EPFL

Thèse n° 7699

Real-Time Nonlinear Model Predictive Control for Fast Mechatronic Systems

Présentée le 19 mai 2022

Faculté des sciences et techniques de l'ingénieur
Laboratoire d'automatique 3
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Petr LISTOV

Acceptée sur proposition du jury

Prof. A. Karimi, président du jury
Prof. C. N. Jones, directeur de thèse
Prof. E. Kerrigan, rapporteur
Prof. R. Schmehl, rapporteur
Prof. G. Ferrari Trecate, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

To strive, to seek, to find, and not to yield.
— Alfred Tennyson

To Olga...

Acknowledgements

I would like to express sincere gratitude to my advisor, Prof. Colin Jones, for giving me the opportunity to pursue my PhD at Laboratoire d'Automatique. I deeply appreciate the freedom of choosing research directions and academic courses that I enjoyed over the past several years. Whatever path I took Colin always provided encouraging support, guidance and had a whole bag of creative ideas. Finally, thanks to his talent for attracting funds I could buy as many embedded computers and build as many robots as a PhD student could possibly only dream about.

I would also like to say a big thank you to Prof. Eric Kerrigan, Prof. Roland Schmehl, Prof. Giancarlo Ferrari-Trecate and Prof. Alireza Karimi for taking their time to read the dissertation and for providing valuable feedback and criticism during the defence.

Visits to other labs within the AWESCO training network always were a big source of inspiration. I am very grateful to Prof. Moritz Diehl and members of his group Misha Katliar, Andrea Zanelli, Robin Verschueren, Dimitris Kouzoupis and Per Rutquist for hosting me at the University of Freiburg. It is during this visit that the architecture of what later would become PolyMPC was outlined after enjoyable discussions and brainstorms with the lab members. Many thanks to Prof. Roy Smith and Eva Ahbe for welcoming me at ETHZ during my second secondment. It was a great pleasure to discover the world of stochastic processes together. Overall, AWESCO was a great and enriching experience and I am grateful to all PI's that made it possible and to all my fellow student researchers.

This thesis would not be possible without fruitful collaborations with brilliant master students at EPFL. Thanks to Michael Spieler with whom we developed and deployed on a microcontroller an early version of PolyMPC. The flight experiments would hardly be possible without Johannes Waibel with whom we spent many hours together building, programming and flying airplanes. It was a pleasure to host Johannes Schwarz for his master thesis at the lab. His enquiring mind and deep questions helped me to grow as an expert in control. I truly enjoyed our collaboration that led to the novel stochastic optimal control algorithms development. Huge thanks to master students, Jean Pierre Allamaa, Gregoire Du Pasquier, and Arrival Racing

Acknowledgements

Team members, Pavel Savinkov, Rinat Shagiev and Max Kumskoy, for their hard work creating and testing the predictive controller for the autonomous racing car. Finally, thanks to Raphaël Linsen and Albéric Lajarte for unforgettable experience building a thrust vector controlled rocket prototype.

I will always remember the days spent at Laboratoire d'Automatic with great warmth. It was an honour to become an invited member of the "Empty head research group" along with Ehsan and Dr. Philippe Müllhaupt with whom we concealed the idea of the first commercial Vasco da Gamma university. I feel lucky to share the lab space with the nicest and smartest people: Georgios, Luca, Altug, Sanket, my old fellow programmer Harsh, Christophe, Sriniketh, Martand, Tafarel, Tomasz, Predrag, Sohail, Ivan and the younger generation: Emilio, Philippe, Clara, Paul, Yingzhao, Wenji, Yang and my new programming fellow Roland. A big thank you to Lloris who helped me with the abstract translation. I am grateful to Dr. Yuning Jiang, Dr. Tony Wood and Emilio for improving the writing quality of this thesis. Thanks to Dr. Christophe Salzmann for sharing his immense knowledge about network protocols and for his readiness to help virtually with any possible hardware and organisational question I could have. I thank Ruth, Eva, Margot and Nicole for providing the best administrative assistance and for managing my numerous trip and procurements.

I am very thankful to my former colleagues at RoboCV for introducing me to robotics and software engineering, particularly to Artur Emagulov and Dr. Ilya Nedelko. It was Ilya who initiated me into the field of numerical optimisation and helped to implement the first nonlinear solver. His fundamental understanding of algorithms, analysis and problem-solving skills influenced my decision to pursue a PhD degree. Many thanks to Artur for making me a better engineer but primarily for being a good friend all these years.

It is hard to find the right words to express all the gratitude to my family for their unconditional support and care. I will be eternally grateful to the two heroic women, my mother Arina and babushka Zinaida Aleksandrovna, who despite very hard times raised me and my brothers. I thank my stepfather Boris for his patience in helping me to prepare for university entrance exams but perhaps most importantly for fostering the right work attitude which I will carry through days.

At last but not least, I say thank you to my beloved wife Olga without whom I would never have started (and finished) this journey.

Lausanne, December 24, 2021 P. L.

Abstract

This thesis presents an efficient and extensible numerical software framework for real-time model-based control. We are motivated by complex and challenging mechatronic applications spanning from flight control of fixed-wing aircraft and thrust vector control drones to autonomous driving.

In the first part, we present *PolyMPC*, a novel C++ software framework for real-time embedded nonlinear optimal control and optimisation. A key feature of the package is a highly optimised implementation of the pseudospectral collocation method that exploits instruction set parallelism available on many modern computer architectures. Polynomial representation of the state and control trajectories allows the tool to be used as a standalone controller and as an efficient solver for low-level tracking controllers in hierarchical schemes. Another distinctive property of *PolyMPC* is that, unlike almost all other software tools for real-time optimal control, it does not employ code generation. Instead, we leverage the flexibility of templates and static polymorphism in C++ to optimise computations, statically allocate memory and dispatch algorithmic optimisation at compilation time. Algorithmically, the choice is made towards computational speed. For nonlinear problems, we combine a sequential quadratic programming (SQP) strategy with the alternating direction method of multipliers (ADMM) for quadratic programs (QP), which is especially favourable for embedded applications thanks to the low computational cost per iteration. The user is not limited to the available implementations and has the option of interfacing several other established QP and NLP solvers. *PolyMPC* has been tested on many embedded platforms and shown to provide better performance than other similar tools for numerical optimal control.

In the second part, the developed numerical methods and software are used to experimentally study optimisation-based control of airborne wind energy (AWE) systems. For this purpose, we designed and built a small-scale prototype of a single-line rigid-wing AWE kite which comprises an aircraft fitted with necessary sensors and computers and a fully autonomous ground station for tether control. The prototype serves as a research platform for studying flight navigation and control systems thanks to very flexible custom mission management and control software. We further develop a dynamic optimisation based methodology for

parameter identification and provide a validated flight simulator that matches well the real behaviour of the system. Finally, a model-predictive path following flight controller is designed and tested in real-world experiments.

The third part of the thesis is concerned with the application of real-time nonlinear model predictive control (NMPC) to autonomous driving at the limits of handling, which requires high sampling rates and robustness of the motion control system. We propose a dynamic optimization-based hierarchical framework for the local refinement of the racing lines that takes into account the nonlinear vehicle and actuator dynamics, adaptive tyre constraints, and the safety corridor around the initial path. The top layer receives a discrete obstacle-free local path computed by a coarse planner and transforms it into auto-differentiable look-up tables (LUT) for efficient continuous sampling. The *PolyMPC*-based nonlinear model predictive control algorithm computes an optimal trajectory that steers the race car to the path while respecting dynamic and actuator constraints. This continuous trajectory is then sampled by a fast discrete-time linear quadratic regulator (LQR) at the stabilisation layer. We provide rigorous processor-in-the-loop (PiL) benchmarks using validated high-fidelity simulation tools. Our solution provides an improvement in lap time over the current state of the art method and minimises interference of emergency stabilisation systems.

Separately, we investigated the problem of safe trajectory planning under parametric model uncertainties motivated by automotive applications. We use generalised polynomial chaos expansions for efficient nonlinear uncertainty propagation and distributionally robust inequalities for chance constraint approximation. Inspired by tube-based model predictive control, an ancillary feedback controller is used to control the deviations of stochastic modes from the nominal solution, and therefore, decrease the variance. Our approach reduces conservatism related to nonlinear uncertainty propagation while guaranteeing constraint satisfaction with a high probability. The performance is demonstrated on the example of a trajectory optimisation problem for a simplified vehicle model with uncertain parameters.

Key words: Nonlinear Model Predictive Control, Pseudospectral Methods in Optimal Control, Flight Control, Stochastic Optimal Control

Résumé

Cette thèse présente un logiciel efficace, extensible, et en temps réel pour le contrôle basé sur des modèles (MPC). Nous sommes motivés par des applications mécatroniques très complexes et exigeantes allant du contrôle de vol d'aéronefs à voilure fixe et du vecteur de poussée de drones à la conduite autonome.

Dans la première partie, nous présentons *PolyMPC*, un nouveau logiciel en C++ pour le contrôle et l'optimisation embarqué et en temps réel de systèmes non linéaires. Une de ses caractéristiques clés est une implémentation hautement optimisée de la méthode de collocation pseudospectrale qui exploite le parallélisme d'instructions disponible sur de nombreuses architectures informatiques modernes. Une représentation polynomiale des trajectoires du système (états et commandes) permet d'utiliser *PolyMPC* comme contrôleur autonome et comme guidage efficace pour des contrôleurs de suivi de bas niveau dans des schémas hiérarchiques. Une autre propriété notable de *PolyMPC* est que, contrairement à presque tous les autres logiciels de contrôle et optimisation en temps réel, il n'utilise pas de génération de code. A la place, nous tirons parti de la flexibilité des modèles et du polymorphisme statique en C++ pour optimiser les calculs, allouer la mémoire statiquement, et répartir l'optimisation algorithmique au moment de la compilation. Pour le choix des algorithmes, les décisions sont prises en fonctions de leur vitesse de calcul. Pour les problèmes non linéaires, nous combinons la stratégie de programmation quadratique séquentielle (SQP) avec la méthode des multiplicateurs à directions alternées (ADMM) pour les problèmes quadratiques (QP) qui est particulièrement intéressante pour les applications embarquées de par son faible coût de calcul par itération. L'utilisateur n'est pas limité aux implémentations disponibles et a la possibilité d'interfacer plusieurs autres solveurs pour des QP et d'autres programmes non linéaires (NLP) existants. *PolyMPC* a été testé sur de nombreuses plates-formes embarquées et s'est avéré offrir de meilleures performances que d'autres outils similaires pour le contrôle prédictif basé sur des modèles.

Dans la deuxième partie, les méthodes numériques et les logiciels développés sont utilisés pour étudier expérimentalement le contrôle et l'optimisation des systèmes aéroportés d'énergie éolienne (AWE). À cette fin, nous avons conçu et construit un prototype à petite échelle

d'un cerf-volant AWE à voilure rigide et à une seule ligne, qui comprend un aéronef équipé des capteurs et des calculateurs nécessaires et une station au sol entièrement autonome pour le contrôle de l'ancrage. Ce prototype sert de plate-forme de recherche pour l'étude des systèmes de navigation et de contrôle de vol grâce à un logiciel de gestion et de contrôle de mission sur mesure et très flexible. De plus, nous développons une méthodologie d'optimisation dynamique pour l'identification des paramètres et fournissons un simulateur de vol validé qui correspond bien au comportement réel du système. Enfin, un contrôleur de suivi de trajectoires de vol basé sur les prédiction du modèle est conçu et testé lors d'expériences réelles.

La troisième partie de la thèse s'intéresse à l'application en temps réel d'un contrôleur prédictif basé sur un modèle non linéaire (NMPC) à la conduite autonome d'un véhicules aux actions limitées qui nécessite des taux d'échantillonnage élevés et un système de contrôle de mouvement robuste. Nous proposons une architecture hiérarchique basée sur l'optimisation dynamique pour le raffinement des lignes de course locales, laquelle prend en compte les dynamiques non linéaires du véhicule et des actionneurs, les contraintes adaptatives des pneus et le corridor de sécurité autour du parcours initial. La couche supérieure reçoit un parcours local discret et sans obstacle calculé par un planificateur plus grossier et le transforme en tables de consultation (LUT) auto-différentiables pour un échantillonnage continu efficace. L'algorithme de NMPC, basé sur *PolyMPC*, calcule la trajectoire optimale pour diriger la voiture de course sur le parcours tout en respectant les contraintes du parcours et des actionneurs. Cette trajectoire continue est ensuite échantillonnée par un régulateur quadratique linéaire à temps discret (LQR) rapide au niveau de la couche de stabilisation. Nous fournissons des références précises pour comparer notre méthode à l'aide d'outils de simulation haute fidélité validés et en utilisant le procédé de processor-in-the-loop (PiL). Notre solution permet d'améliorer le temps au tour par rapport à la méthode de pointe actuelle et de minimiser les interférences des systèmes de stabilisation d'urgence.

Séparément, motivés par des applications automobiles, nous avons étudié le problème de la planification de trajectoires sûres lorsque des incertitudes paramétriques de modèle existent. Nous utilisons l'expansion polynomiale généralisée du chaos pour une propagation efficace de l'incertitude non linéaire et des inégalités robustes sous l'incertitude pour l'approximation des contraintes aléatoires. Inspiré du contrôle prédictif basé sur des tubes (tube-based MPC), un contrôleur de rétroaction auxiliaire est utilisé pour contrôler les écarts des modes stochastiques par rapport à la solution nominale et, par conséquent, diminuer la variance. Notre approche permet de réduire le conservatisme lié à la propagation de l'incertitude non linéaire tout en garantissant la satisfaction des contraintes avec une probabilité élevée. Sa performance est démontrée pour un problème d'optimisation de trajectoire pour un modèle de véhicule simplifié avec des paramètres incertains.

Mots clés : Contrôle prédictif de modèle non linéaire, méthodes pseudospectrales en contrôle optimal, contrôle de vol, contrôle optimal stochastique

Contents

Chapter 1

Introduction

1.1 Introduction and Contributions

The last decade has seen a surge of interest in optimisation-based solutions for real-time control and decision making for various robotic and mechatronic systems both in industry and in the research community. Initially developed to control slow chemical processes, model predictive techniques are becoming more and more important for mechatronic systems with faster dynamics, which includes industrial manipulators, advanced driving assistants and cruise and flight control systems to name a few. This has become possible thanks to active research in computational methods and software tailored to model predictive control problems. Despite recent advances, however, embedded deployment of nonlinear model-predictive control algorithms in real applications remains a challenging task. In this thesis, we present a novel computational and software framework for real-time embedded nonlinear optimal control that aims to reduce the complexity of embedded deployment. Furthermore, we demonstrate how this framework enables novel control applications in the areas of airborne wind energy and autonomous driving.

1.1.1 PolyMPC: software for fast embedded Nonlinear Model Predictive Control

The first part of the thesis concerns a novel C++ framework for fast embedded nonlinear optimisation and optimal control called *PolyMPC*. *PolyMPC* is a collection of carefully optimised algorithms for quadratic and nonlinear optimisation problems, pseudospectral transcription of optimal control problems and spectral projections. The tool is cross-platform and provides competitive or better performance compared to state-of-the-art open-source optimal control software. *PolyMPC* supports dense and sparse computations, and thanks to its design, enables

simple algorithm customisation. The remainder of this section provides details about this software tool.

In Part 1, we outline the major design choices and technical requirements for *PolyMPC*. The software employs static polymorphism and other modern C++ design patterns to allow the user cost-free customisation and extension of available algorithms. *PolyMPC* leverages the powerful template mechanism of C++ to analyse the problem data, allocate memory and dispatch optimisations at compile time. The software can adaptively allocate data statically or dynamically depending on the problem and available stack sizes or programmer preference. Once the data is allocated, *PolyMPC* makes sure that at no point temporary objects are created, which helps to improve runtime and avoids possible memory fragmentation, which is critical for embedded platforms.

The chapter also provides the necessary background on pseudospectral collocation methods for nonlinear optimal control. We argue that collocation techniques could be a good alternative to classical shooting schemes for real-time nonlinear predictive control as they typically require fewer optimisation variables for the same prediction horizon. Polynomial parametrisation also becomes advantageous for hierarchical control schemes since state and control trajectories can be cheaply and continuously re-sampled.

PolyMPC targets fast real-time systems and, therefore, prioritizes computational speed over robustness. The toolbox offers two custom implementations of the alternating directions method of multipliers (ADMM) for quadratic problems (QP) with slightly different splitting strategies. ADMM iterations are known to be computationally cheap and provide moderate accuracy results for reasonably scaled QPs. To enhance the robustness of the method, if necessary, the user is allowed to define a preconditioning routine. For nonlinear optimisation problems (NLPs) we implement a sequential programming (SQP) algorithm with a collection of regularisation routines and line search routines. For the generation of sensitivities, we use an existing forward mode automatic differentiation library and extend it with some commonly used vector functions. Finally, *PolyMPC* provides interfaces to some established QP and NLP solvers.

We conclude with some examples of nonlinear optimisation and optimal control problems, as well as benchmark against popular open-source and commercial tools for model predictive control.

1.1.2 Identification and Flight Control of Rigid-Wing Airborne Wind Energy Kites

The PolyMPC toolbox was initially developed to study experimentally if nonlinear model predictive control (NMPC) techniques can increase the robustness, accuracy and flexibility of

airborne wind energy (AWE) flight control systems. AWE systems are a new technology for energy harvesting from high altitude winds. Unlike conventional wind turbines, AWE systems do not require massive concrete and steel structures to hold the blades. Instead, the power is produced by a rigid or soft flying wing, often referred to as a kite, tethered to the ground station. One variant of the technology foresees a two-phase operation of an AWE kite: During the power generating phase, or traction phase, the kite moves away from the ground station in cross-wind, reeling out the tether and spinning the rotor of the generator. In the de-powering phase (or retraction phase) the kite glides back to the ground station at a low angle of attack, during which time the generator switches to act as a motor and reels in the loose tether. The difference between the traction and re-traction phases constitutes the power yield. Another AWE concept has the kite continuously flying a certain orbit in cross-wind, typically a circle. In this case, the power is generated by onboard turbines and transmitted to the ground station via a conductive tether.

Despite offering potential improvements over classical wind turbines, AWE systems are very complex from the state estimation, identification and control perspectives. To become practically viable, the control systems should possess robustness against unexpected wind disturbances while preserving the flying vehicle within the safe flight envelope. In this thesis, we studied how nonlinear optimisation-based controllers can achieve these requirements. To perform experimental studies, we designed and built a small-scale prototype of a fixed-wing AWE system and tested our algorithms in real flight conditions. For this purpose, a commercially available styrofoam glider was fitted with necessary sensors, an embedded computer, a low-level flight controller and a tether release mechanism to enable fully autonomous flight operation. The software includes the onboard mission control module, several flight controllers, a custom telemetry module, a ground station control unit and a lightweight command line interface for in-flight management of the system. In order to test the algorithms in the lab, a kite simulator with graphical visualisation was also implemented.

Chapter 3 presents the state of the art in control of rigid-wing AWE kites and Chapters 4 and 5 of the thesis present our work on modelling and identification of the prototype. We employ a control-oriented six degree of freedom (DoF) model with quaternion parametrisation of rotations, a linearised aerodynamic model and a tether that is modelled as a viscoelastic element. We rely on standard fixed-wing identification manoeuvres to collect data and the model parameter estimation problem is posed as a nonlinear dynamic optimisation problem and solved for multiple experiments using the PolyMPC toolbox.

The contribution of Chapter 6 is a real-time optimisation-based path following algorithm for full-body motion control of a kite. The algorithm allows for a flexible geometric path specification, does not require preliminary trajectory optimisation and explicitly encodes a safe flight envelope and we show the real-time capability of the NMPC implementation on an

embedded platform. To improve the robustness in real flight conditions, we implement and compare two hierarchical schemes: high-frequency feed-forward re-sampling and low-level tracking of the optimal trajectories. Finally, the control scheme is validated on real flight experiments.

1.1.3 Predictive Control for Autonomous Racing

The third part of the thesis explores real-time NMPC for motion control of a racing car at the limits of handling. The aim of the project was to develop an embedded optimisation-based controller to steer an electric vehicle in a realistic racing scenario while respecting dynamic safety bounds. The simulation and experimental parts of the project were carried out in collaboration with the Arrival Racing team in the context of the Roborace autonomous competition. We further investigated techniques to improve robustness and to ensure safe autonomous driving in the case of model parametric uncertainty.

Chapter 7 provides background on the mathematical modelling of a racing car both in Cartesian and curvilinear reference frames. Further, a nonlinear path following model predictive control algorithm for racing line tracking is presented and a PolyMPC-based implementation is adapted for a real sensing system and car hardware limitations. A fast re-sampling scheme with the existing low-level controller is implemented to enhance the robustness during the experimental racing sessions. Processor-in-the-loop simulation results with an industry-grade simulator and embedded automotive platform conclude the chapter.

In Chapter 8, a stochastic NMPC approach is proposed for the case when some of the model parameters cannot be established precisely or change quickly over time. Unknown parameters are modelled as random variables, and the model equations thus become stochastic differential equations. Polynomial chaos expansion (PCE) is proposed as a numerical tool for nonlinear uncertainty propagation, and safety chance constraints are approximated by distributionally robust inequalities. In order to reduce the conservatism of the stochastic approach, we proposed a novel optimal control problem formulation with a pre-stabilising controller. The performance of the stochastic NMPC algorithm is demonstrated using a simplified car model in simulation.

1.2 Collaborations

The first part of the thesis was done in collaboration with Michael Spieler, who wrote the first prototype of the SQP solver during his master thesis. The second part would be impossible without Johannes Waibel, who contributed a lot to the hardware and software development of

the AWE prototype, implemented a backup geometric controller and masterfully piloted the aircraft. The third part is the result of the fruitful collaboration of many people: Jean-Pierre Allamaa contributed to the curvilinear path following NMPC formulation, he also developed the PolyMPC interface to Simulink. We are grateful to Pavel Savinkov and other engineers from the Arrival Racing Team for providing us with the industrial-grade simulation tools and helping to deploy the NMPC controller on the embedded hardware. Johannes Schwarz has contributed to the stochastic NMPC simulation and proposed the idea of an ancillary controller for stochastic NMPC during his master thesis in the laboratory.

1.3 Publications

- JP. Allamaa, **P. Listov**, H. Van der Auweraer, C. Jones, S. Tong Duy, "*Real-Time Nonlinear MPC Strategy with Full Vehicle Validation for Autonomous Driving*", submitted to IEEE American Control Conference (ACC), (under review)
- **P. Listov***, J. Schwarz*, C. Jones, "*Stochastic Optimal Control for Autonomous Driving Applications via Polynomial Chaos Expansions*", submitted to Optimal Control Applications and Methods, (under review).
- R. Linsen*, **P. Listov***, A. Lajarte*, R. Schwan, C. Jones, "*Optimal Thrust Vector Control of an Electric Small-Scale Rocket Prototype*", submitted to IEEE International Conference on Robotics and Automation, 2022 (under review).
- Y. Jiang*, **P. Listov***, C. Jones, "*Block BFGS based Distributed Optimization for NMPC using PolyMPC*", European Control Conference (ECC), 2021.
- E.Ahbe, **P. Listov**, A. Ianneli, R. Smith, "*Feedback Control Design Maximizing the Region of Attraction of Stochastic Systems Using Polynomial Chaos Expansion*", 21st IFAC World Congress, 2020.
- **P. Listov**, C. Jones, "*PolyMPC: An Efficient and Extensible Tool for Real-Time Nonlinear Model Predictive Tracking and Path Following for Fast Mechatronic Systems*", Optimal Control Applications and Methods, 2020.
- **P. Listov**, T. Faulwasser, C. Jones, "*Nonlinear Model Predictive Path Following Control of a Fixed-Wing Single-Line Kite*", Book of Abstracts of the International Airborne Wind Energy Conference (AWEC), 2017.

*The authors contributed equally

Articles in preparation:

- **P. Listov***, JP. Allamaa*, G. Du Pasquier, J. Schwarz, P. Savinkov, C. Jones, "*Real-Time Nonlinear Model Predictive Control for Autonomous Racing*"
- **P. Listov***, J. Waibel*, C. Jones, "Identification and Predictive Control of an Airborne Wind Energy System"

Part I

PolyMPC: Software for Fast Embedded NMPC

Chapter 2

Software Design and Algorithms

2.1 Purpose and Design Goals

Optimal control aims at finding a control function that minimises (or maximises) a given criterion subject to system dynamics, typically governed by differential equations, and path constraints. A generic continuous-time Optimal Control Problem (OCP) is often written in the following form:

$$\underset{x(\cdot),u(\cdot),p}{\text{minimize}}\ J[x(t_0),u(\cdot),p] = \Phi[x(t_f),t_f,p] + \int_{t_0}^{t_f} L[x(\tau),u(\tau),\tau,p]\,d\tau \tag{2.1a}$$

$$\text{s.t. } \forall \tau \in [t_0,t_f] \tag{2.1b}$$

$$\dot{x}(\tau) = f(x(\tau),u(\tau),\tau,p),\ x(t_0) = x_0 \tag{2.1c}$$

$$g(x(\tau),u(\tau),\tau,p) \leq 0 \tag{2.1d}$$

$$\psi(x(t_f),t_f,p) \leq 0 \tag{2.1e}$$

where $\tau \in \mathbb{R}$ denotes time, $x(\tau) \in \mathbb{R}^{n_x}$ is the state of the system, $u(\tau) \in \mathbb{R}^{n_u}$ is the vector of control inputs and $p \in \mathbb{R}^{n_p}$ is the vector of parameters. The function $\Phi : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is the terminal cost function and $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is called the running cost. The system dynamics are given by the function $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$, $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_g}$ is the path constraint function, and finally $\psi : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_\varphi}$ is the terminal constraint function.

Due to the significant progress both in the theory and the development of numerical methods over the last decades, optimal control has become a mature technology with many applications in the aerospace, robotics, chemical and manufacturing industries.

Nonlinear Model Predictive Control (NMPC) is an advanced feedback control method that utilizes a dynamic model of a system to iteratively solve a finite time OCP and apply the optimal solution $u^*(x_0, t_0)$ in a receding horizon fashion. MPC theory is concerned with designing the functions Φ and φ such that the closed-loop system is stable and the recursive feasibility is guaranteed, i.e. a solution to (2.1) exists at every iteration. Practical deployment of NMPC algorithms still remains challenging due to the high computational demand of the method.

Numerical Methods for Optimal Control

A globally optimal control function for (2.1) can be recovered from the solution of the Hamilton-Jacobi-Bellman (HJB) equation [1, 2]. However, except for certain specific cases, a solution to the HJB equation is hard or impossible to find analytically. Therefore, in practice it is usually discretised and solved with algorithms based on the dynamic programming recursion, which are known to suffer from the so called curse of dimensionality, i.e. computational complexity grows exponentially with the problem dimension.

Numerical approaches for computing local OCP solutions are usually divided into two main categories: the indirect and direct methods. The indirect approach requires a solution of a boundary value problem for a system of differential algebraic equations (DAE) resulting from the necessary optimality conditions [3, 4]. The main disadvantage of these methods is a high sensitivity to the unspecified boundary conditions [4]. Hence, the reliability of the numerical solution depends on a good initial guess, which is not always available. Another difficulty limiting the practical application of the indirect methods is the inability to treat inequality constraints efficiently, since the sequence of constrained and unconstrained arcs should be guessed prior to solving the problem. The direct, or mathematical programming methods, on the other hand, seek to transform a continuous time OCP into a finite dimensional nonlinear program (NLP) by employing certain parametrizations of the state and control functions. The ability to handle inequality constraints efficiently, as well as the active development of nonlinear programming algorithms and software in the last several decades, has made the direct methods a favoured choice for real-time optimal control.

Direct OCP transcription methods differ in their parametrisation of control and state trajectories, as well as integration methods for differential equations that govern dynamical systems. In the single-shooting, or sequential approach, the prediction horizon is typically evenly divided into segments. Within each time segment, the control is assumed to be constant (zero-order hold assumption), and explicit integration techniques are used to eliminate the state variables. This approach is primarily applied to the systems governed by well-conditioned stable differential equations. For stiff and unstable dynamical systems, single-shooting often

is not efficient due to insufficient numerical stability of explicit integration techniques and high sensitivity of the solution to the control inputs [5].

Unlike single-shooting, the direct multiple-shooting approach developed by Bock [6, 7] does not eliminate the state variables, and therefore can benefit from available implicit integration schemes, which allows for the handling of unstable nonlinear systems. Efficient numerical implementations exploit separability of the cost function and system constraints inherent to the method [8].

Pseudospectral collocation is another numerical technique that employs a polynomial approximation of the state and control trajectories to discretize continuous-time OCPs. Initially developed for solving differential equations, the collocation method was adopted for optimal control problems by Biegler [9] and further analysed and developed by Ross [10], Benson [11] and Huntington [12] in the early 2000s. One distinctive feature of this method is that the solution of a system of differential equations and optimisation of the cost function happen simultaneously. It also inherits the stability properties of the collocation method for ODEs, and therefore is particularly useful for stiff nonlinear and unstable systems. Furthermore, for some smooth problems the method can exhibit very fast, or spectral, convergence [13].

Software for Optimal Control

There exist a number of software tools that allow one to formulate and solve optimal control problems. Among those based on pseudospectral collocation methods probably the most well-known is the commercial package *GPOPS-II* [14]. Originally developed in Matlab, *GPOPS-II* has been very recently released in C++. The software employs the Gauss pseudospectral method with Legendre collocation nodes to solve multi-phase OCPs using interfaces to nonlinear solvers *IPOPT* [15] or *SNOPT* [16]. An important feature of the tool is the hp mesh refinement algorithm that allows one to adjust the order of interpolating polynomials and the length of subintervals to achieve a high accuracy of the optimal solution, even for discontinuous systems. This accuracy comes at the expense of computational resources, which renders the use of this tool impossible in real-time applications. An open-source alternative to *GPOPS-II* in Matlab is *ICLOCS2* [17], which additionally offers multiple-shooting and direct collocation methods, and an interface to a sequential programming code *WORHP* [18].

However, these packages rely on *Matlab*, which is neither free nor available on every platform. The software tool *PSOPT* [19] overcomes this limitation, as it is a free and open-source C++ package that implements pseudospectral discretization on Legendre-Gauss-Lobatto (LGL) and Chebyshev-Gauss-Lobatto (CGL) nodes, along with the p mesh refinement method. Similar to *GPOPS-II* and *ICLOCS*, it can handle multi-stage problems and relies on the *IPOPT* nonlinear

optimization solver. A possible inconvenience for control systems developers is the necessity to input model equations, cost and constraint functions in the scalar form, which can be very difficult and error prone for complex systems. Another free and open-source alternative is a *Python* package *mpopt* [20] which was co-developed by the author of the thesis. It features several collocation schemes, *h* grid refinement and problem formulation using the symbolic framework *CasADi* [21]. Some commercial software packages for optimal control employing pseudospectral collocation methods include *DIRCOL* [22], *DIDO* and *SOS* [23].

Another popular software for direct optimal control that does not rely on spectral methods is *ACADO* [24], which is a highly optimized C++ framework that implements single- and multiple-shooting discretization combined with various explicit and implicit integration routines, as well as a custom sequential programming (SQP) solver. For OCP modelling, *ACADO* implements a symbolic algebra system that allows for automatic differentiation of the user-provided expressions and generation of efficient tailored C code for solving the problem in real-time. The successor of *ACADO*, *acados* [25] does not provide a modelling language but rather requires the user to allocate data and implement cost, constraints, system dynamics and their sensitivities with the respect to the state and control in the C language directly. This approach allows for better modularity of the software, since it avoids the code generation step, but demands significantly larger coding effort and is prone to errors in memory management. To simplify the process, *acados* offers interfaces to *Matlab* and *Python* where OCPs can be formulated using the symbolic framework *CasADi* [21] and be exported back to the C language.

The previously mentioned software *CasADi* is not specific to optimal control applications, but provides a set of tools and interfaces to various numerical codes allowing the implementation of different dynamic optimisation algorithms. At the core of the toolbox is a symbolic framework that transforms user-specified scalar and matrix expressions into topologically sorted directed acyclic graphs which allows the reuse of repeating expressions and the optimisation of memory usage. Such a graph representation helps to efficiently evaluate expressions and their sensitivities using forward or backward mode automatic differentiation. Therefore, the software serves as a powerful prototyping instrument that does not require a lot of coding effort or skills from the user. As a result of this flexibility and generality, however, *CasADi* does not typically posses the same level of computational performance as more specialised real-time optimal control tools. For embedded deployment, *CasADi* is able to generate C code for the large subset of available functionality including a custom SQP solver.

PolyMPC

One can conclude that software for real-time and embedded optimal control follows one of the two design patterns. In the first one, a high level modelling language is used to describe an OCP

such that a code generation engine can create a self-contained low level code, typically in the C language. This approach is particularly attractive for quick NMPC prototyping and testing. On the downside, the code generation obscures the algorithmic components from the control or optimisation expert, i.e. it is hard to single out or extend the algorithms. Thus, the user is limited to a set of algorithms already implemented in a package. The modular design pattern, conversely, allows more flexibility and full control of the code at the expense of programming effort from the user. Most of the packages pursuing modular design, however, are highly optimised for a particular structure of discretisation or optimisation methods developed by the authors and therefore, adding an algorithm that does not fit into a given structure is difficult without significant loss of performance. Furthermore, procedural languages like C do not favour generic programming and many levels of abstractions.

This section presents *PolyMPC*, an open-source C++ library for real-time embedded nonlinear optimal control that combines high computational performance, modularity and a simple and intuitive user interface. *PolyMPC* is a lightweight collection of optimised and loosely coupled tools for nonlinear optimisation and optimal control which includes several QP and NLP solvers, quasi-Newton Hessian approximation and Hessian regularisation methods, a polynomial interpolation and approximation module and an implementation of the Chebyshev pseudospectral collocation method, a continuous algebraic Riccati equation (CARE) solver, linear quadratic regulator (LQR), and more. The design goals of the project that guided the development are:

Performance: the software is designed to create real-time NMPC algorithms for fast mechatronic systems that potentially can run on low-power computation platforms. Therefore, it is important to avoid unnecessary memory allocations, copying and expensive calls of virtual methods while exploiting vectorisation capabilities of modern processors. To allow for embedded deployment on microcontrollers without an operating system the software supports fully static memory allocation.

Usability: a key factor in the development is to create an efficient implementation of state-of-the-art methods while preserving simplicity and a user-friendly interface. The use of modern frameworks for linear algebra and automatic differentiation allows the user to formulate problems using intuitive vector notation.

Modularity: each algorithmic component of *PolyMPC* should be usable independently and interaction of the components should happen at zero computational cost. Moreover, interfaces and implementations should be logically separated, i.e. for each family of algorithms (QP, NLP solvers, OCP discretisation

etc) a unifying interface should be provided.

Extensibility: the tool is designed in a manner that allows the user to formulate OCPs as well as easily utilize and modify the building blocks of the algorithm by changing or adding a corresponding implementation. It is further important that the code is kept concise and clear.

Key design choices

The choice of the programming language is crucial to achieve the aforementioned design goals. The object-oriented (OO) paradigm in C++ facilitates the development of a cleaner, modular and extensible code. At the same time, modern C++ follows the *zero-overhead principle* that states [26, 27]:

1 "You don't pay for what you don't use."

2 "What you do use is just as efficient as what you could reasonably write by hand."

This means, it is possible in principle to write a program using abstractions that will be at least as efficient as with a low level code. It is further important to note that *C++* is a quickly evolving language with the standard being updated every three years with new concepts and library features. While the newest language standards possess some attractive features such as an improved aligned memory allocation, concurrency, type deduction and more convenient conditional compilation, not all vendors of operating systems and compilers support them yet. Therefore, at the moment, in the *PolyMPC* implementation we strictly adhere to the *C++11* standard, which is supported on the majority of computational platforms that can run *C++* code.

A very important example of this principle is the curiously recurring template pattern (CRTP) [28] that is widely used in *PolyMPC* to abstract interfaces and implementation methods, to customise the existing algorithms, and to supply user-defined functions for OCP and NLP formulations. CRTP achieves a similar effect to the virtual function call system, but without memory and performance overheads related to dynamic polymorphism: creating and storing virtual method tables, runtime method lookups and so on. The compile-time dispatching of the functions also often results in a better optimised machine code since the program flow is transparent to the compiler. The downside of this approach is the impossibility of changing function implementations at runtime, but since we target embedded applications and prioritise computational speed over flexibility, the static polymorphism remains well motivated.

Another important paradigm in the language to implement static polymorphism is called function templates [26]. In procedural languages like *C*, if one needs a function to operate with different data types, it is necessary to define this function with all possible combinations of data types, or to overload it. The template mechanism provides a more elegant solution to this problem, where yet undefined data types serve as template parameters. Moreover, the function is only fully defined when it becomes instantiated with particular data types. This feature helps to avoid unnecessary computational overheads. For example, if only a numerical value of a user-defined function is required by an algorithm, it can be called with one of the standard floating point types, e.g. *double* or *float*. However, if an algorithm requires, for instance, derivatives of the function, it can be called with a special *AutoDiff* scalar type which carries the sensitivity information, and therefore is more computationally expensive. The compiler will generate different machine code for these two different function calls. Function templates also allow the parametrisation of the complete workflow of an optimisation or MPC algorithm by a preferred scalar type, e.g. it is easy to switch between single-precision or double-precision floating point numbers without any changes to the code.

It is possible to further speed-up computations by specifying a custom implementation for a particular data type through a so called partial class (or full) template specialisation mechanism. In *PolyMPC* this approach is employed to optimise sensitivity computations of some vector-valued functions where first and second order sensitivities can be found in closed form or can be vectorised efficiently, for instance, affine functions or quadratic forms. The compiler performs a name lookup to detect if a functor was called with a particular type and replaces a generic implementation with an optimised one.

Inlining is another feature in *C++* that reduces the overhead associated with function calls. If the *inline* specifier is used ahead of a function declaration, the compiler will expand the function body at the point where it is called. This technique is typically used to optimise frequently used small functions. In general, an extensive use of templates and inline functions leads to the generated machine code growing in size and can cause losses in computational performance. Therefore, in *PolyMPC* we run benchmarks to determine which functions should be inlined.

Linear algebra

Linear algebra (LA) is a crucial component of any optimisation or optimal control algorithm. For LA, *PolyMPC* relies on a fast and lightweight *C++* template library *Eigen* [29] that possesses several interesting and important features that will be briefly highlighted in the following. The library exploits the CRTP idiom to implement the expression templates technique [30, 29], which allows one to avoid creating temporary objects, reduces the number of memory ac-

cesses and performs loop unrolling if necessary. Internally, *Eigen* performs aligned memory allocation and explicitly vectorises computations using hardware-specific single instruction, multiple data (SIMD) intrinsics (*AVX, SSE, SSE2, Neon, AltiVec*). Besides high computational performance *Eigen* provides a user-friendly interface for matrix expression manipulations, supports dense and sparse matrix formats, direct and iterative linear system solvers, polynomial and geometric modules and a basic automatic differentiation module. To summarise, the particular LA library was chosen for its high performance, active maintenance by the research and industrial communities and its good documentation. All these factors also contribute to usability and extensibility of the *PolyMPC* package itself.

Memory management

Eigen provides the user with an option of heap or stack allocation of dense matrix objects. For computational performance reasons and keeping in mind embedded deployment, *PolyMPC* requires that the dimensions of the objects in the generic OCP (2.1) or the dimensions of the optimisation variables and the number of constraints for optimisation problems are known at compile time. Provided problem dimensions, the software can estimate the sizes and the necessary amount of memory for every object that will be created during the optimisation procedure and decide whether to allocate each of them statically or on the heap depending on the available stack size. For embedded applications, if the provided stack size is insufficient, the compilation will fail with a corresponding warning. Otherwise, larger objects will be allocated on the heap. This approach also allows one to perform compile-time checks on the consistency of matrix expressions and the avoid invalid memory accesses at runtime. Static memory allocation for sparse matrices is impossible since, in general, the sparsity pattern of a matrix can change during computations, which will then require memory re-allocation. Whenever possible, *PolyMPC* will precompute the sparsity pattern of matrices and perform block-updates using the low-level column compressed storage (CCS) format without memory re-allocation.

CasADi

For historical reasons, some functionality in *PolyMPC* is duplicated using the *CasADi* framework. This is useful for validating the sensitivity computations and solutions to nonlinear problems. For embedded deployment, this interface is inefficient and we recommend to use the *Eigen* interface instead.

Summary

This section briefly introduced the key implementation and programming concepts that were used in *PolyMPC* development. We make extensive use of the static polymorphism paradigm and templates in *C++* to achieve high computational performance while preserving the modularity of each algorithmic module in the software. Our memory management approach enables embedded deployment on systems where dynamic memory allocators are not available. In the next two sections, we provide some details on the algorithmic modules currently implemented in *PolyMPC*.

2.2 Pseudospectral Methods for Optimal Control

The global Gauss pseudospectral method with application to some optimal problems in aerospace was proposed in [11]. The authors established the equivalence between optimality conditions on the NLP and the discretised OCP, and proposed a method for co-state estimation. This work was extended in [12] for Legendre and Radau collocation schemes, as well as for local, or piece-wise polynomial, approximation of state and control trajectories. Superior convergence properties of the method were observed compared to other OCP discretisation techniques for smooth problems. The method has since become widely popular to solve trajectory optimization problems that arise in aerospace applications [31, 10] since it provides high numerical accuracy for long prediction horizons. The pseudospectral transcription of these problems results, however, in large-scale non-convex NLPs that can take minutes to solve for complex systems using modern software. Recent advances in the computational performance of available hardware make it possible to apply pseudospectral collocation to generate real-time model predictive feedback control laws for complex and highly nonlinear systems such as unmanned aerial vehicles, mobile robots, self-driving cars, and manipulators.

We suggest that the pseudospectral collocation method is suitable for real-time model predictive control applications for several reasons. First, as was observed in the literature before, the method can exhibit fast, or even spectral, convergence in practice for smooth problems. Spectral convergence implies that the error of approximating a solution to a problem decays exponentially with the order of approximation polynomials, i.e. very few parameters are needed to find an accurate approximation of the solution. Second, for the shooting methods that are typically used in real-time NMPC codes, one needs to find a trade-off between the computation time and number of shooting intervals as an insufficient number of intervals can lead to suboptimality and inaccurate representation of the trajectories. This renders the application of these methods for higher dimensional systems or for problems where longer prediction horizon is desirable challenging. Thanks to the rich parametrisation of control

and state trajectories, collocation methods require fewer parameters and smaller NLPs. Additionally, due to the polynomial parametrisation, state and control trajectories can be very efficiently resampled by a low-level controller in hierarchical control schemes. To the best of the author's knowledge, there is no openly available software that allows for real-time embedded deployment of pseudospectral collocation based controllers. The contribution of this chapter, therefore, is a highly optimised implementation of the Chebyshev pseudospectral collocation method.

In the following, we provide some preliminaries on the pseudospectral collocation method, describe the Chebyshev collocation method implemented in *PolyMPC* and discuss several issues that one can face in practice when applying such a method.

Pseudospectral Collocation Method

The software is designed to solve generic nonlinear optimal control problems of the form (2.1). An example of using Chebyshev pseudospectral transcription of the optimal control problem can be found in [32]. In this section we provide the basic relations from spectral approximation methods and extend previous work to the case of local approximation of the domain.

Approximation of Differential Constraints

Assuming the approximate solution for the state and control trajectories are represented by an appropriately chosen set of basis functions $\varphi_k(\cdot)$:

$$\begin{aligned} x(t) \approx x^N(t) = \sum_{k=0}^{N} x_k \varphi_k(t) \\ u(t) \approx u^N(t) = \sum_{k=0}^{N} u_k \varphi_k(t) \end{aligned} \tag{2.2}$$

the collocation approach requires that equation (2.1c) is satisfied exactly at a set of collocation points $t_j \in (t_0,\ t_f)$:

$$\frac{dx^N}{dt} - f(x^N, u^N, t_j, p)\Big|_{t=t_j} = 0,\ j = 1, ..., N \tag{2.3}$$

With some initial conditions: $x^N(t_0) = x_0$. Formulas (2.2) represent a function expansion in the nodal basis, where x_k is the value of $x(t)$ at the node t_k and φ_k denotes a specific cardinal

basis function, which satisfies the condition: $\varphi_j(t_i) = \delta_{ij},\ i,j = 0,...,N$, where:

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{2.4}$$

A common choice of the cardinal basis in pseudospectral methods is the characteristic Lagrange polynomial of order N. This means that each basis function reproduces one function value at a particular point in the domain. The general expression for these polynomials is given by:

$$\varphi_k(t) = \prod_{j,\ j\neq k}^{N} \frac{(t - t_j)}{(t_k - t_j)},\ k = 0,...,N \tag{2.5}$$

Using the approximations above one can compute derivatives of the state trajectory at the nodal points:

$$(x^N(t_j))' = \sum_{k=0}^{N} x_k \dot{\varphi}_k(t_j) = \sum_{k=0}^{N} x_k D_{jk} \tag{2.6}$$

where D_{jk} are elements of the interpolation derivative matrix $\mathbf{D}$. And the approximate solution to the (2.1c) satisfies a system of algebraic equations:

$$\sum_{k=0}^{N} x_k D_{jk} = f(x_j, u_j, t_j, p), \quad j = 0,\ldots,N \tag{2.7}$$

Chebyshev Collocation Method

A careful choice of collocation points is crucial for numerical accuracy and convergence of the collocation method. An equidistant spacing of the collocation nodes is known to cause an exponential error growth of the polynomial interpolation near the edges of the domain [33, 34, 35, 36] an effect called Runge's phenomenon. Initially developed for solving partial differential equations in fluid dynamics, the Chebyshev spectral method provides an elegant solution to this problem. The idea is to project the solution of (2.1c) onto the set of Chebyshev polynomials $T_k(x)$, that are orthogonal on the interval $[-1,1]$ with respect to the weight function $\omega(x) = 1/\sqrt{1-x^2}$:

$$\int_{-1}^{1} T_n(x)\, T_m(x) \omega(x)\, dx = 0,\ n \neq m \tag{2.8}$$

So in addition to (2.2) the state and control trajectories can be expressed in terms of Chebyshev transform coefficients, or in the nodal basis:

$$\begin{aligned} x^N(t) &= \sum_{k=0}^{N} \tilde{x}_k T_k(t) \\ u^N(t) &= \sum_{k=0}^{N} \tilde{u}_k T_k(t) \end{aligned} \tag{2.9}$$

Where $\tilde{x}_k$ and $\tilde{u}_k$ are spectral coefficients of the Chebyshev transform. Representation (2.9) is employed in numerical methods based on the weak formulation, for instance the Chebyshev-Tau approach. In the context of collocation this representation is useful to speed-up the computation of the approximate derivative (2.6) as will be discussed later. The k-th order Chebyshev polynomial can be expressed as:

$$T_k(t) = \cos(k \cdot \arccos(t)) \tag{2.10}$$

Another feature of Chebyshev polynomials that makes them computationally attractive compared to other Jacobi polynomials is that collocation points have explicit formulas. In the *PolyMPC* we employ the most commonly used set of $N+1$ Chebyshev Gauss-Lobatto (CGL) points:

$$t_j = \cos\left(\frac{\pi j}{N}\right), \; j = 0, ..., N \tag{2.11}$$

Using Chebyshev polynomials in the collocation scheme is not efficient because they do not satisfy the cardinality condition (2.4), and therefore we will be using CGL nodes, which allow us to avoid Runge's phenomenon, in combination with Lagrange polynomials. Equipped with the set of collocation points, it is possible to compute the $(N+1) \times (N+1)$ interpolation derivative matrix $\mathbf{D}$ [34]:

$$D_{jk} = \begin{cases} \frac{c_j}{c_k} \frac{(-1)^{j+k}}{t_j - t_k} & j \neq k \\ -\frac{t_k}{2(1-t_k^2)} & 1 \leq j = k \leq N-1 \\ \frac{2N^2+1}{6} & j = k = 0 \\ \frac{2N^2+1}{6} & j = k = N \end{cases} \tag{2.12}$$

where

$$c_i = \begin{cases} 2 & i = 0 \; or \; N \\ 1 & \text{otherwise} \end{cases} \tag{2.13}$$

To account for the cases when the dimension of the state space n is larger than one, we will write the "composite" differentiation matrix $\mathbb{D}$ as the Kronecker product between the

$(N+1)\times(N+1)$ interpolation derivative matrix $\mathbf{D}$ and the $n\times n$ identity matrix:

$$\mathbb{D} = \mathbf{D} \otimes \mathbb{I}_{n\times n},\ \mathbb{D} \in \mathbb{R}^{(N+1)n\times(N+1)n} \tag{2.14}$$

It is worth noting that obtaining derivative approximates with matrix multiplication requires $2N^2$ operations. For large order polynomial approximation N one can benefit from the Fast Chebyshev Transform (similar to the Fast Fourier Transform) that has a lower computational complexity of $5N(\log_2 N+2)$. For this approach, one has to differentiate the function approximation in Chebyshev transform space and then transform these spectral derivatives back to physical space.

In most cases the time domain of the optimal control problem does not coincide with the interval [-1, 1] where the orthogonality of Chebyshev polynomials is preserved. Therefore, the time and system dynamics equations must be transformed accordingly:

$$\begin{aligned} t &= \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2},\ \tau \in [-1,1] \\ \dot{x}(\tau) &= \frac{t_f - t_0}{2} f(x(\tau), u(\tau), \tau, p) \end{aligned} \tag{2.15}$$

Putting these expressions in compact matrix form we obtain:

$$\begin{aligned} &\mathbb{X} = [x_N; \cdots, x_j; \cdots, x_1; x_0]^T = [x(1); \cdots; x(t_j); \cdots; x(t_1); x(-1)] \\ &\mathbb{U} = [u_N; \cdots, u_j; \cdots, u_1; u_0]^T = [u(1); \cdots; u(t_j); \cdots; u(t_1); u(-1)] \\ &\mathbb{F}(\mathbb{X}, \mathbb{U}, \{\tau_j\}, p) = \frac{t_f - t_0}{2}[f(x_N, u_N, 1, p); \cdots; f(x_j, u_j, \tau_j, p); \cdots; f(x_0, u_0, -1, p)] \end{aligned} \tag{2.16}$$

Here (;) denotes the vertical concatenation, and the CGL points are ordered backwards in time following the convention from the classic numerical literature. Finally (2.2) can be written as a system of nonlinear algebraic equations:

$$\mathbb{D}\mathbb{X} - \mathbb{F}(\mathbb{X}, \mathbb{U}, \{\tau_j\}, p) = 0 \tag{2.17}$$

Cost Function Approximation

The Mayer term of the cost function (2.1a) can be trivially approximated at the last collocation point:

$$\Phi[x(\tau_f)], \tau_f, p = \Phi[x_N, 1, p] = \Phi[x(1), 1, p] \tag{2.18}$$

For the Lagrange term approximation using the CGL set of points it is convenient to utilize the

Clenshaw-Curtis quadrature integration scheme, as suggested in [33]. The method allows one to find a set of weights $\{\omega_j\} \in \mathbb{R},\ j = 0, \cdots, N$ such that the following approximation is exact for all polynomials $p(t)$ of order less than N and corresponding weight function $\omega(t)$:

$$\sum_{j=0}^{N} p(t_j)\omega_j = \int_{-1}^{1} p(t)\omega(t)dt,\ p(t) \in \mathbb{P}_N \tag{2.19}$$

Where $\mathbb{P}_N$ is the space of polynomials of degree less or equal than N and $p(t)$ is a Chebyshev approximation of the integrated function:

$$L(x(t), u(t), t, p) \approx p(t) = \sum_{k=0}^{N} a_k T_k(t) \tag{2.20}$$

The symmetric quadrature weights are given by:

$$\begin{aligned} &\omega_0 = \omega_N = \frac{1}{N^2-1}, \text{ for even N} \\ &\omega_j = \omega_{N-j} = \frac{2}{N}\left[1 - \sum_{k=1}^{N/2-1} \frac{2}{1-4k^2}\cos\left(\frac{2\pi jk}{N}\right) - \frac{1}{1-N^2}\cos\left(\frac{N\pi j}{N}\right)\right],\ j = 1, \cdots, N/2 \end{aligned} \tag{2.21}$$

and

$$\begin{aligned} &\omega_0 = \omega_N = \frac{1}{N^2}, \text{ for odd N} \\ &\omega_j = \omega_{N-j} = \frac{2}{N}\left[1 - \sum_{k=0}^{\lfloor (N-1)/2 \rfloor} \frac{2}{1-4k^2}\cos\left(\frac{2\pi jk}{N}\right)\right],\ j = 1, \cdots, \lfloor N/2 \rfloor \end{aligned} \tag{2.22}$$

Therefore the cost function can be approximated with the following formula:

$$J(x(t), u(t)) \approx \Phi[x_N, 1, p] + \frac{t_f - t_0}{2} \sum_{j=0}^{N} L(x_j, u_j, \tau_j, p)\omega_j \tag{2.23}$$

Constraint Discretization

The constraints defined in (2.1d) are simply enforced at the collocation points:

$$\begin{aligned} &g(\mathbb{X}, \mathbb{U}, \{t_j\}, p) = [g(x_N, u_N, t_f, p); \cdots; g(x_j, u_j, t_j, p); \cdots; g(x_0, u_0, t_0, p)] \leq 0 \\ &\underline{X} \leq \mathbb{X} \leq \overline{X},\ \underline{X} = \mathbb{1}^{N+1} \otimes \underline{x},\ \overline{X} = \mathbb{1}^{N+1} \otimes \overline{x} \\ &\underline{U} \leq \mathbb{U} \leq \overline{U},\ \underline{U} = \mathbb{1}^{N+1} \otimes \underline{u},\ \overline{U} = \mathbb{1}^{N+1} \otimes \overline{u} \end{aligned} \tag{2.24}$$

And similarly for the terminal constraints (2.1e):

$$\psi(x_N, t_f, p) = \psi(x_N, t_f, p) \leq 0 \tag{2.25}$$

Multiple Interval Extension

The Chebyshev nodes are clustered on the edges of the collocation interval, and therefore sometimes it is desired to split the entire domain to achieve a more uniform distribution of collocation points. This is particularly important when dealing with discontinuous cost or dynamics functions.

The software allows one to split the time horizon of the optimal control problem into S equal subintervals with N collocation points in each interval. Vectors of collocation coefficients now

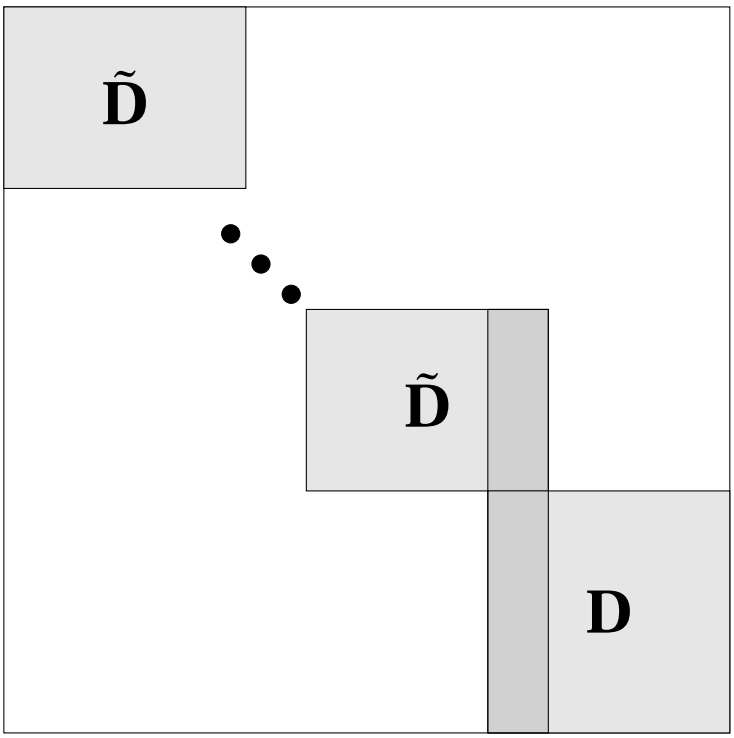


Figure 2.1 – Structure of the composite differentiation matrix $\mathbf{D}_c$ for multiple interval CGL collocation scheme.

have the form:

$$\begin{aligned} \mathbb{X} &= [x_N^{S-1}; \cdots; x_j^s; \cdots; x_1^1; x_N^0; \cdots; x_1^0; x_0^0] \in \mathbb{R}^{(SN+1)n} \\ \mathbb{U} &= [u_N^{S-1}; \cdots; u_j^s; \cdots; u_1^1; u_N^0; \cdots; u_1^0; u_0^0] \in \mathbb{R}^{(SN+1)m} \end{aligned} \tag{2.26}$$

Here the superscript denotes the index of an interval and the subscript the index of a colloca-

tion point within this interval.

The structure of the composite Chebyshev Gauss-Lobatto differentiation matrix $\mathbf{D}_c$ is presented in Figure 2.1. Here $\mathbf{D}$ is computed according to (2.12), $\tilde{\mathbf{D}} = \mathbf{D}[0:N-1;0:N]$. The last row is removed to enforce the continuity of the state trajectory at the splitting points- the end point of interval s is the starting point of interval $s+1$: $x_N^s = x_0^{s+1}$. Also the first and the last columns of neighbouring matrices are aligned. Again the full differentiation matrix is the Kronecker product between the composite derivative matrix and the identity matrix of size $n \times n$:

$$\mathbb{D} = \mathbf{D}_c \otimes \mathbb{I}_{n\times n},\ \mathbb{D} \in \mathbb{R}^{(SN+1)n\times(SN+1)n} \tag{2.27}$$

The collocation equations become:

$$\begin{aligned} \mathbb{F}(\mathbb{X},\mathbb{U},\{\tau_j\},p) &= \frac{t_f - t_0}{2S}[f(x_N^{S-1},u_N^{S-1},\tau_j^{S-1},p),\cdots,f(x_j^s,u_j^s,\tau_j^s,p),\cdots,f(x_0^0,u_0^0,\tau_j^0,p)]^T \\ \mathbb{D}\mathbb{X} - \mathbb{F}(\mathbb{X},\mathbb{U},p) &= 0 \end{aligned} \tag{2.28}$$

The cost function is computed in the following way:

$$J(\mathbb{X},\mathbb{U},p) = \Phi[x_N^{S-1}] + \frac{t_f - t_0}{2S}\sum_{s=0}^{S-1}\sum_{j=1}^{N} c_j L(x_j^s,u_j^s,\tau_j^s,p)\omega_j + \frac{t_f - t_0}{2S}L(x_0^0,u_0^0,\tau_0^0,p)\omega_0 \tag{2.29}$$

where

$$c_j = \begin{cases} 2 & j = N \\ 1 & otherwise \end{cases} \tag{2.30}$$

Nonlinear Optimization Problem

Using (2.28) and (2.29) the optimal control problem (2.1) can be written as a nonlinear optimization problem:

$$\begin{aligned} &\underset{\mathbb{X},\mathbb{U},p}{\text{minimize}}\ J(\mathbb{X},\mathbb{U},p) \\ &\text{subject to:} \\ &\mathbb{D}\mathbb{X} - \mathbb{F}(\mathbb{X},\mathbb{U},p) = 0 \\ &g(\mathbb{X},\mathbb{U},p) \le 0 \\ &\underline{X} \le \mathbb{X} \le \overline{X} \\ &\underline{U} \le \mathbb{U} \le \overline{U} \\ &\underline{p} \le p \le \overline{p} \\ &x_0^0 = x(t_0) \end{aligned} \tag{2.31}$$

This optimisation problem can be solved using generic nonlinear optimisation algorithms available in *PolyMPC*, which will be outlined in the next section.

Discussion

It is important to note that unlike in [14] or [19] the full set of collocation equations along with the initial condition are considered, as suggested by [37] and [38]. This is essential for real-time NMPC applications since one is most interested in the value of the optimal control signal at the initial time. Otherwise, if the starting point is left uncollocated as is the case for the Gauss or one of the Gauss-Radau schemes, an extrapolation of the optimal control function is required to find its value at the t_0 time instant. In the case when the final point is uncollocated, one has to use quadrature formulas in order to compute the terminal state.

The chosen approach is also different from standard collocation techniques where usually the last n or the first and last n equations in (2.28) are replaced by specific boundary conditions. This means that the system of equality constraints in (2.31) is overdetermined for some fixed vector $\mathbb{U}$, and therefore cannot be simulated forward and can only be used in the optimal control problem context. Moreover, since the $\mathbb{D}$ matrix is singular, it is harder to determine whether the linearized constraints would satisfy the linear independence constraint qualification (LICQ).

In general, even if the LICQ is met, finding a solution to (2.28) can be challenging for complex nonlinear and stiff systems especially in the context of nonlinear optimisation problems. One possible solution for this problem is to apply a continuation strategy as in [39], i.e. the original system (2.28) is replaced by a simpler system of equations $\tilde{\mathbb{F}}(\mathbb{X},\mathbb{U},p)$ for which a solution can be easily found and the original solution is gradually recovered by decreasing the homotopy parameter λ:

$$\mathbb{H}(\mathbb{X},\mathbb{U},p,\lambda) = \lambda\tilde{\mathbb{F}}(\mathbb{X},\mathbb{U},p) + (1-\lambda)\mathbb{F}(\mathbb{X},\mathbb{U},p) \tag{2.32}$$

Here, $\mathbb{H}(\mathbb{X},\mathbb{U},p,\lambda) \rightarrow \mathbb{F}(\mathbb{X},\mathbb{U},p)$ as $\lambda : 1 \rightarrow 0$. This approach is not very suitable for real-time control but rather for large scale problems where long simulation horizons are desirable; for instance trajectory optimisation or parameter identification for highly complex systems. *PolyMPC* features an implementation of the pseudo arclength continuation strategy [40] in the *experimental* subset of the package.

Another relaxation strategy is to augment the original Lagrange term of the cost function (2.1a) with a squared residual function of the system dynamics:

$$J[x(t_0),u(\cdot),p] = \Phi[x(t_f),t_f,p] + \int_{t_0}^{t_f} L[x(\tau),u(\tau),\tau,p] + \rho\|\dot{x}(\tau) - f(x(\tau),u(\tau),\tau,p)\|_2^2 d\tau \tag{2.33}$$

where ρ is a large scalar. Authors in [41] augment the cost with the integral of the ODE residual and adds a logarithmic barrier for inequality constraints to obtain an unconstrained optimisation problem. An alternative solution refinement technique based on the integral square residual minimisation is proposed in [42] which, however, requires that the problem is first solved with a standard direct collocation approach to obtain a bound for the cost function. In *PolyMPC* this relaxation strategy is available only using the *CasADi* interface.

Another potential source of problems is a discontinuous optimal control trajectory which often appears in problems where systems have affine input or where there is no control regularisation, for example. In these cases, an optimal control solution obtained with the pseudospectral approach can exhibit undesired oscillations related to Gibb's phenomenon. The classical way to tackle this problem is through grid refinement: changing the length or amount of collocation segments (h-refinement), increasing the order of approximating polynomials (p-refinement), or both combined (hp-refinement). Again, these methods are not particularly suitable for real-time applications as the optimisation problem has to be solved for each refinement iteration. Alternatively, a heuristic approach that often leads to reduced oscillations in practice is to constrain the control signal derivative which can be achieved by introducing an auxiliary control variable or done directly using the spectral derivative approximation as suggested in [43].

Another interesting observation is about the sparsity pattern of the differentiation matrix $\mathbb{D}$. In the case of global collocation, that is, there is only one interpolation segment, $\mathbb{D}$ consists of $(N+1) \times (N+1)$ diagonal matrices of size $n \times n$. When there is more than one interpolating segment, the sparsity pattern is depicted in Figure 2.1. One can observe that the larger the number of interpolating segments, the sparser the differentiation matrix (and overall constraint Jacobian). On the other hand, the spectral convergence of the Chebyshev collocation method is not guaranteed for multiple interval interpolation.

Implementation Details

Given user-defined functions to evaluate the cost, system dynamics and constraints, the number of segments and order of approximating polynomials, *PolyMPC* creates a class that allows the evaluation of discretised system dynamics and constraints along with their sensitivities, as well as the value, gradient and Hessian of the discretised cost and Lagrangian of (2.31). We take advantage of the separability of the discretised cost function (2.29) and constraints(2.28), which is characteristic of the Lobatto schemes, to compute or update the sensitivities in a block-wise manner. This is very important since for embedded applications we are limited to the forward mode automatic differentiation (AD) by operator overloading [44] where memory can be statically preallocated. Backward mode AD codes typically require dynamic memory

management to compute and store the tape or expression trace (except for codes based on a source code transformation method embedded in the compiler). Forward mode AD is less efficient than backward mode for functions where the number of outputs is a lot lower than the number of inputs. Exploiting function separability alleviates this disadvantage and this is the reason why the integral squared residual method, discussed earlier, is only available in the *CasADi* interface which features the backward mode AD implementation. The derivative operator in (2.33) makes use of all collocation points in a segment and therefore, breaks the separability of the cost function, and makes the problem denser. Computing a dense Hessian by forward mode AD introduces large overheads.

For a sparse implementation, we precompute the sparsity patterns of the Jacobians and Hessians and allocate exactly the required amount of memory. However, block updates of sparse matrices are typically very inefficient due to the high cost of random insertion, thus this feature is not available in *Eigen*. In our implementation, we exploit the knowledge of the sparsity pattern to work directly with data in CCS format, which allows one to perform vectorised block updates and therefore, significantly improve the performance.

2.3 Optimisation Kernel

PolyMPC is a collection of carefully optimised and loosely coupled numerical optimisation algorithms. This means that all available methods can be used completely independently of each other and of a particular problem structure. At the same time, the software architecture allows one to seamlessly combine the algorithmic blocks employing the template mechanism in C++ to enable the solution of complex nonlinear optimisation and optimal control problems. In the following, we present a brief overview of the available algorithms for numerical optimisation.

QP Solvers

There exist a number of software tools for solving the quadratic problems (QP) arising in optimal control that are potentially suitable for embedded applications. They are typically divided into two groups:

Sparsity-exploiting Among generic sparsity-exploiting solvers, *OOQP* [45] is a C++ program based on the interior-point method (IPM) [46]. Another tool, *OSQP*, uses the alternating direction method of multipliers (ADMM) and features code-generation capabilities for embedded deploy-

ment.

Structure-exploiting The software tools in this cohort exploit the QP structure inherent in optimal control problems. *qpOASES* [47] and *PRESAS* [48] are both block-structured active set solvers written in C. *HPIPM* [8] and *FORCES* [49] are IPM-based solvers optimised for problems resulting from multiple-shooting transcription.

PolyMPC solves quadratic programs with two-sided linear inequality constraints of the form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \tfrac{1}{2} x^T P x + q^T x \\ \text{subject to} \quad & l \leq Ax \leq u \\ & x_l \leq x \leq x_u \end{aligned} \tag{2.34}$$

where $x \in R^n$ is an optimisation variable, $x_l \in \mathbb{R}^n \cup -\infty$ and $x_u \in \mathbb{R}^n \cup +\infty$ are box constraints, $P \in \mathbb{S}_+^n$ positive semidefinite, $q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $l \in \mathbb{R}^m \cup -\infty$ and $u \in \mathbb{R}^m \cup +\infty$. For equality constraints $l_i = u_i$.

ADMM

The ADMM algorithm was chosen for its cheap and well vectorisable iterations that are particularly suitable for embedded applications. We first present a standard splitting approach suggested in the literature [50, 51, 52]. This will require the introduction of an extended constraint matrix $\tilde{A} = [A;\ I]$, where I is an $n \times n$ identity matrix, and an additional variable $z \in \mathbb{R}^{m+n}$ to write an equivalent QP:

$$\begin{aligned} \underset{x,z}{\text{minimize}} \quad & \tfrac{1}{2} x^T P x + q^T x \\ \text{subject to} \quad & l \leq z \leq u \\ & \tilde{A}x = z \end{aligned} \tag{2.35}$$

One can then rewrite the modified QP (2.35) by introducing auxiliary variables $\tilde{x}$ and $\tilde{z}$.

$$\begin{aligned} \underset{x,z,\tilde{x},\tilde{z}}{\text{minimize}} \quad & \tfrac{1}{2} \tilde{x}^T P \tilde{x} + q^T \tilde{x} + \mathscr{I}_{\tilde{A}x=z}(\tilde{x},\tilde{z}) + \mathscr{I}_{\mathscr{C}}(z) \\ \text{subject to} \quad & (\tilde{x},\tilde{z}) = (x,z) \end{aligned} \tag{2.36}$$

where $x \in \mathbb{R}^n$, $z \in \mathbb{R}^{m+n}$. $\mathscr{I}_{\tilde{A}x=z}$ is the indicator function for set $\{(x,z) \mid \tilde{A}x = z\}$ to enforce $\tilde{A}\tilde{x} = \tilde{z}$ and $\mathscr{I}_{\mathscr{C}}$ is the indicator function for set $\mathscr{C} = [l,u] = \{z \mid l_i \leq z_i \leq u_i, i = 1 \cdots m+n\}$ to enforce $l \leq z \leq u$.

The augmented Lagrangian for this problem is

$$\begin{aligned} L_{\sigma\rho}(x,z,\tilde{x},\tilde{z},w,y) = \quad & \tfrac{1}{2}\tilde{x}^T P\tilde{x} + q^T\tilde{x} + \mathscr{I}_{\tilde{A}x=z}(\tilde{x},\tilde{z}) + \mathscr{I}_{\mathscr{C}}(z) \\ & + \tfrac{\sigma}{2}\|\tilde{x} - x + \sigma^{-1}w\|_2^2 + \tfrac{\rho}{2}\|\tilde{z} - z + \rho^{-1}y\|_2^2 \end{aligned} \tag{2.37}$$

with dual variables (w, y) and corresponding penalty parameters $(\sigma, \rho) > 0$.

Problem (2.36) can be solved using the following ADMM iterations:

$$(\tilde{x}^{k+1}, \tilde{z}^{k+1}) \leftarrow \underset{(\tilde{x},\tilde{z}):\tilde{A}\tilde{x}=\tilde{z}}{\operatorname{argmin}} \frac{1}{2}\tilde{x}^T P\tilde{x} + q^T\tilde{x} + \frac{\sigma}{2}\|\tilde{x} - x^k + \sigma^{-1}w^k\|_2^2 + \frac{\rho}{2}\|\tilde{z} - z^k + \rho^{-1}y^k\|_2^2 \tag{2.38}$$

$$x^{k+1} \leftarrow \tilde{x}^{k+1} + \sigma^{-1}w^k \tag{2.39}$$

$$z^{k+1} \leftarrow \Pi_{\mathscr{C}}\left(\tilde{z}^{k+1} + \rho^{-1}y^k\right) \tag{2.40}$$

$$w^{k+1} \leftarrow w^k + \sigma(\tilde{x}^{k+1} - x^{k+1}) \tag{2.41}$$

$$y^{k+1} \leftarrow y^k + \rho(\tilde{z}^{k+1} - z^{k+1}) \tag{2.42}$$

The final algorithm is listed in Alogrithm 1. A relaxation is added to update steps (2.39)–(2.42) using the parameter $\alpha \in (0,2)$ and $r_{prim} = \|Ax - z\|_\infty$ and $r_{dual} = \|Px + q + A^T y\|_\infty$ are the primal and dual residuals accordingly.

Algorithm 1 ADMM Solver

1: **given:** initial values x^0, z^0, y^0 and parameters $\rho > 0, \sigma > 0, \alpha \in (0,2),\ \varepsilon_p, \varepsilon_d > 0$
2: **while** $r_{prim} \geq \varepsilon_p \ ||\ r_{dual} \geq \varepsilon_d$ **do**
3: $\quad (\tilde{x}^{k+1}, \nu^{k+1}) \leftarrow$ solve linear system $\begin{bmatrix} P + \sigma I & \tilde{A}^T \\ \tilde{A} & -\rho^{-1}I \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ \nu^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \rho^{-1}y^k \end{bmatrix}$
4: $\quad \tilde{z}^{k+1} \leftarrow z^k + \rho^{-1}(\nu^{k+1} - y^k)$
5: $\quad x^{k+1} \leftarrow \alpha\tilde{x}^{k+1} + (1-\alpha)x^k$
6: $\quad z^{k+1} \leftarrow \Pi_{\mathscr{C}}(\alpha\tilde{z}^{k+1} + (1-\alpha)z^k + \rho^{-1}y^k)$
7: $\quad y^{k+1} \leftarrow y^k + \rho(\alpha\tilde{z}^{k+1} + (1-\alpha)z^k - z^{k+1})$
8: **end while**

boxADMM

For dense problems with box constraints the splitting described before is not very efficient due to the potentially large number of zero values in $\tilde{A}$ and the linear system at the step (2) in Algorithm 1 that have to be stored. We therefore propose an additional splitting which aims at reducing memory consumption and better vectorisation – we call this solver *boxADMM*. As

before, auxiliary variables $z, \tilde{z} \in \mathbb{R}^m$ and $\tilde{x}, \xi \in \mathbb{R}^n$ are introduced to pose an equivalent QP:

$$\begin{aligned} \underset{x,\tilde{x},z,\tilde{z},\xi}{\text{minimize}} \quad & \tfrac{1}{2}\tilde{x}^T P \tilde{x} + q^T \tilde{x} + \mathscr{I}_{\tilde{A}x=z}(\tilde{x},\tilde{z}) + \mathscr{I}_{\mathscr{C}}(z) + \mathscr{I}_{\mathscr{B}}(\xi) \\ \text{subject to} \quad & (\tilde{x},\tilde{z}) = (x,z) \\ & \tilde{x} = \xi \end{aligned} \tag{2.43}$$

where $\mathscr{I}_{Ax=z}$ is the indicator function for set $\{(x,z) \mid Ax = z\}$ to enforce $A\tilde{x} = \tilde{z}$, $\mathscr{I}_{\mathscr{C}}$ is the indicator function for set $\mathscr{C} = [l,u] = \{z \mid l_i \le z_i \le u_i, i = 1...m\}$ to enforce $l \le z \le u$ and $\mathscr{I}_{\mathscr{B}}$ is the indicator function for box constraints $\mathscr{B} = [x_l, x_u] = \{\xi \mid x_l \le \xi \le x_u\}$.

The *boxADMM* algorithm 2 is derived following the standard procedure from [50]:

Algorithm 2 boxADMM Solver

given: initial values $x^0, z^0, \xi^0, y^0, \lambda^0$ and parameters $\rho > 0, \beta > 0, \sigma > 0, \alpha \in (0,2)$, $\varepsilon_p, \varepsilon_d > 0$
while $r_{prim} \ge \varepsilon_p \;||\; r_{dual} \ge \varepsilon_d$ **do**
  $(\tilde{x}^{k+1}, \nu^{k+1}) \leftarrow \text{solve} \begin{bmatrix} P + \sigma I + \beta I & A^T \\ A & -\rho^{-1} I \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ \nu^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q + \beta \xi^k - \lambda^k \\ z^k - \rho^{-1} y^k \end{bmatrix}$
  $\tilde{z}^{k+1} \leftarrow z^k + \rho^{-1}(\nu^{k+1} - y^k)$
  $x^{k+1} \leftarrow \alpha \tilde{x}^{k+1} + (1-\alpha) x^k$
  $z^{k+1} \leftarrow \Pi_{\mathscr{C}}(\alpha \tilde{z}^{k+1} + (1-\alpha) z^k + \rho^{-1} y^k)$
  $\xi^{k+1} \leftarrow \Pi_{\mathscr{B}}(x^{k+1} + \beta^{-1} \lambda^k)$
  $y^{k+1} \leftarrow y^k + \rho(\alpha \tilde{z}^{k+1} + (1-\alpha) z^k - z^{k+1})$
  $\lambda^{k+1} \leftarrow \lambda_k + \beta(x^{k+1} - \xi^{k+1})$
end while

Where $\lambda \in \mathbb{R}^n$ is a vector of Lagrange multipliers associated with box constraints of the original problem (2.34), $r_{prim} = max(\|Ax - z\|_\infty, \|x - \xi\|_\infty)$ and $r_{dual} = \|Px + q + A^T y + \lambda\|_\infty$ are the primal and dual residuals correspondingly.

Both the *ADMM* and the *boxADMM* algorithms are statically (specified at compile time) parametrised by:

Problem dimensions	Dimension of the optimisation variable and number of constraints (excluding box constraints). This allows the allocation of the necessary memory and compile-time data consistency checks.
Scalar type	*double* (default), *float, complex* or user-defined scalar types.
Matrix format	*DENSE* (default) or *SPARSE*. This allows one to have a unifying interface for sparse and dense linear algebra.
Linear system solver	QP solvers support all direct and iterative, dense and linear solvers available in *Eigen*. The user also can provide a custom linear solver given it is derived

from one of the base solver classes in *Eigen*.

Besides two custom ADMM implementations, *PolyMPC* interfaces an established ADMM solver *OSQP*, which additionally includes an infeasibility detection feature, and a Goldfarb-Idnani active-set solver *QPMAD*.

Our *ADMM* implementation has several attractive features compared to *OSQP*. First, operations (3-8) of Algorithm 2 are explicitly vectorised so a significant speed-up is expected when SIMD optimisations are enabled. Second, our implementation supports both dense and sparse linear algebra, where only sparse computations are available in *OSQP*. Third, *OSQP* supports static memory allocation only through code generation, which requires that the ρ-update mechanism is not available and factorisation of the KKT matrix happens only once. *PolyMPC* on the other hand, supports all algorithmic features for sparse and dense static matrices.

Nonlinear Problem

We are concerned with solving nonlinear optimisation programs (NLP) of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c(x) = 0 \\ & g_l \leq g(x) \leq g_u \\ & x_l \leq x \leq x_u \end{aligned} \tag{2.44}$$

with $x \in \mathbb{R}^n$, equality constraint $c(x) : \mathbb{R}^n \to \mathbb{R}^m$, inequality constraint $g(x) : \mathbb{R}^n \to \mathbb{R}^g$, and box constraint defined by vectors $x_l \in \mathbb{R}^n$ and $x_u \in \mathbb{R}^n$. Before describing a numerical method for solving these NLPs in *PolyMPC* we introduce basic definitions and concepts in nonlinear optimisation.

For the problem (2.44), the *Lagrangian function* is defined as:

$$\mathscr{L}(x, \lambda_c, \lambda_g, \lambda_x) = f(x) + \lambda_c^T c(x) + \lambda_g^T g(x) + \lambda_x^T x \tag{2.45}$$

Where $\lambda_c, \lambda_g, \lambda_x$ are the Lagrange multipliers corresponding to the constraints $c(x) = 0$, $g_l \leq g(x) \leq g_u$ and $x_l \leq x \leq x_u$ respectively.

The $i-$th constraint $g_{l_i} \leq g_i(x) \leq g_{u_i}$ is called *inactive* if the inequalities hold strictly, otherwise it is called *active*. A subset of of all active constraints will be denoted by a subscript $\mathscr{A}$, e.g.,

$g_{\mathscr{A}}$. A point $x^\star$ is called *regular* if it is feasible and the following matrix has full row rank:

$$\begin{bmatrix} \nabla_x c(x^\star) \\ \nabla_x g(x^\star)_{\mathscr{A}} \end{bmatrix} \tag{2.46}$$

The last conditions are referred to as the linear independence constraint qualification (LICQ).

The Karush-Kuhn-Tucker conditions (KKT conditions) state the necessary conditions of optimality for an NLP: for a regular and locally optimal point $x^\star$ there exist Lagrange multipliers $\lambda_c^\star$, $\lambda_g^\star$, $\lambda_x^\star$ such that the following holds:

Stationarity:

$$\nabla_x \mathscr{L}(x^\star, \lambda_c^\star, \lambda_g^\star, \lambda_x^\star) = 0 \tag{2.47}$$

Primal feasibility:

$$c(x^\star) = 0, g_l \le g(x^\star) \le g_u, x_l \le x^\star \le x_u \tag{2.48}$$

Dual feasibility and complimentarity:

$$(\lambda_g^\star)_i \begin{cases} \ge 0 & \text{if the i-th upper bound of } g(x) \text{ is active} \\ \le 0 & \text{if the i-th lower bound of } g(x) \text{ is active} \\ = 0 & \text{if the i-th constraint is inactive} \end{cases}$$

$$(\lambda_x^\star)_i \begin{cases} \ge 0 & \text{if the i-th upper bound of } x \text{ is active} \\ \le 0 & \text{if the i-th lower bound of } x \text{ is active} \\ = 0 & \text{if the i-th box constraint is inactive} \end{cases}$$

SQP Solver

The sequential quadratic programming (SQP) approach approximates the problem (2.44) by a constrained quadratic subproblem (2.49) at the current iteration x^k, and the minimizer of this subproblem $p^\star$ is used to compute the next iterate x^{k+1}. For equality-constrained NLPs, solving this subproblem is equivalent to applying a Newton iteration to the KKT conditions (2.47), (2.48) [46].

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & \tfrac{1}{2} p^T H p + h^T p \\ \text{subject to} \quad & c(x^k) + A_c p = 0 \\ & g_l - g(x^k) \le A_g p \le g(x^k) - g_u \\ & x_l - x^k \le p \le x_u - x^k \end{aligned} \tag{2.49}$$

with $A_c \in \mathbb{R}^{m \times n}$ and $A_g \in \mathbb{R}^{g \times n}$ the Jacobian of the constraints $c(x)$ and $g(x)$ at point x^k, the Hessian of the Lagrangian $H = \nabla_{xx} L(x^k, \lambda^k)$, $\lambda^k = [\lambda_c^k, \lambda_g^k, \lambda_x^k]^T$ and objective gradient

$h = \nabla_x f(x^k)$.

An extended version of the line search SQP algorithm 18.3 from [46] is presented in Algorithm 3. Here r_{prim} denotes the maximum constraint violation of the problem (2.31) at the iterate x^k, and $\alpha^0 = 1$.

Algorithm 3 SQP

given: $x^0, \lambda^0, p^0, p_\lambda^0, \varepsilon_p, \varepsilon_\lambda, \varepsilon > 0$
while $\alpha^k \|p^k\|_\infty \geq \varepsilon_p \;||\; \alpha^k \|p_\lambda^k\|_\infty \geq \varepsilon_\lambda \;||\; r_{prim} \geq \varepsilon$ **do**
    $(H, h, c, A_c, g, A_g) \leftarrow$ linearization(x^k, λ^k) of the Problem (2.44)
    $H \leftarrow$ apply regularisation(H)
    Apply preconditioning to Problem (2.49)
    $p^k, \hat{\lambda} \leftarrow$ solve Problem (2.49)
    $p^k, \hat{\lambda} \leftarrow$ revert preconditioning
    $p_\lambda^k \leftarrow \hat{\lambda} - \lambda^k$
    $\alpha^k \leftarrow$ line search$(p^k, \hat{\lambda})$
    $x^{k+1} \leftarrow \alpha^k p^k$
    $\lambda^{k+1} \leftarrow \alpha^k p_\lambda^k$
    $k \leftarrow k + 1$
end while

The SQP algorithm requires three template arguments:

Problem — The solver will deduce the problem dimensions and data types from the problem class (an example of an NLP class definition will be shown in the next section) and allocate necessary memory, as the problem itself typically does not require any allocation by the user. The problem will then be used to evaluate user defined functions and sensitivities.

QP Solver — Here any of the available QP solvers can be used: *boxADMM* (default), *ADMM*, *OSQP* or *QPMAD*. The user can provide a custom QP implementation by deriving from the *QPBase* interface class.

Preconditioner — ADMM methods are known to be sensitive to the problem conditioning [52, 51]. However, it is sometimes possible to scale the data to achieve better conditioning and therefore improve the convergence of the QP solver. At the moment, the user can choose between no preconditioning (default) and a heuristic Ruiz matrix equilibration algorithm [53].

Furthermore, the solver in *PolyMPC* is designed such that every step of the algorithm can be

customised by the user. The rest of the section briefly outlines the steps required for the SQP algorithm.

Sensitivity Computation

For sensitivity computation, the software relies on the forward mode AD implemented using the operator overloading method in *Eigen*. We extend the existing functionality of the library to deal with some vector-valued functions where the sensitivity computations can be efficiently vectorised or derivative expressions are known in the closed form and do not require the AD procedure. Our current solution relies on the template specialisation mechanism which allows compile-time selection of a particular implementation based on the scalar type. This mechanism is demonstrated on a simplified example of a quadratic form differentiation in Listing 1.

Listing 1 Vectorised sensitivity computation of a quadratic form. The AD type is a complex structure and therefore the computation of the quadratic form will be performed coefficient-wise as in a generic *quad_form* function. By providing a specialised implementation for the *adscalar_t* we take advantage of the explicit vectorisation mechanism in *Eigen*

```
// general scalar
template<typename scalar_t>
scalar_t quad_form(Ref<vector_t<scalar_t>> x, const Ref<const real_vector_t>& y)
{
    return x.dot(A * x);
}

// autodiff
template<>
adscalar_t quad_form<adscalar_t>(Ref<vector_t<adscalar_t>> x,
                                 const Ref<const real_matrix_t>& A)
{
    real_vector_t xt;
    for (Index i  = 0; i < Size; ++i)
        xt(i) = x(i).value();

    // vectorised matrix-vector multiplication and dot product
    xt = A * xt;
    Scalar value = xt.dot(xt);

    // vectorised dot product
    real_vec_t derivative;
    for (Index i  = 0; i < Size; ++i)
        derivative(i) = 2 * xt.dot(x(i).derivative())


    return adscalar_t(value, derivative);
}
```

For dense problems, the software can generate sensitivities automatically or mix AD and

user-provided functions to evaluate derivatives. For sparse problems, the user must provide functions for sensitivity computations. It is possible to estimate the sparsity pattern of Jacobians and Hessians automatically by using, for example, Bayesian boolean probing [54] or graph colouring techniques [55]. This feature is subject to future development.

Computing the Hessian of the Lagrangian in Algrithm 3 can be quite expensive for moderate-sized and large problems. Therefore, we implement the damped Broyden-Fletcher-Goldfarb-Shanno (BFGS) update method [46] summarised in Algorithm 4

Algorithm 4 Damped BFGS Update

procedure DAMPED BFGS(B, y, s)
  if $s^T y < 0.2 s^T B s$ **then**
    $\theta \leftarrow 0.8(s^T B s)/(s^T B s - s^T y)$
    $r \leftarrow \theta y + (1-\theta) B s$ ▷ Damped update
  else
    $r \leftarrow y$ ▷ Normal update
  end if
  $B^{k+1} \leftarrow B - \frac{B s s^T B}{s^T B s} + \frac{r r^T}{s^T r}$
  return B^{k+1}
end procedure

where B denotes the Hessian approximation, $y = x^{k+1} - x^k$ is the step and $s = \nabla_x \mathscr{L}(x^{k+1}, \lambda^{k+1}) - \nabla_x \mathscr{L}(x^k, \lambda^k)$ is the corresponding change of the gradient of the Lagrangian. In the pseudospectral collocation module, where the sparsity pattern is known we implement dense and sparse block-BFGS updates which further improves the computational performance.

Regularisation

In our numerical implementation, the Hessian of the Lagrangian is evaluated exactly at the first SQP iteration and then updated using the sparsity-preserving block-BFGS method 4 or exact linearisation. The chosen adaptive line-search strategy in the SQP requires that the Hessian is kept positive-definite [46] which is not guaranteed by the algorithm design. For this reason, we approximate the Hessian by a positive definite one through regularization and the simplest way to regularise the Hessian is by adding a multiple of the identity matrix [46]. It is often hard, however, to choose the scaling parameter without knowing the minimal negative eigenvalue. Furthermore, this method perturbs the positive eigenvalues as well, and therefore changes the curvature information. Eigenvalue mirroring is another regularisation technique that corrects only negative eigenvalues. As the name suggests, however, the eigenvalue decomposition

should be available at each solver iteration, which renders the method infeasible for most real-time applications. An alternative convexification method preserving the sparsity pattern is presented in [56]. The method is tailored to the multiple-shooting structure of the problem and assumes explicit integration of the dynamics, i.e. not suitable for the pseudospetral collocation method used in our approach.

To cope with the potentially indefinite Hessian, we propose a heuristic regularization algorithm that modifies the eigenvalues of the Hessian based on Gershgorin spectrum bounds. Gershgorin circle theorem [57] states that every eigenvalue of a square matrix A lies within the union of discs $D(a_{ii}, R_i)$ such that a_{ii} the diagonal element in A is the center of the disc with radius R_i equal to the sum of the off-diagonal entries of every row:

$$R_i = \sum_{i \neq j} |a_{ij}| \tag{2.50}$$

By using the symmetry of the Hessian, it is possible to take advantage of the column-major storage order for both dense and sparse matrices to compute the Gershgorin discs efficiently.

Algorithm 5 Gershgorin regularisation

1: **while** $i \leq N_{rows}$ **do**
2: **if** $a_{ii} - R_i \leq 0$ **then**
3: $a_{ii} \leftarrow -(a_{ii} - R_i) + \varepsilon$
4: **end if**
5: **end while**

This algorithm will conservatively guarantee that the approximated Hessian is positive definite with eigenvalues contained in Gershgorin discs intersecting the real axis at a minimum $\varepsilon > 0$.

Preconditioning

In general, convergence of first order methods is affected by the distribution of the eigenvalues of the matrix that characterises the problem data:

$$M = \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \tag{2.51}$$

Preconditioning is a numerical technique that aims to improve the eigenvalue distribution, typically by transforming data matrices H and A. In the context of quadratic programming in control, several diagonal preconditioners minimising the conditioning number of M were proposed in the literature [52, 58]. These techniques, however, require solving a semi-definite

program which is usually more expensive than solving a QP itself. In [48] a preconditioned conjugate-gradient method is used in combination with a block-structured active-set QP solver.

By default, in *PolyMPC* no preconditioning is performed but, similar to [51] the user has an option of using an implemented Ruiz matrix equilibration algorithm that aims at scaling primal and dual variables $\bar{x} = Dx, \bar{\lambda} = E\lambda$ such that the modified matrix M has columns with equal l_∞-norms. It is important to note that matrix equilibration is a heuristic method and does not guarantee desirable distribution of eigenvalues of a matrix but sometimes improves it in practice. The modular design of *PolyMPC* allows one to provide a custom preconditioning algorithm both for QP data and for iterative linear solvers.

2.4 Examples

This section provides usage examples of the different modules in *PolyMPC*. We start with very simple QP and NLP problems to showcase the simplicity and compactness of the code. In the second part of the section we demonstrate how the software is used to create optimal guidance and real-time nonlinear predictive controllers for a highly nonlinear and unstable system, which emulates thrust vector control (TVC) of a sounding rocket. Both guidance and NMPC algorithms are deployed on a low-power embedded platform and demonstrate the real-time capabilities in real-world flight experiments.

Quadratic Program

The purpose of this example is to demonstrate how one can create an embedded QP solver in just a few lines of code.

$$\begin{aligned} \min_x \ & \frac{1}{2} x^T \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T x \\ s.t. \ & 1 \leq x_1 + x_2 \leq 1 \\ & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq x \leq \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} \end{aligned} \tag{2.52}$$

Solving this problem with *PolyMPC* would require writing the following simple program: We first define a scalar type that will be used for all data types in the solver, in this case *double*. All modules are compatible with *Eigen* data types, so if the user passes a matrix of a wrong size or type, the static checker will return a compilation error with a corresponding message. After

Listing 2 Solving (2.52) with *PolyMPC*

```
using Scalar = double;

Eigen::Matrix<Scalar, 2,2> H;
Eigen::Matrix<Scalar, 2,1> h, xu, xl,;
Eigen::Matrix<Scalar, 1,2> A;
Eigen::Matrix<Scalar, 1,1> al, au;

H << 4, 1, 1, 2;
h << 1, 1;
A << 1, 1;
al << 1; xl << 0, 0;
au << 1; xu << 0.7, 0.7;

/** ADMM method with LDLT linear solver  */
ADMM<2, 1, Scalar> admm_solver;

/** boxADMM with LDLT linear solver */
boxADMM<2, 1, Scalar> box_admm_solver;

/** boxADMM with Conjugate Gradient linear solver */
boxADMM<2, 1, Scalar, DENSE, Eigen::ConjugateGradient,
        Eigen::Lower | Eigen::Upper > cg_admm_solver;

/** solve with 3 different solvers */
admm_solver.solve(H, h, A, al, au, xl, xu);
box_admm_solver.solve(H, h, A, al, au, xl, xu);
cg_admm_solver.solve(H, h, A, al, au, xl, xu);

/** get the solution */
Eigen::Vector2d sol = admm_solver.primal_solution();
```

the problem data was created, the user needs to define the QP solver which is parametrised by: dimension of the optimisation variable, number of constraints (excluding box constraints, which do not require additional memory allocation), scalar type (default: *double*), matrix format (*DENSE/SPARSE*, default: *DENSE*), linear solver (default: *LDLT (DENSE) : Simplicial LDLT (SPARSE)*). It is further possible to provide a hint to the solver if the Hessian is symmetric, lower or upper triangular. In the next commented line we show how an iterative conjugate gradient solver can be used with the very same setup.

Nonlinear Program

In order to demonstrate the NLP interface of the *PolyMPC* package, we consider Problem 71 from the Hock-Schittkowski problem collection [59].

$$\begin{aligned} \min_{x \in \mathscr{R}^4} \; & x_1 x_4 (x_1 + x_2 + x_3) + x_3 \\ s.t. \quad & x_1^2 + x_2^2 + x_3^2 + x_4^2 = 40 \\ & x_1 x_2 x_3 x_4 \geq 25 \end{aligned} \tag{2.53}$$

with the starting point $x^0 = \begin{bmatrix} 1 & 5 & 5 & 1 \end{bmatrix}^T$. The problem definition is in Listing 3.

Listing 3 Solving (2.53) with *PolyMPC*

```
/** HS071 problem as in Ipopt tutorial */
POLYMPC_FORWARD_NLP_DECLARATION(/*Name*/ HS071, /*NX*/ 4, /*NE*/1,
                                                /*NI*/1, /*NP*/0, /*Type*/double);
class HS071 : public ProblemBase<HS071>
{
public:

    template<typename T>
    inline void cost_impl(const Ref<const variable_t<T>>& x,
                          const Ref<const static_parameter_t>& p,
                          T& cost) const
    {
        cost = x(0)*x(3)*(x(0) + x(1) + x(2)) + x(2);
    }

    template<typename T>
    inline void inequality_constraints_impl(const Ref<const variable_t<T>>& x,
                                            const Ref<const static_parameter_t>& p,
                                            Ref<ineq_constraint_t<T>> constraint) const
    {
        // 25 <= x^2 + y^2 <= Inf -> will set bounds once the problem is instantiated
        constraint << x(0)*x(1)*x(2)*x(3);
    }

    template<typename T>
    inline void equality_constraints_impl(const Ref<const variable_t<T>>& x,
                                          const Ref<const static_parameter_t>& p,
                                          Ref<eq_constraint_t<T>> constraint) const
    {
        // x(0)^2 + x(1)^2 + x(2)^2 + x(3)^2 == 40
        constraint << x.squaredNorm() - 40;
    }
};
```

The *POLYMPC_FORWARD_DECLARATION* macro creates type traits for the *HS071* class so that the base class *ProblemBase* can statically preallocate necessary memory. The user must

specify the name, number of variables, number of equality and inequality constraints, size of the parameter vector and a scalar type. As the next step the user needs to implement the cost and constraint functions. The resulting class *HS071* can then be passed to a nonlinear solver.

Guidance and Control of a Sounding Rocket Prototype

Thrust Vector Control (TVC) is a key technology enabling rockets to perform complex autonomous missions, such as active stabilization, orbit insertion, or propulsive landing. This is achieved by independently controlling the thrust direction and magnitude of each of its engines. Compared to aerodynamic control such as fins or a canard, it guarantees a high control authority even in the absence of an atmosphere, i.e. during high altitude launches or exploration of other planetary bodies.

In the following, we present an implementation of an optimal guidance, navigation and control (GNC) system for the motion control of a small-scale electric prototype of a thrust-vectored rocket. The aim of this prototype is to provide an inexpensive platform to explore GNC algorithms for automatic landing of sounding rockets. The guidance and trajectory tracking are formulated as continuous-time optimal control problems and are solved in real-time on embedded hardware using *PolyMPC*. Finally, indoor and outdoor flight experiments are performed to validate the architecture and NMPC performance.

The vehicle portrayed in Figure 2.2 is 60 cm tall, and has a diameter of 26 [cm], totaling a weight of 1.7 kg. It has a maximum thrust of about 2.3 kg. The center of mass is located at a distance of 21.5 cm from the propellers. The batteries are mounted at the top of the vehicle to achieve similarity with rocket dynamics by moving the center of gravity (CoG) further from the propellers.

A Pixhawk 4 mini is used for state estimation and servo control. It contains an inertial measurement unit (IMU) with a magnetometer, as well as an external GPS antenna. The state estimates are accessed through the MAVROS protocol. For indoor flights, the motion capture system Optitrack replaces the GPS measurements.

A Raspberry Pi 4 model B, equipped with a quad-core Cortex-A72 processor and 4 GB of RAM, plays the role of the onboard computer and hosts the guidance, control and disturbance estimation algorithms. Each of the aforementioned algorithms is assigned to a separate core to enhance the performance. The robot operating system (ROS) [60] running on an embedded Ubuntu Server provides the interprogram communication interface between the software components. The GNC architecture can be seen in Figure 2.3. The guidance algorithm computes an optimal trajectory linking the current and the target states, which is then tracked by the nonlinear model predictive controller (NMPC).

(a) TVC Prototype. (b) Two-axis gimbal with mounted propellers

Figure 2.2 – The prototype of a TVC rocket.

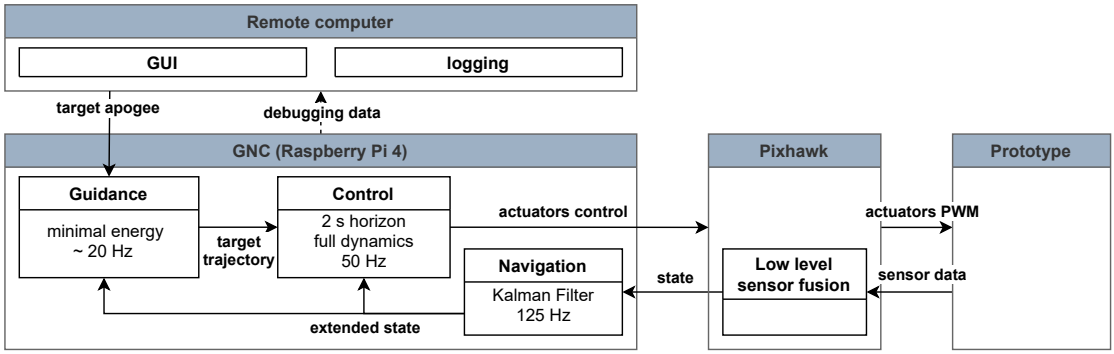


Figure 2.3 – GNC software architecture.

Equations of motion

Throughout the section, the vectors without superscript are assumed to be given in the fixed inertial reference frame (IRF), and the vectors with superscript b are given in the body reference frame (BRF). The position vector is denoted by $p = \begin{bmatrix} x & y & z \end{bmatrix}$, the velocity $v = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}$, and the orientation is defined by the quaternion $q = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}$, while the angular speed in BRF is given by $\omega^b = \begin{bmatrix} \omega_x^b & \omega_y^b & \omega_z^b \end{bmatrix}$. The rotation matrix from BRF to IRF derived from the quaternion q is denoted $R(q)$.

The state vector is denoted $\boldsymbol{x}$, and contains the position vector, the velocity, the orientation q,

and the angular speed in the body frame ω^b:

$$\boldsymbol{x} = \begin{bmatrix} p & v & q & \omega^b \end{bmatrix}^T \tag{2.54}$$

The control vector is denoted $\boldsymbol{u}$ and the prototype is controlled through the command servo angles θ_1 and θ_2, as well as the speed of the bottom and top propellers, P_B and P_T respectively (in % for the rest of the section). It is more convenient, however, to consider the propellers' average command speed as an input $\overline{P} = \frac{P_B + P_T}{2}$ and the command speed difference $P_\triangle = P_T - P_B$:

$$\boldsymbol{u} = \begin{bmatrix} \theta_1 & \theta_2 & \overline{P} & P_\triangle \end{bmatrix}^T \tag{2.55}$$

The state equations are given by generic 6 DoF solid body dynamics. Omitting atmospheric interactions, the forces acting on the vehicle are: gravity mg, thrust F_T^b and total torque M^b in BRF. M^b comprises the torque due to the thrust vector F_T^b and the torque M_P^b caused by the speed difference between the two propellers. The complete dynamics is given by:

$$\begin{aligned} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{q} \\ \dot{\omega^b} \end{bmatrix} = \begin{bmatrix} v \\ \dfrac{R(q)F_T^b}{m} + g \\ \frac{1}{2} q \circ \omega^b \\ I^{-1}(M^b - \omega^b \times (I\omega^b)) \end{bmatrix} \\ M^b = r \times F_T^b + M_P^b \end{aligned} \tag{2.56}$$

Both the thrust F_T^b and the torque M_P^b vectors are determined by the gimbal angles:

$$\frac{F_T^b}{\|F_T^b\|} = \frac{M_P^b}{\|M_P^b\|} = \begin{bmatrix} \sin\theta_2 \\ -\sin\theta_1 \cos\theta_2 \\ \cos\theta_1 \cos\theta_2 \end{bmatrix} \tag{2.57}$$

The relations between the absolute values of F_T^b and M_P^b and the control parameters $\overline{P}, P_\triangle$ are difficult to establish from the physical principles, and therefore, these relations have been identified experimentally:

$$\begin{aligned} \|F_T^b\| &= f_1(\overline{P}, P_\triangle) = a\,\overline{P}^2 + b\,\overline{P} + c \\ \|M_P^b\| &= f_2(\overline{P}, P_\triangle) = c\; I_{zz} P_\triangle \end{aligned} \tag{2.58}$$

Guidance

Similar to [61] and [62], the guidance algorithm is based on a point mass model and uses a minimal energy formulation where terminal time t_f is an optimization variable. The final position $p(t_f)$ is constrained to the small neighborhood of the target position p_t with zero-velocity.

$$
\begin{aligned}
&\min_{F_T(t),\varphi(t),\psi(t),t_f} \int_{t_0}^{t_f} F_T^2(t)\,dt + \rho\eta \\
&s.t. \quad \dot{p}(t) = v(t),\ \dot{v}(t) = \frac{F_T(t)}{m}\begin{bmatrix} \sin\varphi\sin\psi \\ -\cos\varphi\sin\psi \\ \cos\psi \end{bmatrix} + g \\
&\qquad v_{min} \le v(t) \le v_{max} \\
&\qquad F_{min} \le F_T(t) \le F_{max} \\
&\qquad -\psi_{max} \le \psi(t) \le \psi_{max} \\
&\qquad p(t_0) = p_0 \\
&\qquad v(t_0) = v_0 \\
&\qquad z(t_f) = z_t \\
&\qquad (x(t_f) - x_t)^2 + (y(t_f) - y_t)^2 \le \eta \\
&\qquad v(t_f) = 0
\end{aligned}
\tag{2.59}
$$

The propulsion vector of the rocket is defined in spherical coordinates, where F_T is the absolute value of thrust, φ the azimuth angle and ψ the polar angle. Symmetric constraints on the polar angle define the aperture of the reachable cone. Since the attitude of the vehicle is not explicitly considered in the guidance problem, the polar angle is usually related to the tilt angle of the rocket, and thus should not be too large.

The slack variable η weighted by ρ is used to formulate a slack constraint on the target horizontal position, which may be violated when close to the target position.

Since the guidance OCP has a free terminal time, the horizon is scaled to the interval [0, 1], $\tau \equiv \frac{t-t_0}{t_f-t_0}$, the dynamics become $\dot{x} = (t_f - t_0) f(x,u)$, and the horizon length $(t_f - t_0)$ then becomes a variable parameter in the OCP. An initial guess for this parameter is given to accelerate the solving of the OCP using a simple closed form solution.

NMPC Tracking Controller

The NMPC controller uses the full state dynamics (2.56), augmented with the disturbance estimation and has a prediction horizon of two seconds. Close to the target, the length of the horizon is scaled down to match the time to target provided by the guidance algorithm.

$$\begin{aligned}
&\min_{\boldsymbol{u}(t)} \int_{t_0}^{t_f} l(\boldsymbol{x},\boldsymbol{u},t)\,dt + V_f(\boldsymbol{x}_f) \\
&s.t. \quad \dot{\boldsymbol{x}} = f(\boldsymbol{x},\boldsymbol{u}) \\
&\qquad -\theta_{max} \le \theta_1 \le \theta_{max} \\
&\qquad -\theta_{max} \le \theta_2 \le \theta_{max} \\
&\qquad -\dot{\theta}_{max} \le \dot{\theta_1} \le \dot{\theta}_{max} \\
&\qquad -\dot{\theta}_{max} \le \dot{\theta_2} \le \dot{\theta}_{max} \\
&\qquad P_{min} \le \overline{P} + P_{\triangle}/2 \le P_{max} \\
&\qquad P_{min} \le \overline{P} - P_{\triangle}/2 \le P_{max} \\
&\qquad P_{\triangle min} \le P_{\triangle} \le P_{\triangle max} \\
&\qquad 0 \le z
\end{aligned} \tag{2.60}$$

The servo motors are constrained in a range of ±15° and we introduce derivative constraints to limit the maximum rate of input change given by the controller and to smoothen the open-loop trajectories.

The propeller speed models are directly included in the formulation, and along with the constraints on top and bottom propeller speeds P_T and P_B, provide a simple way to deal with the trade-off between roll control (through $P_{\triangle}$) and altitude control (through $\overline{P}$).

Stage Cost

The tracking residual at time t corresponds to the difference between the predicted $x(t)$ and the target guidance trajectory $x_G(t)$. The stage cost combines the squared tracking residual, penalty on the control input and penalty on the deviation from vertical orientation. The components $q_w q_x - q_y q_z$ and $q_w q_y + q_x q_z$ penalize deviations of pitch and yaw angles from zero [63]. The roll angle is not controlled directly, but rather the roll rate ω_z^b, as the final roll angle is not critical for the flight mission.

$$l(\boldsymbol{x},\boldsymbol{u},t) = e(\boldsymbol{x}-\boldsymbol{x}_G(t))^T Q\, e(\boldsymbol{x}-\boldsymbol{x}_G(t)) + \boldsymbol{u}^T R \boldsymbol{u} \tag{2.61}$$

$$e(\boldsymbol{x}) \equiv \begin{bmatrix} p & v & q_w q_x - q_y q_z & q_w q_y + q_x q_z & \omega^b \end{bmatrix}^T \tag{2.62}$$

Terminal Cost In order to improve stability, a continuous-time linear quadratic regulator (LQR) based on a linearization around the zero-speed steady-state (2.63) is used as a terminal controller. The matrices Q and R for the LQR design are identical to the ones used in the stage cost.

$$A = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x_s,u_s} \quad B = \left.\frac{\partial f(x,u)}{\partial u}\right|_{x_s,u_s} \tag{2.63}$$

Note that the states q_w and q_z are fixed in the linearization, as they are not controlled. The matrix of the terminal quadratic cost Q_f is then obtained by solving the continuous time algebraic Riccati equation (CARE). The final cost is then:

$$V_f(\boldsymbol{x}_f) = e(\boldsymbol{x}_f)^T Q_f \, e(\boldsymbol{x}_f) \tag{2.64}$$

Compared to previously proposed methods, this formulation allows for simultaneous tracking of the optimal trajectory and vertical stabilization, and therefore improves the agility of the vehicle.

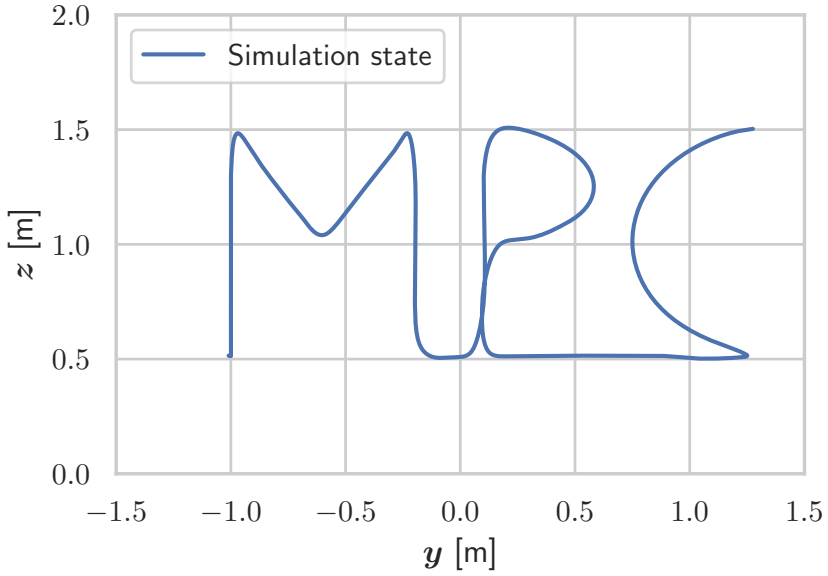
Indoor Experiment

For the GNC system validation, the controller had to track a complex "MPC"-shaped pattern with a constant speed of 0.35 [m/s]. The flight was performed indoors, where Optitrack was used to obtain the position and orientation data. Simulation and real-world flight experiments are in Figure 2.4.

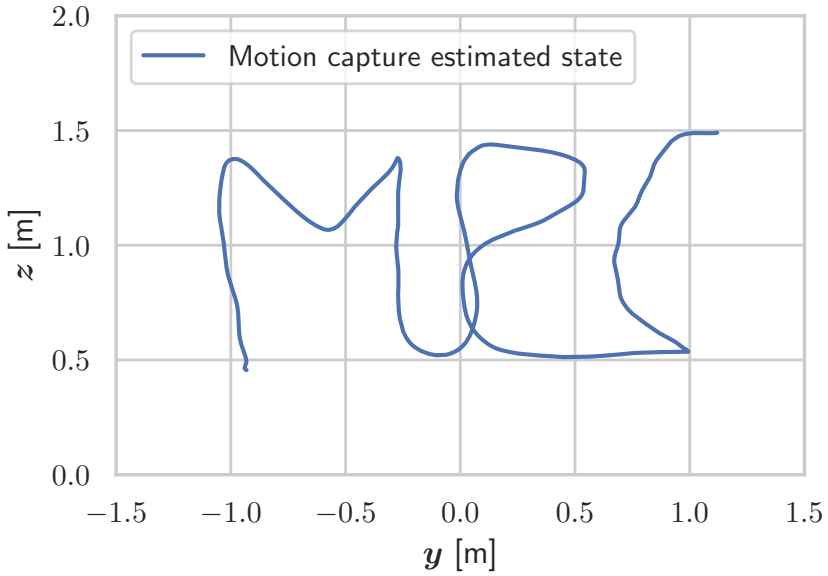
Outdoor Experiment

In order to validate the overall architecture using the guidance and tracking system, the outdoor experiment includes reaching a 3-meter apogee, followed by a controlled descent to a given landing point two meters away from the starting position.

The resulting trajectory can be observed in Figure 2.5.



(a) Simulation



(b) Real flight

Figure 2.4 – Comparison of the NMPC tracking performance in simulation study and indoor flight experiment.

Figure 2.5 – Apogee tracking and controlled descent. It could be observed that the vehicle initially deviates from the vertical trajectory due to the wind disturbance, then recovers and at the altitude of 3 meters computes the descent trajectory.

2.5 Benchmarks

This section consists of two parts: The first presents a number of simulation experiments in order to underline the performance properties of the algorithm, namely scalability with the number of collocation points and the prediction horizon. The second provides a short run-time performance comparison for the same problem implemented using the ACADO code generation tool and the PSOPT library, as well as highlighting the main differences of the considered optimal control tools. The benchmarks are performed using both the *CasADi* and the *Eigen* interfaces of the software on a desktop computer, low-power embedded computer running Linux and a micro-controller (MCU).

An example of an orbit tracking flight controller synthesis for an Airborne Wind Energy (AWE) kite is considered. The state of the kite can be characterized by its position with respect to a ground station in spherical coordinates- elevation angle θ and azimuth φ, as well as heading angle γ, while the tether length L is assumed to be constant. It is possible to directly affect the flight direction, or heading angle, by pulling the control lines of the kite and the control signal u_γ is the rate of change of the heading angle. A more detailed description of the kite mathematical model can be found in [64]. The kite kinematics are given by the following system of ODEs:

$$\begin{aligned} \begin{bmatrix} L & 0 \\ 0 & L\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\varphi} \end{bmatrix} &= \bar{R}_{NK} \begin{bmatrix} 1 & 0 & -E \\ 0 & 0 & 0 \end{bmatrix} R_{NK}^T R_{GN}^T v_w - \bar{R}_{NK} \begin{bmatrix} Ez \\ 0 \end{bmatrix} \\ \dot{\gamma} &= u_\gamma \end{aligned} \tag{2.65}$$

Where $\bar{R}_{NK}$, R_{NK} and R_{GN} are the following rotation matrices:

$$\begin{aligned} R_{GN} &= \begin{bmatrix} -\sin\theta\cos\varphi & -\sin\varphi & -\cos\theta\cos\varphi \\ -\sin\theta\sin\varphi & \cos\varphi & -\cos\theta\sin\varphi \\ \cos\theta & 0 & -\sin\theta \end{bmatrix}, \; R_{NK} = \begin{bmatrix} \bar{R}_{NK} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \\ \bar{R}_{NK} &= \begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix} \end{aligned} \tag{2.66}$$

Other parameters are wind speed v_w, glide ratio E and tether reel-out/in speed z (assumed to be zero in this example). One of the most common trajectories in AWE applications is a “lemniscate" orbit, the parametric equation of which is given below:

$$\begin{aligned} \theta(\tau) &= h + a\sin(2\tau) \\ \varphi(\tau) &= 4a\cos(\tau) \end{aligned} \tag{2.67}$$

State and control constraints:

$$\begin{aligned} 0 \leq \theta(t) &\leq \frac{\pi}{2} \\ -\frac{\pi}{2} \leq \varphi(t) &\leq \frac{\pi}{2} \\ -\pi \leq \gamma(t) &\leq \pi \\ -5 \leq u_\gamma &\leq 5 \end{aligned} \tag{2.68}$$

A simulation study was carried out in order to assess the real-time feasibility of the predictive controller for the kite path following problem. The orbit following controller was implemented using the CasADi interface of PolyMPC with IPOPT as an NLP solver. The kite simulator and controller were set to run asynchronously on a 2.8 GHz Intel Core i7 processor with inter-process communication provided through ROS [60]. The simulator was run at 100 Hz, and the controller node was set to run at 50Hz with the following parameters: number of collocation points- 12; number of subintervals- 3, prediction horizon - 1.5 seconds and the NLP convergence tolerance is set to 10^{-5}. The average computation time for the control is 3 ms and the kite, path and controller parameters used in the simulation study are: $v_w = 5\ [m/s]$, $\mathbf{E} = 7$, $L = 5\ [m]$, $h = \frac{\pi}{6}$, $a = 0.2$, $\dot{\theta}_{ref} = 1.0$, $Q = diag\{500, 500\}$, $R = diag\{0.001, 0.1\}$, $W = diag\{1.0\}$.

Figures 2.6 to 2.9 show respectively the orbit tracking error, the computational delay, the normalised delay distribution, and the closed loop orbit (lemniscate). Furthermore, two computational experiments have been conducted to demonstrate the sensitivity of the computation time against the number of collocation points and prediction horizon length.

In the first experiment the five sets of simulation scenarios are conducted with the total number of Chebyshev collocation points equal to 8, 12, 16, 20 and 24 and each set consists of 10 scenarios with different initial conditions. For the multi-segment scheme, the order of polynomial approximation is chosen to be 4 within each segment. Figure 2.10 depicts the mean and the standard deviation of the computational delay for each of the simulation runs and it can be observed that in both cases computational delay grows almost linearly with the number of collocation points. The multi-segment scheme in this example is consistently faster and has a lower standard deviation of the computational delay after a certain number of collocation points thanks to the efficient sparsity exploiting direct linear algebra solver.

During the second study, the terminal time of the OCP was varied from 1 to 2.5 seconds and the increase of the integration period, in general, worsens the conditioning of the system of nonlinear equations resulting from an implicit integration scheme, or in particular the collocation equations (2.28). This in turn often leads to a higher number of iterations required to solve the corresponding NLP to a desired precision. In this experiment the number of collocation points was 16 for both single- and multi-segment schemes. Figure 2.11 indicates that in either case, the method is not very sensitive to the change of prediction horizon. The

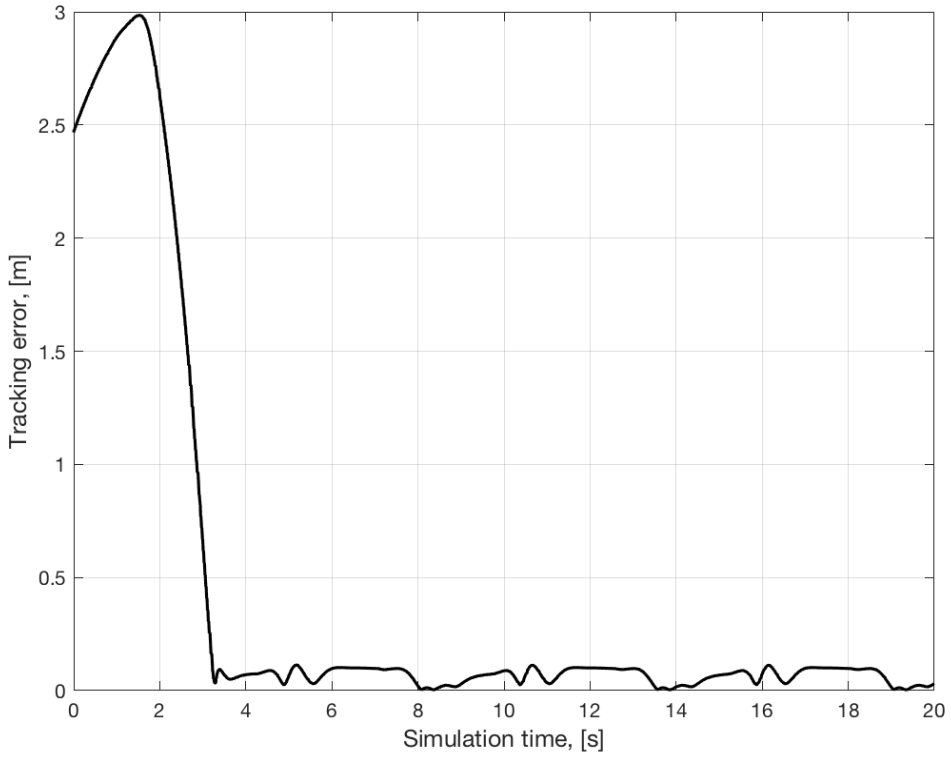


Figure 2.6 – Tracking accuracy of the NMPF controller.

computation time scales linearly with prediction horizon: 250% increase of the horizon leads to a 21% rise in computation time for the single-segment and 13% for multi-segment methods. It is important to note that the multi-segment method is less affected by the increase of horizon as the higher number of segments tends to improve the conditioning of (2.28). For the comparison study the same problem has been implemented using the PSOPT and ACADO toolboxes. The benchmark implementations can be found in open access at the link http://github.com/petlist/simple_kite_benchmark. In the following the important features and differences to the PolyMPC package are highlighted. A short summary of all three programs is presented in Table 2.1.

Since PSOPT uses a similar approximation strategy to PolyMPC, it has been configured to have the same number of collocation points, subintervals as well as convergence tolerances. The mesh refinement was disabled during the simulation. Additionally, it was set to interface the same NLP solver, therefore, the timing differences are mostly due to the source code optimizations and utilised libraries. Some of the key differences are summarised below:

- It follows from Figure 2.12a that the software is not optimized for real-time predic-

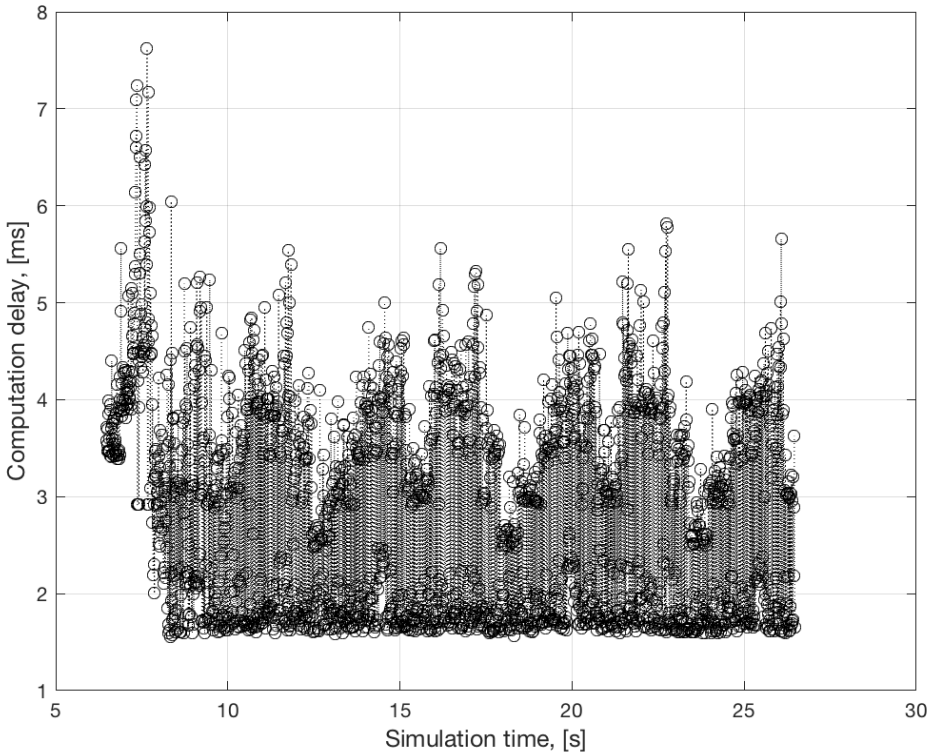


Figure 2.7 – Computational delay of the path following controller.

tive control of systems with fast dynamics, since the computation time was nearly 30 times slower. It was not possible to use the code in the real-time simulation, so the PSOPT-based controller was applied on the data collected from the previously described simulation.

- PSOPT uses CppAD [65] syntax to formulate the OCP, which requires implementation of system equations and cost function in scalar form. This increases the chance of a coding error, particularly in cases with many matrix calculations.

The C++ interface to the ACADO code generation tool was used to create the orbit tracking controller. A multiple-shooting discretization with 12 intervals has been chosen and within each interval the Runge-Kutta scheme of order 4 is utilised for trajectory and sensitivity propagation. QPOASES [47] is set as the quadratic problem solver with the exact Hessian. The number of RTI [66] iterations at each sampling time is defined by the KKT tolerance. Some of the obtained results are given below:

- The ACADO-generated controller is sufficiently fast for real-time control of the considered system. The average computation delay over several periods is only a fraction of

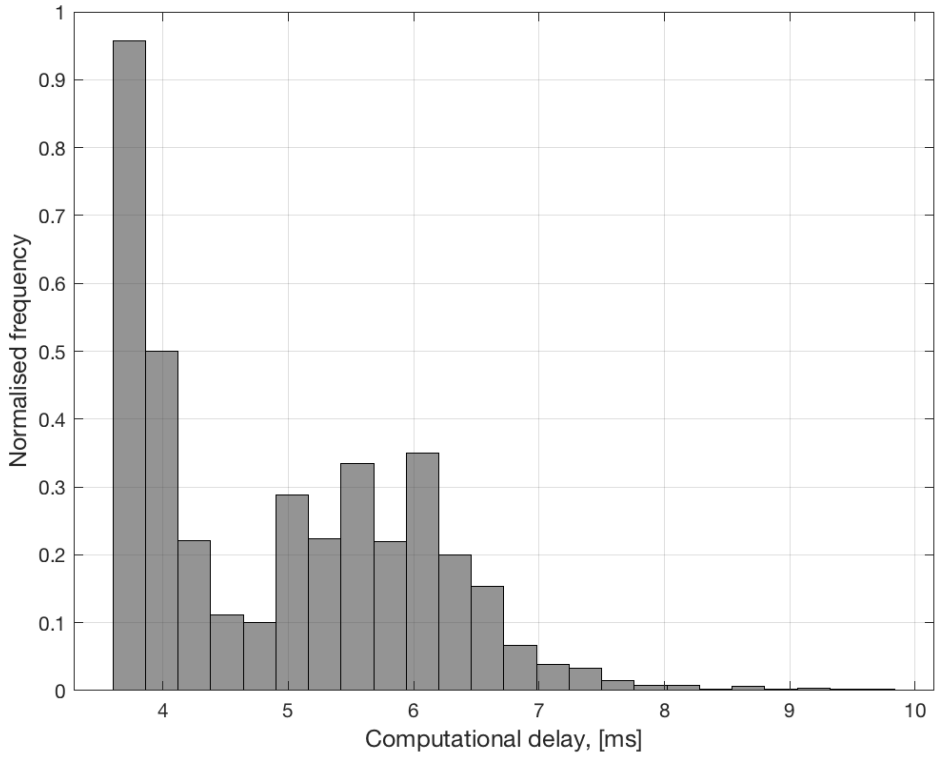


Figure 2.8 – Computational delay of the path following controller.

a millisecond larger than that of PolyMPC. Interestingly, as becomes clear from Figure 2.12b, the standard deviation of the delay is significantly lower than the ones of PolyMPC and PSOPT. This is probably due to the more efficient handling of warm-starting in the ACADO solver.

- Important to note that unlike PSOPT or PolyMPC, the ACADO code generation tool does not directly support economic cost functions. Therefore, the particular benchmark problem has had to be reformulated to the Mayer form OCP.

- ACADO uses its own modelling language which, to the best of the authors understanding, does not support vector operations. This is usually an error-prone approach when dealing with complex motion equations, for example, aircraft flight modelling. PolyMPC on the other hand is designed to work with the CasADi and Eigen data types that naturally support vector operations.

- The code generation tool in ACADO is an attractive feature as it allows the creation of fast and memory-lean embeddable predictive controllers. The negative side of this feature is the restricted debugging and analysis capabilities since the generated C code is usually

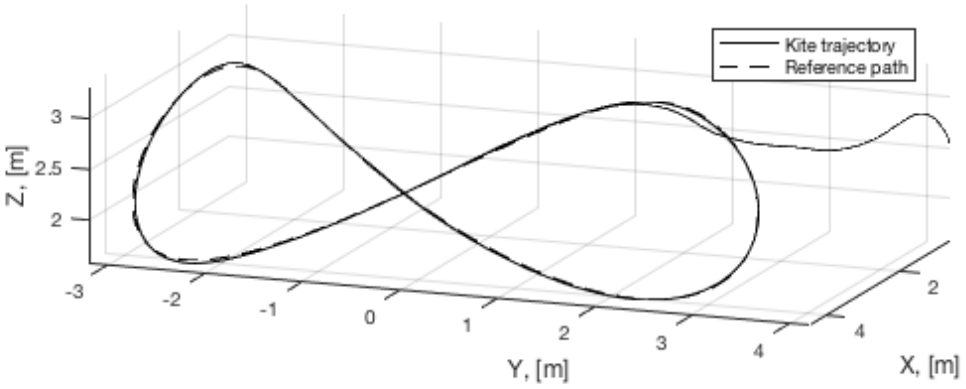


Figure 2.9 – Closed loop path following performance.

hard to understand. In contrast, PolyMPC is designed to be more developer friendly while still providing comparable runtime performance and embedding capabilities.

Name	PolyMPC	PSOPT	ACADO
Delay : mean [ms]	3.11 (CasADi)	100.51	3.29
Delay : std [ms]	1.13 (CasADi)	10.33	0.24
Build system	CMake	Makefile	CMake
Modelling language	CasADi/Eigen	CppAD	ACADO
Supported platforms	Linux/Mac OS/Windows	Linux	Linux/Mac OS/ Windows (req. gcc)
Embedded	Yes	No	Yes
Interfaces	C++	C++	C++/Matlab

Table 2.1 – Timings and software summary

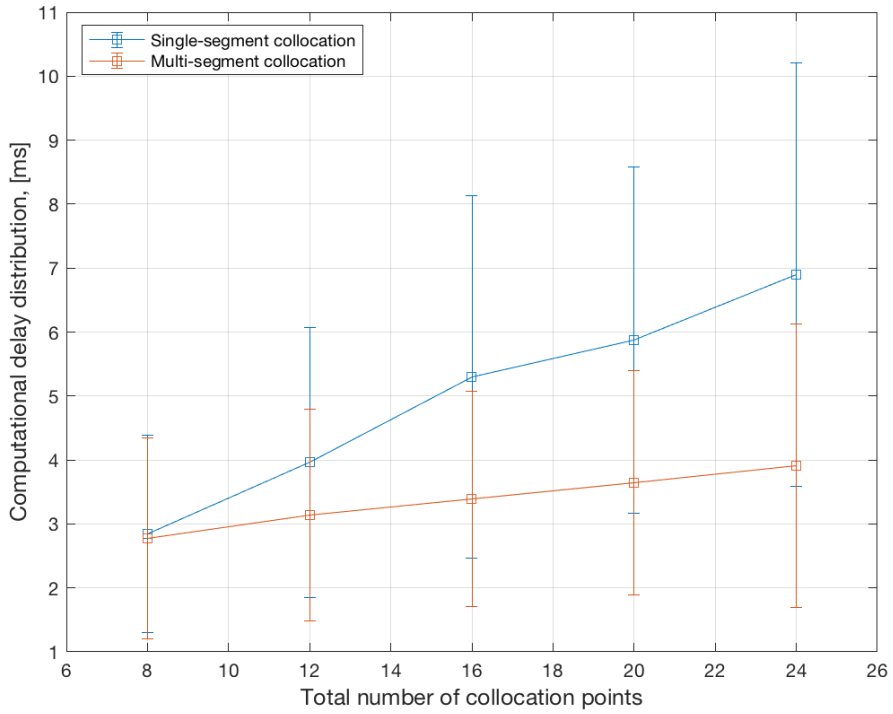


Figure 2.10 – Comparison of computational delays of single- and multi-segment Chebyshev collocation.

Embedded benchmark

The kite control problem was benchmarked on the STM32 microcontroller, the Odroid XU4 and a MacBook Pro. The specifications of the embedded platforms is summarised in Table 2.2. The MCU benchmark code and tutorial on how to deploy *PolyMPC*-based controllers on MCU can be found here: https://github.com/LA-EPFL/polympc-stm32f7.

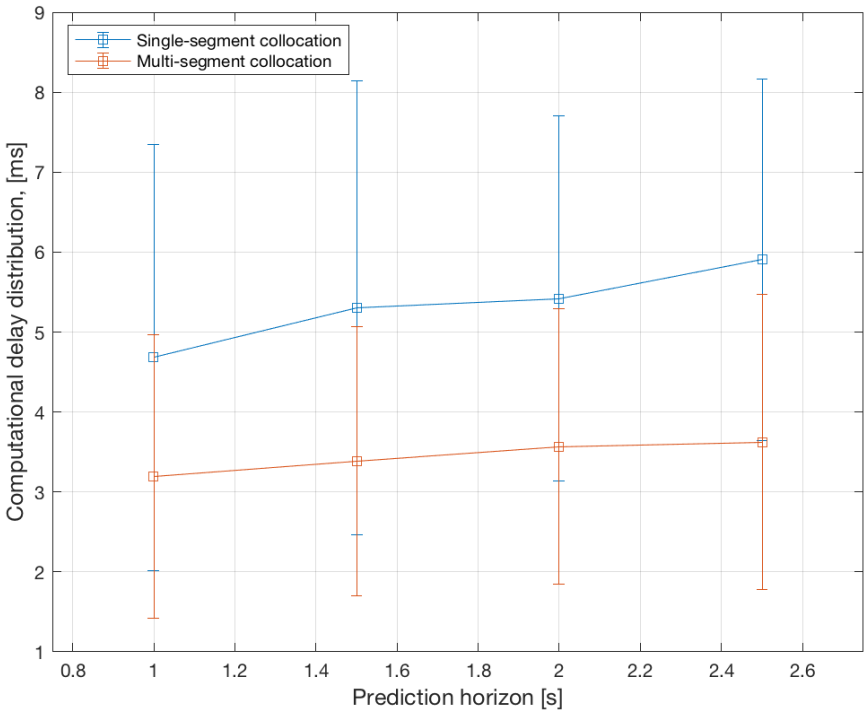


Figure 2.11 – Comparison of computational delays of single- and multi-segment Chebyshev collocation.

Name	Odroid XU4	Nucleo-F767ZI
Platform	Samsung Exynos5422	STM32F767ZI
CPU	8x ARM Cortex™-A15/A7	ARM Cortex™-M7
Architecture	ARMv7-A (32bit)	ARMv7E-M (32bit)
Acceleration	FPU, NEON SIMD, DSP	FPU (DP+SP), DSP
Clock	2GHz	216MHz
RAM	2GB LPDDR3	512KB SRAM
Storage	16GB eMMC/SDCard	2MB flash
OS	Ubuntu 18.04	bare metal
Dimensions	83mm x 58mm	133mm x 70mm
Power consumption	10W - 20W	<1.5W

Table 2.2 – Embedded hardware comparison

For *Nucleo-F767ZI* during embedded testing, the *Eigen* interface was employed with dense linear algebra and single-precision floating point scalars. On the *MacBook* and the *Odroid XU4*,

sparse linear algebra with double-precision floating point numbers was used. Additionally, *NEON* SIMD optimisation was set for *Odroid*. The timing results are shown in Table 2.3

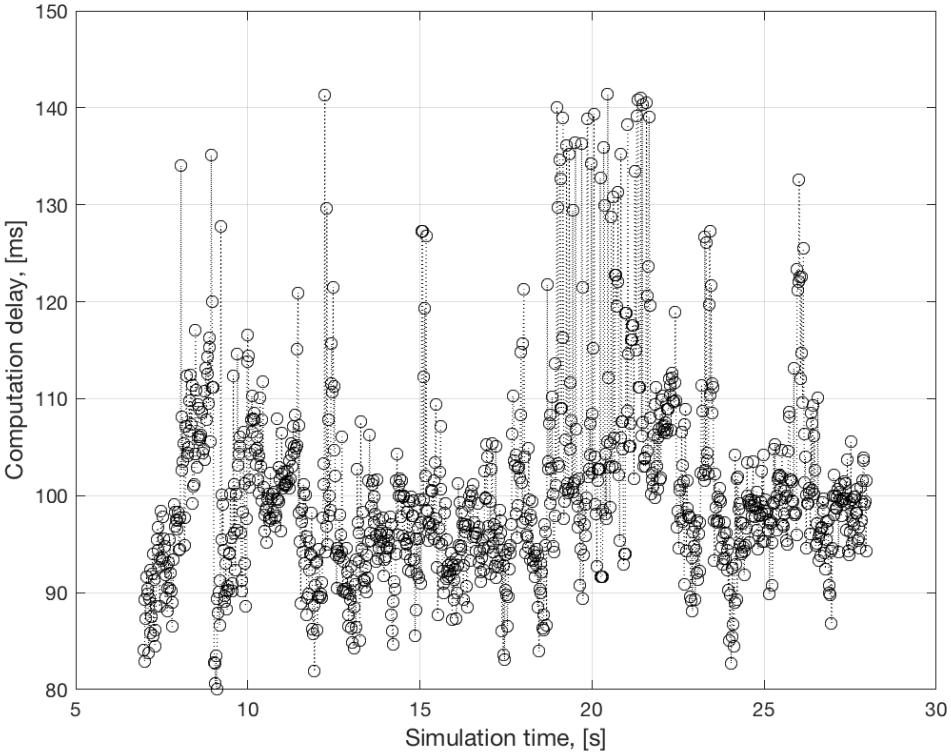
platform	solve time [ms]	factor
MacBook Pro	0.91	1.0
Odroid XU4	4.31	4.8
STM32F767ZI	41.86	46.0

Table 2.3 – Kite benchmark solve time comparison

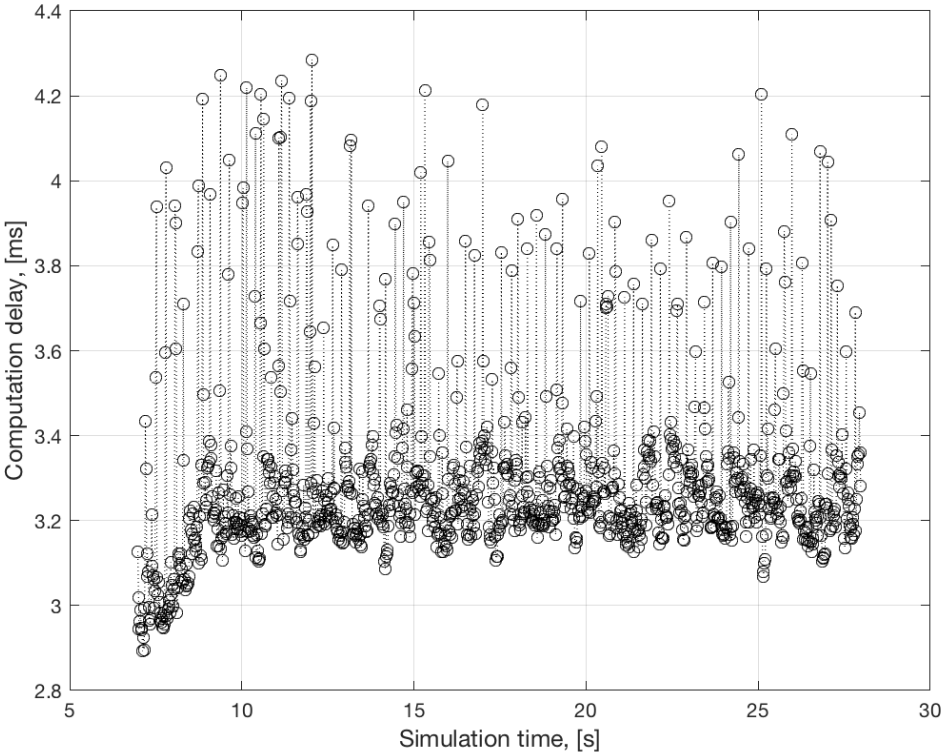
It can be observed that our optimised *Eigen*-based implementation is more than three times faster on average than *CasADi* version or *ACADO*. On the memory consumption aspect, the kite controller object uses 60984 bytes of memory when using single precision floating point types. This fits easily into the 512Kbytes available on the microcontroller.

2.6 Summary

In this chapter, we presented *PolyMPC*, a free and open-source software toolkit for real-time embedded optimisation and optimal control. Thanks to its modular structure and flexibility, the software is helpful for control practitioners and researchers in the field of optimisation and optimisation-based control as it allows full control over the source code. Some of the algorithmic modules already include several ADMM QP solvers, an SQP solver, Chebyshev and Legendre pseudospectral collocation codes, polynomial interpolation and approximation modules. All algorithmic components support dense and sparse linear algebra, and provide an option for a static memory allocation for embedded deployment even on bare metal platforms. *PolyMPC* performance is proven in numerical benchmarks and real-life experiments.



(a) Computational delay of the PSOPT-based path following controller.



(b) Computational delay of the ACADO-based path following controller.

Figure 2.12 – Computational delays produced by PSOPT and ACADO

Part II

Identification and Predictive Flight Control of Rigid-Wing Airborne Wind Energy Kites

Chapter 3

Introduction

Airborne Wind Energy (AWE) is an emerging technology aimed at harvesting wind energy at higher altitudes than conventional wind turbines. Since traditional turbines were widely introduced to the energy sector about four decades ago, blade sizes and consequently the supporting structures have been constantly increasing to maximise the energy yield [67]. Despite the steady growth of world-wide installations, wind variability poses challenges for continuous energy supply and as a result, lead to financial losses to wind farms and electric grid operators [68]. Reaching higher altitudes and therefore steadier winds is difficult due to structural limitations, but recent studies show that by reaching altitudes of approximately 300 to 800 meters, the number of places on earth suitable for wind energy harvesting is augmented significantly [69, 70]. AWE technology is being developed to address the aforementioned limitations of conventional wind turbines by replacing massive blades with an aerodynamic wing connected to a ground station by a flexible and relatively lightweight tether which potentially allows AWE kites, to access much higher altitudes.

There exist several AWE technological concepts depending on the flight operation and electricity production principles. In this thesis, we consider the two, most commonly adopted concepts: lift-mode and drag-mode. The two concepts differ in the way they transform wind energy into electricity and, therefore, in their operation strategy. In lift mode, the aerodynamic force generated by the aircraft flying cross-wind is used to reel out a tether from the ground station and rotate a drum connected to the rotor of the generator. The product of the tether force and the reel-out speed is proportional to the mechanical power that can be converted to electricity. Once the maximum length of the tether is reached, the system switches to a retraction (or recovery) phase where the generator works as a motor and reels in the tether while the aircraft flies towards the ground station in a low-tension regime, typically with high angle of attack. In drag mode, ram-air turbines oriented towards the airstream are mounted on

the aircraft to convert kinetic energy to electricity that is transmitted to the ground station via a conducting tether. For the aircraft, the generator resistance corresponds to additional drag. Besides these operating strategies, existing AWE prototypes differ in their wing structure (rigid or flexible), number of flying vehicles, on-board or ground-based actuation. A comprehensive overview of the existing AWE technologies and concepts can be found in [71, 72, 73].

Even though there is no clearly superior technology, the trend among AWE companies and practitioners over the past few years has been shifting towards rigid-wing single-line kites. The focus of this thesis is on modelling, identification and predictive control of this type of kite. We develop a small-scale, low-cost prototype AWE kite, perform identification and validation of the flying vehicle and tether models and present our work towards embedded hierarchical nonlinear model predictive control for geometric path following.

3.1 State of the art

Flight control of AWE kites is a complex problem since the system is inherently unstable, highly nonlinear, and operates in an uncertain environment. Standard steady-state flight assumptions [74, 75] cannot be applied since the aircraft moves at high accelerations and aerodynamic angles.

For fixed-wing single-line kites, the control systems typically have a modular structure. Rapp et al. [76] propose a complete pumping cycle flight control algorithm based on a state machine and a cascaded path-following controller. The guidance layer defines the desired path and course angles using the projection of the aircraft onto the local tangential plane of a continuously parametrised geometric path. These angles are then used in the path-following layer to compute the bank angle and angle of attack which, in turn, are transformed to roll, pitch, and yaw rates by the attitude loop. Finally, the actuator inputs are deduced by dynamic model inversion. Control in the radial direction is realised through the winch system. Finally, a state machine is used to switch among the operating regimes. Sieberling [77] represents the desired kite trajectory as a collection of scheduled waypoints in the Cartesian reference frame and controls longitudinal and lateral motions independently using cascaded PID loops. Fagiano et al. [78] present a flight control system specific to lemniscate-type trajectories for a drag-mode kite. In this approach, two particular waypoints are placed on a sphere defined by the tether length. The reference course angle is computed as an angular distance between a waypoint and an aircraft when projected on the local tangential plane. The path angle is proportional to the altitude difference between waypoint and aircraft. Once the guidance signal is computed, it is transformed to yaw and pitch rate signals and tracked by low level controllers.

Nonlinear Model Predictive Control (NMPC) can be seen as another guidance method for

rigid-wing AWE kites. Stastny et al. [79] employ a model predictive controller with a first-order kinematic kite model similar to the ones presented in [80, 81] to generate a yaw rate reference for tracking of a tilted circle. The yaw rate is then tracked by low-level PID controllers. The approach was experimentally validated using a small-scale prototype. The NMPC algorithm in their work provides a compromise between circle tracking accuracy and local power optimisation. In [82], the authors take a different approach and let the NMPC algorithm directly control the actuators. A 6 degree-of-freedom (DoF) airplane model is used for predictions and a power-optimal reference trajectory is assumed to be available. The tether is modelled as a geometric constraint which leads to the differential algebraic equations (DAE) constraint in the optimal control formulation. A simulation study was carried out using the *ACADO* toolkit [24]. A key limitation of the solution is the computation time of the NMPC implementation. The authors report an average computation of over 180 milliseconds on a desktop computer which is not suitable for real-time applications. Moreover, one will typically observe a 5x to 10x increase of the computation time when ported to high-end embedded platforms.

3.2 Contributions

Experimental validation of advanced control schemes for rigid-wing AWE kites is particularly challenging due to the high design and manufacturing costs of the prototypes. For this reason, industrial partners are often reluctant to adopt novel control approaches published by the academic community. To address this issue and foster experimental studies of AWE flight control systems, we have developed a complete small-scale prototype of a rigid-wing AWE system capable of operating in lift mode. The prototype features a 1.8 m wing-span foam aircraft equipped with an inertial measurements unit (IMU), global positioning (GPS), an airspeed sensor, a flight controller, and a companion computer running embedded Linux for demanding computations. A portable and fully autonomous ground station comprises high-precision line angle encoders, a load cell for tension measurement and an electric motor for tether control. The versatile flight management software based on *ROS* [60] allows for quick integration of new control algorithms, flexible mission management and switching between regimes and algorithms.

The second contribution of this part is a methodology for identification of the kite model parameters. Here we adopt a multi-experiment averaging technique from [83] and use standard assumptions on the separability of the longitudinal and lateral motions. For each flight experiment, a perturbed 3-2-1-1 input sequence is applied in open-loop fashion. After the experimental data is collected, the *PolyMPC* toolbox is used to solve a large-scale dynamic optimisation problem over several experiments.

Finally, we propose a nonlinear path-following optimal control formulation that is used as a fast trajectory optimisation algorithm in a hierarchical flight control system. The proposed approach allows for any geometric path to be flown while respecting the flight envelope constraints. The efficient embedded implementation is shown to be on average 5x faster than the one presented in [82] and is tested experimentally on the embedded computer. A novel delay compensation mechanism allows running the NMPC algorithm in real-time on the prototype.

Chapter 4

Modelling of a Rigid-Wing AWE Kite

This section presents a control-oriented mathematical model of a single-line rigid-wing AWE kite. In general, mathematical models for this type of AWE kite differ in the level of detail of the aerodynamic wing simulation and tether. For control applications, aerodynamic properties of the flying vehicle are typically characterised by a set of aerodynamic coefficients which can be obtained with specialized computation fluid dynamics (CFD) software by perturbing aircraft around equilibrium flight conditions [75]. In AWE literature examples of such models can be found in [84, 85, 76]. A reduced set of coefficients that neglects the influence of angular motion on the lift, drag and aerodynamic forces is used in [82]. Tethers are modelled as a geometric constraint (rigid rod) [82, 85], lumped mass models where mass points are connected by springs and dampers [86, 87] or a single visco-elastic element [84] as in this thesis. In our work, a full set of aerodynamic coefficients is employed for the flying vehicle simulation, and a smooth approximation of the tether tension force allows using the model both for simulation and dynamic optimisation.

4.1 Modelling: Aircraft

We utilize the aircraft dynamic equations of motion from [75]. The aircraft is assumed to operate in a close neighbourhood of the ground station (GS) and, thus, the effects of the earth shape and rotation as well as variation of the gravity field are neglected. Quaternions are chosen to represent rotations for their computational efficiency and absence of gimbal-lock effect. In the following, we briefly introduce three relevant reference frames that appear in the model.

Body Reference Frame (BRF) the origin is fixed at the center of mass (CoM) of an aircraft, longitu-

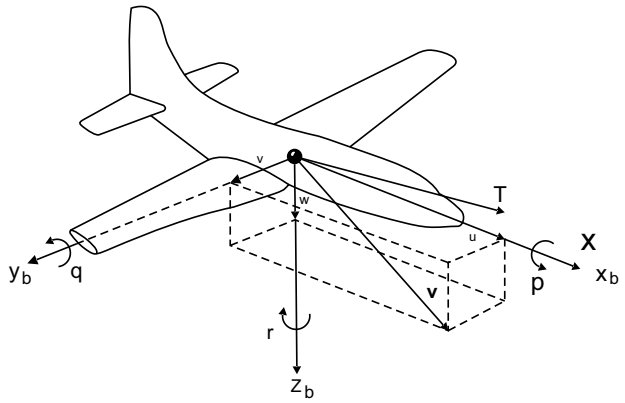


Figure 4.1 – Body Reference Frame. Adopted from [75]

dal axis (Ox_b) - an axis drawn through the body of the vehicle from tail to nose in the normal direction of flight, lateral axis (Oy_b) - an axis is pointing to the starboard and parallel to the wings of a winged aircraft, normal axis (Oz_b) - an axis drawn from top to bottom and perpendicular to the other two axes.

Inertial Reference Frame (NED) the origin is fixed at an any point (at the groundstation (GS) for instance) and does not move over time relative to the ground, (Oz_n) - axis is pointing downward (gravity vector), (Ox_n) and (Oy_n) axes are perpendicular to (Oz_n) and point to the geographic North and East directions correspondingly.

Aerodynamic Reference Frame (ARF) the aerodynamic forces can be conveniently expressed in ARF where the origin is fixed at the CoM of a vehicle, (Ox_a) axis points towards apparent velocity vector, (Oz_a) axis is perpendicular to (Ox_a) and points down, (Oy_a) complements these two axes to a right triangle.

We use the following conventions: a vector $\mathbf{v}$ in reference frame 'a' is given by ${}_a\mathbf{v}$. The operator $(\cdot)$ denotes the transformation of a vector defined in frame 'a' to frame 'b' using a unit quaternion:

$$ {}_b\mathbf{v} = \mathbf{q}_{ba} \cdot {}_a\mathbf{v} := \mathbf{q}_{ba} \otimes \begin{bmatrix} 0 \\ {}_a\mathbf{v} \end{bmatrix} \otimes \mathbf{q}_{ba}^{-1} \tag{4.1} $$

The state vector of a 6 DoF rigid-body model is defined as

$$\mathbf{x} = \begin{pmatrix} {}_b\mathbf{v}_K \\ \boldsymbol{\omega} \\ \mathbf{r} \\ \mathbf{q} \end{pmatrix} \tag{4.2}$$

where ${}_b\mathbf{v}_K = (u, v, w)^T$ is the flight path velocity and $\boldsymbol{\omega} = \left(p, q, r\right)^T$ is the angular velocity in the body frame, $\mathbf{r} = \left(x, y, z\right)^T$ denotes the position in the NED reference frame, and $\mathbf{q} = \left(q_w, q_x, q_y, q_z\right)_{nb}^T = \mathbf{q}_{nb}$ is the quaternion that describes the rotation of the body frame with respect to the NED frame. The control vector is defined as

$$\mathbf{u} = \begin{pmatrix} F_{Thr,0}, & \delta_e, & \delta_r, & \delta_a \end{pmatrix}^T \tag{4.3}$$

with the static thrust (at zero airspeed) $F_{Thr,0}$, and the control surface deflections of the elevator, rudder, and ailerons, δ_e, δ_r, δ_a, respectively. The equations of motion in the body frame are

$${}_b\dot{\mathbf{v}}_K = \frac{1}{m}\sum_i {}_b\mathbf{F}_i - \boldsymbol{\omega} \times {}_b\mathbf{v}_K \tag{4.4}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\left(\sum_i {}_b\mathbf{M}_i - \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}\right) \tag{4.5}$$

where ${}_b\mathbf{F}_i$ and ${}_b\mathbf{M}_i$ are the external forces and moments acting on the aircraft, m is the aircraft mass and $\mathbf{I}$ contains the moments of inertia where I_{xy} and I_{yz} components are neglected [75]

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix} \tag{4.6}$$

The evolution of the aircraft position and attitude of the aircraft are described by the kinematic equations

$$\dot{\mathbf{r}} = \mathbf{q} \cdot {}_b\mathbf{v}_K, \tag{4.7}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \begin{pmatrix} 0 \\ \boldsymbol{\omega} \end{pmatrix} \tag{4.8}$$

For a tethered airplane, the total force consists of aerodynamic, thrust, gravitational, and tether components. Assuming the thrust vector is aligned with the body x-axis, the total moment

consists of an aerodynamic and a tether portion:

$$\begin{aligned} \sum_i {}_b\mathbf{F}_i &= {}_b\mathbf{F}_A + {}_b\mathbf{F}_{Thr} + {}_b\mathbf{F}_G + {}_b\mathbf{F}_T \\ \sum_i {}_b\mathbf{M}_i &= {}_b\mathbf{M}_A(+{}_b\mathbf{M}_T) \end{aligned} \tag{4.9}$$

Aerodynamics, Propulsion and Gravity

With the apparent air velocity

$${}_b\mathbf{v}_A = (u_A, v_A, w_A)^T = {}_b\mathbf{v}_K - \mathbf{q}_{bn} \cdot {}_n\mathbf{v}_W \tag{4.10}$$

where ${}_n\mathbf{v}_W$ is a wind velocity vector expressed in the NED frame, the angle of attack (AoA) α, and the angle of sideslip β are defined as

$$\begin{aligned} \alpha &= \arctan\left(\frac{w_A}{u_A}\right) \\ \beta &= \arcsin\left(\frac{v_A}{V_A}\right) \end{aligned} \tag{4.11}$$

with $V_A = \|{}_b\mathbf{v}_A\|_2$. The aerodynamic lift ($L$), drag ($D$) and side ($Y$) forces can be estimated using the following equations [75]:

$$\begin{aligned} L &= \bar{q}S \cdot C_L \\ D &= \bar{q}S \cdot C_D \\ Y &= \bar{q}S \cdot C_Y \end{aligned} \tag{4.12}$$

Here C_L, C_D, C_Y denote dimensionless aerodynamic coefficients which can be computed using the following linear approximations:

$$\begin{aligned} C_L &= C_{L0} + C_{L\alpha}\alpha + \frac{c}{2V_A} C_{Lq} q + C_{L\delta e}\delta_e \\ C_D &= C_{D0} + \frac{C_L^2}{\pi e \Lambda} \\ C_Y &= C_{Y\beta}\beta + \frac{b}{2V_A}\left(C_{Yp}p + C_{Yr}r\right) + C_{Y\delta r}\delta_r \end{aligned} \tag{4.13}$$

Similarly, for aerodynamic roll (l), pitch (m) and roll (n) moments we have

$$\begin{aligned} l &= \bar{q}Sb \cdot C_l \\ m &= \bar{q}Sc \cdot C_m \\ n &= \bar{q}Sb \cdot C_n \end{aligned} \tag{4.14}$$

and

$$\begin{aligned} C_l &= C_{l\beta}\beta + \frac{b}{2V_A}\left(C_{lp}p + C_{lr}r\right) + C_{l\delta r}\delta_r + C_{l\delta a}\delta_a \\ C_m &= C_{m0} + C_{m\alpha}\alpha + \frac{c}{2V_A}C_{mq}q + C_{m\delta e}\delta_e \\ C_n &= C_{n\beta}\beta + \frac{b}{2V_A}\left(C_{np}p + C_{nr}r\right) + C_{n\delta r}\delta_r + C_{n\delta a}\delta_a \end{aligned} \tag{4.15}$$

with the reference quantities S, b, c, e, and Λ for the wing area, span, chord length, Oswald efficiency, and aspect ratio, respectively, and the dynamic pressure $\bar{q} = \rho/2 \cdot V_A^2$. $C_{L0}, C_{D0}, CL_\alpha$ are aerodynamic coefficients, $C_{L\_}, C_{Y\_}, C_{l\_}, C_{m\_}, C_{n\_}$ are dimensionless stability coefficients and control derivatives. These coefficients characterise dynamics of the aircraft for a defined trimmed flight condition and can be found through wind tunnel test, identification flight experiments, or computational fluid dynamics (CFD) simulations.

The total aerodynamic forces and moments in the body reference frame are

$${}_b\mathbf{F}_A = \mathbf{q}_{ba} \cdot \begin{pmatrix} -D \\ Y \\ -L \end{pmatrix}, \quad {}_b\mathbf{M}_A = \begin{pmatrix} l \\ m \\ n \end{pmatrix} \tag{4.16}$$

with $\mathbf{q}_{ba} = \mathbf{q}_2(\alpha) \otimes \mathbf{q}_3(-\beta)$. We assume that the aerodynamic coefficients are constant, and that the symmetric aircraft does not produce lateral forces or moments for zero sideslip. The thrust and gravitational forces are

$${}_b\mathbf{F}_{Thr} = \begin{bmatrix} F_{Thr} \\ 0 \\ 0 \end{bmatrix}, \qquad {}_b\mathbf{F}_G = \mathbf{q}_{bn} \cdot \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

with the local gravitational acceleration g. The thrust magnitude depends on the airspeed and is approximated from the static thrust command with a second order polynomial [88]:

$$F_{Thr} = F_{Thr,0} \cdot \left(p_2 V_A^2 + p_1 V_A + 1\right) \tag{4.17}$$

where (p_2, p_1) are polynomial coefficients obtained with of a propeller design software.

4.2 Modelling: Ground Station and Tether

This section extends the modelling approach presented in [84]. The tether force comprises the drag, weight and tension components and can be written in the BRF as:

$${}_b\mathbf{F}_T = {}_b\mathbf{F}_{ten} + {}_b\mathbf{D}_T + {}_b\mathbf{W}_T \tag{4.18}$$

This force is applied at the tether attachment point relative to the CoM: ${}_b\mathbf{r}_t = (x_t, 0, z_t)^T$, which results in the torque

$$ {}_b\mathbf{M}_T = {}_b\mathbf{r}_t \times {}_b\mathbf{F}_T \tag{4.19} $$

Tether Tension

The tension force is modelled as a visco-elastic element and points towards the GS. Given the aircraft position ${}_n\mathbf{r}$ and velocity ${}_n\mathbf{v}_K$ in the NED frame, the force can be computed using the following formula

$$ {}_n\mathbf{F}_T = \frac{{}_n\mathbf{r}}{\|{}_n\mathbf{r}\|} \left[K_s(l_t - \|{}_n\mathbf{r}\|) - Kd \frac{{}_n\mathbf{r}^T \, {}_n\mathbf{v}_K}{\|{}_n\mathbf{r}\|} \right] H(l_t - \|{}_n\mathbf{r}\|) \tag{4.20} $$

Where Ks, Kd are the spring and damping coefficients of the line, l_t is a current length of the tether and $H(\cdot)$ is a Heaviside function which activates the tension component of the tether force only when $\|{}_n\mathbf{r}\| > l_t$. In the BRF we have:

$$ {}_b\mathbf{F}_{ten} = \mathbf{q}_{bn} \cdot {}_n\mathbf{F}_T \tag{4.21} $$

Tether Drag

The tether drag results from the component of the apparent velocity that is perpendicular to the tether. Assuming a straight tether, a horizontal projection of the apparent velocity in the NED frame is given by:

$$ {}_n\mathbf{v}_A = \mathbf{q}_{nb} \cdot {}_n\mathbf{v}_K - {}_n\mathbf{v}_W $$

$$ {}_n\mathbf{v}_{A,hor} = \begin{bmatrix} {}_n\mathbf{v}_{A,x} \\ {}_n\mathbf{v}_{A,y} \\ 0 \end{bmatrix} $$

The tether drag force is orthogonal to this projected velocity. Assuming that the airspeed increases linearly from 0 at the ground station to ${}_l\mathbf{v}_{A,hor}$ at the aircraft position, the drag can be computed as

$$ {}_n\mathbf{D}_T = -\frac{1}{8} \rho \, {}_n\mathbf{v}_{A,hor} \left\| {}_n\mathbf{v}_{A,hor} \right\| c_\perp d_T l_t $$

$$ {}_b\mathbf{D}_T = \mathbf{q}_{nl} \, {}_b\mathbf{D}_T $$

where ρ is the air density, $c_\perp, d_T$ are diameter and drag coefficient of the tether correspondingly.

Tether Weight

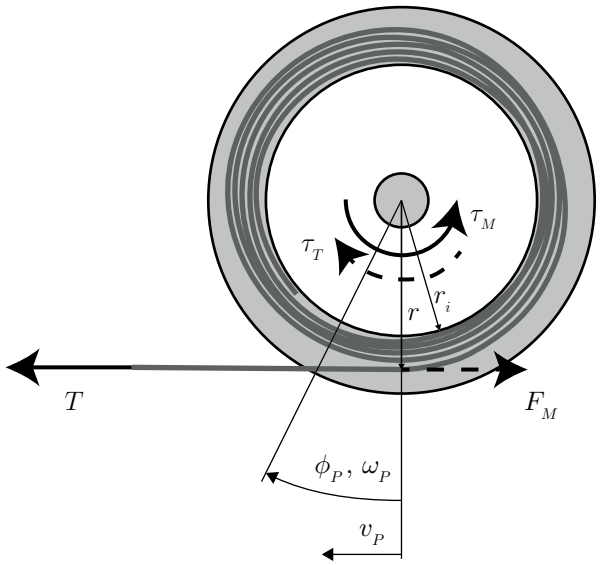


Figure 4.2 – Definition of the forces and moments acting on the pulley.

The total tether mass can be written as

$$m_T = c_m l_t \tag{4.22}$$

where

$$c_m = \rho_T A_T$$

expresses the mass per meter of tether. Assuming that the aircraft carries the whole tether mass the weight portion in the BRF can be computed as

$${}_b\mathbf{W}_T = \mathbf{q}_{bn} \otimes \begin{bmatrix} 0 \\ 0 \\ m_T \cdot g \end{bmatrix} \tag{4.23}$$

Winch Dynamics

The pulley dynamics can be expressed by its angular position φ_P and velocity ω_P, as defined in Figure 4.2. The angular dynamics of the winch is defined by the sum of external torques divided by the angular inertia I of moving parts:

$$\ddot{\varphi}_P = \frac{1}{I}\sum_i \tau_i \tag{4.24}$$

In our case, I is dominated by the angular inertia of the motor. The tether tension $T = \|{}_n\mathbf{F}_T\|$ applies at the variable radius r, which results in a torque $\tau_T = T \cdot r$ that acts on the pulley. The

motor applies a torque τ_M in the opposite direction so that the angular acceleration of the pulley equals

$$\ddot{\varphi}_P = \frac{1}{I}(\tau_T - \tau_M). \tag{4.25}$$

The angular velocity can be converted to its linear equivalent:

$$\begin{aligned} v_P &= \dot{\varphi}_P r \\ \dot{l}_t &= v_P \end{aligned} \tag{4.26}$$

The radius r depends on the tether diameter and the number of turns on the pulley. To convert the tether tension to its torque equivalent on the pulley or the motor torque to its force equivalent, the radius must be approximated by a model. The equivalent quantities are drawn with dashed lines in Figure 4.2.

Power Output

The mechanical power that is transmitted to or from the ground station equals

$$P = T v_P, \tag{4.27}$$

where T is the tether tension and v_P is the linear velocity of the pulley. As for the tension control in the following section, we assume to have mainly a stationary case where the tension equals the torque equivalent induced by the motor. Neglecting all losses in the energy conversion from the battery to the winch dynamics:

$$P = F_M v_P \tag{4.28}$$

with the motor force

$$F_M = \frac{\tau_M}{r}. \tag{4.29}$$

This means, when we reel out the tether at the velocity $v_P > 0$ against the motor force F_M, we transmit the power P to the ground station battery. On the other hand, as soon as the velocity equals 0, there is no power transmitted, no matter how high the motor force is.

4.3 Prototype: Hardware

The AWE system prototype presented in this thesis consists of a commercial electric foam glider connected to a ground station by a tether. The ground station is designed and built specifically for the experimental validation of AWE systems. The prototype allows performing automatic flight experiments with minimal operator intervention and serves as a research

Figure 4.3 – EasyGlider 4 by Multiplex.

Table 4.1 – EasyGlider4 technical data.

Wingspan	1.8 m
Overall Length	1.08 m
Take-off Mass (modified)	1.1 (1.35) kg

platform to experimentally study the performance of flight control systems.

Airplane

The airplane is the EasyGlider 4 by *Multiplex*. It is an electric glider made of ELAPOR foam, see Figure 4.3 and Table 4.1. While the elastic material makes it robust to rough landings or even minor crashes, these favorable properties come at the cost of slightly weaker flight performance.

Electronics and On-Board Computer

Every ordinary remote-controlled (RC) aircraft requires a battery, an RC receiver, and servos to move the control surfaces (ailerons, elevator, rudder). An electric motor also requires an *Electronic Speed Controller* (ESC) that converts the battery voltage to appropriate voltages for the motor and servos and also controls the motor at the low-level. For autonomous operation, it requires a *Flight Controller* (FC) and additional components according to the flight tasks and type of the vehicle.

In the system configuration, which can be seen in Figure 4.4, we use a Pixhawk4 Mini as the FC. It features a built-in *Inertial Measurement Unit* (IMU) and is connected to all crucial avionic components, such as a *Sensirion* SDP3x airspeed sensor, a Pixhawk GPS Module, and an RC receiver for manual control commands from the pilot. All these components are powered through the Pixhawk by a 3DR APM/Pixhawk Power Module v1.0. The servos

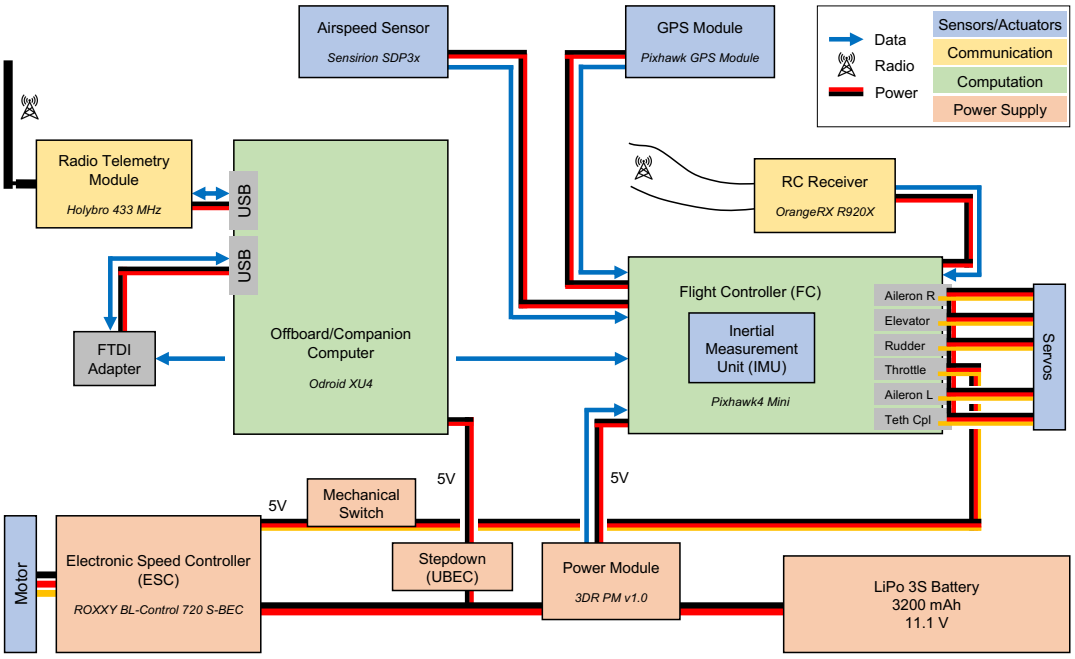


Figure 4.4 – Avionics Architecture.

and the ESC are directly connected to the Pixhawk. The ESC powers all servos through the Pixhawk servo rail circuit. To avoid the servos from randomly running to the extreme positions at system startup, an additional mechanical switch was added for the servo power supply. Finally, there is an Odroid XU4 *Offboard Computer* (often referred to in practice as a *Companion Computer*). The comparably power-demanding Offboard Computer allows for computationally expensive control and estimation algorithms to be executed. In the control system architecture, we use the Offboard Computer for all custom-written software and algorithms. The Offboard Computer is connected to the Pixhawk via an *FTDI* Adapter which bridges from USB to Pixhawk's JST-GH connector. As all relevant flight tasks run on the Odroid, the *Holybro* 433 MHz Radio Telemetry Module enables communication with the ground station. Optionally, the telemetry modules can be connected to the Pixhawk which is necessary for certain configuration tasks. Connecting the telemetry module to the Odroid enables transmission of parameters and real-time data related to the control and navigation algorithms. The Odroid and devices connected to it are powered by an additional step-down DC converter to ensure that the Odroid running power-demanding optimizations will not affect the safety-critical FC or servo power supply.

Assembly

The available space inside the fuselage is very constrained. Apart from increasing the total take-off weight, the majority of the additional components are placed in front of the center

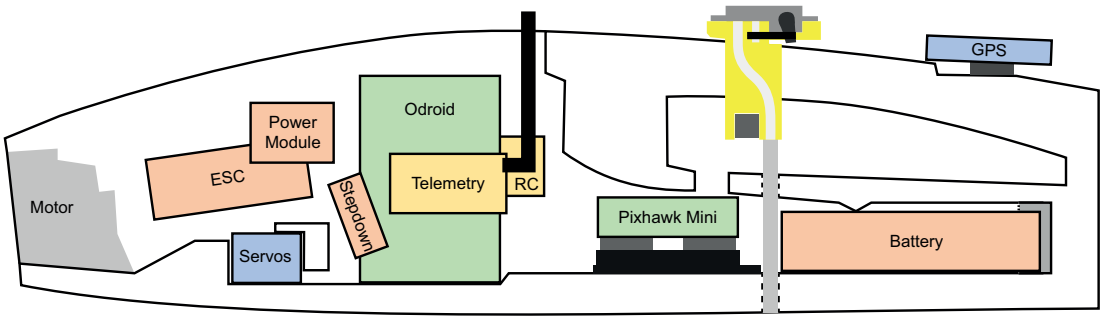


Figure 4.5 – Component arrangement in the fuselage.

of gravity (CoG) and make the aircraft nose-heavy. It is therefore desirable to place as many heavy components as possible close to the tail, so that less additional balancing weights are needed. For this reason, the battery is positioned in the very back of the fuselage which had to be extended by a few millimeters to allow the tether guiding tube to cross the fuselage.

Ideally, the Pixhawk is placed near the CoG because it contains the IMU. Due to electromagnetic interference the RC receiver and the telemetry module are placed on two opposing sides of the Odroid.

The arrangement of the components inside the fuselage is shown by Figure 4.5. The GPS module is fixed on the upper of the fuselage. Together with the tether coupling servo cable, its cable is connected to the Pixhawk.

Ground Station

The Ground Station is shown in Figure 4.6 The main components are a direct current (DC) motor, a pulley that spools the tether and is directly mounted on the motor shaft, and a set of rollers that redirect the tether towards the aircraft. In addition, there is a force sensor to measure the tether tension and two angular sensors that measure the orientation of the rod guiding the tether.

The tether has the diameter of 0.2 mm and a length of $2 \cdot 150$ m and is a fishing line that is rated for 13.6 kg of maximum load. The pulley has been designed to allow for high linear speeds. For safe and reliable operation, protections are placed around all rotating parts. The ground station is equipped with a 48 V battery which allows for its mobile and autonomous use.

The sensors are read out with an Arduino board and the motor is controlled by a *Maxon Motor* control unit. Both are connected to an ordinary laptop computer through USB, or an embedded computer if graphical visualisation is not necessary.

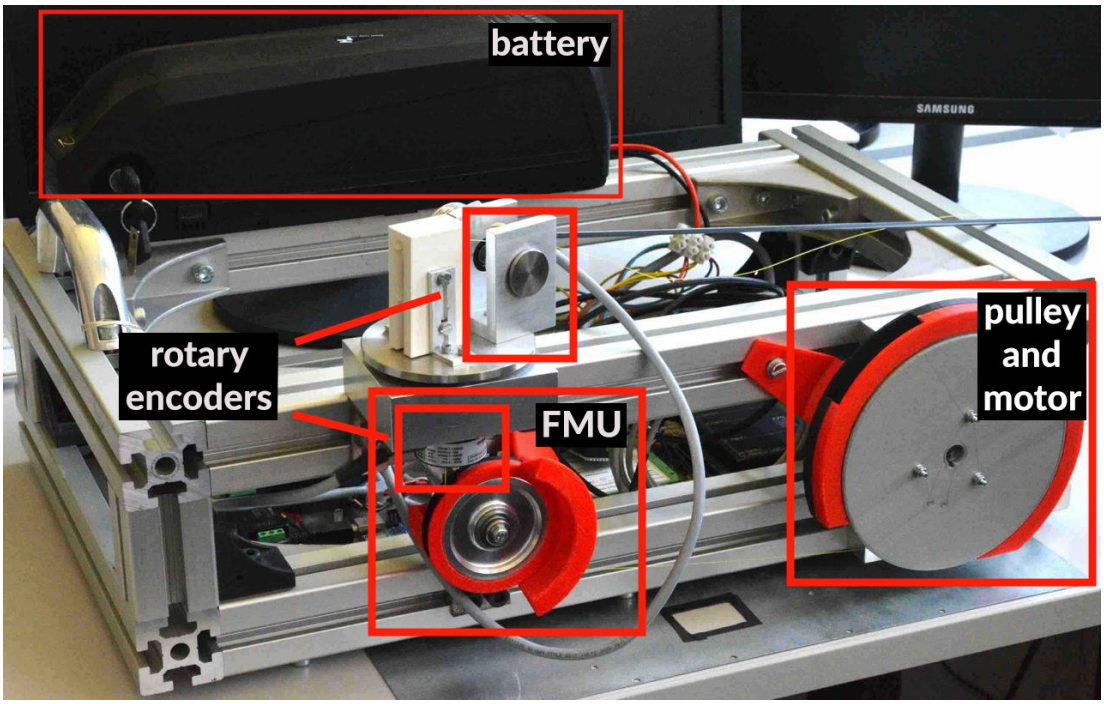


Figure 4.6 – Ground station. FMU statnds for the force measurement unit.

4.4 Prototype: Software

This section presents the software architecture that is used onboard the aircraft and, for the ground station as well as the developing environment. It contains tools for flight log analysis and a software-in-the-loop simulator.

Onboard

The Pixhawk FC runs the open-source autopilot software PX4, which manages all the sensors and actuators. It processes the sensor data by running an *Extended Kalman Filter* (EKF) to obtain state estimates which are sent over the serial port to the Odroid which is running the Ubuntu MATE 18.04 operating system. The inter-program communication is performed using *Robot Operating System* (*ROS*) Melodic [60]. Figure 4.7 demonstrates the *ROS* nodes that run on the Odroid, with the arrows showing the data flow between the nodes. The Odroid interfaces to the PX4, through the *MAVROS* package which publishes all received data streams as ROS topics over the serial interface.

The *Offboard Control* node is the essential autopilot implementation. Offboard Control receives control and flight mission related data such as position, attitude, linear and angular velocities, airspeed, manual control input from the RC receiver, and the status of the FC. It manages the system parameters with a parameter server and switches between flight control

modes depending on the current state and commands from the ground station or pilot. Within one of the flight control modes (attitude hold, identification experiments, AWE regime, etc), it computes all necessary values to navigate, guide and control the aircraft. The resulting control command is then sent to Pixhawk which implements the command. Additionally, it publishes some status and diagnostic information for flight analysis.

The *Air Telemetry* node provides the interface to the ground station. Air Telemetry processes parameter read and write requests from the ground, sends status information and, if necessary, real-time flight data to the ground for monitoring. The parameter server uses the built-in ROS infrastructure *rosparam* and loads the parameters from a YAML-file, which then can be read and overwritten by any node. Every time Offboard Control becomes inactive, that is the pilot takes manual control, all parameters are saved to the file again. When the node is closed, it dumps and clears the parameters from the server. All data streams between nodes are recorded in a ROS BAG-file, with the recording started and stopped by Offboard Control on arming and disarming of the Pixhawk, respectively.

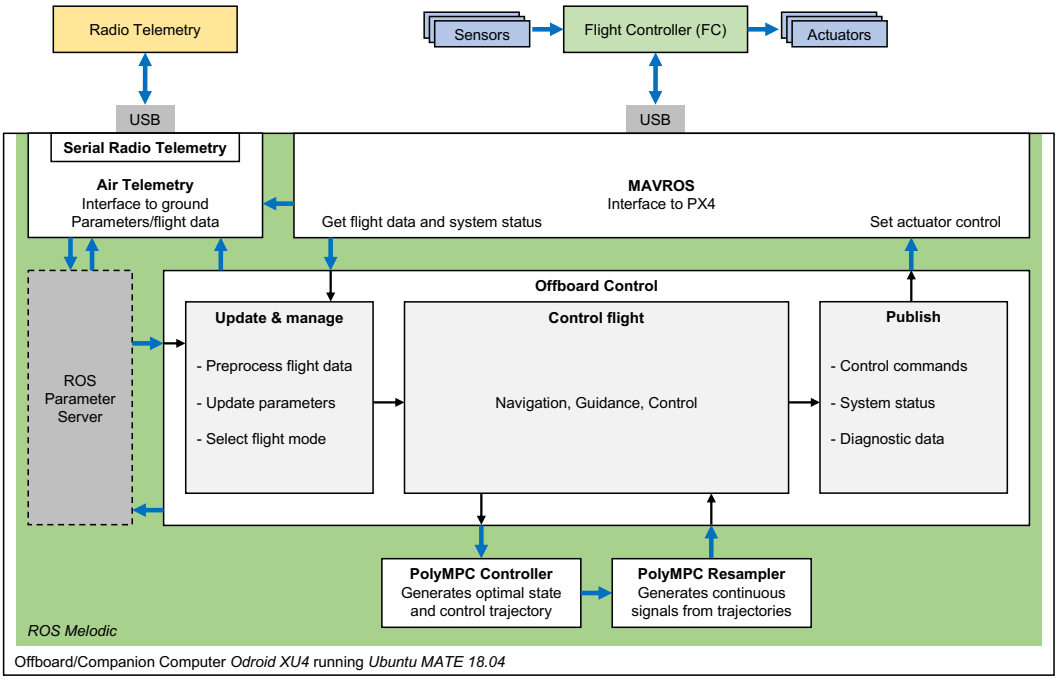


Figure 4.7 – Software architecture on the Offboard Computer.

Offboard Control

The Offboard Control node consists of two processes that run at different rates. Navigation data processing, parameter and flight state machine management executes at a high rate. This includes the decision whether Offboard Control is active or not and which flight control mode is executed. The slow loop contains the high-level controllers, which is all controllers except

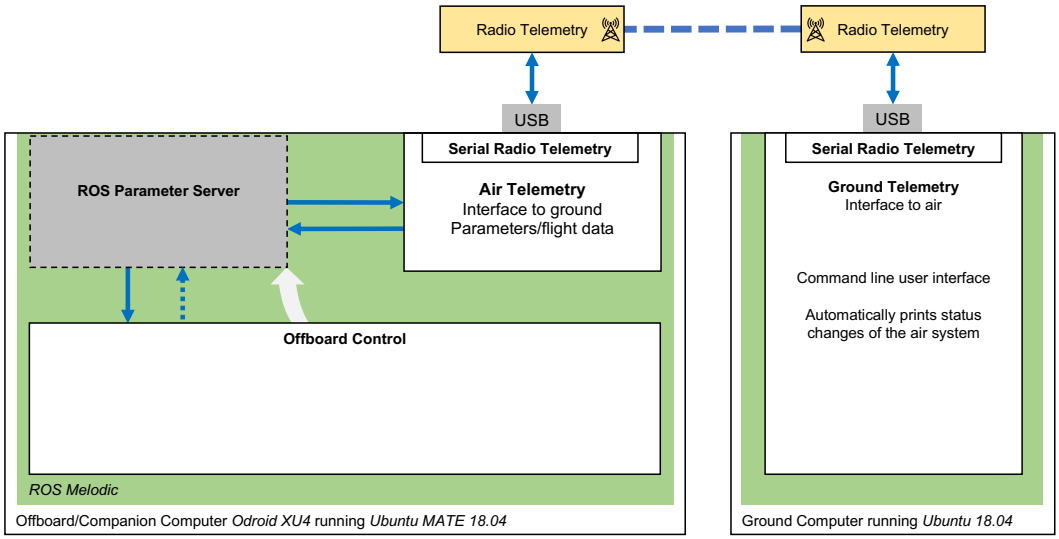


Figure 4.8 – Telemetry air-ground connection.

the velocity, attitude, and rate controllers. The fast loop then processes the references from the high-level controllers at the fastest possible frequency, based on the newest measurements. Besides the use of flight control modes, Offboard Control contains an event-based framework to manage flight missions that enables changing the flight control mode or any parameter within Offboard Control.

Telemetry and Parameter Management

Both Ground and Air Telemetry are built on the *Serial Radio Telemetry* package, which is a custom serial protocol. It enables creation of custom messages to be transmitted over the serial telemetry connection. The main task of the Ground Telemetry node is to change parameters during flight. Our parameter management system is sketched in Figure 4.8. At system launch, Offboard Control initialises the parameter server by loading all parameters from a file. Ground Telemetry provides a command line user interface (CLI) that enables real-time interaction with the parameter server on-board. Whenever a parameter is selected, Ground Telemetry first sends a read request for the current value to the air node. Air Telemetry processes this request, reads out the corresponding value from the ROS parameter server, and sends it to the ground, where it is displayed to the user. The write request works in a similar fashion. If written correctly, the user gets a confirmation. Air telemetry also notifies Offboard Control to update all parameters. For robustness, Ground Telemetry attempts sending parameter requests until it receives a confirmation.

Additionally, every time Offboard Control changes the flight control mode or the PX4 is armed or disarmed, the Air Telemetry node sends a corresponding status message to the ground node which also contains the battery voltage, capacity and flight time.

In the remaining time, when none of the above mentioned tasks is due, Air Telemetry can send flight data down to the GS for monitoring and algorithm tuning. By default, this feature is deactivated to save battery.

Ground Station

The GS software comprises a motor control unit and drivers for a force measurement unit and encoders. The motor control unit includes torque, velocity, and position controllers. The ground station is controlled by several ROS nodes that read out measurements and set references to its controllers. Additional nodes manage custom ground station parameters and an interface to set the motor mode and control reference with the laptop keyboard.

The software package also includes a monitoring tool. It displays the control mode and motor torque or velocity references, and indicates the reeling direction of the tether. Moreover, it monitors all relevant measurements of the ground station.

Flight Log Analysis

All ROS communication in our system is recorded in BAG-files. They allow to replay the system later, with an option to export to CSV-files. The CSV-files are then parsed by topic in Matlab. The implemented script is quite flexible. The data can be selected by the offboard control mode or mission mode, or any other condition derived from the data. Relevant data sequences are then processed further. Above all, the flight log analysis provides plots that visualize the recorded data and make it intuitive.

Simulator

Our AWE software pack also includes a flight simulator and visualization tool to display the flight behavior. For control algorithm debugging it additionally outputs aerodynamic angles and specific non-gravitational forces on the aircraft.

For visualization, the ROS package *RViz* is used, which displays 3D data in different coordinate frames. Figure 4.9 shows the visualization in RViz: it displays all relevant coordinate frames, the wind vector, aircraft body, geodetic and aerodynamic velocities and aircraft position trace. For tethered flight, there is also the tether force vector and a numerical indication of its load magnitude. Furthermore, the specific non-gravitational force vector can be displayed.

4.5 Summary

In this chapter, we presented a dynamical model of a fixed-wing single-line AWE kite. Our approach to the modelling of aerodynamic forces and smooth approximation of tether tension

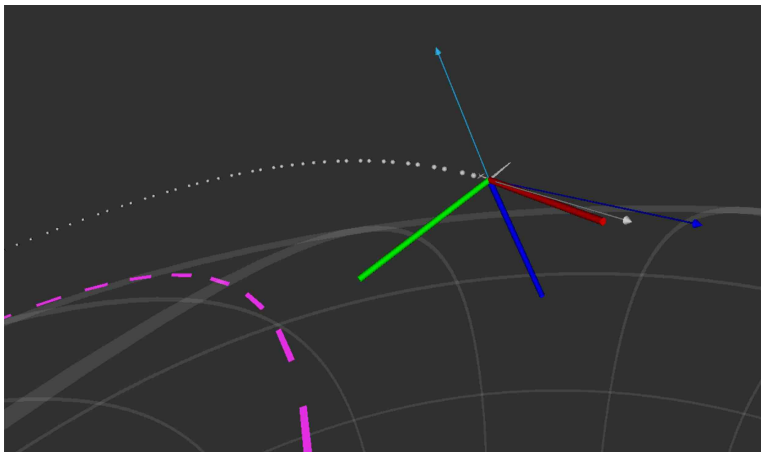


Figure 4.9 – Flight simulation and visualization in RViz. Display of the body reference frame in red-green-blue, the flight path velocity $\mathbf{v}_K$ (white), apparent velocity $\mathbf{v}_A$ (dark blue), and resulting specific non-gravitational force (light blue).

force allow for high fidelity simulation while being computationally feasible for model-based control design. Additionally, a unique small-scale platform has been designed and built to enable the experimental study of AWE kite dynamics and control systems. The flexible software architecture facilitates quick control design iterations and allows switching and tuning of flight controllers in the air without the need for landing.

Chapter 5

Identification of an AWE Kite

5.1 Identification

Accurate modelling of the system is instrumental for designing and validating control and estimation algorithms. A key challenge for any fixed-wing flying vehicle is characterisation of its aerodynamic properties, i.e. interaction with the atmosphere. Dimensionless aerodynamic coefficients, stability and control derivatives introduced in Section 4.1 depend on the profiles of aerodynamic surfaces and geometry of the aircraft. Initial estimates of these coefficients can obtained using an empirical framework DATCOM [89] that performs calculations based on the geometry of a flying vehicle and a vast data base of tunnel experiments. *XFLR5* [90] is a more elaborate software tool that utilises the vortex lattice method (VLM) to compute aerodynamic polars for a specified wing profile for small Reynolds numbers. The software then employs a 3D panel method to estimate the pressure distribution on the aerodynamic surfaces of the aircraft. To estimate stability and control derivatives, XFLR5 computes the trimmed flight conditions and introduces small perturbations to angular rates and control surfaces. Some limitations of VLM include inability to estimate the viscous portion of the drag and small range of aerodynamic angles for which the method can produce reliable results (small angle approximation). Additionally, XFLR5 neglects the influence of the fuselage. Other powerful CFD methods like finite element method (FEM) are able to produce more accurate and reliable estimates but require special qualification and experience. Wind tunnel tests is another method known to produce very accurate aerodynamic characteristics of an aircraft but require expensive infrastructure that is not always available.

In this thesis, initial estimates of the aerodynamic coefficients were obtained with *XFLR5* simulations and refined through an extensive experimental flight campaign. For model identification, similar to Licitra et al [91], we employ a model-based parameter estimation

Table 5.1 – Mass and moments of inertia. Note that I_{xz} is an estimate from XFLR.

m	1.3474 kg	I_{xx}	0.0832 kgm^2
I_{xz}	−0.00215 kgm^2	I_{yy}	0.0667 kgm^2
		I_{zz}	0.1173 kgm^2

Table 5.2 – Geometric values obtained with *XFLR5*.

b	1.8 m
c	0.185 m
Λ	10.016
S	0.323 m^2

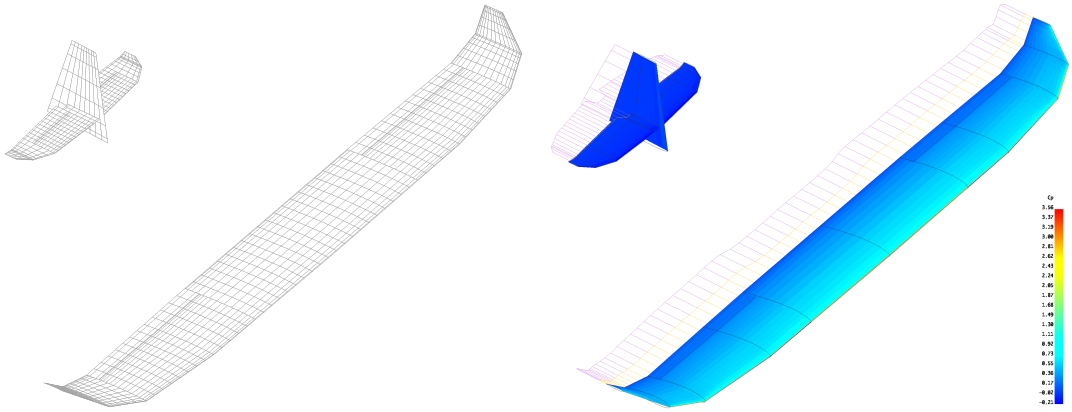
methodology (MBPE) where a full set model equation from Section 4.1 is used to formulate a constrained nonlinear least-squares dynamic optimisation problem. In our approach, however, another method is used for the reference input design. Instead of one optimised control sequence, we propose to use a standard in aerospace field, the so called 3-2-1-1 input sequence for longitudinal and lateral motions with random parameter perturbations in each experiment. Additionally, for each flight experiment, we estimate short-term constant wind disturbances since exact wind measurements are not available on the prototype. Finally, in order to improve robustness and generalisation properties of the model, the multi-experiment averaging strategy from [83] is used where one set of parameters is optimised for several experiments simultaneously. The resulting large scale dynamic optimisation problem that is discretised and solved using *PolyMPC* and the nonlinear solver *Ipopt* [92].

A-Priori Parameters

The mathematical model of a rigid-wing aircraft is over-parametrised, i.e. change in forces or moments can be attributed to or influenced by several parameters, therefore, it is crucial to reduce the search space of the parameter estimation problem. Mass and inertia are determined by measurements and swing experiments as described in [93], see Table 5.1. A first guess for the aerodynamic parameters is obtained with *XFLR5*. After modelling the wing-tail geometry (see Figure 5.1), the software outputs derived geometric quantities, static aerodynamic coefficients, stability and control derivatives. As airfoil profile data is not available from the manufacturer, for simulation we use a NACA 2412 profile for the main wing due to its geometric similarity. *XFLR5* analysis results are shown in Tables 5.2-5.4, and the aerodynamic coefficients are summarised in Table 5.3 and 5.4.

Table 5.3 – Aerodynamic coefficients of the longitudinal dynamics obtained with *XFLR5*.

e	0.97	C_{L0}	0.630	C_{m0}	−0.020	$C_{L\delta e}$	0.275
C_{D0}	0.009	$C_{L\alpha}$	5.347	$C_{m\alpha}$	−0.910	$C_{m\delta e}$	−0.894



(a) (b)

Figure 5.1 – CFD modeling of the wing-tail combination in *XFLR5*. (a) Panel grid used for CFD analysis. (b) The colored surfaces represent the pressure distribution, the yellow and purple lines visualize the viscous and induced drag, respectively.

Table 5.4 – Aerodynamic coefficients of the lateral dynamics obtained with *XFLR*.

$C_{Y\beta}$	−0.307	C_{Yp}	−0.124	C_{Yr}	0.234	$C_{Y\delta r}$	0.203	$C_{l\delta a}$	−0.315
$C_{l\beta}$	−0.097	C_{lp}	−0.540	C_{lr}	0.139	$C_{l\delta r}$	0.010	$C_{n\delta a}$	−0.008
$C_{n\beta}$	0.084	C_{np}	−0.075	C_{nr}	−0.066	$C_{n\delta r}$	−0.074		

The range of control surface deflections and static thrust are determined prior to flight experiments. The maximum deflection angles of the elevator, rudder, and ailerons are measured from photos. For the differentially deflected ailerons, we average the two asymmetric deflections. The static thrust $F_{Thr,0,max}$ at full throttle and battery voltage is measured with a force measurement unit. Thrust decrease related to airspeed is approximated using a free propeller design tool *PropCalc* [94] for a standard propeller of corresponding size and pitch, as seen in Figure 5.2. We fit the speed-thrust curve with a second-order polynomial

$$f(V_A) = p_2' V_A^2 + p_1' V_A + p_0', \tag{5.1}$$

normalized it such that it maps the airspeed to [0, 1], and multiply it by the measured maximum static thrust:

$$F_{Thr} = F_{Thr,0} \cdot \left(p_2 V_A^2 + p_1 V_A + 1\right) \tag{5.2}$$

Results are given in Table 5.5.

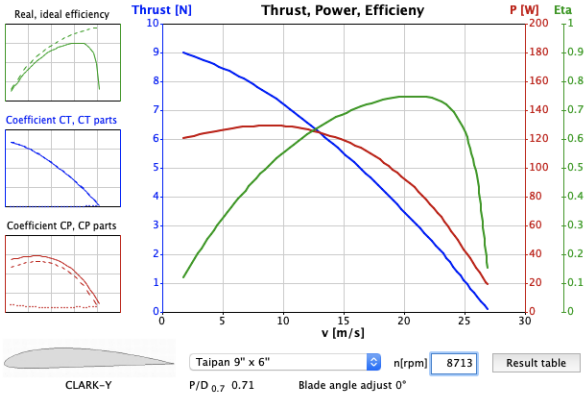


Figure 5.2 – Thrust curve (blue) in PropCalc.

Table 5.5 – Maximum control surface deflections and thrust parameters.

$\delta_{e,max}$	20.0°	0.35 rad	$F_{Thr,0,max}$	6.37 N
$\delta_{r,max}$	26.0°	0.45 rad	p_1	−0.0147
$\delta_{a,max}$	20.0°	0.35 rad	p_2	−0.001

5.2 Identification Experiments

The a-priori aerodynamic coefficients are refined through experimental flight data. After performing a stability analysis of the aircraft dynamic modes in *XFLR5*, we choose to apply 3-2-1-1 square-waves to separately excite the longitudinal and lateral dynamics of the system. This class of inputs has traditionally been used in airplane identification [95]. The unit pulse length is set such that the power density is concentrated around the particular natural frequency of the fast longitudinal mode or the lateral dutch roll mode, as proposed in [95, 91]. Large input amplitudes are favourable for a high signal-to-noise ratio and information-rich trajectories. It is important to note, however, that input design based on the eigenfrequency analysis is a heuristic method and must be adapted and validated during flight experiments as it can lead to insufficient excitations or violation of the flight envelop.

As common in control-oriented applications, the aerodynamic coefficients are assumed to be constant which is only true in a limited range of aerodynamic angles and airspeeds. It is, therefore, undesirable to steer the aircraft to large angles of attack and side slip angles.

To use the longitudinal and lateral motion separation assumption, for every experiment the aircraft is brought to quasi-steady level flight condition: angular rates (p, q, r), velocities (v, w) and roll angle are close to zero and the aircraft flies with a constant airspeed. Some randomness is, however, desired increasing the information content [95]. During longitudinal experiments, when the elevator excites the longitudinal dynamics, the lateral motion (roll and yaw rates) are

stabilised by ancillary controllers. All identification manoeuvres are flown without propulsion to avoid vibration and parasitic forces and moments and we perform as many redundant experiments as possible to minimize the effect of disturbances, e.g. turbulence.

As state and control signals are updated at different rates, they are interpolated with splines and resampled at desired time instants during optimisation. Before optimisation, all identification experiments of the same kind are standardised to start North-headed from the same position. This allows for convenient comparison and avoids resampling issues with periodic state components, such as quaternion. Optionally, data is smoothed if noisy.

Figures 5.3 and 5.4 show data from three exemplary longitudinal and lateral identification experiments, respectively. To simulate transient response of actuators the 3-2-1-1 inputs are filtered by a second-order linear filter.

5.3 Nonlinear Multi-experiment Identification via Dynamic Optimisation

Lagrange polynomial interpolation is utilised to obtain an approximation of the continuous measurements $\mathbf{y}_{meas}(\cdot)$ and control $\mathbf{u}(\cdot)$ trajectories. We then formulate the following continuous-time optimal parameter estimation problem for K experiments of length t_f:

$$\begin{aligned}
\min_{\mathbf{x}^{(k)}(\cdot),\boldsymbol{\theta}} \; & J(\mathbf{x}^{(k)}(\cdot),\boldsymbol{\theta}) = \sum_{k=0}^{K-1} \left(\int_0^{t_f} \left\| \mathbf{y}_{meas}^{(k)}(\tau) - \mathbf{h}(\mathbf{x}^{(k)}(\tau)) \right\|_{\mathbf{W}}^2 d\tau \; + \left\| \boldsymbol{\theta}^{(k)} \right\|_{\mathbf{Q}_{\Delta wind}}^2 \right) \\
\text{subject to} \quad & \mathbf{x}^{(k)}(0) = \mathbf{x}_{meas}^{(k)}(0) \\
& \dot{\mathbf{x}}^{(k)}(\cdot) = \mathbf{f}(\mathbf{x}^{(k)}(\cdot), \mathbf{u}^{(k)}(\cdot), \tilde{\boldsymbol{\theta}}^{(k)}) \\
& \underline{\mathbf{x}} \leq \mathbf{x}^{(k)}(\cdot) \leq \overline{\mathbf{x}} \\
& \underline{\tilde{\boldsymbol{\theta}}} \leq \tilde{\boldsymbol{\theta}}^{(k)} \leq \overline{\tilde{\boldsymbol{\theta}}},
\end{aligned} \tag{5.3}$$

Where $\mathbf{h}(\mathbf{x}(\cdot))$ is the output mapping and quaternions are transformed to Euler angles, roll φ and yaw ψ, to obtain lateral attitude errors. The matrix $\mathbf{W}$ weights the outputs differently according to the type of identification experiment: higher weights are assigned for u, w, q, z, q_y in longitudinal experiments, and for v, p, r, φ, ψ in lateral. In addition, all weights are normalised by typical maximum magnitudes of the respective output variable.

The full multi-experiment parameter vector consists of the aerodynamic parameters $\boldsymbol{\theta}_{aero}$ and an experiment-specific part $\boldsymbol{\theta}_{\Delta wind}^{(k)}$ that parametrises unmeasured short-term wind

measurement error:

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\theta}_{aero} \\ \boldsymbol{\theta}^{(0)}_{\Delta wind} \\ \vdots \\ \boldsymbol{\theta}^{(K-1)}_{\Delta wind} \end{pmatrix}, \quad \tilde{\boldsymbol{\theta}}^{(k)} := \begin{pmatrix} \boldsymbol{\theta}_{aero} \\ \boldsymbol{\theta}^{(k)}_{\Delta wind} \end{pmatrix}$$

Given that one 3-2-1-1 experiment is of a few seconds in length, we assume the short-term wind deviation can be modeled as a constant offset vector $\Delta\mathbf{v}_W$ from the long-term mean $\overline{\mathbf{v}}_W$, i.e., $\boldsymbol{\theta}^{(k)}_{\Delta wind} := \Delta\mathbf{v}^{(k)}_W$ and therefore, $\mathbf{Q}_{\Delta wind}$ is used to regularise the disturbance estimate.

The aerodynamic parameter vector is initialised with an initial guess and bounded to positive multiples of its initial values, such that sign switching is not allowed. The wind offsets are bounded to $\pm(2,2,1)^T$ m/s. Therefore, the parameter bounds $(\underline{\tilde{\boldsymbol{\theta}}}, \overline{\tilde{\boldsymbol{\theta}}})$ are identical for all experiments.

For numerical simplification and to compare tendencies in the estimates, we divide the available identification experiments into groups of six. For each group, we run a multi-experiment parameter estimation and compare the estimates between groups.

Longitudinal experiments show almost perfect lateral stabilization but lateral experiments usually provoke some pitch dynamics. Therefore, longitudinal parameter estimation is performed first, and the subsequent lateral identification already uses the identified longitudinal parameters.

It can be observed that the estimation problem with the full set of longitudinal and lateral parameters is over-parametrised and does not have a unique optimal solution. For instance, the Oswald efficiency e and the pitch moment coefficient show strong co-linearity, and the optimiser drives one value to an upper or lower bound while compensating with the other value. We, therefore, employ some engineering knowledge as described in the following section to reduce the degrees of freedom in the optimization problem and fix some parameters to their a-priori values.

Drag and lift polars

As the nonlinear parameter identification tends to provide ambiguous results, it is desirable to determine and fix as many parameters as possible beforehand to decrease the degrees of freedom of the optimization problem. One approach is to evaluate characteristic curves, or polars. By plotting the lift and drag curves (see Figure 5.5) using accelerometer measurements and angle of attack estimates [95], conclusions about the data quality can be formed. The noisy nature of the data has two sources. First, the aircraft is not in steady flight and nonzero pitch rates introduce additional lift. Second, the angle of attack used to compute the lift and drag coefficients (C_L, C_D) is an estimate and thus introduces uncertainty. Initial experiments

with angle of attack sensors suggest that measurements would narrow the data point envelope considerably. We find that polynomials fitted with the least squares method characterize the noisy data acceptably and obtain data-based coefficients for C_{D0}, e, C_{L0} and C_{La} that can be fixed in the following identification.

Experimental Results

In the following, we present one complete identification procedure. The coefficients C_{D0}, e, C_{L0} and C_{La} that have been identified as described above are considered part of the a-priori parameter set 'xflr-graph'. We use $\approx 80\,\%$ of 81 available longitudinal identification experiments for identification which yields 10 groups of 6 experiments. For each experiment group, we perform a multi-experiment nonlinear parameter estimation, where the remaining optimizable parameters are C_{m0}, $C_{m\alpha}$, C_{Lq}, C_{mq}, and $C_{m\delta e}$. The lift contribution from the elevator $C_{L\delta e}$ is neglected. Figure 5.6 shows the estimates for the 10 experiment groups. We observe a considerable variance in the estimation as well as different estimation branches which indicates that there are several local minima. The estimated trajectory is portrayed in Figure 5.7 and demonstrates the satisfactory quality of the parameter estimates. We choose to determine the final estimates as the median of the identification groups. The updated parameter set is called "xflr-graph-P" ("Pitch" experiment).

In the next step, we identify the lateral dynamics using the "xflr-graph-P" parameter set. Having in general small contribution, the side force coefficients $C_{Y_{(\cdot)}}$ have been set to their a-priori values. The resulting estimates are depicted in Figure 5.8, and a low-error validation experiment is shown in Figure 5.9. Note that with a longer experiment duration, compared to the longitudinal experiments, fitting gets more difficult because the contribution of disturbances becomes stronger. While the lateral linear and angular velocities generally fit well, the attitude angles roll and yaw tend to have increasing errors along the experiment time due to integration of small errors in the velocities. In the longitudinal dynamics, we generally see errors. The experiments reveal how continuous, small disturbances from the atmosphere add up and bring uncertainty into the trajectory within seconds if no significant input is given. We call the final parameter set "xflr-graph-P-YR" ("Yaw-Roll" experiment), as listed in Table 5.6.

For validation, we simulate both the a-priori XFLR model and the best identified model 'xflr-graph-P-YR' with manual inputs that have a duration of multiple seconds. Figure 5.10 shows the two trajectories along with the flight experiment data. While the angular velocities usually fit well across different validation experiments, the linear velocities often show more errors. Accordingly, attitude and position integrate the velocity errors, together with unmeasured disturbances. The quantitative assessment of the model predictive quality is performed using

Table 5.6 – Complete parameter set.

(a) Geometric values from *XFLR5*.

b	1.8 m
c	0.185 m
Λ	10.016
S	0.323 m^2

(b) Aerodynamic coefficients of the longitudinal dynamics after complete identification.

e	0.47	C_{L0}	0.78	C_{m0}	−0.060	$C_{L\delta e}$	0
C_{D0}	0.025	$C_{L\alpha}$	5.3	$C_{m\alpha}$	−0.659	$C_{m\delta e}$	−0.615

(c) Aerodynamic coefficients of the lateral dynamics after complete identification.

$C_{Y\beta}$	−0.307	C_{Yp}	−0.130	C_{Yr}	0.231	$C_{Y\delta r}$	0.203
$C_{l\beta}$	−0.071	C_{lp}	−0.293	C_{lr}	0.176	$C_{l\delta r}$	0
$C_{n\beta}$	0.053	C_{np}	−0.061	C_{nr}	−0.066	$C_{n\delta r}$	−0.057

$C_{l\delta a}$	−0.121
$C_{n\delta a}$	0

the Theil Inequality Coefficient (TIC) which is defined by the following formula:

$$TIC = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}(\tau_i) - \mathbf{h}(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \theta))^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbf{y}^2(\tau_i)} + \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbf{h}(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \theta))^2}} \tag{5.4}$$

where N is a number of test points sampled from validation trajectories. TIC provides a normalised and dimensionless metric of predictive quality of a mathematical model where $TIC = 0$ corresponds to a perfect model fit and $TIC = 1$ indicates that the measured data cannot be explained by the model. Values of $TIC \leq 0.25$ are considered good for rigid-wing aircraft models [91]. Figure 5.11 compares TIC values for the a-priori model based on the CFD simulations and the estimated from flight experiments. It can be observed that the quality of the model was significantly improved after the identification, especially for the angular velocities (p, q, r), the vertical component of the linear velocity (w) as well as for the pitch and roll angle (φ, ψ).

5.4 Summary

This chapter presents a novel model-based parameter estimation methodology for the identification of fixed-wing aircraft. Initial estimates of aerodynamic coefficients are obtained CFD simulations and refined through an extensive experimental flight campaign. In our approach, MBPE is combined with random perturbation of the 3-2-1-1 control sequences across multiple

experiments, which are optimised simultaneously using the *polyMPC* toolbox. Without access to high precision wind speed measurements on-board, the long-term wind disturbances are additionally estimated for each flight experiment. The proposed methodology proved its efficiency on validation flights in different wind conditions and allowed significant improvement of the predictive quality of the aircraft aerodynamic model.

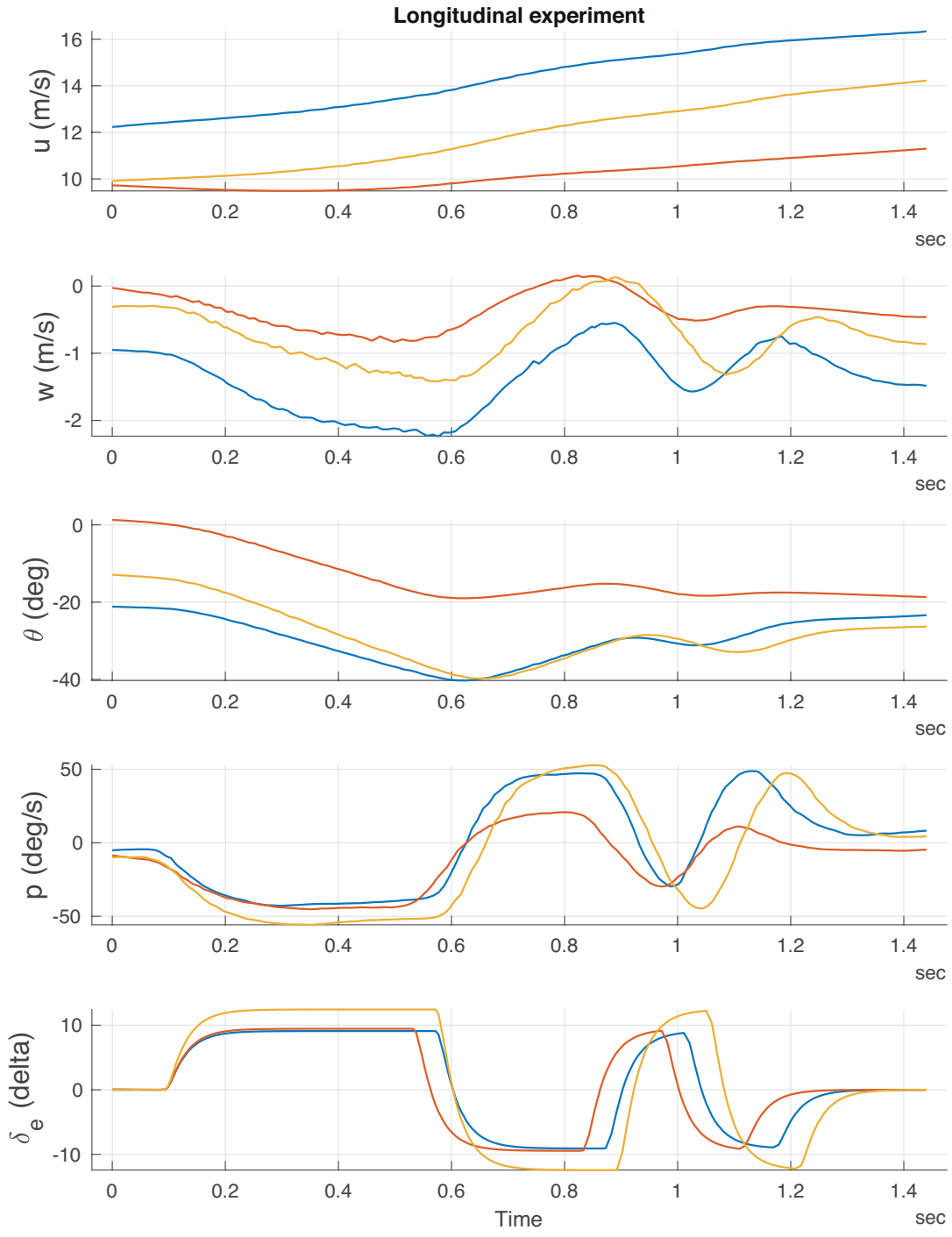


Figure 5.3 – Three example longitudinal identification experiments. Amplitude and pulse duration are randomised to increase the information from multiple experiments.

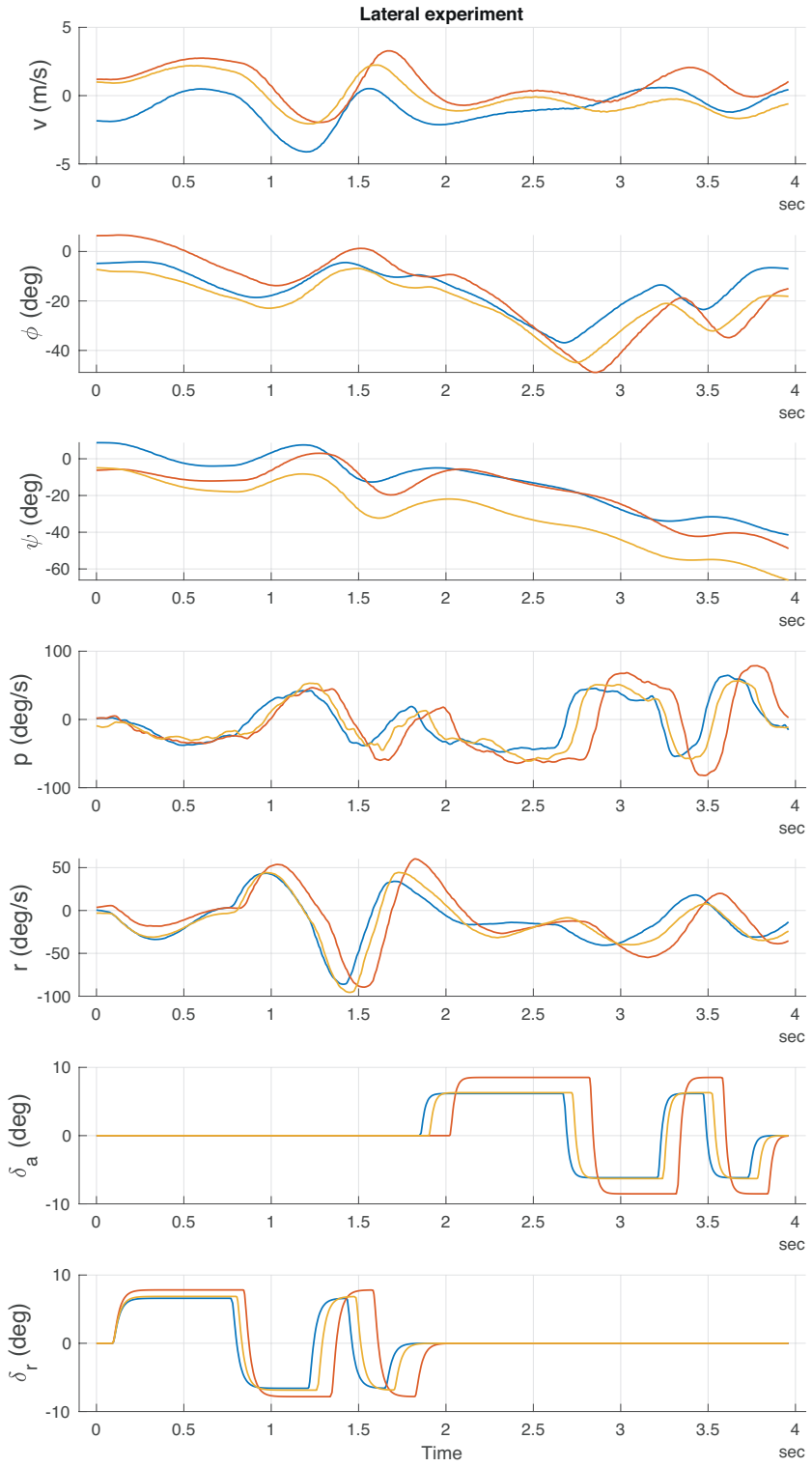


Figure 5.4 – Three example lateral identification experiments. Amplitude and pulse duration are randomised to increase the information from multiple experiments.

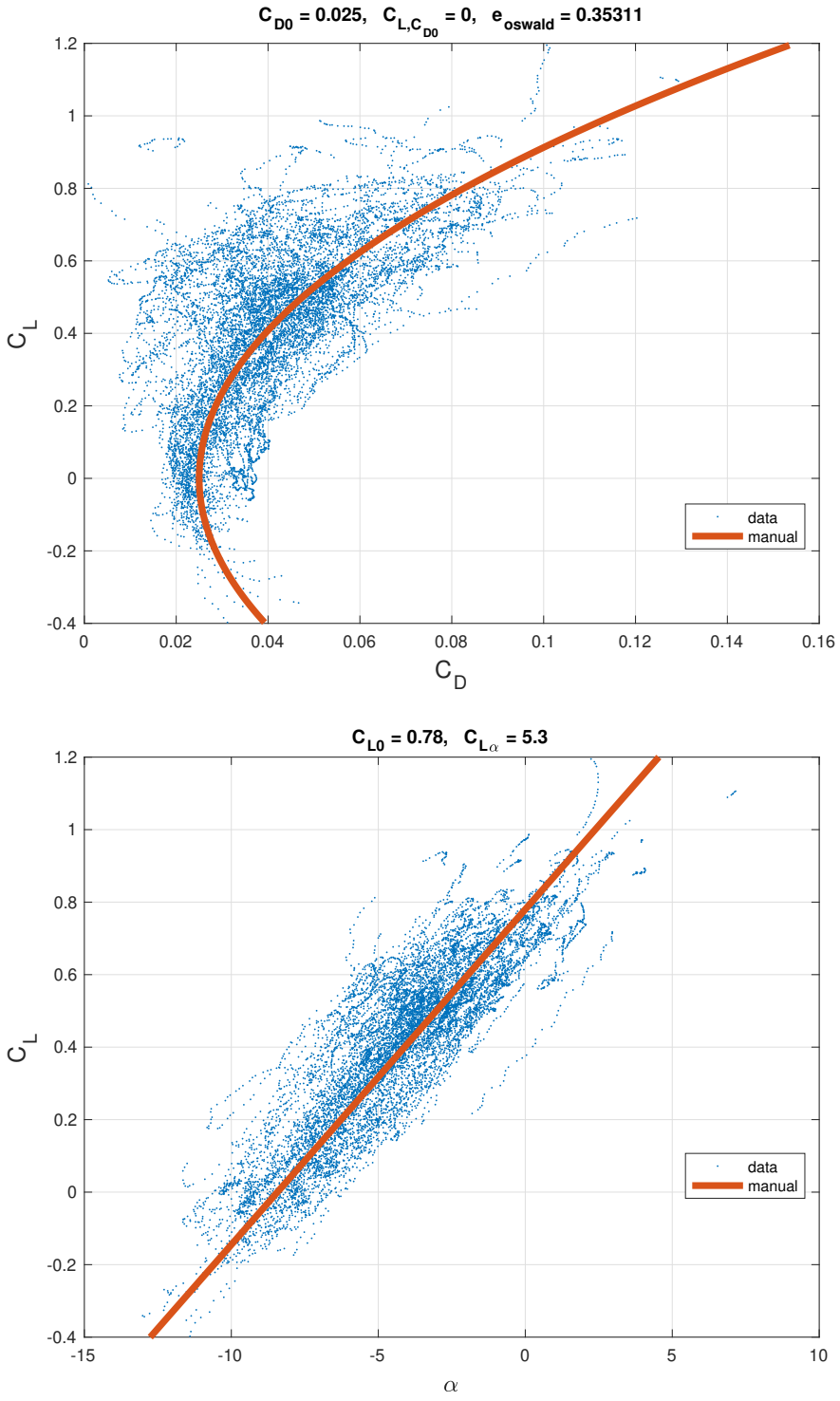


Figure 5.5 – Approximated lift and drag polars. We fit a second-order polynomial $C_D(C_L) = C_{D0} + C_L^2/(\pi e \Lambda)$ for the drag polar (left) and a first-order polynomial $C_L(\alpha) = C_{L0} + C_{L\alpha}\alpha$ for the lift polar (right).

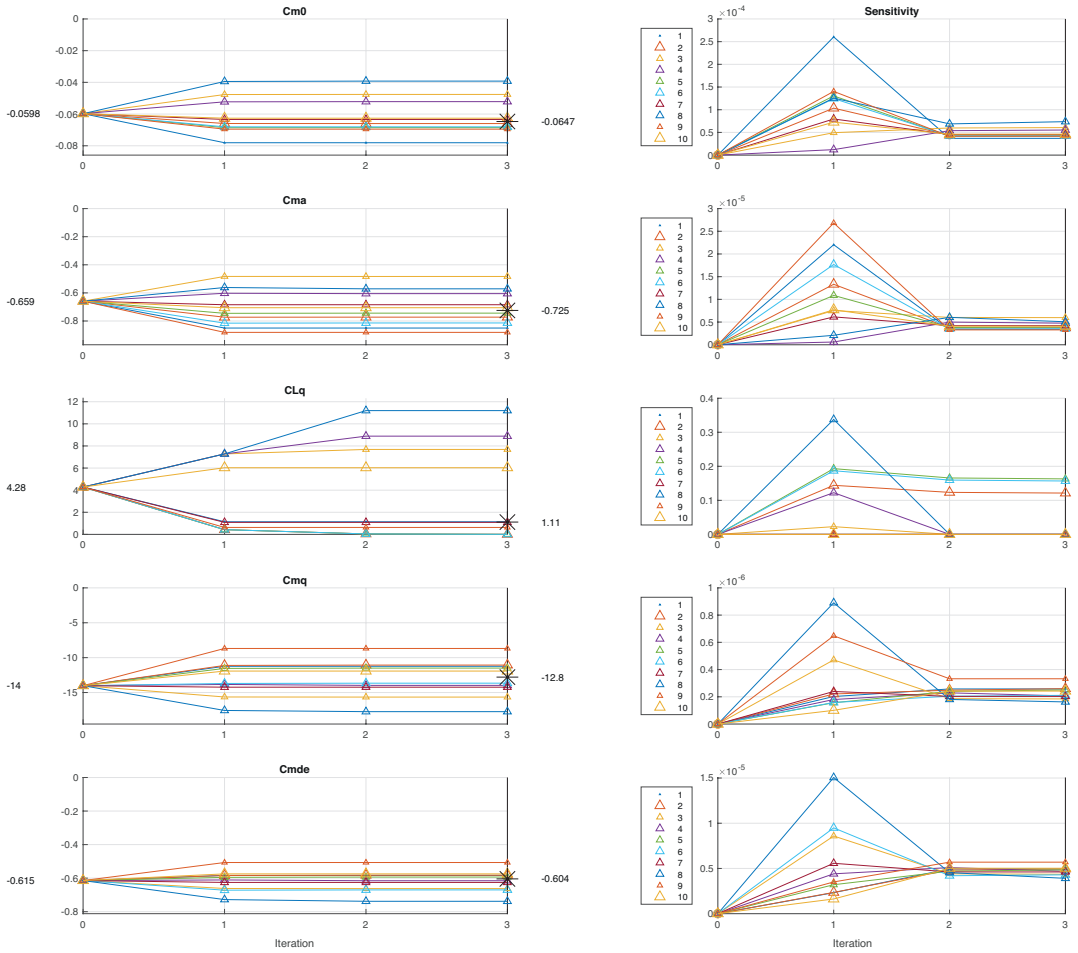


Figure 5.6 – Results of longitudinal nonlinear parameter estimation over three consecutive iterations. Each line represents an identification group of 6 experiments that have been processed together. Larger symbol size represents better fit in terms of the cost. The overall estimates are determined by the median.

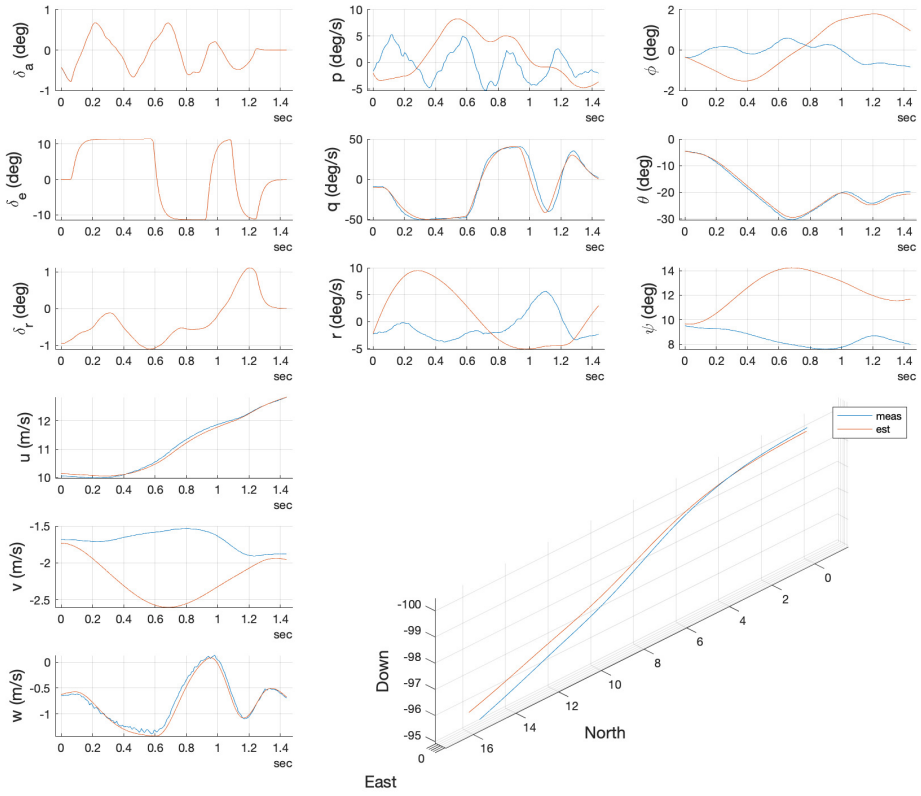
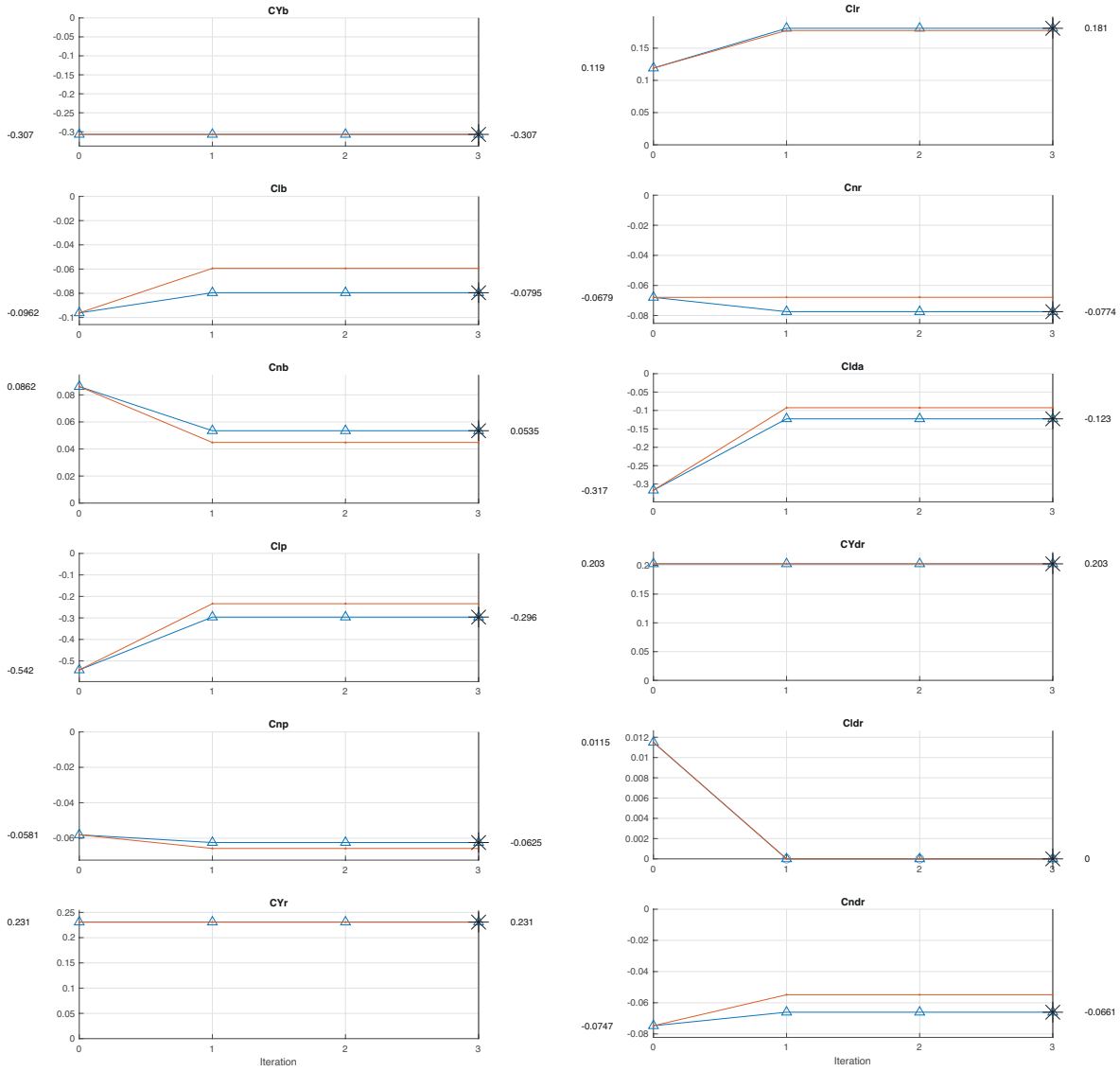


Figure 5.7 – State trajectory fit from longitudinal nonlinear parameter estimation. Note that lateral errors (v, p, q, φ, ψ, y(=East)) are only lightly weighted in longitudinal identification, and their magnitude is small compared to the longitudinal states (u, w, q, θ, x(=North), z(=Down)). The elevator input δ_e induces the longitudinal excitation signal filtered by simulated actuator dynamics, while the lateral inputs δ_a and δ_r stabilize the lateral dynamics.



(a) (b)

Figure 5.8 – Results of lateral nonlinear parameter estimation over 3 (warm-started) iterations. Each line represents an identification group of 10 experiments that have been processed together. Larger symbol size represents better fit in terms of smaller cost. For two experiment groups, the overall estimates are determined by the better fit.

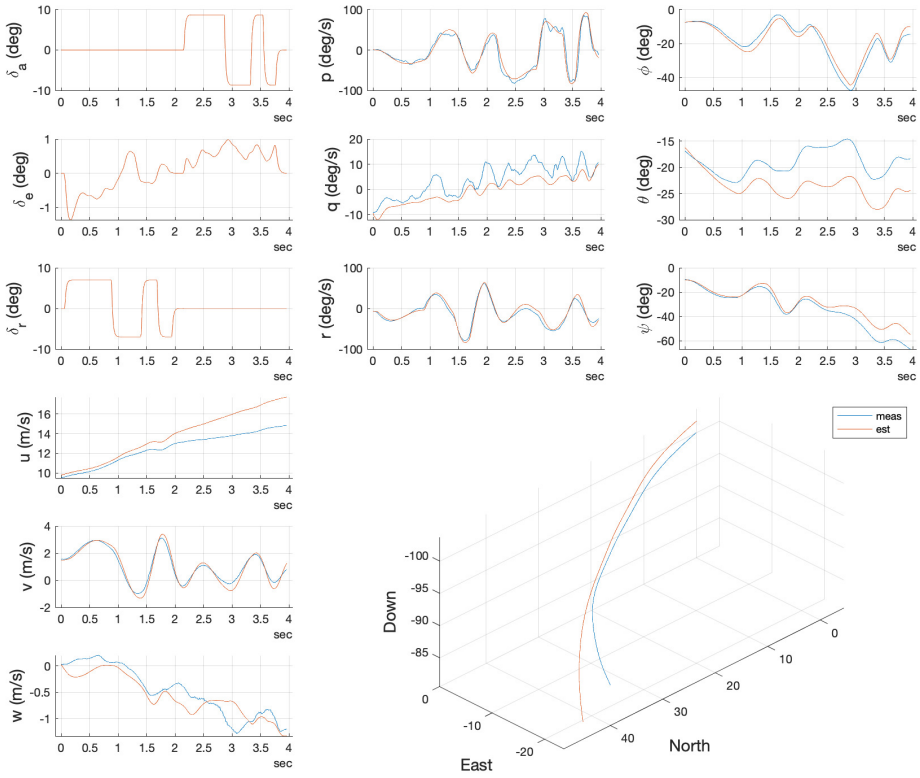


Figure 5.9 – State trajectory fit from lateral nonlinear parameter estimation (best example). Lateral states are about 5 times higher weighted than longitudinal states. The lateral excitation signal is induced by consecutive inputs in rudder and aileron.

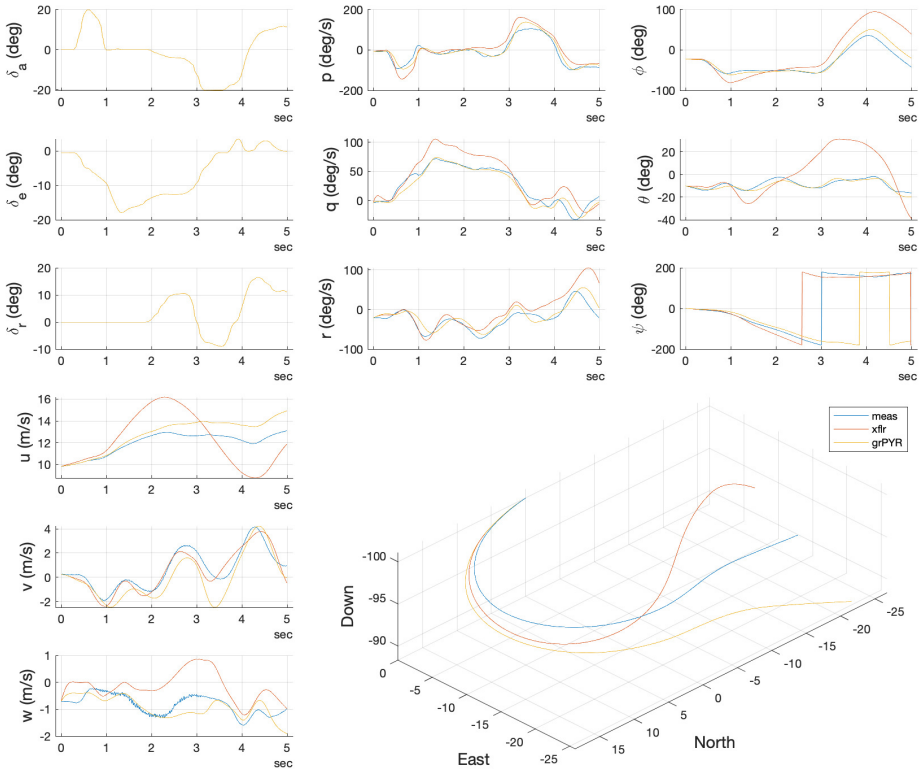


Figure 5.10 – Validation with manual input sequence. All three control surfaces are deflected during the experiment. The best identified model clearly outperforms the a-priori XFLR5 model. Prediction is particularly good for the angular velocities (p, q, r) as well as for the pitch and roll angle (φ, ψ).

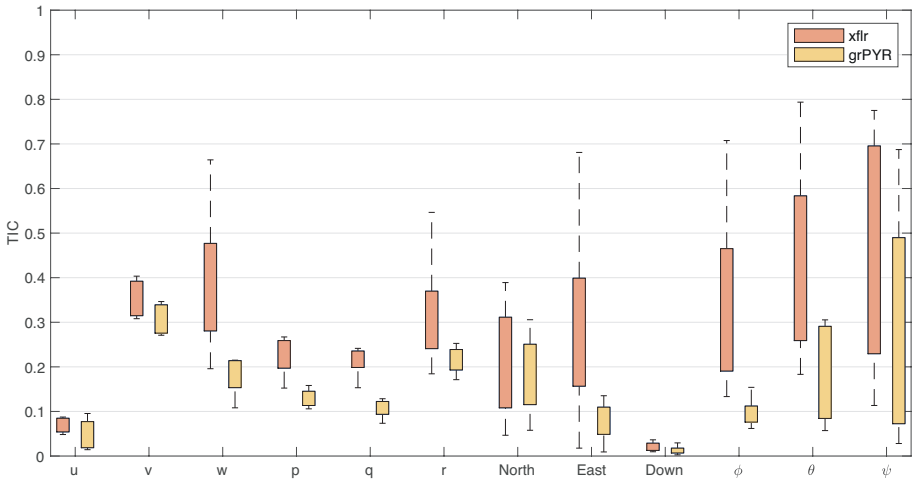


Figure 5.11 – Theil Inequality Coefficient values for 10 validation flight experiments with random pilot input. Orange and yellow boxes denote the 75th percentile of the TIC values for the a-priori (XFLR5) and identified models correspondingly. The dashed vertical lines show extreme low and high TIC values for both models. The best identified model clearly outperforms the a-priori XFLR5 model. Prediction quality is significantly improved for the angular velocities (p, q, r), the vertical component of the linear velocity (w) as well as for the pitch and roll angle (φ, ψ).

Chapter 6

Predictive Path Following Control

In this section, we consider the problem of optimising the flight trajectory given a geometric representation of a desired path. The goal of the optimisation algorithm is to find a trajectory that steers the kite onto this path while respecting dynamic, actuation and flight envelope constraints. This approach is more flexible than in [82] since it does not require a precomputed optimal trajectory but only a geometric curve and possibly a speed profile. Compared to [76], the projection onto the path is done automatically by the optimisation algorithm. Thanks to the look-ahead capabilities, the controller can detect high curvature segments earlier and adjust the trajectory accordingly.

For the predictive controller design, we adopt the optimisation-based path following methodology from [96, 97]. Consider the system output $\mathrm{y} \in \mathbb{R}^{\mathrm{n_y}}$, $y = h(x)$, and a reference path to follow $p : \mathbb{R} \to \mathbb{R}^{\mathrm{n_y}}$ parametrised by a path parameter $\theta \in \mathbb{R}$. The goal then is to find the control signal $u(\cdot)$ that steers the system to this path. That is, to some sequence $\theta(t),\ t \in [t_0, t_f]$ that corresponds to a sequence of points on the path $p(\theta(t))$, we seek for a $u(t)$ to minimize the distance between the path and the system:

$$u^\star(t) = \arg\min_{u(t)} \int_{t_0}^{t_f} \| p(\theta(t)) - h(x(t)) \| \, dt \tag{6.1}$$

A standard approach for finding such an optimal sequence $\theta(t)$ is to assign dynamics to the parameter with a pseudo-input $v(t)$:

$$\dot{\theta}(t) = f_\theta(\theta(t), v(t)) \tag{6.2}$$

In the particular implementation here the choice of $f_\theta(\theta(t), v(t))$ is a second-order linear system: [96]

$$\dot{\mathrm{z}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathrm{z} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \nu = A\mathrm{z} + B\mathrm{v} \tag{6.3}$$

where z consists of the path parameter and its derivative

$$\mathrm{z} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{6.4}$$

The path following problem then becomes:

$$\begin{aligned} \min_{\mathrm{u}(\cdot),\mathrm{v}(\cdot),\mathrm{x}(\cdot),\mathrm{z}(\cdot)} \; & \Phi\left(\mathrm{x}(t_\mathrm{f}), \mathrm{z}(t_\mathrm{f})\right) + \int_{t_0}^{t_\mathrm{f}} L\left(\mathrm{x}(\tau), \mathrm{z}(\tau), \mathrm{u}(\tau)\right) d\tau. \\ \text{s.t. } \forall t \in \left[t_0, t_f\right] : \; & \dot{\mathrm{x}}(t) = \mathrm{f}(\mathrm{x}(t), \mathrm{u}(t)) \\ & \dot{\mathrm{z}}(t) = A\mathrm{z}(t) + B\mathrm{v}(t) \\ & g_l \leq g(\mathrm{x}(t), \mathrm{z}(t), \mathrm{u}(t)) \leq g_u \\ & \mathrm{u}(t) \in \mathscr{U}, \; \mathrm{v}(t) \in \mathscr{V} \\ & \mathrm{x}(t_0) = \mathrm{x}_0 \end{aligned} \tag{6.5}$$

Where x, u, f are the state, control vectors and kite dynamic as introduced in Section 4.1. The associated Lagrange term penalises the deviation from the path, input signal, reference velocity deviations and large aerodynamic angles.

$$L[\mathrm{x}(t), \mathrm{z}(t), \mathrm{u}(t)] = L_\mathrm{path} + L_\mathrm{input} + L_\mathrm{aero} \tag{6.6}$$

Path tracking cost The path tracking portion of the cost is designed to address objective (6.1):

$$L_{path}(\mathrm{x}, \mathrm{z}, \mathrm{u}) = \|\mathrm{h}(\mathrm{x}(t)) - \mathrm{p}(\theta(t))\|_\mathrm{Q}^2 + \|\dot{\theta}(t) - \dot{\theta}_\mathrm{ref}(\theta(t))\|_W^2 \tag{6.7}$$

Aerodynamic cost This term promotes piloting within the desired aerodynamic flight envelope, i.e., keeping the sideslip angle β small

$$L_{aero}(\mathrm{x}, \mathrm{u}) = \begin{bmatrix} \beta \end{bmatrix}^T \mathbf{W}_{aero} \begin{bmatrix} \beta \end{bmatrix}$$

Control cost This term is designed to improve the energy efficiency of the controller by minimizing the deflection of the control surfaces from the neutral position and amount of throttle. Above all, the throttle usage determines the flight duration, significantly draining the battery. Additionally, a penalty on the control signal derivative is included to avoid oscillatory solutions.

$$L(\mathbf{u}, \mathbf{p}) = \mathbf{u}^T \mathbf{W}_u \mathbf{u} + \dot{\mathbf{u}}^T \mathbf{W}_{\delta u} \dot{\mathbf{u}}$$

Flight envelope constraints In a moving atmosphere, it is the apparent velocity rather than the geodetic velocity that is crucial for the aerodynamic forces. It is, therefore, desirable to limit the minimum airspeed to a value where the aircraft does not stall. In addition, the angle of attack α and sideslip angle β shall not exceed certain limits for which the airplane is not designed for. The mentioned quantities can be computed using Equations (4.11) and posed as nonlinear constraints:

$$\underline{V_a} \leq V_a \leq \overline{V_a}$$
$$\underline{\alpha} \leq \alpha \leq \overline{\alpha}$$
$$\underline{\beta} \leq \beta \leq \overline{\beta}$$

Attitude Constraints Quaternions are not very intuitive for representation of attitude constraints, thus, we use Euler angles instead:

$$\underline{\varphi} \leq \varphi \leq \overline{\varphi}$$
$$\underline{\theta} \leq \theta \leq \overline{\theta}$$

here φ and θ are the roll and pitch angles, respectively. We further apply constraints on the change of the control signal:

$$\underline{\mathbf{u}_\delta} \leq \dot{\mathbf{u}} \leq \overline{\mathbf{u}_\delta}$$

The bounds for the control surfaces are symmetrical but the throttle bounds may be asymmetric, to take into account that the real motor runs up slower than it can be cut off:

$$\overline{\mathbf{u}_\delta} = \begin{bmatrix} F_{Thr,0,\delta max}, & \delta_{e,\delta max}, & \delta_{r,\delta max}, & \delta_{a,\delta max} \end{bmatrix}^T$$
$$\underline{\mathbf{u}_\delta} = \begin{bmatrix} F_{Thr,0,\delta min}, & -\delta_{e,\delta max}, & -\delta_{r,\delta max}, & -\delta_{a,\delta max} \end{bmatrix}^T$$

6.1 Hierarchical Control Scheme

The continuous optimal control (8.37) is discretised and solved using the *PolyMPC* toolbox. Even though, our implementation is on average 5 times faster than the solution presented in [82] with a similar setup, i.e. ≈ 40 ms average solve time, this would not be sufficient for real-time embedded flight control applications. Typically, computations on our Offboard computer run 5-10 times slower than on a desktop. And since embedded Linux is used on the prototype, additional delay may be introduced due to the unsteady performance of the operating system scheduler which can change the priority of the process. To address computational delays, a hierarchical control scheme was developed.

PolyMPC implements the pseudospectral collocation method, and therefore, allows for efficient sampling of the optimal control and state trajectories at any given time t using Lagrange interpolation:

$$\mathrm{x}(t) = \sum_{k=0}^{N} \mathrm{x}_k \varphi_k(t), \quad \mathrm{u}(t) = \sum_{k=0}^{N} \mathrm{u}_k \varphi_k(t) \tag{6.8}$$

where $\mathrm{x}_k = \mathrm{x}(t_k)$ and $\mathrm{u}_k = \mathrm{u}(t_k)$ are the state and controls trajectories evaluated at the so-called collocation nodes t_k, and φ_k is a Lagrange polynomial of order k. We leverage this representation to resample the solution of (8.37) at a faster rate in a low-level stabilization loop. In this scenario, the NMPC algorithm operates in the guidance mode providing optimal trajectory and feed-forward control signals.

The hierarchical controller is implemented as two separate asynchronous processes, or nodes. In the following, the word "publishing" refers to broadcasting data over the network. The *PolyMPC Control* node repeatedly solves the OCP (8.37) and publishes the optimal trajectory, in the form of expansion coefficients (6.8). At a high constant rate, the *PolyMPC Resampler* node resamples and publishes these reference trajectories.

In order to account for external disturbances and model mismatch, we use a high-rate feedback control loop that stabilises the aircraft around the most recently computed optimal trajectories. The steering input from the *PolyMPC Control* node is used as a feed-forward component:

$$\mathbf{u}_{cmd} = k_{FF} \cdot \mathbf{u}_{MPC} + \Upsilon\left(\widetilde{\mathbf{x}}_{ref} - \widetilde{\mathbf{x}}\right), \tag{6.9}$$

where $\mathbf{u}_{cmd}$ is the control command applied to the aircraft, $\mathbf{u}_{MPC}$ is the feed-forward control command, $\Upsilon(\cdot)$ is a low-level feedback controller (in our case, PID or LQR), and $\widetilde{(\cdot)}$ is the selection operator. In this work, the angular rates are stabilised. As was mentioned earlier, the highly nonlinear and nonconvex nature of the problem (8.37) can pose significant challenges for the nonlinear solver, which might not converge in assigned time or can get stuck in a local minima. To mitigate the risks and increase the reliability of the system a supervising layer is added. Whether a new optimal trajectory is accepted by the watchdog depends on the status of the nonlinear solver and a solution quality heuristic. As the heuristic, the path tracking portion cost is used and the critical value is estimated prior to the flight tests based on simulation studies. Apart from the trajectory quality the watchdog tracks the period of time since the last optimal solution was accepted because the open-loop solution will inevitably diverge from the actual aircraft state after some time. If the solution provided by *PolyMPC Control* does not satisfy the quality criteria or the solution has not been available for a certain amount of time, the watchdog switches to a simpler backup controller similar to the one presented in [77] that

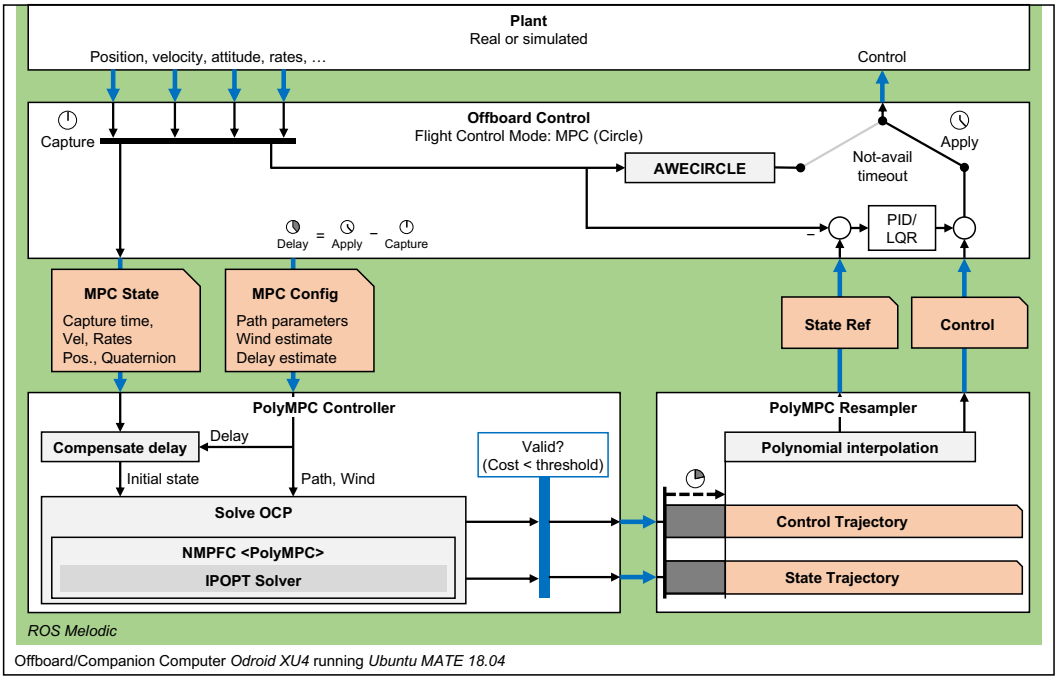


Figure 6.1 – Integration of PolyMPC Control into the Offboard Control architecture with trajectory stabilization extension.

would safely steer the aircraft in the neighbourhood of the desired path. The complete control system structure is depicted in Figure 6.1.

Through the whole control system pipeline, the *Offboard Control* software tracks the timestamps of each event. This allows for exact estimation of delays caused by the NMPC node which can be used for accurate delay compensation as will be explained next.

Delay compensation

Delay compensation is a feature that is particularly relevant when the computation of the control command takes a considerable amount of time, which is usually the case when running optimization algorithms on embedded platforms. Basically, the problem is that when the optimal control command is applied to the plant, its state has already changed considerably in the meanwhile, compared to the state that was used to start the computation.

Figure 6.2 summarises delays that occur in our flight control system as well as the delay compensation mechanism. We use two distinguished time instants:

t_c Capture time. This is the time when the true flight state is first captured by a sensor. The resulting state measurement is then $\mathbf{x}_c = \mathbf{x}(t_c)$.

t_a Apply time. This is the time when a control command that was computed based on the state measurement $\mathbf{x}_c$ at capture time t_c is applied to the plant.

Since *Offboard Control* logs the timestamps of received and published data it can exactly measure the time delay between receiving a state estimate and publishing the control signal. This will be called "measured delay" in Figure 6.2. The additional delay between the apply time and the consequent control surface deflection can be determined experimentally. It is called "external" delay as it accounts for delays in data transmission over local network, serial ports and in actuators.

We can use this information for compensation. After filtering the measured delay using a moving average filter and optionally adding the external delay component, *Offboard Control* publishes the "expected delay". *PolyMPC Control* receives a state $\mathbf{x}_c^{(i)}$ with its capture time $t_c^{(i)}$, along with the expected delay $d_{expected}$, and can therefore compute the apply time of the trajectory which is about to be calculated:

$$t_a^{(i)} = t_c^{(i)} + d_{expected}$$

Based on the flight model presented in Section 4.1, the state is propagated in the future where it is expected to be applied. Simulation (i) therefore runs from $t_c^{(i)}$ to $t_a^{(i)}$ with the initial state $\mathbf{x}_{sim}(t_c^{(i)}) = \mathbf{x}_c^{(i)}$. The question of what control inputs to use for this simulation remains. *PolyMPC Control* keeps a limited number of valid previous trajectories in a buffer, together with their apply times. Simulating along these control trajectories over the expected delay reconstructs what will have happened to the plant until the solution of the present OCP will be applied. If no valid trajectory is available the last input is applied with the zero-order hold assumption.

Figure 6.2 shows three example cases:

1. Iteration (i) is about solving OCP$^{(i)}$. The state is considered constant until $t_a^{(i)}$.
2. Iteration $(i+1)$ is about solving OCP$^{(i+1)}$. The state is considered constant until $t_a^{(i)}$, where traj($t_a^{(i)}$) begins, which was computed in iteration (i).
3. Iteration $(i+2)$ is about solving OCP$^{(i+2)}$. The simulation time is covered by traj($t_a^{(i)}$) from beginning. From $t_a^{(i+1)}$ on, the next newer traj($t_a^{(i+1)}$) is used.

6.2 Experimental Results

Flight Experiments

First MPC tests are performed without tether and with small declination angles of a reference circle. The prediction horizon was set to $5s$ More declined circles often bring the aircraft to its flight performance limits. It considerably loses energy while flying down the circle at high speed and drag. Even at full thrust, the aircraft can potentially stall on the way to the topmost point.

Figure 6.4 shows the flight trajectory of the supervised flight. During the experiments, there was a wind speed of up to $\approx 3\,\mathrm{m/s}$. The computation time to solve the OCP ranged from $\approx 0.3\,\mathrm{s}$ to $1.2\,\mathrm{s}$ which brings necessity to the previously described resampling strategy with angular rates stabilisation was used. The stabilization gains and the model parameter set used for MPC are tuned in flight. The NMPC algorithm follows the circle path with visible oscillations. These oscillations are caused by high computation times on the embedded hardware which means that the aircraft often tracks open-loop trajectories. Currently, the prototype does not have sensors for aerodynamic angles measurements which creates another source of modelling mismatch.

Figure 6.5 shows typical control command signals computed by the MPC controller. They usually contain visible discontinuities between the MPC iterations because the real flight state deviates from the optimal predicted trajectory during solving of the OCP.

6.3 Summary

In this thesis, we present a complete testing platform of an AWE system which allows for affordable validation of novel flight control algorithms. We further present identification methodology for aerodynamic parameters which enables significant improvement of the quality of the aerodynamic models. Finally, an optimisation based path following controller is implemented and tested on an embedded platform.

Simulations with the nominal model show that the NMPC performance deteriorate when adding constant wind that is unknown to the controller. During the outdoor flight test, the PX4 state estimation includes the horizontal wind vector. This estimation is based on the level flight assumption, i.e., the measured airspeed lies in the horizontal plane, which is rarely true in an AWE flight. The future work will be aimed at developing airspeed angles sensors for the prototype which should improve the performance of the controller.

It is expected that the model parameters cannot be identified with high accuracy. In addition, the flight model itself contains considerable simplifications. For instance, the aerodynamic parameters are assumed to be constant with respect to the airspeed and angle-of-attack. At sufficiently high computation rates, however, the sensitivity of the closed-loop performance

to these inaccuracies is reduced. However, for the Odroid computer, the computation times average at $\approx 450\,\text{ms}$ which renders it difficult for reliable real-time control. For the future work, approximate schemes such as real-time iteration will be implemented and tested.

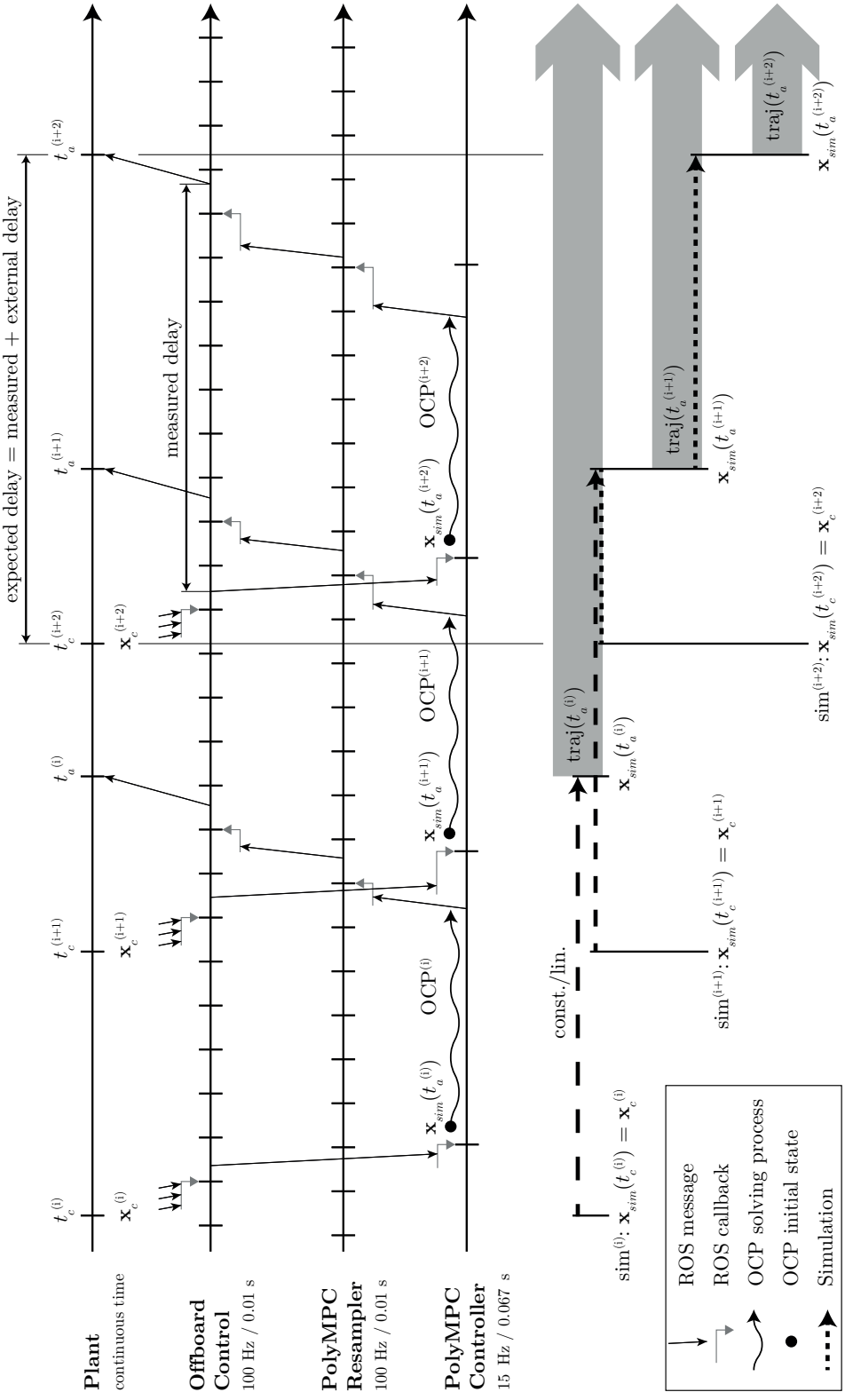


Figure 6.2 – Delay compensation over several trajectories.

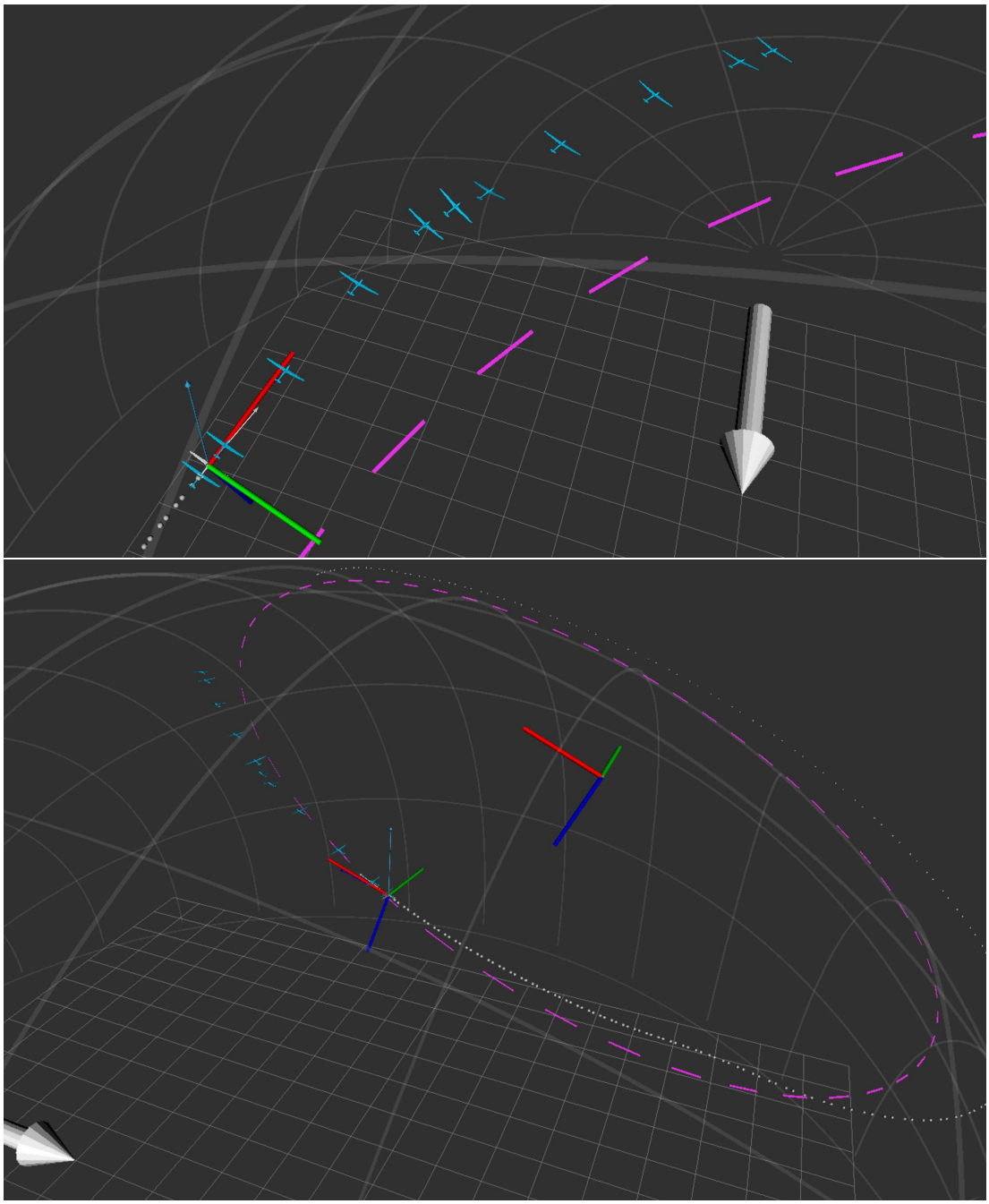


Figure 6.3 – Stabilised NMPFC controller on AWE circle path in simulation. The simulator model and the model used for predictive control differ from each other to simulate the model uncertainty in the real world.

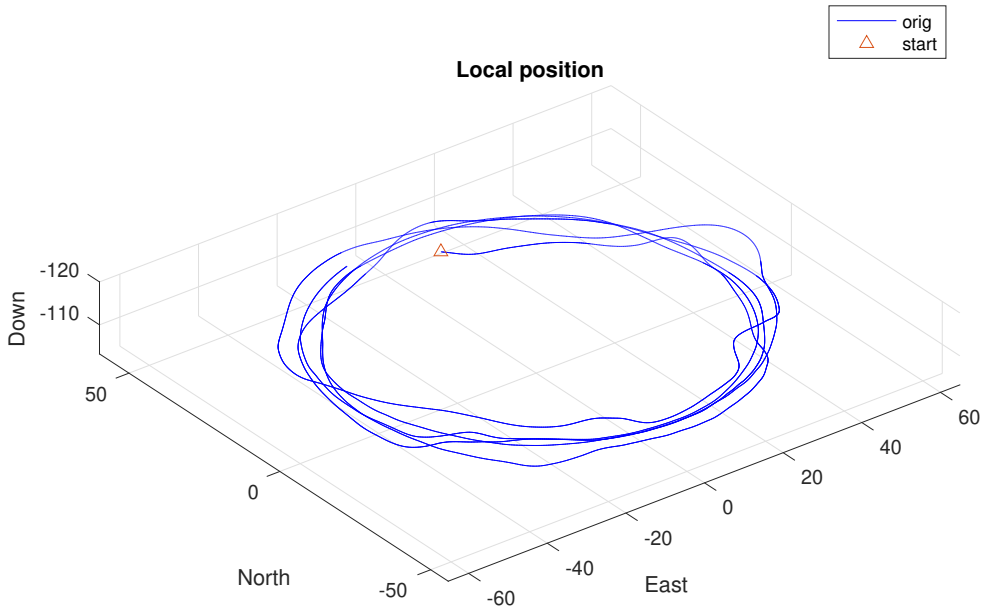


Figure 6.4 – Supervised MPC circle flight (untethered).

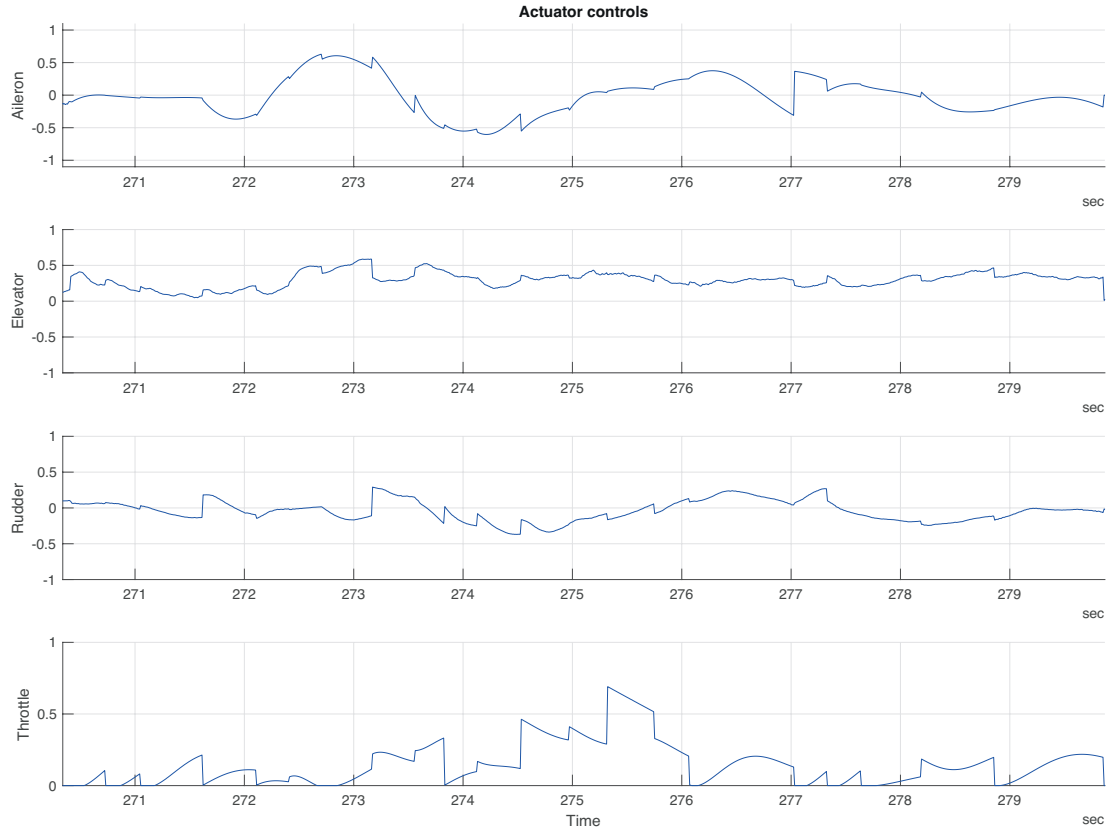


Figure 6.5 – Stabilised MPC control command signals for aileron, elevator, rudder, and thrust. The control commands follow the PX4 convention here, i.e., they are on the range $[-1, 1]$ and $[0, 1]$. The characteristic discontinuities in the signals result from long computation times to solve the OCP while the real trajectory deviates from the last optimal solution.

Part III

Predictive Control for Autonomous Racing

Chapter 7

Predictive Path Following Control for Racing

This chapter presents a new methodology for the real-time dynamic trajectory optimisation in the context of the Roborace competition. In this competition, academic and industrial research teams compete in the development of advanced navigation, trajectory planning and motion control algorithms for fully autonomous electrically powered race cars. During the race, the vehicles similar to the one portrayed in Figure 7.1 drive at high speeds up to 200 [km/h] and accelerations up to 1.3 [g], which in turn requires high sampling rates and robustness of the motion control system. This motivates our work on the real-time trajectory planning considering dynamical limitations of the race car operating at the limits of handling. The developed hierarchical algorithm generates a local trajectory based on typically coarse motion plans while respecting the nonlinear dynamics of the car, tire saturation constraints, delays in the steering and braking mechanisms and power limits of the electric motors.

Our approach reduces the overly aggressive behaviour of the low level tracking systems, often observed in practice, when strong deviations from the desired path occur, which might lead to increased tire slip and activation of the emergency stabilisation systems. This can be explained by the highly nonlinear dynamics of the car at higher speeds and slip angles that are hard to capture with the existing linear controllers that typically assume small deviations from the reference trajectory. The proposed adaptive algorithm relies on the optimal path following control formulation to refine the speed and acceleration profiles, and the driving line geometry in a defined corridor around the local obstacle-free path given the car dynamic limits and current tire and road conditions.

In order to satisfy the strict requirements on the sampling rates for racing applications, we implemented the trajectory optimisation framework in C++ using *PolyMPC* [98], a highly

efficient toolbox for numerical optimisation and optimal control. This allowed us to run the trajectory updates every 10 [ms] on a *Speedgoat* computer (Intel Dual Core-i7, 2.5 GHz) and 20 [ms] on a *NVidia Xavier*, equipped with a 64-bit Quad-Core ARM-57 microprocessor.

Albeit developed for racing applications at extreme speeds, the proposed method can potentially also improve safety and robustness for general advanced driving assistant systems.

The rest of the chapter is organised as follows: In Section 7.1, we provide a brief description of the mathematical model of the car used for optimisation and a short background on the numerical methods for optimal control problems. Then, Section 7.2 presents the key components of the proposed framework: transformation from discrete to continuous path parametrization, path localisation, and formulation of the continuous-time path-following optimal control problem. Finally, in Section 7.4, we demonstrate the hardware-in-the-loop (HIL) performance of the algorithm in racing scenarios using the industrial-grade driving simulator [99].

Figure 7.1 – DevBot 2.0

State of the art

Driving an autonomous vehicle along a racing track involves solving two sub-problems: finding the global, typically time-optimal, or local collision-free trajectory and designing a feedback control algorithm to steer the car along this trajectory. There exist a number of approaches for global racing trajectory and path generation. In [100] the authors iteratively perturb evenly spaced points along the center line of the road to obtain the minimal curvature line by solving a convex QP. The minimum-curvature lines provide a good approximation to the

minimum-time driving trajectories, since in the steady state the maximum cornering speed is inversely proportional to the square-root of the curvature. A similar idea with curvature minimisation is proposed in [101] where a sequence of connected straight lines, clothoids and constant-radius arcs are used to parametrize the racing line. A different approach is suggested by [102], where the authors formulate and efficiently solve a minimum-time mixed-integer nonlinear optimal control problem for a car with a gear box. Unlike the first two approaches, this methodology allows for the computation of the complete trajectory of the vehicle. In [103], researchers extend the idea to spatially varying friction coefficients. The corresponding highly nonlinear optimal control problem is discretized using the direct collocation method and solved offline with *CasADi* [21] and *Ipopt* [15].

Control methods for high-speed racing line tracking usually differ in their assumptions about the vehicle dynamics. The simplest control techniques tend to separate longitudinal and lateral motions of the car. Often, the vehicle is assumed to have neutral steering for lateral control as it allows one to directly relate steering angle and the curvature of the track [104]. For longitudinal control, the accelerated point-mass assumption can be utilised for speed control of the race car [100]. Alternatively, the concept of input-output linearisation can be used to enforce exact trajectory tracking. In [105], the authors consider the center of percussion (CoP) of the front axis as output for I/O linearisation of the simplified bicycle model. They further provide yaw motion stability analysis when tracking straight lines and steady turns. [106] and [107] apply CoP I/O linearisation for lateral motion control of a racing car in an experimental setup. Another approach for lateral motion control is investigated in [108]. The researchers combine linear quadratic regulator (LQR) design for the linearised single-track vehicle model with torque-vectoring. They demonstrate in simulation that this combination improves robustness against the parametric mismatch.

Recent advances in numerical methodologies and software for nonlinear optimal control paved the way for real-time applications of advanced model-based control algorithms in the area of autonomous driving at the limits of handling. These algorithms have an advantage of handling lateral and longitudinal motions simultaneously. One of the first real-time capable implementations of model predictive algorithms for car motion control was presented in [109]. The authors utilise a nonlinear single-track vehicle model with a Pacejka tire model for the optimal path following problem. The code-generation toolkit *ACADO* [24] manages to solve the problem using a real-time iteration (RTI) [66] mode with a sampling rate under 50 [ms]. The vehicle speed used in simulation was limited to 40 [km/h] only, however. Liniger [110] introduces the contouring approach using a similar model with application to small scale race cars. A sufficient sampling rate of 50 [Hz] was achieved by linearisation of the nonlinear optimisation problem and solving a single QP with *FORCES* [111] at each sampling instant. The recent report from the AMZ racing team [112] extends the contouring optimal control

formulation with combined tire-force constraints and solves the corresponding nonlinear OCP to optimality with the commercial toolbox *FORCES PRO* [113] under 65 [ms], and 50 [ms] on average.

When it comes to racing applications, the requirements on computational latency and robustness become much stricter. Due to the extreme driving speeds, the required hard real-time sampling rate of the motion control system is 250 [Hz] on the embedded hardware which renders infeasible the application of nonlinear MPC algorithms for direct actuator control. Constraints on the computation times also limit the prediction horizon of the algorithm, which may lead to crashes if the speed profile is not available globally and potential obstacles make the problem even harder. Several hybrid approaches have been suggested in the literature to cope with these issues. Novi and Liniger [114] propose a hierarchical scheme where a simple point-mass model with track and acceleration constraints is used to compute the path and speed profile over a longer horizon of 250 [m]. An NMPC algorithm with a seven degree-of-freedom (DoF) and much shorter horizon is then used to track the line. Herrmann et al [115] use a graph-based local path planning strategy [116] with an adaptive refinement of the speed-profile. First, a precomputed state-lattice graph is traversed to find a local obstacle-free path, the initial speed profile is estimated from the curvature of the path assuming no slip and then, the speed-profile is refined with a dynamic optimisation-based algorithm that takes into consideration the variable friction coefficient, air drag and engine power limits.

Thanks to the pseudospectral OCP transcription, carefully tuned numerical solvers and software implementation our NMPC algorithm allows sampling times up to an order of magnitude smaller than previously proposed solutions from the literature. Additionally, the continuously parametrised state and control trajectories can be efficiently resampled by a low-level tracking controller if a higher sampling rate is required. Hierarchical structure makes the control system more robust to possible computational delays and convergence issues of the nonlinear solver.

7.1 Modelling of a Racing Car

We start by presenting the ordinary differential equations (ODE) that govern the dynamics of the racing vehicle. A bicycle, or single track vehicle model is chosen here as a trade-off between point-mass models and high fidelity simulation models. In the following we neglect the roll, pitch and heave motions, which is a reasonable simplification for racing cars which typically have a low center of gravity and stiff suspension systems. We further assume the absence of the longitudinal slip as it is significantly lower in practice than the lateral slip. The longitudinal slip also makes the ODEs very stiff and therefore, significantly harder for

optimisation. A planar diagram of the bicycle model is shown in Figure 7.2.

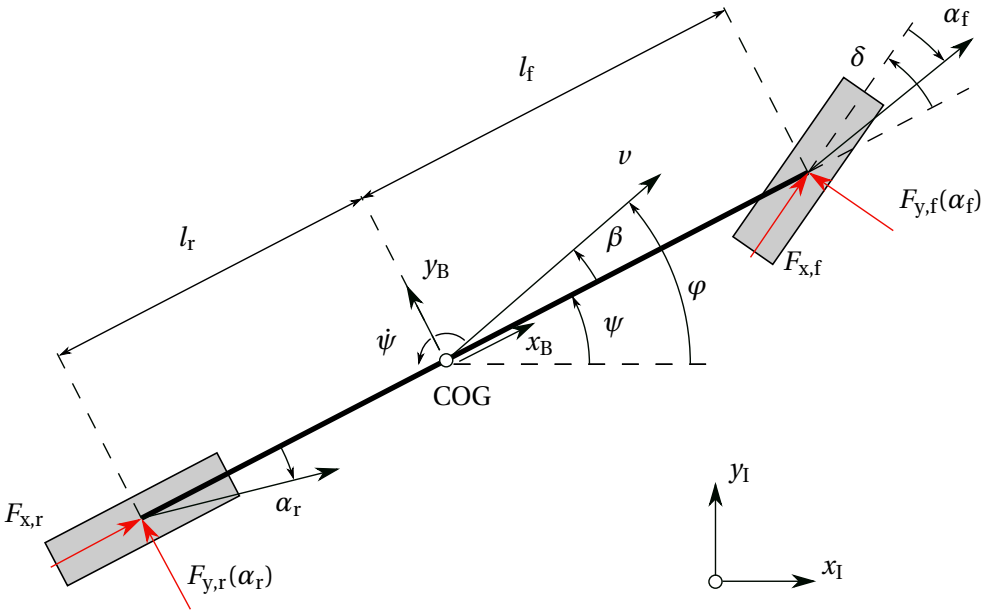


Figure 7.2 – Bicycle model

We first introduce the car dynamics in the natural Cartesian reference frame:

$$\begin{aligned}
\dot{v_x} &= \dot{\psi} v_y + \frac{1}{M}(F_{x,f}\cos\delta - F_{y,f}\sin\delta + F_{x,r} - F_{drag}),\\
\dot{v_y} &= \dot{\psi} v_x + \frac{1}{M}(F_{x,f}\sin\delta + F_{y,f}\cos\delta + F_{y,r}),\\
\dot{\omega} &= \frac{1}{I_z}(-L_r F_{y,r} + L_f(F_{y,f} cos\delta + F_{x,f}\sin\delta)),\\
\dot{X} &= v_x\cos\psi - v_y\sin\psi,\\
\dot{Y} &= v_x\sin\psi + v_y\cos\psi,\\
\dot{\psi} &= \omega,
\end{aligned} \tag{7.1}$$

The first three equations are the dynamics of the system in the car body frame with x pointing forward along the main symmetry axis. The position and orientation in the fixed Inertial reference frame (IRF) is given by X, Y and φ.

This bicycle model takes as input the steering angle δ and the longitudinal forces F_{xf} and F_{xr} on the front and rear axles are summarized in

$$\begin{aligned}
\dot{\eta} &= f(\eta, u)\\
\eta &= [v_x, v_y, \omega, X, Y, \psi]\\
u &= [\delta, F_{xf}, F_{xr}]
\end{aligned} \tag{7.2}$$

There exist several approaches for tire force modelling that differ in numerical complexity. For autonomous driving at moderate speeds, one could use the linear tire model with a constant cornering stiffness. However, for high lateral accelerations in racing scenarios this model becomes inadequate as can be seen in Figure 7.3. It is important to represent the lateral dynamics with a high degree of accuracy, and therefore, the Pacejka magic formula is used for tire models [117]:

$$\begin{aligned} F_{yf} &= F_z D_f \sin\Big(C_f \tan{-1}\big(B_f \alpha_f - E_f(B_f \alpha_f - \tan^{-1}(B_f \alpha_f))\big)\Big), \\ F_{yr} &= F_z D_r \sin\Big(C_r \tan^{-1}\big(B_r \alpha_r - E_r(B_r \alpha_r - \tan^{-1}(B_r \alpha_r))\big)\Big). \end{aligned} \tag{7.3}$$

Front and rear lateral forces each have their own set of magic formula coefficients B, C, D, E and are functions of the slip angles α, which can be modeled as in Equation 7.4.

$$\begin{aligned} \alpha_f &= -\tan^{-1}\left(\frac{v_y + \dot{\psi} L_f}{v_x}\right) + \delta, \\ \alpha_r &= -\tan^{-1}\left(\frac{v_y - \dot{\psi} L_r}{v_x}\right). \end{aligned} \tag{7.4}$$

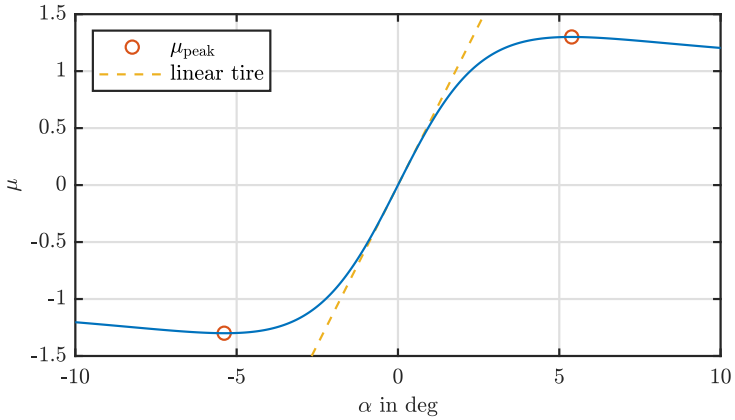


Figure 7.3 – Exemplary curve of the friction coefficient $\mu(\alpha)$ of Pacejka's magic formula.

The navigation information is naturally provided in the Cartesian frame. For the control system, however, it is beneficial to represent the car position in the reference frame linked to the track; the so called Curvilinear reference frame shown in Figure 7.4, which is also local and attached to the car body frame. This transformation allows a natural formulation of the path-following problem, since each point on the path is characterised by a position in the Cartesian frame X_c, Y_c, global heading ψ_c and a curvature κ_c. It is further possible to move from a time to a spatial characterization of the optimal trajectories. Finally, track limits can be added as varying box constraints to the formulation in an efficient manner.

From the position and heading in the Cartesian frame, we calculate the distance w and the

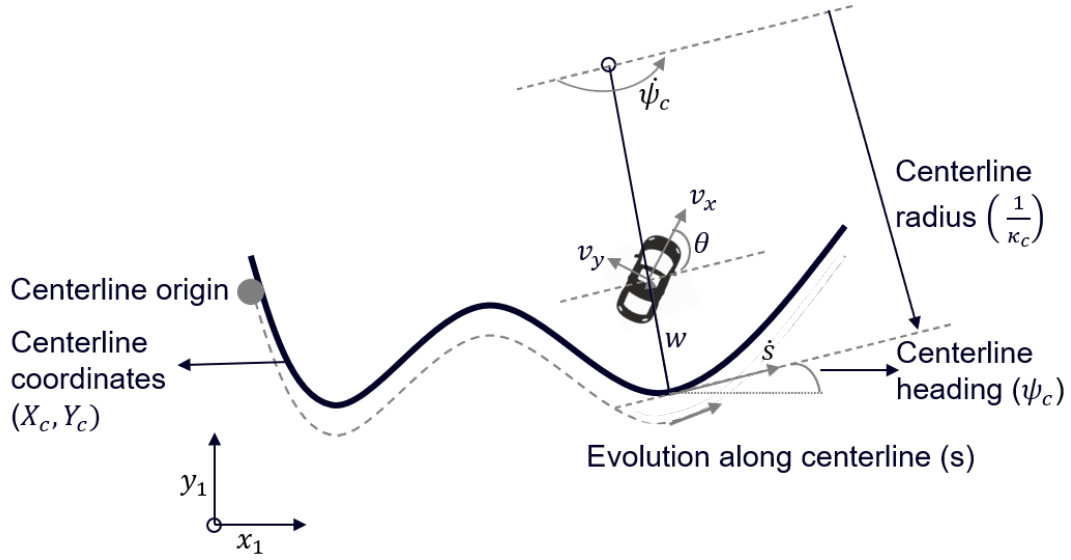


Figure 7.4 – Transformation to the Curvilinear frame

heading deviation θ from the center line, with $w > 0$ to the left of the center line and $\theta > 0$ a counter-clockwise rotation with respect to the tangent to the centerline ψ_c.

$$\begin{aligned} w &= (Y - Y_c)\cos(\psi_c) - (X - X_c)\sin(\psi_c), \\ X &= X_c - w\sin(\psi_c), \\ Y &= Y_c + w\cos(\psi_c), \\ \theta &= \psi - \psi_c. \end{aligned} \tag{7.5}$$

An additional state, s, is added to track the evolution along the racing line. The resulting kinematic equations in the Curvilinear frame are:

$$\begin{aligned} \dot{s} &= \frac{1}{1 - \kappa_c w}(v_x\cos\theta - v_y sin\theta), \\ \dot{w} &= v_x\sin\theta + v_y\cos\theta, \\ \dot{\theta} &= \dot{\psi} - \dot{\psi}_c = \dot{\psi} - \kappa_c\dot{s}. \end{aligned} \tag{7.6}$$

Finally, the race car is affected by actuation delays that can cause instabilities. In order to compensate for these actuation delays and the computation latency of the NMPC algorithm, we introduce a new variable δ_d that denotes the delayed steering input. We approximate the transfer function between δ_d and δ with a first order low-pass filter with a time constant T_d.

$$\frac{\delta_d(s)}{\delta(s)} = \frac{1}{1 + sT_d} \longrightarrow \dot{\delta}_d(t) = \frac{\delta(t) - \delta_d(t)}{T_d} \tag{7.7}$$

State	Description	Unit
v_x	Local longitudinal velocity	$\frac{m}{s}$
v_y	Local lateral velocity	$\frac{m}{s}$
$\dot{\psi}$	Local yaw rate	$\frac{rad}{s}$
s	Distance along center line	m
w	Deviation from center line	m
θ	Heading deviation from center line	rad
X,Y	Global Cartesian coordinates	m
ψ	Yaw angle (heading)	rad
X_c, Y_c	Global center line coordinates	m
ψ_c	Global center line orientation	rad

Table 7.1 – Bicycle model states in curvilinear frame

The list of curvilinear states is presented in Table 7.1, where the velocities and the yaw rate are expressed in terms of the Curvilinear (local) frame and not the global one.

7.2 Trajectory Optimization Framework

In the proposed framework, NMPC acts as a high-level controller that refines the trajectories generated by the coarse discrete planner. It will then produce optimized trajectories and feed-forward control actions that respect the vehicle dynamics, actuator limits and spatial constraints.

Furthermore, the NMPC controller is fit into the existing motion control system with a discrete local planner and a low-level LQR tracking controller [108]. A key benefit of fast trajectory optimization in this application is its ability to handle extreme scenarios like over-steering at high speeds.

The complete trajectory optimization framework is represented in Figure 7.5. Based on the current navigation information, a coarse planner that will be briefly discussed in the next section provides chunks of points representing the obstacle-free local path. In order to use these discrete chunks in our continuous NMPC path formulation, we first fit a cubic spline through an online path parametrization, then compute the car state in the Curvilinear frame and solve a path-following OCP. The NMPC controller computes both the state and feed-forward control trajectories, which are then passed to the LQR controller running at a higher rate.

Coarse Planner and Online Path Parametrization

In this project, a previously developed racing line planner similar to [100] was used. The

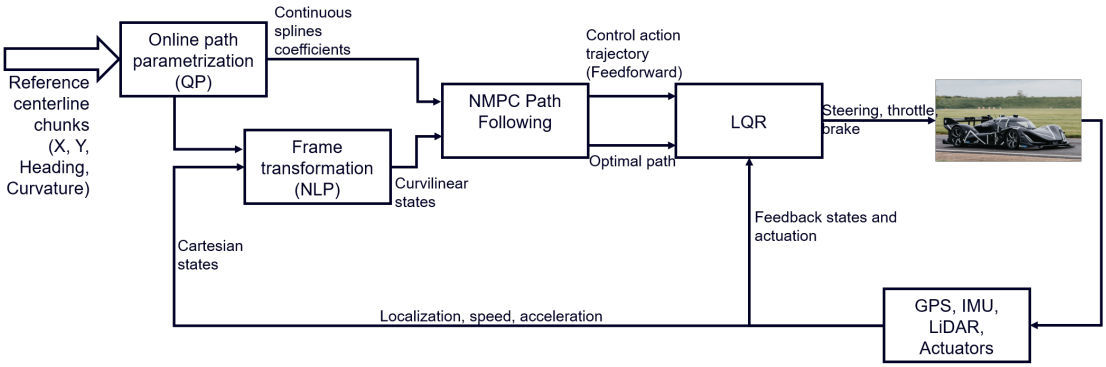


Figure 7.5 – Trajectory optimisation framework

algorithm is initialised with evenly spaced points along the center line of the road and it then searches for an obstacle-free minimal curvature path by perturbing the position of these points in the direction orthogonal to the original path. In order to estimate the speed and acceleration profiles, the method considers a point mass moving along the path. The maximal speed in the turns is defined by the lateral acceleration limits, assuming steady-state motion and the longitudinal acceleration can then be found by maximising the overall speed along the path. Since the method employs a very simplified model of the vehicle it cannot guarantee that the generated path can be always tracked well by the race car. In the proposed scheme, in Figure 7.5, the NMPC layer receives this coarse path and solves a continuous-time optimal path following problem to compute the racing trajectory respecting the car dynamics and actuator constraints.

In the current implementation, the output of the coarse planner is a set of equally spaced points, which we will call a chunk. Each point is characterised by its position, curvature and reference speed. Such a discrete representation is not very efficient for the continuous path following NMPC, and we therefore choose to interpolate the chunk with custom cubic splines that can be efficiently automatically differentiated. In the following, we detail the developed interpolation procedure.

Let y denote one of the path parameters to be approximated by a cubic spline. We divide every data chunk into N_s equidistant segments and fit a cubic spline $y(s)$ parametrised by

polynomial coefficients for each segment $\{P_i\}$.

$$\begin{aligned} \min_{\{P_i\}} \sum_{s=0}^{s_{max}} (y(s) - \tilde{y}(s))^2, \\ \text{subject to: } y(s=0) = \tilde{y}(0), \\ \left.\frac{dy}{ds}\right|_{s=0} = \dot{\tilde{y}}(0), \\ \left.\frac{dy}{ds}\right|_{s=s_{max}} = \dot{\tilde{y}}(s_{max}), \end{aligned}$$

where s_{max} is the maximum spline length and $\tilde{y}(s)$ is data evaluated at the grid points and the derivative constraints enforce continuity of the two neighbouring segments.

We formulate a QP in the form of $\frac{1}{2}(D^T P - \tilde{Y})^2$ where P is the coefficient vector of all segments in one spline, and D is the data vector containing all the grid points.

$$\begin{aligned} D_i &= \begin{bmatrix} 1 & \dots & 1 \\ s_{0,N_i} & \cdots & s_{max,N_i} \\ s_{0,N_i}^2 & \cdots & s_{max,N_i}^2 \\ s_{0,N_i}^3 & \cdots & s_{max,N_i}^3 \end{bmatrix} \in \mathbb{R}^{4 \times N_{c,i}} \\ D &= \begin{bmatrix} D_1 & & \\ & \ddots & \\ & & D_{N_s} \end{bmatrix} \end{aligned} \tag{7.8}$$

where $N_{c,i}$ is the number of points in one segment of the chunk such that $\sum_{i=1}^{N_s} N_{c,i} = N_c$, with N_c the total number of points received in one chunk. In this application 50, $\tilde{Y} \in \mathbb{R}^{N_c \times 1}$ is the vector of true data and $P \in \mathbb{R}^{4 \times 1}$ is the coefficients vector to optimize over.

This least squares formulation has the same minimum as the equivalent QP problem:

$$\begin{aligned} \min_X \frac{1}{2} X^T H X + h^T X, \\ \text{subject to: } AX = b. \end{aligned} \tag{7.9}$$

where H, h, A are the Hessian, gradient and constraint Jacobian of the problem and b is the

constant vector of data input.

$$A = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_{N_s} \end{bmatrix}$$

$$A_{N_i} = \begin{bmatrix} 1 & s_{0,N_i} & s_{0,N_i}^2 & s_{0,N_i}^3 \\ 0 & 1 & 2s_{0,N_i} & 3s_{0,N_i}^2 \\ 0 & 1 & 2s_{max,N_i} & 3s_{max,N_i}^2 \end{bmatrix}$$

$$H = DD^T,$$

$$h = -D\tilde{Y}$$

$$b = [\tilde{y}_1(0), \dot{\tilde{y}}_0(0), \dot{\tilde{y}}_1(s_{max,1}), \ldots, \tilde{y}_{N_s}(0), \dot{\tilde{y}}_{N_s}(0), \dot{\tilde{y}}_{N_s}(s_{max,N_s})]^T$$

A and H do not depend on the data but only on the data grid (in terms of s). Therefore for problems where the chunks are given in the same discrete grid samples, the two matrices can be computed only once, with h being computed for every new data set.

We approximate with splines the center line (X_c, Y_c), desired heading (tangent) (ψ_c) and curvature (κ_c). A cubic spline is also fit for the reference velocity profile ($v_{x,ref}$). Discrete chunks are received in grids spanning over 25m, and we fit a spline with two segments of 12.5m each, sufficient for our formulation. The resulting equality-constrained dense QP problem can be efficiently solved in microseconds on modern hardware.

A clear advantage arising from this continuous spline formulation is instead of carrying 100 or more data points for every road parameter, we are able to transmit $4N_s$ spline coefficients; enough to describe the parameter in question on the spline made out of N_s segments. In this application, as a result of this compression we reduce the amount of data transmitted between blocks by almost 90%.

Frame Transformation and Path Localisation

As mentioned earlier, navigation is represented in the Cartesian frame. Therefore, in order to convert to the curvilinear frame using (7.5), we must first find the projection of the car onto the racing line.

This projection problem is sketched in Figure 7.6 and written as a minimization problem of

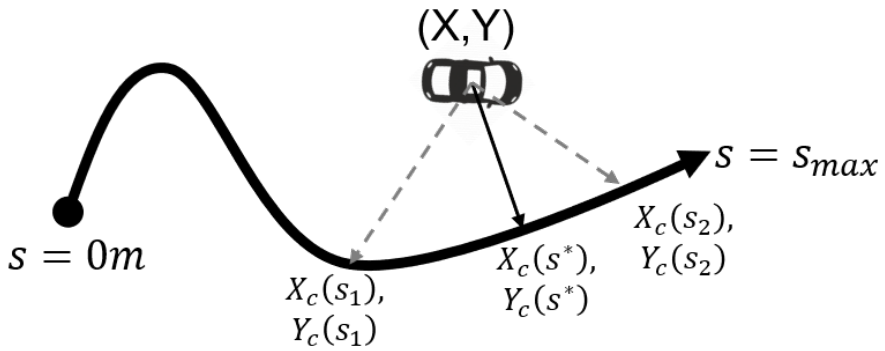


Figure 7.6 – Localization on a path

the form:

$$\begin{aligned} &\min_{s} \|p - p_c(s)\|^2 \\ &\text{subject to} : 0 \leq s \leq s_{max} \end{aligned} \tag{7.10}$$

Where $p = [x, y]^T$ is the current positon of the car and $p_c(s) = [x_c(s), y_c(s)]$ is a parametric path, which in our project is the cubic splines as defined in the previous subsection.

Once an optimal solution of (7.10), $s^\star$, is found, κ_c and $v_{x,ref}$ can be computed and the state vector converted to the Curvilinear frame. This NLP is solved using the PolyMPC toolbox using the dense SQP solver.

Path Following NMPC

At the core of the OCP are the differential equations defined in (7.1) and (7.6). For smooth driving, hard rate constraints on the steering and acceleration rates are required. For this, we augment the OCP system states with the steering control, delayed steering and the two longitudinal forces such that:

$$\begin{aligned} &\frac{d}{dt}\delta = \dot{\delta} = v_1 \\ &\frac{d}{dt}\delta_d = \frac{\delta(t) - \delta_d(t)}{T_d} \\ &\frac{d}{dt}F_{x,f} = \dot{F}_{x,f} = v_2 \\ &\frac{d}{dt}F_{x,r} = \dot{F}_{x,r} = v_3 \end{aligned} \tag{7.11}$$

The resulting augmented state vector is denoted with ξ and control with v. To take into

consideration the delay, the bicycle model dynamics are evaluated with δ_d as indicated in

$$\begin{aligned} \xi &= [v_x, v_y, \omega, s, w, \theta, \delta, \delta_d, F_{x,f}, F_{x,r}] \\ \upsilon &= [\dot{\delta}, \dot{F}_{x,f}, \dot{F}_{x,r}] \\ \dot{\xi} &= F(\xi, \upsilon) = F\big(f(\eta, \{\delta_d, F_{x,f}, F_{x,r}\}), \upsilon\big) \end{aligned} \tag{7.12}$$

The continuous-time nonlinear path following formulation is presented in Equation 7.13. The constraint on the deviation from the center line w is such that at every instant w is bounded between the right and left boundaries of the lane. The slip angles are constrained to avoid tire saturation.

$$\begin{aligned} \min_{\xi(.),\upsilon(.)} \int_{t_0}^{t_f} & l(\xi(t), \upsilon(t), p)\, dt + V_f(\xi(t_f)) \\ \text{subject to: } & \xi_0 = \xi(0) \\ & \dot{\xi}(t) = F(\xi(t), \upsilon(t), p) \qquad t \in [t_0, ..., t_f] \\ & 0 \le \frac{w(t) - w_r(t)}{w_l(t) - w_r(t)} \le 1 \\ & \theta_{min} \le \theta(t_i) \le \theta_{max} \\ & v_{x,min} \le v_x(t) \le v_{x,max} \\ & v_{y,min} \le v_y(t) \le v_{y,max} \\ & \dot{\psi}_{min} \le \dot{\psi}(t) \le \dot{\psi}_{max} \\ & 0 \le s(t) \\ & \alpha_{min} \le \alpha(t) \le \alpha_{max} \\ & \delta_{min} \le \delta(t), \delta_d(t) \le \delta_{max} \\ & F_{x,min} \le F_{x,f}(t), F_{x,r}(t) \le F_{x,max} \\ & \dot{\delta}_{min} \le \dot{\delta(t)} \le \dot{\delta}_{max} \\ & \text{jerk}_{\text{x,min}} \le \dot{F}_{x,f}(t), \dot{F}_{x,r}(t) \le \text{jerk}_{\text{x,max}} \end{aligned} \tag{7.13}$$

We do not assume equal front and rear longitudinal forces so that the control allocation strategies can be added at a later stage. Nevertheless, large differences are penalised in the cost function. The stage cost $l(x, u)$ is defined as:

$$l(\xi(t), \upsilon(t), p) = (\xi^*(t) - \xi^*_{ref}(t))^T Q (\xi^*(t) - \xi^*_{ref}(t)) + \upsilon(t)^T R \upsilon(t) + \sigma (F_{x,r} - F_{x,f})^2 \tag{7.14}$$

with $Q \in \mathbb{R}^{10\times10} \succeq 0, R \in \mathbb{R}^{3\times3} \succ 0, \sigma \in \mathbb{R} \geq 0$

Tracking is equivalent to keeping w and θ close to zero. The coarse planner provides a velocity

profile which helps to keep the optimisation horizon shorter. Another benefit of the track-centered formulation is the position constraint which can be simplified box constraints on the variables, which makes tracking the local racing line within a given corridor simpler.

7.3 Implementation

This section presents some implementation details of the framework modules shown in Figure 7.5. We discuss the different deployment strategies and numerical heuristics that were used to speed-up and robustify the computations.

NVidia Xavier

Since the complete motion control system developed by the Arrival Racing team is not yet available on the NVidia Xavier platform, we apply the spline fitting algorithm (7.9) to the complete pre-recorded racing line and the racing line is not, therefore, updated during the simulation. Finally, communication with the simulator is implemented with ROS [60].

Simulink and Hardware Deployment

Currently, the deployment of the entire motion control system to the target racing platform *Speedgoat* is performed by means of the *Matlab Code Generation Toolbox*. This requires that each component of the framework has to be implemented as independent *Simulink* blocks. The absence of code-generated components and header-only nature of the library allows for simple code integration.

We use S-function builder blocks to call our custom C++ algorithms in Simulink by creating blocks for the several class objects and implementing the different methods inside the S-function builder. The resulting S-function can be connected to any other Simulink block by specifying the required input and output sizes and widths of the signals. Finally, compilation of the S-function builder blocks requires only specification of the path to C++ codes without the need to link to any library or binary. At run-time, every block based on our custom toolbox and codes operates as a standalone executable, which was compiled via Simulink using Microsoft Visual Studio 17.

The simulation study is carried out with a high-fidelity numerical simulator of the vehicle dynamics and low-level trajectory tracking LQR controller. Due to the hard real-time constraints, the trajectory optimisation block is set to execute at a lower frequency than the rest of the motion control system. Using the rate transition blocks in *Simulink*, input to the trajectory optimization is down-sampled and later the output is re-sampled at a higher frequency by the LQR. Although running at a slower rate, the trajectory optimization still operates at almost

70Hz.

Heuristics: Fault detection

The optimisation framework is designed for racing applications at speeds greater than 100 [km/h]. Because of safety concerns, we introduce a trajectory quality measure that is passed down to the low-level tracking level. The quality measure takes into consideration constraint violation, the primal and dual residuals and the computation time. If a fault, or insufficient quality, in the solution is detected, then the LQR controller can switch to the previously computed optimal trajectory or to the coarse path if necessary.

Heuristics: OCP scaling

In order to improve the numerical conditioning of the OCP (7.13) and reduce computation times we introduce a scaling of the state and control variables as recommended in [118]: $\xi^* = \Sigma_\xi \xi$ and $v^* = \Sigma_v v$ such that all components of ξ^* and v^* are of the same order of magnitude. The system dynamics then become:

$$\xi^*(t) = \Sigma_\xi^{-1} F(\Sigma_\xi^{-1}\xi^*(t), \Sigma_v^{-1} v(t), p) \tag{7.15}$$

Matrices Σ_ξ, Σ_v, Σ_ξ^{-1} and Σ_v^{-1} are diagonal in our implementation and do not change at run-time. Note, that all the bounds and weights in (7.13) must also be scaled accordingly.

Heuristics: Variable horizon length

Unlike [109], we do not completely eliminate the time from the OCP. Therefore, we need to ensure that the optimisation algorithm does not attempt the access the path beyond the 25 meters limit, because the data describing the raceline only goes this far. For this, the prediction horizon is conservatively scaled using the maximum allowed speed on the interval:

$$t_f = \frac{s_{max}}{\max(v_{x,ref}(s))} \tag{7.16}$$

7.4 Results

In this section, we present a simulation study with a high-fidelity racing environment on two circuits that feature in the Roborace competition: Bedford (England) and Croix-en-Ternois (France). First, we investigate the computational performance and real-time capability of the NMPC algorithm on three different hardware platforms: Desktop PC, *Speedgoat real-time target machine* and a low-power *NVidia Xavier* embedded computer. Next, we demonstrate

that the proposed hierarchical approach yields better lap time and robustness to unknown road conditions than the existing LQR controller.

Processor-in-the-Loop Benchmark

The presented framework modules are implemented as header-only C++ libraries with a strong adherence to the C++11 coding standard, which has significantly simplified the portability of the software across various hardware platforms. For Processor-in-the-loop (PiL) testing, the code was deployed to a *Speedgoat* along with the existing motion control system through the *Simulink Code Generation* toolbox. Our numerical studies were performed on the Croix-en-Ternois racing track shown in the Figure 7.7 with a high-fidelity simulation tool [99] that runs on a desktop computer.

For the path following problem (7.13), the control and state trajectories are interpolated by a two-segment spline, where in each segment a Lagrange polynomial of order five is collocated on a Chebyshev-Gauss-Lobatto grid. The Hessian of the Lagrangian for the corresponding NLP is computed exactly for the first SQP iteration and updated using a sparsity preserving block-BFGS update step. The maximum number of iterations is set to five and the primal and dual residuals are set to 10^{-5}.

A summary of the tested computational platforms and computational benchmark is shown in Table 7.2.

Platform	PC	Speedgoat	NVidia Xavier
System	Windows	QNX 7.1	Linux
Compiler	Visual C++	qcc	gcc
OCP (7.13)	6.21/7.9	10.07/11.37	15.43/18.21
Spline fit. (7.9)	0.021/0.027	0.032/0.037	0.053/0.061
Path loc. (7.10)	0.012/0.16	0.019/0.023	0.044/0.051

Table 7.2 – PiL Simulation benchmark: average and maximum computation times in milliseconds [avg/max] on various platforms. Our implementation is faster than other implementations in the literature [114, 110, 112] for a similar OCP complexity and prediction horizon length. It also shows good scalability on embedded platforms thanks to efficient memory management and vectorisation.

Racing Performance

As shown in the previous subsection, the NMPC algorithm can successfully track a racing line and satisfies hard real-time constraints on the automotive hardware. As a next step, we compare the racing performance of the developed hierarchical control scheme to an existing

and carefully tuned LQR controller. In the provided racing scenario the look-ahead range is limited to 25 [m], NMPC is set to run at 16 [ms] and the optimal control and state trajectories are tracked by the LQR at a constant rate of 250 [Hz]. An initial race line is provided by a coarse planner discussed in Section 7.2 which uses a quasi-static car model and therefore, tends to overestimate feasible speed. NMPC is allowed to change the speed profile and geometry of the initial line within a corridor dynamically provided by the planner.

Figure 7.7 illustrates the path driven by the NMPC algorithm using the high fidelity simulator. In Figure 7.8 one can observe the difference in the driving performance of NMPC and LQR algorithms. We notice that the LQR steering is smoother as our predictive controller chooses to steer more aggressively to initiate a turn and then performs a sawing motion past the apex. Interestingly enough, such a steering strategy is often used by professional pilots to counteract the oversteering which occurs when the vehicle decelerates and consequently the load increases on the front axle. This load transfer from the rear to the front axle during a cornering manoeuvre reduces the grip of the rear tires and leads to higher lateral slip. By rapidly steering in the opposite direction NMPC controller avoids oversteering and allows passing the turns up to 3 [km/h] faster than LQR. As a consequence, our controller shows 0.6 [s] better lap time than the existing professionally tuned controller which is a substantial difference for racing, especially in simulation. Additional high-order oscillations are introduced by the low-level tracking controller as follows from Figure 7.9c. Deviations from the initial race are shown in Figures 7.9a,7.9b. These deviations are higher for the NMPC algorithm than for the existing LQR controller as it is allowed to change the line geometry in order to better reflect the dynamic constraints of the car. It is evident from Figure 7.7, however, that the modified line does not violate racing circuit boundaries. Finally, during this one-lap race, NMPC algorithm always returned valid solutions, i.e. the backup controller was never active.

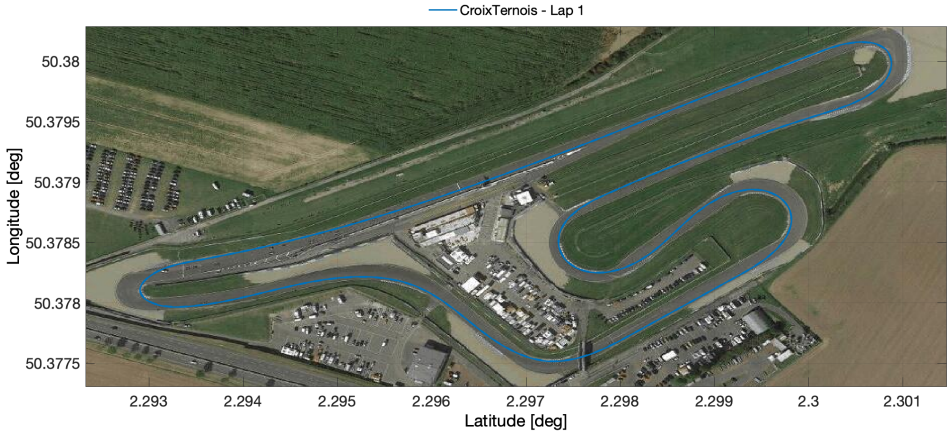
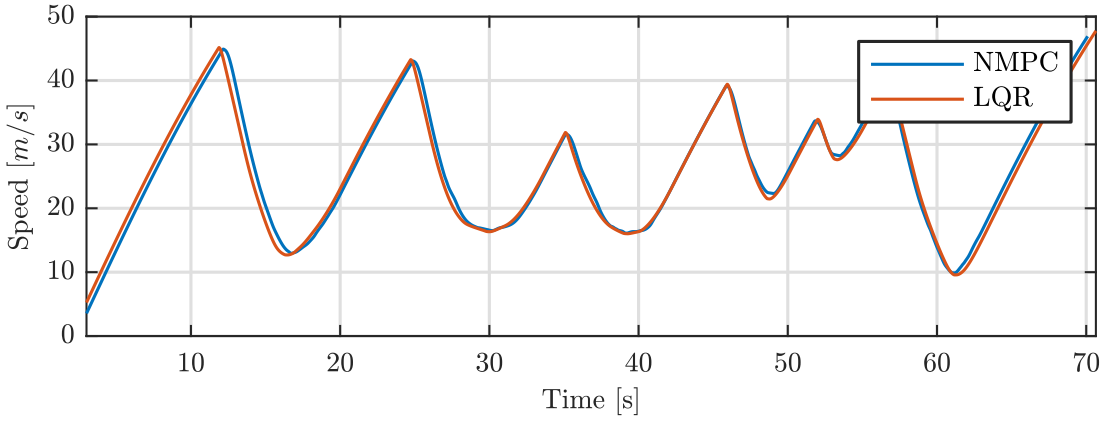
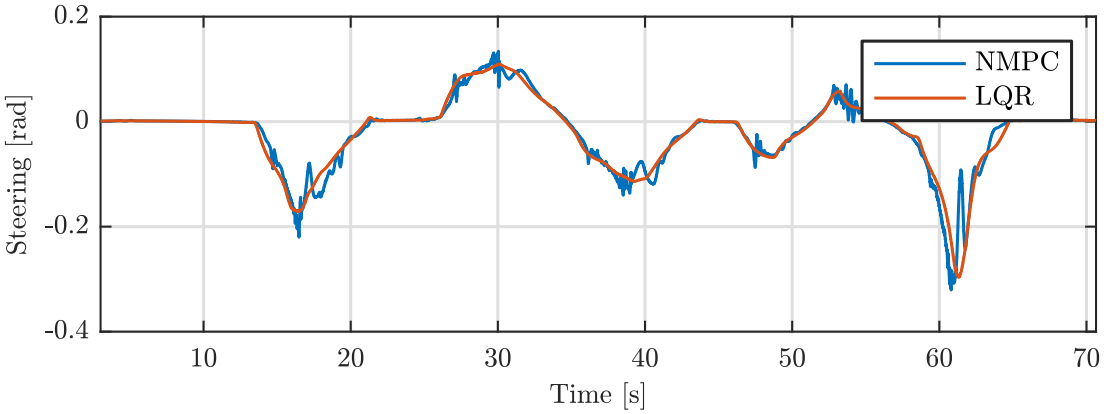


Figure 7.7 – Closed-loop trajectory on the Croix-en-Ternois circuit.

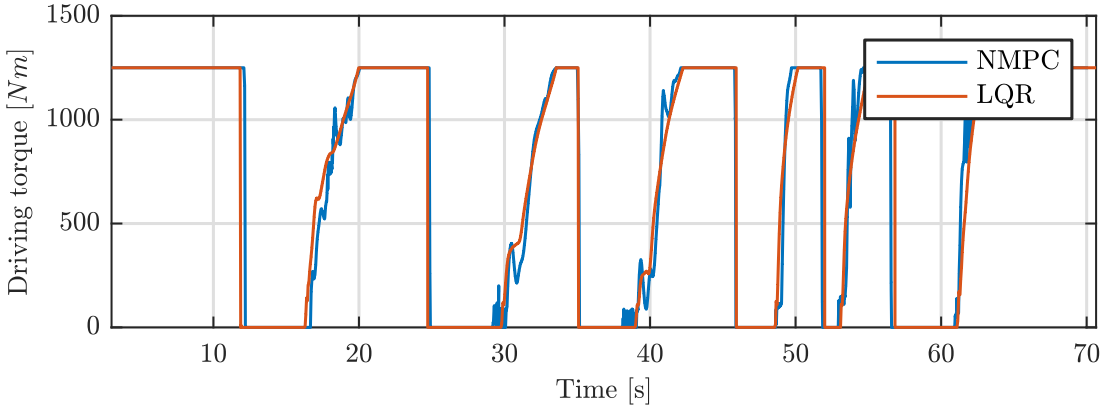
Robustness



(a) Speed profiles of the NMPC and LQR controllers. The NMPC algorithm drives the car slightly faster during cornering manoeuvres by exploiting lateral slip predictions

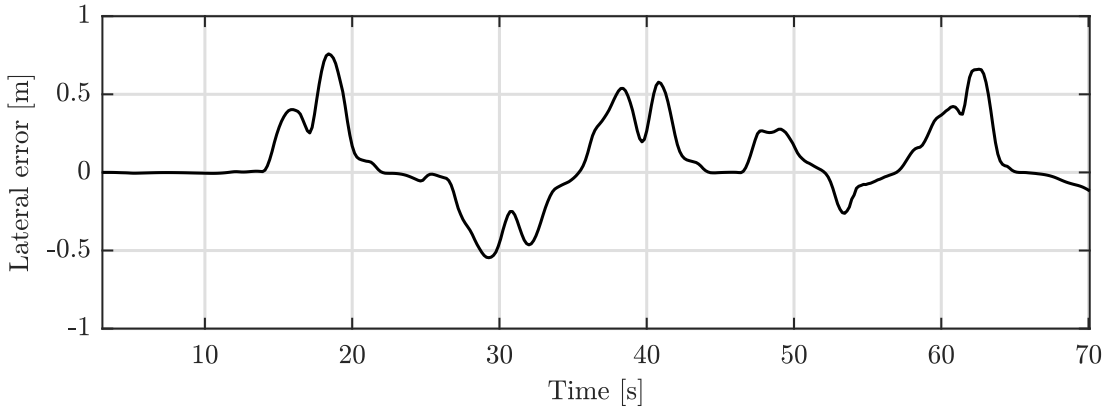


(b) Steering profiles of the NMPC and LQR controllers. NMPC counteracts oversteering by reducing steering angle at the apex where braking effort is maximal.

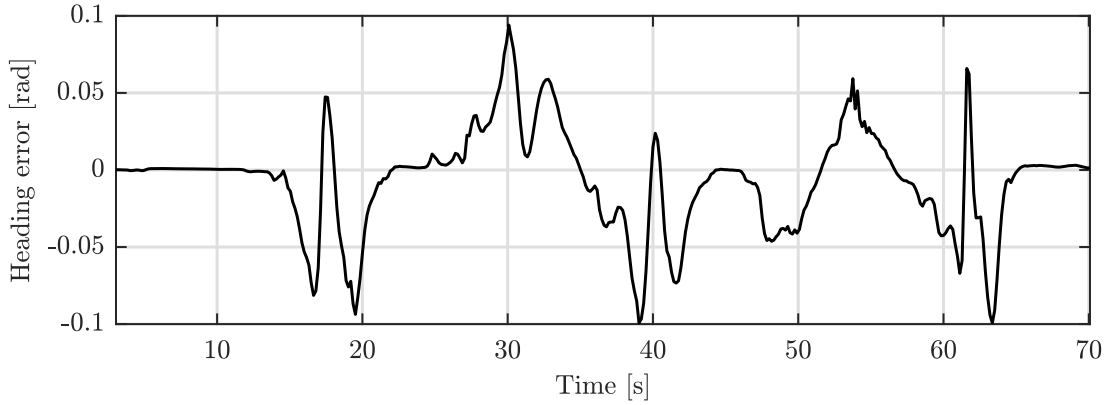


(c) Driving torque applied by the NMPC and LQR controllers.

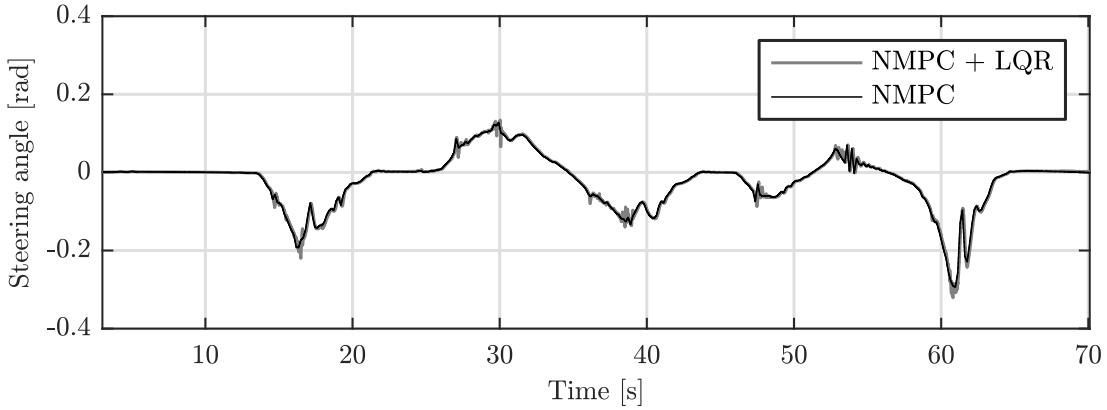
Figure 7.8 – Comparison of the speed profiles and control strategies of the NMPC and LQR controllers along the Croix-en-Ternois circuit.



(a) Deviation from the initial race line. Even though, the deviation from the initial line reaches 0.76 meters the car remains within the track bounds.



(b) Deviation from the intended heading, i.e. tangent direction of the initial race line.



(c) Feedforward and feedback portions of the steering signal. Solid black line is the feedforward signal provided by NMPC, solid grey line is feedback portion coming from the low-level LQR controller tracking the updated race line at a higher rate. The LQR controller is provided by the Roborace engineers as a black box.

Figure 7.9 – NMPC tracking of the initial path and the stabilising LQR controller

In the next experiment, we explore the robustness of our control scheme to uncertain environment parameters. The racing scenario is set at the Bedford circuit that features a chicane, highlighted in Figure 7.10 that requires fast switching of the steering angle at high speed. In this scenario, the road friction coefficient was reduced by 15% in the simulation environment which corresponds to a light drizzle in reality. The tuning of NMPC and LQR controllers remain as in the previous experiment and the racing line and speed profile are optimised for a dry circuit with a maximum allowed lateral acceleration of 1.1 $[g]$.

Figure 7.11 demonstrates how the LQR controller fails to pass the chicane and goes off the track while the NMPC algorithm is able to steer the car very close to the original path. One can notice that the LQR controller initiates left turn a little later than NMPC which means that it has to steer more aggressively. However, with less grip on the front axle the clearly understeers as can be seen in Figure 7.12a and triggers activation of the traction system around 52 [s] that reduces the steering input and torque demand to improve the grip and lower the slip angles. Since the NMPC can predict the tire saturation at high slip angles it initiates the turns earlier and does not use the full range of the steering wheel.

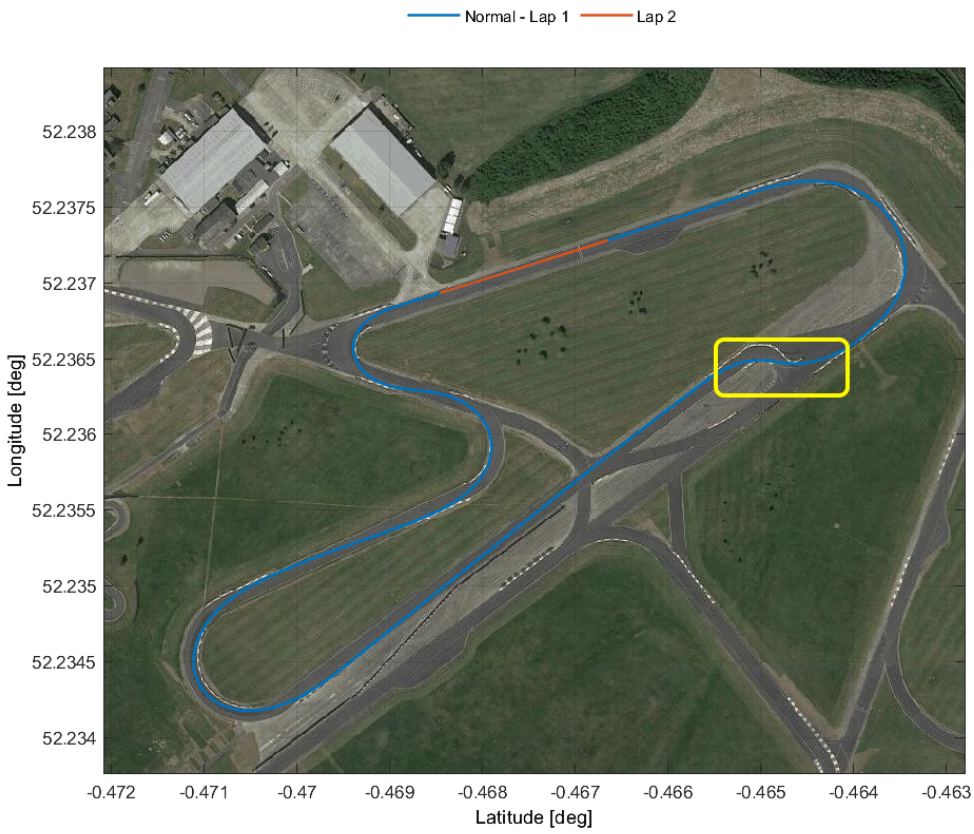


Figure 7.10 – Path driven by the NMPC controller on the Bedford circuit.

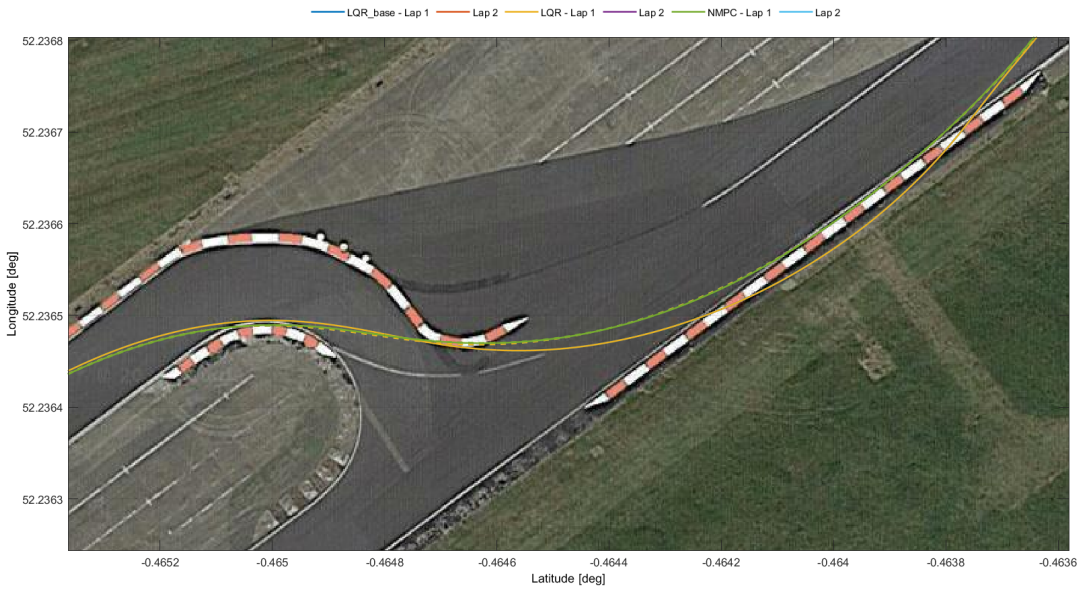
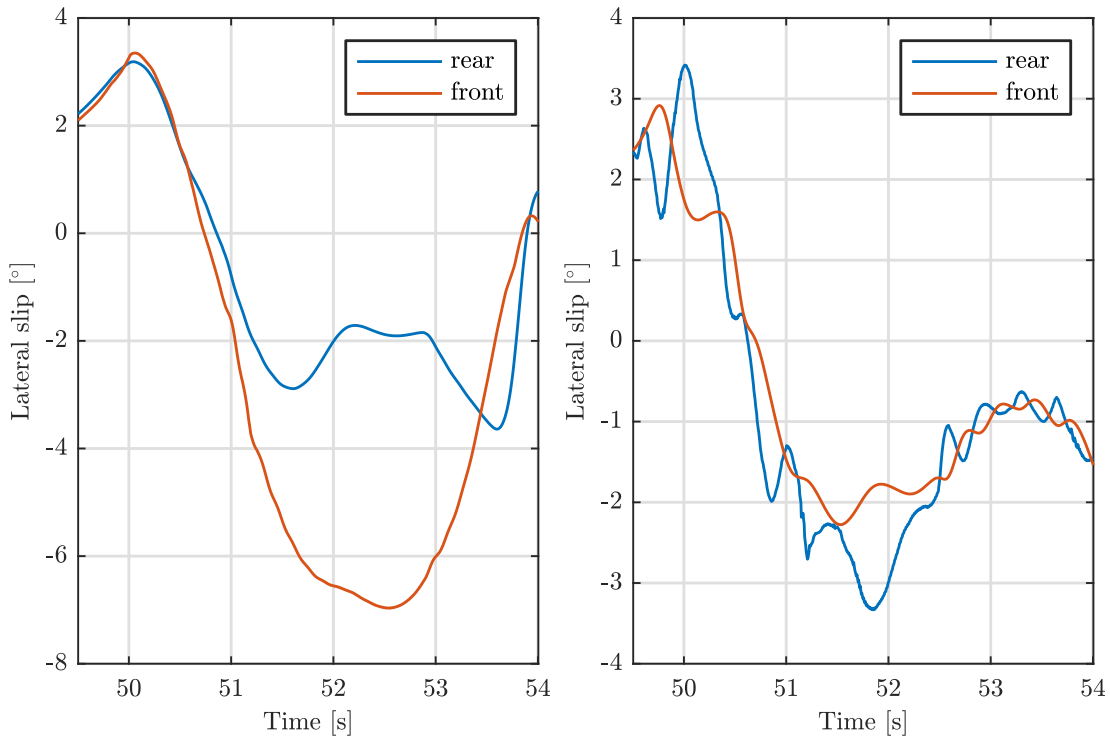


Figure 7.11 – Comparison of the LQR and NMPC controllers driving chicane on a slightly wet circuit. The dashed yellow line is the original racing path, the solid yellow line is the path driven by LQR and solid green line is path driven by NMPC. The car driven by LQR understeers due to aggressive steering at high speed and goes of the track, whereas the NMPC controller follows the racing path very closely.

7.5 Summary

The presented real-time embedded trajectory optimisation framework for local path refinement demonstrated two key advantages of optimisation-based solutions for autonomous electric vehicle control at the limits of handling. First, it allows to improve the racing performance at the limits of handling over the state of the art controller in simulation. Second, due to predictive capabilities it has shown to be more robust to the road friction uncertainty and to reduce the involvement of the low-level car traction control systems during high speed tracking of the racing lines. Importantly, the pseudospectral collocation OCP transcription and efficient implementation allowed for very high sampling rates even on automotive grade embedded platforms. The method is expected to be tested on the real track in the near future.



(a) Lateral slip angles during driving the chicane with front wheel and understeering.

(b) Lateral slip angles during driving the chicane with LQR. High steering input causes a high slip angle of theNMPC.

Figure 7.12 – Comparison of lateral slip angles. The maximum lateral slip of the front wheel is three times lower for NMPC than for LQR.

Chapter 8

Stochastic NMPC for Safe Autonomous Driving

In optimal control problems, uncertainty can be modelled in different ways: parametric or structural uncertainty of the plant model, the state measurements being subject to uncertainty, or unmeasured disturbances entering model equations as noise. In the presence of path constraints, quantifying and predicting state evolution uncertainty is necessary to allow for a robust trajectory optimisation and control design. Constraint violation due to uncertainty can be reduced by keeping sufficient distance from the constraints, or tightening them. On the other hand, an overly conservative handling of uncertainty leads to poor performance, and thus better estimation of the state uncertainty leads to an improved compromise. [119] [120]

We propose a computational methodology for reducing the conservatism of stochastic nonlinear optimal control problems with parametric uncertainty. The approach builds on the idea of approximating stochastic processes governed by nonlinear controlled differential equations with uncertain parameters using the generalised polynomial chaos expansion and controlling spectral modes of such expansion with an ancillary feedback controller. Advantages of the method are illustrated with a safe trajectory optimisation problem relevant for autonomous driving applications.

In order to handle uncertainties in the system, robust optimal control methodologies have been developed. The methods can be classified by the way they handle constraints, the type of uncertainty they address, and the type of uncertainty prediction. Constraints can be either violated with a probability greater or equal zero, so called chance constraints, or with probability zero, which implies that the uncertainties or disturbances have to be bounded. The uncertainty can either be time variant or constant. The prediction can either be open-loop without consideration of a feedback controller or closed loop.

Tube-based model predictive control (MPC) schemes handle disturbances by an ancillary stabilizing feedback controller, and do an open-loop prediction for the nominal model. To account for the interference of the ancillary controller, constraints are tightened in the open-loop prediction [121]. This implies that a stabilizing controller and the amount of tightening have to be appropriately designed. Both tasks are challenging for nonlinear systems and are subject to current research. Mayne et al [122] suggests a dual-MPC approach to handle additive time varying disturbances, where both the ancillary and the nominal controller are model predictive controllers. Singh [123] develops a contraction-based framework to design the ancillary controller and to define the constraint tightening for input- and disturbance-affine nonlinear systems. Nubert [124] develops a Lyapunov-based constraint tightening methodology for nonlinear systems assuming the ancillary controller is known and applies the theory to a feedback-linearized robotic manipulator.

While classic tube-based MPC assumes the disturbances to be bounded and guarantees constraint satisfaction for all disturbance realizations, stochastic optimal control introduces chance constraints, which means that constraint satisfaction is only guaranteed up to a certain probability [120]. The authors in [125] combines stochastic MPC and the so called scenario-based approach to address highway driving scenarios, where the manoeuvres of the surrounding vehicles are uncertain. Simple linear point mass models facilitate computation. The scenario approach randomly samples time varying uncertainty and solves the OCP without considering feedback. The constraints are formulated as chance constraints and, as is common for scenario-based approaches, the confidence of their satisfaction is influenced by the number of samples [126].

In order to avoid sampling for the uncertainty propagation, approaches based on so called generalized polynomial chaos (gPC) have been developed [120]. They transform a stochastic ordinary differential equation (ODE) into a deterministic ODE by approximating the random processes by polynomials with time varying coefficients; for a detailed explanation see Section 8.1. Still, the probability density function of the uncertain states needs to be reconstructed to enforce chance constraints, which is done in [127] by efficient sampling. Since the prediction does not consider feedback in the open-loop, trajectories of realizations of the uncertainties might diverge quickly, which leads to a growing variance and thus conservatism [121]. Authors in [128] use Gaussian processes and a race car model linearized around the reference trajectory to run stochastic nonlinear MPC with open-loop predictions. To reduce conservatism, chance constraints are only considered for the short initial part of the optimization horizon.

A key challenge for general nonlinear systems is that there are no closed form solutions for the uncertainty propagation, which is crucial to reduce conservatism. Generalized polynomial chaos (gPC) can serve as a method to treat time invariant uncertainties, e.g., in the parameters

or the initial conditions of initial value problems. [120] An example of linear stochastic MPC with gPC is given in [129], where the authors use the Galerkin projection method and a distributionally robust probabilistic inequality to steer a stable linear system. An advantage of the Galerkin projection for linear ODEs is that expansion coefficients, or the so called Galerkin tensor, can be computed offline. Fagiano [130] proposes a regularised stochastic collocation-based MPC method for stabilisation of nonlinear systems. In their approach, however, the constraints are satisfied only in expectation.

We combine the idea of stochastic and tube-based MPC algorithms: a nominal optimal control formulation is augmented by a stochastic system stabilized around the nominal system by an ancillary feedback controller, whose parameters can also be subject to optimization. The applied control action is then the sum of the nominal MPC and the ancillary controller. In order to ensure satisfaction of input constraints, the control authority of the ancillary controller is restricted by a saturation function. The path constraints are formulated as chance constraints using a spectral representation of uncertainties and distributionally robust probabilistic inequalities, which tighten the constraints of the nominal MPC in an adaptive fashion taking the predicted uncertainty of the stabilized stochastic system into account.

Contributions

In this thesis, we present several contributions to the field of robust trajectory planning and control for autonomous vehicles.

First, time invariant uncertainties are considered using a stochastic nonlinear OCP with gPC. Thereafter, the computational toolbox *PolyMPC* is extended to transform stochastic OCPs with chance constraints into deterministic OCPs using gPC expansions. Using open-loop predictions, especially for unstable systems, in the optimisation algorithm leads to conservatism, which is demonstrated for an example problem of generating a vehicle trajectory respecting road boundaries. To overcome this conservatism, a robust stochastic optimal control formulation is proposed. The formulation combines the idea of a tube-based approach with the capability of gPC based stochastic optimal control: an ancillary feedback controller stabilizes the uncertain system around the nominal optimal trajectory, and gPC predicts the stochastic evolution of the stabilized system. Chance constraints on the uncertain system implicitly tighten the constraints of the nominal OCP, which circumvents the problem of constraint tightening in the construction of classical tube-based MPC for nonlinear systems.

Simulation studies show the comparison of stochastic nonlinear trajectory optimisation, and the proposed formulation of robust stochastic nonlinear optimal control using the Dubin's car model with uncertain parameters.

Additionally, a numerical study demonstrates the properties of the polynomial chaos approach

applied to the sensitivity analysis of a nonlinear model of a single track dynamic model of a car with the Pacejka tire model. The focus is on the influence of the uncertain tire model parameters on the lateral dynamics of the car which is the most relevant for control applications.

8.1 Polynomial Chaos Expansion

In this work, we are concerned with parametric uncertainties in the plant model which lead to the following general stochastic optimal control problem:

$$\begin{aligned} &\min_{\mathrm{u}(t)} \ \Phi(\mathrm{x}(t_\mathrm{f},\xi)) + \int_{t_0}^{t_\mathrm{f}} L(\mathrm{x}(\tau,\xi),\mathrm{u}(\tau))\, d\tau. \\ &\text{s.t. } \dot{\mathrm{x}}(t,\xi) = \mathrm{f}(\mathrm{x}(t,\xi),\mathrm{u}(t),\xi) \\ &\Pr[g_i(\mathrm{x}(t,\xi),\mathrm{u}(t)) \le 0] \le \varepsilon_i \quad \forall i = 0 \dots n_g \\ &\mathrm{x}(t_0,\xi) = \mathrm{x}_0 \end{aligned} \tag{8.1}$$

where $\xi \in \mathbb{R}^{n_\xi}$ is a random variable defined on Ξ with probability density function $\rho(\xi)$; $t \in \mathbb{R}$ denotes time, $\mathrm{x}(\tau,\xi) \in \mathbb{R}^{n_x}$ is a stochastic process describing the evolution of the system state, $u \in \mathbb{R}^{n_u}$ is a vector of control inputs. The function $\Phi : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal cost function, or the Mayer term, and $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is called the running cost, or the Lagrange term. In a stochastic formulation, Lagrange and Mayer terms typically depend on the expected value $\mathrm{E}[\mathrm{x}(t,\xi)]$ and the variance $\mathrm{Var}[\mathrm{x}(t,\xi)]$. The uncertain system dynamics is given by the function $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_x}$, $g_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ are the path constraints, and finally, ε_i are the probabilities with which i-th constraint should be satisfied. In the following, we briefly explain how this problem can be efficiently transformed to a numerically tractable OCP that can be handled by the standard tools.

Generalized polynomial chaos (gPC) approximates random variables using polynomial basis functions $\Psi_\mathrm{i}(\xi) : \Xi \to \mathbb{R}$ such that the expansion $\tilde{p}(\xi) \in \mathbb{R}$ up to polynomial degree $\mathrm{N_{gPC}}$ of the random variable $p(\xi) \in \mathbb{R}$ is

$$p(\xi) \approx \tilde{p}(\xi) = \sum_{\mathrm{k}=0}^{\mathrm{N_{gPC}}} p_\mathrm{k} \Psi_\mathrm{k}(\xi). \tag{8.2}$$

The choice of basis functions is crucial for the quality of the approximation. For a gPC expansion, the key idea is to use polynomials that are orthogonal with respect to the probability density function (PDF) $\rho(\xi)$ of the random variable ξ. In the space of square-integrable

functions orthogonality is defined as

$$\langle \Psi_i, \Psi_j \rangle_{L^2(\rho)} := \int_\Xi \Psi_i \Psi_j \rho(\xi) d\xi = \gamma_i \delta_{ij} \tag{8.3}$$

where $\langle \cdot, \cdot \rangle_{L^2(\rho)}$ denotes the inner product, Ξ is the domain of ξ, δ_{ij} is the Kronecker delta, and γ_i is the normalization constant

$$\gamma_i = \langle \Psi_i, \Psi_i \rangle_{L^2(\rho)}. \tag{8.4}$$

Polynomials orthogonal with respect to any PDF $\rho(\xi)$ can be constructed by a three-term recurrence relation which defines a relation between Ψ_{k+1}, Ψ_k, and Ψ_{k-1} with $\Psi_0 = 1$ and $\Psi_{-1} = 0$ or by using the Gram-Schmidt orthogonalisation procedure [131][132]. Orthogonal polynomials for some commonly used distributions can be found for example in [133].

The orthogonality property can be used to determine the expansion coefficients p_k by the projection

$$p_k = \frac{1}{\gamma_k} \langle p(\xi), \Psi_k \rangle_{L^2(\rho)}, \tag{8.5}$$

which results in the truncation error $e(\xi) = p(\xi) - \tilde{p}(\xi)$ being orthogonal to the approximation basis:

$$\langle e(\xi), \Psi_i \rangle_{L^2(\rho)} = 0, \quad \text{for } i = 0, ..., N_{gPC}. \tag{8.6}$$

It can be shown [131] that the expected value and variance of the random variable $\tilde{p}(\xi)$ can be expressed in terms of approximation coefficients (8.5) as

$$\begin{aligned} \mathrm{E}[\tilde{p}(\xi)] &= p_0, \\ \mathrm{Var}[\tilde{p}(\xi)] &= \sum_{k=1}^{N_{gPC}} \gamma_k p_k^2. \end{aligned} \tag{8.7}$$

In the case of p being a stochastic process $p(t, \xi)$, performing the same operations leads to the approximation

$$p(t, \xi) \approx \tilde{p}(t, \xi) = \sum_{k=0}^{N_{gPC}} p_k(t) \Psi_k(\xi) \tag{8.8}$$

with the stochastic modes $p_k(t)$ and time dependent expected value and variance. Additionally, if $p \in \mathbb{R}^{n_p}$, the coefficients are also in $\mathbb{R}^{n_p}$. $\mathrm{Var}[p] \in \mathbb{R}^{n_p}$ then denotes elementwise variance of p.

Galerkin Projection

is a numerical procedure that allows one to determine the evolution of the stochastic process $\mathrm{x}(t,\xi)$ driven by a controlled stochastic ordinary differential equation (SODE):

$$\dot{\mathrm{x}}(t,\xi) = \mathrm{f}(\mathrm{x}(t,\xi),\mathrm{u}(t),\xi) \tag{8.9}$$

As before, the process $\mathrm{x}(t,\xi)$ is approximated by a corresponding polynomial basis:

$$\mathrm{x}(t,\xi) \approx \tilde{\mathrm{x}}(t,\xi) = \sum_{\mathrm{i}=0}^{\mathrm{N_{gPC}}} \mathrm{x_i}(t)\Psi_\mathrm{i}(\xi) \tag{8.10}$$

Galerkin projection determines a governing ODE for the stochastic modes $\mathrm{x_k}(t)$ by a weak formulation of the stochastic ODE from Equation 8.9 which needs needs to be rewritten in the residual form

$$\dot{\tilde{\mathrm{x}}}(t,\xi) - \mathrm{f}(\tilde{\mathrm{x}}(t,\xi),\mathrm{u}(t),\xi) = \mathrm{R}\left(\dot{\tilde{\mathrm{x}}}(t,\xi),\tilde{\mathrm{x}}(t,\xi),\mathrm{u}(t),\xi\right). \tag{8.11}$$

In order to eliminate ξ and the residual, the Galerkin approach then demands the projection of the residual onto the gPC expansion's basis function Ψ_j be zero:

$$\langle \mathrm{R}(\cdot),\Psi_\mathrm{j}\rangle_{L^2(\rho)} = 0, \quad \text{for } \mathrm{j} = 0,\ldots,\mathrm{N_{gPC}}. \tag{8.12}$$

For each j the projection of the derivative part of the residual $\dot{\tilde{\mathrm{x}}}(t,\xi)$ reduces to

$$\langle \dot{\tilde{\mathrm{x}}}(t,\xi),\Psi_\mathrm{j}\rangle_{L^2(\rho)} = \gamma_\mathrm{j}\dot{\mathrm{x}}_\mathrm{j}(t) \tag{8.13}$$

due to the orthogonality and the separation of t and ξ in the gPC expansion. The right hand side of the SODE, $\mathrm{f}(\tilde{\mathrm{x}}(t,\xi),\mathrm{u}(t),\xi)$, is less trivial for general nonlinear systems and the projection integral

$$\langle \mathrm{f}(\cdot),\Psi_\mathrm{j}\rangle_{L^2(\rho)} = \int_\Xi \mathrm{f}\left(\sum_{\mathrm{i}=0}^{\mathrm{N_{gPC}}} \mathrm{x_i}(t)\Psi_\mathrm{i}(\xi),\mathrm{u}(t),\xi\right)\Psi_\mathrm{j}\rho(\xi)d\xi \tag{8.14}$$

needs to be calculated numerically, e.g., by Gauss quadrature with a sufficient number of integration nodes. By defining the expanded state X as

$$\mathrm{X} = \begin{bmatrix} \mathrm{x_0} \\ \mathrm{x_1} \\ \vdots \\ \mathrm{x_{N_{gPC}}} \end{bmatrix}, \tag{8.15}$$

where $\mathrm{x_i}$ are the modes of the spectral expansion defined in (8.10) and the j-th projection of

the right hand side as

$$\mathrm{f_j} = \langle \mathrm{f}(\cdot), \Psi_\mathrm{j} \rangle_{L^2(\rho)} = \mathrm{f_j}(\mathrm{X}(t), \mathrm{u}(t)), \tag{8.16}$$

and the expanded dynamics F by

$$\mathrm{F} = \begin{bmatrix} \mathrm{f}_0 \\ \mathrm{f}_1 \\ \vdots \\ \mathrm{f}_{\mathrm{N_{gPC}}} \end{bmatrix} \tag{8.17}$$

one can compactly write a deterministic ODE for the stochastic modes or expanded states, respectively,

$$\dot{\mathrm{X}}(t) = \mathrm{F}(\mathrm{X}(t), \mathrm{u}(t)). \tag{8.18}$$

This has the same form as the dynamics of the deterministic OCP and can be treated using standard numerical integration methods.

8.2 Distributionally Robust Constraints

The transformation of chance constraints into a deterministic formulation is another challenging step. Evaluating probabilities requires integration of the PDF of the term $\mathrm{g}(\mathrm{x}(t,\xi),\mathrm{u}(t))$ in chance constraints as introduced in (8.1), which becomes elaborate for several reasons. Assuming that the gPC approximation of $\mathrm{x}(t,\xi)$ is known, determining its PDF is still not straightforward, especially if $\mathrm{x}(t,\xi)$ is not globally invertible. The same holds for g if it is a nonlinear function in x. Even if the PDF is known, evaluating joint chance constraints that arise from multivariate ξ and the resulting multivariate integration becomes computationally demanding. Popular methods to solve these integrals are Monte Carlo methods which are based on function evaluations with samples drawn from the underlying distribution [131]. The scenario approach is another sampling-based method to handle chance constraints. Analytic approaches on the other hand render chance constraints deterministic by using deterministic bounds, which ensure constraint satisfaction with higher or equal probability than necessary. A good overview over several analytic and sampling based methods to handle constraints can be found in [134]. Despite their conservatism analytic methods are ideal for optimal control due to their lower calculation costs. If the shape of the constraint PDF is not known in advance, so called distributionally robust chance constraints can be used. These methods define probabilistic bounds that are valid for any PDF [120]. One such bound is the Chebyshev inequality, which only depends on the expected value $\mu = \mathrm{E}[y]$ and variance $\sigma^2 = \mathrm{Var}[y]$ of the

random variable y: [135]

$$Pr[|y-\mu| > t] \leq \frac{\sigma^2}{t^2}. \tag{8.19}$$

In other words, it states a bound on the probability of y being in the set $y > \mu + t \cup y < \mu - t$, i.e., the two tails of the domain of y centered around μ. In the chance constraint from (8.1), the probability of $y < 0$ needs to be bounded, consequently t has to be set to μ. One can write

$$Pr[y < 0] \leq Pr[|y-\mu| > \mu] \leq \frac{\sigma^2}{\mu^2} \leq \varepsilon \tag{8.20}$$

such that it is sufficient to demand, under consideration that $\mu > 0$ has to hold to satisfy the inequality,

$$\mu^2 - \frac{1}{\varepsilon}\sigma^2 \geq 0. \tag{8.21}$$

In addition to the conservatism introduced by distributional robustness, the Chebyshev inequality bounds the probability of y being in one of the two tails of the domain, whereas only one tail is needed in the chance constraint. Calafiore [136] suggests another inequality for one tail, which becomes after some simplification

$$\mu - \sqrt{\frac{1-\varepsilon}{\varepsilon}}\sigma \geq 0 \implies Pr[y < 0] \leq \varepsilon. \tag{8.22}$$

which gives a better bound for small ε and significantly better bounds for ε closer to 1. To avoid calculating σ as a square root of the variance and again with $\mu > 0$, one can write

$$\mu^2 - \frac{1-\varepsilon}{\varepsilon}\sigma^2 \geq 0, \tag{8.23}$$

Both bounds are applicable to any probability distribution where the mean and variance are defined. The downside of this generality is that the provided bounds usually are not sharp. An example comparing both bounds for a time slice of a random process will be provided in the Section 8.5. The expected value and variance required for these bounds can be efficiently calculated from the gPC expansions as introduced in the previous section. Projection of a general nonlinear constraint onto the gPC basis results in the expansion coefficients being nonlinear functions of all state coefficients X and the input u,

$$\mathrm{g}(\mathrm{x}(t,\xi),\mathrm{u}(t)) \approx \tilde{\mathrm{g}}(\mathrm{x}(t,\xi),\mathrm{u}(t)) = \sum_{\mathrm{k}=0}^{\mathrm{N_{gPC}}} \mathrm{g_k}(\mathrm{X}(t),\mathrm{u}(t))\Psi_\mathrm{k}(\xi). \tag{8.24}$$

If the constraints are affine in $\mathrm{x}(t,\xi)$, i.e., $\mathrm{g}(\mathrm{x}(t,\xi),\mathrm{u}(t))$ has the form $\mathrm{a}(\mathrm{u}(t))^\mathrm{T}\mathrm{x}(t,\xi) + b(\mathrm{u}(t))$, its

gPC coefficients are linear combinations of the state coefficients,

$$g_{\mathrm{lin,k}}(x_k(t), u(t)) = a(u(t))^T x_k(t) + \delta_{k0} b(u(t)), \tag{8.25}$$

where the first coefficient and thus the expected value is shifted by b.

Putting expected value and variance of the constraint's gPC expansion and the Inequality (8.23) together results in a deterministic nonlinear inequality constraint on the expanded state X and input u.

Discussion

For unstable systems state trajectories of different uncertainty realizations may diverge quickly, which leads to a growing state variance, although a feedback controller reacting on the as unknown disturbances could reduce the variance. Consequently, an open-loop stochastic solution tends to overestimate the variance and thus becomes conservative.

It is important to note that, similar to the spectral methods for partial differential equations (PDE), the gPC expansion can suffer from the so called curse of dimensionality, i.e. the computational complexity grows exponentially with the number of independent random variables in the SODE. One solution to reduce the computational complexity is to use sparse grids [131][137]. Related, one of the assumptions necessary for the efficient use of gPC is a restriction to time invariant uncertainties, as uncertainties that are uncorrelated in time, e.g., white noise, would lead to an infinite number of random variables. Approaches exist to incorporate noise at the cost of a higher computational effort [138, 139, 140].

The quality of the gPC expansion depends on the number of basis functions and the regularity of the states in the random space. If the state distribution differs significantly from the distribution of the random variable ξ, which might be the case for long prediction horizons, the quality of the gPC expansion might deteriorate over time. Time dependent gPC therefore adapts the choice of basis functions over time [141, 142].

8.3 Stochastic Optimal Control with Prestabilising Controller

To overcome the conservatism linked to a rapidly growing variance of stochastic predictions, a novel stochastic optimal control formulation is proposed. The formulation combines the idea of a tube-based approach with the capability of gPC to efficiently solve stochastic ODEs: an ancillary feedback controller stabilizes the uncertain system represented by the spectral modes of the gPC expansion around the nominal optimal trajectory. Chance constraints on the uncertain system implicitly tighten the constraints of the nominal OCP, which circumvents

the problem of constraint tightening in the construction of classical tube-based MPC for nonlinear systems. As the performance of the ancillary controller is predicted by gPC, failures are avoided by falling back into the conservative mode of stochastic optimal control. This means that the chance constraints will still be satisfied even if the ancillary controller fails to stabilise the system. To the best of the authors knowledge, such approach to the variance control has not been explored in the literature before.

Algorithm The proposed stochastic OCP considers both the nominal and stochastic system: the nominal dynamics $\hat{\mathrm{f}}: \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_u}} \to \mathbb{R}^{\mathrm{n_x}}$ mapping the nominal control $\hat{\mathrm{u}} \in \mathbb{R}^{\mathrm{n_u}}$ and nominal state $\hat{\mathrm{x}} \in \mathbb{R}^{\mathrm{n_x}}$ to the nominal state derivatives, the uncertain dynamics $\mathrm{f}: \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_u}} \times \mathbb{R}^{\mathrm{n_\xi}} \to \mathbb{R}^{\mathrm{n_x}}$ mapping control $\mathrm{u} \in \mathbb{R}^{\mathrm{n_u}}$ and state $\mathrm{x} \in \mathbb{R}^{\mathrm{n_x}}$ to the state derivatives in dependence of the uncertainty $\xi \in \mathbb{R}^{\mathrm{n_\xi}}$, the cost $J: \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_u}} \to \mathbb{R}$ on nominal states and inputs, and inequality chance constraints as introduced in (8.1) on state x and input u. An ancillary controller $\boldsymbol{\mu}: \mathbb{R}^{\mathrm{n_x}} \times \mathbb{R}^{\mathrm{n_{pc}}} \to \mathbb{R}^{\mathrm{n_u}}$ parametrized by $\mathrm{p_c} \in \mathbb{R}^{\mathrm{n_{pc}}}$ limits the deviation of x from the reference $\hat{\mathrm{x}}$ to stabilize the uncertain system around the nominal trajectory. The predicted control applied to the uncertain system is consequently

$$\mathrm{u}(t,\xi;\mathrm{p_c}(t)) = \hat{\mathrm{u}}(t) + \boldsymbol{\mu}\left(\Delta\mathrm{x}(t,\xi);\mathrm{p_c}(t)\right), \tag{8.26}$$

with the uncertain deviation from the reference

$$\Delta\mathrm{x}(t,\xi) = \mathrm{x}(t,\xi) - \hat{\mathrm{x}}(t). \tag{8.27}$$

As the ancillary controller saturates, a lower and upper bound possibly depending on the controller parameters can be given by

$$|\boldsymbol{\mu}\left(\Delta\mathrm{x}(t,\xi);\mathrm{p_c}(t)\right)| \leq \boldsymbol{\mu}_{\max}(\mathrm{p_c}(t)) \tag{8.28}$$

where $\boldsymbol{\mu}_{\max}$ controls the level of authority of the ancillary controller. The OCP then becomes

$$\begin{aligned}
\min_{\mathrm{u}(t),\mathrm{p_c}(t)} \quad & J[\hat{\mathrm{x}}(t),\hat{\mathrm{u}}(t),\mathrm{Var}[\mathrm{x}(t,\xi)]] \\
\text{s.t. } & \dot{\hat{\mathrm{x}}}(t) - \hat{\mathrm{f}}(\hat{\mathrm{x}}(t),\hat{\mathrm{u}}(t)) = 0, \\
& \hat{\mathrm{x}}(t_0) = \mathrm{E}[\mathrm{x}(t_0,\xi)], \\
& \dot{\mathrm{x}}(t,\xi) - \mathrm{f}(\mathrm{x}(t,\xi),\hat{\mathrm{u}}(t) + \boldsymbol{\mu}\left(\Delta\mathrm{x}(t,\xi);\mathrm{p_c}(t)\right),\xi) = 0, \\
& \mathrm{x}(t_0,\xi) = \mathrm{x}_0, \\
& \Pr[g_i(\mathrm{x}(t,\xi),\mathrm{u}(t)) \leq 0] \leq \varepsilon_i \quad \forall i = 0 \ldots n_g \\
& \mathrm{u_{min}} + \boldsymbol{\mu}_{\max}(\mathrm{p_c}(t)) \leq \hat{\mathrm{u}}(t) \leq \mathrm{u_{max}} - \boldsymbol{\mu}_{\max}(\mathrm{p_c}(t)), \\
& \mathrm{g_c}(\mathrm{p_c}(t)) \leq 0,
\end{aligned} \tag{8.29}$$

where g_i denotes all inequality constraints on states and inputs other than the inputs constraint and $\mathrm{g_c}$ denotes inequality constraints on the ancillary controller parameters.

The proposed methodology can be applied not only for robust trajectory planning for uncertain systems but also in receding horizon fashion for model predictive control. In the MPC scenario, until the new solution of the OCP (8.29) is available MPC applies the sum of the nominal and the ancillary controller:

$$\mathrm{u}(t) = \hat{\mathrm{u}}(t) + \boldsymbol{\mu}\left(\mathrm{x_{meas}}(t) - \hat{\mathrm{x}}(t); \mathrm{p_c}(t)\right). \tag{8.30}$$

Discussion

All parts of the OCP can be treated with the methods presented in the previous sections. In our work, the stochastic dynamics and the chance constraints are transformed to a standard OCP formulation by the Galerkin projection method and distributionally robust chance constraint as outlined in Sections (8.1, 8.2). The resulting deterministic OCP can then be transformed into an optimization problem by pseudospectral collocation using the *PolyMPC* toolbox.

Special attention should be paid to the structure of the OCP: the ancillary controller and its parameters $\mathrm{p_c}(t)$ shape the dynamics of the uncertain system and can thus reduce the state variance if chosen accordingly. The state variance influences the chance constraint g_i and a smaller variance allows the expected value to be closer to the deterministic inequality constraints. If the nominal dynamics are chosen to be close to the dynamics of the expected states of the uncertain system, this leads to less tightened constraints for the nominal states and a cost reduction.

To avoid the constraints being the only coupling mechanism of the uncertain system and the nominal systems, and thus the only influence of the controller parameters, a cost on the state variance can be introduced. In the case of failing to stabilize the uncertain dynamics, the nominal solution can adapt to prevent the state variance to grow, for instance, by slowing down the vehicle as will be shown in the simulation studies.

8.4 Optimal Path Planning

Vehicle Modelling

The kinematic bicycle, or Dubin's car, model has three states $[x, y, \psi]$, where x and y are the position of the lumped rear wheel and ψ is the heading angle of the vehicle. By using the steering angle δ and the velocity v as inputs, the states can be controlled and follow the

dynamics derived by simple kinematic relations

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos(\psi) \\ v\sin(\psi) \\ \frac{v}{l}\tan(\delta) \end{bmatrix} \tag{8.31}$$

with wheelbase l, i.e., the distance between front and rear axis, as the only parameter of the system. For small angles, the steering angle and curvature of the driven path are proportional and the resulting relation is the Ackermann steering angle for curvature κ [143]:

$$\delta_{\text{acker}} = \kappa \cdot l. \tag{8.32}$$

The dynamic bicycle is the same as in Section 7.1 .

Path Following

In this section, we consider the problem of optimising a vehicle trajectory given a geometric representation of the center line of the road and its boundaries. It is common for modern vision-based road recognition systems to compute the road boundaries as parametrised curves, for example splines. The goal of the optimisation algorithm is to find a trajectory within the road that respects dynamic and actuation constraints of the vehicle. To achieve this we adopt the optimisation-based path following methodology from [96][97].

Consider the system output $\mathrm{y} \in \mathbb{R}^{\mathrm{n_y}}$, $\mathrm{y} = \mathrm{h(x)}$, and a reference path to follow $\mathrm{x}_p : \mathbb{R} \to \mathbb{R}^{\mathrm{n_y}}$ parametrized by a path parameter $\theta \in \mathbb{R}$. The goal then is to find the control signal $u(\cdot)$ that steers the system to this path. That is, for some sequence $\theta(t),\ t \in [t_0, t_f]$ corresponding to a sequence of points on the path $p(\theta(t))$, one seeks to find a $u(t)$ to minimize the distance between the path and the system:

$$u^\star(t) = \underset{u(t)}{\arg\min} \int_{t_0}^{t_f} \|\mathrm{x}_p(\theta(t)) - h(x(t))\|\, dt \tag{8.33}$$

A standard approach to pick an optimal sequence $\theta(t)$ is to assign dynamics to the parameter:

$$\dot{\theta}(t) = f_\theta(\theta(t), v(t)) \tag{8.34}$$

In the particular implementation here the choice of $f_\theta(\theta(t), v(t))$ is a second order linear system which is often used for mechatronic systems as it allows to control speed and acceleration profiles: [96]

$$\dot{\mathrm{z}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathrm{z} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v = A\mathrm{z} + Bv \tag{8.35}$$

where z consists of the path parameter and its derivative

$$\mathrm{z} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}. \tag{8.36}$$

The path following problem then becomes:

$$\begin{aligned} \min_{\mathrm{u}(t),\mathrm{v}(t)} \; & \Phi\left(\mathrm{x}(t_\mathrm{f}), \mathrm{z}(t_\mathrm{f})\right) + \int_{t_0}^{t_\mathrm{f}} L\left(\mathrm{x}(\tau), \mathrm{z}(\tau), \mathrm{u}(\tau)\right) d\tau. \\ \text{s.t. } & \dot{\mathrm{x}}(t) = \mathrm{f}(\mathrm{x}(t), \mathrm{u}(t)) \\ & \dot{\mathrm{z}}(t) = A\mathrm{z}(t) + B\mathrm{v}(t) \\ & g(\mathrm{x}(t), \mathrm{z}(t), \mathrm{u}(t)) \geq 0 \\ & \mathrm{u}(t) \in \mathscr{U},\ \mathrm{v}(t) \in \mathscr{V} \\ & \mathrm{x}(t_0, \xi) = \mathrm{x}_0 \end{aligned} \tag{8.37}$$

Where f is the vehicle dynamic equations, $\mathscr{U}$ and $\mathscr{V}$ are box constraints on the vehicle and virtual inputs. The associated Lagrange term cost penalises the geometric path deviation, the input the reference velocity deviations. The Mayer term can also be complemented by a cost on the reference velocity:

$$\begin{aligned} L[\mathrm{x}(t), \mathrm{z}(t), \mathrm{u}(t)] &= L_\mathrm{path} + L_\mathrm{input} + L_\mathrm{velocity} \\ &= \|\mathrm{h}(\mathrm{x}(t)) - \mathrm{x}_p(\theta(t))\|_\mathrm{Q} + \|\mathrm{u}(t)\|_\mathrm{R} + \|\dot{\theta}(t) - \dot{\theta}_\mathrm{ref}(\theta(t))\|_W, \\ \Phi[\mathrm{x}(t_\mathrm{f})] &= \|\mathrm{h}(\mathrm{x}(t_f)) - \mathrm{x}_p(\theta(t_f))\|_\mathrm{Q} + \|\mathrm{x}(t_\mathrm{f}) - \mathrm{x}_\mathrm{ref}(t_\mathrm{f})\|_{\mathrm{Q_f}} + \|\mathrm{z}(t_\mathrm{f}) - \mathrm{z}_\mathrm{ref}(t_\mathrm{f})\|_{W_\mathrm{f}} \end{aligned} \tag{8.38}$$

Road Boundaries

In our implementation, the geometric curves representing the reference path $\mathrm{x}_\mathrm{p}(\theta_\mathrm{p})$ and the parametric curves describing the right and left boundaries of the road $\mathrm{x}_{\mathrm{b,r}}(\theta_\mathrm{p})$ and $\mathrm{x}_{\mathrm{b,l}}(\theta_\mathrm{p})$ are synchronised. Synchronisation means that the curves are parametrised such that for any path parameter θ_p^* in the domain, the points $\mathrm{x}_{\mathrm{b,r}}(\theta_\mathrm{p}^*)$ and $\mathrm{x}_{\mathrm{b,l}}(\theta_\mathrm{p}^*)$ are the closest possible to $\mathrm{x}_\mathrm{p}(\theta_\mathrm{p}^*)$ in the sense of the second norm. The condition of being on the inner side of the road boundary is replaced by the condition to be on the inner side of its tangent, which reduces to a simple dot product when synchronised curves are used:

$$g_{\mathrm{b,i}} = \left(\mathrm{x}_\mathrm{p}(\theta) - \mathrm{x}_{\mathrm{b,i}}(\theta)\right)^\mathrm{T} \left(\mathrm{x}_\mathrm{cog} - \mathrm{x}_{\mathrm{b,i}}(\theta)\right) \geq 0, \tag{8.39}$$

where x_cog are the coordinates of the vehicle's center of gravity (CoG) as depicted in Figure 8.1 and $\mathrm{i} \in \{\mathrm{r}, \mathrm{l}\}$ is the right or left road boundary. In order to account for the vehicle's width and orientation, the road boundary constraints are tightened by half the width of the vehicle.

An advantage of such a constraint formulation is its linearity, which is helpful when computing the impact of uncertainty: as described in Section 8.2, note that the path parameter θ is not subject to uncertainty but can be seen as an internal state of the controller. Thus, Equation (8.39) is linear in the uncertain states and chance constraints can be formulated in a simple manner.

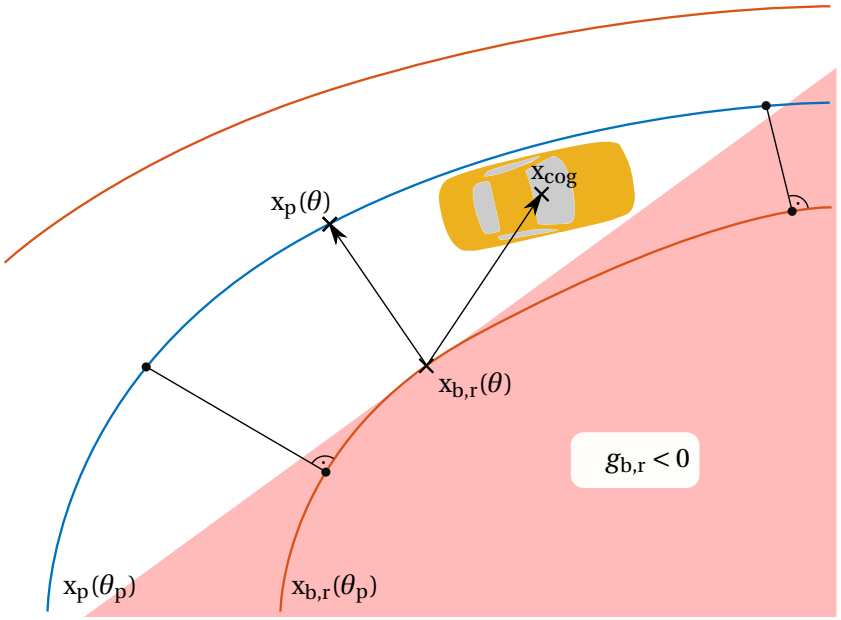


Figure 8.1 – Tangential approximation of the right road boundary, so that x_{cog} cannot enter the red region where $g_{b,r} < 0$.

Stochastic Cost

A common choice to account for uncertainty in the cost is to consider the expected value and variance of the state [119] [144]. Lagrange and Mayer terms in (8.38) then become

$$\begin{aligned} L[x(t,\xi),z(t),u(t)] &= \|h(E[x(t,\xi)]) - x_p(\theta(t))\|_Q + \|u(t)\|_R + \|\dot{\theta}(t) - \dot{\theta}_{ref}(\theta(t))\|_W + q_{Var}^T \mathrm{Var}[x(t,\xi)], \\ \Phi[x(t_f,\xi),z(t_f)] &= \|h(E[x(t_f,\xi)]) - x_p(\theta(t_f))\|_{Q_f} + \|z(t_f) - z_{ref}(\theta(t_f))\|_{W_f}. \end{aligned} \tag{8.40}$$

where $\mathrm{Var}[\cdot]$ denotes the diagonal entries of the covariance matrix.

8.5 Trajectory Optimisation under Parametric Uncertainties

In addition to the time dimension of states and controls, the dimension of the uncertainty has to be examined for stochastic MPC. The following sections show and interpret results from a sensitivity analysis of the dynamic bicycle, the solutions of the stochastic OCPs with the kinematic bicycle, and the effect of an ancillary controller in an OCP of the robust stochastic OCP scheme.

On Uncertainty Visualisation

This section briefly explains the generation of PDFs and the visualization of the road boundaries as chance constraints.

We start with the visualisation of the stochastic process governed by a SODE and chance constraints as described in Section 8.1. For a better understanding of results a good visualization can be helpful, e.g., for the construction of a PDF of a random variable. Following [131], the PDF of a random variable $p(\xi)$ is the push-forward of the PDF of ξ under the map p. If $p(\xi)$ is invertible, one can write

$$\rho_\xi(\xi(p^*))|\mathrm{d}\xi| = \rho_p(p^*)|\mathrm{d}p| \tag{8.41}$$

with ρ_ξ and ρ_p denoting the PDFs of ξ and p, respectively, i.e., the probability of an element in ξ is assigned to the probability of the corresponding element in p. It follows that

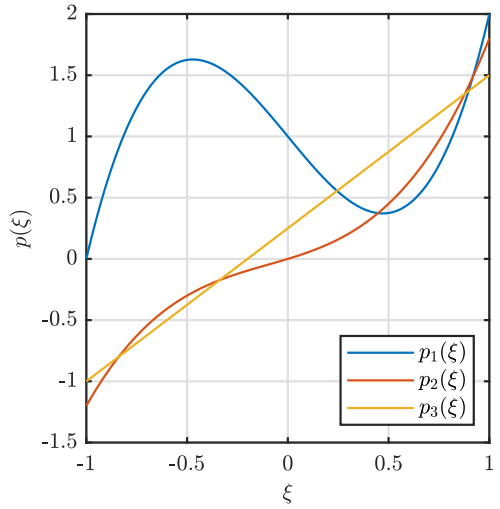
$$\rho_p(p^*) = \left|\left(\frac{\mathrm{d}p}{\mathrm{d}\xi}\right)^{-1}\right| \rho_\xi(\xi(p^*)), \tag{8.42}$$

i.e., the PDF is scaled by the inverse of the slope of $p(\xi)$. Consequently, the PDF can reach infinity if the slope becomes zero. If $p(\xi)$ is not invertible (8.42) is not correct, but the right hand side can be adapted:

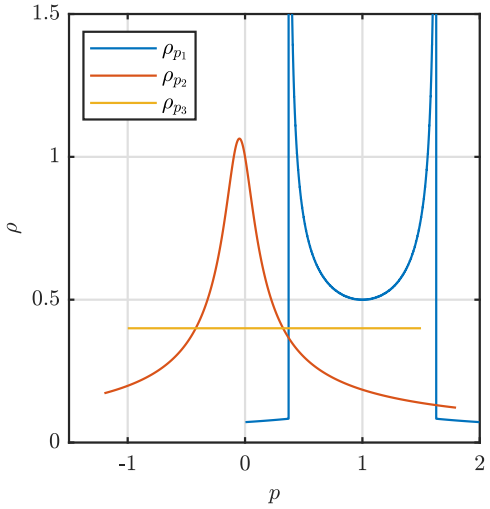
$$\rho_p(p^*) = \sum_{i=1}^{N} \left|\left(\frac{\mathrm{d}p}{\mathrm{d}\xi}\right)^{-1}\right| \rho_\xi(\xi_{i,p^*}), \tag{8.43}$$

Figure (8.2) shows the push forward of a uniform distribution through some simple polynomials as an example.

An interesting random variable to analyze is the chance constraint of a road boundary violation given the vehicle's current position $\mathrm{x}(\xi)$, path and boundary vectors $\mathrm{x_p}$ and $\mathrm{x_b}$. One way to visualise is to calculate the tightest boundary parallel to the original boundary that does not



(a) Polynomials $p_i(\xi)$.



(b) PDFs corresponding to the push-forwards through $p_i(\xi)$.

Figure 8.2 – Push-forward of a uniform distribution through different polynomials $p_i(\xi)$.

violate the chance constraints. Equation 8.39 can be reformulated as

$$\Pr[\gamma \mathrm{n}^{\mathrm{T}} \mathrm{x}_{cog}(\xi) - \gamma \mathrm{n}^{\mathrm{T}} \left(\mathrm{x}_{\mathrm{p}} - \gamma \mathrm{n}\right) \geq 0] \geq \varepsilon \tag{8.44}$$

where $\gamma = \|\mathrm{x}_{\mathrm{p}} - \mathrm{x}_{\mathrm{b}}\|_2$ is the distance from boundary to the path and $\mathrm{n} = \frac{\mathrm{x}_{\mathrm{p}} - \mathrm{x}_{\mathrm{b}}}{\gamma}$ is the normalized vector from boundary to path. The tightest bound of a chance constraint for probability of violation ε- γ_ε can be found by applying one of the distributionally robust chance constraints from Section 8.2. Equation 8.44 can be reformulated to

$$\Pr[\mathrm{n}^{\mathrm{T}} \mathrm{x}_{cog}(\xi) - \mathrm{n}^{\mathrm{T}} \mathrm{x}_{\mathrm{p}} - \gamma_\varepsilon = g_{\gamma_\varepsilon}(\xi) \geq 0] \geq \varepsilon \tag{8.45}$$

and using the chance constraint from Equation (8.23) leads to

$$\mathrm{E}[g_{\gamma_\varepsilon}(\xi)] - \sqrt{\frac{1-\varepsilon}{\varepsilon} \mathrm{Var}[g_{\gamma_\varepsilon}(\xi)]} \geq 0. \tag{8.46}$$

Calculation of the expected value and variance is then straightforward if the gPC expansion coefficients x_i of $\mathrm{x}(\xi)$ are known. As stated in Equation (8.25), orthogonality of the polynomials implies

$$\begin{aligned} g_{\gamma_\varepsilon,0} &= -\mathrm{n}^{\mathrm{T}} \mathrm{x}_{cog,0} + \mathrm{n}^{\mathrm{T}} \mathrm{x}_{\mathrm{p}} + \gamma_\varepsilon, \\ g_{\gamma_\varepsilon,i} &= -\mathrm{n}^{\mathrm{T}} \mathrm{x}_{cog,\mathrm{i}}, \end{aligned} \tag{8.47}$$

and the expected value and variance can be calculated according to (8.7). Using $\sigma_{\gamma_\varepsilon} =$

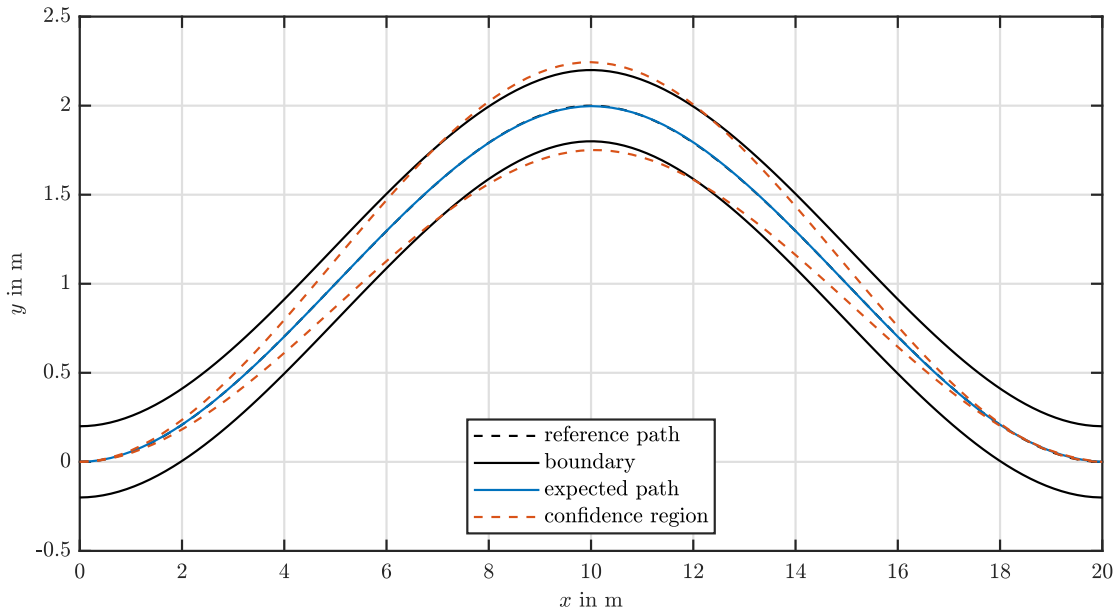
$\sqrt{\mathrm{Var}[g_{\gamma_\varepsilon}(\xi)]}$, the estimate of boundary becomes

$$\gamma_\varepsilon = \sqrt{\frac{1-\varepsilon}{\varepsilon}}\sigma_{\gamma_\varepsilon} + \mathrm{n}^{\mathrm{T}}\left(\mathrm{x}_{\mathrm{cog},0} - \mathrm{x}_{\mathrm{p}}\right). \tag{8.48}$$

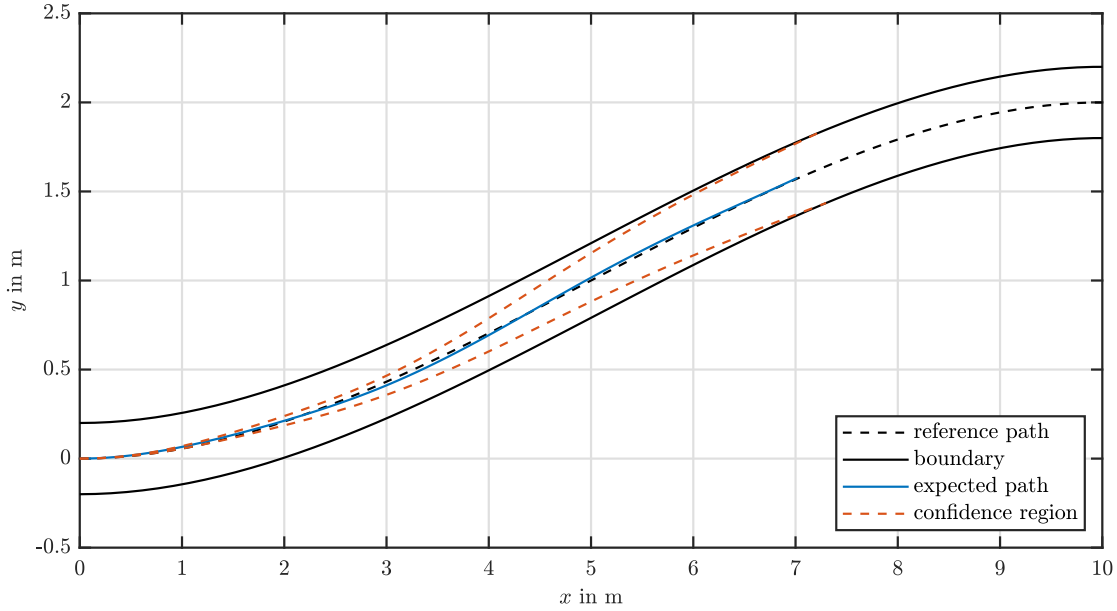
Optimal Control Problem

The kinematic bicycle introduced above is used to demonstrate the capability of gPC expansions of a stochastic OCP. In the test case, the vehicle has an uncertain wheelbase between $0.95 \cdot L_{\mathrm{nominal}}$ and $1.05 \cdot L_{\mathrm{nominal}}$ modelled by a uniform distribution that scales the influence of the steering angle on the driven curvature. The OCP is parametrized with path-, velocity- and regularization cost as listed in Table 8.2b, the gPC expansion has three basis functions, and the vehicle is supposed to follow a cosine with wavelength 20 [m] along the x-axis with an amplitude of 1 [m] in the y-direction, a constant longitudinal velocity of 10 [m/s], and boundaries with a distance of 0.2 [m] from the center line of the path.

Figure 8.3a shows the optimal solution for a horizon length of 2 [s] not enforcing boundary chance constraints. The expected path of the vehicle is shown with the solid blue line, and the dashed red lines denote the boundaries of a 95% confidence region, i.e. the region where all possible realisations of the vehicle trajectories will lie with the 95% probability. It can be observed that these boundaries cross the road boundaries at around 7[m], and therefore, the vehicle will leave the road with high probability from some realisations of the uncertain parameter if the control was applied in open-loop. In fact, due to the kinematic relations in the model, it is not possible to reduce the size of the tube, as any steering increases the variance. Figure 8.3b shows solution to the same stochastic problem but the chance constrained enforced. The vehicle has to considerably slow down so that the chance constraints are satisfied. This conservatism is avoidable in reality, however, if one considers the feedback correction that will keep the vehicle closer to the center line. In the following, we demonstrate how the method introduced in Section 8.3 allows to avoid the conservatism of the open-loop predictions while guaranteeing chance constraint satisfaction.



(a) OCP without consideration of the road boundaries as chance constraints. The red dash lines define the confidence tube, i.e. the tube that contains all possible realisations of the vehicle trajectory with probability 95%. The tube crosses the road boundaries which will lead to constraint violation for some realisations of the uncertain parameter.



(b) OCP with consideration of the road boundaries as chance constraints. Due to the kinematic relations in the model, the vehicle cannot influence the effect of uncertainty while proceeding in x-direction, and therefore has to reduce the speed.

Figure 8.3 – Path solution of the stochastic OCP for the kinematic bicycle with uncertain wheelbase following a cosine with constant velocity.

	q_x	q_y	w	$q_{x,f}$	$q_{y,f}$	$r_{\dot{\delta}}$	$r_{\dot{v}}$	$r_{\dot{\nu}}$
unit	$1/(\mathrm{m}^2\cdot\mathrm{s})$	$1/(\mathrm{m}^2\cdot\mathrm{s})$	s/m^2	$1/\mathrm{m}^2$	$1/\mathrm{m}^2$	s	$\mathrm{s}^3/\mathrm{m}^2$	$\mathrm{s}^5/\mathrm{m}^2$
value	1e2	1e2	1e2	1e2	1e2	1e0	1e0	1e0

(a) Parameters for MPC with the kinematic bicycle.

	m	J	l_f	l_r	C_D	C_R				
unit	kg	kgm^2	m	m	kg/m	N				
value	1200	1400	1.6	1.4	1	600				
	B_f	C_f	D_f	E_f	$C_{\mathrm{lin,f}}$	B_r	C_r	D_r	E_r	$C_{\mathrm{lin,r}}$
unit	1/rad				N/rad	1/rad				N/rad
value	10.3	2.4	1.3	1	1.77e5	10	2.7	1.3	1	2.20e5

(b) Model parameters for the dynamic bicycle.

	r_δ	$r_{F_{\mathrm{x,f}}}$	$r_{F_{\mathrm{x,r}}}$	r_ν	$r_{\dot{\delta}}$	$r_{\dot{F}_{\mathrm{x,f}}}$	$r_{\dot{F}_{\mathrm{x,r}}}$	$r_{\dot{\nu}}$
unit	1/s	$1/(\mathrm{s}\cdot\mathrm{N}^2)$	$1/(\mathrm{s}\cdot\mathrm{N}^2)$	$\mathrm{s}^3/\mathrm{m}^4$	s	s/N^2	s/N^2	$\mathrm{s}^5/\mathrm{m}^2$
value	1e3	1e−6	1e−6	1e−1	1e4	1e−4	1e−4	1e−1

(c) Regularization cost parameters for MPC on the dynamic bicycle.

setup		q_x	q_y	w	$q_{x,f}$	$q_{y,f}$	w_f	$q_{\psi,f}$
	unit	$1/(\mathrm{m}^2\cdot\mathrm{s})$	$1/(\mathrm{m}^2\cdot\mathrm{s})$	s/m^2	$1/\mathrm{m}^2$	$1/\mathrm{m}^2$	$\mathrm{s}^2/\mathrm{m}^2$	
1	value	1e5	1e5	1e6	1e5	1e5		
2	value	1e5	1e5	1e3	1e5	1e5	1e3	
3	value	1e5	3e2	1e3	1e5	3e2	1e3	1e5
4	value	1e5	1e4	1e3	1e5	1e4	1e3	1e4

(d) Velocity- and path cost parameters for MPC on the dynamic bicycle.

Table 8.1 – Model- and MPC parameters for the kinematic- and dynamic bicycle.

Stochastic Optimal Control with a Prestabilising Controller

The major drawback of the open-loop predictions in stochastic OCP is the conservatism of the prediction as illustrated in the previous section. Stochastic model predictive control proposed in Section 8.3 circumvents this problem by applying an ancillary controller in a tube-based MPC fashion. To test this assertion the same parameters and tuning as in the stochastic OCP are used in the proposed robust formulation. Note that the OCP parameters listed in Table 8.2b have a slightly different meaning, as q_x, q_y, $q_{x,f}$, and $q_{y,f}$ now parametrize the cost on the nominal system of the robust stochastic OCP in contrast to the cost on the expected values in the stochastic OCP. The ancillary feedback controller applies an additional steering command that depends on the lateral path deviation d_lat and the orientation error d_or to reduce the

setup	parameters	minimal factor	maximal factor
1	D_f, D_r	0.95	1.05
2	B_f, C_f, B_r, C_r	$\sqrt{0.95}$	$\sqrt{1.05}$
3	B_r, C_r	$\sqrt{0.90}$	1.0

(a) Parameter variations for sensitivity analysis. The dependency of parameter p on ξ is given by $p(\xi) = 0.5((p_{\min}(1-\xi) + p_{\max}(1+\xi))$. The minimal/maximal parameter value is the minimal/maximal factor times the nominal parameters from Table 8.1b.

	q_x	q_y	w	$q_{x,f}$	$q_{y,f}$	r_δ	$r_{\dot{v}}$	$r_{\dot{\nu}}$
unit	$1/(\mathrm{m}^2\cdot\mathrm{s})$	$1/(\mathrm{m}^2\cdot\mathrm{s})$	s/m^2	$1/\mathrm{m}^2$	$1/\mathrm{m}^2$	s	$\mathrm{s}^3/\mathrm{m}^2$	$\mathrm{s}^5/\mathrm{m}^2$
value	1e4	1e4	1e3	1e4	1e4	1e1	1e1	1e0

(b) Parameters for the stochastic open-loop OCP, and the robust stochastic OCP.

p_sat	$p_\mathrm{slope,lat}$	$p_\mathrm{slope,or}$
rad	rad/m	
2e−2	3e−1	1

(c) Ancillary controller parameters for the robust stochastic OCP.

Table 8.2 – Parameter variation for open-loop sensitivity, parameters of the stochastic open-loop OCP, and for the robust stochastic OCP.

effect of the uncertain wheelbase. The heuristic proportional steering feedback law is given by

$$\mu_\delta(t,\xi) = -p_\mathrm{sat}\cdot\tanh\left(\frac{p_\mathrm{slope,lat}}{p_\mathrm{sat}}d_\mathrm{lat}(t,\xi) + \frac{p_\mathrm{slope,or}}{p_\mathrm{sat}}d_\mathrm{or}(t,\xi)\right) \tag{8.49}$$

The saturation is implemented with the tanh function and the coefficient p_sat that defines the range of the ancillary control signal. The values can found in Table 8.2c. As the saturation limit p_sat is set to 0.02 [rad], the ancillary lateral controller is responsible for at most one percent of the available steering angle of $\pi/6$. Three different setups show the effect of the additional controller: the first setup assumes the same uncertainty on the wheelbase as in the stochastic OCP from i.e., between 0.95 to 1.05 times the nominal wheelbase, to compare the open-loop solution with the robust solution. The second setup has an uncertainty with a larger range of 0.90 to 1.10 times the nominal wheelbase which leads to the saturation of the ancillary controller. Finally, the third setup uses the same uncertainty as in the second experiment but applies the road boundaries as chance constraints.

Figure 8.4 shows the optimal solution of the first setup. Since the nominal and expected value of the wheelbase parameter coincide the nominal dynamics are close to the expected stochastic dynamics and as the initial value of the nominal system is equal to the expected initial value of the stochastic system, the nominal and expected paths coincide. The ancillary controller reduces the variance of the lateral path error and stabilizes the uncertain system

around the nominal system. Contrary to the optimal solution of the open-loop stochastic OCP shown in Figure 8.3a, the road boundary chance constraints are not violated and the vehicle can reach the desired end point of the horizon at $x = 20$ [m]. As the ancillary controller depends on the uncertain lateral path deviation and orientation error, it becomes a random process. Its PDF is shown in Figure 8.6 at $x = 15$ [m] and is drawn from the linear region of the tanh function as the controller output μ is far from saturation.

A larger uncertainty in the second setup leads to a larger variance of the lateral path error and consequently the ancillary controller comes closer to saturation. Figure 8.5a shows how the boundary constraints are violated despite the stabilizing controller. The kinematic nature of the vehicle is preserved despite the ancillary controller and the vehicle cannot reach the desired end point at $x = 20$ [m] in the third setup, where the chance constraints are considered as shown in Figure 8.5b. To go as far as possible, the vehicle takes advantage of the small variance at the apex of the turn and takes the shortest possible path. For both setups two and three, the ancillary controller is close to its saturation of 0.02 [rad], which equals to 1.15 [deg] as shown in Figure 8.6 for the end of the optimization horizon. Because of the decreasing slope of tanh in the region of saturation, the maximal or minimal ancillary controller output is more likely than for the solution of in first setup from Figure 8.6 with less uncertainty and thus less controller saturation.

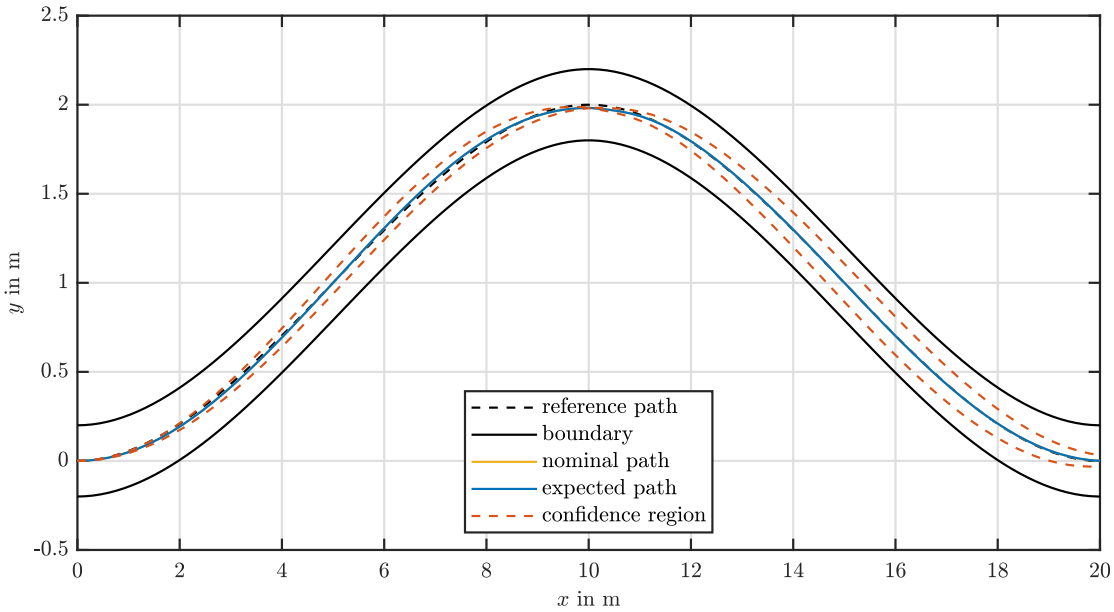
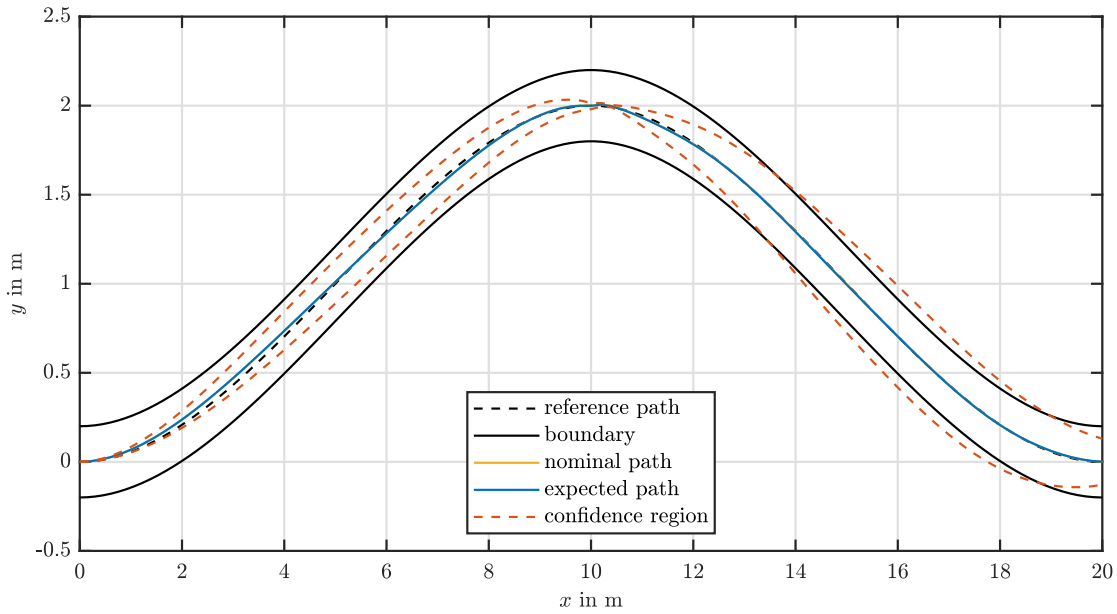
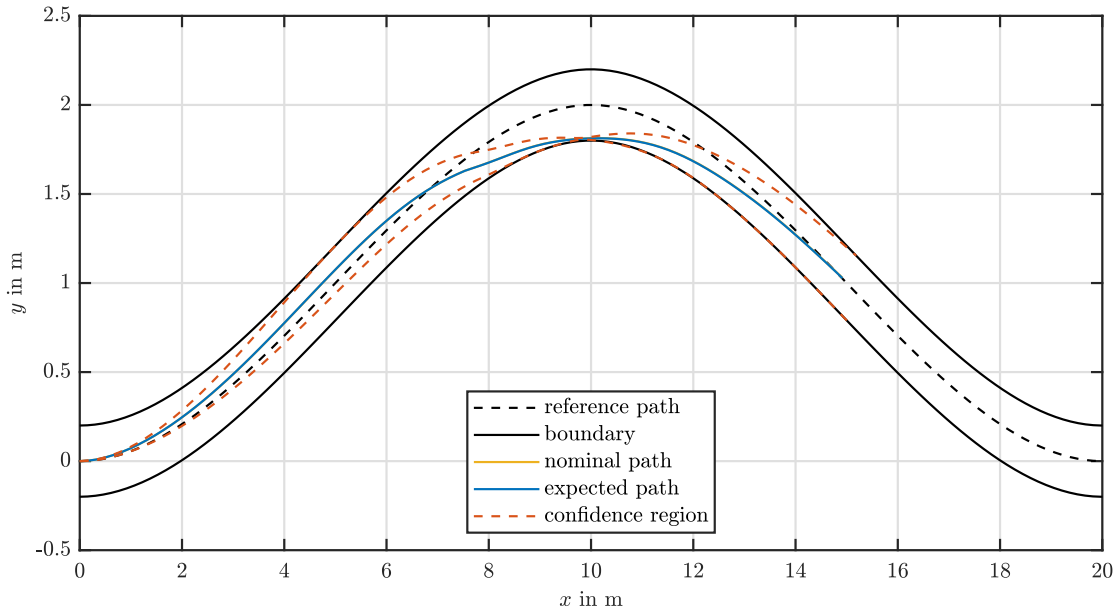


Figure 8.4 – Path solution of the stochastic OCP with the ancillary controller for the kinematic bicycle with uncertain wheelbase following a cosine with constant reference velocity. The ancillary controller reduces the variance and the vehicle reaches the desired end point at 20m.



(a) Solution without enforcing the chance constraints. Saturation of the ancillary controller leads to violation of road boundary constraints for some realisations of the uncertain parameter.



(b) Solution with consideration of the chance constraints. Due to the ancillary controller saturation and kinematic nature of the vehicle, the desired end point at 20m cannot be reached. To go as far as possible within the horizon of 2s, the optimiser exploits the kinematic relations in the model and finds the shortest path that less steering input and thus reduces the variance further.

Figure 8.5 – Path solution of the robust stochastic OCP for the kinematic bicycle with uncertain wheelbase following a cosine with constant reference velocity. Because of a large uncertainty between 0.9 and 1.0 times the nominal wheelbase, the variance grows faster than in the solution of Figure 8.4.

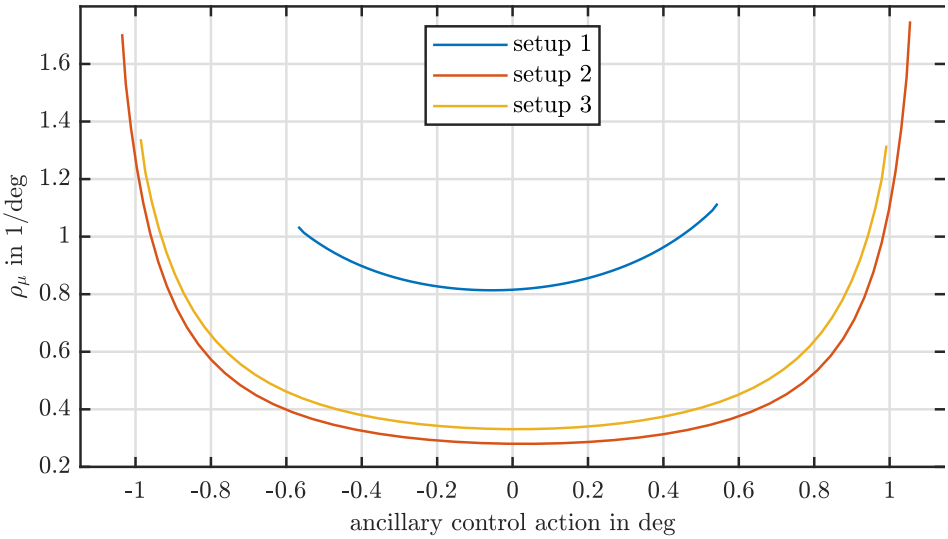


Figure 8.6 – PDF of the ancillary controller output at $x = 15$m for the first setup with an uncertainty in the range of 0.95 to 1.05 times the nominal wheelbase, the second setup with a range of 0.90 to 1.10 without consideration of the road boundaries as chance constraints, and the third setup with a range of 0.90 to 1.10 with consideration of chance constraints. For all setups, the ancillary controller saturates at 1.15deg.

8.6 Sensitivity Analysis

In this section, the Galerkin projection method introduced in Section 8.1 is applied to examine the sensitivity of the car dynamics with respect to the tire parameters. For this numerical study we use the recorded control trajectory computed by the NMPC algorithm using the dynamic bicycle with nominal parameters. Simulation is done the SODE with fixed control input for different parameters that are often uncertain in practice. The control inputs come from a double lane change manoeuvre as shown in Figure 8.7 planned with a $1.5[g]$ acceleration limit with the vehicle parameter of setup 4 as listed in Tables 8.1b and 8.1d. Inputs and reference states are taken from an optimization horizon of length $2[s]$ starting right before the second turn of the double lane change manoeuvre and ending after the third turn. As both right and left turns as well as a fast transient in between are within the horizon and the vehicle drives close to its limits, high sensitivities on the parameters can be expected.

The sensitivities to three different Pacejka model parameters are investigated using uniform distributions of several parameters: the system for the first sensitivity analysis has an uncertain D-parameter for the front and rear axes varying between $0.90 \cdot D_{\text{nominal}}$ and D_{nominal} corresponding proportionally to $\xi = -1$ and $\xi = 1$, respectively. Apart from its influence on the linear cornering stiffness, D is proportional to the maximally transmittable lateral tire force. The second sensitivity analysis retains the neutral behavior of the vehicle by varying

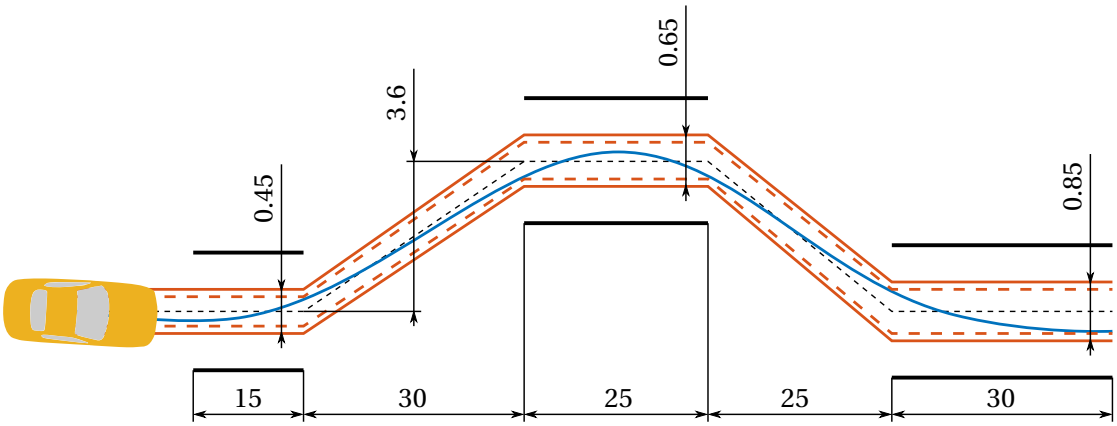


Figure 8.7 – Double lane change manoeuvre.

the linearized cornering stiffness of the front and rear axes in the same proportion. Since $C_{\text{lin}} \propto B \cdot C$, B and C are varied equally to maintain the overall shape of the Pacejka tire curve. C_{lin} is varied between $0.95 \cdot C_{\text{lin,nominal}}$ and $1.05 \cdot C_{\text{lin,nominal}}$ again corresponding proportionally to $\xi = -1$ and $\xi = 1$, respectively. The last sensitivity study weakens the rear axes by reducing the rear linear cornering stiffness only, with $C_{\text{lin,r}}$ varying $0.90 \cdot C_{\text{lin,nominal}}$ and $C_{\text{lin,nominal}}$ for ξ between $\xi = -1$ and $\xi = 1$, which renders the vehicle oversteering for small rear cornering stiffnesses. All setups are listed in Table 8.2a.

Figure 8.8 shows different realisations of the driven path of the dynamic bicycle model with uncertain D. y-position and the orientation deviate considerably from the nominal solution, as the vehicle saturates the rear tire and does not manage to get into the second turn for some realizations of ξ. To verify the approximation of gPC and the Galerkin projection, the solution of the stochastic modes evaluated at different ξ_i are compared to samples of solutions of the original ODE with parameters evaluated at the same ξ_i. Despite the rich gPC basis with polynomials up to degree eleven, the regular sampling and the gPC solution diverge. Figure 8.9 shows the reason for the divergence with the states y and $\dot{\psi}$ as an example: the gPC expansion has to approximate a kink in y and a step in $\dot{\psi}$, which leads to oscillating solutions related to the Gibbs' phenomenon [131]. Because of the poor regularity of the functions to approximate, the gPC coefficients only converge slowly in comparison to the coefficients of the solution for an uncertainty in the rear cornering stiffness as shown in Figure 8.10.

The sensitivity to the changes in the linear cornering stiffnesses is easier to approximate and a smaller basis for the gPC expansion is sufficient. In this scenario, Legendre polynomials up to degree six are used. Figure 8.11 shows a comparison of trajectory samples of the second experiment maintaining the vehicle's neutral behavior and the third experiment with the oversteering behavior. While the neutral vehicle stays close to the nominal solution, the oversteering vehicle integrates up deviations in the yaw dynamics and thus y-coordinate

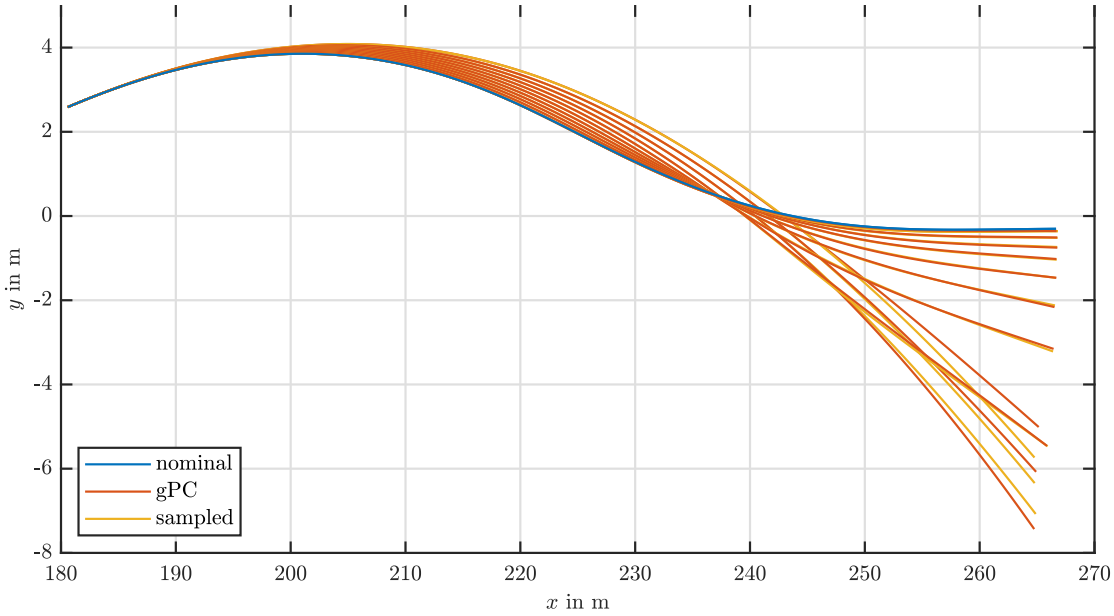
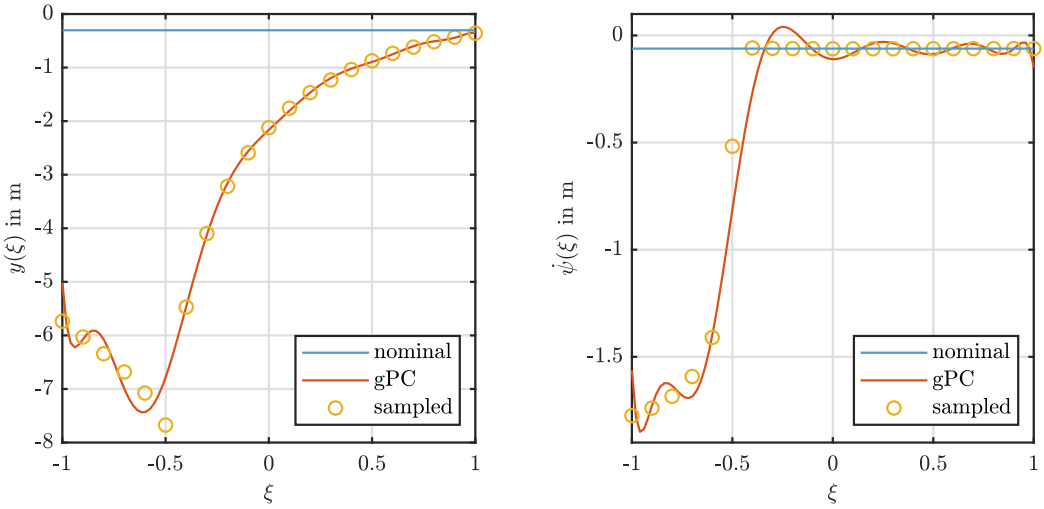


Figure 8.8 – Sensitivity for parameter variation of D of front an rear axis. The solutions with different realizations deviate from the nominal solution due to tire saturation for small D. The realizations are calculated from the gPC expansion of the system states determined by the expanded ODE and are compared to trajectory samples of the original ODE through the same parameter realizations. Errors come from the poor regularity of some system states to be expanded as shown in Figure 8.9.

deviates considerably from the nominal solution. The yaw dynamic error arises mainly in the transient phase between the turns where the tires generate yaw acceleration. Due to the lower rear stiffness, the vehicle drives with a higher slip angle β in the first turn of the horizon to counteract the centripetal acceleration. The higher slip angle also leads to higher lateral forces at the front axis, which results in a higher yaw acceleration and thus higher yaw rate than the nominal model. In the transient phase after the turn, the yaw rate must change sign rapidly, which implies high yaw acceleration in the opposite direction. The nominal model achieves this by changing the steering to the left, reducing the front lateral tire force and leads to a moment on the vehicle. The higher slip angle countersteers the force reduction and thus the yaw acceleration as shown in Figure 8.12b and the error takes the integrator chain from yaw rate to orientation. In the end of the last turn, the slip angle reduces to zero again, and the delayed yaw dynamics help the vehicle to reduce it as depicted in Figure 8.12a.

Figure 8.13 shows the nominal optimal trajectory, the OCP solution with the nominal model for the controller and worst case model as plant, the expected value of the uncertain model, the envelope of the solutions of the uncertain model and the chance constraints for a violation probability of five percent. While the worst case solution tightly follows the nominal prediction, the open-loop prediction of the uncertainty diverges as explained before. Together with the



(a) GPC expansion trying to approximate a step in y.

(b) GPC expansion trying to approximate a kink in $\dot{\psi}$.

Figure 8.9 – Gibbs' phenomenon [131] in the polynomial approximation of y and $\dot{\psi}$ at the end of the optimization horizon.

chance constraints the prediction of the open-loop system is very conservative and does not resemble the feedback solution. For the y-coordinate, which points into the same direction as the boundary normal, Figure 8.14 shows the expansion, PDF, and chance constraints at the end of the horizon.

8.7 Summary

The stochastic nonlinear OCP controller uses gPC expansions to approximate the uncertain states. The implemented projection algorithm generates a deterministic ODE governing the evolution of the expansion coefficients of a stochastic ODE with arbitrary right hand sides using the Galerkin projection and numerical quadrature. Its potential and limits are demonstrated in a sensitivity analysis of the dynamic bicycle. Road boundaries are reformulated to chance constraints, i.e., constraint satisfaction can only be guaranteed to a certain probability. Their effect on the stochastic OCP is shown with the Dubin's car model.

Finally, a stochastic nonlinear OCP formulation with an ancillary controller is proposed to overcome the conservatism of the open-loop predictions in stochastic OCP. The key idea is to stabilise the uncertain system around a nominal optimal trajectory by an ancillary feedback controller. The gPC expansions are used for efficient nonlinear uncertainty propagation. The idea is verified in simulation using the Dubin's car model.

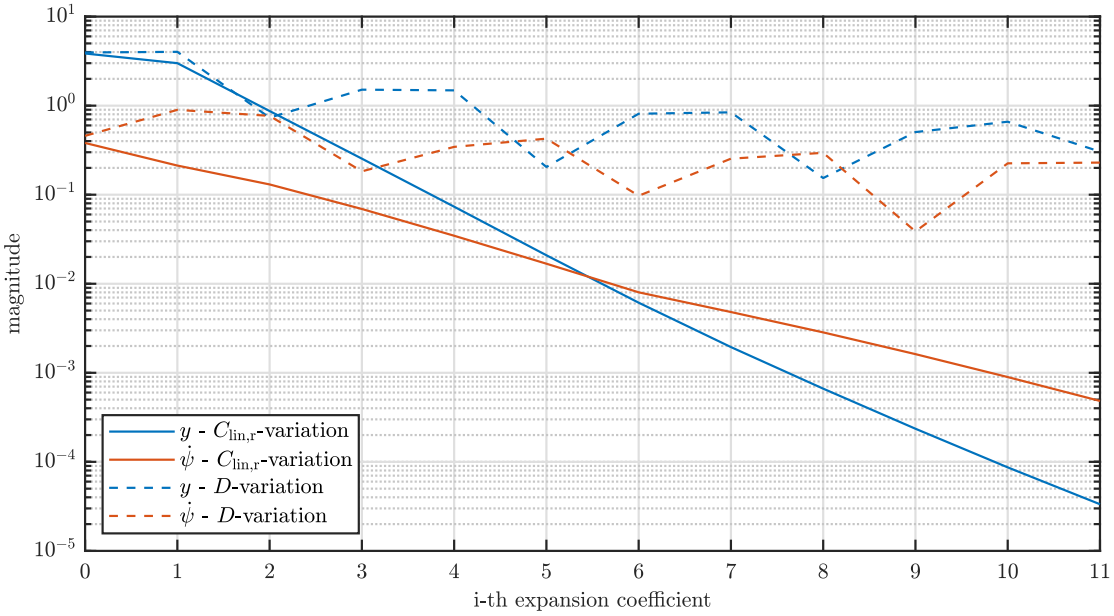


Figure 8.10 – Maximal magnitude of the i-th expansion coefficient of the approximation of states y and $\dot{\psi}$. The gPC solution of the system with D-variation shows a slower decay than the system with variation of the rear axis' cornering stiffness.

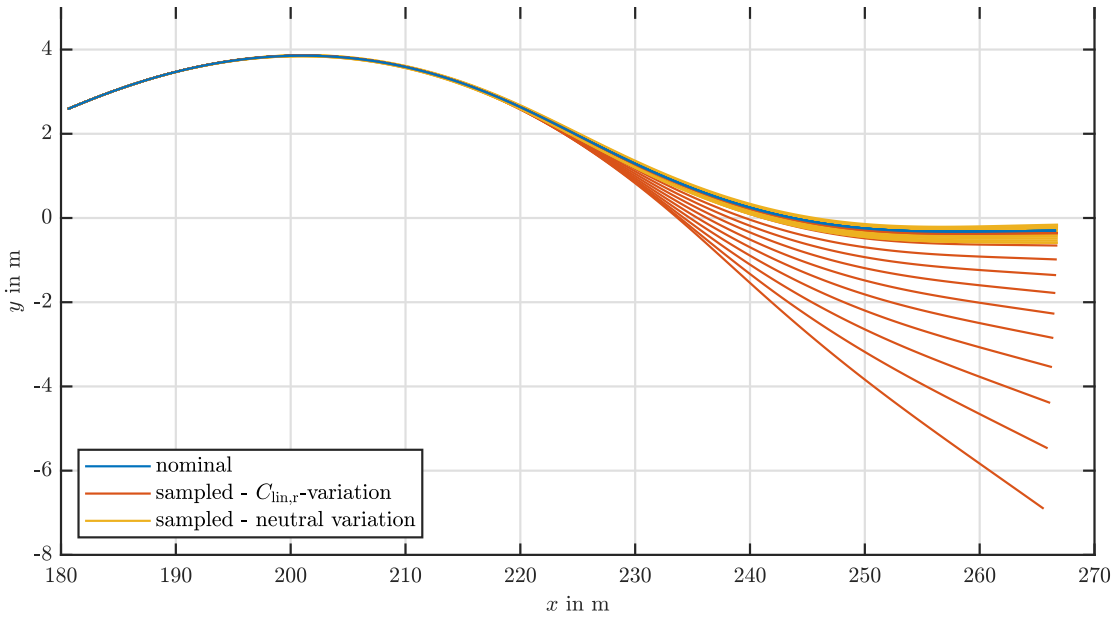
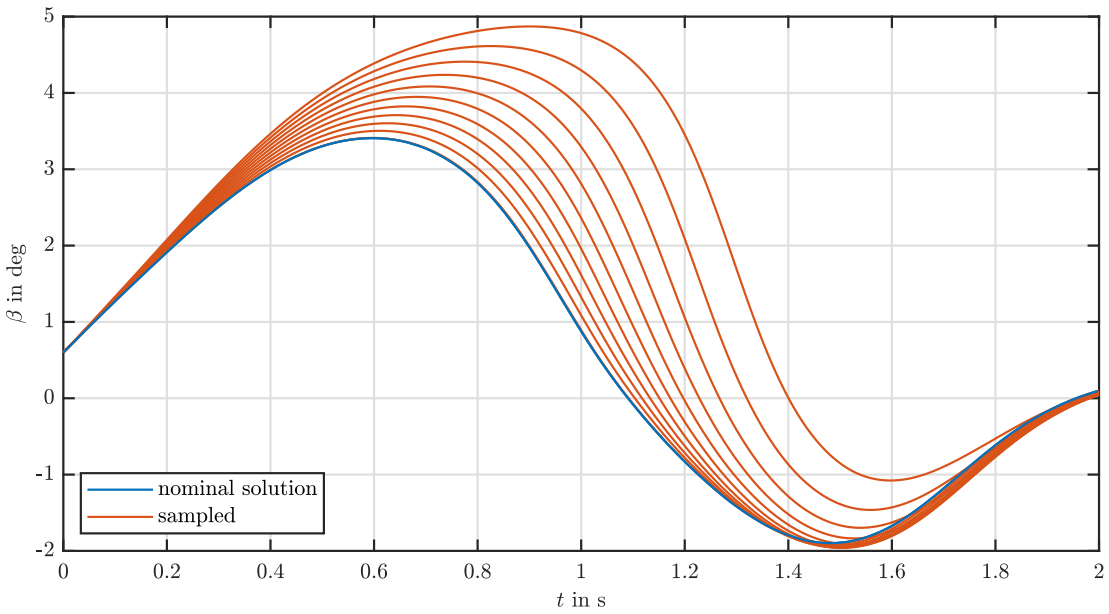
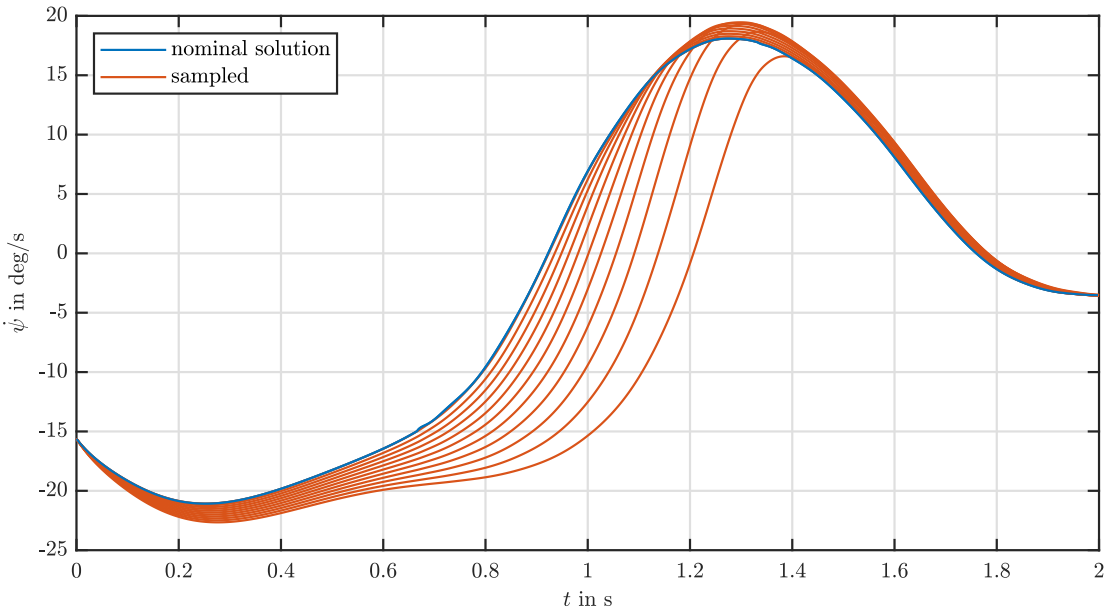


Figure 8.11 – System with variation of front and rear cornering stiffness maintaining the neutral behavior of the vehicle compared with a variation of the rear cornering stiffness leading to an oversteering behavior. While the sampled trajectories of the oversteering vehicle deviate significantly from the nominal solution, the neutral vehicle is less sensitive.



(a) Slip angle β for different realizations of a weakened rear axis. Because of the weaker rear axis higher side slip angles are necessary to withstand the centripetal forces.



(b) Slip angle $\dot{\psi}$ for different realizations of a weakened rear axis.The yaw acceleration is delayed for a weaker rear axis, as the higher side slip angle reduces the effect of the steering angle in the transient phase after the first turn.

Figure 8.12 – Yaw rate $\dot{\psi}$ and side slip angle β for different realizations of a weakened rear axis. The rear cornering stiffness varies between 0.90 and 1.0 times the nominal stiffness.

In general, stochastic nonlinear optimal control is computationally demanding and poses significant challenges to nonlinear optimisation solvers. Distributed numerical frameworks for nonlinear optimal control[145] could be explored in the future to improve robustness and

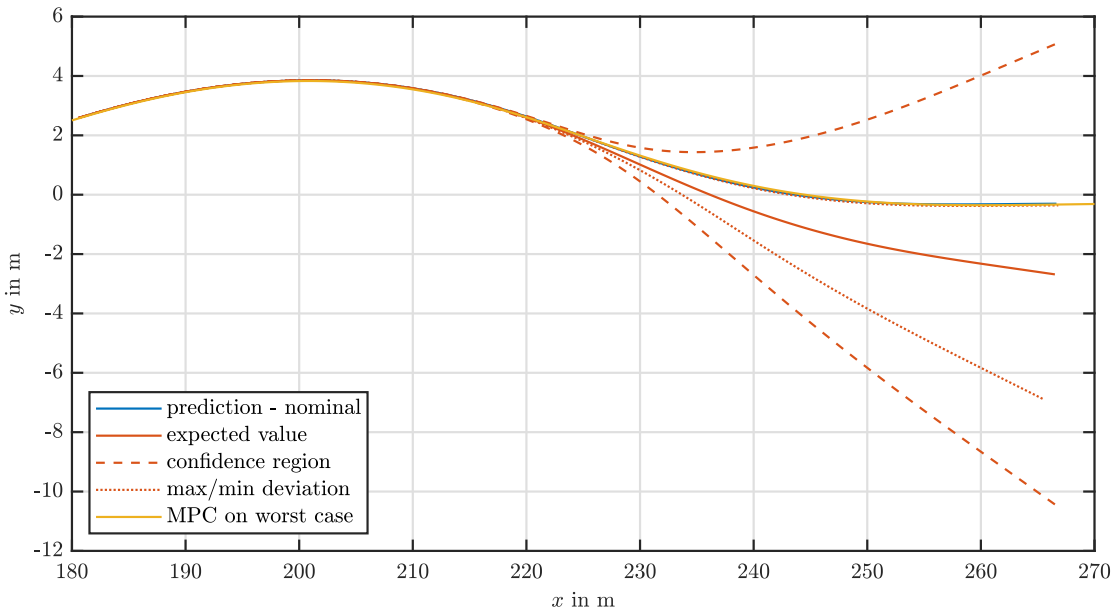
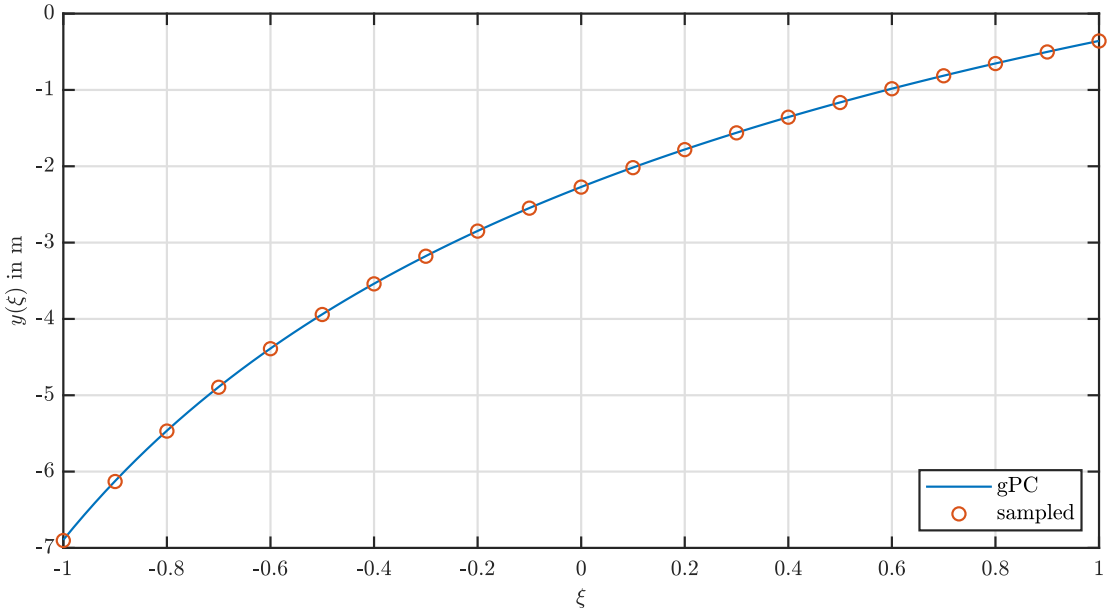


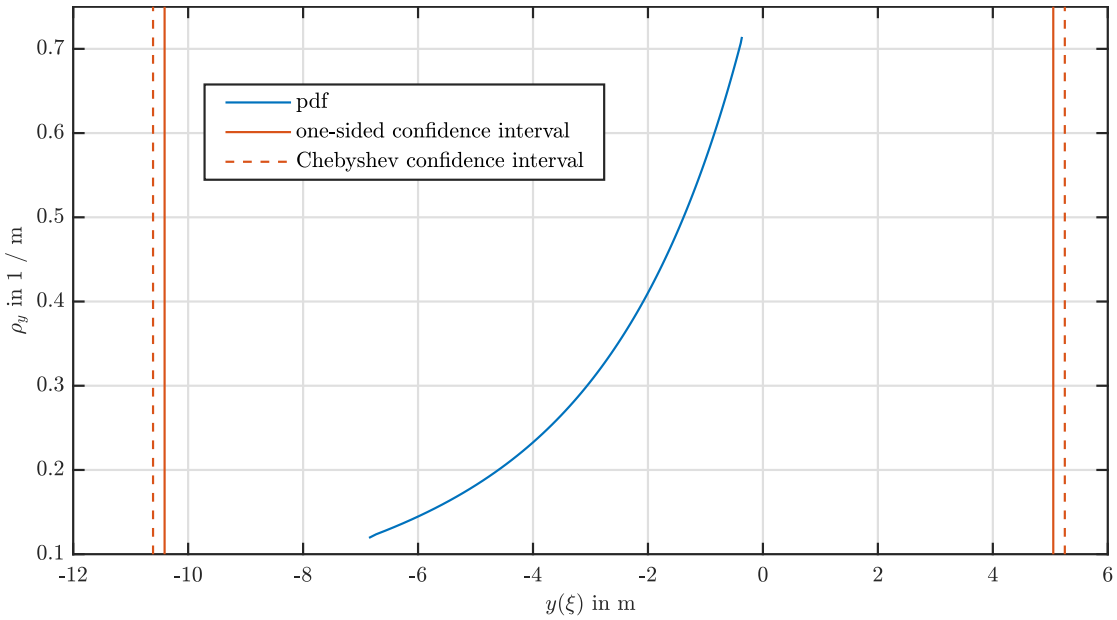
Figure 8.13 – Expected value, envelope as max/min deviations, and chance constraint of system with variation of the rear cornering stiffness compared to the nominal solution and the MPC solution with nominal model in the controller and worst case model as plant. While MPC manages to track the nominal solution tightly despite the uncertainty, the open-loop prediction diverges, amplified by the conservative chance constraints.

facilitate real-time applications of the proposed methodology.

While the gPC methodology demonstrated its efficiency for nonlinear uncertainty propagation in dynamical systems, it can be observed from the presented sensitivity study that the conservatism of chance constraints can be further reduced if sharper probabilistic bounds for chance constraints were available. Furthermore, since at every time slice the stochastic state is characterised by polynomials it is potentially possible to cast the chance constraints as polynomial positivity condition by exploring connection with the Bernstein polynomial basis [146]. These directions are also subject for the future research.



(a) GPC expansion and sampled trajectories of y at the end of the horizon. GPC approximates well the shape of $y(\xi)$ and a basis with polynomials of lower order can be considered.



(b) PDF as a push-forward of the uniform distribution of ξ through y, and chance constraints coming from Chebyshev's inequality as well as the alternative one-sided chance constraint introduced in Section 8.2.

Figure 8.14 – Expansion, PDF and chance constraints for y at the end of the horizon.

Part IV
Conclusions and outlook

Conclusions and outlook

In this thesis, we presented a novel computational and software framework for real-time embedded nonlinear optimal control that reduces the complexity of embedded deployment of optimisation-based control solutions. Furthermore, we demonstrated the application of our framework to real-time model-based motion control of several complex mechatronic systems: guidance and control of a TVC rocket prototype, identification and flight control of a fixed-wing AWE kite and autonomous driving at the limits of handling.

The main contribution of the first part is *PolyMPC*, an open-source C++ framework for real-time embedded optimisation and optimal control. Thanks to the modular design, efficiency and flexibility, the software targets not only control practitioners but also researchers in the field of optimisation and optimisation-based control as it provides full access to the code and allows simple modifications and extensions of the existing algorithmic modules. The computational efficiency is achieved with a highly optimised implementation of the Chebyshev pseudospectral collocation method and utilisation of instruction set parallelism for optimisation routines. Besides numerical benchmarks, we demonstrate the application of the software to embedded optimal guidance and motion control of a small-scale TVC rocket prototype.

Currently, *PolyMPC* exploits only instruction set parallelism in computations. For large-scale problems, e.g. stochastic optimal control or optimisation of PDEs, it is interesting to take advantage of the modern multi-core hardware architectures and distributed numerical optimisation techniques. Initial benchmarks for parallel sensitivity evaluation demonstrated that standard multi-threading and distributed computation frameworks *OpenMP* [147] and *MPI* [148] introduce considerable overhead related to thread spawning and communication corresponding to some problems considered in this thesis. From the software engineering perspective, the non-blocking thread pools [149] appear as a better alternative for parallel sensitivity computation. Numerically, ALADIN [150] provides an attractive numerical framework for distributed computation in optimal control.

An important future feature of *PolyMPC* could be generic block storage of static matrices. This

extension will allow the reduction of memory usage and enable the application of the software using sparse representations on low-end microcontrollers. The development of a high-level Python interface to the tool is another research project that is currently under development at the host laboratory.

In the second part, we presented a fully functioning AWE system prototype that serves as a low-cost platform for flight control system validation. Besides the hardware, a reliable flight simulation tool is crucial for control system development. Therefore, we developed an identification methodology for aerodynamic parameters which allows one to significantly improve the quality of the aerodynamic models. Initial estimates of the aerodynamic coefficients were obtained with CFD simulations and refined through an extensive experimental flight campaign. In our approach, we combine a model-based parameter estimation methodology, random perturbations of the 3-2-1-1 control sequences, short-term wind velocity estimation and multi-experiment averaging.

The second contribution is a real-time optimisation-based path following algorithm for full-body motion control of a kite. The algorithm allows for a flexible geometric path specification, does not require preliminary trajectory optimisation and explicitly encodes a safe flight envelope. The real-time capability of the NMPC implementation on an embedded platform is demonstrated during flight experiments. To improve the robustness in real flight conditions, we implemented a hierarchical scheme with low-level tracking of the optimal trajectories. Finally, the control scheme is validated on real flight experiments. To the best of our knowledge, it is the first NMPC implementation capable of direct actuator control of a fixed-wing aircraft.

Simulations and real flight experiments highlighted the sensitivity of the model-based control algorithms to accurate model parameters and wind estimation. Future work will be aimed at developing airspeed angle sensors for the prototype which should improve the performance of the flight controller. Additionally, sensitivity to the model parameters can be reduced in practice by higher sampling rates, and it is, therefore, interesting to investigate approximate optimal control schemes [66] or [151].

In the third part of the thesis, we develop model-based algorithms for real-time control of electric racing cars at the limits of handling. Our first contribution is an implementation of a hierarchical NMPC scheme that increases racing performance and improves robustness to uncertain road friction coefficients over the existing state of the art controller. The control algorithm is tested in a HiL platform using an industrial-grade racing simulator and automotive embedded hardware. Future work will aim at experimental testing of our implementation in a racing scenario.

In continuation of the work on NMPC for autonomous driving, we propose a stochastic NMPC approach for the case when some of the model parameters cannot be established precisely

or change quickly over time. Unknown parameters are modelled as random variables, and the model equations thus becomes stochastic differential equations. Polynomial chaos expansion (PCE) is used as a numerical tool for nonlinear uncertainty propagation, and safety chance constraints are approximated by distributionally robust inequalities. To reduce the conservatism of the stochastic approach, we proposed a novel optimal control problem formulation with a pre-stabilising controller. The performance of the stochastic NMPC algorithm is demonstrated on a simplified car model in simulation.

As mentioned before, the proposed stochastic optimal control methodology can benefit from distributed and parallel computation frameworks to enable their wider adoption in practice. Furthermore, since at every time slice the stochastic state is characterised by polynomials it is potentially possible to cast the chance constraints as polynomial positivity condition by exploring a connection with the Bernstein polynomial basis [146] for example.

Bibliography

[1] D. E. Kirk, *Optimal Control Theory: An Introduction*, ser. Dover Books on Electrical Engineering Series. Dover Publications, 2004.

[2] R. Bellman, *Dynamic Programming*. Dover Publications, 1957.

[3] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes*. New York: John Wiley & Sons, 1962.

[4] A. E. Bryson and Y.-C. Ho, *Applied optimal control: optimization, estimation, and control*. Hemisphere Publishing Corporation, 1975.

[5] J. B. Rawlings, Q. D. Mayne, and M. Diehl, *Model Predictive Control: theory, computation, and design*. Madison, Wisconsin: Nob Hill Publishing, 2017.

[6] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems*," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984, 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984.

[7] H. G. Bock, M. M. Diehl, D. B. Leineweber, and J. P. Schlöder, "A direct multiple shooting method for real-time optimization of nonlinear dae processes," in *Nonlinear Model Predictive Control*, F. Allgöwer and A. Zheng, Eds. Basel: Birkhäuser Basel, 2000, pp. 245–267.

[8] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020, 21st IFAC World Congress.

[9] L. T. Biegler, "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation," *Computers & Chemical Engineering*, vol. 8, no. 3, pp. 243–247, 1984.

[10] F. Fahroo, D. B. Doman, and A. D. Ngo, "Footprint generation for reusable launch vehicles using a direct pseudospectral method," ser. American Control Conference, vol. 3, 2003, pp. 2163–2168.

[11] D. Benson, "A gauss pseudospectral transcription for optimal control," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[12] G. Huntington, "Advancement and analysis of gauss pseudospectral transcription for optimal control problems," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[13] W. Kang, I. M. Ross, and Q. Gong, *Pseudospectral Optimal Control and Its Convergence Theorems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 109–124. [Online]. Available: https://doi.org/10.1007/978-3-540-74358-3_8

[14] M. A. Patterson and A. V. Rao, "GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 1, p. 1, 2014.

[15] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006.

[16] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[17] Y. Nie, O. Faqir, and E. C. Kerrigan, "ICLOCS2: Try this optimal control problem solver before you try the rest," in *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018, pp. 336–336.

[18] C. Büskens and D. Wassel, "The ESA NLP Solver WORHP," in *Modeling and Optimization in Space Engineering*, G. Fasano and J. D. Pintér, Eds. Springer New York, 2013, vol. 73, pp. 85–110.

[19] V. M. Becerra, "Solving complex optimal control problems at no cost with PSOPT," ser. 2010 IEEE International Symposium on Computer-Aided Control System Design (CACSD). IEEE, 2010, pp. 1391–1396.

[20] D. Thammisetty, P. Listov, and C. Jones, "mpopt: Multi-Phase Dynamic Optimisation," https://mpopt.readthedocs.io/en/latest/, accessed: 2021-10-25.

[21] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2018.

[22] O. von Stryk, "User's guide for dircol version 2.1," *Simulation and Systems Optimization Group, Technische Universität Darmstadt*, 1999.

[23] J. Betts, "Sparse optimization suite, sos, user's guide, release 2015.11," 2016.

[24] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 685–704, 2014.

[25] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, October 2021.

[26] B. Stroustrup, "Foundations of c++," in *Proceedings of the 21st European Conference on Programming Languages and Systems*, ser. ESOP'12. Berlin, Heidelberg: Springer-Verlag, 2012, p. 1–25. [Online]. Available: https://doi.org/10.1007/978-3-642-28869-2_1

[27] ——, *The C++ Programming Language*, 4th ed. Addison-Wesley Professional, 2013.

[28] D. Abrahams and A. Gurtovoy, *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth Series)*. Addison-Wesley Professional, 2004.

[29] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," 2010, http://eigen.tuxfamily.org.

[30] D. Vandevoorde, N. M. Josuttis, and D. Gregor, *C++ Templates: The Complete Guide (2nd Edition)*, 2nd ed. Addison-Wesley Professional, 2017.

[31] K. F. Graham and A. V. Rao, "Minimum-time trajectory optimization of low-thrust earth-orbit transfers with eclipsing," *Journal of Spacecraft and Rockets*, vol. 53, no. 2, pp. 289–303, 2016.

[32] F. Fahroo and I. M. Ross, "Direct trajectory optimization by a chebyshev pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 160–166, 2002.

[33] L. N. Trefethen, *Spectral methods in MATLAB*, ser. Software, Environments and Tools. SIAM, 2000, vol. 10.

[34] C. G. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral methods: Fundamentals in single domains*. Springer, 2010.

[35] B. Fornberg, *A Practical Guide to Pseudospectral Methods*. Cambdridge University Press, 1998.

[36] J. P. Boyd, *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[37] G. N. Elnagar and M. A. Kazemi, "Pseudospectral chebyshev optimal control of constrained nonlinear dynamical systems," *Computational Optimization and Applications*, vol. 11, no. 2, pp. 195–217, 1998.

[38] F. Fahroo and I. M. Ross, "Costate estimation by a legendre pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 270–277, 2001.

[39] S. Gros, M. Zanon, and M. Diehl, "A relaxation strategy for the optimization of airborne wind energy systems," in *2013 European Control Conference (ECC)*, 2013, pp. 1011–1016.

[40] E. L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*. Berlin, Heidelberg: Springer-Verlag, 1990.

[41] M. P. Neuenhofen and E. C. Kerrigan, "An integral penalty-barrier direct transcription method for optimal control," in *59th IEEE Conference on Decision and Control, CDC 2020, Jeju Island, South Korea, December 14-18, 2020*. IEEE, 2020, pp. 456–463.

[42] Y. Nie and E. C. Kerrigan, "Efficient and more accurate representation of solution trajectories in numerical optimal control," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 61–66, 2020.

[43] ——, "Efficient implementation of rate constraints for nonlinear optimal control," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 329–334, 2021.

[44] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed. USA: Society for Industrial and Applied Mathematics, 2008.

[45] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Trans. Math. Softw.*, vol. 29, no. 1, p. 58–81, Mar. 2003. [Online]. Available: https://doi.org/10.1145/641876.641880

[46] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[47] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[48] R. Quirynen and S. Di Cairano, "Presas: Block-structured preconditioning of iterative solvers within a primal active-set method for fast model predictive control," *Optimal Control Applications and Methods*, vol. 41, no. 6, pp. 2282–2307, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/oca.2652

[49] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012, pp. 668–674.

[50] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, p. 1–122, Jan. 2011. [Online]. Available: https://doi.org/10.1561/2200000016

[51] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[52] G. Stathopoulos, H. Shukla, A. Szucs, Y. Pu, and C. N. Jones, *Operator Splitting Methods in Control*. Now Foundations and Trends, 2016.

[53] D. Ruiz, "A scaling algorithm to equilibrate both rows and columns norms in matrices," Rutherford Appleton Lab, Tech. Rep., 2001.

[54] A. Griewank and C. Mitev, "Detecting Jacobian sparsity patterns by Bayesian probing," *Mathematical Programming, Ser. A*, vol. 93, no. 1, pp. 1–25, 2002.

[55] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

[56] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "A sparsity preserving convexification procedure for indefinite quadratic programs arising in direct optimal control," *SIAM Journal on Optimization*, vol. 27, 04 2017.

[57] S. Gerschgorin, "Uber die abgrenzung der eigenwerte einer matrix," *Izvestija Akademii Nauk SSSR, Serija Matematika*, vol. 7, no. 3, pp. 749–754, 1931.

[58] F. Rey, D. Frick, A. Domahidi, J. L. Jerez, M. Morari, and J. Lygeros, "Admm prescaling for model predictive control," *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 3662–3667, 2016.

[59] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*. Berlin, Heidelberg: Springer-Verlag, 1981.

[60] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," ser. ICRA workshop on open source software, vol. 3. IEEE, 2009.

[61] B. Acikmese and S. R. Ploen, "Convex programming approach to powered descent guidance for mars landing," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[62] B. Açıkmeşe, J. M. Carson, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2104–2113, 2013.

[63] B. Graf, "Quaternions and dynamics," 2008.

[64] S. Diwale, A. Alessandretti, I. Lymperopoulos, and C. N. Jones, "A nonlinear adaptive controller for airborne wind energy systems," ser. American Control Conference (ACC). IEEE, 2016, pp. 4101–4106.

[65] B. Bell, "CppAD: a package for c++ algorithmic differentiation," 2019, http://www.coin-or.org/CppAD.

[66] B. Houska, H. J. Ferreau, and M. Diehl, "An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[67] M. G. Molina and P. E. Mercado, "Modelling and control design of pitch-controlled variable speed wind turbines," in *Wind Turbines*, I. Al-Bahadly, Ed. Rijeka: IntechOpen, 2011, ch. 16.

[68] W. Katzenstein and J. Apt, "The cost of wind power variability," *Energy Policy*, vol. 51, pp. 233–243, 2012, renewable Energy in China.

[69] C. L. Archer, L. Delle Monache, and D. L. Rife, "Airborne wind energy: Optimal locations and variability," *Renewable Energy*, vol. 64, pp. 180–186, 2014.

[70] M. Sommerfeld, C. Crawford, A. Monahan, and I. Bastigkeit, "Lidar-based characterization of mid-altitude wind conditions for airborne wind energy systems," *Wind Energy*, vol. 22, no. 8, pp. 1101–1120, 2019.

[71] A. Cherubini, A. Papini, R. Vertechy, and M. Fontana, "Airborne wind energy systems: A review of the technologies," *Renewable and Sustainable Energy Reviews*, vol. 51, pp. 1461–1476, 2015.

[72] U. Ahrens, M. Diehl, and R. Schmehl, *Airborne Wind Energy*. Berlin, Heidelberg: Springer, 11 2013.

[73] C. Vermillion, M. Cobb, L. Fagiano, R. Leuthold, M. Diehl, R. S. Smith, T. A. Wood, S. Rapp, R. Schmehl, D. Olinger, and M. Demetriou, "Electricity in the air: Insights from two

decades of advanced control research and experimental flight testing of airborne wind energy systems," *Annual Reviews in Control*, 2021.

[74] B. Etkin and L. D. Reid, *Dynamics of flight : stability and control*, 3rd ed. New York ; Brisbane: Wiley, 1996.

[75] R. F. Stengel, *Flight Dynamics*. Princeton University Press, 2004.

[76] S. Rapp, R. Schmehl, E. Oland, and T. Haas, "Cascaded pumping cycle control for rigid wing airborne wind energy systems," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 11, pp. 2456–2473, 2019.

[77] S. Sieberling, "Flight guidance and control of a tethered glider in an airborne wind energy application," in *Advances in Aerospace Guidance, Navigation and Control*, Q. Chu, B. Mulder, D. Choukroun, E.-J. van Kampen, C. de Visser, and G. Looye, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 337–351.

[78] L. Fagiano, E. Nguyen-Van, F. Rager, S. Schnez, and C. Ohler, "Autonomous takeoff and flight of a tethered aircraft for airborne wind energy," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 151–166, 2018.

[79] T. Stastny, E. Ahbe, M. Dangel, and R. Siegwart, "Locally power-optimal nonlinear model predictive control for fixed-wing airborne wind energy," in *2019 American Control Conference (ACC)*, 2019, pp. 2191–2196.

[80] S. Costello, G. François, and D. Bonvin, "Real-time optimizing control of an experimental crosswind power kite," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 2, pp. 507–522, 2018.

[81] T. A. Wood, H. Hesse, A. U. Zgraggen, and R. S. Smith, "Model-based flight path planning and tracking for tethered wings," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 6712–6717.

[82] S. Gros, M. Zanon, and M. Diehl, "Control of airborne wind energy systems based on nonlinear model predictive control and moving horizon estimation," in *2013 European Control Conference (ECC)*, 2013, pp. 1017–1022.

[83] L. Ljung, *System Identification (2nd Ed.): Theory for the User*. USA: Prentice Hall PTR, 1999.

[84] P. Listov, T. Faulwasser, and C. N. Jones, "Nonlinear model predictive path following control of a fixed-wing single-line kite," in *Book of Abstracts of the International Airborne Wind Energy Conference (AWEC 2017)*, M. Diehl, R. Leuthold, and R. Schmehl, Eds. Freiburg, Germany: University of Freiburg | Delft University

of Technology, 2017, p. 90. [Online]. Available: http://resolver.tudelft.nl/uuid:febc767a-8360-45d4-b737-071723420bec

[85] E. C. Malz, J. Koenemann, S. Sieberling, and S. Gros, "A reference model for airborne wind energy systems for optimization and control," *Renewable Energy*, 2019.

[86] U. Fechner, R. van der Vlugt, E. Schreuder, and R. Schmehl, "Dynamic model of a pumping kite power system," *Renewable Energy*, vol. 83, pp. 705–716, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148115003080

[87] J. Koenemann, P. Williams, S. Sieberling, and M. Diehl, "Modeling of an airborne wind energy system with a flexible tether model for the optimization of landing trajectories," *IFAC-PapersOnLine*, vol. 50, pp. 11 944–11 950, 2017.

[88] D. J. J. Anderson, *Aircraft Performance and Design*. USA: McGraw-Hill Education, 1999.

[89] R. Finck and M. A. C. S. L. MO., *USAF (United States Air Force) Stability and Control DATCOM (Data Compendium)*. Defense Technical Information Center, 1978. [Online]. Available: https://books.google.ch/books?id=80S2NQAACAAJ

[90] "Xflr5 guidelines," https://sourceforge.net/projects/xflr5/files/Guidelines.pdf. [Online]. Available: https://sourceforge.net/projects/xflr5/files/Guidelines.pdf

[91] G. Licitra, A. Bürger, P. Williams, R. Ruiterkamp, and M. Diehl, "Optimal input design for autonomous aircraft," *Control Engineering Practice*, vol. 77, pp. 15–27, 2018.

[92] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, 2006, vol. 106, no. 1.

[93] M. Koken, "The experimental determination of the moment of inertia of a model airplane," Williams Honors College, Honors Research Projects 585, 2017.

[94] H. Schenk, "Propcalc," http://www.drivecalc.de/PropCalc/index.html. [Online]. Available: http://www.drivecalc.de/PropCalc/index.html

[95] V. Klein and E. A. Morelli, *Aircraft system identification: theory and practice*, ser. AIAA education series. Reston, VA: American Institute of Aeronautics and Astronautics, 2006, oCLC: ocm68263458.

[96] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, 2016.

[97] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1505–1511, 2017.

[98] P. Listov and C. Jones, "PolyMPC: An efficient and extensible tool for real-time nonlinear model predictive tracking and path following for fast mechatronic systems," *Optimal Control Applications and Methods*, vol. 41, no. 2, pp. 709–727, 2020.

[99] J. Herman, J. Francis, S. Ganju, B. Chen, A. Koul, A. Gupta, A. Skabelkin, I. Zhukov, M. Kumskoy, and E. Nyberg, "Learn-to-race: A multimodal control environment for autonomous racing," 2021. [Online]. Available: https://arxiv.org/abs/2103.11575

[100] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," *Vehicle System Dynamics*, vol. 58, no. 10, pp. 1497–1527, 2020. [Online]. Available: https://doi.org/10.1080/00423114.2019.1631455

[101] P. A. Theodosis and J. C. Gerdes, "Generating a Racing Line for an Autonomous Racecar Using Professional Driving Techniques," ser. Dynamic Systems and Control Conference, vol. ASME 2011 Dynamic Systems and Control Conference and Bath/ASME Symposium on Fluid Power and Motion Control, Volume 2, 10 2011, pp. 853–860. [Online]. Available: https://doi.org/10.1115/DSCC2011-6097

[102] C. Kirches, S. Sager, H. G. Bock, and J. P. Schlöder, "Time-optimal control of automobile test drives with gear shifts," *Optimal Control Applications and Methods*, vol. 31, no. 2, 2010.

[103] F. Christ, A. Wischnewski, A. Heilmeier, and B. Lohmann, "Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients," *Vehicle System Dynamics*, vol. 59, no. 4, pp. 588–612, 2021.

[104] W. F. Milliken and D. L. Milliken, *Race car vehicle dynamics*, 1994.

[105] S. C. Peters, E. Frazzoli, and K. Iagnemma, "Differential flatness of a front-steered vehicle with tire force control," *IEEE International Conference on Intelligent Robots and Systems*, vol. 02139, pp. 298–304, 2011.

[106] K. Kritayakirana and J. C. Gerdes, "Autonomous vehicle control at the limits of handling," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 4, pp. 271–296, 2012.

[107] J. H. Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," *Proceedings of the American Control Conference*, pp. 188–193, 2013.

[108] C. Chatzikomis, A. Sorniotti, P. Gruber, M. Zanchetta, D. Willans, and B. Balcombe, "Comparison of path tracking and torque-vectoring controllers for autonomous electric vehicles," *IEEE Transactions on Intelligent Vehicles*, 2018.

[109] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles," *2013 European Control Conference (ECC)*, pp. 4136–4141, 2013.

[110] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[111] A. Domahidi, "Forces: Fast optimization for real-time control on embedded systems," *Available at Forces ETHZ*, 2012.

[112] J. Kabzan, M. I. Valls, V. J. F. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart, "Amz driverless: The full autonomous racing system," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21977

[113] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, 2020.

[114] T. Novi, A. Liniger, R. Capitani, and C. Annicchiarico, "Real-time control for at-limit handling driving on a predefined path," *Vehicle System Dynamics*, vol. 0, no. 0, pp. 1–30, 2019. [Online]. Available: https://doi.org/00423114.2019.1605081

[115] T. Herrmann, A. Wischnewski, L. Hermansdorfer, J. Betz, and M. Lienkamp, "Real-time adaptive velocity optimization for autonomous electric cars at the limits of handling," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[116] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios," *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3149–3154, 2019.

[117] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle System Dynamics*, vol. 21, no. sup001, pp. 1–18, 1992.

[118] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*, 2nd ed. Society for Industrial and Applied Mathematics, 2010. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718577

[119] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, 2016.

[120] T. A. N. Heirung, J. A. Paulson, J. O'Leary, and A. Mesbah, "Stochastic model predictive control — how does it work?" *Computers and Chemical Engineering*, vol. 114, pp. 158–170, 2018. [Online]. Available: http://dx.doi.org/10.1016/j.compchemeng.2017.10.026

[121] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. M. Scokaert, C. National, and F. Telecom, "Constrained model predictive control : Stability and optimality," vol. 36, 2000.

[122] D. Q. Mayne, E. C. Kerrigan, E. J. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[123] S. Singh, M. Pavone, and J.-j. E. Slotine, "Tube-Based MPC: a Contraction Theory Approach," *IEEE Conference on Decision and Control (CDC)*, 2016.

[124] J. Nubert, K. Johannes, V. Berenz, F. Allg, and S. Trimpe, "Safe and Fast Tracking Control on a Robot Manipulator: Robut MPC and Neural Network Control," 2019.

[125] T. Brudigam, M. Olbrich, M. Leibold, and D. Wollherr, "Combining Stochastic and Scenario Model Predictive Control to Handle Target Vehicle Uncertainty in an Autonomous Driving Highway Scenario," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, pp. 1317–1324, 2018.

[126] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annual Reviews in Control*, vol. 33, no. 2, pp. 149–157, 2009.

[127] S. Streif, M. Karl, and A. Mesbah, "Stochastic Nonlinear Model Predictive Control with Efficient Sample Approximation of Chance Constraints," no. October, 2014. [Online]. Available: http://arxiv.org/abs/1410.4535

[128] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars," in *2018 European Control Conference (ECC)*. IEEE, jun 2018, pp. 1341–1348. [Online]. Available: https://ieeexplore.ieee.org/document/8550162/

[129] S. Lucia, P. Zometa, M. Kögel, and R. Findeisen, "Efficient stochastic model predictive control based on polynomial chaos expansions for embedded applications," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 3006–3012.

[130] L. Fagiano and M. Khammash, "Nonlinear stochastic model predictive control via regularized polynomial chaos expansions," in *IEEE Conference on Decision and Control (CDC)*, 2012, pp. 142–147.

[131] T. Sullivan, *Introduction to Uncertainty Quantification*. Springer Berlin Heidelberg, 2015.

[132] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010. [Online]. Available: http://www.jstor.org/stable/j.ctv7h0skv

[133] D. Xiu and G. E. Karniadakis, "The wiener–askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, 2002.

[134] M. Vitus, "Stochastic Control via Chance Constrained Optimization and its Application to Umanned Aerial Vehicles," Ph.D. dissertation, 2012.

[135] J. A. Rice, *Mathematical Statistics and Data Analysis*, 2006.

[136] G. C. Calafiore and L. E. Ghaoui, "On distributionally robust chance-constrained linear programs," *Journal of Optimization Theory and Applications*, vol. 130, no. 1, pp. 1–22, 2006.

[137] J. Winokur, "Adaptive sparse grid approaches to polynomial chaos expansions for uncertainty quantification," Ph.D. dissertation, Duke University, 2015.

[138] Y. Xu, L. Mili, and J. Zhao, "A novel polynomial-chaos-based kalman filter," *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 9–13, 2019.

[139] H. C. Ozen and G. Bal, "A dynamical polynomial chaos approach for long-time evolution of SPDEs," *Journal of Computational Physics*, vol. 343, pp. 300–323, 2017.

[140] U. Konda, P. Singla, T. Singh, and P. Scott, "Uncertainty Evolution In Stochastic Dynamic Models Using Polynomial Chaos," *Lairs.Eng.Buffalo.Edu*, 2009. [Online]. Available: http://lairs.eng.buffalo.edu/wiki/images/1/10/RC30.pdf

[141] M. Gerritsma, J. B. van der Steen, P. Vos, and G. Karniadakis, "Time-dependent generalized polynomial chaos," *Journal of Computational Physics*, vol. 229, no. 22, pp. 8333–8363, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2010.07.020

[142] V. Heuveline, "A local time–dependent Generalized Polynomial Chaos method for Stochastic Dynamical Systems," *A local time–dependent Generalized Polynomial Chaos method for Stochastic Dynamical Systems*, no. 04, 2011.

[143] H. Pacejka and I. Besselink, *Tire and Vehicle Dynamics*. Elsevier Ltd, 2012.

[144] A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz, "Stochastic nonlinear model predictive control with probabilistic constraints," *Proceedings of the American Control Conference*, pp. 2413–2419, 2014.

[145] P. Piprek, S. Gros, and F. Holzapfel, "A distributed robust optimal control framework based on polynomial chaos," in *CEAS Specialist Conference on Guidance, Navigation & Control (EuroGNC 2019)*, 2019.

[146] T. Leth, "Polynomials in the bernstein basis and their use in stability analysis," Ph.D. dissertation, Aalborg University, 2017.

[147] L. Dagum and R. Menon, "OpenMP: An industry-standard api for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, p. 46–55, Jan. 1998.

[148] M. P. Forum, "MPI: A message-passing interface standard," USA, Tech. Rep., 1994.

[149] R. P. Garg and I. A. Sharapov, "Techniques for optimizing applications: High performance computing," 2001.

[150] B. Houska and Y. Jiang. Springer, 2021, ch. Distributed Optimization and Control with ALADIN, p. 135–163.

[151] V. M. Zavala and L. T. Biegler, "The advanced-step nmpc controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109808004196

Petr Listov

Route de la Maladière, 4
1022 Chavannes-près-Renens
Switzerland
+41 (075) 412 3713
plistov@gmail.com
Date of Birth: 14.06.1989

Summary

- PhD in Electrical Engineering with 8+ years of experience in algorithms and numerical software for motion planning and control
- Solid expertise in numerical optimisation, simulation, predictive control and robotic systems

Work Experience

2016–2021 **PhD Candidate**, *Ecole Polytechnique Fédérale de Lausanne*, Switzerland.

- Developed a software framework for fast embedded nonlinear optimisation and optimal control. The software has been used in several research projects and teaching robotic practicals at the University.
- Demonstrated first autonomous flight of a small-scale airborne wind energy (AWE) kite prototype using embedded optimal control. The platform led to 2 master theses and one follow-on PhD thesis.
- Co-developed an NMPC algorithm for autonomous racing at Roborace competition with Arrival Racing Team.
- Supervised 14 master students, assisted with Control Systems and Model Predictive Control courses, co-designed robotic practicals, twice coached teams in interdisciplinary robotic competition.

March–July 2019 **Consultant**, *Arrival Ltd*, London, UK.

- Implemented fast motion planning and NMPC algorithms for automatic parking in dynamic environments using my software framework. The work prompt the company to further explore embedded optimisation solutions and led to a follow-on project with the company's racing division.

2013–2016 **Lead Research Engineer**, *RoboCV*, Moscow, Russia.

- Developed an NMPC algorithm for trajectory tracking and precise pallet handling of the company autonomously guided vehicles (AGV).
- Created an encoder calibration methodology using gradient-free optimisation. This approach facilitated quick and simple calibration of customer's vehicles at the deployment sight.
- Developed a pose estimation algorithm based on AR markers and camera.
- Led a team of two engineers to develop a motion planning technique for AGV in warehouse environment. The algorithm would allow global and local real-time path re-planning in the presence of static and dynamic obstacles.
- Actively participated in trial and deployment robotisation projects at Volkswagen (33 robots deployed) and Samsung (9 robots deployed) plants in Moscow as well as several major retail companies in Russia.

2012–2013 **Junior Researcher**, *RoboCV*, Moscow, Russia.

- Created a simulation model for the company mobile robot prototype in Gazebo.
- Developed a predictive path matching algorithm for autonomous navigation of mobile robots in warehouse environment.
- In collaboration with Sputnix LLC, developed and deployed an EKF-based orbit parameters estimation algorithm using limited GPS data and the SGP-4 ballistic model.

Education

2016–2021 **PhD Candidate in Electrical Engineering**,
Automatic Control Laboratory, EPFL.
Thesis title: Real-Time Embedded NMPC for Fast Mechatronic Systems
Thesis director: Prof. Colin N. Jones

2006–2012 **Master in Automatic Control Systems for Flight Vehicles, 4.7/5**,
Department of Computer Science and Control Systems,
Bauman Moscow State Technical University, Moscow, Russia.
Thesis title: Vision-Based Navigation for Unmanned Aerial Vehicle

Languages

English Full professional proficiency
German Limited working proficiency
French Elementary proficiency
Russian Native proficiency

Software Tools

Programming languages: C++, C, Matlab, Python
Mathematical programming: Eigen, CasADi, Boost (BGL, OdeInt) SciPy, TensorFlow, OpenCV, OpenMP\MPI
Robotics and aerodynamics: ROS\2, Gazebo, XFLR5, VRPN
Development tools: QtCreator, Git, CLion, PyCharm

Publications

[1] E. Ahbe, P. Listov, A. Iannelli, and R. S. Smith. Feedback control design maximizing the region of attraction of stochastic systems using polynomial chaos expansion. *IFAC-PapersOnLine*, 53(2):7197–7203, 2020. 21th IFAC World Congress.

[2] Y. Jiang, P. Listov, and C. Jones. Block BFGS based distributed optimization for NMPC using PolyMPC. *Accepted to European Control Conference*, 2021.

[3] P. Listov, T. Faulwasser, and C. Jones. Nonlinear model predictive path following control of a fixed-wing single-line kite. In *Book of Abstracts of the International Airborne Wind Energy Conference (AWEC 2017)*, page 90, Freiburg, Germany, 2017. University of Freiburg | Delft University of Technology.

[4] P. Listov and C. Jones. PolyMPC: An efficient and extensible tool for real-time nonlinear model predictive tracking and path following for fast mechatronic systems. *Optimal Control Applications and Methods*, 41(2):709–727, 2020.