Doctoral theses at NTNU, 2023:265

Erlend Magnus Lervik Coates

Nonlinear Attitude and Path-Following Control of Fixed-Wing Aircraft

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

Erlend Magnus Lervik Coates

Nonlinear Attitude and Path-Following Control of Fixed-Wing Aircraft

Thesis for the Degree of Philosophiae Doctor

Trondheim, September 2023

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

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

ISBN 978-82-326-7230-1 (printed ver.)
ISBN 978-82-326-7229-5 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

ITK-report 2023-19-W

Doctoral theses at NTNU, 2023:265

Printed by NTNU Grafisk senter

To Amalie, Ellinor & Oskar.

Summary

The last decade has seen an increased use of small fixed-wing unmanned aerial vehicles (UAVs) in a multitude of civil, commercial and scientific applications. Autopilots are automatic control systems that provide the basic low-level control functions that maintain aircraft attitude, speed, altitude and course heading. As we move towards higher levels of automation, robust and reliable autopilots are needed to enable safe and efficient autonomous operation of UAVs. Conventional autopilots are designed using linear control theory to operate close to some nominal operating conditions. However, the underlying physics are highly nonlinear, causing performance deterioration and, possibly, instability when approaching the edge of the flight envelope.

This thesis focuses on applying state-of-the-art nonlinear control algorithms to develop autopilots that can provide fixed-wing UAVs with the capability to operate in extended flight envelopes, with more aggressive manoeuvres, and in a wider range of weather conditions. Although most of the results of this thesis are valid for fixed-wing aircraft in general, simulation studies and experimental trials are carried out for fixed-wing UAVs.

The thesis contributions are presented in three parts. Part I of the thesis develops novel geometric reduced-attitude controllers, particularly well-suited for fixed-wing aircraft performing banked-turn manoeuvres. Instead of using roll and pitch (Euler) angles, a global, singularity-free representation on the unit two-sphere is used that is independent of the yaw/heading angle of the aircraft and allows for geodesic (shortest path) rotations. The reduced-attitude representation lets us decouple 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). Posing the control problem on the sphere has the advantage of opening up a wide range of tools for control systems on spheres, including different choices of potential functions and methods for global stabilization and tracking using hybrid control. Almost-global and global asymptotic stability is proven using Lyapunov methods for smooth and hybrid controllers, respectively. Adaptive and robust variants of the control design are developed to account for uncertain aerodynamic effects and turbulent wind disturbances. The control performance is demonstrated in MATLAB simulations as well as using more realistic software-in-the-loop simulations.

In part II, we revisit the path-following guidance problem in three dimensions. First, by formulating the path-following error directly in the inertial frame, we propose a class of guidance laws for regular parameterized paths that, unlike most approaches existing in the literature, do not require the explicit construction of a path frame. Based on an inner-outer loop control paradigm, the guidance law generates a normal acceleration command that is normal to the flow-relative velocity vector (as the lift force). This allows for a natural decomposition of the desired vehicle acceleration for aerial vehicles in coordinated turns: tangential acceleration for airspeed control and normal acceleration for guidance generated through bank-to-turn manoeuvres, i.e., by tilting the lift vector. By using cascade arguments, we show that the proposed design leads to almost global stability results and thus relaxes the set of feasible initial conditions compared to existing methods. The efficacy of the proposed guidance law is demonstrated in a simulation study.

In part III, we explore the use of deep reinforcement learning (DRL) for attitude control of UAVs. We show that DRL can successfully learn to perform attitude control of fixed-wing UAVs, requiring as little as three minutes of flight data. The proposed method is based on the Soft Actor-Critic algorithm and improves upon the data efficiency of the existing literature by at least an order of magnitude, providing an important step towards enabling the learning of reinforcement learning controllers entirely on the real UAV. We initially train our model in a simulation environment and then deploy the learned controller on the UAV in flight tests, demonstrating comparable performance to the state-of-the-art ArduPlane PID attitude controller with no further online learning required.

Preface

This thesis is submitted in partial fulfilment of the requirements for the degree of Philosophiae Doctor (PhD) at the Norwegian University of Science and Technology (NTNU). I carried out the work at the Department of Engineering Cybernetics under the supervision of main supervisor Professor Thor Inge Fossen and co-supervisor Professor Tor Arne Johansen. The Research Council of Norway has co-funded this work through the Centres of Excellence funding scheme with grant number 223254 (NTNU AMOS) and the IKTPLUSS funding scheme with grant number 261791 (AutoFly).

Acknowledgments

This thesis is the result of collaboration and support from a long list of people. Thor Inge and Tor Arne, thank you for letting me dive into the fascinating field of flight control. You are both a great inspiration, and your solution-oriented mindset, timely feedback and wholehearted support helped me through these years.

Dirk, thanks for holding out with me in D447. I appreciate our long days at the lab and all the field trips to Breivika. It's not about the destination but the journey. Big thanks to Eivind for our fruitful collaboration on DRL. The experimental parts of this thesis would not have been possible without Kristoffer, Martin and Pål at the UAV-Lab, who contributed with their knowledge and experience on the LSTS codebase, fieldwork and piloting.

During my PhD, I was lucky enough to enjoy several short research trips to France. Tarek and Antonio, I appreciate our discussions. Your insight has helped me better navigate the realms of applied nonlinear control.

Thanks to the Sevendof team for introducing me to unmanned aircraft, in particular, Glenn, who got me onboard and with whom I also shared several years at ITK. A big shoutout goes to all colleagues at ITK and NTNU AMOS for sharing numerous lunches, coffee breaks and social events.

I would also like to thank Ole Morten, Torleif and all my student as-

sistants for our cooperation in TTK4215, giving structure to days at work when struggling to find a path forward in the early days of my PhD. Thanks to the NTNU COVID-19 Taskforce for filling the initial period of the pandemic with meaningful distractions. Thanks to Omega FK, ITK Cageball and Charlottenlund Sportsklubb ("Kimmes disipler") for letting me hone my football skills during my years in Trondheim.

My automatic control journey has now taken me back to where it all started, in Ålesund. Thanks to all my good colleagues at the Department of ICT and Natural Sciences for your support, which has enabled me to (finally) complete this thesis alongside other tasks.

This endeavour would not have been possible without the loving support from you, Amalie. My greatest achievement during these years is our little family. Thanks to Ellinor and Oskar for your patience and for bringing a lot of laughter and joy into our daily lives. Thanks to my parents-in-law for babysitting and for letting me spend time on a desert island writing for this thesis. Finally, thanks Mom and Dad for even more babysitting, for your unconditional support and for giving me a good base for further study during our early days together.

Contents

Summary **iii**

Preface **v**

Contents **vii**

List of figures **xi**

List of tables **xvii**

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Background 6
- 1.3 Contributions 9
- 1.4 Publications 10
- 1.5 Outline 12

2 Preliminaries **13**
- 2.1 Notation and Definitions 13
- 2.2 Stability Definitions 16
- 2.3 Kinematics of Flight 16
- 2.4 Aircraft Equations of Motion 21
- 2.5 The Coordinated-Turn Equation 24
- 2.6 ArduPlane 25

I Geometric Attitude Control Laws for Fixed-Wing Aircraft **29**

3 Introduction to Part I **31**
- 3.1 Introduction 31

3.2 Outline of Part I 33
3.3 Related Work 33

4 Reduced-Attitude Control for Fixed-Wing Aircraft Using Geometric Methods on the Two-Sphere 35
4.1 Introduction 35
4.2 Reduced-Attitude Representation 36
4.3 Problem Statement 39
4.4 Control Design 42
4.5 Turn Coordination 50
4.6 Comparison With Controller Based on Euler Angles 52
4.7 Reference Filter 53
4.8 Numerical Example 54
4.9 Chapter Summary 56

5 Hybrid Control of Fixed-Wing Aircraft for Large-Angle Attitude Maneuvers on the Two-Sphere 59
5.1 Introduction 59
5.2 Hybrid Control Design 60
5.3 Numerical Example 69
5.4 Chapter Summary 73

6 Almost Global Geometric Reduced-Attitude Tracking Control of Fixed-Wing Aircraft 75
6.1 Introduction 75
6.2 Problem Definition 76
6.3 Turn Coordination 79
6.4 Control Laws - Nominal Case 82
6.5 Robustness Considerations 88
6.6 Simulation Results 95
6.7 Chapter Summary 102

7 Robust Reduced-Attitude Control of Fixed-Wing UAVs Using a Generalized Multivariable Super-Twisting Algorithm 103
7.1 Introduction 103
7.2 Literature Review 105
7.3 The Generalized Multivariable Super-Twisting Algorithm . 108
7.4 Problem Definition 110
7.5 Control Law 111
7.6 Simulation Study 115

7.7 Chapter Summary 117

II Path-Following Control 123

8 Introduction to Part II 125

9 Almost Global Three-Dimensional Path-Following Guidance Law for Arbitrary Curved Paths 127

9.1 Introduction 127
9.2 Problem Formulation 129
9.3 Guidance controller Design 131
9.4 Heading Control 135
9.5 Analysis of the Complete Closed-Loop System 136
9.6 Relation to Classical Line-of-Sight Guidance Laws Using Angles 137
9.7 Simulation Results 138
9.8 Chapter Summary 139

III Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs 143

10 Introduction to Part III 145

10.1 Introduction 145
10.2 Outline of Part III 147
10.3 Related Work 147
10.4 Reinforcement Learning 150

11 Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization 155

11.1 Introduction 155
11.2 Method 156
11.3 Results and Discussion 162
11.4 Chapter Summary 166

12 Data-Efficient Deep Reinforcement Learning for Attitude Control of Fixed-Wing UAVs: Field Experiments 167

12.1 Introduction 167
12.2 Method 168
12.3 Experimental Results 177
12.4 Discussion 181

12.5 Chapter Summary 188

13 Experimental Platform for Evaluation of Advanced Low-Level Control Algorithms 189

13.1 Introduction 189
13.2 Related work 191
13.3 Control algorithms: Overview 192
13.4 System Architecture 195
13.5 Test procedures 201
13.6 Experimental results 203
13.7 Lessons learned 206
13.8 Chapter Summary 208

Concluding Remarks 209

14 Conclusions and Future Directions 211

References 215

List of figures

1.1 Small fixed-wing unmanned aerial vehicles (UAVs). 1
1.2 Unmanned systems in marine operations. Illustration by NTNU AMOS. 2
1.3 The observational pyramid. Illustration by NTNU AMOS. . . . 3
1.4 A decade of drone investments (Source: Drone Industry Insights). 4
1.5 Number of hits on keyword "UAV" at IEEEXplore and ScienceDirect (Accessed 9/12-2022). 5

4.1 Reduced-attitude representation illustrated with a section of the two-sphere corresponding to $\theta = 0$. 37
4.2 Reduced-attitude representation. The aircraft is shown from the side, illustrated with a section of the two-sphere with a non-zero roll angle. 38
4.3 Block diagram of a guidance, navigation and control (GNC) architecture for fixed-wing aircraft. 51
4.4 Tracking scenario: Roll, pitch and angular velocity error. 55
4.5 Tracking scenario: Angle of attack (AoA), sideslip angle (SSA) and control surface deflection angles. 55
4.6 Regulation scenario: Euler angle comparison. 56
4.7 Regulation scenario: Angle of attack, sideslip angle and norm of control input. 57
4.8 Regulation scenario: Trajectories on the two-sphere. Solid: Geometric controller. Dashed: Controller based on Euler angles. . . 57

5.1 Control surface deflections, respectively aileron δ_a, elevator δ_e and rudder δ_r, for the continuous controller (blue) and the hybrid controller (orange). 70
5.2 Attitude response towards the reference (green) represented by roll angle ϕ and pitch angle θ for the continuous controller (blue) and the hybrid controller (orange). 70

5.3 Angular rate response compared to the reference (green), represented by roll rate p, pitch rate q and yaw rate r, for the continuous controller (blue) and the hybrid controller (orange). 71
5.4 The relative velocity v_a^b represented by airspeed V_a, angle of attack α (AOA) and sideslip angle β (SSA) for the continuous controller (blue) and the hybrid controller (orange). 72
5.5 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. 72

6.1 Roll and pitch angles vs. references, and velocity tracking error e_ω. 97
6.2 Reduced attitude Γ (blue) and the reduced-attitude reference Γ_d (red). 97
6.3 Magnitudes of $\omega^{\|}$ (red) vs $\omega_d^{\|}$ (blue), both parallel to Γ. 98
6.4 Disturbance estimates (blue) vs their true values (red). 98
6.5 a) Flow angles: Angle of attack α and sideslip angle β; b) Control surface deflections: Aileron δ_a, elevator δ_e and rudder δ_r. 99
6.6 Path on the two-sphere for the 17 first seconds. Red asterix marks the initial configuration, while the constant reference is marked green. 99
6.7 SITL: a) Geographical plot showing the horizontal path of the unmanned aerial vehicle (UAV); b) Altitude in meters (above home position). 100
6.8 SITL: Roll and pitch angles (red) vs reference angles (blue). . . 101
6.9 SITL: Reduced-attitude vector Γ (red) vs reference attitude (blue). 101
6.10 SITL: Control input. Control surface deflections are normalized to $[-1, 1]$ and the throttle to $[0, 1]$. 102

7.1 Wind turbulence generated by Dryden turbulence model: Linear wind velocity (top) and angular velocity (bottom). 117
7.2 Roll and pitch angles (red) vs references (blue). 118
7.3 Reduced-attitude vector Γ (blue) vs reference Γ_d (red). 118
7.4 The 17 first seconds of simulated trajectory shown as a path on the sphere. Initial condition marked red. Reference marked green. 119
7.5 Commanded control surface deflections. 119
7.6 Sliding variable σ (top) and integrator state z (bottom). 120
7.7 $\omega_d^{\|}$ (red) vs $\omega^{\|}$ (blue). 120

7.8 Airspeed, angle of attack and sideslip angle. 121

9.1 Desired path. 130
9.2 The wind triangle. From the Pythagorean theorem, $Y = \sqrt{V_w^2 - (v_w^\top \eta)^2}$ and $X = \sqrt{(v_w^\top \eta)^2 + V_a^2 - V_w^2}$. Finally, $V = v_w^\top \eta + X$ to derive (9.19) 134
9.3 Projection to the horizontal plane. The magnitude of $\mathbf{\Pi}_{e_3}^\perp(\eta_r)$ equals $\cos(\gamma_r)$, and $\|\mathbf{\Pi}_{e_3}^\perp(\mathbf{\Pi}_{\eta_r}^\perp(\tilde{\xi}))\| = |y_e|$. 137
9.4 Path following of helical path. Desired path $\mathcal{P}$ in blue, and path traced out by the vehicle in red. The direction of travel is upwards. 139
9.5 From top to bottom: the magnitude of normal acceleration (normal to η_a), the magnitude of the orthogonal component of the position error, and the angle between desired and actual relative heading. 140
9.6 Top: V_r converges to V as the component of $\tilde{\xi}$ in the direction of η_r (bottom) goes to zero and η is aligned with η_r. V_a can be seen to be constant. 140
9.7 Azimuth and elevation angles. For each respective vector, the azimuth angle is $\chi_* = \text{atan2}(e_2^\top \eta_*, e_1^\top \eta_*)$, and the elevation angle is $\gamma_* = \arcsin(-e_3^\top \eta_*)$. 141

10.1 Agent-environment interaction in reinforcement learning (RL) (adapted from [235]). 150

11.1 The RL controller trained episodically with a single setpoint and no wind or turbulence generalizes well to many wind conditions and continuous tracking of setpoints (shown with dashed lines marked by crosses). Here subjected to severe wind and turbulence disturbances with a magnitude of 20 m/s. 165
11.2 Comparison of the proportional-integral-derivative (PID) and RL controllers tasked with tracking the dashed green line. 166

12.1 Architecture of the RL controller, superscript $\sim$ signifies normalized states. 171
12.2 The aerodynamic coefficients of the UAV in a longitudinal (top three) and a lateral (bottom three) chirp signal test sequence for elevator and aileron, respectively, based on IMU data (blue) and model prediction (orange). 174
12.3 Fly-by-wire A (FBWA) mode using references (dashed orange line) from the pilot, showing the attitude states and right (blue) and left (orange) elevon signals. 178

12.4 Position plot showing how the RL controller is able to take references from ArduPlane's guidance system in Auto mode, and effectively follow prescribed paths. First, a loiter, then a square waypoint pattern. 179

12.5 Comparison between the RL controller and the ArduPlane PID controller for steps in the roll (top) and pitch (bottom) references (dashed orange line). 180

12.6 Response of the RL controller where the steady-state error has been estimated and references (dashed orange line) adjusted to compensate. 182

12.7 Oscillatory attitude response of initial flight experiments, for one controller trained with original (left) and one with reduced (right) C_{m_q}. 183

12.8 Open-loop level-flight response of the RL controller when perturbing one input at a time. The x-axis is in the units of the corresponding state. The lines are elevon outputs mapped to aileron (orange) and elevator (blue). 185

12.9 The learning phase of the proposed RL controller, showing normalized mean episode reward and error-proportional gains. The solid line represents a rolling average mean value while the shaded region represents one standard deviation over three randomly seeded controllers. Base refers to the method as presented in Section 12.2, the fully-connected (FC) version replaces the convolutional layer with an FC layer, and the h1 version has no history in the input. 186

12.10 An RL controller that has trained for only 3 minutes of real-time flight in the simulation environment is airworthy, able to track references (dashed orange line) reasonably well. 187

13.1 Hardware configuration of the experimental platform. 196

13.2 nonlinear model predictive control (NMPC) benchmarking results for the employed single-board computer (SBC) for different prediction horizons N in a direct multiple-shooting scheme with 0.1 s shooting interval. 197

13.3 Experimental results: Initial switch to geometric attitude control (GAC). The plots show the tracking response for roll and pitch references (dashed, black). The bottom subplot depicts the virtual deflections of the aileron (blue) and the elevator (orange). . . . 204

13.4 Performance of GAC after a short tuning procedure. The plots show the tracking response for roll and pitch references (dashed, black). The bottom subplot depicts the virtual deflections of the aileron (blue) and the elevator (orange). 205
13.5 Experimental results with deep reinforcement learning (DRL) showing (a) oscillatory attitude response (blue) vs reference angles (dashed, black) in initial flight experiments, and (b) after a few design iterations, the controller achieves good tracking performance. See [32] for details. 207

List of tables

11.1 Constraints and ranges for initial conditions and target setpoints used during training. 160
11.2 PID controller parameters. 161
11.3 Performance metrics for the RL controller and the baseline PID controller on the evaluation scenarios. Both controllers exhibit strengths in different aspects — the best value in each circumstance is shown in bold. 164

12.1 Hyperparameters of the RL algorithm 169
12.2 Initial conditions for the episodes are uniformly sampled from the indicated ranges. 176
12.3 Linearly approximated gains at level-flight, where each input is perturbed in isolation. 186

13.1 Features of the compared controllers. 193

Glossary

ADP adaptive dynamic programming.

ANN artificial neural network.

AoA angle of attack.

AoS angle of sideslip.

AUV autonomous underwater vehicle.

CAPS conditioning for action policy smoothness.

CFD computational fluid dynamics.

CG centre of gravity.

DDPG Deep Deterministic Policy Gradient.

DOF degree of freedom.

DRL deep reinforcement learning.

EKF extended Kalman filter.

ESC electronic speed controller.

FBWA fly-by-wire A.

FC fully-connected.

FPGA field-programmable gate array.

FVI fitted value iteration.

GAC geometric attitude control.

GAE generalized advantage estimate.

GNC guidance, navigation and control.

GNSS global navigation satellite system.

HER Hindsight Experience Replay.

I2C Inter-Integrated circuit.

IMC Inter Module Communication.

IMU inertial measurement unit.

KL Kullback–Leibler.

LAN local area network.

LOS line-of-sight.

LQR linear quadratic regulator.

MC Monte Carlo.

MDP Markov decision process.

MIMO multiple-input multiple-output.

MLP multilayer perceptron.

MPC model predictive control.

MUX multiplexer.

NED north-east-down.

NMPC nonlinear model predictive control.

OCP optimal control problem.

ODE ordinary differential equation.

PI proportional-integral.

PID proportional-integral-derivative.

PPO proximal policy optimization.

PWM pulse-width modulation.

QP quadratic program.

RL reinforcement learning.

RNN recurrent neural network.

ROS Robot Operating System.

RRT Rapidly-Exploring Random Tree.

RTL	return to launch.
SAC	Soft Actor-Critic.
SBC	single-board computer.
SITL	software-in-the-loop.
TRPO	Trust Region Policy Optimization.
UAV	unmanned aerial vehicle.
VTOL	vertical take-off and landing.

Chapter 1

Introduction

1.1 Motivation

In recent years, technological advancements have led to increased use of small unmanned aerial vehicles (UAVs) in civil, commercial, and scientific applications. UAVs, also known as drones, unmanned aircraft systems (UAS) or remotely piloted aircraft systems (RPAS), are aircraft without any humans onboard that can either be under remote control or fly automatically. UAVs are employed extensively to increase safety and efficiency in a plethora of tasks.

Fixed-wing UAVs, as illustrated in Fig. 1.1, have superior range and endurance when compared to rotary-wing UAVs (like the ubiquitous quadcopter), which enable applications such as infrastructure inspection, search and rescue, aerial surveillance and mapping, environmental monitoring, and medical transportation [242][1]. UAVs can be deployed standalone or as part of a network of unmanned vehicles performing coordinated operations (Fig. 1.2).

[1] In this thesis when referring to *small* fixed-wing UAVs, we mean UAVs in the size of "model airplanes" from 50 cm and up to a few meters [70]. These vehicles can be carried around by one person and easily hand-launched or launched using a catapult.

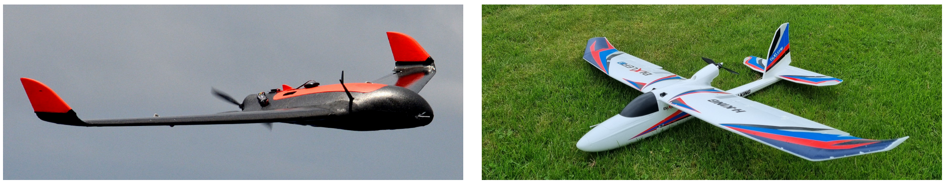

(a) Skywalker X8 (NTNU UAV-Lab). **(b)** H-King Bixler 3 (author photo).

Figure 1.1: Small fixed-wing unmanned aerial vehicles (UAVs).

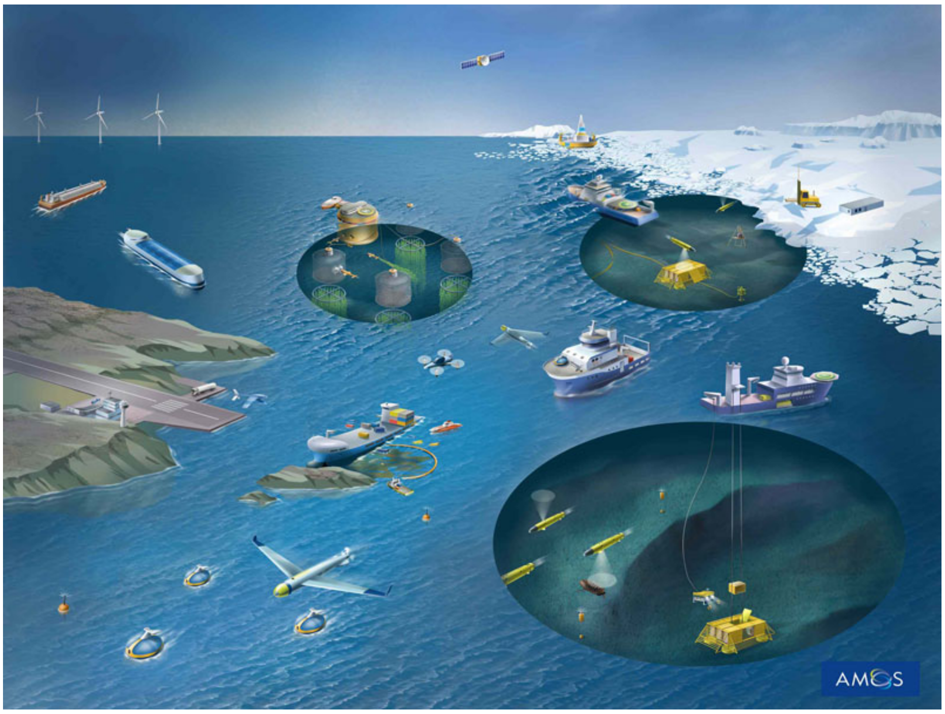


Figure 1.2: Unmanned systems in marine operations. Illustration by NTNU AMOS.

In the observational pyramid illustrated in Fig. 1.3, vehicles operating in different temporal and spatial scales do biological mapping of an area using a range of different sensors.

The interest in UAVs has skyrocketed in the last decade. In Fig. 1.4, we see exponential growth in global drone investments, which also carries over into academia. Figure 1.5 shows how the number of publications containing the keyword “UAV” has increased exponentially as well (based on queries in the databases of IEEEXplore and ScienceDirect). For guidance, navigation and control (GNC) of these vehicles, several open-source hardware and software systems exist. For instance, the ArduPilot [13] and PX4 [174] software suites have been widely adopted by industry, research institutions, as well as hobbyists. Over the last decade, open-source ecosystems have stimulated many drone startups and considerable growth in the unmanned systems industry.

A critical component of any GNC architecture is the *autopilot*, which is the automatic control system that provides the basic low-level control functions that maintain aircraft orientation (attitude), speed, altitude and course

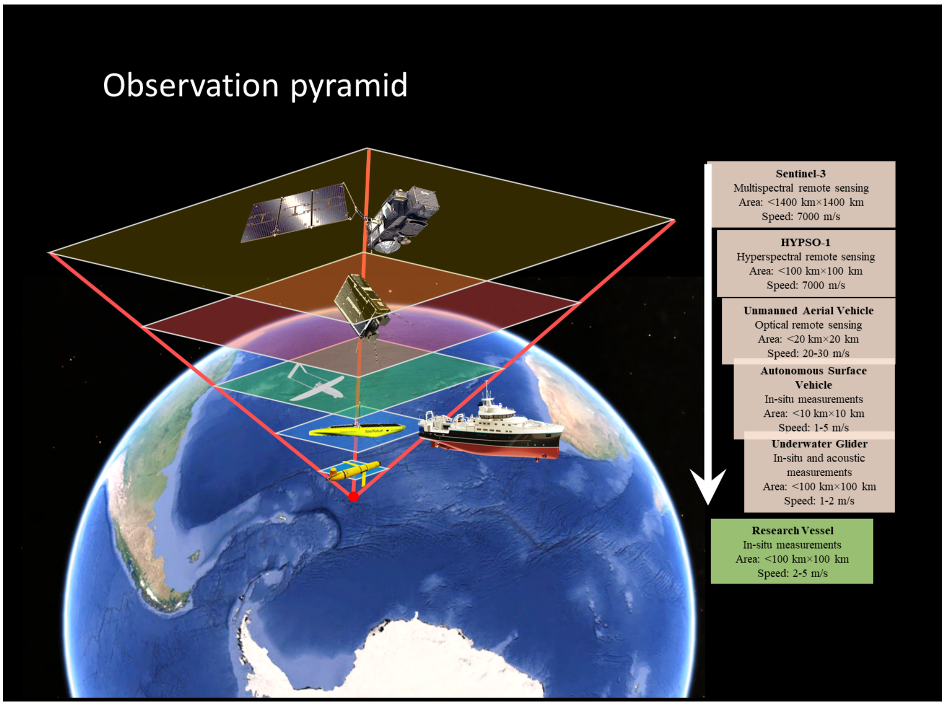


Figure 1.3: The observational pyramid. Illustration by NTNU AMOS.

heading [19]. Since the first aircraft autopilot was developed by Lawrence Sperry[2] of the Sperry Gyroscope Company in 1912 [233], automatic flight control systems have evolved from basic stability augmentation systems such as the yaw damper [231] to more elaborate systems such as automatic landing systems [166]. The autopilot is a key enabling technology for the widespread use of UAVs, with current systems providing functionality for fully automated flight. Autopilots for fixed-wing UAVs are typically designed using cascaded single-variable loops under assumptions of decoupled longitudinal and lateral motion, using classical linear control theory [19]. The dynamics of fixed-wing aircraft are, however, strongly coupled and nonlinear. Nonlinear terms in the equations of motion include kinematic nonlinearities (rotations and Coriolis effects), actuator saturation and aerodynamic nonlinearities, which are uncertain and difficult to model [113, 180].

The linear and decoupled controllers are designed to operate close to some nominal operating conditions (trim conditions), and the flight performance is usually reliable and well-tested for nominal flight (for non-

[2]As a curiosity, Lawrence Sperry is also widely regarded as the founder of the "Mile High Club".

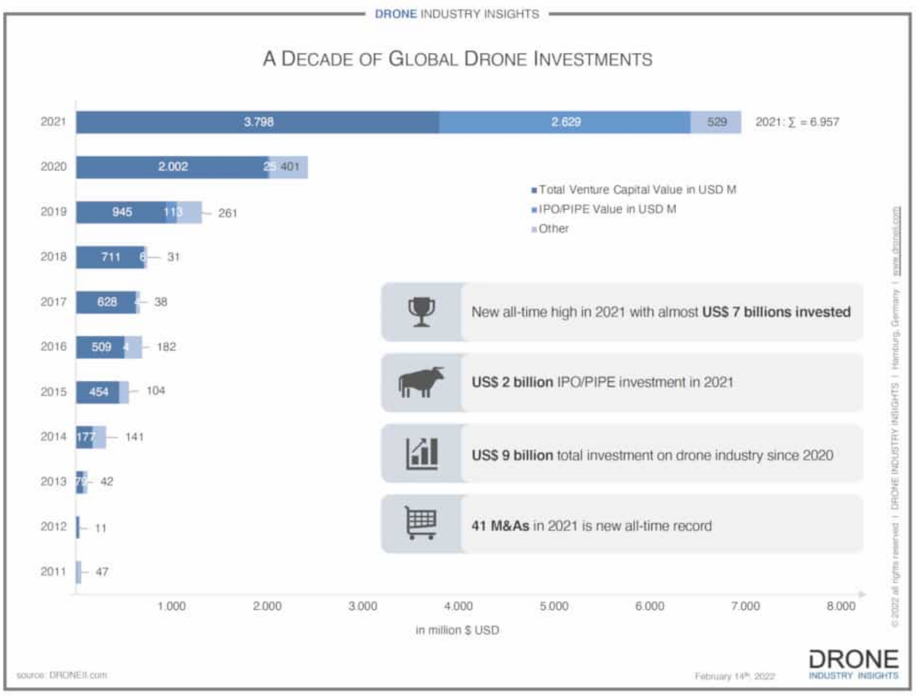


Figure 1.4: A decade of drone investments (Source: Drone Industry Insights).

aggressive manoeuvres under relatively calm wind conditions) but requires conservative safety limits in the allowable range of flight conditions and manoeuvres (flight envelope protection) because linear controllers applied to nonlinear systems typically result in a limited region of attraction [125]. For more aggressive flight trajectories, nonlinear aerodynamic effects become increasingly significant, causing performance deterioration and, in the worst case, instability. A practical approach to this problem is to limit the range of flight conditions the UAV is operated in, referred to as the flight envelope [233].

The nonlinear nature of the underlying physics suggests that state-of-the-art methods in nonlinear control theory should instead be used to design more advanced autopilots that are less conservative, more capable of agile manoeuvring, and allow for a wider flight envelope. Such functionality will increase the capabilities of fixed-wing UAVs and can lead to new innovative use cases and products. Although promising results exist using nonlinear methods such as dynamic inversion [137], backstepping [73, 228], and adaptive control [138], these methods have arguably not been developed to the mature level required for implementation in UAV autopilots. Many non-

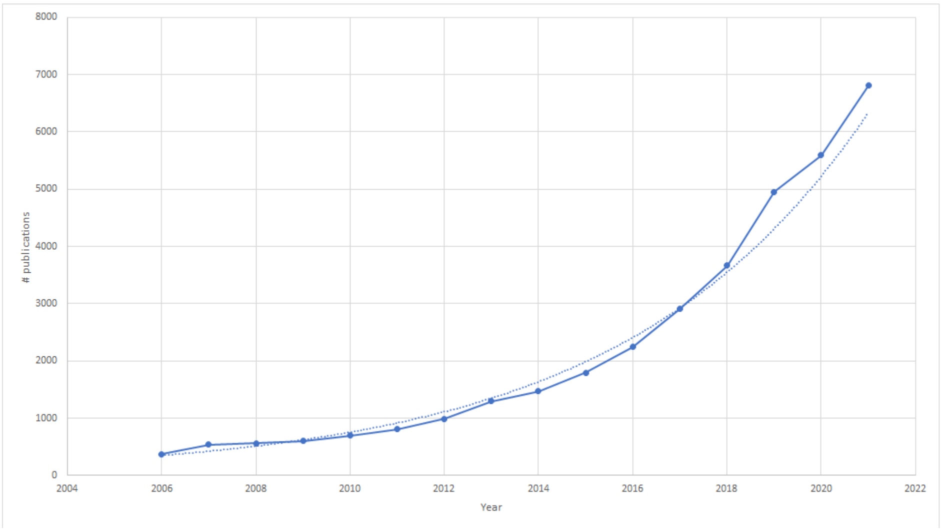


Figure 1.5: Number of hits on keyword “UAV” at IEEEXplore and ScienceDirect (Accessed 9/12-2022).

linear control algorithms have not been sufficiently tested outside nominal operating conditions for fixed-wing UAVs. Moreover, flight control methods developed for high-performance aircraft are not easily adopted since fixed-wing UAVs typically operate at low speeds with small stall margins, [106].

Objectives

As we move towards higher levels of automation and to enable safe and efficient autonomous operation of UAVs, we need robust autopilots that can handle a range of environmental conditions, including turbulent wind conditions, and operate in the presence of highly uncertain aerodynamics [177]. The main research objective of this thesis is to develop nonlinear autopilot designs for fixed-wing UAVs that extend the flight envelope for more aggressive manoeuvres and allows the UAVs to operate in a wider range of weather conditions. In particular, we consider to:

- Use Lyapunov methods [125] to design attitude controllers with large regions of attraction that explicitly deal with kinematic and aerodynamic nonlinearities.
- Use Lyapunov methods to develop a guidance controller suitable for fixed-wing UAVs that allows for aggressive manoeuvring in windy conditions.

- Assess the applicability of alternative control design methods based on recent progress in machine learning, in particular, deep reinforcement learning (DRL).
- Develop an experimental platform to allow for implementation and testing of the developed algorithms in outdoor flight experiments with fixed-wing UAVs.

1.2 Background

We now proceed by providing further background for the contributions of this thesis. The next paragraph is from [31] and the remainder of this section from [58].

Examples of nonlinear control methods applied to UAVs include gain scheduling [85], linear parameter varying (LPV) control [215], dynamic inversion (feedback linearization) [123], adaptive backstepping [214], sliding mode control [48], nonlinear model predictive control [167], nonlinear H-infinity control [84], dynamic inversion combined with mu-synthesis [176], model reference adaptive control [138] and L1 adaptive control [122]. Automated agile and aerobatic manoeuvring is treated in [150] and [40]. Several of these methods require a more or less accurate aerodynamic model of the UAV. A model-free method based on fuzzy logic can be found in [133]. Fuzzy control falls under the category of intelligent control systems, which also includes the use of neural networks. An adaptive backstepping controller using a neural network to compensate for aerodynamic uncertainties is given in [144], while a genetic neuro-fuzzy approach for attitude control is taken in [68]. The state of the art in intelligent flight control of small UAVs is discussed in [218].

As underactuated vehicles, conventional fixed-wing aircraft have fewer control inputs than the dimension of their configuration space. One or more propellers provide a thrust force in the longitudinal direction, but the forces orthogonal to the thrust axis (lift, side force) are not directly controllable. Therefore, fixed-wing UAVs have to resort to using guidance schemes [36], where the UAV's geometric path in 3-D space is controlled by specifying course and flight path angle commands to lower-level autopilots [116]. Due to the fact that small fixed-wing UAVs experience winds that are large relative to their operating airspeeds [19], *path-following* methods [217] are usually preferred over *trajectory tracking* control [3]. In path following, the goal is to reach and follow a geometric path without any temporal constraints. This also deals with performance limitations of trajectory tracking for systems with nonminimum phase characteristics, such as aircraft [4]. See [234] and [196] for a comparison of different path-following algorithms for fixed-wing UAVs in two and three dimensions, respectively. For a recent survey with a focus on quadrotor UAVs, see [216].

Guidance and control systems for unmanned vehicles can be *integrated*, or *separated* [219]. For integrated guidance and control (IGC) systems, the guidance system and inner-loop autopilot are designed simultaneously, taking cross-coupling effects into account. On the other hand, in separated guidance and control (SGC), inner and outer loops are designed separately, with modularity and cross-platform use in mind [51]. Examples of separate guidance algorithms for fixed-wing UAVs include nonlinear guidance laws [193, 194], vector-field path following [139, 187] and a guidance law based on nested saturations [195]. In [122], path following is achieved by using an existing commercial inner-loop autopilot but augmented with an $\mathcal{L}_1$ adaptive controller to deal with modelling uncertainty and environmental disturbances. While most guidance algorithms use only kinematic models, an integrated approach is presented in [119] that uses a simple model of the aerodynamic forces acting on the aircraft. Common to all the mentioned approaches, both IGC and SGC, is the reliance on attitude control in the innermost loop, i.e. controlling the aircraft orientation relative to an inertial reference frame.

Several different attitude representations have been employed for fixed-wing UAV path following, including Euler angles [155], rotation matrices [56] and unit quaternions [5]. Minimal representations such as Euler angles are often used because of their intuitive interpretation but suffer from gimbal-lock singularities [164]. Unit quaternions [250] are singularity-free but provide a double cover of $SO(3)$, the space of 3-D rotations. This might lead to *unwinding*, where the UAV unnecessarily makes a full rotation, even when arbitrarily close to the target attitude [26, 54]. Rotation matrices, on the other hand, provide a global and unique representation. This has led to a significant research effort into so-called *geometric attitude control*, where singularity-free controllers are designed directly on $SO(3)$, using rotation matrices, that avoid the unwinding phenomenon and often control the system along *geodesics*, i.e., paths of minimum length in rotation space [24, 42, 53, 120, 129, 141]. These advantages are desirable when the controlled vehicle is subject to large angle rotations, e.g. a fixed-wing UAV recovering from large attitude errors resulting from severe wind gusts [117].

Fixed-wing UAVs use one of two main mechanisms for turning, *bank-to-turn*, where a lateral acceleration is generated by reorienting the lift-force by rolling, or banking, the UAV, or *skid-to-turn*, where turning is achieved by generating a sideslip angle, which in turn generates a lateral force that turns the vehicle [46]. In [77], these methods are combined to reduce lateral distortion of camera images gathered by a fixed-wing UAV. In general, bank-to-turn is often preferred over skid-to-turn because, for most aircraft, the lift force is of orders of magnitude greater than thrust forces [6]. Thus, the course angle, yaw angle, and turn rate of aircraft are not controlled directly but rather through banked-turn manoeuvres. For aircraft in coordinated turns, i.e., with zero sideslip angle, the coordinated-turn equation provides a simple relationship between roll angle and resulting turn rate and is for this reason often used in autopilot design [19, 44, 65, 92, 149, 232], including those used in state-of-the-art open-source autopilots [13, 174].

Controllers designed using rotation matrices or quaternions control the full attitude and, therefore, cannot be directly applied to fixed-wing aircraft using banked turn manoeuvres. One approach could be to feedback the true yaw angle into the desired rotation matrix and, as such, use a rotation error representation for roll and pitch only. However, this representation is highly redundant, as nine parameters are used to parametrize a two-dimensional subspace. A simpler approach that does not require the full machinery of working in $SO(3)$ is to consider a reduced-attitude representation, evolving on the two-sphere, $\mathbb{S}^2 \subset \mathbb{R}^3$ [41]. In this space of reduced attitude, all rotations that are related by a rotation about some fixed axis are considered the same [54]. Control systems with reduced attitude evolving on $\mathbb{S}^2$ have previously been studied in the context of spin-axis [41] and boresight axis [203] control for satellites, pendulum stabilization [53], path-following control of underwater vehicles [253], thrust-vector control for multirotor UAVs [47, 108] and for general rigid bodies [143, 172, 205]. Controllers developed on $\mathbb{S}^2$ are relatively simple compared to those developed using rotation matrices and require fewer matrix operations.

It is well-known that a desired attitude (full or reduced) cannot be globally stabilized using continuous state-feedback control laws [26]. This stems from the topological properties of $SO(3)$ and $\mathbb{S}^2$, which are compact, boundaryless manifolds that are not diffeomorphic to any Euclidean space. The largest possible attraction basins under continuous feedback are *almost global*, i.e., excluding a zero-measure set, which corresponds to the stable manifolds of additional unstable equilibrium points [145]. However, global asymptotic stability can be achieved by using tools from hybrid dynamical systems, where hysteresis-based switching ensures that all trajectories converge to the desired equilibrium [21, 47, 142, 143, 169, 172, 173].

1.3 Contributions

The main contributions of this work are focused on applied nonlinear control of fixed-wing aircraft and, in particular, small fixed-wing UAVs. While most of the developments apply to fixed-wing aircraft in general, all simulation studies and experiments are carried out using small fixed-wing UAVs. A large focus has been put on the low-level control of these vehicles, i.e., attitude control, using both formal, geometric methods grounded on Lyapunov stability theory, as well as data-driven methods based on deep reinforcement learning. Furthermore, there are contributions to path-following guidance in three dimensions, especially suited for fixed-wing UAVs flying in wind fields that are large compared to their airspeed. Finally, a significant amount of work has been put into the development of a suitable test platform for experimental evaluation of the developed control algorithms on a physical UAV operating in an outdoor environment.

The thesis contributions can be summarized as follows:

- Proposed a novel geometric attitude controller for fixed-wing aircraft using a reduced-attitude representation evolving on the $\mathbb{S}^2$ that is invariant to the yaw/heading angle and, therefore, especially well suited for fixed-wing aircraft performing banked turn manoeuvres. The methodology is backed up by stability analysis and simulation results, which show that the controller is more efficient than when using Euler angles (roll and pitch).
- Developed hybrid geometric attitude controllers for fixed-wing aircraft that render the desired reduced attitude globally asymptotically stable, thus extending the nominal controller in the previous contribution and overcoming the obstruction to global stabilization that exists when using continuous feedback.
- Further refined the nominal reduced-attitude controller to a more practical design. In particular, several assumptions are relaxed through a new, simpler backstepping-based design, and the efficacy of the proposed design is demonstrated through implementation in ArduPilot and realistic software-in-the-loop (SITL) simulations.
- Proposed a new robust reduced-attitude controller for fixed-wing UAVs using sliding-mode control concepts. In particular, we design a novel sliding surface and employ a generalized multi-variable super-twisting algorithm. The performance of the controller is demonstrated through a simulation study with highly turbulent conditions.
- Designed a class of path-following guidance laws for three-dimensional, regularly parameterized paths that, unlike most approaches existing in

the literature, do not require the explicit construction of a path frame. Through cascade arguments, we show that the proposed design leads to almost global stability results and thus relaxes the set of feasible initial conditions compared to existing methods. Furthermore, the guidance law generates a normal acceleration command that is normal to the flow-relative velocity vector (as the lift force). This allows for a natural decomposition of the desired vehicle acceleration for aerial vehicles in coordinated turns: tangential acceleration for airspeed control and normal acceleration for guidance generated through bank-to-turn manoeuvres, i.e., by tilting the lift vector.

- Proposed and demonstrated, in simulation, the viability of using DRL to control the attitude and speed of fixed-wing UAVs. Moreover, the simulation results show that the DRL controller is more robust against turbulent wind disturbances when compared to conventional proportional-integral-derivative (PID) controllers.
- Demonstrated a successful real-life control application of DRL. We present the results of field experiments where attitude control policies trained in a simulation environment are successfully transferred to the field, showing comparable performance to a state-of-the-art industrial autopilot. Using a particularly data-efficient approach, we are able to learn to control the UAV with only three minutes of flight data.
- Developed an experimental platform well suited for early-stage rapid testing of computationally demanding, low-level control algorithms requiring direct access to the actuators. This work is used for the DRL experiments in Chapter 12 as well as the model predictive control (MPC) experiments in [210].

1.4 Publications

This thesis is based on eight research papers, of which seven are published in peer-reviewed international journals and conferences, and one is currently under review for possible publication. These are listed below, chronologically by date of publication.

[31] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[61] E. M. Coates, D. Reinhardt, and T. I. Fossen. Reduced-attitude control of fixed-wing unmanned aerial vehicles using geometric methods on

the two-sphere. *21st IFAC World Congress, Berlin, Germany. IFAC-PapersOnLine*, 53:5749–5756, 2020.

[209] D. Reinhardt, E. M. Coates, and T. A. Johansen. Hybrid control of fixed-wing UAVs for large-angle attitude maneuvers on the two-sphere. *21st IFAC World Congress, Berlin, Germany. IFAC-PapersOnLine*, 53:5717–5724, 2020.

[58] E. M. Coates and T. I. Fossen. Geometric reduced-attitude control of fixed-wing UAVs. *Applied Sciences*, 11(7), 2021.

[59] E. M. Coates, J. B. Griffiths, and T. A. 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)*, 2021.

[62] E. M. Coates, D. Reinhardt, K. Gryte, and T. A. 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)*, 2022.

[32] E. Bøhn, E. M. Coates, D. Reinhardt, and T. A. Johansen. Data-efficient deep reinforcement learning for attitude control of fixed-wing UAVs: Field experiments. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[60] E. M. Coates, T. Hamel, and T. I. Fossen. Almost global three-dimensional path-following guidance law for arbitrary curved paths. In *62nd IEEE Conference on Decision and Control (CDC) (accepted)*, 2023.

During the PhD period, I have also contributed to the following papers that are related to but not part of this thesis:

[63] E. M. Coates, A. Wenz, K. Gryte, and T. A. Johansen. Propulsion system modeling for small fixed-wing UAVs. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[161] B. Løw-Hansen, N. C. Müller, E. M. Coates, T. A. Johansen, and R. Hann. Identification of an electric uav propulsion system in icing conditions. *SAE International Conference on Icing of Aircraft, Engines, and Structures*, 2023.

[126] Ø. K. Kjerstad and E. M. Coates. A cascaded heading control design with motion constraint handling for marine surface vessels. In *2023 European Control Conference (ECC)*, 2023.

[210] D. Reinhardt, E. M. Coates, and T. A. Johansen. Low-level nonlinear model predictive attitude and speed control of fixed-wing unmanned aerial vehicles. *Control Engineering Practice*, submitted.

1.5 Outline

The thesis consists of 14 chapters, divided into three main parts. Before commencing with Part I, in Chapter 2, we first introduce mathematical notation and review basic modelling and control used throughout the thesis. Part I presents the results on geometric attitude control, path-following guidance is covered in Part II, and Part III is about DRL. Finally, in Chapter 14, we provide some concluding remarks, including possible future research directions.

Chapter 2

Preliminaries

2.1 Notation and Definitions

Positive (resp. non-negative) real numbers are denoted $\mathbb{R}_{>0}$ ($\mathbb{R}_{\geq 0}$) and $\mathbb{N}$ is the set of natural numbers. Let $\mathbb{R}^n$ denote the n-dimensional Euclidean space with the standard basis $\{e_1, \dots, e_n\}$ and Euclidean norm $\|x\| = \sqrt{x^\top x}$. When $n = 1$, we denote the absolute value of x by $|x|$.

For compactness, explicit time arguments will be used for state variables only when considering specific solutions and for signals and functions in general when we want to highlight that time-varying exogenous signals are considered.

The set of 3×3 symmetric positive definite matrices is denoted $\mathcal{P}_+^3$, the identity matrix of dimension $n \times n$ is denoted by I_n, and the maximum and minimum eigenvalues of a square matrix A are denoted $\lambda_{\max}^A$, $\lambda_{\min}^A$, respectively. For a symmetric matrix $A = A^\top \in \mathbb{R}^{n \times n}$, and $x \in \mathbb{R}^n$, we have the following inequality for quadratic forms:

$$\lambda_{\min}^A \|x\|^2 \leq x^\top A x \leq \lambda_{\max}^A \|x\|^2. \tag{2.1}$$

The induced 2-norm of a matrix A is $\|A\| = \sigma_{\max}^A$, where $\sigma_{\max}^A$ is the largest singular value of A. For square, real symmetric positive semidefinite matrices A, $\lambda_{\max}^A = \sigma_{\max}^A$.

Sets

The set $\mathbb{B} := \{x \in \mathbb{R}^3 : \|x\| \leq 1\}$ is the closed unit ball in $\mathbb{R}^3$, while $\mathbb{B}_2 := \{x \in \mathbb{R}^3 \colon \|x\| \leq 2\}$ is the closed 3-ball of radius two. We denote the boundary of $\mathbb{B}_2$ by $\partial\mathbb{B}_2 := \{x \in \mathbb{R}^3 \colon \|x\| = 2\}$.

The set $\mathbb{S}^2 \subset \mathbb{R}^3$ is the two-sphere embedded in $\mathbb{R}^3$. $\mathbb{S}^2$ is the set of three-dimensional unit vectors and is defined as

$$\mathbb{S}^2 := \{x \in \mathbb{R}^3 \colon \|x\| = 1\}. \tag{2.2}$$

The tangent space at a point $p \in \mathbb{S}^2$ is identified with the vectors that are orthogonal to p:

$$\mathrm{T}_p\mathbb{S}^2 := \{v \in \mathbb{R}^3 \colon p^\top v = 0\},$$

and the normal space $\mathrm{N}_p\mathbb{S}^2$ is the set of vectors parallel to p:

$$\mathrm{N}_p\mathbb{S}^2 := \{w \in \mathbb{R}^3 \colon w^\top v = 0 \text{ for all } v \in \mathrm{T}_p\mathbb{S}^2\}.$$

The tangent bundle $\mathrm{T}\mathbb{S}^2$ is the set

$$\mathrm{T}\mathbb{S}^2 := \{(p, v) \colon p \in \mathbb{S}^2, v \in \mathrm{T}_p\mathbb{S}^2\}. \tag{2.3}$$

The three-dimensional special orthogonal group is the set of three-dimensional rotation matrices, given by

$$\mathrm{SO}(3) := \{R \in \mathbb{R}^{3\times 3} \colon R^\top R = I_3, \det R = 1\}. \tag{2.4}$$

The three-sphere, which is a double cover of SO(3), is the set of four-dimensional unit vectors,

$$\mathbb{S}^3 := \{x \in \mathbb{R}^4 \colon \|x\| = 1\}. \tag{2.5}$$

The lie algebra of SO(3) is denoted by $\mathfrak{so}(3)$ and consists of all 3×3 matrices that are skew-symmetric:

$$\mathfrak{so}(3) = \{A \in \mathbb{R}^{3\times 3} \colon A^\top = -A\}. \tag{2.6}$$

For $u, v \in \mathbb{R}^3$, the map $\boldsymbol{S} \colon \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined as

$$\boldsymbol{S}(u) = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \tag{2.7}$$

and maps the vector u to the skew-symmetric matrix such that the (anticommutative) cross-product operation in $\mathbb{R}^3$ can be performed as a matrix-vector multiplication: $\boldsymbol{S}(u)v = u \times v = -v \times u$.

Algebraic Identities

For any $u, v, w \in \mathbb{R}^3$, the scalar triple product $u \cdot (v \times w) \equiv u^\top \boldsymbol{S}(v) w$ is invariant to even permutations:

$$u^\top \boldsymbol{S}(v) w = v^\top \boldsymbol{S}(w) u = w^\top \boldsymbol{S}(u) v. \tag{2.8}$$

Another useful identity that can easily be derived from the Jacobi identity for cross-products is the following:

$$\boldsymbol{S}(\boldsymbol{S}(a)b) = \boldsymbol{S}(a)\boldsymbol{S}(b) - \boldsymbol{S}(b)\boldsymbol{S}(a). \tag{2.9}$$

For $p \in \mathbb{S}^2$, we have that

$$\boldsymbol{S}^3(p) = -\boldsymbol{S}(p). \tag{2.10}$$

Projection Operators

Any square matrix $A \in \mathbb{R}^{n\times n}$ can be written as the sum of a symmetric and skew-symmetric part, $A = \mathbf{sym}(A) + \mathbf{skew}(A)$, where $\mathbf{sym}(A) = (A + A^\top)/2$ and $\mathbf{skew}(A) = (A - A^\top)/2$ are the symmetric and skew-symmetric projection operators, respectively.

With $p \in \mathbb{S}^2$ and $x \in \mathbb{R}^3$, define the orthogonal and parallel projection operators $\boldsymbol{\Pi}_p^\perp \colon \mathbb{R}^3 \to \mathrm{T}_p\mathbb{S}^2$ and $\boldsymbol{\Pi}_p^{\|} \colon \mathbb{R}^3 \to \mathrm{N}_p\mathbb{S}^2$ by

$$\boldsymbol{\Pi}_p^\perp(x) = \left(I_3 - pp^\top\right) x = -\boldsymbol{S}^2(p)x, \qquad \boldsymbol{\Pi}_p^{\|}(x) = pp^\top x. \tag{2.11}$$

Then, any vector $x \in \mathbb{R}^3$ can be written as the sum $x = \boldsymbol{\Pi}_p^\perp(x) + \boldsymbol{\Pi}_p^{\|}(x)$. For any $v \in \mathrm{T}_p\mathbb{S}^2$ and $x \in \mathbb{R}$,

$$v^\top \boldsymbol{\Pi}_p^\perp(x) = v^\top x. \tag{2.12}$$

Saturation Functions

For $x \in \mathbb{R}$, and $\Delta > 0$, the classical saturation function is defined as

$$\mathrm{sat}^\Delta(x) = \begin{cases} x, & \text{if } |x| \le \Delta \\ \mathrm{sgn}(x)\Delta, & \text{otherwise.} \end{cases} = \min(1, \Delta/|x|)x. \tag{2.13}$$

The straightforward vector-valued extension ($x \in \mathbb{R}^n$) is then given by $\mathrm{sat}^\Delta(x) = \min(1, \Delta/\|x\|)x$. Finally, $\bar{\mathrm{sat}}^\Delta(x)$ ($\Delta > 0, x \in \mathbb{R}^n$) denotes a sufficiently smooth approximation of the vector-valued saturation function. A typical example can be constructed using the hyperbolic tangent function: $\bar{\mathrm{sat}}^\Delta(x) = \Delta \tanh(\|x\|/\Delta)x/\|x\|$ [119].

2.2 Stability Definitions

Definition 2.1. [142] An equilibrium solution of a dynamical system is said to be *almost globally asymptotically stable* if it is asymptotically stable with an almost global domain of attraction, i.e. the domain of attraction is the entire state space excluding a set of Lebesgue measure zero.

Definition 2.2. [142] An equilibrium solution of a dynamical system is *almost semiglobally exponentially stable* if it is asymptotically stable, and for almost all initial states, there exist finite controller gains or parameters such that the corresponding trajectory exponentially converges to the origin, i.e. the set of initial states that cannot exponentially converge to the origin has zero measure.

2.3 Kinematics of Flight

2.3.1 Vehicle Configuration

We consider aircraft that, for the purposes of this thesis, can be modelled with sufficient accuracy as a single rigid body. The configuration of the vehicle can then be fully specified by the position of a point on the body and the vehicle's orientation.

We make use of standard right-handed coordinate systems for aircraft navigation and control [19]. Let $\{n\}$ denote a local north-east-down (NED) tangent frame, assumed inertial, and let $\{b\}$ denote a body-fixed frame, rigidly attached to the centre of mass of the vehicle, with the x-axis pointing forward in the longitudinal direction, and with the y-axis pointing towards the right wing.

Let $p \in \mathbb{R}^3$ denote the position of the origin of $\{b\}$ relative to $\{n\}$, and $v \in \mathbb{R}^3$ the linear velocity, both defined with respect to $\{n\}$. Then

$$\dot{p} = v. \tag{2.14}$$

Further, let $R_b^n \in \mathrm{SO}(3)$ be the rotation matrix that transforms vectors from $\{b\}$ to $\{n\}$. The matrix R_b^n globally and uniquely describes the orientation, or attitude, of $\{b\}$ relative to $\{n\}$. In particular, the columns of R_b^n are the axes of $\{b\}$ decomposed in $\{n\}$. The time evolution of R_b^n is governed by the kinematic differential equation [80]

$$\dot{R}_b^n = R_b^n \boldsymbol{S}(\omega), \tag{2.15}$$

where $\omega = [p \;\; q \;\; r]^\top \in \mathbb{R}^3$ is the angular velocity of $\{b\}$ relative to $\{n\}$, expressed in $\{b\}$.

2.3.2 Alternative Attitude Representations

Although the rotation matrix R_b^n, i.e. a matrix of nine numbers, represents the orientation globally and uniquely, the constraints (2.4) suggest that more compact attitude representations may be used. In this thesis, we also use both Euler angles and unit quaternions, both of which are extensively used in aerospace applications. We give a brief description of these below. Many other attitude representations can be found in the literature, for which we refer the reader to [163, 226].

Euler Angles

The orientation of a rigid body can also be represented using three coordinates $\Theta = [\phi \;\; \theta \;\; \psi]^T$, where $\phi \in [-\pi, \pi]$, $\theta \in [-\pi/2, \pi/2]$ and $\psi \in [-\pi, \pi]$ are the roll, pitch and yaw angles respectively. The rotation matrix R_b^n can be calculated from Θ using the mapping

$$\boldsymbol{R}_b^n(\Theta) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}, \tag{2.16}$$

where $c_{(\cdot)}$ and $s_{(\cdot)}$ are shorthand notation for $\cos(\cdot)$ and $\sin(\cdot)$, respectively. Analogous to (2.15), the time evolution of Θ is governed by

$$\dot{\Theta} = \boldsymbol{T}_\Theta(\Theta)\omega. \tag{2.17}$$

The matrix $\boldsymbol{T}_\Theta(\Theta)$ relating the angular velocity ω to the time derivative of the Euler angles can be calculated from Θ as follows:

$$\boldsymbol{T}_\Theta(\Theta) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}, \tag{2.18}$$

where $\sec\theta = 1/\cos\theta$. The inverse relation is given by

$$\omega = \boldsymbol{T}_\Theta^{-1}(\Theta)\dot{\Theta} \tag{2.19}$$

with

$$\boldsymbol{T}_\Theta^{-1}(\Theta) = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \tag{2.20}$$

The Euler angles Θ (also called nautical angles, Cardan angles or Tait-Bryan angles) is a minimal three-parameter representation of $SO(3)$ that parameterize a sequence of three elemental rotations, one out of 12 possible sequences. The predominant sequence used in vehicle navigation and control [80, 233] is the intrinsic z-y-x sequence of basic rotations, where (2.16) is the result of the following sequence of basic rotations:

$$\boldsymbol{R}_b^n(\Theta) = R_{z,\psi} R_{y,\theta} R_{x,\phi}, \tag{2.21}$$

where

$$R_{z,\psi} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.22}$$

$$R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.23}$$

$$R_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \tag{2.24}$$

In the rest of this thesis, we will refer to Θ as simply Euler angles. Since the mapping $\Theta \mapsto \boldsymbol{R}_b^n(\Theta)$ is surjective, the Euler angle parameterization has no geometric singularities. However, there is a kinematic singularity present in (2.17). When $|\theta|$ approaches $\pm\pi/2$, the matrix $\boldsymbol{T}_\Theta^{-1}(\Theta)$ in (2.20) becomes rank-deficient and the $\tan\theta$- and $\sec\theta$-terms in (2.18) goes to infinity, causing the derivate $\dot{\Theta}$ to be undefined. This phenomenon is referred to as gimbal-lock [164] and leads to the loss of a degree of freedom in the virtual gimbal system defined by the Euler angle representation. With $|\theta| = \pi/2$, the yaw angle ψ and roll angle ϕ rotations have the same effect on the vehicle orientation. With reference to (2.16) we see that e.g. $\theta = \pi/2$ results in

$$\boldsymbol{R}_b^n(\Theta) = \begin{bmatrix} 0 & -\sin(\psi-\phi) & \cos(\psi-\phi) \\ 0 & \cos(\psi-\phi) & \sin(\psi-\phi) \\ -1 & 0 & 0 \end{bmatrix}. \tag{2.25}$$

Unit Quaternions

Although the Euler angles provide a compact and intuitive attitude representation, the gimbal-lock singularity precludes efficient numerical simulation of arbitrary attitude manoeuvres. Moreover, (2.16) and (2.18) contain a lot of trigonometric terms that are computationally expensive to evaluate. A popular singularity-free attitude representation for efficient numerical simulation is the four-parameter unit quaternion representation (also known as Euler parameters, Euler-Rodrigues parameters or versors).

A unit quaternion $Q \in \mathbb{S}^3$ (satisfying $Q^\top Q = 1$) can be identified with a vector of four numbers $Q = [Q_1 \;\; Q_{2:4}]^\top$, where Q_1 is the scalar part, and $Q_{2:4} = [Q_2 \;\; Q_3 \;\; Q_4]^\top$ is the vector part. The unit quaternion can be identified with a single rotation by an angle φ about some axis k as follows [164]:

$$Q = \begin{bmatrix} \cos(\frac{\varphi}{2}) \\ k\sin(\frac{\varphi}{2}) \end{bmatrix}. \tag{2.26}$$

Further, the time derivative of the unit quaternion is related to the angular velocity ω through the kinematic differential equation

$$\dot{Q} = \boldsymbol{T}_Q(Q)\omega, \tag{2.27}$$

where

$$\boldsymbol{T}_Q(Q) = \frac{1}{2}\begin{bmatrix} Q_1 I_3 + \boldsymbol{S}(Q_{2:4}) \\ -Q_{2:4}^\top \end{bmatrix}. \tag{2.28}$$

The rotation matrix R_b^n can be calculated from Q using

$$\boldsymbol{R}_b^n(Q) = I_3 + 2Q_1\boldsymbol{S}(Q_{2:4}) + 2\boldsymbol{S}^2(Q_{2:4}). \tag{2.29}$$

The mapping $Q \mapsto \boldsymbol{R}_b^n(Q)$ is two-to-one. Antipodal points on $\mathbb{S}^3$, i.e. Q and $-Q$, for any unit quaternion Q, results in the same attitude. This is easily verified by insertion into (2.29) and stems from the fact that $\mathbb{S}^3$ is a double cover of SO(3).

During simulation, small numerical round-off errors cause the unit quaternion Q to deviate from the unit norm constraint. One option is to renormalize every timestep according to $Q \leftarrow Q/\|Q\|$. Another option is to use Corbett-Wright orthogonality control (see e.g. [19]) to maintain the unit-norm constraint during simulation and modify (2.27) to

$$\dot{Q} = \boldsymbol{T}_Q(Q)\omega - \frac{\lambda}{2}(\|Q\|^2 - 1)Q, \tag{2.30}$$

where $\lambda > 0$ is the normalization gain.

An algorithm to convert between unit quaternions and Euler angles can be found in [19].

2.3.3 The Wind Triangle

Let the wind speed be denoted by $v_w \in \mathbb{R}^3$. The air-relative velocity expressed in the inertial frame, $v_a \in \mathbb{R}^3$, is then given by

$$v_a := v - v_w. \tag{2.31}$$

In spherical coordinates (V_a, χ_a, γ_a), we can express v_a as

$$v_a = V_a \begin{bmatrix} \cos(\gamma_a)\cos(\chi_a) \\ \cos(\gamma_a)\sin(\chi_a) \\ -\sin(\gamma_a) \end{bmatrix}, \tag{2.32}$$

where the magnitude of v_a, $V_a := \|v_a\| \in \mathbb{R}_{\geq 0}$ is the airspeed, and the azimuth and elevation angles $\chi_a \in [-\pi, \pi]$, $\gamma_a \in [-\pi/2, \pi/2]$ are the airmass-referenced course and flight-path angles, respectively. When decomposed in $\{b\}$, the relative velocity is normally parameterized in terms of spherical coordinates (V_a, α, β) as

$$v_a^b := (R_b^n)^\top v_a = V_a \begin{bmatrix} \cos(\alpha)\cos(\beta) \\ \sin(\beta) \\ \sin(\alpha)\cos(\beta) \end{bmatrix} = \begin{bmatrix} \mathrm{u}_a \\ \mathrm{v}_a \\ \mathrm{w}_a \end{bmatrix}, \tag{2.33}$$

where $\alpha \in [-\pi, \pi]$ the angle of attack (AoA), and $\beta \in [-\pi/2, \pi/2]$ is the angle of sideslip (AoS). From (2.33), we get the relation

$$\beta = \arcsin\left(\frac{\mathrm{v}_a}{V_a}\right), \quad V_a \neq 0, \tag{2.34}$$

and

$$\alpha = \operatorname{atan2}(\mathrm{w}_a, \mathrm{u}_a), \quad \beta \neq \pm\frac{\pi}{2}. \tag{2.35}$$

The AoA and AoS thus parameterize the orientation of airflow relative to the aircraft and are therefore important quantities when characterizing aerodynamic performance.

Further, the wind velocity is decomposed into

$$v_w = v_{w,s} + R_b^n v_{w,g}, \tag{2.36}$$

Here, $v_{w,s}$ is the steady part, expressed in $\{n\}$, representing the constant (or slowly varying) mean wind velocity. Moreover, a stochastic term $v_{w,g}$ represents gusts and turbulence, expressed in $\{b\}$. Similarly, rotational disturbances are modelled through the wind angular velocity ω_w. The air-relative angular velocity is then defined as:

$$\omega_a = \omega - \omega_a = \begin{bmatrix} p_a \\ q_a \\ r_a \end{bmatrix}. \tag{2.37}$$

The stochastic components of the wind, given by $v_{w,g}$ and ω_w, can be generated by passing white noise through shaping filters given by the Dryden velocity spectra [240][168].

2.3.4 The Wind Frame

Aerodynamic forces are described with reference to the relative airflow, typically expressed using the so-called wind frame $\{w\}$. The x-axis of $\{w\}$ points in the direction of v_a, and the z-axis lies in the symmetry plane of the aircraft, pointing downwards. The axes of $\{w\}$ expressed in $\{b\}$ are the columns of the rotation matrix R_w^b, which transforms vectors from the wind frame to the body-fixed frame. In terms of α and β, we have the mapping

$$\boldsymbol{R}_w^b(\alpha,\beta) = \begin{bmatrix} \cos(\alpha)\cos(\beta) & -\cos(\alpha)\sin(\beta) & -\sin(\alpha) \\ \sin(\beta) & \cos(\beta) & 0 \\ \cos(\beta)\sin(\alpha) & -\sin(\alpha)\sin(\beta) & \cos(\alpha) \end{bmatrix}. \tag{2.38}$$

2.4 Aircraft Equations of Motion

Following [19], the aircraft is modelled as a rigid body of mass $m \in \mathbb{R}_{>0}$ and positive definite inertia matrix $J = J^\top \in \mathbb{R}^{3\times 3}$ with a body frame $\{b\}$ rigidly attached to its centre of mass, moving relative to a NED frame assumed to be inertial $\{n\}$. A standard model of aircraft dynamics is given by the Newton-Euler equations affected by forces and moments due to gravity, aerodynamics, and propulsion effects [19, 233]:

$$m\dot{v} = mge_3 + TR_b^n e_1 + F_a \tag{2.39}$$

$$J\dot{\omega} = \boldsymbol{S}(J\omega)\omega + M_a + M_p, \tag{2.40}$$

where $g \in \mathbb{R}_{>0}$ the gravitational acceleration, $F_a \in \mathbb{R}^3$ the aerodynamic force, expressed in $\{n\}$, and $T \in \mathbb{R}_{\geq 0}$ the thrust force, which is assumed to be aligned with the x-axis of $\{b\}$. M_a and $M_p \in \mathbb{R}^3$ are the aerodynamic moment and propeller moment, respectively, about the vehicle's centre of gravity (CG), expressed in $\{b\}$.

2.4.1 Propulsion Effects

Without loss of generality, we consider propeller-driven aircraft with a single propeller driven by an electric motor. Let $\delta_t \in [0, 1]$ denote the propeller throttle and $\rho \in \mathbb{R}_{>0}$ the density of air. The propeller thrust T can be modelled as [18, 78]

$$\begin{aligned} V_d &= V_a + \delta_t(k_m - V_a) \\ T &= \frac{1}{2}\rho S_p C_p V_d \left(V_d - V_a\right), \end{aligned} \tag{2.41}$$

where V_d is the discharge velocity of air from the propeller and $k_m, S_p, C_p \in \mathbb{R}_{>0}$ is the motor constant, propeller disc area and efficiency factor, respectively.

The propeller moment M_p is mainly due to a reaction torque from the rotating propeller, for which a simple model is given by [19]

$$M_p = \begin{bmatrix} k_Q \delta_t^2 \\ 0 \\ 0 \end{bmatrix}, \tag{2.42}$$

where $k_Q \in \mathbb{R}_{>0}$ is an appropriate constant.

2.4.2 Aerodynamic Forces and Moments

Aerodynamic forces and moments are, in general, nonlinear functions that are difficult to model accurately. Identification of parameters for even simple linear models from flight data remains a challenging problem [113], [180]. Let $u \in \mathbb{R}^p$ denote a vector of p control surface deflections. Following [19], [233] we model the aerodynamic torque as a control-affine (in u) function of u, the angular velocity ω and the body-fixed relative velocity $v_a^b \in \mathbb{R}^3$ as

$$M_a = M_a(\omega, v_a^b) + G(\omega, v_a^b)u, \tag{2.43}$$

where $M_a(\omega, v_a^b) \in \mathbb{R}^3$ is the aerodynamic torque that is independent of u and $G(\omega, v_a^b) \in \mathbb{R}^{3\times p}$ is the control effectiveness matrix. In (2.43), the typical dependence on α, β is captured through v_a^b (cf. (2.33)). Reynolds and Mach number effects are usually ignored for small UAVs moving at airspeeds well below the speed of sound [19].

The aerodynamic forces and moments are formulated in terms of dimensionless aerodynamic coefficients C_* that are, in general, nonlinear functions of α, β and ω. The aerodynamic moment vector, and control effectiveness matrix $G(\omega, v_a^b)$ can be written in general form as [233]

$$M_a(\omega, v_a^b) = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} bC_l(\alpha,\beta,\omega) \\ cC_m(\alpha,\beta,\omega) \\ bC_n(\alpha,\beta,\omega) \end{bmatrix} \tag{2.44}$$

$$G(\omega, v_a^b) = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} bC_{l_u}(\alpha,\beta,\omega)^\top \\ cC_{m_u}(\alpha,\beta,\omega)^\top \\ bC_{n_u}(\alpha,\beta,\omega)^\top \end{bmatrix}, \tag{2.45}$$

where $S, b, c \in \mathbb{R}_{>0}$ are the planform area of the wings, wingspan and aerodynamic chord, respectively. The functions C_l, C_m and C_n are roll, pitch and yaw moment coefficients, respectively, while the vector-valued functions C_{l_u}, C_{m_u} and C_{n_u} map control surface deflections to the resulting torque vector.

Simplified Model for Attitude Control

A first approximation of the aerodynamic moments that is commonly used in the literature [19, 233], and can be useful for control design is given by

$$M_a = h(v_a^b) + V_a D\omega + V_a^2 B u,$$

with

$$\begin{aligned}
h(v_a^b) &= \frac{\rho V_a^2 S}{2} \begin{bmatrix} bC_{l_\beta}\beta \\ c(C_{m_0} + C_{m_\alpha}\alpha) \\ bC_{n_\beta}\beta \end{bmatrix} \\
D &= \frac{\rho S}{4} \begin{bmatrix} b^2 C_{l_p} & 0 & b^2 C_{l_r} \\ 0 & c^2 C_{m_q} & 0 \\ b^2 C_{n_p} & 0 & b^2 C_{n_r} \end{bmatrix} \\
B &= \frac{\rho S}{2} \begin{bmatrix} bC_{l_{\delta_a}} & 0 & bC_{l_{\delta_r}} \\ 0 & cC_{m_{\delta_e}} & 0 \\ bC_{n_{\delta_a}} & 0 & bC_{n_{\delta_r}} \end{bmatrix},
\end{aligned}$$

where the parameters C_* are dimensionless aerodynamic coefficients. In combination with (2.40), the aircraft rotational dynamics can be written as

$$J\dot{\omega} = \boldsymbol{S}(J\omega)\omega + h(v_a^b) + V_a D\omega + V_a^2 B u + M_p. \tag{2.46}$$

Aerodynamic Forces

The components of the aerodynamic force decomposed in the wind frame are $[-D \;\; Y \;\; -L]^\top$, where D is the drag force, Y the side force and L the lift force, defined in terms of aerodynamic coefficients as follows:

$$\begin{bmatrix} D \\ Y \\ L \end{bmatrix} = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} C_D(\alpha, \beta, \omega_a, u) \\ C_Y(\alpha, \beta, \omega_a, u) \\ C_L(\alpha, \beta, \omega_a, u) \end{bmatrix}. \tag{2.47}$$

To transform to the inertial frame, we perform the mapping

$$F_a = R_b^n \boldsymbol{R}_w^b(\alpha, \beta) \begin{bmatrix} -D \\ Y \\ -L \end{bmatrix}. \tag{2.48}$$

Expressions for the force coefficients can be found in any textbook on flight mechanics [19, 231, 233].

Control Surfaces

A typical set of control surfaces is given by $u = [\delta_a \;\; \delta_e \;\; \delta_r]^\top$, where $\delta_a, \delta_e, \delta_r \in \mathbb{R}$ are the aileron, elevator and rudder control surface deflection angles, respectively. These are used to actively control the attitude of the aircraft and produce control moments about each respective axis.

Some aircraft, like the Skywalker X8 in Fig. 1.1a are tailless and do not have any active control moments generated by a rudder. The rotational dynamics, (2.46), thus become underactuated. Instead, the X8 has one elevon control surface on each wing. Collective and differential elevon deflections can be mapped to "virtual" aileron and elevator deflections δ_a and δ_e which are then used as input to the aerodynamic model.

$$\begin{bmatrix}\delta_a \\ \delta_e\end{bmatrix} = \begin{bmatrix}-0.5 & 0.5 \\ 0.5 & 0.5\end{bmatrix}\begin{bmatrix}\delta_{e,r} \\ \delta_{e,l}\end{bmatrix}. \tag{2.49}$$

Actuator Dynamics

Denoting commands with superscript c, the control surface dynamics are given by the following second-order transfer function:

$$\frac{\delta_i(s)}{\delta_i^c(s)} = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}, \quad i = a, e, r, \tag{2.50}$$

and δ_i^c is the commanded deflection angle for actuator δ_i.

2.5 The Coordinated-Turn Equation

A simple relation between roll angle and the resulting turn rate for aircraft is the coordinated-turn equation, which for $|\phi| \neq \pi/2$ can be written as [19]

$$\dot{\psi} = \frac{g}{V_a}\tan\phi. \tag{2.51}$$

The coordinated-turn equation is derived under the assumption of zero sideslip angle and zero acceleration along the body-fixed y-axis. This is in general a desirable flight condition as passenger comfort is increased (in the case of manned aircraft) and drag (which depends on sideslip angle) is decreased. Related expressions also exist that instead relate the roll angle to the course angle or (inertial-frame) lateral acceleration. See [92] for a detailed discussion on the assumptions behind the coordinated-turn equation.

From a guidance and control perspective, the coordinated-turn equation is very useful as it provides a simple way to translate course angle [81, 187] or lateral acceleration [194] commands from an outer-loop guidance system to roll angle references for the inner-loop autopilot. In addition, the coordinated-turn equation can be used to (approximately) calculate a feed-forward signal for $\dot{\psi}$ required in a coordinated turn. The coordinated turn equation (2.51) has an alternative formulation in terms of the course angle [19], which is often used to perform course control. Course control based on the coordinated-turn equation is thoroughly studied in [92].

2.6 ArduPlane

This section presents the main equations used for attitude control and guidance in ArduPlane [13], which is a state-of-the-art open-source autopilot for fixed-wing UAVs. The equations are based on ArduPlane, Release 4.0.9, which is the most recent stable release (as of August 2021). This controller is used as a baseline and to support the discussion in various chapters of this thesis.

2.6.1 Attitude Control

The ArduPlane attitude controller consists of two cascaded single-input-single-output (SISO) feedback loops. The elevator controls pitch angle, while the ailerons are used for roll control. The outer loop consists of proportional controllers, where desired roll and pitch rates $p_r, q_r \in \mathbb{R}$ are calculated according to

$$p_r = k_\phi \left(\phi_r - \phi\right) \tag{2.52}$$

$$q_r = k_\theta \left(\theta_r - \theta\right) + q_{ct}, \tag{2.53}$$

where $k_\phi, k_\theta > 0$ and q_{ct} is the pitch rate offset needed to maintain a constant pitch angle during coordinated turns, given by

$$q_{ct} = \sin(\phi)\cos(\theta)\frac{g}{V_a}\tan(\phi). \tag{2.54}$$

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

$$\delta_a = k_{p,p}\nu^2 \left(p_r - p\right) + \int_0^t k_{i,p}\nu^2 \left(p_r - p\right) d\tau + k_{ff,p}\nu p_r \tag{2.55}$$

$$\delta_e = -k_{p,q}\nu^2 \left(q_r - q\right) - \int_0^t k_{i,q}\nu^2 \left(q_r - q\right) d\tau - k_{ff,q}\nu q_r, \tag{2.56}$$

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 [19].

For UAVs equipped with a rudder, additional control loops utilize the extra control surface for turn coordination.

For elevon planes like the Skywalker X8, the aileron and elevator deflection angles are virtual control signals that are mapped to elevon control actions using the inverse of (2.49):

$$\delta_l = \delta_e + \delta_a \tag{2.57}$$

$$\delta_r = \delta_e - \delta_a. \tag{2.58}$$

2.6.2 Guidance

Lateral guidance is performed using a nonlinear guidance law [193], [194], often called $L1$ guidance, by commanding a lateral acceleration $a_{s_{cmd}}$ using

$$a_{s_{cmd}} = K_{L_1} \frac{V_g^2}{L_1} \sin(\varphi), \tag{2.59}$$

where $V_g := \|v\|$ is the ground speed of the UAV, L_1 is the distance to a reference point on the desired path, ahead of the UAV, K_{L_1} is a tuning parameter, and φ is the angle between the ground speed vector and the L_1 vector pointing from the UAV to the reference point on the path. The desired lateral acceleration is then converted into a desired roll angle ϕ_d using a simplified version of the coordinated turn equation:

$$\phi_d = \cos(\theta) \operatorname{atan}\left(\frac{a_{s_{cmd}}}{g}\right). \tag{2.60}$$

The pitch angle reference is calculated using the total energy control system (TECS) [136]. TECS is based on energy principles, and accounts for dynamic coupling in the longitudinal dynamics of the aircraft by simultaneously controlling altitude and airspeed using pitch and throttle. The pitch angle is used to control the energy distribution E_D, i.e., the difference between (specific/per mass) potential and kinetic energy, given by

$$E_D = gh - \frac{1}{2}V_a^2, \tag{2.61}$$

where h is the altitude. For a desired altitude $h_d \in \mathbb{R}$ and desired airspeed $V_{a,d} \in \mathbb{R}_{>0}$, define the desired specific energy distribution $E_{D,d} = gh_d - V_{a,d}^2/2$ and the error $\tilde{E}_D = E_{D,d} - E_D$. Then, the pitch reference is prescribed as follows:

$$\theta_d = k_1 \tilde{E}_D + k_2 \int_0^t \tilde{E}_D d\tau + k_3 \dot{\tilde{E}}_D + k_4 \dot{E}_{D,d}, \tag{2.62}$$

where $k_i \in \mathbb{R}_{>0}, i = 1 \ldots 4$ are tuning gains.

Part I

Geometric Attitude Control Laws for Fixed-Wing Aircraft

Chapter 3

Introduction to Part I

This chapter is based on material found in the following articles:

[61] E. M. Coates, D. Reinhardt, and T. I. Fossen. Reduced-attitude control of fixed-wing unmanned aerial vehicles using geometric methods on the two-sphere. *21st IFAC World Congress, Berlin, Germany. IFAC-PapersOnLine*, 53:5749–5756, 2020.

[58] E. M. Coates and T. I. Fossen. Geometric reduced-attitude control of fixed-wing UAVs. *Applied Sciences*, 11(7), 2021.

3.1 Introduction

The attitude control system provides the primary stabilization function in autopilots for fixed-wing aircraft. It enables aircraft 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 [19]. The yaw angle, course, and turn rate of aircraft are typically not controlled directly but through banked turn manoeuvres, where the horizontal component of the lift force provides a centripetal force. Therefore, instead of controlling the full attitude, the control objective is to accurately track time-varying roll and pitch angles while simultaneously satisfying the condition of a coordinated turn, Eq. (2.51).

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 3×3 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 [164]. 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 [54]. 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. In contrast, 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 [117].

Controllers designed on SO(3), or using quaternions [250], control the full attitude and, therefore, can not be directly applied to fixed-wing aircraft using banked turn manoeuvres. Instead of studying the full attitude, some authors consider a reduced-attitude representation, evolving on the two-sphere, $\mathbb{S}^2 \subset \mathbb{R}^3$ [41]. In this space of reduced attitude, all rotations related by a rotation about some fixed axis are considered the same [53]. Control systems with reduced attitude evolving on $\mathbb{S}^2$ have previously been studied in the context of spin-axis stabilization of satellites [41], pendulum stabilization [53], path-following control of underwater vehicles [253], control of multirotor UAVs [47] and for general rigid bodies [172].

Part I of this thesis covers novel contributions to the topic of geometric attitude control, adapted specifically for fixed-wing aircraft. The key innovation lies in a careful choice of kinematic representation: To control roll and pitch in a multivariable non-decoupled manner, I propose to use a global representation that evolves on $\mathbb{S}^2$, more specifically a vector representation of reduced attitude that is invariant to rotations about the inertial gravity axis, i.e. independent of the yaw/heading angle of the aircraft. This makes it well-suited for banked turn manoeuvres. The chosen attitude representation is singularity-free and can be exploited to apply proportional feedback along the shortest path in the natural configuration space, giving it an advantage over conventional design methods using Euler angles.

The reduced-attitude representation allows for a convenient decomposition of the dynamics and a corresponding natural 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, one degree of freedom is left to do turn rate control, which essentially performs turn coordination, providing damping about the inertial z-axis and reducing the sideslip angle.

The resulting control design is lightweight, has no singularities, and can be deployed in conjunction with state-of-the-art hierarchical flight control architectures that rely on roll and pitch control in the inner loop, such as [19], and those implemented in open-source autopilots such as ArduPlane [13], and PX4 [174]. The method applies to aircraft with fully actuated rotational dynamics, e.g., those equipped with a full set of control surfaces, such as ailerons, elevator, and rudder. Posing the control problem on $\mathbb{S}^2$ has the advantage of opening up a wide range of tools for control systems on spheres, including different choices of potential functions and methods for global stabilization and tracking using hybrid control.

3.2 Outline of Part I

We first presented the main concepts in [61] and further developed the idea in a series of papers [58, 59, 209]. The remainder of Part I of the thesis consists of four chapters, based on each of these papers. In Chapter 4, the main idea is presented before expanding this work in several directions, considering global stabilization using hybrid control (Chapter 5), a refined more practical design (Chapter 6) and robust control using super-twisting sliding mode control (Chapter 7).

3.3 Related Work

Reduced-attitude control has been extensively applied for thrust-vector control of multirotor UAVs, e.g. [108]. Fixed-wing aircraft, on the other hand, are subject to additional aerodynamic forces and moments that make control of such vehicles fundamentally different. Besides, the reduced-attitude representation used in this thesis (gravity direction represented in body-fixed frame) differs from the thrust direction of multirotors (body-fixed axis represented in the inertial frame). The representation used here is similar to that used to stabilize the inverted equilibrium manifold of the 3-D pendulum in [52, 53].

The idea of separately controlling reduced attitude and another variable decoupled from the reduced-attitude vector is not entirely new. In [162], the reduced attitude is steered along a geodesic path, while the full attitude is stabilized. In [82, 130], the attitude control of a quadrotor is decoupled into thrust-vector control on $\mathbb{S}^2$ and control of the angle of rotation about the thrust vector. A similar approach is taken in [37] with a control allocation strategy prioritizing reduced-attitude correction over yaw errors. Different rotational error metrics for quadrotor control, defined in terms of both full and reduced attitude, are compared in [229]. Finally, in [103], a vector-projection algorithm is used for trajectory tracking for an agile fixed-wing UAV (where aerodynamics are dominated by the propeller). The roll angle is decoupled from the reference attitude so that thrust and lift forces can be pointed to achieve position tracking. Compared to these works, I propose to simultaneously control reduced attitude and an *angular velocity* around the reduced-attitude vector.

Other related work can be found in [191, 192] and [178, 179], where nonlinear attitude controllers for fixed-wing UAVs are developed using quaternions, and that also use a model of the rotational dynamics. In [191, 192], the translational and rotational subsystems are decoupled by estimating the higher-order derivatives of the angle of attack and sideslip angle. This enables controllers for the two subsystems to be designed separately. In [178], a nonlinear PID controller for fixed-wing UAV (full) attitude control is presented. The control law is based on unit quaternions and compensates for aerodynamic coupling effects using integral action. This approach is extended in [179] to apply also to rudderless (i.e., underactuated in attitude) fixed-wing UAVs by using a projection of the quaternion error to a yaw-free subspace. In [201], a gain-scheduled attitude controller based on Euler angles is given. An algorithm for automatic tuning is provided, and the control system is verified experimentally in a wind tunnel.

Chapter 4

Reduced-Attitude Control for Fixed-Wing Aircraft Using Geometric Methods on the Two-Sphere

The main results of this chapter can be found in the following article:

[61] E. M. Coates, D. Reinhardt, and T. I. Fossen. Reduced-attitude control of fixed-wing unmanned aerial vehicles using geometric methods on the two-sphere. *21st IFAC World Congress, Berlin, Germany. IFAC-PapersOnLine*, 53:5749–5756, 2020.

Some supporting material is also taken from

[58] E. M. Coates and T. I. Fossen. Geometric reduced-attitude control of fixed-wing UAVs. *Applied Sciences*, 11(7), 2021.

4.1 Introduction

In this chapter, we develop a smooth, nonlinear reduced-attitude controller for fixed-wing aircraft in a coordinate-free manner. We use a global, singularity-free reduced-attitude representation on the two-sphere $\mathbb{S}^2$, which is particularly well-suited for fixed-wing aircraft since it does not depend on the yaw angle and enables traditional banked-turn manoeuvres.

We combine dynamic inversion, feedforward terms, and PD-like feedback to create the control law. Using Lyapunov methods, we establish almost semiglobal exponential tracking of reduced attitude and, in the case of constant references, almost global asymptotic stability. Due to topological constraints, the latter is the best possible stability result for continuous attitude control systems [26].

In addition to being singularity-free, the geometric approach offers the benefit of directing the proportional action along the shortest path on the sphere toward the reference. Finally, we establish the relation to a classical approach using Euler angles and demonstrate increased efficiency in numerical simulations with a model of a small fixed-wing UAV. The results of this chapter lay the foundations for a novel attitude control framework for fixed-wing aircraft, which we expand in several directions in subsequent chapters.

Chapter Outline

We introduce our choice of reduced-attitude representation in Section 4.2 before proceeding with the problem definition in Section 4.3. The core controller design is presented in Section 4.4 while turn coordination is treated in Section 4.5. A comparison between the geometric controller and a controller based on Euler angles is performed in Section 4.6, and a reduced-attitude reference filter based on roll and pitch angle references is provided in Section 4.7. In Section 4.8, we present the simulation results before giving a summary of the chapter in Section 4.9.

4.2 Reduced-Attitude Representation

Recall from Chapter 2 that $R_b^n \in \mathrm{SO}(3)$ is the rotation matrix that transforms vectors from $\{b\}$ to $\{n\}$, and that $e_3 = [0\ \ 0\ \ 1]^\top$ represents the inertial z-direction (direction of gravitational acceleration). We employ the following reduced-attitude representation:

$$\Gamma = (R_b^n)^\top e_3 \in \mathbb{S}^2, \tag{4.1}$$

which is interpreted as the inertial z-axis, expressed in $\{b\}$. By expanding (4.1) using the roll-pitch-yaw Euler-angle parametrisation of R_b^n, Eq. (2.16), the reduced-attitude vector Γ can be expressed in terms of the roll angle $\phi \in [-\pi, \pi]$ and pitch angle $\theta \in [-\pi/2, \pi/2]$ as follows:

$$\boldsymbol{\Gamma}(\phi, \theta) = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix}. \tag{4.2}$$

Observe that this particular choice of attitude representation is invariant to changes in the heading/yaw angle ψ. The reduced attitude representation is illustrated in Figure 4.1, where a section of the sphere corresponding to $\theta = 0$ is shown. Figure 4.2 shows another section where the aircraft is shown from the side with a nonzero roll angle. As shown, the vector Γ is expressed in the body-fixed frame, and the vector Γ points towards the ground.

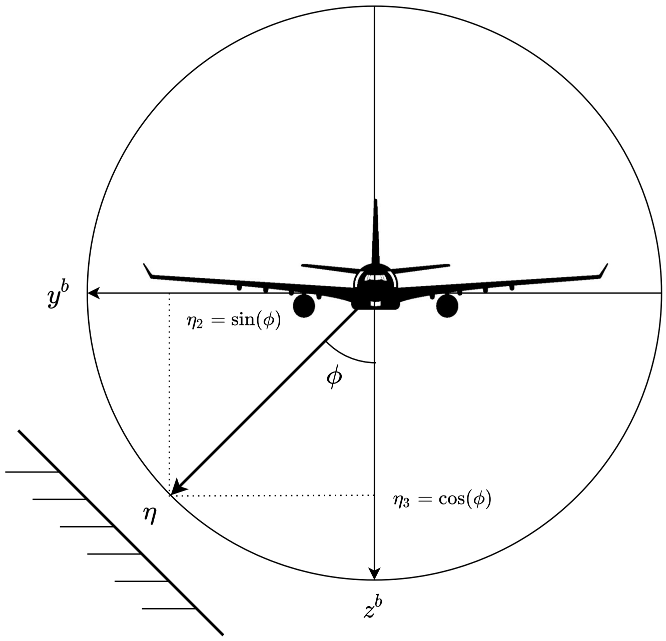


Figure 4.1: Reduced-attitude representation illustrated with a section of the two-sphere corresponding to $\theta = 0$.

The reduced-attitude representation (4.1) is the same as the one considered for the 3-D pendulum in [53], but different compared to the one used for thrust-vector control for multirotor UAVs, which is the thrust direction in the inertial-frame, $R_b^n e_1$ [108].

Note that in light of Eq. (4.1), Γ is equivalent to the third row of the matrix R_b^n, which also appears as the third column of $\boldsymbol{T}_\Theta^{-1}(\Theta)$ in Eq. (2.19). Furthermore, from the mapping (4.2), we recognise ϕ and θ as azimuth and elevation angles, respectively, of the vector Γ. In the language of differential geometry, Eq. (4.2) is a parametrisation of $\mathbb{S}^2$ (excluding the poles) and ϕ, θ are local coordinates. This parametrisation is not valid on the whole sphere since ϕ is not unique at the poles. This is in line with the gimbal-lock singularity of the Euler-angle parametrisation of the full orientation. Instead of utilizing the azimuth and elevation angles ϕ, θ for control of reduced attitude, as done in the existing aerospace and robotics literature, we employ the vector itself, i.e. without using local coordinates on $\mathbb{S}^2$. This lets us work

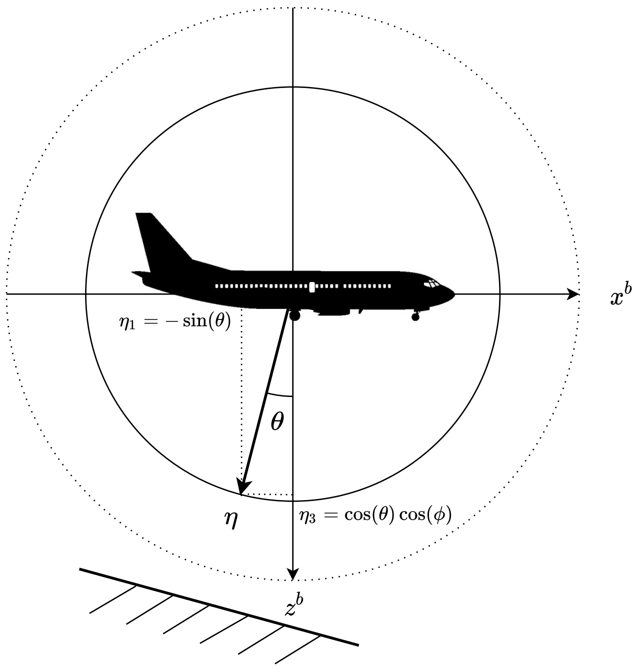


Figure 4.2: Reduced-attitude representation. The aircraft is shown from the side, illustrated with a section of the two-sphere with a non-zero roll angle.

with $\mathbb{S}^2$ globally in a singularity-free manner without the need for multiple coordinate charts.

Further, let $\omega \in \mathbb{R}^3$ be the angular velocity of the body-fixed frame relative to the inertial frame, expressed in the body-fixed frame. From (4.1) and (2.15) we derive the following kinematic differential equation for the reduced-attitude vector Γ:

$$\dot{\Gamma} = \Gamma \times \omega. \tag{4.3}$$

Using (2.11), we can perform an orthogonal decomposition of the angular velocity ω with respect to Γ such that $\omega = \omega^{\perp} + \omega^{\parallel}$, where

$$\omega^{\perp} := \mathbf{\Pi}_{\Gamma}^{\perp}(\omega) \in \mathrm{T}_{\Gamma}\mathbb{S}^2 \qquad \omega^{\parallel} := \mathbf{\Pi}_{\Gamma}^{\parallel}(\omega) \in \mathrm{N}_{\Gamma}\mathbb{S}^2. \tag{4.4}$$

Applying this decomposition of ω in combination with (4.3) gives

$$\dot{\Gamma} = \Gamma \times (\omega^{\perp} + \omega^{\parallel}) = \Gamma \times \omega^{\perp}. \tag{4.5}$$

The parallel component $\omega^{\parallel}$ is the angular velocity about the inertial z-axis (expressed in the body-fixed frame) and does not influence $\dot{\Gamma}$.

Remark 4.1. Note that since the two-sphere $\mathbb{S}^2$ is a two-dimensional manifold, in principle two actuators are sufficient to control reduced attitude. However, since Γ is fixed in the inertial frame, the two required control directions vary with the orientation of the vehicle and are thus not fixed in $\{b\}$. Therefore, we need three linearly independent body-fixed actuators to make the reduced attitude fully controllable throughout the configuration space. In this chapter, we consider only aircraft with fully actuated rotational dynamics, and at each time instant, we will use the remaining degree of freedom (DOF) to stabilize $\omega^{\parallel}$.

For $|\theta| \neq \pi/2$, we can find an expression for $\omega^{\parallel}$ in terms of Euler angles. From (2.11), (2.19)-(2.20), (4.2) and (4.4) we have that, for $|\theta| \neq \pi/2$,

$$\omega^{\parallel} = \Gamma\Gamma^{\top}\omega = \Gamma\Gamma^{\top}\boldsymbol{T}_{\Theta}^{-1}(\Theta)\dot{\Theta} = \Gamma\Gamma^{\top}\left(\dot{\phi}e_1 + \dot{\theta}\begin{bmatrix}0\\ \cos(\phi)\\ -\sin(\phi)\end{bmatrix} + \dot{\psi}\Gamma\right) \tag{4.6}$$

$$= \left(\dot{\phi}\boldsymbol{\Gamma}(\phi,\theta)^{\top}e_1 + \dot{\theta}\boldsymbol{\Gamma}(\phi,\theta)^{\top}\begin{bmatrix}0\\ \cos(\phi)\\ -\sin(\phi)\end{bmatrix} + \dot{\psi}\Gamma^{\top}\Gamma\right)\Gamma = \left(-\dot{\phi}\sin\theta + \dot{\psi}\right)\Gamma. \tag{4.7}$$

If the roll angle ϕ is constant, we get the simplified expression

$$\omega^{\parallel} = \dot{\psi}\Gamma. \tag{4.8}$$

We end this section by calculating expressions for the time derivatives of the projections of the angular velocity vector given by Eq. (4.4). From differentiating (4.4) and by using (4.5) and (2.8) we get

$$\dot{\omega}^{\perp} := \frac{d}{dt}\omega^{\perp} = \boldsymbol{\Pi}_{\Gamma}^{\perp}(\dot{\omega}) + \omega^{\perp} \times \omega^{\parallel} \tag{4.9}$$

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

4.3 Problem Statement

4.3.1 Aircraft Attitude Dynamics

For control design purposes, we consider (4.5) and the rotational dynamics (2.40) in control-affine form:

$$\dot{\Gamma} = \Gamma \times \omega^{\perp} \tag{4.11}$$

$$J\dot{\omega} = f(\omega, v_a^b, \delta_t) + G(\omega, v_a^b)u, \tag{4.12}$$

where

$$f(\omega, v_a^b, \delta_t) = \boldsymbol{S}(J\omega)\omega + M_a(\omega, v_a^b) + M_p(\delta_t) \tag{4.13}$$

defines the drift term, i.e. the torque that is independent of the control input u. Further, we consider aircraft moving with a strictly positive airspeed $V_a > 0$ for which Eq. (4.12) is fully actuated, i.e. the control effectiveness matrix $G(\omega, v_a^b)$ has full row rank. Without loss of generality, we assume that $G(\omega, v_a^b)$ is square and invertible and that the aircraft is equipped with a typical set of linearly independent control surfaces such that $u = [\delta_a \;\; \delta_e \;\; \delta_r]^\top$.

Remark 4.2. From (2.45), it is clear that a strictly positive airspeed is necessary for $G(\omega, v_a^b)$ to have full rank. To generate control moments, aircraft use the control surfaces to deflect the airstream, so a sufficiently large airspeed is needed to ensure controllability.

In what follows, the throttle δ_t and relative velocity v_a^b, and therefore also α, β and V_a (as functions of v_a^b), are treated as known time-varying input signals.

Remark 4.3. Note that since the translational subsystem (2.39) depends on R, ω and u, the relative velocity v_a^b is not truly an exogenous signal. Nevertheless, we assume that v_a^b is available either through measurements of the relative wind or estimated using standard sensors [115, 239]. This should be considered during tuning and when integrating the attitude controller developed in this chapter in GNC systems, for instance by using bandwidth separation between inner and outer control loops. Formally, we consider the system (4.12) along the (bounded) solutions $v_a^b(t)$ of (2.15), (2.31), (2.33) and (2.39), and that the terms of (4.12) satisfy the following assumption to ensure bounded control torques:

Assumption 1. The aerodynamic drift term and control-effectiveness matrix satisfy

$$\|M_a(\omega, v_a^b)\| \leq c_{M_a} \qquad\qquad \|G^{-1}(\omega, v_a^b)\| \leq c_{G^{-1}} \tag{4.14}$$

for some positive constants c_{M_a} and $c_{G^{-1}}$.

4.3.2 Reference System

Analogous to (4.5), let a time-varying reference trajectory for the reduced attitude $\Gamma_d(t) \in \mathbb{S}^2$ satisfy

$$\dot{\Gamma}_d = \Gamma_d \times \omega_d^\perp, \tag{4.15}$$

where the desired angular velocity $\omega_d^\perp \in \mathrm{T}_{\Gamma_d}\mathbb{S}^2$ is twice continuously differentiable. Further, there exist positive constants $c_{\omega_d^\perp}, c_{\dot{\omega}_d^\perp}$ such that

$$\|\omega_d^\perp\| \leq c_{\omega_d^\perp}, \qquad \|\dot{\omega}_d^\perp\| \leq c_{\dot{\omega}_d^\perp}. \tag{4.16}$$

A method that generates reference trajectories $\Gamma_d(t), \omega_d^\perp(t), \dot{\omega}_d^\perp(t)$ satisfying (4.15) based on trajectories parameterised by Euler angles is given in Section 4.7.

Consider the projection of $\omega_d^\perp$ onto the tangent space $\mathrm{T}_\Gamma\mathbb{S}^2$, given by $\mathbf{\Pi}_\Gamma^\perp(\omega_d^\perp) \in \mathrm{T}_\Gamma\mathbb{S}^2$. In a similar fashion to (4.9), the time derivative can be found to satisfy

$$\frac{d}{dt}\left(\mathbf{\Pi}_\Gamma^\perp(\omega_d^\perp)\right) = \mathbf{\Pi}_\Gamma^\perp(\dot{\omega}_d^\perp) + \omega^\perp \times \mathbf{\Pi}_\Gamma^\parallel(\omega_d^\perp) + \mathbf{\Pi}_\Gamma^\parallel(\omega_d^\perp \times \omega^\perp). \tag{4.17}$$

4.3.3 Control Objective

The goal is to design a smooth control input u such that $\Gamma = \Gamma_d$ is an asymptotically stable equilibrium point.

It follows from (4.5), (4.9) and (4.12) that only u for which $J^{-1}G(\omega, v_a^b)u \in \mathrm{T}_\Gamma\mathbb{S}^2$ affects the reduced attitude. Let an orthogonal decomposition of the control input vector u be given by $u = u^\perp + u^\parallel$ such that $J^{-1}G(\omega, v_a^b)u^\perp \in \mathrm{T}_\Gamma\mathbb{S}^2$, and $J^{-1}G(\omega, v_a^b)u^\parallel \in \mathrm{N}_\Gamma\mathbb{S}^2$.

The primary control objective can be stated as follows:

Reduced-Attitude Tracking

Design a smooth state-feedback control law $u^\perp$ such that $\Gamma(t) \to \Gamma_d(t)$ and $\omega^\perp(t) \to \omega_d^\perp(t)$ as $t \to \infty$.

A special case of reduced-attitude tracking is the case when Γ_d is constant and $\omega_d \equiv 0$:

Reduced-Attitude Regulation

Design a smooth state-feedback control law $u^\perp$ such that $\Gamma(t) \to \Gamma_d$ constant and $\omega^\perp(t) \to 0$ as $t \to \infty$.

In both cases, due to topological restrictions to global stabilization on compact manifolds, the strongest stability results that we can achieve are almost global [26].

Secondary Objective

The remaining control direction provided by $u^{\|}$ is in the null space of the orthogonal projection and as such does not interfere with the control of reduced attitude. It should be used for secondary objectives such as turn coordination and to stabilize the parallel component of angular velocity, $\omega^{\|}$.

4.4 Control Design

This section presents the main result of this chapter, where a control law $u^{\perp}$ is given. The design of a control law using the remaining control direction, given by $u^{\|}$ is treated in Section 4.5.

4.4.1 Potential Function

Let a smooth configuration error function $\Psi\colon \mathbb{S}^2 \times \mathbb{S}^2 \to [0, 2]$ be defined by

$$\Psi(\Gamma, \Gamma_d) = \frac{1}{2}\|\Gamma - \Gamma_d\|^2 = 1 - \Gamma_d \cdot \Gamma = 1 - \cos\nu, \tag{4.18}$$

where ν is the angle between Γ and Γ_d. The function Ψ measures the "distance" between two points Γ and Γ_d on $\mathbb{S}^2$, and is clearly positive definite with respect to $\Gamma = \Gamma_d$. There are two critical points: A minimum when $\Gamma = \Gamma_d$, and a maximum when $\Gamma = -\Gamma_d$. In subsequent Lyapunov analysis, Ψ will be used as pseudo-potential energy.

4.4.2 Error States

To design proportional feedback on $\mathbb{S}^2$, let a configuration error vector $e_\Gamma\colon \mathbb{S}^2 \times \mathbb{S}^2 \to \mathrm{T}_\Gamma\mathbb{S}^2$ be given by

$$e_\Gamma = \Gamma \times \Gamma_d, \tag{4.19}$$

and define the angular velocity error as

$$e_\omega = \omega^{\perp} - \mathbf{\Pi}_\Gamma^{\perp}(\omega_d^{\perp}) = \mathbf{\Pi}_\Gamma^{\perp}(\omega - \omega_d^{\perp}) \in \mathrm{T}_\Gamma\mathbb{S}^2. \tag{4.20}$$

From (4.9) and (4.17), the derivative of e_ω can be written as

$$\dot{e}_\omega = \mathbf{\Pi}_\Gamma^{\perp}(\dot{\omega} - \dot{\omega}_d^{\perp} + \omega^{\perp} \times (\omega^{\|} - \mathbf{\Pi}_\Gamma^{\|}(\omega_d^{\perp}))) - \mathbf{\Pi}_\Gamma^{\|}(\omega_d^{\perp} \times e_\omega). \tag{4.21}$$

The error vector e_Γ can be viewed as a gradient vector field on $\mathbb{S}^2$ induced by the potential function Ψ [145], and as $\|e_\Gamma\| = |\sin\nu|$, it vanishes at the critical points of Ψ. The error vector e_Γ is geodesic in the sense that its direction defines an axis of rotation which connects Γ and Γ_d with the shortest possible curve on $\mathbb{S}^2$. In particular, the gradient w.r.t to Γ of $\Psi(\Gamma, \Gamma_d)$, viewed as a function from $\mathbb{R}^3 \times \mathbb{R}^3$ to $\mathbb{R}$ is given by $\nabla_\Gamma \Psi(\Gamma, \Gamma_d) = -\Gamma_d$. However, for functions on general manifolds, the derivative of the function at a point maps between tangent spaces [95]. Further, in light of (2.12), we define a gradient vector field on $\mathbb{S}^2$ by projecting the gradient to the tangent space as $\dot{\Gamma} = -\boldsymbol{\Pi}_\Gamma^\perp(\nabla_\Gamma \Psi(\Gamma, \Gamma_d)) = \boldsymbol{\Pi}_\Gamma^\perp(\Gamma_d)$. To find the rotation axis to apply torque, we form the cross-product

$$\Gamma \times \boldsymbol{\Pi}_\Gamma^\perp(\Gamma_d) = -\boldsymbol{S}^3(\Gamma)\Gamma_d = \boldsymbol{S}(\Gamma)\Gamma_d = e_\Gamma, \tag{4.22}$$

where we have used (2.10).

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

Remark *4.4.* Other configuration error vectors (with corresponding potential functions) on $\mathbb{S}^2$ could be used in place of (4.19), without changing the general approach considered in this chapter. The advantage of using (4.19) for proportional feedback is that it is simple, smooth and globally defined. However, there are some performance issues, since for initial reduced attitudes arbitrarily close to $-\Gamma_d$, the control action will be close to zero, and the reduced attitude will stay there for an extended period of time before converging to the desired reduced attitude. Some alternative error vectors that do not vanish when approaching $-\Gamma_d$ (at the cost of being undefined at this point) can be found in [41], [52] and [205]. Another advantage of using (4.18), (4.19) is that they are compatible with the concept of synergistic potential functions [172], where methods from hybrid control theory can be applied to overcome the topological obstruction to global stabilization that exists when using continuous control laws [26].

4.4.3 Control Law

The control law is a combination of dynamic inversion, feedforward terms and PD-like feedback in terms of e_Γ and e_ω:

$$\begin{aligned} u^\perp = G^{-1}(\omega, v_a^b) J \Big(& - k_p e_\Gamma - \boldsymbol{\Pi}_\Gamma^\perp(K_d e_\omega) \\ & - \boldsymbol{\Pi}_\Gamma^\perp(J^{-1} f(\omega, v_a^b, \delta_t)) - \omega^\perp \times (\omega^\parallel - \boldsymbol{\Pi}_\Gamma^\parallel(\omega_d^\perp)) + \boldsymbol{\Pi}_\Gamma^\perp(\dot{\omega}_d^\perp) \Big), \end{aligned} \tag{4.23}$$

where $k_p \in \mathbb{R}_{>0}$ and $K_d \in \mathcal{P}_+^3$ are tuning parameters (proportional and derivative-like gains respectively).

A notable property of the control law (4.23) is that since the control effectiveness matrix $G(\omega, v_a^b)$ given by (2.45) contains a factor of V_a^2, the inverse matrix $G^{-1}(\omega, v_a^b)$ contains a factor of $1/V_a^2$. This means that the control law includes airspeed scaling, a feature often found in commercial and open-source autopilots [13, 174]. Also note that instead of compensating for the entire drift term $f(\omega, v_a^b, \delta_t)$, only the orthogonal projection is compensated for.

The control law (4.23) is based on proportional action that is proportional to the error term e_Γ as defined by (4.19), whose direction defines an axis of revolution for the direct, shortest rotation connecting Γ and Γ_d (forming a *geodesic* curve on the sphere, i.e. part of a great circle). This is convenient when dealing with large rotation errors and is a property that is not shared with controllers based on Euler angles. We show in later sections that this construction makes the geometric controller spend less control energy than the controller based on Euler angles.

The error vector e_Γ is zero when $\Gamma = \Gamma_d$. However, this is also the case when $\Gamma = -\Gamma_d$. Naturally, this choice of configuration error leads to an additional undesired equilibrium point at $\Gamma = -\Gamma_d$, but due to the topology of the sphere (it is a compact manifold), this is unavoidable when using continuous feedback [26]. The presence of more than one equilibrium prevents us from designing globally stabilizing feedback laws. A suitable notion of stability in this context is the concept of *almost global* asymptotic stability as defined in Definition 2.1. As shown in e.g. [172], we can make the region of attraction global using hybrid control. While the scope of the present chapter is limited to considering smooth control laws, we explore the application of hybrid methods to the attitude control of fixed-wing aircraft in Chapter 5.

4.4.4 Main Result

We state the main stability properties of the closed-loop system in the following proposition, which is the main result of this chapter:

Proposition 4.1 (Reduced-Attitude Tracking): Consider the reduced-attitude error dynamics defined by (4.12), (4.5) and (4.21), assuming $V_a \geq \underline{V}_a > 0$ and that the control effectiveness matrix $G(\omega, v_a^b)$ has full rank. With $k_p \in \mathbb{R}_{>0}$ and $K_d \in \mathcal{P}_+^3$, let a smooth tracking control law $u^\perp$ satisfying $J^{-1}G(\omega, v_a^b)u^\perp \in \mathrm{T}_\Gamma\mathbb{S}^2$ be given by Eq. (4.23). With δ_t, v_a^b treated as bounded, time-varying exogenous signals, the following holds:

1. The closed-loop error system has two equilibrium solutions given by $(\Gamma, e_\omega) = (\pm\Gamma_d, 0)$.
2. The desired equilibrium $(\Gamma_d, 0)$ is exponentially stable, with region of exponential convergence given by

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

$$\|e_\omega(0)\| \leq \sqrt{2k_p\left(\overline{\Psi} - \Psi(\Gamma(0), \Gamma_d(0))\right)}, \tag{4.25}$$

   for some $\overline{\Psi} < 2$, where 2 is the maximum value of Ψ, attained at $\Gamma = -\Gamma_d$.
3. The additional undesired equilibrium $(-\Gamma_d, 0)$ is unstable.
4. Additionally, if $\omega_d^\perp = 0$, the desired equilibrium $(\Gamma_d, 0)$ is almost globally asymptotically stable.

Proof. We break the proof into four parts, starting with the equilibrium points.

Part 1: Equilibrium Points By differentiating (4.19), applying the identity (2.9) and combining with (4.12), (4.21), (4.23) gives the non-autonomous closed-loop error system

$$\dot{e}_\Gamma = -\boldsymbol{S}(\omega_d^\perp)e_\Gamma - \boldsymbol{S}(\Gamma_d)\boldsymbol{S}(\Gamma)e_\omega \tag{4.26}$$

$$\dot{e}_\omega = -k_p e_\Gamma - \boldsymbol{\Pi}_\Gamma^\perp(K_d e_\omega) - \boldsymbol{\Pi}_\Gamma^\parallel(\omega_d^\perp \times e_\omega), \tag{4.27}$$

which gives the equilibrium condition

$$\begin{bmatrix} -\boldsymbol{S}(\omega_d^\perp) & -\boldsymbol{S}(\Gamma_d)\boldsymbol{S}(\Gamma) \\ -k_p I_3 & -\boldsymbol{S}^2(\Gamma)K_d\boldsymbol{S}^2(\Gamma) - \Gamma\Gamma^\top\boldsymbol{S}(\omega_d^\perp) \end{bmatrix} \begin{bmatrix} e_\Gamma \\ e_\omega \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{4.28}$$

where we have used the fact that $-\boldsymbol{S}^2(\Gamma)e_\omega = e_\omega$. For (4.28) to be satisfied for all t, where the time-dependence is implicit through $\Gamma_d(t), \omega_d^\perp(t)$, we get $e_\Gamma = e_\omega = 0$. Note that, when $\omega_d^\perp = 0$, the matrix above is rank-deficient, with $v = [0 \;\; w]^\top$ as a basis of the null space, where w is parallel to Γ. But e_ω lies in $\mathrm{T}_\Gamma\mathbb{S}^2$. Thus, equilibrium solutions are given by $(\Gamma, e_\omega) = (\pm\Gamma_d, 0)$.

Part 2: Exponential Tracking Consider the Lyapunov-like function $V_1\colon \mathrm{T}\mathbb{S}^2 \times \mathbb{S}^2 \to \mathbb{R}_{\geq 0}$

$$V_1(\Gamma, e_\omega, \Gamma_d) = k_p\Psi(\Gamma, \Gamma_d) + \frac{1}{2}e_\omega^\top e_\omega. \tag{4.29}$$

Differentiating along closed-loop trajectories of (4.27) gives

$$\dot{V}_1 = k_p e_\omega^\top e_\Gamma + e_\omega^\top \left(-k_p e_\Gamma - \mathbf{\Pi}_\Gamma^\perp (K_d e_\omega) - \mathbf{\Pi}_\Gamma^\parallel (\omega_d^\perp \times e_\omega)\right) \tag{4.30}$$

$$= -e_\omega^\top K_d e_\omega \leq -\lambda_{\min}^{K_d} \|e_\omega\|^2 \leq 0, \tag{4.31}$$

where the last term in (4.30) disappears since $e_\omega \in \mathrm{T}_\Gamma \mathbb{S}^2$, and we have used (2.12). For initial conditions satisfying (4.24), (4.25), we get $V_1(t_0) \leq k_p \overline{\Psi}$. Since $V_1(t)$ is non-increasing, we get:

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

which means that $\Psi(\Gamma(t), \Gamma_d(t)) \leq \overline{\Psi}$. For the sublevel set

$$L_2 = \left\{\Gamma, \Gamma_d \in \mathbb{S}^2 \colon \Psi(\Gamma, \Gamma_d) \leq \overline{\Psi}\right\} \tag{4.33}$$

we can bound Ψ by

$$\frac{1}{2}\|e_\Gamma\|^2 \leq \Psi(\Gamma, \Gamma_d) \leq \frac{1}{2 - \overline{\Psi}}\|e_\Gamma\|^2. \tag{4.34}$$

Now, consider the Lyapunov function candidate

$$V_2(\Gamma, e_\omega, \Gamma_d) = V_1 + \epsilon e_\omega^\top e_\Gamma, \tag{4.35}$$

where $\epsilon > 0$. Using (4.34), we can derive upper and lower bounds

$$\frac{1}{2} z^\top M_1 z \leq V_2 \leq \frac{1}{2} z^\top M_2 z, \tag{4.36}$$

where $z = [\|e_\Gamma\| \ \|e_\omega\|]^\top$ and

$$M_1 = \begin{bmatrix} k_p & -\epsilon \\ -\epsilon & 1 \end{bmatrix}, \qquad M_2 = \begin{bmatrix} \frac{2k_p}{2-\overline{\Psi}} & \epsilon \\ \epsilon & 1 \end{bmatrix}. \tag{4.37}$$

Differentiating V_2 along the closed-loop trajectories gives

$$\dot{V}_2 = -e_\omega^\top K_d e_\omega + \epsilon \dot{e}_\omega^\top e_\Gamma + \epsilon e_\omega^\top \dot{e}_\Gamma. \tag{4.38}$$

The cross terms can be bounded as follows:

$$\|e_\omega^\top \dot{e}_\Gamma\| \leq \|e_\omega\| \, \|\omega_d^\perp\| \, \|e_\Gamma\| + \|e_\omega\|^2 \leq c_{\omega_d^\perp} \|e_\omega\| \|e_\Gamma\| + \|e_\omega\|^2 \tag{4.39}$$

$$\|e_\Gamma^\top \dot{e}_\omega\| \leq -k_p \|e_\Gamma\|^2 + \lambda_{\max}^{K_d} \|e_\Gamma\| \|e_\omega\|, \tag{4.40}$$

which leads to

$$\dot{V}_2 = -e_\omega^\top K_d e_\omega + \epsilon \dot{e}_\omega^\top e_\Gamma + \epsilon e_\omega^\top \dot{e}_\Gamma \leq -z^\top M_3 z, \tag{4.41}$$

where the matrix M_3 is given by

$$M_3 = \begin{bmatrix} \epsilon k_p & -\frac{\epsilon}{2}\left(c_{\omega_d^\perp} + \lambda_{\max}^{K_d}\right) \\ -\frac{\epsilon}{2}\left(c_{\omega_d^\perp} + \lambda_{\max}^{K_d}\right) & \lambda_{\min}^{K_d} - \epsilon \end{bmatrix} \tag{4.42}$$

If ϵ is chosen to satisfy

$$\epsilon < \min\left\{ \sqrt{k_p}, \frac{4k_p \lambda_{\min}^{K_d}}{4k_p + \left(c_{\omega_d^\perp} + \lambda_{\max}^{K_d}\right)^2} \right\}, \tag{4.43}$$

then M_1, M_2 and M_3 are all positive definite.

By following similar arguments as in the proof of Theorem 4.10 in [125], we get that $V_2(t)$ and $\|z(t)\|$ converge exponentially to zero, which in turn means that $(\Gamma(t), e_\omega(t))$ converge exponentially to $(\Gamma_d(t), 0)$, with the region of exponential convergence given by (4.24) and (4.25).

Part 3: Instability of Undesired Equilibrium To show that the undesired equilibrium is unstable, define

$$W = 2k_p - V_2 \geq -\frac{1}{2}\|e_\omega\|^2 - \epsilon\|e_\omega\|\|e_\Gamma\| + k_p(2 - \Psi(\Gamma, \Gamma_d)). \tag{4.44}$$

At the undesired equilibrium $(-\Gamma_d, 0)$, we have $W = 0$, and $\dot{W} = -\dot{V}_2$ is positive definite from (4.41). Now consider Γ arbitrarily close to $-\Gamma_d$. In this case, the term $2 - \Psi(\Gamma, \Gamma_d)$ is positive, and we can choose $\|e_\omega\|$ sufficiently small such that $W > 0$ and $\dot{W} > 0$. By Theorem 4.3 in [125], the equilibrium $(-\Gamma_d, 0)$ is unstable.

Part 4: Almost Global Regulation When $\omega_d^\perp = 0$, (4.31) reduces to

$$\dot{V}_1 = -(\omega^\perp)^\top K_d \omega^\perp \leq 0, \tag{4.45}$$

so the set given by

$$\begin{aligned} \Omega := \{(\Gamma, \omega^\perp) \in \mathbb{S}^2 \times \mathrm{T}_\Gamma\mathbb{S}^2 : \\ V_1(\Gamma(t), \omega^\perp(t)) \leq V_1(\Gamma(t_0), \omega^\perp(t_0))\} \end{aligned} \tag{4.46}$$

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, $\omega^{\perp} = 0$, which when inserted into (4.27) and using (4.19) leads to $\Gamma = \pm\Gamma_d$. By Theorem 4.4 in [125] (LaSalle), every solution starting in Ω then converges asymptotically to one of the equilibrium solutions $(\pm\Gamma_d, 0)$. Local asymptotic stability of the desired equilibrium point, as well as the instability of the undesired equilibrium, follows from Part 2 and Part 3 of the proof. To establish almost global asymptotic stability of the desired equilibrium, we will study the local structure of the undesired equilibrium.

Let a perturbation of the equilibrium solution $(\Gamma(t), \omega^{\perp}(t)) = (\Gamma_e, 0)$ be given in terms of a perturbation parameter ϵ as $(\Gamma_\epsilon(t,\epsilon), \omega_\epsilon^{\perp}(t,\epsilon)) = (e^{-\epsilon S(\eta(t))}\Gamma_e, \epsilon\delta\omega(t))$, which satisfies $\eta(t)\cdot\Gamma_e = \delta\omega(t)\cdot\Gamma_e = 0$ for all t. Now, consider the perturbed equations of motion (4.5), (4.27) given by

$$\dot{\Gamma}_\epsilon(t,\epsilon) = \Gamma_\epsilon(t,\epsilon) \times \omega_\epsilon^{\perp}(t,\epsilon) \tag{4.47}$$

$$\begin{aligned}\dot{\omega}_\epsilon^{\perp}(t,\epsilon) = -k_p\Gamma_\epsilon(t,\epsilon) \times \Gamma_d - K_d\omega_\epsilon^{\perp}(t,\epsilon)& \\ + \Gamma_\epsilon(t,\epsilon)\Gamma_\epsilon^{\top}(t,\epsilon)K_d\omega_\epsilon^{\perp}(t,\epsilon).&\end{aligned} \tag{4.48}$$

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

$$A(-\Gamma_d) := A = \begin{bmatrix} 0 & I_3 \\ -k_p\boldsymbol{S}^2(\Gamma_d) & -\boldsymbol{S}^2(\Gamma_d)K_d\boldsymbol{S}^2(\Gamma_d) \end{bmatrix}, \tag{4.49}$$

where the relation $-\boldsymbol{S}^2(\Gamma_d)\delta\omega = \delta\omega$ 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

$$Cx = \begin{bmatrix} \Gamma_e^{\top} & 0 \\ 0 & \Gamma_e^{\top} \end{bmatrix} \begin{bmatrix} \eta \\ \delta\omega \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{4.50}$$

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

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

$$v_{i2} = \lambda_i v_{i1} \tag{4.51}$$

$$-k_p\boldsymbol{S}^2(\Gamma_d)v_{i1} - \boldsymbol{S}^2(\Gamma_d)K_d\boldsymbol{S}^2(\Gamma_d)v_{i2} = \lambda_i v_{i2}. \tag{4.52}$$

Inserting (4.51) into (4.52) and pre-multiplying with the complex conjugate transpose $\bar{v}_{i1}^{\top}$ of v_{i1} gives

$$a\lambda^2 + b\lambda - c = 0, \tag{4.53}$$

where

$$a = \bar{v}_{i1}^{\top} v_{i1} > 0 \tag{4.54}$$

$$b = \bar{v}_{i1}^{\top}[-\boldsymbol{S}^2(\Gamma_d)]K_d[-\boldsymbol{S}^2(\Gamma_d)]v_{i1} \geq 0 \tag{4.55}$$

$$c = k_p \bar{v}_{i1}^{\top}[-\boldsymbol{S}^2(\Gamma_d)]v_{i1} \geq 0, \tag{4.56}$$

since the matrix $-\boldsymbol{S}^2(\Gamma_d)$ is positive semi-definite. The coefficients b and c are only (simultaneously) zero when $-\boldsymbol{S}^2(\Gamma_d)v_{i1} = 0$, i.e. when v_{i1} has the form $v_{i1} = z_1\Gamma_d$ for some $z_1 \in \mathbb{C}$. In this case, (4.53) reduces to $a\lambda^2 = 0$, so $\lambda_1 = 0$ is an eigenvalue of A with algebraic multiplicity two, corresponding to the eigenvector $v_1 = [z_1\Gamma_d^{\top} \;\; 0^{\top}]^{\top}$. However, since A has rank five, the geometric multiplicity of λ_1 is one. To get a full Jordan basis, we choose the generalized eigenvector $v_2 = [z_2\Gamma_d^{\top} \;\; z_1\Gamma_d^{\top}]^{\top}$, which satisfies $(A - \lambda_1 I)v_2 = Av_2 = v_1$, for $z_2 \in \mathbb{C}$.

The solutions of the linearized system defined by (4.49) can be written in terms of its Jordan form as

$$x(t) = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_1 t}(v_1 t + v_2) + \sum_{i=3}^{6} c_i h_i(t), \tag{4.57}$$

where the functions $h_i(t)$ depend on the vectors v_j, $j \in \{1, \ldots, 6\}$, the eigenvalues λ_k, $k = \{2, 3, 4, 5\}$ and their multiplicities. The constants c_i depend on the intial condition $x(0) = \sum_{i=1}^{6} c_i v_i$, which satisifies $Cx(0) = 0$. Since the vectors v_1, v_2 do not satisfy the constraints (4.50), $c_1 = c_2 = 0$, the solution $x(t)$ does not depend on λ_1.

Since $A \in \mathbb{R}^{6\times6}$ is a rank five matrix, we know that no other eigenvectors are parallel to v_1, so for all remaining eigenvector pairs, $a, b, c > 0$. Since (4.53) 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 $(\Gamma, \omega^{\perp}) = (-\Gamma_d, 0)$ 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 [94]. 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. □

For almost all $\Psi(\Gamma(t_0), \Gamma_d(t_0)), e_\omega(t_0)$ (excluding $\Gamma(t_0) = -\Gamma_d(t_0)$), some k_p can be chosen such that (4.24), (4.25) is satisfied. The equilibrium $(\Gamma_d, 0)$ is therefore said to be *almost semiglobally exponentially stable* [142].

Remark 4.5. By exploiting passivity-properties in the Lyapunov design, the term $-\omega^\perp \times (\omega^\parallel - \mathbf{\Pi}_\Gamma^\parallel(\omega_d^\perp))$ in (4.23) could be replaced by $-\mathbf{\Pi}_\Gamma^\perp(\omega_d^\perp) \times (\omega^\parallel - \mathbf{\Pi}_\Gamma^\parallel(\omega_d^\perp))$, without changing $\dot{V}$. This would also remove the seemingly unneeded cross-product term in (4.58). However, this would give a closed-loop system that depends on $\omega^\parallel$, and invalidate several arguments used in the proof.

Parts of the proof are inspired by [143], where a tracking controller for a double integrator system on $\mathbb{S}^2$ is presented. However, [143] considers an inertial-frame representation of reduced attitude, as opposed to (4.1), which is defined in the body-fixed frame. Also, no dynamics or parallel components of the angular velocity are considered. In addition to compensating for the dynamics, compared to [143], the controller (4.23) allows a matrix gain K_d, projects the feedforward term $\dot{\omega}_d^\perp$ to $\mathrm{T}_\Gamma\mathbb{S}^2$, and adds an additional term $-\omega^\perp \times \omega^\parallel$ to compensate for the "Coriolis" term that appears when $\omega^\parallel$ is nonzero.

In the special case of regulation, where $\omega_d^\perp = 0$, and $e_\omega = \omega^\perp$, the control law (4.23) reduces to

$$\begin{aligned} u^\perp = G^{-1}(\omega, v_a^b) J \Big(& - k_p e_\Gamma - \mathbf{\Pi}_\Gamma^\perp(K_d \omega^\perp) \\ & - \mathbf{\Pi}_\Gamma^\perp(J^{-1} f(\omega, v_a^b, \delta_t)) - \omega^\perp \times \omega^\parallel \Big). \end{aligned} \tag{4.58}$$

The closed-loop system in this case is autonomous. This means that LaSalle's invariance theorem [125] can be applied. Inspired by the methodology presented for the 3-D pendulum in [53], 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 [53] evolve on $\mathbb{R}^5$, since the state space includes the full angular velocity ω. In [145], 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 [53], but is adjusted to use a linearization on $\mathbb{R}^4$ instead of $\mathbb{R}^5$, inspired by [145], but done in full generality. In addition, a matrix gain K_d is used instead of a scalar.

4.5 Turn Coordination

The control law defined by (4.23) does not inject damping about the axis defined by Γ. The control $u^\parallel$ can be utilized to do this. The coordinated-turn

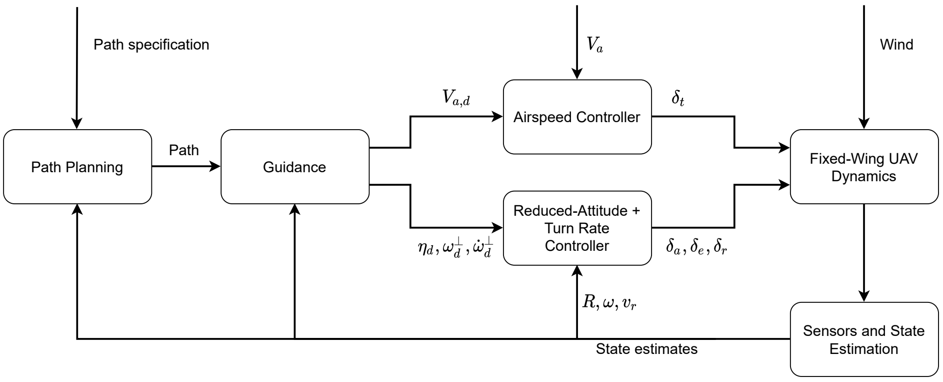


Figure 4.3: Block diagram of a guidance, navigation and control (GNC) architecture for fixed-wing aircraft.

equation can be used to (approximately) calculate a feed-forward signal for the component of angular velocity parallel to Γ required in a coordinated turn. Thus, we combine (4.8) with the coordinated-turn equation, (2.51), to generate a reference for $\omega^{\|}$:

$$\omega_d^{\|} = \frac{g}{V_a}\tan(\phi)\Gamma = \frac{g\Gamma_2}{V_a\Gamma_3}, \quad |\Gamma_1| \neq 1. \tag{4.59}$$

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 (4.59) or by using the reference Γ_d instead, which we can constrain as we want. A controller that adds damping about Γ without interfering with the banked-turn manoeuvre is then given by

$$u^{\|} = G^{-1}(\omega, v_a^b)J\left(-k_{tc}(\omega^{\|} - \omega_d^{\|}) - \mathbf{\Pi}_\Gamma^{\|}(J^{-1}f(\omega, v_a^b, \delta_t)) + \mathbf{\Pi}_\Gamma^{\|}(\dot{\omega}_d^{\|})\right), \tag{4.60}$$

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

As an alternative to (4.60), for some $k_\beta \in \mathbb{R}_{>0}$, consider the following control law, which is designed to drive the sideslip angle to zero:

$$u^{\|} = G^{-1}(\omega, v_a^b)J\,\mathbf{\Pi}_\Gamma^{\perp}(k_\beta \beta e_3). \tag{4.61}$$

We continue the discussion on the design of $\omega_d^{\|}$ in Chapter 6, Section 6.3.

Figure 4.3 shows a block diagram that illustrates how the combined reduced-attitude and turn rate controller integrates into a typical GNC architecture for fixed-wing aircraft. The references for reduced-attitude, angular velocity, and angular acceleration are generated by some outer-loop guidance controller, and the reduced-attitude control law is combined with a control law for airspeed control, e.g. a PI-controller [19]. The controller uses estimates of the rotation matrix R, the angular velocity ω, as well as the relative velocity v_a^b, which are all made available through a state estimation module. The use of v_a^b is relaxed in Chapter 6.

4.6 Comparison With Controller Based on Euler Angles

In this section, the structure of the geometric controller presented in Section 4.4 is compared to a controller based on Euler (azimuth and elevation) angles. For $K_\omega \in \mathcal{P}_+^3$, consider the cascaded dynamic inversion based controller

$$u = G^{-1}(\omega, v_a^b) J \Big(- K_\omega(\omega - \bar{\omega}_d) - J^{-1} f(\omega, v_a^b, \delta_t) \Big), \tag{4.62}$$

where the bar in $\bar{\omega}_d$ is introduced to distinguish it from $\omega_d^\perp$ in (4.15). The desired angular velocity is computed using (2.51) and linear feedback from the roll and pitch regulation errors $\tilde{\phi} := \phi - \phi_d$, $\tilde{\theta} := \theta - \theta_d$ as follows:

$$\bar{\omega}_d = \boldsymbol{T}_\Theta^{-1}(\Theta) \begin{bmatrix} -k_\phi \tilde{\phi} \\ -k_\theta \tilde{\theta} \\ \frac{g}{V_a} \tan(\phi) \end{bmatrix} \tag{4.63}$$

where $k_\phi, k_\theta \in \mathbb{R}_{>0}$, and $\boldsymbol{T}_\Theta^{-1}(\Theta)$ is given by (2.20).

Remark 4.6. This controller has a similar structure as the control architecture used in the PX4 open source autopilot [174], but there the dynamic inversion term is replaced by an integral term.

To compare the geometric controller from Section 4.4 with the Euler angle controller (4.62)-(4.63), we consider the regulation case with $\omega_d^\perp = 0$, and set $K_d = K_\omega = k_d I_3$, $k_\theta = k_\phi = k_p/k_d$, and (4.60) is used for $u^\parallel$ with $k_{tc} = k_d$.

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

$$\dot{\omega} = -k_p e_{\theta\phi} - k_d \omega + k_d \omega_d^\parallel, \tag{4.64}$$

where the error vector $e_{\theta\phi}$ is given by

$$e_{\theta\phi} := \begin{bmatrix} \tilde{\phi} & \tilde{\theta}\cos(\phi) & -\tilde{\theta}\sin(\phi) \end{bmatrix}^\top . \tag{4.65}$$

For the geometric controller, if we ignore the cross-product term in (4.58) and the derivative term in (4.60), the closed-loop angular velocity dynamics becomes

$$\dot{\omega} = -k_p e_\Gamma - k_d \omega + k_d \omega_d^{\parallel}. \tag{4.66}$$

The only difference between (4.66) and (4.64) lies in the proportional error vectors. By calculating $\Gamma \cdot e_{\theta\phi} = -\tilde{\phi}\sin(\theta) \neq 0$, we see that $e_{\theta\phi} \notin \mathrm{T}_\Gamma\mathbb{S}^2$, so the proportional action has a different direction than the geodesic direction defined by e_Γ. The error vectors also have different magnitudes, but this can be changed using a different potential function (see Remark 4.4). In the simulation study of Section 4.8.2 we normalize the magnitude of the error vectors by replacing e_Γ with $e'_\Gamma = \|e_{\theta\phi}\| \cdot e_\Gamma / \|e_\Gamma\|$, which enables us to compare the controllers on equal grounds.

4.7 Reference Filter

An expression for the reduced-attitude vector Γ in terms of the Euler angles roll and pitch is given by (4.2). Differentiating leads to

$$\dot{\Gamma} = \begin{bmatrix} -\cos(\theta)\dot{\theta} \\ -\sin(\theta)\sin(\phi)\dot{\theta} + \cos(\theta)\cos(\phi)\dot{\phi} \\ -\sin(\theta)\cos(\phi)\dot{\theta} - \cos(\theta)\sin(\phi)\dot{\phi} \end{bmatrix}. \tag{4.67}$$

For $\omega^\perp \in \mathrm{T}_\Gamma\mathbb{S}^2$ we can invert (4.5) using (4.2) and (4.67) to get

$$\omega^\perp = \dot{\Gamma} \times \Gamma = \begin{bmatrix} \cos^2(\theta)\dot{\phi} \\ \cos(\phi)\dot{\theta} + \sin(\theta)\cos(\theta)\sin(\phi)\dot{\phi} \\ -\sin(\phi)\dot{\theta} + \sin(\theta)\cos(\theta)\cos(\phi)\dot{\phi} \end{bmatrix}, \tag{4.68}$$

with derivative

$$\dot{\omega}^\perp = \ddot{\Gamma} \times \Gamma, \tag{4.69}$$

where $\ddot{\Gamma} = [\ddot{\Gamma}_1,\ \ddot{\Gamma}_2,\ \ddot{\Gamma}_3]^\top$ can been found by differentiating (4.67), resulting in

$$\ddot{\Gamma}_1 = \sin(\theta)\dot{\theta}^2 - \cos(\theta)\ddot{\theta} \tag{4.70}$$

$$\ddot{\Gamma}_2 = -\cos(\theta)\sin(\phi)(\dot{\theta}^2 + \dot{\phi}^2) - 2\sin(\theta)\cos(\phi)\dot{\theta}\dot{\phi} - \sin(\theta)\sin(\phi)\ddot{\theta} + \cos(\theta)\cos(\phi)\ddot{\phi} \tag{4.71}$$

$$\ddot{\Gamma}_3 = -\cos(\theta)\cos(\phi)(\dot{\theta}^2 + \dot{\phi}^2) + 2\sin(\theta)\sin(\phi)\dot{\theta}\dot{\phi} - \sin(\theta)\cos(\phi)\ddot{\theta} - \cos(\theta)\sin(\phi)\ddot{\phi}. \tag{4.72}$$

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 [80]), the relations (4.2) and (4.67) - (4.72) can be used to generate continuous signals $\Gamma_d(t)$, $\omega_d^{\perp}(t)$, $\dot{\omega}_d^{\perp}(t)$, which are needed to implement the tracking controller (5.8).

4.8 Numerical Example

This section presents some simulation results using a model of the Aerosonde UAV [19] with a simple proportional-integral (PI) controller for airspeed and a constant reference of 35 m/s. In the tracking example, (4.61) is used for $u^{\parallel}$, while (4.60) is used in the regulation case. The angular velocity is initialized to zero in both cases, while the controller parameters are set to $k_p = 9.5$, $K_d = 8I$ and $k_\beta = 10$.

4.8.1 Tracking

Consider a tracking scenario, where a trajectory $(\Gamma_d(t),\ \omega_d^{\perp}(t), \dot{\omega}_d^{\perp}(t))$ has been generated using (4.2) and (4.67) - (4.72), $\phi_d(t) = 60\frac{\pi}{180}\cos(0.1 \cdot 2\pi t)$, $\theta_d(t) = 30\frac{\pi}{180}\cos(0.08{\cdot}2\pi t)$ and their analytical first and second order derivatives. Initial reduced attitude is set using $\phi(0) = -70°$ and $\theta(0) = -30°$. Fig. 4.4 shows that reduced attitude, visualized using roll and pitch angles, converges to the desired trajectory from large initial errors, while the angular velocity error goes to zero. The angle of attack, sideslip angle and control surface deflection angles, which attain reasonable values throughout the manoeuvre, are displayed in Fig. 4.5.

4.8.2 Regulation

Now consider a regulation case, where $\omega_d^{\perp} = 0$. The constant reference is generated using the mapping (4.2) with $\phi_d = 60°$ 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 4.6, the magnitude of the error vector (4.19) is scaled for comparison with the Euler angle controller. Fig. 4.6 shows that the UAV performs a banked turn manoeuvre with approximately constant turn rate, and roll and pitch angles converge in both cases. For this specific manoeuvre, the geometric controller seems to have a slightly faster response in pitch. The difference can be more clearly understood by looking at Fig. 4.8. The response of the geometric controller is shown to make the UAV take the shortest path between Γ and Γ_d, while the controller based on Euler angles does not. Fig. 4.7 indicates that this makes the geometric controller spend

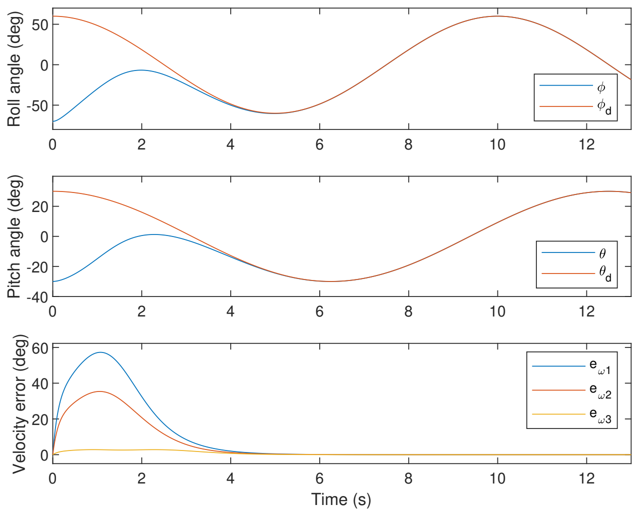


Figure 4.4: Tracking scenario: Roll, pitch and angular velocity error.

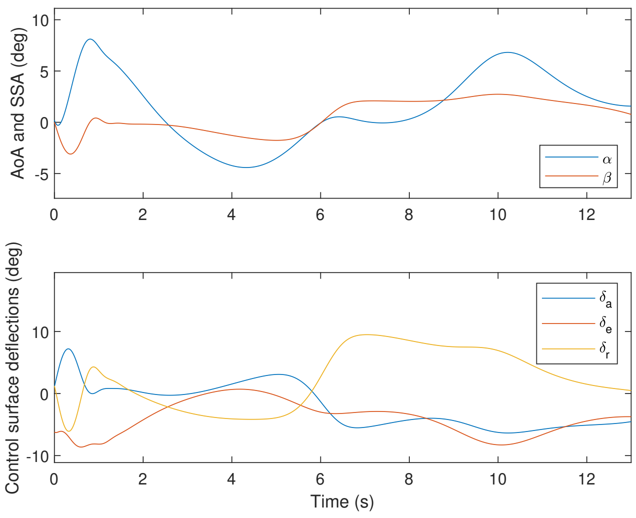


Figure 4.5: Tracking scenario: Angle of attack (AoA), sideslip angle (SSA) and control surface deflection angles.

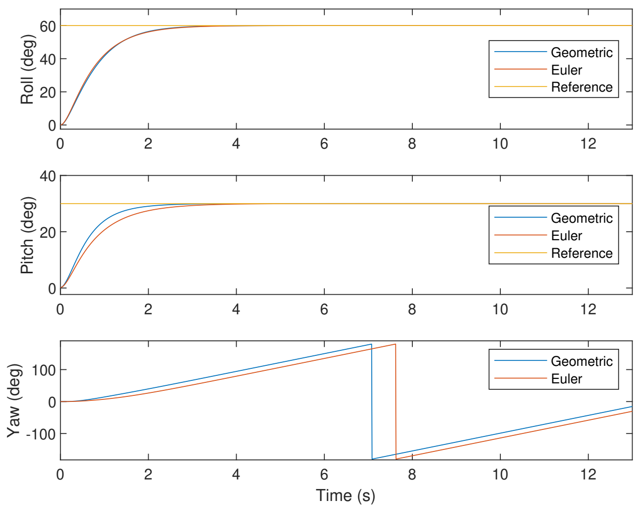


Figure 4.6: Regulation scenario: Euler angle comparison.

less control energy. However, the sideslip angle is smaller, which results in a reduced magnitude of the compensated drift term $f(\omega, v_a^b, \delta_t)$. Further investigation should compare the two controllers in more realistic scenarios with uncertain $f(\omega, v_a^b, \delta_t)$, for instance, compensated using integral action.

4.9 Chapter Summary

In this chapter, we introduced a novel geometric reduced-attitude controller especially suited for fixed-wing aircraft, with several advantages over conventional roll and pitch angles. The main novelty is designing the controller using a kinematic representation on the two-sphere that is independent of the aircraft's heading/yaw angle. Almost semiglobal exponential stability and almost global asymptotic stability properties are shown through Lyapunov stability analysis for tracking and regulation objectives, respectively.

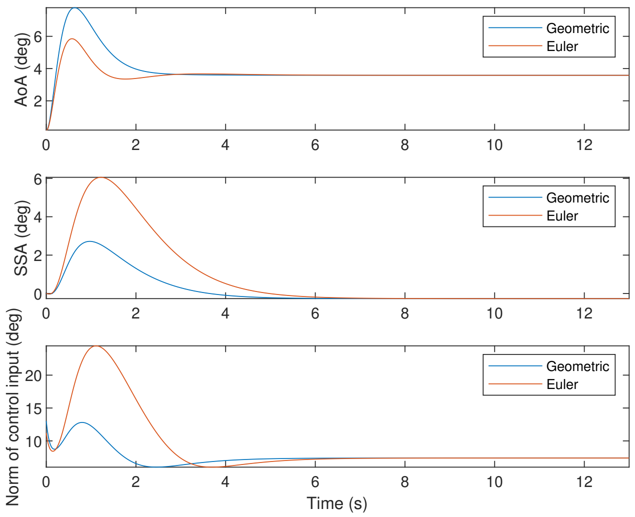


Figure 4.7: Regulation scenario: Angle of attack, sideslip angle and norm of control input.

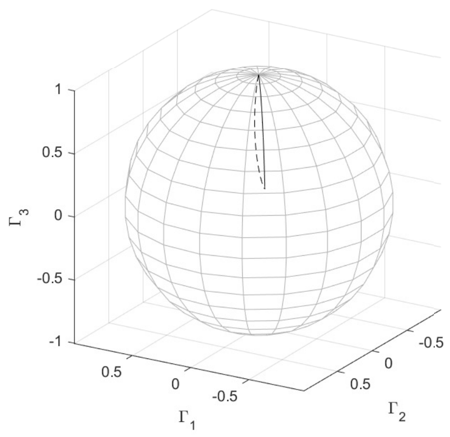


Figure 4.8: Regulation scenario: Trajectories on the two-sphere. Solid: Geometric controller. Dashed: Controller based on Euler angles.

Chapter 5

Hybrid Control of Fixed-Wing Aircraft for Large-Angle Attitude Maneuvers on the Two-Sphere

This chapter is based on the following article:

[209] D. Reinhardt, E. M. Coates, and T. A. Johansen. Hybrid control of fixed-wing UAVs for large-angle attitude maneuvers on the two-sphere. *21st IFAC World Congress, Berlin, Germany. IFAC-PapersOnLine*, 53:5717–5724, 2020.

5.1 Introduction

As explained in the previous chapter, the topological properties of compact manifolds such as the two-sphere preclude global stability results [26]. The strongest possible stability results using continuous feedback are almost global. Continuous controllers, such as (4.23), generate an additional unstable (saddle) equilibrium point where the gradient of the potential function vanishes. This problem may seem purely academic since in practice, small perturbations cause the system trajectories to leave the unstable point. However, there are also performance issues [53]: due to the continuous dependence on initial conditions [125], trajectories starting close to the stable manifold of the unstable equilibrium can stay near this point for an extended period of time before converging to the reference [145]. Global asymptotic stability can be achieved using discontinuous feedback (e.g. [132]). However, discontinuous feedback is not robust to arbitrarily small measurement noise, which can, in the worst case, cause the system to get stuck at the unstable equilibrium point [169]. In fact, [170] shows that when an equilibrium point cannot be globally stabilized by continuous feedback, it cannot be robustly globally stabilized using discontinuous feedback either. To overcome the topological obstructions, hybrid controllers for robust global tracking of full attitude have been proposed in [169] using quaternions, and on SO(3) in [21, 142, 173]. Similarly, for reduced attitude on $\mathbb{S}^2$, global asymptotic stability was achieved using hybrid control in [143, 172] and [47] with application to general rigid bodies and multirotor UAVs.

To the best of our knowledge, these methods have not been targeted at fixed-wing aircraft. In this chapter, we expand on the result of Chapter 4 and inspired by [47, 143, 172], we propose a hybrid controller for fixed-wing aircraft that guarantees global exponential tracking of reduced-attitude references on the two-sphere. The design is based on the notion of synergistic potential functions [171] and overcomes the topological obstruction to global attitude stabilization using a hysteresis-based switching law between two different gradient-based proportional feedbacks. The controller is well-suited for the recovery from large attitude disturbances for which we demonstrate the performance in a numerical example.

Chapter Outline

The rest of this chapter is structured as follows: In Section 5.2 we present the hybrid controller design. The simulation results can be found in Section 5.3. Finally, we provide a summary of the chapter in Section 5.4.

5.2 Hybrid Control Design

The dynamic model, reference model and control objective are the same as in the previous chapter, as stated in Section 4.3. The goal of this chapter is to achieve global asymptotic stability by designing a hybrid controller, in particular using the notion of synergistic potential functions [171].

5.2.1 Hybrid Dynamical Systems

We proceed by introducing the framework presented in [86] where a hybrid system can be defined in the following form:

$$\dot{\xi} \in \mathcal{F}(\xi), \;\; \xi \in \mathcal{C} \tag{5.1}$$

$$\xi^{+} = \mathcal{G}(\xi), \;\; \xi \in \mathcal{D}. \tag{5.2}$$

When the state ξ is inside the flow set $\mathcal{C}$, its continuous motion is governed by the differential inclusion $\dot{\xi} \in \mathcal{F}(\xi)$. Complementary, when the state is inside the jump set $\mathcal{D}$ it evolves in the form of discrete jumps with its dynamics governed by the difference equation $\xi^+ = \mathcal{G}(\xi)$. Solutions to (5.1)-(5.2) are parametrised by $(t, i) \in \mathbb{R}_{\geq 0} \times \mathbb{N}$, i.e. both the amount of time passed, t, and i, the number of discrete jumps.

In the following, we describe the design of a hybrid controller which employs proportional feedback based on a synergistic potential function as presented in [172], 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 $\mathcal{D}$.

5.2.2 Potential Function

The design of a synergistic potential functions follows the approach presented in [143, 172] and employs a discrete logic variable $q \in \mathcal{Q}$ that specifies at each time instant which mode, out of two, is active. To this end, let $\mathcal{Q} = \{0, 1\}$. The *nominal mode* $q = 0$ drives the reduced attitude towards the nominal reference Γ_d. The *expelling mode* $q = 1$ will be designed such that the critical points of its potential function are at a maximum distance to both the nominal reference and its antipodal point, i.e. they evolve on the great circle on $\mathbb{S}^2$ orthogonal to Γ_d.

Let the reference in the expelling mode be denoted by $s_d \in \mathbb{S}^2$ and satisfy

$$\dot{s}_d = s_d \times \omega_d^\perp. \tag{5.3}$$

It follows from (4.15), (5.3) and the identity (2.8) that when s_d is initialized orthogonal to Γ_d, i.e. satisfies $s_d(0) \cdot \Gamma_d(0) = 0$, it holds that $s_d(t) \cdot \Gamma_d(t) = 0$ for all future time.

As in [172] we define the synergistic potential functions as

$$\Psi_q(\Gamma, \Gamma_d, s_d) = \begin{cases} 1 - \Gamma \cdot \Gamma_d & \text{if } q = 0 \\ a + b(1 - \Gamma \cdot s_d) & \text{if } q = 1, \end{cases} \tag{5.4}$$

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 Γ_d. The gradient of Ψ_q with respect to Γ is given by

$$\nabla_\Gamma \Psi_q(\Gamma, \Gamma_d, s_d) = \begin{cases} -\Gamma_d & \text{if } q = 0 \\ -b s_d & \text{if } q = 1. \end{cases} \tag{5.5}$$

Note that since $\mathbb{S}^2$ is a compact manifold and b is finite, $\|\nabla_\Gamma \Psi_q(\Gamma, \Gamma_d, s_d)\|$ is bounded.

5.2.3 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 constructing a gradient vector field on $\mathbb{S}^2$ (see Section 4.4.2) and use a proportional feedback proportional to the error vector

$$e_{\Gamma q} = -\Gamma \times \nabla_\Gamma \Psi_q(\Gamma, \Gamma_d, s_d) \in \mathrm{T}_\Gamma \mathbb{S}^2. \tag{5.6}$$

Depending on the mode q, $e_{\Gamma q}$ then becomes

$$e_{\Gamma q} = \begin{cases} \Gamma \times \Gamma_d & \text{if } q = 0 \\ b(\Gamma \times s_d) & \text{if } q = 1. \end{cases} \tag{5.7}$$

Further, the angular velocity error $e_\omega \in \mathrm{T}_\Gamma \mathbb{S}^2$ is defined as before, by (4.20).

5.2.4 Control Law

In the following, we let $x = (\Gamma, \omega^\perp, \Gamma_d, s_d, \omega_d^\perp) \in \Xi$ be the continuous state of the hybrid system which evolves in the space $\Xi := \mathbb{S}^2 \times \mathrm{T}_\Gamma \mathbb{S}^2 \times \mathbb{S}^2 \times \mathbb{S}^2 \times \mathrm{T}_{\mathbf{\Gamma}_d} \mathbb{S}^2$. The full state is then given by $\xi = (x, q) \in \Xi \times \mathcal{Q}$.

In the nominal mode, the control law equals (4.23), but to accommodate the expelling mode, we replace e_Γ in the proportional feedback term of (4.23) with $e_{\Gamma q}$ to get a hybrid feedback law:

$$u^\perp = G^{-1}(\omega, v_a^b) J\Big(-\mathbf{\Pi}_\Gamma^\perp (J^{-1} f(\omega, v_a^b, \delta_t)) + \kappa(x, q)\Big), \tag{5.8}$$

where

$$\kappa(x, q) = -\omega^\perp \times (\omega^\parallel - \mathbf{\Pi}_\Gamma^\parallel(\omega_d^\perp)) + \mathbf{\Pi}_\Gamma^\parallel(\dot{\omega}_d^\perp) - k_p e_{\Gamma q} - \mathbf{\Pi}_\Gamma^\perp(K_d e_\omega), \tag{5.9}$$

with proportional gain $k_p \in \mathbb{R}_{>0}$ and positive definite gain matrix $K_d = {K_d}^\top \in \mathbb{R}^{3\times 3}$.

As was the case with the nominal control law (4.23) this control law only affects $\dot{\omega}^\perp$. As discussed in Section 4.5, this leaves the possibility for an independent design of a control law $u^\parallel \in \mathrm{N}_\Gamma \mathbb{S}^2$ for other objectives such as turn coordination.

5.2.5 Jump Map

The binary logic variable q changes according to the equations

$$\dot{q} = 0 \qquad (x, q) \in \mathcal{C} \tag{5.10}$$

$$q^+ = 1 - q \quad (x, q) \in \mathcal{D}. \tag{5.11}$$

During the continuous flow of the system, the mode stays constant, while (5.11) describes how q toggles between $q = 0$ and $q = 1$ in the jump set.

In the nominal mode, the error vector $e_{\Gamma q}$ drives the reduced attitude in the direction on the sphere that lies on the path of minimal distance between Γ and Γ_d, known as the minimal geodesic path [41]. To extend this path to the case where the expelling mode is active, s_d^+ is chosen as

$$s_d^+ \in g_{s_d}(\Gamma, \Gamma_d, s_d) := \begin{cases} \frac{\Gamma_d \times \Gamma}{\|\Gamma_d \times \Gamma\|} \times \Gamma_d & \text{if } \Gamma \neq \pm\Gamma_d \\ s_d & \text{otherwise,} \end{cases} \tag{5.12}$$

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 that the mapping is well-defined also when $\Gamma = \pm\Gamma_d$.

5.2.6 Closed-loop System

We can now describe the closed-loop dynamics of the hybrid system. The continuous kinematics of Γ, Γ_d, s_d are given by (4.5), (4.15) and (5.3), respectively. The closed-loop dynamics for $\omega^\perp$ are given by (4.9), (4.12) and the control law (5.8). Both references are governed by the same kinematic equation as the reduced-attitude vector. Furthermore, the uniform bounds (4.16) can be equivalently stated as

$$\omega_d^\perp \in c_{\omega_d^\perp}\mathbb{B} \tag{5.13}$$

$$\dot{\omega}_d^\perp \in c_{\dot{\omega}_d^\perp}\mathbb{B}. \tag{5.14}$$

The differential inclusion, (5.14) allows us to formulate an autonomous closed-loop system such that hybrid invariance principles can be applied in the stability analysis [169].

The resulting continuous motion of the closed-loop system is governed by

$$\begin{bmatrix} \dot{\Gamma} \\ \dot{\omega}^{\perp} \\ \dot{\Gamma}_d \\ \dot{s}_d \\ \dot{\omega}_d^{\perp} \\ \dot{q} \end{bmatrix} \in \mathcal{F}(x,q) := \begin{bmatrix} \Gamma \times \omega^{\perp} \\ \kappa(x,q) + \omega^{\perp} \times \omega^{\parallel} \\ \Gamma_d \times \omega_d^{\perp} \\ s_d \times \omega_d^{\perp} \\ c_{\dot{\omega}_d^{\perp}} \mathbb{B} \\ 0 \end{bmatrix}. \tag{5.15}$$

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 governed by

$$\begin{bmatrix} \Gamma^{+} \\ \omega^{\perp +} \\ \Gamma_d^{+} \\ s_d^{+} \\ \omega_d^{\perp +} \\ q^{+} \end{bmatrix} = \mathcal{G}(x,q) := \begin{bmatrix} \Gamma \\ \omega^{\perp} \\ \Gamma_d \\ g_{s_d}(\Gamma, \Gamma_d, s_d) \\ \omega_d^{\perp} \\ 1-q \end{bmatrix}. \tag{5.16}$$

5.2.7 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 the synergy gap [172]. It is defined as

$$\mu(\Gamma, q) = \Psi_q(\Gamma, \Gamma_d, s_d) - \min_{\nu \in \mathcal{Q}} \Psi_\nu(\Gamma, \Gamma_d, s_d). \tag{5.17}$$

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

$$\mathcal{C} = \{(x,q) : \mu(\Gamma, q) \leq \delta\}, \tag{5.18}$$

$$\mathcal{D} = \{(x,q) : \mu(\Gamma, q) \geq \delta\}. \tag{5.19}$$

The following proposition establishes conditions on the potential function such that it is synergistic and positive definite relative to Γ_d:

Proposition 1: Let the sets $\mathcal{C}$, $\mathcal{D}$ be given by (5.18) and (5.19), with synergy gap μ defined in (5.17). Then the potential function Ψ_q in (5.4) is a synergistic potential function with gap exceeding δ satisfying

$$0 < \delta < \min\{2 - a - b, a - 1, a + 2b - 1\}. \tag{5.20}$$

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 $q \in \mathcal{Q}$ as

$$\mathrm{Crit}\Psi_q = \{(\Gamma, \Gamma_d, s_d) \in (\mathbb{S}^2)^3 : e_{\Gamma q} = 0\}. \tag{5.21}$$

From the definition of $e_{\Gamma q}$ it follows that at all critical points, the reduced attitude Γ is parallel to $\nabla_\Gamma \Psi_q$.

The set of all critical points follows as $\cup_{q\in\mathcal{Q}}\mathrm{Crit}\,\Psi_q = \{(\pm\Gamma_d), (\pm s_d)\}$. For $\{\Psi_q\}_{q\in\mathcal{Q}}$ to be synergistic relative to Γ_d with gap exceeding δ, the condition $\cup_{q\in\mathcal{Q}}\mathrm{Crit}\,\Psi_q \setminus \{\Gamma_d\} \subset \mathcal{D}$ needs to be satisfied. At $(\Gamma, q) = (-\Gamma_d, 0)$, the potential function evaluates to $\Psi_0(-\Gamma_d) = 2$ and $\Psi_1(-\Gamma_d) = a + b$, where we use the fact that $s_d \cdot \Gamma_d = 0$. To include that point in the jump set, the synergy gap needs to satisfy

$$\mu(-\Gamma_d, 0) = 2 - a - b > \delta > 0. \tag{5.22}$$

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

$$\mu(-s_d, 1) = a - 1 > \delta > 0 \tag{5.23}$$
$$\mu(+s_d, 1) = a + 2b - 1 > \delta > 0. \tag{5.24}$$

Then by [172, Proposition 1], the potential function Ψ_q is synergistic relative to Γ_d with synergy gap exceeding δ given by (5.20). □

The closed-loop hybrid system is defined such that it satisfies the hybrid basic conditions [86, Assumption 6.5] which makes it nominally robust to measurement noise. The next proposition provides a collection of these conditions.

Proposition 2: Consider the sets $\mathcal{C}$ in (5.18), $\mathcal{D}$ in (5.19) and the maps $\mathcal{F}$, $\mathcal{G}$ in (5.15),(5.16). Then, the following is satisfied:

(i) The sets $\mathcal{C}$ and $\mathcal{D}$ are closed.

(ii) The map $\mathcal{F}$ is outer semicontinuous and locally bounded relative to $\mathcal{C}$ and $\mathcal{F}(x, q)$ is convex for every $(x, q) \in \mathcal{C}$.

(iii) 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 $q \in \mathcal{Q}$ and that the minimum of two continuous functions is continuous. The synergy gap μ in (5.17) then is the difference between 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 $(x, q) \in \mathcal{C}$ such that $\omega_d^\perp$, $\dot{\omega}_d^\perp$ 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 semicontinuity follows from [86, Lemma 5.10], which shows (ii). Further, note that $\mathbb{S}^2$ is compact and hence $s_d^+ = g_{s_d}(\Gamma, \Gamma_d, s_d) \in \mathbb{S}^2$ is locally bounded relative to $\mathcal{D}$ and the graph of $g_{s_d} : \mathbb{S}^2 \times \mathbb{S}^2 \mapsto \mathbb{S}^2$ given by

$$\operatorname{gph} g_{s_d} = \{(\Gamma, \Gamma_d, s_d) \in \mathbb{S}^2 \times \mathbb{S}^2 \times \mathbb{S}^2 : s_d \in g_{s_d}(\Gamma, \Gamma_d, s_d\} \tag{5.25}$$

is closed. Since $\mathcal{D}$ is closed, outer semicontinuity of g_{s_d} relative to $\mathcal{D}$ follows from [86, Lemma 5.10]. □

5.2.8 Main Result

We summarize the stability results with two propositions.

Proposition 3: Let the parameters a, b, $\delta \in \mathbb{R}$ be chosen such that Ψ_q in (5.4) satisfies (5.20) according to Proposition 1. Consider the closed-loop hybrid system $\mathcal{H} = (\mathcal{C}, \mathcal{F}, \mathcal{D}, \mathcal{G})$ with $\mathcal{F}$, $\mathcal{G}$ defined in (5.15), (5.16) and the sets $\mathcal{C}$, $\mathcal{D}$ given by (5.18), (5.19). Then the set

$$\mathcal{A} = \{(x, q) \in \Xi \times \mathcal{Q} : \Gamma = \Gamma_d,\ \omega^\perp = \omega_d^\perp\} \tag{5.26}$$

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

Proof. During flows of $\mathcal{H}$, Ψ_q evolves according to

$$\dot{\Psi}_q = e_{\Gamma q} \cdot e_\omega. \tag{5.27}$$

The time derivative of the hybrid configuration error vector is given by

$$\dot{e}_{\Gamma q} = -\boldsymbol{S}(\omega_d^\perp) e_{\Gamma q} + \boldsymbol{S}(\nabla_\Gamma \Psi_q) \boldsymbol{S}(\Gamma) e_\omega. \tag{5.28}$$

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

$$V(x, q) = k_p \Psi_q + \frac{1}{2} e_\omega{}^\top e_\omega. \tag{5.29}$$

Analogous to (4.31), it follows from (5.27) and (4.21) that for $(x, q) \in \mathcal{C}$ along solutions of the closed-loop system

$$\dot{V}(x, q) \leq -\lambda_{\min}^{K_d} \|e_\omega\|^2 := u_c(x). \tag{5.30}$$

It follows from K_d being positive definite that $u_c(x) \leq 0$ such that $V(x,q)$ is non-increasing along flows. In $\mathcal{D}$, the mode is switched to the lower potential which leads to the difference during jumps

$$V(\mathcal{G}(x,q)) - V(x,q) = -k_p\delta := u_d. \tag{5.31}$$

This shows that the growth of $V(x,q)$ along solutions to $\mathcal{H}$ is bounded by $u_c(x) \leq 0$ and $u_d < 0$. Note that by requiring the reference to be bounded, the dynamics of the closed-loop system in (5.15) and (5.16) are autonomous and hybrid invariance principles can be applied. Then by [86, Theorem 8.8] we have that for an arbitrary $c \in V(\Xi, \mathcal{Q})$, each precompact solution to $\mathcal{H}$ converges to the nonempty set that is the largest weakly invariant subset of

$$\Omega = V^{-1}(c) \cap \mathrm{cl}\,(u_c^{-1}(0)), \tag{5.32}$$

where $V^{-1}(c)$ denotes the preimage of the Lyapunov function candidate at c and $u_c^{-1}(0)$ denotes the preimage of u_c at 0 for which $\mathrm{cl}\,(u_c^{-1}(0))$ gives the closure. Then from (5.30) we see that $\mathrm{cl}\,(u_c^{-1}(0))$ leads to $e_\omega = 0$ which implies $\dot{e}_\omega = 0$. We substitute this into (4.21) to see that $e_{\Gamma q} = 0$ which gives $(\Gamma, q) \in \mathrm{Crit}\,\Psi$. Since all critical points except $(\Gamma_d, 0)$ are included in $\mathcal{D}$, it follows that $\mathcal{A}$ is the largest weakly invariant subset of Ω. Thus all precompact solutions of $\mathcal{H}$ converge to $\mathcal{A}$. Further note that $\mathcal{A}$ is compact and $\mathrm{cl}\,(\mathcal{C}) \cup \mathcal{D} = \Xi \times \mathcal{Q}$ and therefore $\mathcal{G}(\mathcal{D}) \subset \mathrm{cl}\,(\mathcal{C}) \cup \mathcal{D}$. Since $V(x,q)$ is positive definite with respect to $\mathcal{A}$ it follows from [86, Theorem 8.8] and [86, Corollary 8.9 (iii)] that $\mathcal{A}$ is globally asymptotically stable. $\square$

It follows from (5.7) and (4.20) that inclusion in the set $\mathcal{A}$ implies $e_{\Gamma q} = 0$ and $e_\omega = 0$. 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 4: Let the parameters a, b, $\delta \in \mathbb{R}$ be chosen such that Ψ_q in (5.4) satisfies (5.20) according to Proposition 1. Consider the closed-loop hybrid system $\mathcal{H} = (\mathcal{C}, \mathcal{F}, \mathcal{D}, \mathcal{G})$ with $\mathcal{F}$, $\mathcal{G}$ defined in in (5.15), (5.16) and the sets $\mathcal{C}$, $\mathcal{D} \subset \Xi \times \mathcal{Q}$ given by

$$\mathcal{C} = \{(x,q) : \mu(\Gamma, q) \leq \delta \ \text{ or } \ \|e_\omega\| \geq B_{e_\omega}\}, \tag{5.33}$$

$$\mathcal{D} = \{(x,q) : \mu(\Gamma, q) \geq \delta \ \text{ and } \ \|e_\omega\| \leq B_{e_\omega}\} \tag{5.34}$$

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

$$\mathcal{A} = \{(x,q) \in \Xi \times \mathcal{Q} : \Gamma = \Gamma_d,\ \omega^\perp = \omega_d^\perp\} \tag{5.35}$$

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

Proof. The aim of this proof is to show global exponential stability as defined in [236]. We first show that the potential function is uniformly quadratic [42]. Since all critical points other than Γ_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 [142]) as

$$\Psi_q(\Gamma, \Gamma_d, s_d) \leq \frac{1}{2}\gamma \|e_{\Gamma q}\|^2. \tag{5.36}$$

To show a lower bound, we use scaling in each mode and define b_q as $b_0 = 1$ and $b_1 = b$. Analogous to (4.34), it then follows from (5.7), that the potential function is uniformly bounded by

$$\frac{1}{2b_q}\|e_{\Gamma q}\|^2 \leq \Psi_q(\Gamma, \Gamma_d, s_d) \leq \frac{1}{2}\gamma \|e_{\Gamma q}\|^2. \tag{5.37}$$

Using (5.29), let a Lyapunov function candidate be defined as

$$V_\epsilon(x, q) = V(x, q) + \epsilon e_\omega^\top e_{\Gamma q} \tag{5.38}$$

for some $\epsilon \in \mathbb{R}_{>0}$. From (5.37) and defining $z = \begin{bmatrix}\|e_{\Gamma q}\| & \|e_\omega\|\end{bmatrix}^\top$ we see that $V_\epsilon(x, q)$ can be bounded by

$$\frac{1}{2}z^\top M_1 z \leq V_\epsilon(x, q) \leq \frac{1}{2}z^\top M_2 z, \tag{5.39}$$

where the matrices $M_1, M_2 \in \mathbb{R}^{2\times 2}$ are given by

$$M_1 = \begin{bmatrix}\frac{k_p}{b_q} & -\epsilon \\ -\epsilon & 1\end{bmatrix}, \qquad M_2 = \begin{bmatrix}k_p\gamma & \epsilon \\ \epsilon & 1\end{bmatrix}. \tag{5.40}$$

Next, we show that there exists $\lambda > 0$ such that along solutions of the closed-loop dynamics, $V_\epsilon(t, i) := V_\epsilon(x(t, i), q(t, i))$ can be bounded by

$$V_\epsilon(t, i) \leq V_\epsilon(0, 0)\exp(-\lambda t). \tag{5.41}$$

During jumps, the difference in V_ϵ is given by

$$\begin{aligned}
V_\epsilon(\mathcal{G}(x, q)) - V_\epsilon(x, q) &= -k_p\delta + \epsilon {e_\omega}^\top(e_{\Gamma q^+} - e_{\Gamma q}) && (5.42)\\
&\leq -k_p\delta + \epsilon\|e_\omega\|\|\Gamma \times (\Gamma_d - b s_d)\| && (5.43)\\
&\leq -k_p\delta + \epsilon\|e_\omega\|(\|\Gamma_d\| + |b|\|s_d\|) && (5.44)\\
&\leq -k_p\delta + \epsilon B_{e_\omega}(1 + |b|). && (5.45)
\end{aligned}$$

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

$$\epsilon \leq \frac{k_p\delta}{B_{e_\omega}(1 + |b|)}, \tag{5.46}$$

which shows that $V_\epsilon(x,q)$ is non-increasing during jumps. Next we show that there exists a positive definite matrix $M_3 \in \mathbb{R}^{2\times 2}$ such that along flows, $V_\epsilon(x,q)$ satisfies

$$\dot{V}_\epsilon(x,q) \le -z^\top M_3 z. \tag{5.47}$$

The upper bound for $\dot{V}(x,q)$ is given by (5.30) and it remains to find a bound for the time-derivative of the cross-term in (5.38). From (5.28), (4.21) we see that

$$\begin{aligned}\frac{d}{dt}(e_\omega^\top e_{\Gamma q}) = e_\omega^\top(\boldsymbol{S}(\nabla_\Gamma \Psi_q)\boldsymbol{S}(\Gamma))e_\omega - k_p\|e_{\Gamma q}\|^2 & \\ - {e_{\Gamma q}}^\top(K_d + \boldsymbol{S}(\omega_d^\perp))e_\omega & \quad (5.48)\\ \le \|\nabla_\Gamma\Psi_q\|\|e_\omega\|^2 - k_p\|e_{\Gamma q}\|^2 & \\ + (\lambda_{\max}^{K_d} + B_{\omega_d})\|e_{\Gamma q}\|\|e_\omega\|. & \quad (5.49)\end{aligned}$$

From (5.30) and (5.47) we find that

$$M_3 = \begin{bmatrix} \epsilon k_p & -\frac{\epsilon}{2}(\lambda_{\max}^{K_d} + B_{\omega_d}) \\ -\frac{\epsilon}{2}(\lambda_{\max}^{K_d} + B_{\omega_d}) & \lambda_{\min}^{K_d} - \epsilon\|\nabla_\Gamma\Psi_q\| \end{bmatrix}. \tag{5.50}$$

The matrices M_1, M_2, M_3 are positive definite for any ϵ satisfying

$$\epsilon < \min_{q\in\mathcal{Q}}\left\{\sqrt{\frac{k_p}{b_q}}, \frac{4\lambda_{\min}^{K_d}}{4k_p\|\nabla_\Gamma\Psi_q\| + (\lambda_{\max}^{K_d} + B_{\omega_d})^2}\right\}, \tag{5.51}$$

which shows that (5.41) is satisfied for all initial conditions with $\lambda = \lambda_{\min}^{M_3}$. We can then apply [236, Theorem 1] to conclude global exponential convergence of $V_\epsilon(x,q)$ to zero. Then $V_\epsilon = 0(x,q)$ if and only if $\Psi_q = 0$ and $e_\omega = 0$ and hence $\Gamma \to \Gamma_d$ and $\omega^\perp \to \omega_d^\perp$, which shows that $\mathcal{A}$ is globally exponentially stable. □

Note in the proof to Proposition 4 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 4.

5.3 Numerical Example

We use a model of the Aerosonde UAV with nonlinear aerodynamics as described in [19]. We compare two controllers: first, a continuous controller that employs the nominal mode throughout the simulation and second, the

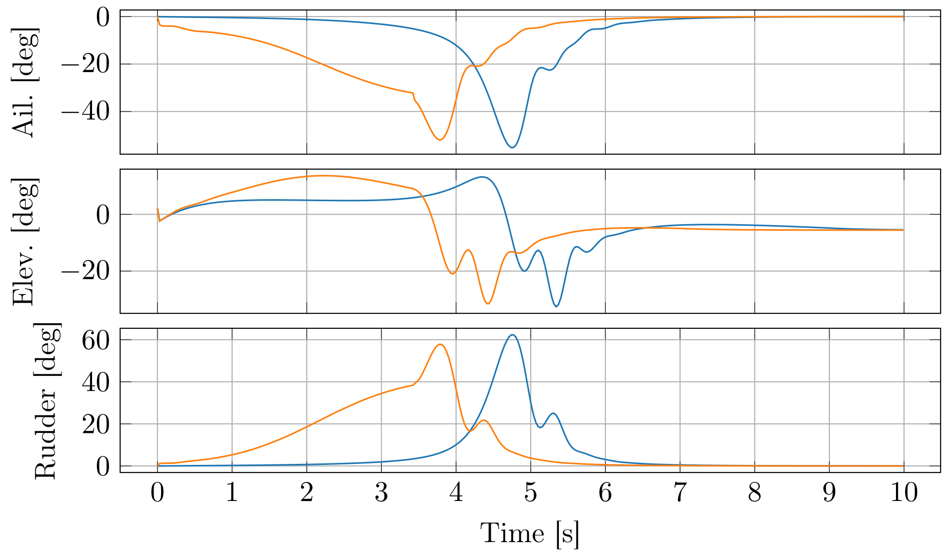


Figure 5.1: Control surface deflections, respectively aileron δ_a, elevator δ_e and rudder δ_r, for the continuous controller (blue) and the hybrid controller (orange).

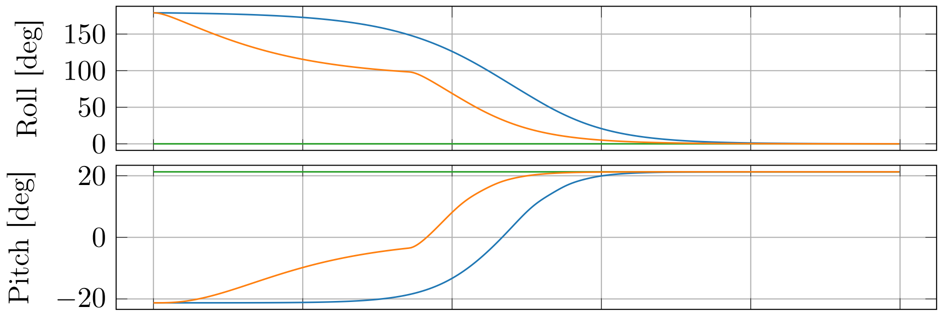


Figure 5.2: Attitude response towards the reference (green) represented by roll angle ϕ and pitch angle θ for the continuous controller (blue) and the hybrid controller (orange).

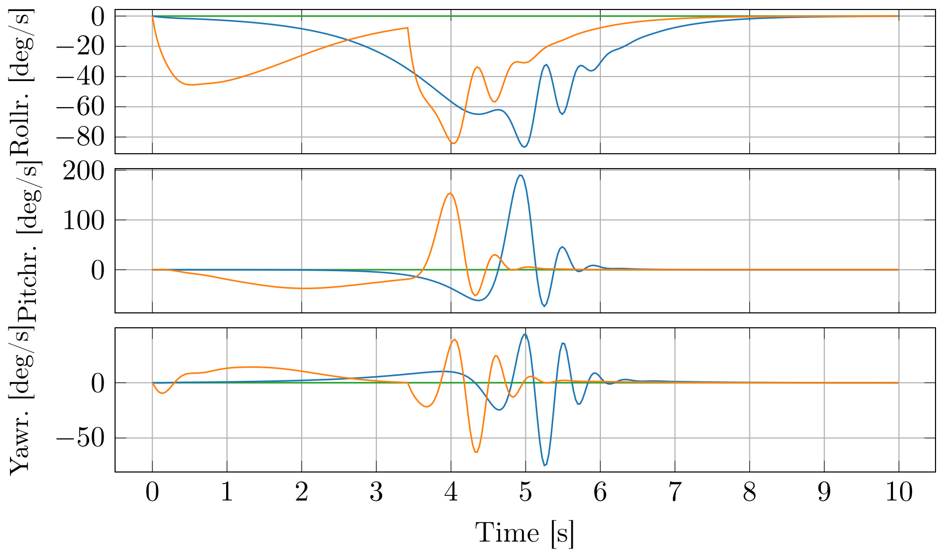


Figure 5.3: Angular rate response compared to the reference (green), represented by roll rate p, pitch rate q and yaw rate r, for the continuous controller (blue) and the hybrid controller (orange).

presented hybrid controller. In the simulations, the airspeed of the UAV is controlled via a PI controller using the propeller throttle, δ_t.

The controller parameters are chosen as $k_p = 9.5$, $K_d = 8I_3$, $a = 1.25$ and $b = 0.6$. We simulate the recovery from a large initial attitude disturbance and set the initial state such that $\Gamma(0) = -\exp(e_1\epsilon)\Gamma_d$ with $\epsilon = \pi/180$. In terms of Euler angles, this corresponds to a roll angle of -179 degrees and a pitch angle of -21.26 degrees. The initial yaw angle is set to zero. The reference is parameterized according to (4.2) 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 Figure 5.2, the continuous controller remains close to the initial attitude for up to 3 seconds, whereas the hybrid controller reacts instantly and uses the expelling potential, which has a larger gradient, until 3.5 seconds into the simulation before switching to the nominal potential (cf. Figure 5.5). As a consequence, the hybrid controller recovers faster from the descend at a lower speed (cf. Figure 5.4) and returns to ascending flight two seconds before the continuous controller, with similar actuator usage (cf.

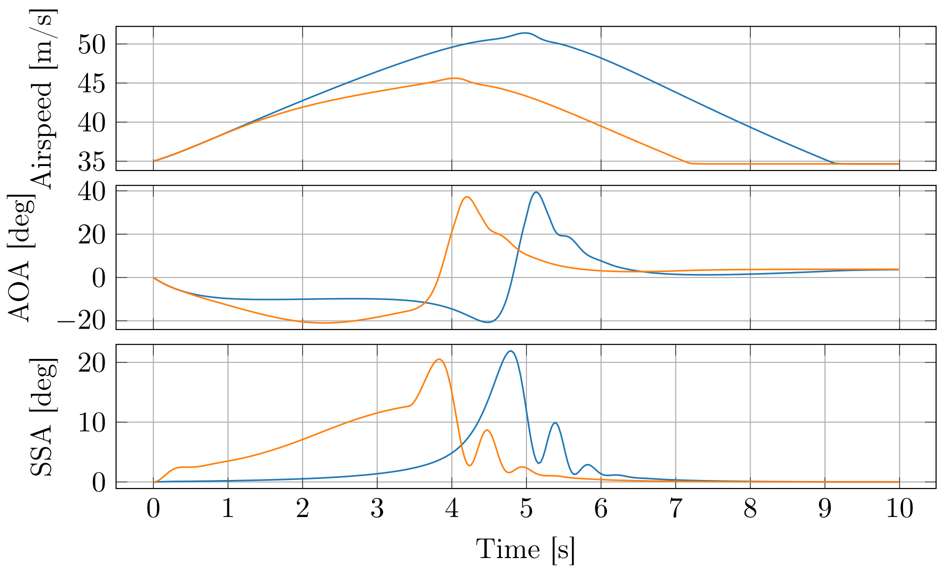


Figure 5.4: The relative velocity v_a^b represented by airspeed V_a, angle of attack α (AOA) and sideslip angle β (SSA) for the continuous controller (blue) and the hybrid controller (orange).

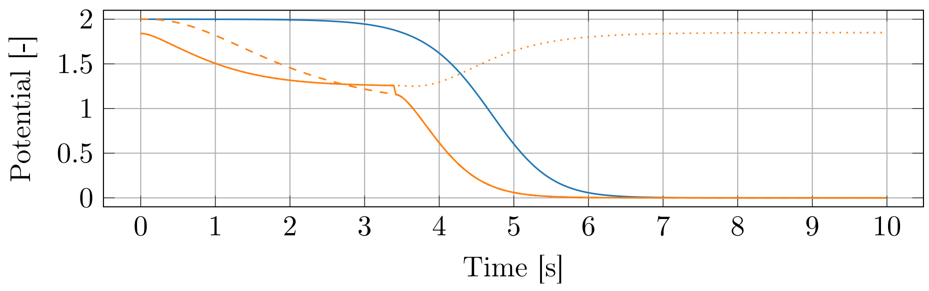


Figure 5.5: Trajectories of the potential functions for the continuous controller (blue) and the hybrid controller (orange). The values for the nominal potential function Ψ_0 (dashed), the expelling potential function Ψ_1 (dotted), and the activated potential function Ψ_q (solid) are shown.

Figure 5.1). However, a drawback of the hybrid controller is the deceleration close to the expelling reference observed in Figure 5.3 and Figure 5.5. This suggests using a dynamic extension in which the control action is given by a dynamic weighting of both configuration error vectors as done in [21] or [172]. Future work will also address the 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.

5.4 Chapter Summary

In this chapter, we extended the results of Chapter 4 by using hybrid control, in particular synergistic potential functions, to overcome the topological obstructions to robust global reduced-attitude tracking. Using two different jump sets, we were able to prove global asymptotic and global exponential stability, respectively, using Lyapunov stability analysis for hybrid dynamical systems. The efficacy of the design was illustrated in a simulated scenario where a fixed-wing UAV is subject to large initial attitude errors.

Chapter 6

Almost Global Geometric Reduced-Attitude Tracking Control of Fixed-Wing Aircraft

This chapter is based on the following article:

[58] E. M. Coates and T. I. Fossen. Geometric reduced-attitude control of fixed-wing UAVs. *Applied Sciences*, 11(7), 2021.

6.1 Introduction

In the two preceding chapters, we introduced a nonlinear autopilot design for multivariable reduced-attitude control of fixed-wing aircraft. To control roll and pitch, we employ vector coordinates evolving on $\mathbb{S}^2$ that are independent of the yaw/heading angle. There are, however, some shortcomings of the nominal control law (4.23). It uses dynamic inversion, which requires a lot of knowledge about the system dynamics. It is well known that control laws based on dynamic inversion are sensitive to model uncertainty and often use excessive control energy to cancel system nonlinearities [125].

In this chapter, we build upon the design methodology presented in Chapter 4. By using a pertinent model of the aircraft rotational dynamics, we exploit model structure to design more robust and practical controllers that are less dependent on perfect knowledge about the system dynamics. The stability results are also less restrictive than in Chapter 4. Through the use of an extended version of Barbalat's lemma [175], we prove almost global asymptotic tracking with a simpler proof than that of Proposition 4.1. The main simplification results from a redesign of the angular velocity error, where we now also include the parallel part of angular velocity.

Using Lyapunov theory, almost global asymptotic stability is established for three controllers: one constructed based on an energy-like Lyapunov function, a variation of this based on a backstepping procedure, and lastly, an adaptive version of the latter that estimates the net aerodynamic moment caused by the translational dynamics (flow angles). This alleviates the need for expensive flow angle measurement equipment, as well as the knowledge of an accurate aerodynamic model. Furthermore, we show that only a rough estimate of the input matrix is needed to achieve ultimate boundedness.

A limitation of the controller (4.23) is that the proportional gain is restricted to a scalar. The backstepping-based controllers allow for a matrix gain. This is an essential feature when implementing attitude controllers on real aircraft since it allows for different gains for each axis of rotation. This is achieved by prescribing a "geodesic" intermediate angular velocity reference (in the direction of the shortest path on $\mathbb{S}^2$) instead of a geodesic proportional torque.

The controller design can be used with state-of-the-art guidance systems for fixed-wing aircraft and is implemented in the open-source autopilot ArduPilot, where we demonstrate the suitability of the proposed attitude control algorithm through realistic SITL simulations of a fixed-wing UAV.

Chapter Outline

The rest of the chapter is organized as follows: we start with the problem definition in Section 6.2, presenting the dynamic model used, error variables and stating the control objective. We give an expanded treatment on turn coordination in Section 6.3 before continuing with the nominal control design in Section 6.4. In Section 6.5, we consider control design for uncertain models, including a backstepping-based control law with integral action. Simulation results are presented in Section 6.6, and finally, a chapter summary is given in Section 6.7.

6.2 Problem Definition

6.2.1 Control-Oriented Model

In this chapter, compared to the previous two, we use a more elaborate dynamic model to exploit model structure in the design. We base our control design on Eq. (2.46):

$$J\dot{\omega} = \boldsymbol{S}(J\omega)\omega + h(v_a^b) + V_a D\omega + V_a^2 B u + M_p(\delta_t).$$

In horizontal, level flight, the angular velocity ω is zero. To ensure equilibrium flight ("trim conditions"), define

$$u_{\text{trim}} = (V_a^*)^{-2} B^{-1} \left[-h(v_a^{b*}) - M_p^* \right],$$

where V_a^*, v_a^{b*}, $M_p(\delta_t^*)$ are the trim airspeed, trim relative velocity and trim throttle setting, respectively. If $u = u_{\text{trim}}$ and $\omega = 0$, then $\dot{\omega} = 0$ during trimmed flight. Now define

$$\Delta(v_a^b, t) = V_a^2 B u_{\text{trim}} + h(v_a^b) + M_p(\delta_t), \tag{6.1}$$

which represents the deviation from trimmed flight. We can now combine (6.1), the rotational dynamics (2.46) and the reduced-attitude kinematics (4.3) to obtain the following model that will be the basis for control design:

$$\dot{\Gamma} = \Gamma \times \omega \tag{6.2}$$

$$J\dot{\omega} = \boldsymbol{S}(J\omega)\omega + V_a D\omega + V_a^2 B \left[u - u_{\text{trim}}\right] + \Delta(v_a^b, t). \tag{6.3}$$

The state is represented by $(\Gamma, \omega) \in \mathbb{S}^2 \times \mathbb{R}^3$, the control input is $u \in \mathbb{R}^3$, and we consider δ_t and v_a^b (and thus V_a) as exogenous bounded inputs.

For control design purposes, we assume the following:

Assumption 2. The airspeed V_a is strictly positive and bounded with bounded derivative: $0 < V_{\min} \leq V_a \leq V_{\max}$, $|\dot{V}_a| \leq c_{\dot{V}_a}$.

Assumption 3. The vector $\Delta(v_a^b, t)$ and its derivative $\dot{\Delta}(v_a^b, t)$ are bounded.

Assumption 4. The control effectiveness matrix B is square and invertible.

Assumption 5. The damping matrix D satisfies $x^\top D x \leq 0, \forall x \in \mathbb{R}^3$.

Remark 6.1. Assumption 3 is an assumption on the translational dynamics, which is assumed to affect the rotational dynamics through the exogenous signal v_a^b (may also be considered as "internal dynamics"). In practice, since we are dealing with a physical system, $\Delta(v_a^b, t)$ and $\dot{\Delta}(v_a^b, t)$ will always be bounded. However, since the control surfaces are bounded, we would want the upper bounds to be relatively small. In particular, during nominal flight, the angle of attack α is usually small, and the lift coefficient is such that a perturbation in α tends to be restored [19]. However, if the stall angle of attack is reached, the slope of the lift coefficient changes such that the α-dynamics might go unstable, which in turn results in a high aerodynamic moment $\Delta(v_a^b, t)$.

Remark 6.2. A square matrix B corresponds to aircraft with fully actuated rotational dynamics, i.e. three independent actuators. Now B is invertible if it has full rank. 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. Therefore, the full rank assumption is reasonable for most common fully actuated control surface configurations.

Remark 6.3. Assumption 5 is a dissipation assumption and is equivalent to requiring that $\mathbf{sym}(D)$ has nonpositive eigenvalues. In nominal flight conditions, this will be true for most airframes [233] but can be relaxed by using a higher derivative gain (adding damping to the system). See Remark 6.7.

6.2.2 Error Functions

The goal is to design a state-feedback control law $u \in \mathbb{R}^3$ to make the reduced attitude $\Gamma \in \mathbb{S}^2$ asymptotically track a smooth, time-varying reference $\Gamma_d \in \mathbb{S}^2$, satisfying (4.15), and at the same time drive $\omega^{\parallel}$ to $\omega_d^{\parallel}$, where $\omega_d^{\parallel} \in \mathrm{N}_\Gamma\mathbb{S}^2$ denotes the desired value of $\omega^{\parallel}$, yet to be specified.

We require that all references are bounded, as specified in the following assumption:

Assumption 6. The angular velocity references $\omega_d^{\perp}$, $\omega_d^{\parallel}$ and their derivatives $\dot{\omega}_d^{\perp} := \frac{d}{dt}\omega_d^{\perp}$, $\dot{\omega}_d^{\parallel} := \frac{d}{dt}\omega_d^{\parallel}$ can be bounded a priori by

$$\begin{aligned} \|\omega_d^{\parallel}\| &\leq c_{\omega_d^{\parallel}} & \|\dot{\omega}_d^{\parallel}\| &\leq b_{\dot{\omega}_d^{\parallel}}\|\omega\| + c_{\dot{\omega}_d^{\parallel}} \\ \|\omega_d^{\perp}\| &\leq c_{\omega_d^{\perp}} & \|\dot{\omega}_d^{\perp}\| &\leq c_{\dot{\omega}_d^{\perp}}, \end{aligned} \tag{6.4}$$

where $c_{\omega_d^{\parallel}}, c_{\dot{\omega}_d^{\parallel}}, c_{\omega_d^{\perp}}, c_{\dot{\omega}_d^{\perp}}, b_{\dot{\omega}_d^{\parallel}} \in \mathbb{R}_{>0}$ are appropriate constant parameters.

We base our design on the same potential function and configuration error vector as before, i.e., $\Psi(\Gamma, \Gamma_d)$ and e_Γ given by (4.18) and (4.19), respectively. However, to simplify the design process and utilize (6.3) in a way that the resulting controller design is more robust to model uncertainty, we redefine the angular velocity error as follows:

$$e_\omega := \omega - \omega_d \in \mathbb{R}^3, \tag{6.5}$$

where $\omega_d := \mathbf{\Pi}_\Gamma^{\perp}(\omega_d^{\perp}) + \omega_d^{\parallel}$.

As before, the error terms, e_Γ and e_ω, are compatible in the sense that $\dot{\Psi} = e_\omega^\top e_\Gamma$, which cancels with the proportional feedback term in the subsequent Lyapunov analysis.

From (6.3), the derivative of e_ω satisfies

$$J\dot{e}_\omega = \boldsymbol{S}(J\omega)\omega + V_a D\omega + V_a^2 B\left[u - u_{\text{trim}}\right] + \Delta(v_a^b, t) - J\dot{\omega}_d. \tag{6.6}$$

6.2.3 Control Objective

The control objective is to simultaneously achieve $\omega^{\|} \to \omega_d^{\|}$ and $\Gamma \to \Gamma_d$, $\omega^{\perp} \to \omega_d^{\perp}$ as $t \to \infty$. These two objectives are orthogonal and not in conflict with each other. In particular, note that from our definition of e_ω, Eq. (6.5), and (4.4), e_ω can be decomposed into two orthogonal parts:

$$e_\omega = \underbrace{(\omega^{\perp} - \mathbf{\Pi}_{\Gamma}^{\perp}(\omega_d^{\perp}))}_{\in \mathrm{T}_\Gamma \mathbb{S}^2} + \underbrace{(\omega^{\|} - \omega_d^{\|})}_{\in \mathrm{N}_\Gamma \mathbb{S}^2}. \tag{6.7}$$

This means that as e_ω converges to zero, $\omega^{\perp} \to \mathbf{\Pi}_{\Gamma}^{\perp}(\omega_d^{\perp})$ and $\omega^{\|} \to \omega_d^{\|}$, in a decoupled manner. If in addition $e_\Gamma = 0$, then $\mathbf{\Pi}_{\Gamma}^{\perp}(\omega_d^{\perp}) = \omega_d^{\perp}$.

We now state the control objective as follows:

Almost global reduced-attitude tracking

Design a state-feedback control law u such that for almost all initial conditions, $e_\Gamma(t_0), e_\omega(t_0)$, $\Gamma(t) \to \Gamma_d(t)$ and $e_\omega(t) \to 0$ as $t \to \infty$.

As we consider continuous feedback on a compact configuration manifold, almost global asymptotic stability is the best possible achievable result [26]. In our setting, if the equilibrium point $(\Gamma, e_\omega) = (\Gamma_d, 0)$ is almost globally asymptotically stable, then almost all trajectories converge to it, except for those with initial velocity (depending on the initial configuration error) that are *exactly* such that $\omega^{\perp}(t) - \mathbf{\Pi}_{\Gamma}^{\perp}(\omega_d^{\perp}(t)) = 0$ when $\Gamma(t) = -\Gamma_d(t)$. This set of initial conditions has a dimension lower than the dimension of the state space and therefore has measure zero. As shown in Chapter 5, we can make the region of attraction global using hybrid control. Indeed, we can extend the work presented in this chapter in a similar manner, but this is outside of the scope of this thesis.

6.3 Turn Coordination

Before continuing with the controller design, we proceed by following up on the topic of turn coordination in Section 4.5 with a discussion on different design choices for $\omega_d^{\|}$.

Motivated by (4.7) and the coordinated-turn equation (2.51), we propose the following design for $\omega_d^{\|}$ that satisfies Assumption 6:

$$\omega_d^{\|} = \left(\frac{g}{V_a} \tan \phi_d - \dot{\phi}_d \sin \theta_d \right) \Gamma, \tag{6.8}$$

where θ_d, ϕ_d and $\dot{\phi}_d$ are consistent with $\Gamma_d, \omega_d^\perp$ (through (4.2) and (4.68)). Clearly, since (4.7) is only valid for $|\theta| \neq \pi/2$, and (6.8) contains $\tan\phi_d$, we must restrict the desired reduced-attitude as follows:

Assumption 7. The desired reduced-attitude Γ_d is such that $|\theta_d| \leq c_{\theta_d} < \pi/2$ and $|\phi_d| \leq c_{\phi_d} < \pi/2$, for some $c_{\theta_d}, c_{\phi_d} \in \mathbb{R}_{>0}$, and $\Gamma_d, \phi_d, \theta_d$ satisfy Eq. (4.2).

Remark 6.4. We stress that the mentioned singularities at $\phi = \pm\pi/2$ and $\theta = \pm\pi/2$ are only present for the reference angles. The allowed reference orientations cover most typical flight conditions except for certain aerobatic manoeuvres. The controller design, however, is globally defined, which enables recovery from large reduced-attitude errors, e.g. resulting from large wind gusts.

Alternative design choices for $\omega_d^\parallel$: It is possible to consider some variations of the preceding design of $\omega_d^\parallel$. We now present a few of these options but leave it as an exercise for the reader to fully explore these possibilities.

- An alternative to (6.8) is to define $\omega_d^\parallel$ in terms of Γ_d and then project to $\mathrm{N}_\Gamma\mathbb{S}^2$:

$$\begin{aligned}\omega_d^\parallel &= \mathbf{\Pi}_\Gamma^\parallel(\left(\frac{g}{V_a}\tan(\phi_d) - \dot{\phi}_d \sin\theta_d\right)\Gamma_d) \\ &= \left(\frac{g}{V_a}\tan(\phi_d) - \dot{\phi}_d \sin\theta_d\right)(\Gamma_d^\top \Gamma)\Gamma.\end{aligned}$$

  The extra term $\Gamma_d^\top\Gamma = \cos(\nu)$ puts less emphasis on turn coordination when errors in reduced attitude are large.

- Eq. (6.8) only satisfies the coordinated-turn equation (2.51) asymptotically, as $\Gamma \to \Gamma_d$ (and $\phi \to \phi_d$). One might consider to instead use the actual value of ϕ instead of ϕ_d (as in Section 4.5), but in this case, we cannot guarantee a priori that $\omega_d^\parallel$ and its derivative are bounded. This means that the subsequent stability analysis needs to be adjusted. A pragmatic solution could be to use a saturation function in combination with (6.8).

Variants of the preceding design can be developed by incorporating turn coordination concepts to drive sideslip angle (cf. Eq. (4.61)) or lateral acceleration to zero, or by including a yaw damper to increase dutch-roll damping [233]. This can be combined with the turn-rate feedforward (6.8) by using a high-pass (washout) filter [233]. As long as the resulting angular velocity command is parallel to Γ, it does not interfere with the reduced-attitude control objective.

To summarize, the expression for the total desired angular velocity ω_d in (6.5) is

$$\omega_d = \mathbf{\Pi}_\Gamma^\perp(\omega_d^\perp) + \omega_d^\parallel,$$

where $\omega_d^\parallel \in \mathrm{N}_\Gamma\mathbb{S}^2$ is given by (6.8), and $\omega_d^\perp \in \mathrm{T}_{\Gamma_d}\mathbb{S}^2$.

Eq. (6.8) satisfies Assumption 6 with

$$c_{\omega_d^\parallel} = \frac{g}{V_{\min}} \tan c_{\phi_d} + c_{\dot\phi_d} \sin c_{\theta_d},$$

where $c_{\dot\phi_d}$ is a bound for $\dot\phi_d$, i.e. $|\dot\phi_d| \leq c_{\dot\phi_d}$. Furthermore, $\dot\omega_d^\parallel$ can be bounded using appropriate constants $b_{\dot\omega_d^\parallel}$ and $c_{\dot\omega_d^\parallel}$ that depends on the bounds on the airspeed, reference angles and their derivatives. See Section 6.3.1 for details. Also, we write

$$\|\omega_d\| \leq \|\omega_d^\perp\| + \|\omega_d^\parallel\| \leq c_{\omega_d^\perp} + c_{\omega_d^\parallel} := c_{\omega_d}. \tag{6.9}$$

We proceed by calculating an explicit expression for $\dot\omega_d$, which is needed in the control laws.

6.3.1 Time Derivative of Desired Velocities

The total time derivative of ω_d is

$$\dot\omega_d = \frac{d}{dt}[\mathbf{\Pi}_\Gamma^\perp(\omega_d^\perp)] + \dot\omega_d^\parallel,$$

with

$$\frac{d}{dt}[\mathbf{\Pi}_\Gamma^\perp(\omega_d^\perp)] = \mathbf{\Pi}_\Gamma^\perp(\dot\omega_d^\perp) + \omega^\perp \times \mathbf{\Pi}_\Gamma^\parallel(\omega_d^\perp) - \mathbf{\Pi}_\Gamma^\parallel(\omega^\perp \times \omega_d^\perp),$$

and

$$\begin{aligned}\dot\omega_d^\parallel = &\left(\frac{g}{V_a}\tan\phi_d - \dot\phi_d \sin\theta_d\right) \boldsymbol{S}(\Gamma)\omega^\perp - \frac{g}{V_a^2}\tan(\phi_d)\dot V_a \Gamma \\ &+ \left(\frac{g}{V_a}\frac{1}{\cos^2(\phi_d)}\dot\phi_d - \ddot\phi_d \sin\theta_d - \dot\phi_d\dot\theta_d\cos\theta_d\right)\Gamma.\end{aligned}$$

Let $|\dot\theta_d| \leq c_{\dot\theta_d}$ and $|\ddot\phi_d| \leq c_{\ddot\phi_d}$, for some $c_{\dot\theta_d}, c_{\ddot\phi_d}, c_{\dot V_a} \in \mathbb{R}_{>0}$. Then, the norm of $\dot\omega_d^\parallel$ can be bounded as follows:

$$\|\dot\omega_d^\parallel\| \leq b_{\dot\omega_d^\parallel}\|\omega\| + c_{\dot\omega_d^\parallel},$$

where $c_{\dot\omega_d^\parallel} = c_{\omega_d^\parallel}$ and

$$b_{\dot\omega_d^\parallel} = \frac{g}{V_{\min}\cos^2(c_{\phi_d})} + c_{\ddot\phi_d}\sin c_{\theta_d} + c_{\dot\phi_d}c_{\dot\theta_d} + \frac{g}{V_{\min}^2}c_{\dot V_a}\tan c_{\phi_d}.$$

6.4 Control Laws - Nominal Case

In this section, we present nominal state-feedback control laws assuming perfect knowledge of the rotational dynamics. Two different controllers are presented: one based on an energy-like Lyapunov function and another based on the backstepping procedure. Although perfect model knowledge is assumed, we do not perform feedback linearization/dynamic inversion but rather exploit structural properties of the model, such as skew-symmetry and positive definiteness of matrices. This way, we avoid cancelling "good" terms, while other terms are dominated in the stability proof.

6.4.1 Control Design Based on an Energy-Like Lyapunov Function

Proposition 6.1: Consider the tracking error dynamics (6.6), and for $k_p > 0, K_d = K_d^\top > 0$, define the control input as

$$u = u_{\text{trim}} + \frac{1}{V_a^2} B^{-1} \left[u_{\text{pd}} + u_{\text{ff}} - \Delta(v_a^b, t) \right], \tag{6.10}$$

where

$$u_{\text{pd}} = -k_p e_\Gamma - K_d e_\omega \tag{6.11}$$
$$u_{\text{ff}} = J\dot{\omega}_d - \boldsymbol{S}(J\omega_d)\omega_d - V_a D \omega_d, \tag{6.12}$$

and the matrix K_d is chosen such that

$$\lambda_{\min}^{K_d} - \lambda_{\max}^{J} c_{\omega_d} \geq \gamma, \tag{6.13}$$

for some $\gamma > 0$. Then the following holds:

i) There are two closed-loop equilibria, given by $(\Gamma, e_\omega) = (\pm\Gamma_d, 0)$.

ii) The equilibrium $(\Gamma, e_\omega) = (-\Gamma_d, 0)$ is unstable.

iii) The desired equilibrium $(\Gamma, e_\omega) = (\Gamma_d, 0)$ is almost globally asymptotically stable.

iv) The desired equilibrium $(\Gamma, e_\omega) = (\Gamma_d, 0)$ is locally exponentially stable. In addition, if the initial conditions $\Gamma(0), \Gamma_d(0), e_\omega(0)$ satisfy

$$\Psi(\Gamma(0), \Gamma_d(0)) < 2 \tag{6.14}$$
$$k_p \Psi(\Gamma(0), \Gamma_d(0)) + \frac{1}{2} e_\omega^\top(0) J e_\omega(0) < 2k_p, \tag{6.15}$$

then the energy-like function $V(t) := k_p \Psi(\Gamma, \Gamma_d) + (1/2) e_\omega^\top J e_\omega$ converges exponentially to zero.

v) $\omega^{\perp} \to \omega_d^{\perp}$ and $\omega^{\parallel} \to \omega_d^{\parallel}$ as $t \to \infty$.

Proof. The control law (6.10) in combination with (6.6) results in the closed-loop error dynamics

$$J\dot{e}_\omega = -k_p e_\Gamma - [K_d - V_a D + \boldsymbol{S}(\omega_d)J]\, e_\omega + \boldsymbol{S}(J(e_\omega + \omega_d))e_\omega. \tag{6.16}$$

Note that the system is time-varying due to the presence of ω_d and V_a.

Part 1: Equilibrium solutions When $\Gamma = \pm\Gamma_d$, then $e_\Gamma = 0$. By substituting $(e_\Gamma, e_\omega) = (0, 0)$ into (6.16) and (4.26), we see that $(\Gamma(t), e_\omega(t)) = (\pm\Gamma_d(t), 0)$ indeed represent equilibrium solutions of the closed-loop dynamics. To see that all solutions converge to either of these equilibria, consider the Lyapunov function candidate

$$V(\Gamma, \Gamma_d, e_\omega) = k_p \Psi(\Gamma, \Gamma_d) + \frac{1}{2} e_\omega^\top J e_\omega \geq 0, \tag{6.17}$$

whose time derivative along the solutions of (6.16), (4.3) and (4.15) satisfies

$$\dot{V} = k_p e_\omega^\top e_\Gamma + e_\omega^\top J \dot{e}_\omega = -e_\omega^\top [K_d - V_a D + \boldsymbol{S}(\omega_d)J] e_\omega.$$

By Assumption 5 and the gain condition (6.13) (inspired by [20]), we get

$$\dot{V} \leq -\left(\lambda_{\min}^{K_d} - \lambda_{\max}^{J} c_{\omega_d}\right) \|e_\omega\|^2 \leq -\gamma \|e_\omega\|^2 \leq 0. \tag{6.18}$$

From $V \geq 0$ and $\dot{V} \leq 0$ we get that e_ω is bounded. In addition, the limit $V_\infty = \lim\limits_{t\to\infty} V(t)$ exists and is finite ([112], Lemma 3.2.3). This means that $\int_{t_0}^{\infty} \dot{V} d\tau = V_\infty - V(t_0)$. Therefore, the function $W = \gamma\|e_\omega\|^2$ satisfies $\int_{t_0}^{\infty} W d\tau \leq -\int_{t_0}^{\infty} \dot{V} d\tau = V(t_0) - V_\infty < \infty$, so the limit $\lim\limits_{t\to\infty} \int_{t_0}^{t} W d\tau$ exists and is finite. By definition, e_Γ is bounded. Since e_ω is bounded, $\dot{e}_\omega$ is bounded. This follows from (6.16) and the boundedness of ω_d and V_a. Now, since $\dot{W} = 2\gamma e_\omega^\top \dot{e}_\omega$ is bounded, it follows that $W(t)$ is a uniformly continuous function. From Barbalat's Lemma ([112], Lemma 3.2.6), $W(t)$ (and thus $e_\omega(t)$) converges to zero as $t \to \infty$.

To show that e_Γ also converges to zero, we need the following technical lemma:

Lemma 6.1. *Let $x(t)$ denote a solution to the differential equation $\dot{x} = a(t) + b(t)$ with $a(t)$ a uniformly continuous function. Assume that $\lim_{t\to\infty} x(t) = c$ and $\lim_{t\to\infty} b(t) = 0$, with c a constant value. Then, $\lim_{t\to\infty} \dot{x}(t) = 0$ [108, 175].*

From (6.16) we can write $J\dot{e}_\omega = a(t) + b(t)$, where

$$\begin{aligned} a(t) &:= -k_p e_\Gamma \\ b(t) &:= -\left[K_d - V_a D + \boldsymbol{S}(\omega_d)J\right] e_\omega + \boldsymbol{S}(J(e_\omega + \omega_d))e_\omega \end{aligned}$$

Since e_ω converges to zero, we know that $b(t)$ converges to zero. From (4.26), the derivative of $a(t)$ is given by

$$\dot{a}(t) = -k_p \dot{e}_\Gamma = k_p \left[\boldsymbol{S}(\omega_d^\perp)e_\Gamma + \boldsymbol{S}(\Gamma_d)\boldsymbol{S}(\Gamma)e_\omega\right],$$

which is bounded because e_Γ, e_ω and $\omega_d^\perp$ are bounded. Therefore, $a(t)$ is uniformly continuous, and convergence of e_Γ to zero follows from Lemma 6.1.

To summarize, all solutions converge to one of the two equilibria given by $(\Gamma, e_\omega) = (\pm\Gamma_d, 0)$.

Remark 6.5. The Lyapunov function (6.17) is quadratic, and as we will see, leads to exponential stability, which in general leads to good performance and robustness to perturbations [125]. In [105], it is shown that non-quadratic Lyapunov functions could lead to better performance. Therefore, future work based on the general approach presented in this paper might explore whether different Lyapunov functions than (6.17), possibly non-quadratic ones, could lead to better performance.

Part 2: Instability of the undesired equilibrium point At the equilibrium point $(\Gamma, e_\omega) = (-\Gamma_d, 0)$, the value of the Lyapunov function is $V = 2k_p$. To show that this equilibrium is unstable, it suffices to show that for any neighbourhood $\mathcal{U}$ around this point, one can find Γ^*, e_ω^* such that $V < 2k_p$. Since V is non-increasing and all solutions converge to either of the two equilibria, any solution starting at (Γ^*, e_ω^*) must converge to $(\Gamma, e_\omega) = (\Gamma, 0)$. Consider Γ^* arbitrarily close to $-\Gamma_d$, say at an angle ϵ away from $-\Gamma_d$. Then, $\psi(\Gamma, \Gamma_d) = 1 - \cos(\pi - \epsilon) \approx 2 - \epsilon$ and $V \approx k_p(2-\epsilon) + e_\omega^{*\top} J e_\omega^*/2$. This means that if we choose e_ω^* small enough, then $V < 2k_p$ and we conclude that the equilibrium point is unstable.

Remark 6.6. This line of reasoning parallels that of Chetaev's Theorem, for which a version for time-invariant systems is given in Theorem 4.3 in [125].

Part 3: Stability of the desired equilibrium We proceed by studying the asymptotic stability of the equilibrium point where $\Gamma = \Gamma_d$. To this end, consider again the Lyapunov function candidate (6.17). From [143], we know that in some neighborhood of $(\Gamma, e_\omega) = (\Gamma_d, 0)$, V can be lower and upper bounded by

$$\frac{k_p}{2}\|e_\Gamma\|^2 + \frac{\lambda_{\min}^J}{2}\|e_\omega\|^2 \le V \le \frac{k_p}{2-\bar{\Psi}}\|e_\Gamma\|^2 + \frac{\lambda_{\max}^J}{2}\|e_\omega\|^2.$$

$\dot{V} \leq 0$ together with the positive definite bounds on V makes the equilibrium point $(\Gamma, e_\omega) = (\Gamma_d, 0)$ uniformly stable (Theorem 4.8 in [125]).

Convergence combined with Lyapunov stability leads to asymptotic stability of the desired equilibrium point $(\Gamma, e_\omega) = (\Gamma_d, 0)$. The stable manifold of the unstable equilibrium is less than the dimension of the state space of the system and therefore has measure zero. The region of attraction to the stable equilibrium point excludes this manifold, so we conclude that the desired equilibrium is almost globally asymptotically stable.

Part 4: Exponential stability To show exponential stability, let $\epsilon > 0$ (arbitarily small) and consider the Lyapunov function (6.17) augmented with a cross-term:

$$V_\epsilon = V + \epsilon e_\Gamma^\top J e_\omega.$$

which is positive definite for small ϵ. The time derivative of V_ϵ along the closed-loop trajectories satisfies

$$\dot{V}_\epsilon = \dot{V} + \epsilon e_\Gamma^\top J \dot{e}_\omega + \epsilon e_\omega^\top J \dot{e}_\Gamma.$$

We calculate the last two terms separately. From (6.16) and (4.26) we get

$$\begin{aligned}\epsilon e_\Gamma^\top J \dot{e}_\omega &= -\epsilon k_p e_\Gamma^\top e_\Gamma + \epsilon e_\Gamma^\top \boldsymbol{S}(J e_\omega) e_\omega - \epsilon e_\Gamma^\top \left[K_d - V_a D + \boldsymbol{S}(\omega_d) J - \boldsymbol{S}(J\omega_d)\right] e_\omega \\ &\leq -\epsilon k_p \|e_\Gamma\|^2 + \epsilon \lambda_{\max}^J \|e_\omega\|^2 + \epsilon(\lambda_{\max}^{K_d} + V_{\max}\sigma_{\max}^D + 2\lambda_{\max}^J c_{\omega_d})\|e_\Gamma\|\|e_\omega\|,\end{aligned}$$

since $\|e_\Gamma\| \leq 1$.

$$\epsilon e_\omega^\top J \dot{e}_\Gamma = -\epsilon e_\omega^\top J \left[\boldsymbol{S}(\omega_d^\perp) e_\Gamma + \boldsymbol{S}(\Gamma_d)\boldsymbol{S}(\Gamma) e_\omega\right] \leq \epsilon \lambda_{\max}^J c_{\omega_d^\perp} \|e_\Gamma\|\|e_\omega\| + \epsilon \lambda_{\max}^J \|e_\omega\|^2.$$

Combining this with (6.18) gives

$$\begin{aligned}\dot{V}_\epsilon \leq -\epsilon k_p \|e_\Gamma\|^2 - (\gamma - 2\epsilon\lambda_{\max}^J)\|e_\omega\|^2 + \epsilon(\lambda_{\max}^{K_d} + V_{\max}\sigma_{\max}^D \\ + \lambda_{\max}^J(2c_{\omega_d} + c_{\omega_d^\perp})\|e_\Gamma\|\|e_\omega\|\end{aligned}$$

Let $x := [\|e_\Gamma\| \;\; \|e_\omega\|]^\top$. Then, we can write $\dot{V}_\epsilon \leq -x^\top M x$, where the matrix M is given by

$$M = \begin{bmatrix} \epsilon k_p & -\frac{\epsilon\xi}{2} \\ -\frac{\epsilon\xi}{2} & \gamma - 2\epsilon\lambda_{\max}^J \end{bmatrix},$$

where $\xi = \lambda_{\max}^{K_d} + V_{\max}\sigma_{\max}^D + \lambda_{\max}^J(2c_{\omega_d} + c_{\omega_d^\perp})$. The matrix M is positive definite if the following inequality is satisfied:

$$\epsilon < \frac{4k_p\gamma}{8k_p\lambda_{\max}^J + \xi^2}.$$

Since ϵ can be chosen arbitrarily small, this inequality can always be satisfied. With $\dot{V}_\epsilon$ negative definite, and with quadratic bounds on V_ϵ, we conclude that the desired equilibrium is exponentially stable. For estimation of the region of exponential convergence, see [61].

Part 5: Convergence of angular velocities Since e_Γ converges to zero, $\mathbf{\Pi}_\Gamma^\perp(\omega_d^\perp) \to \omega_d^\perp$. Now, Eq. (6.7) proves our point due to the orthogonality of the two parenthesized terms. □

Remark 6.7. If Assumption 5 is not satisfied, it is not difficult to show that the result still holds if K_d is chosen such that $\lambda_{\min}^{K_d} > \lambda_{\max}^{J} c_{\omega_d} + V_{\max}\sigma_{\max}^{\mathbf{sym}(D)}$.

Remark 6.8. The region of exponential convergence to the desired equilibrium point can be made (almost) arbitrarily large by increasing k_p ("semi-global" property). However, the region of convergence can never include the unstable equilibrium point and its corresponding unstable manifold [145].

6.4.2 Backstepping Design

A disadvantage of the controller design in the previous section is that the scalar proportional gain k_p is restrictive. In this section, we present a backstepping controller that allows for a matrix proportional gain, which gives the flexibility for the control to be more aggressive along certain body-fixed axes, which is important due to geometric and aerodynamic asymmetries of aircraft. In the previous section, the proportional action defines a *torque* that is aligned with the axis of shortest rotation. The backstepping controller, on the other hand, defines a *desired angular velocity* that generates a geodesic curve on the sphere.

To this end, define the virtual control signal

$$\varphi(\Gamma, \Gamma_d, \omega_d^\perp) := -\kappa e_\Gamma + \mathbf{\Pi}_\Gamma^\perp(\omega_d^\perp) \in \mathrm{T}_\Gamma\mathbb{S}^2, \tag{6.19}$$

where $\kappa \in \mathbb{R}_{>0}$ is a user specified parameter. We will show that $\omega^\perp = \varphi(\Gamma, \Gamma_d, \omega_d^\perp)$ solves the kinematic reduced-attitude tracking problem (see the proof of Proposition 2). Now, introduce the tracking-error signal

$$z := \underbrace{\left(\omega^\perp - \varphi(\Gamma, \Gamma_d, \omega_d^\perp)\right)}_{\in \mathrm{T}_\Gamma\mathbb{S}^2} + \underbrace{(\omega^\parallel - \omega_d^\parallel)}_{\in \mathrm{N}_\Gamma\mathbb{S}^2} = \omega - \bar{\omega}_d, \tag{6.20}$$

where $\bar{\omega}_d = \varphi(\Gamma, \Gamma_d, \omega_d^\perp) + \omega_d^\parallel$. Note that $\bar{\omega}_d = \omega_d - \kappa e_\Gamma$ and z can be written as $z = e_\omega + \kappa e_\Gamma$. Due to orthogonality properties, z defined as in (6.20) has the nice property that as z converges to zero, $\omega^\perp \to \varphi(\Gamma, \Gamma_d, \omega_d^\perp)$, which stabilizes the desired reduced-attitude, and at the same time, $\omega^\parallel$ converges to $\omega_d^\parallel$.

Proposition 6.2: Consider the tracking error dynamics (6.6), and for $k_1 > 0, K_2 = K_2^\top > 0$, define the control input as

$$u = u_{\text{trim}} + \frac{1}{V_a^2} B^{-1} \left[u_{\text{pd}} + u_{\text{ff}} - \Delta(v_a^b, t) \right] \tag{6.21}$$

where

$$u_{\text{pd}} = -k_1 e_\Gamma - K_2 z \tag{6.22}$$
$$u_{\text{ff}} = J\dot{\bar{\omega}}_d - \boldsymbol{S}(J\bar{\omega}_d)\bar{\omega}_d - V_a D \bar{\omega}_d, \tag{6.23}$$

and the matrix K_2 is chosen such that

$$\lambda_{\min}^{K_2} - \lambda_{\max}^{J}(c_{\omega_d} + \kappa) \geq \gamma, \tag{6.24}$$

for some $\gamma > 0$. Then the following holds:

i) There are two closed-loop equilibria, given by $(\Gamma, z) = (\pm\Gamma_d, 0)$.

ii) The equilibrium $(\Gamma, z) = (-\Gamma_d, 0)$ is unstable.

iii) The desired equilibrium $(\Gamma, z) = (\Gamma_d, 0)$ is almost globally asymptotically stable.

iv) The desired equilibrium $(\Gamma, z) = (\Gamma_d, 0)$ is locally exponentially stable. In addition, if the initial conditions $\Gamma(0), \Gamma_d(0), z(0)$ satisfy

$$\Psi(\Gamma(0), \Gamma_d(0)) < 2 \tag{6.25}$$
$$k_1 \Psi(\Gamma(0), \Gamma_d(0)) + \frac{1}{2} z^\top(0) J z(0) < 2k_1, \tag{6.26}$$

then the energy function $V_2(t) := k_1 \Psi(\Gamma, \Gamma_d) + (1/2) z^\top J z$ converges exponentially to zero.

v) $\omega^\perp \to \omega_d^\perp$ and $\omega^\parallel \to \omega_d^\parallel$ as $t \to \infty$.

Proof. We begin by establishing that $\omega^\perp = \varphi(\Gamma, \Gamma_d, \omega_d^\perp)$ stabilizes the desired reduced-attitude with $V_1 = k_1 \Psi(\Gamma, \Gamma_d)$ as a Lyapunov function. The derivative of V_1 under the stated virtual control becomes $\dot{V}_1 = -\kappa k_1 \|e_\Gamma\|^2$. When $\omega^\perp \neq \varphi(\Gamma, \Gamma_d, \omega_d^\perp)$ we use (6.20) and get:

$$\dot{V}_1 = -\kappa k_1 \|e_\Gamma\|^2 + k_1 z^\top e_\Gamma,$$

since $e_\Gamma^\top \omega^\parallel = e_\Gamma^\top \omega_{ct} = 0$.

From (6.6) and (6.20), the derivative of z satisfies

$$J\dot{z} = \left[\boldsymbol{S}(J(z + \bar{\omega}_d)) + V_a D \right](z + \bar{\omega}_d) + V_a^2 B \left[u - u_{\text{trim}} \right] + \Delta(v_a^b, t) - J\dot{\bar{\omega}}_d. \tag{6.27}$$

In closed loop with the control law (6.21), we get

$$J\dot{z} = -k_1 e_\Gamma - [K_2 - V_a D + \boldsymbol{S}(\bar{\omega}_d)J]\, z + \boldsymbol{S}(J(z + \bar{\omega}_d))z. \tag{6.28}$$

Let a Lyapunov function candidate for the complete system be given by

$$V_2 = V_1 + \frac{1}{2} z^\top J z,$$

whose total time derivative satisfies

$$\begin{aligned}\dot{V}_2 &= -\kappa k_1 e_\Gamma^\top e_\Gamma + z^\top \left[k_1 e_\Gamma + J\dot{z}\right] \leq -\kappa k_1 e_\Gamma^\top e_\Gamma - z^\top \left[K_2 + \boldsymbol{S}(\bar{\omega}_d)J\right] z \\ &\leq -\kappa k_1 \|e_\Gamma\|^2 - \left(\lambda_{\min}^{K_2} - (\lambda_{\max}^{J} c_{\omega_d} + \kappa)\right) \|z\|^2 \leq -\kappa k_1 \|e_\Gamma\|^2 - \gamma \|z\|^2,\end{aligned}$$

where we have used (6.24), (6.28) and Assumption 5.

The rest of the proof follows the same arguments as in the proof of Proposition 1. □

As for the previous design, a statement similar to Remark 6.7 holds true also here.

The control laws (6.10) and (6.21) might seem similar at first glance, but by inserting $z = e_\omega + \kappa e_\Gamma$ we can rewrite Eq. (6.21) in terms of e_ω and ω_d:

$$\begin{aligned} u_{\text{pd}} &= -\left[K(t) + \kappa^2 \boldsymbol{S}(J e_\Gamma)\right] e_\Gamma - \left[K_2 - \kappa J \boldsymbol{S}(\Gamma_d)\boldsymbol{S}(\Gamma)\right] e_\omega \\ u_{\text{ff}} &= J\dot{\omega}_d - \boldsymbol{S}(J\omega_d)\omega_d - V_a D \omega_d, \end{aligned}$$

where $K(t) = k_1 + \kappa[K_2 - V_a D - J\boldsymbol{S}(\omega_d^\perp) + \boldsymbol{S}(\omega_d)J - \boldsymbol{S}(J\omega_d)]$, and the time-dependence is implicit through V_a, $\omega_d^\perp$ and ω_d. Here, the feed-forward part is the same as (6.12), but the change of variables imposed by the backstepping procedure has introduced a time-varying matrix proportional gain $K(t)$, a time-varying derivative gain, as well as a nonlinear feedback-term $-\kappa^2 \boldsymbol{S}(J e_\Gamma) e_\Gamma$.

6.5 Robustness Considerations

There are a few drawbacks to the controller designs presented in Section 6.4. In particular, the control laws (6.10) and (6.21) require the knowledge of the inertia matrix J, the damping matrix D, the input-matrix B, and the aerodynamic moment $\Delta(v_a^b, t)$. In this section, we focus on the control law (6.21) and state some properties regarding robustness to uncertainty in our model estimates. In addition, an adaptive version of (6.21) is presented that provides integral action by estimating $\Delta(v_a^b, t)$ under a slowly time-varying assumption.

Assumption 8. The aerodynamic moment disturbance $\Delta(v_a^b, t)$ is slowly varying, satisfying $\dot{\Delta}(v_a^b, t) \approx 0$.

6.5.1 Integral Action

The assumption that $\Delta(v_a^b, t)$ is known is particularly restrictive. The aerodynamics of aircraft is highly uncertain. Moreover, the explicit computation of $\Delta(v_a^b, t)$ requires knowledge of the surrounding flow field. Although the airspeed can be measured using a small pitot-static tube, equipment that measures the flow angles α and β is usually not readily available for some aircraft such as small UAVs. There exists some available technologies [2], but such equipment can be expensive, too large or too heavy, or just impractical to install on small UAVs that often perform belly landings [239]. Several approaches for flow angle estimation have been proposed in the literature [115, 206, 239, 251], but it remains a challenging problem. Therefore, we focus our attention to instead estimating the aerodynamic moments directly. The control input during trim, u_{trim} can often be quite easily identified during manual flight, so we turn our attention to estimating $\Delta(v_a^b, t)$ instead of $h_r(v_a^b)$. This also removes the need for an explicit estimate of M_p.

Proposition 6.3: Consider the tracking error dynamics (6.6), and let $\hat{\Delta}$ be an estimate of $\Delta(v_a^b, t)$. Define the estimation error $\tilde{\Delta} := \hat{\Delta} - \Delta(v_a^b, t)$, let K_2, K_3 be symmetric, positive definite matrices and define the control input as

$$u = u_{\text{trim}} + \frac{1}{V_a^2} B^{-1} \left[u_{\text{pd}} + u_{\text{ff}} - \hat{\Delta} \right] \tag{6.29}$$

where

$$u_{\text{pd}} = -k_1 e_\Gamma - K_2 z \tag{6.30}$$
$$u_{\text{ff}} = J\dot{\bar{\omega}}_d - \boldsymbol{S}(J\bar{\omega}_d)\bar{\omega}_d - V_a D\bar{\omega}_d, \tag{6.31}$$

where the update law for $\hat{\Delta}$ is given by

$$\dot{\hat{\Delta}} = K_3 z, \tag{6.32}$$

and the matrix K_2 is chosen such that

$$\lambda_{\min}^{K_2} - \lambda_{\max}^{J}(c_{\omega_d} + \kappa) \geq \gamma, \tag{6.33}$$

for some $\gamma > 0$. Then the following holds:

i) There are two closed-loop equilibria, given by $(\Gamma, z, \tilde{\Delta}) = (\pm\Gamma_d, 0, 0)$.

ii) The equilibrium $(\Gamma, z, \tilde{\Delta}) = (-\Gamma_d, 0, 0)$ is unstable.

iii) The desired equilibrium $(\Gamma, z, \tilde{\Delta}) = (\Gamma_d, 0, 0)$ is almost globally asymptotically stable and locally exponentially stable.

Proof. By combining (6.27) with the control law (6.29), we now get the closed-loop error dynamics

$$J\dot{z} = -k_1 e_\Gamma - \left[K_2 - V_a D + \boldsymbol{S}(\bar{\omega}_d)J\right] z + \boldsymbol{S}(J(z+\bar{\omega}_d))z - \tilde{\Delta}. \tag{6.34}$$

This is similar to (6.28), but with the extra estimation-error term $-\tilde{\Delta}$. Again, calculating $\dot{V}_2$ gives

$$\dot{V}_2 \leq -\kappa k_1 \|e_\Gamma\|^2 - \gamma \|z\|^2 - z^\top \tilde{\Delta}.$$

For some symmetric, positive definite gain matrix $K_3 = K_3^\top > 0$, let an augmented Lyapunov function candidate be given by

$$V_3 = V_2 + \frac{1}{2}\tilde{\Delta}^\top K_3^{-1} \tilde{\Delta}.$$

Differentiating V_3 gives

$$\dot{V}_3 = \dot{V}_2 + \tilde{\Delta}^\top K_3^{-1} \dot{\tilde{\Delta}} \leq -\kappa k_1 \|e_\Gamma\|^2 - \gamma \|z\|^2 + \tilde{\Delta}^\top K_3^{-1}\left[\dot{\hat{\Delta}} - K_3 z\right].$$

If the update law for $\hat{\Delta}$ is chosen as

$$\dot{\hat{\Delta}} = K_3 z,$$

we are left with

$$\dot{V}_3 \leq -\kappa k_1 \|e_\Gamma\|^2 - \gamma \|z\|^2.$$

By Barbalat's lemma, $\dot{V}$ goes to zero asymptotically, and so does e_Γ and z, and therefore also e_ω. From (6.34), we have $J\dot{z} = a(t) + b(t)$, where

$$\begin{aligned} a(t) &:= -\tilde{\Delta} \\ b(t) &:= -k_1 e_\Gamma - \left[K_2 - V_a D + \boldsymbol{S}(\bar{\omega}_d)J\right] z + \boldsymbol{S}(J(z+\bar{\omega}_d))z \end{aligned}$$

Since e_Γ and z converges to zero, we know that $b(t)$ converges to zero. The derivative of $a(t)$ is given by

$$\dot{a}(t) = -\dot{\hat{\Delta}} = -K_3 z$$

which is bounded by the boundedness of z. Therefore, $a(t)$ is uniformly continuous. From Lemma 6.1, we get that $\tilde{\Delta}$ converges to zero as well. The rest of the proof follows closely that of Proposition 1.

To show exponential stability, for $\epsilon > 0$, consider

$$V_4 = V_3 + \epsilon \tilde{\Delta}^\top J z,$$

whose time derivative satisfies

$$\dot{V}_4 = \dot{V}_3 + \epsilon\tilde{\Delta}^\top J\dot{z} + \epsilon z^\top J\dot{\tilde{\Delta}} \leq -\kappa k_1\|e_\Gamma\|^2 - \gamma\|z\|^2 + \epsilon z^\top JK_3 z + \epsilon\tilde{\Delta}^\top J\dot{z}$$

From (6.34), we calculate the last term as

$$\begin{aligned}\epsilon\tilde{\Delta}^\top J\dot{z} &= -\epsilon\tilde{\Delta}^\top\tilde{\Delta} + \epsilon\tilde{\Delta}^\top\boldsymbol{S}(J(z+\bar{\omega}_d))z - \epsilon k_1\tilde{\Delta}^\top e_\Gamma - \epsilon\tilde{\Delta}^\top\left[K_2 - V_a D + \boldsymbol{S}(\bar{\omega}_d)J\right]z\\ &\leq -\epsilon\|\tilde{\Delta}\|^2 + \epsilon\lambda_{\max}^J(c_{\omega_d}+\kappa)\|z\|^2\|\tilde{\Delta}\| + \epsilon k_1\|e_\Gamma\|\|\tilde{\Delta}\|\\ &\quad + \epsilon\left(\lambda_{\max}^{K_2} + \sigma_{\max}^{D(V_a)} + \lambda_{\max}^J(c_{\omega_d}+\kappa)\right)\|z\|\|\tilde{\Delta}\|\end{aligned}$$

Now, for z such that $\|z\| \leq c_z$, we can write $\dot{V}_4 \leq -\bar{x}^\top N\bar{x}$, where the matrix N is given by

$$N = \begin{bmatrix}\kappa k_1 & 0 & -\frac{\epsilon k_1}{2}\\ 0 & \gamma - \epsilon\lambda_{\max}^J\lambda_{\max}^{K_3} & -\frac{\epsilon\xi}{2}\\ -\frac{\epsilon k_1}{2} & -\frac{\epsilon\xi}{2} & \epsilon\end{bmatrix},$$

with $\xi = \lambda_{\max}^{K_2} + V_{\max}\sigma_{\max}^D + (1+c_z)\lambda_{\max}^J(c_{\omega_d}+\kappa)$ and $\bar{x} = [\|e_\Gamma\|\ \|z\|\ \|\tilde{\Delta}\|]^\top$.

The parameter ϵ can be chosen small enough such that N is positive definite. Requiring a positive determinant leads to a second-order inequality in ϵ of the form $a\epsilon^2 - b\epsilon + c > 0$, where a, b, c are positive coefficients. Clearly, since $c > 0$, there exists some ϵ (arbitrarily small) that satisfies this inequality. □

While the controller in the previous section is of PD type, this is a PID controller with feedforward terms. Integral action removes any steady-state error between the desired and actual angular velocity.

6.5.2 Uncertain Model

Sometimes it is desirable not to include integral action in the inner loops of cascaded control systems. Therefore, we focus on a version of the controller that uses a fixed—possibly time- and state-varying, but bounded—disturbance estimate. This estimate does not necessarily equal the true value of Δ. Besides, for some hierarchical flight control loops, the reference velocities are not made available for the inner loop. We thus remove the assumption that $\omega_d^\perp$ is known in the control design, and thus redefine the backstepping variable as $\hat{z} = \omega - \hat{\omega}_d$, and $\hat{\omega}_d = \omega_d^\parallel - \kappa e_\Gamma$. The backstepping procedure in the previous section has provided us with a strict Lyapunov function that can be used to show uniform ultimate boundedness of the solutions of the closed-loop system.

To account for model uncertainties, consider again the control law (6.21)

$$u = u_{\text{trim}} + \frac{1}{V_a^2}\hat{B}^{-1}\left[u_{\text{pd}} + u_{\text{ff}} - \hat{\Delta}(v_a^b, t)\right] \tag{6.35}$$

where

$$u_{\text{pd}} = -k_1 e_\Gamma - K_2\hat{z} \tag{6.36}$$

$$u_{\text{ff}} = \hat{J}\dot{\hat{\omega}}_d' - \boldsymbol{S}(\hat{J}\hat{\omega}_d)\hat{\omega}_d - V_a\hat{D}\hat{\omega}_d, \tag{6.37}$$

and $\hat{\Delta}(v_a^b, t), \hat{B}, \hat{J}, \hat{D}$ are our best estimates of $\Delta(v_a^b, t), B, J, D$, respectively. The term $\dot{\hat{\omega}}_d'$ is defined as the parts of $\dot{\hat{\omega}}_d$ that do not require reference angular velocities or accelerations:

$$\dot{\hat{\omega}}_d' = \left(\frac{g}{V_a}\tan\phi_d\right)\dot{\Gamma} - \frac{g}{V_a^2}\tan(\phi_d)\dot{V}_a\Gamma - \kappa(\dot{\Gamma}\times\Gamma_d). \tag{6.38}$$

The next proposition states that, for sufficiently small model uncertainties, and sufficiently small $\omega_d^\perp$, the solutions are ultimately bounded. This is essentially a local input-to-state stability (ISS) property [125].

To parametrize the model uncertainty, let $\delta B := B\hat{B}^{-1}$, $E := I - \delta B$, $\tilde{J} := \hat{J} - J$, and $\tilde{D} := \hat{D} - D$. For compactness, we define $c_J = \|\tilde{J}\| + \|E\|\|\hat{J}\|$ and $c_D = \|\tilde{D}\| + \|E\|\|\hat{D}\|$.

Proposition 6.4: Consider the tracking error dynamics (6.6) and the perturbed controller (6.35). Assume that δB satisifes $x^\top \delta B x > 0, \forall x \neq 0$. Then, there exists some gain matrix $K_2 = K_2^\top > 0$ such that the matrix $\mathbf{sym}(\delta B K_2)$ is positive definite. If the matrix K_2 is chosen such that

$$\lambda_{\min}^{\mathbf{sym}(\delta B K_2)} > \frac{a^2}{4\kappa k_1} + \lambda_{\max}^J(c_{\omega_d^\parallel} + \kappa) + c_J(\kappa + b_{\dot{\omega}_d^\parallel}), \tag{6.39}$$

where $a = k_1\|E\| + k_1 c_{\omega_d^\perp} + \kappa c_D + c_J\kappa\left(2\kappa + 2c_{\omega_d^\parallel} + b_{\dot{\omega}_d^\parallel}\right)$, and if c as defined by Equation (6.44) is sufficiently small, then the solutions of the closed-loop system are uniformly ultimately bounded, with an ultimate bound that depends on the controller parameters, the model estimation errors, and the reference velocity bounds.

Proof. The derivative of $\hat{z}$ satisfies

$$J\dot{\hat{z}} = \left[\boldsymbol{S}(J(z + \hat{\omega}_d)) + V_a D\right](z + \hat{\omega}_d) + V_a^2 B\left[u - u_{\text{trim}}\right] + \Delta(v_a^b, t) - J\dot{\hat{\omega}}_d. \tag{6.40}$$

In closed loop with the control law (6.35), we get

$$J\dot{\hat{z}} = \delta B \left[-k_1 e_\Gamma - K_2\hat{z} - \tilde{\Delta}\right] + \left[V_a D + \boldsymbol{S}(\hat{\omega}_d)J\right]\hat{z} + \boldsymbol{S}(J(\hat{z}+\hat{\omega}_d))\hat{z} + d_1, \tag{6.41}$$

where

$$\begin{aligned} d_1 = {} & E\Delta(v_a^b, t) + [\tilde{J} - E\hat{J}]\dot{\hat{\omega}}_d - V_a[\tilde{D} - E\hat{D}]\hat{\omega}_d + \boldsymbol{S}(\tilde{J}\hat{\omega}_d)\hat{\omega}_d \\ & - E\boldsymbol{S}(\hat{J}\hat{\omega}_d)\hat{\omega}_d - \delta B\hat{J}(\dot{\hat{\omega}}_d - \dot{\hat{\omega}}_d'). \end{aligned}$$

The time derivative of $V_1 = k_1\Psi(\Gamma, \Gamma_d)$ now becomes

$$\begin{aligned} \dot{V}_1 = k_1\left(\omega - \omega_d^{\|} - \boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp)\right)^\top e_\Gamma &= k_1\left(\hat{z} - \kappa e_\Gamma - \boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp)\right)^\top e_\Gamma \\ &= -\kappa k_1\|e_\Gamma\|^2 + k_1\hat{z}^\top e_\Gamma - k_1 e_\Gamma^\top\boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp). \end{aligned}$$

Let a Lyapunov function candidate be given by $\hat{V}_2 = V_1 + \hat{z}^\top J\hat{z}/2$, whose time derivative satisfies

$$\begin{aligned} \dot{\hat{V}}_2 = \dot{V}_1 + \hat{z}^\top J\dot{\hat{z}} &= -\kappa k_1\|e_\Gamma\|^2 - k_1 e_\Gamma^\top\boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp) + \hat{z}^\top\left[k_1 e_\Gamma + J\dot{\hat{z}}\right] \\ &= -\kappa k_1\|e_\Gamma\|^2 - \hat{z}^\top\left[\delta B K_2 + \boldsymbol{S}(\hat{\omega}_d)J\right]\hat{z} + \hat{z}^\top d_2 \end{aligned}$$

where $d_2 = d_1 - \delta B\tilde{\Delta} + k_1 E e_\Gamma - k_1 e_\Gamma^\top\boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp)$.

The time derivative of $\hat{\omega}_d$ is

$$\dot{\hat{\omega}}_d = \dot{\omega}_d^{\|} - \kappa\dot{e}_\Gamma,$$

where $\dot{e}_\Gamma$ can be bounded using

$$\|\dot{e}_\Gamma\| \le \|\omega_d^\perp\| + \|e_\omega\| \le 2\|\omega_d^\perp\| + \|\hat{z}\| + \kappa\|e_\Gamma\|.$$

From Assumption 6 we get

$$\begin{aligned} \|\dot{\hat{\omega}}_d\| &\le b_{\dot{\omega}_d^{\|}}\|\omega\| + c_{\dot{\omega}_d^{\|}} + 2\kappa\|\omega_d^\perp\| + \kappa\|\hat{z}\| + \kappa^2\|e_\Gamma\| \\ &\le \kappa(\kappa + b_{\dot{\omega}_d^{\|}})\|e_\Gamma\| + (\kappa + b_{\dot{\omega}_d^{\|}})\|\hat{z}\| + \left(b_{\dot{\omega}_d^{\|}}c_{\omega_d^{\|}} + c_{\dot{\omega}_d^{\|}} + 2\kappa c_{\omega_d^\perp}\right), \end{aligned}$$

where we have used $\|\omega\| \le \|\hat{z}\| + \|\omega_d^{\|}\| + \kappa\|e_\Gamma\|$, from the definition of $\hat{z}$.

Now, it is not difficult to show that d_2 satisfies

$$\|d_2\| \le a\|e_\Gamma\| + b\|\hat{z}\| + c,$$

where

$$a = k_1\|E\| + k_1 c_{\omega_d^\perp} + \kappa c_D V_{\max} + c_J\kappa\left(2\kappa + 2c_{\omega_d^\parallel} + b_{\dot\omega_d^\parallel}\right) + \kappa c_{\dot\phi_d}\sin c_{\theta_d} \tag{6.42}$$

$$b = c_J(\kappa + b_{\dot\omega_d^\parallel}) + c_{\dot\phi_d}\sin c_{\theta_d} \tag{6.43}$$

$$\begin{aligned} c = {} & c_J\left((c_{\omega_d^\parallel})^2 + b_{\dot\omega_d^\parallel}c_{\omega_d^\parallel} + c_{\dot\omega_d^\parallel} + 2\kappa c_{\omega_d^\perp}\right) + c_D V_{\max} c_{\omega_d^\parallel} + \|E\|\|\Delta\| + \|\delta B\|\|\tilde\Delta\| \\ & + c_{\omega_d^\parallel} c_{\dot\phi_d}\sin c_{\theta_d} + \kappa c_{\omega_d^\perp} + \frac{g}{V_{\min}\cos^2(c_{\phi_d})}c_{\dot\phi_d} + c_{\dot\phi_d}c_{\dot\theta_d} + c_{\ddot\phi_d}\sin c_{\theta_d}. \end{aligned} \tag{6.44}$$

Inserting this into the expression for $\dot{\hat V}_2$ gives

$$\begin{aligned} \dot{\hat V}_2 &\le -\kappa k_1\|e_\Gamma\|^2 - \left[\lambda_{\min}^{\mathbf{sym}(\delta B K_2)} - \lambda_{\max}^J(c_{\omega_d^\parallel} + \kappa)\right]\|\hat z\|^2 + \|\hat z\|\|d_2\| \\ &\le -\kappa k_1\|e_\Gamma\|^2 - \left[\lambda_{\min}^{\mathbf{sym}(\delta B K_2)} - \lambda_{\max}^J(c_{\omega_d^\parallel} + \kappa) - c_J(\kappa + b_{\dot\omega_d^\parallel})\right]\|\hat z\|^2 \\ &\qquad + a\|e_\Gamma\|\|\hat z\| + c\|\hat z\| \\ &\le -\kappa k_1\|e_\Gamma\|^2 - \gamma\|\hat z\|^2 + a\|e_\Gamma\|\|\hat z\| + c\|\zeta\| \\ &= -\zeta^\top L\zeta + c\|\zeta\|, \end{aligned}$$

where $\zeta = [\|e_\Gamma\| \;\; \|\hat z\|]^\top$ and

$$L = \begin{bmatrix} \kappa k_1 & -\frac{a}{2} \\ -\frac{a}{2} & \gamma \end{bmatrix},$$

which is positive definite if $\gamma > a^2/(4\kappa k_1)$. Clearly, the matrix $\mathbf{sym}(\delta B K_2)$ must be positive definite, and sufficiently so, satisfying

$$\lambda_{\min}^{\mathbf{sym}(\delta B K_2)} > \frac{a^2}{4\kappa k_1} + \lambda_{\max}^J(c_{\omega_d^\parallel} + \kappa) + c_J(\kappa + b_{\dot\omega_d^\parallel}).$$

If this holds, and c is sufficiently small, then the solutions are ultimately bounded with an ultimate bound that depends on the controller gains, model estimation errors, and the bounds on reference velocities [125]. $\square$

In essence, the matrix K_2 can be chosen such that the controller is robust to model uncertainties, even when the derivatives of the reduced-attitude reference are not available. However, a necessary condition is that the uncertainty in the input matrix is not too large. The condition $x^\top \delta B x > 0$ implies that the control direction is known up to an error of 90 degrees.

Remark 6.9. Global stability of the nominal system is a necessary condition for (global) ISS. Due to the topological obstruction to global stabilization on compact manifolds such as $\mathbb{S}^2$, a relaxed property of almost (global) ISS has been proposed in [9], and sufficient conditions based on dual Lyapunov techniques (density functions) [207] are given. In [243], a combination of Lyapunov and density functions are used to show almost ISS for systems with rotational degrees of freedom, illustrated using a perturbed nonlinear observer. In [10], a complementary set of tools is given, based on Lyapunov functions and the theory of stable and unstable manifolds of dynamical systems. It is shown that the downward equilibrium of a perturbed pendulum with friction is almost ISS. This is a system that is very much of a similar nature to the one considered in this paper. In [22], robustness on SO(3) is considered in the context of nonlinear complementary filters in the presence of measurement errors. "Divergence" of trajectories on SO(3) is defined as trajectories that converge to the manifold of maximum distance, i.e., the manifold of all rotations of angle 180 degrees. In contrast to [10], only kinematic systems are considered. While almost ISS could probably be shown in our case, the result of [10] only considers a perturbation that is independent of the state. Since in our case, the perturbation is state-dependent, we settle for a local property.

6.6 Simulation Results

In this section, simulation results are presented. We show results from an ideal Matlab-environment, as well as realistic software-in-the-loop simulations where discretization effects and simulated sensor noise are present. In both cases, reduced-attitude references are generated from roll and pitch angle references using Equation (4.2). A topic for future work is to investigate if a tailor-made guidance scheme can be designed that directly produces a reduced-attitude vector reference.

6.6.1 Matlab

Figures 6.1-6.6 show the simulation results for the adaptive backstepping controller (6.29) applied to a simulation model of the Aerosonde UAV [19]. The controller uses perfect estimates of the matrices B, J, and D but no information about $\Delta(v_a^b, t)$. While the control surface deflections are controlled by the attitude controller, a PI controller is used to control airspeed using throttle [19]. The airspeed reference is constant and set to $35\,\mathrm{m\,s^{-1}}$. The attitude controller parameters are set to $\kappa = 1, k_1 = 1, K_2 = \mathrm{diag}(7, 5, 7)$ and $K_3 = \mathrm{diag}(40, 30, 40)$. During the first 20 seconds, the reduced-attitude reference is constant, corresponding to $\phi_d = 60\,\text{deg}$ and $\theta_d = 15\,\text{deg}$, which might correspond to a sharp, climbing turn. During the last 20 seconds, we use the time-varying reference $A\cos(2\pi f(t-20))$, with amplitude A equal to the initial 20 seconds, and $f = 0.1\,\text{Hz}$ for roll and $f = 0.08\,\text{Hz}$ for pitch. The initial conditions are set to $\omega(0) = 0, \phi(0) = -40\,\text{deg}$ and $\theta(0) = -20\,\text{deg}$.

Figure 6.1 shows the tracking performance in terms of roll and pitch angles, while Figure 6.2 shows the vector coordinates $\Gamma \in \mathbb{S}^2$. The errors converge quickly from large initial values, and the velocity errors are kept close to zero throughout the maneuver. During the latter half of the simulated trajectory, a slightly deteriorated tracking performance is observed in pitch. This also applies to the turn rate, visualized in Figure 6.3. This is explained by looking at Figure 6.4: When stabilizing a constant reference, the assumption that aerodynamic moments are slowly-varying applies quite well, and the disturbance estimates converge towards their true values. When tracking a time-varying trajectory, however, this assumption seems to break down, which has a negative impact on tracking performance. This variation seems to be attributed to the variations in the angle of attack seen in Figure 6.5. Nevertheless, this simulated case study shows adequate tracking performance for both constant and time-varying reference trajectories. The effect of turn-coordination can be seen by observing the sideslip angle in Figure 6.5. During the first 20 seconds, the sideslip angle is reduced to zero. During the last 20 seconds, some variation is seen, but the sideslip angle is still kept at small values (less than 2 degrees). The control surface deflections are shown in the bottom half of Figure 6.5, and are smooth and well below the saturation limits, which in the simulation is set to $\pm 20\,\text{deg}$. Finally, the start of the maneuver is illustrated as a path on the two-sphere in Figure 6.6, which is an alternative to showing roll and pitch angles when visualizing reduced-attitude trajectories.

6.6.2 Software-in-the-Loop Simulation

This section showcases the efficacy of the control design via realistic SITL simulations. The controller is implemented in the ArduPilot [13] open-source autopilot framework for fixed-wing UAVs. We simulate our code using ArduPilot's SITL framework, using the JSBSim flight dynamics engine with a model of a SIG Rascal 110. Roll and pitch reference angles are provided by ArduPilot's guidance system, described in Section 2.6.2.

Figures 6.7-6.10 shows the result of a simulation run of the adaptive backstepping controller (6.29). However, as the derivatives of the reduced-attitude reference are not available in the ArduPilot code, we use a version of (6.29) where no information about the reference velocities is used in

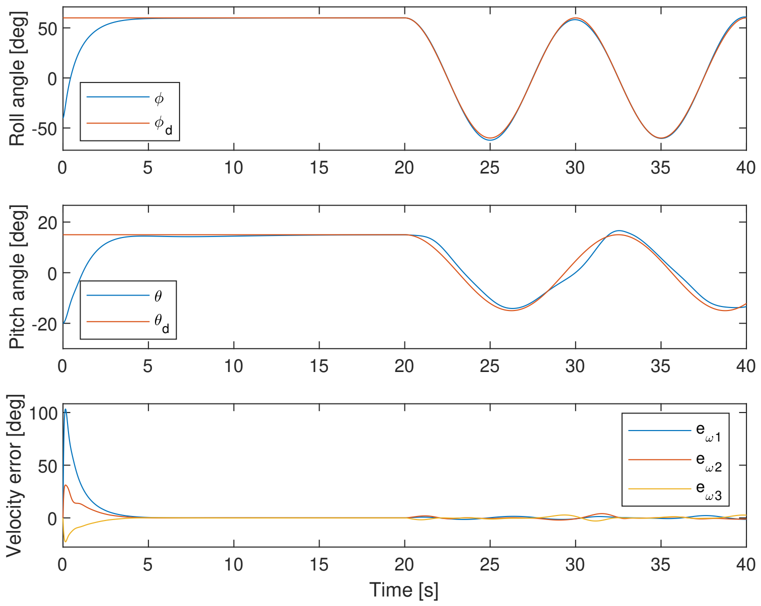


Figure 6.1: Roll and pitch angles vs. references, and velocity tracking error e_ω.

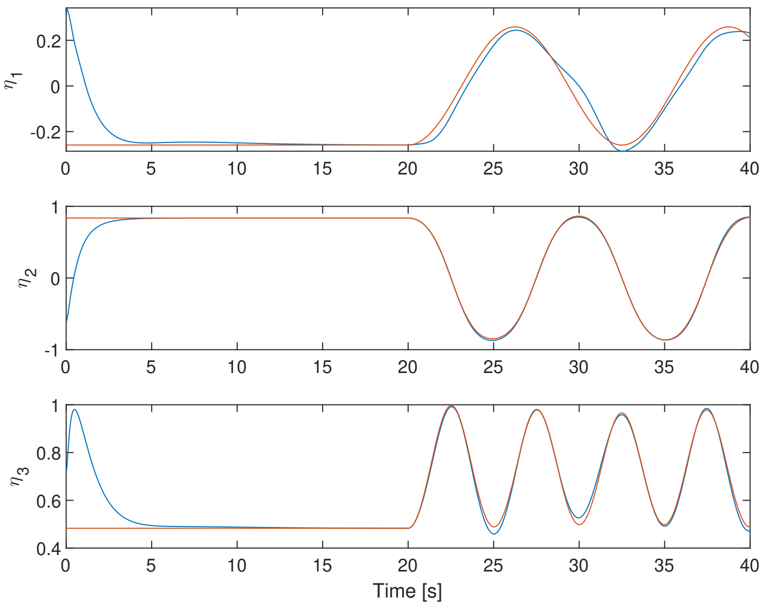


Figure 6.2: Reduced attitude Γ (blue) and the reduced-attitude reference Γ_d (red).

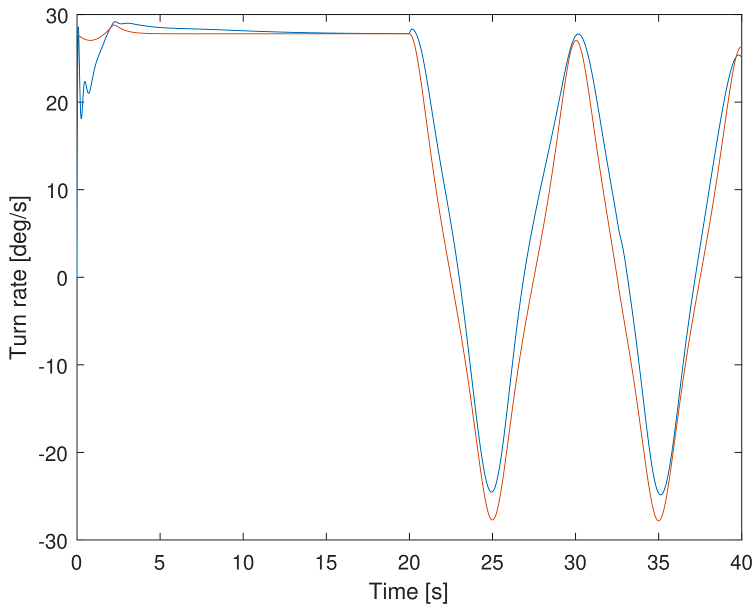


Figure 6.3: Magnitudes of $\omega^{\|}$ (red) vs $\omega_d^{\|}$ (blue), both parallel to Γ.

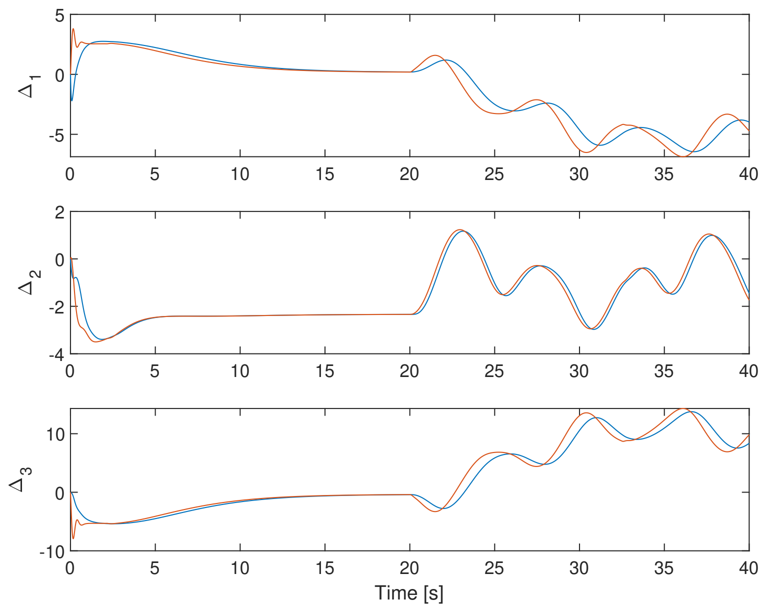


Figure 6.4: Disturbance estimates (blue) vs their true values (red).

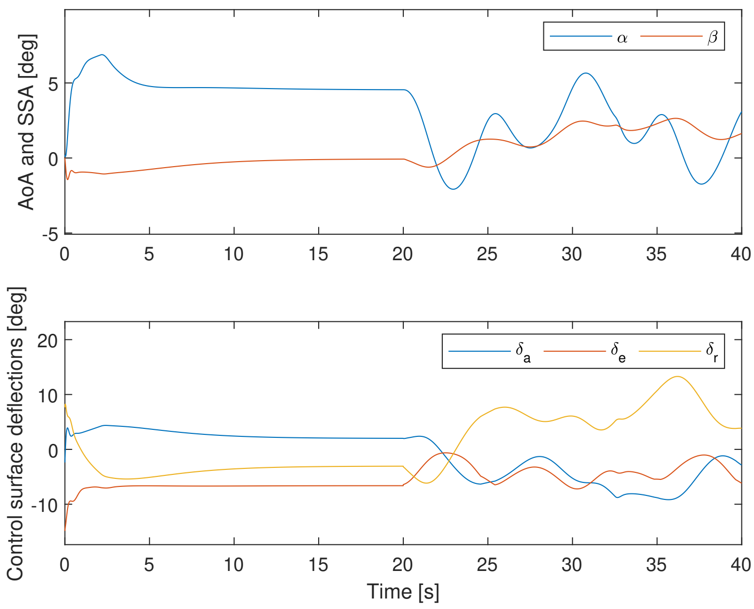


Figure 6.5: a) Flow angles: Angle of attack α and sideslip angle β; b) Control surface deflections: Aileron δ_a, elevator δ_e and rudder δ_r.

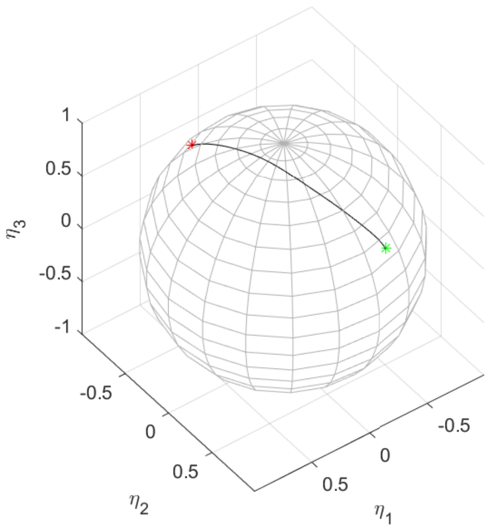


Figure 6.6: Path on the two-sphere for the 17 first seconds. Red asterix marks the initial configuration, while the constant reference is marked green.

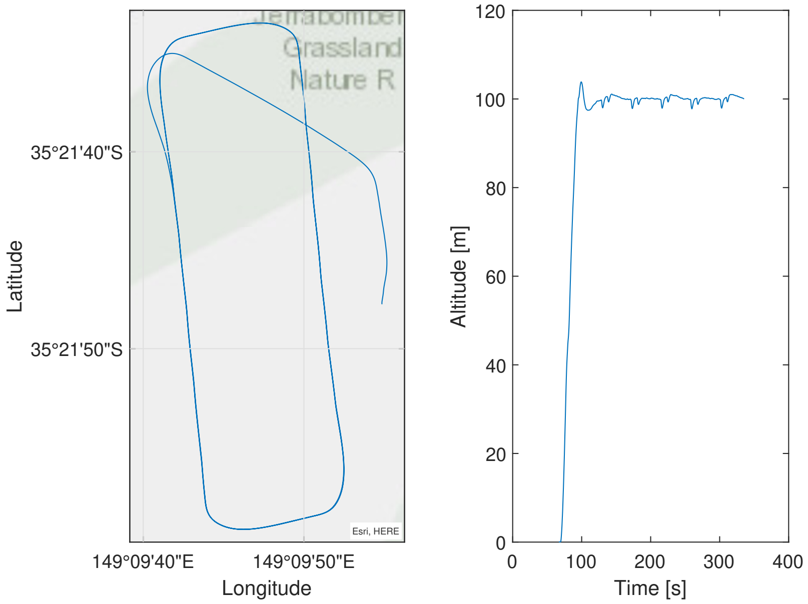


Figure 6.7: SITL: a) Geographical plot showing the horizontal path of the UAV; b) Altitude in meters (above home position).

the feedforward part of the controller. in addition, up to 20 percent uncertainty is added to all elements of the matrices J, B and D. This makes the controller more akin to (6.35), but with added integral action. The controller parameters are set to $\kappa = 2, k_1 = 10, K_2 = \text{diag}(5, 7, 5)$ and $K_3 = \text{diag}(0.1, 0.25, 0.1)$.

The simulated UAV is tasked with following a square pattern, shown in Figure 6.7. The actual horizontal position and altitude are shown, from takeoff and until a few rounds have been completed. It is clear that the proposed reduced-attitude controller successfully integrates into the ArduPilot infrastructure. Roll and pitch responses are shown in Figure 6.8, while Figure 6.9 shows the vector coordinates Γ. The UAV tracks the reference well, except when there are large steps in roll angle going into a turn. This is where a feedforward from the reference velocity could help reduce the errors. Anyhow, the errors are relatively small and do not interfere with the overall control objective. Figure 6.7 shows that the altitude is kept approximately constant at 100 meters, with only minor drops in altitude during sharp turns. The control input is shown in Figure 6.10. Except for some large spikes in the control surface deflections (due to large steps in roll reference when going into sharp turns), the control input is well behaved.

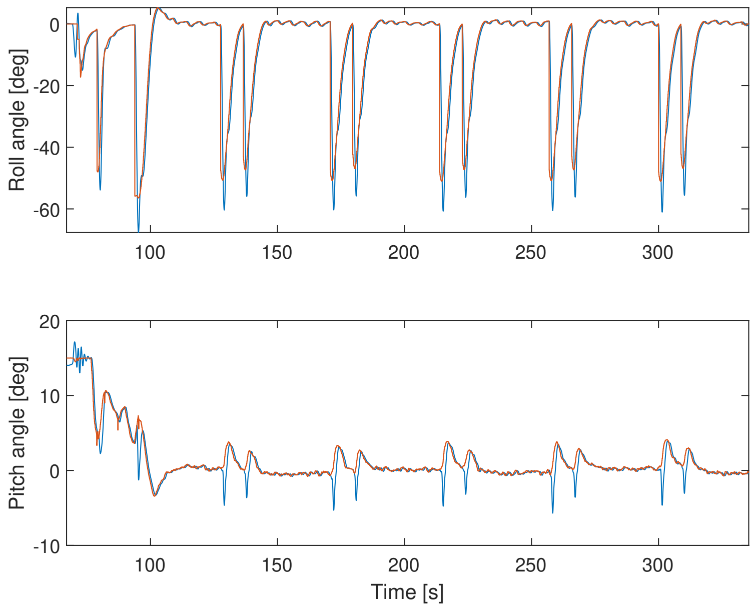


Figure 6.8: SITL: Roll and pitch angles (red) vs reference angles (blue).

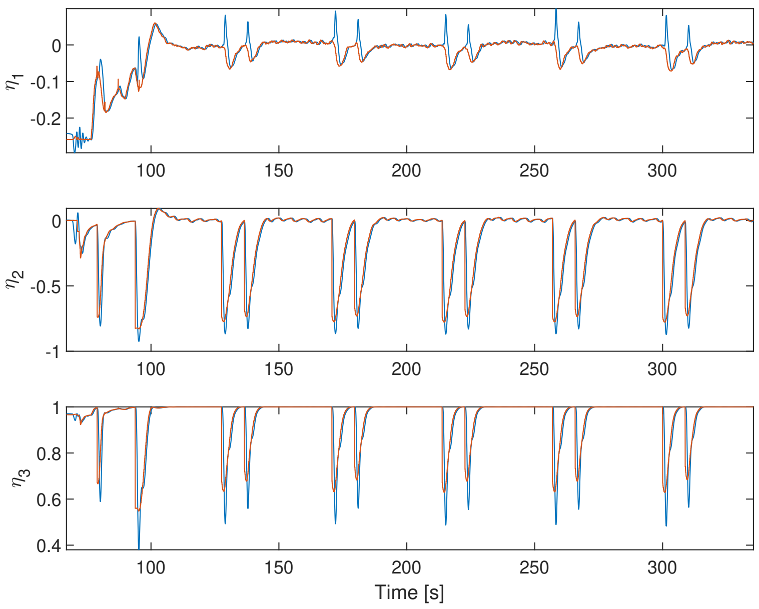


Figure 6.9: SITL: Reduced-attitude vector Γ (red) vs reference attitude (blue).

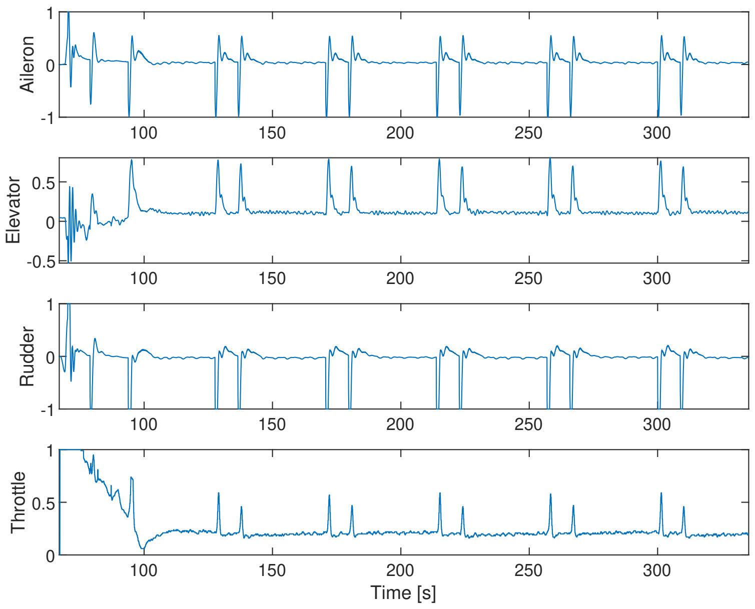


Figure 6.10: SITL: Control input. Control surface deflections are normalized to $[-1, 1]$ and the throttle to $[0, 1]$.

6.7 Chapter Summary

In this chapter, building upon the nominal control design of Chapter 4, we have designed more practical reduced-attitude controllers for fixed-wing aircraft. In particular, the controllers are less dependent on known system dynamics, and stability proofs are both simpler and less restrictive. Using a pertinent aerodynamic model of the rotational dynamics, almost global asymptotic stability is established for the proposed controllers. The efficacy of the presented approach was demonstrated using Matlab simulations as well as more realistic SITL simulations, where the control law shows that it successfully completes the defined control objectives in the presence of uncertain aerodynamics and reference velocities and integrates into a state-of-the-art open-source autopilot for fixed-wing UAVs.

Chapter 7

Robust Reduced-Attitude Control of Fixed-Wing UAVs Using a Generalized Multivariable Super-Twisting Algorithm

This chapter is based on the following article:

[59] E. M. Coates, J. B. Griffiths, and T. A. 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)*, 2021.

7.1 Introduction

Attitude control of fixed-wing unmanned aerial vehicles (UAV) is a challenging control problem, mainly caused by the highly uncertain aerodynamics of such vehicles. This is particularly true in heavy wind and turbulent conditions, as wind speeds often exceed 50 per cent of typical airspeeds flown by small UAVs [19]. Therefore, robust control laws could potentially enable the use of UAVs in a broader range of environmental conditions. Large wind gusts may bring the UAV far away from nominal conditions where longitudinal/lateral-decoupling assumptions are valid [233]. This motivates the use of multivariable methods to effectively deal with coupled uncertainties and disturbances.

Sliding mode control (SMC) [224, 241] is a robust control method that has gained popularity over the last decades due to its insensitivity to matched disturbances, i.e., disturbances that enter the system through the same channel as the control input. The robustness properties of SMC are attributed to a discontinuous control signal that also introduces a significant drawback to SMC design: the *chattering phenomenon*. Chattering is experienced as high-frequency oscillations in the control signal, which is undesirable in many control applications, including flight control. This has led to significant efforts to reduce chattering while retaining the desirable robustness properties. Several chattering-reduction techniques exist, including continuous approximations of the discontinuous SMC ("boundary-layer" methods) [43, 125], as well as higher-order SMC algorithms [147, 148].

A particular class of second-order SMC is the super-twisting algorithm (STA) [146]. STA applies to systems of relative degree one, and a continuous control signal is achieved by applying the discontinuous control input behind an integrator. While the stability properties of STA were originally proven using geometric methods or homogeneity approaches [148], Lyapunov methods have recently been applied [182, 183] to show finite-time convergence of the sliding-variable and its first derivative to zero.

The Lyapunov stability proof has enabled the research community to explore a range of extensions to the basic STA, including variable gain STA [88], adaptive STA [225], and generalized STA (GSTA) [181]. In [50], the GSTA is applied to systems whose perturbations and uncertain control coefficients are time- and state-dependent. An adaptive version of the GSTA was recently published in [34].

A multivariable super-twisting algorithm (MSTA) for multi-input-multi-output (MIMO) systems was recently presented in [186], based on unit-vector control [97, 98] and the Lyapunov ideas in [183]. While it is sometimes possible to decouple a MIMO system into several scalar single-input systems, this approach may fail if the interaction between the system variables is too strong [69]. Several extensions to the MSTA have appeared since [186] was published, including variable-gain MSTA [246], adaptive MSTA (AMSTA) [69, 71, 107, 237], the generalized MSTA (GMSTA) [160], and adaptive generalized MSTA (AGMSTA) [249]. A drawback of these algorithms is that perfect knowledge of the input matrix is needed. This is relaxed in [245] and [124], which give modified conditions on the parameters for the MSTA in [186] that allow for both symmetric [245] and non-symmetric [124] uncertain input matrices.

Operation of fixed-wing unmanned aerial vehicles (UAVs) outside their nominal operating conditions require autopilots that can effectively compensate for highly uncertain aerodynamics and coupled disturbances due to turbulent winds. In this chapter, we apply a generalized multivariable super-twisting algorithm [160] to the robust reduced-attitude control problem for fixed-wing UAVs. We design a novel sliding surface based on geometric methods to perform reduced-attitude tracking while simultaneously stabilizing a turn rate based on the coordinated-turn equation. In contrast to Chapter 4 and Chapter 5, where perfect model knowledge is assumed, and Chapter 6, where a nonlinear PID controller is designed using backstepping to account for slowly-varying aerodynamic uncertainties, we extend these results to a more general class of model uncertainties and wind disturbances. The controller is validated through simulations, and it is shown that the controller robustly stabilizes the desired trajectory in the presence of model uncertainties and highly turbulent wind conditions.

In the master's thesis [89], several different approaches to multivariable super-twisting attitude control of fixed-wing UAVs were compared. The GM-STA generally performed favourably and is, therefore, the focus of this chapter. Although STA has been applied to fixed-wing UAVs and aircraft previously [15, 49, 96, 102, 156, 204, 262], we apply a multivariable, generalized STA in combination with the proposed sliding surface, to achieve robust reduced-attitude tracking. Different geometric approaches to sliding-surface design have also been considered in [99, 131, 157] using rotation matrices and quaternions, respectively.

Chapter Outline

The rest of the chapter is organized as follows: A literature review on previous applications of multivariable super-twisting control is given in Section 7.2. Preliminaries on the generalized multivariable super-twisting algorithm are presented in Section 7.3 and in Section 7.4, the problem definition is stated. The control law, including the sliding surface design, is presented in Section 7.5, and in Section 7.6 we present a simulation case study. Finally, concluding remarks are given in Section 7.7.

7.2 Literature Review

This literature review aims to give an overview of related applications of the MSTA and its extensions, with a focus on aerospace control systems and attitude control designs. Section 7.2.1 presents research on the use of the MSTA, while section 7.2.2 presents several papers in which extensions of the MSTA are employed in aerospace and attitude control design.

7.2.1 Multivariable STA Applied to Attitude Control Design

An example of the use of the MSTA in attitude control problems in aerospace applications is given in [256]. In this paper, the control objective is to achieve continuous finite-time tracking of a trajectory for a spacecraft in the presence of external disturbances. Lyapunov methods are employed to prove the finite-time convergence of the closed-loop system to the equilibrium. Numerical simulations are performed to show the performance characteristics of the proposed controller. The simulations show the robustness of the controller in the presence of cyclic disturbance torques. Additionally, the resulting control signal is shown to be continuous and the error trajectories reach the equilibrium point in finite time.

Another example of the application of the MSTA in attitude control design is presented in [72]. In this paper, a robust trajectory tracking controller for a small unmanned helicopter with model uncertainties and external disturbances is designed. The controller is paired with a disturbance observer which is designed based on both backstepping and the MSTA. The system with the controller and disturbance observer is proven to be globally asymptotically stable through Lyapunov-analysis. Lastly, the proposed control design is compared through simulations with another control system design based only on backstepping. The tracking performance of the method proposed in [72] is shown to be more effective than the backstepping method, with smaller overshoot, faster tracking, and less chattering in the system.

The MSTA has also been applied in control design for attitude control of quadrotor UAVs, such as in [238]. Here, the control objective is to achieve continuous finite-time trajectory tracking. The finite-time stability of the closed-loop system is determined using the Lyapunov method and the homogeneous technique [238]. The performance of the control strategy is confirmed first through numerical simulations, then through experimental tests on a quadrotor UAV indoors. During the performance testing, the proposed method is compared with a PID controller. The controller based on the MSTA demonstrates better robustness and higher tracking accuracy than the PID controller due to the ability of the MSTA to reject disturbances.

7.2.2 Extensions of the Multivariable STA Applied to Attitude Control Design

An example of the application of the AMSTA in attitude control design is presented in [69] where an autopilot based on the AMSTA for a reusable launch vehicle (RLV) is presented. The control objective is to design the control torque so that the RLV can track the guidance commands in finite time in the presence of model uncertainties and external disturbances with unknown boundaries [69]. The paper states that the AMSTA based controller can adapt to additive and multiplicative perturbations with unknown boundaries while avoiding gain overestimation. It is confirmed through simulations that the proposed controller is effective and robust, and can provide fast and accurate tracking, and chattering suppression.

The control strategy presented in [71] is another approach to AMSTA. In this paper, an AMSTA is implemented for a hypersonic vehicle with the control objective of tracking a velocity and altitude reference in the presence of bounded but unknown perturbations. While the control objective in this article may not be to perform attitude control, the problem of following a velocity and altitude reference is in many ways similar to attitude control. Designing control systems for hypersonic vehicles is challenging due to the high-speed flight conditions and severe aerodynamic uncertainties [71]. It is, therefore, necessary to design a robust and effective controller to guarantee adequate tracking of the references. In [71], the controller is combined with a disturbance observer to further improve the tracking results.

In [237] an attitude control design for a quadrotor UAV using an AMSTA is presented. The UAV is assumed to be affected by unknown external disturbances with unknown bounds so that gain adaptation is necessary to avoid overestimation of the control gains. The proof of finite-time stability of the closed-loop dynamics is derived using the Lyapunov technique [237]. A comparison between using the AMSTA and single-input ASTA controllers is investigated, where the single-input design exhibits better accuracy in theory, while in practice the multivariable design proved to be easier to tune and gave the best results.

The AMSTA has also been employed in the control design for other autonomous vehicles, for example in [107]. In this paper, an AMSTA control strategy is applied to the problem of designing a lane-keeping control for four-wheel independently actuated autonomous vehicles. Since it is difficult to measure the lateral velocity [107] a high-order sliding mode observer is included in the system. Lyapunov methods are employed to prove the finite-time convergence of the closed-loop system. To verify the effectiveness of the proposed control strategy, simulations were performed in which the proposed controller was compared to a more traditional SMC approach. The simulations show the robustness of the adaptive controller in yielding a high-performance, fast and accurate lane-keeping control in a faulty steering situation [107].

An example of the application of the GMSTA to a problem that is similar to the problem of attitude control is presented in [227]. In this paper, an event-based GMSTA is used to perform path tracking to achieve safe navigation of a nonholonomic mobile robot in an unknown indoor environment. The event-triggered condition is obtained using Lyapunov theory to minimize the utilization of the resources. In addition to the Lyapunov analysis, a sensitivity analysis of the proposed controller is performed to ensure that the GMSTA is more robust than the MSTA from [186]. The performance of the proposed control strategy is compared to several other controllers through experiments for obstacle avoidance applications. The effectiveness of the proposed controller is shown in terms of error convergence rate and disturbance rejection capability with minimum control effort compared to the other controllers.

Another extension of the MSTA is the previously mentioned AGMSTA. A version of this algorithm is implemented in [249] in the control scheme for a space robot with coupled uncertainties and external disturbances. The space robot is a system with strongly coupled characteristics between the robot platform and the manipulators due to the absence of a fixed base. The advantages of the proposed controller over the original MSTA presented in [186] are that there are fewer conditions and parameters to design, and the ability to compensate for both Lipschitz continuous disturbances and state-dependent uncertainties in finite-time [249]. In addition to the AGMSTA, a sliding mode disturbance observer is introduced in the system to alleviate the system conservatism and improve convergence rate and accuracy [249]. Numerical simulations show the efficiency of the proposed control design in achieving tracking of the reference trajectory while compensating for external disturbances and coupled uncertainties. The simulations also show that improved convergence accuracy and rate are achieved when the disturbance observer is added to the control design.

7.3 The Generalized Multivariable Super-Twisting Algorithm

Sliding mode control design typically consists of two design steps:

1. Design of a sliding variable $\sigma \in \mathbb{R}^m$, such that, when the system dynamics are constrained to $\sigma = 0$, the compensated dynamics are of reduced order, and satisfy some control objective specified by the designer.
2. Design of a reaching law to control σ to zero.

Suppose that the sliding dynamics for a nonlinear system can be written on the form

$$\dot{\sigma} = a(t,x) + b(t,x)u + \gamma(t,\sigma), \tag{7.1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, $a(t,x) \in \mathbb{R}^m$ and $b(t,x) \in \mathbb{R}^{m\times m}$ are known functions, and $\gamma(t,\sigma,z) \in \mathbb{R}^m$ is an unknown disturbance. Assuming that the control-effectiveness matrix $b(t,x)$ is invertible, define the control u as

$$u = b^{-1}(t,x)\left(\bar{u} - a(t,x)\right). \tag{7.2}$$

The generalized multivariable super-twisting algorithm (GMSTA) presented in [160] is

$$\bar{u} = -k_1\phi_1(\sigma) + k_3\xi \tag{7.3}$$
$$\dot{\xi} = -k_2\phi_2(\sigma), \tag{7.4}$$

where $k_1, k_2, k_3 \in \mathbb{R}_{>0}$ and

$$\phi_1(\sigma) = \left(c_1\|\sigma\|_2^{-d_1} + c_2 + c_3\|\sigma\|_2^{d_2}\right)\sigma \tag{7.5}$$
$$\phi_2(\sigma) = \left(c_1(1-d_1)\|\sigma\|_2^{-d_1} + c_2 + c_3(1+d_2)\|\sigma\|_2^{d_2}\right)\phi_1(\sigma) \tag{7.6}$$

with

$$c_1 > 0, \quad c_2, c_3 \geq 0, \quad 0 < d_1 \leq \frac{1}{2}, \quad d_2 > 0.$$

The control law (7.2)-(7.4) transforms the system (7.1) into

$$\dot{\sigma} = -k_1\phi_1(\sigma) + k_3\xi + \gamma(t,\sigma) \tag{7.7}$$
$$\dot{\xi} = -k_2\phi_2(\sigma). \tag{7.8}$$

Let $z_1 = A\sigma \in \mathbb{R}^m$, where $A \in \mathbb{R}^{m\times m}$ is invertible. Furthermore, decompose the disturbance into $\gamma(t,\sigma) = \gamma_1(t,\sigma) + \gamma_2(t,\sigma)$. Then the change of variables $z_2 = \xi + (1/k_3)\gamma_2(t,\sigma)$ turns the system into

$$\dot{z}_1 = A\left[-k_1\phi_1(A^{-1}z_1) + k_3 z_2\right] + \delta_1(t,z_1) \tag{7.9}$$
$$\dot{z}_2 = -k_2\phi_2(A^{-1}z_1) + \delta_2(t,z_1), \tag{7.10}$$

where $\delta_1(t,z_1) \triangleq A\gamma_1(t,\sigma)$ and $\delta_2(t,z_1) \triangleq (1/k_3)d/dt[\gamma_2(t,\sigma)]$. If the perturbations $\delta_1(t,z_1), \delta_2(t,z_1)$ satisfy

$$\delta_1(t,\sigma) = AG_1(t)\phi_1(\sigma) \tag{7.11}$$
$$\delta_2(t,\sigma) = G_2(t)\phi_2(\sigma), \tag{7.12}$$

for some (possibly unknown) element-wise bounded matrices $G_1(t), G_2(t)$, then there exists controller parameters $k_i, i = [1,3]$ such that the origin of (7.9)-(7.10) is finite-time stable. An explicit algorithm to calculate such parameters, based on bounds on the elements of $G_1(t), G_2(t)$, can be found in [160]. It should also be noted that the right-hand-side of (7.9)-(7.10) is discontinuous, and solutions should be interpreted in the sense of Filippov [76].

A special case of the GMSTA arises if we consider $k_3 = 1, c_3 = 0$ and $d_1 = 1/2$. From (7.3)-(7.4), we are left with

$$\bar{u} = -c_1 k_1 \frac{\sigma}{\|\sigma\|^{\frac{1}{2}}} - c_2 k_1 \sigma + \xi \tag{7.13}$$

$$\dot{\xi} = -\frac{c_1^2 k_2}{2} \frac{\sigma}{\|\sigma\|} - c_1 c_2 k_2 \frac{\sigma}{\|\sigma\|^{\frac{1}{2}}} - c_2^2 k_2 \sigma, \tag{7.14}$$

which has (up to a constant in the second term of (7.14)) the same form as the adaptive GMSTA in [249], which is a multivariable extension of the GSTA [50, 181]. Furthermore, the GMSTA (7.13)-(7.14) reduces to the original MSTA in [186] if the second term in (7.14) is removed. In [186] and [249], the class of disturbances for which there exists controller parameters k_i such that finite-time stability is preserved, are those that satisfy

$$\|\delta_1(t, \sigma)\| \le \bar{\delta}_1 \|\sigma\| \tag{7.15}$$

$$\|\delta_2(t, \sigma)\| \le \bar{\delta}_2, \tag{7.16}$$

for some $\bar{\delta}_1, \bar{\delta}_2 \in \mathbb{R}_{>0}$.

We also note that if we in addition take $c_1 = 0$ and $c_2 = 1$, then $\phi_1(\sigma) = \phi_2(\sigma) = \sigma$ and (7.13)-(7.14) turns into

$$\bar{u} = -k_1 \sigma + \xi \tag{7.17}$$

$$\dot{\xi} = -k_2 \sigma, \tag{7.18}$$

which is a linear, proportional-integral controller in the sliding variable σ, akin to a typical backstepping controller.

7.4 Problem Definition

The control objective is the same as in the previous chapter, i.e., to simultaneously achieve $\omega^{\parallel} \to \omega_d^{\parallel}$ and $\Gamma \to \Gamma_d$, $\omega^{\perp} \to \omega_d^{\perp}$ as $t \to \infty$ (see Section 6.2.3).

As in previous chapters, the geometric methodology is based on the potential function $\Psi(\Gamma, \Gamma_d)$ and the corresponding configuration error vector e_Γ as defined by Eq. (4.18) and Eq. (4.19), respectively.

As in Chapter 6 we base our design on the rotational dynamics (2.46). However, in contrast to Chapter 6, we get an extra term, $-V_a D \omega_w$, related to the possibly non-zero angular velocity of the wind:

$$J\dot{\omega} = [\boldsymbol{S}(J\omega) + V_a D]\,\omega + V_a^2 Bu + h(v_a^b) + M_p(\delta_t) - V_a D\omega_w. \tag{7.19}$$

We treat ω_w, the relative velocity v_a^b (and thus also V_a, α, β) and the throttle setting δ_t as time-varying exogenous inputs. As before, the moment vectors $h(v_a^b)$ and $M_p(\delta_t)$ are assumed to be bounded, and the input matrix B is assumed to be square and invertible.

Remark 7.1. Wind and turbulence enters Equation (7.19) through ω_w, a time-varying airspeed V_a, and v_w, which affects $h(v_a^b)$ through v_a^b.

Remark 7.2. The assumption that $h(v_a^b)$ is bounded is an implicit assumption on stable internal dynamics. In section 7.5, a sliding variable σ is chosen, which corresponds to choosing a controlled output for the system. In particular, a necessary condition for SMC algorithms to be applicable is that the zero dynamics (the internal dynamics of the system when confined to the sliding surface) is stable. This is related to similar issues in input-output feedback linearization. See e.g.. [125] for a detailed discussion on these matters.

7.5 Control Law

7.5.1 Sliding Surface Design

Inspired by the backstepping design in Chapter 6, let the sliding variable $\sigma \in \mathbb{R}^3$ be given by

$$\sigma = \omega - \omega_d, \tag{7.20}$$

where $\omega_d = \boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp) - \kappa e_\Gamma + \omega_d^\parallel$, and $\kappa \in \mathbb{R}_{>0}$ is a design parameter. We can rewrite (7.20) as

$$\sigma = \underbrace{\left(\omega^\perp - \boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp) + \kappa e_\Gamma\right)}_{\in \mathrm{T}_\Gamma\mathbb{S}^2} + \underbrace{\left(\omega^\parallel - \omega_d^\parallel\right)}_{\in \mathrm{N}_\Gamma\mathbb{S}^2}. \tag{7.21}$$

Due to the orthogonality of the two parts, it is clear that when constrained to the sliding surface defined by $\sigma = 0$, the system dynamics collapse to $\omega^\parallel = \omega_d^\parallel$ and $\omega^\perp = \boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp) - \kappa e_\Gamma$. To show that the latter leads to convergence of Γ to Γ_d, we use the potential function (4.18) as a Lyapunov function candidate. $\Psi(\Gamma, \Gamma_d)$ is positive definite w.r.t. the equilibrium point $\Gamma = \Gamma_d$. The time derivative of $\Psi(\Gamma, \Gamma_d)$ along the trajectories of (4.5) and (4.15) satisfies

$$\dot{\Psi}(\Gamma, \Gamma_d) = e_\Gamma^\top \left[\omega^\perp - \boldsymbol{\Pi}_\Gamma^\perp(\omega_d^\perp)\right] = -\kappa\|e_\Gamma\|^2 < 0, \forall e_\Gamma \neq 0. \tag{7.22}$$

This shows that on the sliding surface, the desired reduced-attitude equilibrium $\Gamma = \Gamma_d$ is asymptotically stable. The fact that $\Psi(\Gamma, \Gamma_d)$ is constant for $e_\Gamma = 0$, means that there is an additional equilibrium point at $\Gamma = -\Gamma_d$. Since this represents a maximum of the Lyapunov function, and $\dot{\Psi}(\Gamma, \Gamma_d) < 0$ in any neighbourhood around this point, the equilibrium $\Gamma = -\Gamma_d$ is unstable. When the sliding surface is reached, convergence to the desired equilibrium is thus guaranteed for all initial reduced attitude configurations except the unstable point.

By using geometric concepts on the two-sphere $\mathbb{S}^2$, the sliding variable (7.20) combines the objectives of reduced-attitude tracking and coordinated banked turns, without singularities and with no conflicts between the two objectives. With the chosen error representation, (4.19), the closed-loop system does not exhibit the unwinding behavior [54]. In addition, errors in reduced attitude are compensated by commanding an angular velocity that is directed along the shortest path (a geodesic) on the sphere. As pointed out in previous chapters as well, potential shaping can be achieved by modifying (4.19), e.g. by multiplication with a nonlinear gain. Nevertheless, this does not change the general design procedure outlined above for the geometric sliding surface design on $\mathbb{S}^2$.

7.5.2 Reaching Law

The role of the reaching law is to control the sliding variable σ to zero in finite time. To this end, let a model-based feedforward term $u_{\mathrm{ff}} \in \mathbb{R}^3$ be given by

$$u_{\mathrm{ff}} = \hat{J}\dot{\omega}_d - V_a\hat{D}\omega_d - \boldsymbol{S}(\hat{J}\omega_d)\omega_d, \tag{7.23}$$

where $\hat{J}, \hat{D}$ are estimates of J, D, respectively. Also, let $\hat{h}(v_a^b)$ and $\hat{M}_p$ be estimates of $h(v_a^b)$ and M_p. In light of (7.2), let $a(t, x) = \hat{h}(v_a^b) + \hat{M}_p - u_{\mathrm{ff}}$ and $b^{-1}(t, x) = (1/V_a^2)B^{-1}J$ such that

$$u = \frac{1}{V_a^2}B^{-1}\left[J\bar{u} + u_{\mathrm{ff}} - \hat{h}(v_a^b) - \hat{M}_p\right], \tag{7.24}$$

and $\bar{u}$ is given by (7.3)-(7.4).

Remark 7.3. The feedforward term (7.23) is motivated by the fact that if $\hat{J} = J$ and $\hat{D} = D$, the controller $u = (1/V_a^2)B^{-1}[-K_\sigma\sigma + u_{\mathrm{ff}} - h(v_a^b) - M_p]$ achieves global asymptotic stability of the origin $\sigma = 0$. To see this, consider the Lyapunov function candidate

$$V_\sigma(\sigma) = \frac{1}{2}\sigma^\top J\sigma, \tag{7.25}$$

whose time derivative along the closed-loop trajectories satisfies

$$\begin{aligned}\dot{V}_\sigma(\sigma) &= \sigma^\top \left[-K_\sigma\sigma + V_a D\sigma + \boldsymbol{S}(J\omega)\sigma - \boldsymbol{S}(\omega_d)J\sigma\right] && (7.26)\\ &= -\sigma^\top \left[K_\sigma - V_a D + \boldsymbol{S}(\omega_d)J\right]\sigma && (7.27)\\ &\le -\left(\lambda_{\min}^{K_\sigma} - c_{\omega_d}\lambda_{\max}^{J} - V_{\max}\|D\|\right)\|\sigma\|^2 && (7.28)\\ &< 0, \quad \forall\sigma \ne 0, && (7.29)\end{aligned}$$

if $\lambda_{\min}^{K_\sigma} > c_{\omega_d}\lambda_{\max}^{J} + V_{\max}\|D\|$. Here, $\|\omega_d\| \le c_{\omega_d}$, $V_{\max}$ is an upper bound on V_a, and $\lambda_{\min}^{A}, \lambda_{\max}^{A}$ denotes the min and max eigenvalues of a matrix A, respectively. In many cases, the matrix D provides natural damping such that the preceding gain condition can be relaxed.

7.5.3 Disturbance Bounds

Let $z_1 = J\sigma$. Then

$$\dot{z}_1 = J\left[-k_1\phi_1(\sigma) + k_3\xi\right] + \gamma(t,\sigma,\omega,\omega_d,\dot{\omega}_d), \tag{7.30}$$

where

$$\begin{aligned}\gamma(t,\sigma,\omega,\omega_d,\dot{\omega}_d) = {}& V_a D\sigma + \boldsymbol{S}(J\omega)\sigma - \boldsymbol{S}(\omega_d)J\sigma - \tilde{M}_p\\ &- \tilde{h}(v_a^b) + \tilde{J}\dot{\omega}_d - V_a\tilde{D}\omega_d - \boldsymbol{S}(\tilde{J}\omega_d)\omega_d,\end{aligned} \tag{7.31}$$

and $\tilde{J} = \hat{J} - J$, $\tilde{D} = \hat{D} - D$, $\tilde{h}(v_a^b) = \hat{h}(v_a^b) - h(v_a^b)$ and $\tilde{M}_p = \hat{M}_p - M_p$. Now, with reference to (7.9)-(7.10), write γ in (7.31) as $\gamma = \gamma_1(t,\sigma,\omega) + \gamma_2(t,\omega_d,\dot{\omega}_d)$, with

$$\begin{aligned}\gamma_1(t,\sigma,\omega) &= V_a D\sigma + \boldsymbol{S}(J\omega)\sigma - \boldsymbol{S}(\omega_d)J\sigma && (7.32)\\ \gamma_2(t,\omega_d,\dot{\omega}_d) &= -\tilde{M}_p - \tilde{h}(v_a^b) + \tilde{J}\dot{\omega}_d - V_a\tilde{D}\omega_d - \boldsymbol{S}(\tilde{J}\omega_d)\omega_d. && (7.33)\end{aligned}$$

We note that γ_1 is vanishing at $\sigma = 0$, and that γ_2 vanishes as the model error ("tilde"-terms) goes to zero.

Let $z_2 = \xi + (1/k_3)J^{-1}\gamma_2$, $\delta_1 \triangleq \gamma_1(t,\sigma,\omega)$ and $\delta_2 \triangleq (1/k_3)J^{-1}\dot{\gamma}_2$. Then we recover the closed-loop structure of (7.9)-(7.10).

From (7.32), we get that, for any $c_\omega \in \mathbb{R}_{>0}$ such that $\|\omega\| \le c_\omega$, the following inequality holds:

$$\|\delta_1\| \le \big[\underbrace{V_{\max}\|D\| + (c_\omega + c_{\omega_d})\lambda_{\max}^{J}}_{\triangleq\bar{\delta}_1}\big]\|\sigma\|. \tag{7.34}$$

Therefore, the condition (7.15) is satisfied, and we move on to consider (7.16). Without calculating δ_2 explicitly, we see from (7.33) that $\dot{\gamma}_2$ is a function of time, $v_a^b, \dot{v}_r, \omega_d, \dot{\omega}_d$ and $\ddot{\omega}_d$:

$$\delta_2 = \frac{1}{k_3} J^{-1} \dot{\gamma}_2 = \delta_2(t, v_a^b, \dot{v}_r, \omega_d, \dot{\omega}_d, \ddot{\omega}_d) \tag{7.35}$$

By the definition of ω_d, (7.20), it is clear that $\dot{\omega}_d$ also depends on ω and $\ddot{\phi}_d$. Thus $\dot{\omega}$ enters into the expression for $\ddot{\omega}_d$. To satisfy (7.16), bounded relative velocities and accelerations, as well as bounded angular accelerations, are needed. Due to the smoothness of the equations of motion, (2.39) and (2.40), this will be satisfied in any compact domain of the state space. Additionally, the wind velocity and acceleration is bounded.

Although we do not explicitly calculate $\bar{\delta}_2$ in (7.16), we conclude that the value of $\bar{\delta}_2$ has a strong dependence on wind acceleration, angular velocity, and acceleration, as well as on how demanding the desired trajectory is. Instead of using such bounds to calculate controller gains for (7.3), (7.4), we argue that explicit gain calculation formulas derived from Lyapunov analysis often are conservative, and rather tune the parameters by trial and error. Another reason is that in practice, actuator dynamics, saturation effects, and discrete controls are present, all of which affect performance but are not explicitly treated in the analysis.

7.5.4 Input-Matrix Uncertainty

The MSTA in [186], as well as the GMSTAs in [249] and [160], all assume that the input matrix is fully known. In [245], a modification to the MSTA is provided which is robust to an uncertain input matrix, but only symmetric, positive definite input matrices are considered. This is relaxed in [124], where it is shown that the MSTA is robust to non-symmetric uncertain input matrices, but for large asymmetries, the algorithm becomes unstable. For the GMSTA however, no results on input-matrix uncertainty can be found, and it is unclear how such uncertainties affect the stability properties of the algorithm. Nevertheless, in this chapter, we simulate with input matrix uncertainty, and for the UAV model considered, asymmetry is relatively small.

7.6 Simulation Study

7.6.1 Simulation Environment

The six-degree-of-freedom nonlinear dynamic model of Section 2.4 is implemented in MATLAB/Simulink with parameters for the Aerosonde UAV from [19]. This also includes models for propeller thrust and aerodynamic forces. In our simulations, the UAV is subject to severe turbulence, shown in Figure 7.1, generated by the Dryden turbulence model [168]. The wind disturbance enters the equations of motion through (2.31).

The controller, given by Eq. (7.24), is run at discrete time instants at a constant rate of 50 Hz with zero-order hold on the commanded control surface deflection angles. This is the same as the sampling frequency of the ArduPilot fixed-wing autopilot [13]. The ode45 method is used for numerical integration between controller updates. To make the simulation model more realistic, measurement noise and a time delay of one timestep are added to the discrete measurements. Based on logged data from ArduPilot running on a Pixhawk autopilot [200] standing still on a desk, the measurement noise of the angular rate sensors is approximated as a zero-mean Gaussian noise process with a variance of $2.2\times10^{-7}\,\mathrm{rad}^2\,\mathrm{s}^{-2}$. In the logged data, the attitude estimates provided by the onboard extended Kalman filter show very little to no variance. Therefore, measurement noise in the attitude estimates is not considered in the simulation model. The discrete control surface deflection commands are run through the (continuous) second-order actuator dynamics defined by (2.50) with $\zeta = 1/\sqrt{2}$, $\omega_0 = 30\,\mathrm{rad\,s}^{-1}$. Magnitude and rate constraints of $\pm30\,\mathrm{deg}$ and $\pm200\,\mathrm{deg/s}$, respectively, are added to (2.50) using conditional integration.

When simulating our controller, the airspeed is controlled separately using the proportional-integral (PI) controller

$$\delta_t = \delta_{t,0} - 0.15(V_a - V_{a,d}) - 0.25\int_0^t (V_a - V_{a,d})d\tau, \tag{7.36}$$

where $V_{a,d} = 35\,\mathrm{m\,s}^{-1}$ is the airspeed reference, $\delta_t \in [0, 1]$ is the throttle, and $\delta_{t,0}$ is the nominal throttle calculated by a trim procedure for straight and level horizontal flight at $V_a = 35\,\mathrm{m\,s}^{-1}$.

7.6.2 Controller Implementation

Controller parameters are set to $\kappa = 1, c_1 = 0.6, c_2 = 1, c_3 = 0, k_1 = 5, k_2 = 20, k_3 = 1, d_1 = 1/2$. This corresponds to the special case (7.13)-(7.14). This is done to simplify tuning as it is still unclear what to gain in practice by allowing more general parameter values. This will be the topic of further investigation in the future. The parameters were tuned by trial and error to achieve low tracking errors without noticeable chattering in the control input.

In the implementation, $\|\sigma\|$ in (7.13) and (7.14) is replaced by $\|\sigma\| + \epsilon$ for some small $\epsilon > 0$ to avoid division by zero during simulation. A value of $\epsilon = 0.01$ seems to work well.

In (7.24), to really test the robustness of the controller, no model estimates are used, i.e., $u_{\text{ff}} = \hat{h}(v_a^b) = \hat{M}_p = 0$. In place of the true inertia and input matrices in (7.24), the following matrices are used to introduce uncertainty in the control directions in the simulation:

$$\hat{J} = \frac{A + A^\top}{2} \tag{7.37}$$

$$A = \begin{bmatrix} 1.2 & 0 & 0.1 \\ 0 & 1.13 & 0 \\ 0.1 & 0 & 0.8 \end{bmatrix} J \tag{7.38}$$

$$\hat{B} = \begin{bmatrix} 0.9 & 0 & 0.1 \\ 0 & 1.12 & 0 \\ -0.12 & 0 & 1.0 \end{bmatrix} B, \tag{7.39}$$

where J and B are the true matrices with parameters from [19].

7.6.3 Results

Figures 7.2-7.8 show the results of a simulation run where for the 20 first seconds, $\phi_d = 60\,\text{deg}$ and $\theta_d = 15\,\text{deg}$ are constant. In the following 40 seconds, $\phi_d = 60\cos(0.2\pi(t-20))\,\text{deg}$ and $\theta_d = 15\cos(0.16(t-20))\,\text{deg}$. These expressions are differentiated analytically, and $\omega_d^\perp$ is calculated as described in [209], Appendix E. The initial conditions are $\phi(0) = -40\,\text{deg}$, $\theta(0) = -20\,\text{deg}$ and $\omega(0) = 0$.

Despite the turbulent wind conditions and the fact that no feed-forward, including reference time derivatives, is used, the GMSTA achieves excellent tracking of reduced attitude. Tracking results for both constant and time-varying references are shown in terms of roll and pitch angles in Figure 7.2, and in terms of the vector coordinates Γ in Figure 7.3. To better visualize the spherical parameterization, a path on the sphere is shown in Figure 7.4, corresponding to the first 17 seconds of the simulation. No significant chattering can be observed in the control signal shown in Figure 7.5.

The sliding variable σ and GMSTA integrator state z is shown in Figure 7.6. Some oscillations of the sliding variable around $\sigma = 0$ can be

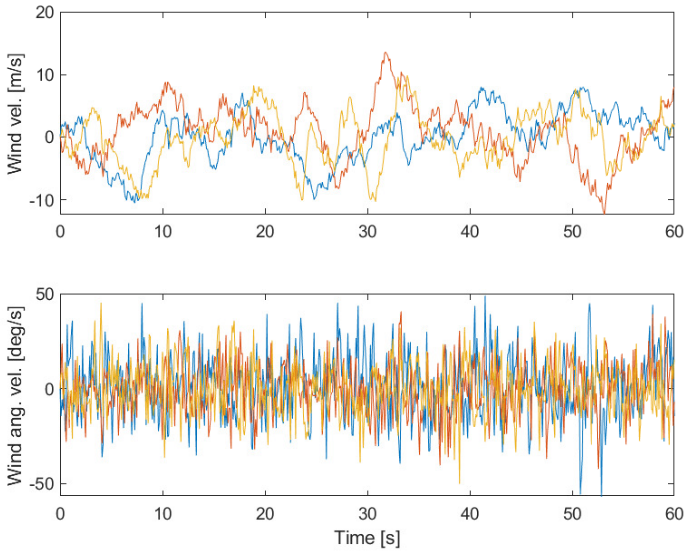


Figure 7.1: Wind turbulence generated by Dryden turbulence model: Linear wind velocity (top) and angular velocity (bottom).

observed. Theoretically, σ should stay at zero once the sliding surface is reached. However, the analysis does not consider discrete implementations and actuator dynamics. We have also tuned the controller to avoid significant chattering in the control signal, which conflicts with the objective of perfect disturbance rejection. In practice, due to the high degree of turbulence, such oscillations are to be expected.

The variation of σ seems to affect the parallel component of angular velocity the most, seen as high-frequent oscillations of $\omega^{\|}$ around $\omega_d^{\|}$ in Figure 7.7, which should be investigated further. Although there are some oscillations, the GMSTA fulfils the control objective of simultaneous reduced-attitude tracking and control of $\omega^{\|}$ to $\omega_d^{\|}$. Finally, airspeed, angle of attack, and sideslip are plotted in Figure 7.8, which displays the turbulent conditions. In light of (2.51), the sideslip angle is kept small throughout the manoeuvre.

7.7 Chapter Summary

This chapter has presented a solution to the robust reduced-attitude control problem for fixed-wing UAVs based on a generalized multivariable super-twisting algorithm. The sliding surface is designed by applying methods from

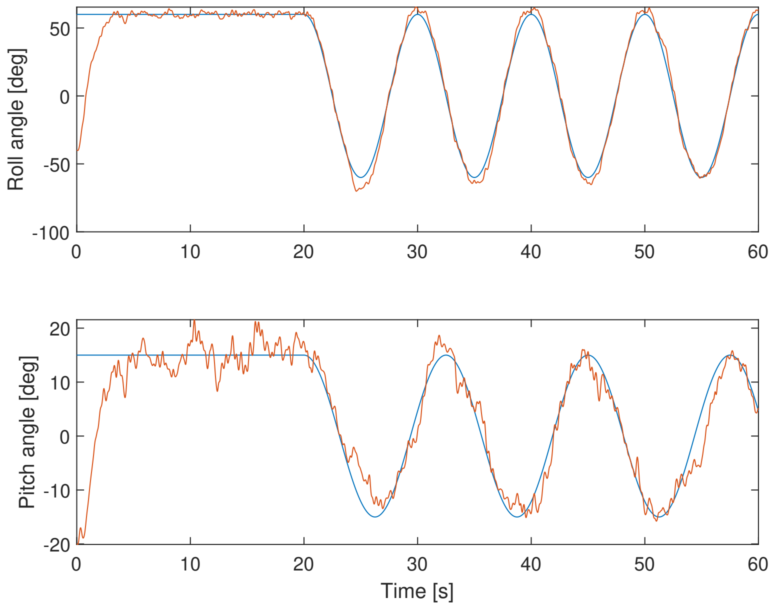


Figure 7.2: Roll and pitch angles (red) vs references (blue).

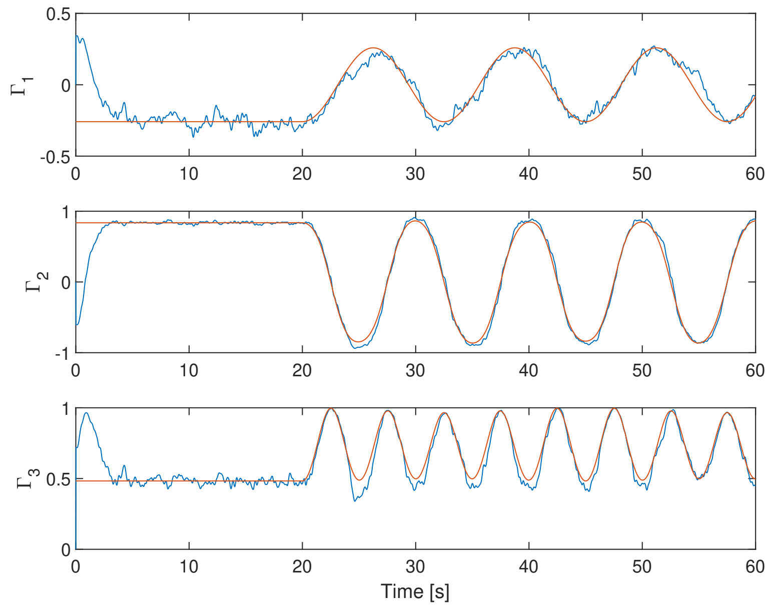


Figure 7.3: Reduced-attitude vector Γ (blue) vs reference Γ_d (red).

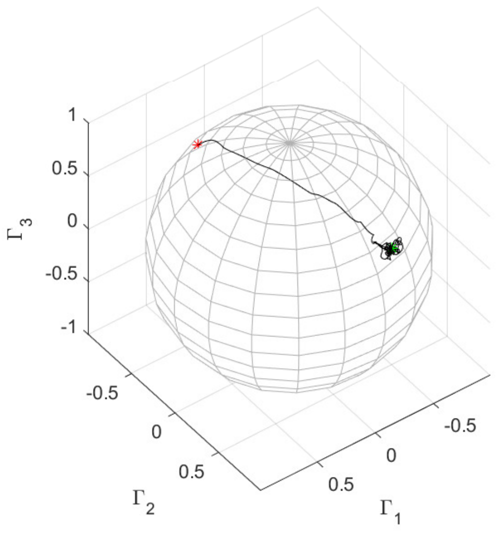


Figure 7.4: The 17 first seconds of simulated trajectory shown as a path on the sphere. Initial condition marked red. Reference marked green.

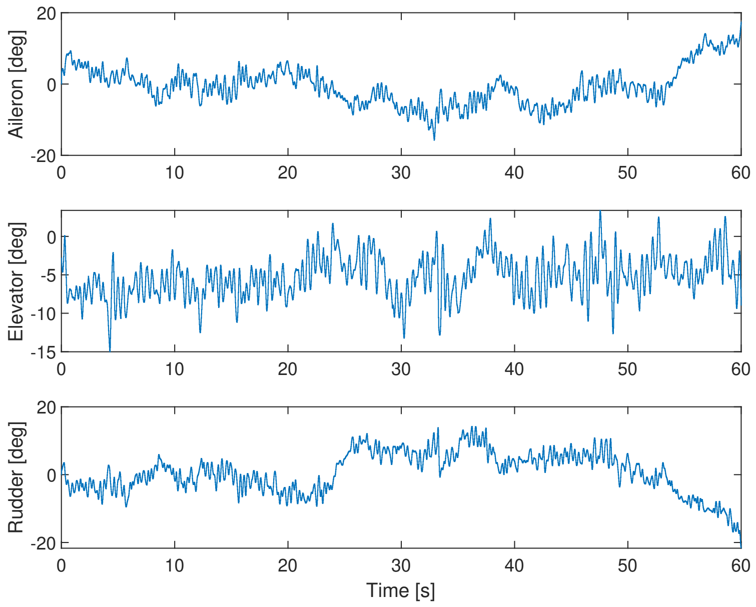


Figure 7.5: Commanded control surface deflections.

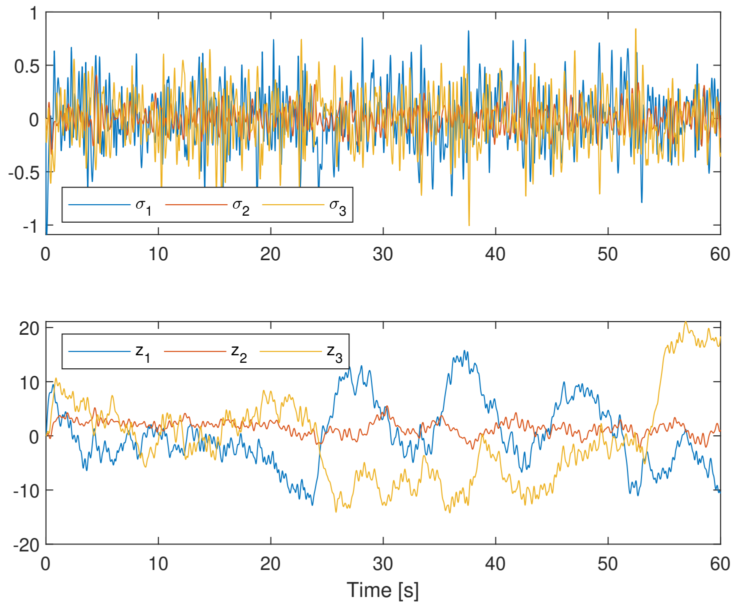


Figure 7.6: Sliding variable σ (top) and integrator state z (bottom).

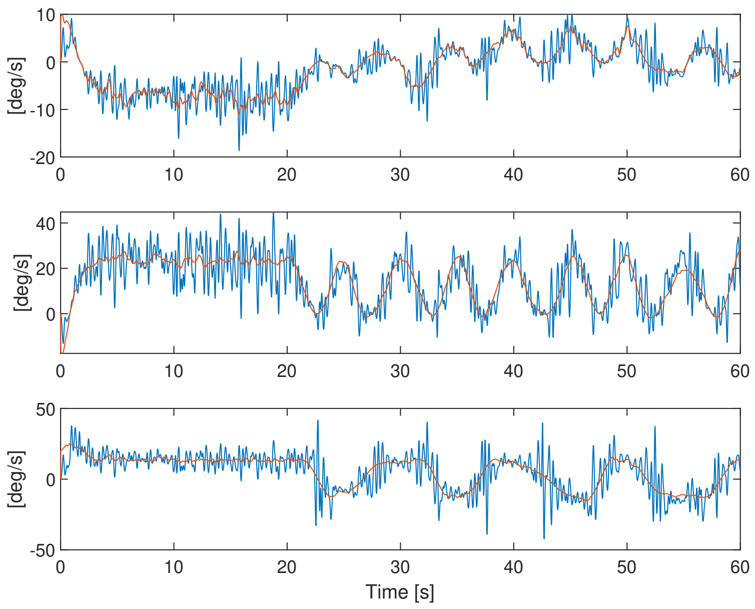


Figure 7.7: $\omega_d^{\parallel}$ (red) vs $\omega^{\parallel}$ (blue).

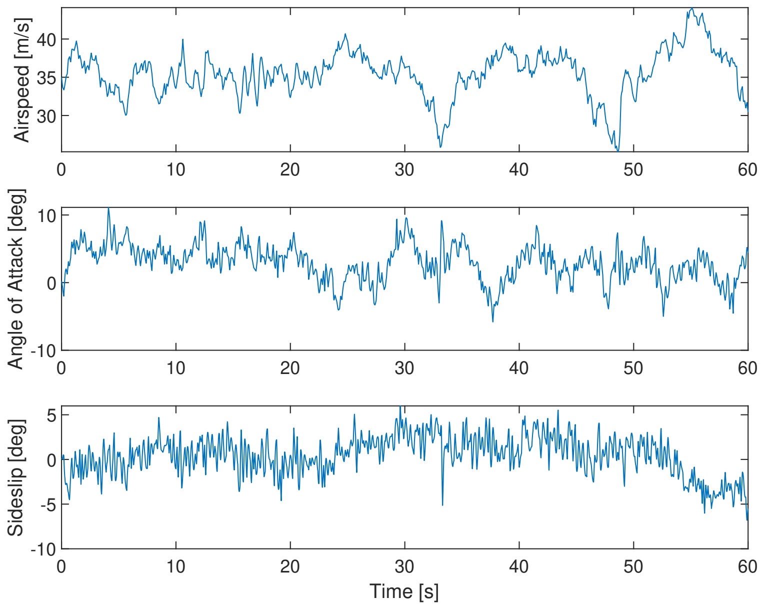


Figure 7.8: Airspeed, angle of attack and sideslip angle.

geometric attitude control combined with the coordinated-turn equation. The efficacy of the design has been demonstrated in a simulated environment. To evaluate the practical applicability of the methods, the simulation model includes several elements that can make real-life application of sliding mode control theory challenging: discrete measurements and controls, time delay, measurement noise, saturated actuator dynamics, as well as a turbulence model.

Part II

Path-Following Control

Chapter 8

Introduction to Part II

In Part II, which consists of one main chapter, we revisit the path-following guidance problem in three dimensions. First, by formulating the path-following error directly in the inertial frame, we propose a class of guidance laws for regular parametrised paths that, unlike most approaches existing in the literature, do not require the explicit construction of a path frame. Based on an inner-outer loop control paradigm, the guidance law generates a normal acceleration command that is normal to the flow-relative velocity vector (as the lift force). This allows for a natural decomposition of the desired vehicle acceleration for aerial vehicles in coordinated turns: tangential acceleration for airspeed control and normal acceleration for guidance generated through bank-to-turn manoeuvres, i.e., by tilting the lift vector. By using cascade arguments, we show that the proposed design leads to almost global stability results and thus relaxes the set of feasible initial conditions compared to existing methods. The efficacy of the proposed guidance law is demonstrated in a simulation study.

Chapter 9

Almost Global Three-Dimensional Path-Following Guidance Law for Arbitrary Curved Paths

This chapter is based on the following article:

[60] E. M. Coates, T. Hamel, and T. I. Fossen. Almost global three-dimensional path-following guidance law for arbitrary curved paths. In *62nd IEEE Conference on Decision and Control (CDC) (accepted)*, 2023.

9.1 Introduction

In the last decade, advances in hardware technology and control algorithms have led to increased use of small UAVs in a wide range of civil, commercial, and scientific applications [19].

In high-performance applications, the position of the UAV is typically controlled by either trajectory tracking or path-following algorithms [3]. The objective of path following is to ensure that the vehicle's position converges to and follows a desired geometric path without any temporal constraints [36].

For fixed-wing aircraft, path-following algorithms are usually preferred over trajectory tracking for the following reasons: Aerodynamic characteristics and performance depend on the air-relative velocity, so accurate trajectory tracking can be difficult. This is especially true for small fixed-wing

UAVs, where wind speeds comprise a high percentage of the total speed. Furthermore, the underactuated and nonminimum phase characteristics of typical fixed-wing aircraft further complicate the matter. Moreover, in [4], it is shown that path-following removes the fundamental performance limitations present in trajectory tracking for nonminimum phase systems.

During the last three decades, many path-following methods have appeared in the literature. We refer the reader to the recent review [109] for an in-depth treatment. The existing literature on 3D path following, e.g. [55, 56, 119], solve the control problem by specifying a guidance law in terms of *normal acceleration*, normal to the (ground) velocity vector of the aircraft. However, the control of fixed-wing aircraft is better defined in terms of the air-relative velocity vector for the following reasons:

- Performance specifications (lift, drag, stall, efficiency, etc.) of fixed-wing aircraft are defined using airspeed. Therefore, the airspeed should be carefully monitored and controlled during flight.
- A fixed-wing aircraft in a coordinated turn generate normal accelerations by reorienting the lift force through a banking ("bank-to-turn") maneuver. The lift force is normal to the relative velocity.
- Decomposing the desired acceleration in components orthogonal to, and in the direction of, the relative velocity allows us to decouple the guidance from airspeed control.
- In many applications, the discrepancy between the two different normal accelerations is ignored. However, this difference can be significant for fixed-wing aircraft flying in wind, resulting in disturbances that deteriorate the control performance. This is especially true for small UAVs, which often experience significant wind speeds compared to their airspeed.

In this chapter, we propose a class of guidance laws for almost-global path-following of any viable path in three dimensions, especially suited for fixed-wing aircraft. We employ an inner-outer loop control paradigm: In the outer loop, a desired heading vector (the velocity direction) is chosen that steers the vehicle toward the path. This is achieved using a simple line-of-sight (LOS)-like feedback law, and in contrast to most existing methods for 3D path-following [36, 55, 56, 119], we do not require the definition of a moving coordinate frame attached to the path. The desired heading vector is converted to a *relative* heading vector by carefully considering the wind triangle. In the inner loop, a control law on the two-sphere achieves the desired relative heading by specifying a normal acceleration, *normal to the relative velocity* as is natural for fixed-wing aircraft. The resulting closed-loop system is analyzed as a cascade and is shown to be almost globally asymptotically stable. The effectiveness of our approach is demonstrated in a simulation case study.

The main contributions can be summarized as follows:

1. We propose a novel guidance law for *arbitrary regular curved paths in 3D* with *almost global stability properties*. This contrasts with many existing works that either consider only a specific type of path, such as straight lines or circular paths, or constrain the allowable range of initial conditions.
2. The control input is the normal acceleration, *normal to the relative velocity*, making the proposed guidance law particularly suited for fixed-wing aircraft.
3. The control design and implementation is carried out *without any path frames* such as Frenet-Serret or parallel transport frames. This, in conjunction with the cascade design approach, results in a simple and modular design.
4. Finally, we establish the connection between the proposed guidance law and the classical LOS guidance law in 2D, showing that the proposed design is a natural generalization of LOS guidance to 3D.

Chapter Outline

This chapter is organized as follows: the problem formulation is stated in Section 9.2. Section 9.3 explains the derivation of the outer-loop guidance controller, which includes the desired relative heading as well as an update law driving the progression of the reference point on the desired path. The inner-loop heading control is discussed in Section 9.4, while Section 9.5 covers the analysis of the complete closed-loop system. The relation between our proposed guidance law and the classical LOS guidance law in 2D is established in Section 9.6. Numerical simulation results are presented in Section 9.7 before we summarize the chapter in Section 9.8.

9.2 Problem Formulation

Let $a \in \mathbb{R}^3$ denote an acceleration vector, and consider some $p \in \mathbb{S}^2$. Then $\mathbf{\Pi}_p^{\perp}(a) \in \mathrm{T}_p\mathbb{S}^2$ is the normal acceleration, normal to p, and $\mathbf{\Pi}_p^{\|}(a) \in \mathrm{N}_p\mathbb{S}^2$ is the tangential acceleration, in the direction of p. Let $a = \dot{v}$ and $v = V\eta$ with $V > 0$ and $\eta \in \mathbb{S}^2$. From the product rule, $\dot{v} = V\dot{\eta} + \dot{V}\eta$. It follows that $\mathbf{\Pi}_\eta^{\perp}(a) = V\dot{\eta}$ and $\mathbf{\Pi}_\eta^{\|}(a) = \dot{V}\eta$.

9.2.1 Dynamical Model

We address the guidance problem by considering the vehicle's second-order kinematics. We use a simplified model in which the vehicle is treated as a point particle moving in a uniform, constant wind field:

$$\dot{\xi} = v = V_a \eta_a + v_w \tag{9.1}$$

$$\dot{\eta}_a = \frac{1}{V_a} a_a^{\perp}, \tag{9.2}$$

where the vehicle's position and velocity are represented by $\xi \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$, respectively, and $v_w \in \mathbb{R}^3$ is the constant wind velocity, all expressed with respect to an inertial reference frame $\mathcal{F}_I = \{e_1, e_2, e_3\}$. By assuming that the ground speed $V = \|v\|$ never crosses zero, the velocity vector v is split into its ground speed $V = \|v\|$ and heading vector $\eta = v/V \in \mathbb{S}^2$. To account for the effects of wind on the vehicle's motion, we similarly rewrite the air-relative velocity $v_a = v - v_w$ as $V_a \eta_a$, with $V_a = \|v_a\|$ representing the airspeed and $\eta_a \in \mathbb{S}^2$ is the air-relative heading vector. The normal acceleration, $a_a^{\perp} \in \mathrm{T}_{\eta_a}\mathbb{S}^2$, is the control input for the path-following problem considered here. We assume that some inner-loop autopilot system controls $a_a^{\perp}$ to its desired value, e.g. using the method presented in [140]. For airspeed control, we assume that there is an independent inner-loop control law that uses the thrust input that stabilizes V_a to the desired value V_a^d (typically constant) greater than the stall airspeed V_a^s. From now on, we assume that V_a is constant and $V_a \geq V_a^s > V_w$, with V_w the wind magnitude $\|v_w\|$.

9.2.2 Path-Following Control Problem

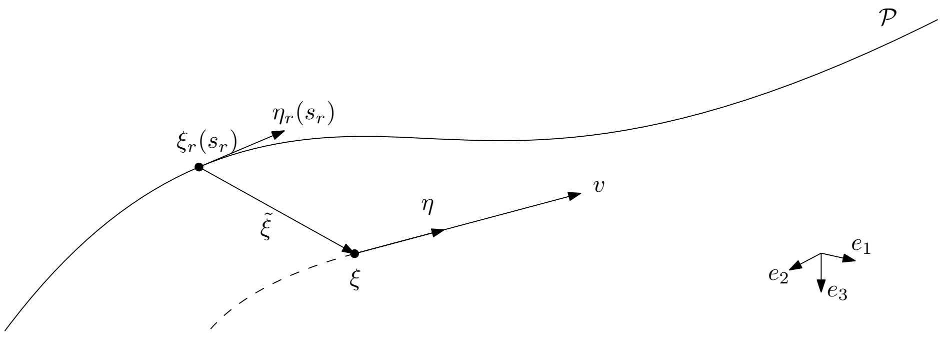


Figure 9.1: Desired path.

Let the desired path be represented by a regular curve $\mathcal{P}$ in 3D space with a desired orientation, parametrised by signed arc length $s_r \in \mathbb{R}$ (s_r increases as the point on the path moves along the curve).

We denote a reference position on the path by $\xi_r(s_r) \in \mathbb{R}^3$, and the tangent vector at $\xi_r(s_r)$ by $\eta_r(s_r) \in \mathbb{S}^2$. For regular curves, $\eta_r = \partial\xi_r/\partial s_r$ is always well defined. The reference velocity (on the path) is then given by

$$\dot{\xi}_r = (\partial\xi_r/\partial s_r)\dot{s}_r = V_r\eta_r, \tag{9.3}$$

where $V_r := \dot{s}_r$ is the "signed speed" on the path. For ease of notation, we denote $\eta_r(s_r)$ by η_r for the remainder of this chapter.

Let $\tilde{\xi} = \xi - \xi_r$ be the position error. From (9.1) and (9.3) it follows that

$$\dot{\tilde{\xi}} = v - V_r\eta_r(s_r) = V_a\eta_a + v_w - V_r\eta_r(s_r) \tag{9.4}$$

$$\dot{s}_r = V_r \tag{9.5}$$

As the regulation of V_a is ensured independently of the system under consideration, we can consider η_a and V_r as virtual controls to steer the vehicle towards the path and then follow the path onward in the direction of η_r by the asymptotic stabilization of the equilibrium $\tilde{\xi} = 0$. These key concepts are depicted in Fig. 9.1, which illustrates the main quantities involved in the control process.

We simplify the control design by dividing the path-following control problem into two subproblems:

1. Guidance that consists in defining η_a^d and V_r that ensure the asymptotic stabilization of $\tilde{\xi} = 0$ when $\eta_a \equiv \eta_a^d$ and consequently $V_r = V$.
2. Heading control that uses the normal acceleration $a_a^\perp$ to drive the actual direction η_a towards the desired direction η_a^d.

The resulting closed-loop system is then analyzed using the cascaded dynamical systems theory.

9.3 Guidance controller Design

The guidance controller design is based on the model (9.4)–(9.5). By using η_a as virtual control and assigning an update law for V_r, one can impose the dynamics $\dot{\tilde{\xi}} = f_1(\tilde{\xi}, s_r)$ for some function $f_1(\tilde{\xi}, s_r)$ such that for all t and s_r, $f_1(0, s_r) = 0$ and $\tilde{\xi}^\top f_1(\tilde{\xi}, s_r) < 0$ when $\tilde{\xi} \neq 0$. The global asymptotic (local exponential) stability proof of the origin $\tilde{\xi} = 0$ follows directly using the function $W_1(\tilde{\xi}) = (1/2)\|\tilde{\xi}\|^2$ along with standard Lyapunov stability arguments [125, Theorem 4.9]. Note however that $f_1(\tilde{\xi}, s_r)$ should be bounded since $\|\mathbf{\Pi}_{\eta_r}^\perp(f_1(\tilde{\xi}, s_r))\| \leq V_a$ and hence precludes the global exponential stability of $\tilde{\xi} = 0$, which is consistent with the prior work on guidance in 2D [81].

Using the fact that V_r is a degree of freedom that can be used at will in the η_r direction, it is natural to rewrite $f_1 = f_1(\tilde{\xi}, s_r)$ in two components: one in the direction of η_r and the other in the orthogonal space: $f_1(\tilde{\xi}, s_r) = \mathbf{\Pi}^{\|}_{\eta_r}(f_1(\tilde{\xi}, s_r)) + \mathbf{\Pi}^{\perp}_{\eta_r}(f_1(\tilde{\xi}, s_r))$.

At this stage, the virtual control input η_a^d and V_r enter into the equation via (9.4) using $\eta_a^d \equiv \eta_a$. The exact manner in which the virtual control input η_a^d and the input V_r enter imposes a class of feasible feedback $f_1(\tilde{\xi}, s_r)$ as discussed below.

9.3.1 Specifying the Progression of the Path Reference Point

Recalling (9.4) along with the fact that $\dot{\tilde{\xi}} = f_1(\tilde{\xi}, s_r)$ and pre-multiplying by $\eta_r^\top$ on both sides of the equation, one verifies that:

$$\eta_r^\top v - V_r \eta_r^\top \eta_r = \eta_r^\top f_1(\tilde{\xi}, s_r). \tag{9.6}$$

Since $\eta_r \in \mathbb{S}^2$, one can rewrite (9.6) as follows:

$$V_r = \eta_r^\top v - \eta_r^\top f_1(\tilde{\xi}, s_r) = \eta_r^\top v - \eta_r^\top \mathbf{\Pi}^{\|}_{\eta_r}(f_1(\tilde{\xi}, s_r)). \tag{9.7}$$

From there, one can choose the following feedback in the direction of η_r (or equivalently in the image of $\eta_r \eta_r^\top$):

$$\mathbf{\Pi}^{\|}_{\eta_r}(f_1(\tilde{\xi}, s_r)) = -k_1 \mathbf{\Pi}^{\|}_{\eta_r}(\tilde{\xi}), \tag{9.8}$$

A saturated version of the form:

$$\mathbf{\Pi}^{\|}_{\eta_r}(f_1(\tilde{\xi}, s_r)) = -\bar{\text{sat}}^{\Delta_1}\left(k_1 \eta_r^\top \tilde{\xi}\right) \eta_r, \tag{9.9}$$

with Δ_1 the saturation limit, is also a possible choice.

9.3.2 Specifying the Desired Heading Vector

To specify the desired relative heading vector η_a^d, we insert first the expression of V_r, (9.7), into (9.4) to get

$$\dot{\tilde{\xi}} = \mathbf{\Pi}^{\perp}_{\eta_r}(v) + \mathbf{\Pi}^{\|}_{\eta_r}(f_1(\tilde{\xi}, s_r)) = f_1(\tilde{\xi}, s_r). \tag{9.10}$$

By projecting both sides of the above equation by using $\mathbf{\Pi}^{\perp}_{\eta_r}(\cdot)$, one arrives at the condition:

$$\mathbf{\Pi}^{\perp}_{\eta_r}(V\eta) = \mathbf{\Pi}^{\perp}_{\eta_r}(f_1(\tilde{\xi}, s_r)), \tag{9.11}$$

or equivalently,

$$\mathbf{\Pi}^{\perp}_{\eta_r}(V_a\eta_a + v_w) = \mathbf{\Pi}^{\perp}_{\eta_r}(f_1(\tilde{\xi}, s_r)). \tag{9.12}$$

From there, one has to select $\eta_a^d \in \mathbb{S}^2$ to fulfil the objective. However, solving (9.12) for η_a can be challenging as it requires cancelling the wind effect and ensuring normalization of $\eta_a \in \mathbb{S}^2$. Therefore, to simplify this process, we will perform the assignment in two steps: 1) specify first η^d that satisfies (9.11) and then 2) transform η^d to η_a^d by involving the measure or an estimate of the wind according to (9.12). See the prior work [115] on how the wind can be estimated using standard sensors. We also refer the reader to the recent survey [239].

Desired heading η^d

The simplest choice for η^d is:

$$\eta^d = \frac{\eta_r - k_2\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi})}{\|\eta_r - k_2\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi})\|}, \tag{9.13}$$

where $k_2 > 0$. By referring to (9.11) it is clear that in this case, $\mathbf{\Pi}^{\perp}_{\eta_r}(f_1(\tilde{\xi}, s_r)) = -k_2V\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi})/\|\eta_r - k_2\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi})\|$. One can consider the following option to independently manage the aircraft's behaviour in both the horizontal and vertical planes:

$$\eta^d = \frac{\eta_r - (k_2\mathbf{\Pi}^{\perp}_{e3} + k_3e_3e_3^{\top})\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi})}{\|\eta_r - (k_2\mathbf{\Pi}^{\perp}_{e3} + k_3e_3e_3^{\top})\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi})\|}, \tag{9.14}$$

in which case

$$\mathbf{\Pi}^{\perp}_{\eta_r}(f_1(\tilde{\xi}, s_r)) = -\frac{V\mathbf{\Pi}^{\perp}_{\eta_r}((k_2\mathbf{\Pi}^{\perp}_{e3} + k_3e_3e_3^{\top})\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi}))}{\|\eta_r - (k_2\mathbf{\Pi}^{\perp}_{e3} + k_3e_3e_3^{\top})\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi})\|}, \tag{9.15}$$

and $k_2, k_3 > 0$. One verifies that if $k_2 = k_3$, (9.14) reduces to (9.13).

Remark 9.1. The expressions (9.13) and (9.14) are closely related to the classical *line-of-sight guidance laws* in 2D [81], see Section 9.6 for further details.

Similarly to (9.9) saturation functions can be introduced in the design of η^d to limit the maximum approach angle (w.r.t. η_r) when $\mathbf{\Pi}^{\perp}_{\eta_r}(\tilde{\xi})$ is large.

Desired relative heading η_a^d

To convert the desired heading vector to a desired *relative* heading vector, we will carefully consider the *wind triangle.*

Consider the wind triangle relating the velocity vector v, the air-relative velocity v_a and the wind velocity v_w:

$$v_a = v - v_w, \tag{9.16}$$

or equivalently:

$$V_a \eta_a = V\eta - v_w. \tag{9.17}$$

Dividing by V_a gives the following expression for the relative heading vector as a function of V, η, V_a, and the wind velocity v_w:

$$\eta_a = \frac{V}{V_a}\eta - \frac{v_w}{V_a}. \tag{9.18}$$

From geometrical considerations (see Fig. 9.2), one directly derives the following expression for the actual ground speed V as a function of V_a, η and v_w:

$$V = \eta^\top v_w + \sqrt{(\eta^\top v_w)^2 + V_a^2 - V_w^2}. \tag{9.19}$$

From there, one defines the expression for the desired relative heading:

$$\eta_a^d = \frac{V_d}{V_a}\eta^d - \frac{v_w}{V_a}. \tag{9.20}$$

where

$$V_d = v_w^\top \eta^d + \sqrt{(v_w^\top \eta^d)^2 + V_a^2 - V_w^2}. \tag{9.21}$$

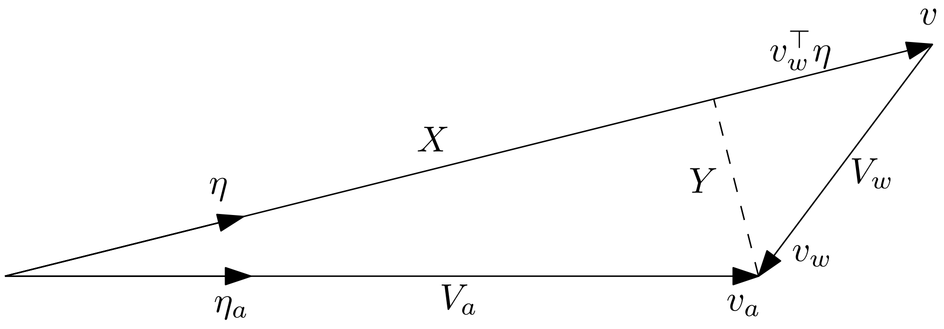


Figure 9.2: The wind triangle. From the Pythagorean theorem, $Y = \sqrt{V_w^2 - (v_w^\top \eta)^2}$ and $X = \sqrt{(v_w^\top \eta)^2 + V_a^2 - V_w^2}$. Finally, $V = v_w^\top \eta + X$ to derive (9.19)

It is straightforward to verify by direct computation of $\|\eta_a^d\|$ that η_a^d is a unit vector and that V_d is the normalizing factor required to compensate for v_w in (9.12). In particular, by inserting $\eta_a \equiv \eta_a^d$ with η_a^d defined by (9.20) into (9.12) one arrives at $\mathbf{\Pi}_{\eta_r}^{\perp}(V_d\eta_d)$. Finally, one notes that when $v_w = 0$, $\eta_a^d = \eta^d$.

9.4 Heading Control

In this section, we solve the (relative) heading control problem by choosing a control law for normal acceleration $a_a^{\perp}$ in (9.2), ensuring that the actual relative heading vector η_a tracks the desired heading vector η_a^d.

Note that the set of all possible heading vectors η_a is the two-sphere $\mathbb{S}^2$, and for each point $\eta_a \in \mathbb{S}^2$, the normal acceleration $a_a^{\perp} \in \mathrm{T}_{\eta_a}\mathbb{S}^2$ is a vector in the tangent space at η_a.

Several control laws for systems evolving on spheres can be found in the literature, e.g. [41, 53, 64]. We propose to use the following:

Proposition 9.1: Consider the system (9.2) and a desired relative heading vector $\eta_a^d \in \mathbb{S}^2$. Chose the following expression for $a_a^{\perp}$:

$$a_a^{\perp} = V_a^2 k_\eta \mathbf{\Pi}_{\eta_a}^{\perp}(\eta_a^d) + V_a \eta_a \times \left(\dot{\eta}_a^d \times \eta_a^d\right), \quad k_\eta > 0. \tag{9.22}$$

Then,

1. there are two closed-loop equilibria, given by $\eta_a = \pm\eta_a^d$.
2. the desired equilibrium $\eta_a = \eta_a^d$ is almost globally asymptotically stable and locally exponentially stable, uniformly in t_0.
3. the undesired equilibrium $\eta_a = -\eta_a^d$ is unstable.

Proof. Let $\tilde{\eta}_a := \eta_a - \eta_a^d \in \mathbb{B}_2$ denote the relative heading tracking error and consider the storage function given by

$$W_2(\tilde{\eta}_a) = \frac{1}{2}\tilde{\eta}_a^\top \tilde{\eta}_a = 1 - \eta_a^\top \eta_a^d. \tag{9.23}$$

One verifies that the time derivative of $W_2(\tilde{\eta}_a)$ is:

$$\begin{aligned} \dot{W}_2(\tilde{\eta}_a) &= \tilde{\eta}_a^\top \dot{\tilde{\eta}}_a = -V_a k_\eta {\eta_a^d}^\top \mathbf{\Pi}_{\eta_a}^{\perp}(\eta_a^d) && (9.24)\\ &= -V_a k_\eta \|\eta_a \times \eta_a^d\|^2 \le 0. && (9.25) \end{aligned}$$

The stated properties in the proposition then follow from standard Lyapunov arguments, see e.g. [58, Proposition 1]. □

Remark 9.2. Since $\mathbb{S}^2$ is a compact manifold, almost global asymptotic stability is the strongest stability result possible using continuous feedback [26]. However, global asymptotic stability can be achieved using hybrid control, where hysteresis-based switching ensures that *all* trajectories converge to the desired equilibrium [47, 143, 172, 209].

9.5 Analysis of the Complete Closed-Loop System

Proposition 9.2 (Main Result)**:** Consider the second-order kinematics (9.1)–(9.2) under the control (9.22) with the desired relative heading defined as in (9.20)–(9.21), and V_r given by (9.7). Then the equilibrium point $(\tilde{\xi}, \tilde{\eta}_a) = (0, 0)$ is almost globally exponentially stable and locally exponentially stable, uniformly in t_0.

Proof. To simplify the analysis, we consider the closed-loop system along the solutions $s_r(t)$ of (9.5), which lets us analyze the system as a cascade. The rigorous foundation that allows us to do this can be found in [158] and is based on the notion of *forward completeness* [11], which can easily be shown to hold for our system considered here.

Following the developments of Section 9.3 and substituting $\tilde{\eta}_a = \eta_a - \eta_a^d$ then leads to

$$\dot{\tilde{\xi}} = f_1(\tilde{\xi}, s_r(t)) + V_a \mathbf{\Pi}_{\eta_r}^{\perp}(\tilde{\eta}_a). \tag{9.26}$$

By construction, the origin $\tilde{\xi} = 0$ of the unperturbed system $\dot{\tilde{\xi}} = f_1(\tilde{\xi}, s_r(t))$ is uniformly globally asymptotically (locally exponentially) stable. Further, the perturbation term $V_a \mathbf{\Pi}_{\eta_r}^{\perp}(\tilde{\eta}_a)$ is bounded, since $\tilde{\eta}_a \in \mathbb{B}_2$. Moreover, from Proposition 9.1 we know that for all initial conditions except the (zero measure) set where $\|\tilde{\eta}_a\| = 2$, $\tilde{\eta}_a$ is converging (ultimately exponentially) to zero and thus the integral $\int_{t_0}^{\infty} \|\tilde{\eta}_a(t)\| dt$ is bounded. By standard cascade arguments, e.g. [159, Theorem 2.1], with a minor technicality being that $\tilde{\eta}_a$ belongs to the compact space $\mathbb{B}_2$, one concludes that the equilibrium point $(\tilde{\xi}, \tilde{\eta}_a) = (0, 0)$ is almost globally asymptotically stable, and locally exponentially stable, uniform in t_0. □

The region of attraction to the desired equilibrium point only excludes initial conditions in the set $\mathbb{R}^3 \times \partial\mathbb{B}_2$ which has zero measure in $\mathbb{R}^3 \times \mathbb{B}_2$. The region of attraction is thus *almost global.* In practice, however, convergence is essentially global: any slight perturbation of $\|\tilde{\eta}(t_0)\|$ would cause the heading to converge to the desired heading. Global asymptotic stability can be achieved using hybrid control, where hysteresis-based switching ensures that all trajectories converge to the desired equilibrium [47, 143, 172, 209].

9.6 Relation to Classical Line-of-Sight Guidance Laws Using Angles

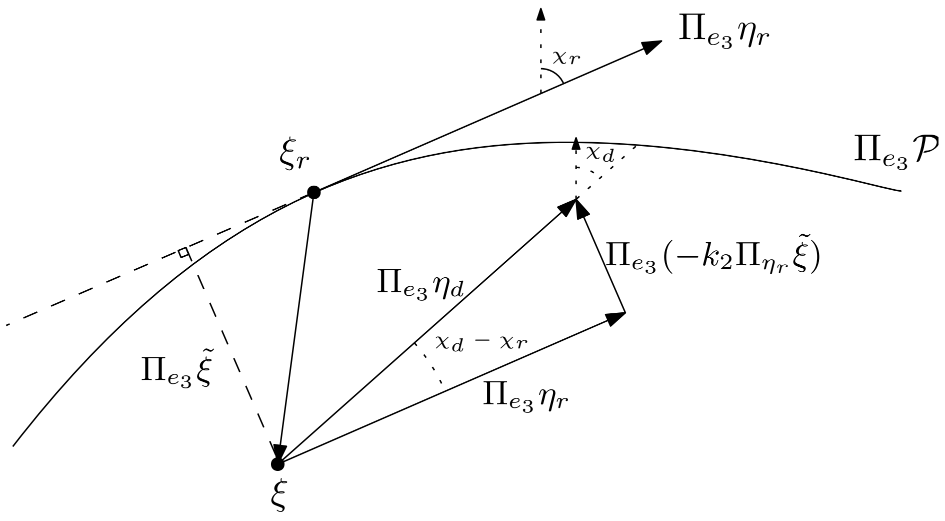


Figure 9.3: Projection to the horizontal plane. The magnitude of $\mathbf{\Pi}_{e_3}^{\perp}(\eta_r)$ equals $\cos(\gamma_r)$, and $\|\mathbf{\Pi}_{e_3}^{\perp}(\mathbf{\Pi}_{\eta_r}^{\perp}(\tilde{\xi}))\| = |y_e|$.

Let parametrisations of η_r and η^d based on the commonly used spherical coordinates be given by:

$$\eta_r = \begin{bmatrix} \cos(\gamma_r)\cos(\chi_r) \\ \cos(\gamma_r)\sin(\chi_r) \\ -\sin(\gamma_r) \end{bmatrix}, \quad \eta^d = \begin{bmatrix} \cos(\gamma_d)\cos(\chi_d) \\ \cos(\gamma_d)\sin(\chi_d) \\ -\sin(\gamma_d) \end{bmatrix} \tag{9.27}$$

where the pairs χ_r, γ_r and χ_d, γ_d are the azimuth and elevation angles of η_r, η^d, respectively.

We can derive a relation between the guidance laws (9.13)–(9.14) and the classical angle-based line-of-sight (LOS) guidance laws in 2D (e.g. [36, 81]) by projecting to the horizontal plane using the projection $\mathbf{\Pi}_{e_3}^{\perp}(\cdot)$. The projection is illustrated in Fig. 9.3. By considering the projection of the right triangle made up by η^d and its components in the direction of, and orthogonal to η_r, Equations (9.13) and (9.14) can be expressed in terms of azimuth (course) angles as

$$\chi_d = \chi_r + \arctan\left(\frac{-k_2 y_e}{\cos(\gamma_r)}\right), \quad \gamma_r \neq \pm\frac{\pi}{2} \tag{9.28}$$

where y_e is the *cross-track error*, i.e. the horizontal component of $\mathbf{\Pi}_{\eta_r}^{\perp}(\tilde{\xi})$, given by $y_e := (\mathbf{\Pi}_{\eta_r}^{\perp}(\tilde{\xi}))^{\top}(e_3 \times \eta_r)^{\top}/\|e_3 \times \eta_r\|$. The expression (9.28) is equivalent to that given in [36, 81] with $\Delta = \cos(\gamma_r)/k_2$. The control laws (9.13)–(9.14) thus *provide natural generalizations of 2D LOS guidance laws to 3D* using unit vectors in $\mathbb{S}^2$.

9.7 Simulation Results

In this section, we consider a simulation scenario to demonstrate the effectiveness of the proposed guidance law.

The specific choice of feedback used are given by (9.9) and (9.13) with the following set of control parameters: $k_1 = 20$, $\Delta_1 = 50$, $k_2 = 0.01$, and $k_\eta = 0.025$. Further, the simulation has been set up with $V_a = 18\,\mathrm{m\,s^{-1}}$ and $v_w = [10\,\mathrm{m\,s^{-1}}, 0, 0]^\top$. The desired path is the arc length parametrisation of a helix of radius $R = 200\,\mathrm{m}$ and vertical separation $h = 100\,\mathrm{m}$:

$$\xi_r(s_r) = \begin{bmatrix} R\cos(\frac{s_r}{\sqrt{R^2+c^2}}) \\ R\sin(\frac{s_r}{\sqrt{R^2+c^2}}) \\ \frac{c}{\sqrt{R^2+c^2}} s_r \end{bmatrix}, \tag{9.29}$$

where $c = h/(2\pi)$. The initial conditions used are $\xi(t_0) = [0, 0, 0]^\top$ (in the center of the helix) and $\eta_a(t_0) = [-1, 0, 0]^\top$, resulting in a large initial heading error.

The simulation results are shown in Figs. 9.4–9.7 and show that after an initial transient, perfect path following is achieved, even for a very large initial heading error.

Fig. 9.4 shows how the vehicle (red) starts in the centre of the helix and then converges to the path. In the bottom of Fig. 9.5, the angle θ between η_a and η_a^d is shown to converge rapidly to zero after an initial error of around 150 degrees. The "cross-track error" $\|\mathbf{\Pi}_{\eta_r}^{\perp}(\tilde{\xi})\|$ (middle) seemingly converges linearly to zero. This is because the vehicle is moving towards the path with a constant airspeed V_a. The bump in the magnitude of the normal acceleration (top) occurring around 30 sec is also seen in Fig. 9.6 and is caused by the wind. As the vehicle moves around the helix with constant airspeed, the ground speed V varies, depending on the angle of the velocity with respect to the non-zero wind. One also observes that when $\tilde{\xi}$ has converged close to zero that η is aligned with η_r (see Fig. 9.7 where the time evolution of these vectors are plotted using spherical coordinates) and that V_r converges to V. From Fig. 9.7 one also sees how η_a^d is offset from η^d to account for the wind, and how η_a converge to η_a^d, and η to η_d (and in turn, to η_r), respectively.

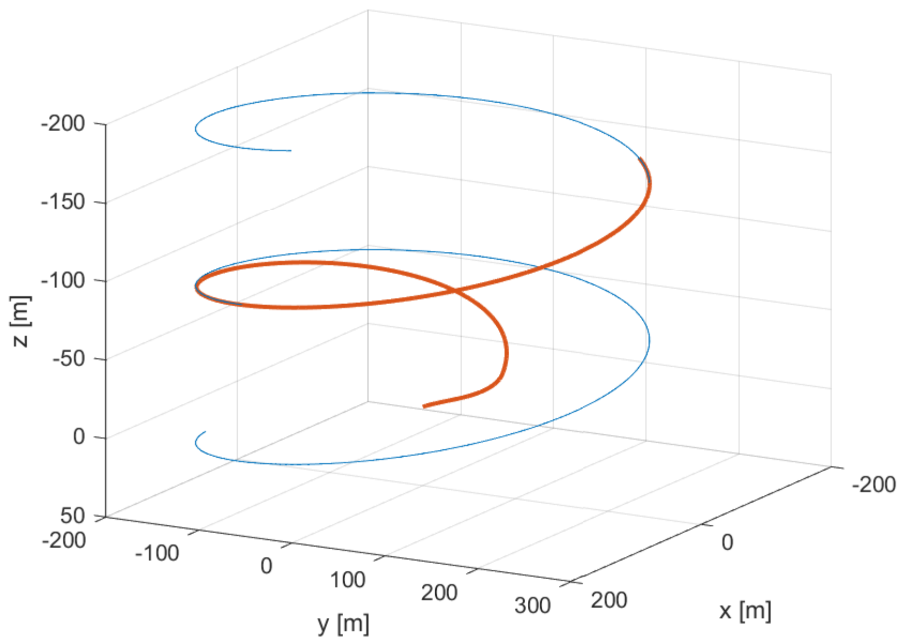


Figure 9.4: Path following of helical path. Desired path $\mathcal{P}$ in blue, and path traced out by the vehicle in red. The direction of travel is upwards.

9.8 Chapter Summary

In this chapter, we presented a class of novel path-following guidance laws with almost global stability properties for any feasible curved path in 3D. Working with the path-following error directly in the inertial frame means there is no need to construct explicit “path frames” moving along the path. Using a cascade approach, we provided conditions for choosing suitable feedback laws. Further, using the normal acceleration normal to the relative velocity as the control input makes the guidance law especially suited for fixed-wing aircraft. The simulated results demonstrate the efficacy of the proposed method.

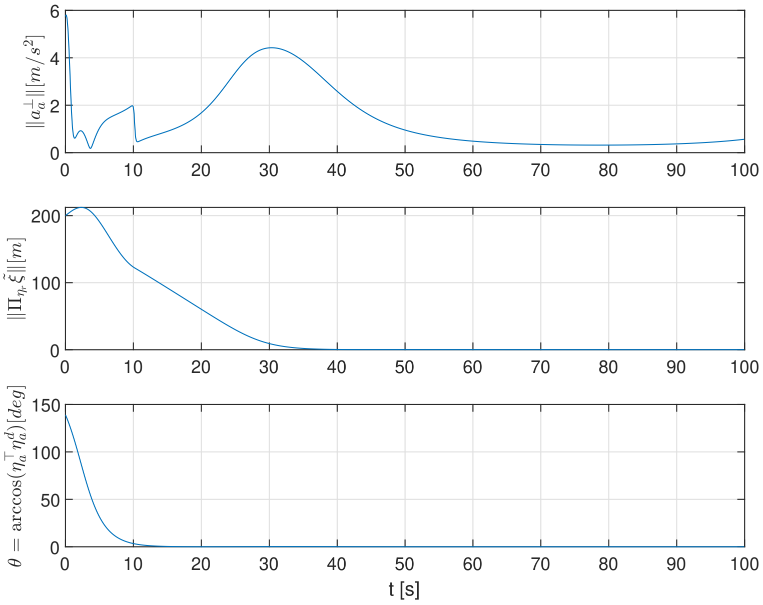


Figure 9.5: From top to bottom: the magnitude of normal acceleration (normal to η_a), the magnitude of the orthogonal component of the position error, and the angle between desired and actual relative heading.

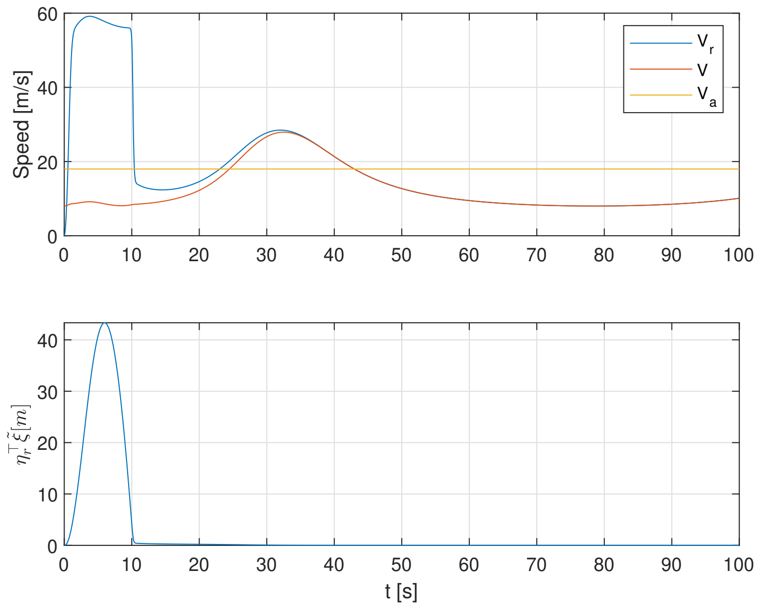


Figure 9.6: Top: V_r converges to V as the component of $\tilde{\xi}$ in the direction of η_r (bottom) goes to zero and η is aligned with η_r. V_a can be seen to be constant.

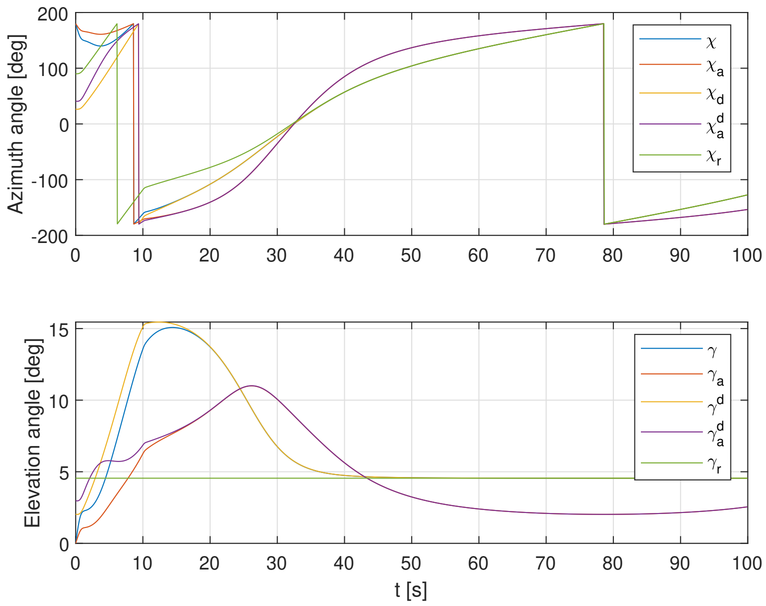


Figure 9.7: Azimuth and elevation angles. For each respective vector, the azimuth angle is $\chi_* = \text{atan2}(e_2^\top \eta_*, e_1^\top \eta_*)$, and the elevation angle is $\gamma_* = \arcsin(-e_3^\top \eta_*)$.

Part III

Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs

Chapter 10

Introduction to Part III

This chapter is based on material found in the following articles:

[31] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[32] E. Bøhn, E. M. Coates, D. Reinhardt, and T. A. Johansen. Data-efficient deep reinforcement learning for attitude control of fixed-wing UAVs: Field experiments. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

10.1 Introduction

Many challenging control problems arise during advanced operation of fixed-wing UAVs, such as aerobatic maneuvering [40], perching [66], deep-stall landing [166], recovery from loss of control [67], flying in strong wind fields [154], or performing vertical take-off and landing (VTOL) transitions between hover and forward flight [12]. Fixed-wing UAVs, as illustrated in Fig. 1.1a, have superior range and endurance when compared to multirotor UAVs. However, the control of such vehicles is challenging due to highly coupled, underactuated nonlinear dynamics, as well as uncertain aerodynamics affected by wind disturbances that make up a large fraction of the vehicle's airspeed.

A class of methods that have shown promising results for challenging control problems is deep reinforcement learning (DRL). Reinforcement learning (RL) [25, 235] is a computational approach to solving sequential decision problems under uncertainty or alternatively, learning approximate (suboptimal) solutions to stochastic optimal control problems, often through learning from interaction with the (true or simulated) environment. DRL is the combination of reinforcement learning (RL) algorithms with artificial neural networks (ANNs) as function approximators, which is the state-of-the-art approach for many complex decision-making problems with long time horizons such as game-playing [221], dexterous in-hand robotic manipulation [8], and object manipulation [220]. It can handle continuous state and action spaces, highly complex and nonlinear system dynamics, and in general, does not require a model of the system to be controlled. Furthermore, the online execution of an RL controller is often very computationally efficient. This should make DRL an enticing alternative for problems where accurate identification of the system is difficult and traditional control approaches are unable to yield sufficient control performance. Despite this potential, DRL has yet to be widely adopted for control systems and notably lacks demonstrations of control applications outside of simulations. One of the main contributing factors to this is the lack of safety guarantees and the ability to formulate operating constraints, both in the exploration and exploitation phase, which is further complicated by the data-intensive nature of DRL.

One approach to mitigate the challenges of RL for control is foregoing online exploration entirely and learning the controller exclusively from historical data with no further interaction with the system to be optimized, i.e. offline RL [151]. The latter is a radical alteration of the RL problem introducing new challenges and requiring its own set of learning algorithms. It could nevertheless be an important future tool for problems such as control of UAVs — where data collection carries a high risk, and accurate modelling is difficult. A more common approach is performing part of or the whole exploration phase in a simulation of the target system. A downside of this approach is that while RL in principle is a model-free optimization framework, the success of the transfer from the simulated environment to the real target environment is highly affected by the accuracy of the simulation model, the lack of which is referred to as the reality gap. One should therefore make a great effort in minimizing the reality gap through sim-to-real measures, which aim at robustifying the learned controller and emulating effects such as latency and measurement noise present in the real control system. For a recent survey on sim-to-real methods in the context of control and robotics, see [261].

For the sim-to-real learning approach to be useful for practical flight control applications, controllers trained in simulation need to transfer well to control the real UAV. Before attempting advanced problems like e.g. deep-stall landing, it makes sense to first attempt simpler problems and identify what factors are important for controllers to transfer well to reality.

In Part III of the thesis, we consider the application of DRL methods to low-level attitude control of fixed-wing UAVs.

10.2 Outline of Part III

First, in Chapter 11, which is based on [31], we apply the proximal policy optimization (PPO) algorithm to control the attitude and speed of a fixed-wing UAV. The controller is trained in a simulator environment developed using the OpenAI Gym framework [38] and a flight dynamics model of the Skywalker X8 flying wing. The DRL controller is evaluated in simulations and compared against PID controllers using a range of performance metrics. Next, in Chapter 12, based on [32], we demonstrate successful real-world flight experiments using a Soft Actor-Critic (SAC) algorithm. We adopt the method of exploring and learning in a simulator environment and iteratively adjust the model and simulation environment with insights from flight experiments. We apply domain randomization and other sim-to-real measures in order to reduce the reality gap. Finally, in Chapter 13, based on [62], we detail the experimental platform used for flight-testing of DRL controllers as well as other high-performance flight control algorithms tested at our lab. We summarize the challenges and lessons learned and document the architecture of our experimental platform in a best-practice manner.

Before moving on to controller design in Chapter 11, we now give an overview of related work and then end the present chapter with a brief intro to RL and the specific DRL algorithms used in this work.

10.3 Related Work

In general, the application of RL to UAV platforms has been limited compared to other robotics applications due to data collection with UAV systems carrying a significant risk of fatal damage to the aircraft. RL have been proposed as a solution to many high-level tasks for UAVs such as the higher-level path planning and guidance tasks, alongside tried and tested traditional controllers providing low-level stabilization. In the work of Gandhi et al. [83], a UAV is trained to navigate in an indoor environment by gathering a sizable dataset consisting of crashes, giving the UAV ample experience of how *not* to fly. In [101], the authors tackle the data collection problem by constructing a pseudo flight environment in which a fixed-wing UAV and the surrounding area are fitted with magnets, allowing for adjustable magnetic forces and moments in each DOF. In this way, the UAV can be propped up as one would do when teaching a baby to walk, and thereby experiment without fear of crashing in a setting more realistic than simulations.

Imanberdiyev et al. [94] [111] developed a model-based RL algorithm called TEXPLORE to efficiently plan trajectories in unknown environments subject to constraints such as battery life. In [260], the authors use MPC to generate training data for an RL controller, thereby guiding the policy search and avoiding the potentially catastrophic early phase before an effective policy is found. Their controller generalizes to avoid multiple obstacles, compared to the singular obstacle avoided by the MPC controller in training, does not require full state information like MPC does, and is computed at a fraction of the time. With the advent of DRL, it has also been used for more advanced tasks such as enabling intelligent cooperation between multiple UAVs [110], and for specific control problems such as landing [202]. RL algorithms have also been proposed for attitude control of other autonomous vehicles, including satellites [255] and underwater vehicles. Carlucho et al. [45] apply an actor-critic DRL algorithm to low-level attitude control of an autonomous underwater vehicle (AUV) and find that the derived control law transfers well from simulation to real-world experiments.

Of work addressing problems more similar in nature to the one in this part of the thesis, i.e. low-level attitude control of UAVs, one can trace the application of RL methods back to the works of Bagnell and Schneider [16] and Ng er al. [189], both focusing on helicopter UAVs. Both employed methods based on offline learning from data gathered by an experienced pilot, as opposed to the online self-learning approach proposed in this thesis. The former focuses on control of a subset of the controllable states while keeping the others fixed, whereas the latter work extends the control to all six degrees of freedom. In both cases, the controllers exhibit control performance exceeding that of the original pilot when tested on real UAVs. In [1], the latter work was further extended to include difficult aerobatic manoeuvres such as forward flips and sideways rolls, significantly improving upon the state-of-the-art. Cory and Tedrake [66] present experimental data of a fixed-wing UAV perching manoeuvre using an approximate optimal control solution. The control is calculated using a value iteration algorithm on a model obtained using nonlinear function approximators and unsteady system identification based on motion capture data. Bou-Ammar et al. [35] compared an RL algorithm using fitted value iteration (FVI) for approximation of the value function, to a non-linear controller based on feedback linearization, on their proficiency in stabilizing a quadcopter UAV after an input disturbance. They find the feedback-linearized controller to have superior performance. Recently, Koch et al. [128] applied three state-of-the-art RL algorithms to control the angular rates of a quadcopter UAV. They found PPO to perform the best of the RL algorithms, outperforming the PID controller on nearly every metric.

The literature on RL-based attitude control of UAVs is dominated by quadcopters, and most works operate exclusively in simulated environments [23, 74, 128, 153, 248]. When it comes to works presenting real-world flight experiments we have identified the following: [57, 79, 127, 135, 185, 247, 254]. Of these, only [57], their follow-up work [79], and [247] use a fixed-wing UAV design. [57] and [79] study the specific problem of controlling a perched landing, [247] fixes the aircraft in a wind-tunnel and limit themselves to controlling the pitch of the UAV, while we study the attitude control problem of an unconstrained vehicle in a 3D outdoor environment. Moreover, the data requirement of their methods is on the order of millions of samples. The other aforementioned works also require millions of samples of data, with the notable exception of [135]. Their method uses model-based RL and involves learning a forward dynamics model that is used in an MPC scheme that controls the quadrotor. While this method is very sample efficient, the MPC controller is too computationally complex to run aboard the UAV and therefore necessitates continuous communication with an external computer, while our controller can run on a fraction of the computational power embedded in the UAV. For a more general overview of the application of RL to UAVs see [14] and the related works section of [31].

Other related control methods that have been proposed for attitude control of fixed-wing UAVs include simple PID loops [19], the linear quadratic regulator (LQR) [138], adaptive dynamic programming (ADP) [75, 259, 263], and MPC [208]. ADP is similarly to RL a data-driven optimal control scheme. In the works of [75, 259, 263], the cost function is assumed to have a quadratic form and the optimal control law is derived from the Hamilton-Jacobi-Bellman (HJB) equation, which is solved numerically by approximating the cost-function using a value iteration scheme. Of note is also MPC which has inherent support for system constraints, multivariate objectives, high interpretability, and the ability to incorporate domain knowledge in the dynamics model. Motivated by these properties, the authors of this work have developed an MPC controller for simultaneous control of the UAV's attitude and airspeed, allowing the controller to consider the coupling between pitch and airspeed dynamics (note that we have previously demonstrated that RL is also capable of learning this coupled control problem [30, 31]). Flight experiments with this MPC were conducted using the same experimental platform as this work, albeit with a more powerful processing unit. See [208] for details and experimental results, as well as the discussion in [62]. A in-depth comparison between MPC and the RL controller is

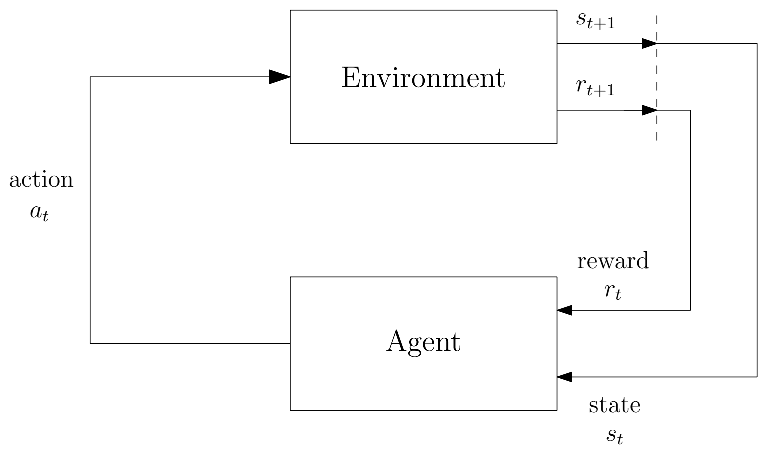


Figure 10.1: Agent-environment interaction in RL (adapted from [235]).

outside the scope of this thesis.

10.4 Reinforcement Learning

The RL optimization framework, illustrated by Fig. 10.1, consists of two main parts, an environment that is to be controlled and an agent that observes the state of the environment and selects actions to maximize the rewards it receives. The environment is typically described as a Markov decision process (MDP), which is defined by a 5-tuple of components $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$: A set of states, $\mathcal{S}$, a set of actions available to the agent, $\mathcal{A}$, a transition function $\mathcal{T}(s_t, a_t) = s_{t+1}$ which describe the evolution of the states as a function of actions and previous states, a reward function $R(s, a)$ quantifying the utility of states and accompanying actions, and finally, a discount factor $\gamma \in [0, 1)$, weighing the relative value of immediate and future rewards.

We consider the episodic finite-horizon formulation of RL. An episode is a trajectory of states and actions $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ of maximum length T with a distribution of initial states $\mathcal{S}_0$. The policy π_ϑ is a parameterized function that maps states to actions, describing the agent's behaviour (analogous to a control law in control terminology). The RL objective can then be stated as finding the parameters ϑ of the optimal policy π^*_ϑ that maximizes the return G over the distribution of the initial conditions of the episode:

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t) \tag{10.1}$$

$$\pi_\vartheta^* = \underset{\vartheta}{\operatorname{argmax}} \, \mathbb{E}_{\tau \sim \mathcal{T}(\mathcal{S}_0, \pi_\vartheta)} \left[G(\tau) \right] \tag{10.2}$$

where $\sim$ signifies that the trajectories τ are sampled from $\mathcal{T}(\mathcal{S}_0, \pi)$, i.e. the distribution of trajectories given by the transition dynamics $\mathcal{T}$, the initial state distribution $\mathcal{S}_0$, and the action-distribution of the policy π.

Further, we define the value function $V^\pi(s)$ and action-value function $Q^\pi(s, a)$ as follows:

$$V^\pi(s) = \sum_t \mathbb{E}_{\pi_\vartheta}[R(s_t, a_t) \mid s] \tag{10.3}$$

$$Q^\pi(s, a) = \sum_t \mathbb{E}_{\pi_\vartheta}[R(s_t, a_t) \mid s, a]. \tag{10.4}$$

Current state-of-the-art DRL algorithms for continuous state and action spaces, notably Deep Deterministic Policy Gradient (DDPG) [152], Trust Region Policy Optimization (TRPO) [222], PPO [223] and SAC [100], are generally policy-gradient methods, where some parameterization of the policy is iteratively optimized through estimating the gradients. They are model-free, meaning they make no attempt at estimating the state-transition function. Thus they are very general and can be applied to many problems with little effort at the cost of lower sample efficiency. These methods generally follow the actor-critic architecture, wherein the actor module, i.e. the policy, chooses actions for the agent and the critic module evaluates how good these actions are, i.e. it estimates the expected long-term reward, which reduces the variance of the gradient estimates.

Policy Gradient Methods

One class of RL methods that attempt to solve (10.2) are policy gradient algorithms. Policy gradient algorithms work by estimating the policy gradient and then applying a gradient ascent algorithm to the gradient estimate. The gradients are estimated in a Monte Carlo (MC) fashion by running the policy in the environment to obtain samples of the gradient of the policy loss $J^{PG}(\vartheta)$t [235]:[1]

[1] τ represents trajectories of the form $(s_0, a_0, s_1, a_1, s_2, a_2, \ldots, s_T)$

$$\nabla_\vartheta J(\vartheta) = \mathbb{E}_{\tau\sim\pi_\vartheta(\tau)}\left[\left(\sum_{t=0}^{T}\nabla_\vartheta \log \pi_\vartheta(a_t|s_t)\right)G(\tau)\right] \tag{10.5}$$

$$\vartheta_{\text{new}} = \vartheta_{\text{old}} + \eta\nabla_\vartheta J^{\text{SAC}}(\pi_\vartheta), \tag{10.6}$$

and (10.6) shows the gradient ascent scheme used to arrive at the optimal policy in which $\eta > 0$ is the learning rate and J^{SAC} is the SAC objective function.

In practice, these gradients are obtained with automatic differentiation software on a surrogate loss objective, whose gradients are the same as (10.5), and are then backpropagated through the ANN to update ϑ.

In this section, π is the policy network (that is, the control law) which is optimized wrt. its parameterization ϑ, in this case the ANN weights. The policy network takes the state, s, as its input, i.e. the observation vector, and outputs an action, a, consisting of the elevator, aileron and throttle setpoints. For continuous action spaces, the policy network is tasked with outputting the moments of a probability distribution, in this case, the means and variances of a multivariate Gaussian, from which actions are drawn. During training, actions are randomly sampled from this distribution to increase exploration, while the mean is taken as the action when training is completed.

Actor-Critic Methods

The central challenge in policy gradient methods lies in reducing the variance of the gradient estimates such that consistent progress towards better policies can be made. The actor-critic architecture makes a significant impact in this regard by reformulating the reward signals in terms of the *advantage*, defined as:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{10.7}$$

The advantage function (10.7) measures how good an action is compared to the other actions available in the state, such that good actions have positive rewards and bad actions have negative rewards. One thus has to be able to estimate the average reward of the state, i.e. the value function $V(s)$.[2] This is the job of the critic network, a separate ANN trained in a supervised manner to predict the value function with ground truth from the reward

[2]The value function $V^\pi(s)$ is the expected long-term reward of being in state s and then following policy π, as opposed to the $Q^\pi(s,a)$-function which focuses on the long term reward of taking a specific action in the state, and then following the policy.

values in the gathered samples. Several improvements such as generalized advantage estimate (GAE) are further employed to reduce variance of the advantage estimates.

10.4.1 The Proximal Policy Optimization Algorithm

PPO is a model-free, on-policy, actor-critic, policy-gradient method. It aims to retain the reliable performance of TRPO algorithms, which guarantee monotonic improvements by considering the Kullback–Leibler (KL) divergence of policy updates, while only using first-order optimization. PPO maximizes the surrogate objective function

$$L(\vartheta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\vartheta)\hat{A}_t, \operatorname{clip}\left(r_t(\vartheta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t \right) \right], \tag{10.8}$$

in which $\hat{A}$ and $\hat{\mathbb{E}}$ denotes the empirically obtained estimates of the advantage function and expectation, respectively, and $r_t(\vartheta)$ is the probability ratio

$$r_t(\vartheta) = \frac{\pi_\vartheta(a_t, s_t)}{\pi_{\vartheta_{old}}(a_t, s_t)}. \tag{10.9}$$

Vanilla policy gradients require samples from the policy being optimized, which after a single optimization step are no longer usable for the improved policy. For increased sample efficiency, PPO uses importance sampling to obtain the expectation of samples gathered from an old policy $\pi_{\vartheta_{old}}$ under the new policy we want to refine π_ϑ. In this way, each sample can be used for several gradient ascent steps. As the new policy is refined, the two policies will diverge, increasing the variance of the estimation, and the old policy is therefore periodically updated to match the new policy. For this approach to be valid, the state transition function must be similar between the two policies, which is ensured by clipping the probability ratio (10.9) to the region $[1-\epsilon,\ 1+\epsilon]$.[3] This also gives a first-order approach to trust region optimization: The algorithm is not too greedy in favouring actions with positive advantage and not too quick to avoid actions with a negative advantage function from a small set of samples. The minimum operator ensures that the surrogate objective function is a lower bound on the unclipped objective and eliminates increased preference for actions with negative advantage function. PPO also makes use of several actors simultaneously gathering samples with the policy to increase the sample batch size. PPO is outlined in Algorithm 1.

[3]The clip operator saturates the variable in the first argument between the values supplied by the two following arguments.

Algorithm 1: PPO

```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy $\pi_{\vartheta_{old}}$ in environment for T time steps
        Compute advantage estimates $\hat{A}_t$ for $t = 1, 2, \ldots, T$
    end
    Optimize surrogate L wrt. $\vartheta$.
    $\vartheta_{old} \leftarrow \vartheta$
end
```

10.4.2 Soft Actor Critic

SAC [100] is an actor-critic algorithm whose defining characteristic is its entropy regularization, meaning that it is jointly maximizing the expected rewards as in (10.2) and maximizing the entropy of the policy:

$$\pi^*_\vartheta = \underset{\vartheta}{\operatorname{argmax}}\, \mathbb{E}_{\tau\sim\mathcal{T}(\mathcal{S}_0,\pi_\vartheta)} \left[\sum_{t=0}^{T} \gamma^t \left(R(s_t, a_t) + \chi \mathcal{H}(\vartheta | s_t, a_t) \right) \right] \tag{10.10}$$

where $\mathcal{H}(\vartheta|s,a) = \mathbb{E}_{a\sim\pi_\vartheta(a|s)}\left[-\log \pi_\vartheta(a|s)\right]$ is the entropy, equal to the negative log probability of the action-distribution of the policy in the state in question and χ is the entropy coefficient. For brevity, we limit our discussion about SAC to the implementation of the policy and instead refer the reader to the original paper [100] for details on the objective function. The policy is implemented as follows:

$$\pi_\vartheta(s_t) = \tanh(\mu_\vartheta(s) + \sigma_\vartheta(s) \odot \xi), \;\; \xi \sim \mathcal{N}(0, I), \tag{10.11}$$

where μ_ϑ and σ_ϑ are two parameterized deterministic functions of the input, representing the mean and covariance of the output, respectively. The notation $\odot$ denotes element-wise matrix multiplication and ξ is independently sampled Gaussian noise. The entropy can therefore be controlled in a state-dependent manner through the σ_ϑ function. Finally, the output is saturated with the hyperbolic tangent function, which squashes the Gaussian's infinite support to the domain $[-1, 1]$, limiting the adverse effects of extreme noise values and giving bounded outputs.

Chapter 11

Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization

This chapter is based on the following article:

[31] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

11.1 Introduction

Contemporary autopilot systems for UAVs are far more limited in their flight envelope as compared to experienced human pilots, thereby restricting the conditions UAVs can operate in and the types of missions they can accomplish autonomously. In this chapter, we propose to use DRL to handle the nonlinear attitude control problem and to extend the flight envelope for fixed-wing UAVs. The objective is to explore the application of RL methods to low-level control of fixed-wing UAVs and produce a proof-of-concept RL controller capable of stabilizing the attitude of the UAV to a given attitude reference. To this end, we implement an OpenAI Gym environment [38] with a flight simulator tailored to the Skywalker X8 flying wing, where the RL controller is tasked with controlling the attitude (the roll and pitch angles) as well as the airspeed of the aircraft. To the best of the authors' knowledge, this is the first reported work to use DRL for attitude control of fixed-wing UAVs.

Chapter Outline

The rest of the chapter is organized as follows: First, Section 11.2 outlines the approach taken to develop the RL controller, presenting the configuration of the RL algorithm and the key design decisions taken, and finally how the controller is evaluated. In Section 11.3, we present simulation results and evaluate the controller in light of the approach described in Section 11.2 before ending with a chapter summary in Section 11.4.

11.2 Method

11.2.1 Control Objective

The objective is to control the UAV's attitude, so a natural choice of controlled variables are the roll, pitch and yaw angles. However, the yaw/heading angle of the aircraft is typically not controlled directly but through banked-turn manoeuvres. In addition, it is desirable to stay close to some nominal airspeed to ensure energy-efficient flight, avoid stalling, and maintain control surface effectiveness which is proportional to airspeed squared. The RL controller is therefore tasked with controlling the roll and pitch angles, ϕ and θ, and the airspeed V_a to desired reference values.

11.2.2 RL Algorithm

The RL controller is constructed using PPO [223], described in Section 10.4.1. PPO was chosen for several reasons: first, PPO was found to be the best performing algorithm for attitude control of quadcopters in [128], and secondly, PPO's hyperparameters are robust for a large variety of tasks, and it has high performance and low computational complexity. It is therefore the default choice of algorithm in OpenAIs projects. Our implementation is based on the OpenAI Baselines implementation [104].

11.2.3 UAV Simulation Model

To train and test the RL controller, we have implemented a simulation model of a Skywalker X8 flying wing (Fig. 1.1a) integrated with the OpenAI Gym environment [38], which enables easy integration with OpenAI Baselines. We have since made this software publicly available [27, 28].

The simulation model is based on the general aircraft equations of motion presented in Section 2.3 and Section 2.4, tailored to the Skywalker X8 and based on previous modelling efforts. Aerodynamic coefficients are taken from [93], based on wind tunnel experiments of the X8 and simple computational fluid dynamics (CFD) simulations. In addition to the linear coefficients typically used in flight mechanics [19, 233], the drag coefficient C_D contains extra terms quadratic in β and the (virtual) elevator deflection δ_e. Furthermore, the flight simulator was designed with the goal of being valid for a wide array of flight conditions and therefore includes additional nonlinear effects in the aerodynamic model. Therefore, as an attempt to extend the range of validity to a greater range of α- and β-values, lift, drag and pitch moment coefficients are made nonlinear using Newtonian flat plate theory [19, 91].

The propeller thrust T_p is given by Eq. (2.41) with parameters based on wind-tunnel experiments presented in [63]. The propeller moment is given by Eq. (2.42) where $k_\Omega = 797.1268$ and $k_Q = 1.1871 \times 10^{-6}$. These numbers are estimated based on the same experimental data used in [63]. Gyroscopic moments are assumed negligible. The throttle dynamics is given by a first-order (unity DC gain) low-pass filter with time constant $T = 0.2$.

The X8 is equipped with elevons, one control surface on each wing, that are moved collectively to produce a pitch moment and differentially to produce a roll moment. The elevon control surface dynamics are modelled by magnitude- and rate-limited second-order low-pass filters (2.50) with natural frequency $\omega_0 = 100\,\mathrm{rad\,s^{-1}}$ and damping ratio $\zeta = 1/\sqrt{2}$. The angular deflections and rates are constrained to $\pm 30\,\mathrm{deg}$ and $\pm 200\,\mathrm{deg/s}$, respectively. The aerodynamic model is defined in terms of virtual aileron and elevator deflections. The true elevon deflection angles are therefore transformed using Eq. (2.49) before evaluating the aerodynamic forces and moments.

11.2.4 Controller Architecture

In DRL, the controller policy is made up of an ANN. The input to the policy is the observation vector, and the outputs are control actions. The policy network is an extended version of the default two hidden layers, 64 nodes multilayer perceptron (MLP) policy of the OpenAI Baselines implementation: the observation vector is first processed in a convolutional layer with three filters spanning the time dimension for each component, before being fed to the default policy. This allows the policy to construct functions on the time evolution of the observation vector while scaling more favourably in parameter count with increasing observation vector size compared to a fully connected input layer.

Observation Vector

The observation vector (i.e. the input to the RL algorithm) contains information obtained directly from state feedback of states typically measured by standard sensor suites. No sensor noise is added. To promote smooth actions it also includes a moving average of previous actuator setpoints. Moreover, since the policy network is a feed-forward network with no memory, the observation vector at each time step consists of these values for several previous time steps to facilitate learning of the dynamics.

The choice of observation vector supplied to the RL controller proved to be significant for its rate of improvement during training and its final performance. We found that reducing the observation vector to only the essential components, i.e. the current airspeed and roll and pitch angles, the current angular velocities of the UAV, and the state errors, helped the RL controller improve significantly faster than other, larger observation vectors.[1] Including values for several previous time steps (five was found to be a good choice) further accelerated training, possibly making dynamics learning easier for the memoryless feed-forward policy.

The components of the observation vector are expressed in different units and also have different dynamic ranges. ANNs are known to converge faster when the input features share a common scale, such that the network does not need to learn this scaling itself. The observation vector is therefore normalized. This is accomplished with the VecNormalize class of [104], which estimates each observation component's running mean and variance and normalizes based on these estimates.

Action Space

The action space of the controller is three-dimensional, consisting of commanded virtual elevator and aileron angles as well as the throttle. Elevator and aileron commands are mapped to commanded elevon deflections using the inverse of the transformation given by (2.49).

A known issue in optimal control is that continually switching between maximum and minimum input is often optimal in the sense of maximizing the objective function, it wears unnecessarily on the actuators in practice. Since PPO samples its outputs from a Gaussian distribution during training, a high variance will generate highly fluctuating actions. This is not much of a problem in a simulator environment but could be an issue if trained online on

[1]Essential here refers to the factors' impact on performance for this specific control task. One would for instance expect α and β to be essential factors when operating in the more extreme and nonlinear regions of the state space.

a real aircraft. PPO's hyperparameters are tuned w.r.t. a symmetric action space with a small range (e.g. -1 to 1). Adhering to this design also has the benefit of increased generality, training the controller to output actions as a fraction of maximal and minimal setpoints. Therefore, the actions produced by the controller are saturated to this range and subsequently scaled to fit the actuator ranges as described in Section 11.2.3.

11.2.5 Reward Function and Training Procedure

At each time step t, the controller receives an immediate reward R_t, and it aims at developing a control law that maximizes the sum of future discounted rewards. In accordance with traditional control theory, where one usually considers the cost to be minimized rather than rewards to be maximized, the immediate reward returns to the RL controller are all negative rewards in the normalized range of -1 to 0:

$$R_t = -(R_\phi + R_\theta + R_{V_a} + R_{\delta^c}), \tag{11.1}$$

where

$$\begin{aligned}
R_\phi &= \text{sat}^{\gamma_1}\left(\frac{|\phi - \phi^d|}{\zeta_1}\right) \\
R_\theta &= \text{sat}^{\gamma_2}\left(\frac{|(\theta - \theta^d)|}{\zeta_2}\right) \\
R_{V_a} &= \text{sat}^{\gamma_3}\left(\frac{|V_a - V_a^d|}{\zeta_3}\right) \\
R_{\delta^c} &= \text{sat}^{\gamma_4}\left(\frac{\sum_{j\in[a,e,t]}\sum_{i=0}^{4}|\delta^c_{j t-i} - \delta^c_{j t-1-i}|}{\zeta_4}\right)
\end{aligned}$$

and

$$\begin{aligned}
&\zeta_1 = 3.3,\ \zeta_2 = 2.25,\ \zeta_3 = 25,\ \zeta_4 = 60 \\
&\gamma_1 = 0.3,\ \gamma_2 = 0.3,\ \ \gamma_3 = 0.3, \gamma_4 = 0.1
\end{aligned}$$

In this reward function, L_1 was chosen as the distance metric between the current and desired states (denoted with superscript d).[2] Furthermore, a cost is attached to changing the actuator setpoints to address oscillatory control behaviour. Commanded control setpoint of actuator j at time step t is denoted $\delta^c_{j_t}$, where $j \in [a, e, t]$. The importance of each component of the

[2]The L_1 distance has the advantage of punishing small errors harsher than the L_2 distance and therefore encourages eliminating small steady-state errors.

Table 11.1: Constraints and ranges for initial conditions and target setpoints used during training.

Variable	Initial Condition	Target
ϕ	$\pm 150°$	$\pm 60°$
θ	$\pm 45°$	$\pm 30°$
ψ	$\pm 60°$	-
ω	$\pm 60\,°/\mathrm{s}$	-
α	$\pm 26°$	-
β	$\pm 26°$	-
V_a	$12 - 30\,\mathrm{m/s}$	$12 - 30\,\mathrm{m/s}$

reward function is weighted through the γ factors. To balance the disparate scales of the different components, the values are divided by the variables' approximate dynamic range, represented by the ζ factors.

The reward function is one of the major ways the designer can influence and direct the behaviour of the agent. One of the more popular alternatives to L_1 norm and clipping to achieve saturated rewards are the class of exponential reward functions, and notably the Gaussian reward function as in [45]. Analyzing different choices of the reward function was not given much focus as the original choice gave satisfying results.

The PPO RL controller was initialized with the default hyperparameters in the OpenAI Baselines implementation [104] and ran with 6 parallel actors. The controller is then trained in an episodic manner to assume control of an aircraft in motion and orient it towards some new reference attitude. Although the task at hand is not truly episodic in the sense of having natural terminal states, episodic training allows one to adjust episode conditions to suit the agent's proficiency and admits greater control of the agent's exploration of the state space. The aircraft's initial state and reference setpoints are randomized in the ranges shown in Table 11.1. Episode conditions are made progressively more difficult as the controller improves, beginning close to target setpoints and in stable conditions until finally spanning the entire range of values given in Table 11.1. The chosen ranges allow the RL controller to demonstrate that it is capable of attitude control and facilitates comparison with the PID controller as it is expected to perform well in this region. According to [19], a typical sampling frequency for autopilots is 100 Hertz, and the simulator, therefore, advances 0.01 seconds at each time step. Each episode is terminated after a maximum of 2000 time steps,

Table 11.2: PID controller parameters.

Parameter	Value	Parameter	Value
k_{p_V}	0.5	k_{d_ϕ}	0.5
k_{i_V}	0.1	k_{p_θ}	-4
k_{p_ϕ}	1	k_{i_θ}	-0.75
k_{i_ϕ}	0	k_{d_θ}	-0.1

corresponding to 20 seconds of flight time, and no wind or turbulence forces are enabled during training.

11.2.6 Evaluation

Representing the state-of-the-art in model-free control, fixed-gain PID controllers for roll, pitch and airspeed were implemented to provide a baseline comparison for the RL controller:

$$\delta_t^c = -k_{p_V}(V_a - V_a^d) - k_{i_V}\int_0^t (V_a - V_a^d)d\tau \tag{11.2}$$

$$\delta_a^c = -k_{p_\phi}(\phi - \phi^d) - k_{i_\phi}\int_0^t (\phi - \phi^d)d\tau - k_{d_\phi}p \tag{11.3}$$

$$\delta_e^c = -k_{p_\theta}(\theta - \theta^d) - k_{i_\theta}\int_0^t (\theta - \theta^d)d\tau - k_{d_\theta}q \tag{11.4}$$

The throttle is used to control airspeed, while virtual aileron and elevator commands are calculated to control roll and pitch, respectively. The PID controllers were manually tuned using a trial-and-error approach until achieving acceptable transient responses and low steady-state errors for a range of initial conditions and setpoints. The wind was turned off in the simulator during tuning. The integral terms in (11.2)-(11.4) are implemented numerically using forward Euler. Controller gains are given in Table 11.2.

The same aerodynamic model used for training is also used for evaluation purposes, with the addition of disturbances in the form of wind to test generalization capabilities. The controllers are compared in four distinct wind and turbulence regions: light, moderate, severe and no turbulence. Each setting consists of a steady wind component, with randomized orientation and a magnitude of 7 m/s, 15 m/s, 23 m/s and 0 m/s respectively, and additive turbulence given by the Dryden turbulence model [240]. Note that a wind speed of 23 m/s is a substantial disturbance, especially when considering the Skywalker X8's nominal airspeed of 18 m/s. For each wind setting, 100 sets of initial conditions and target setpoints are generated, spanning the ranges shown in Table 11.1. The reference setpoints are set to 20-30 degrees and 3-4 m/s deviation from the initial state for the angle variables and airspeed, respectively. Each evaluation scenario is run for a maximum of 1500 time steps, corresponding to 15 seconds of flight time, which should be sufficient time to allow the controller to regulate to the setpoint.

The reward function is not merely measuring the proficiency of the RL controller but is also designed to facilitate learning. To compare, rank and evaluate different controllers, one needs to define additional evaluation criteria. To this end, the controllers are evaluated on the following criteria: **Success/failure**, whether the controller is successful in controlling the state to within some bound of the setpoint. The state must remain within the bounds for at least 100 consecutive time steps to be counted as a success. The bound was chosen to be $\pm 5°$ for the roll and pitch angles, and ± 2m/s for the airspeed. **Rise time**, the time it takes the controller to reduce the initial error from 90 % to 10 %. As these control scenarios are not just simple step responses and may cross these thresholds several times during the episode, the rise time is calculated from the first time it crosses the lower threshold until the first time it reaches the upper threshold. **Settling time**, the time it takes the controller to settle within the success setpoint bounds and never leave this bound again. **Overshoot**, the peak value reached on the opposing side of the setpoint wrt. the initial error, expressed as a percentage of the initial error. **Control variation**, the average change in actuator commands per second, where the average is taken over time steps and actuators. Rise time, settling time, overshoot and control variation are measured only when the episode is counted as a success. When comparing controllers, the success criterion is the most important, as it is indicative of stability as well as achieving the control objective. Secondly, low control variation is important to avoid unnecessary wear and tear on the actuators. While success or failure is a binary variable, rise time, settling time and overshoot give additional quantitative information on the average performance of the successful scenarios.

11.3 Results and Discussion

The controller was trained on a desktop computer with an i7-9700k CPU and an RTX 2070 GPU. The model converges after around two million time steps of training, which on this hardware takes about an hour. This is relatively little compared to other applications of DRL, and suggests that the RL controller has additional capacity to master more difficult tasks. Inference with the trained model takes 800 microseconds on this hardware, meaning that the RL controller can operate at the assumed autopilot sampling frequency of 100 Hertz in flight.

The RL controller generalizes well to situations and tasks not encountered during training. Even though the controller is trained with a single setpoint for each episode, Figure 11.1 shows that the controller is perfectly capable of adapting to new setpoints during flight. This result was also found by Koch et al. [128] for quadcopters. The generalization capability also holds for unmodeled wind and turbulence forces. The controller is trained with no wind estimates present in the observation vector and no wind forces enabled in the simulator. However, as Table 11.3 shows, it is still able to achieve tracking to the setpoint when steady wind and turbulence is enabled in the test environment. Table 11.3 should be read as a quantitative analysis of performance in conditions similar to normal operating conditions, while Figure 11.1 and 11.2 qualitatively shows the capabilities of the controllers on significantly more challenging tasks.

Table 11.3 shows that both controllers can achieve convergence to the target for the evaluation tasks, with neither controller clearly outperforming the other. The RL controller has an advantage over the PID controller on the success criterion and seems to be more robust to the turbulence disturbance. It achieves convergence in the attitude states in all situations, unlike the PID controller, and is also notably more successful in moderate and severe turbulence conditions. The PID controller has considerably lower control variation for the simple settings with little or no wind, but its control variation grows fast with increasing disturbance. At severe turbulence, the RL controller has the least control variation.

The two controllers perform similarly w.r.t. settling time and rise time, each having the edge in different states under various conditions, while the PID controller performs favourably when measured on overshoot. All in all, this is an encouraging result for the RL controller, as it is able to perform similarly as the established PID controller in its preferred domain, while the RL controller is expected to make its greatest contribution in the more nonlinear regions of the state space.

A comparison of the two controllers is shown in Figure 11.2 on a scenario involving fairly aggressive maneuvers, which both are able to execute. Figure 11.1 and 11.2 illustrate an interesting result; the RL controller is able to eliminate steady-state errors. While the PID controller has integral action to mitigate steady-state errors, the control law of the RL controller is only a function of the last few states and references. This might suggest that the RL controller has learned some feed-forward action, including nominal inputs in each equilibrium state, thus removing steady-state errors in most

Table 11.3: Performance metrics for the RL controller and the baseline PID controller on the evaluation scenarios. Both controllers exhibit strengths in different aspects — the best value in each circumstance is shown in bold.

		Success (%)				Rise time (s)			Settling time (s)			Overshoot (%)			Control variation
Setting	Controller	ϕ	θ	V_a	All	ϕ	θ	V_a	ϕ	θ	V_a	ϕ	θ	V_a	(s^{-1})
No turbulence	RL	100	100	**100**	**100**	**0.265**	0.661	**0.825**	**1.584**	1.663	2.798	21	24	**31**	0.517
	PID	100	100	98	98	1.344	**0.228**	0.962	2.050	**1.364**	**2.198**	**4**	**17**	35	**0.199**
Light turbulence	RL	100	100	**100**	**100**	**0.210**	0.773	**0.744**	**1.676**	1.806	2.738	28	33	**36**	0.748
	PID	100	100	99	99	1.081	**0.294**	0.863	2.057	**1.638**	**2.369**	**6**	**20**	43	**0.457**
Moderate turbulence	RL	**100**	**100**	**98**	**98**	**0.192**	1.474	0.934	**2.167**	**2.438**	4.085	54	54	74	0.913
	PID	100	93	90	87	0.793	**0.525**	**0.864**	2.764	2.563	**3.460**	**34**	**35**	**70**	**0.781**
Severe turbulence	RL	**100**	**100**	**92**	**92**	**0.166**	1.792	1.585	**2.903**	**3.280**	7.049	122	93	156	**1.083**
	PID	99	96	87	86	0.630	**0.945**	**1.343**	3.576	5.256	**5.470**	**92**	**80**	**122**	1.117

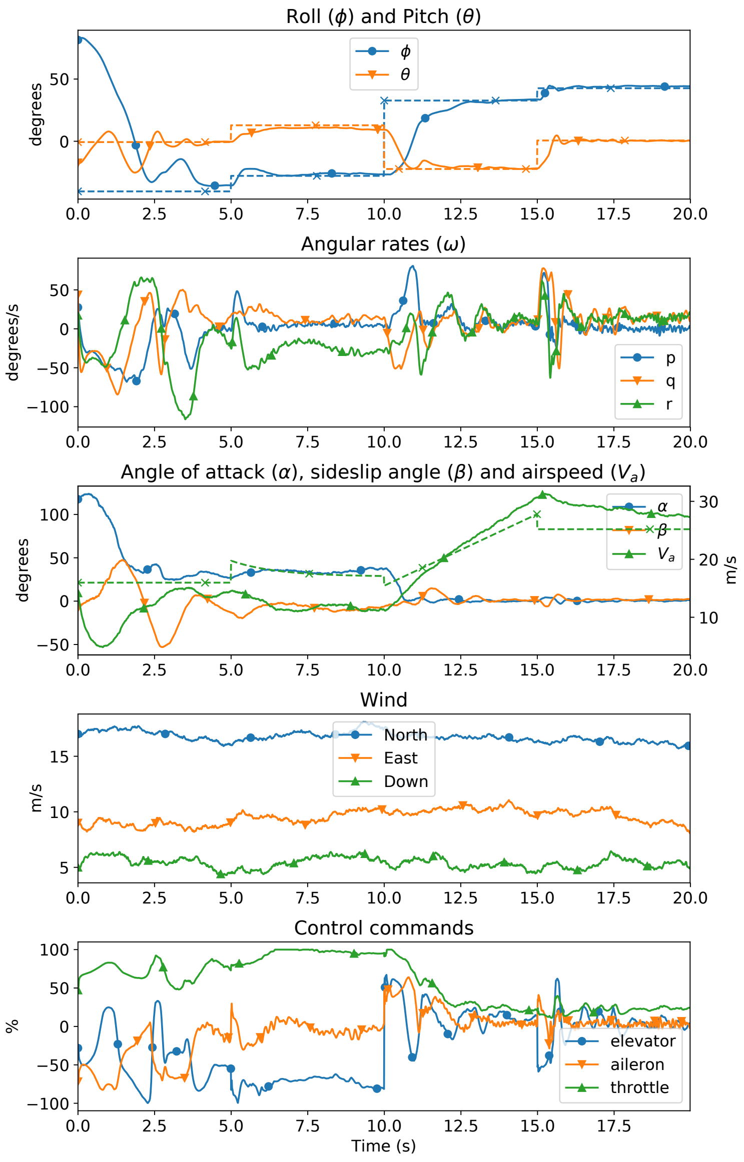


Figure 11.1: The RL controller trained episodically with a single setpoint and no wind or turbulence generalizes well to many wind conditions and continuous tracking of setpoints (shown with dashed lines marked by crosses). Here subjected to severe wind and turbulence disturbances with a magnitude of 20 m/s.

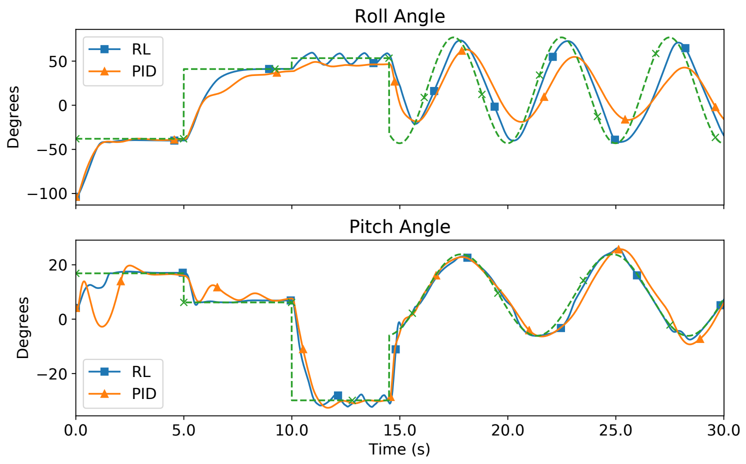


Figure 11.2: Comparison of the PID and RL controllers tasked with tracking the dashed green line.

cases. Another possibility is that steady-state errors are significantly reduced through high-gain feedback, but the low control variation shown for severe turbulence in Table 11.3 indicates that the gain is not excessive. Future work should include integral error states in the observations and evaluate the implications on training and flight performance.

11.4 Chapter Summary

In this chapter, we have proposed to use the DRL algorithm PPO to control the attitude and airspeed of a fixed-wing UAV. Both training and evaluation of the controller were carried out in simulation. The ease at which the RL controller learns to control the UAV for the tasks presented in this chapter, and its ability to generalize to turbulent wind conditions, suggests that DRL is a good candidate for nonlinear flight control design. However, a central unanswered question here is the severity of the reality gap, or in other words, how transferable the strategies learned in simulations are to real-world flight. We address this problem in the next chapter, where we target real-life flight experiments using DRL.

Chapter 12

Data-Efficient Deep Reinforcement Learning for Attitude Control of Fixed-Wing UAVs: Field Experiments

This chapter is based on the following article:

[32] E. Bøhn, E. M. Coates, D. Reinhardt, and T. A. Johansen. Data-efficient deep reinforcement learning for attitude control of fixed-wing UAVs: Field experiments. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

12.1 Introduction

In this chapter, we extend the work presented in the preceding chapter and show that DRL can successfully learn to perform attitude control of a fixed-wing UAV, requiring as little as three minutes of flight data. The proposed method is based on the SAC algorithm and improves upon the data efficiency of the existing literature by at least an order of magnitude, providing an important step towards enabling the learning of RL controllers entirely on the real UAV. We initially train our model in a simulation environment and then deploy the learned controller on the UAV in flight tests, demonstrating comparable performance to the state-of-the-art ArduPlane PID attitude controller with no further online learning required.

To the best of our knowledge, this is the first work to demonstrate through field experiments the efficacy of RL for attitude control of fixed-wing UAVs, a class of UAV design generally considered to be significantly more complex to control than the multirotor which is common in the literature.

Chapter Outline

The rest of this chapter is organized as follows: Section 12.2 describes our method in detail and presents our thoughts on what parameters are important for the learning problem. Section 12.3 presents the main experimental results, followed by a discussion and analysis part in Section 12.4. Finally, Section 12.5 concludes with a chapter summary and ideas for future research directions.

12.2 Method

The control objective of the RL controller is to control the attitude of the aircraft to the desired reference attitude. Compared to the previous chapter, the airspeed is not a controlled variable. We use standard aircraft nomenclature and coordinate systems [19], as well as a roll-pitch-yaw Euler angle parameterization of attitude. The heading/yaw angle is typically not controlled directly, but rather through banked-turn maneuvers [19]. Therefore, the natural choice of controlled states are the roll angle ϕ, and the pitch angle θ. We assume that the UAV is equipped with control surfaces such that the roll and pitch angles are controllable (an assumption that is satisfied by most UAV designs). The Skywalker X8 seen in Fig. 1.1a is used in our field experiments. It has two elevon control surfaces, one on each wing, which can be driven together to produce a pitching moment, or driven differentially to produce a rolling moment. In addition, it has a propeller that can produce a thrust force along the longitudinal axis of the UAV. In the simulation environment, a PI-controller is used to control airspeed using the propeller throttle [31].

As a general approach, we tested new ideas in the simulation environment and made extensive use of sim-to-sim experiments where we studied how the controller transferred from simulation with one set of parameters to a simulation with another set of parameters. We then tested the most promising controllers in flight experiments in the field and adjusted our approach based on the insight we gathered from the flight experiments. The simulation environment software is made open-source and is available at [29].

Table 12.1: Hyperparameters of the RL algorithm

Hyperparameter	Value	Description
γ	0.99	Discount factor of MDP
η	$3 \cdot 10^{-4}$	Learning rate
Buffer size	$5 \cdot 10^{5}$	Size of experience replay buffer
Batch size	128	Number of samples in minibatch
τ	0.005	Polyak averaging factor for target networks
χ	auto	Entropy coefficient, learned automatically
Target entropy	-2	Target entropy for the automatic η learning
Train freq	100	Update parameters after every train freq steps
Gradient steps	100	Number of gradient steps per parameter update
N goals	4	Number of imagined goals for HER per sample
HER strategy	Future	Goal selection strategy for HER

12.2.1 Reinforcement Learning Algorithm

To develop the RL controller we use the SAC algorithm and augment the collected data using Hindsight Experience Replay (HER) [7], based on the implementation [104], with the hyperparameters listed in Table 12.1. We chose the SAC algorithm because it is off-policy, and therefore has comparatively high data efficiency among RL methods, and furthermore the policy is explicitly trained to handle perturbations from the inherent randomness, which tends to yield more robust policies that transfer better than non-entropy-regularized algorithms. Note that we employ the technique of initializing the replay buffer of the algorithm with 5k data samples (corresponding to 100 seconds of flight at 50Hz), which is a common technique in RL to help the policy with the initial exploration phase. This data is entirely independent of the learning controller being trained and is obtained by uniformly sampling random actions from the set of possible actions in the simulator environment. This data could also stem from other sources such as historical data gathered by a human pilot or another controller, which might be more suitable when performing exploration exclusively in the field. Since this data is independent of the learning controller, we do not count it towards the data requirement of our method and do not include it in the learning curve graphs. During field experiments we set $\sigma_{\vartheta} = 0$ in Eq. (10.11) as this tends to yield better performance and smoother outputs [100].

12.2.2 Controller Architecture and State Design

We identified in our previous work that limiting the state vector to only the most useful information and reducing redundancy is important for the rate of convergence and to prevent the controller from learning spurious relationships. This has also been observed in other research [23]. At every time step, we measure the following information:

$$\begin{aligned} m_t = \big[&p_t, q_t, r_t, \alpha_t, \beta_t, V_{a,t}, \delta_{r,t-1}, \delta_{l,t-1}, \\ &I_{\phi,t}, I_{\theta,t}, \phi_t, \theta_t, e_{\phi,t}, e_{\theta,t}\big]^\top \end{aligned} \tag{12.1}$$

$$I_{*,t} = \gamma^I I_{*,t-1} + e_{*,t}, \ \ \gamma^I = 0.99, \ \ I_0 = 0, * \in \{\phi, \theta\} \tag{12.2}$$

where t is the time index, $\omega_t = [p_t, q_t, r_t]^\top$ is the angular velocity in the body-fixed frame, α_t is the angle-of-attack, β_t is the sideslip-angle, $V_{a,t}$ is the airspeed, $\delta_{\{r,l\},t-1}$ represent the previous output of the RL controller, in this case the commanded deflection angles of the right and left elevons, $e_{*,t} = *_t - *_{r,t}$ is the state tracking error where subscript r denotes the state reference, I_* is the integrator of the state error and γ^I is the integrator decay rate. The integration decay scheme follows [254], and facilitates boundedness of the integrator state. Lastly, because ANNs are known to converge faster given normalized inputs, the measurements are normalized using running estimates of mean and variance for each component before being fed to the controller.

Due to unmeasured effects such as turbulence and the sim-to-real measures described in Section 12.2.5, the attitude control problem is partially observable. Furthermore, to enable the controller to adapt to the dynamics of the field experiments, we wish to enhance the controller with the capability of inferring the dynamics around the current state. A common approach to achieve this effect is to use a recurrent neural network (RNN) [197, 247]. However, we found that using a one-dimensional convolution over the time dimension as the input layer yielded similar control performance, and therefore prefer it since it is significantly less complex than the RNN. We therefore include the h last measurements (12.1) in the state vector, where $\hat{m}_t$ indicates a noisy measurement to be defined in Section 12.2.5:

$$s_t = [\hat{m}_t, \hat{m}_{t-1}, \dots, \hat{m}_{t-h}]^\top \tag{12.3}$$

$$[\delta_{r,t}, \delta_{l,t}]^\top = \pi_\vartheta(s_t) + [\delta_{r,\text{trim}}, \delta_{l,\text{trim}}]^\top \tag{12.4}$$

such that the total size of the state vector is $|s_t| = |m_t| \cdot h$. The convolutional input layer scales favorably in number of learned parameters compared to

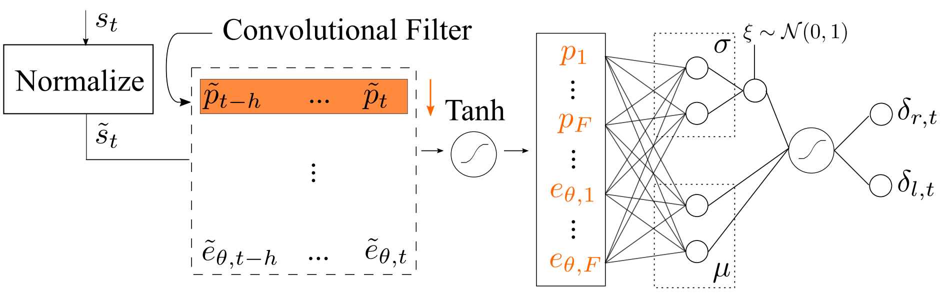


Figure 12.1: Architecture of the RL controller, superscript $\sim$ signifies normalized states.

a fully-connected (FC) layer: it scales linearly in $|m_t|$ as opposed to multiplicative for the FC layer, and it is constant for h. The convolutional layers' output size is $F \cdot |m_t|$ where F is the number of learned convolutional filters, and each filter has size h. The memory capacity of the state vector can therefore be increased as required to give sufficient history to infer the dynamics, with only a slight increase in the number of parameters. Through a small grid search using rate of learning and asymptotic performance as metrics we found $F = 8$ and $h = 10$ to work well. The complete RL controller architecture is shown in Fig. 12.1.

The output of the RL controller is the commanded states of the controlled system's actuators relative to the trim-point (12.4). The nominal elevon deflection angles $\delta_{r,\text{trim}} = \delta_{l,\text{trim}} = 0.045$ are calculated using a standard trim routine for horizontal, wings-level flight based on the model in Section 12.2.4 [19]. The target UAV for the field experiments, the Skywalker X8, has elevon actuators and we therefore chose to have the controller output the desired deflection angles of these directly, in order to provide RL with as direct control as possible. This choice is fairly arbitrary, however, and experiments showed that outputting virtual elevator and aileron angles (the sum and difference, respectively, of the elevon angles defined by (2.57)-(2.58)) instead yield similar performance.

12.2.3 Reward and Objective Design

We found sparse rewards to yield better results than shaped rewards, both in terms of rate of learning and in terms of asymptotic performance. A sparse reward is one that is nonzero only for some subset of the state space. It has the benefit that it is easier to formulate than hand-crafted shaped rewards, and would therefore be more transferable to other UAVs with fewer adjustments necessary. The reward is formulated as follows:

$$R(s_t, a_t) = \lambda^{\phi} B(e_{\phi,t}) + \lambda^{\theta} B(e_{\theta,t}) + \lambda^{\dot{\phi}} B(\dot{\phi}_t) + \lambda^{\dot{\theta}} B(\dot{\theta}_t) \tag{12.5}$$

$$B(\cdot) = \begin{cases} 1 & \text{if } |\cdot| \leq \cdot^{b} \\ 0 & \text{otherwise} \end{cases} \tag{12.6}$$

$$e^{b}_{\phi_t} = 3^{\circ},\ \ e^{b}_{\theta_t} = 3^{\circ},\ \ \dot{\phi}^{b}_{t} = 4.3^{\circ}/s,\ \ \dot{\theta}^{b}_{t} = 4.3^{\circ}/s \tag{12.7}$$

$$\lambda^{\phi} = 0.5,\ \ \lambda^{\theta} = 0.5,\ \ \lambda^{\dot{\phi}} = 0.167,\ \ \lambda^{\dot{\theta}} = 0.167 \tag{12.8}$$

where superscript b refers to the goal-bound on the variable and the λs are weighting factors. This reward ensures that the controller tracks the setpoints with accuracy as specified by the bound, while the rewards on the derivatives of the controlled states discourage high rates. Our method is not very sensitive to the size of the bound, but generally larger bounds accelerate learning at the expense of tracking accuracy.

When transferring from a simulator environment to the field, it is important to consider how the actuation system impacts the effects of actions. That is, while high-gain bang-bang control may be an optimal strategy in the simulator, frequently changing the setpoints of the actuators introduces considerable wear due to the high currents generated as a result of the switching. In our previous work [31] (and indeed in other works [128]) this problem is addressed through a term in the reward that discourages changing the setpoints. We now take a different approach to this problem, using the conditioning for action policy smoothness (CAPS) method [184]:

$$J^{\text{TS}}(\pi_{\vartheta}) = ||\pi_{\vartheta}(s_t) - \pi_{\vartheta}(s_{t+1})||_2 \tag{12.9}$$

$$J^{\text{SS}}(\pi_{\vartheta}) = ||\pi_{\vartheta}(s_t) - \pi_{\vartheta}(\hat{s}_t)||_2, \qquad \hat{s}_t \sim \mathcal{N}(s_t, 0.01) \tag{12.10}$$

This method adds two regularization terms to the loss, a temporal smoothness term (12.9) and a spatial smoothness term (12.10). As the authors demonstrate, this method is more successful in generating controllers that yield smooth control signals compared to the action reward-term approach. Additionally, removing the action term from the reward simplifies the problem of learning the action-value function since the reward now contains fewer disparate parts, thereby accelerating learning. Instead, the gradient ascent scheme calculating the parameter updates is conditioned towards policies that are smooth in the output. Finally, we add a regularization term on the pre-activation $\pi^{\text{PA}}_{\vartheta}$ (that is, before applying the hyperbolic tangent in (10.11)) of the output:

$$J^{\mathrm{PA}}(\pi_\vartheta) = ||\pi_\vartheta^{\mathrm{PA}}(s_t)||_2 \tag{12.11}$$

This helps in reducing the gain of the controller, especially for small errors, as it essentially tells the controller that it needs to have a strong benefit to move the actuators away from the trim-point. Additionally, we find it accelerates learning as the controller is biased towards non-aggressive control, which in conjunction with HER means the controller quickly discovers how to achieve the sparse stabilizing objective. Thus, the objective we optimize is defined as:

$$\begin{aligned} J(\pi_\vartheta) = J^{\mathrm{SAC}}(\pi_\vartheta) + \lambda^{\mathrm{TS}} J^{\mathrm{TS}}(\pi_\vartheta) + \lambda^{\mathrm{SS}} J^{\mathrm{SS}}(\pi_\vartheta) +& \\ \lambda^{\mathrm{PA}} J^{\mathrm{PA}}(\pi_\vartheta)& \end{aligned} \tag{12.12}$$

$$\lambda^{\mathrm{TS}} = 5 \cdot 10^{-2},\ \ \lambda^{\mathrm{SS}} = 10^{-1},\ \ \lambda^{\mathrm{PA}} = 10^{-4} \tag{12.13}$$

12.2.4 UAV Model

The simulated environment is based on the same model as described in Section 11.2.3, but updated based on recent modelling efforts. An estimate of the inertia matrix is provided in [92] based on bifilar pendulum experiments, and complementary CFD results presented in [252].

We collected a short data series to assess the quality of the dynamic model. To excite the system dynamics, we used the actuator signals from the baseline attitude controller and perturbed them with chirp signals before mapping them to the elevon deflections. The start and end frequencies of the chirp signals were 8 Hz and 12 Hz, respectively. A dynamic mode analysis of the model indicates that this is the dominant frequency spectrum of the X8. The signal duration was 15 seconds and we used a peak-to-peak amplitude of 20 degrees.

The aerodynamic coefficients that are calculated based on recorded sensor data and the inertia matrix of the vehicle are shown in Fig. 12.2. Following [19], the coefficient subscript L, D, Y, l, m, n denotes lift, drag, side force, roll moment, pitch moment and yaw moment, respectively. These results show that despite the modeling efforts, there are still significant discrepancies between the predicted and measured data, particularly in the pitching moment coefficient, C_m.

12.2.5 Sim-to-Real Measures

The main sim-to-real measure employed in the method is domain randomization. As shown in Section 12.2.4, there is a significant reality gap, and

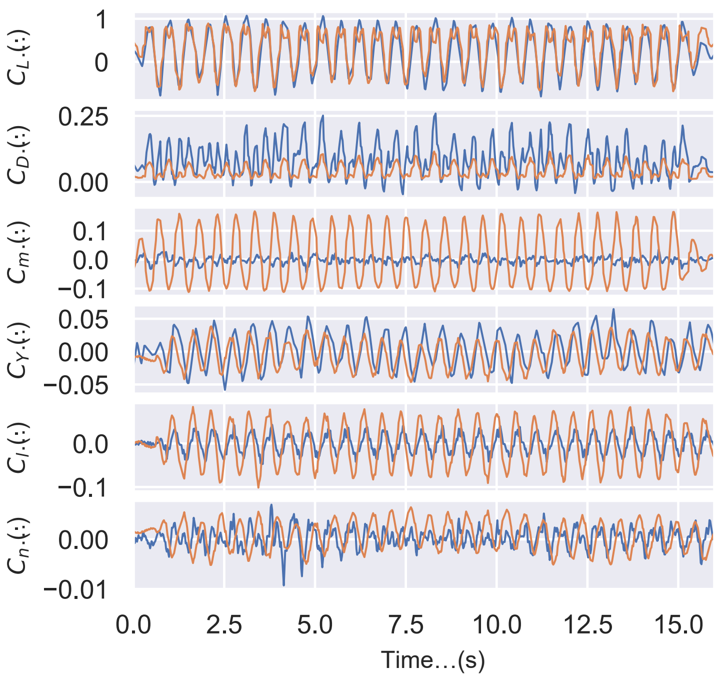


Figure 12.2: The aerodynamic coefficients of the UAV in a longitudinal (top three) and a lateral (bottom three) chirp signal test sequence for elevator and aileron, respectively, based on IMU data (blue) and model prediction (orange).

as such we want to avoid the RL controller overfitting to the simulation environment. The intuition behind the domain randomization technique is that learning over a distribution of possible UAV models should robustify the controller. To this end, we assess the uncertainty in the estimate of every parameter of the UAV model and use this uncertainty to construct a probability distribution over its range of probable values (see [29] for details). The coefficients of the rate-dependent terms of the UAV model have larger sampling ranges since these are not estimated based on wind tunnel data but rather on uncertain CFD simulations [93]. The sampled values are also clipped as indicated to avoid extreme unrealistic values. At the start of every episode, we sample a value for each parameter from its distribution, together constituting one realization of the UAV model.

The UAV sensor suite is subject to noise in its measurements. To model these, we first estimated the real hardware's noise characteristics, then we emulated this in the simulator environment. We model the measurement noise as an Ornstein-Uhlenbeck (OU) process (12.14), which in addition to white noise gives rise to effects like measurement drift:

$$\hat{m}_t = m_t + w_t,\ \ w_t \sim OU(\mu_m, \sigma_m, \theta_m),\ \ \mu_m = 0,\ \ \theta_m = 1$$
$$\sigma_m = 0.005 \cdot [1.5, 1.5, 1.5, 1, 1, 15, 0, 0, 0, 0, 1, 1, 0, 0]^\top \qquad (12.14)$$

where $\mu_m, \sigma_m, \theta_m$ are the mean, variance, and rate of mean-reversion parameters of the measurement noise. Note that we do not add noise to the error and integrator states, as these are already affected by the noise in the measurement of the state that is used to calculate the error, while the previous output of the controller is by nature free of noise. The sensor suite has an update rate of 50Hz, and we therefore chose this as the control frequency as well. In the simulation environment we add exponentially distributed noise on top of the fixed control frequency in order to simulate sensor timing-variability:

$$\Delta_t = \Delta_0 + z_t,\ \ z_t \sim \mathrm{Exp}(\kappa),\ \ \kappa \sim \mathcal{U}(250, 1000) \qquad (12.15)$$

where Δ_t is the simulation step size at step t, $\Delta_0 = 0.02\,\mathrm{s}$ is the base control frequency, and z_t is exponentially distributed noise whose parameter κ is drawn uniformly at the start of every episode.

Although not strictly sim-to-real measures, the adjustments to the RL objective described in Section 12.2.3 in the form of CAPS and the pre-activation term also serve to improve the transferability from simulation to reality, as they encourage less aggressive lower-gain control. Another major effect present in the field is atmospheric disturbances such as wind and turbulence. We model turbulence with the Dryden turbulence model [19], and a steady wind component whose direction and magnitude between $0\,\mathrm{ms}^{-1}$ and $15\,\mathrm{ms}^{-1}$ is sampled at the start of each episode. The last effect we found was highly impactful for successful transfer was the actuation delay, i.e. the time it takes before the output of the controller is applied to the system, a result which was also found in [247]. The simulator contains a constant actuation delay of 100 ms, while we believe this is a significant overestimation of the delay of the real system, we motivate this choice in Section 12.4.2.

12.2.6 Simulator Episode Design

The standard design of episodes for UAV control in the literature seems to be short episodes with a single constant desired attitude [23, 128]. We found that having constant setpoints accelerates learning, however, the operation of the UAV in the field typically sees the navigation system continuously

Table 12.2: Initial conditions for the episodes are uniformly sampled from the indicated ranges.

Variable	Range	Unit	Variable	Range	Unit
ϕ	-40, 40	degrees	ϕ_r	-60, 60	degrees
θ	-15, 15	degrees	θ_r	-25, 20	degrees
V_a	13, 26	m/s	α	-8, 8	degrees
ω	-60, 60	degrees	β	-10, 10	degrees
$\delta_{r,l}$	-30, 30	degrees			

update the desired attitude. To align these considerations, we employ fairly long episodes of length 900 steps (18 s) where setpoints are kept constant but resampled every 150 steps. To ensure sufficient diversity of the state trajectories and transitions used to update the parameters of the RL controller, we sample initial conditions as shown in Table 12.2. Considering that the main objective of the simulation environment is to ready the controller for the field, we sample initial conditions mostly from the linear region of the model, as this is where the UAV model is assumed to have the most validity. Note that while the range of initial conditions is somewhat limited, there is nothing stopping the controller from exploring the full state space. Furthermore, since the initial states of the actuators are also randomized the sampled initial conditions could cause instability, such that the controller must learn to recover.

12.2.7 Experimental Platform

Our custom avionics stack is based on the low-level control architecture developed at the NTNU UAV-lab [62]. It consists of a Cube Orange flight controller running the (industry standard) ArduPlane open-source autopilot [13], and a Raspberry Pi 4 running the DUNE Uniform Navigation Environment [199]. During experiments, the total flight weight of the Skywalker X8 is 3.8 kg.

The RL controller is implemented as a DUNE task in C++ with the ANN implemented in TensorFlow. Sensor data and state estimates from ArduPlane are sent through a serial connection to the Raspberry Pi, providing all necessary data for the RL controller. The ANN controller output is converted to PWM duty cycle and sent to the elevon servos using a PCA9685 servo driver, interfaced through I2C from the Raspberry Pi. A PWM multiplexer supports switching between the RL controller output and ArduPlane.

This enables the pilot to always take control during testing, either through manual controls, or through ArduPlane's standard autopilot. This switching mechanism enables us to safely engage the RL controller in flight, while takeoff and landing are performed by the pilot operating our tried and tested avionics stack [264]. The experimental platform is discussed in more detail in Chapter 13.

12.3 Experimental Results

This section presents the main experimental results, collected during two days of flight experiments at Agdenes Airfield, Breivika, Norway in September 2021. During the first day, we enjoyed calm weather and perfect flight conditions, with a mean wind speed (as estimated by ArduPlane's Kalman Filter) of less than $4\,\mathrm{m\,s^{-1}}$. The second day of flight tests, however, presented challenging weather conditions, with frequent gusts and a mean wind speed of approximately $12.5\,\mathrm{m\,s^{-1}}$ (70% of the Skywalker X8's cruise speed of $18\,\mathrm{m\,s^{-1}}$).

We present three types of data, differing mainly by how roll and pitch angle references are provided:

1. References are given by the pilot, mimicking ArduPlane's fly-by-wire A (FBWA) mode (Section 12.3.1).
2. References are provided by ArduPlane's guidance system, which is set to track a rectangular waypoint pattern (Section 12.3.2).
3. References are set by a predefined, automated series of steps (Section 12.3.3). Similar maneuvers are also performed with an implementation of the ArduPlane PID attitude controller, with the response compared to that of the RL controller.

In contrast to the training phase, where the throttle actuator used to control airspeed is operated by a PI controller (see [31] for details), the throttle is either under manual control by the pilot (FBWA) or controlled by ArduPlane (auto/steps). In figures presenting flight results, the dashed orange line corresponds to state reference, while in the elevon plots, the blue and orange lines correspond to the right and left elevon, respectively, with minimum and maximum deflections of $-30°$ and $+30°$.

12.3.1 FBWA Mode

Fig. 12.3 shows an excerpt from the flight experiments where a human pilot decides the desired attitude of the UAV. The RL controller is able to closely track the desired attitude even for the most difficult and aggressive maneuvers, while producing smooth outputs for the actuators. We do however note a consistent steady-state error. Towards the end of the maneuver, we observe that the roll response is non-symmetric, that is, rolling to the left (towards negative roll angles) is slower than rolling in the opposite direction. We investigate and discuss this matter, as well as the steady-state error, in Section 12.4.

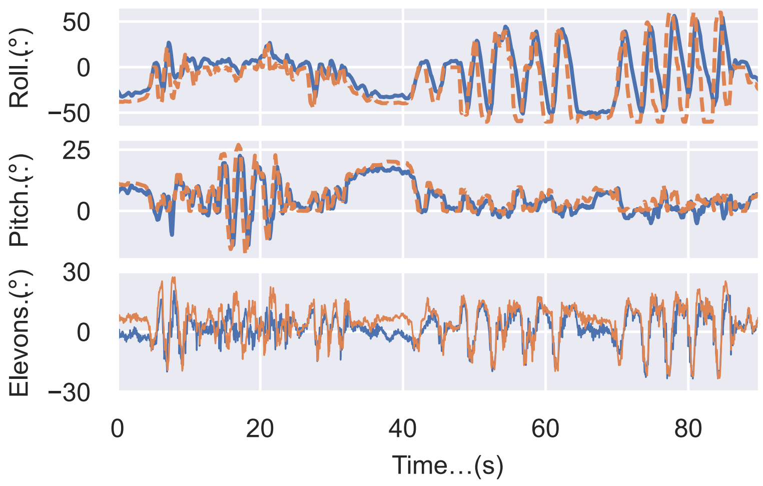


Figure 12.3: FBWA mode using references (dashed orange line) from the pilot, showing the attitude states and right (blue) and left (orange) elevon signals.

12.3.2 Auto mode

Fig. 12.4 shows the results for the RL controller operating with references provided from the ArduPlane guidance system, set to track a square waypoint pattern. Before tracking the square, the UAV loiters in a circular pattern for a while. Despite some steady-state offset, especially for the roll angle error, the UAV successfully completes the specified mission. This is because the outer-loop guidance controller can compensate for this error, and still achieve convergence when faced with disturbances such as wind and offset in inner-loop control. This is similar to how a pilot supplying manual references would offset the references to keep the intended course.

During turns, a certain altitude drop is seen from the right part of Fig. 12.4. This is caused by the aggressive turn radius accompanied by drops in the pitch angle. This effect can be reduced by tuning the guidance system to be less aggressive (e.g. by increasing the turn radius) or reducing the maximum allowable roll angle setpoint, which is set to be 55 degrees. A

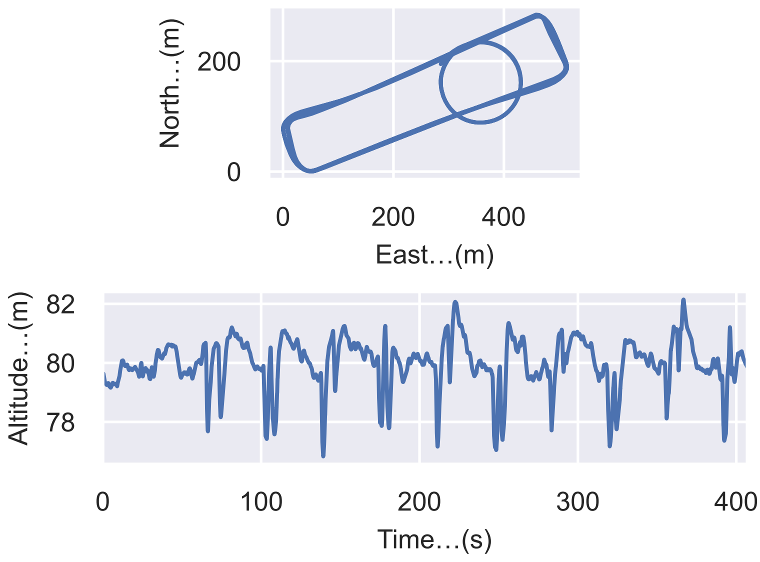


Figure 12.4: Position plot showing how the RL controller is able to take references from ArduPlane's guidance system in Auto mode, and effectively follow prescribed paths. First, a loiter, then a square waypoint pattern.

similar drop in pitch angle is also seen when using the default ArduPlane attitude controller.

12.3.3 Step sequences and Comparison with ArduPlane PIDs

Step responses for the RL controller, as well as the ArduPlane PID controller, are displayed in Fig. 12.5. The RL controller shows comparable transient performance to that of ArduPlane, the main difference being the steady-state error of the RL controller. Additionally, the pitch response of the PID controller is slightly more aggressive. However, this could be changed by tuning the controller.

The control signals generated by the RL controller are relatively smooth and well-behaved but include some high-frequency components not seen in the PID response. Apart from that, the control input looks qualitatively similar, with similar magnitude. For a quantitative comparison we employ the smoothness metric defined in [184] which jointly considers the amplitudes and frequencies of the control signals:

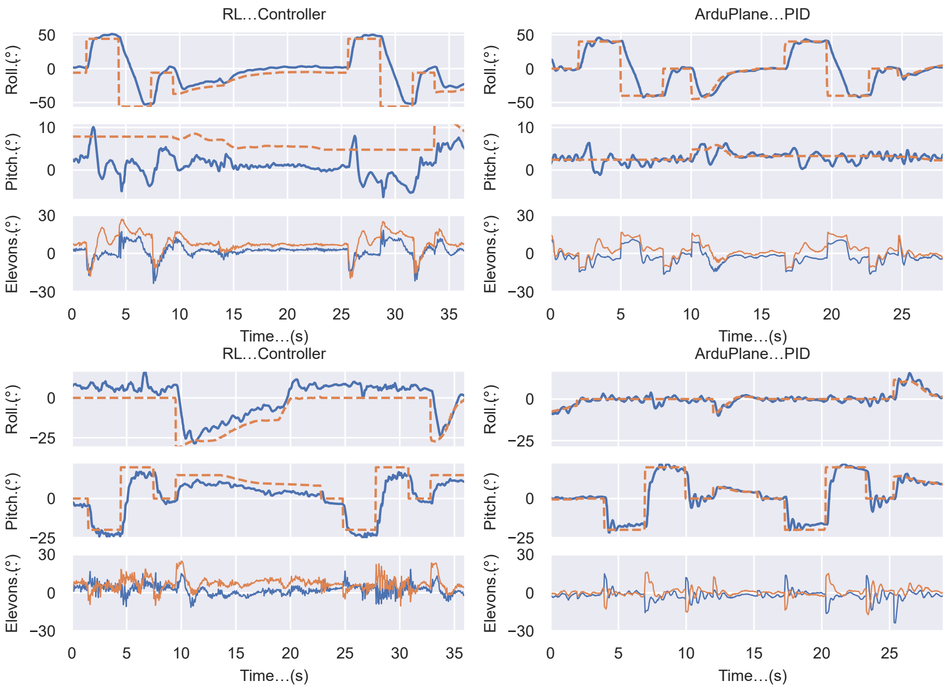


Figure 12.5: Comparison between the RL controller and the ArduPlane PID controller for steps in the roll (top) and pitch (bottom) references (dashed orange line).

$$Sm = \frac{2}{nf_s}\sum_{i=1}^{n} M_i f_i \tag{12.16}$$

where M_i is the amplitude of the i'th frequency component f_i, and f_s is the sampling frequency. On this metric the PID measures at $6.20 \cdot 10^{-4}, 6.30 \cdot 10^{-4}$ for the roll and pitch maneuvers in Fig. 12.5, respectively, while RL measures 2% and 44% higher at $6.30 \cdot 10^{-4}, 9.05 \cdot 10^{-4}$. This metric shows that RL has comparable smoothness in its output with the PID controller, but also indicates the higher frequency components of the RL controller's output in Fig. 12.5. It is not clear why there is such a discrepancy between the two maneuvers for the RL controller, but this data is as mentioned subjected to considerable turbulence and wind, and could therefore be caused by transient gusts.

While the former results were gathered on a calm day with virtually no wind, these maneuvers are executed in harsh wind conditions on day two. The UAV also suffered structural damage (not while under RL control) after collecting the PID data, before redoing the experiments with RL. The vehicle had to be repaired with a new wing and duct tape, causing a change in the UAV's trim point. Thus, the presented results demonstrate the RL controller's robustness towards model mismatch and varying wind conditions, including heavy gusts.

To achieve a fair comparison, the ArduPlane PID is implemented in the same software stack and ran with the same hardware architecture as the RL controller (described in Section 12.2.7). In particular, this means that any increased signal latency introduced in our setup does not affect the comparison.

12.4 Discussion

The experimental results of Section 12.3 show that the RL controller performs well compared to a state-of-the-art open-source autopilot, and is robust to disturbances caused by harsh wind conditions. The control performance of the RL controller across the various flight modes speaks to its ability to generalize further than just the maneuvers encountered during training. In particular, no guidance controller was present during training.

Despite the promising results, there is room for improvement. In this section, we further discuss how performance can be improved, the iterative development process, and training, and we perform a linear analysis to gain further insight into the behavior of the RL controller.

As noted in Section 12.3.1, the roll response of the RL controller is non-symmetric, meaning that rolling towards the left wing is slower than rolling to the right. This is supported by the pilot's qualitative assessment during flight. We found that this effect was caused by an overestimation of the simulated propeller torque, which was found to be less prominent on the physical UAV than expected, presumably due to the mechanical mounting of the propeller. This causes a bias in the RL controller, which has learned to counteract the propeller torque. To remedy this, we trained a new RL controller in a simulation model without propeller torque, which was briefly tested in the field to verify our hypothesis. The new controller exhibited a more symmetric roll response, as expected.

12.4.1 Steady-State Errors

We experimented with several techniques in order to address the steady-state error of the RL controller observed in flight experiments: pure integrator (no decay), higher decay factor (e.g. 0.999), having integration separate from the ANN controller with learned integration gains, shaped rewards,

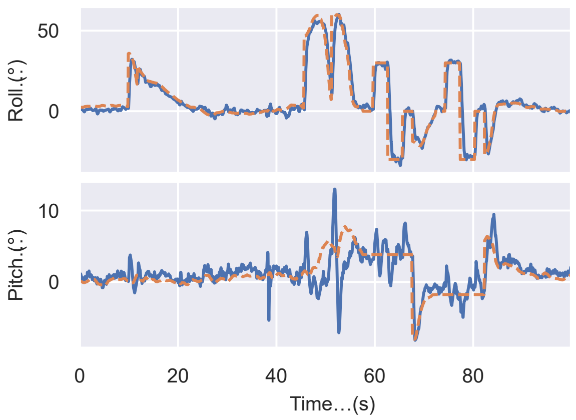


Figure 12.6: Response of the RL controller where the steady-state error has been estimated and references (dashed orange line) adjusted to compensate.

and training with input disturbances. Whereas some of these measures reduced the steady-state error to some degree, none were successful in entirely eliminating it.

We note that there is no consistent steady-state error in the simulator in the same way we observe in field experiments, i.e. consistently over or under the reference with a consistent magnitude. The controller has learned to use integral action to reduce steady-state error from disturbances in the simulator, but not in a way that transfers to the field. This could be because the controller is overfitting to the simulator, thus the larger tracking errors in the field combined with the hyperbolic tangent saturating functions of the ANN causes the integrator states' effect on the output to saturate prematurely.

An effective way to address this problem is to estimate the steady-state error and then add the estimated value to the references provided to the RL controller, as was done in the flight experiment shown in Fig. 12.6. As can be seen, this simple technique can fully compensate for the steady-state error and may also be automated using an integrator in an outer loop to estimate the steady-state error [165]. Furthermore, there are compelling arguments for not having integral action in the inner-loop attitude controller, as adding integral action to the controller necessarily reduces the phase margins and the achievable bandwidth [19].

12.4.2 Oscillations: Illustration of Iterative Development

Initial field experiments were characterized by excessive oscillations in the attitude response of the UAV, especially in pitch, necessitating halving the RL controller's outputs in order to keep the aircraft airborne. These oscilla-

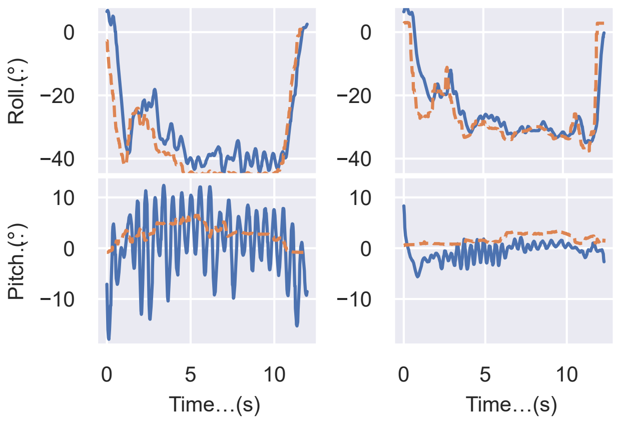


Figure 12.7: Oscillatory attitude response of initial flight experiments, for one controller trained with original (left) and one with reduced (right) C_{m_q}.

tions were not present in the simulator, as such, we suspected that this was (at least in part) caused by the simulator overestimating the natural damping present in the aircraft. Therefore, we reduced the C_{m_q} (pitch damping) parameter by a factor of 10. While this reduced the oscillations somewhat, there were still significant oscillations in the response, see Fig. 12.7.

We estimated a typical actuation latency for the system of $10\,\mathrm{ms}^{-1}$. In sim-to-sim experiments, where we raised the latency of the control system during the exploitation phase of a controller trained with $10\,\mathrm{ms}^{-1}$ latency, we observed similar oscillatory responses as in the flight experiments and noted its relationship with increasing latency. We then trained an RL controller where the latency was set to $100\,\mathrm{ms}^{-1}$ during the learning phase. This controller trained with higher latency almost entirely eliminated the oscillations to the levels shown in the field experiment figures. Favoring robust controller design, we increased the base latency of the simulation environment to $100\,\mathrm{ms}^{-1}$, even though we believe that this is a significant overestimation of the true latency of the real system.

12.4.3 Linear Analysis

In order to better understand how the RL attitude controller operates, we analyze its sensitivity to the input variables. In Fig. 12.8 we have plotted the open-loop response of the controller as a function of a single perturbed input. The rest of the state vector is kept constant at the steady-flight value, i.e. zero for all variables except the airspeed V_a which is set to the cruising speed of $18\,\mathrm{m\,s}^{-1}$, and the angle-of-attack α and pitch angle θ, which are kept at the trim values necessary to generate lift for level flight, while the input values for previous time steps are kept constant in the time dimension. To be able to compare the results with ArduPlane, we translate the elevon outputs into virtual elevator and aileron commands using the inverse of (2.57)-(2.58). The figures and tables are presented in terms of these virtual commands, which also have a more intuitive and straightforward effect on the roll and pitch angles.

The saturating effect of the hyperbolic tangent nonlinearity on the RL controller is distinctly present in the responses. This is a desired effect as we know that any input should have a bounded effect on the output, which gives robustness towards possible measurement errors or misalignment of the dynamics of the simulation and real environments. The controller makes use of all its inputs, with the previous outputs of the controller having the most significance for the current output (the typical values for most states in Fig. 12.8 are close to the level-flight value in the center and will thus use a limited range of the response curve, while the previous output of the controller frequently employs the full range). This makes sense as the controller is conditioned towards smooth outputs, as described in Section 12.2.3, which means that a reasonable initial guess of any action is to be similar to the previous action. Moreover, the fact that the previous output (left and right elevons) do not have a symmetric effect on the subsequent output (there is no mechanism enforcing symmetry in the learning controller) could be a motivating factor to instead employ (virtual) elevator and aileron as outputs of the RL controller.

Since the control authority of the actuators increases with airspeed, we investigated if the RL controller has learned to scale its outputs depending on airspeed, an effect that is included in the ArduPlane controller. With no further documentation (due to space constraints), we state that this is not the case. However, it has learned to bias the response, essentially shifting the curves in Fig. 12.8 up, as airspeed increases in order to compensate for the change in trim-point with airspeed.

To estimate sensitivities wrt. an input we take a linear approximation of its response curve by using the slope of the tangent line at level-flight conditions. The result is shown in Table 12.3, and compared to the ArduPlane PID controller gains (see Appendix 2.6.1). The RL controller is noticeably more aggressive in the pitch error, while simultaneously introducing more damping through the angular velocity component q. This is evident in Fig. 12.5 where the RL controller exhibits less oscillation in the pitch response. The estimated gains for the integrator states in Table 12.3 are not representative of the response curves for these states, as the response curve exhibits cubic characteristics with a small opposing region around the level-flight value. Thus, for these states, a linear approximation over a larger region would be more descriptive. Overall, the gains of the RL controller are similar to those of the PID controller, which increases the trust in the RL controller. On the other hand, the dynamic aspect caused by integral states and data from previous time steps increases the complexity of the analysis and thus limits the conclusions that can be drawn from it.

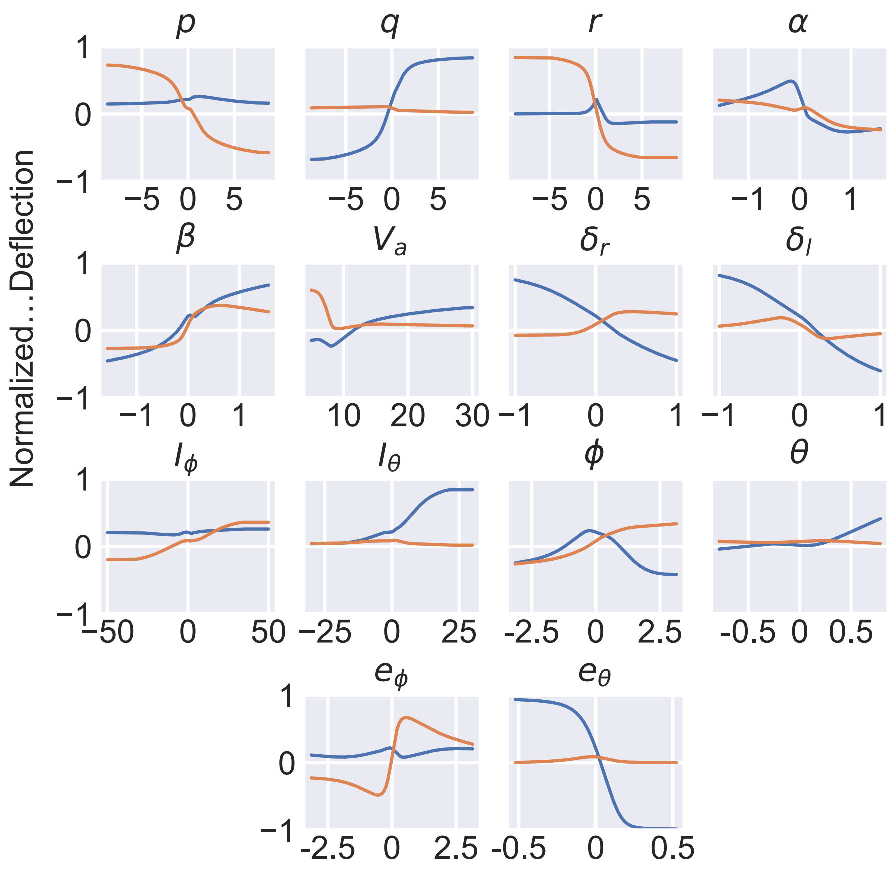


Figure 12.8: Open-loop level-flight response of the RL controller when perturbing one input at a time. The x-axis is in the units of the corresponding state. The lines are elevon outputs mapped to aileron (orange) and elevator (blue).

12.4.4 Data Requirements

In this section, we attempt to quantify the data efficiency of our method. We start by presenting the learning phase of the controller in the simulation environment (Fig. 12.9), demonstrating that our method produces

Table 12.3: Linearly approximated gains at level-flight, where each input is perturbed in isolation.

Controller	$\frac{\partial \delta_a}{\partial e_\phi}$	$\frac{\partial \delta_e}{\partial e_\theta}$	$\frac{\partial \delta_a}{\partial I_\phi}$	$\frac{\partial \delta_e}{\partial I_\theta}$	$\frac{\partial \delta_a}{\partial p}$	$\frac{\partial \delta_e}{\partial q}$
RL	1.268	-3.320	-0.005	0.006	-0.008	0.223
PID	1.630	-1.081	0.052	-0.052	-0.024	0.031

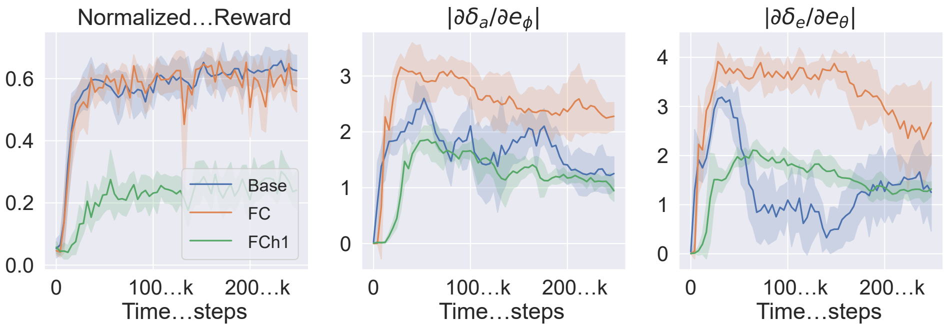


Figure 12.9: The learning phase of the proposed RL controller, showing normalized mean episode reward and error-proportional gains. The solid line represents a rolling average mean value while the shaded region represents one standard deviation over three randomly seeded controllers. Base refers to the method as presented in Section 12.2, the FC version replaces the convolutional layer with an FC layer, and the h1 version has no history in the input.

proficient controllers with a number of data samples on the order of 10s of thousands, and then demonstrate that the learning controllers are flight-worthy in the field after experiencing only three minutes of real-time flight data (Fig. 12.10). It is difficult to compare this result directly to the existing literature, for reasons outlined in the literature review of Section 12.1, that is, no other reported work study the full attitude control problem of fixed-wing UAVs using RL. Other works study either a limited version of the attitude control problem, or consider other aircraft designs (e.g. quadcopters) whose dynamics are more linear and controllable than the dynamics of a fixed-wing UAV. Nonetheless, works in the existing literature report a data requirement on the order of 100s of thousands or millions of data samples, meaning our result presents a significant improvement in terms of data efficiency. Further, we would argue that this improvement in data efficiency is significant because it suggests that learning the controller entirely in the real world (i.e. no simulation required) is possible. Collecting data for a few minutes of flight seems reasonably achievable through some form of safe ex-

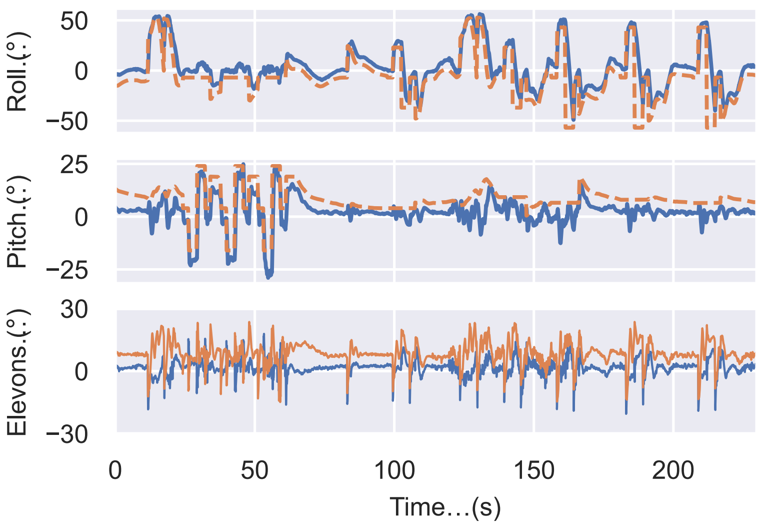


Figure 12.10: An RL controller that has trained for only 3 minutes of real-time flight in the simulation environment is airworthy, able to track references (dashed orange line) reasonably well.

ploration or guided learning where, for instance, the learning controller is monitored by a pilot or other proficient controller that can assume control should dangerous situations arise.

The evolution of the learning phase for the RL controller as a function of time steps is shown in Fig. 12.9. For every version in Fig. 12.9, we train three controllers each with a different initial random seed and average the results over the controllers. The rewards are normalized so that 1 corresponds to attaining the maximum reward as defined in (12.5) at every step (although this is not physically achievable) and 0 corresponds to obtaining no rewards at all. Base refers to the RL controller as presented in Section 12.2 that was used in the field experiments. To assess the contribution of the convolutional input layer, we trained one version where the input layer is replaced with an FC layer, and further to test the importance of the history of states in the state vector we train one model with an FC input layer and with $h = 1$ (labeled FCh1).

The base version learns fast, reaching convergent performance after around 40k time steps. This corresponds to about 13 minutes of real-time flight or 16 minutes of wall-clock training time when trained entirely on an i7-9700 Intel CPU with a single data-generating-agent in the simulator[1]. We veri-

[1]Note that the wall-clock training time can essentially be made arbitrarily short with parallelization of more agents and more powerful computing hardware

fied in the field that the RL controller is flightworthy long before this point: The controller shown in Figure 12.10 exhibits decent and stable performance (although unable to follow the most aggressive maneuvers) after only 10k time steps of training in the simulator (corresponding to three minutes of real-time flight). We conjecture that the data efficiency of our method limits overfitting to the simulation environment and therefore transfers better to the field, although this statement requires further evidence such as a bench of learning controllers with varying time in simulation to verify.

We find that our training method is stable in the sense that the performance differences between controllers with different seeds are small, within a few percent. The FCh1 version without state history is never able to learn to consistently stabilize the UAV at the desired attitude in the time frames we considered. The FC version with state history on the other hand achieves comparable rewards to the base version, showing the importance of history in the RL input state. The base version reaches peak performance slightly faster than the FC version, and further its proportional gains are considerably lower. This is also evidenced by the smoothness metric (12.16) for which the base version scores 50% lower than the FC version. The lower gains and the smoothness metric indicate that the convolutional input layer provides a superior ability to predict the system response and thus provide smoother responses in attitude and control signals, while the FC version is more reactive and oscillatory.

12.5 Chapter Summary

This chapter has presented a data-efficient method for learning attitude controllers for fixed wing UAVs using RL. The learning controller is able to operate directly on the nonlinear dynamics and therefore could extend the flight envelope and capabilities of autopilots. The high data efficiency of the presented method facilitates transfer to the control of the real UAV by limiting overfitting to the simulated model. We demonstrate that the learned controller has comparable performance to the existing state-of-the-art ArduPlane PID autopilot and is capable of tracking prescribed paths from a guidance system while generating smooth actuation signals and attitude responses. Key factors behind the success of the method were robustifying the controller through increasing its phase margins by learning with significant actuation delay and diversifying the simulated dynamics, as well as incentivizing non-aggressive control through sparse rewards and additional objective terms enforcing temporal and spatial smoothness in the controller outputs.

Chapter 13

Experimental Platform for Evaluation of Advanced Low-Level Control Algorithms

This chapter is based on the following article:

[62] E. M. Coates, D. Reinhardt, K. Gryte, and T. A. 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)*, 2022.

13.1 Introduction

Inner-loop control algorithms in state-of-the-art autopilots for fixed-wing UAVs are typically designed using linear control theory to operate in relatively conservative flight envelopes. In this work, we seek to extend the flight envelopes of fixed-wing UAVs to allow for more aggressive manoeuvring and operation in a wider range of weather conditions. As a result of this, we have successfully flight-tested several advanced nonlinear algorithms for attitude control of fixed-wing UAVs, including those presented in the preceding chapters, namely DRL and geometric attitude control (GAC) but also nonlinear model predictive control (NMPC) [210]. Each of these algorithms demands different capabilities in the onboard avionics stack. In particular, NMPC is very computationally demanding, as a nonlinear optimization program needs to be solved at every controller update. DRL is more computationally efficient but requires access to some framework for artificial neural networks, which is not provided by the standard autopilots. Therefore, we have established a common hardware platform, system architecture, and operating procedures for flight experiments of low-level nonlinear control algorithms for fixed-wing UAVs. The flexible embedded platform, briefly introduced in Section 12.2.7, is capable of running computationally demanding low-level controllers that require direct actuator control. For safe operation and rapid development cycles, this platform can be deployed in tandem with well-tested standard autopilots. In this chapter, we summarize this work with a focus on the challenges and lessons learned and document our experimental platform in a best-practice manner.

There does not seem to be any documented standard solution or best practice for computationally demanding fixed-wing UAV control architectures that require low-level access to the actuators. Motivated by this, and based on experience with extensive field testing of a wide range of nonlinear control algorithms, the main contributions of this chapter are:

- We describe novel experimental results from flight testing of several advanced nonlinear control algorithms for attitude control of fixed-wing UAVs while articulating the key challenges and lessons learned.

- A set of constructive guidelines on how to deploy an experimental platform that is well-suited for the evaluation of control algorithms that require a lot of computational resources and direct access to the actuators. Our approach also allows for switching between the experimental algorithm and the standard autopilot, allowing safe experimental testing at an early stage.

In summary, this lowers the threshold for other researchers and engineers to employ new low-level control algorithms for fixed-wing UAVs. The individual components are off-the-shelf and thus readily available. This lowers the threshold for other researchers and engineers to employ new low-level control algorithms for fixed-wing UAVs. Case studies from outdoor field experiments are provided to demonstrate the efficacy of our research platform.

Chapter Outline

The rest of this chapter is structured as follows: In Section 13.3 we describe the control algorithms that have been successfully implemented and tested using our experimental platform. The system architecture is presented in Section 13.4, and our test procedure, for both ground testing and flight experiments, is described in Section 13.5. In Section 13.6, we summarize our results, and in Section 13.7 we discuss the lessons learned before giving some concluding remarks in Section 13.8.

13.2 Related work

We classify algorithms as either *high-level* (guidance/outer-loop) controllers or *low-level* (inner-loop) controllers. Whereas the high-level algorithms are typically implemented on a single-board computer, transmitting references to standard low-level control loops running on some commercial or open-source autopilot (e.g ArduPilot [13] or PX4 [174]), the low-level algorithms need direct access to the actuators. A comprehensive review of different control architectures is not in the scope of this chapter. Although we mention some notable works that include experimental verification of high-level controllers, our focus is on low-level control and the computing platforms and system architectures used.

The low-level algorithms can be further classified concerning their requirements to the embedded computing platform as either: (a) *lightweight*, or (b) *computationally intensive*. Lightweight algorithms are easily integrated into open-source autopilots and can be run directly on hardware platforms such as the Pixhawk or CubePilot series of autopilots. For instance, in [190], the low-level control framework is based on explicit MPC using linearized models and implemented directly on the resource-constrained onboard avionics. The unified guidance and low-level control architecture in [118] is implemented directly on a mRo PixRacer with a 180Mhz ARM Cortex M4 processor by modifying the existing PX4 middleware.

Among the more computationally intensive algorithms, the NMPC scheme in [17] first poses a feasibility problem to generate dynamically feasible paths from an initial guess found via a Rapidly-Exploring Random Tree (RRT) algorithm combined with spline smoothing. Then, a time-varying LQR is used to follow the nominal trajectories calculated by the planner in a receding-horizon manner. Their experiments are conducted in a controlled lab environment with a motion capture system and desktop computer for nonlinear optimization. The desired actuator signals are transmitted to the vehicle through radio communication. The authors in [230] identify second-order models of the autopilot-controlled low-level dynamics which is used in their MPC guidance algorithm. The computing platform is an Intel UP board running a Robot Operating System (ROS) node. In [121] MPC controllers for trajectory-tracking of both fixed-wing and rotary-wing UAVs are presented, with a special focus on ROS integration. ROS is also used in [257], where an optimal path-following controller for windy conditions transmits roll and pitch angle references to low-level controllers running in PX4 on a Pixhawk.

Looking wider, into the realm of multirotor UAVs, [188] presents a controller for unified trajectory optimization and tracking with a hexacopter

UAV. They explicitly aim at improving performance by giving the controller direct actuator access. The optimal controller is implemented on a stationary Intel Core i7 processor that sends the velocity commands of the rotors to the flight control unit. State estimation is done with an optical motion capture system. The work in [258] aims at showing that current optimal control solvers have become fast enough to handle the computational burden that is coming with highly dynamic robotics applications such as the low-level control of multirotors for which they provide experimental results.

DRL algorithms are computationally efficient during online execution, but often require more flexibility than is typically provided by standard flight controllers. Some specialized solutions exist, e.g. in [198], a low-level RL-based controller for multirotors is validated experimentally using a PixRacer flight control board. In [135], a model-based RL algorithm for low-level control of a Quadrotor is validated using the open-source Crazyflie 2.0 quadrotor. pulse-width modulation (PWM) commands are calculated on a ROS server running on the ground and sent to the UAV using radio. The Neuroflight neural network-based flight control firmware is presented in [127], where experimental validation of a low-level RL controller for a quadcopter is carried out using a 216MHz ARM Cortex-M7 microcontroller.

13.3 Control algorithms: Overview

In this section, we describe the main control algorithms evaluated using our experimental platform, namely NMPC [210], DRL [32], GAC [58], as well as a PID benchmark implementation based on the ArduPilot [13] fixed-wing attitude controller. The capabilities needed to effectively run these algorithms online define the requirements for the system architecture presented in Section 13.4. This section also provides the necessary background for the experimental results of Section 13.6.

An in-depth discussion of the experimental controllers is out of scope of this work, and more details can be found in each of the respective references [32, 58, 210]. A comparison of the most significant features of the controllers is given in Tab. 13.1.

A significant part of our controllers is, to some extent, relying on accurate models of the UAV. We also depend on dynamic models in our simulators, discussed in Section 13.5. Our dynamic models are based on previous and ongoing modelling efforts, in particular [63, 92, 93].

Table 13.1: Features of the compared controllers.

	GAC	DRL	NMPC
Demand for comp. resources	Low	Low	High
Needs dynamic model	Low	Medium/High	High
Optimality wrt. cost/reward	Low	Medium	High
Constraint satisfaction	No	No	Yes
Interpretable performance	Yes	No	Yes
Open software available	Not needed	Yes	Yes
Available stability proofs	High	Low	Medium

13.3.1 PID Benchmark

As a benchmark, we use the widely adopted ArduPilot [13] open-source autopilot and compare our methods to the ArduPlane PID attitude controller for fixed-wing UAVs, given in Section 2.6.1. To get a fair comparison, we implemented a version of this in the same system as our algorithms. This means that all algorithms run on the same hardware, in the same software environment, and with the same communication latencies.

13.3.2 Nonlinear Model Predictive Control (NMPC)

NMPC allows us to explicitly encode the flight envelope in the controller design as nonlinear constraints in an optimal control problem (OCP) that can be solved regularly to obtain an optimal control input trajectory. The OCP over a prediction horizon T usually has the form

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

where the cost consists of l, denoting a stabilizing stage cost that is often in quadratic form, and a penalty for the slack variables $\mathbf{s}$ that are included for constraint relaxation to guarantee the feasibility of the approximating quadratic program (QP). The constraints include the initial condition, denoted by $\mathbf{x}_0$, dynamic constraints in form of the continuous model, denoted by the vector ordinary differential equation (ODE) defined by $\mathbf{f}$, and inequality constraints to reflect operational and actuator limits, denoted by $\mathbf{h}$. Optimal performance with respect to a defined cost function and prediction in addition to constraint satisfaction by use of the actuation system in an integrated multiple-input multiple-output (MIMO) fashion are traits that are hard to achieve with the existing autopilot software such as ArduPlane. The drawback however is its increased computational complexity and requirement for a dynamic model of the UAV, [212], which demands powerful embedded computing platforms and a considerable engineering effort in system identification [211]. The field of MPC is quite mature and conditions for performance and stability guarantees exist on a theoretical level but may be hard to formally verify for a particular system [90].

13.3.3 Deep Reinforcement Learning (DRL)

In addition to NMPC, we looked at DRL, a set of data-driven methods for approximate optimal control, where the controller is implemented as an ANN [31]. These methods can operate on and optimize control performance for the full nonlinear dynamics in a model-free manner (at least in theory), their online operation is generally highly computationally efficient, and can exhibit (nearly) arbitrarily complex behavior. A downside of the DRL methods is the lack of interpretability of the deep neural network; stability, robustness, and constraint satisfaction properties are not guaranteed. In addition, DRL requires a lot of training data, a challenge that is further complicated for flight control applications by the high inherent risk associated with data collection using a suboptimal controller. A common approach is to instead train the controller in a simulator environment, which motivates the need for a good model. To deal with model mismatch when transferring the trained controller to the field, "Sim2Real" measures such as domain randomization [32] are typically applied.

13.3.4 Geometric Attitude Control (GAC)

A third approach we considered is to apply nonlinear control methods based on Lyapunov stability theory, in particular GAC. Such methods require only a fraction of the computational resources required by NMPC, and stability and robustness guarantees can be given under some assumptions for the particular system under study, even with limited knowledge of system models. However, constraint handling is difficult to address and optimality is not addressed. Traditionally, the orientation/attitude of aircraft in 3D space is parametrized using Euler angles, which have singularities for certain orientations, which in itself contributes to a more limited flight envelope. In our work, we employ a global, nonsingular reduced attitude representation on the two-sphere. In addition to avoiding singularities, this enables more efficient, shortest path (geodesic) rotation maneuvers, [61].

13.4 System Architecture

In this section, we describe our experimental platform, which has evolved over several years as a result of a wide range of research activities carried out at the NTNU UAV-lab. An alternative system architecture is described in [264], 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 to satisfy the following requirements:

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.
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.
6. A 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. 13.1. We proceed by describing each main element of the system architecture. The following discussion concerns fixed-wing UAVs, given that we focus our research on this type of UAV. Note, however, that it is straightforward to translate the hardware architecture and outlined test procedures to other UAV morphologies. The only requirement is access to the actuators through PWM signals and a fallback controller, which is usually available on off-the-shelf platforms.

13.4.1 UAV Platform

Our platform is built around a Skywalker X8 airframe, depicted in Fig. 1.1a. 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,

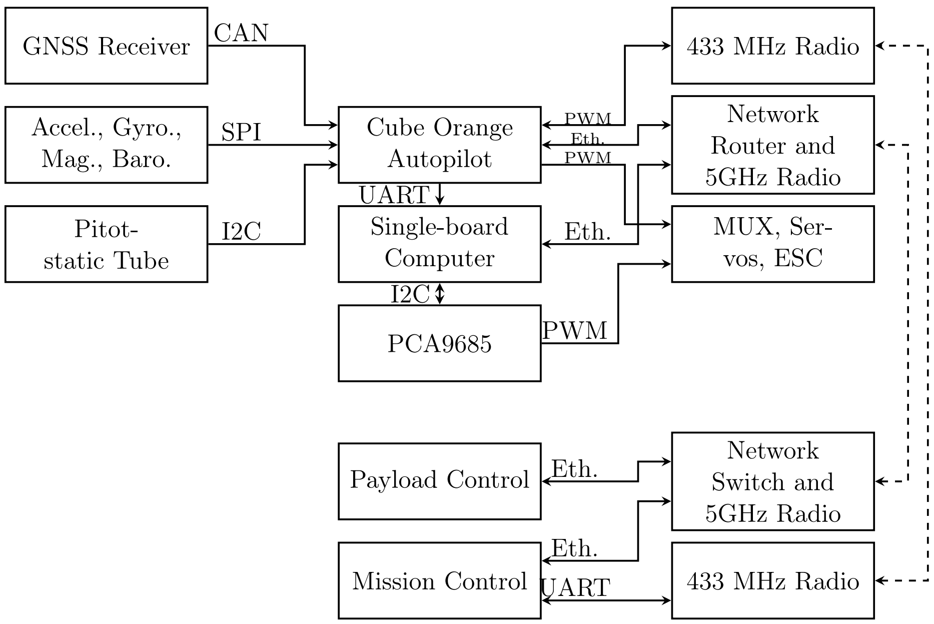


Figure 13.1: Hardware configuration of the experimental platform.

or collectively, to produce a pitch acceleration. The control signals consist of PWM signals to the two servo motors that actuate the elevons, and the throttle signal (also PWM) to a consumer-grade electronic speed controller (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 [13]. The sensor suite consists of triple redundant inertial measurement units (IMUs) with magnetometers, pressure sensors for altitude and airspeed (pitot-static tube), and a global navigation satellite system (GNSS) receiver.

13.4.2 Payload Computer

Alongside the Cube Orange, we use the Khadas Vim3[2] single-board computer (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.

[1]https://cubepilot.org/

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

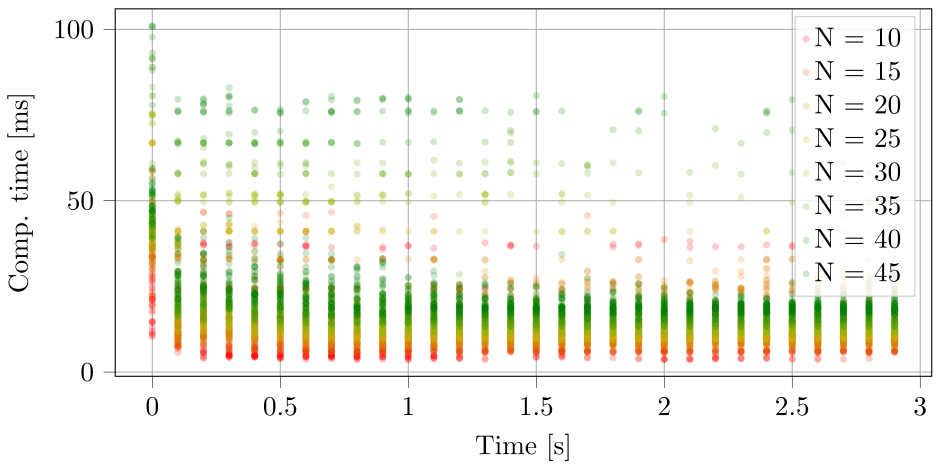


Figure 13.2: NMPC benchmarking results for the employed SBC for different prediction horizons N in a direct multiple-shooting scheme with 0.1 s shooting interval.

On the SBC we run the DUNE Uniform Navigation Environment. DUNE is part of the LSTS toolchain [199] developed at the Underwater Systems and Technology Laboratory (LSTS), University of Porto. DUNE allows us to (similarily to ROS) write different tasks that run independently from each other on separate threads or processes while exchanging data using a message bus mechanism.

The NMPC is implemented using acados [244], 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 Khadas Vim3 are depicted in Fig. 13.2, which show the closed-loop runtime of the solver to find solutions to the 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. Therefore, we chose an update period of 50 ms in the experiments, which led to satisfactory performance. More details can be found in [212].

The DRL controller is implemented as a DUNE task in C++ with the ANN implemented in TensorFlow. Benchmarking tests show that the controller can 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.

13.4.3 State Estimates

State estimates, including estimates of the local wind velocity, are provided by ArduPilot's extended Kalman filter (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 amounts 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 were not powerful enough to handle the high data throughput over the serial link.

13.4.4 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 13.1. 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 [134]. 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.

13.4.5 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.

When switching between different controllers, we reset integral terms to zero to avoid potential stability or performance issues. In ArduPlane this is done every time a switch happens between "MANUAL" mode and stabilized modes. For our custom controllers implemented in DUNE, we do this through dedicated parameters.

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 [114].

13.4.6 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 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 similar 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".

13.4.7 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 [264] for details).

We use the ArduPilot-compatible ground control software Mission Planner running on a dedicated 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.

13.4.8 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 in a repeatable manner. 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 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.

13.5 Test procedures

This section describes 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.

13.5.1 Python Simulator

As an initial 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 algorithms' robustness to modeling errors. Also, initial tuning guidelines are established during this stage. Our Python-based DRL test bench is publicly available online[3]. This is the same simulator that is also used for training of the DRL algorithms, using our OpenAI Gym wrapper[4].

13.5.2 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 setpoints, *as if the UAV was*

[3]`https://github.com/eivindeb/pyfly`

[4]`https://github.com/eivindeb/fixed-wing-gym`

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 our controllers.

13.5.3 Lab Testing

We conduct system integration tests on the physical hardware at the lab, checking all communication channels and verifying 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. Moreover, the magnitude of the deflection is an indicator of the controller response.

13.5.4 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 experiments, the team communicates using 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 pattern of waypoints.

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.

13.6 Experimental results

As a result of this work, we have been able to perform a series of successful outdoor flight experiments to evaluate the algorithms described in Section 13.3. A detailed description of the specific experiments and the results obtained will appear in separate manuscripts. See [32] and [210] for DRL and NMPC results, respectively.

In this section, we demonstrate the efficacy of our experimental platform by presenting some of our results, with a special focus on the switching mechanism. In particular, we show initial results for GAC, based on [58], and look at one of our earliest attempts at closed-loop flight using DRL.

13.6.1 Geometric Attitude Control (GAC)

See Fig. 13.3 for an early attempt during the tuning process of GAC. Initially, after takeoff, the pilot uses ArduPlane's fly-by-wire A (FBWA) mode

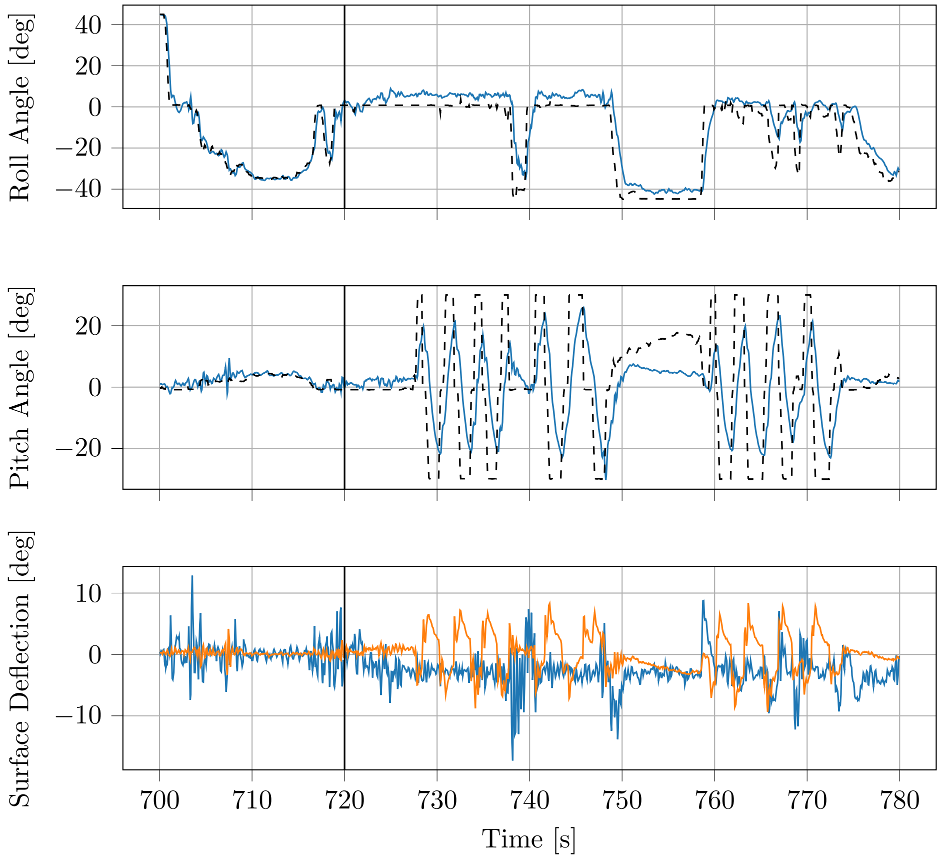


Figure 13.3: Experimental results: Initial switch to GAC. The plots show the tracking response for roll and pitch references (dashed, black). The bottom subplot depicts the virtual deflections of the aileron (blue) and the elevator (orange).

to control the UAV's roll and pitch angles using manual stick input. At 720 s (marked by the vertical line), the pilot switches the MUX to engage closed-loop operation of the GAC. During this initial test, integral action was disabled, explaining the increased offset visible after the switch. The switch between the PWM outputs from the Cube and those from the SBC is seamless, and any potential communication delays in the hardware architecture do not seem to have a significant effect on the controller.

Fig. 13.4 shows the performance of the GAC controller after further tuning. The steady-state offset has been removed, and the test shows good performance through repeated step responses in both roll and pitch channels. The parallel pipeline, including a solid backup system and in-air switching between the two, enabled a safe and comfortable tuning process. A more

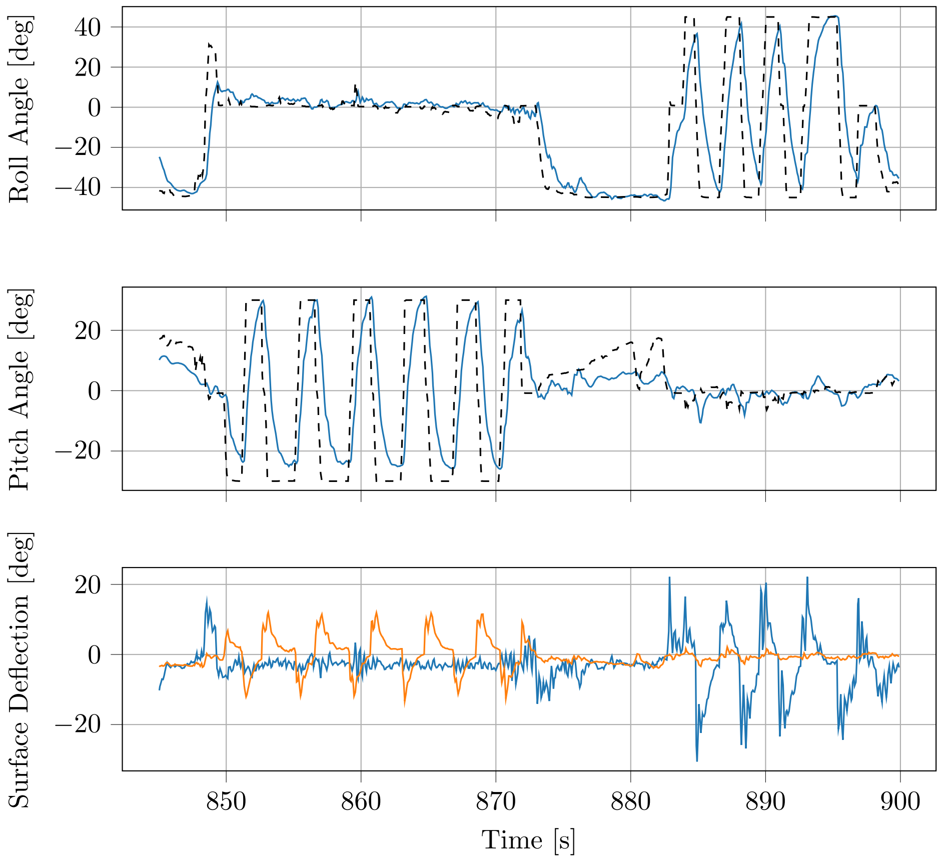


Figure 13.4: Performance of GAC after a short tuning procedure. The plots show the tracking response for roll and pitch references (dashed, black). The bottom subplot depicts the virtual deflections of the aileron (blue) and the elevator (orange).

thorough evaluation of the GAC experiments is set to appear in a separate article.

13.6.2 Deep Reinforcement Learning (DRL)

Fig. 13.5 shows one of the first attempts during flight testing of the DRL controller. Again, after takeoff and checking that all systems behave as expected, the pilot gave control over the servos to the DRL controller by flipping the MUX switch. The closed-loop pitch response was highly oscillatory (unstable), causing the pilot to switch to manual control of the UAV. After tweaking a few parameters to scale down the magnitude of the controller outputs (while the UAV was still flying), we were able to reduce the oscillations. Fig. 13.5 shows a marginally stable response where the pilot was comfortable leaving the experimental algorithm in control. After some design iterations with rapid test cycles, we obtained results comparable to a well-tuned benchmark PID controller. See [32] for details.

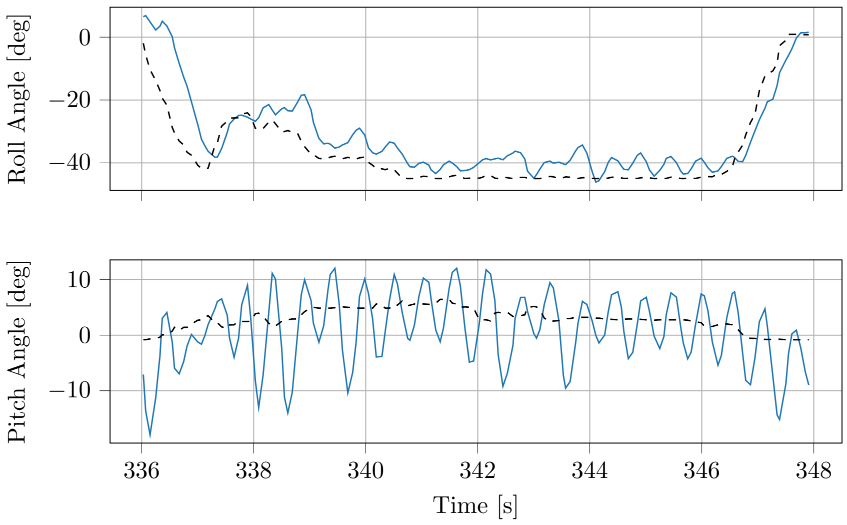
These examples illustrate how our system lowers the threshold for high-risk tests of experimental low-level algorithms. The main takeaways are (a) the switch does not interfere when our algorithm works, and in case it doesn't, our system saves the day (although we did crash a few times due to human errors), and (b) the serial communication between the Cube and the SBC provide state estimates with a low enough latency to be satisfactory for our purposes.

13.7 Lessons learned

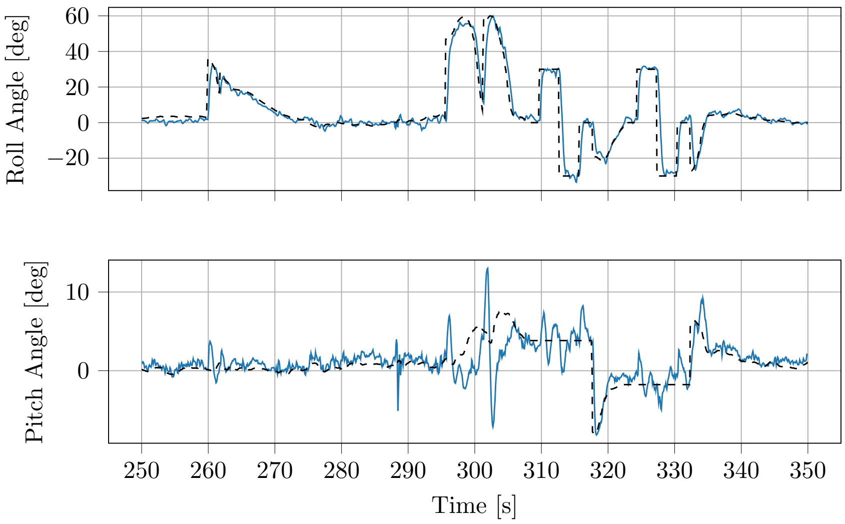
In this section, we discuss some lessons learned based on the experience gained in this project.

Previous work at our lab has focused on different aspects of state estimation for autonomous vehicles, including GNSS-aided inertial navigation [39] and estimation of airflow angles [115, 251]. However, in our work, the focus has been on control. To provide the SBC with the state estimates needed to run our control algorithms, we chose a pragmatic solution instead of spending time on implementing a custom solution: to utilize the already existing navigation solution provided by ArduPlane and send this to the SBC using the MAVLink protocol. Initially, we were worried that the latency of this link could cause problems with the closed-loop operation of our algorithms or that the achieved update frequency would be too low. However, this has not caused any issues to this day. Therefore, instead of spending too much time and resources on a perfect solution, we choose something simple and stick with it until something better is needed.

An unexpected problem that was particularly time-consuming during the preparations of the experiments was the electromagnetic interference between components of the flight stack. Being unaware of the fact that this is the source of error makes it hard to debug components of the system. For example, on our test platform, an earlier controller board interfered with the GNSS antenna, ultimately causing the EKF to diverge in a non-deterministic way. It took a considerable amount of time until we discovered a correlation with the distance between the GNSS antenna and the controller board. Early integration tests with alternatives for each hardware component is our lesson learned in this case, assuming that an engineering team with expertise in electromagnetic interference is not part of the crew.



(a) Early DRL experiment.



(b) Performance after a few design iterations.

Figure 13.5: Experimental results with DRL showing (a) oscillatory attitude response (blue) vs reference angles (dashed, black) in initial flight experiments, and (b) after a few design iterations, the controller achieves good tracking performance. See [32] for details.

Before attempting the first flight experiments of our model-based designs, we lingered for too long because initial model validation efforts showed some discrepancies with flight data. Looking back, we would have instead performed more early flight tests before iteratively improving the model [211, 213]. For instance, the NMPC has proved to be robust against modelling errors. Do not underestimate the robustness of feedback control.

The pilot is an essential part of the crew and can be a good resource when doing experimental work. Supporting with experience from the field and practical aspects of the UAV, it is a good idea to keep the pilot in close communication and discuss ideas early to get additional insights into the feasibility of the case study. Create an open environment where ideas can be freely discussed.

A working experimental platform is a solid foundation for rapid prototyping of practical control designs. However, it has taken some time to get there. Performing flight experiments with fixed-wing UAVs is an outdoor sport. Weather conditions, travel time to the airfield, and the size of our test crew are all elements that make this a substantial undertaking.

13.8 Chapter Summary

We provided a detailed description of a flight-stack architecture and experiment protocols to test advanced nonlinear control algorithms. The hardware architecture consists of off-the-shelf components researchers can integrate into existing flight platforms with minimum effort. Finally, we demonstrate the practical use with examples of low-level motion control algorithms from our lab and conclude with lessons learned to help other researchers avoid pitfalls we discovered while building our test infrastructure.

Concluding Remarks

Chapter 14

Conclusions and Future Directions

This thesis has contributed to the theory and practice of nonlinear autopilot design for fixed-wing aircraft with a particular focus on UAV applications. Each chapter contains a summary section at the end. In this chapter, we conclude this thesis by highlighting some future research directions, structured by parts.

Part I: Geometric Attitude Control Laws for Fixed-Wing Aircraft

In Chapter 4, the simulation results show that the geometric reduced-attitude controller is more efficient than the Euler-angle-based (roll/pitch) controller in the sense that it uses less control energy. However, the difference in the angle of attack and sideslip angle causes a difference between the dynamic inversion terms of the two controllers. Therefore, the increase in efficiency should be investigated further in a more detailed comparison between the two approaches by using Monte Carlo simulation.

Chapter 5 presented a hybrid controller to overcome the topological obstruction to global attitude stabilization. Although the results here are intriguing, further studies should assess if we gain anything (robustness or performance) with the extra machinery compared to simpler discontinuous control designs using simulated measurement noise with representative noise levels. Further, a drawback of the hybrid controller is the deceleration close to the expelling reference as shown in Figure 5.3 and Figure 5.5. This suggests using a dynamic extension in which the control action is given by a dynamic weighting of both configuration error vectors as done in [21] or

[172]. This would essentially mean extending the backstepping-based design of Chapter 6 with the hybrid design principles outlined here. Future work will also address the performance of the hybrid controller in the face of non-vanishing disturbances and model perturbations. An interesting topic here would be to combine hybrid and robust controller designs. This is not necessarily straightforward since the jump between potential functions would cause a discontinuous jump in the sliding variable.

The controllers designed in Chapter 6 and Chapter 7 have been implemented in the ArduPlane open-source autopilot. SITL simulation results are presented in Chapter 6. Initial flight test results, some of which are shown in Chapter 13, demonstrate the performance of the results. Although the simulation results and initial flight test results look promising, further validation of the presented approach is warranted. In particular, extensive flight experiments on a physical UAV platform should be carried out to evaluate the real contribution compared to state-of-the-art autopilots for fixed-wing UAVs. One additional practical aspect that has not been covered in this chapter and needs to be considered in real-life applications is the use of anti-windup mechanisms in conjunction with the super-twisting control law. Anti-windup in the context of the super-twisting algorithm has recently been treated in [87].

Part II: Path-Following Control

The results of Part II could be extended in a few different directions. One downside of the approach is the reliance on a known wind estimate. The method can be improved by the addition of a wind estimator, which would (cf. Section 9.6) result in a 3D adaptive/integral LOS-like algorithm [33]. Another extension could be to use similar methods as in Chapter 5 to achieve global asymptotic stability also for the path-following controller. Furthermore, a coupled stability analysis with an inner-loop attitude controller is a natural next step, including saturation limits on the normal acceleration (and thus on the angle of attack).

Part III: Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs

Further work in this direction should investigate if DRL solutions to more complex flight control problems also transfer well to the field, e.g. deep-stall landings or end-to-end path following. The problem of limited integral action should also be further investigated. Moreover, learning from real data, be it historical or generated online by the learning controller, is an intriguing research direction that would alleviate the need for accurate nonlinear models. The data efficiency of our method shows that this should, in fact, be feasible.

References

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1–8. MIT Press, 2007.

[2] P. Abichandani, D. Lobo, G. Ford, D. Bucci, and M. Kam. Wind measurement and simulation techniques in multi-rotor small unmanned aerial vehicles. *IEEE Access*, 8:54910–54927, 2020.

[3] 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.

[4] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic. Path-following for nonminimum phase systems removes performance limitations. *IEEE Transactions on Automatic Control*, 50(2):234–239, 2005.

[5] T. S. Andersen and R. Kristiansen. Quaternion Path-Following in Three Dimensions for a Fixed-Wing UAV Using Quaternion Blending. *2018 IEEE Conference on Control Technology and Applications, CCTA 2018*, pages 1597–1602, 2018.

[6] J. D. Anderson. *Introduction to Flight.* McGraw Hill, 1989.

[7] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5055–5065, Red Hook, NY, USA, 2017. Curran Associates Inc.

[8] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al.

Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[9] D. Angeli. An almost global notion of input-to-state stability. *IEEE Transactions on Automatic Control*, 49(6):866–874, 2004.

[10] D. Angeli and L. Praly. Stability robustness in the presence of exponentially unstable isolated equilibria. *IEEE Transactions on Automatic Control*, 56(7):1582–1592, 2011.

[11] D. Angeli and E. D. Sontag. Forward completeness, unboundedness observability, and their lyapunov characterizations. *Systems & Control Letters*, 38(4-5):209–217, dec 1999.

[12] A. Anglade, J.-M. Kai, T. Hamel, and C. Samson. Automatic control of convertible fixed-wing drones with vectorized thrust. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 5880–5887, 2019.

[13] Ardupilot [online]. `http://www.ardupilot.org`, 2021. Accessed: May 10, 2021.

[14] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, and G. Casalino. Drone deep reinforcement learning: A review. *Electronics*, 10(9), 2021.

[15] A.-R. Babaei, M. Malekzadeh, and D. Madhkhan. Adaptive super-twisting sliding mode control of 6-dof nonlinear and uncertain air vehicle. *Aerospace Science and Technology*, 84:361 – 374, 2019.

[16] J. A. Bagnell and J. G. Schneider. Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods. In *2001 IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[17] M. Basescu and J. Moore. Direct NMPC for post-stall motion planning with fixed-wing UAVs. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9592–9598, 2020.

[18] R. W. Beard. UAVBOOK Supplement. Additional thoughts on propeller thrust model. Technical report, Princeton University Press, 2014.

[19] R. W. Beard and T. W. McLain. *Small Unmanned Aircraft*. Princeton University Press, Princeton, NJ, 2012.

[20] S. Berkane. *Hybrid Attitude Control and Estimation On SO(3)*. PhD thesis, University of Western Ontario, 2017.

[21] S. Berkane, A. Abdessameud, and A. Tayebi. Hybrid global exponential stabilization on so(3). *Automatica*, 81:279–285, 2017.

[22] S. Berkane and A. Tayebi. On the design of attitude complementary filters on SO(3). *IEEE Transactions on Automatic Control*, 63(3):880–887, 2018.

[23] N. Bernini, M. Bessa, R. Delmas, A. Gold, E. Goubault, R. Pennec, S. Putot, and F. Sillion. A few lessons learned in reinforcement learning for quadcopter attitude control. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[24] S. Bertrand, T. Hamel, H. Piet-Lahanier, and R. Mahony. Attitude tracking of rigid bodies on the special orthogonal group with bounded partial state feedback. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 2972–2977, 2009.

[25] D. P. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, Nashua, NH, 2019.

[26] S. P. Bhat and D. S. Bernstein. A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon. *Syst. Control Lett.*, 39(1):63–70, jan 2000.

[27] E. Bøhn. Fixed-wing aircraft gym environment. `https://github.com/eivindeb/fixed-wing-gym`, 2019.

[28] E. Bøhn. Pyfly. `https://github.com/eivindeb/pyfly`, 2019.

[29] E. Bøhn. Project code and details. `https://github.com/eivindeb/rluav`, 2021.

[30] E. Bøhn. *Reinforcement Learning for Optimization of Nonlinear and Predictive Control*. PhD thesis, Norwegian University of Science and Technology, 2022.

[31] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen. Deep reinforcement learning attitude control of fixed-wing UAVs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[32] E. Bøhn, E. M. Coates, D. Reinhardt, and T. A. Johansen. Data-efficient deep reinforcement learning for attitude control of fixed-wing UAVs: Field experiments. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[33] E. Borhaug, A. Pavlov, and K. Y. Pettersen. Integral los control for path following of underactuated marine surface vessels in the presence of constant ocean currents. In *2008 47th IEEE Conference on Decision and Control*, pages 4984–4991, 2008.

[34] I. G. Borlaug, K. Y. Pettersen, and J. T. Gravdahl. The generalized super-twisting algorithm with adaptive gains. In *2020 European Control Conference (ECC)*, pages 1624–1631, 2020.

[35] H. Bou-Ammar, H. Voos, and W. Ertel. Controller design for quadrotor UAVs using reinforcement learning. In *2010 IEEE International Conference on Control Applications*, pages 2130–2135. IEEE, sep 2010.

[36] M. Breivik and T. I. Fossen. Principles of guidance-based path following in 2d and 3d. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 627–634, 2005.

[37] D. Brescianini and R. D'Andrea. Tilt-prioritized quadrocopter attitude control. *IEEE Transactions on Control Systems Technology*, 28(2):376–387, 2020.

[38] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, June 2016. arXiv: 1606.01540.

[39] T. H. Bryne, J. M. Hansen, R. H. Rogne, N. Sokolova, T. I. Fossen, and T. A. Johansen. Nonlinear observers for integrated insgnss navigation: Implementation aspects. *IEEE Control Systems Magazine*, 37(3):59–86, 2017.

[40] E. Bulka and M. Nahon. Automatic control for aerobatic maneuvering of agile fixed-wing uavs. *J Intell Robot Syst*, 93:85–100, 2019.

[41] F. Bullo, R. Murray, and A. Sarti. Control on the sphere and reduced attitude stabilization. *IFAC Proc. Vol.*, 28(14):495–501, 1995.

[42] F. Bullo and R. M. Murray. Tracking for fully actuated mechanical systems: a geometric framework. *Automatica*, 35(1):17–34, jan 1999.

[43] J. A. Burton and A. S. I. Zinober. Continuous approximation of variable structure control. *International Journal of Systems Science*, 17:875–885, 1986.

[44] D. Cabecinhas, C. Silvestre, P. Rosa, and R. Cunha. Path-following control for coordinated turn aircraft maneuvers. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007.

[45] I. Carlucho, M. De Paula, S. Wang, Y. Petillot, and G. G. Acosta. Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning. *Robotics and Autonomous Systems*, 107:71–86, sep 2018.

[46] L. H. Carter and J. S. Shamma. Gain-scheduled bank-to-turn autopilot design using linear parameter varying transformations. *Journal of Guidance, Control and Dynamics*, 19(5):1056–1063, 1996.

[47] P. Casau, C. G. Mayhew, R. G. Sanfelice, and C. Silvestre. Robust global exponential stabilization on the n-dimensional sphere with applications to trajectory tracking for quadrotors. *Automatica*, 110:108534, dec 2019.

[48] H. Castañeda, O. S. Salas-Peña, and J. de León-Morales. Extended observer based on adaptive second order sliding mode control for a fixed wing UAV. *ISA Transactions*, 66:226–232, jan 2017.

[49] H. Castañeda, O. S. Salas-Peña, and J. de León-Morales. Robust flight control for a fixed-wing unmanned aerial vehicle using adaptive super-twisting approach. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 228(12):2310–2322, 2014.

[50] I. Castillo, L. Fridman, and J. A. Moreno. Super-twisting algorithm in presence of time and state dependent perturbations. *International Journal of Control*, 91(11):2535–2548, 2018.

[51] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. Philip Chen. Review of advanced guidance and control algorithms for space/aerospace vehicles. *Progress in Aerospace Sciences*, 122:100696, 2021.

[52] N. Chaturvedi and H. McClamroch. Asymptotic stabilization of the inverted equilibrium manifold of the 3-D pendulum using non-smooth feedback. *IEEE Trans. Autom. Control*, 54(11):2658–2662, nov 2009.

[53] N. Chaturvedi, N. McClamroch, and D. Bernstein. Asymptotic smooth stabilization of the inverted 3-D pendulum. *IEEE Trans. Autom. Control*, 54(6):1204–1215, jun 2009.

[54] N. Chaturvedi, A. Sanyal, and H. McClamroch. Rigid-body attitude control. *IEEE Control Syst.*, 31(3):30–51, jun 2011.

[55] N. Cho, Y. Kim, and S. Park. Three-dimensional nonlinear differential geometric path-following guidance law. *Journal of Guidance, Control, and Dynamics*, 38(12):2366–2385, 2015.

[56] V. Cichella, I. Kaminer, V. Dobrokhodov, E. Xargay, N. Hovakimyan, and A. Pascoal. Geometric 3d path-following control for a fixed-wing UAV on so(3). In *AIAA Guidance, Navigation, and Control Conference*, page 6415, 2011.

[57] R. J. Clarke, L. Fletcher, C. Greatwood, A. Waldock, and T. S. Richardson. Closed-loop q-learning control of a small unmanned aircraft. In *AIAA Scitech 2020 Forum*, page 1234, 2020.

[58] E. M. Coates and T. I. Fossen. Geometric reduced-attitude control of fixed-wing UAVs. *Applied Sciences*, 11(7), 2021.

[59] E. M. Coates, J. B. Griffiths, and T. A. 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)*, 2021.

[60] E. M. Coates, T. Hamel, and T. I. Fossen. Almost global three-dimensional path-following guidance law for arbitrary curved paths. In *62nd IEEE Conference on Decision and Control (CDC) (accepted)*, 2023.

[61] E. M. Coates, D. Reinhardt, and T. I. Fossen. Reduced-attitude control of fixed-wing unmanned aerial vehicles using geometric methods on the two-sphere. *21st IFAC World Congress, Berlin, Germany. IFAC-PapersOnLine*, 53:5749–5756, 2020.

[62] E. M. Coates, D. Reinhardt, K. Gryte, and T. A. 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)*, 2022.

[63] E. M. Coates, A. Wenz, K. Gryte, and T. A. Johansen. Propulsion system modeling for small fixed-wing UAVs. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[64] P. Corke and R. Mahony. Sensing and control on the sphere. In *Springer Tracts in Advanced Robotics*, pages 71–85. Springer Berlin Heidelberg, 2011.

[65] J. J. Corona-Sánchez, Óscar Roberto Guzmán Caso, and H. Rodríguez-Cortés. A coordinated turn controller for a fixed-wing aircraft. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(5):1728–1740, 2019.

[66] R. Cory and R. Tedrake. Experiments in fixed-wing uav perching. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[67] T. Cunis, D. Liao-McPherson, I. Kolmanovsky, and L. Burlion. Model-predictive spiral and spin upset recovery control for the generic transport model simulation. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1–7, 2020.

[68] H. A. de Oliveira and P. F. F. Rosa. Genetic neuro-fuzzy approach for unmanned fixed wing attitude control. In *2017 International Conference on Military Technologies (ICMT)*. IEEE, may 2017.

[69] Q. Dong, Q. Zong, B. Tian, and F. Wang. Adaptive-gain multivariable super-twisting sliding mode control for reentry rlv with torque perturbation: Sliding mode control for reentry rlv. *International Journal of Robust and Nonlinear Control*, 27, 07 2016.

[70] P. G. Fahlstrom, T. J. Gleason, and M. H. Sadraey. *Introduction to UAV Systems*. John Wiley & Sons, Hoboken, NJ, 2022.

[71] W. Fan and B. Tian. Adaptive multivariable super-twisting sliding mode controller and disturbance observer design for hypersonic vehicle. *Mathematical Problems in Engineering*, 2016:1–9, 08 2016.

[72] X. Fang, A. Wu, Y. Shang, and C. Du. Multivariable super twisting based robust trajectory tracking control for small unmanned helicopter. *Mathematical Problems in Engineering*, 2015:1–13, 05 2015.

[73] J. Farrell, M. Sharma, and M. Polycarpou. Backstepping-based flight control with adaptive function approximation. *Journal of Guidance, Control, and Dynamics*, 28(6):1089–1102, 2005.

[74] F. Fei, Z. Tu, D. Xu, and X. Deng. Learn-to-recover: Retrofitting uavs with reinforcement learning-assisted flight control under cyber-physical attacks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7358–7364. IEEE, 2020.

[75] S. Ferrari and R. F. Stengel. Online adaptive critic flight control. *Journal of Guidance, Control, and Dynamics*, 27(5):777–786, sep 2004.

[76] A. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Springer Science & Business Media, Dordrecht, The Netherlands, 1988.

[77] T. Fisher and R. Sharma. Rudder-augmented trajectory correction for small unmanned aerial vehicle to minimize lateral image errors. *Journal of Aerospace Information Systems*, 15(9), 2018.

[78] P. Fitzpatrick. Calculation of thrust in a ducted fan assembly for hovercraft. Technical report, Hovercraft Club of Great Britain, 2003.

[79] L. J. Fletcher, R. J. Clarke, T. S. Richardson, and M. Hansen. Reinforcement learning for a perched landing in the presence of wind. In *AIAA Scitech 2021 Forum*, page 1282, 2021.

[80] T. I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley-Blackwell, 2011.

[81] T. I. Fossen and K. Y. Pettersen. On uniform semiglobal exponential stability (USGES) of proportional line-of-sight guidance laws. *Automatica*, 50(11):2912–2917, 2014.

[82] K. Gamagedara, M. Bisheban, E. Kaufman, and T. Lee. Geometric controls of a quadrotor uav with decoupled yaw control. In *2019 American Control Conference (ACC)*, 2019.

[83] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955, Sept. 2017.

[84] G. A. Garcia, S. Kashmiri, and D. Shukla. Nonlinear control based on h-infinity theory for autonomous aerial vehicle. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2017.

[85] C. V. Girish, F. Emilio, H. P. Jonathan, and L. Hugh. Nonlinear flight control techniques for unmanned aerial vehicles. In *Handbook of*

Unmanned Aerial Vehicles, pages 577–612. Springer Netherlands, aug 2014.

[86] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness.* Princeton University Press, 2012.

[87] M. A. Golkani, S. Koch, R. Seeber, M. Reichhartinger, and M. Horn. An anti-windup scheme for the super-twisting algorithm. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6947–6952, 2019.

[88] T. Gonzalez, J. A. Moreno, and L. Fridman. Variable gain super-twisting sliding mode control. *IEEE Transactions on Automatic Control*, 57(8):2100–2105, 2012.

[89] J. B. Griffiths. Multivariable super-twisting control of fixed-wing UAVs. Master's thesis, Norwegian University of Science and Technology (NTNU), 2020.

[90] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control*, pages 45–69. Springer International Publishing, Cham, 2017.

[91] K. Gryte. High Angle of Attack Landing of an Unmanned Aerial Vehicle. Master's thesis, Norwegian University of Science and Technology, 2015.

[92] K. Gryte. *Precision control of fixed-wing UAV and robust navigation in GNSS-denied environments.* PhD thesis, Norwegian University of Science and Technology (NTNU), 2020.

[93] K. Gryte, R. Hann, M. Alam, J. Roháč, T. A. Johansen, and T. I. Fossen. Aerodynamic modeling of the skywalker x8 fixed-wing unmanned aerial vehicle. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 826–835, 2018.

[94] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* Springer-Verlag, New York, NY, 1983.

[95] V. Guillemin and A. Pollack. *Differential Topology.* American Mathematical Society, 1974.

[96] R. Guruganesh, B. Bandyopadhyay, H. Arya, and G. K. Singh. Design and hardware implementation of autopilot control laws for MAV using super twisting control. *J Intell Robot Syst*, 90:455–471, 2018.

[97] S. Gutman. Uncertain dynamical systems–a Lyapunov min-max approach. *IEEE Transactions on Automatic Control*, 24(3):437–443, 1979.

[98] S. Gutman and G. Leitmann. Stabilizing control for linear systems with bounded parameter and input uncertainty. In J. Cea, editor, *Optimization Techniques Modeling and Optimization in the Service of Man Part 2*, pages 729–755, Berlin, Heidelberg, 1976. Springer Berlin Heidelberg.

[99] G. C. Gómez Cortés, F. Castaños, and J. Dávila. Sliding motions on so(3), sliding subgroups. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6953–6958, 2019.

[100] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *35th International Conference on Machine Learning*, pages 1861–1870, Stockholm, Sweden, 2018. PMLR.

[101] J.-H. Han, D.-K. Lee, J.-S. Lee, and S.-J. Chung. Teaching micro air vehicles how to fly as we teach babies how to walk. *Journal of Intelligent Material Systems and Structures*, 24(8):936–944, 2013.

[102] J. L. Hernandez, I. González-Hernández, and R. Lozano. Super-twisting control in a solar unmanned aerial vehicle: Application to solar tracking. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 379–384, 2018.

[103] J. C. Hernández Ramírez and M. Nahon. Nonlinear vector-projection control for agile fixed-wing unmanned aerial vehicles. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5314–5320, 2020.

[104] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

[105] M. Hosseinzadeh and M. J. Yazdanpanah. Performance enhanced model reference adaptive control through switching non-quadratic lyapunov functions. *Systems & Control Letters*, 76:47–55, 2015.

[106] A. Hovenburg, T. A. Johansen, and R. Storvold. Mission performance trade-offs of battery-powered suas. In *Int. Conf. Unmanned Aircraft Systems, Miami*, 2017.

[107] C. Hu, Y. Qin, H. Cao, X. Song, K. Jiang, J. J. Rath, and C. Wei. Lane keeping of autonomous vehicles based on differential steering with adaptive multivariable super-twisting control. *Mechanical Systems and Signal Processing*, 125:330 – 346, 2019.

[108] M. Hua, T. Hamel, P. Morin, and C. Samson. A control approach for thrust-propelled underactuated vehicles and its application to vtol drones. *IEEE Transactions on Automatic Control*, 54(8):1837–1853, 2009.

[109] N. Hung, F. Rego, J. Quintas, J. Cruz, M. Jacinto, D. Souto, A. Potes, L. Sebastiao, and A. Pascoal. A review of path following control strategies for autonomous robotic vehicles: Theory, simulations, and experiments. *Journal of Field Robotics*, dec 2022.

[110] S.-M. Hung and S. N. Givigi. A Q-Learning Approach to Flocking With UAVs in a Stochastic Environment. *IEEE Transactions on Cybernetics*, 47(1):186–197, jan 2017.

[111] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen. Autonomous navigation of UAV by using real-time model-based reinforcement learning. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–6. IEEE, nov 2016.

[112] P. A. Ioannou and J. Sun. *Robust Adaptive Control.* Dover Publications, Inc., Mineola, NY, 2012.

[113] R. V. Jategaonkar. *Flight Vehicle System Identification: A Time-Domain Methodology, Second Edition.* American Institute of Aeronautics and Astronautics, Inc, Reston, VA, 2015.

[114] T. A. Johansen. Toward dependable embedded model predictive control. *IEEE Systems Journal*, 11(2):1208–1219, 2017.

[115] T. A. Johansen, A. Cristofaro, K. Sorensen, J. M. Hansen, and T. 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.

[116] T. A. Johansen and T. I. Fossen. Guidance, navigation, and control of fixed-wing unmanned aerial vehicles. In M. H. Ang, O. Khatib, and B. Siciliano, editors, *Encyclopedia of Robotics*, pages 1–9. Springer Berlin Heidelberg, Berlin, Heidelberg, 2020.

[117] T. A. Johansen, A. Zolich, T. Hansen, and A. J. Sorensen. Unmanned aerial vehicle as communication relay for autonomous underwater vehicle — Field tests. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 1469–1474, 2014.

[118] J.-M. Kai, A. Anglade, T. Hamel, and C. Samson. Design and experimental validation of a new guidance and flight control system for scale-model airplanes. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4270–4276, 2018.

[119] J.-M. Kai, T. Hamel, and C. Samson. A unified approach to fixed-wing aircraft path following guidance and control. *Automatica*, 108:108491, oct 2019.

[120] U. V. Kalabić, R. Gupta, S. Di Cairano, A. M. Bloch, and I. V. Kolmanovsky. MPC on manifolds with an application to the control of spacecraft attitude on SO(3). *Automatica*, 76:293–300, 2017.

[121] M. Kamel, T. Stastny, K. Alexis, and R. 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.

[122] I. Kaminer, A. Pascoal, E. Xargay, N. Hovakimyan, C. Cao, and V. Dobrokhodov. Path following for small unmanned aerial vehicles using l1 adaptive augmentation of commercial autopilots. *Journal of Guidance, Control, and Dynamics*, 33(2):550–564, mar 2010.

[123] Y. Kawakami and K. Uchiyama. Nonlinear controller design for transition flight of a fixed-wing UAV with input constraints. In *AIAA Guidance, Navigation, and Control Conference.* American Institute of Aeronautics and Astronautics, jan 2017.

[124] T. A. Keijock, E. V. L. Nunes, and L. Hsu. On the fragility of multivariable super-twisting algorithm for non-symmetric uncertain input matrix. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 7857–7862, 2019.

[125] H. K. Khalil. *Nonlinear Systems.* Pearson, 3rd edition, 2002.

[126] Ø. K. Kjerstad and E. M. Coates. A cascaded heading control design with motion constraint handling for marine surface vessels. In *2023 European Control Conference (ECC)*, 2023.

[127] W. Koch, R. Mancuso, and A. Bestavros. Neuroflight: Next Generation Flight Control Firmware. *arXiv:1901.06553 [cs]*, Sept. 2019. arXiv: 1901.06553.

[128] W. Koch, R. Mancuso, R. West, and A. Bestavros. Reinforcement learning for uav attitude control. *ACM Trans. Cyber-Phys. Syst.*, 3(2), Feb. 2019.

[129] D. E. Koditschek. The application of total energy as a lyapunov function for mechanical control systems. *Contemporary mathematics*, 97:131, 1989.

[130] D. Kooijman, A. P. Schoellig, and D. J. Antunes. Trajectory tracking for quadrotors with attitude control on $\mathcal{S}^2 \times \mathcal{S}^1$. In *2019 18th European Control Conference (ECC)*, pages 4002–4009, 2019.

[131] A. B. Krishna, A. Sen, and M. Kothari. Robust geometric control of a helicopter using sliding mode control. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 737–743, 2020.

[132] R. Kristiansen, P. Nicklasson, and J. Gravdahl. Satellite attitude control by quaternion-based backstepping. *IEEE Transactions on Control Systems Technology*, 17(1):227–232, jan 2009.

[133] S. Kurnaz, O. Cetin, and O. Kaynak. Fuzzy logic based approach to design of flight control and navigation tasks for autonomous unmanned aerial vehicles. *Journal of Intelligent and Robotic Systems*, 54(1-3):229–244, oct 2008.

[134] P. Ladosz, M. Coombes, J. Smith, and M. Hutchinson. A generic ros based system for rapid development and testing of algorithms for autonomous ground and aerial vehicles. In A. Koubaa, editor, *Robot Operating System (ROS): The Complete Reference (Volume 3)*, pages 113–153. Springer International Publishing, Cham, 2019.

[135] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robotics and Automation Letters*, 4(4):4224–4230, 2019.

[136] A. A. Lambregts. Vertical flight path and speed control autopilot design using total energy principles. In *Aiaa Guid. Control. Conf*, 1983.

[137] S. H. Lane and R. F. Stengel. Flight control design using nonlinear inverse dynamics. In *1986 American Control Conference*, pages 587–596, 1986.

[138] K. A. Lavretsky Eugene; Wise. *Robust and Adaptive Control With Aerospace Applications*. Springer, 2012.

[139] D. A. Lawrence, E. W. Frew, and W. J. Pisano. Lyapunov vector fields for autonomous unmanned aircraft flight control. *Journal of Guidance, Control and Dynamics*, 31(5):1220–1229, 2008.

[140] F. Le Bras, T. Hamel, R. Mahony, C. Barat, and J. Thadasack. Approach maneuvers for autonomous landing using visual servo control. *IEEE Transactions on Aerospace and Electronic Systems*, 50(2):1051–1065, apr 2014.

[141] T. Lee. Exponential stability of an attitude tracking control system on SO(3) for large-angle rotational maneuvers. *Systems and Control Letters*, 61(1):231–237, 2012.

[142] T. Lee. Global exponential attitude tracking controls on SO(3). *IEEE Trans. Autom. Control*, 60(10):2837–2842, oct 2015.

[143] T. 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.

[144] T. Lee and Y. Kim. Nonlinear adaptive flight control using backstepping and neural networks controller. *Journal of Guidance, Control, and Dynamics*, 24(4):675–682, jul 2001.

[145] 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, 2011.

[146] A. Levant. Sliding order and sliding accuracy in sliding mode control. *International Journal of Control*, 58(6):1247–1263, 1993.

[147] A. Levant. Higher-order sliding modes, differentiation and output-feedback control. *International Journal of Control*, 76(9-10):924–941, 2003.

[148] A. Levant. Homogeneity approach to high-order sliding mode design. *Automatica*, 41(5):823–830, 2005.

[149] S. Leven, J. Zufferey, and D. Floreano. A minimalist control strategy for small uavs. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2873–2878, 2009.

[150] J. M. Levin, A. A. Paranjape, and M. Nahon. Agile maneuvering with a small fixed-wing unmanned aerial vehicle. *Robotics and Autonomous Systems*, 116:148–161, jun 2019.

[151] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[152] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv:1509.02971 [cs, stat]*, Sept. 2015. arXiv: 1509.02971.

[153] X. Lin, Y. Yu, and C. Sun. Supplementary reinforcement learning controller designed for quadrotor uavs. *IEEE Access*, 7:26422–26431, 2019.

[154] C. Liu and W.-H. Chen. Disturbance rejection flight control for small fixed-wing unmanned aerial vehicles. *Journal of Guidance, Control and Dynamics*, 39(12):2804–2813, 2016.

[155] C. Liu, O. McAree, and W.-H. Chen. Path-following control for small fixed-wing unmanned aerial vehicles under wind disturbances. *International Journal of Robust and Nonlinear Control*, 23(15):1682–1698, 2013.

[156] J. Liu, M. Sun, Z. Chen, and Q. Sun. Super-twisting sliding mode control for aircraft at high angle of attack based on finite-time extended state observer. *Nonlinear Dynamics*, 99:2785–2799, 2020.

[157] B. T. Lopez and J.-J. E. Slotine. Sliding on manifolds: Geometric attitude control with quaternions, 2020.

[158] A. Loria. From feedback to cascade-interconnected systems: Breaking the loop. In *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008.

[159] A. Loría and E. Panteley. 2 cascaded nonlinear time-varying systems: Analysis and design. In *Advanced Topics in Control Systems Theory*, pages 23–64. Springer London, aug 2005.

[160] F. López-Caamal and J. A. Moreno. Generalised multivariable supertwisting algorithm. *International Journal of Robust and Nonlinear Control*, 29(3):634–660, 2019.

[161] B. Løw-Hansen, N. C. Müller, E. M. Coates, T. A. Johansen, and R. Hann. Identification of an electric uav propulsion system in icing conditions. *SAE International Conference on Icing of Aircraft, Engines, and Structures*, 2023.

[162] J. Markdahl, J. Hoppe, L. Wang, and X. Hu. A geodesic feedback law to decouple the full and reduced attitude. *Systems and Control Letters*, 102:32–41, 2017.

[163] F. L. Markley. Attitude error representations for kalman filtering. *Journal of Guidance, Control, and Dynamics*, 26(2):311–317, mar 2003.

[164] F. L. Markley and J. L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control.* Springer, New York, NY, 2014.

[165] A. B. Martinsen. End-to-end training for path following and control of marine vehicles. Master's thesis, NTNU, 2018.

[166] S. Mathisen, K. Gryte, S. Gros, and T. A. Johansen. Precision deep-stall landing of fixed-wing uavs using nonlinear model predictive control. *J Intell Robot Syst*, 101(24), 2021.

[167] S. H. Mathisen, K. Gryte, T. Johansen, and T. I. Fossen. Non-linear model predictive control for longitudinal and lateral guidance of a small fixed-wing UAV in precision deep stall landing. In *AIAA Infotech @ Aerospace.* American Institute of Aeronautics and Astronautics, jan 2016.

[168] MathWorks. Dryden Wind Turbulence Model (Continuous). `https://se.mathworks.com/help/aeroblks/drydenwindturbulencemodelcontinuous.html`, 2020.

[169] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel. Quaternion-based hybrid control for robust global attitude tracking. *IEEE Transactions on Automatic Control*, 56(11):2555–2566, 2011.

[170] C. G. Mayhew and A. R. Teel. On the topological structure of attraction basins for differential inclusions. *Systems and Control Letters*, 60(12):1045–1050, 2011.

[171] C. G. Mayhew and A. R. Teel. Synergistic potential functions for hybrid control of rigid-body attitude. In *Proceedings of the 2011 American Control Conference*, pages 875–880, 2011.

[172] C. G. Mayhew and A. 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, jul 2013.

[173] C. G. Mayhew and A. R. Teel. Synergistic hybrid feedback for global rigid-body attitude tracking on SO (3). *IEEE Transactions on Automatic Control*, 58(11):2730–2742, 2013.

[174] L. Meier, D. Honegger, and M. 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.

[175] A. Micaelli and C. Samson. Trajectory tracking for unicycle-type and two-steering-wheels mobile robots. INRIA, Tech. Rep. 2097, 1993.

[176] M. G. Michailidis, K. Kanistras, M. Agha, M. J. Rutherford, and K. P. Valavanis. Robust nonlinear control of the longitudinal flight dynamics of a circulation control fixed wing UAV. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, dec 2017.

[177] M. G. Michailidis, M. J. Rutherford, and K. P. Valavanis. A survey of controller designs for new generation UAVs: The challenge of uncertain aerodynamic parameters. *Int. J. Control Autom. Syst.*, 18:801–816, 2020.

[178] Y. Mitikiri and K. Mohseni. Attitude control of micro/mini aerial vehicles and estimation of aerodynamic angles formulated as parametric uncertainties. *IEEE Robotics and Automation Letters*, 3(3):2063–2070, 2018.

[179] Y. Mitikiri and K. Mohseni. Globally stable attitude control of a fixed-wing rudderless UAV using subspace projection. *IEEE Robotics and Automation Letters*, 4(2):1395–1401, 2019.

[180] E. A. Morelli and V. Klein. *Aircraft System Identification: Theory and Practice, Second Edition*. Sunflyte Enterprises, Williamsburg, VA, 2016.

[181] J. A. Moreno. A linear framework for the robust stability analysis of a generalized super-twisting algorithm. In *2009 6th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pages 1–6, 2009.

[182] J. A. Moreno and M. Osorio. A lyapunov approach to second-order sliding mode controllers and observers. In *2008 47th IEEE Conference on Decision and Control*, pages 2856–2861, 2008.

[183] J. A. Moreno and M. Osorio. Strict Lyapunov functions for the super-twisting algorithm. *IEEE Transactions on Automatic Control*, 57(4):1035–1040, 2012.

[184] S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko. Regularizing action policies for smooth control with reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2021.

[185] S. Mysore, B. Mabsout, K. Saenko, and R. Mancuso. How to train your quadrotor: A framework for consistently smooth and responsive flight control via reinforcement learning. *ACM Transactions on Cyber-Physical Systems*, 5(36):1–24, oct 2021.

[186] I. Nagesh and C. Edwards. A multivariable super-twisting sliding mode approach. *Automatica*, 50(3):984 – 988, 2014.

[187] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007.

[188] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. 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.

[189] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.

[190] 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.

[191] E. Oland and R. Kristiansen. A decoupled approach for flight control. *Modeling, Identification and Control*, 37(4):237–246, 2016.

[192] E. Oland, R. Kristiansen, and J. T. Gravdahl. A comparative study of different control structures for flight control with new results. *IEEE Transactions on Control Systems Technology*, 28(2):291–305, 2020.

[193] S. Park, J. Deyst, and J. How. A new nonlinear guidance logic for trajectory tracking. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, aug 2004.

[194] S. Park, J. Deyst, and J. P. How. Performance and Lyapunov Stability of a Nonlinear Path Following Guidance Method. *Journal of Guidance, Control, and Dynamics*, 30(6):1718–1728, 2007.

[195] J. Patrikar, V. R. Makkapati, A. Pattanaik, H. Parwana, and M. Kothari. Nested saturation based guidance law for unmanned aerial vehicles. *J. Dyn. Sys., Meas., Control*, 141(7):071008 (13 pages), 2019.

[196] G. V. Pelizer, N. B. F. da Silva, and K. R. L. J. Branco. Comparison of 3d path-following algorithms for unmanned aerial vehicles. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 498–505, 2017.

[197] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[198] C.-H. Pi, Y.-W. Dai, K.-C. Hu, and S. Cheng. General purpose low-level reinforcement learning control for multi-axis rotor aerial vehicles. *Sensors*, 21(13), 2021.

[199] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa. The lsts toolchain for networked vehicle systems. In *2013 MTS/IEEE OCEANS - Bergen*, pages 1–9, 2013.

[200] Pixhawk [online]. `https://pixhawk.org/`, 2021. Accessed: May 10, 2021.

[201] P. Poksawat, L. Wang, and A. Mohamed. Gain Scheduled Attitude Control of Fixed-Wing UAV with Automatic Controller Tuning. *IEEE Transactions on Control Systems Technology*, 26(4):1192–1203, 2018.

[202] R. Polvara, M. Patacchiola, S. Sharma, J. Wan, A. Manning, R. Sutton, and A. Cangelosi. Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 115–123. IEEE, jun 2018.

[203] C. M. Pong and D. W. Miller. Reduced-attitude boresight guidance and control on spacecraft for pointing, tracking, and searching. *Journal of Guidance, Control and Dynamics*, 38(6):1027–1035, 2015.

[204] K. Raj, V. Muthukumar, S. N. Singh, and K. W. Leeb. Finite-time sliding mode and super-twisting control of fighter aircraft. *Aerospace Science and Technology*, 82-83:487–498, 2018.

[205] M. Ramp and E. Papadopoulos. Attitude and angular velocity tracking for a rigid body using geometric methods on the two-sphere. In *2015 European Control Conference (ECC)*, pages 3238–3243, 2015.

[206] C. Ramprasadh and H. Arya. Multistage-fusion algorithm for estimation of aerodynamic angles in mini aerial vehicle. *Journal of Aircraft*, 49(1):93–100, 2012.

[207] A. Rantzer. A dual to lyapunov's stability theorem. *Systems & Control Letters*, 42(3):161–168, 2001.

[208] D. Reinhardt. *On nonlinear and optimization-based control of fixed-wing unmanned aerial vehicles.* PhD thesis, Norwegian University of Science and Technology, 2022.

[209] D. Reinhardt, E. M. Coates, and T. A. Johansen. Hybrid control of fixed-wing UAVs for large-angle attitude maneuvers on the two-sphere. *21st IFAC World Congress, Berlin, Germany. IFAC-PapersOnLine*, 53:5717–5724, 2020.

[210] D. Reinhardt, E. M. Coates, and T. A. Johansen. Low-level nonlinear model predictive attitude and speed control of fixed-wing unmanned aerial vehicles. *Control Engineering Practice*, submitted.

[211] D. Reinhardt, K. Gryte, and T. A. Johansen. Modeling of the skywalker x8 fixed-wing UAV: Flight tests and system identification. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022.

[212] D. Reinhardt and T. A. 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.

[213] D. Reinhardt, M. D. Pedersen, K. Gryte, and T. A. Johansen. A symmetry calibration procedure to compensate for sensor-to-airframe misalignments in wind tunnel data. In *2022 Conference on Control Technologies and Applications (CCTA)*, 2022.

[214] W. Ren and E. Atkins. Nonlinear trajectory tracking for fixed wing UAVs via backstepping and parameter adaptation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, aug 2005.

[215] D. Rotondo, A. Cristofaro, K. Gryte, and T. A. Johansen. LPV model reference control for fixed-wing UAVs. *IFAC-PapersOnLine*, 50(1):11559–11564, 2017.

[216] B. Rubí, R. Pérez, and B. Morcego. A survey of path following control strategies for uavs focused on quadrotors. *J Intell Robot Syst*, 98:241–265, 2020.

[217] C. Samson. Path following and time-varying feedback stabilization of a wheeled mobile robot. In *Proceedings of the international conference on advanced robotics and computer vision (ICARCV)*, 1992.

[218] F. Santoso, M. A. Garratt, and S. G. Anavatti. State-of-the-art intelligent flight control systems in unmanned aerial vehicles. *IEEE Transactions on Automation Science and Engineering*, 15(2):613–627, apr 2018.

[219] F. Santoso, M. A. Garratt, and S. G. Anavatti. State-of-the-art integrated guidance and control systems in unmanned vehicles: A review. *IEEE Systems Journal*, pages 1–12, 2020.

[220] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. Aparicio Ojea, E. Solowjow, and S. Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5548–5555, 2020.

[221] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al.

Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[222] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. *arXiv:1502.05477 [cs]*, Feb. 2015. arXiv: 1502.05477.

[223] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, July 2017. arXiv: 1707.06347.

[224] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant. *Sliding Mode Control and Observation.* Birkhäuser, New York, NY, USA, 2014.

[225] Y. B. Shtessel, J. A. Moreno, F. Plestan, L. M. Fridman, and A. S. Poznyak. Super-twisting adaptive sliding mode control: A Lyapunov design. In *49th IEEE Conference on Decision and Control (CDC)*, pages 5109–5113, 2010.

[226] M. D. Shuster. A survey of attitude representations. *The Journal of the Astronautical Sciences*, 41(4):439–517, Oct. 1993.

[227] P. Singh, P. Agrawal, A. Nandanwar, L. Behera, N. K. Verma, S. Nahavandi, and M. Jamshidi. Multivariable event-triggered generalized super-twisting controller for safe navigation of nonholonomic mobile robot. *IEEE Systems Journal*, pages 1–12, 2020.

[228] L. Sonneveldt, Q. P. Chu, and J. A. Mulder. Nonlinear flight control design using constrained adaptive backstepping. *Journal of Guidance, Control, and Dynamics*, 30(2):322–336, mar 2007.

[229] A. Spitzer and N. Michael. Rotational error metrics for quadrotor control, 2020.

[230] T. Stastny and R. 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.

[231] R. F. Stengel. *Flight dynamics.* Princeton University Press, 2004.

[232] J. Stephan, O. Pfeifle, S. Notter, F. Pinchetti, and W. Fichter. Precise tracking of extended three-dimensional dubins paths for fixed-wing aircraft. *Journal of Guidance, Control, and Dynamics*, 43(12):2399–2405, dec 2020.

[233] B. L. Stevens, F. L. Lewis, and E. N. Johnson. *Aircraft control and simulation: Dynamics, controls design, and autonomous systems.* John Wiley & Sons, Hoboken, NJ, 2016.

[234] P. B. Sujit, S. Saripalli, and J. B. 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.

[235] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT Press, Cambridge, MA, 2018.

[236] A. R. Teel, F. Forni, and L. Zaccarian. Lyapunov-based sufficient conditions for exponential stability in hybrid systems. *IEEE Transactions on Automatic Control*, 58(6):1591–1596, 2013.

[237] B. Tian, J. Cui, H. Lu, Z. Zuo, and Q. Zong. Adaptive finite-time attitude tracking of quadrotors with experiments and comparisons. *IEEE Transactions on Industrial Electronics*, 66(12), 2019.

[238] B. Tian, L. Liu, H. Lu, Z. Zuo, Q. Zong, and Y. Zhang. Multivariable finite time attitude control for quadrotor uav: Theory and experimentation. *IEEE Transactions on Industrial Electronics*, 65(3):2567–2577, 2018.

[239] P. Tian, H. Chao, M. Rhudy, J. Gross, and H. Wu. Wind sensing and estimation using small fixed-wing unmanned aerial vehicles: A survey. *Journal of Aerospace Information Systems*, 18(3), 2021.

[240] U.S. Department of Defense. U.S. Military Specification MIL-F-8785C. *Washington, D.C.: U.S. Department of Defense*, 1980.

[241] V. Utkin, J. Guldner, and J. Shi. *Sliding Mode Control in Electro-Mechanical Systems.* CRC Press, Boca Raton, FL, USA, 2009.

[242] K. P. Valavanis and G. J. Vachtsevanos, editors. *Handbook of Unmanned Aerial Vehicles.* Springer, Dordrecht, 2015.

[243] J. Vasconcelos, A. Rantzer, C. Silvestre, and P. J. Oliveira. Combination of lyapunov and density functions for stability of rotational motion. *IEEE Transactions on Automatic Control*, 56(11):2599–2607, 2011.

[244] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl. acados:

a modular open-source framework for fast embedded optimal control. *arXiv preprint*, 2019.

[245] P. V. N. M. Vidal, E. V. L. Nunes, and L. Hsu. Multivariable super-twisting algorithm for a class of systems with uncertain input matrix. In *2016 American Control Conference (ACC)*, pages 7201–7206, 2016.

[246] P. V. N. M. Vidal, E. V. L. Nunes, and L. Hsu. Output-feedback multivariable global variable gain super-twisting algorithm. *IEEE Transactions on Automatic Control*, 62(6):2999–3005, 2017.

[247] D. Wada, S. A. Araujo-Estrada, and S. Windsor. Unmanned aerial vehicle pitch control under delay using deep reinforcement learning with continuous action in wind tunnel test. *Aerospace*, 8(9), 2021.

[248] Y. Wang, J. Sun, H. He, and C. Sun. Deterministic policy gradient with integral compensator for robust quadrotor control. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(10):3713–3725, 2020.

[249] J. Wei, J. Yuan, and Z. Wang. Adaptive multivariable generalized super-twisting algorithm based robust coordinated control for a space robot subjected to coupled uncertainties. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(9):3244–3259, 2019.

[250] J.-Y. Wen and K. Kreutz-Delgado. The attitude control problem. *IEEE Trans. Autom. Control*, 36(10):1148–1162, 1991.

[251] A. Wenz and T. A. Johansen. Moving horizon estimation of air data parameters for uavs. *IEEE Transactions on Aerospace and Electronic Systems*, 56(3):2101–2121, June 2020.

[252] A. Winter, R. Hann, A. Wenz, K. Gryte, and T. A. Johansen. Stability of a flying wing uav in icing conditions. In *8th European Conference for Aeronautics and Aerospace Sciences (EUCASS)*, 2019.

[253] M. Wrzos-Kaminska, K. Pettersen, and J. Gravdahl. Path following control for articulated intervention-AUVs using geometric control of reduced attitude. *IFAC-PapersOnLine*, 52(16):192–197, 2019.

[254] J. Xu, T. Du, M. Foshey, B. Li, B. Zhu, A. Schulz, and W. Matusik. Learning to fly: computational controller design for hybrid uavs with reinforcement learning. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[255] K. Xu, F. Wu, and J. Zhao. Model-based deep reinforcement learning with heuristic search for satellite attitude control. *Industrial Robot: An International Journal*, pages IR–05–2018–0086, oct 2018.

[256] L. Xuehui, S. Shenmin, and G. Yong. Multivariable super-twisting sliding mode approach for attitude tracking of spacecraft. In *2015 34th Chinese Control Conference (CCC)*, pages 5789–5794, 2015.

[257] J. Yang, C. Liu, M. Coombes, Y. Yan, and W.-H. Chen. Optimal path following for small fixed-wing uavs under wind disturbances. *IEEE Transactions on Control Systems Technology*, 29(3):996–1008, May 2021.

[258] A. Zanelli. Nonlinear Model Predictive Control of a Human-sized Quadrotor. *2018 European Control Conference (ECC)*, pages 1542–1547, 2018.

[259] C. Zhang, G. Zhang, and Q. Dong. Adaptive dynamic programming-based adaptive-gain sliding mode tracking control for fixed-wing unmanned aerial vehicle with disturbances. *International Journal of Robust and Nonlinear Control*, oct 2022.

[260] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. *CoRR*, abs/1509.06791, 2015.

[261] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[262] Z. Zhen, C. Yu, S. Jiang, and J. Jiang. Adaptive super-twisting control for automatic carrier landing of aircraft. *IEEE Transactions on Aerospace and Electronic Systems*, 56(2):984–997, 2020.

[263] Y. Zhou, E.-J. van Kampen, and Q. Chu. Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback. *Journal of Guidance, Control, and Dynamics*, 40(2):493–496, feb 2017.

[264] A. Zolich, T. A. Johansen, K. Cisek, and K. Klausen. Unmanned aerial system architecture for maritime missions. design amp; hardware description. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 342–350, 2015.

ISBN 978-82-326-7230-1 (printed ver.)
ISBN 978-82-326-7229-5 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)