between NMPC updates. However, the most suitable references would be the zero vector for each, which is what the pure GC assumes. Thus, in this case the references generated by the NMPC may actually cause a degrading performance.

6.4.3 Recovery

To show that the NMPC is a viable alternative to the hybrid controller discussed in Section 4.5, we simulate a similar scenario as for the hybrid case in which the UAV needs to be recovered from $\phi_0 = -175$ deg to $\phi_{\text{ref}} = \theta_{\text{ref}} = 0$. However, different to

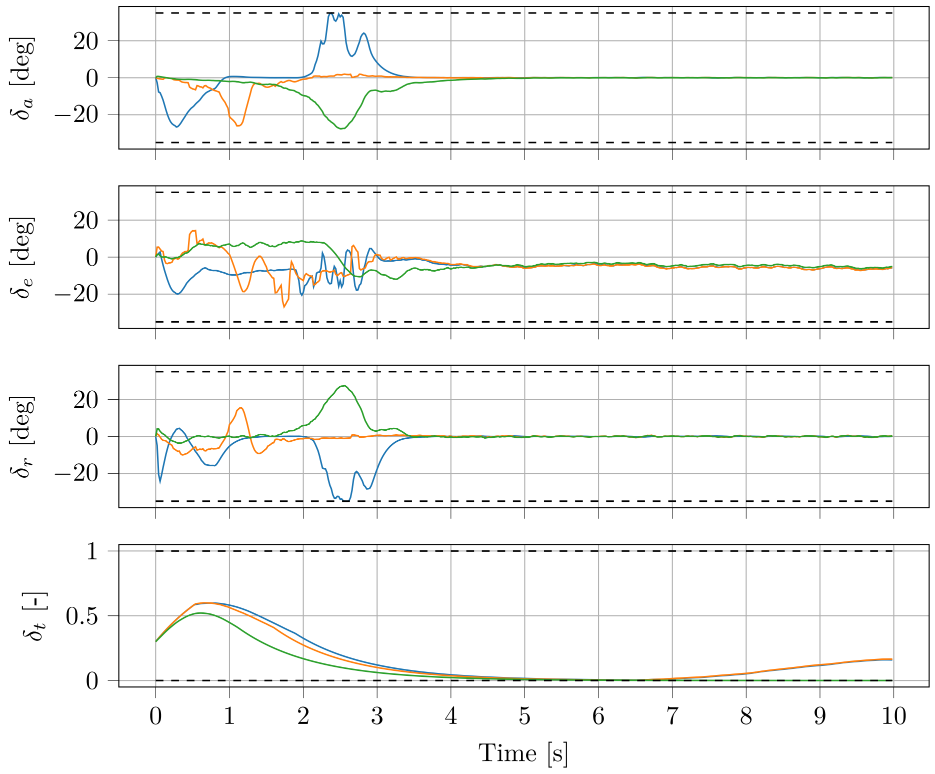

Figure 6.2: Input trajectories of the controllers for the global stabilization case including that of the pure NMPC (blue), combined controller (orange) and the pure geometric controller (green).

the scenario as discussed in Section 4.6, the initial pitch angle is set to $\theta_0 = 0\,\text{deg}$. Considering that this is a highly dynamic scenario in which fast solutions of the NLP solver are desirable, we choose a warm-start for both MPCs in which the initial guess is initialized as the shifted trajectory of the optimal state predictions of the previous iteration.

In this case the continuous geometric attitude controller suffers from a performance loss due to the close unstable equilibrium in the vicinity of the initial state which results in reduced proportional action. A consequence of this is that the attitude stays close to the initial roll angle for a longer time as can be seen in Fig. 6.3, which at $t = 2\,\text{s}$ leads to a violation of the airspeed constraint. Both MPCs show significantly faster response by making use of all actuators (see Fig. 6.2), as contrasted by the pure GC which does not use aileron and rudder to a notable extent up to this point into the simulation.

However, it is notable that the slower response of the geometric control law carries over to the mixed controller, when compared to the pure NMPC. The most prominent indicator is that the pure NMPC leads to operation at the limit of the angular rate constraints Fig. 6.4 to stabilize the UAV at the commanded attitude.

Table 6.1: Performance indices for airspeed and attitude errors.

Var	Intensity	NMPC	NMPC GC	GC
$J_{e_{V_a}}$ [m/s]	low	0.187	0.190	0.202
	medium	0.430	0.458	0.368
	high	0.499	0.510	0.512
J_{e_ϕ} [deg]	low	0.033	0.016	0.020
	medium	0.072	0.029	0.037
	high	0.113	0.047	0.060
J_{e_θ} [deg]	low	0.038	0.021	0.019
	medium	0.090	0.064	0.035
	high	0.137	0.106	0.043
J_u [deg]	low	3.323	3.327	3.372
	moderate	3.499	3.341	3.418
	high	3.729	3.304	3.358

Further, note that the mixed controller and the pure NMPC start to enacting a different strategy at $t = 2\,\mathrm{s}$ to keep the UAV close to the constraint set. The magnitude of the constraint violation is subject to the tuning of the cost $\mathbf{P}$ of the slack variables.

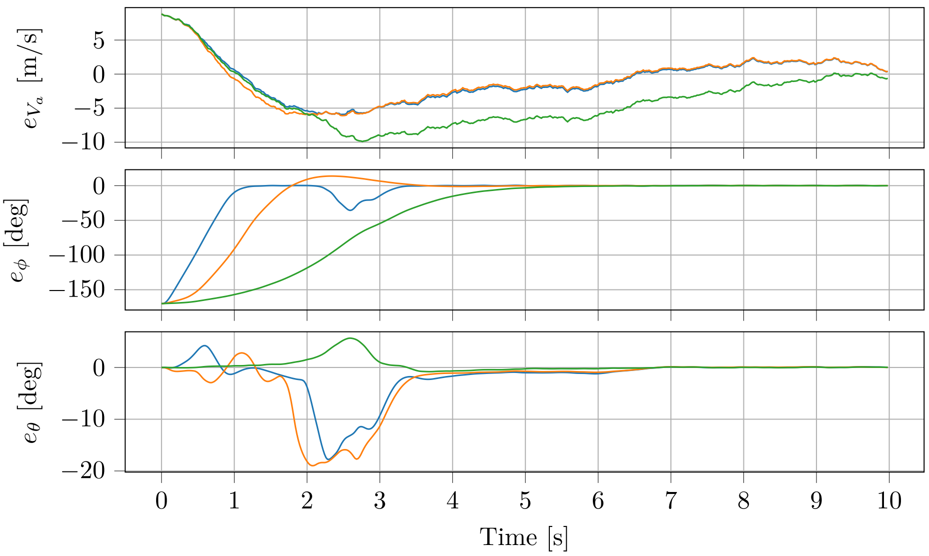

Figure 6.3: Speed and attitude trajectories of the controllers for the global stabilization case including that of the pure NMPC (blue), combined controller (orange) and the pure geometric controller (green).

While this case shows that the NMPC can act as a viable alternative to hybrid control regarding the problem of performance loss in the vicinity of the unstable equilibrium on the two-sphere, it has to be noted that this example is however purely academic as such as it requires a globally valid dynamic model. The suc-

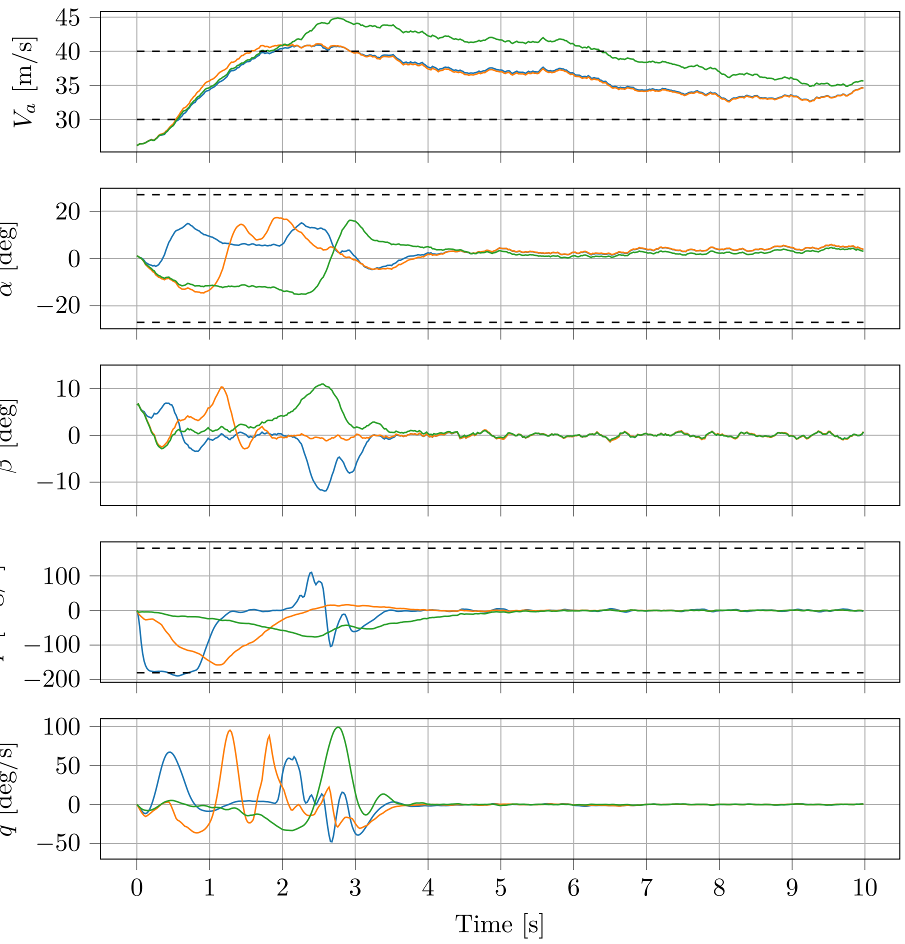

Figure 6.4: Aerodynamic data and angular rates for the controllers for the recovery case including that of the pure NMPC (blue), combined controller (orange) and the pure geometric controller (green). The state constraints (dashed) are included to ensure nominal flight conditions and model validity below the stall regime.

cessful identification of the dominant dynamics of a model for the entire state space of the attitude dynamics can surely be regarded as a hard task in practice.

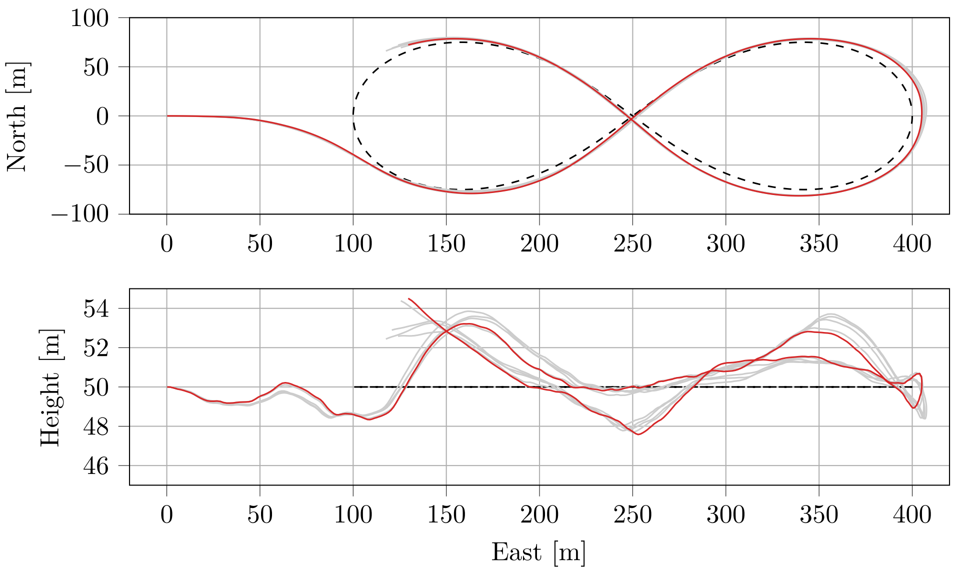

Figure 6.5: Benchmark simulation comparing the combined controller (MPCGC, plotted in red) introduced in this chapter to the previous controller designs (gray). The reference path is plotted in black, dashed.

6.5 Benchmark Scenario

As a geometric attitude controller, we use a similar design as discussed in the benchmark of Chapter 4, but without the integral action. The resulting control law for the geometric attitude controller has therefore a simple PD structure

$$\mathbf{u} = \mathbf{G}(\mathbf{v}_r)^{\dagger}(-k_p\mathbf{e}_\Gamma - \mathbf{K}_d\mathbf{e}_\omega). \tag{6.39}$$

The same control law is consequently also used in the definition of OCP. The gains for the geometric controller in combination with the MPC are the same as in Section 4.8, i.e. $k_p = 20.0$ and $\mathbf{K}_i = \mathbf{K}_d = \text{diag}(2.0, 2.0, 2.0)$. The difference is now that the desired angular velocity $\boldsymbol{\omega}_d$ is not set to zero, but determined by the MPC. The MPC is tuned with the following cost matrices

$$\mathbf{Q} = \text{diag}(q_{V_a},\ q_{\Gamma,x},\ q_{\Gamma,y},\ q_{\Gamma,z}) = \text{diag}(0.01,\ 30,\ 30,\ 30) \tag{6.40a}$$

$$\mathbf{R} = \text{diag}(r_{\dot{\omega}_x},\ r_{\dot{\omega}_y},\ r_{\dot{\omega}_z},\ r_{\dot{\delta}_t}) = \text{diag}(10^{-3},\ 10^{-3},\ 10^{-3},\ 10^{-1}) \tag{6.40b}$$

The geometric attitude controller is updated at 50 Hz and the MPC at 20 Hz. The trajectories for the position are plotted in Fig. 6.5 and the distance, error signals and actuator signals are depicted in Fig. 6.6. In this case, the comparison to the geometric controller derived in Chapter 4 is particularly interesting, as it allows us to gauge the performance improved achieved by adding the MPC into the loop.

The plots for all controllers and the performance metrics are given in Appendix A. The mean squared error (MSE) of the roll angle, $J_{e,\phi}$, and pitch angle,

$J_{e,\theta}$, shows that the combined controller achieves better or comparable performance. The MSE concerning the airspeed, J_{e,V_a} also got improved, which can be attributed to the coupling between pitch angle and airspeed that is taken into account by the MPC. As shown for the tuning steps Fig. A.1, the combined controller actively uses a pitch-down or pitch-up motion to respectively accelerate or decelerate the UAV.

Regarding the actuator velocities captured by the smoothness metric $J_{f,\cdot}$, the combined controller is commanding notably faster set-point changes, which can be mitigated by a less aggressive tuning of the MPC. It is however the rule for all MPC designs in this thesis to favor tracking performance over actuator usage.

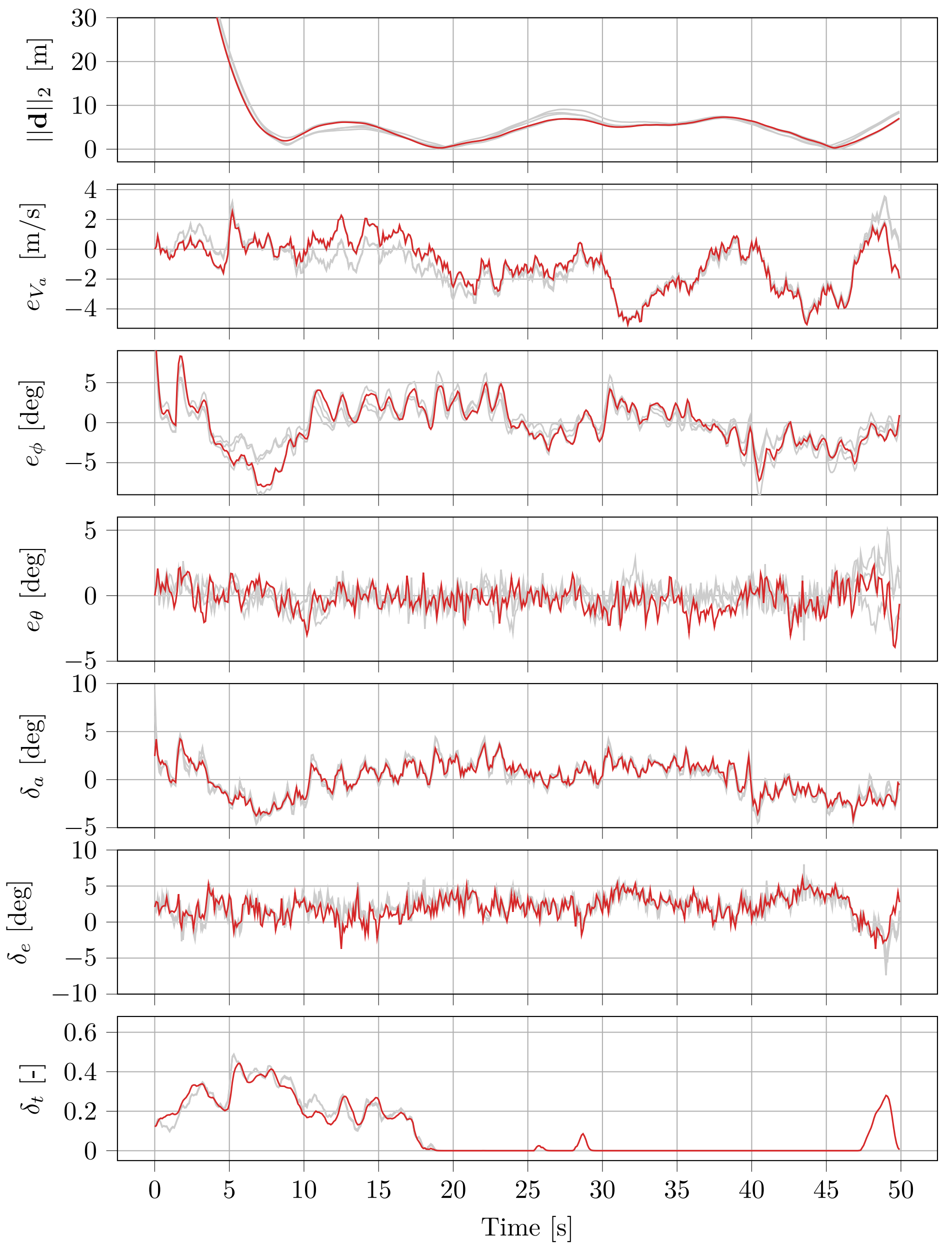

Figure 6.6: Benchmark simulation comparing the combined controller (MPCGC, plotted in red) introduced in this chapter to the previous controller designs (gray). The first subplot includes the distance to the reference path $\|\mathbf{d}\|_2$, and the following subplots include the error signals for airspeed, roll and pitch, as well as the actuator signals.

6.6 Chapter Summary

We looked into a cascade of NMPC and smooth geometric attitude control as introduced in Chapter 4, where the NMPC tracks linear acceleration commands from a guidance module and generates the reference signals for angular acceleration, angular rate and the reduced attitude vector. This makes the cascaded controller not only an alternative to the hybrid extension, but exploits the tracking controller design of the geometric attitude controller.

The controller design was compared to the smooth geometric attitude controller and a NMPC with direct actuator access in simulation scenarios. Significant performance improvements to the smooth geometric attitude controller were evident, and moreover constraint satisfaction was achieved. Disturbance rejection capabilities of the cascaded controller were better than that for the NMPC with direct actuator access which in turn proved to be the best choice in a recovery scenario.

We therefore put a stronger emphasis on the NMPC with direct actuator access in the preceding chapter to see if the required update rates can be achieved on the experimental platform introduced in Chapter 2. Note that as for the NMPC in Chapter 5, the controller designed in this chapter includes the coupling between pitch control and airspeed control in addition to the angle of attack constraints. It is therefore also suited for operations at low airspeeds where the UAV is at increased risk of stalling.

Chapter 7

Extended Aerodynamic Modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle

7.1 Introduction

The experimental results of the preceding chapter revealed that the controller included a dynamic model that has potential for improvements. Our experiments were the first to evaluate the model proposed by Gryte et al. [71] with in-flight tests. In their paper, Gryte et al. merely auto-validate the identified models against the measured wind tunnel data, resulting in favorable scores of the coefficient of determination very close to 1. However, the resulting score is even negative when we compared the accelerations obtained through the force model and the observations based on data from IMUs. This indicates that using the model in a model-based controller does not improve performance but makes it worse in some situations. Considering that the targeted use case by Gryte et al. is model-based flight control or model-based estimation or aerodynamic parameters, we see it crucial to apply the fitness metric to actual flight data.

With a closer look at the wind tunnel data that serves as a foundation for the following model identification process, we identified four possible modifications that can improve the model. First, the dataset includes asymmetries concerning the longitudinal plane that divides the airframe into the fuselage's left and right sides. Gryte et al. mention the possible misalignment of the body-fixed frame and the wind tunnel at zero sideslip angle to be the cause for the asymmetries. However, instead of finding a proper rotation, they addressed the problem of asymmetric measurements in an ad-hoc manner by setting specific parameters to zero to obtain the desired symmetries in the model. We propose a different approach to minimize the asymmetries in a least-squares sense using a calibration of the data prior to identifying the model. Second, the lever arm of the attack point of the force vector to the center of mass is implicit in the model. Gryte et al. tweaked the vector describing the lever arm to have the pitch moment aligned with their experience. We instead explicitly model the generalized forces at the aerodynamic reference point, the location of the force sensors during the experiments, and identify the vector to center of mass based on flight data. Third, we have a more thorough look at the subset of the obtained data suitable for identifying aerodynamic linear coefficients in the sideslip angle. We moreover reduce the drag model's complexity and simultaneously improve its fitness. Fourth, the final set of parameters proposed by Gryte et al. is a blend of parameters identified by the wind tunnel, results from a Vortex-Lattice method (XFLR), and additional manual tweaks to the parameters based on experience from flight experiments. The result is a set of parameters that is not optimal in either of the original identification procedures. We present a different approach that combines the use of wind tunnel and flight data, which leads to a consistent parameter vector.

This chapter aims to find an improved model compared by dealing with the outlined shortcomings of the described identification process. We begin by presenting a novel method to calibrate the dataset for the symmetry concerning the longitudinal plane of the airframe and follow with the modified model structure before identifying the aerodynamic coefficients. A significant part will be identified based on the same wind tunnel data used by Gryte et al. However, instead of finding the damping coefficients for the rotational motion through a Vortex-Lattice method, we use actual flight data. Finally, evaluate the new model compared to the baseline model from Gryte et al. and a model entirely based on the estimated coefficients from the flights. The outlined procedure to find an improved model is illustrated in Fig. 7.1.

The following notation related to the aerodynamics is different to the notation used in the preceding chapters. This is a consequence of our effort to describe the aerodynamics in a manner that takes geometric properties into account, in particular for the design of the calibration procedure in Section 7.2.2. Another motivation was to describe the aerodynamics moments and forces as a function of the relative linear and angular velocities, which would reduce the need for trigonometric terms, i.e. for the angle of attack and sideslip angle, and thus reduce the nonlinearities in a model-based controller.

7.1.1 Related Work

The work for this chapter is not the first to consider the aerodynamic model of the X8. In addition to [71], which is a continuation of [69], several publications consider this popular airframe. Farhadi et al. [131] identify a lateral model of the X8 using flight data, in a combination of the output-error method and ordinary least squares regression. Gan et al. [60] identify the static aerodynamic coefficients from a Reynolds-averaged Navier-Stokes (RANS) computational fluid dynamics (CFD) program. Winter et al. [192] also provide a RANS CFD analysis, further using time-series CFD simulations to study the effect on the dynamic coefficients in situations where the airfoil is subject to ice aggregation. All three papers rely to some extent on our previous work [69, 71]. System identification based on flight data is also a well-established field [82, 97], and has been widely used in modelling of fixed-wing UAVs, both in the time- [112, 167] and frequency domain [123, 182]. More recently, Kaiser et al. [88] presented results where they identify the longitudinal dynamics

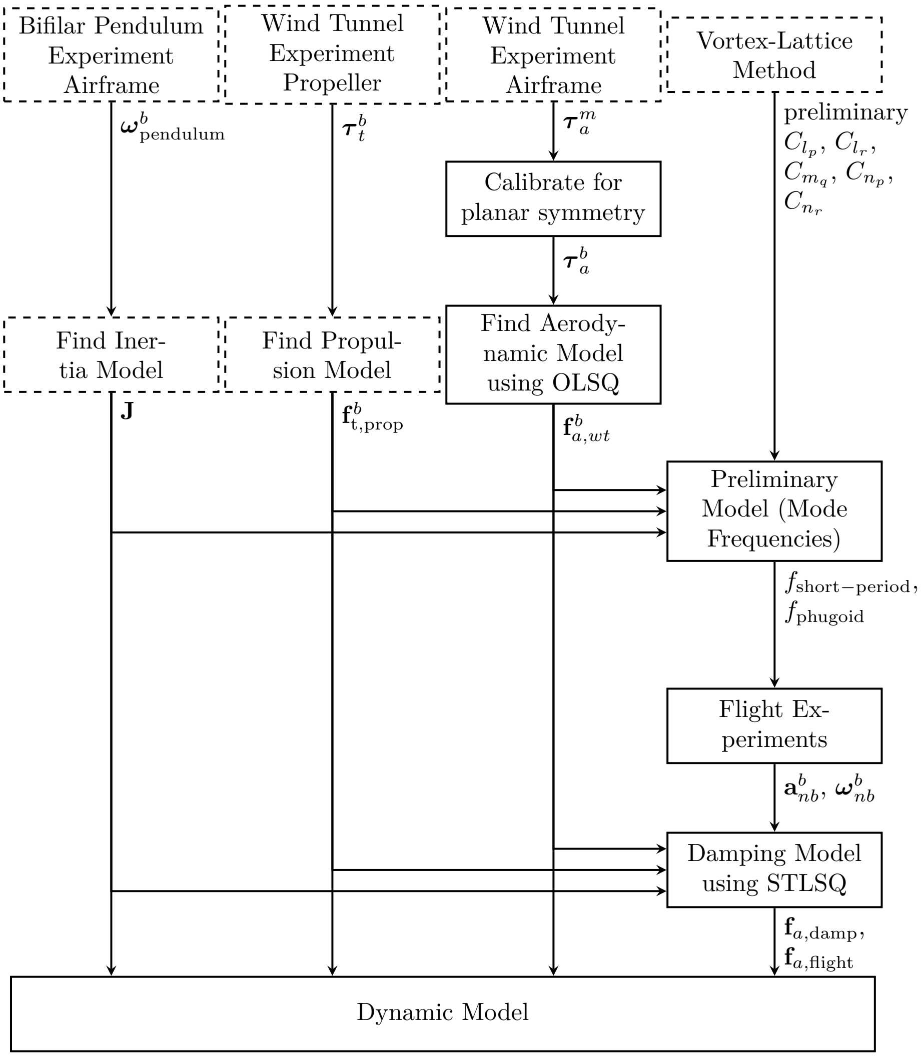
Bifilar Pendulum Experiment Airframe
Wind Tunnel Experiment Propeller
Wind Tunnel Experiment Airframe
Vortex-Lattice Method
$\boldsymbol{\omega}^b_{\text{pendulum}}$
$\boldsymbol{\tau}^b_t$
$\boldsymbol{\tau}^m_a$
preliminary C_{l_p}, C_{l_r}, C_{m_q}, C_{n_p}, C_{n_r}
Calibrate for planar symmetry
$\boldsymbol{\tau}^b_a$
Find Inertia Model
Find Propulsion Model
Find Aerodynamic Model using OLSQ
$\mathbf{J}$
$\mathbf{f}^b_{\text{t,prop}}$
$\mathbf{f}^b_{a,wt}$
Preliminary Model (Mode Frequencies)
$f_{\text{short-period}}$, f_{phugoid}
Flight Experiments
$\mathbf{a}^b_{nb}$, $\boldsymbol{\omega}^b_{nb}$
Damping Model using STLSQ
$\mathbf{f}_{a,\text{damp}}$, $\mathbf{f}_{a,\text{flight}}$
Dynamic Model

Figure 7.1: System Identification Flow. Previous work that this chapter is built on is marked in dashed boxes and details can be found in the references [39, 70, 71].

of a fighter aircraft using the Sparse Identification of Nonlinear Dynamics (SINDy) proposed by Brunton et al. [16].

Fixed-wing UAV designs are usually symmetric with respect to the longitudinal plane, such that the left-hand side mirrors the right-hand side of the airframe. The goal is for the generalized aerodynamic forces to be symmetric for equivalent maneuvering capabilities during turns in either direction. To identify the forces for a given airframe design, engineers often collect data that capture the forces in wind tunnel tests or flight experiments. In either case, one would expect the magnitude of the forces to be equal for symmetric use of the actuators and mirrored relative velocities concerning the plane of symmetry. However, this is not the case for the collected data when the coordinate axes of the force and moment measurement equipment are not aligned with the coordinate axes of the body-fixed coordinate frame, which is usually the case. This asymmetry then propagates to the identified models and can be problematic in model-based control, which is the use-case we are targeting.

The misalignment can be kept small by careful mounting procedures, such that it is possible to calibrate for remaining asymmetries through proper post-processing. However, it appears that a systematic calibration method to do this has not been addressed, whereas engineering researchers focused on the compensation of other error sources and effects. Molinari et al. [132] consider disturbances due to the airframe and wall of the wind tunnel blocking the airstream. They isolate the airfoil in a pitching motion such that symmetries to sideslip angle variations are not a factor. Damljanovic et al. [42] also focus on wall interference. They note possible offsets between coordinate axes of the measurement equipment and airframe but state that the effects are in the order of magnitude of measurement noise and disregard further calibration.

Ocokoljic et al. [142] extensively discuss wind tunnel calibration tests, including the test section, measurement instruments, and standard models for calibration. Jindeog et al. [83] consider possible measurement biases due to thermal effects but only within the longitudinal plane where asymmetries in sideslip angle variations can not be observed. Simmons et al. [167] focus on experiments design and system identification techniques and do not discuss asymmetries.

Holsten et al. [79] notice asymmetries in data series that they expect to be symmetrical but do not offer a solution to compensate for this. The data provided by Ol et al. [144] looks to be sufficiently symmetric, but they do not give detailed information on possible post-processing steps. They note, however, the need for cautious treatment of the actuators. In particular, when testing off-the-shelf radio-controlled UAVs, consumer-grade servos to actuate the control surfaces can introduce significant offsets to the reference deflections due to the hysteresis band. A possible reason for neglecting symmetry considerations in the literature may be that this problem is likely more pervasive for airframes of fixed-wing UAVs that are small and have a low cost.

In Section 7.2.2, we discuss an approach to find a static rotation that aligns the coordinates of the measurement equipment in a wind tunnel experiment with the coordinate axes of the body-fixed frame of a fixed-wing UAV. The necessary assumption that the UAV is symmetric concerning its longitudinal plane is satisfied by most classes of small fixed-wing UAVs and aircraft.

Our approach builds on the formulation of a symmetry condition along the coordinate axes of the force and moment measurement equipment to formulate a suitable error function. We then formulate an optimization problem that finds a transformation that minimizes the asymmetries in the generalized forces (i.e. linear forces and angular momentums) for symmetric relative velocities in a least-squares sense. We base our work on a dataset from a wind tunnel. However, it is straightforward to apply the proposed method to flight data and find the orientation of a relative velocity sensor with respect to the body-fixed frame.

7.2 Wind Tunnel Model

7.2.1 The Dataset

The experiments conducted by Gryte et al. are thoroughly described in their paper [71]. The collected dataset includes variations of the airspeed, angle of attack, and sideslip angle, as well as surface deflections. The collected data is based on the assumption of decoupled dynamics in the lateral and longitudinal plane at small aerodynamic angles. This means that it includes rotations of the lateral plane at zero angles of attack and rotations of the longitudinal plane at zero sideslip angle. The angle of attack mainly was varied between $-10\,\text{deg}$ and $15\,\text{deg}$ degrees with some measurements at higher values and low airspeeds to identify forces in the stall regime. Gryte et al. tested five uniformly distributed elevator deflections between $-20\,\text{deg}$ and $20\,\text{deg}$ at airspeeds set to either $18\,\text{m/s}$ or $21\,\text{m/s}$. With zero aileron deflection and sideslip angle, the data points available for the identification of the longitudinal coefficients are given by

$$(V_a,\, \alpha,\, \delta_e) \in \{18, 21\} \times [-10, 15] \times \{-20, -10, 0, 10, 20\}. \tag{7.1}$$

The dataset for identifying the lateral coefficients includes sideslip angle variations within $-15\,\text{deg}$ and $15\,\text{deg}$ at the same airspeeds as for the longitudinal tests. Based on the assumption of the aileron to be symmetric around zero, the dataset only includes negative deflections. At zero elevator deflection and angle of attack, the lateral data points are given by

$$(V_a,\, \beta,\, \delta_a) \in \{18, 21\} \times [-15, 15] \times \{-20, -10, 0\}. \tag{7.2}$$

Forces and moments due to the vehicle's weight were compensated during each run. The airframe was carefully mounted onto the force sensor to align the measurement axes with the axes of the body-fixed frame. The measurement axes are assumed to be in alignment with the axes of the wind tunnel at zero angles of attack and sideslip angles. However, the measured dataset does not appear to be symmetric concerning the longitudinal plane of the airframe, which is shown in Fig. 7.2 for a sweep of sideslip angles. We will first address this problem and show how one can find a static rotation matrix to improve the symmetry of the available dataset.

7.2.2 Calibration of the Dataset for Planar Symmetry

Problem

We have an airframe for which we impose the assumption

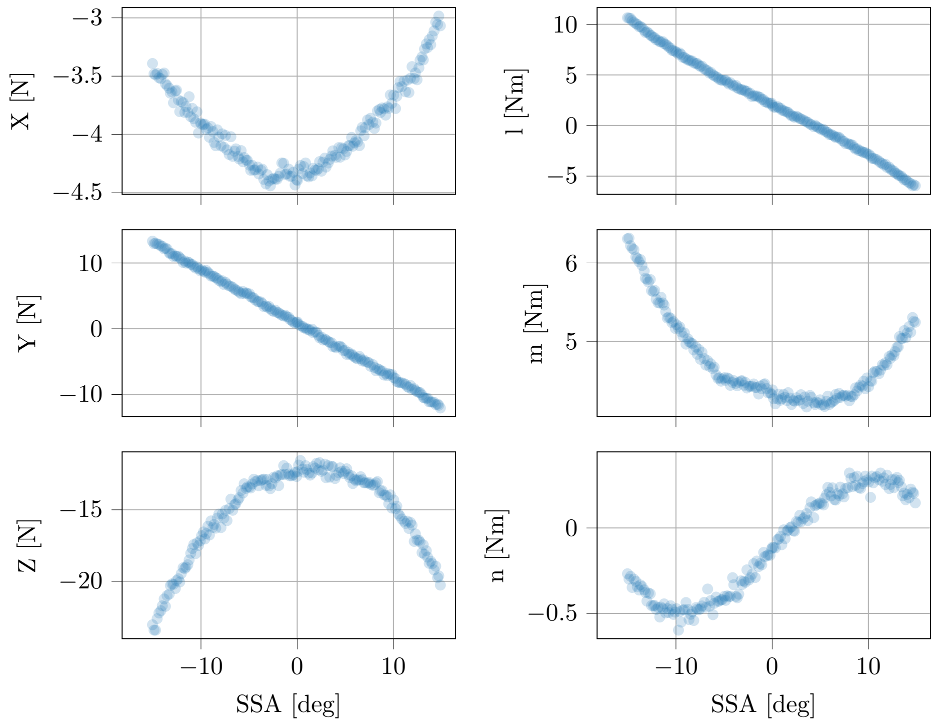

Figure 7.2: A sweep of the sideslip angle with $\alpha = \delta_a = \delta_e = 0$ and $V_a = 21\text{m/s}$. The measured data does not appear to be symmetric to the longitudinal plane (zero sideslip angle) of the airframe.

Assumption 12. The geometry of the airframe is symmetric with respect to the xz-plane.

However, a set of measurement data recorded in the wind tunnel does not show this symmetry when the xz-plane is rotated, as depicted in Fig. 7.2. We assume a misalignment between the body-fixed frame that includes the symmetry plane and the measurement frame in which the data is recorded. Let the orientation of the body-fixed frame relative to the measurement frame be denoted by $\mathbf{R}_{mb} \in \mathrm{SO}(3)$. The problem is to find the rotation $\mathbf{R}_{mb}$ merely based on the available measurement data that is to be used for model identification and without a prior symmetry calibration routine in the experiments.

Symmetry and Transformations

Let the relative linear velocity $\mathbf{v}_r \in \mathbb{R}^3$ and the relative angular velocity $\boldsymbol{\omega}_r \in \mathbb{R}^3$ be concatenated into one vector $\boldsymbol{\nu}_r = [\boldsymbol{v}_r;\, \boldsymbol{\omega}_r]$. Let the vector of generalized forces be denoted by $\boldsymbol{\tau} = [\mathbf{f}^b;\, \mathbf{m}^b]$, and suppose $\mathbf{f} : \mathbb{R}^6 \to \mathbb{R}^6$ denotes a mapping from velocities to forces. The general symmetry condition that we impose reads as

$$\mathbf{M}\mathbf{f}(\mathbf{M}\boldsymbol{\nu}_r^b) = \mathbf{f}(\boldsymbol{\nu}_r^b), \tag{7.3}$$

with the case of xz-symmetry being defined by the symmetry matrix $\mathbf{M}$ given as

$$\mathbf{M} \triangleq \operatorname{diag}(1, -1, 1, -1, 1, -1). \tag{7.4}$$

Now suppose that the body-fixed frame $\{b\}$ and the sensor-fixed frame $\{m\}$ are not aligned, but that the orientation of $\{b\}$ with respect to the coordinate axes of $\{m\}$ is given by a rotation $\mathbf{R}_{mb} \in \mathrm{SO}(3)$. Then we can express the velocity in $\{m\}$ as

$$\boldsymbol{\nu}_r^m = \mathbf{T}\boldsymbol{\nu}_r^b \tag{7.5}$$

where

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{mb} & \mathbf{0} \\ \mathbf{S}(\mathbf{r}_{mb}^m)\mathbf{R}_{mb} & \mathbf{R}_{mb} \end{bmatrix}. \tag{7.6}$$

The variable $\mathbf{r}_{mb}^m \in \mathbb{R}^3$ denotes the position of the origin of $\{b\}$ with respect to $\{m\}$ in coordinates of $\{m\}$. We make a few simplifying assumptions

Assumption 13. The sensor is placed at the origin of the body-fixed frame, i.e. $\mathbf{r}_{mb}^m \approx \mathbf{0}$ such that $\mathbf{T} \approx \operatorname{blkdiag}(\mathbf{R}_{mb}, \mathbf{R}_{mb})$ and $\mathbf{T}^{-1} = \mathbf{T}^\top$.

Note that the origin of $\{b\}$ does not necessarily coincide with the center of mass.

Assumption 14. The wind tunnel produces a homogenous air stream with a negligible angular velocity component, i.e. $\boldsymbol{\omega}_r \approx \mathbf{0}$.

Let the coordinate frame of the wind tunnel be described by $\{w\}$. Each data point includes measurements of the relative linear velocity vector in the sensor frame, i.e. $\mathbf{v}_r^m$, typically recorded in terms of airspeed $V_a \in \mathbb{R}$, angle of attack $\alpha \in \mathbb{R}$ and sideslip angle $\beta \in \mathbb{R}$. This can be thought of as magnitude and spherical direction of $\mathbf{v}_r^m$. To parameterize $\mathbf{v}_r^m$ from the data points, we define the map $\boldsymbol{\mu} : \mathbb{R}^3 \to \mathbb{R}^3$ as

$$\mathbf{v}_r^m = \boldsymbol{\mu}(V_a, \alpha, \beta) \triangleq \mathbf{R}_{mw}(\alpha, \beta)\mathbf{v}_r^w \tag{7.7}$$

and its inverse

$$(V_a, \alpha, \beta) = \boldsymbol{\mu}^{-1}(\mathbf{v}_r^m) \triangleq \left(\|\mathbf{v}_r^m\|_2, \arctan\left(\frac{w_r}{u_r}\right), \arcsin\left(\frac{v_r}{\|\mathbf{v}_r^m\|_2}\right) \right). \tag{7.8}$$

The rotation matrix $\mathbf{R}_{mw}(\alpha, \beta) = \mathbf{R}_{ms}(\alpha)\mathbf{R}_{sw}(\beta)$ is a composition of

$$\mathbf{R}_{ms}(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}, \quad \mathbf{R}_{sw}(\beta) = \begin{bmatrix} \cos(\beta) & \sin(\beta) & 0 \\ -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.9}$$

which are the known rotation matrices between body-fixed frame, stability frame and wind frame [9].

Interpolating the Dataset

Our goal is to formulate an optimization problem to find the rotation matrix $\mathbf{R}_{mb}$ based on the available force measurements in the dataset from which expressions for both sides of Eq. (7.3) can be derived. To be able to use gradient-based optimization methods, we need to approximate the measurements by functions that are continuously differentiable with respect to the sideslip angle. In this subsection we give an outline how this can be done using polynomial interpolation. For each sweep of sideslip angles, the airspeed V_a (indicated by the Reynolds Number), and the angle of attack α and elevator deflection δ_e are constant. Moreover, we only consider sweeps during which the aileron deflection δ_a is set to zero, given that a non-zero value would violate the symmetry condition. The aileron deflections are assumed to be offset-free.

To interpolate between data points, we fit polynomial functions of the sideslip angle to the recorded aerodynamic coefficients given as

$$f_{C_k}(\beta) = \sum_{i=0}^{n} a_i \beta^i. \tag{7.10}$$

Given the measurements $(C_X, C_Y, C_Z, C_l, C_m, C_n)$, the coefficients a_i in the functions f_{C_k} can be found by using linear regression for each sweep of the sideslip angle. Upon inspection of the measured aerodynamic coefficients with varying sideslip angle, we found a good compromise between a reasonable fit and low polynomial order for the polynomial order $n = 4$ for C_D, C_L, C_m, $n = 3$ for C_l, C_n and $n = 1$ for C_Y.

We then concatenate the functions f_{C_k} to approximate the function $\mathbf{f}$ in Eq. (7.3) by $\hat{\mathbf{f}}$ defined as

$$\hat{\mathbf{f}}(\boldsymbol{\nu}_r^m|\delta_a, \delta_e) \triangleq \begin{bmatrix} f_{C_x}(\beta) & f_{C_Y}(\beta) & f_{C_Z}(\beta) & bf_{C_l}(\beta) & cf_{C_m}(\beta) & bf_{C_n}(\beta) \end{bmatrix}^\top, \tag{7.11}$$

where $b, c \in \mathbb{R}_{\geq 0}$ denote wingspan and chord length of the airframe, respectively. Note that angle of attack and control surface deflections are implicit in the polynomials f_{C_k}. In addition to the angle of attack, the airspeed and sideslip angle can be obtained by the map $\boldsymbol{\mu}^{-1}$.

To obtain the reflected relative velocity vector in $\{m\}$, the measured relative velocity vector first needs to be transformed to $\{b\}$ using Eq. (7.5), then reflected using Eq. (7.4), and transformed back to $\{m\}$ again. This leads to the reflected relative velocity vector $\boldsymbol{\nu}_r'^m$ given as

$$\boldsymbol{\nu}_r'^m = \mathbf{TMT}^{-1}\boldsymbol{\nu}_r^m, \tag{7.12}$$

in which the expression $\mathbf{TMT}^{-1}$ can be interpreted as a similarity transformation of the reflection matrix $\mathbf{M}$ to the measurement frame $\{m\}$.

Optimization Problem

The symmetry condition Eq. (7.3) can be used to formulate a symmetry error in the body-fixed frame based on the interpolation polynomials as

$$\mathbf{e} = \mathbf{MT}^{-1}\hat{\mathbf{f}}(\mathbf{TMT}^{-1}\boldsymbol{\nu}_r^m|0, \delta_e) - \mathbf{T}^{-1}\hat{\mathbf{f}}(\mathbf{T}^{-1}\boldsymbol{\nu}_r^m|0, \delta_e). \tag{7.13}$$

We use $\mathbf{e}$ to formulate a cost function as the sum over all sideslip angle sweeps with zero aileron deflection. Let the set of all suitable sweeps be denoted by $\mathcal{D}$ and the samples in each sweep be denoted by $N_{\mathcal{D}}$, then the cost function is defined as

$$J = \sum_{\mathcal{D}} \sum_{i}^{N_{\mathcal{D}}} \mathbf{e}_i. \tag{7.14}$$

Suppose the matrix $\mathbf{R}_{mb}$ is parameterized by at set of Euler angles. Then it can be shown that the similarity transformation $\mathbf{TMT}^{-1}$ is invariant to rotations around a vector that is normal to the xz-plane, i.e. pitch rotations. This means that it is enough to use roll angle $\phi \in [\underline{\phi}, \overline{\phi}]$ and yaw angle $\psi \in [\underline{\psi}, \overline{\psi}]$, where $\underline{\cdot}, \overline{\cdot}$ denote a respective lower and upper bound. The parameterization of $\mathbf{R}_{mb}$ is

$$\mathbf{R}_{mb}(\phi,\, \psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\cos(\phi)\sin(\psi) & \cos(\phi)\cos(\psi) & \sin(\phi) \\ \sin(\phi)\sin(\psi) & -\sin(\phi)\cos(\psi) & \cos(\phi) \end{bmatrix}. \tag{7.15}$$

We then find the optimal values ϕ^*, ψ^* by solving the optimization problem

$$\phi^*, \psi^* \triangleq \arg\min_{\phi,\psi} J(\phi, \psi) \tag{7.16a}$$

$$\text{s.t.} \quad \underline{\phi} \le \phi \le \overline{\phi} \tag{7.16b}$$

$$\underline{\psi} \le \psi \le \overline{\psi} \tag{7.16c}$$

and denote the resulting rotation matrix as $\mathbf{R}^*_{mb}$ and the transformation as $\mathbf{T}^*$.

Remark 7.1. The outlined problem of finding a static rotation matrix to minimize the cost function Eq. (7.14) is similar to Wahba's problem [187] if Assumption 13 is satisfied. If one can show equivalence of the cost function given here to the loss function used in [187], it is possible to use Davenport's q-method [119] to arrive at an explicit solution to the optimization problem.

Results of the Symmetry Calibration

We use algorithmic differentiation provided by CasAdi [6] and the interior-point optimization implemented in interior point optimizer (Ipopt) [186] to solve the optimization problem defined by Eq. (7.16a) - Eq. (7.16c). The optimal solution to the calibration problem is given by $\phi^* = -0.90\,\text{deg}$ and $\psi^* = 1.58\,\text{deg}$, which is in the range that can not be corrected for via visual inspection of the operator conducting the wind tunnel experiment. The result for a sweep of the sideslip angle during which $\alpha = \delta_a = \delta_e = 0$ and $V_a = 21\,\text{m/s}$ is depicted in Fig. 7.3. Clearly, the measured forces and moments do not look symmetric with respect to the longitudinal plane, i.e. when $\beta = 0$. However, when plotted over the calibrated sideslip angle, most of the forces and moments look symmetric. The exception is the roll moment in the upper right plot. A possible cause for this can be that the airframe that was used in the experiments has asymmetries with respect to the xz-plane such that Assumption 12 is not satisfied. Another explanation is that the wind tunnel may be producing a vortex around its longitudinal axis that would

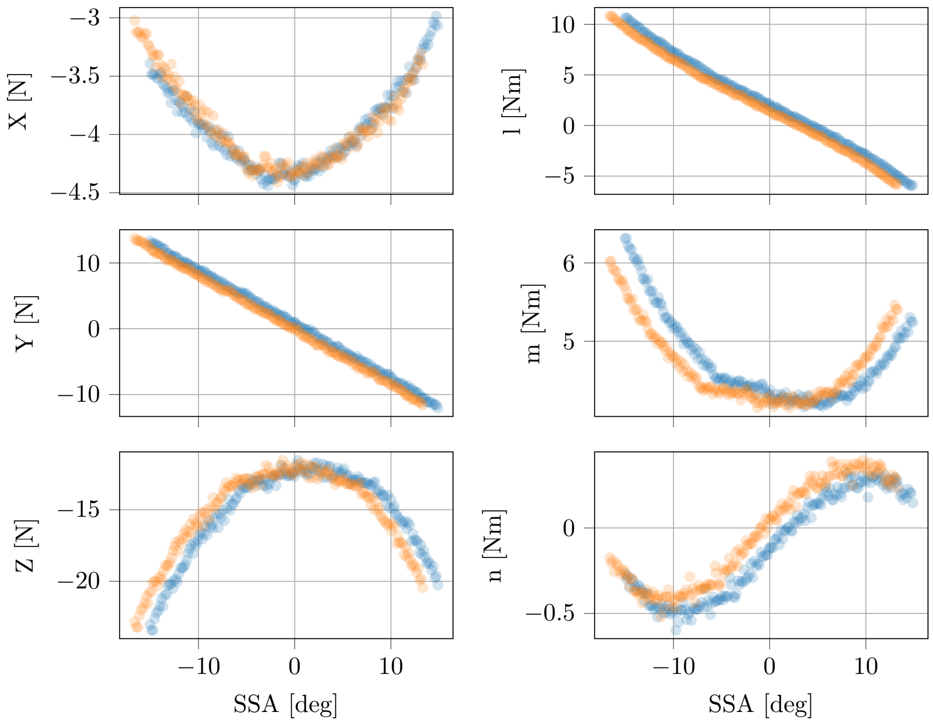

Figure 7.3: A sweep of the sideslip angle with $\alpha = \delta_a = \delta_e = 0$ and $V_a = 21\mathrm{m/s}$. The data over the measured sideslip angle (blue) and the data over the calibrated sideslip angle (orange) are plotted. The calibration angles are $\phi^* = -0.90\,\mathrm{deg}$ and $\psi^* = 1.58\,\mathrm{deg}$.

produce a non-zero relative angular velocity and therefore violate Assumption 14. In the latter case, the calibration could be extended to estimating the relative angular velocity

$$\boldsymbol{\omega}_r^w = \begin{bmatrix} p_r & 0 & 0 \end{bmatrix}^\top, \tag{7.17}$$

with $p_r \in \mathbb{R}$ as an additional decision variable in the optimization.

Another possibility is that the control surface deflections were not symmetric and therefore inducing an additional roll moment. The resulting aerodynamic model of the roll moment shows that an aileron deflection of less than 2.5 deg is enough to explain the offset of approximately 2 N m in Fig. 7.3. Mapped to the elevon deflection, this amounts to 1.25 deg on each side, which is within the hysteresis band for the consumer-grade servos that were used during the tests.

Now that the measurement from the wind tunnel are calibrated, we proceed with the identification of the aerodynamic model based on this dataset.

7.2.3 Dataset for Model Identification

The aerodynamic force coefficients were computed based on the measured body-fixed forces, airspeed and geometric factors. The body-fixed forces X, Y, $Z \in \mathbb{R}$

are transformed to the forces referred to as drag, crosswind and lift, denoted by D, C, $L \in \mathbb{R}$ as follows:

$$\mathbf{f}_a^w = \begin{bmatrix} D \\ C \\ L \end{bmatrix} = \mathbf{R}_{wb}(\alpha, \beta) \begin{bmatrix} -X \\ Y \\ -Z \end{bmatrix} . \tag{7.18}$$

The moments in the directions roll, pitch and yaw are denoted by l, n, $n \in \mathbb{R}$ and modeled in the body-fixed frame. The dimensionless coefficients are given by

$$\begin{aligned} C_D &= \frac{D}{\bar{q} S_{\text{wing}}}, \quad C_C = \frac{C}{\bar{q} S_{\text{wing}}}, \quad C_L = \frac{L}{\bar{q} S_{\text{wing}}} \\ C_l &= \frac{l}{\bar{q} S_{\text{wing}} b}, \quad C_m = \frac{m}{\bar{q} S_{\text{wing}} c}, \quad C_n = \frac{n}{\bar{q} S_{\text{wing}} b}, \end{aligned} \tag{7.19}$$

and plotted in Fig. 7.4 for their respective variations of the control surfaces and aerodynamic angles. Higher angles of attack measurements beyond those used for model identification are included in the figures to determine the linear region below the stall angle. We see that for identifying a linear model in C_L and C_m, the angle of attack measurements need to be restricted to angles below 12 deg, which is referred to as stall angle.

Having a look at the lateral coefficients, we can see that nonlinear effects are notable for sideslip angles that are either below -5 deg or above 5 deg. The nonlinearities appear negligible for roll and pitch moment, but are clearly visible for the yaw moment. Regarding zero aileron deflections, identifying the yaw moment coefficient would in the case of ideal measurements lead to a model that is invariant to sideslip angle variations. Moreover, the effect of aileron deflections on the yaw moment coefficient seems to be saturated for aileron deflections below -10 deg (or above 10 deg due to symmetry), which should be considered when selecting a subset of the data for identification of a model for C_n that is linear in δ_a. None of these effects are taken into account by Gryte et al. [71].

7.2.4 Model Identification

Model Structure

We modify the structure of the existing model with alterations to the drag coefficient. For now, we focus on a subset of the aerodynamic coefficients that can be identified based on wind tunnel experiments in which the airframe is statically mounted to a pan-tilt-unit. The dynamic coefficients that capture effects of the angular rate are therefore not included. The expressions for the identifiable aero-

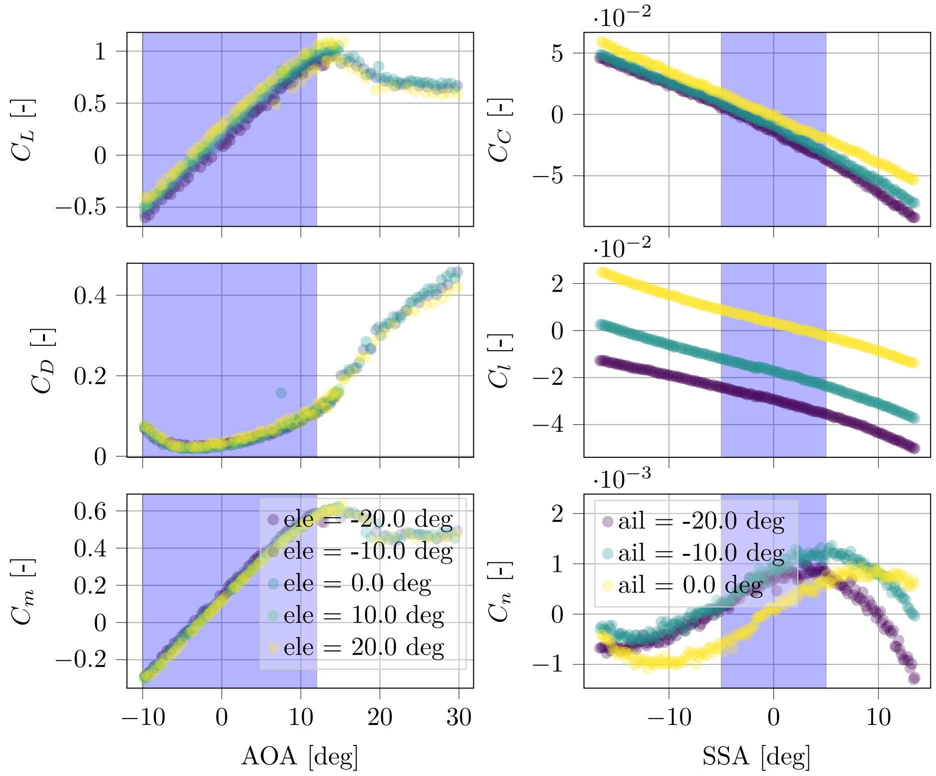

Figure 7.4: Measured aerodynamic coefficients for varying aerodynamic angles and surface deflections. The moment coefficients in the right column are with respect to the sensor position. The interval of the angle of attack $[-10\,\text{deg}, 12\,\text{deg}]$ and the sideslip angle $[-5\,\text{deg}, 5\,\text{deg}]$ (marked in blue) are suitable for identification of a linear model.

dynamic coefficients are given by

$$\begin{aligned}
C_D &= C_{D_0} + C_{D_\alpha}\alpha + C_{D_{\alpha\delta_e}}\alpha\delta_e + C_{D_{\alpha^2}}\alpha^2 \\
C_C &= C_{C_0} + C_{C_\beta}\beta + C_{C_{\delta_a}}\delta_a \\
C_L &= C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e \\
C_l &= C_{l_0} + C_{l_\beta}\beta + C_{l_{\delta_a}}\delta_a \\
C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta_e}}\delta_e \\
C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_{\delta_a}}\delta_a.
\end{aligned} \tag{7.20}$$

The model of the drag coefficient does not include the effects of sideslip, given that the sensitivity of the drag force to sideslip angle variations was not significant. We also replace the quadratic elevator term by a mixed term with the angle of attack, reflecting the fact that a negative/positive elevator deflection at a positive angle of attack actually decreases/increases the area of the UAV that is orthogonal to the air stream.

Table 7.1: Parameters of the wind tunnel model

C_{D_0}	0.023617	C_{m_0}	0.051656	C_{L_0}	0.058192
C_{D_α}	0.012051	C_{m_α}	2.409198	C_{L_α}	3.996278
$C_{D_{\alpha\delta_e}}$	0.075081	$C_{m_{\delta_e}}$	-0.043286	$C_{L_{\delta_e}}$	0.242942
$C_{D_{\alpha 2}}$	1.725632				
C_{l_0}	0.001839	C_{C_0}	-0.002544	C_{n_0}	0.000058
C_{l_β}	-0.064541	C_{C_β}	-0.23371	C_{n_β}	0.006828
$C_{l_{\delta_a}}$	0.094302	$C_{C_{\delta_a}}$	0.036065	$C_{n_{\delta_a}}$	-0.004462

Remark 7.2. Gryte et al. [71] also saw a notable drop in the coefficient of determination for the drag model when auto-validating it against the wind tunnel measurements. They however assumed misalignment of the airframe to be the issue and did not conclude that the aerodynamic coefficient is not significantly affected by sideslip angle variations.

Parameter Estimation

The model structure in Eq. (7.20) is well-suited for linear regression which can be used to identify models of the form

$$\mathbf{z} = \mathbf{X}\boldsymbol{\Theta} + \boldsymbol{\epsilon} \tag{7.21}$$

where $\mathbf{z} \in \mathbb{R}^N$ is a vector of N measurements, $\mathbf{X} \in \mathbb{R}^{N \times p}$ is the regressor matrix composed of the model terms, and $\boldsymbol{\Theta} \in \mathbb{R}^p$ is the vector of model parameters. The part of the measurements that are not explained by the model are captured by the residual $\boldsymbol{\epsilon} \in \mathbb{R}^N$. Gauss showed that a cost function composed of the sum of squares of the residuals

$$J(\boldsymbol{\Theta}) = \frac{1}{2}(\mathbf{z} - \mathbf{X}\boldsymbol{\Theta})^\top(\mathbf{z} - \mathbf{X}\boldsymbol{\Theta}), \tag{7.22}$$

is minimized by the solution

$$\hat{\boldsymbol{\Theta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{z}. \tag{7.23}$$

Now, for example, to find an estimate of the parameter vector for the drag coefficient model $\hat{\boldsymbol{\Theta}}$, we use the regressor matrix and measurement vector

$$\mathbf{X}_D = \begin{bmatrix} 1 & \alpha_i & \alpha_i\delta_{e_i} & \alpha_i^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_N & \alpha_N\delta_{e_N} & \alpha_N^2 \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} C_{D_i} \\ \vdots \\ C_{D_N} \end{bmatrix}. \tag{7.24}$$

The parameters of the other coefficients are identified in the same way by parameterizing regressor matrix and measurement vector according to the model structure in Eq. (7.20). A problem with ordinary least squares is that it will fit the given parameter vector to the measurements, regardless of how well its elements explain

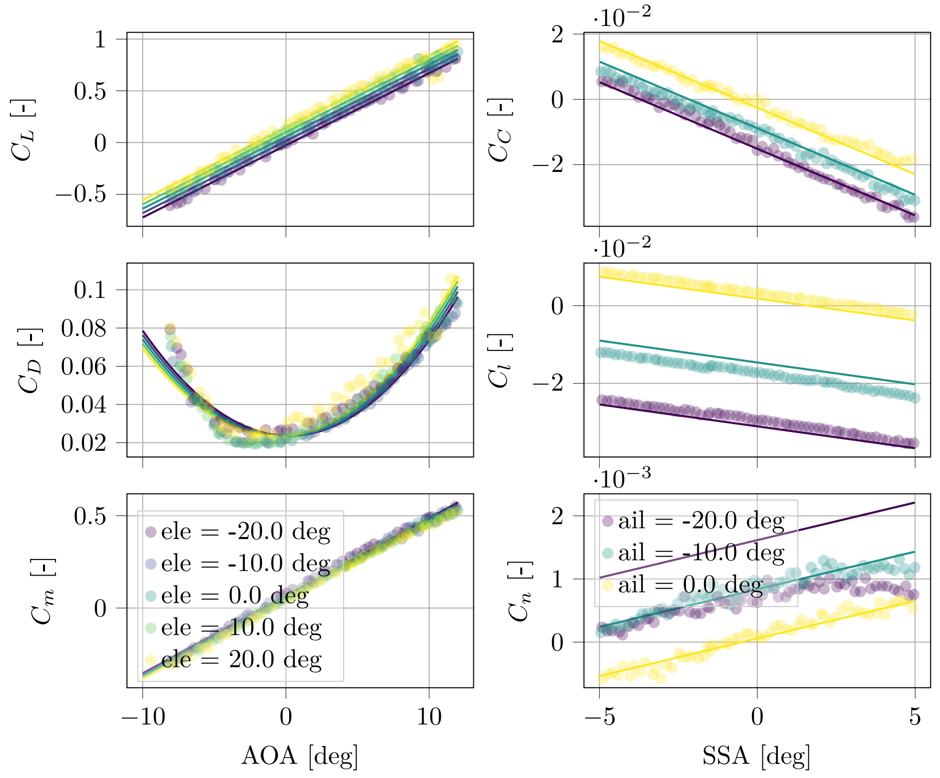

Figure 7.5: Aerodynamic models based on the wind tunnel data. The moment coefficients are with respect to the sensor position.

them, which requires a good confidence in the proposed model structure. Regarding the preceding discussion of the measurements, it is clear that a linear model is appropriate for all coefficients except the drag coefficient, which also requires quadratic terms. The resulting model parameters are summarized in Table 7.1 and plotted against the measurements in Fig. 7.5. The linear models correlate well with the measured data in general. The notable exception is the yaw moment coefficient where a model that is linear in the aileron deflection significantly differs from the measurements at higher deflections due to the saturation discussed previously. The drag coefficient model could be extended by polynomial terms of the angle of attack up to order four to better capture the data at negative angles. Negative angles of attack are usually not part of the nominal flight conditions, and we prioritize model simplicity over global accuracy in this case.

The coefficients C_l, C_m, C_n model the aerodynamic moments at the point where the sensor was located during the wind tunnel experiments. Let this point be referred to as *aerodynamic reference* point. This is in contrast to [71], who propose a model with respect to the center of mass of the airframe. The position of the point of the force measurements relative to the center of mass was manually adjusted until the equilibrium point of the pitch moment was shifted to an angle of attack that approximately matched the experience from flight tests. Instead of implicitly assuming a fixed offset, we identify the vector based on data collected in flight tests. The identification of this lever arm, together with a suitable model that augments the wind tunnel model with additional damping, is the subject of the next section.

7.3 Model Augmentation based on Flight Experiments

So far, we have found an aerodynamic model based on data from a wind tunnel. The wind tunnel model maps relative linear velocities and the surface deflections to aerodynamic forces and is denoted by $\mathbf{f}_{\text{wt}} : \mathbb{R}^5 \to \mathbb{R}^6$ which reads as

$$\begin{bmatrix} \mathbf{f}^w_{a,\text{wt}} \\ \mathbf{m}^b_{a,\text{wt}} \end{bmatrix} = \mathbf{f}_{\text{wt}}(\boldsymbol{\nu}^b_r, \delta_a, \delta_e) = \mathbf{f}_{\text{wt}}(\mathbf{v}^b_r, \delta_a, \delta_e) + \mathbf{f}_{\text{damp}}(\boldsymbol{\nu}^b_r). \tag{7.25}$$

The aim of this section is to use data collected during flight experiments to augment $\mathbf{f}_{\text{wt}}$ with a damping model that further considers the effect of angular velocities onto the generalized aerodynamic forces. We thus seek a function $\mathbf{f}_{\text{damp}} : \mathbb{R}^6 \to \mathbb{R}^6$, and the final augmented model $\mathbf{f}_{\text{wt,aug}} : \mathbb{R}^8 \to \mathbb{R}^6$ such that the modeled generalized aerodynamic forces are given by

$$\begin{bmatrix} \mathbf{f}^w_{a,\text{wt,aug}} \\ \mathbf{m}^b_{a,\text{wt,aug}} \end{bmatrix} = \mathbf{f}_{\text{wt,aug}}(\boldsymbol{\nu}^b_r, \delta_a, \delta_e) = \mathbf{f}_{\text{wt}}(\mathbf{v}^b_r, \delta_a, \delta_e) + \mathbf{f}_{\text{damp}}(\boldsymbol{\nu}^b_r). \tag{7.26}$$

We will further identify a separate model that is entirely based on the collected flight data and includes effects of the throttle δ_t. Let this model be denoted by $\mathbf{f}_{\text{flight}} : \mathbb{R}^9 \to \mathbb{R}^6$, and it's resulting forces be given by

$$\begin{bmatrix} \mathbf{f}^w_{a,\text{flight}} \\ \mathbf{m}^b_{a,\text{flight}} \end{bmatrix} = \mathbf{f}_{\text{flight}}(\boldsymbol{\nu}^b_r, \boldsymbol{\delta}), \quad \text{with } \boldsymbol{\delta} = \begin{bmatrix} \delta_a & \delta_e & \delta_t \end{bmatrix}^\top. \tag{7.27}$$

A brief outline of the flight experiments to identify the new models reads as follows: Starting from wings-level horizontal flight, we induced oscillations of the control surfaces as a chirp signal with frequency range from 4 Hz to 8 Hz and minimum and maximum set to −5 deg and 5 deg, respectively. We moreover used step signals that were added as doublets to the elevators and in the form of 1-2-1 signals to the ailerons as depicted in Fig. 7.6. The relative velocities from Ardupilot's wind velocity observer, IMU data and actuator set-points are recorded and used for the following identification procedure.

We assume that a model of the propulsion is available that maps airspeed and throttle set-point to a propulsion force and let the result be denoted by $\mathbf{f}^b_{\text{t,prop}} \in \mathbb{R}^3$. For example the propulsion force model identified in [39] models the propulsion force vector $\mathbf{f}^b_{\text{t,prop}} = \begin{bmatrix} T & 0 & 0 \end{bmatrix}^\top$ with

$$\begin{aligned} C_T &= \delta_t(V_a + \delta_t(k_m - V_a))(k_m - V_a) \\ T &= \rho S_{\text{prop}} \eta_{\text{prop}} C_T. \end{aligned} \tag{7.28}$$

The propeller parameters are $S_{\text{prop}} = 0.108$, $\eta_{\text{prop}} = 0.248$ and the engine parameter is $k_m = 37.42$. For more details on the identified thrust model, see [39]. A last

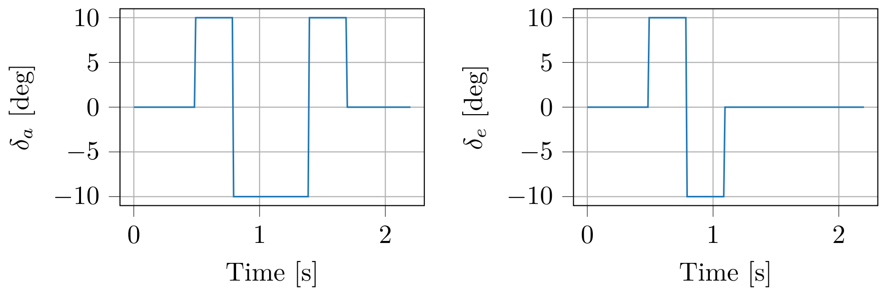

Figure 7.6: Step inputs used in the flight experiments for model identification and evaluation.

assumption on existing models is knowledge of the moment of inertia $\mathbf{J}$, which has been identified based on bifilar pendulum tests and is given by

$$\mathbf{J} = \begin{bmatrix} 0.335 & 0 & -0.029 \\ 0 & 0.14 & 0 \\ -0.029 & 0 & 0.40 \end{bmatrix} \tag{7.29}$$

The observed generalized aerodynamic forces are then computed as

$$\mathbf{f}^w_{a,z} = m\mathbf{R}_{wb}\mathbf{a}^b_{nb} - \mathbf{f}^b_{\text{t,prop}} \tag{7.30}$$

$$\mathbf{m}^b_{a,z} = \mathbf{J}^b\dot{\boldsymbol{\omega}}^b_{nb} - \mathbf{J}^b\boldsymbol{\omega}^b_{nb} \times \boldsymbol{\omega}^b_{nb}, \tag{7.31}$$

where $\mathbf{a}^b_{nb} \in \mathbb{R}^3$ denotes the linear acceleration measured by the IMU and m denotes the mass of the UAV. The observed angular acceleration $\dot{\boldsymbol{\omega}}^b_{nb} \in \mathbb{R}^3$ is computed using the centered difference formula based on the estimated angular velocity.

In the following section we will first discuss how to find a lever arm between the center of mass and the aerodynamic reference point, i.e. the position of the measurement equipment in the wind tunnel tests. The lever arm will be used to update the moments of the wind tunnel model. After that follows a discussion on how to find the damping model, $\mathbf{f}_{\text{damp}}$, and the model that is fully identified based on flight data, $\mathbf{f}_{\text{flight}}$.

7.3.1 Distance from the Aerodynamic Reference Point to the Center of Mass

Let the coefficients that model the aerodynamic moment at the aerodynamic reference point used in the wind tunnel be denoted by $C_{\{l,m,n\},ar}$ and the values at the center of mass be denoted by $C_{\{l,m,n\},cm}$. Their difference is determined by the lever arm $\mathbf{r}_{ar,cm}$ and the aerodynamic force coefficients as

$$\begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix}_{cm} = \begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix}_{ar} - \text{diag}(b,c,b)^{-1}\mathbf{S}\left(\mathbf{R}_{wb}(\alpha,\beta)^\top \begin{bmatrix} -C_D \\ C_C \\ -C_L \end{bmatrix}\right)\mathbf{r}_{ar,cm}, \tag{7.32}$$

which follows from simple mechanics of a force-inducing moment given by the cross-product of the lever arm and the force vector.

We can again use linear regression to find $\mathbf{r}_{ar,cm}$. Given N measurements, let for each sample i be

$$\mathbf{X}_i = -\operatorname{diag}(b,c,b)^{-1}\mathbf{S}\left(\mathbf{R}_{wb}(\alpha,\beta)^\top \begin{bmatrix} -C_D \\ C_C \\ -C_L \end{bmatrix}\right), \quad \mathbf{z}_i = \begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix}_z - \begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix}_{ar} \tag{7.33}$$

which can be vertically concatenated to give the regressor matrix $\mathbf{X} \in \mathbb{R}^{3N\times 3}$ and measurement vector $\mathbf{z} \in \mathbb{R}^{3N}$. Using the measured data gives the result

$$\mathbf{r}_{ar,cm} = \begin{bmatrix} -0.226 & -0.02 & 0.144 \end{bmatrix}, \tag{7.34}$$

which is in line with the vector found in [71].

Remark 7.3. Considering the dimensions of the Skywalker X8, an offset of 0.144 m in the z-direction may seem too high. However, keep in mind that this reflects the different orientations of the sensor frames of the force balance used in the wind tunnel experiments and the frame of the IMU in the flight experiments.

7.3.2 Aerodynamic Models from Flight Observations

In this subsection, we will first give a brief outline of the method that we use to identify the models $\mathbf{f}_{\text{damp}}$ and $\mathbf{f}_{\text{flight}}$ before briefly showing how to apply it.

Sparse Identification of Nonlinear Dynamics

Ordinary least squares that we used up until now has the drawback that it gives a solution that fits the proposed parameter vector to the measurements, even if a subset of the parameters actually does not explain the measurements in a meaningful way. There are several methods that address this problem by introducing an additional penalty on the size of the coefficients. A prominent method is *Ridge Regression* using an additional l_2 regularization. Other methods such as *Lasso* or *Elastic Net* fit sparse models by including l_1 or l_0 regularization sparse models with fewer terms. In this direction, Brunton et al. [16] developed sparse identification of nonlinear dynamics (SINDy).

Since its first publication in 2016, SINDy received a lot of attention and has recently been made available as an open-source toolbox implemented in Python [44]. The idea is to use a library of symbolic functions such as polynomials and optimize over the induced function space to have a linear combination of nonlinear functions describe the dynamical system that best fits the collected measurement data. The underlying assumption is that the dynamics have a sparse representation in the function space that is described by the library. We use SINDy to find an expression for the damping forces of the airframe, and give a brief outline on how it works.

Assume an autonomous dynamic system that is not affected by external disturbances. Its dynamics can be described by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{7.35}$$

for which the function $\mathbf{f}$ is to be determined from data of the state $\mathbf{x}(t)$ or its derivative $\dot{\mathbf{x}}(t)$. The data at sampling times $t_1, t_2, \dots, t_m$ is arranged into matrices $\mathbf{X}, \dot{\mathbf{X}} \in \mathbb{R}^{m \times n_x}$ given as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^\top(t_1) \\ \mathbf{x}^\top(t_2) \\ \vdots \\ \mathbf{x}^\top(t_m) \end{bmatrix}, \qquad \dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^\top(t_1) \\ \dot{\mathbf{x}}^\top(t_2) \\ \vdots \\ \dot{\mathbf{x}}^\top(t_m) \end{bmatrix}. \tag{7.36}$$

Then a function library $\mathbf{\Theta}(\mathbf{X})$ consisting of nonlinear candidate functions based on $\mathbf{X}$ is used, for example

$$\mathbf{\Theta}(\mathbf{X}) = \begin{bmatrix} 1 & \mathbf{X} & \mathbf{X}^{P_2} \end{bmatrix} \tag{7.37}$$

where the matrix $\mathbf{X}^{P_2}$ includes quadratic nonlinearities:

$$\mathbf{X}^{P_2} = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \dots & x_2^2(t_1) & \dots & x_{n_x}^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \dots & x_2^2(t_2) & \dots & x_{n_x}^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1^2(t_m) & x_1(t_m)x_2(t_m) & \dots & x_2^2(t_m) & \dots & x_{n_x}^2(t_m) \end{bmatrix}. \tag{7.38}$$

The coefficients $\mathbf{\Xi} = [\boldsymbol{\xi}_1 \; \boldsymbol{\xi}_2 \; \dots \; \boldsymbol{\xi}_{n_x}]$ are then used to set up a sparse regression problem to select the nonlinear candidate functions that model the time-derivatives $\dot{\mathbf{X}}$ through a linear combination of the features that are included in the function library

$$\dot{\mathbf{X}} = \mathbf{\Xi}\mathbf{\Theta}(\mathbf{X})^\top. \tag{7.39}$$

The minimization problem seeks the coefficients according to the objective

$$\boldsymbol{\xi}_k = \arg\min_{\boldsymbol{\xi}_k} \|\dot{\mathbf{X}}_k - \boldsymbol{\xi}_k \mathbf{\Theta}(\mathbf{X})^\top\|_2 + \alpha \|\boldsymbol{\xi}_k\|_1 \tag{7.40}$$

where the $\|\cdot\|_1$ promotes sparsity in the function space. A trade-off between low model complexity and sufficient accuracy can be made by tuning the parameter α. The result of the sparse symbolic regression, which is implemented in [44], is the coefficient vector $\mathbf{\Xi}$ from which we can construct the governing equations as

$$\dot{x}_k = f_k(\mathbf{x}) \approx \mathbf{\Theta}(\mathbf{x}^\top)\boldsymbol{\xi}_k \tag{7.41}$$

with $\mathbf{\Theta}(\mathbf{x})$ denoting the vector of symbolic functions that were proposed as candidates.

Damping Model Augmentation to the Wind Tunnel Model

Now, to find expressions for the damping terms to augment the wind tunnel model, we use the error vectors of the generalized aerodynamic forces. Let the values obtained by the wind tunnel model be denoted by $\mathbf{f}^w_{a,\mathrm{wt}}$, $\mathbf{m}^b_{a,\mathrm{wt}}$. The goal is to find a damping model $\mathbf{f}_{\mathrm{damp}}$ that minimizes a residual $\boldsymbol{\epsilon}$ of the difference between the wind tunnel model and the observations $\mathbf{f}^w_{a,z}$, $\mathbf{m}^b_{a,z}$, which can be formulated as

$$\begin{bmatrix} \mathbf{f}^w_{a,z} - \mathbf{f}^w_{a,\mathrm{wt}} \\ \mathbf{m}^b_{a,z} - \mathbf{m}^b_{a,\mathrm{wt}} \end{bmatrix} = \mathbf{f}_{\mathrm{damp}}(\mathbf{x}) + \boldsymbol{\epsilon}, \quad \text{with } \mathbf{x} = \boldsymbol{\nu}^b_r. \tag{7.42}$$

Table 7.2: The coefficient matrix Ξ for the damping augmentation to the wind tunnel model. The coefficients that are 0 are structural zeros and not rounded.

	dX	dY	dZ	dl	dm	dn
1	-15.01	-2.88	-398.74	3.22	0.29	0
u	0.52	0.21	48.51	-0.21	0	0
v	0	6.89	18.54	0.66	0	0
w	8.73	2.02	-41.30	0	-0.24	0
p	2.70	2.07	-8.62	-0.67	0	0
q	-5.60	-0.26	50.31	0.16	0.50	0
r	-4.89	-26.10	21.09	32.12	-0.12	-0.18
u^2	0	0	-1.43	0	0	0
uv	0	-0.33	-1.06	0	0	0
uw	-0.48	-0.11	3.04	0	0	0
up	-0.16	-0.14	0.50	0	0	0
uq	0.28	0	-3.30	0	0	0
ur	0.23	1.86	-1.16	-2.16	0	0
v^2	0.16	0	-2.03	0	0	0
vw	0	0.16	-0.72	0	0	0
vp	0	0	-0.17	0	0	0
vq	0	-0.15	0.56	0	0	0
vr	0.85	-0.20	-7.35	0	0	0
w^2	-0.41	0	0.30	0	-0.15	0
wp	0	0.44	0	0.44	0	-0.17
wq	0	0	0.44	0	0.24	0
wr	0.13	0	0.84	0.87	0	0
p^2	-0.26	0	-0.57	0	0	0
pq	0.21	-0.95	-0.48	-0.14	0	0
pr	-0.26	-0.23	-4.86	0.36	-0.66	0.39
q^2	0.86	0	-0.70	0	-0.23	0
qr	-0.53	-0.37	-0.46	-2.52	0	0
r^2	2.77	-0.76	-43.44	-0.49	-0.56	0

The variables to explain the errors are the relative velocities $\boldsymbol{\nu}_r^b$, which build the polynomial function library in the form of Eq. (7.37). We used six sequences of oscillating input disturbances to the aileron and elevator over a duration of 15 s as training data. Finding the error models with the STLSQ as optimizer and $\alpha = 0.05$ with the threshold set to 0.1 gives the coefficient matrix Ξ shown in Table 7.2. The coefficient vectors for the rotational damping model have a desirable sparse structure which shows that a simple model is sufficient to explain the difference between the observed moment and the linear model from the wind tunnel. The coefficient vectors for the force error model are less sparse, which indicates that a function library based on second order polynomials of the relative velocities is not sufficient to capture the difference between the wind tunnel model and the observations. For instance angle of attack, sideslip angle, airspeed or their products are not included.

The extension of the feature library to rational numbers of the variables, trigonometric functions or fractional exponents is straight forward and may give models that are more accurate and sparse.

Remark 7.4. The training data can also be used to find the aerodynamic damping coefficients that that were used in [71] by means of ordinary least squares. However, this leads to a degrading fitness compared to the raw wind tunnel model where damping is neglected.

Full model of the aerodynamic forces

A model that is identified based on the available flight data can be used for comparison. We seek a model $\mathbf{f}_{\text{flight}}$ for the generalized aerodynamic forces as a function of the relative velocities and the actuators $\boldsymbol{\delta} = \begin{bmatrix}\delta_a & \delta_e & \delta_t\end{bmatrix}^\top$. This can be formulated as

$$\begin{bmatrix}\mathbf{f}_{a,z}^{w} \\ \mathbf{m}_{a,z}^{b}\end{bmatrix} = \mathbf{f}_{\text{flight}}(\mathbf{x}) + \boldsymbol{\epsilon} \quad \text{with } \mathbf{x} = \begin{bmatrix}\boldsymbol{\nu}_r^b \\ \boldsymbol{\delta}\end{bmatrix}. \tag{7.43}$$

We use the same training data and parameterization of the optimizer as for the damping model. The coefficients of $\mathbf{f}_{\text{flight}}$ are given in Table 7.3. Note again the sparsity of the resulting coefficient vectors, indicating a good choice of the function library. A problem that shows in this model is that the data for identification includes mostly constant throttle so that there is little information on how it affects the dynamics. The result of this is the high magnitudes of the coefficients related to the terms that include δ_t. A more rigorous test design would be needed to accurately model the effect of the throttle. During this test campaign, however, the primary purpose was to find the rotational damping coefficients to augment the wind tunnel model. A more thorough identification of a model that is entirely based on data collect in flight is part of future work.

7.4 Flight Results and Discussion

The models are compared against the baseline model presented in [71] in test sequences including chirp signals and step perturbations to elevator and aileron. The modeled aerodynamic forces for the respective disturbances to the aileron and elevator are depicted in Fig. 7.7 and Fig. 7.8. A set of different chirp signals was used to identify the damping model and the full aerodynamic model based on flight data, and no step perturbations were used in the identification process.

We use the coefficient of determination (R^2) as a metric to evaluate the model fitness in terms of the generalized aerodynamic forces and the resulting accelerations. The results are summarized in Table 7.4. They indicate that our model obtained from the wind tunnel without damping already outperforms the baseline model in all forces except for the side force Y. However, the lateral acceleration $\dot{v}$ shows that the aerodynamic force in this direction is not significant compared to the coriolis term. The damping augmentation further improves the wind tunnel model except for the roll moment l and the resulting acceleration $\dot{p}$. This suggests an overfitting of the roll damping to the training data.

Table 7.3: The coefficient matrix Ξ for the generalized forces identified with flight data.

	X	Y	Z	l	m	n
1	27.72	-5.94	-200.19	-8.80	3.19	-1.04
u	-0.97	0.41	23.41	0.52	-0.22	0
v	2.49	-1.81	-15.41	0.20	0	0.50
w	11.27	0	-23.38	-0.64	-0.57	0
p	4.96	3.54	-16.26	0	0	0
q	-10.49	0.23	56.32	0.56	0.86	0
r	20.82	-9.70	-77.45	-5.75	-5.21	-0.52
δ_a	0	0	0.10	0	0	0
δ_e	0	0	0	0	-0.10	0
δ_t	-118.23	17.33	14.27	22.95	-3.23	3.29
u^2	0	0	-0.89	0	0	0
uv	-0.11	0	0.87	0	0	0
uw	-0.66	0	0.98	0	0	0
up	-0.36	-0.17	1.18	0	0	0
uq	0.67	0	-4.15	0	0	0
ur	-1.53	0.76	5.85	0.79	0.31	0
$u\delta_a$	0	0	0	0	0	0
$u\delta_e$	0	0	-0.12	0	0	0
$u\delta_t$	2.20	-0.93	7.05	-1.12	0.37	0
v^2	0	0	0	0	0	0
vw	0.18	0	-0.78	0	0	0
vp	0	0	0	0	0	0
vq	0	0	0.47	-0.10	0	0
vr	-0.39	-0.29	-0.69	0	0	0
$v\delta_a$	0	0	0	0	0	0
$v\delta_e$	0	0	0	0	0	0
$v\delta_t$	-1.40	1.24	3.31	-1.23	0	-0.45
w^2	0.47	0	-0.36	0	0	0
wp	0	0.21	0	0.26	0	0
wq	-0.66	0	1.21	0	0.12	0
wr	2.41	0	-6.73	0.72	0	0
$w\delta_a$	0	0	0	0	0	0
$w\delta_e$	0	0	0	0	0	0
$w\delta_t$	2.02	-0.31	-3.06	0.79	0.28	0
p^2	0	0	-1.36	0	0	0
pq	0	-0.39	0	0	0	0
pr	0.16	0	-3.12	-0.41	-0.62	0
$p\delta_a$	0	0	0	0	0	0
$p\delta_e$	0	0.10	0	0	0	0
$p\delta_t$	2.16	0	-6.49	-0.93	0	0
q^2	0.96	0	-2.36	0	-0.23	0
qr	-2.98	-0.49	4.33	-1.84	0	-0.30
$q\delta_a$	0	0	0	0	0	0
$q\delta_e$	0	0	0	0	0	0
$q\delta_t$	-3.78	0	17.89	-0.42	-0.64	0
r^2	2.61	-2.61	5.17	-2.45	0	0.45
$r\delta_a$	0	0	-0.75	0	0	0
$r\delta_e$	-0.18	0	0.37	0.12	0	0
$r\delta_t$	2.41	-5.04	-19.24	-12.51	0	-0.71
δ_a^2	0	0	0	0	0	0
$\delta_a\delta_e$	0	0	0	0	0	0
$\delta_a\delta_t$	0	0.34	0	0.69	0	0
δ_e^2	0	0	0	0	0	0
$\delta_e\delta_t$	0	0.14	-1.19	0	-0.12	0
δ_t^2	80.40	-1.86	-106.05	-4.67	-2.28	-2.86

Table 7.4: The R^2 scores for the entire flight, including the modeled aerodynamic forces and the resulting accelerations.

	X	Y	Z	l	m	n	$\dot{u}$	$\dot{v}$	$\dot{w}$	$\dot{p}$	$\dot{q}$	$\dot{r}$
baseline	-3	-0.27	-1.5	-3.1	-1.3	-0.63	-1.3	0.93	-0.19	-3.1	-1.3	-0.62
wt	-1.6	-0.5	-1	0.093	-0.096	0.21	-0.52	0.91	0.036	0.089	-0.096	0.036
wt,aug	0.68	0.12	0.63	-0.76	0.7	0.37	0.69	0.94	0.82	-0.77	0.7	0.26
flight	0.71	0.24	0.81	0.6	0.72	-0.0047	0.65	0.95	0.91	0.6	0.72	-0.26

Another possible explanation is the feedback controller in the loop that will introduce disturbances into the observations of the open-loop damping.

A more rigorous test campaign in which the UAV is operated in open-loop for the collection of the training data can mitigate this problem. Upon inspection of the trajectories depicted in Fig. 7.7 and Fig. 7.8, we see that the roll model from the wind tunnel is already capturing the most important transients of the observed roll moment, which should be sufficient for model-based control. Similar arguments can be made for the yaw moment n, which in contrast to the roll moment, is improved by the damping augmentation. The most improvements to the wind tunnel model are in the pitch moment m, for which we see drastic improvements by adding a small set of additional terms, as can be seen in Table 7.2.

Regarding the modeled forces X, Y, Z, the R^2 scores and the depicted trajectories show that the wind tunnel model without damping augmentation is a better fit than the baseline model. However, both models appear to capture all relevant transients and thus differ from the observations by a slowly time-varying offset. The variance of this offset is further reduced for the trajectories of the accelerations, which are not shown here. Simple forms of integral action are well-suited to compensate for this type of model mismatch, such that the additional complexity introduced by the damping augmentation is not necessary for model-based control.

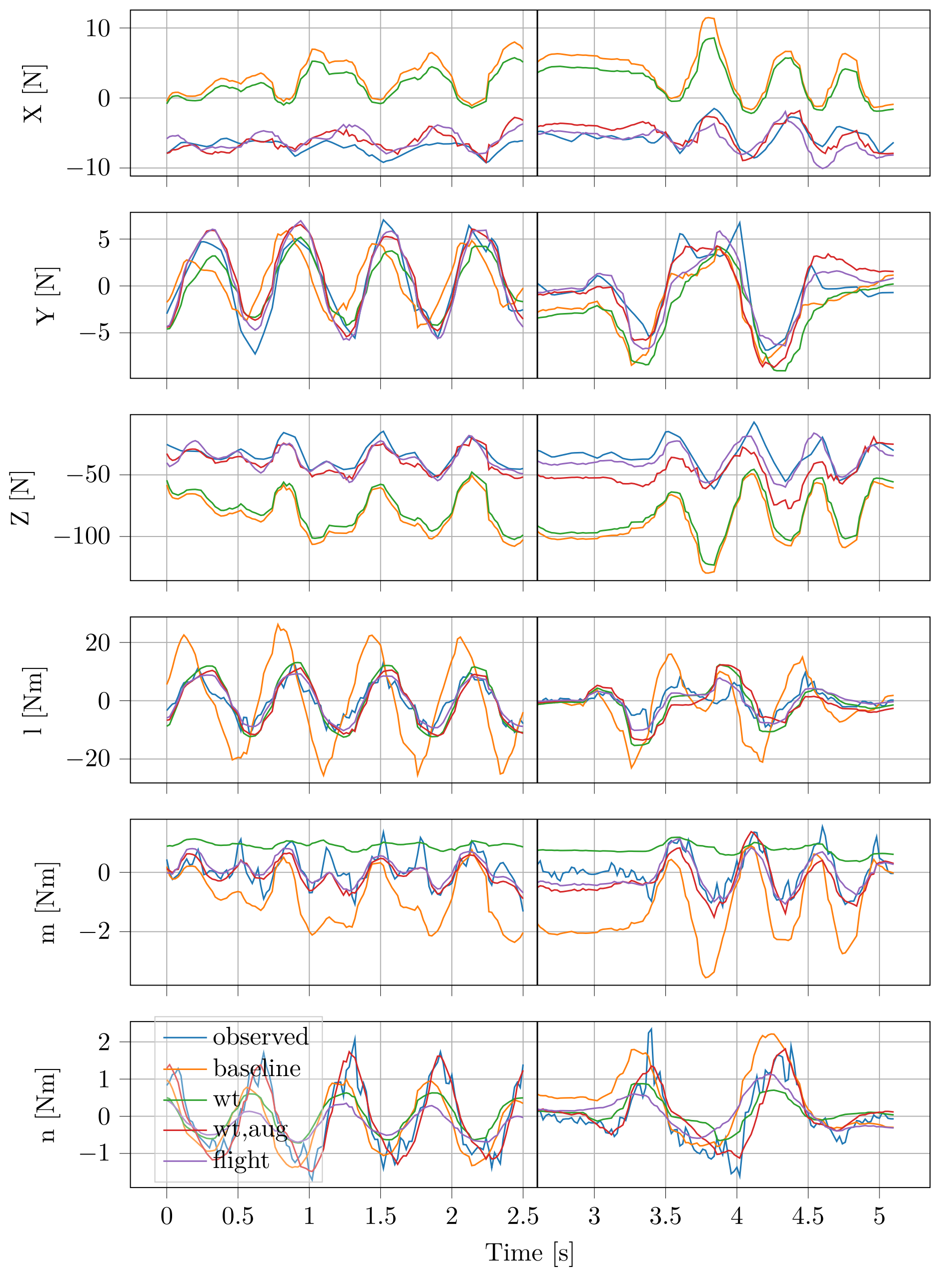

Figure 7.7: Forces for aileron disturbances. The chirp disturbance was active during the first half and the step disturbance during the second half.

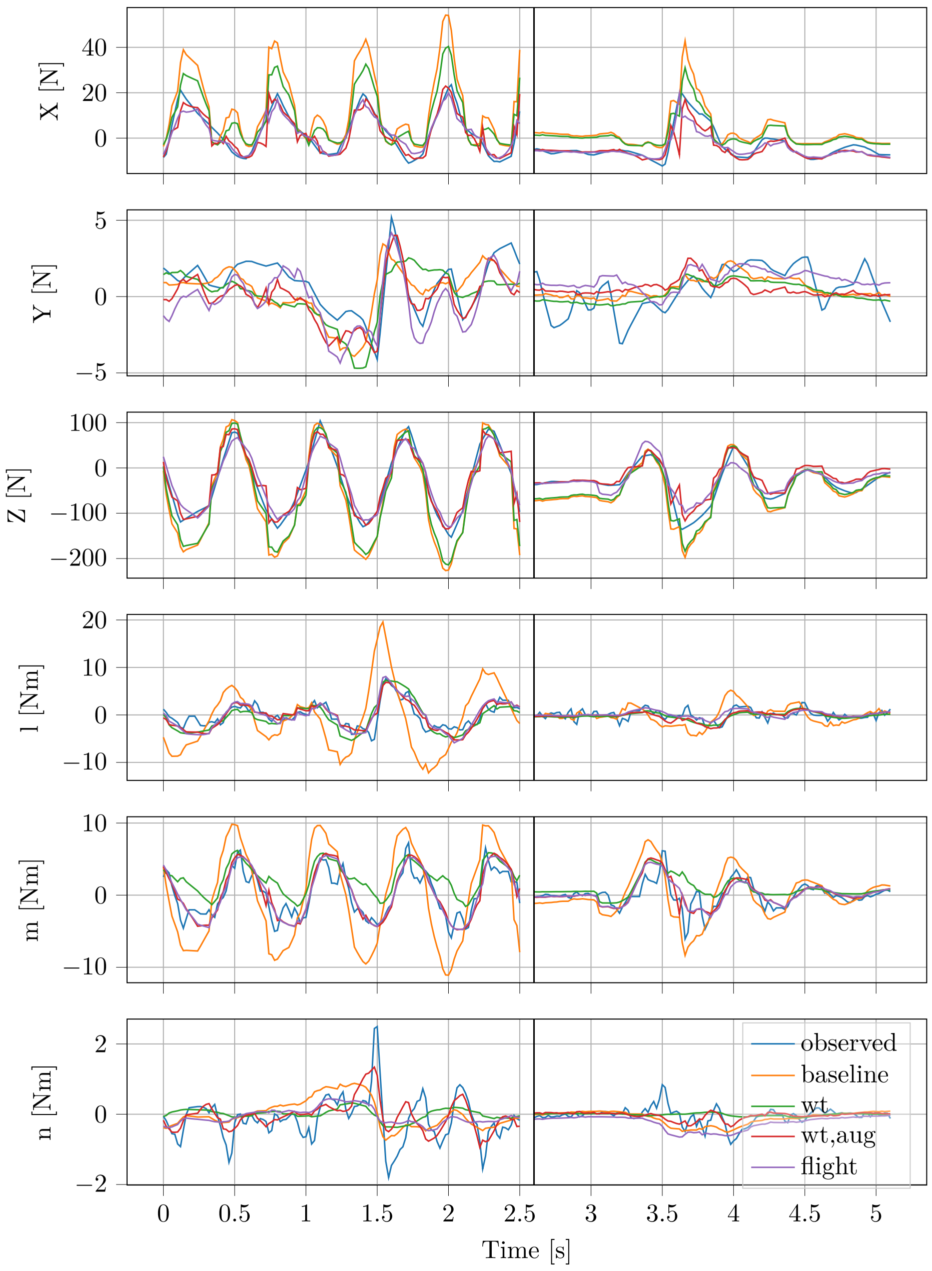

Figure 7.8: Forces for elevator disturbances. The chirp disturbance was active during the first half and the step disturbance during the second half.

7.5 Chapter Summary

In this chapter, we looked at the shortcomings of the existing model that we used for the dynamic model of the NMPC in Chapter 5. A symmetry calibration procedure that finds a static transformation for the wind tunnel data and a more careful consideration of the subset of data in which the aerodynamic coefficients are linear helped to improve the model compared to the baseline model [71].

Additional damping augmentation using SINDy further lifted the quality of the model which we demonstrated using flight observations. We also used SINDy to identify a model that is entirely based on in-flight data collections, which does not require access to a wind tunnel and therefore helps to significantly reduce cost and effort.

We discussed the implications of the additional complexity in light of the model mismatch for model-based control. However, a more rigorous design of the test campaign may improve the model quality, in particular larger variations to the throttle and open-loop training sequences for the control surfaces are needed.

Chapter 8

Direct Nonlinear Model Predictive Control for the Path-Following Control Problem

The content of this chapter includes preliminary results and is not part of a conference or journal publication.

8.1 Introduction

The controller designs of the preceding chapters focus on low-level motion control. Critical components such as the dynamic model and the computing platform were the respective subject of Chapter 7 and Chapter 3. This chapter aims to widen the scope to guidance control and modify the presented NMPC of Chapter 5 to a path-following controller for which we presented successful experiments using the controller with the developed model and experimental platform.

This motivates a modification to a path-following NMPC to tackle the guidance control problem. Only moderate alterations to the system dynamics and cost function are required, and we follow ideas based on the general discussions by Faulwasser and Findeisen [54]. We will look into the controller design and a numerical example demonstrating the approach. The chapter starts with a brief review of existing guidance algorithms for small fixed-wing UAVs and AUVs to provide the necessary context for the contribution of this chapter.

The path-following problem is well-studied and different approaches to steer an underactuated vehicle to planar or three-dimensional Euclidean paths exist in the literature. Sujit et al. [178] published a survey paper comparing established algorithms for fixed-wing UAVs. They include the standard methods in off-the-shelf autopilot implementations based on the L1 guidance law proposed by Park et al. [147] and TECS by Lambregts [101]. An approach for a general class of underactuated vehicles is the Lyapunov-based nonlinear design by Aguiar and Hespanha [2], which guarantees global convergence to an arbitrarily small neighborhood of the path.

Focusing on small UAVs, Nelson et al. [137] discuss a Lyapunov-based design that uses vector fields for straight-line and orbital path-following. Another interesting discussion on path-following control based on vector fields in arbitrarily strong wind fields is given by Furieri et al. [59]. Capello et al. [23] propose a waypoint-based guidance algorithm with outer-loop PID control that does not require a model. Cichella et al. [34] present a geometric approach and provide experiments that demonstrate the ability of the controller to follow position and speed profiles independently. Kai et al. [87] discuss a more model-based unified approach using parallel transport frames. All these approaches have proven stability properties and have a computational footprint that is small enough to implement them on low-cost hardware.

However, a significant part of the controllers are limited to simple path geometries such as straight lines or orbits and do not address constraint satisfaction or optimal performance, which NMPC is well-suited to handle. The use of NMPC for path-following control based on control-augmented dynamics with an off-the-shelf autopilot in the loop for low-level motion control is discussed by Stastny and Siegwart [174]. The topic of constraint output path-following using MPC is generally addressed by Faulwasser and Findeisen [54]. They propose to augment the system dynamics with an additional integrator chain that determines the motion of the reference along a given output path.

Applications that use this idea for the guidance of underactuated mobile robots include the work by Alessandretti and Aguiar [3] for following planar curves and the more recent approach by Yang et al. [194]. Dauer et al. [43] published similar work targeting helicopters where they use dynamic model inversion for low-level motion control. An application for airborne wind energy systems is presented by Diwale et al. [47] with a stronger emphasis on theoretic stability properties.

The reviewed path-following control methods, including the control-augmented NMPC, are guidance controllers based on kinematic models. They assume a modular architecture with sufficient bandwidth separation to generate reference signals that low-level motion controllers can track. The contribution of this chapter is an integrated NMPC design to follow defined position and speed profiles with direct actuator access and without relying on separate control loops.

The chapter is structures as follows. The next section gives a brief description of the path-following control objective before we show how cubic Bézier curves can be used to generate a set of successive path segments that can be generated from a list of waypoints that is available in most GNC architectures. The section then continues with the NMPC design that stabilizes the position error and airspeed, with references to the previous discussion on NMPC to keep the presentation short. The next section then demonstrates the performance of the controller in a numerical example following a periodic Lissajous curve, before the chapter ends with a discussion on the results and future work toward an experimental verification.

8.2 Control Objective

In conventional operations of aerial robotics applications, the output path is usually a parametric geometric curve in three-dimensional Euclidean space on which the reference position, here denoted by $\mathbf{p}^n_{nb,\text{ref}}$, evolves. Let the path $\mathcal{P}$ be parameterized by the path variable $\gamma \in [\underline{\gamma},\ \overline{\gamma}]$ as

$$\mathcal{P} = \left\{ \mathbf{p}^n_{nb,\mathrm{ref}} \in \mathbb{R}^3 | \gamma \in [\underline{\gamma}, \overline{\gamma}] \rightarrow \mathbf{p}^n_{nb,\mathrm{ref}} = \mathbf{p}(\gamma) \right\}. \tag{8.1}$$

Under the system dynamics outlined in Chapter 2 the objective is to achieve convergence of the position to the path, i.e.

$$\lim_{t\to\infty} \left\| \mathbf{p}^n_{nb}\left(\mathbf{x}(t)\right) - \mathbf{p}^n_{nb,\mathrm{ref}}\left(\gamma(t)\right) \right\| = 0, \tag{8.2}$$

while satisfying safety-related constraints imposed by aerodynamics and actuator limits. Instead of designing the NMPC to track low-level attitude references the objective end-to-end control from the path definition to the actuator set-points. This means that we aim for a controller design that achieves guidance and low-level motion control in one integrated approach instead of the timescale separated modular approach described in Chapter 2. A suitable NMPC can be designed based on the framework of constrained output path-following MPC proposed by [54] in which the controller determines the evolution of a reference on the path as part of the OCP by means of additional ordinary differential equations (ODEs) referred to as *timing law*. The resulting control algorithm will be discussed in Section 8.3.

8.2.1 Reference Path using Cubic Bézier Curves

A flexible path parameterization are cubic Bézier curves, which can be generated by a set of waypoints, which we briefly discuss now. Given a set of control knots $[\mathbf{P}_0, \dots, \mathbf{P}_3]$ a cubic Bézier curve $\mathbf{F}$ is defined as

$$\mathbf{F}(s) = (1-s)^3\mathbf{P}_0 + 3(1-s)^2 s\mathbf{P}_1 + 3(1-s)s^2\mathbf{P}_2 + s^3\mathbf{P}_3, \quad s \in [0,1], \tag{8.3}$$

which can be used to parameterize path segments for $s = \gamma + 1$. We use this for both straight line segments and transitions. Let $\gamma \in [-1, 0]$, then the reference position using cubic Bézier curves is given by

$$\mathbf{p}^n_{nb,\mathrm{ref}}(\gamma) = \gamma^3\mathbf{P}_0 + 3\gamma^2(\gamma+1)\mathbf{P}_1 + 3\gamma(\gamma+1)^2\mathbf{P}_2 + (\gamma+1)^3\mathbf{P}_3, \tag{8.4}$$

which facilitate penalties on the path variable that incentivize the controller to reach the end of the path. However, using $\gamma = s$ and Eq. (8.3) is also possible when the squared path variable is not part of the stage cost.

In normal waypoint tracking operations for UAVs an ordered list of waypoints is available from which a list of path parameterizations can be generated. We distinguish between straight line segments, that are commonly used in operation, and transitions between straight lines. Even though the complexity of cubic splines is unnecessary to parameterize straight lines, we use it to keep a consistent formulation of the reference which makes the implementation easier regarding most OCP solvers. Let a transition radius be denoted by $r \in \mathbb{R}$ and assume that the initial path segment is a straight line defined by two waypoints $[\mathbf{w}_{k-1}, \mathbf{w}_k]$. Suitable control knots $\mathbf{S}_i$ to parameterize a straight line are given by

$$\mathbf{S}_0 = \mathbf{w}_{k-1} + r\frac{\mathbf{w}_k - \mathbf{w}_{k-1}}{\|\mathbf{w}_k - \mathbf{w}_{k-1}\|_2} \tag{8.5}$$

$$\mathbf{S}_3 = \mathbf{w}_k - r\frac{\mathbf{w}_k - \mathbf{w}_{k-1}}{\|\mathbf{w}_k - \mathbf{w}_{k-1}\|_2} \tag{8.6}$$

$$\mathbf{S}_1 = \mathbf{S}_0 + 0.25(\mathbf{S}_3 - \mathbf{S}_0) \tag{8.7}$$

$$\mathbf{S}_2 = \mathbf{S}_0 + 0.75(\mathbf{S}_3 - \mathbf{S}_0). \tag{8.8}$$

A transition between two straight line segments defined by $[\mathbf{w}_{k-1}, \mathbf{w}_k]$ and $[\mathbf{w}_k, \mathbf{w}_{k+1}]$ can be heuristically parameterized as

$$\mathbf{T}_0 = \mathbf{S}_3 = \mathbf{w}_k - r\frac{\mathbf{w}_k - \mathbf{w}_{k-1}}{\|\mathbf{w}_k - \mathbf{w}_{k-1}\|_2} \tag{8.9}$$

$$\mathbf{T}_3 = \mathbf{w}_k + r\frac{\mathbf{w}_{k+1} - \mathbf{w}_k}{\|\mathbf{w}_{k+1} - \mathbf{w}_k\|_2} \tag{8.10}$$

$$\mathbf{T}_1 = \mathbf{w}_k + \varepsilon\frac{\mathbf{T}_0 - \mathbf{w}_k}{\|\mathbf{T}_0 - \mathbf{w}_k\|} \tag{8.11}$$

$$\mathbf{T}_2 = \mathbf{w}_k + \varepsilon\frac{\mathbf{T}_3 - \mathbf{w}_k}{\|\mathbf{T}_3 - \mathbf{w}_k\|}, \tag{8.12}$$

with ε as a tuning parameter that determines the distance of the inner control knots to the corner. A more elaborate scheme to find the control knots of the corner spline segments is the G2 Continuous Cubic Bézier Spiral Path Smoothing (G2CBS) by Yang and Sukkarieh [195]. It allows to define a maximum curvature constraint that will be satisfied by a resulting set of control knots

$$\mathbf{B}_c = \begin{bmatrix}\mathbf{B}_0 & \mathbf{B}_1 & \mathbf{B}_2 & \mathbf{B}_3\end{bmatrix} \in \mathbb{R}^{3\times4} \tag{8.13}$$

$$\mathbf{E}_c = \begin{bmatrix}\mathbf{E}_3 & \mathbf{E}_2 & \mathbf{E}_1 & \mathbf{E}_0\end{bmatrix} \in \mathbb{R}^{3\times4}, \tag{8.14}$$

for which Yang and Sukkarieh provide an analytical solution that can be implemented in a simple algorithm. This approach was used by Basescu and Moore [8] for control of an agile UAV via time-varying LQR in confined spaces. However, constraints can be directly implemented in our MPC formulation such that it is not necessary to use curvature constrained paths and simulations did not show significant differences in performance to the more simple spline transition using $\mathbf{T}_i$. The control knots enter as parameters into the OCP. They are updated to the control knots of the next path segment at $\gamma \geq \gamma_1$ and the path variable is reset to $\gamma = \gamma_0$.

Remark 8.1. Simulations show that a naive switching of path segments with the resulting sudden discontinuities in the path variables dynamics can be problematic. Other authors using similar controller designs present examples based on periodic path definitions and do not specifically address problems of switching path segments. See for example Faulwasser et al. [55] for industrial robots or Yang et al. [194] for UAV. However, in follow-up work, Faulwasser et al. [54] provide demonstrations for robotic manipulators that follow more complex geometries using polynomial splines with up to 1750 path segments. The results show that switching path segments should not be a problem with careful implementation of the NLP solver. Suitable modifications to the solver are part of ongoing work, and we will only consider periodic paths for the remainder of the chapter.

8.3 Controller Design

8.3.1 Dynamic Model

In this chapter we use the dynamic model in the body-fixed frame. Recall the ordinary differential equations from Chapter 2

$$\dot{\mathbf{p}}^n_{nb} = \mathbf{R}_{nb}\mathbf{v}^b_{nb} \tag{8.15}$$

$$\dot{\mathbf{v}}^b_{nb} = \frac{1}{\mathrm{m}}(\mathbf{R}^\top_{wb}\mathbf{f}^w_a + \mathbf{f}^b_t) + \mathbf{R}^\top_{nb}\mathbf{g}^n - \boldsymbol{\omega}^b_{nb} \times \mathbf{v}^b_{nb} \tag{8.16}$$

$$\dot{\mathbf{R}}_{nb} = \mathbf{R}_{nb}\mathbf{S}(\boldsymbol{\omega}^b_{nb}) \tag{8.17}$$

$$\mathbf{J}\dot{\boldsymbol{\omega}}^b_{nb} = \mathbf{S}(\mathbf{J}\boldsymbol{\omega}^b_{nb})\boldsymbol{\omega}^b_{nb} + \mathbf{m}^b. \tag{8.18}$$

To enable the NMPC to manipulate the position reference, the system state is augmented with the path variable γ and a double-integrator as timing law such that the acceleration of the path variable is part of the control inputs. Let the integrator state be denoted by $\mathbf{z} = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^\top \in \mathbb{R}^2$ and the control variable to manipulate the acceleration be denoted by $\nu \in \mathbb{R}$. The dynamic equations for the path variable then read

$$\dot{z}_1 = z_2 \tag{8.19}$$

$$\dot{z}_2 = \nu. \tag{8.20}$$

The actuator signal of the UAV is again denoted by $\boldsymbol{\delta} \in \mathbb{R}^{n_\delta}$, reflecting that different configurations are readily handled by the solver. We will however continue with the familiar flying-wing configuration of the Skywalker X8 to illustrate the approach, i.e. use aileron, elevator and throttle as actuators such that $\boldsymbol{\delta} = \begin{bmatrix} \delta_a & \delta_e & \delta_t \end{bmatrix}^\top$. The state vector $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{n_x}$ and control input vector $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^{n_u}$ are defined as

$$\mathbf{x} = \begin{bmatrix} {\mathbf{p}^n_{nb}}^\top & {\mathbf{v}^b_{nb}}^\top & {\mathbf{r}_x}^\top & {\mathbf{r}_y}^\top & {\mathbf{r}_z}^\top & {\boldsymbol{\omega}^b_{nb}}^\top & \boldsymbol{\delta}^\top & \mathbf{z}^\top \end{bmatrix}^\top, \tag{8.21}$$

$$\mathbf{u} = \begin{bmatrix} \dot{\boldsymbol{\delta}}^\top & \nu \end{bmatrix}, \tag{8.22}$$

where the rotation matrix is again decomposed as $\mathbf{R}_{nb} = \begin{bmatrix} \mathbf{r}_x & \mathbf{r}_y & \mathbf{r}_z \end{bmatrix} \in \mathrm{SO}(3)$.

In the following, we use a fourth order explicit Runge-Kutta scheme to integrate the actuator dynamics and the continuous dynamics defined by Eq. (8.15) - Eq. (8.20) and let $\mathbf{f}_{\mathrm{RK4}} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ denote the integrator function.

8.3.2 Reference and Cost

The primary objective is to find control actions such that the path-following error denoted by $\mathbf{e} = \mathbf{p}^n_{nb} - \mathbf{p}^n_{nb,\mathrm{ref}}(\gamma)$ is converging to zero. To improve convergence of the direction of travel of the UAV to the direction of the path, we define the vector $\boldsymbol{\eta}$ as the direction of the velocity vector in the inertial frame, effectively describing course and flight-path angle. The reference is the direction of the path $\boldsymbol{\eta}_{\text{ref}}$, and both are defined as

$$\boldsymbol{\eta}(\mathbf{x}) = \frac{\mathbf{R}_{nb}\mathbf{v}^b_{nb}}{\|\mathbf{R}_{nb}\mathbf{v}^b_{nb}\|_2}, \quad \boldsymbol{\eta}_{\text{ref}}(\mathbf{x}) = \frac{\nabla_\gamma \mathbf{p}^n_{nb,\text{ref}}(\gamma)}{\|\nabla_\gamma \mathbf{p}^n_{nb,\text{ref}}(\gamma)\|_2} \tag{8.23}$$

Using the reference $\boldsymbol{\eta}_{\text{ref}}$ allows for dropping the penalty term or lower constraint on γ to ensure forward motion along the path, which is the original approach proposed by Faulwasser and Findeisen [54]. The initial motivation for this cost term however is our observation that it helps to stabilize the dynamics even for large mismatches in the damping derivatives of the model. We also add a penalty on the deviation of the airspeed V_a from a given reference $V_{a,\text{ref}}$, which is usually the nominal cruise speed of the UAV. We then define the stabilizing stage cost as a sum of quadratic terms

$$l(\mathbf{x},\mathbf{u}) = \|k_p(\mathbf{p}^n_{nb} - \mathbf{p}^n_{nb,\text{ref})}(\gamma))\|^2_{\mathbf{Q}_p} + \|\boldsymbol{\eta}(\mathbf{x}) - \boldsymbol{\eta}_{\text{ref}}(\mathbf{x})\|^2_{\mathbf{Q}_v} + q_{V_a}(V_a - V_{a,\text{ref}})^2 + \|\mathbf{u}\|^2_{\mathbf{R}}, \tag{8.24}$$

with symmetric and positive-definite weighting matrices $\mathbf{Q}_p$, $\mathbf{Q}_\eta, \in \mathbb{R}^{3\times 3}$, $\mathbf{R} \in \mathbb{R}^{n_u \times n_u}$ and $q_{V_a} > 0$. The scaling parameter $k_p \in \mathbb{R}_{\geq 0}$ is included to define a distance to the path where the magnitude of the position error is equal to the error between velocity vector and path direction, which facilitates more intuitive tuning. The path reference is parameterized through control knots of the Bézier curve that enter the cost function through $\mathbf{p}^n_{nb,\text{ref}}$. For an easier notation, the reference is not included in the argument list of the cost function.

8.3.3 Constraints

We include the constraints on airspeed, angle of attack and actuators as discussed in Chapter 5 given by Eq. (5.14a) - Eq. (5.14d) and Eq. (5.16a) - Eq. (5.16f). Additional constraints related to the path variable are needed to ensure that the reference position is constrained to the defined path, formulated as

$$\gamma - \underline{\gamma} \geq 0 \tag{8.25}$$

$$-\gamma + \overline{\gamma} \geq 0. \tag{8.26}$$

Note that this is not necessary in cases where the path is periodic in the path variable and the operation does not require the system to stop at the end of the path. For example compare the endpoint of a robotic end effector in industrial manufacturing where the end-effector is not allowed to continue and a periodic surveillance pattern in UAV operations.

Remark 8.2. Faulwasser and Findeisen [54] moreover include a constraint expressed as $\dot{\gamma}(t) = \dot{z}_1 \geq 0$ to ensure monotonous forward motion along the path. We do not include this constraint to address cases where wind conditions are so severe that forward motion along the path may not feasible due to significant differences in the inertial and relative velocity. Note however that the positive airspeed constraint and alignment with the path direction through the stage cost result in forward motion in most flight conditions.

As in Chapter 5, constraints are needed to enforce the slack variables to be non-negative and possibly limit the actuator rates and acceleration along the path. The constraint expressions are given by

$$\mathbf{s} \geq 0, \quad \mathbf{u} - \underline{\mathbf{u}} \geq 0, \quad -\mathbf{u} + \overline{\mathbf{u}} \geq 0,, \tag{8.27}$$

where the inequality denotes an element-wise operation. For a more compact notation, let the constraints again be represented by $\mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{s}) \geq 0$.

8.3.4 Nonlinear Program and Implementation

As in Chapter 5, we use direct multiple-shooting [13] with an explicit Runge-Kutta integration scheme of order four to integrate $\mathbf{f}$ and let $\mathbf{f}_{\mathrm{RK4}}$ denote the resulting integrator function. The system is discretized into N steps with the resulting shooting interval $\Delta t = T/N$. The MPC scheme is then based on solving the NLP at time t for the predictions at $k \in [0, ..., N]$ given by

$$\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \sum_{k=0}^{N-1} l(\mathbf{x}(k|t),\mathbf{u}(k|t)) + \frac{1}{2}\mathbf{s}^\top \mathbf{P} \mathbf{s} \tag{8.28}$$

$$\text{s. t.} \quad \mathbf{x}(0|t) = \mathbf{x}(t) \tag{8.29}$$

$$\mathbf{x}(k+1|t) = \mathbf{f}_{\mathrm{RK4}}(\mathbf{x}(k|t), \mathbf{u}(k|t), \mathbf{d}(0|t)) \tag{8.30}$$

$$\mathbf{h}(\mathbf{x}(k|t), \mathbf{u}(k|t), \mathbf{s}) \geq 0. \tag{8.31}$$

The disturbance $\mathbf{d}(t)$ is again included for offset-free control and estimated by the disturbance observer presented in Chapter 5. As in the preceding chapter on NMPC, we implement the controller and the simulator for the disturbance observer using the open-source software package acados [185] and employ the Realtime-Iteration SQP solver based on [46] with the HPIPM presented in [57] for the solutions of the underlying QPs.

8.4 Simulation Study

We conduct a simulation study to test the performance of the proposed controller in following 3D curves. The controller can be used to follow parameterized geometric paths that are more general than circles or straight-line segments. We therefore use a Lissajous curve as done in [194] to define the reference path which is described by

$$\mathbf{p}^n_{nb,\mathrm{ref}}(\gamma) = \begin{bmatrix} 150\cos(2\pi\gamma) & 75\sin(4\pi\gamma) & -200 + 30\cos(\gamma) \end{bmatrix}^\top. \tag{8.32}$$

The curve is periodic in the path parameter, such that it is not necessary to switch between path segments and therefore avoid discrete jumps of the path variable. As discussed previously, we found naive resets of the path variable to be problematic when testing with spline curves and using a periodic path geometry enables the study of effects of prediction horizon and parametric and environmental disturbances, which is the primary objective of the simulation study. All simulations are run on a lab computer which includes eight CPU cores of the model AMD Ryzen 7 Pro 5850U.

8.4.1 Controller Parameterization

The direct multiple shooting scheme is parameterized with a discretization interval length $\Delta t = 0.1\,\mathrm{s}$. The controller is parameterized with the weights

$$\begin{gathered}\mathbf{Q}_p = \mathrm{diag}(50,50,50) \quad \mathbf{Q}_\eta = \mathrm{diag}(20,20,20) \quad q_{V_a} = 0.3 \quad k_p = 0.02. \\ \mathbf{R} = \mathrm{diag}(0.1,0.1,0.1,1) \quad \mathbf{P} = 10^3\mathbf{I}_6. \end{gathered} \tag{8.33}$$

The disturbance estimator, which is discussed in detail in Chapter 5, is parameterized with the gain matrix

$$\mathbf{L} = 0.03\mathbf{I}_4. \tag{8.34}$$

The simulation of the UAV uses a fourth order explicit Runge-Kutta integrator with step size set to 0.01 s. The disturbance estimator is updated at every simulation step, i.e. with a 100 Hz update rate, and the controller is updated at 20 Hz.

8.4.2 Initial Conditions and Wind

The initial flight condition of the UAV is set to a trimmed wings-level flight with direction towards the path at its nominal cruise speed with 18 m/s. There is no static wind component in the simulations, but highly dynamic gust winds generated by the Dryden wind model as described in [9]. A generated wind sequence is depicted in Fig. 8.1. The initial states are given by

$$\mathbf{p}^n_{nb} = \begin{bmatrix}-200\\0\\-200\end{bmatrix}\mathrm{m}, \quad \mathbf{v}^b_{nb} = \begin{bmatrix}17.99\\0\\0.55\end{bmatrix}\mathrm{m/s}, \quad \begin{bmatrix}\phi\\\theta\\\psi\end{bmatrix} = \begin{bmatrix}0\\1.76\\180.0\end{bmatrix}\mathrm{deg}, \quad \boldsymbol{\omega}^b_{nb} = \begin{bmatrix}0\\0\\0\end{bmatrix} \tag{8.35}$$

with the actuators and the control variables of the NMPC initially set to

$$\begin{bmatrix}\delta_a\\\delta_e\\\delta_t\end{bmatrix} = \begin{bmatrix}0.0\\2.10\\0.12\end{bmatrix}, \quad \begin{bmatrix}\gamma\\z\end{bmatrix} = \begin{bmatrix}-0.5\\0\end{bmatrix}, \quad \mathbf{u} = \mathbf{0}_{4\times1}, \tag{8.36}$$

where the control surfaces deflections $\delta_a,\ \delta_e$ are given in degrees.

8.4.3 Effect of the Prediction Horizon

The simulations include results for the prediction horizons $N \in \{10, 15, \cdots, 40\}$ under the gust wind profile shown in Fig. 8.1. The convergence to the path for each prediction horizon is depicted in Fig. 8.2. The controller is able to converge to the path for all prediction horizons and keep the UAV on the path despite the significant wind gusts. The shorter prediction horizons are stabilized through the reference for the direction of the inertial vector which, compared to longer prediction horizons, results in prioritizing alignment with the path direction and thus a slower convergence to the path. This is confirmed by inspecting the plotted distance and airspeed error in Fig. 8.3 which shows similar actuator usage after the initial convergence to the path for prediction horizons $N \geq 20$. For $N \geq 30$

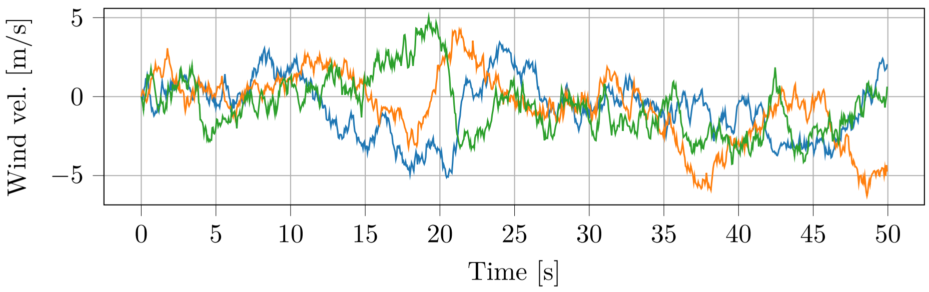

Figure 8.1: The wind velocity vector in the body-fixed frame coordinates along the x-axis (blue), y-axis (orange) and z-axis (green).

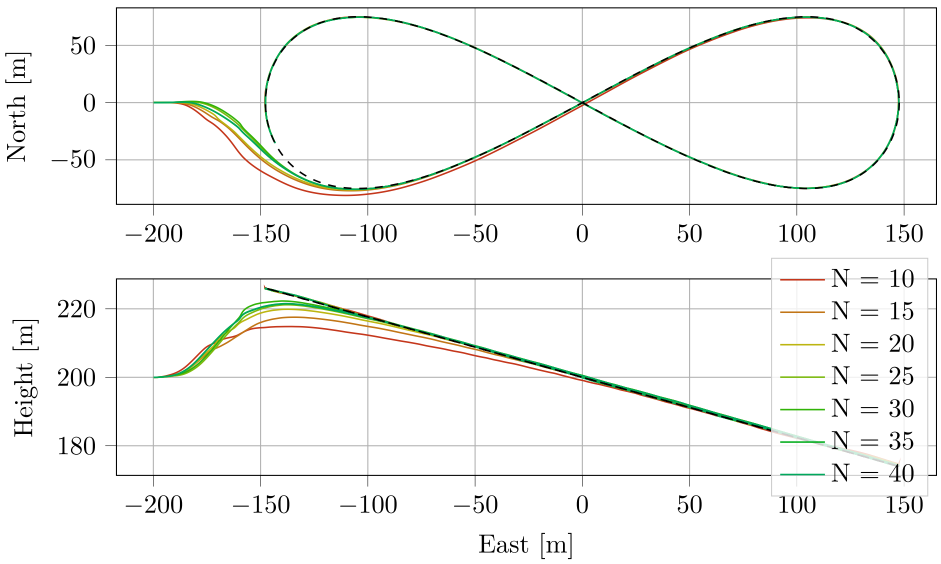

Figure 8.2: Position convergence to the reference path (dashed) given as a tilted Lemniscate for different prediction horizons N for the shooting interval $\Delta t = 0.1\,\mathrm{s}$. The shorter prediction horizons are stabilized through the reference for the direction of the inertial velocity vector and have a slower convergence to the path.

the performance is not distinguishable, indicating a good compromise between stabilizing horizon length and computational demand. We will therefore continue using $N = 30$ in the following Monte-Carlo study.

8.4.4 Effect of parametric Disturbances and Wind

We conduct a Monte-Carlo simulation study to test the controller under turbulent wind conditions and significant model mismatch. The gust winds are again generated through the Dryden model resulting in qualitatively similar wind profiles as the one shown in Fig. 8.1. To study the effect of model mismatch, the UAV is sim-

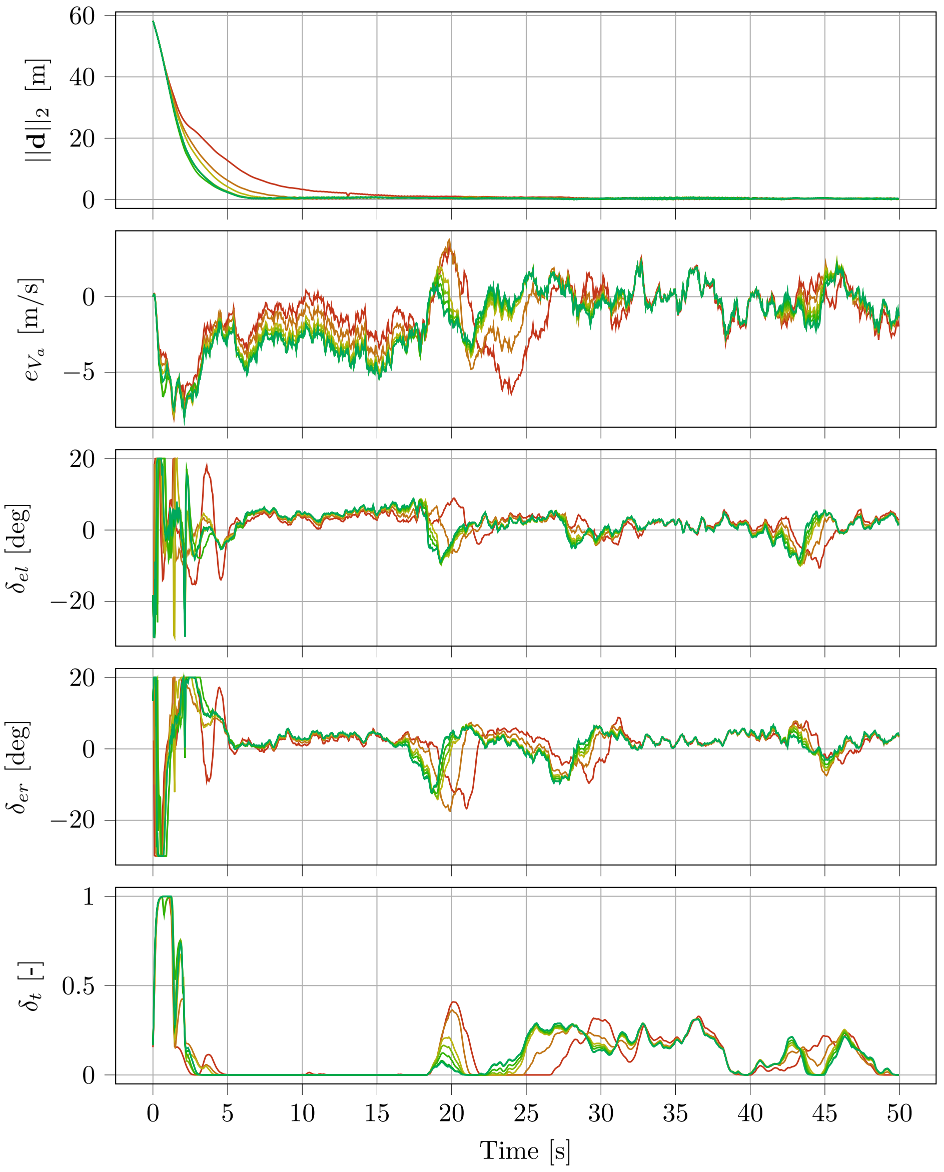

Figure 8.3: Distance to the path and airspeed error for varying horizon lengths ($N = 10$ in read to $N = 40$ in green). The left/right elevon deflections and throttle set-point are also shown. The shorter prediction horizons prioritize stabilization of the airspeed to cruise speed at the expense of slower convergence to the path.

ulated using the nominal model parameters that were used throughout the thesis. The dynamic model of the controller has the same structure except for the actuator dynamics which we again model as first-order systems as described in the simulation study of Chapter 5. To simulate parametric disturbances the parameters of the controller model are drawn from a normal distribution $\mathcal{N}(\rho, \sigma^2)$ with nominal parameter $\rho \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}$. The standard deviation for each parameter distribution were chosen as summarized in Table 8.1. The distributions for the static coefficients are parameterized with smaller standard deviations which reflects our experience that they can be determined with more accuracy, e.g. from wind-tunnel experiments, compared to the dynamic coefficients which require flight tests or alternative CFD methods. See Chapter 7 for a more detailed discussion regarding model identification efforts. In extreme cases the dynamic coefficients drawn from the distribution are even allowed to have a different sign than the nominal parameter, which amounts to a severe parametric disturbance.

The Monte-Carlo study includes 100 simulations starting from the initial condition described in the previous section. Out of the 100 simulations, the controller was able to converge to the reference in 97% of the cases. The remaining 3% of the simulations were interrupted due to 10 consecutive fails of the underlying QP solver. The distance to the path and deviation of the airspeed from the reference are shown in Fig. 8.4 with their respective mean values and 1σ-band around it. The initial convergence to the path during the first five seconds shows a robust performance along all parameter perturbations. It is notable that convergence to the path is significantly prioritized over tracking the airspeed reference which is accelerated to its upper bound of 25 m/s. A smaller scaling value k_p can help to mitigate this effect. An alternative is to temporarily tighten the airspeed limits until the UAV is at a shorter distance to the path where the problem is less pronounced. An approach that would change the structure of the problem is to use multidimensional paths as suggested by Matschek et al. [122] to conceptually define a tube around the original path as a reference. The systematic airspeed error until approximately 25 s into the simulation indicates that the decreasing height profile introduces as conflicting objective where the NMPC needs to find a compromise. A deviation of 2 m/s is however in the acceptable range.

The closed-loop computation times of the NLP solver includes the evolution of the maximum computation times which is mostly in a band around 10 ms and rarely exceeds 20 ms. However, the resources of the lab computer used to run the simulations significantly exceed the resources that are available onboard the UAV as outlined in Chapter 3. The evaluation of the runtime on the targeted hardware is part of future work in preparation of the flight experiments.

8.5 Discussion and Future Work

We discussed the design of a path-following NMPC following ideas proposed by Faulwasser and Findeisen [54] with minor modifications. Initial simulation results show good path-following capabilities even for smaller prediction horizons due to the additional penalty term on the direction of the linear velocity vector. However, there are some challenges that we did not address. The most important one is a

Table 8.1: Controller parameters are sampled at the start of each simulation according to these distributions, where ρ denotes the nominal parameter value that is used by the simulated UAV model. The static coefficients can be determined in wind-tunnel experiments with significantly less uncertainty compared to the dynamic coefficients, which is expressed through the standard deviation of the distribution.

Parameter	Distribution
$C_{D_0}, C_{D_{\alpha_1}}, C_{D_{\alpha_2}}, C_{D_{\beta_1}}, C_{D_{\beta_2}}, C_{D_{\delta_e}}$ $C_{L_0}, C_{L_\alpha}, C_{L_{\delta_e}}, C_{Y_\beta}, C_{Y_{\delta_e}}, C_{l_\beta}, C_{l_{\delta_a}}$ $C_{m_0}, C_{m_\alpha}, C_{m_{\delta_e}}, C_{m_{\text{fp}}}, C_{n_\beta} C_{n_{\delta_a}}$ $C_{\text{prop}}, J_{\{x,y,z,xz\}}, M, a_0, k_\Omega, k_{T_p}, k_{\text{motor}}$	$\mathcal{N}(\rho, \rho \cdot 0.1)$
$C_{D_p}, C_{L_q}, C_{Y_p}, C_{Y_r}, C_{l_p}, C_{l_r}, C_{m_q}, C_{n_p}, C_{n_r}$	$\mathcal{N}(\rho, \rho \cdot 0.5)$

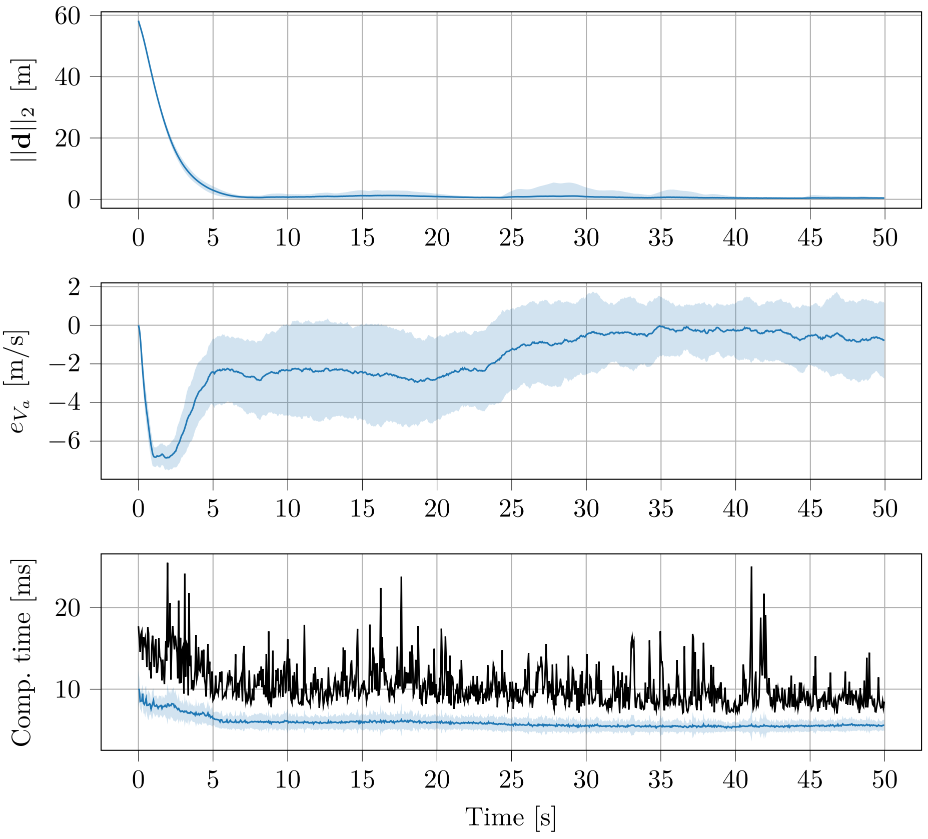

Figure 8.4: Results of the Monte-Carlo study including the distance to the path, airspeed error, and computation times needed by the NLP solver. The respective mean value (blue, solid) with a band indicating addition and substraction of one standard deviation (blue, shaded) are shown. The maximum time (black, solid) is also shown. The controller is able to keep the UAV on the path despite significant parametric disturbances and severe wind conditions. The worst case computation times indicate that the required update rates for flight experiments can be met by the controller.

suitable switching logic that enables to transition between different paths which is a major ingredient to stitching together a sequence of parametric cubic Bézier curves such that almost arbitrarily complex geometries can be used. Resetting the path variable at each transition leads to significant discrete jumps in the OCP which are not well-handled by the current controller design, even in the case of sufficiently smooth geometries. Another topic that we did not discuss is the initial convergence to the path from a wider range of initial conditions, and it can be expected that the short prediction horizons of the given simulation study will not suffice for closed-loop stability of the defined path when the UAV is at increasingly longer distance to the path in opposite direction of travel. Finally, the evaluation of the closed-loop computation times on the embedded hardware remains to show that the real-time application is feasible, which can be seen as the last milestone towards experimental verification. So far, computation times are only evaluated on a more powerful lab computer. Critical components such as the dynamic model and the hardware and software architecture were the subject of Chapter 7 and Chapter 3, respectively, and used to successfully apply NMPC for low-level motion control as shown in Chapter 5.

8.6 Benchmark Scenario

The path-following MPC presented in this chapter has a different control objective compared to the low-level motion controllers of the preceding chapters. It is therefore not possible to use the same tuning procedure based on the reference steps as outlined in Appendix A. We instead use a simple tuning that achieved satisfactory performance when following a loiter pattern in simulations. This resulted in the cost matrices

$$\mathbf{Q} = \operatorname{diag}(q_{V_a},\ q_{e,N},\ q_{e,E},\ q_{e,D},\ q_{\eta,x},\ q_{\eta,y},\ q_{\eta,z}) = \operatorname{diag}(1, 1, 10, 1, 1, 1, 1) \tag{8.37}$$

$$\mathbf{R} = \operatorname{diag}(r_{\dot{\delta}_a},\ r_{\dot{\delta}_e},\ r_{\dot{\delta}_t}) = \operatorname{diag}(1, 1, 1). \tag{8.38}$$

The prediction horizon is set to $N = 30$ and the length of the shooting intervals is set to $\Delta t = 0.1\,\mathrm{s}$, matching the LLMPC and MPCGC. In contrast to the preceding control algorithms, the benchmark simulation for the PFMPC gives notably different results, as shown for the position trajectories in Fig. 8.5 and the distance, error signals and actuator signals in Fig. 8.6. The PFMPC is able to keep the UAV close to the path at comparable airspeed-tracking performance and significantly less rapid actuator changes, which is shown in Fig. 8.6 and confirmed by the scores on the respective metrics in Table A.2. The decisive difference for the PFMPC in comparison to the LLMPC and MPCGC is that it integrates guidance and low-level motion control into one control algorithm with an internal dynamic state. Instead of reacting to commands from a guidance controller as the LLMPC and MPCGC, the PFMPC can re-plan the trajectory to the path whenever necessary and therefore achieves better path-following performance at less control effort. The LLMPC and MPCGC on the other hand have to use constant reference signals along the prediction horizon that match the current attitude and speed command, which significantly reduces the comparative benefits of a predictive controller.

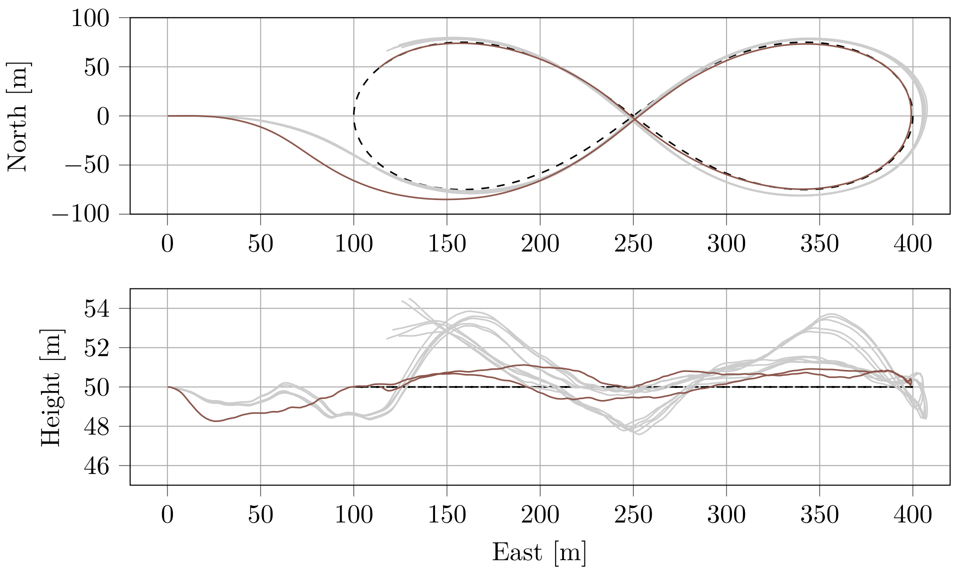

Figure 8.5: Benchmark simulation comparing the path-following MPC (PFMPC, plotted in brown) introduced in this chapter to the previous controller designs (gray). The PFMPC keeps the UAV significantly closer to the reference path (black, dashed).

It is however also important to note that the NDGPFG which we employed as a guidance controller does not include an integrator state in the implementation that we used in the benchmark simulations. Augmenting the NDGPFG with integral action would enable a more accurate comparison. Nonetheless, the arguments regarding the benefits of the integrated approach of the PFMPC over the LLMPC and MPCGC, which are run in a reactive fashion, still hold. A last note concerns the comparably big airspeed error of the PFMPC in the first five seconds during the convergence to the path. This effect is due to initializing the position of the UAV at a distance to the path where the PFMPC prioritizes to path-following performance over airspeed. As soon as the UAV is close enough to the reference path, the airspeed is giving higher priority and the corresponding error trajectory is similar to those of the LLMPC and MPCGC. To reduce the effect of the initial convergence to the performance metrics, they are evaluated for the interval $t \in [10, 50]$.

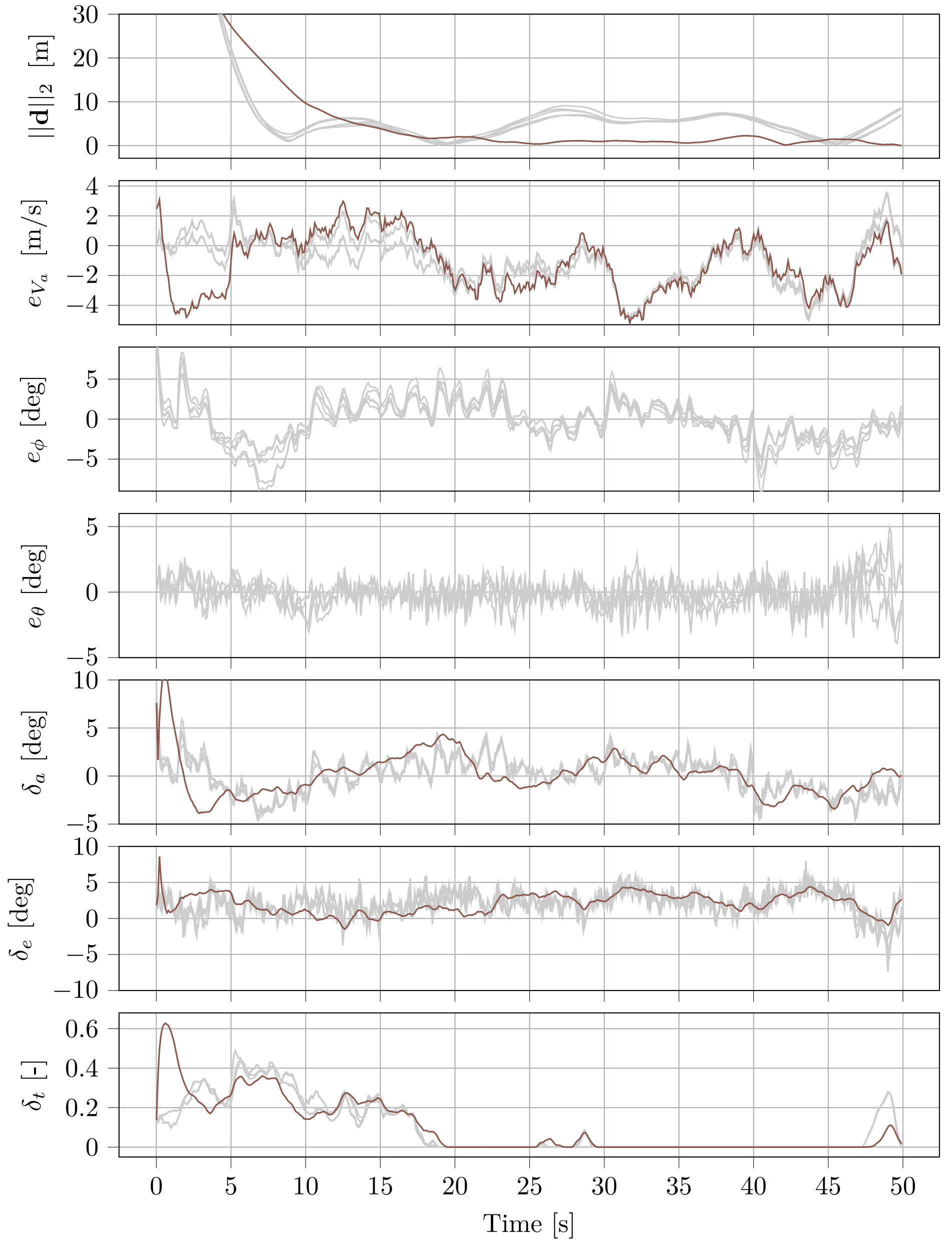

Figure 8.6: Benchmark simulation comparing the path-following MPC (PFMPC, plotted in brown) introduced in this chapter to the previous controller designs (gray). The PFMPC keeps the UAV significantly closer the reference path at less control effort.

8.7 Chapter Summary

This content of this chapter is preliminary research on tackling the path-following problem by using NMPC with direct access to the actuators instead of relying on off-the-shelf autopilots which is the state of the art so far. The controller design follows the general proposition on constrained path-following NMPC by Faulwasser and Findeisen [54]. We use an additional penalty term on the deviation of the linear velocity vector to the path direction which helps to stabilize the path-following error for even smaller prediction horizons at the cost of potentially slower initial convergence to the path. A simulation study demonstrates the controller's robustness to gust winds and parametric disturbances in a Monte-Carlo study. We discussed future work toward employing cubic Bézier curves as parameterized path and remaining work towards flight experiments.

Chapter 9

Concluding Remarks and Future Work

9.1 Conclusion

Each chapter that included contributions ends with a summary and a brief conclusion of the related work. The purpose of this chapter is to widen the scope and hint to possible future directions that start at open ends, which this thesis did not cover. We will discuss suggestions for further work in the order in which the contributions appeared.

Chapter 3: Experimental Platform

The work of this chapter is not a contribution that attempts to give answers to research questions but work that facilitates the testing of experimental algorithms for the control of fixed-wing UAVs. A description of the hardware components onboard the UAV and test procedures in simulations and experiments summarize our experience in the Autofly project and may help other researchers avoid pitfalls by providing lessons learned.

Future work should consider establishing on-site facilities at the research institute for agile fixed-wing UAVs similar to the facilities used by Basescu and Moore [8] or Bulka et al. [17]. Performing flight experiments with fixed-wing UAVs is an outdoor sport. Weather conditions, travel time to the airfield, and the size of the test crew are all elements that make this a substantial undertaking. In retrospect, we could have first set up some test facilities at our department (possibly indoors) for more rapid development cycles, using a smaller model airplane. In this way, the complete tool-chain could have been tested thoroughly before moving on to larger vehicles in outdoor experiments. Preliminary tests are not limited to control algorithms but can include data collections for model identification.

Chapter 4: Geometric Attitude Control

This chapter presented geometric attitude control laws for tracking roll and pitch angle reference signals. We designed the control laws directly on the two-sphere instead of relying on more conventional attitude representations and discussed the possible benefits of this approach. A rigorous Lyapunov analysis provided proof of almost global asymptotic stability of constant attitude references in closed-loop with the proposed control law and almost semi-global exponential stability for tracking time-varying attitude reference signals. The extension to hybrid feedback rendered time-varying attitude references globally exponentially stable. Numerical results show the viability of the controllers, and initial experiments demonstrated its applicability in flight tests. This contribution addresses the research objective to design Lyapunov-based attitude control algorithms with proven global stability properties to recover the UAV from loss of control and stabilize it from attitudes far from the reference, which we demonstrated in simulations.

The theoretical results are strong, but the dynamic model inversion imposes a strong assumption on the model's accuracy. Work that progresses from this thesis's results should analyze the stability properties under parametric uncertainties and external disturbances. E. Coates, who collaborated on both papers, pursued this direction and published further work [35, 36]. This work also uses a backstepping design to relax the need for knowing angular acceleration references. Another important topic that the resulting controller does not address is the limits of the actuators and safety constraints. One possible approach is to modify the output of the control methods to (virtual) control moments and then map these to the actuator suite through control allocation algorithms [183].

Further, the geometric attitude controllers should be subjected to a more thorough experimental verification. So far only preliminary flight tests have been conducted to demonstrate the control performance when tracking roll and pitch references from a guidance module.

Chapter 5: Direct Nonlinear Model Predictive Control for Attitude and Speed Control

The contribution of this chapter is a NMPC design for low-level motion control of fixed-wing UAVs with direct access to the actuators instead of relying on off-the-shelf control boards that track attitude reference signals generated by a guidance-level MPC. Experiments demonstrated that real-time control is feasible using state-of-the-art numerical algorithms and hardware. The performance in comparison to industry-standard PID controllers was either matched or exceeded. This chapter addresses the second research task to develop NMPC designs that can function as a reference method to assess the performance of the Lyapunov-based controllers with a smaller computational footprint. Given the promising results on the real-time applicability of the controller, we further pursued experiments that showed superior results to industry-standard PID controllers.

There are three main directions for future work regarding the experiments of the controller:

1. Improve the dynamic model of the controller. We have already looked at some shortcomings of the aerodynamic model and possibilities to increase its quality in Chapter 7.

2. The improved aerodynamic model should be used in the dynamic constraints to repeat the experiments described in this chapter.
3. It would be very interesting to conduct experiments with designs that include more aggressive maneuvers where the safety-related constraints are active over a considerable period.

Another intriguing enhancement is to use learning-based MPC in challenging maneuvers. One possibility is to augment the developed NMPC with a data-driven approach such as Gaussian process regression [191] to capture the error of the nominal dynamic model. Successful applications of this approach include problems similar to improving the aerodynamic model of fixed-wing UAVs. Prominent examples are learning tire friction in racing cars or improving the aerodynamic model of multi-rotor UAVs. Hewing et al. [77] and Torrente et al. [184] demonstrated successful applications in experiments. Another promising way to enhance our controller would be the combination of RL and NMPC, which Gros and Zanon [67] thoroughly discuss.

Chapter 6: Coupled Nonlinear Model Predictive Control and Geometric Attitude Control

This chapter exploits two other contributions, i.e., the presentation of the geometric controller in Chapter 4 and the NMPC for low-level motion control with direct actuator access in Chapter 5. The cascade of the geometric controller and the NMPC showed promising results in numerical simulations. The NMPC is a viable approach to generate the angular rate references for the low-level geometric attitude controller. Assuming full dynamic model inversion of the nominal geometric attitude controller and the resulting requirements for an accurate model, the use of an additional NMPC in a cascade does not add much to the necessary information of the dynamic model. However, it does require other computing resources to employ the numerical solver on a suitable SBC. When working on this controller, it was not clear if it was possible to run the NMPC design with direct actuator access in real-time such that the solver could meet the required update rates for the fast dynamics of the UAV. Seeing the successful experiments of the NMPC design in Chapter 5 and its overall better performance in the simulation study, we did not pursue this approach any further. It would, however, be interesting to see it tested in flight experiments. The work that led to this thesis contributed sufficiently to the infrastructure at the UAV Lab to see this type of experiment as low-hanging fruit.

Chapter 7: Extended Aerodynamic Modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle

This chapter reflects work on system identification efforts that concluded in an improved aerodynamic model of the Skywalker X8 airframe. The chapter begins with developing a method to increase the symmetry of generalized aerodynamic force measurements of an airframe that is by design symmetric to its longitudinal plane. Identifying the dynamic coefficients based on flight data is well-established and not a result of this chapter. However, the application of the SINDy method proposed by Brunton et al. [16] to augment the model from the wind tunnel with additional damping is, to the best of my knowledge, first presented in this thesis. The extended model builds on previous work at the UAV Lab that identified a propulsion model and the matrix of inertia. Both were key to isolating the aerodynamic force contributions to the observed generalized forces. Considering that this work was always secondary to work on control design, it is fair to assume that a more focused modeling effort in this direction can further improve the accuracy of the model and consequently improve the performance of the model-based control algorithms of the preceding chapters.

The work is primarily intended to contribute to the model-based controller designs. However, it also helps to improve the simulation models to assess the controller performance before experimental verification.

Future work should include more advanced measurement equipment of the relative velocity vector in the flight experiments and, in general, high-resolution sensors to log more accurate data at higher rates to improve the quality of the datasets. Such measurement equipment is available at the UAV Lab. We moreover observed that using PX4 instead of ArduPilot would enable updating state estimates at more reliable and higher rates. The estimates of the ArduPilot EKF, which we used for the data collection, were updated at 0.02s periods. This update rate is 20x slower than what can be achieved using high resolution IMUs or the rates in the demonstrations to identify the longitudinal dynamics by Kaiser et al. [88].

Follow-up work should also consider a more thorough experiment design regarding variations in the throttle input and its effect on the dynamics. Our focus was primarily on the impact of the control surface deflections. Related work on the assessment of maneuvering qualities of fixed-wing UAVs has been presented by Capello et al. [22], which may complement the identification efforts to see if the maneuvering capabilities of the model matches the observations from experiments.

Chapter 8: Direct Nonlinear Model Predictive Control for the Path-Following Control Problem

The controller designs in this thesis focus in most parts on low-level motion control. This chapter includes a wider scope and presents work on designing a NMPC for the path-following problem. The NMPC design of Chapter 5 is modified to follow parametric curves in three-dimensional Euclidean space. Initial results using the framework by Faulwasser et al. [54] are promising but limited to numerical examples. The resulting controller is similar to the one that Yang et al. [194] recently published, but with the difference that we consider the full dynamic model instead of a kinematic guidance model for controller design. It remains to evaluate the real-time feasibility on our experimental platform before an experimental verification can be conducted.

9.2 Other Future Work

While the previous section gave an outline of the road ahead from each chapter's contributions, there is more future work to enhance the results of this thesis. The initial plan was to demonstrate the use of the developed controllers in challenging case studies as outlined in Section 1.3.

However, we were not as quick to build our algorithms to the mature level needed to include them in more advanced case studies. Consequently, the results of this thesis are limited to simulations and initial experiments to demonstrate the controllers in maneuvers that have the sole purpose of proving their practical usability.

Appendices

Appendix A

Benchmark Comparison

A.1 Tuning

Table A.1: Parameterization for each controller in the benchmark scenario.

Controller	Parameters	Values
PID	$k_{p,\phi}$, $k_{i,\phi}$, $k_{d,\phi}$, $k_{p,\theta}$, $k_{i,\theta}$, $k_{d,\theta}$, k_{p,V_a}, k_{i,V_a}	1.00, 0.10, 0.10, 2.00, 0.50, 0.10, 0.08, 0.05
AP	k_{ϕ}, $k_{p,p}$, $k_{i,p}$, $k_{ff,p}$, k_{θ}, $k_{p,q}$, $k_{i,q}$, $k_{ff,q}$, k_{p,V_a}, k_{i,V_a}	3.00, 0.17, 0.03, 0.30, 5.00, 0.20, 1.20, 0.30, 0.08, 0.05
GC	k_p, $K_{d,x}$ $K_{d,y}$ $K_{d,z}$, $K_{i,x}$ $K_{i,y}$ $K_{i,z}$	20.00, 2.00, 2.00, 2.00, 2.00, 2.00, 2.00
MPCGC	q_{V_a}, $q_{\Gamma,x}$, $q_{\Gamma,y}$, $q_{\Gamma,z}$, $r_{\dot{\delta}_a}$, $r_{\dot{\delta}_e}$, $r_{\dot{\delta}_t}$	0.01, 30.00, 30.00, 30.00, 0.01, 0.01, 0.01
MPCGC	q_{V_a}, $q_{\Gamma,x}$, $q_{\Gamma,y}$, $q_{\Gamma,z}$, $r_{\dot{\omega}_x}$, $r_{\dot{\omega}_y}$, $r_{\dot{\omega}_z}$	0.01, 30.00, 30.00, 30.00, 1e-3, 1e-3, 1e-3
PFMPC	q_{V_a}, $q_{e,N}$, $q_{e,E}$, $q_{e,D}$, $r_{\dot{\delta}_a}$, $r_{\dot{\delta}_e}$, $r_{\dot{\delta}_t}$	1.00, 1.00, 10.00, 1.00, 1.0, 1.00, 1.00, 1.00, 1.00, 1.00

All low-level motion controllers are tuned based on a step for each channel, i.e. roll, pitch, and airspeed, as depicted in Fig. A.1. All controllers have been tuned such that they achieve similar performance for each step. For a fair comparison between the low-level controllers, for both MPCs, the reference along the prediction horizon is set to the reference at the initial conditions, meaning they do not have information about the coming step change. The step responses are mostly in-distinguishable, except for the qualitative difference between the MPCs and the reactive controllers regarding the coupling between airspeed and pitch control. At the step decrease in airspeed from 21 m/s to 15 m/s, the MPCs actively pitch up to decelerate. Later at the pitch step to 10 deg, the MPCs predict a decrease in airspeed and increase the throttle early to avoid a drop in airspeed. Similar behavior can be achieved by applying a TECS, but the reactive low-level motion controllers (PID, AP, GC) can not do more than decreasing the throttle to the lower limit and let the drag decelerate. The parameterizations from the tuning procedure are summarized in Table A.1 which also includes the parameterization for the PFMPC.

A.2 Performance Metrics and Figures

The trajectories for the positions, distance to the path, error signals and actuator usage are plotted in Fig. A.2 and Fig. A.3. From Fig. A.3, the actuator usage of the low-level MPCs is similar to the reactive controllers, i.e. the PID, AP and GC, which can be best seen for the aileron deflections δ_a. The path-following performance in

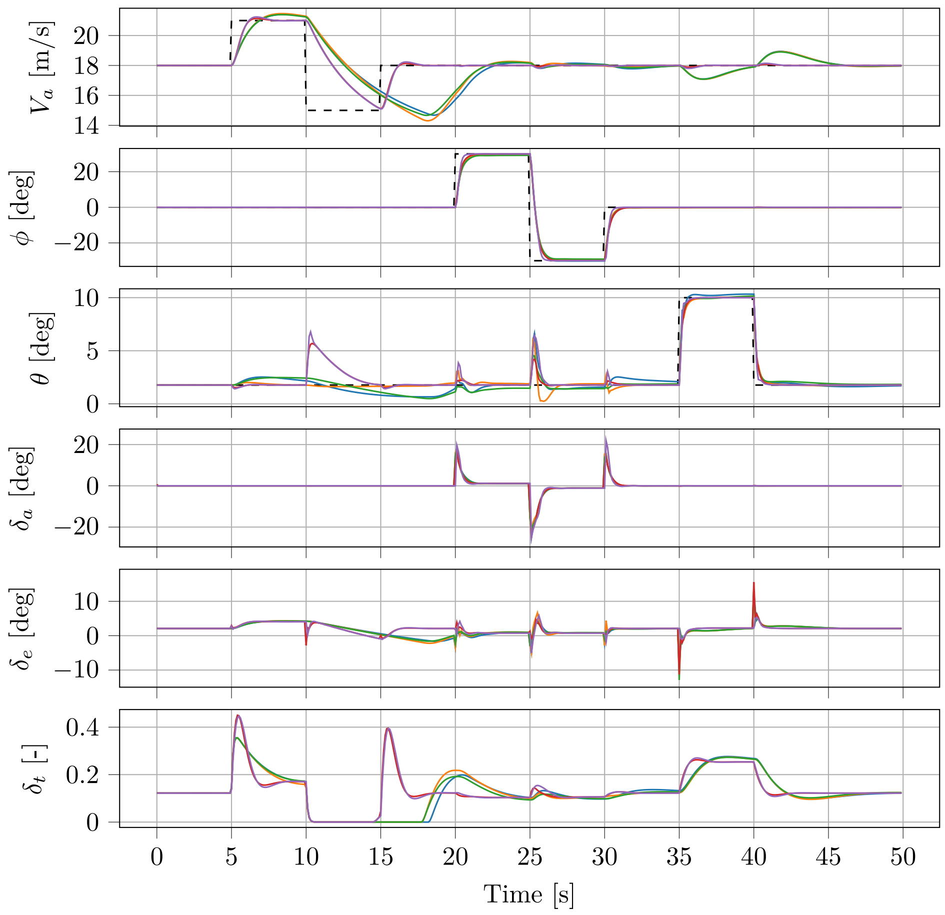

Figure A.1: Step responses for each controller before applying them to the benchmark. All controllers are tuned for comparable performance on each step. One exception is the negative in airspeed which the MPCs are better equipped to handle by pitching up to decelerate faster.

Table A.2: Performance metrics as given by Eq. (2.74) for the basic PID (PID), Ardupilot (AP), Geometric Controller (GC), low-level MPC (LLMPC), the combination of MPC and GC (MPCGC) and the path-following MPC (PFMPC).

	$J_{f_{\delta_a}}$	J_{f,δ_e}	J_{f,δ_t}	J_{u,δ_a}	J_{u,δ_e}	J_{u,δ_t}	$J_{e,d}$	J_{e,V_a}	$J_{e,\phi}$	$J_{e,\theta}$
PID	1.49	1.93	0.08	1.37	2.53	0.04	4.39	1.78	1.52	0.72
AP	1.86	2.45	0.08	1.40	2.55	0.04	4.86	1.78	1.53	0.78
GC	0.94	1.70	0.09	1.30	2.51	0.04	4.82	1.77	2.41	0.77
LLMPC	2.82	6.22	0.07	1.42	2.37	0.05	4.23	1.67	1.42	0.83
MPCGC	1.46	4.16	0.07	1.31	2.35	0.05	4.42	1.66	2.06	0.80
PFMPC	0.53	0.77	0.05	1.34	2.09	0.05	1.84	1.86		

loop with the guidance controller is therefore very similar for all controllers, as can be seen from the position plot in Fig. A.2 and the performance MSE of the distance $J_{e,d}$ in Table A.2.

The MPC with direct access to the actuators (LLMPC) achieves the best performance in the metrics for the path distance, airspeed (J_{e,V_a}) and roll tracking ($J_{e,\phi}$). The fact that the pitch angle tracking, as indicated by $J_{e,\theta}$, is worse compared to the reactive controllers (PID, AP, GC), suggests that a trade-off between airspeed and pitch tracking is being made. It is important to note that the LLMPC is also using significantly larger changes in the control surfaces, as indicated by J_{f,δ_a}, J_{f,δ_e}, which is to some extent mitigated for the combination of the MPC and the GC (MPCGC). However, the tuning of the $\mathbf{R}$ matrix of the LLMPC has a direct impact on this performance metric and can be used in cases where actuator set-point changes are too fast.

The behavior of the PFMPC is significantly different, due to the fact that the controller has a dynamic state to control the reference point on the path. This allows the PFMPC to replan within its prediction horizon instead of purely reacting to a higher-level guidance controller as is the case for the low-level controllers. The smoothness metric $J_{f,\cdot}$ in Table A.2 is consequently significantly better for the PFMPC when compared to the reactive controllers. This is not surprising when considering the relatively high cost of the elements of $\mathbf{R}$ in the PFMPC, as given in Table A.1.

The overall goal of keeping the UAV close to the reference path is significantly better achieved by the PFMPC compared to the low-level motion controllers in loop with the guidance controller. However, a fair comparison on a guidance level would require to the NDGPFG to be augmented with integral action to minimize the visible offset in the cross-track error that can be seen in Fig. A.2. The main goal of this running example was to compare the low-level motion controllers, where it is only required to use equal guidance controllers where the path-following performance is not critical.

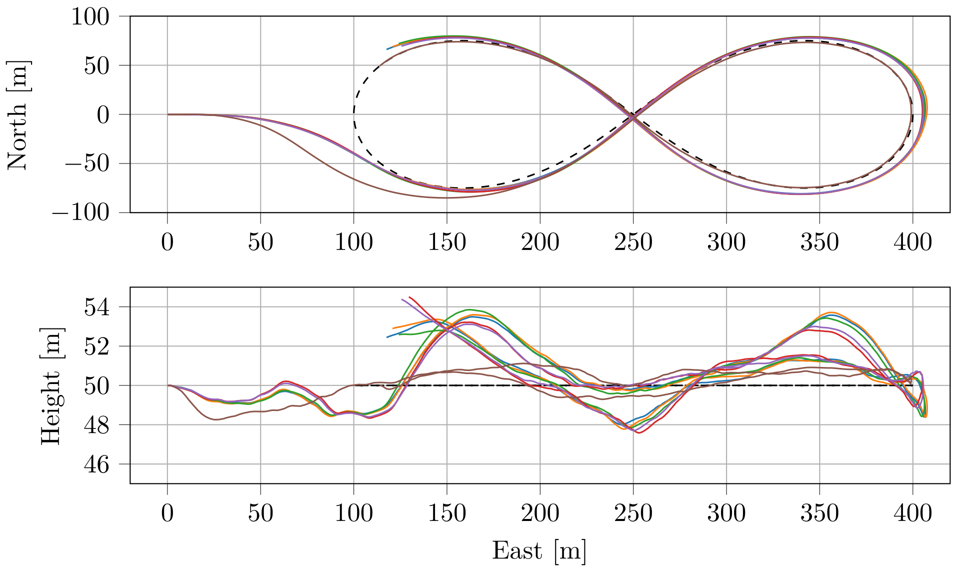

Figure A.2: The position of the UAV in the benchmark simulation for all controllers.

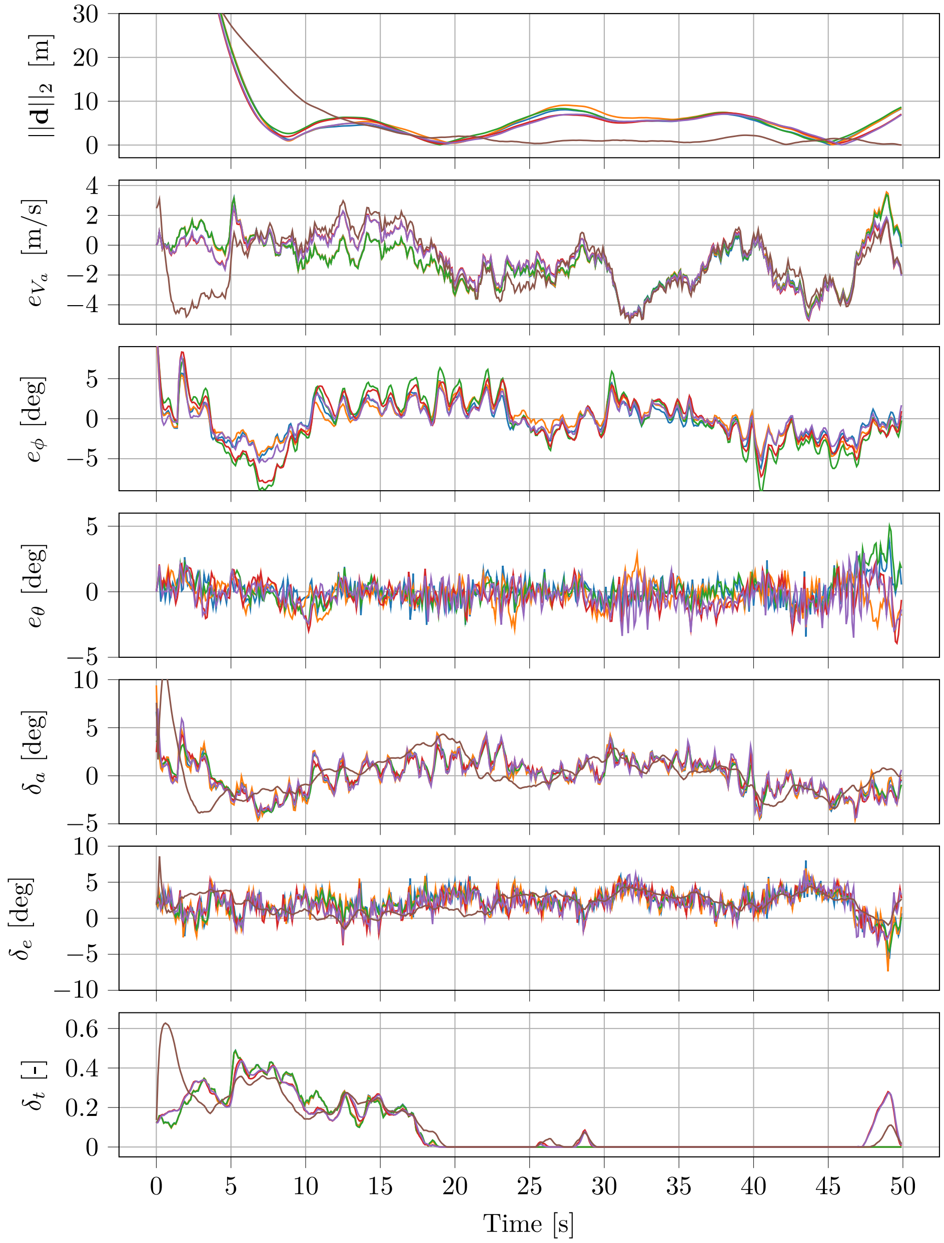

Figure A.3: Distance to the path $\|\mathbf{d}\|$, errors for airspeed, roll, pitch, and actuator usage for all controllers in the benchmark scenario.

Appendix B

Experiment Plots for Chapter 5: Direct Nonlinear Model Predictive Control for Attitude and Speed Control

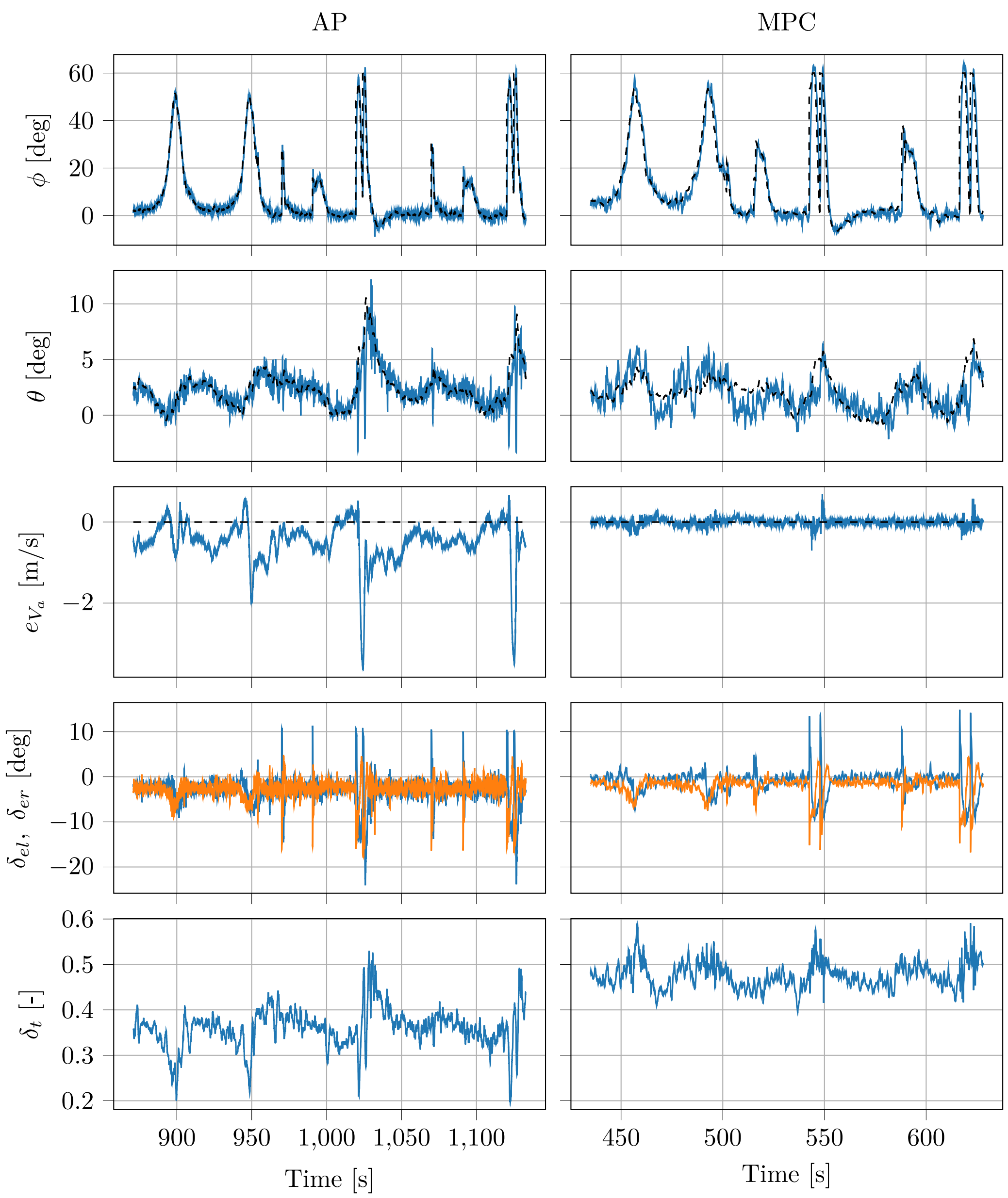

Figure B.1: Tracking performance for both controllers when following two circular paths in the beginning and two rectangular paths in the end. The reference (dashed) is shown for roll and pitch angle. The deflections for the left elevon (blue) and the right elevon (orange) are also shown.

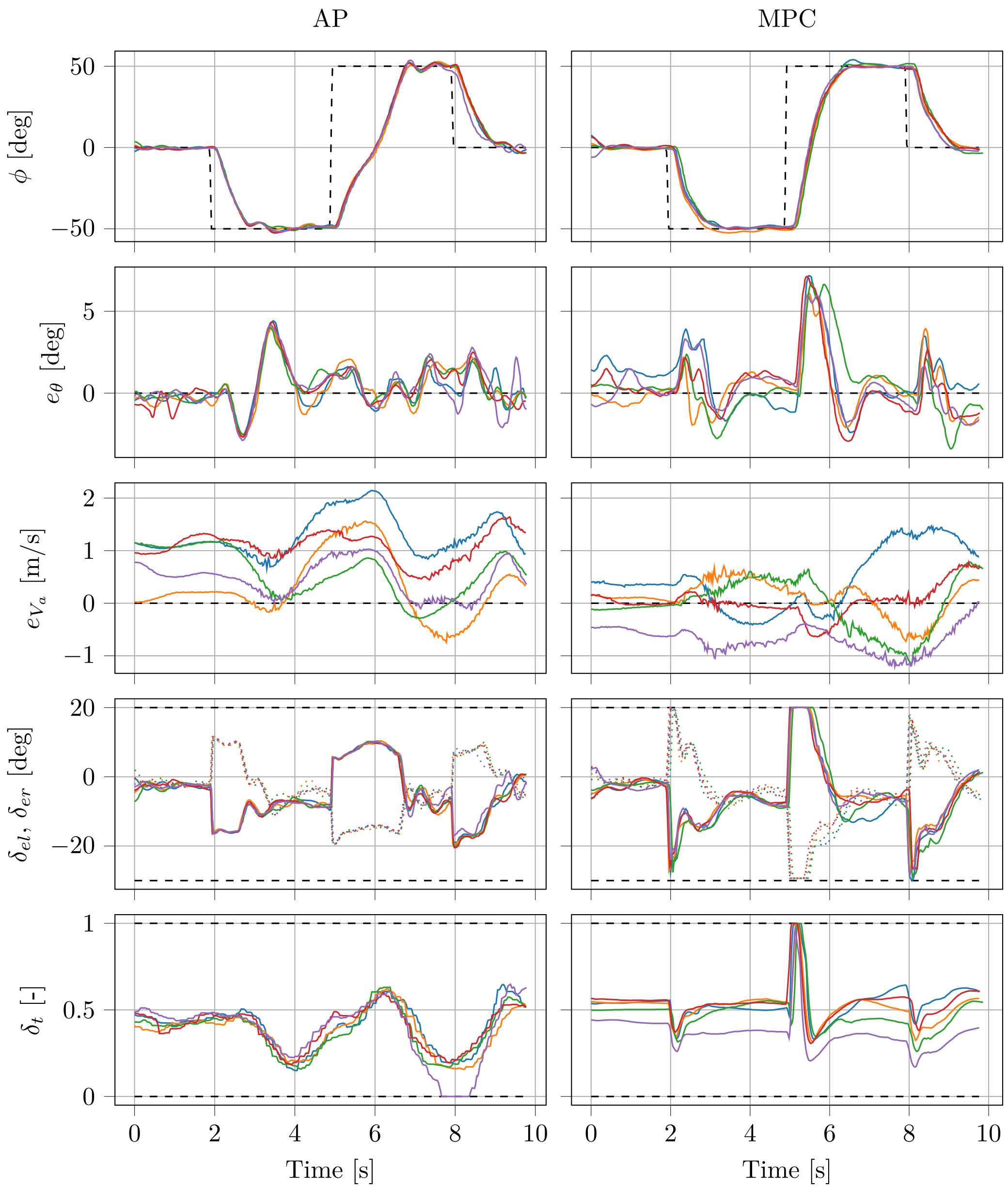

Figure B.2: Responses to roll reference steps including the roll angle trajectories and their reference (dashed) in the first subplot. The colors encode different runs for the same controller.

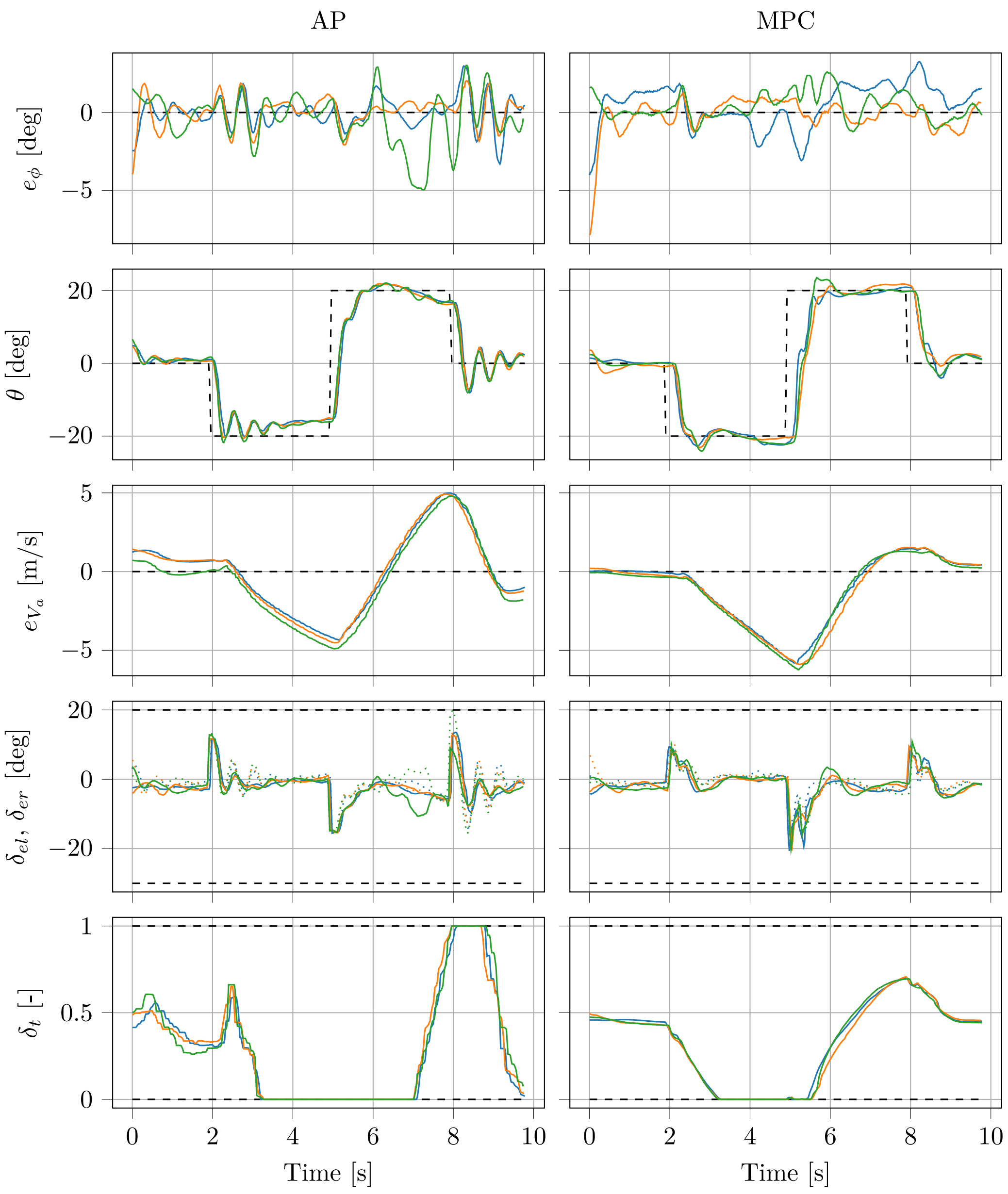

Figure B.3: Responses to pitch reference steps (dashed) for the MPC and the ArduPlane Controller. The colors encode different runs for the same controller.

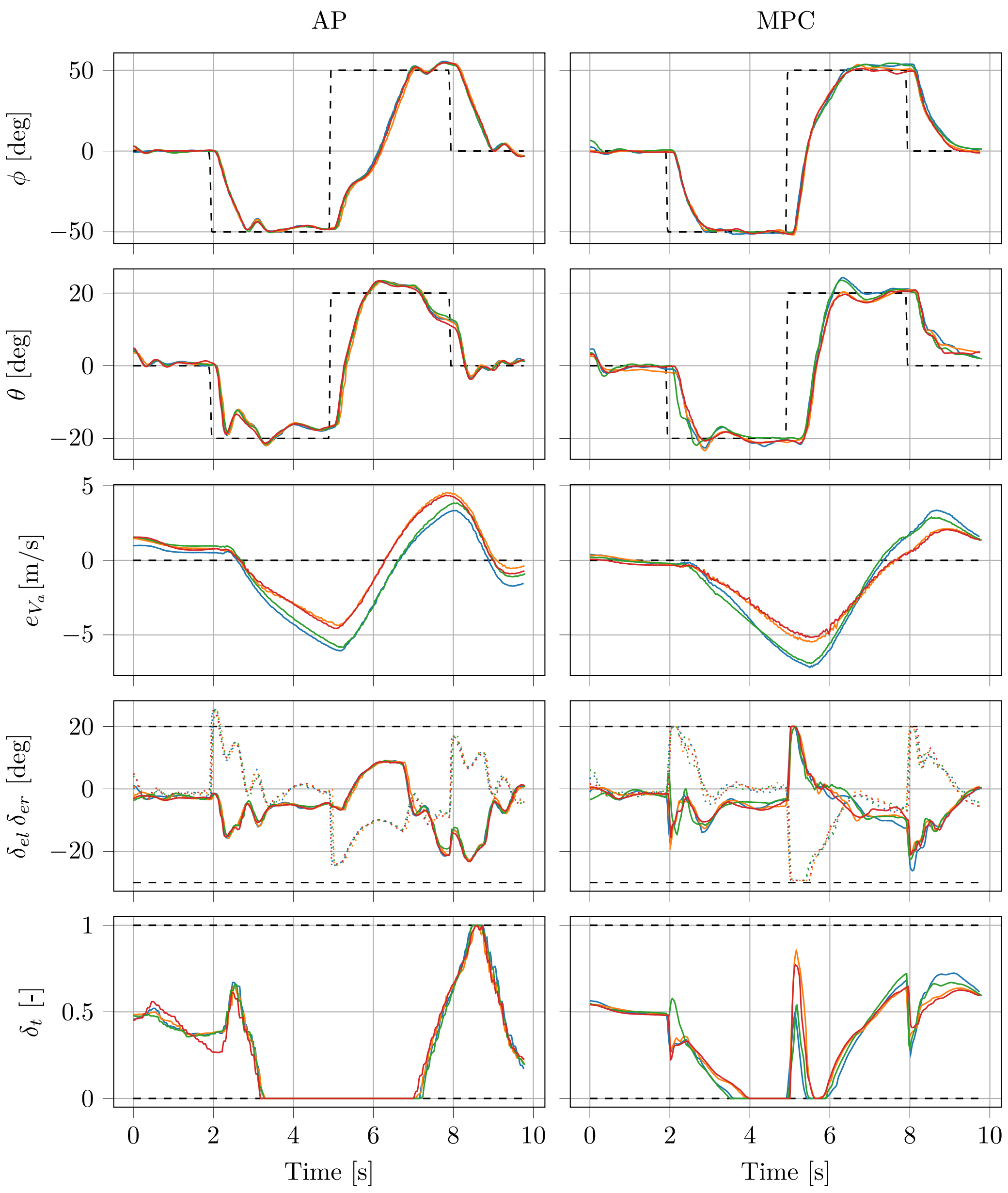

Figure B.4: Responses to a simultaneous step change in both roll and pitch angle reference (dashed) shown in the first two subplots.

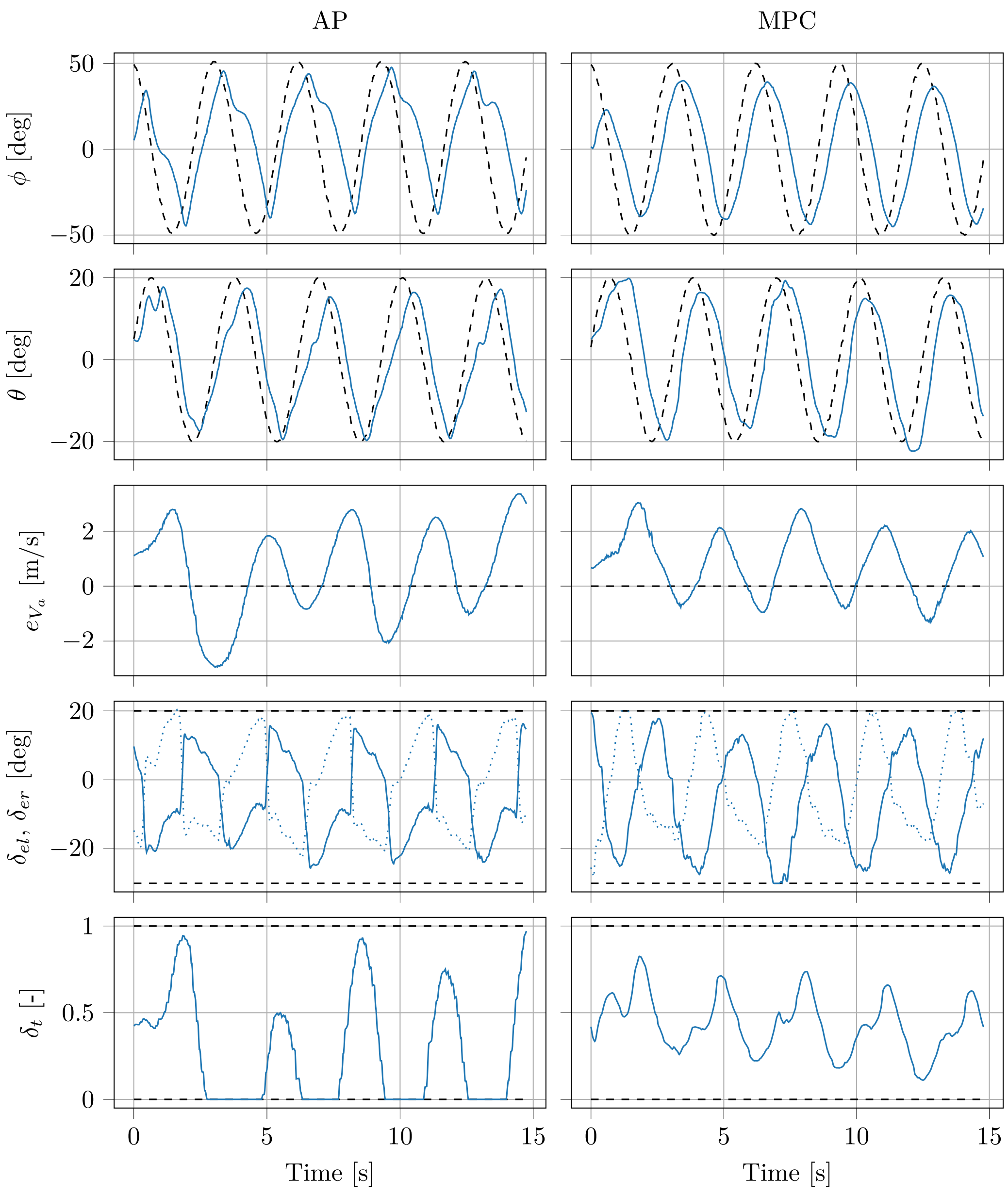

Figure B.5: Responses to simultaneously oscillating references (dashed) for roll and pitch.

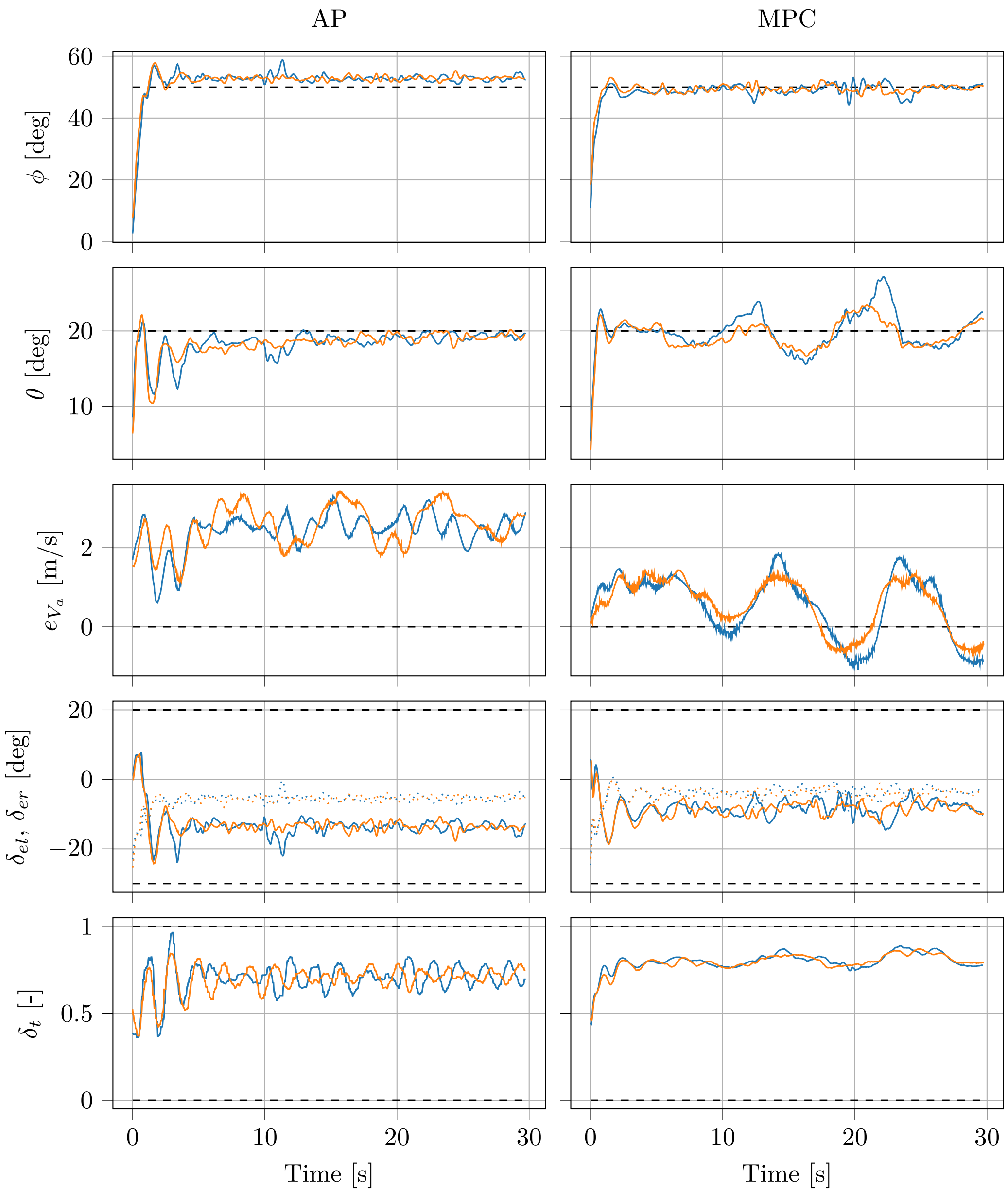

Figure B.6: Responses to a simultaneous reference step (dashed) for roll and pitch angle that would result in the UAV following a narrow upward spiral. The colors encode different runs for the same controller.

References

[1] Evan Ackerman and Eliza Strickland. Medical delivery drones take flight in east africa. *IEEE Spectrum*, 55(1):34–35, 2018.

[2] A. P. Aguiar and J. P. Hespanha. Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, Aug 2007.

[3] Andrea Alessandretti and A. Pedro Aguiar. A planar path-following model predictive controller for fixed-wing Unmanned Aerial Vehicles. In *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, pages 59–64, July 2017.

[4] Andrea Alessandretti, A. Pedro Aguiar, and Colin N. Jones. Trajectory-tracking and path-following controllers for constrained underactuated vehicles using Model Predictive Control. In *2013 European Control Conference (ECC)*, pages 1371–1376, July 2013.

[5] Kostas Alexis, Christos Papachristos, Roland Siegwart, and Anthony Tzes. Robust Model Predictive Flight Control of Unmanned Rotorcrafts. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 81(3-4):443–469, 2016.

[6] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

[7] ArduPilot open source drone software. `https://ardupilot.org/`. Accessed: 2021-12-03.

[8] Max Basescu and Joseph Moore. Direct NMPC for Post-Stall Motion Planning with Fixed-Wing UAVs. pages 9592–9598, 2020.

[9] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice.* Princeton University Press, 2012.

[10] Soulaimane Berkane, Abdelkader Abdessameud, and Abdelhamid Tayebi. Hybrid global exponential stabilization on SO(3). *Automatica*, 81:279–285, 2017.

[11] Sanjay P. Bhat and Dennis S. Bernstein. A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon. *System & Control Letters*, 39(1):63–70, 2000.

[12] Jean-Marc Biannic, Pierre Apkarian, and William L. Garrard. Parameter Varying Control of a High-Performance Aircraft. *Journal of Guidance, Control, and Dynamics*, 20(2):225–231, 1997.

[13] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[14] Eivind Bøhn, Erlend M Coates, Dirk Reinhardt, and Tor Arne Johansen. Data-Efficient Deep Reinforcement Learning for Attitude Control of Fixed-Wing UAVs: Field Experiments. *arXiv preprint arXiv:2111.04153, submitted to IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[15] Kasper T. Borup, Thor I. Fossen, and Tor A. Johansen. A Nonlinear Model-Based Wind Velocity Observer for Unmanned Aerial Vehicles. *IFAC-PapersOnLine*, 49(18):276 – 283, 2016. 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016.

[16] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[17] Eitan Bulka and Meyer Nahon. A unified control strategy for autonomous aerial vehicles. *Autonomous Robots*, 45(6):859–883, Sep 2021.

[18] Francesco Bullo. Stabilization of relative equilibria for underactuated systems on Riemannian manifolds. *Automatica*, 36(12):1819 – 1834, 2000.

[19] Francesco Bullo and Andrew D Lewis. *Geometric Control of Mechanical Systems*, volume 49 of *Texts in Applied Mathematics*. Springer-Verlag, 2005.

[20] Francesco Bullo and Richard M Murray. Tracking for fully actuated mechanical systems: a geometric framework. *Automatica*, 35(1):17–34, 1999.

[21] Franceso Bullo, Richard M. Murray, and Augusto Sarti. Control on the Sphere and Reduced Attitude Stabilization. *IFAC Proceedings Volumes*, 28(14):495–501, 1995. 3rd IFAC Symposium on Nonlinear Control Systems Design 1995, Tahoe City, CA, USA, 25-28 June 1995.

[22] Elisa Capello, Giorgio Guglieri, Paolo Marguerettaz, and Fulvia Quagliotti. Preliminary assessment of flying and handling qualities for mini-UAVs. *Journal of Intelligent & Robotic Systems*, 65(1):43–61, Jan 2012.

[23] Elisa Capello, Giorgio Guglieri, and Fulvia Quagliotti. A Waypoint-Based Guidance Algorithm for mini UAVs. *IFAC Proceedings Volumes*, 46(30):120–125, 2013. 2nd IFAC Workshop on Research, Education and Development of Unmanned Aerial Systems.

[24] Pedro Casau, Christopher G. Mayhew, Ricardo G. Sanfelice, and Carlos Silvestre. Robust global exponential stabilization on the n-dimensional sphere with applications to trajectory tracking for quadrotors. *Automatica*, 110:108534, 2019.

[25] Pedro Casau and Carlos Silvestre. Global asymptotic stabilization of spherical orientation by synergistic hybrid feedback with application to reduced attitude synchronization. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1536–1541. IEEE, 2018.

[26] Haiyang Chao, Yongcan Cao, and YangQuan Chen. Autopilots for Small Fixed-Wing Unmanned Air Vehicles: A Survey. In *2007 International Conference on Mechatronics and Automation*, pages 3144–3149, Aug 2007.

[27] N. Chaturvedi and H. McClamroch. Asymptotic Stabilization of the Inverted Equilibrium Manifold of the 3-D Pendulum Using Non-Smooth Feedback. *IEEE Transactions on Automatic Control*, 54(11):2658–2662, Nov 2009.

[28] Nalin Chaturvedi, Amit K Sanyal, and N Harris McClamroch. Rigid-Body Attitude Control. *IEEE Control Systems Magazine*, 31(3):30–51, 2011.

[29] Nalin A. Chaturvedi, Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Nonlinear dynamics of the 3D pendulum. *Journal of Nonlinear Science*, 21(1):3–32, 2011.

[30] Nalin A Chaturvedi, N Harris McClamroch, and Dennis S Bernstein. Asymptotic smooth stabilization of the inverted 3D pendulum. *IEEE Transactions on Automatic Control*, 54(6):1204–1215, 2009.

[31] H. Chen and F. Allgöwer. A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability. *Automatica*, 34(10):1205 – 1217, 1998.

[32] Namhoon Cho, Youdan Kim, and Sanghyuk Park. Three-Dimensional Nonlinear Differential Geometric Path-Following Guidance Law. *Journal of Guidance, Control, and Dynamics*, 38(12):2366–2385, 2015.

[33] Girish Chowdhary, Eric N. Johnson, Rajeev Chandramohan, M. Scott Kimbrell, and Anthony Calise. Guidance and Control of Airplanes Under Actuator Failures and Severe Structural Damage. *Journal of Guidance, Control, and Dynamics*, 36(4):1093–1104, 2013.

[34] Venanzio Cichella, Isaac Kaminer, Vladimir Dobrokhodov, Enric Xargay, Naira Hovakimyan, and Antonio Pascoal. Geometric 3D path-following control for a fixed-wing UAV on SO (3). In *AIAA Guidance, Navigation, and Control Conference*, page 6415, 2011.

[35] Erlend M. Coates and Thor I. Fossen. Geometric Reduced-Attitude Control of Fixed-Wing UAVs. *Applied Sciences*, 11(7), 2021.

[36] Erlend M. Coates, Jenny Bogen Griffiths, and Tor Arne Johansen. Robust Reduced-Attitude Control of Fixed-Wing UAVs Using a Generalized Multivariable Super-Twisting Algorithm. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 454–464, 2021.

[37] Erlend M. Coates, Dirk Reinhardt, and Thor I. Fossen. Reduced-Attitude Control of Fixed-Wing Unmanned Aerial Vehicles Using Geometric Methods on the Two-Sphere. *IFAC-PapersOnLine*, 53(2):5749–5756, 2020. 21st IFAC World Congress.

[38] Erlend M. Coates, Dirk Reinhardt, Kristoffer Gryte, and Tor Arne Johansen. Toward Nonlinear Flight Control for Fixed-Wing UAVs: System Architecture, Field Experiments, and Lessons Learned. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, accepted.

[39] Erlend M. Coates, Andreas Wenz, Kristoffer Gryte, and Tor Arne Johansen. Propulsion System Modeling for Small Fixed-Wing UAVs. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[40] Michael V Cook. *Flight dynamics principles: a linear systems approach to aircraft stability and control.* Butterworth-Heinemann, 2012.

[41] Rick Cory and Russ Tedrake. Experiments in Fixed-Wing UAV Perching. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2010.

[42] Dijana Damljanovi, Jovan Isakovi, and Boko Rauo. T-38 wind-tunnel data quality assurance based on testing of a standard model. *Journal of Aircraft*, 50(4):1141–1149, 2013.

[43] Johann C. Dauer, Timm Faulwasser, Sven Lorenz, and Rolf Findeisen. *Optimization-based Feedforward Path Following for Model Reference Adaptive Control of an Unmanned Helicopter.*

[44] Brian de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, 2020.

[45] Stefano Di Cairano and Ilya V. Kolmanovsky. Real-time optimization and model predictive control for aerospace and automotive applications. In *2018 Annual American Control Conference (ACC)*, pages 2392–2409, 2018.

[46] Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*, pages 391–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[47] Sanket Diwale, Timm Faulwasser, and Colin N. Jones. Model Predictive Path-Following Control for Airborne Wind Energy Systems. *IFAC-PapersOnLine*, 50(1):13270–13275, 2017. 20th IFAC World Congress.

[48] Richard C.. Dorf and Robert H Bishop. *Modern Control Systems.* Pearson Prentice Hall, 2008.

[49] Emad Ebeid, Martin Skriver, Kristian Husum Terkildsen, Kjeld Jensen, and Ulrik Pagh Schultz. A survey of open-source UAV flight controllers and flight simulators. *Microprocessors and Microsystems*, 61:11–20, 2018.

[50] Utku Eren, Anna Prach, Baaran Bahadr Koçer, Saša V. Raković, Erdal Kayacan, and Behçet Açkmee. Model Predictive Control in Aerospace Systems: Current State and Opportunities. *Journal of Guidance, Control, and Dynamics*, 40(7):1541–1566, 2017.

[51] Bernard Etkin. *Dynamics of atmospheric flight.* Courier Corporation, 2012.

[52] Joonyup Eun, Byung Duk Song, Sangbok Lee, and Dae-Eun Lim. Mathematical investigation on the sustainability of uav logistics. *Sustainability*, 11(21), 2019.

[53] Jay Farrell. *Aided navigation: GPS with high rate sensors.* McGraw-Hill, Inc., 2008.

[54] T. Faulwasser and R. Findeisen. Nonlinear Model Predictive Control for Constrained Output Path Following. *IEEE Transactions on Automatic Control*, 61(4):1026–1039, April 2016.

[55] Timm Faulwasser, Janine Matschek, Pablo Zometa, and Rolf Findeisen. Predictive path-following control: Concept and implementation for an industrial robot. In *2013 IEEE International Conference on Control Applications (CCA)*, pages 128–133, Aug 2013.

[56] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control.* John Wiley & Sons, 2011.

[57] Gianluca Frison and Moritz Diehl. HPIPM: a high-performance quadratic programming framework for model predictive control. arXiv, 2020.

[58] Gianluca Frison, Dimitris Kouzoupis, Tommaso Sartor, Andrea Zanelli, and Moritz Diehl. BLASFEO. *ACM Transactions on Mathematical Software*, 44(4):130, Aug 2018.

[59] Luca Furieri, Thomas Stastny, Lorenzo Marconi, Roland Siegwart, and Igor Gilitschenski. Gone with the wind: Nonlinear guidance for small fixed-wing aircraft in arbitrarily strong windfields. In *2017 American Control Conference (ACC)*, pages 4254–4261, 2017.

[60] Ze Feng Ted Gan, Jiajun Kevin Feng, Fidel Khouli, Mostafa SA ElSayed, and Fred Nitzsche. Development and optimization of flight dynamics, control laws and avionics system for a uav with a multi-scale optimized blended wing body configuration. *CASI Aero19*, 2019.

[61] Gonzalo Garcia and Shahriar Keshmiri. Nonlinear model predictive controller for navigation, guidance and control of a fixed-wing uav. In *AIAA Guidance, Navigation, and Control Conference*, 2012.

[62] Francisco Gavilan, Rafael Vazquez, and Eduardo F Camacho. An iterative model predictive control algorithm for UAV guidance. *IEEE Transactions on Aerospace and Electronic Systems*, 51(3):2406–2419, 2015.

[63] Chowdhary Vinayak Girish, Frazzoli Emilio, How P. Jonathan, and Liu Hugh. *Nonlinear Flight Control Techniques for Unmanned Aerial Vehicles*, pages 577–612. Springer Netherlands, Dordrecht, 2015.

[64] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness.* Princeton University Press, 2012.

[65] Jared A. Grauer and Eugene A. Morelli. Generic Global Aerodynamic Model for Aircraft. *Journal of Aircraft*, 52(1):13–20, 2015.

[66] S. Gros, R. Quirynen, and M. Diehl. Aircraft control based on fast non-linear MPC & multiple-shooting. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1142–1147, Dec 2012.

[67] Sébastien Gros and Mario Zanon. Data-Driven Economic NMPC Using Reinforcement Learning. 65:636–648, 2020.

[68] Lars Grüne and Juergen Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms.* Springer International Publishing, 2017.

[69] Kristoffer Gryte. High angle of attack landing of an unmanned aerial vehicle. Master's thesis, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, 2015.

[70] Kristoffer Gryte. *Precision control of fixed-wing UAV and robust navigation in GNSS-denied environments.* PhD thesis, Norwegian University of Science and Technology (NTNU), June 2020.

[71] Kristoffer Gryte, Richard Hann, Mushfiqul Alam, Jan Rohác, Tor Arne Johansen, and Thor I Fossen. Aerodynamic modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle. *International Conference of Unmanned Aerial Systems*, 2018.

[72] John Guckenheimer and Philip Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* Springer-Verlag, New York, NY, 1983.

[73] Rohit Gupta, Uros V. Kalabic, Stefano Di Cairano, Anthony M. Bloch, and Ilya V. Kolmanovsky. Constrained spacecraft attitude control on SO(3) using fast nonlinear model predictive control. *Proceedings of the American Control Conference*, 2015-July:2980–2986, 2015.

[74] Torleiv Haaland Bryne. *Nonlinear Observer Design for Aided Inertial Navigation of Ships*. PhD thesis, Norwegian University of Science and Technology, 2017.

[75] T. Hamel and R. Mahony. Attitude estimation on SO(3) based on direct inertial measurements. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2170–2175, 2006.

[76] L. Hewing, J. Kabzan, and M. N. Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, pages 1–8, 2019.

[77] L. Hewing, A. Liniger, and M. N. Zeilinger. Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars. In *2018 European Control Conference (ECC)*, pages 1341–1348, June 2018.

[78] Hanno Hildmann and Ernö Kovacs. Review: Using Unmanned Aerial Vehicles (UAVs) as Mobile Sensing Platforms (MSPs) for Disaster Response, Civil Security and Public Safety. *Drones*, 3(3), 2019.

[79] J. Holsten, T. Ostermann, and D. Moormann. Design and wind tunnel tests of a tiltwing uav. *CEAS Aeronautical Journal*, 2(1):69–79, Dec 2011.

[80] Jonathan P. How, Emilio Frazzoli, and Girish Vinayak Chowdhary. *Linear Flight Control Techniques for Unmanned Aerial Vehicles*, pages 529–576. Springer Netherlands, Dordrecht, 2015.

[81] Richard A Hyde. *H-infinity aerospace control design: a VSTOL flight application*. Springer Science & Business Media, 2013.

[82] Ravindra V. Jategaonkar. *Flight Vehicle System Identification : A Time-domain Methodology*. AIAA, 2 edition, 2015.

[83] Chung Jindeog, Lee Jangyeon, Sung Bongzoo, and Koo Samok. Wind tunnel test of an unmanned aerial vehicle (uav). *KSME international journal*, 17(5):776–783, 2003.

[84] Tor A Johansen. Toward dependable embedded model predictive control. *IEEE Systems Journal*, 11(2):1208–1219, 2017.

[85] Tor A. Johansen, Andrea Cristofaro, Kim Sorensen, Jakob M. Hansen, and Thor I. Fossen. On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors. *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pages 510–519, 2015.

[86] Tor A Johansen, Artur Zolich, Torkel Hansen, and Asgeir J. Sorensen. Unmanned aerial vehicle as communication relay for autonomous underwater vehicle - Field tests. *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 1469–1474, 2014.

[87] J.-M. Kai, T. Hamel, and C. Samson. A unified approach to fixed-wing aircraft path following guidance and control. *Automatica*, 108:108491, 2019.

[88] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2219):20180335, 2018.

[89] Uroš Kalabić, Rohit Gupta, Stefano Di Cairano, Anthony Bloch, and Ilya Kolmanovsky. MPC on Manifolds with an Application to SE (3). In *2016 American Control Conference (ACC)*, pages 7–12. IEEE, 2016.

[90] Uroš V. Kalabić, Rohit Gupta, Stefano Di Cairano, Anthony M. Bloch, and Ilya V. Kolmanovsky. MPC on manifolds with an application to the control of spacecraft attitude on SO(3). *Automatica*, 76:293–300, 2017.

[91] Mina Kamel, Kostas Alexis, Markus Achtelik, and Roland Siegwart. Fast nonlinear model predictive control for multicopter attitude tracking on SO(3). In *2015 IEEE Conference on Control Applications (CCA)*, pages 1160–1166, 2015.

[92] Mina Kamel, Michael Burri, and Roland Siegwart. Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles. *IFAC-PapersOnLine*, 50(1):3463–3469, 2017.

[93] Mina Kamel, Thomas Stastny, Kostas Alexis, and Roland Siegwart. Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In *Robot Operating System (ROS)*, pages 3–39. Springer, 2017.

[94] Konstantinos Kanistras, Goncalo Martins, Matthew J. Rutherford, and Kimon P. Valavanis. Survey of unmanned aerial vehicles (uavs) for traffic monitoring. In *Handbook of Unmanned Aerial Vehicles.* International Conference of Unmanned Aerial Systems, 2013.

[95] Hassan K Khalil. *Nonlinear Systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[96] Waqas Khan. *Dynamics Modeling of Agile Fixed-Wing Unmanned Aerial Vehicles.* PhD thesis, McGill University, 2016.

[97] Vladislav Klein. *Aircraft System Identification - Theory and Practice.* AIAA, 2006.

[98] D K Kufoalor and Tor Arne Johansen. Reconfigurable fault tolerant flight control based on nonlinear model predictive control. In *American Control Conference (ACC), 2013*, pages 5128–5133. IEEE, 2013.

[99] J. Köhler, M. A. Müller, and F. Allgöwer. A novel constraint tightening approach for nonlinear robust model predictive control. In *2018 Annual American Control Conference (ACC)*, pages 728–734, June 2018.

[100] Pawel Ladosz, Matthew Coombes, Jean Smith, and Michael Hutchinson. *A Generic ROS Based System for Rapid Development and Testing of Algorithms for Autonomous Ground and Aerial Vehicles*, pages 113–153. Springer International Publishing, Cham, 2019.

[101] A. A Lambregts. Vertical Flight Path and Speed Control Autopilot Design Using Total Energy Principles. In *Aiaa Guid. Control. Conf*, 1983.

[102] S. H. Lane and R. F. Stengel. Flight control design using non-linear inverse dynamics. *Automatica*, 24(4):471–483, 1988.

[103] Kevin A Lavretsky Eugene; Wise. *Robust and Adaptive Control With Aerospace Applications*. Springer, 2013.

[104] Chirl Hwa Lee, Myoung Ho Shin, and Myung Jin Chung. A design of gain-scheduled control for a linear parameter varying system: an application to flight control. *Control Engineering Practice*, 9(1):11–21, 2001.

[105] Dae Young Lee, Rohit Gupta, Uros V. Kalabic, Stefano Di Cairano, Anthony M. Bloch, James W. Cutler, and Ilya V. Kolmanovsky. Constrained Attitude Maneuvering of a Spacecraft with Reaction Wheel Assembly by Nonlinear Model Predictive Control. In *2016 American Control Conference (ACC) Boston Marriott Copley Place July 6-8, 2016. Boston, MA, USA*, 2016.

[106] T. Lee, M. Leok, and N. H. McClamroch. Stable manifolds of saddle equilibria for pendulum dynamics on S2 and SO(3). In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 3915–3921. IEEE, 2011.

[107] Taeyoung Lee. Global Exponential Attitude Tracking Controls on SO(3). *IEEE Transactions on Automatic Control*, 60(10):2837–2842, 2015.

[108] Taeyoung Lee. Optimal Hybrid Controls for Global Exponential Tracking on the Two-Sphere. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3331–3337, 2016.

[109] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. *Global Formulations of Lagrangian and Hamiltonian Dynamics on Manifolds*. Springer, 2017.

[110] Frederik S. Leira, Tor Arne Johansen, and Thor I. Fossen. Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera. In *2015 IEEE Aerospace Conference*, pages 1–10, 2015.

[111] A. Levant, A. Pridor, R. Gitizadeh, I. Yaesh, and J. Z. Ben-Asher. Aircraft Pitch Control via Second-Order Sliding Technique. *Journal of Guidance, Control, and Dynamics*, 23(4):586–594, 2000.

[112] Giovanni Licitra, Adrian Bürger, Paul Williams, Richard Ruiterkamp, and Moritz Diehl. Aerodynamic model identification of an autonomous aircraft for airborne wind energy. *Optimal Control Applications and Methods*, 2019.

[113] Jakob Mahler Hansen. *Nonlinear Observers for Inertial Navigation Systems aided by Real- Time Kinematic Global Satellite Navigation* . PhD thesis, The Norwegian University of Science and Technology, 2017.

[114] Robert Mahony, Tarek Hamel, and Jean-Michel Pflimilin. Nonlinear Complementary Filters on the Special Orthogonal Group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, 2008.

[115] Martina Mammarella and Elisa Capello. A Tube-based Robust MPC for a Fixed-wing UAV: an Application for Precision Farming. *arXiv preprint arXiv:1805.04295.*

[116] Martina Mammarella and Elisa Capello. A Robust MPC-based autopilot for mini UAVs. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1227–1235, June 2018.

[117] Martina Mammarella and Elisa Capello. Tube-based robust mpc processor-in-the-loop validation for fixed-wing uavs. *Journal of Intelligent & Robotic Systems*, pages 1–20, 2020.

[118] Martina Mammarella, Elisa Capello, Fabrizio Dabbene, and Giorgio Guglieri. Sample-based SMPC for tracking control of fixed-wing UAV. *IEEE Control Systems Letters*, 2018.

[119] F. L. Markley and J. L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control.* Springer-Verlag New York, 2014.

[120] I. Masubuchi, J. Kato, M. Saeki, and A. Ohara. Gain-scheduled controller design based on descriptor representation of LPV systems: application to flight vehicle control. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, volume 1, pages 815–820 Vol.1, 2004.

[121] Siri Mathisen, Kristoffer Gryte, Sebastien Gros, and Tor Arne Johansen. Precision Deep-Stall Landing of Fixed-Wing UAVs Using Nonlinear Model Predictive Control. *Journal of Intelligent & Robotic Systems*, 101(1):24, Dec 2020.

[122] Janine Matschek, Tobias Bäthge, Timm Faulwasser, and Rolf Findeisen. *Nonlinear Predictive Control for Trajectory Tracking and Path Following: An Introduction and Perspective*, pages 169–198. Springer International Publishing, Cham, 2019.

[123] Justin J Matt, Steven G Hagerott, Benjamin C Svoboda, Haiyang Chao, and Harold P Flanagan. Frequency domain system identification of a small flying-wing uas. In *AIAA SCITECH 2022 Forum*, page 2407, 2022.

[124] Christopher G Mayhew, Ricardo G Sanfelice, and Andrew R Teel. Quaternion-Based Hybrid Control for Robust Global Attitude Tracking. *IEEE Transactions on Automatic Control*, 56(11):2555–2566, 2011.

[125] Christopher G. Mayhew and Andrew R. Teel. Synergistic potential functions for hybrid control of rigid-body attitude. *Proceedings of the 2011 American Control Conference*, pages 875–880, 2011.

[126] Christopher G. Mayhew and Andrew R. Teel. Global stabilization of spherical orientation by synergistic hybrid feedback with application to reduced-attitude tracking for rigid bodies. *Automatica*, 49(7):1945–1957, 2013.

[127] D. McFarlane and K. Glover. A loop-shaping design procedure using H-∞ synthesis. *IEEE Transactions on Automatic Control*, 37(6):759–769, June 1992.

[128] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240, 2015.

[129] Mariann Merz and Tor Arne Johansen. Feasibility study of a circularly towed cable-body system for UAV applications. *2016 International Conference on Unmanned Aircraft Systems, ICUAS 2016*, pages 1182–1191, 2016.

[130] Michail G. Michailidis, Konstantinos Kanistras, Mohammed Agha, Matthew J. Rutherford, and Kimon P. Valavanis. Nonlinear Control of Fixed-Wing UAVs with Time-Varying Aerodynamic Uncertainties Via μ-Synthesis. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6314–6321, 2018.

[131] Rahman Mohammadi Farhadi, Vyacheslav Kortunov, Andrii Molchanov, Tatiana Solianyk, et al. Estimation of the lateral aerodynamic coefficients for skywalker x8 flying wing from real flighttest data. 2018.

[132] Giulio Molinari, Andres F. Arrieta, Michel Guillaume, and Paolo Ermanni. Aerostructural performance of distributed compliance morphing wings: Wind tunnel and flight testing. *AIAA Journal*, 54(12):3859–3871, 2016.

[133] Joseph Moore, Rick Cory, and Russ Tedrake. Robust post-stall perching with a simple fixed-wing glider using LQR-Trees. *Bioinspiration & biomimetics*, 9(2):025013, 2014.

[134] Manfred Morari and Urban Maeder. Nonlinear offset-free model predictive control. *Automatica*, 48(9):2059–2067, 2012.

[135] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, Young-Han Nam, and Mérouane Debbah. A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems. *IEEE Communications Surveys Tutorials*, 21(3):2334–2360, 2019.

[136] Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing Action Policies for Smooth Control with Reinforcement Learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1810–1816, 2021.

[137] Derek R Nelson, D Blake Barber, Timothy W McLain, and Randal W Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007.

[138] Robert C Nelson et al. *Flight stability and automatic control*, volume 2. WCB/McGraw Hill New York, 1998.

[139] Michael Neunert, Cédric De Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1398–1404, 2016.

[140] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, July 2018.

[141] Francesco Nex and Fabio Remondino. UAV for 3D mapping applications: a review. *Applied geomatics*, 6(1):1–15, 2014.

[142] Goran Ocokoljic, Dijana Damljanovic, De Vukovic, and Boško Rašuo. Contemporary frame of measurement and assessment of wind-tunnel flow quality in a low-speed facility. *FME Transactions*, 46(4):429–442, 2018.

[143] P. Oettershagen, A. Melzer, S. Leutenegger, K. Alexis, and R. Siegwart. Explicit model predictive control and L1-navigation strategies for fixed-wing UAV path tracking. In *22nd Mediterranean Conference on Control and Automation*, pages 1159–1165, June 2014.

[144] Michael Ol, Cale Zeune, Trenton White, and Thomas Kudla. Wind tunnel evaluation of powered static aerodynamics of an aerobatic uav. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013.

[145] Espen Oland. *Nonlinear Control of Fixed-Wing Unmanned Aerial Vehicles*. PhD thesis, The Arctic University of Norway, 2014.

[146] Sanghyuk Park. Autonomous aerobatics on commanded path. *Aerospace Science and Technology*, 22(1):64–74, 2012.

[147] Sanghyuk Park, John Deyst, and Jonathan How. A New Nonlinear Guidance Logic for Trajectory Tracking. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–16, 2004.

[148] Jose Pinto, Paulo S. Dias, Ricardo Martins, Joao Fortuna, Eduardo Marques, and Joao Sousa. The LSTS toolchain for networked vehicle systems. *OCEANS 2013 MTS/IEEE Bergen: The Challenges of the Northern Dimension*, 2013.

[149] Panagiotis Radoglou-Grammatikis, Panagiotis Sarigiannidis, Thomas Lagkas, and Ioannis Moscholios. A compilation of UAV applications for precision agriculture. *Computer Networks*, 172:107148, 2020.

[150] Michalis Ramp and Evangelos Papadopoulos. Attitude and angular velocity tracking for a rigid body using geometric methods on the two-sphere. *2015 European Control Conference, ECC 2015*, 2(1):3238–3243, 2015.

[151] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[152] Dirk Reinhardt, E. M. Coates, and Tor Arne Johansen. Low-level Nonlinear Model Predictive Attitude and Speed Control of Fixed-Wing Unmanned Aerial Vehicles. *Control Engineering Practice*, submitted.

[153] Dirk Reinhardt, Erlend. M. Coates, and Tor Arne Johansen. Hybrid Control of Fixed-Wing UAVs for Large-Angle Attitude Maneuvers on the Two-Sphere. *IFAC-PapersOnLine*, 53(2):5717–5724, 2020. 21st IFAC World Congress.

[154] Dirk Reinhardt, Kristoffer Gryte, and Tor Arne Johansen. Modeling of the Skywalker X8 Fixed-Wing UAV: Flight Tests and System Identification. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, accepted.

[155] Dirk Reinhardt and Tor Arne Johansen. Nonlinear Model Predictive Attitude Control for Fixed-Wing Unmanned Aerial Vehicle based on a Wind Frame Formulation. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 503–512, June 2019.

[156] Dirk Reinhardt and Tor Arne Johansen. Control of Fixed-Wing UAV Attitude and Speed based on Embedded Nonlinear Model Predictive Control. *IFAC-PapersOnLine*, 54(6):91–98, 2021. 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021.

[157] Dirk Reinhardt and Tor Arne Johansen. Nonlinear Model Predictive Control combined with Geometric Attitude and Speed Control for Fixed-Wing UAVs. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 465–475, 2021.

[158] Dirk Reinhardt, Morten D. Pedersen, Kristoffer Gryte, and Tor Arne Johansen. A Symmetry Calibration Procedure for Sensor-to-Airframe Misalignments in Wind Tunnel Data. In *2022 Conference on Control Technologies and Applications (CCTA)*, accepted.

[159] David Rohr, Matthias Studiger, Thomas Stastny, Nicholas R. J. Lawrance, and Roland Y. Siegwart. Nonlinear Model Predictive Velocity Control of a VTOL Tiltwing UAV. *IEEE Robotics and Automation Letters*, 6:5776–5783, 2021.

[160] Wilson J. Rugh and Jeff S. Shamma. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

[161] Rolf Rysdyk. Course and heading changes in significant wind. *Journal of guidance, control, and dynamics*, 30(4):1168–1171, 2007.

[162] Searchwing.org a model aircraft to safe lives, 2020. `https://sea-watch.org/en/searchwing-org-a-model-aircraft-to-safe-lives/`.

[163] Michael Selig. Modeling Full-Envelope Aerodynamics of Small UAVs in Realtime. In *AIAA Atmospheric Flight Mechanics Conference*, page 7635, 2010.

[164] Ilya A. Shkolnikov and Yuri B. Shtessel. Aircraft Nonminimum Phase Control in Dynamic Sliding Manifolds. *Journal of Guidance, Control, and Dynamics*, 24(3):566–572, 2001.

[165] Yuri Shtessel, Christopher Edwards, Leonid Fridman, and Arie Levant. *Sliding mode control and observation*, volume 10. Springer, 2014.

[166] Yuri Shtessel, Mohammed Taleb, and Franck Plestan. A novel adaptive-gain supertwisting sliding mode controller: Methodology and application. *Automatica*, 48(5):759–769, 2012.

[167] Benjamin M. Simmons, Hunter G. McClelland, and Craig A. Woolsey. Nonlinear model identification methodology for small, fixed-wing, unmanned aircraft. *Journal of Aircraft*, 56(3):1056–1067, 2019.

[168] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Citeseer, 2007.

[169] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, volume 1. Prentice hall Englewood Cliffs, NJ, 1991.

[170] G Allan Smith and George Meyer. Aircraft automatic flight control system with model inversion. *Journal of Guidance, Control, and Dynamics*, 10(3):269–275, 1987.

[171] Frantisek Sobolic and Jonathan How. Nonlinear Agile Control Test Bed for a Fixed Wing Aircraft in a Constrained Environment. In *AIAA Infotech@Aerospace Conference*, 2009.

[172] Joan Sola. Quaternion kinematics for the error-state Kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[173] L Sonneveldt. Adaptive backstepping flight control for modern fighter aircraft. *... doctoral dissertation, Delf ...*, pages 1–296, 2010.

[174] Thomas Stastny and Roland Siegwart. Nonlinear Model Predictive Guidance for Fixed-wing UAVs Using Identified Control Augmented Dynamics. *2018 International Conference on Unmanned Aircraft Systems, ICUAS 2018*, pages 432–442, 2018.

[175] Thomas J Stastny, Adyasha Dash, and Roland Siegwart. Nonlinear mpc for fixed-wing uav trajectory tracking: Implementation and flight experiments. In *AIAA Guidance, Navigation, and Control Conference*, page 1512, 2017.

[176] Robert F Stengel. *Flight dynamics*. Princeton University Press, 2015.

[177] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[178] P. B. Sujit, Srikanth Saripalli, and Joao Borges Sousa. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems*, 34(1):42–59, 2014.

[179] Russ Tedrake. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832), 2020.

[180] Russ Tedrake, Ian R. Manchester, Mark Tobenkin, and John W. Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[181] Andrew R Teel, Fulvio Forni, and Luca Zaccarian. Lyapunov-Based Sufficient Conditions for Exponential Stability in Hybrid Systems. *IEEE Transactions on Automatic Control*, 58(6):1591–1596, 2013.

[182] Mark B Tischler and Robert K Remple. *Aircraft and rotorcraft system identification*. American Institute of Aeronautics and Astronautics Reston, VA, 2012.

[183] Johannes Tjønnås and Tor A. Johansen. Adaptive control allocation. *Automatica*, 44(11):2754–2765, 2008.

[184] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza. Data-Driven MPC for Quadrotors. *IEEE Robotics and Automation Letters*, pages 1–1, 2021.

[185] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Jonathan Frey, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados: a modular open-source framework for fast embedded optimal control. *arXiv preprint*, 2019.

[186] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[187] Grace Wahba. A least squares estimate of satellite attitude. *SIAM Review*, 7(3):409–409, 1965.

[188] J.T.-Y. Wen and K. Kreutz-Delgado. The attitude control problem. *IEEE Transactions on Automatic Control*, 36(10):1148–1162, Oct 1991.

[189] A. Wenz and T. A. Johansen. Real-Time Moving Horizon Estimation of Air Data Parameters and Wind Velocities for fixed-wing UAVs. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 998–1006, Sep. 2020.

[190] Andreas Wenz and Tor Arne Johansen. Estimation of wind velocities and aerodynamic coefficients for UAVs using standard autopilot sensors and a Moving Horizon Estimator. *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, pages 1267–1276, 2017.

[191] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[192] Adrian Winter, Richard Hann, Andreas Wenz, Kristoffer Gryte, and Tor Arne Johansen. Stability of a flying wing uav in icing conditions. In *8th European Conference for Aeronautics and Space Sciences (EUCASS), Madrid*, 2019.

[193] M. Wrzos-Kaminska, K.Y. Pettersen, and J.T. Gravdahl. Path following control for articulated intervention-AUVs using geometric control of reduced attitude. *IFAC-PapersOnLine*, 52(16):192–197, 2019.

[194] Jun Yang, Cunjia Liu, Matthew Coombes, Yunda Yan, and Wen-Hua Chen. Optimal Path Following for Small Fixed-Wing UAVs Under Wind Disturbances. *IEEE Transactions on Control Systems Technology*, 29(3):996–1008, May 2021.

[195] Kwangjin Yang and Salah Sukkarieh. An Analytical Continuous-Curvature Path-Smoothing Algorithm. *IEEE Transactions on Robotics*, 26(3):561–568, 2010.

[196] Kwangjin Yang, Salah Sukkarieh, and Yeonsik Kang. Adaptive nonlinear model predictive path tracking control for a fixed-wing unmanned aerial vehicle. In *AIAA Guidance, Navigation, and Control Conference*, page 5622, 2009.

[197] Andrea Zanelli. Nonlinear Model Predictive Control of a Humand-sized Quadrotor. *2018 European Control Conference (ECC)*, pages 1542–1547, 2018.

[198] Chunhua Zhang and John M. Kovacs. The application of small unmanned aerial systems for precision agriculture: a review. *Precision Agriculture*, 13(6):693–712, Dec 2012.

[199] Artur Zolich, Tor Arne Johansen, Krzysztof Cisek, and Kristian Klausen. Unmanned aerial system architecture for maritime missions. design & hardware description. *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems, RED-UAS 2015*, pages 342–350, 2016.

ISBN 978-82-326-6666-9 (printed ver.)
ISBN 978-82-326-6479-5 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)