Technische Universität München
TUM School of Computation, Information and Technology

Predictive Longitudinal Vehicle Motion Tracking Using Offset-Free Receding Horizon Control and Deep Reinforcement Learning

Martin Büchel, Dipl.-Ing.

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Martin Schulz

Prüfer der Dissertation:

1. Prof. Dr.-Ing. habil. Alois C. Knoll
2. Prof. Dr. Igor Gilitschenski

Die Dissertation wurde am 4. Januar 2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 29. Oktober 2023 angenommen.

Abstract

Automated vehicles have access to predictive information, which is unavailable for manually operated ones or is afflicted with high uncertainty. The additional information includes the desired vehicle trajectory and future road grade values. This thesis investigates how predictive information can be exploited to improve longitudinal motion tracking of automated vehicles. A method from Receding Horizon Control is evaluated and compared to results from an alternative approach based on deep reinforcement learning. Another focus is the development of a combined state and parameter estimator to determine the unknown and time-varying values of vehicle mass and driving resistance to achieve offset-free tracking.

Receding Horizon Control, also called model predictive control, has been primarily used in the process industry, applying linear models to regulate to a constant set-point. It has only been recently applied for automotive control, where short calculation cycles are mandatory. We investigate a more complex nonlinear case, controlling to a time-varying reference trajectory instead of a constant set-point. Additionally, it is uncommon to include advance knowledge about future disturbances in the framework of predictive tracking control with time-varying trajectories. Disturbances are mostly unknown, and their consideration increases the complexity of the control problem. We show that advance knowledge about future road grades can be exploited to impact tracking performance positively. This is only possible due to a combined state and parameter observer, which can estimate the time-varying parameters inherent to the physical vehicle dynamics model.

Therefore, one essential contribution of this thesis is a comparative study of different recursive methods to estimate the vehicle state and the parameters relevant to longitudinal motion. A first family of algorithms formulates the problem as augmented state space equations, modeling the parameters as additional states. This approach proves reliable but tends to show only slow convergence to the true values. Here, Bayesian algorithms based on Kalman Filtering and particle filtering were applied. An alternative approach is to reformulate the motion equations into a form that is linear in the parameters. This is shown to be challenging due to various reasons. Among these is the presence of measurement noise in the variables and insufficient system excitation during phases of constant velocity. Existing approaches show satisfying convergence to regions in the proximity of the true parameter values in various simulation studies but do not robustly show offset-free behavior. Also, this approach lacks the possibility of providing filtered values of the state variables.

This motivated the development of a novel, model-free smoothing algorithm, which can achieve improved performance compared to existing approaches. The algorithm is based on local polynomial approximation. It is able to leverage knowledge about the interrelation between measurement channels, namely that they are higher-order derivatives of a base channel. The optimal mathematical solution can be efficiently calculated and shows similarities to the ones found in convolutional neural network layers. One can also use this method to calculate additional higher-order derivatives not present in the measurement channels. As a further contribution, we show how combining this novel smoothing algorithm with a hybrid and recursive state and parameter estimator can reduce the estimation error. The next step is to combine this approach with a model predictive controller to achieve an adaptive, nonlinear solution for offset-free tracking.

Then, we investigate an alternative approach and apply model-free reinforcement learning to the predictive trajectory tracking problem. Here, an agent without prior knowledge of the system behavior learns the control policy. Including advance knowledge about future disturbances is, again, uncommon. Its inclusion raises questions during the practical implementation, for which we propose a solution capable of increasing the learning rate. Based on experiments in a simulation environment, we analyze the performance of a solution found by model-free deep reinforcement learning. We can show that the learned controller finds a solution with a performance close to the optimal one given by the previously developed model predictive control approach.

In the description of the methodical background, which lies in the intersection of control systems theory and machine learning, an emphasis was put on easing readers from one field to enter the other. An extensive review of historical developments on longitudinal control for automated vehicles and a broad outlook on future work complement this thesis.

Zusammenfassung

Automatisierte Fahrzeuge haben Zugriff auf Informationen, die im manuellen Betrieb nicht verfügbar sind. Die vorliegende Arbeit untersucht, wie diese Informationen gewinnbringend für die automatisierte Fahrzeuglängsführung eingesetzt werden können. Dazu wird eine Methode der modellprädiktiven Regelung entwickelt und einem Verfahren des Verstärkungslernens gegenübergestellt. Als weiterer Schwerpunkt ergibt sich die Entwicklung von Zustands- und Parameterschätzverfahren zur Ermittlung der genauen Fahrzeuggeschwindigkeit, der Fahrzeugmasse sowie den Luft- und Rollwiderstandswerten. Dies ermöglicht, eine bleibende Regelabweichung zu eliminieren.

Die modellprädiktive Regelung findet bereits länger in der Prozesstechnik Anwendung, in der typischerweise langsame Regelzyklen vorherrschen und die Regelgröße auf einen konstanten Wert hin stabilisiert wird. Wir betrachten hingegen die Folgeregelung zeitvarianter Referenzen und berücksichtigen Vorwissen über zukünftig eintretende Störungen. Letzteres ist bislang unüblich, da dies eine weitere Erhöhung der Komplexität zur Folge hat. Das Wissen über zukünftige Fahrbahnsteigungen als bekannte Störgröße kann jedoch vorteilhaft zur Steigerung der Regelgüte verwendet werden. Dies ist aber lediglich möglich, wenn die dem Verfahren inhärenten, zeitvarianten Fahrzeugparameter durch einen Schätzmechanismus berechnet werden.

Ein wesentlicher Beitrag dieser Arbeit untersucht deshalb vergleichend rekursive Schätzverfahren, um die für die Regelung notwendigen Zustandsgröße zusammen mit den Fahrzeugparametern hochgenau zu ermitteln. Ein Ansatz betrachtet das Problem als erweiterte Zustandsgleichungen, bei der die unbekannten Parameter als zusätzliche Zustände modelliert werden. Dazu werden Verfahren basierend auf Varianten der Kalman- sowie Partikelfilterung geprüft. Diese erweisen sich als zuverlässig, eine Konvergenz auf die Realwerte findet jedoch nur relativ langsam statt. Eine alternative Vorgehensweise besteht darin, die Gleichungen der Fahrzeuglängsdynamik in eine Formulierung umzuwandeln, die linear in den Parametern ist. Dies zeigt sich durch den in den Variablen vorhandenen Messfehler einerseits und durch eine nicht ausreichende Systemanregung während Fahrten mit konstanter Geschwindigkeit als Herausforderung, die sich nur durch eine Kombination aus verschiedenen Gegenmaßnahmen bewältigen lässt. Existierende Verfahren zeigen zwar ein befriedigendes Konvergenzverhalten in die Nähe der echten Werte, können aber eine bleibende Abweichung nicht zufriedenstellend eliminieren. Außerdem fehlt in diesen Verfahren die Möglichkeit, neben den Parametern auch gefilterte Werte der Zustände zu berechnen.

Aus dieser Problemstellung heraus wurde deshalb ein neuartiges Verfahren entwickelt, das eine im Vergleich zu existierenden Methoden verbesserte Glättung von dynamischen Größen ermöglicht. Dies beruht auf der gleichzeitigen Schätzung mehrerer Messkanäle und der Verwertung des Vorwissens darüber, dass die einzelnen verrauschten Messkanäle jeweils zeitliche Ableitungen zueinander darstellen. Die sich als Lösung ergebende mathematische Formulierung zeigt dabei Ähnlichkeit zu Faltenden Neuronalen Netzwerken, und lässt sich nicht nur zur Ermittlung von geglätteten Werten, sondern auch zur Berechnung von weiteren, in den Messungen nicht enthaltenen Ableitungen existierender Signale verwenden.

Es wird gezeigt, wie durch die Kombination dieses neuartigen Glättungsverfahrens mit einem rekursiven Parameterschätzer die Qualität der Lösung deutlich verbessert werden kann. Im nächsten Schritt wird der so vorgeschlagene Zustands- und Parameterschätzer mit dem Regler verbunden, wodurch sich ein adaptives, nichtlineares und modellprädiktives Verfahren ergibt.

Ein weiterer Beitrag untersucht einen alternativen Ansatz aus dem Gebiet des tiefen Verstärkungslernens zur Lösung des Folgeregelproblems. Hierbei soll ein Agent selbständig das optimale Regelverhalten erlernen, ohne dabei Vorwissen über das Systemverhalten zu benötigen.

Die Verwendung von Wissen über zukünftig eintretende Störungen ist auch hier unüblich, weshalb sich bei der praktischen Umsetzung neuartige Fragestellungen ergeben. Es wird eine Lösung vorgeschlagen, die den Lernvorgang unter gewissen Voraussetzungen beschleunigt. In Simulationsexperimenten wird gezeigt, dass der so erhaltene Regler auf Basis des modellfreien Verstärkungslernens sehr nahe an die optimale Lösung herankommen kann.

Bei der Beschreibung des methodischen Hintergrundes, der in der Überschneidung der Forschungsgebiete Regelungstechnik und maschinellem Lernen liegt, wurde Wert darauf gelegt, den Lesern einen Einstieg in Inhalte des jeweils anderen Feldes zu erleichtern. Ein ausführlicher Rückblick auf die historische Entwicklung bestehender Konzepte der automatisierten Fahrzeuglängsregelung sowie ein umfassender Ausblick runden die Arbeit ab.

Contents

1 Introduction

"Prediction is very difficult, especially if it's about the future."

– Niels Bohr, *1885-1962*

This thesis focuses on the control module responsible for longitudinal vehicle motion tracking for automated vehicles. By incorporating predictive knowledge, typically not available in manually driven vehicles, the motion tracking module can be improved to calculate optimal commands to both power-train and brakes. This predictive knowledge, also called advance knowledge, is, for example, the desired trajectories given by a planning module with a certain look-ahead and information about future disturbances resulting from changing road grades. Various reasons exist for why it is desirable to improve the tracking capability of automated vehicles. For example, traffic throughput of smart intersections can be increased if vehicles are able to move at minimal inter-vehicle distances. Passenger comfort can be increased, and energy consumption can be reduced. A detailed discussion of these motivational reasons is provided in Chapter 2.

First, we discuss the general vehicle control architecture of automated vehicles and how this should be different from current implementations in today's vehicles, which were designed for manual driving together with Advanced Driver Assistance Systems (ADAS). We will then investigate how methods from optimal control and reinforcement learning can be applied to design such a predictive longitudinal motion tracking controller. We will look into methods that are designed following a model-free approach and model-based methods that rely on a physical system model derived from first principles. A difficulty for any of these methods is that the longitudinal motion behavior changes over time due to changing vehicle parameters, such as the vehicle mass, the aerodynamic drag, and the rolling resistance. As an example, we want to mention the use case of people movers, which see a fluctuating number of passengers between each stop, changing the vehicle mass substantially. Nevertheless, smooth motion tracking is always desired to ensure passenger comfort and safety. For this reason, adaptation schemes are required to achieve optimal, offset-free tracking at all times. As an essential part of this thesis, we will also look into state and parameter estimation methods to obtain an adaptive control solution, considering the nonlinear behavior of the longitudinal vehicle motion.

In the remainder of this chapter, we will first discuss what we mean by longitudinal motion tracking in Section 1.1. Then, we want to present the goals and research questions this thesis aims to answer in Section 1.2. We anticipate some of the results by listing the main contributions in Section 1.3, and providing a summary of publications in Section 1.4. The structure of this thesis will be outlined in Section 1.5 before providing some taxonomy in Section 1.6 and commonly used notation in Section 1.7.

1.1 Longitudinal vehicle motion tracking

First, we want to define more in detail what we mean by the *longitudinal motion tracking task* and how this is different from other controllers typically present in an automated vehicle. Figure 1.1 depicts a typical software architecture of automated vehicles focusing on

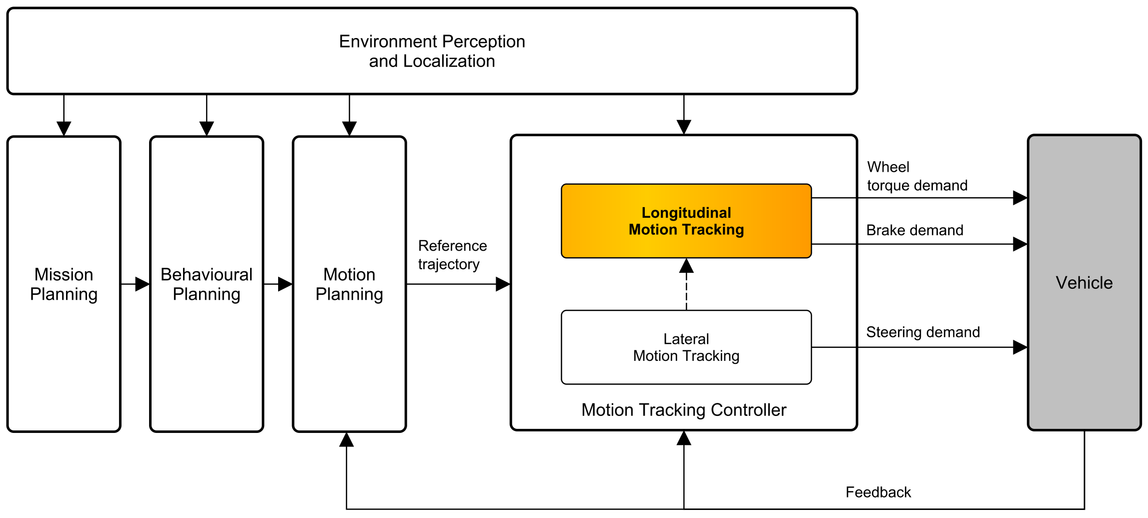

Figure 1.1: Schematic overview of different planning and control modules in a typical software architecture of an automated vehicle. This thesis focuses on the longitudinal motion control module.

motion planning and control modules. Here, this should solely explain the functionality and boundaries of the longitudinal motion tracking module as discussed in this thesis (see also Section 4 and references therein). Similar block diagrams can be found in the literature about automated vehicles or mobile robots, while variations in terminology, structure, and interfaces exist. The decision-making process of an automated vehicle can be separated into different layers. A mission planning layer (or sometimes denoted strategic layer) calculates a route to a target destination, for example, on a map. Some behavioral planning layers typically make decisions about lane selection and lane changes, as well as when to stop at a crossing or when to enter. This layer is also often called the tactical or maneuver layer. A motion planning layer calculates detailed vehicle trajectories to realize decisions made within the higher-level modules. It usually considers vehicle kinematics and geometry and avoids collisions with obstacles. Therefore, the motion planning layer is often called the reactive planning layer. On the vehicle side, motion is realized by longitudinal and lateral actuators. Longitudinal motion can be controlled via power-train and brakes, creating torques at the wheels. Steering modules realize lateral motion.

Given reference trajectories, which are calculated in a motion planner, the *motion tracking* module has the purpose of making sure that the actuators realize the desired motion. Such a clear separation between these modules is only sometimes present in current literature, as will be discussed more in detail in Section 4, but already preempts one of the proposals in this thesis.

The motion tracking task can be further divided into the following sub-tasks: A lateral motion control module is responsible for actuating the vehicle's steering. In specific scenarios, it also influences the longitudinal motion, for example, at the limit of friction. The longitudinal motion tracking controller receives a desired longitudinal motion behavior from the motion planner and transforms this input to actuator inputs of acceleration and brake. One of the assumptions within this thesis is that the input from the motion planning module is provided in the form of a reference trajectory, which, per definition, contains predictive information about desired vehicle states together with a time stamp, indicating at which time in the future this state should be reached. A feedback loop enables compensation of deviations between the vehicle's true and desired state. The optimal design of the longitudinal motion tracking control module, as just described, is the subject of investigation in this thesis.

1.2 Focus of this work

Please refer to Section 1.6 for a taxonomy related to the topic of longitudinal vehicle control.

This thesis investigates how a predictive longitudinal motion tracking control module for automated vehicles can be designed to achieve optimal performance regarding some cost function. We aim to exploit all necessary predictive information that is expected to be available in automated vehicles and incorporate this information into the control solution. We also want to provide unique control interfaces to upper-level modules and actuators in a modular fashion, clearly separating motion planning tasks from motion execution. Doing so enables a solution capable of working in all possible driving scenarios. We aim to encapsulate the complexity inherent to longitudinal vehicle motion dynamics and hide complexity from upper-level modules and actuators. This includes providing a solution to the control allocation problem, which in our case means calculating the control inputs to the partly redundant actuators of power-train and brakes.

We also want to provide a more formal description, while some points might only be made clear after reading the system description and analysis in Chapter 3. To summarize, this thesis aims to:

- Design a longitudinal vehicle motion reference tracking controller for a nonlinear, time-varying system with input constraints, considering actuator delays.
- Incorporate advance knowledge in the form of a time-varying reference trajectory and a disturbance input known over a specific prediction horizon.
- Deal with nonlinear plant behavior and with unknown system parameters (within bounds).
- Compensate for the time-varying character of the plant.
- Provide an optimal solution while considering state and input constraints to achieve offset-free tracking.
- Also, we consider a stochastic system where measurement noise is present, and the solution should be robust against measurement noise, including outliers.
- The system should be robust against model uncertainties and external disturbances.

In order to find solutions, approaches of two different domains are investigated. We investigate how results from the machine learning community can be applied, and also study solutions developed by the controls community. More in detail, we will apply deep reinforcement learning techniques for continuous control, and also develop an adaptive, nonlinear model predictive control scheme. Adaptation to time-varying parameters will be achieved in the control-based solution by means of combined state and parameter estimation. We will investigate variations and extensions of existing joint and dual Kalman-type and information filtering techniques.

1.2.1 Research questions

We want to formulate the following research questions, for which we aim to find answers within this thesis.

- Can optimal longitudinal trajectory tracking of time-varying references be achieved for automated vehicles, even in the presence of (1) actuator delays, (2) unknown and time-variant vehicle parameters, and (3) measurement noise?
- Can modern deep reinforcement learning be successfully applied to the problem of predictive motion tracking in general and longitudinal vehicle motion tracking in particular?
- Can predictive information about future disturbances be incorporated into the reinforcement learning framework?
- How does a reinforcement learning controller perform compared to the solution obtained by model predictive control?
- How robust is a model predictive controller to changes in the model due to time-varying vehicle parameters, and can offset-free tracking still be achieved without sacrificing performance?
- How and under which conditions can we learn the true unknown and time-varying parameters in an online recursive fashion?

1.2.2 Out of scope of this work

Since the focus of this thesis is solely on designing a longitudinal motion tracking module, as depicted above, the following topics are not targeted within this thesis.

We assume that a motion planning module provides target trajectories with a certain look-ahead, but we do not deal with the question of how this planning module is designed. The same applies to lower-level modules, which include brakes, power-train, and engine control, as well as questions about how a power split should be optimally performed in hybrid power-trains. This includes energy management of Hybrid Electric Vehicles (HEVs), gear shift optimization, or brake pressure control.

Providing predictive trajectories with sufficient look-ahead as an output to the actuator control modules will offer the necessary degree of freedom to allow for optimizations within these units. We are also aware that feedback from actuators would also be necessary to optimize the overall performance. However, these topics are not discussed further and are left to different investigations. The envisioned hierarchical structure allows for encapsulating complexity away from higher-layer modules.

We aim to provide a solution solely for a longitudinal trajectory tracking module while generically supporting any possible scenario an automated vehicle will encounter. Hence, ADAS specific solutions, like Cruise Control (CC), Adaptive Cruise Control (ACC), Cooperative Adaptive Cruise Control (CACC), Predictive Cruise Control (PCC), would have to be solved within some higher, reactive motion planning layer, and therefore these topics are not addressed in our discussion. This includes discussions and investigations about string stability.

While we believe that increased motion tracking performance can positively impact scenarios like vehicle platooning and smart intersections, we will also have to leave investigations to quantify this impact to future work.

As a limitation, this work focuses on dynamic scenarios such that a passenger will perceive the ride throughout as comfortable. This implies intentionally leaving vehicle maneuvers at the limits of handling out of the scope of this thesis. One might argue that, especially in mixed traffic, a vehicle has to be able to perform an emergency brake maneuver. This is true, but we would rather see any emergency scenario, also for redundancy reasons, be covered by a separate emergency control layer. This means that existing ADAS systems like *Anti-lock Braking* and *Emergency Braking* would never be integrated into a motion tracking module as investigated, while *Hill Descent Control* might be. Nevertheless, including such scenarios is left to future work.

Also, we will not consider any lateral vehicle motion within this thesis. While there certainly is an interplay between longitudinal and lateral motion concerning considerations about both passenger comfort and limits of handling, these critical questions are considered part of a higher-level module and, hence, are also not covered by this thesis. Of course, the list of what we do *not* address within this thesis cannot be exhaustive.

1.3 Research contribution

While we will provide a detailed list of the contributions achieved within each relevant chapter, here we want to give an overview of all contributions presented within this thesis. The following summarizes the research contribution made in this thesis:

- A broad literature review on longitudinal vehicle motion control is presented in Section 5.2.

- A novel controller architecture to realize longitudinal vehicle motion of automated vehicles was proposed, published in the author's work [BK16a]. It includes a predictive motion tracking control module, which can track time-varying reference trajectories while using all relevant, predictive information that is assumed to be available in future automated vehicles. This includes advance knowledge in the form of future road slope information, which acts as known disturbance on the controller.

- The predictive motion tracker is realized as Adaptive Nonlinear Model Predictive Control (ANMPC), receiving vehicle parameter estimates from a state and parameter observer. A formulation of the ANMPC controller is given, which achieves offset-free tracking for the system under consideration and solves the control allocation problem within this same framework. This builds on work presented in Buechel and Knoll [BK16a] and Buechel and Knoll [BK16b].

- Since we combine model predictive control with state estimation and parameter learning techniques, we provide a detailed analysis of related work on vehicle mass and parameter estimation and discuss many existing algorithms in detail.

- We conducted a comparative simulation study of different state and parameter estimation methods. These include joint estimation schemes based on Extended Kalman

Filters, Unscented Kalman Filters, and particle filters, which are based on a formulation as an ordinary differential equation. It partly builds on work already published in Buechel and Knoll [BK16a]. In addition, Kalman Filter methods with various enhancements applied to a formulation based on a Linear-in-Parameters (LiP) model are considered.

- Experimental results are given to investigate the effect of signal smoothing prior to performing the above-mentioned state and parameter estimation algorithms.

- Two novel smoothing algorithms are proposed, which we name Derivative-exploiting Polynomial Least Squares (DeePLS) and Derivative-exploiting Polynomial Kalman Smoother (DeePKS) in Chapter 5. The algorithms can operate on multiple measurement channels simultaneously and incorporate knowledge of different measurement channels being higher-order derivatives of a base channel. DeePLS can be seen as a generalization and extension of Savitzky-Golay [SG64] smoothing (a method widely used within the chemical processing community), while DeePKS is a recursive version of the former. DeePKS can also be seen as the generalization of Polynomial Kalman Smoother (PKS) [RB+14]. The generic formulation allows the incorporation of irregularly sampled data and delayed and missing measurements. Both algorithms are shown to reduce estimation errors compared to their existing counterparts. The DeePLS algorithm is presented in Buechel [Bue22].

- Additionally, an interpretation of the DeePLS algorithm was provided as a combined concatenation and convolution layer with a physics-informed, optimal kernel. This brings valuable insights for physics-informed machine learning algorithms working on multi-channel time-series data that involve derivatives.

- A proposed hybrid estimation scheme combines state estimation based on a formulation as an ordinary differential equation with a parameter estimator based on a LiP model. One crucial ingredient in this estimation scheme is to perform signal smoothing prior to state and parameter estimation, using the DeePLS algorithm proposed in Chapter 5. This improves estimation performance substantially in various simulation studies.

- As an alternative solution, we investigate the application of using Deep Reinforcement Learning (DRL) for predictive tracking control and start with a broad review of various Reinforcement Learning (RL) algorithms suited to solve the continuous tracking control problem.

- We extend the framework of DRL to include predictive information, including advance knowledge about future disturbances, and propose a solution that was previously published in [BK18] for longitudinal motion tracking of automated vehicles.

- A novel research question arises when trying to train reinforcement learning agents in the context of predictive motion tracking: how to design the reference trajectories the agent experiences during learning. We propose to apply a method already known within the systems identification community and empirically show that this increases the sample efficiency during learning compared to a naive approach. This suggestion was already published by the author in [BK18].

- A comparison between the method of Reinforcement Learning (RL) for predictive, continuous control and Model Predictive Control (MPC) applied to longitudinal motion tracking is provided. This can be found in Section 9.2.2.

- We provide an analysis of the proposed, learning-based, model predictive control solution and identify relevant building blocks to achieve robust learning even during phases of insufficient excitation in Section 9.2.1.

- A broad outlook on future work related to the topics covered in this thesis is given in Section 9.3.

1.4 Summary of publications

Parts of this dissertation build on the following papers, which were published in the course of this thesis:

1. Buechel and Knoll [BK16a]. "A parameter estimator for a model based adaptive control scheme for longitudinal control of automated vehicles" proposes the predictive control architecture together with a joint state and parameter estimator.

2. Buechel and Knoll [BK16b]. "An adaptive nonlinear model predictive controller for longitudinal motion of automated vehicles" combines and evaluates the proposed joint state and parameter estimator with the predictive controller, which is capable of exploiting knowledge about future disturbances.

3. Buechel and Knoll [BK18]. "Deep reinforcement learning for predictive longitudinal control of automated vehicles" compares the previously proposed model predictive control approach with a solution based on deep reinforcement learning.

4. Buechel [Bue22]. "Physics-informed kernels for convolutional smoothing of multichannel time-series data" presents a novel smoothing approach.

The following publications were presented during the author's time as a guest at the Technical University of Munich, as well as projects while working at the research institute of the Free State of Bavaria, *fortiss GmbH*. While some may share ideas with this thesis, they are not a core contribution to this work.

1. Buechel, Frtunikj, et al. [BF+15]. "An automated electric vehicle prototype showing new trends in automotive architectures".

2. Buechel, Hinz, et al. [BH+17]. "Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles".

3. Buechel, Schellmann, et al. [BS+19]. "Fortuna: Presenting the 5G-connected automated vehicle prototype of the project PROVIDENTIA".

4. Hinz, Buechel, et al. [HB+17]. "Designing a far-reaching view for highway traffic scenarios with 5G-based intelligent infrastructure".

5. Hinz, Eichinger, Buechel, and Knoll [HE+17]. "Proactive video-based use of telecommunications technologies in innovative motorway scenarios".

6. Kessler, Bernhard, Buechel, et al. [KB+19]. "Bridging the gap between open source software and vehicle hardware for autonomous driving".

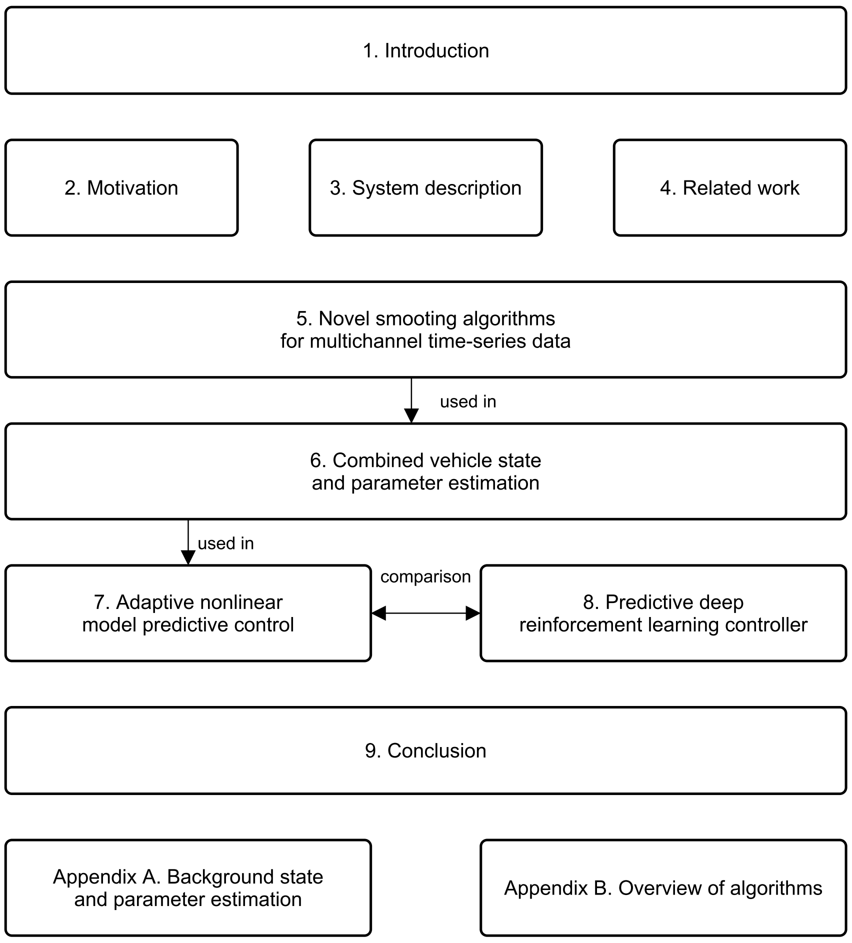
1. Introduction
2. Motivation
3. System description
4. Related work
5. Novel smooting algorithms for multichannel time-series data
used in
6. Combined vehicle state and parameter estimation
used in
comparison
7. Adaptive nonlinear model predictive control
8. Predictive deep reinforcement learning controller
9. Conclusion
Appendix A. Background state and parameter estimation
Appendix B. Overview of algorithms

Figure 1.2: Thesis chapter overview.

1.5 Thesis outline

A graphical illustration of the chapters of this thesis and their inter-dependencies is given in Figure 1.2. This thesis is structured into the following chapters.

Introduction (Chapter 1) This chapter provides an overview of this document. It depicts the research questions treated and lists the contributions achieved within this thesis. A summary of the publications of the author of this thesis is included. Also, we provide some taxonomy to clarify some essential concepts and list some mathematical notation used within this thesis.

Motivation (Chapter 2) Here, we give some background about longitudinal vehicle control for automated vehicles, which is necessary to elaborate on the motivation and the solutions proposed in this thesis. After defining different levels of vehicle automation, we adumbrate the history of automated vehicles and vehicle control systems. We then define the tasks of longitudinal vehicle control in detail and elaborate on the difference between longitudinal control for manual vehicles with ADAS systems and automated vehicles. An analysis of the physical effects determining the behavior of the control system is done, which later helps to systematically derive the proposed control structure. To underline the importance of the longitudinal controller for automated vehicles, we analyze future trends and the impact on control quality on various evaluation criteria.

System Description (Chapter 3) We derive a first-principles model of the longitudinal vehicle dynamics, which is then used in the sections dedicated to model-based control and estimation. A discussion of these equations, including ranges of expected vehicle parameters, follows. This underlines the motivation to build an adaptive and predictive control scheme.

Related Work (Chapter 4) While topic-specific related work sections are included in each of the subsequent chapters, this chapter aims to depict an overview of the state of the art in longitudinal vehicle control in general.

Novel Smoothing Algorithms for Multichannel Time-Series Data (Chapter 5) In this chapter, we propose two novel algorithms for smoothing multichannel time-series data, incorporating knowledge about the interrelation between measurement channels. Results are used in the combined vehicle state and parameter estimator in Chapter 6.

State and Parameter Estimation (Chapter 6) In Chapter 6, we present a combined state and parameter estimator that can track unknown and time-varying parameters. The proposed solution is used in combination with the nonlinear model predictive controller presented in Chapter 7. The proposed solution is built on the smoothing algorithm presented in Chapter 5. The work builds on material published already in [BK16a], but contains substantial unpublished contributions.

Adaptive Nonlinear Model Predictive Control (Chapter 7) This chapter builds the core of the proposed model-based control approach. As can be seen in Figure 1.2, it will be using results from Chapter 5 and Chapter 6. The author of this thesis previously published preliminary results using the same approach in Buechel and Knoll [BK16b], but Chapter 7 contains substantial, previously unpublished work.

Predictive Deep Reinforcement Learning Controller (Chapter 8) Chapter 8 investigates an alternative solution to the model predictive control approach discussed in Chapter 7, based on deep reinforcement learning. The consideration of known future disturbances in RL schemes poses novel challenges. We will present a proposal that improves the learning rate compared to state-of-the-art algorithms. Results of this chapter were previously published in the author's paper [BK18].

Conclusion (Chapter 9) Chapter 9 concludes with a summary of the thesis and gives an outlook on future research directions. A comparison between model predictive control and reinforcement learning is given, together with an analysis of the structure of the ANMPC approach seen from a machine learning perspective.

Appendix A: Background State and Parameter Estimation Appendix A provides the necessary background for the algorithms used in Chapter 6.

Appendix B: Overview of Algorithms An overview of some of the algorithms used as building blocks of the proposed estimator solution or as baselines is given in Appendix B.

1.6 Taxonomy

In this section, we want to introduce or clarify some terminology that will be used throughout this thesis. Nevertheless, we will introduce additional terms and expressions within the relevant chapters if their understanding requires some context.

Levels of vehicle automation This thesis adopts the taxonomy of different levels of vehicle automation defined by Society of Automotive Engineers (SAE) International [SAE18]. This is currently the most commonly used definition within the automotive community. SAE defines six mutually exclusive levels of automation, ranging from no driving automation (level 0) to full driving automation (Level 5). For details, please refer to [SAE18]. Level 0 has no automation, meaning that this refers to fully manual vehicles. In Level 1 vehicles, the longitudinal or the lateral control is automated, while Level 2 vehicles are equipped with controls in both dimensions. Levels 0 to 2 are only safe, with a human driver monitoring the environment at all times and reacting immediately in case the automation fails. Systems of Level 1 and Level 2 automation are typically also within the group referred to as Advanced Driver Assistance System (ADAS) systems. At the same time, the term ADAS also includes passive systems, which do not actively interfere with vehicle controls. This could be, for example, a forward collision warning or lane-keeping feature. In conditional automation Level 3, the automation system is already capable of monitoring the driving task and, in case of failure, warns the driver and expects him to take over within a predefined period of time. A Level 4 vehicle works without the driver's expectation to take over but is restricted to a limited number of driving tasks. In contrast, the remaining driving tasks must be carried out by a driver. Examples of Level 4 driving would be a piloted parking application or a highway pilot. Full, unlimited automation is defined as Level 5, for which a driver (and hence a steering wheel) would become obsolete.

Throughout this thesis, we will denote vehicles with an Automated Driving System (ADS) of Level 3 to Level 5 as Automated Vehicles (AVs). As we will learn in Section 4, longitudinal control solutions for AVs will typically be fitting within some vehicle architecture similar to what we presented in Figure 1.1. On the contrary, ADAS systems are typically designed following a structure as discussed in Section 1.6.

Platooning Platooning is when several vehicles form road trains with virtual axles, following each other with small inter-vehicle distances. The first vehicle in the road train, the lead vehicle, might be manually driven while the followers operate in automated mode.

String stability One challenge with automated vehicle-following systems is the problem of string stability. Spacing errors might be propagated throughout a vehicle platoon, and oscillations might be amplified. For safety reasons, uniform boundedness of all vehicle states needs to be guaranteed. Manually driven vehicles in a platoon lose the string stability property when inter-vehicle distances are too small, which can result in rear-end collisions in the worst case. For a precise definition of string stability, see [Kai74]. See also Section 1.6 and Section 4.3.5.

Cruise Control In the Cruise-Control scenario, the vehicle can automatically maintain a desired, constant target velocity, and the driver is responsible for reducing speed in case of violating a safety distance to a potential lead vehicle. This is considered to be of SAE Level 1.

Adaptive Cruise Control Adaptive Cruise Control (ACC) additionally enhances a Cruise Control system with the ability to keep a safe distance between the ego-vehicle and a preceding lead vehicle. This requires a sensing system that provides the current distance between the vehicles. Radar sensors are primarily used for this purpose due to the lower costs and higher robustness compared to Lidar sensors. Limitations are given regarding the performance of vehicle platoons using ACC. Due to dynamic delays and lags present in the longitudinal vehicle dynamics, the safe distance cannot be set as a constant (constant-clearance-following policy) but has to be increased with the platoon's velocity (constant-time-headway policy). Otherwise, vehicle platoons consisting of many vehicles with ACC systems following each other have been proven to lose the property of string stability (see Section 1.6). Keeping a constant time gap instead of a constant clearance between the vehicles means their distance is increased proportionally to the vehicle's speed.

Alarmingly, a recent study that conducted various experiments with production vehicles suggests that many commercial ACC systems are not string stable [GG+19]. The reason might be owed to using far too simplifying assumptions during the control design process [XG11]. Refer to Section 4.3.5 for more information. This is where vehicle platoons using CACC systems come in place, as will be discussed in the next section.

Cooperative Adaptive Cruise Control and Platooning Nevertheless, small distances between vehicles are desired to increase traffic throughput and reduce energy consumption by reducing air drag. As discussed above, to overcome the burden of losing string stability, Cooperative Adaptive Cruise Control (CACC) systems come in place. By enabling a communication channel between the front vehicle of a platoon and its followers, investigations showed that one could reduce the distance between vehicles compared to a regular platoon using ACC only.

In CACC systems, control laws under the constant-clearance-following policy can theoretically be applied without losing the string stability property. Shladover, Nowakowski, Lu, and Hoogendoorn [SN+14] still suggested that it would be more likely to use a constant-time-gap-following policy in a production CACC system for safety reasons. At the same time, the full potential for fuel efficiency improvements can only be demonstrated using a constant-clearance policy. Nevertheless, the time-headway can be reduced dramatically with CACC systems (in the order of around 1.4 seconds when driving manually to a value of 0.6 seconds when using CACC according to [SN+14].) This would increase lane capacity from about 2000 vehicles per hour to almost 4000 vehicles per hour.

Predictive Cruise Control Compared to other Cruise Control solutions, PCC systems typically do not only provide an acceleration set-point, which is then forwarded to the low-level control unit. Instead, PCC solutions use predictive information, for example, about the status of traffic lights at intersections, to generate a reference velocity, which is given to a lower-level (see Section 1.6) tracking controller. See also [DAF21] for more information.

High-level and low-level controllers of ADAS solutions Longitudinal controls for ADAS systems are typically designed for a limited scope of scenarios, like keeping a certain constant vehicle speed or following a preceding vehicle. The control architecture of ADAS systems is typically divided into a high-level and a low-level controller. A high-level longitudinal motion control module takes the velocities of the ego vehicle and lead vehicle and the relative distance as inputs and aims to achieve the desired set speed and time gap between the ego vehicle and the lead vehicle. It then typically calculates a desired vehicle acceleration set-value, given to a lower-level longitudinal control module. For example, the lower-level controller calculates the desired wheel torque value and compensates for disturbances like road grade changes or wind forces. The lower-level control module also splits between commands to the engine or power-train and the brakes. Many variations exist, in which, for example, compensation for the nonlinearity in the power-train is also integrated into the low-level controller, and engine torque or throttle requests are calculated instead. Historically, evolution has been taking place from pure Cruise Control solutions over ACC systems to CACCs or PCCs.

Motion planning Motion planning for automated vehicles is typically defined as the problem of finding a path or trajectory that brings the vehicle from an initial configuration to a target state. One can distinguish between *path* and *trajectory* planning. *Path planning* is typically referred to as the problem of "planning the future motion of the vehicle in the configuration space". Trajectory planning is the "search of a time-parametrized solution" [GK+18]. The information is often encoded in waypoints (with a low spatial resolution) or trajectories (with a resolution that suits the sampling time of the underlying tracking controller). Vehicle speed profiles are either separately generated (in the most straightforward case, only constant velocities) or implicitly included in the trajectories.

Motion tracking and vehicle motion control The module, which performs the *motion tracking* task, will be called *motion tracking module*, interchangeably with *motion tracking controller* and the shorthanded version *tracking controller*. In some occurrences, we will use the term *vehicle motion controller* when it should be clear from the context that we refer to the tracking module. See also Section 1.1 for more details.

Remark 1. *According to the SAE [SAE16], longitudinal vehicle motion control is defined as "the activities necessary for the real-time, sustained regulation of the x-axis component of vehicle motion. [. . .] Longitudinal vehicle motion control includes maintaining a set speed and detecting a preceding vehicle in the path of the subject vehicle, maintaining an appropriate gap to the preceding vehicle, and applying propulsion or braking inputs to cause the vehicle to maintain that speed or gap."*

Although some of the definitions of the SAE are adopted in this thesis (see Section 1.6), this definition is not in accordance with the one given above. The sub-task of detecting a preceding vehicle we considered realized within the environment perception block, and we consider maintaining an appropriate gap as part of the behavioral and motion planning blocks depicted in Figure 1.1.

Predictive tracking control We use the term predictive tracking controller to describe an algorithm that solves the task of tracking time-varying reference trajectories. This is in contradiction to set-point controllers, which operate under the restrictive assumption that reference trajectories consist of constant future desired values. In addition, in predictive tracking, advance knowledge might be included to increase the controller's tracking performance and disturbance rejection capabilities.

Model predictive control A model predictive controller makes use of a model which describes the dynamic evolution of the controlled process over time. See Chapter 7 for an introduction to the topic.

Advance knowledge In this dissertation, we will borrow the term *advance knowledge*, which was already used before in the context of MPC by Rossiter, Schuurmans, and Grinnell [RS+99] in 1999. We will use *advance knowledge* to refer to any given future information that has an impact on the plant behavior. This information can be about desired trajectory set-points, time-variant reference trajectories, or any quantity that impacts the plant behavior, like future disturbances. In our case, we will also use the term *advance information* interchangeably with *advance knowledge*, and mostly mean the information about future disturbances, together with time-varying trajectories, since this is regarded as given in the example under investigation. Nevertheless, by using *advance knowledge*, we want to emphasize that we want to stay more general than in our specific example.

Control allocation problem We want to borrow the following definition, given in [Har21] "The control allocation problem is that of distributing a desired total control effort among a redundant set of actuators". In the case of longitudinal vehicle motion tracking, the redundant set of actuators are brakes and the engine. Although brakes can only generate a negative wheel torque and engines typically operate to induce a positive torque at the wheels, the latter may also operate within the negative torque range, which introduces redundancy. This is especially the case when electric engines within a vehicle's power-train operate in recuperation mode and when combustion engines produce a negative drag torque. Then, to maximize the recuperated energy and/or reduce wear and tear of brakes, it is desirable to apply them only when necessary. See Section 7.3.10 for a more control-oriented discussion.

Reinforcement learning related terminology Since terminology related to reinforcement learning is more easily presented in the context of an introduction to this broad field, please refer to Section 8.2.

Filtering and smoothing We can define *filtering* as the causal process of estimating a signal at a particular time given past noisy observations. In contrast, *smoothing* uses past *and* future data to calculate the estimate. This makes smoothing non-causal.

1.7 Notation

While we will introduce mathematical symbols when appearing within each chapter, we quickly want to highlight some of the most essential mathematical notation used within this thesis. $\mathbb{R}$ denotes the set of real numbers. For both scalars and vectors, we use italic lowercase, e.g., x, s. An exception is made in Chapter 5, where we will explicitly mention the cases where bold symbols are used for vectors. Matrices are in italic upper case, e.g., A, B, and the symbol $\mathbb{I}$ denotes the Identity matrix. Constants are in upright lower case, e.g., gravity constant g.

For any vector $x \in \mathbb{R}^n$, we write $||x||$ to denote the Euclidean norm, and its square as $||x||^2 = x^\mathsf{T} x$, while the weighted squared ℓ_2-norm is denoted as $||x||_W^2 = x^\mathsf{T} W x$. The absolute value of some number x is denoted as $x = |-x|$. We will use a MATLAB®-style notation, for example for the vertical concatenation of vectors $y_i \in \mathbb{R}^{m\times 1}$ we write $[y_1; y_2; \ldots; y_n] \in \mathbb{R}^{n\cdot m\times 1}$, while the horizontal concatenation of vectors $y_i \in \mathbb{R}^{m\times 1}$ is written as $[y_1, y_2, \ldots, y_n] \in \mathbb{R}^{m\times n}$. Similarly, $A_{[r-k:r,:]}$ denotes a matrix that consists of a selection of A, with the rows given by the indices $r-k$ to r, while : denotes that all columns are used. We further use $\hat{\theta}$ to denote that we are talking about the estimation of the parameter vector θ.

We denote $x \sim \mathcal{N}(\mu, \sigma)$ for a random variable x drawn from a normal distribution with mean μ and standard deviation σ. The expression $r \sim E$ will read as "sampled from", in this example a reward r is sampled from an environment E. $\mathbb{E}[X]$ is used for the expectation of a random variable X, i.e., $\mathbb{E}[X] \doteq \sum_\mathrm{x} p(x)x$, while $p\big(s', r|s, u\big)$ denotes the conditional probability, in this example of the transition to state s' with reward r given state s and action u.

2 Motivation

"Safety, Comfort, Speed and Economy: [...] From the beginning of time, whenever people have tried to get from one place to another, they have kept these same basic aims in mind. The first is their desire for self-preservation; the second is their desire for a pleasant trip; the third is their desire to reach their goal quickly; and the fourth is their desire to spend as little money and effort on the way as possible. Now, for self-preservation, read safety*; for a convenient and pleasant trip, read* comfort*; for a quick arrival, read* speed*; and for a saving of expense and effort, read* economy.*"*

– Norman Bel Geddes, *Magic Motorways (1940)*

Since the invention of automobiles in the 19th century and of modern computers in the second half of the 20th century, many efforts have been made to make automobiles be what the etymological meaning of their name[1] promises them to be: *self-moving*. Here, we allow a modern interpretation, such that a vehicle is not only moving without being pulled by animals but capable of driving in the absence of a driver. Efforts in this direction started surprisingly early, even at a time when automobiles were purely mechanical masterpieces. It was already around the year 1900 when the first predecessors of cruise control systems appeared. After this period, mechanical parts have been gradually substituted by mechatronic solutions in order to increase passenger safety and comfort and to reduce energy consumption and emissions.

The first electronic throttle devices, appearing in the 1980s, enabled them to fit vehicles with electronic cruise control systems. The first cruise controls were only built to keep the vehicle at a constant speed for highway applications. In the late 1990s, these systems were improved using sensor information, allowing the speed to be adapted according to the movement of a preceding vehicle.

Since the 1980s, many research prototype vehicles have proven that, equipped with a sensor set to perceive its environment together with electronic steer- and drive-by-wire solutions, and given enough computational power, it is possible to achieve the vision of creating fully automated vehicles.

Automated vehicles not only perceive the current state of their surroundings through their sensors but also have some capability to predict what will happen in the future - at least over a certain time horizon. Maps provide information about road characteristics like curvature or road slope. Perception and prediction algorithms calculate information about the trajectories

[1] ***automobile***. From: Greek "$\alpha\upsilon\tau o\varsigma$" ("self") and Latin "mobilis" ("movable").

of other road users. All this information is typically used in decision-making and planning modules to derive future actions with a certain look-ahead. This can help to increase the safety of passengers and other road users, especially vulnerable road users.

In fully automated vehicles, drivers will become passengers - meaning that they will not have to actively drive a car but instead be driven by the vehicle. This is why they are expected to dedicate their travel time to other tasks [SS15]: working, reading, sleeping, or browsing the internet will be among them. Partaking in these activities, a passenger's focus on the driving experience itself will fade. This leads to increased comfort requirements - especially for passengers who are prone to motion sickness.

With increased vehicle automation and a high market penetration of such vehicles, new cooperative scenarios will be made possible. These include the formation of vehicle platoons and cooperative intersections, where vehicles entering from different directions are given small alternating time slots to cross without stopping. Both scenarios are aiming for increased traffic flow and reduced energy consumption.

Hence, the primary aims of passenger transport, as stated in the epigraph by American visionary Norman Bel Geddes in 1940, still hold, but with an enhancement. The desire for safety, comfort, speed, and economy nowadays is extended by a desire for sustainability. This leads to the need for technology to either reduce fuel consumption and emissions when looking at vehicles driven by Internal Combustion Engines (ICEs) or to increase efficiency and reduce energy consumption in general for electrical power-trains. In order to achieve these goals, automated vehicles with modular vehicle architectures[2] require control modules that are capable of transforming motion trajectories, desired by some planning module, into actuator commands both for lateral and longitudinal motion.

For the reader to better understand the motivation behind this thesis work, it helps to understand the historical developments of longitudinal vehicle motion control systems leading to today's automated vehicle efforts. Also, we want to draw possible expectations about future trends in this domain. The benefit for the reader to discuss the topics in this chapter more in detail is intended to be two-fold: First, it will help to differentiate between various components described in existing literature related to *longitudinal vehicle motion control*, and understand differences and similarities to the module proposed in this thesis. Second, it should serve to understand the benefits we expect from having a predictive vehicle motion tracking module with improved tracking capability, as will be suggested throughout the remainder of this thesis.

2.1 Longitudinal vehicle motion control - history and future

2.1.1 Brief history of vehicle automation

Long before electronic versions of vehicle controls appeared in literature, the mechanical predecessors of cruise controls could be found as early as 1900. The Wilson-Pilcher vehicle (see Figure 2.1) was equipped with a governor acting on the throttle valve and the spark ignition timing. The Automotor Journal [The04] stated: "By this arrangement, the driver can regulate the normal speed of the engine by varying the strength of the governor-spring .... He is also able to control the engine speed by a foot-pedal .... The strength of this spring can be varied by the driver from the steering pillar, the hand-lever".

A few mechanical speed regulators existed before electronic solutions appeared: The first car that was marketed having an "Auto-Pilot" seems to be the 1958 Chrysler-Imperial [Clu04].

[2] Alternative suggestions like end-to-end architectures will be discussed in Section 2.1.2

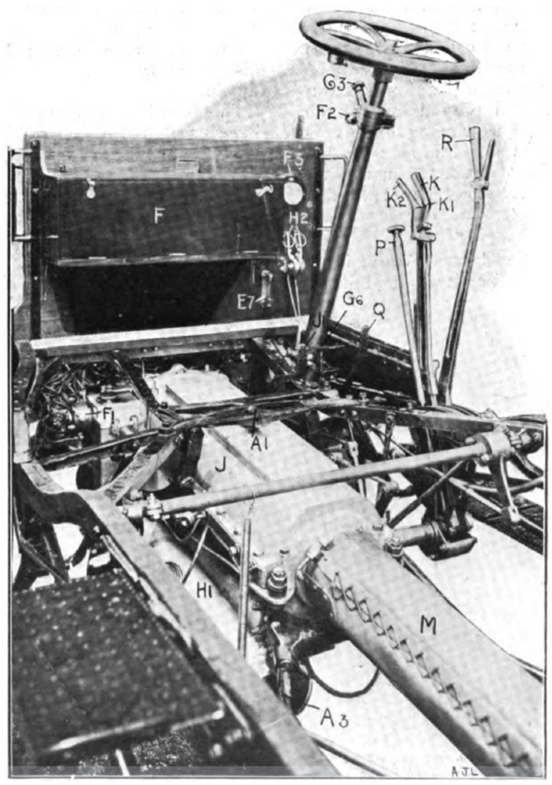

FIG. 6.—Central Portion of the Wilson-Pilcher 6-Cylinder Chassis, showing the Regulating Levers and Pedals, the Gear-Box, and the Dash.

Figure 2.1: A picture from "The Automotor Journal 1904" [The04] showing the speed control lever (G3) of the Wilson-Pilcher vehicle, which was possibly the world's first Cruise Control system. (Copyright: Creative Commons CC0 1.0 Universal)

The system was built on a US patent from 1948 by Ralph R. Teetor, which mentions a "Speed control device for resisting operation of the accelerator" [Tee48]. According to Shladover [Shl95], who described the history of Advanced Vehicle Control Systems (AVCSs), the development of automated longitudinal vehicle control systems is closely related to a vision of Automated Highway Systems (AHSs), which was depicted during the General Motors "Futurama" exhibition held at the 1939-40 New York World Fair. The main character of Futurama, visionary Norman Bell Geddes, stated that to increase safety and comfort, human drivers would have to be eliminated [Nor40]. According to his vision, radio-controlled electric cars should be "... on advanced highways propelled by electromagnetic fields provided by circuits embedded in the roadway". He believed that this would have already happened in his future, 1960. Moreover, it was in the year 1960 that General Motors Corporation published a report [Gar60] including a description of an automatic speed control system "for electronic highways". According to Shladover [Shl95], the system was proven on test tracks, so Geddes' estimation was correct, but the system was far from reaching the maturity necessary for series production. Fenton and Mayan [FM91] reports a long-range program on various aspects of automated highway operations at The Ohio State University between 1964 and 1980, with the principal emphasis on electronic controllers for both longitudinal and lateral vehicle control.

Several sources [Con18; Wik18] list Daniel Aaron Wisner as the inventor of the "Automotive Electronic Cruise Control," who filed two patents on his inventions in the year 1968 [Wis68b; Wis68a]. "The advantage of electronic speed control over its mechanical predecessor was that it could be integrated with electronic accident avoidance and engine management systems”. It is further stated that by 1974, many car manufacturers, for example AMC, GM, Chrysler, and Ford, offered vehicles with an electronic cruise control.

Shladover [Shl95] reports early parallel developments on automatic steering control in the 1960s in the United States, Europe and Japan. In England, experiments were conducted in the late 1960s [Car70], while in Japan, a car followed an inductive cable at a speed of 100 km/h without any driver in 1967 at MITI’s Mechanical Engineering Laboratory [Osh65]. In the 1970s, several projects investigating longitudinal control systems were conducted, such as the Cabinentaxi system in Germany [Hes72] or the ARAMIS system in France [Mau74]. Meanwhile, in Japan, the first tests of visual vehicle control were performed in 1977 at the Mechanical Engineering Laboratory in Tsukuba [TY+79], driving at speeds up to 30 km/h.

Although not in a scientific context, it may inspire many researchers worldwide, including the author of this thesis; the year 1982 has to be mentioned. This year was when television series “Knight Rider” was first screened, introducing the intelligent and self-driving car “KITT”.

Shladover as well as Xiao and Gao [XG10] list the years 1986/87 as the beginning of the "modern" history of advanced vehicle control systems, with the PATH Program (started in 1986) in Berkeley, California and the PROMETHEUS Program (1987 - 1994) in Europe.

Dickmanns [Dic02] mentions 1987 as a milestone in the development of automated vehicles, guided by machine vision, with the presentation of the VaMoRs vehicle of the Universität der Bundeswehr München. The demonstrator could navigate with engaged driving automation (both lateral and longitudinal control) on a free section of the German Autobahn over 20 km at speeds up to almost 100 km/h. As one outcome of the PROMETHEUS program, the concept of ACC was presented by Broqua [Bro91] in the year 1991. Several prototype vehicles were demonstrated during the program. “ViTA” was a modified van by Daimler-Benz, which first demonstrated stop-and-go capability enabled by computer vision in 1991. The automated Mercedes SEL500 sedan vehicles “VaMP” and “VITA2” were showcased at the final demonstration of the PROMETHEUS project in 1994. Until then, “already thousands of kilometers have been driven in regular three-lane traffic including convoy driving and transition into this mode from free-lane driving at speeds up to 130 km/h” [Dic02].

In 1989, neural network-based road following was demonstrated on the test vehicle “ALVINN” - Autonomous Land Vehicle In a Neural Network [Pom89]. The neural network processed camera images and point cloud data from a Lidar to output steering commands.

In 1995, two independent demonstrations showed (partially) automated long-distance highway driving. Carnegie Mellon University (CMU)’s test vehicle “NavLab5” crossed the USA from east to west, but only as Level 1 demonstrator controlling the lateral control, while the longitudinal control was left to the driver. “VaMP” was a Level 2 demonstrator driving 1600 km from Munich to Denmark, with only 5 % intervention by the safety driver.

It was also in 1995 when the first Electronic Stability Control (ESC) systems and Global Navigation Satellite Systems (GNNSs) were introduced in production vehicles. A group around A. Broggi demonstrated their test vehicle “ARGO” in 1998, letting it drive with engaged automation on standard roads [Bro99]. Nine years after the presentation of an ACC system, Mercedes was the first to offer it in a production vehicle as an option called “Distronic” in the year 2000 [Dic02]. It used radar sensors to detect the lead vehicle.

In the Project “Demo 2000” [TK+01], a cooperative driving system for platooning with inter-vehicle communication was showcased. The first parking assist function in a production vehicle was released in 2003 by Toyota in the hybrid passenger car Prius [CNN03]. The year 2004 started another milestone phase in research of automated driving vehicles, when Defense Advanced Research Projects Agency (DARPA) offered a million dollar prize for the winner of the “DARPA Grand Challenge 2004” [DAR04]. None of the 15 finalists were able to complete the difficult desert track. In the renewed “DARPA Grand Challenge 2005”, already

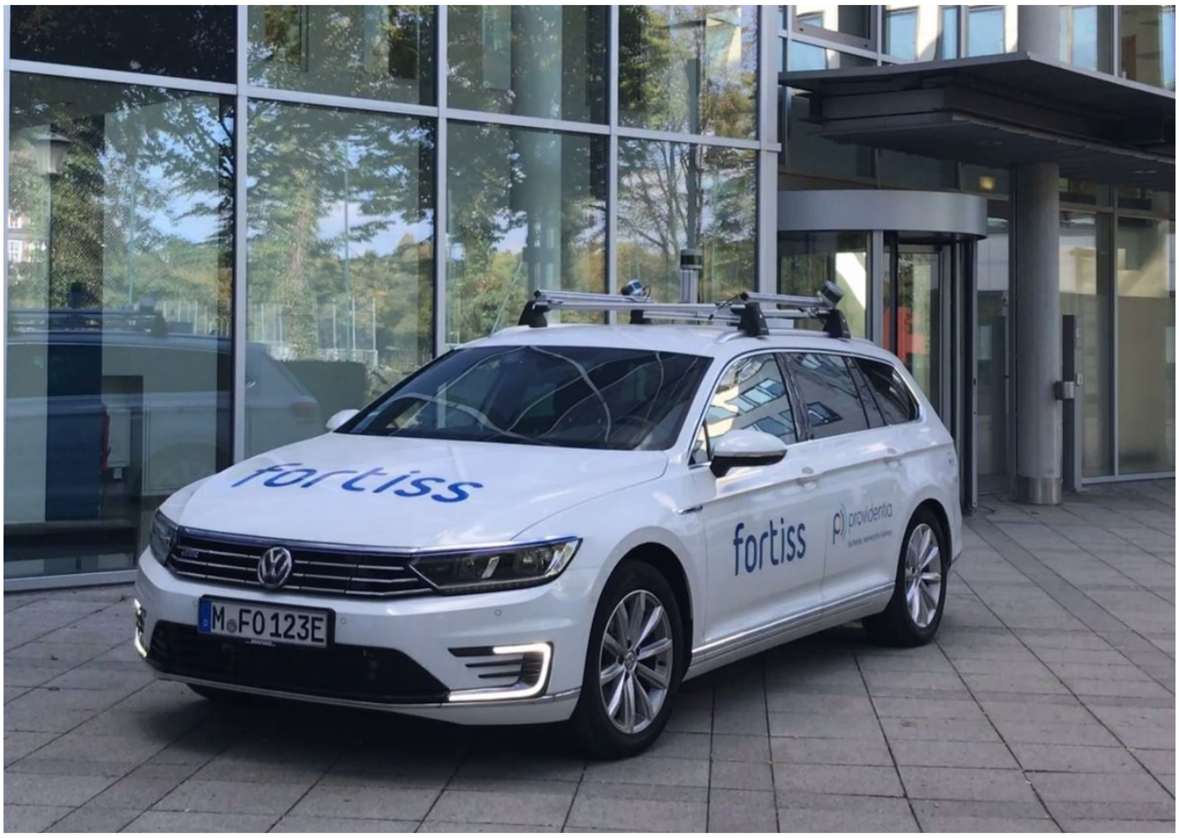

Figure 2.2: *fortuna*, the vehicle prototype built under the author's supervision during this thesis, is an example of a research prototype vehicle equipped with a driving automation system. (Photo by Martin Büchel)

five teams completed the track in the Mojave Desert, and the Stanford Racing Team won the $2 million prize with its vehicle "Stanley" [TM+06].

In 2007, the DARPA Urban challenge was held. Vehicles had to demonstrate capabilities to drive in (non-public) traffic, merge, pass, and cross intersections. The CMU with its vehicle "BOSS" was the winning team, followed by "Junior", Stanford University's vehicle. More information on the vehicles of the DARPA Urban Challenge can be found in [BI+09].

In 2010, A. Broggi's group demonstrated a long-distance ride of four vehicles equipped with a driving automation system from Parma, Italy, to arrive at the 2010 World Expo in Shanghai, China, with almost no human intervention [BC+12a], at the "VISLAB Intercontinental Autonomous Challenge". The SARTRE Project ("Safe Road Trains for the Environment") aimed to push the development of road trains (Platoons) on highways. It was funded by the European Commission and took place between 2009 and 2012. The Grand Cooperative Driving Challenges, held in 2011 and 2016, challenged international teams in a competition to demonstrate highway platooning scenarios. Vehicles were equipped with communication devices to interconnect to show cooperative driving capabilities [PS+12; EC+16].

Commercial development of automated vehicles was made public in 2010, with the foundation of Google's self-driving car project, running at the time of writing under Alphabet's subsidiary company Waymo. Traditional carmakers answered with demonstrations of their capabilities in this field. Also in 2010, a modified Audi TT named "Shelley" climbed the legendary mountain race course "Pikes Peak" at the limits of handling [FT+12; KG12]. Daimler demonstrated the memorial "Bertha Benz Drive", repeating the historic automobile journey from August 1888, but 125 years later, and this time with engaged driving automation system [ZD+13]. BMW followed to publish in 2015 about their testing efforts and experiences gained during the years starting from Spring 2011 [AR+15].

A newcomer to the market, Tesla, released its first version of Level 2 ADAS called "Autopilot" in 2014 [Tes15]. After a few accidents with fatalities, some countries forced Tesla to remove the name "Autopilot" in favor of a name that leads to less expectations of customers concerning safety [Lom]. Audi initially claimed the world's first level 3 production vehicle to be the 2017 Audi A8. The system, named traffic jam pilot, was designed to operate in traffic up to 60 km/h on highways and multi-lane roads with a physical barrier separating the two directions of traffic. Nevertheless, at the time of writing, the company was still waiting for homologation of the ADAS system [Fle18].

Since then, an exploding number of other research institutes, car companies, and newcomers have been actively developing vehicles with driving automation capability during the last decade, and it would be out of the scope of this document to list them all. Nevertheless, we want to mention two prototype vehicles that were created under the author's supervision during his time working at the research institute fortiss: First, the vehicle *fortuna* (see Figure 2.2) is a 5th Generation Telecommunication Standard (5G) connected vehicle with an automated driving system ready to be used for tests on public roads. It is equipped with a prototypical sensor set ready to provide Level 5 capabilities [BS+19]. Second, the electric vehicle prototype of the project Robust and Reliant Automotive Computing Environment for Future eCars (RACE) (see Figure 2.3 served as a demonstrator of a novel vehicle Electric / Electronic (E/E) architecture, capable of delivering fail-operational behavior - a requirement for Level 5 automated driving [BF+15].

2.1.2 Future trends

This section looks into possible future developments arising from increased vehicle automation and discusses the connection of various aspects to the longitudinal vehicle motion tracking module under investigation.

Development towards Level 5 autonomy

Reaching Level 5 autonomy induces that the Operational Desing Domain (ODD) as described in Section 1.6 will be increased to grow to the entire space covered by today's manual vehicles. This also means that a shift away from driving primarily on flat roads as highways will happen, for which a conventional ADAS system was designed. This means that future vehicle control systems must be robust against the influences of high road grades on vehicle dynamics.

Instead of solely designing the control system robust in a reactive manner, we want to go one step further and propose to actively consider these influences by making use of the advance knowledge of a known disturbance fed into the longitudinal controller. Another proposal of this thesis (both already published in [BK16a]) is to design the motion tracking module as a unified interface for the different driving tasks appearing in all possible scenarios. This includes ACC scenarios, platooning, low- and high-speed driving, stop-and-go traffic, parking scenarios, and all other driving tasks appearing on rural roads, highways, and cities, which include smart connected intersections (as described in Section 2.1.2).

Increase of computational in-vehicle capacity

Vehicle automation leads to a high demand for computational power. Additional sensor data processing, calculation of perception algorithms, mission planning, behavioral planning modules, and additional vehicle control functions - all need more computational resources

Figure 2.3: The vehicle prototype of the project RACE, also built under the author's supervision while working on this thesis. It demonstrates a novel vehicle software and E/E architecture for future automated vehicles. (Photo by Martin Büchel)

compared to today's production vehicles. Most of today's prototype demonstrators for automated driving are equipped with various additional computers fitted into the trunk, all with high-performance CPUs and GPUs. Due to the necessity of an increased in-vehicle computing capacity, more resources will be available for novel tasks with a higher degree of automation and longitudinal control functions.

Centralized vehicle architectures

Many research activities highlight the trend towards centralizing E/E architectures of future vehicles [BC+12b; CL+12; SC+13]. With a centralized vehicle architecture, additional information from sensors and other vehicle functions will become available to any other function.

An example of a vehicle demonstrator showing new trends in automotive E/E architectures is the one built under the author's supervision during their time at the research institute fortiss GmbH. It is shown in Figure 2.3, and details can be found in the author's publication [BF+15]. In the future, the control system designer will have fewer restrictions when defining the interfaces than today with current production vehicles.

Connectivity

During the last decade, progress in technology and standardization of Vehicle to-Everything (V2X) communication interfaces can be seen, and this progress will continue. With additional information coming from other vehicles, infrastructure, and a back-end server, automated and connected vehicles can get more information about environmental and road conditions, the precise position of vehicles, and other traffic participants. With the new telecommunications standard 5G, V2X communication will be reliable for the first time, with guaranteed latencies, enabling to build safety-critical functions based on mobile communication. This will give connected vehicles more reliable information about future trajectories of other vehicles. This information can be exploited to plan more reliable trajectories over a longer prediction horizon (see also Section 2.1.2). One example of a 5G connected vehicle prototype is the one presented by the author of this thesis in [BS+19], the vehicle *fortuna* (see Figure 2.2).

Smart infrastructure

Future systems with smart infrastructure are investigated, for example, in the Providentia project [HB+17; HE+17], to which the author of this thesis also contributed during the time of this thesis at the research institute fortiss.

One of the targets of this project was to demonstrate how a vehicle's (limited) field-of-view can be enhanced by smart infrastructure. This was achieved using sensor data coming from stationary equipment placed along a highway and transferred to the vehicle using V2X communication.

In this way, the vehicle's environment model can be enhanced, and traffic participants that might otherwise be occluded will be made visible. Also, it is possible to provide additional predictive data to the vehicles. For example, predictions of future trajectories of other traffic participants can be computed by the computing units of the infrastructure. Hence, more reliable predictions with an increased prediction horizon will be made available (due to extensive computational power in the back-end). Again, a trend is visible here that future vehicles will have access to more precise information over a longer time horizon.

Cooperative driving

Due to enhancements in technology and standardization of V2X communication, future automated vehicles will be enabled to plan cooperatively and interactively. Anticipated planning, considering other nearby vehicles' plans, helps achieve sustainable, more comfortable, and safer trajectories. At large scale, cooperative driving will help to robustly reduce traffic congestion. Two special cases of cooperative driving will be discussed in more detail. Further information about platooning will be mentioned in Section 2.1.2, and smart intersections are discussed in Section 2.1.2

Platooning

Vehicle platoons can potentially increase the road traffic flow rate while decreasing energy consumption and emissions. The main effect of energy consumption reduction is due to the aerodynamic drag reduction. Regarding to [Ala11], a reduction of over 65 % is possible. However, this examination is only valid for inter-vehicle distances of less than 5 meters. In contrast, at 10 m of distance, only a reduction of 40 % is possible, and at 20 m, this value decreases to around 30 %. Further information about vehicle platoon control can be found in [SK+01] and [SP+04].

Shladover stated in [Shl95] that "*Dynamic response of actuation system*s (engine and brake control systems) can be *a serious limitation* to longitudinal control system performance. *Fast response of these subsystems is needed* to ensure that high-precision constant-spacing vehicle follower control systems perform effectively and with asymptotic stability. [...] The *largest technical issue* in the design of regulation-layer controllers *for longitudinal control* is *ensuring robustness with respect to parameter variations, external force loadings, and sensor anomalies* under the full range of real-world operating conditions. This will require extensive experimental work on full-scale vehicles in realistic operating environments."

Only if the low-level longitudinal controller, combined with the actuators, can follow a given trajectory without substantial deviations can the upper-level platoon controllers be tuned less conservatively. Hence, small inter-vehicle distances can be realized. Seiler, Pant, and Hedrick investigated *the negative effect of the time constant* and dead-times *of the longitudinal acceleration control* in [SP+04] on the relative vehicle distance and fuel consumption. *They identified that these effects pose fundamental limitations to longitudinal vehicle platoon control*.

Although one can expect that some progress has been made regarding actuator dynamics by improving the electro-mechanical components, limitations still exist. With physical hardware involved, the possibility of reducing time constants and delays within the actuators is limited. A predictive acceleration controller, partly able to compensate for these limitations, is an indirect enabler to reduce inter-vehicle distances of platoons and, hence, fuel consumption.

Smart intersections

Various researchers have proposed smart intersections for automated vehicles. A centralized controller, as in Tachet, Santi, and Sobolevsky [TS+16], Au, Zhang, and Stone [AZ+15] or Wuthishuwong and Traechtler [WT13], could be used to enable traffic crossing from both sides of an intersection simultaneously without stopping. In order to realize such a future scenario, time slots are assigned to automated vehicles, and they have to pass the intersection at precisely this time slot. The strategies above assume that deviations from the planned trajectories are minor or not present at all. If, in reality, deviations would be greater than a defined tolerance, this would lead to fatal outcomes. Hence, the throughput through an intersection directly depends on the longitudinal variance. This, in return, requires that, for optimal intersection throughput, each vehicle's motion tracker can operate under the restriction of small tracking error tolerances.

Mobility as a Service

The trend known under the term Mobility-as-a-Service (MaaS) will potentially lead to more automated micro-buses on the roads, with people frequently hopping on and off the vehicle. This means that one can expect a very high variance in the vehicle's gross mass, acting as a disturbance to a longitudinal motion tracking module.

Artificial intelligence and end-to-end architectures

In the past, various End-to-End solutions, probably starting with [Pom89], until more recent attempts [BD+16; MS17a] were demonstrated as an alternative to rather classical approaches following the well-known “Sense-Plan-Act” scheme (as also depicted in Figure 1.1).

End-to-end approaches use raw sensor data (mainly camera images) and train deep neural networks that directly act on the vehicle's actuators. Currently, these approaches lack the possibility to integrate a desired behavior, like, for example, following traffic rules, and are hard to debug. This makes it challenging to fulfill the safety requirements of automated vehicles. Although it is impressive that the approach already works for simple lane-keeping scenarios, how such approaches can be scaled to more complex traffic scenarios is currently under investigation. We want to believe that although data-centric approaches will become very important, some kind of modular, hierarchical software structure will be present in future automated vehicles. Very recent developments published by the company Wayve AI [KH+18] underpin this opinion, as they stated that they use a rather “conventional” low-lever actuator control module while standing in general for an automated driving solution based on an end-to-end approach. This means that a longitudinal motion tracking module will still play an important role when shifting to rather data-centric approaches.

2.2 Impact of longitudinal motion tracking on automated vehicles

In this section, we want to discuss the impact of longitudinal motion tracking on automated vehicles. We want to analyze the benefit of a high-precision motion tracking module by looking at the following aspects: safety, comfort, speed, economy, efficiency, and sustainability.

Safety

Clearly, any undesired longitudinal motion of an automated vehicle endangers the safety of passengers and other road users and has to be avoided at all times. This could be unexpected breaking for no apparent reason, leading to a following vehicle crash into the ego-vehicle. Undesired acceleration could cause accidents by running into obstacles or other road users. Especially when looking at scenarios of vehicle platooning and smart intersections (as depicted by Sections 2.1.2 and 2.1.2), it is evident these can only be safely realized, when strong guarantees can be given that a vehicle's true trajectory will only deviate from the desired one by some sufficiently small error margins. While violations of these safety margins might even cause vehicles to crash, it is also highly desirable to realize very small error tolerances, as otherwise, these scenarios cannot be realized to their full potential.

Comfort

In automated vehicles, drivers will become passengers, and therefore, they can engage in other activities, like sleeping, reading, working, writing text messages, or browsing the internet. A study performed at the University of Michigan [SS15] concluded that "37 % of drivers will engage in activities that increase the severity and frequency of motion sickness". Further, "6-12 % would experience moderate or severe motion sickness while being a passenger of a self-driving car".

Hence, the importance of ride comfort will rise with the degree of vehicle automation. The motion tracking control unit is responsible for the quality of the vehicle's motion, for example, by reducing jerks that the passenger can feel. Therefore, the motion-tracking module will have an increased influence on this subjective perception.

For this reason, we want to give some more profound insights into how physical properties and their comfort ranges can define a comfortable ride. Elbanhawi [Elb15] investigated ride comfort measures in automated cars and concluded that traditional factors for comfort, like air quality, sound and noise, temperature, and vibrations, will be removed by the factors of naturality, disturbances, apparent safety, and motion sickness in automated vehicles. Naturality, the sense of being driven by a human driver, is influenced by lateral and longitudinal vehicle motion. Some unwanted effects caused by a control module, like oscillations, are causing an unnatural feeling. Elbanhawi mentions that disturbances, following their taxonomy, can be both road disturbances (causing vertical vibrations) or load disturbances (causing horizontal deviations). The latter has to be compensated by longitudinal control. While the safety aspect mentioned by Elbanhawi is discussed in Section 2.2, we want to discuss the effect of motion sickness in more detail. Schaedler [Sch17] compared several studies on the influencing factors for motion sickness and came to the conclusion that not only lateral acceleration but also longitudinal acceleration has an impact on motion sickness, and Vogel, Kohlhaas, and Baumgarten [VK+82] conducted experiments giving this evidence. Schaedler further analyzed the physical quantities which can be directly related to causing discomfort. The main goal for a comfortable ride is to minimize forces acting on the passenger, mainly caused by acceleration and its time derivative jerk. For acceleration, a discomfort threshold of around $1\,\mathrm{m/s^2}$ to $1.5\ \mathrm{m/s^2}$ is suggested, while for jerk, a maximum value of $3\ \mathrm{m/s^3}$ is suggested. [ES15] defines the jerk threshold for discomfort at 0.5 to $0.9\,\mathrm{m/s^3}$.

The influence of higher derivatives like snap, crackle, and pop seem less well understood, while Schaedler proposes reducing the values. [ES15] further suggests reducing the frequency content of acceleration, as horizontal acceleration in the range of 0.1 - 0.5 Hz seems to have a negative effect, causing motion sickness.

Speed

In the sense of Norman Bel Geddes, who we quoted at the beginning of this chapter, we can define the increased speed of a journey as a reduction of the time needed to drive from a starting point to a target in traffic. We can look at this evaluation criteria as a road traffic throughput that can be achieved. Reducing inter-vehicle distances, for example, with platooning when driving on roads and highways, in smart intersections, or by cooperative driving maneuvers - all these scenarios help to increase speed.

We already discussed in Section 2.1.2 that an improved longitudinal control quality is an enabler for these intelligent transportation systems, mainly due to improved error tolerances. We can most likely achieve this by an advanced motion tracking controller, with better reference tracking capability, and by a compensation of the "parasitic actuator delays and lags", as termed by [XD+08] in the context of string stability of vehicle platoons.

Efficiency

Overshoots and oscillations around the motion reference trajectories can be seen as directly linked to the tracking performance of a longitudinal vehicle motion tracker. In a vehicle, this means that kinetic energy will have to be produced during phases of overshoots, which cannot be fully recuperated during phases of under-shoots. This will immediately lead to decreased energy efficiency. Hence, improving the tracking performance of a motion-tracking module will automatically lead to improved energy efficiency.

We will later present the results of a simulation study that suggests a potential of up to around three percent in fuel consumption reduction when comparing a predictive controller to a conventional PI controller.

Economy

The fuel or energy consumption reductions mentioned in the previous subsection 2.2 will help to reduce ride costs. Reduced travel times, as mentioned in Section 2.2, will also impact the ride costs. Another potential source of reduction is if inter-vehicle distances can be reduced while platooning. Hence, tracking control quality directly and indirectly has the potential to reduce travel costs.

Sustainability

European emission standards, for example, limit various exhaust gases, namely carbon monoxide (CO), hydrocarbon (THC), nitrogen oxides (NOx), and particulates, for vehicles with internal combustion engines.

It would be clearly out of scope in this thesis to sufficiently derive and explain the effects that lead to high emission values in an ICE engine. From the author's professional working experience in combustion engines, it can be stated that dynamic transitions often cause certain emissions to peak. Hence, a driver's personal driving style has an important effect on exhaust gas emissions in manual vehicles. Anticipating the torque which needs to be produced in the future helps to operate a combustion engine at low emission levels.

In automated vehicles, emission reductions are possible because of two effects: First, the vehicle defines its “driving style”, meaning that highly dynamic driving patterns can be avoided by constraining desired trajectories calculated in a planning module. Including such constraints will most likely be an engineering target to maintain certain passenger comfort. However, a second reason is that the trajectories from the planning module are known with a certain look-ahead, which can be exploited both within the motion tracking module and also in the actuator control module - which, in the case of a combustion engine, will be the engine control unit. This is one of the reasons for this thesis to suggest using a predictive motion tracking module, which can also forward the desired torque values to a power-train control unit. After this suggestion was published in the author’s work [BK16a] and [BK16b], a study performed by Ma and Wang [MW19] recently could demonstrate NOx reductions of 13 % in the investigated driving cycle when using model predictive engine control, which could be easily combined with the proposed tracking module.

Realizing this potential would not be possible with a conventional, acceleration set-point-based low-level controller, as probably still can be found in most of today’s vehicles equipped with ADAS systems.

2.3 Summary

We can summarize the following trends arising from a future shift of vehicle automation levels towards SAE Level 5. First, increased longitudinal motion tracking accuracy will play a crucial role in enabling scenarios like vehicle platooning and smart intersections to reach their full potential. Predictive motion tracking has the potential to compensate or even eliminate “parasitic delays and lags” in the context of vehicle platooning. Different from typical operation domains of conventional ADAS systems, when driving on roads with high road slopes, the forces acting on a fully automated vehicle cannot be neglected. Nevertheless, one can expect that future road slope information will be available to the vehicle. Hence, the availability of advance information can be exploited for improved vehicle motion tracking within automated vehicles.

Further, including predictive information in a motion tracking module is an enabler to reduce energy consumption, and there is potential to reduce emissions in ICE-driven vehicles when predictive control of actuators is realized. For example, up to 13 % of NOx reductions could recently be demonstrated with predictive engine control on known target velocity profiles [MW19]. Increased passenger comfort is a distinctive feature for automated vehicles, and it is directly impacted by motion tracking performance. The use of advanced predictive control methods, which are computationally more expensive, is enabled by increased computational resources and centralized information systems, which we can expect to be available in future vehicles.

3 System Description

"If you can't explain it simply, you don't understand it well enough."

– Albert Einstein

We want to derive the vehicle dynamics by building the equilibrium of forces acting on the vehicle, and enhance the equations by including the power-train dynamics. A similar derivation can be found in [Raj11], but we additionally want to include the terms from resulting forces due to road grade changes and wind effects.

The following assumptions are made: The influence of power-train elasticity and backlash effects is small and we neglect it in the following analysis. The effect of tire slip is neglected, since it is small during normal operation, where we assume our planning module will plan trajectories far away from physical limits. It is assumed that the effect of power losses during gear shifts is compensated by a power-train control unit and therefore only the case of a transmission in steady state is considered.

3.1 Longitudinal vehicle dynamics

A sketch of the longitudinal forces acting on the vehicle is given in Figure 3.1. Applying Newton's second law of motion and building the dynamic equilibrium of forces, one will get:

$$m \cdot \dot{v} = F_{\mathrm{tire,f}} + F_{\mathrm{tire,r}} - F_{\mathrm{pitch}} - F_{\mathrm{aero}} - F_{\mathrm{r,f}} - F_{\mathrm{r,r}}, \tag{3.1}$$

with vehicle mass m and vehicle speed v. $F_{\mathrm{tire,f}}$ and $F_{\mathrm{tire,r}}$ are the tire forces in longitudinal direction of front and rear tires, F_{pitch} is the resulting gravity force, F_{aero} is the aerodynamic drag force, and $F_{\mathrm{r,f}}$, $F_{\mathrm{r,r}}$ are the rolling resistance forces on the wheels.

3.1.1 Gravity force component

F_{pitch} is the resulting gravity force on the vehicle due to a time varying road slope $\varphi(t)$, and the gravity constant g:

$$F_{\mathrm{pitch}} = m \cdot \mathrm{g} \cdot \sin\varphi(t). \tag{3.2}$$

Please note that the terms road slope and road grade are used interchangeable within this thesis, being identical to the *road plane elevation angle* as defined per ISO8855 [ISO11].

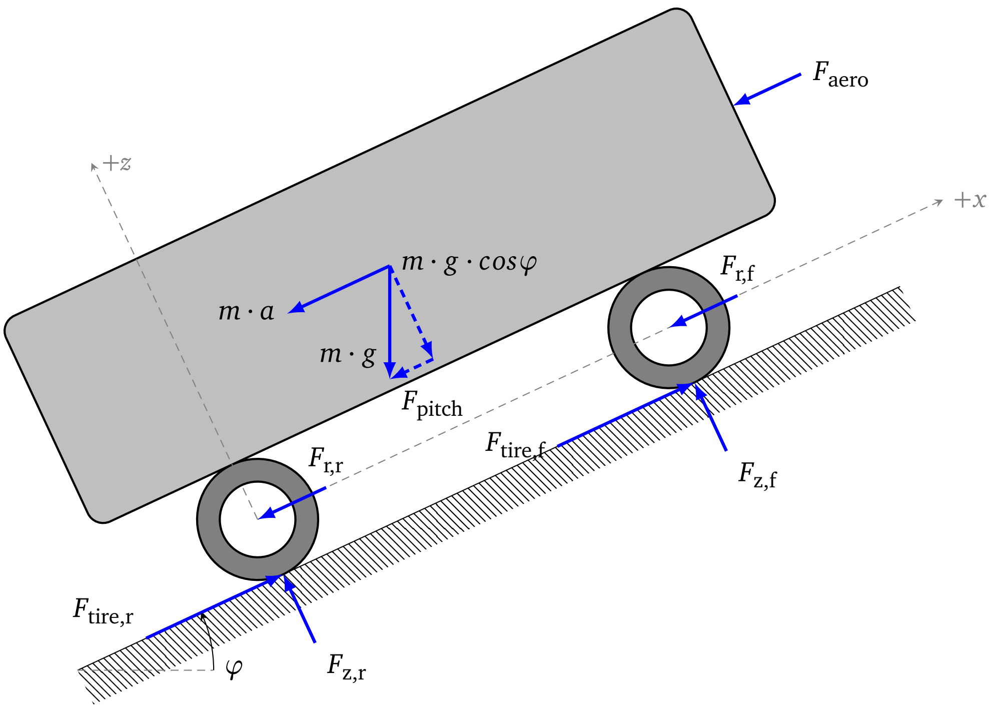

Figure 3.1: Longitudinal vehicle forces.

3.1.2 Aerodynamic drag

The aerodynamic drag force in longitudinal direction F_{aero} was reported to be modeled as

$$F_{\mathrm{aero}} = \frac{1}{2} \cdot \rho_{\mathrm{air}} \cdot c_{\mathrm{x}}(\psi_{\mathrm{a}}) \cdot A_{\mathrm{v}}(\psi_{\mathrm{a}}) \cdot v_{\mathrm{a}}^2, \tag{3.3}$$

with air density ρ_{air}, the dimensionless longitudinal aerodynamic drag coefficient c_{x}, the frontal area of the vehicle A_{v} and the air approach velocity v_{a} [Hak18]. $A_{\mathrm{v}}(\psi_{\mathrm{a}})$ is also often denoted cross-sectional area and, together with $c_{\mathrm{x}}(\psi_{\mathrm{a}})$, a function of the air approach angle ψ_{a}. Air density is often calculated by the relation stated by the general gas equation $\rho_{\mathrm{air}} = \frac{p_{\mathrm{a}}}{R_{\mathrm{air}} T_{\mathrm{amb}}}$ under the assumption of air being an ideal gas. In this equation, p_{a} and T_{amb} are ambient pressure and temperature, respectively, while $R_{\mathrm{air}} = 287.058 \ [Jkg^{-1}K^{-1}]$ is the specific gas constant for idealized air. Measurement values are typically provided by an ambient pressure and ambient temperature sensor, at least in vehicles with combustion engines. We can lump air density, drag coefficient and frontal area into the parameter:

$$C_{\mathrm{d}} = \frac{1}{2} \cdot \frac{p_{\mathrm{a}}}{R_{\mathrm{air}} T_{\mathrm{amb}}} \cdot c_{\mathrm{x}} \cdot A_{\mathrm{v}}. \tag{3.4}$$

If we define v_{wind} as the longitudinal wind speed in forward direction of the vehicle, we can rewrite for the aerodynamic drag force:

$$F_{\mathrm{aero}} = C_{\mathrm{d}} \cdot \mathrm{sign}\,(v - v_{\mathrm{wind}}) \cdot (v - v_{\mathrm{wind}})^2 \,. \tag{3.5}$$

A more detailed discussion of the influence of side-wind components on the longitudinal aerodynamic drag coefficient c_x is out of scope of this thesis. More detailed information and empirical data can be found in [Hak18], p. 154f. Instead, we want to highlight the fact that the lumped aerodynamic drag coefficient C_{d} is hence an *unkown, time varying* parameter.

3.1.3 Rolling resistance

$F_{\mathrm{r,f}}$ and $F_{\mathrm{r,r}}$ are the rolling resistance forces of front and rear wheels. As in [Raj11], we approximate these forces by a linear correlation between the coefficient of rolling resistance C_{r} and the (lumped) vertical tire force F_{z}, and write for rear and front wheels together:

$$F_{\mathrm{r,f}} + F_{\mathrm{r,r}} = F_{\mathrm{r}} = C_{\mathrm{r}} F_{\mathrm{z}} = C_{\mathrm{r}} \cdot m \cdot \mathrm{g} \cdot \cos\varphi. \tag{3.6}$$

Considering that rolling resistance forces are always directed against the vehicle's moving direction, this results with vehicle speed v in:

$$F_{\mathrm{r,f}} + F_{\mathrm{r,r}} = F_{\mathrm{r}} = C_{\mathrm{r}} F_{\mathrm{z}} = C_{\mathrm{r}} \cdot \mathrm{sign}(v) \cdot m \cdot \mathrm{g} \cdot \cos\varphi. \tag{3.7}$$

Often, C_{r} is regarded constant, which is a simplification which does not hold in practice. More sophisticated models include other, nonlinear dependencies, e.g. on vertical tire force, tire pressure, tire temperature and vehicle speed (see [Hak18] pp. 137). For driving on paved roads, the following polynomial dependency has been modeled [RH95]:

$$C_{\mathrm{r}}(v) = c_{\mathrm{r}_0} + c_{\mathrm{r}_1} v + c_{\mathrm{r}_4} v^4, \tag{3.8}$$

while the curve was found to be approximately linear at lower vehicle speeds of up to around 80 km/h [Hak18].

3.1.4 Toe-in resistance

Haken [Hak18] notes that for stability and comfort reasons, wheels are typically symmetrically pointing towards the center axis of the vehicle at a small toe-in angle $\delta_{\mathrm{w}} \approx \alpha_{\mathrm{w}} \ll 1$ which is approximately the same as the side slip angle α_{w}. This causes longitudinal forces depending on the lateral force F_{s} according to

$$F_{\mathrm{tr}} = F_{\mathrm{s}} \cdot sin(\alpha_{\mathrm{w}}). \tag{3.9}$$

Typically, the toe-in resistance can be neglected in relation to the rolling resistance [Hak18], p. 145.

3.1.5 Cornering resistance

While driving around corners at a radius r_{v}, the centrifugal force $F_{\mathrm{c}} = m a_{\mathrm{y}} = \frac{1}{r_{\mathrm{v}}} m v^2$ acts on the center of gravity. This needs to be compensated by lateral forces on the wheels, which cause longitudinal resistance components. Under rather strong simplifications, these can be modeled by the single track model, often also named bicycle model. Then, one can approximately state for the cornering resistance F_{cr} [Hak18], p. 147:

$$F_{\mathrm{cr}} = \frac{m^2 v_{\mathrm{x}}^4}{4 r_{\mathrm{v}}^2 C_{\mathrm{s}}}, \tag{3.10}$$

using the cornering stiffness C_{s}. Haken [Hak18] mentions that the cornering resistance is hence quadratic in the lateral acceleration value and has magnitudes which are approximately the same as the rolling resistance at values of 4 ms^{-1} lateral acceleration and a curve radius r_{v} of 100 m. See the reference [Hak18], p. 147 for more details on the simplifying assumptions.

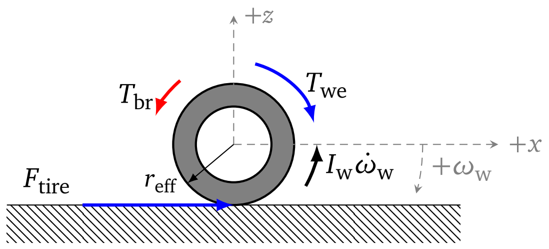

Figure 3.2: Rotational forces and torques acting on the wheel.

3.1.6 Wheel dynamics

Wheel forces can be controlled by inducing torques to the wheel. This is realized by torques created trough a vehicles power-train as well as by friction brakes. Wheel dynamics is described by the equilibrium of forces and moments acting on the wheel (see Figure 3.2):

$$I_{\text{w}} \cdot \dot{\omega}_{\text{w}} = T_{\text{we}} - T_{\text{br}} - r_{\text{eff}} \cdot F_{\text{tire}}, \tag{3.11}$$

with I_{w} being the lumped inertia of all wheels including drive shaft, differential and gearbox elements rotating with the wheel, T_{we} the torque at the wheel induced by the engine, the braking torque T_{br} and the effective wheel radius r_{eff}. Please note that “engine” is used here interchangeable with “power-train” as a torque producing unit in case of more complex power-train configurations, as with hybrid vehicles. Equation (3.11) can be transformed into:

$$F_{\text{tire}} = \frac{1}{r_{\text{eff}}}(T_{\text{we}} - T_{\text{br}} - I_{\text{w}} \cdot \dot{\omega}_{\text{w}}). \tag{3.12}$$

With the relation between angular velocity of the wheel and the linear vehicle velocity, assuming no slip occurs:

$$\omega_{\text{w}} = \frac{v}{r_{\text{eff}}}. \tag{3.13}$$

Building the time derivative of (3.13) one gets

$$\dot{\omega}_{\text{w}} = \frac{\dot{v}}{r_{\text{eff}}}, \tag{3.14}$$

and by inserting (3.14) into (3.12), one gets the equation for the tire forces as

$$F_{\text{tire}} = \frac{1}{r_{\text{eff}}} \cdot (T_{\text{we}} - T_{\text{br}}) - \frac{I_{\text{w}}}{r_{\text{eff}}^2} \cdot \dot{v}. \tag{3.15}$$

We can simplify above equation (3.15), for readability reasons, by introducing the “mass equivalent” resulting from the (lumped) wheel inertia m_{I} as

$$m_{\text{I}} \doteq \frac{I_{\text{w}}}{r_{\text{eff}}^2}. \tag{3.16}$$

Note that the brake torque can only act in direction against wheel rotation. Hence, if we want to define F_{tire} for both positive and negative vehicle speeds, we need to rewrite (3.15) as

$$F_{\text{tire}} = \frac{1}{r_{\text{eff}}} \cdot (T_{\text{we}} - \text{sign}(v) T_{\text{br}}) - m_{\text{I}} \cdot \dot{v}. \tag{3.17}$$

3.1.7 Resulting equation for longitudinal vehicle motion

Hence, the resulting vehicle dynamics can be described condensing the sum of front and rear tire forces to F_{tire}, as:

$$m \cdot \dot{v} = F_{\mathrm{tire}} - m \cdot g \cdot (\sin\varphi + C_{\mathrm{r}} \cdot \mathrm{sign}(v) \cdot \cos\varphi) - C_{\mathrm{d}} \cdot \mathrm{sign}\,(v - v_{\mathrm{wind}}) \cdot (v - v_{\mathrm{wind}})^2 . \qquad (3.18)$$

Inserting (3.17) into (3.18), one obtains

$$(m + m_{\mathrm{I}}) \cdot \dot{v} = \frac{T_{\mathrm{we}} - \mathrm{sign}(v) T_{\mathrm{br}}}{r_{\mathrm{eff}}} - m \cdot g \cdot (\sin\varphi + C_{\mathrm{r}} \cdot \mathrm{sign}(v) \cdot \cos\varphi) - C_{\mathrm{d}} \cdot \mathrm{sign}\,(v - v_{\mathrm{wind}}) \cdot (v - v_{\mathrm{wind}})^2 . \qquad (3.19)$$

3.2 Power-train and brake dynamics

We derive the power-train equations for a conventional power-train as depicted in Figure 3.3. It consists of an IC engine or an electric motor, a torque converter, a gearbox, a differential, and wheels with brakes. Since we are interested in the sum of all forces acting on the vehicle instead of individual wheels, we omit the fact that the torques are distributed to the wheels. Instead, we reduce all wheels to a single wheel which produces the sum of all forces as a result from the gearbox output torque, and include the inertia of a rotational parts rotating with wheel speed ω_{w}.

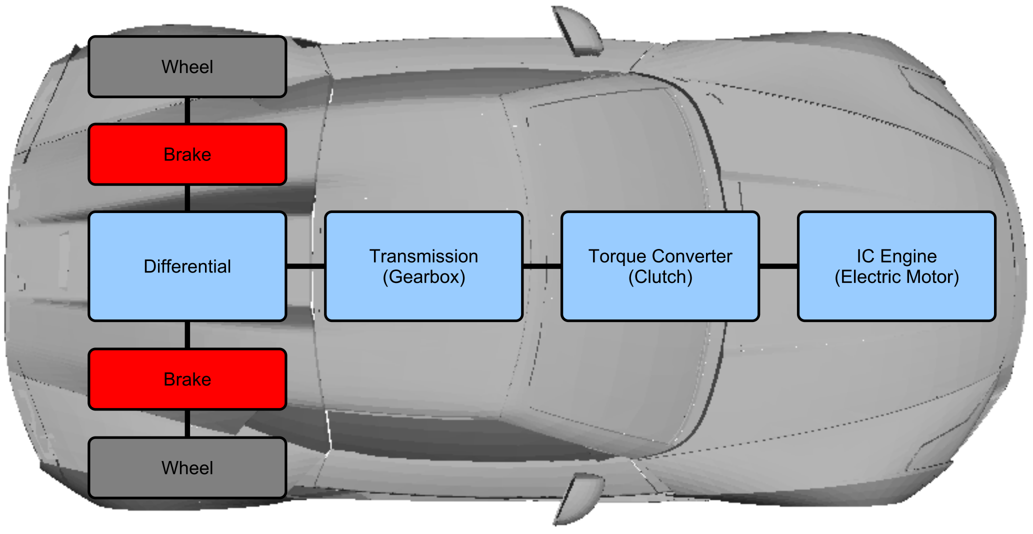

Figure 3.3: Schematic overview of typical components of a single engine driven power-train

3.2.1 Gearbox

For simplification, we neglect drive shaft elasticity, backlash effects and dynamics during gear shift. With this assumptions, introducing the gear ratio $i_{\mathrm{g}} = \omega_{\mathrm{g}}/\omega_{\mathrm{w}}$ and the transmission

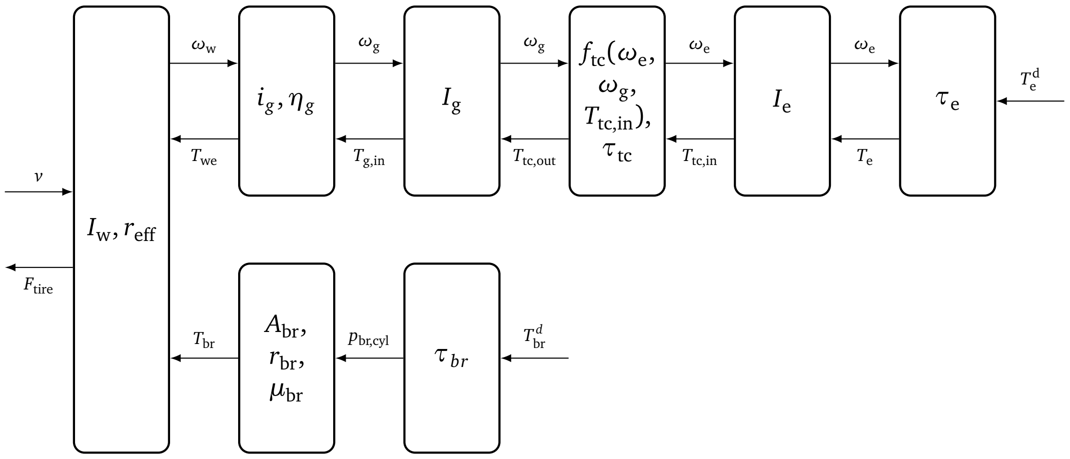

Figure 3.4: Block diagram of power-train model.

efficiency η_g, we have the following relation between torque into and out of the gearbox and hence the equations for the transmission are given with (3.14) as

$$T_{we} = \eta_g \cdot i_g \cdot T_{g,in} \tag{3.20}$$

$$\omega_g = i_g \cdot \omega_w = \frac{i_g}{r_{eff}} \cdot v. \tag{3.21}$$

We insert this back into the wheel equation (3.17) to get

$$F_{tire} = \frac{1}{r_{eff}} \cdot (\eta_g \cdot i_g \cdot T_{g,in} - T_{br}) - m_I \cdot \dot{v}. \tag{3.22}$$

3.2.2 Gearbox inertia

With the total inertia I_g of all parts necessarily rotating with the gearbox rotational speed ω_g, we can write

$$I_g \cdot \dot{\omega}_g = T_{tc,out} - T_{g,in}.$$

Solving for $T_{g,in}$ we obtain

$$T_{g,in} = T_{tc,out} - i_g \cdot \frac{I_g}{r_{eff}} \dot{v}.$$

We can insert this back into (3.20) to get

$$T_{we} = \eta_g \cdot i_g \cdot T_{tc,out} - \eta_g \cdot i_g^2 \cdot \frac{I_g}{r_{eff}} \dot{v}. \tag{3.23}$$

We can insert this back into (3.22) to get

$$F_{tire} = \frac{1}{r_{eff}} \cdot \left(\eta_g \cdot i_g \cdot (T_{tc,out} - i_g \cdot \frac{I_g}{r_{eff}} \dot{v}) - T_{br} \right) - m_I \cdot \dot{v}. \tag{3.24}$$

which is equivalent to

$$F_{tire} \cdot r_{eff} = \eta_g \cdot i_g \cdot T_{tc,out} - \frac{1}{r_{eff}} (\eta_g \cdot i_g^2 \cdot I_g + I_w) \dot{v} - T_{br}. \tag{3.25}$$

We introduce $I_{w,g}$ as

$$I_{w,g} = \eta_g \cdot i_g^2 \cdot I_g + I_w, \tag{3.26}$$

to simplify (3.25) to

$$F_{\mathrm{tire}} = \frac{1}{r_{\mathrm{eff}}} \cdot \left(\eta_{\mathrm{g}} \cdot i_{\mathrm{g}} \cdot T_{\mathrm{tc,out}} - T_{\mathrm{br}}\right) - \frac{I_{\mathrm{w,g}}}{r_{\mathrm{eff}}^2} \cdot \dot{v}. \tag{3.27}$$

3.2.3 Torque converter

A torque converter is an essential part of an automatic power-train. It allows to prevent an ICE from stalling when the vehicle is at standstill or at very low speed. Building high-fidelity models which are suited for real-time control applications is an ongoing research topic [Lee17; AM12; Li16], as well as finding a methodology for the parameter identification process of these models [MS17b]. [Lee17] proposed a model based on four first-order nonlinear differential equations to approximate the dynamic behavior of the hydrodynamic effects. A widely uses model for control purposes is the dynamic model of Kotwicki [Kot82]. It is a zero order model, describing the relation of the input torque to the torque converter $T_{\mathrm{tc,in}}$, to the output torque $T_{\mathrm{tc,out}}$, as a nonlinear function of

$$T_{\mathrm{tc,out}} = f_{\mathrm{tc}}(\omega_{\mathrm{g}}, \omega_{\mathrm{e}}, T_{\mathrm{tc,in}}) = f_{\mathrm{tc}}(i_{\mathrm{g}}\omega_{\mathrm{w}}, \omega_{\mathrm{e}}, T_{\mathrm{tc,in}}) = f_{\mathrm{tc}}(\frac{i_{\mathrm{g}}}{r_{\mathrm{eff}}}v, \omega_{\mathrm{e}}, T_{\mathrm{tc,in}}). \tag{3.28}$$

Another model suitable for real-time simulation is given in [SAE00]. It describes the behavior of a torque converter in the following way:

$$T_{\mathrm{tc,out}} = T_{\mathrm{tc,in}} \cdot R_{\mathrm{tc,tq}}(i_{\mathrm{tc}}), \tag{3.29}$$

with R_{tc} being a nonlinear function of the speed ratio $i_{\mathrm{tc}} = \frac{\omega_{\mathrm{g}}}{\omega_{\mathrm{e}}}$. The speed ratio follows the equation

$$\tau_{\mathrm{tc,in}}\dot{T}_{\mathrm{tc,in}} + T_{\mathrm{tc,in}} = sgn(1 - i_{\mathrm{tc}})(\frac{\omega_{\mathrm{e}}}{K(i_{\mathrm{tc}})})^2, \tag{3.30}$$

with the time constant $\tau_{\mathrm{tc,in}}$ and the capacity factor $K(i_{\mathrm{tc}})$ also being a nonlinear function (typically approximated by a look-up table) of the speed ratio i_{tc}. In order to optimize for simulation performance, the recommendation is to set the time constant to zero to neglect the transient behavior.

We can summarize these findings to express the turbine torque as a function of the impeller torque given by the following equations:

$$\overline{T}_{\mathrm{tc,out}} = f_{\mathrm{tc}}(\frac{i_{\mathrm{g}}}{r_{\mathrm{eff}}}v, \omega_{\mathrm{e}}, T_{\mathrm{tc,in}}) \tag{3.31}$$

$$\tau_{\mathrm{tc}}\dot{T}_{\mathrm{tc,out}} + T_{\mathrm{tc,out}} = \overline{T}_{\mathrm{tc,out}} \tag{3.32}$$

$$\omega_{\mathrm{e}} = g_{\mathrm{tc}}(\omega_{\mathrm{g}}, T_{\mathrm{tc,out}}). \tag{3.33}$$

In case of a torque converter in normal operation, we can write for the torque induced by the engine at the wheel:

$$T_{\mathrm{we}} = \eta_{\mathrm{g}} \cdot i_{\mathrm{g}} \cdot T_{\mathrm{tc,in}} \cdot R_{\mathrm{tc,tq}}(i_{\mathrm{tc}}) - \eta_{\mathrm{g}} \cdot i_{\mathrm{g}}^2 \cdot \frac{I_{\mathrm{g}}}{r_{\mathrm{eff}}} \cdot \dot{v}. \tag{3.34}$$

In the case of a locked torque converter, it will be that

$$T_{\mathrm{tc,out}} = T_{\mathrm{tc,in}}, \tag{3.35}$$

and

$$\omega_\mathrm{e} = \omega_\mathrm{g} = i_\mathrm{g} \cdot \omega_\mathrm{w} = \frac{i_\mathrm{g}}{r_\mathrm{eff}} \cdot v, \tag{3.36}$$

and we can write for the tire force:

$$F_\mathrm{tire} = \frac{1}{r_\mathrm{eff}} \cdot \left(\eta_\mathrm{g} \cdot i_\mathrm{g} \cdot T_\mathrm{tc,in} - T_\mathrm{br}\right) - \frac{I_\mathrm{w,g}}{r_\mathrm{eff}^2} \cdot \dot{v}, \tag{3.37}$$

and for the wheel torque induced by the power-train:

$$T_\mathrm{we} = \eta_\mathrm{g} \cdot i_\mathrm{g} \cdot T_\mathrm{tc,in} - \eta_\mathrm{g} \cdot i_\mathrm{g}^2 \cdot \frac{I_\mathrm{g}}{r_\mathrm{eff}} \cdot \dot{v}. \tag{3.38}$$

3.2.4 Engine inertia

The engine dynamics can be derived from the equilibrium of moments

$$I_\mathrm{e} \cdot \dot{\omega}_\mathrm{e} = T_\mathrm{e} - T_\mathrm{tc,in}, \tag{3.39}$$

with the engine inertia I_e, the engine speed ω_e, the engine net output torque after losses T_e and T_tc, which represents the load on the engine from the torque converter. We can transform this to

$$T_\mathrm{tc,in} = T_\mathrm{e} - I_\mathrm{e} \cdot \dot{\omega}_\mathrm{e}. \tag{3.40}$$

Then, we can write for the torque at the wheel induced by the engine:

$$T_\mathrm{we} = \eta_\mathrm{g} i_\mathrm{g} \cdot (T_\mathrm{e} - I_\mathrm{e} \cdot \dot{\omega}_\mathrm{e}) \cdot R_\mathrm{tc,tq}(i_\mathrm{tc}) - \eta_\mathrm{g} i_\mathrm{g}^2 \cdot \frac{I_\mathrm{g}}{r_\mathrm{eff}} \cdot \dot{v} \tag{3.41}$$

$$T_\mathrm{we} = \eta_\mathrm{g} i_\mathrm{g} \cdot R_\mathrm{tc,tq}(i_\mathrm{tc}) \cdot T_\mathrm{e} - \eta_\mathrm{g} i_\mathrm{g} \cdot R_\mathrm{tc,tq}(i_\mathrm{tc}) I_\mathrm{e} \dot{\omega}_\mathrm{e} - \eta_\mathrm{g} i_\mathrm{g}^2 \cdot \frac{I_\mathrm{g}}{r_\mathrm{eff}} \cdot \dot{v} \tag{3.42}$$

With $\dot{\omega}_\mathrm{e} = \dot{\omega}_g / i_\mathrm{tc} = \frac{i_\mathrm{g}}{i_\mathrm{tc}} \frac{\dot{v}}{r_\mathrm{eff}}$ this yields

$$T_\mathrm{we} = \eta_\mathrm{g} i_\mathrm{g} R_\mathrm{tc,tq}(i_\mathrm{tc}) \cdot T_\mathrm{e} - \eta_\mathrm{g} i_\mathrm{g}^2 \left(\frac{R_\mathrm{tc,tq}(i_\mathrm{tc})}{i_\mathrm{tc}} I_\mathrm{e} + I_\mathrm{g}\right) \frac{1}{r_\mathrm{eff}} \cdot \dot{v}. \tag{3.43}$$

For the case of a locked torque converter this is simplified to

$$T_\mathrm{we} = \eta_\mathrm{g} i_\mathrm{g} \cdot T_\mathrm{e} - \eta_\mathrm{g} i_\mathrm{g}^2 \cdot \left(I_\mathrm{e} + I_\mathrm{g}\right) \frac{1}{r_\mathrm{eff}} \cdot \dot{v}. \tag{3.44}$$

In any case, the wheel torque is of the form:

$$T_\mathrm{we} = \frac{1}{\alpha(\cdot)} T_\mathrm{e} - \beta(\cdot) \dot{v}, \tag{3.45}$$

where $\alpha(\cdot) = 1 / \left(\eta_\mathrm{g} i_\mathrm{g} R_\mathrm{tc,tq}(i_\mathrm{tc})\right)$ is a function of the speed ratios and the power-train efficiency, and $\beta(\cdot) = \eta_\mathrm{g} i_\mathrm{g}^2 \left(\frac{R_\mathrm{tc,tq}(i_\mathrm{tc})}{i_\mathrm{tc}} I_\mathrm{e} + I_\mathrm{g}\right) \frac{1}{r_\mathrm{eff}}$ is additionally dependent on the power-train inertia. From above equation, we can see that the wheel torque induced by the engine is reduced by a term dependent on the vehicle acceleration $\dot{v} = a$. Inverting (3.45) we get

$$T_\mathrm{e} = \alpha(\cdot) T_\mathrm{we} + \alpha(\cdot) \beta(\cdot) \dot{v}. \tag{3.46}$$

3.2.5 Engine torque response

The engine net torque T_e results from a torque request T_e^d to the engine control module. Torque build up is delayed by various effects. First, looking at ICEs, actuator delays of throttle, valves, Exhaust Gas Recirculation (EGR) and waste-gate actuators exist. Second, air flow effects through the intake manifold system of an ICE have to be considered. Third, within Engine Control Module (ECM), various limiting functions exist which are necessary to meet emission regulations and have an effect on the dynamic engine response. A comprehensive description of the effects is out of scope of this document. The interested reader is referred to Rajamani [Raj11].

As an approximation, we can model the torque build up with a transfer function including first-order dynamic behavior, similar as in [SK99; Dav13], with the time constant τ_e for engine dynamics:

$$\tau_\mathrm{e}\dot{T}_\mathrm{e} + T_\mathrm{e} = T_\mathrm{e}^\mathrm{d} \tag{3.47}$$

$$\dot{T}_\mathrm{e} = \frac{1}{\tau_\mathrm{e}}\left(T_\mathrm{e}^\mathrm{d} - T_\mathrm{e}\right). \tag{3.48}$$

Typical time constants are in the range of $\tau_\mathrm{e} = 0.15\,\mathrm{s}$ to $0.4\,\mathrm{s}$ regarding to [May99; Ise06; XD+08]. Xiao, Darbha, and Gao also highlight the effect of these "parasitic delays and lags" [XD+08] to string stability of vehicle platoons. The engine torque is limited between engine speed dependent values of a drag torque and a maximal engine torque:

$$T_\mathrm{drag}(\omega_\mathrm{e}) < T_\mathrm{e} < T_\mathrm{e}^\mathrm{max}(\omega_\mathrm{e}). \tag{3.49}$$

3.2.6 Power-train torque response

If we want to see how the engine torque response relates to the response of the torque induced by the engine at the wheel, we can investigate the following. From (3.46) we can get by differentiation using the product rule:

$$\dot{T}_\mathrm{e} = \dot{\alpha}(\cdot)T_\mathrm{we} + \alpha(\cdot)\dot{T}_\mathrm{we} + \dot{\alpha}(\cdot)\beta(\cdot)\dot{v} + \alpha(\cdot)\dot{\beta}(\cdot)\dot{v} + \alpha(\cdot)\beta(\cdot)\ddot{v}, \tag{3.50}$$

and inserting the above into (3.47) and some algebra, we get

$$\frac{\tau_\mathrm{e}\alpha}{\alpha + \tau_\mathrm{e}\dot{\alpha}}\dot{T}_\mathrm{we} + T_\mathrm{we} = \frac{T_\mathrm{e} - \tilde{\alpha}\dot{v} - \tilde{\beta}\ddot{v}}{\alpha + \tau_\mathrm{e}\dot{\alpha}}, \tag{3.51}$$

where we introduced $\tilde{\alpha} = \tau_\mathrm{e}\dot{\alpha}\beta + \tau_\mathrm{e}\alpha\dot{\beta} + \alpha\beta$ and $\tilde{\beta} = \tau_\mathrm{e}\alpha\beta$ for readability. By defining

$$\tau_\mathrm{pwt} = \frac{\tau_\mathrm{e}\alpha}{\alpha + \tau_\mathrm{e}\dot{\alpha}}, \tag{3.52}$$

and

$$T_\mathrm{we}^\mathrm{d} = \frac{T_\mathrm{e} - \tilde{\alpha}\dot{v} - \tilde{\beta}\ddot{v}}{\alpha + \tau_\mathrm{e}\dot{\alpha}}, \tag{3.53}$$

we can state

$$\tau_\mathrm{pwt}\dot{T}_\mathrm{we} + T_\mathrm{we} = T_\mathrm{we}^\mathrm{d}. \tag{3.54}$$

From here we can see that the power-train torque also follows a first order differential equation (assuming the engine torque response is given by the same). Note, that in general, the time constant τ_{pwt} is time variant. An exception is the case of a locked torque converter, were we have both $\dot{\alpha} = 0$ and $\dot{\beta} = 0$, and the formulae above results in

$$\tau_{\text{e}} \dot{T}_{\text{we}} + T_{\text{we}} = \frac{T_{\text{e}} - \beta \dot{v} - \tau_{\text{e}} \beta \ddot{v}}{\alpha} = T_{\text{we}}^{\text{d}}. \tag{3.55}$$

At this point, it is clear that the interplay between the engine torque and wheel torque induced it is quite complex, especially for an active torque converter. For the rest of the thesis, we will assume that the power-train dynamics can be sufficiently described by a first order dynamics with a time-invariant power-train time constant τ_{pwt}. If this is chosen to be a worst case value, we can be assume that engine demand torque values calculated from a longitudinal vehicle controller which outputs a wheel torque demand value will be in a feasible range. We do not treat the problem of correctly modeling the torque converter behavior or of calculating a suitable engine input value. We rather assume that such a controller is in place which correctly controls a demanded wheel torque, for example by the inverse relation resulting from the definition of T_{we}^{d} in (3.53) which we can rewrite as:

$$T_{\text{e}} = T_{\text{we}}^{\text{d}} (\alpha + \tau_{\text{e}} \dot{\alpha}) + \tilde{\alpha} \dot{v} + \tilde{\beta} \ddot{v}. \tag{3.56}$$

3.2.7 Brake torque

Simple models of the brake torque T_{br} as a linear function of the brake pressure acting on the brake cylinder were proposed, for example [DO+02]:

$$T_{\text{br}} = A_{\text{br}} \cdot r_{\text{br}} \cdot \mu_{\text{br}} \cdot p_{\text{br}}, \quad p_{\text{br}} \geq 0, \tag{3.57}$$

with the effective brake pad area A_{br}, the effective brake disc radius r_{br}, the brake friction coefficient μ_{br} and the brake pressure p_{br}. These models neglect nonlinearity and temperature dependence. The brake torque can only have positive values, acting against the rotational direction of the wheel. [GW+97] modeled the dynamic behavior of the effective brake pressure in dependence of the brake cylinder pressure $p_{\text{br,cyl}}$ with the following first order dynamics:

$$\tau_{\text{br}} \dot{p}_{\text{br}} + p_{\text{br}} = p_{\text{br,cyl}}, \tag{3.58}$$

with time constants τ_{br} between 50 ms and 250 ms.

A more elaborate model was proposed by Martinez, Velenis, et al. in [MV+15]. She proposed a method to estimate the brake torque values from wheel speed sensor measurements. It includes a model of the temperature behavior of the brake to take the temperature dependence of the friction coefficient into account. The caliper pressure is modeled as a second order under-damped system with first order delays.

For the remainder of this thesis, we want to assume that a brake controller is available which is capable of estimating the brake parameters well enough to provide a torque based interface. Then, we can model the brake torque T_{br} dependent on a request T_{br}^{d} to the brake controller, approximated as a response with first order dynamics, with time constant τ_{br} as

$$\tau_{\text{br}} \dot{T}_{\text{br}} + T_{\text{br}} = T_{\text{br}}^{d}. \tag{3.59}$$

Regarding constraints, we assume the brake torque is limited between time invariant limits given as:

$$0 \leq T_{\text{br}} \leq T_{\text{br}}^{\text{max}}. \tag{3.60}$$

3.3 Summary of longitudinal vehicle equations

We want to revisit the equation describing longitudinal vehicle dynamics from Section 3.1.7. We can restate (3.19), by making $\dot{v}$ explicit, as:

$$\dot{v} = \frac{T_{\mathrm{we}} - \mathrm{sign}(v) T_{\mathrm{br}}}{(m + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{mg}{m + m_{\mathrm{I}}} \cdot (\sin\varphi + C_{\mathrm{r}} \cdot \mathrm{sign}(v) \cdot \cos\varphi) \\ - \frac{C_{\mathrm{d}}}{m + m_{\mathrm{I}}} \cdot \mathrm{sign}\,(v - v_{\mathrm{wind}}) \cdot (v - v_{\mathrm{wind}})^2 . \quad (3.61)$$

If we additionally restate the power-train and brake torque dynamics from (3.54) and (3.59), respectively, we wan summarize the following system of ordinary differential equations, describing the longitudinal vehicle motion *including power-train and brake dynamics*:

$$\dot{v} = \frac{T_{\mathrm{we}} - \mathrm{sign}(v) T_{\mathrm{br}}}{(m + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{mg}{m + m_{\mathrm{I}}} \cdot (\sin\varphi + C_{\mathrm{r}} \cdot \mathrm{sign}(v) \cdot \cos\varphi) \\ - \frac{C_{\mathrm{d}}}{m + m_{\mathrm{I}}} \cdot \mathrm{sign}\,(v - v_{\mathrm{wind}}) \cdot (v - v_{\mathrm{wind}})^2 \quad (3.62a)$$

$$\dot{T}_{\mathrm{we}} = (-)\frac{1}{\tau_{\mathrm{pwt}}} T_{\mathrm{we}} + \frac{1}{\tau_{\mathrm{pwt}}} T^{d}_{\mathrm{we}} \quad (3.62b)$$

$$\dot{T}_{\mathrm{br}} = (-)\frac{1}{\tau_{\mathrm{br}}} T_{\mathrm{br}} + \frac{1}{\tau_{\mathrm{br}}} T^{d}_{\mathrm{br}} . \quad (3.62c)$$

3.4 Discussion of vehicle dynamics equations

In the following, we want to discuss the quantities affecting longitudinal vehicle motion more in detail.

3.4.1 Effective wheel radius

Due to dynamic forces acting on the wheels, the effective wheel radius r_{eff} will increase compared to that of an un-deformed wheel at higher rotational speeds. It is assumed that the effective wheel radius can be obtained using an approach similar to [CG05]. Nevertheless, for simplification in the dynamic equations of a prediction model, we will assume that it can be treated as time invariant within short prediction horizons.

3.4.2 Wheel inertia

We previously defined the “mass equivalent” resulting from the lumped wheel inertia I_{w} in dependency of the effective wheel radius in (3.16) as $m_{\mathrm{I}} \doteq \frac{I_{\mathrm{w}}}{r^2_{\mathrm{eff}}}$. From this dependency it is clear that the assumptions about the effective wheel radius also apply for the mass equivalent m_{I}. The wheel inertia of one single mounted wheel of a passenger car has been found to lie in a range between approximately 0.6 and 1.3 kg/m^2 in a study performed in Metz, Akouris, Agney, and Clark [MA+90]. Also, slight differences were found to exist between new and worn tires. This leads to values of m_{I} for a four-wheeled passenger car, assuming a wheel radius of 0.3 m, in a range of about 25 to 60 kg. Nevertheless, in this study, we will treat the wheel inertia and the resulting mass equivalent as known and time invariant.

3.4.3 Vehicle mass

Changes in the vehicle mass result in almost direct proportional change in the resulting wheel torque needed to achieve a given vehicle acceleration. Due to a changing number of passengers and a changing vehicle load, the gross vehicle mass is known to vary. For example, an existing passenger car known to have a high load capacity, like a Ford Mondeo 1.6 TDCi ECOnetic [Webc], has a net weight of 1515 kg, and can carry up to 2690 kg including a carrier [Webb], which results in a additional mass of over ±75 % compared to the empty vehicle. For heavy duty vehicles with a net weight of around 10.000 kg and a gross weight of 40.000 kg [Weba], the resulting mass increase is even up to ±400 %. While changes of vehicle mass due to modifications in the load or the passenger number usually might be expected to happen during standstill of the vehicle, the vehicle mass decreases additionally due to fuel consumption effects in presence of internal combustion engines. Nevertheless, it might be also of interest to detect changes due to the vehicle loosing freight.

In order to calculate a correct wheel torque demand value in a model-based feed-forward controller, an estimation of the vehicle mass is necessary. If parameter errors are present, this will lead to deviations in the desired vehicle acceleration. These errors have to be compensated by a longitudinal controller.

3.4.4 Road plane elevation angle

A high road slope φ results in a longitudinal vehicle force which is proportional to the vehicle mass, as could be seen from (3.2). We assume the road slope φ is available the controller of an automated vehicle with a certain look-ahead. This information can be either made available from high-definition maps in combination with a precise localization module, or by the vehicle's sensors, or a combination of these.

Previous literature (see Chapter 4 and Section 7.3) often assumed that the road slope angle is not available from measurements and some work suggested estimators for this parameter. As will be seen there, the forces resulting from changes in road slope were mostly neglected in previous literature and treated as disturbance to the control system.

3.4.5 Rolling resistance

The rolling resistance coefficient C_{r} is only constant for a certain combination of tire and road surface, and at a constant tire pressure. Typical values for C_{r} have been reported to be in the range of around 0.013 - 0.015 for passenger cars on dry asphalt surfaces according to [WA+04; Won08].

Different tire pressures, tire wear or switching to a different tire (e.g. winter tires) may result in a change in rolling resistance of up to ±60 % [XG10].

Hence, one has to expect that the rolling resistance coefficient will be changing when driving on changing road surface conditions. Errors resulting from a wrong coefficient value in a model based control scheme will either have to be treated as a disturbance, or efforts have to be made to compensate by estimating the rolling resistance and treating it as time-variant value $C_{\mathrm{r}}(t)$.

3.4.6 Aerodynamic drag

The aerodynamic drag coefficient C_d of a given vehicle will vary when changes to the vehicle's surface are present. Additional loads on the roof, like for example roof boxes or sensors mounted to the vehicle, the impact to the vehicle's aerodynamics when pulling a trailer, aerodynamic effects resulting from side-winds and also open windows will have an influence on its value. A high variation of the coefficient is given in Platooning scenarios, where a reduction of the aerodynamic drag value is desired by design. Typical values of the *nominal* aerodynamic drag area $c_\mathrm{w}A_\mathrm{v}$ as in (3.4) are between 0.25 and 2.45 m^2 [Wik20; Huc08]. At a given air density of 1.2 kg/m^2 this would result in lumped drag coefficients C_d of between 0.15 and 1.5, with most passenger cars at around 0.4 to 0.65.

Regarding modifications of the vehicle's aerodynamic shape (for example due to carrying goods on a rooftop or pulling a caravan), Hucho reports values of $c_\mathrm{w} \cdot A_\mathrm{v}$ from 0.87 (0.48) m^2 up to 3.24 (3.11) m^2 ([Huc08] p. 258, values in brackets are from a second car type). This results in a factor of up to 6.5 the nominal value. A bicycle mounted on a rooftop was reported to double the nominal drag coefficient (Hucho [Huc08], p. 337).

Investigations of the aerodynamic behavior of vehicle platoons showed that fuel reductions to values of less than 0.9 % of their nominal values can be realized (see [Huc08], Figure 4.136. Unfortunately, no data for distances less than 3 meters is given). In general, the average aerodynamic platoon resistance is reduced with decreasing inter-vehicle distances. Nevertheless, highly nonlinear effects due to turbulence in the air flow occur at very low distances. This resulted in experiments (Ewald 1984) to a reduction of the average c_w values to around 75% due to a reduction of the vehicle distances down to 0.5 car length in a platoon on 10 vehicles. Lowering the distances further down *increased* the reduction at first to above 80%, falling down to the minimum achieved at zero distance of around 55% of the nominal values (given with $c_\mathrm{w} = 0.17$). Alam [Ala11] reported reductions of up to 70 % of the c_w value for the following vehicle, and up to 15 % for the lead vehicle. While the data reported from Ewald 1984 with passenger cars suggest that fuel consumption can be reduced at lower distances, an optimal distance of around 9 meters was suggested by the Promote-Chaffeur project (see [Huc08], p 679). This could be because only two vehicle were considered in their measurements, and a profound discussion would exceed the scope of this work. In any case, the results suggest that precise longitudinal control is important from two perspectives: either to maintain the "sweetspot" at half a vehicle length to avoid turbulent behavior - or to further reduce the aerodynamic drag by, possibly, another 20 %.

Hence, the possibility of considering predictive models of expected air drag changes in a longitudinal control framework might be essential to safely reduce the inter-vehicle distances into the zone of turbulent behavior.

3.4.7 Experimental determination of rolling resistance and aerodynamic drag

For a given setup and for reference value determination, values of both rolling resistance coefficient C_r and aerodynamic drag constant C_d can be obtained by performing coast down tests for which the vehicle speed trajectory is measured after letting a vehicle roll with detached power-train until it stops. Due to wind disturbances, these test are performed in two directions on a plane surface to compensate for wind forces and road slope disturbances. Methodologies are described in [HK+85] or [DJ+09].

3.4.8 Acceleration and velocity

We assume that the acceleration signal (corrected for bias, but noisy) is known from an accelerometer. The velocity is available from wheel speed sensors. In order to be used within a control loop, both signals need to be filtered.

3.4.9 Power-train torque generation and estimation

Torque generation

Let us restate relation (3.56) of engine torque value one needs in order to produce the demanded wheel torque induced at the wheel, which we derived in Section 3.2.6, and rewrite it in a more generic fashion:

$$T_{\mathrm{e}} = T_{\mathrm{we}}^{\mathrm{d}}(\alpha + \tau_{\mathrm{e}}\dot{\alpha}) + \tilde{\alpha}\dot{v} + \tilde{\beta}\ddot{v}.$$

The second and third terms in this equation are vehicle acceleration and jerk dependent values arising from power-train inertia. This means that if we want to correctly compensate for this inertia by means of feed-forward control, it is mandatory to tell a power-train control unit the desired time derivatives of the vehicle speed. This information is implicitly contained in vehicle speed trajectory information (since future values are included by definition). Hence, if we design a predictive longitudinal trajectory tracking module, we can sufficiently serve a power-train module with feed-forward control information.

Torque estimation

We assume that the wheel torque is not directly accessible from a measurement system, since torque sensors are usually omitted in passenger vehicles due to cost constraints and robustness targets. Nevertheless, we can assume that an estimation of this quantity is available from the engine and power-train control unit [WS+18a]. During power-train development, precise torque measurements from engine and power-train dynamometers are used in order to calibrate integrated torque estimation functions within the ECM, the Hybrid Control Unit (HCU) and the Transmission Control Unit (TCU) [BC+07]. These functions give on-line estimations of the engine or wheel torque. Especially the determination of the torque created by ICEs is a non-trivial task. The ECM provides an estimation of the engines indicated torque and torque losses as a multidimensional function of various engine parameters. Depending on the type of engine, these parameters could be relative load, air/fuel ratio, spark advance, intake and exhaust valve timings, valve lift and number of active cylinders for SI engines. In Diesel engines, the resulting torque is mainly determined by the injection quantity and start of combustion, which is given by the injection pattern of many pre-, main- and post injections. These parametric torque models are calibrated for each engine with high effort on a stationary engine test bed, collecting measurement data in the whole variation space over weeks. The typical accuracy achieved by an EMS torque structure is around 5-10 % of the real torque measured on an engine test bed [BC+07; LB+05].

In the case of electric engines, the indicated torque is proportional to the measured electrical current and hence a torque estimation is also available in vehicles with electric or hybrid power-trains. Additionally to the torque estimations, we assume that within the Power-train Control Module (PCM), the value of desired power-train torque demand builds the interface to the power-train control unit. For this reason, the author proposed in [BK16a] to include the power-train torque estimations from the ECM or PCM as a virtual measurement value within the longitudinal vehicle motion tracking module, or to be more specific, in a state and parameter estimator which aids the controller. More details will be given in chapters 6 and 7.

3.5 System analysis

In this section, we provide an analysis, or rather various different reformulations of the longitudinal vehicle motion model. These formulations will be used in different context within the upcoming chapters of this thesis, and depending on the structure of these formulations, different methods can be applied. This includes a generic model for a forward moving vehicle in Section 3.5.1, systems of Ordinary Differential Equations (ODEs) which will be used for control and estimation purposes in Sections 3.5.2 and Section 3.5.4 for their time discrete counterparts, a system description which allows for simulation purposed including standstill in Section 3.5.3, as well as a formulation as an explicit algebraic equation which is Linear-in-Parameters in Section 3.5.5.

3.5.1 Generic formulations of longitudinal vehicle dynamics

In order to get a better understanding of the nonlinear vehicle dynamics we want to rewrite the equations from Section 3.3 in different generic forms. Depending on the purpose, these expressions will be specifically useful. We start by investigating the dynamics as a scalar algebraic equation, before writing the equations in the common state space form. This can lead to a system of ODEs or Differential-algebraic equations (DAEs), which we can write in vector form.

To make it concise with existing literature mostly from the field of control systems, we introduce a state vector which we denote x. Unless stated otherwise, we interpret as $x = (v \quad T_{\mathrm{we}} \quad T_{\mathrm{br}})^{\mathsf{T}}$. Also, a control vector $u = (T_{\mathrm{we}}^{d} \quad T_{\mathrm{br}}^{d})^{\mathsf{T}}$ will be used, where d denotes the desired values as input to the power-train control unit. Further, we have the the disturbance $\varphi(\cdot)$, where $(\cdot)$ denotes a dependency which is not yet specified any further. During this discussion, we will mostly limit our investigations for the moment to the case of a forward moving vehicle where $v(t) > 0$ to make the presentation easier.

3.5.2 Simplified motion dynamic equations

In (3.61) we expressed the vehicle motion dynamics as ordinary differential equation, dependent on the torques from power-train and brakes as well as the wind speed. If we neglect the (typically unknown) wind speed as $v_{\mathrm{wind}} = 0$, we get:

$$\dot{v} = \frac{T_{\mathrm{we}} - \mathrm{sign}(v) T_{\mathrm{br}}}{(m + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{mg}{m + m_{\mathrm{I}}} (\sin\varphi + C_{\mathrm{r}} \mathrm{sign}(v) \cos\varphi) - \frac{C_{\mathrm{d}}}{m + m_{\mathrm{I}}} \mathrm{sign}(v)\, v^2. \tag{3.63}$$

If we further distinguish forward and backward vehicle motion this can be split into the following equations:

$$\dot{v} = \frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{(m + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{mg}{m + m_{\mathrm{I}}} (\sin\varphi + C_{\mathrm{r}} \cos\varphi) - \frac{C_{\mathrm{d}}}{m + m_{\mathrm{I}}} \cdot v^2, \qquad \forall v \geq 0, \tag{3.64}$$

and

$$\dot{v} = \frac{T_{\mathrm{we}} + T_{\mathrm{br}}}{(m + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{mg}{m + m_{\mathrm{I}}} (\sin\varphi - C_{\mathrm{r}} \cos\varphi) + \frac{C_{\mathrm{d}}}{m + m_{\mathrm{I}}} \cdot v^2, \qquad \forall v < 0. \tag{3.65}$$

Restricting vehicle speed to positive, forward motion $v \geq 0$, we can hence summarize a simplified system for longitudinal vehicle motion dynamics, derived from (3.62),

$$\dot{v} = \frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{(m + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{mg}{m + m_{\mathrm{I}}} (\sin\varphi + C_{\mathrm{r}} \cos\varphi) - \frac{C_{\mathrm{d}}}{m + m_{\mathrm{I}}} \cdot v^2 \tag{3.66a}$$

$$\dot{T}_{\mathrm{we}} = \frac{-1}{\tau_{\mathrm{pwt}}} T_{\mathrm{we}} + \frac{1}{\tau_{\mathrm{pwt}}} T_{\mathrm{we}}^{d} \tag{3.66b}$$

$$\dot{T}_{\mathrm{br}} = \frac{-1}{\tau_{\mathrm{br}}} T_{\mathrm{br}} + \frac{1}{\tau_{\mathrm{br}}} T_{\mathrm{br}}^{d}. \tag{3.66c}$$

We want to also write this in vector form as follows :

$$\begin{pmatrix} \dot{v} \\ \dot{T}_{\mathrm{we}} \\ \dot{T}_{\mathrm{br}} \end{pmatrix} = \begin{pmatrix} \frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{(m + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{mg}{m + m_{\mathrm{I}}} (\sin\varphi + C_{\mathrm{r}} \cos\varphi) - \frac{C_{\mathrm{d}}}{m + m_{\mathrm{I}}} \cdot v^2 \\ \frac{-1}{\tau_{\mathrm{pwt}}} T_{\mathrm{we}} + \frac{1}{\tau_{\mathrm{pwt}}} T_{\mathrm{we}}^{d} \\ \frac{-1}{\tau_{\mathrm{br}}} T_{\mathrm{br}} + \frac{1}{\tau_{\mathrm{br}}} T_{\mathrm{br}}^{d} \end{pmatrix}. \tag{3.67}$$

In above system, actuator delays for engine and brake are included as first order dynamics.

System of ordinary differential equations

We can write the longitudinal vehicle dynamics as a system of nonlinear explicit ODEs of the form

$$\dot{x}(t) = f\,(x(t), u(t), d(t, \cdot)) \tag{3.68a}$$

$$y(t) = h(x(t)), \tag{3.68b}$$

with the state vector $x(t)$, the input vector $u(t)$ and the output vector $y(t)$. The time varying disturbance function $d\,(t, \cdot)$ is assumed to be known. From (3.67) which stated the longitudinal vehicle equation in vector form, we can rewrite:

$$\dot{x}(t) = \begin{pmatrix} \frac{1}{(m+m_{\mathrm{I}})\, r_{\mathrm{eff}}} x_2 + \frac{-1}{(m+m_{\mathrm{I}})\, r_{\mathrm{eff}}} x_3 + \frac{-C_{\mathrm{d}}}{m+m_{\mathrm{I}}} \cdot {x_1}^2 + \frac{-mg}{m+m_{\mathrm{I}}} (\sin d + C_{\mathrm{r}} \cos d) \\ \frac{-1}{\tau_{\mathrm{pwt}}} x_2 + \frac{1}{\tau_{\mathrm{pwt}}} u_1 \\ \frac{-1}{\tau_{\mathrm{br}}} x_3 + \frac{1}{\tau_{\mathrm{br}}} u_2. \end{pmatrix} \tag{3.69a}$$

$$y(t) = x(t) = \begin{pmatrix} v & T_{\mathrm{we}} & T_{\mathrm{br}} \end{pmatrix}^{\mathsf{T}}. \tag{3.69b}$$

System of semi-explicit differential algebraic equations

A different way to write the system dynamics equations of longitudinal vehicle motion is that of a system of semi-explicit DAEs. This allows to include algebraic states in addition to a differential state vector $x(t)$, which in a standard state space form has to appear in its differential form $\dot{x}(t)$ on the left hand side of the state space equation. A system of semi-explicit DAEs of is typically written in the general form

$$\dot{x} = f(x, z, u, d) \tag{3.70a}$$

$$0 = g(x, z, u, d) \tag{3.70b}$$

$$y = h(x, z). \tag{3.70c}$$

with the differential state vector $x(t)$, and a nonlinear state dynamics function f, which is also dependent on the algebraic state vector $z(t)$. The nonlinear function g is an algebraic equation, by definition not depending on any differentials. As usual, we have the input vector $u(t)$ and the known disturbance $d(t)$. The output vector $y(t)$ is a function h of both differential states as well as algebraic states.

Viewing the system dynamics as system of semi-explicit DAEs makes it possible, in a state estimation scheme, to also include algebraic states as additional measurements. In the case of longitudinal vehicle motion, in our case the vehicle acceleration, which is available as (noisy) measurement observation, it might be useful to view the dynamics

In order to write the longitudinal vehicle dynamics equations in this fashion, we will now directly rewrite it in a less general form than (3.70a), additionally exploiting linear relationships within the general nonlinear functions f and g. With some algebra, (7.23) can be brought into a form in which the state dynamics is linear in differential states, algebraic states, inputs and disturbances - and the algebraic equation is further linear in parameters. Additionally, the measurements are linear in the states (algebraic and differential). The system just described has the general form

$$\dot{x} = Ax + Cz + Bu + Dd \tag{3.71a}$$

$$0 = \theta^{\mathsf{T}} G(x,z,d) \tag{3.71b}$$

$$y = H\begin{pmatrix} x \\ z \end{pmatrix}, \tag{3.71c}$$

while in our case it will be that $D = 0$. Above, G is a nonlinear measurement matrix of states, algebraic states, control input and disturbance, and θ is a parameter vector. Considering that $x_1 = \dot{v} = a = z$ and using (3.69a) we can write:

$$\dot{x} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\frac{1}{\tau_{\text{pwt}}} & 0 \\ 0 & 0 & -\frac{1}{\tau_{\text{br}}} \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} z + \begin{pmatrix} 0 & 0 \\ \frac{1}{\tau_{\text{pwt}}} & 0 \\ 0 & \frac{1}{\tau_{\text{br}}} \end{pmatrix} u \tag{3.72a}$$

$$0 = \begin{pmatrix} -C_{\text{d}} & \frac{1}{r_{\text{eff}}} & -\frac{1}{r_{\text{eff}}} & -(m+m_{\text{I}}) & -mg & -mgC_{\text{r}} \end{pmatrix} \begin{pmatrix} {x_1}^2 \\ x_2 \\ x_3 \\ z \\ \sin d \\ \cos d \end{pmatrix} \tag{3.72b}$$

$$y = H\begin{pmatrix} x \\ z \end{pmatrix}. \tag{3.72c}$$

Partitioned system of semi-explicit differential algebraic equations

We can also re-write (3.72) in a way which emphasizes the fact that the state vector contains states which will always follow the inputs, according to the actuator dynamics, and states which we really want to control. We do this by partitioning the state and write:

$$\dot{x} = \begin{pmatrix} \dot{x}^s \\ \dot{x}^u \end{pmatrix} = \begin{pmatrix} f^s(x^s,z,d) \\ f^u(x^u,u) \end{pmatrix} \tag{3.73a}$$

$$0 = g(x^s,x^u,z,d,\theta) \tag{3.73b}$$

$$y = h(x^s,z). \tag{3.73c}$$

In case of the longitudinal vehicle dynamics, this yields a system of semi-explicit DAEs in the form:

$$\dot{x}^s = f^s(x^s, z) \tag{3.74a}$$

$$\dot{x}^u = A^u x^u + B^u u \tag{3.74b}$$

$$0 = g_{\mathrm{DAE}}(x^s, x^u, z, d, \theta) \tag{3.74c}$$

$$y = h(x^s, z), \tag{3.74d}$$

with linear actuator dynamics. Specifically, this yields

$$\dot{x}^s = z \tag{3.75a}$$

$$\dot{x}^u = \begin{pmatrix} -\frac{1}{\tau_{\mathrm{pwt}}} & 0 \\ 0 & -\frac{1}{\tau_{\mathrm{br}}} \end{pmatrix} x^u + \begin{pmatrix} \frac{1}{\tau_{\mathrm{pwt}}} & 0 \\ 0 & \frac{1}{\tau_{\mathrm{br}}} \end{pmatrix} u \tag{3.75b}$$

$$0 = -\theta_2 {x_1^s}^2 + \frac{x_2^u - x_3^u}{r_{\mathrm{eff}}} - (\theta_1 + m_{\mathrm{I}})z - \theta_1 g(\sin d - \theta_3 \cos d) \tag{3.75c}$$

$$y = \begin{pmatrix} x^s \\ z \end{pmatrix}, \tag{3.75d}$$

with the algebraic state being the vehicle longitudinal acceleration $\dot{x}^s = z = a$, the actuator states $\dot{x}^u = [x_2^u,\, x_3^u] = [T_{\mathrm{we}},\, T_{\mathrm{br}}]$, and the parameter vector $\theta = [m,\, C_{\mathrm{d}},\, C_{\mathrm{r}}]$. Note that we kept the original indexing of the state parameter vector with the idea to create less confusion.

3.5.3 System for simulation purposes

For simulation purposes, another system representation is of interest. Since we are interested to cover the full range of possible velocities, including negative ones, we consider the algebraic constraints as given in (3.63). Additionally, for numeric stability, we approximate the $sign(v)$ functions with the smooth functions $\tanh(c_{\mathrm{t}}v)$, with c_{t} being a real positive scaling factor to make the function steeper at the origin. This allows to simulate a vehicle also during standstill conditions. We obtain the following DAE:

$$\dot{x} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\frac{1}{\tau_{\mathrm{pwt}}} & 0 \\ 0 & 0 & -\frac{1}{\tau_{\mathrm{br}}} \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} z + \begin{pmatrix} 0 & 0 \\ \frac{1}{\tau_{\mathrm{pwt}}} & 0 \\ 0 & \frac{1}{\tau_{\mathrm{br}}} \end{pmatrix} u \tag{3.76a}$$

$$0 = \begin{pmatrix} -C_{\mathrm{d}} & \frac{1}{r_{\mathrm{eff}}} & -\frac{1}{r_{\mathrm{eff}}} & -(m + m_{\mathrm{I}}) & -mg & -mgC_{\mathrm{r}} \end{pmatrix} \begin{pmatrix} \tanh(c_t x_1) {x_1}^2 \\ x_2 \\ \tanh(c_t x_1) x_3 \\ z \\ \sin d \\ \tanh(c_t x_1) \cos d \end{pmatrix}. \tag{3.76b}$$

This representation will be used, for example, for the experiments carried out with Simulation Environment 6.3 and 7.1.

3.5.4 Discrete time state space forms

Integral form

We can write system (3.66) from Section 3.5.2 as a time-discrete system in state space form under the assumption that inputs u, disturbances d and parameters θ remain constant during the sampling duration and discrete measurements are available at multiples of the sampling time $T_s = t_k - t_{k-1}$. The system then takes the form:

$$x_k = x_{k-1} + \int_{t_{k-1}}^{t_k} \begin{pmatrix} \frac{x_2(t)-x_3(t)}{(m+m_\mathrm{I})\, r_\mathrm{eff}} - \frac{mg}{m+m_\mathrm{I}}(\sin\varphi + C_\mathrm{r}\cos\varphi) + \frac{-C_\mathrm{d}}{m+m_\mathrm{I}} \cdot x_1(t)^2 \\ \frac{-1}{\tau_\mathrm{pwt}} x_2(t) + \frac{1}{\tau_\mathrm{pwt}} u_{1,k-1} \\ \frac{-1}{\tau_\mathrm{br}} x_3(t) + \frac{1}{\tau_\mathrm{br}} u_{2,k-1}. \end{pmatrix} \tag{3.77a}$$

$$y_k = x_k = (v_k, T_{\mathrm{we},k}, T_{\mathrm{br},k})^\mathsf{T}. \tag{3.77b}$$

Realization of this system can be done by (approximately) solving the integral using different ODE solvers, or by implementing approximations using Runge-Kutta methods or the forward Euler method, as described next. Approximation errors have to be considered when applying these methods.

Forward Euler method

One possibility of obtaining a time discrete version of the state space model presented in Section 7.4.2 is by the forward-Euler method, where we have to define a sampling time T_s. Doing so, we obtain a time discrete version of (7.23) with $k \in \mathbb{N}$:

$$x_{k+1} = f_k(x_k) + g_k(d_k) + B_k \cdot u_k \tag{3.78a}$$

$$y_k = H x_k \tag{3.78b}$$

$$\tag{3.78c}$$

$$x_k = \begin{pmatrix} v_k \\ T_{\mathrm{we},k} \\ T_{\mathrm{br},k} \end{pmatrix} \tag{3.78d}$$

$$f_k(x_k) = \begin{pmatrix} v_k + T_s \cdot \left(\frac{T_{\mathrm{we},k} - T_{\mathrm{br},k}}{(m+m_\mathrm{I})\, r_\mathrm{eff}} - \frac{C_\mathrm{d}}{m+m_\mathrm{I}} \cdot v_k^2 \right) \\ \left(1 - \frac{T_s}{\tau_{pwt}+T_s}\right) T_{\mathrm{we},k} \\ \left(1 - \frac{T_s}{\tau_\mathrm{br}+T_s}\right) T_{\mathrm{br},k} \end{pmatrix} \tag{3.78e}$$

$$g(d_k) = \begin{pmatrix} T_s \cdot \frac{-mg}{m+m_\mathrm{I}} (\sin\varphi_k + C_\mathrm{r}\cos\varphi_k) \\ 0 \\ 0 \end{pmatrix} \tag{3.78f}$$

$$B_k \cdot u_k = \begin{pmatrix} 0 \\ \frac{T_s}{\tau_{pwt}+T_s} \cdot u_{1,k} \\ \frac{T_s}{\tau_\mathrm{br}+T_s} \cdot u_{2,k} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ \frac{T_s}{\tau_{pwt}+T_s} & 0 \\ 0 & \frac{T_s}{\tau_\mathrm{br}+T_s} \end{pmatrix} \cdot u_k \tag{3.78g}$$

$$H = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}. \tag{3.78h}$$

This can also be expressed more compactly, but loosing information about the structure of the system as:

$$v_{k+1} = v_k + T_s \left(\frac{T_{\mathrm{we},k} - T_{\mathrm{br},k}}{(m + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{1}{m + m_{\mathrm{I}}} \left(mg\left(\sin\varphi_k + C_{\mathrm{r}}\cos\varphi_k\right) + C_{\mathrm{d}} v_k^2 \right) \right) \tag{3.79a}$$

$$T_{\mathrm{we},k+1} = \frac{T_s}{\tau_{\mathrm{pwt}} + T_s} \left(T_{\mathrm{we},k}^d - T_{\mathrm{we},k}\right) + T_{\mathrm{we},k} \tag{3.79b}$$

$$T_{\mathrm{br},k+1} = \frac{T_s}{\tau_{\mathrm{br}} + T_s} \left(T_{\mathrm{br},k}^d - T_{\mathrm{br},k}\right) + T_{\mathrm{br},k}. \tag{3.79c}$$

3.5.5 Explicit algebraic equation linear in unknown parameters

In order to apply linear regression techniques, as was already done for the estimation of vehicle mass and tractive forces in [AE16; AE+16; Rho16], an implicit algebraic formulation, which is linear in the unkown parameters is necessary. We will refer to this as the LiP formulation. For this purpose, the equilibrium of forces expressed in the implicit algebraic equation in (3.72b) needs to be re-written in the form

$$y^\theta = \theta^\mathsf{T} \phi. \tag{3.80}$$

in which y^θ is a known observation vector (for the system under investigation even a scalar), ϕ is a known regressor, and the unknown parameters are all collected in the parameter vector θ. The equation in this form reads as

$$\frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{r_{\mathrm{eff}}} - a m_{\mathrm{I}} = \begin{bmatrix} m & C_{\mathrm{d}} & m \cdot C_{\mathrm{r}} \end{bmatrix} \begin{pmatrix} a + g\sin\varphi(\cdot) \\ v^2 \\ g\cos\varphi(\cdot) \end{pmatrix}, \tag{3.81}$$

which can be easily derived from (3.64) or (3.72b) by bringing all the terms with unknown parameters to the right hand side:

$$\frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{r_{\mathrm{eff}}} - a m_{\mathrm{I}} = a m + C_{\mathrm{d}} \cdot v^2 + mg\sin\varphi(\cdot) + mgC_{\mathrm{r}}\cos\varphi(\cdot). \tag{3.82}$$

This establishes a linear relation from the parameter vector θ and a regressor vector ϕ to a fictional, in our case scalar, "observation vector"

$$y^\theta = \frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{r_{\mathrm{eff}}} - a m_{\mathrm{I}}. \tag{3.83}$$

The regressor ϕ is further given as

$$\phi = \begin{pmatrix} a + g\sin\varphi(\cdot) \\ v^2 \\ g\cos\varphi(\cdot) \end{pmatrix}, \tag{3.84}$$

which clearly is depending on the variables of the vehicle speed v, which we considered elsewhere as *differential state*, the *algebraic state* a, and the *disturbance* φ. The parameter vector θ hence is defined as

$$\theta = \begin{bmatrix} m & C_{\mathrm{d}} & m \cdot C_{\mathrm{r}} \end{bmatrix}^\mathsf{T}. \tag{3.85}$$

Remark 2. *Establishing a linear relationship with a vector containing only unknown parameters is only possible if we allow the product $m \cdot C_{\mathrm{r}}$ in the parameter vector.*

While we can assume that all the variables in both the observation vector and the regressor can be measured, we clearly cannot assume in practice that perfect measurements can be obtained. Also, the general assumption often used to establish linear regression techniques, namely that noise is only present as additive term in above equation, resulting in

$$y^{\theta} = \theta^{\mathsf{T}} \phi + \nu, \tag{3.86}$$

is unfortunately not given in practice. This will be important when discussing solution strategies for the parameter estimation problem in Section 6.4.2.

3.6 Conclusion

In this chapter, we introduced equations describing the control system of longitudinal vehicle motion, including the predominating physical effects. We learned that the acceleration of the vehicle is highly influenced by its mass and forces acting on the vehicle. A parameter with high impact is the vehicle mass, which is variable due to a changing number of passengers and load, but not directly accessible to measurements. Precise knowledge about the vehicle mass is valuable in any model based control law. External longitudinal forces acting on the vehicle are, among others, dependent on the road slope φ. We assume that information about future road slope changes will be available in automated vehicles. Aerodynamic drag and rolling resistance coefficients are also time varying parameters, and changes are directly proportional to the resulting forces. We further discussed that the wheel torque build up is delayed due to various effects. Anticipated planning could compensate this effect. To summarize, we want to highlight:

- The quantities vehicles mass, aerodynamic drag and road friction cannot be considered known and always constant during real world vehicle operation
- Vehicle mass is known to vary around ± 50 % for passenger cars and up to ± 400 % for light duty vehicles
- Aerodynamic drag can vary up to ± 30 % - 70 %
- Road friction might vary around ± 60 %
- Wheel torque build up is delayed with a time constant of between 0.15 s to 0.4 s.

We also learned that we can reformulate the equations governing longitudinal vehicle motion into different forms, and for example find a system of ODEs, or an equation which is linear in the unknown parameters.

4 Related Work

"There is no such thing as a new idea. It is impossible. We simply take a lot of old ideas and put them into a sort of mental kaleidoscope. We give them a turn and they make new and curious combinations. We keep on turning and making new combinations indefinitely; but they are the same old pieces of colored glass that have been in use through all the ages."

– Mark Twain's Own Autobiography: The Chapters from the North American Review

This chapter gives an overview of existing literature about longitudinal vehicle motion control in general, including solutions presented to solve the motion tracking problem. Literature related to the state and parameter estimation problem will be presented, after having given the necessary background, within Chapter 6, which is fully dedicated to this topic. The same applies to publications around the topics of MPC, which is treated within Chapter 7 and can be found in Section 7.3, as well as RL, which are discussed in Chapter 8, Section 8.3.

Introductions to previous developments regarding longitudinal motion control and explanation of different types of ADAS systems can be found in Hedrick, Tomizuka, and Varaiya [HT+94], Rajamani [Raj11], Dang [Dan12] and Horn and Watzenig [HW14]. A recent survey, including a discussion of future challenges, can be found in Guanetti, Kim, and Borrelli [GK+18].

While a comprehensive discussion would be out of the scope of this thesis, we will mention some prominent examples. We will start with what we regard as ADAS solutions before focusing on controllers that were explicitly designed for the longitudinal motion-tracking task. We want to mention that often, a clear separation is not possible. This, for example, is the case when the proposed solutions contain a motion tracking module that only works together with the proposed, scenario-specific solution, targeting a higher-level goal.

4.1 Longitudinal motion control for ADAS

ADAS controllers are typically designed for specific tasks, like following a leading vehicle on a highway. This has the drawback that the control architecture often does not allow it to be applied to the general automated driving problem, which covers all possible scenarios.

Early longitudinal motion controllers were implemented to solve the most basic Cruise Control problem, which consists of maintaining a constant reference speed value, mostly in flat terrain as on highways. Rajamani [Raj11] gives a valuable introduction to these longitudinal motion control solutions, typically structured into higher- and lower-level controllers.

High-level motion controllers typically calculate a desired vehicle acceleration set-value command, which a low-level controller demands. For the higher-level controllers, the lower-level system, which contains vehicle dynamics and dynamic power-train behavior, was typically modeled as first-order dynamics with constant time delays. This separation has the advantage that the higher-level system can be typically formulated as a linear control problem, which facilitates control design.

Standard solutions applied a Proportional-Integral (PI) controller based on a control error calculated as the difference between the actual measured vehicle speed and the given reference speed *set-point value*. The output of this PI controller is a desired acceleration, which is then fed into the lower-level control module, which acts on the actuator. Such, the low-level controller transforms the output of the higher-level module (typically the desired, *current value* of the acceleration) into a throttle value *at the current time point*, inverting the vehicle and power-train equation as given for example in [CD95].

In earlier implementations, instead of calculating an engine torque input value, mean value models of the engine steady state dynamics had to be additionally inverted to calculate a throttle value set point since torque-based ECMs were not available at that time. Additionally, vehicle equations were simplified by neglecting the influence of the road grade for two reasons: first, no road grade measurement value or estimation was available. Second, Cruise Control scenarios were typically designed to be restricted to application on flat roads.

With the development of enhancements like ACC and platooning solutions, the focus was put on controllers, which additionally performed the vehicle-following task. A review of the development of ACC systems is given in Dang [Dan12], Björnander and Grunske [BG08] and Xiao and Gao [XG10]. Solutions providing string stability guarantees (see Section 1.6 for a definition and Section 4.3.5 for literature discussing the influence of actuator time lags and delays on the string stability of the platoon) were often the focus of the presentations. Another novelty was the application of ACC systems in stop&go scenarios, as appearing in traffic jams and urban traffic. Here, the vehicle needs to operate in the low-speed range and be able to come to a stop. Emergency braking scenarios have to be included, where the deceleration range is higher, typically up to negative acceleration values of -8 $[m/s^2]$.

In the following, we want to give a more detailed overview of existing solutions, categorized into the applied control methodology.

4.1.1 Linear control schemes

Pure linear controllers were proposed early to the ACC problem. Ioannou, Xu, et al. [IX+93] studied the application of Proportional-Integral-Derivative (PID) controllers with fixed gain and adaptive gains. A PI controller, which was augmented with a linear driver model, could be designed by Persson, Botling, Hesslow, and Johansson [PB+99]. A linear, optimal controller with string stability was designed by Liang and Peng [LP99]. Following a constant-time-headway policy, simple linear controllers could be proved to guarantee string stability [RZ02]. More advanced ACC solutions include the work of Moon and Yi [MY08; MM+09], who proposed to achieve naturalistic driving behavior by adapting acceleration limits to values obtained from a human driving database.

In order to meet satisfactory performance in real traffic situations, with abruptly changing traffic patterns, vehicle cut-ins, lane changes, and hard braking of the preceding vehicle, more elaborate control designs are necessary, according to Widmann, Daniels, et al. [WD+00].

4.1.2 Sliding-mode control schemes

Swaroop, Hedrick, and Choi [SH+01] designed a robust adaptive sliding-mode controller for a platooning scenario, which already included adaptation to changing vehicle mass, drag coefficient, and rolling resistance. The limiting assumption was that road grade influence was fully neglected. Another sliding-mode controller was proposed by Lu, Hedrick, et al. [LH+02]. It can be applied for both ACC and CACC. The approach from Bartono [Bar04] includes both the solution to the ACC problem and the lower-level controller. The higher-level control is based on a nonlinear sliding-mode approach. For the adaptive lower-level controller, see Section 4.2.8. Another approach using sliding-mode control is given in Ferrara and Pisu [FP04]. The control scheme utilizes state information about the speed and acceleration of the leading vehicle, which is not measured. For this purpose, they introduce an observer, which allows them to estimate the unknown information.

4.1.3 Fuzzy control schemes

Several solutions to cruise control problems were proposed based on fuzzy controllers. Naranjo, González, García, and Pedro [NG+07] applied an adaptive fuzzy logic controller to the ACC problem. Fuzzy reasoning rules, which aim to emulate human driving, were incorporated into the approach. The fuzzy PID controllers were proposed for higher- and lower-level control. The approach is able to cover both highway and stop-and-go scenarios.

Lin, Thi, and Wang [LT+17] presented an adaptive neuro-fuzzy predictive control design, maintaining a safe distance to the lead vehicle while saving energy and ensuring passenger comfort. The controller adapts to the behavior of the preceding vehicle in order to generate predictive information about its motion. This information is then used in a fuzzy neural network inference controller, which approximates the cost function of the optimal control problem. Further examples for fuzzy control schemes were Tsai, Hsieh, and Chen [TH+10], Cabello, Acuña, et al. [CA+11] and Mohtavipour, Mollajafari, and Naseri [MM+18].

4.1.4 Model predictive control

As mentioned in Eskandarian [Esk12], finding optimized controller performance while handling actuator and state constraints with conventional control approaches can be challenging. This is primarily due to the desire to meet multiple contradictory design objectives. In the context of ACC solutions, these objectives typically are minimizing error when regulating a time-gap to the preceding vehicle, preserving string stability, and increasing riding comfort while minimizing fuel or energy consumption.

Model predictive control is a framework that can handle multiple objectives under state and actuator constraints. This, together with recent progress in real-time capable MPC schemes, explains the increasing number of publications suggesting this methodology for the ACC and CACC problem.

A solution to the Adaptive Cruise Control problem based on model predictive control is given by Corona, Lazar, De Schutter, and Heemels [CL+06]. They combine the motion planning and longitudinal control layer in a hybrid MPC. It is based on a nonlinear vehicle equation in which air and friction resistance are considered, but no road grade or vehicle mass variation is taken into account. The nonlinearities are then eliminated by transforming the problem into a piece-wise affine system to apply mixed-integer linear programming to solve the underlying MPC optimization problem. This reduces the computational load while accepting a sub-optimal solution to the original problem. A similar approach using combined state equations for vehicle and power-train dynamics, augmented by the control variables of the ACC problem, was presented in [SO+12]. Vehicle parameters were considered as known and constant. Naus, Bleek, et al. [NB+08] proposed a linear model predictive controller, implemented as explicit MPC in order to reduce computational requirements.

Keulen, Naus, and Jager [KN+09] proposed an explicit model predictive controller to design a predictive cruise controller for a Hybrid Electric Vehicle. It combined the generation of an energy optimal speed reference with "disturbances" coming from an ACC module in a sequential manner.

Asadi and Vahidi [AV11] utilized upcoming traffic signal information to improve fuel economy and reduce trip time and proposed using MPC to realize a combined high-level and low-level control scheme explicitly suited for this task. Li, Li, Rajamani, and Wang [LL+11] included driver desired characteristics of the ACC behavior to the multi-objective optimization in their MPC scheme. Restrictions on the acceleration level were also given as constraints. Li, Jia, Li, and Cheng [LJ+15] proposed different methods of calculating the optimal control solution within a model predictive control framework for fuel-economy-oriented Adaptive Cruise Control. Qiu, Ting, et al. [QT+15] proposed a two-layer approach to the Adaptive Cruise Control problem. The first layer applies an ancillary control law based on linear model predictive control to calculate a desired vehicle acceleration. As a specialty, it treats the acceleration and velocity of the leading vehicle as an unknown disturbance to a nominal plant behavior. This helps to smooth out the behavior of the leading vehicle. The low-level control module uses a simple PID controller to calculate throttle and brake input signals. This approach cannot use any advance information by design on the actuator side.

Alrifaee, Liu, and Abel [AL+15] proposed a variant called Economic Model Predictive Control (EMPC) for application to the Adaptive Cruise Control problem of electric vehicles. In EMPC, instead of trying to stabilize a system to equilibrium, control actions are found that minimize the actuator output necessary to maintain the system within its boundaries. Schmied, Waschl, Quirynen, and Diehl [SW+15] uses Nonlinear Model Predictive Control (NMPC) to design a CACC system that predicts the velocity profile of a predecessor vehicle out of distance measurements and information about oncoming traffic lights. They demonstrate the potential for fuel and emission savings, which arises mainly from the relaxed formulation of the vehicle following the task by introducing a tolerance on the inter-vehicle distance. This allows for reducing peak vehicle acceleration, which is the main driver for emissions and fuel consumption, as discussed in Section 2.2. Zhu, Chen, and Xiong [ZC+16b] suggests a two-layer approach where the upper layer is defined as a linear MPC problem, which creates a demand acceleration for the lower level, and nonlinearities of the vehicle model are incorporated in a lower-level feed-forward control. Aerodynamic drag, rolling resistance, and road grade influences are neglected and treated as disturbances. Turri, Kim, et al. [TK+17] presented a model predictive controller for "eco-platooning". It is based on the assumption that a forecast of future accelerations can be shared by the lead vehicle. The controller is able to compute trajectories that avoid any active braking by incorporating a terminal constraint in the MPC formulation. This allows them to consider safety margins to the lead vehicle over a longer horizon than the prediction horizon used in the controller, which helps to reduce the computational load. Like many other ADAS solutions, road grade influences were neglected.

Xin, Xu, et al. [XX+22] used MPC for optimal energy management of a fuel cell hybrid electric vehicle and included online vehicle mass estimation to improve performance. Contrary to the proposal of this thesis, vehicle parameter estimation was limited to vehicle mass only, while the aerodynamic drag coefficient and the rolling resistance coefficient were assumed to be constants. By including a vehicle mass estimation scheme alone, they found that fuel consumption could be reduced by 0.1 %. They also found that their estimation algorithm based on Recursive Least Squares (RLS) with multiple forgetting performed poorly in some scenarios, especially when trying to track a (quasi) continuously varying vehicle mass. Nevertheless, they left further investigation about the reasons and how to improve performance to future work. They did not mention the difficulty of vehicle mass estimation during phases of insufficient excitation.

4.1.5 Dynamic programming

Hellström [Hel10] proposed an approach based on dynamic programming to reduce the fuel consumption of heavy-duty vehicles. This could be classified as PCC solution. A similar approach was realized and evaluated for sports cars by Radke [Rad13]. Using future road and traffic information, the solution generates optimal, fuel-saving vehicle speed trajectories. The approach is modular. The first module applies dynamic programming to generate a predictive vehicle speed profile within certain upper and lower speed limits. An optimal gear selection is also provided. The time-varying speed profile is then tracked by a PI controller with feed-forward to compensate for the stationary vehicle and power-train forces. It is modeled as an ideal plant (no time delay) and includes road grade changes. No adaptation to changes in the aerodynamic drag or rolling resistance is present. It uses a simple vehicle mass adaptation, but no results on the robustness of the approach to changing drag and resistance parameters are provided.

4.1.6 Adaptive dynamic programming

Zhao, Wang, and Liu [ZW+13] proposed an actor-critic adaptive dynamic programming approach to the (linear) ACC problem. Zhao, Hu, et al. [ZH+14] applied supervised adaptive programming to solve the ACC problem both for highways as well as stop-and-go scenarios in a two-layer approach. The bottom layer, which manages throttle and brake commands, uses a fuzzy gain scheduling scheme based on a PID control from [ZD12].

A similar but more sample-efficient approach was proposed by Wang, Zhao, and Cheng [WZ+18a]. They suggested an adaptive cruise controller using adaptive dynamic programming with experience replay.

4.1.7 Other ADAS solutions

A reference model-based nonlinear controller for the high-level ACC control was suggested by Martinez and Canudas-de-Wit [MC07]. The reference model is nonlinear and ensures safety constraints and comfort criteria, focusing on designing inter-vehicle distances achieved by the approach closer to human behavior. However, nonlinearities within the vehicle dynamics act as unknown disturbances to the control loop.

Zhu, Dai, et al. [ZD+17b] presented an adaptive longitudinal control method for the ACC problem based on a combination of a higher-level controller using neural dynamic programming and an internal model structure for the low-level control. The latter was based on the internal model control approach from [WS+13].

Lefèvre, Carvalho, and Borrelli [LC+16] combined Robust Model Predictive Control (RMPC) with a learning-based approach to adopt human driving styles. Car-following strategies are learned off-line from human demonstrations. This model is used to compute desired base accelerations. The reference acceleration is then modified by a predictive controller, which enforces a set of comfort and safety constraints. The approach is designed specifically for car-following scenarios. It assumes that a lower-level controller exists, which can perfectly track the desired acceleration but does not further provide a solution. Hence, the work targeted in this thesis goes towards a solution to this problem and perfectly complements such an interesting approach.

4.2 Longitudinal motion tracking controllers

As mentioned above, typical ADAS solutions often designed the high-level control separately from a low-level ADAS control module.

In most cases, the lower-level control module is designed in order to include power-train dynamics as well as engine dynamics. Few authors suggested control modules dedicated solely to vehicle motion tracking while leaving power-train and engine dynamics to a separate module. Nevertheless, we want to include such combined approaches in the following presentation.

As an interface, most approaches calculated a desired acceleration value in the high-level module, which was then supposed to be realized by the lower-level module.

4.2.1 Feed-forward control

In many cases, starting with early publications that were presented during the PATH program, low-level modules were realized as simply as feed-forward controls. By inverting both vehicle dynamics equation, engine, and power-train dynamics, throttle values could commanded to the engine, as in [CD95]. They built on power-train models, which have been presented, as examples, by McMahon, Hedrick, and Shladover [MH+90] and Hedrick, Tomizuka, and Varaiya [HT+94]. These approaches typically work under the assumption of known and constant parameters within all underlying equations. Control errors arising from variations and un-modeled dynamics had to be fully compensated by the higher-layer modules.

Some feed-forward part is typically included in the solutions described in the following. We want to give an overview of the literature, again categorized, as far as possible, into the family of primary control approaches suggested in the publication.

4.2.2 Proportional-Integral control schemes

Apart from lower-level solutions mentioned to be combined with higher-level controllers for ADAS, some literature exists that is dedicated to AVs. Most of the early literature in this category can be linked to publications around the DARPA challenges (see Section 2.1.1). In Thrun, Montemerlo, et al. [TM+06], we find that Stanford Racing Team's vehicle, Stanley, was equipped with a drive-by-wire system and a PID controller to generate velocity controls, working at 20Hz. [UA+08] and [BF+09] mention the implementation of a simple proportional-integral (PI) controller in combination with feed-forward terms to linearize throttle and brake dynamics. No adaptation to a changing vehicle mass or consideration of road grade is mentioned. Another vehicle from the DARPA challenge, Team AnnieWAY [GL+12], applied an integral anti-windup feedback controller. The same controller was used in Daimler's vehicle Bertha [ZD+13]. The solution did not actively consider forces resulting from aerodynamic drag and road slope, but these were treated as disturbances. Gehrling [Geh00] proposed a solution in which the low-level control is based on a PID controller with a feed-forward term. The influence of forces and effects of a changing vehicle mass and road grade is not actively considered but left to the integral control action. It mentions the necessity of calculating the throttle angle, lacking a torque-based ECM structure.

We already mentioned in Section 4.1.4 that Qiu, Ting, et al. [QT+15] proposed to use a simple PID controller to calculate throttle and brake input signal for the low-level part of the MPC approach, which was the main contribution of the paper. This approach cannot make use of any advance information by design on the actuator side.

4.2.3 Fuzzy control

Zhongpu and Dongbin [ZD12] applies a hybrid PID control strategy to realize a given vehicle acceleration by acting on the throttle and brake. A fuzzy gain scheduling scheme based on a PID control is used to control the throttle. A hybrid feed-forward and feedback control is applied for acting on the brake. The fuzzy parameters are optimized off-line with a particle swarm optimization algorithm. No evaluation on the robustness of the approach to changing vehicle parameters is given. As already mentioned in Section 4.1.3, Naranjo, Jiménez, Gómez, and Zato [NJ+12] presented a fuzzy controller for the low-level drive-by-wire system of a prototypical, automated vehicle and presented results from vehicle tests. In general, solutions based on fuzzy logic benefit from the fact that they do not require a detailed model but suffer from a very high tuning effort.

4.2.4 Gain scheduling

The application of gain scheduling PI and Linear Quadradic (LQ) controllers was examined in [SO+11]. Hunt, Johansen, et al. [HJ+00] also used gain scheduling mechanisms in their approach (see Section 4.2.6).

4.2.5 Internal Model Control

Wang, Sun, et al. [WS+13] presents an Internal Model Control structure for the speed tracking of autonomous land vehicles and conducts an experimental study. The model includes non-parametric models obtained by vehicle experiments.

4.2.6 Neural network control

Hunt, Johansen, et al. [HJ+00] applied a neural-network-based controller, which was trained by supervised learning. The network structure was a Takagi-Sugeno type fuzzy local controller network [TS85], including up to second order Auto Regressive with Exogenous Input (ARX) model. Gain scheduling mechanisms were used to adapt the controller behavior to the nonlinear plant. For model identification, the vehicle dynamics were excited by Pseudo-Random Binary Sequence (PRBS) signals. Due to the a-priori identification of the plant behavior, no adaptation to time-varying vehicle parameters is given with this approach.

4.2.7 Model-free control

Model-free control [FJ13] is a control concept that works by approximating complex nonlinear dynamics by an ultra-local model of the form

$$\dot{y} = F + \alpha u,$$

which is only valid during a very short sample time period. In the above equation, y is the control output, F represents both un-modeled and neglected dynamics, u is the control input, and α is a parameter that has to be tuned by the practitioner.

Based on the group's previous works [CD+09], which was evaluated as robust to parameter changes, Polack, D'Andréa-Novel, et al. [PD+17] suggest adaptive model-free control to track a reference target speed value. They did not take any engine dynamics into account in their simulations. Menhour, D'Andrea-Novel, et al. [MD+18] compared the model-free control approach to a flatness-based approach and classical PID control. They conclude that model-free control can obtain better results without the need for a vehicle model.

As a disadvantage of the approach, no constraints can be considered, and no predictive behavior can be realized. Also, since no model is involved, no meaningful parameter estimates can be obtained in such an approach.

4.2.8 Adaptive control

Early contributions as [XI94] suggested using Adaptive Control schemes for the tracking task while acting on the throttle. They only considered linear and no actuator dynamics, and their results showed poor tracking behavior in real-world experiments. BMW Group's Highly Automated Driving Project mentioned to apply a combination of controllers from Werling [Wer10] and Bartono [Bar04]. The first is a combined lateral and longitudinal control strategy and will be discussed in Section 4.3. The lower-level controller from Bartono [Bar04] inputs a desired vehicle acceleration and calculates the necessary engine torque via an inverse power-train model. It includes a disturbance estimator, which estimates the sum of all disturbing forces resulting from vehicle mass changes, changes in air drag, rolling resistance, or wind. This approach can realize offset-free reference point tracking, but no predictive behavior and constraints can be handled. The disturbance estimator provides a value for lumped disturbances, making distinguishing between the effects of different parameter values impossible. Also, in such a setting, although one could argue that for (a reactive) controller, the knowledge of the true vehicle parameters might not be necessary, this is beneficial for various other engineering tasks, as discussed in the introduction.

4.2.9 Model Predictive Motion Tracking

Very little work was found proposing to apply MPC for longitudinal vehicle motion tracking. Murayama and Yamakita [MY09] as well as Sakai, Kanai, and Yamakita [SK+10] propose specialized solutions using NMPC for speed tracking. Their approach is not modular and directly includes actuator control, and they provide strategies to control engine parameters like variable valve lift. Hence, this approach is particular and limited to applications in vehicles with internal combustion engines.

As mentioned in Section 4.1.4, Asadi and Vahidi [AV11] included vehicle dynamics as well as power-train equations into the suggested MPC approach, which is limited to the scenarios at intersections, which were the focus of their work. After the author's publications [BK16a] and [BK16b] appeared, which suggested applying MPC for the longitudinal motion tracking task in a modular structure, very recently, Walz and Hohmann; Walz, Schucht, Reger, and Hohmann; Walz and Hohmann [WH19a; WS+18b; WH19b] also followed to suggest a similar approach. His work is different in that it focuses on low-speed vehicle control and on scenarios like, for example, climbing a curb. He could show in vehicle tests that a model predictive controller can improve the tracking behavior considerably by taking into account predictive information about the road profile, including obstacles. Nevertheless, while significant advances were made to include tire dynamics into the vehicle motion model, his work was done under the assumption of known vehicle parameters, like vehicle mass. In the present thesis, these parameters are considered unknown, and a strong focus will be given to the parameter estimation problem under realistic assumptions in the remainder of this work.

4.3 Other topics

4.3.1 Bi-level control

The work published by Werling [Wer10] can be categorized as a solution for AVs and was specially designed for driving in structured environments in which way-point information is exploited to generate motion trajectories. He proposes a bi-level control strategy for both lateral and longitudinal control. The approach consists of three hierarchical control elements. The first solves the motion trajectory planning problem, calculating collision-free trajectories to follow given way-points along the road. A polynomial approach is used to provide optimal trajectory sets, of which the collision-free one with the smallest initial jerk value is given to a lower-level motion controller. The lower-level controller takes the trajectory as input and calculates the desired tire force. Depending on the current vehicle speed, it switches between different nonlinear control laws. The first is applied at high vehicle speed and based on exact linearization. For slow driving maneuvers, different control laws are calculated depending on the driving direction, based on Lyapunov theory. A third control element converts the resulting longitudinal tire forces into a throttle position and brake pressure. Apart from a static feed-forward control law, it contains an anti-windup integrator to compensate for disturbances like wind drag, road slope, and vehicle mass changes. Lacking a torque interface to the engine, a mapping procedure to calibrate the inverse Spark Ignition (SI) engine characteristics is also mentioned.

4.3.2 Limits of handling

Some authors focused on solutions at the limits of handling. This is an important research direction, where considering full vehicle body dynamics and parameter estimation, including road friction coefficients, plays a crucial role. A broad literature exists on related topics. Nevertheless, we focus on comfortable control solutions far away from these limits and see vehicle maneuvers at the limits of handling out of the scope of this thesis. Still, we want to mention a few publications containing longitudinal control solutions. Reschka, Böhmer, et al. [RB+12] presented the longitudinal control approach of the vehicle “Leonie” of the Stadtpilot project. They introduced the estimation of a “grip” value, which estimates the road-tire friction in slippery road conditions. Kritayakirana and Gerdes [KG12] proposed a controller based on the concept of the friction circle and demonstrated its performance in a spectacular demonstration. An automated Audi TTS vehicle [FT+12] was climbing the Pikes Peak Hill Climb, a legendary dirt road track in Colorado, USA.

4.3.3 Parking solution

A different topic is addressed in Bu and Tan [BT07]. They proposed a solution specifically designed to stop a heavy-duty vehicle at a designated parking position.

4.3.4 Solutions specific to prototypical AVs

Some publications can be found that present control solutions specific to prototypical hardware implementations of automated vehicles. These are suited for vehicles that were equipped with additional electro-mechanical actuators to realize interfaces for longitudinal motion control. These devices act directly on the throttle and brake pedals designed for human operation. Typically, these devices had to be introduced because of a lack of availability of electronic software interfaces when retrofitting vehicles. Another reason might have been to keep the vehicle controls unmodified since modifications might have impacted the type-approval necessary to operate on public roads. Examples of such solutions are ([UA+08] and ([BF+09]. These additional actuator components add time delays, lags, and backlash effects to the control loop. From a control system perspective, direct access to the electronic throttle is preferable to facilitate control.

4.3.5 Influence of actuator delays on string stability

String stability has been defined as the property to guarantee that inter-vehicle spacing errors in a platoon do not amplify as they propagate through the platoon [Kai74].

As already mentioned in Section 1.6 and 1.6, actuator lags and time delays given within the power-train and engine dynamics have a negative effect not only on the vehicle dynamics of a single vehicle but highly influence the dynamics of the vehicle platoon. It was also shown that reducing lags and time delays pose a significant challenge for the efficient implementation of vehicle platooning solutions [XG11]. A systematic study on limitations of spacing policies for platoons was done in Solyom and Coelingh [SC13]. Xiao and Gao [XG11] investigated the practical string stability of homogeneous and heterogeneous platoons of Adaptive Cruise Control vehicles, considering "parasitic time delays and lags of actuators". They analyzed control systems with sliding-mode controllers. It concludes that the negative effect of time delays is more significant than that of time lags. Ghasemi, Kazemi, and Azadi [GK+15] provides theoretical results for the stability of a platoon by considering time delay and lag. Chen, Wang, Alkim, and Arem [CW+18] also mentions the "detrimental effects on string stability" of feedback delay and actuator lag. It investigates the possibilities of compensating for uncertain variations in the time delay using a robust min-max MPC approach for the higher-level controller.

Chehardoli and Homaeinezhad [CH17a] also investigated different theoretical approaches to prove string stability for the problem formulation of linear time-varying delay systems. The time delays on the actuator side are assumed to have values between 0.05 and 0.15 seconds. Alarmingly, a recent study that conducted various experiments with production vehicles suggests that many commercial ACC systems are not string stable [GG+19]. The reason might be owed to using far too simplifying assumptions during the control design process [XG11].

4.4 Not covered in this review

Many other topics have been treated within the existing literature, which can be seen related to motion control in general, but not with longitudinal motion tracking specifically, and are therefore not covered in this review:

The optimization of specific power-train components, for example *energy management* of HEVs, *Gear shift optimization* are left to the power-train control unit in the suggested modular framework and therefore not treated here more in detail, although we learned that some authors directly combined cruise control solutions, motion tracking and power-train optimization into unified frameworks. By suggesting to forward predictive trajectories from a motion tracking unit to a power-train controller, as done by this thesis, we enable the power-train controller to be designed by optimal, predictive control principles but make the higher layers independent from any power-train specific characteristics.

A broad range of literature deals with different aspects of vehicle platooning, like, for example, *platoon formation and coordination*. We only want to refer to an overview given in Shladover, Nowakowski, Lu, and Hoogendoorn [SN+14] and not treat this topic further. *Eco-routing* deals with finding an optimal route considering timing and energy consumption criteria Guanetti, Kim, and Borrelli [GK+18], but are left out from this review. Other systems have a connection to longitudinal control but are not treated here. These are, for example, ADAS systems like *Anti-lock Braking* , *Hill Descent Control* and *Emergency Braking*.

Also not covered within this chapter, since some more topic-specific background is needed to properly discuss the existing literature, is work related to Nonlinear Model Predictive Control (NMPC), Reinforcement Learning (RL) and smoothing algorithms. We will, therefore, include sections within the particular chapters with related work for these topics. Therefore, for literature that is specific to NMPC theory and applications, please refer to Section 7.3; for the one related to RL refer to Section 8.3 and for polynomial filtering and smoothing algorithms to Section 5.3.

4.5 Conclusion

Various solutions to the longitudinal vehicle motion control problem exist. Many authors suggested a hierarchical control structure in which a higher-level control module calculates a desired acceleration value, and a lower-level controller acts as a motion-tracking unit and transforms this desired acceleration into actuator commands for the engine and brake.

Although this work focuses on the motion tracking part, we included a discussion on the high-level control modules to understand their tasks, differences between the motion controller and a motion tracking unit, and interfaces between these modules. We learned that high-level control modules arose from the desire to provide solutions to ADAS specific scenarios, evolving from highway cruising over vehicle following towards driving in a platoon. For AVs, a broad range of additional driving scenarios has to be covered. On the other hand, on the actuator side, vehicle power trains also faced an evolution, starting from gasoline engines, with the throttle-plate angle being an adequate interface, towards complex hybrid power-trains, where additional actuators exist and much energy-saving potential lies in optimizing the power split between these power sources.

With a broad range of power-train variants on one side and more complex driving scenarios that are automated on the other side, a modular vehicle architecture is beneficial from an engineering point of view. This includes the longitudinal control modules. Therefore, one of the proposals, which builds the boundary conditions for the rest of the investigations done in this thesis, is to design a motion-tracking unit capable of serving as an interface between the motion planning unit and any existing or future power-train unit.

Shifting our attention back to the literature we discussed, we dare to formulate the following statements: Looking at the control algorithms that have been suggested, they evolved from simple PID controllers in the early history of vehicle automation to more elaborate solutions. In recent years, a rising number of authors proposed the application of MPC schemes, although primarily as solutions to scenario-specific ADAS solutions. Benefits can be seen from the possibilities to include predictive information on one hand, as well as considering state and actuator constraints.

Actuator delays were typically not considered in the tracking module but in the higher-level controllers designed for cruise control scenarios. This limits the modularity of the overall architecture and might need to be revised when designing a system capable of solving generic driving tasks.

Regarding vehicle motion tracking controllers, many approaches use simplifying assumptions to reduce model complexity. In most of the existing literature, the authors suggested not considering forces resulting from tire resistance, aerodynamic drag, or gravity due to inclined roads, meaning that the resulting effects must be treated as a disturbance to the tracking controller. If these effects were considered at all, the related parameters are most widely assumed, both known and time-invariant. In literature including adaptations to changing vehicle parameters, for example in Bartono [Bar04] or Polack, D'Andréa-Novel, et al. [PD+17], they were treated as a single resulting disturbance force, taking away the possibility of learning the true physical parameters, and hence correctly predicting into the future.

To the best of the author's knowledge, at the time of first suggesting one of this thesis proposal in [BK16a] and [BK16b], no existing publication was found that suggests incorporating advance information about future disturbances resulting from changing road grades and design the motion tracking unit in a way that predictive information can still be forwarded to the actuators. Therefore, we will dedicate the following sections of this thesis to designing such a model predictive longitudinal tracking control module while looking into various methods. In each section, we will include a methodology-specific analysis of related work.

5 Novel Smoothing Algorithms for Multichannel Time-Series Data

"Our beauty lies in this extended capacity for convolution."

– Thomas Pynchon

A slightly modified version of this chapter has been published by the author of this thesis as a preprint in [Bue22].

5.1 Introduction

For the controller we will propose in Chapter 7, we need estimations of the true values of the engine speed, which serve as control inputs. In addition, knowing the true values of both engine speed and acceleration can reduce Errors-in-Variables' effect in the parameter estimation problem, as discussed in Chapter 6. While a model-based approach of estimating the states in a joint and dual estimation scheme is also discussed in Chapter 6, this chapter is dedicated to a model-free approach for state estimation. Typically, model-free filtering and smoothing algorithms for time-series data, for example, moving average, Butterworth, or Savitzky-Golay filters, operate only on a single measurement channel. If filtering of multiple channels, like in our case, speed and acceleration, needs to be performed, this happens by running two concurrent and independent filters. Hence, information on the interrelation between measurement channels, namely that acceleration is the derivative of speed, cannot be incorporated. Algorithms of the Kalman Filter family, as often used, for example, in tracking problems, would be able to incorporate such information but need additional assumptions about the evolution of the modeled quantities, in the form of, e.g., constant velocity or constant acceleration assumptions, which might be too restrictive to obtain sufficient results.

This leads to the main question we want to answer in this chapter: How can we incorporate the knowledge of measurement channels being derivatives of each other in an otherwise model-free smoothing algorithm?

As a result, a novel algorithm will be derived in this chapter, which can be used for smoothing and filtering noisy, multi-channel time series data involving derivatives. A slightly modified version of this chapter has been written by the author of this thesis and published, by the time of writing, as a preprint in Buechel [Bue22]. Similar to the Savitzky-Golay convolutional smoothing and differentiation algorithm [SG64], the proposed algorithm is based on local polynomial regression. It can, therefore, be seen as a generalization of the Savitzky-Golay (SaG) algorithm, which can be applied to non-uniformly sampled multi-channel signals consisting of a base channel together with their derivatives. Measurement data of such a structure is given in many physical applications, for example, where noisy motion data is available from measurements, including distance, velocity, and acceleration, or a subset of those mentioned above. Filtering and smoothing motion data is also beneficial when solving the control problem treated in this thesis. While the first of the two proposed algorithms operates in a receding horizon fashion, the second can be interpreted as a recursive version of the first.

The original SaG-smoother is restricted to process scalar-valued, uniformly sampled time-series data over a symmetric measurement window with an odd-numbered amount of measurement points. A variety of extensions and generalizations have been proposed earlier, for example, to overcome the restriction of the measurement window having to be of odd size and uniformly sampled [LY+05; SC+19], or to smooth multi-dimensional image data [SA+04]. We will discuss some of these extensions and other related proposals in more detail in Section 5.2.

The main contribution of the proposal in this chapter lies in performing the local approximation by means of weighted polynomial least squares on all signal channels concurrently, exploiting the knowledge about the differential relation between elements in the measurement vectors. We will see that this leads to improved smoothing performance compared to operating on each signal individually.

The proposed algorithm can be interpreted in various ways. First, as an extension and generalization of SaG-smoothing [SG64], while it degrades to be identical in case it is applied to a single channel signal obeying the restrictions mentioned earlier for SaG-smoothing.

Second, the algorithm can be interpreted as a method to efficiently calculate physics-informed optimal kernels for convolutional layers in Machine Learning (ML) algorithms for time-series data. Hence, it can be seen as an “approach to fix [. . .] weights within the Neural Network (NN) to physically meaningful values and make them non-modifiable during training” [WJ+22]. Based on polynomial regression, not only smoothing but also differentiation of the signal channels can be efficiently performed from the polynomial coefficients of the local approximation, up to the order of the polynomial used in the regression, in the same fashion as in Savitzky and Golay [SG64]. The resulting derivative will inherit the smoothness properties of the polynomial, which approximates the base signal. As an example, acceleration estimations can be obtained from noisy velocity data. An estimation of a given signal’s (delayed) time-derivative is needed in various control algorithms, for example, in Concurrent Learning Model Reference Adaptive Control [CJ10b]. The proposal was motivated by the desire to obtain smoothed values of vehicle motion data of distance, velocity, and acceleration from noisy measurement data. This occurs in a variety of vehicle control and robotics problems. Filtering motion data is also beneficial when solving the control problem treated in this thesis.

As stated earlier, the original SaG-smoothing works on the principle of local polynomial regression. It was, according to Menon and Seelamantula [MS14], motivated by Weierstrass’s theorem [Bra59], which states that “every continuous function defined in a closed interval can be uniformly approximated, as closely as desired, by a polynomial function” [MS14].

At the time of writing, we found more than 18.000 citations of the original paper [SG64] in Google Scholar and more than 400.000 search results entering the term “Savitzky-Golay” on the search engine Google. This indicates that Savitzky-Golay smoothing is an extensively used tool. According to Schafer [Sch11], it is applied especially within the chemical process domain, while it might be less known in other domains.

SaG smoothing [SG64] can be viewed in many different ways that are mathematically equivalent. First, it can be seen as a "*generalized moving average filter*" [SA+04] or a generalization of the "*Finite Impulse Response (FIR) averager filter*" [Orf10]. Since the smoothed values are obtained by fitting a polynomial approximation of the noisy data points over the sliding window by least squares regression, it has also been called a "*digital smoothing polynomial filter*" or a "*least squares smoothing filter*" [Orf10]. Savitzky and Golay [SG64] showed that the calculation of can also be performed by a *convolution operation* with a FIR of a characteristic polynomial function, or as stated by Schafer [Sch11]: "we can think of least squares smoothing as a shift-invariant discrete convolution process." Orfanidis [Orf10] states that "The Savitzky-Golay FIR smoothing filters, also known as polynomial smoothing, or least squares smoothing filters [...], are generalizations of the FIR averager filter that can preserve better the high-frequency content of the desired signal, at the expense of not removing as much noise as the averager" [Orf10], p. 427. Since the convolution operation has also been named the "sliding dot-product of a flipped kernel with the signal" [Roh21], we can also look at it as a way of obtaining optimal kernels in one-dimensional convolutional layers of neural networks. Additionally, we want to look at it as a model-free moving horizon estimation algorithm.

This chapter will be organized as follows: After presenting related work and some existing generalizations and extensions in Section 5.2, we will present the background on polynomial filtering in Section 5.3. The derivation includes the connection between polynomial least squares and the calculation by means of a convolution operation, as well as presenting an efficient way of calculating the convolution vector using QR decomposition. It also includes the PKS algorithm [RB+14; Rho16], a recent proposal for a recursive version of Savitzky-Golay smoothing. This will give the foundation to derive the proposed algorithms neatly in Section 5.4. We will evaluate the proposed algorithms in Section 5.5 before concluding in Section 5.6.

5.2 Related work

Savitzky and Golay [SG64] derived polynomial least squares smoothing by considering various convolution functions and hence established the connection between polynomial least squares smoothing and the convolution operation. While showing that zero-order polynomial smoothing results in the moving average smoother, they provided various tables of convolution kernels for smoothing and differentiation for a variety of polynomial orders and window sizes. Madden [Mad78] published "Comments on the Savitzky-Golay convolution method for least squares fit smoothing and differentiation of digital data," in which they extended the tables to greater window sizes while correcting some errors in the original numeric tables. Orfanidis [Orf10] presented a derivation of the SaG-smoother in matrix form and discussed how closed solutions to the matrix inverses involved in the calculation can be obtained.

Later in 2004, Sühling, Arigovindan, Hunziker, and Unser [SA+04] discussed multi-resolution moment filters, among them an image noise-reduction method, which can be viewed as a multi-dimensional extension of SaG filtering suited for image data. While the original SaG smoothing and differentiation filter was restricted to work on uniformly sampled data of an odd window length, Luo, Ying, and Bai [LY+05] extended it to work on even-numbered data points within the moving window. Schafer [Sch11] provided a derivation of the SaG smoother using matrix notation and discussed its frequency domain characteristics.

To alleviate the performance degradation of the original SaG algorithm in the presence of non-normally distributed measurement noise, Menon and Seelamantula [MS14] introduced a robust version of SaG-smoothing. The solution is achieved by ℓ_1-norm error minimization through the technique of iteratively re-weighted least squares under the assumption of heavy-tailed, Laplacian-distributed noise. Solano-Araque, Colin, et al. [SC+19] proposed to determine vehicle acceleration from noisy, non-uniformly sampled speed data for control purposes. Their work can also be seen as a generalization of the SaG algorithm in the sense that it also does not require the measurement samples to be uniformly sampled. However, contrary to the proposal of this article, it does not exploit any derivative information but solely performs differentiation of a scalar signal in the fashion of SaG-differentiation [SG64].

Rhode, Bleimund, and Gauterin [RB+14] presented the Polynomial Kalman Smoother (PKS), which can be seen as a recursive version of SaG-smoothing by means of Kalman filtering. It is derived by defining the time shift of a polynomial function forward in time as a linear matrix operation, which allows the application of the linear Kalman filter framework. The PKS is limited to scalar-valued signals but can operate as a recursive algorithm on single data points instead of a moving window. This reduces memory resources for computation and theoretically directly allows causal filtering without delay, although typically, a delayed estimation is provided Nevertheless, in [RB+14], they also did not exploit the interdependence between velocity and acceleration data but applied separate filters concurrently for each measurement channel.

The algorithms that will be derived in the course of this chapter were originally motivated by the work on PKS [RB+14], due to their proposal to formulate the time-shift as a linear matrix operation, which led to the insight that differentiation of a polynomial can also be formulated as a linear matrix operation. To the best of the author's knowledge, this is the first time to propose smoothing algorithms based on local polynomial regression, which exploits the interrelation of time derivatives between measurement channels. While the first algorithm DeePLS we propose can be seen as a generalization of Savitzky-Golay smoothing, the second algorithm DeePKS is a generalization of its recursive variant, the PKS.

5.3 Background polynomial filtering and smoothing

The Savitzky-Golay algorithm [SG64] is a data smoothing method for scalar-valued, uniformly sampled time series data. It is often referred to as SaG-filter, despite the fact that typically smoothing is performed rather than filtering. The algorithm is based on a model-free moving horizon approach on smoothing and is derived under the assumption that a scalar trajectory can be locally approximated by a polynomial model of a given order n at any given point in time. As already mentioned in the introduction, this assumption is formulated in the Weierstrass' theorem [Bra59]. The Savitzky-Golay algorithm can be viewed as a generalization of moving average smoothing. In the original proposal [SG64], the polynomial coefficients are estimated from data samples given in an *equidistant* time window of length $2M + 1$, which defines the moving window. The resulting polynomial coefficients are optimal in a least squares sense regarding the error between the polynomial function estimate and the noisy measurements available from the trajectory. The uniformly sampled time-series data is centered around the smoothing point, which allows for very efficient calculation of the smoothed value since its calculation is reduced to knowing the zero-th-order polynomial coefficient. A more general derivation of the SaG smoothing algorithm as the one presented in the original paper [SG64], which can also be applied in case measurements are given at arbitrary points in time and for non-symmetric measurement windows, is presented in Section 5.3.1. There, the derivation is directly given in matrix form, which allows for a neater presentation. This derivation is similar to what can be found in Orfanidis [Orf10] and Schafer [Sch11] and results in equations for implementation similar to [Mat19c]. This, together with the preliminaries presented in Section 5.4, builds the necessary foundation to derive the proposed algorithm in Section 5.4.3.

5.3.1 Generic Polynomial Least Squares smoothing

In this subsection, we want to present the equations for the Savitzky-Golay smoother. This builds a good foundation to derive the proposed algorithm. For this presentation, we will directly use matrix notation, in a similar fashion than presented in [Sch11], while we stay even more generic to also allow for non-uniformly sampled data and non-symmetric measurement windows.

Matrix form of polynomial evaluation

Let $\boldsymbol{\theta} \in \mathbb{R}^{r\times 1}$ be defined as

$$\boldsymbol{\theta} \doteq \begin{pmatrix}\theta_n & \theta_{n-1} & \dots & \theta_1 & \theta_0\end{pmatrix}^{\mathsf{T}}, \tag{5.1a}$$

where θ_i corresponds to the i-th order coefficient of a time dependent polynomial function $p_n(t) : \mathbb{R}^1 \to \mathbb{R}^1$ of order n and $\boldsymbol{\theta}$ is of length $r = n+1$. Note that for reasons which will be made clear in Section 5.4.2, the vector $\boldsymbol{\theta}$ is sorted such that the last element is the zero order coefficient θ_0. Then we can evaluate the polynomial function by calculating the vector inner product

$$p_n(t) \doteq \sum_{i=0}^{n} c_i t^i = \boldsymbol{\tau}_n(t)\boldsymbol{\theta}, \tag{5.1b}$$

with the time-feature vector $\boldsymbol{\tau}(t) \in \mathbb{R}^{1\times r}$:

$$\boldsymbol{\tau}(t) \doteq \begin{pmatrix}t^n & t^{n-1} & \dots & t^1 & t^0\end{pmatrix}, \tag{5.1c}$$

containing decreasing powers of the time value t at which the polynomial function shall be evaluated.

Vector-valued polynomial evaluation

We can also evaluate the polynomial at various points in time given by a time vector $\boldsymbol{t} \in \mathbb{R}^{m\times 1}$ defined in $[t_{\min} \in \mathbb{R}^-, t_{\max} \in \mathbb{R}^+]$ and

$$\boldsymbol{t} \doteq \begin{pmatrix}t_{\min} & \dots & t_0 = 0 & \dots & t_{\max}\end{pmatrix}^{\mathsf{T}}, \tag{5.2}$$

which we define such that it includes the time $t_0 = 0$. In order to do so, we define the feature matrix $\mathcal{T}_n \in \mathbb{R}^{m\times r}$ for an n-th order polynomial such that

$$\mathcal{T}_n(\boldsymbol{t}) \doteq \begin{pmatrix}\boldsymbol{t}^{\odot n} & \boldsymbol{t}^{\odot n-1} & \dots & \boldsymbol{t}^{\odot 0}\end{pmatrix}, \tag{5.3}$$

which is of the form

$$\mathcal{T}_n(\boldsymbol{t}) = \begin{pmatrix} t_1^n & t_1^{n-1} & \dots & t_1^0 \\ t_2^n & t_2^{n-1} & \dots & t_2^0 \\ \vdots & \vdots & \ddots & \vdots \\ t_m^n & t_m^{n-1} & \dots & t_m^0 \end{pmatrix} = \begin{pmatrix}\boldsymbol{\tau}(t_1) \\ \boldsymbol{\tau}(t_2) \\ \vdots \\ \boldsymbol{\tau}(t_m)\end{pmatrix}. \tag{5.4}$$

Note that the matrix $\mathcal{T}$ is a right-left flipped version of the well studied *Vandermonde* matrix [Tur66], which also appears in modern implementations of the SaG algorithm [Mat19c]. The fact that we have a flipped version results from our definition of the coefficient and time-feature vectors to carry the zero-order coefficient at the end instead of the beginning.

The response vector $\boldsymbol{x}(\boldsymbol{t})$, which contains the responses of the polynomial function $\mathcal{P}_n(\boldsymbol{t}) : \mathbb{R}^{1\times m} \to \mathbb{R}^{m\times 1}$ at all times given in $\boldsymbol{t}$ can then be calculated by the matrix product

$$\boldsymbol{x}(\boldsymbol{t}) = \mathcal{P}_n(\boldsymbol{t}) = \mathcal{T}_n(\boldsymbol{t})\boldsymbol{\theta}. \tag{5.5}$$

Note that since we defined $t_0 = 0$ to be in $\boldsymbol{t}$, and evaluating a polynomial at its origin is corresponding its zero-order coefficient, we have that

$$x_0 = x(t_0 = 0) = p_n(0) = \theta_0. \tag{5.6}$$

Polynomial Least Squares

Now let $\boldsymbol{y}(\boldsymbol{t})$ be a vector of noisy measurements defined at the times given in $\boldsymbol{t}$. Under a Gaussian noise assumption, it is reasonable to obtain estimates of the parameter vector $\boldsymbol{\theta}$ by performing least squares regression. To be slightly more general, we want to minimize the cost functional J as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ J = \|\boldsymbol{y}(\boldsymbol{t}) - \mathcal{T}\boldsymbol{\theta}\|_W^2 \right\}. \tag{5.7}$$

which is a weighted linear least squares problem with the diagonal weight matrix $W = \operatorname{diag}\{w\}$ and the vector of weights, denoted by w, shall consist of positive, real weights. Choosing W as the identity $\boldsymbol{I}$ would result in regular least squares. While a standard analytical solution to problem (5.7) is known to be

$$\hat{\boldsymbol{\theta}} = \left(\mathcal{T}^\mathsf{T} W \mathcal{T}\right)^{-1} \mathcal{T}^\mathsf{T} W \boldsymbol{y}(\boldsymbol{t}), \tag{5.8}$$

it can be solved more efficiently by means of QR-decomposition. This leads to the solution [Orf10]:

$$QR = W^{\frac{1}{2}} \mathcal{T} \tag{5.9}$$

$$\hat{\boldsymbol{\theta}} = R^{-1}\left(Q^\mathsf{T} W^{\frac{1}{2}} \boldsymbol{y}\right), \tag{5.10}$$

where Q, R are obtained from an economy-sized QR-decomposition.

Evaluation of full polynomial

If we are less interested in the polynomial coefficient vector itself but want to use the polynomial as smoothed approximation $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ at the entire time window defined in $\boldsymbol{t}$, we could do this by inserting (5.60) in (5.5) which leads to an expression similar to what can be found in [Orf10; Mat19c]:

$$\begin{aligned}
\hat{\boldsymbol{x}} &= \mathcal{T} R^{-1}\left[Q^\mathsf{T}\left(W^{\frac{1}{2}} \boldsymbol{y}\right)\right] && \text{(5.11a)}\\
&= \mathcal{T} R^{-1}\left\{\left[W^{\frac{1}{2}}\left(\mathcal{T} R^{-1}\right)\right]^\mathsf{T}\left(W^{\frac{1}{2}} \boldsymbol{y}\right)\right\} && \text{(5.11b)}\\
&= \left(\mathcal{T} R^{-1}\right)\left(\mathcal{T} R^{-1}\right)^\mathsf{T} W \boldsymbol{y} && \text{(5.11c)}\\
&= T T^\mathsf{T} W \boldsymbol{y}, && \text{(5.11d)}
\end{aligned}$$

where we substituted

$$Q = W^{\frac{1}{2}} \mathcal{T} R^{-1}, \tag{5.12}$$

and (5.12) directly follows from (5.59). As an advantage, since R is upper triangular per definition of the QR-decomposition,

$$T = \mathcal{T} R^{-1}, \tag{5.13}$$

can be efficiently calculated by forward-substitution without the need to build the inverse. From (5.14) we can observe that vector $\hat{\boldsymbol{x}}$, representing smoothed values over the entire measurement window, can be calculated by a single matrix multiplication with the measurement vector $\boldsymbol{y}$ as

$$\hat{\boldsymbol{x}} = K \boldsymbol{y}, \tag{5.14}$$

once the kernel matrix $K \doteq T T^\mathsf{T} W$ is calculated.

Polynomial smoothing

Eventually, if we are not even interested in the full estimation vector $\hat{\boldsymbol{x}}$, but want to obtain smoothing at a single value $\hat{x}_i$, within the measurement window, we can write

$$\hat{x}_i = \boldsymbol{\kappa}_i \boldsymbol{y}_i = K_{[i,:]} \boldsymbol{y}_i, \tag{5.15}$$

where $\boldsymbol{\kappa}_i$ denotes the i-th row of the matrix K, corresponding to the index in $\boldsymbol{t}$ where $t = t_i$. To perform smoothing for an entire measurement sequence of arbitrary length, one can proceed in a receding horizon fashion, and repeatedly calculate (5.15) for each measurement window, while moving it along the sequence. Note that the calculation given in (5.15) can only be applied for the steady-state part (for a definition of steady-state and transient part in this context see [Orf10], p.432) of the signal smoothing operation, where the center point of the measurement window resides within the signal $\boldsymbol{y}$, while the transients could be calculated using the full kernel matrix K.

Calculation as a convolution operation

Using the vector product $\hat{\boldsymbol{x}}_i = \boldsymbol{\kappa}_i \boldsymbol{y}_i$ from (5.15) in each step while moving the measurement window forward in time is equivalent to performing the convolution operation

$$\hat{x}_i = \boldsymbol{y} * \operatorname{flip}\{\boldsymbol{\kappa}_i\}. \tag{5.16}$$

using a flipped version of the kernel vector $\boldsymbol{\kappa}_i$. This follows directly from the definition of the discrete finite convolution. Note that if the measurement window is defined symmetric around a center point at which the smoothing should be performed, the vector $\boldsymbol{\kappa}_i$ is also symmetric and hence the convolution can be directly done as $\boldsymbol{y} * \boldsymbol{\kappa}_i$. This is the case in the definition of the original SaG-algorithm. We named the matrix K earlier as *kernel matrix*, while more exactly, we should term it *flipped kernel matrix*.

Polynomial differentiation

A nice property of polynomial smoothing is the possibility to easily perform differentiation of the smoothed signal up to the given polynomial order used for the estimation. If one is not only interested in the smoothed value $x_0 = x(0)$, but additionally in the first k derivatives at time $t_i = 0$, which we denote (in reverse order) as

$$\hat{\boldsymbol{x}}_0^{(k):(0)} \doteq \begin{pmatrix} x^{(k)}(0) & x^{(k-1)}(0) & \dots & \dot{x}(0) & x(0) \end{pmatrix}^{\mathsf{T}}, \tag{5.17}$$

one can write, similar to what can be found in [Orf10; Mat19c], for any $k \leq n$

$$\hat{\boldsymbol{x}}_0^{(k):(0)} = \boldsymbol{g}_k \odot \boldsymbol{\theta}_{[r-k:r]}. \tag{5.18}$$

Here,

$$\boldsymbol{g}_k = \begin{pmatrix} k! & (k-1)! & \dots & 1! & 0! \end{pmatrix}^{\mathsf{T}} \tag{5.19}$$

denotes a column vector of factorials starting from k down to zero. We want to emphasize that it is sufficient to know the last k coefficients from the vector $\boldsymbol{\theta}$, which we denoted by $\boldsymbol{\theta}_{[r-k:r]}$. We will use this fact in our algorithm proposal in Section 5.4.2.

Time shift

If we consider the polynomial function p_n in (5.1b) given at a certain discrete time k:

$$\boldsymbol{x}_k(t) = \tau(t)\boldsymbol{\theta}_k, \tag{5.20}$$

the value $\boldsymbol{x}_k(t)$ is the evaluation of the polynomial at the time relative to the time at the index k, so in abolute term this would be at $k\Delta t + t$. Evaluating the polynomial function at the relative time $t + \Delta t$ will then be

$$\boldsymbol{x}_k(t + \Delta t) = \tau(t + \Delta t)\boldsymbol{\theta}_k, \tag{5.21}$$

where the same coefficient vector $\boldsymbol{\theta}_k$ is used, and only the time operator has to be calculated at the shifted time $(t + \Delta t)$. Now we want to write this with the increased index $k+1$ as

$$\boldsymbol{x}_{k+1}(t) = \tau(t)\boldsymbol{\theta}_{k+1}, \tag{5.22}$$

from where we observe that if we have that $\boldsymbol{x}_k \neq \boldsymbol{x}_{k+1}$, the coefficient vectors also must be different, hence $\boldsymbol{\theta}_k \neq \boldsymbol{\theta}_{k+1}$. Since per definition of the time indices we have that both

$$\boldsymbol{x}_{k+1}(t) = \boldsymbol{x}_k(t + \Delta t) \tag{5.23}$$
$$\tau(t + \Delta t)\boldsymbol{\theta}_k = \tau(t)\boldsymbol{\theta}_{k+1}, \tag{5.24}$$

we can calculate the coefficient vector $\boldsymbol{\theta}_{k+1}$, considering that the vector or matrix given by $\mathcal{T}_n(t)$ is generally not square and hence not invertible, using the Moore-Penrose Pseudo-inverse to get the relation

$$\boldsymbol{\theta}_{k+1} = \mathcal{T}(t)^{\dagger}\mathcal{T}(t + \Delta t)\boldsymbol{\theta}_k. \tag{5.25}$$

This can now also be written as a linear matrix equation of the form

$$\boldsymbol{\theta}_{k+1} = \mathcal{S}\boldsymbol{\theta}_k, \tag{5.26}$$

where we defined a shift operation matrix $\mathcal{S}$ as

$$\mathcal{S} \doteq \mathcal{T}(t)^{\dagger}\mathcal{T}(t + \Delta t). \tag{5.27}$$

Please note that although mathematically correct, calculating the shift matrix this way is not recommended due to the numerical inaccuracy involved, but it is easy to demonstrate the existence of a square shift matrix this way.

Another algorithm for calculating $\mathcal{S}$ for a unit time shift is given in [RB+14] and the reference herein, and it is based on binomial coefficients. We provide a modified and more general version below, which is also suited for time shifts different from $\Delta t = 1$, while it also considers that our coefficient vector $\boldsymbol{\theta}$ has been defined in reverse order, starting from the coefficient of order $n = r - 1$:

$$\mathcal{S}_{i,j} = \begin{cases} \frac{r-j}{(r-i)!(i-j)!} \cdot \Delta t^{(i-j)} & \forall j \leq i \\ 0 & \forall j > i \end{cases}, \tag{5.28}$$

which is nothing else than the binomial coefficient of $r - j$ over $i - j$ for all $j \leq i$. It yields a squared lower triangular matrix of size $r \times r$, which, for example, for a third order polynomial is

$$\mathcal{S}^3(\Delta t = 1) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{5.29}$$

In above formula, the factor $\Delta t^{(i-j)}$ was inspired by the code example provided with [Rho16], where in the comments, credits were given to Bleimund [Ble13] in his hand-written notes.

5.3.2 Polynomial Kalman smoothing

As already stated in the related work section 5.2, Rhode [Rho16] proposed a recursive version of SaG-smoothing using the framework of linear Kalman filtering. The PKS algorithm is built on the local polynomial approximation of the signal which is smoothed, similar as in SaG-smoothing. The core idea is the conclusion that one can formulate the time evolution of the coefficients of the local polynomial, in linear, discrete state space form. This enables the application of the linear Kalman filter framework, as we will show next.

5.3.3 The Polynomial Kalman Smoothing algorithm

Before presenting the PKS algorithm [Rho16], we want to give a derivation in our own words, as background for the derivation of our generalized version of PKS, which we propose in Section 5.4.4. As stated in (5.26), moving a polynomial approximation forward in time, can be stated by the linear matrix equation

$$\boldsymbol{\theta}_{k+1} = \mathcal{S}\boldsymbol{\theta}_k, \tag{5.30}$$

with the shift matrix as defined in (5.28). We further stated in (5.5) that we can calculate the estimated output vector for all times within the moving window, according the linear matrix operation:

$$\hat{\boldsymbol{x}}(\boldsymbol{t}) = \mathcal{T}_n(\boldsymbol{t})\boldsymbol{\theta}. \tag{5.31}$$

and to obtain a single value given at certain time of this window, we had in (5.1b)

$$x(t) = \tau(t)\boldsymbol{\theta}.$$

If we consider an equidistant time grid, this equals for the rightmost value in the current measurement window to

$$\hat{x}(t_{m_r}) = \tau(t_{m_r})\hat{\boldsymbol{\theta}} = \tau(m_r\Delta t)\hat{\boldsymbol{\theta}}, \tag{5.32}$$

where m_r is the length of the right window. Now we can define a system

$$\boldsymbol{\theta}_{k+1} = \mathcal{S}\boldsymbol{\theta}_k + \omega \tag{5.33}$$

$$y = H\hat{\boldsymbol{\theta}} + \nu, \tag{5.34}$$

with the state vector $\boldsymbol{\theta}$, the dynamic matrix $\mathcal{S}$ and the measurement matrix becomes the vector $H = \tau(t_m)$, and the noise terms ω and ν as usual. We now can apply standard linear Kalman filter techniques to estimate the parameters $\hat{\boldsymbol{\theta}}$, from where we can obtain the smoothed value at the window center as

$$\hat{x}(t_0) = \hat{x}_{k-m_r} = \tau(t_0)\hat{\boldsymbol{\theta}}. \tag{5.35}$$

[Rho16] proposed to use a Kalman Filter variant using a forgetting factor λ, to obtain the Polynomial Kalman Smoother (PKS) algorithm given in Algorithm 1, which we present with changed notation for convenience.

5.4 Proposed algorithms

Before we present the novel algorithm for smoothing time-series data consisting of a base channel and its derivatives, we want to give a formal problem formulation for the algorithm to be derived.

Algorithm 1 Polynomial Kalman Smoother [RB+14; Rho16]

```
Require: $\mathcal{S}$, $\mathcal{T}$, $H$, forgetting factor $\lambda$
1: Inputs:
       $\hat{\boldsymbol{\theta}}^{+}_{k-1}$, $P^{+}_{k-1}$, $y_k$
2: $\hat{\boldsymbol{\theta}}^{-}_{k} = \mathcal{S}\hat{\boldsymbol{\theta}}^{+}_{k-1}$    ▷ A priori state estimate
3: $P^{-}_{k} = \mathcal{S}P^{+}_{k-1}\mathcal{S}^{\mathsf{T}}$    ▷ A priori estimation error covariance
4: $K_k = P^{-}_{k}H^{\mathsf{T}}\left(HP^{-}_{k}H^{\mathsf{T}} + \lambda\right)^{-1}$    ▷ (optimal) Kalman Gain (A.75)
5: $P^{+}_{k} = \frac{1}{\lambda}\left(P^{-}_{k} - K_k H P^{-}_{k}\right)$    ▷ Estimation error covariance
6: $\hat{\boldsymbol{\theta}}^{+}_{k} = \hat{\boldsymbol{\theta}}^{-}_{k} + K_k(y_k - H\hat{\boldsymbol{\theta}}^{-}_{k})$    ▷ A posteriori state estimate
7: $\hat{x}_{k-m_r} = \boldsymbol{\tau}(m_r\Delta t)\hat{\boldsymbol{\theta}}^{+}_{k}$
8: return $\hat{x}_{k-m_r}, \hat{\boldsymbol{\theta}}^{+}_{k}, P^{+}_{k}$
```

5.4.1 Problem formulation

We are given measurements y_0 of a signal $x(t) : \mathbb{R}^{+} \to \mathbb{R}$ which is differentiable on $\mathbb{R}^{+}$ of class C^{j}. Additionally, we are a given a number of up to j measurement channels resulting from the derivatives $\dot{x}(t), \ddot{x}(t), \ldots, x^{(j)}(t)$. We seek to find estimations of the channels $\hat{x}, \hat{\dot{x}}, \hat{\ddot{x}}, \ldots, \hat{x}^{(j)}$, or a subset of those, at discrete points in time. We apply a moving horizon or moving window approach, where in each iteration ι, we observe the measurements given within a time window defined in the non-empty set $\mathcal{U}_\iota = \{\mathbb{R} | t_\iota - \Delta t_l \leq t_\iota \leq t_\iota + \Delta t_r\}$ around the window center at t_ι. We denote the concatenation of vectors containing all measurements given within $\mathcal{U}_\iota$ as

$$\boldsymbol{y}_0^{j} = \left[\boldsymbol{y}^{(0)}; \boldsymbol{y}^{(1)}; \ldots; \boldsymbol{y}^{(j)}\right] \in \mathbb{R}^{m}, \tag{5.36}$$

and of their corresponding time stamps *relative* to the window center with

$$\boldsymbol{t} = \left[\boldsymbol{t}_0; \boldsymbol{t}_1; \ldots; \boldsymbol{t}_j\right] \in \mathbb{R}^{m}, \tag{5.37}$$

with $m = m_0 + m_1 + \cdots + m_j$ being the sum of the number of measurements of each channel within the window. Similar, the concatenation of vectors

$$\boldsymbol{x}_0^{j} = [\boldsymbol{x}(\boldsymbol{t}_0); \dot{\boldsymbol{x}}(\boldsymbol{t}_1); \ldots; \boldsymbol{x}^{(j)}(\boldsymbol{t}_j)] \tag{5.38}$$

denotes their corresponding true, but unknown, signal values. Assuming that the measurements are subject to independent, Gaussian measurement noise, we can write

$$\boldsymbol{y}_0^{j} = \boldsymbol{x}_0^{j}(\boldsymbol{t}) + \ \boldsymbol{\xi} \ \approx \ \mathcal{P}_n^{(0):(j)}(\boldsymbol{t}|\boldsymbol{\theta})) \ + \ \boldsymbol{\xi}, \tag{5.39}$$

with noise vector $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{0}, \Sigma)$ and $\Sigma = \operatorname{diag}\left\{\boldsymbol{\sigma}^{\circ 2}\right\}$ is the diagonal covariance matrix of white noise with standard deviations σ_i. Above, $\mathcal{P}_n^{(0):(j)}(\boldsymbol{t}|\boldsymbol{\theta}) : \mathbb{R}^{m\times 1} \to \mathbb{R}^{m\times 1}$ are the polynomial mappings of relative times given in $\boldsymbol{t}$, by which the channels in $\boldsymbol{x}_0^{j}$ are locally approximated, parameterized by their coefficient vector $\boldsymbol{\theta}$. We aim to find the estimates $\hat{\boldsymbol{x}}_0^{j}$ by using a least squares approach, through estimates of the parameter vector $\hat{\boldsymbol{\theta}}$, which is the argument of the optimization

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ \| \boldsymbol{y} - \mathcal{P}_n^{(0):(j)}(\boldsymbol{t}|\boldsymbol{\theta}) \|_W \right\}. \tag{5.40}$$

From there, we can calculate the estimates $\hat{\boldsymbol{x}}$ using the polynomial mapping as $\hat{\boldsymbol{x}} = \mathcal{P}_n^{(0):(j)}\left(\boldsymbol{t}|\hat{\boldsymbol{\theta}}\right)$, while in most cases, it will be sufficient to obtain the smoothed values in the window center at $t = 0$ as

$$\hat{\boldsymbol{x}} = \mathcal{P}_n^{(0):(j)}\left(\boldsymbol{0}|\hat{\boldsymbol{\theta}}\right). \tag{5.41}$$

Defining $\hat{\boldsymbol{x}}_0^{(k):(0)} \doteq [\hat{x}^{(k)}(0);\dots;\dot{\hat{x}}(0);\hat{x}(0)]$, we can formulate the following:

Problem 1. *Given measurement tuples $\langle \boldsymbol{y}, \boldsymbol{t}\rangle$ of a system following the relation described in (5.39), and a diagonal weight matrix W, find estimates $\hat{\boldsymbol{x}}_0^{(k):(0)}$ at the window center of the signal channel x and its derivatives up to the order $k \le j$ from the argument of the minimization of the cost functional in (5.40).*

In the following Sections 5.4.2, 5.4.2 and 5.4.2 we will present some preliminaries to derive the solution to Problem 1 in Section 5.4.3, which will result in Algorithm 2.

5.4.2 Preliminaries

Proposed alternative calculation of kernel vectors

In this section, we want to derive a modified version of calculating the kernel vector $\boldsymbol{\kappa}_i$. This method theoretically allows to obtain $\boldsymbol{\kappa}_i$ more efficiently, which might be beneficial when dealing with non-uniformly sampled data. For invariant time vectors $\boldsymbol{t}$, as it is the case in the original Savitzky-Golay setting, the projection or kernel matrix K and its central row are also invariant, and hence it suffices to calculate them once in advance. In case of non-uniformly sampled signals, the vectors $\boldsymbol{t}$ of relative times will become time-variant. This means that the kernel matrix K has to be repeatedly recalculated within each iteration. We therefore want to present a theoretically more efficient way of calculating the estimated value $\hat{x}_0$, or more generally, the vector $\hat{\boldsymbol{x}}_0^{(k):(0)}$ of all derivatives up to order k. We had in (5.18) that

$$\hat{\boldsymbol{x}}_0^{(k):(0)} = \boldsymbol{g}_i \odot \hat{\boldsymbol{\theta}}_{[r-k:r]}$$

can be used to evaluate the polynomial p_n for x_0 at time $t_i = 0$ and get subsequent time derivatives in one step, where only the parameters of order zero to k are needed. While full vector $\hat{\boldsymbol{\theta}}$ containing the polynomial coefficients was given in (5.60) as

$$\hat{\boldsymbol{\theta}} = R^{-1}(Q^T W^{\frac{1}{2}} \boldsymbol{y}), \tag{5.42}$$

we can calculate the reduced vector $\hat{\boldsymbol{\theta}}_{[r-k:r]}$ directly from reduced matrices as

$$\hat{\boldsymbol{\theta}}_{[r-k:r]} = \tilde{R}^{-1} \tilde{Q}^{\mathsf{T}} W^{\frac{1}{2}} \boldsymbol{y}, \tag{5.43}$$

where

$$\tilde{Q}^{\mathsf{T}} = Q^{\mathsf{T}}_{[r-k:r,:]} \tag{5.44}$$

$$\tilde{R} = R_{[r-k:r,r-k:r]} \tag{5.45}$$

are the last k rows of the transposed matrix Q and the lower right block of size k of R. The vector of desired derivatives up to order k can hence be rewritten using the reduced kernel matrix $\tilde{K}_i$ as

$$\hat{\boldsymbol{x}}_0^{(k):(0)} = \tilde{K}_k \boldsymbol{y}, \tag{5.46}$$

with

$$\tilde{K}_k = \boldsymbol{g}_k \odot \left(\tilde{R}^{-1} \tilde{Q}^{\mathsf{T}} W^{\frac{1}{2}}\right) = \boldsymbol{g}_k \odot \tilde{A}_k. \tag{5.47}$$

where we introduced

$$\tilde{A}_k \doteq \tilde{R}^{-1} \tilde{Q}^{\mathsf{T}} W^{\frac{1}{2}}. \tag{5.48}$$

Note that above calculation can be efficiently obtained by means of backward-substitution without the need of inverting R, because R is upper triangular [Bon21], p.19, (2.2). For the case one only wants to calculate the zero order coefficient $\hat{\theta}_0 = \hat{x}_0$, the matrices in (5.43) even reduce to the last diagonal entry of the matrix R and the last row of the transposed matrix Q^T:

$$\rho = R_{[r,r]} \tag{5.49}$$

$$\boldsymbol{q} = Q^\mathsf{T}_{[r,:]} \tag{5.50}$$

and we are able to write

$$\hat{\theta}_0 = \frac{1}{\rho}\boldsymbol{q} W^{\frac{1}{2}} \boldsymbol{y}. \tag{5.51}$$

It is worth mentioning that for this to work, we had to use a vector $\boldsymbol{\theta}$ defined in a way that the *last* element corresponds to the zero-order coefficient θ_0. This theoretically reduces the computational complexity for the calculation of A from $\mathcal{O}(r^2)$ to $\mathcal{O}(k^2)$ [FM67], while it remains to be investigated how this reduction can improve the overall performance of the algorithm, given a certain overhead of indexing into the full matrices. See also Section 5.5.4.

Differentiation matrix

In this section, we want to present a generic way of performing differentiation of polynomial functions using a linear matrix operation. This helps to derive the proposed algorithm. While (5.18) allows to find x_i and its derivatives only at time $t_i = 0$, we need to write down the differentiation of an entire vector $\boldsymbol{x}$ in a different way. We can write for the zero-th, first, and j-th derivative of $\mathcal{P}_n(t)$:

$$\boldsymbol{x}(t) = \mathcal{T}_n(t)\cdot\boldsymbol{\theta} = \mathcal{T}_n(t)\cdot\mathcal{D}_n^0\cdot\boldsymbol{\theta} \tag{5.52a}$$

$$\dot{\boldsymbol{x}}(t) = \frac{d}{dt}\cdot\{\mathcal{T}_n(t)\cdot\boldsymbol{\theta}\} = \mathcal{T}_n(t)\cdot\mathcal{D}_n^1\cdot\boldsymbol{\theta} \tag{5.52b}$$

$$\boldsymbol{x}^{(j)}(t) = \frac{d^{(i)}}{dt^{(j)}}\cdot\{\mathcal{T}_n(t)\cdot\boldsymbol{\theta}\} = \mathcal{T}_n(t)\cdot\mathcal{D}_n^j\cdot\boldsymbol{\theta}. \tag{5.52c}$$

where we used differentiation matrix $\mathcal{D}_n \in \mathbb{R}^{r\times r}$ of order n for $\mathcal{P}_n(t)$ given as

$$\mathcal{D}_n \doteq \begin{pmatrix} 0 & \cdots & 0 & 0 & 0 \\ n & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 2 & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}, \tag{5.53}$$

and $\mathcal{D}^0 = \mathbb{I}$ is the identity matrix. This finding is similar to the one found in [Ami16], (1.4) for polynomials of a monomial basis, but respecting the fact that we ordered the polynomial coefficients in $\boldsymbol{\theta}$ such that the last element corresponds to the zero order coefficient.

Integration operator

Although not necessarily needed for the derivation of the DeePLS algorithm, we want to mention that integration as the inverse operation to differentiation can also be defined using the derivative operation matrix $\mathcal{D}$. $\mathcal{D}$ is square but does not have full rank, hence it is not invertible. This becomes clear since when differentiating, the zero order term information is lost during the operation, and therefore the operation is not invertible without loss of information. Nevertheless, all other coefficients appear in the differentiated polynomial, and we can recover the original $\theta^{(0)}$ only by setting the lost term $c_0 \doteq 0$. This can be done using the Moore-Penrose pseudo-inverse of the differentiation operation matrix, and we can define the integration operation as the inverse of differentiation as

$$\mathcal{D}^{-j} = \mathcal{D}^{(j)\dagger}. \tag{5.54}$$

We can see from

$$\int \dot{x} = x(t) - x(0) = x(t) - c_0 = \tau_n(t)\mathcal{D}^{(1)\dagger}\mathcal{D}^{(1)}\theta^{(0)}, \tag{5.55}$$

that only part of the equation is recovered on the right hand side by the expression $\mathcal{D}^{\dagger}\mathcal{D}^{(1)}$ since the zero order coefficient c_0 only appears on the left side of the equation. Indeed, one can show that

$$\mathcal{D}^{\dagger}\mathcal{D}^{(1)} = \begin{pmatrix} 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 0 \end{pmatrix}, \tag{5.56}$$

which is the identity matrix with a zero instead of a one at the element in lower right corner of the matrix. We will use this in order to calculate the integral of a velocity signal defined by their polynomial coefficients in the simulation example presented in Section 5.5.1.

5.4.3 Derivative-exploiting Polynomial Least Squares (DeePLS)

Now we are ready to derive the main proposal of this article. This can be seen as an enhancement and generalization of polynomial smoothing, and can be applied if a multitude of measurement channels is available, which are derivatives of each other. Since the proposed algorithm is able to exploit the interrelation between the measurement channels, we want to name it as *Derivative-exploiting Polynomial Least Squares (DeePLS)*. It uses a moving window approach to calculate smoothed values by operating on all given and noisy measurement channels within the current window concurrently, and fitting a polynomial by means of weighted least squares. Standard polynomial least squares, as described earlier in Section 5.3.1, as well as the SaG algorithm, to the contrary, only operate on a single measurement channel.

In the following, we want to look at the evaluation steps within a single window, while these steps have to be repeated in a receding horizon fashion if we desire to smooth (or filter) an entire measurement sequence. The DeePLS algorithm will hence be the solution for Problem 1: It can easily be seen from (5.52) that the vertical concatenation of several derivatives of $\boldsymbol{x}^{(i)}$ can be expressed as

$$\boldsymbol{x}_j^0 = \begin{pmatrix} \boldsymbol{x}^{(0)}(t) \\ \boldsymbol{x}^{(1)}(t) \\ \vdots \\ \boldsymbol{x}^{(j)}(t) \end{pmatrix} = \mathcal{P}_n^{(0):(j)}(\boldsymbol{t}\,|\,\boldsymbol{\theta}) = \begin{pmatrix} \mathcal{T}_n(\boldsymbol{t}_0)\mathcal{D}_n^0 \\ \mathcal{T}_n(\boldsymbol{t}_1)\mathcal{D}_n^1 \\ \vdots \\ \mathcal{T}_n(\boldsymbol{t}_j)\mathcal{D}_n^j \end{pmatrix} \boldsymbol{\theta} \doteq \Phi(\boldsymbol{t}_j^0)\boldsymbol{\theta}, \tag{5.57}$$

which is a linear matrix operation. Remember that $\mathcal{D}_n^0 = \mathbb{I}$ was defined to be the identity matrix of appropriate size. We can rewrite Problem 1 with

$$\Phi(\boldsymbol{t}_j^0) \doteq \begin{pmatrix} \mathcal{T}_n(\boldsymbol{t}_0)\mathcal{D}_n^0 \\ \mathcal{T}_n(\boldsymbol{t}_1)\mathcal{D}_n^1 \\ \vdots \\ \mathcal{T}_n(\boldsymbol{t}_j)\mathcal{D}_n^j \end{pmatrix}, \tag{5.58}$$

as the weighted linear least squares problem (5.7)

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ J = \|\boldsymbol{y}(\boldsymbol{t}) - \Phi\boldsymbol{\theta}\|_W^2 \right\}. \tag{5.7 revisited}$$

with diagonal weight matrix $W = \operatorname{diag}\{w\}$ and the vector of weights denoted w consisting of positive, real weights. This has the well known solution (see for example [Man80])

$$QR = W^{\frac{1}{2}}\Phi \tag{5.59}$$

$$\hat{\boldsymbol{\theta}} = R^{-1}Q^{\mathsf{T}}W^{\frac{1}{2}}\boldsymbol{y} = A\boldsymbol{y}, \tag{5.60}$$

where Q, R are obtained from an economy-sized QR-decomposition, and

$$A \doteq R^{-1}Q^{\mathsf{T}}W^{\frac{1}{2}}. \tag{5.61}$$

Combining (5.60) and (5.18), we can state that the vector $\hat{\boldsymbol{x}}_0^{(k):(0)}$ of derivatives of the polynomial function $\mathcal{P}(0|\boldsymbol{\theta})$, evaluated at the center $t = 0$, is given, for any $k \leq n$, as

$$\hat{\boldsymbol{x}}_0^{(k):(0)} = \boldsymbol{g}_k \odot \hat{\boldsymbol{\theta}}_{[r-k:r]} = \boldsymbol{g}_k \odot \tilde{A}_k\boldsymbol{y} = \tilde{K}_k\boldsymbol{y}, \tag{5.62}$$

where

$$\boldsymbol{g}_k \doteq \begin{pmatrix} k! & (k-1)! & \dots & 1! & 0! \end{pmatrix}^{\mathsf{T}} \tag{5.19 revisited}$$

denotes a column vector of factorials starting from k down to zero, and

$$\tilde{A}_k \doteq A_{[r-k:r,:]} \tag{5.63}$$

is the sub-matrix formed by the lower k rows of A, and

$$K_k \doteq \boldsymbol{g}_k \odot \tilde{A}_k. \tag{5.64}$$

A summary of the proposed DeePLS algorithm can be found below in Algorithm 2.

Structure of the DeePLS algorithm

Figure 5.1 illustrates the structure of the DeePLS algorithm, operating on uniformly sampled data (left) and sparse, non-uniformly sampled data (right). Data processing of time-series measurements of a base channel (blue dots) together with their derivatives (red dots) is illustrated in a top to bottom fashion. The measurement vectors containing data within the moving window are first concatenated into a single vector in a multi-channel concatenation layer. The resulting vector is then convoluted with the physics-informed kernel, which is the main building block of DeePLS. In this example, we illustrate a kernel which consists of three rows. Convolution with the first row (light blue) results in the smoothed value of the base channel at the window center (blue triangle). The smoothed value for the derivative $\hat{\dot{x}}$ at the center position is obtained by convoluting the concatenated measurement vector with the second row of the kernel. Additionally, the illustration shows the possibility to obtain the second derivative of the base channel as a latent variable (yellowish), for which a third row in the kernel matrix was used. On the right hand side, we can see how the algorithm is applied to sparse and non-uniformly sampled data. This is possible as long as enough measurement points are available (on all channels together) for the regression to be well-defined. Otherwise, no meaningful kernel can be calculated to obtain smoothing.

To summarize, the DeePLS algorithm can be realized by a neural network structure, which consists of a concatenation layer and a convolution layer. The kernel of the latter can be designed in an optimal fashion thanks to the interpretation as a polynomial least squares smoothing algorithm. This network structure can be used as smoothing layer and allows to process non-uniformly sampled and sparse time-series data. Additionally, derivatives of signals can be calculated as latent variables, which potentially improve capabilities and learning of neural networks applied to physical systems.

Algorithm 2 Proposed Derivative-exploiting Polynomial Least Squares (DeePLS) algorithm.

```
Require: weight matrix $W$, differentiation matrix $\mathcal{D}_n$ of polynomial order $n$ from (5.53),
    vector of factorials $\boldsymbol{g}_k$ from (5.19), maximum order $k$ of derivatives to be calculated
 1: Inputs:
        $< \boldsymbol{y}^0, \boldsymbol{t}^0 >, \dots, < \boldsymbol{y}^j, \boldsymbol{t}^j >$
 2: $\boldsymbol{y}_0^j \leftarrow$ concatenate$\{\boldsymbol{y}_0; \dots; \boldsymbol{y}_j\}$
 3: for all $l$ in $[0, j]$ do
 4:     $\mathcal{T}_n(\boldsymbol{t}_l) \leftarrow \begin{pmatrix} \boldsymbol{t}_l^{\odot n} & \boldsymbol{t}_l^{\odot n-1} & \dots & \boldsymbol{t}_l^{\odot 0} \end{pmatrix}$    ▷ (5.3)
 5: end for
 6: $\Phi \leftarrow \begin{pmatrix} \mathcal{T}_n(\boldsymbol{t}_0)\mathcal{D}_n^0 \\ \mathcal{T}_n(\boldsymbol{t}_1)\mathcal{D}_n^1 \\ \vdots \\ \mathcal{T}_n(\boldsymbol{t}_j)\mathcal{D}_n^j \end{pmatrix}$    ▷ (5.58)
 7: $Q, R \leftarrow \mathrm{qr}\left\{W^{\frac{1}{2}}\Phi\right\}$    ▷ (5.59)
 8: $\tilde{Q}^\mathsf{T} \leftarrow$ last $k$ rows of $Q^\mathsf{T}$    ▷ (5.44)
 9: $\tilde{R} \leftarrow$ lower-right block sized $k \times k$ of $R$    ▷ (5.45)
10: $\tilde{A}_k \leftarrow \tilde{R}^{-1}\tilde{Q}^\mathsf{T}W^{\frac{1}{2}}$ (by backward - substitution)    ▷ (5.48)
11: $K_k \leftarrow \boldsymbol{g}_k \odot \tilde{A}_k$    ▷ (5.64)
12: $\hat{\boldsymbol{x}}_0^{(k):(0)} \leftarrow \tilde{K}_k \boldsymbol{y}$    ▷ (5.62)
13: return $\hat{\boldsymbol{x}}_0^{(k):(0)}$
```

Discussion of the proposed DeePLS algorithm

DeePLS reduces to the original SaG algorithm [SG64] in case that (a) no derivative measurement information is available and only the zero-order channel is present in the signal, and (b) for the case that the signal is uniformly sampled, and (c) the window is defined symmetrically around the smoothing point $t_0 = 0$.

The DeePLS algorithm consists of two main parts: (a) the calculation of optimal kernels and (b) their convolution with a concatenated measurement vector, which is produced from moving window data of all measurement channels. The kernels are optimal in the sense that they minimize the weighted squared errors between the signal and a local polynomial approximation and can be efficiently calculated by means of QR-decomposition. The convolution operation is identical to a vector dot product operation.

For the solution of the weighted least squares problem (5.40) to be well-defined, we need at least a total number of $m \geq r$ measurements contained in the measurement window, as we have r unknown parameters. For $m = r$, an exact solution is obtained, and no smoothing effect occurs. The fact that the number of measurement points available for the regression is given as $m = m_0 + m_1 + \cdots + m_j$, and hence always higher than treating each channel individually, has two implications. First, smoothing is already possible when looking at smaller window sizes compared to conventional, single-channel polynomial smoothing. This reduces the time delay for causal filtering. Second, additional measurement values are available compared to single channel SaG over the same time span. This effect is one explanation of why the smoothing errors can be reduced with the proposed method.

Note that in case of uniformly sampled data and time-invariant weights, kernel matrix K_k typically remains constant, and steps 2 to 11 of Algorithm 2 can be moved out to an initialization phase and do not have to be performed within each iteration. This follows for the fact that for time-invariant relative time vectors $\boldsymbol{t}$ and time-invariant weights w, matrix Φ is also time-invariant. Another reason for time-variant relative time vectors $\boldsymbol{t}$ could be sparse measurement vectors due to missing values. Then, the computational load for each iteration

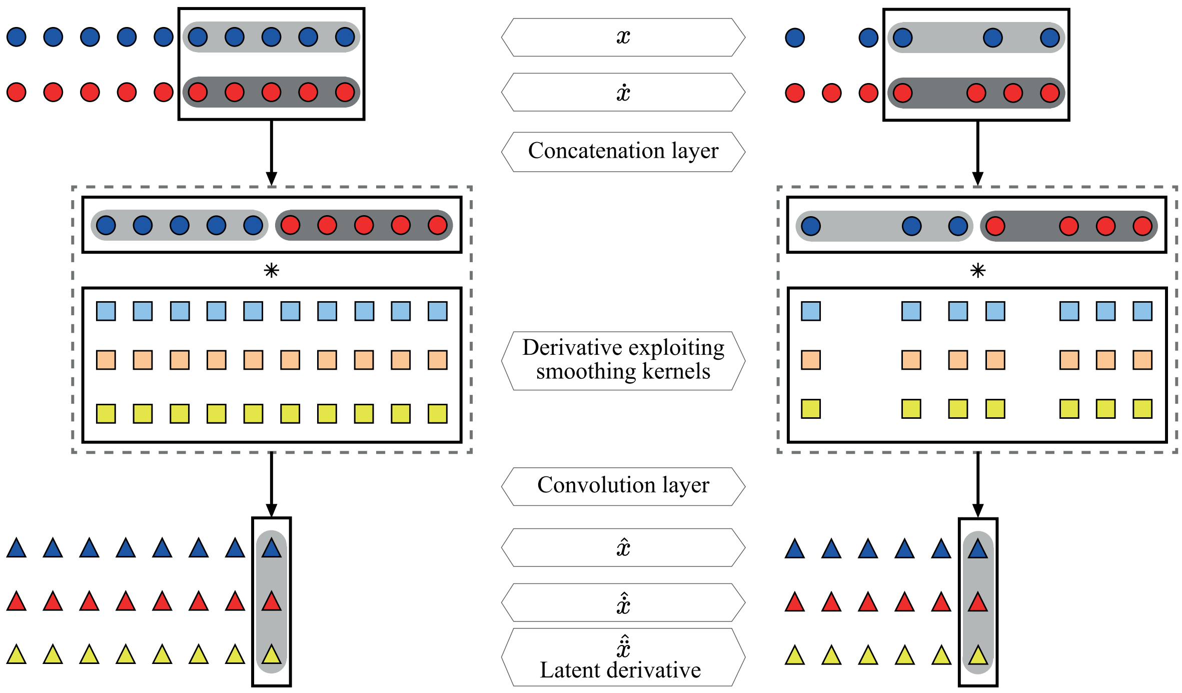

Figure 5.1: Illustration of the structure of the DeePLS algorithm. Left: operating on uniformly sampled data. Right: DeePLS for re-sampling of non-uniformly sampled or sparse measurements.

reduces to the multiplication of matrix $\tilde{K}_k$ with the enhanced measurement vector $\boldsymbol{y}$.

Note that, for ease of notation, the formulation of Algorithm 2 is given such that all subsequent derivatives have to be present, but matrix Φ can easily be reformulated by omitting missing measurement channels if only a subset of time-derivatives is given.

Hyper-parameters of DeePLS are (a) the polynomial order of the local approximation and (b) the size of the measurement window. Additionally, in DeePLS, it is necessary to (c) provide adequate weights for the weighted least squares operation. Theoretically, in the presence of Gaussian noise, matrix W has to be constructed square diagonal with inverses of the variances $\sigma_0^2, \sigma_1^2, \dots, \sigma_j^2$ for each corresponding signal in the main diagonal (see Section 5.5.2). Arbitrary weights could be used instead, for example, to put more emphasis on less recent measurements. This could be desirable when choosing the smoothing points $t_0 = 0$ such that they lie on the right edge of the window, and hence $\Delta t_r = 0$, which allows the use of the framework as a causal filter instead of a smoother one. Typically, when used as a smoothing algorithm, one might choose the smoothing point to a central position within the moving window because the estimation error obtained from the polynomial regression elsewhere is higher.

Note that the calculation given in (5.18) can only be applied for the steady-state part (see [Orf10], p.432) of the signal smoothing operation, while the transients need to be calculated from (5.58) using an appropriate time vector.

5.4.4 Derivative-exploiting Polynomial Kalman Smoothing (DeePKS)

In analogy to the derivation of the Polynomial Kalman Smoother (PKS), we can formulate a recursive version of the previously proposed DeePLS algorithm using the Kalman Filter framework. We will name the resulting algorithm Derivative-exploiting Polynomial Kalman Smoother (DeePKS). Let us assume as for the derivation of the DeePLS algorithm, that we are observing a multi-channel measurement signal, consisting of a base channel and some of its higher order derivatives. We are given a collection of measurement tuples

$$\mathcal{Y}_k = \left\{ < \boldsymbol{y}_k^0, \boldsymbol{t}_k^0 >, \ldots, < \boldsymbol{y}_k^j, \boldsymbol{t}_k^j > \right\}, \tag{5.65}$$

where $\boldsymbol{y}^i$ is the measurement vector belonging to the i-th derivative, together with their corresponding time stamps relative to the center point of the measurement window at the current iteration k. With the concatenation of all measurements, $\boldsymbol{y} \in \mathbb{R}^m$ and their true, unnoisy counterparts $\boldsymbol{x} \in \mathbb{R}^m$, we can write

$$\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{x}(\boldsymbol{t}) + \xi = \Phi(\boldsymbol{t})\boldsymbol{\theta} + \xi, \tag{5.66}$$

where the matrix $\Phi(\boldsymbol{t})$ was defined in (5.58) as

$$\Phi(\boldsymbol{t}) \doteq \begin{pmatrix} \mathcal{T}_n(\boldsymbol{t}^{(0)})\mathcal{D}_n^0 \\ \mathcal{T}_n(\boldsymbol{t}^{(1)})\mathcal{D}_n^1 \\ \vdots \\ \mathcal{T}_n(\boldsymbol{t}^{(j)})\mathcal{D}_n^j \end{pmatrix}. \tag{5.67}$$

Note that Φ is defined for an arbitrary amount of time-value vectors with relative times around the center of a measurement window, and since the information to which channel a certain time value is belonging to is encoded in $\mathcal{Y}$, we could also write:

$$\boldsymbol{y}_{\text{new}} = \Phi_{\boldsymbol{t}_{\text{new}}}\boldsymbol{\theta} + \xi \tag{5.68}$$

to denote that we wanted to calculate the Φ-matrix from the time vectors of new, unseen measurement tuples defined at $\boldsymbol{t}_{\text{new}}$, and further define $\Phi_{\boldsymbol{t}_{\text{new}}} = \Phi(\mathcal{Y}_{\text{new}})$. Note that this also includes the case of new observations only given at times at the rightmost edge of the measurement window. Note also that (5.68) is a measurement equation that is linear in the parameters $\boldsymbol{\theta}$. In addition, given an estimate for the parameter vector $\hat{\boldsymbol{\theta}}$, we could define a different matrix Φ to obtain all the estimates of all channels but only at the center point t_0 and write

$$\hat{\boldsymbol{x}}_{t_0} = \Phi(t_0)\hat{\boldsymbol{\theta}}. \tag{5.69}$$

We are now ready to express the following linear state-space system by combining (5.30) and (5.68) as follows

$$\boldsymbol{\theta}_{k+1} = \mathcal{S}\boldsymbol{\theta}_k + \omega_k \tag{5.70}$$

$$\boldsymbol{y}_{\text{new},k} = \Phi_{\boldsymbol{t}_{\text{new}},k}\boldsymbol{\theta}_k + \xi_k. \tag{5.71}$$

Above, $\boldsymbol{\theta}$ was a vector containing the parameters of the *current* local approximation, $\mathcal{S}$ was the shift matrix from (5.28), which allows to propagate the parameters forward in time to the next center point of the measurement window in the next iteration. The measurement tuples

$$\mathcal{Y}_{new} = \left\{ < \boldsymbol{y}_{\text{new}}^0, \boldsymbol{t}_{\text{new}}^0 >, \ldots, < \boldsymbol{y}_{\text{new}}^j, \boldsymbol{t}_{\text{new}}^j > \right\} \tag{5.72}$$

contain new, unseen measurement values together with their time stamps relative to the current center point of the measurement window. ω and ξ are noise vectors of appropriate size.

Similar to the derivation of the Polynomial Kalman Smoother (PKS), we can now use a linear Kalman filter to obtain estimates $\hat{\boldsymbol{\theta}}$ at each iteration. This is the optimal, recursive solution under the assumptions that (1) the parameters follow the random walk process given in (5.70) and (2) the noise vectors are drawn from zero mean normal distributions, where

$$\omega \sim \mathcal{N}(0, Q) \tag{5.73}$$

$$\xi \sim \mathcal{N}(0, R). \tag{5.74}$$

Also, $\boldsymbol{y}_{\text{new}}$ should contain, as per definition, only previously unseen measurements. From the Kalman filter, we obtain the estimated polynomial parameters, which we can then use to obtain estimates at desired positions relative in time and of arbitrary channels of the base channel and its derivatives by using the relation

$$\hat{\boldsymbol{x}}_{\mathrm{t}_i,k} = \Phi_{\mathrm{t}_i,k}\, \hat{\boldsymbol{\theta}}_k. \tag{5.75}$$

The resulting steps of the proposed Derivative-exploiting Polynomial Kalman Smoother DeePKS are summarized in Algorithm 3.

Algorithm 3 Proposed Derivative-exploiting Polynomial Kalman Smoother (DeePKS)

```
Require: Shift matrix $\mathcal{S}$                    ▷ as defined in (5.28)
1: Inputs:
     $\mathcal{Y}_k^{new} = \left\{ < \boldsymbol{y}_{\text{new}}^0, \boldsymbol{t}_{\text{new}}^0 >, \dots, < \boldsymbol{y}_{\text{new}}^j, \boldsymbol{t}_{\text{new}}^j > \right\}, \hat{\boldsymbol{\theta}}_{k-1}^+, P_{k-1}^+, R_k, Q_{k-1}$
2: $\hat{\boldsymbol{\theta}}_k^- = \mathcal{S}\hat{\boldsymbol{\theta}}_{k-1}^+$                    ▷ A priori state estimate
3: $P_k^- = \mathcal{S}P_{k-1}^+\mathcal{S}^{\mathsf{T}} + Q_{k-1}$                    ▷ A priori estimation error covariance
4: $H = \Phi\left(\mathcal{Y}_k^{new}\right)$                    ▷ as defined in (5.67)
5: $K_k = P_k^- H_k^{\mathsf{T}}\left(H_k P_k^- H_k^{\mathsf{T}} + R_k\right)^{-1}$                    ▷ Kalman Gain (A.75)
6: $P_k^+ = \left(P_k^- - K_k H_k P_k^-\right)$                    ▷ Estimation error covariance
7: $\hat{\boldsymbol{\theta}}_k^+ = \hat{\boldsymbol{\theta}}_k^- + K_k(\boldsymbol{y}_{\text{new},k} - H_k\hat{\boldsymbol{\theta}}_k^-)$                    ▷ A posteriori state estimate
8: $\hat{\boldsymbol{x}}_{\mathrm{t}_i,k} = \Phi(t_i)\hat{\boldsymbol{\theta}}_k^+$                    ▷ Estimates of channels values at desired time $\mathrm{t}_i$
9: return $\hat{\boldsymbol{x}}_{\mathrm{t}_i,k}, \hat{\boldsymbol{\theta}}_k^+, P_k^+$
```

Discussion of the proposed DeePKS algorithm

In the same way as the PKS algorithm [Rho16] can be interpreted as a recursive version of the well-known SaG algorithm [SG64], DeePKS can be seen as a recursive version of DeePLS. Hence, similar to before, DeePKS reduces to PKS in case (a) no derivative measurement information is available, and only the zero-order channel is present in the signal, and (b) for the case that the signal is uniformly sampled. Having said that, the following advantages compared to PKS smoothing arise:

- DeePKS is able to incorporate information from multiple measurement channels in each step
- The interrelation between these channels can be leveraged
- Thanks to its generic formulation, the application to non-uniformly sampled measurement data is enabled
- It is able to process multiple measurements in one calculation step. This includes not only data from the different, interrelated measurement channels but also delayed measurement tuples.

Both PKS and DeePKS are able to be used as causal filters instead of smoothing by setting the center point to the rightmost edge(s) of the measurement window. Nevertheless, as with any filtering technique, this will deteriorate the performance of the algorithms compared to the smoothing result. The polynomial approximation works best at a true center point, which is supported by data on both sides to the left and the right. Although we do not show these results for the sake of brevity, this holds still true even for the recursive variant, which never really performs any fitting directly to the data.

Also, for both PKS and DeePKS, recursive filtering and smoothing are realized by means of Kalman Filtering. This is possible only by leveraging a state (aka parameter) propagation model. This model is based on the assumption that the polynomial coefficients remain constant, except for the random walk part of the equation with additive Gaussian noise. This means that if the true process follows a trajectory that is described by a polynomial (of the order chosen in the algorithms), both PKS and DeePKS are theoretically able to asymptotically converge to the true trajectories. This assumption will, in practice, not hold. We will continue a discussion of how this behavior could still be leveraged in the future in Section 9.3.

5.5 Evaluation

In order to evaluate the proposed DeePLS and DeePKS algorithm, we run a simulation experiment to obtain smoothed estimations and compare the results to the ones calculated by the standard SaG algorithm and its recursive counterpart PKS. We want to look at an example which seems realistic in the context of longitudinal vehicle control, and design a sufficiently differentiable longitudinal velocity trajectory, together with their corresponding trajectories of distance and acceleration based on quintic polynomials. These have been used earlier in the context of automated vehicle trajectory planning, for example in Werling [Wer10]. Find details about the experiments and hyper-parameters in Sections 5.5.1 and 5.5.2. An evaluation of the computational times needed to execute the algorithms is given in Section 5.5.4.

5.5.1 Experiment description

We define a set of polynomials using data tuples given in Table 5.1. Each row in Table 5.1 defines a quintic polynomial starting at time t_{start} and ending at t_{end}. The velocities at the start and end points are given by v_{start} and v_{end}, respectively. The polynomial coefficients are obtained with the additional condition that the acceleration at the start- and end-times has to be zero to achieve smooth transitions with phases of piece-wise constant velocity trajectories, which are used to fill the gaps between the polynomials.

Based from this reference velocity data, trajectories for distance and acceleration are then calculated by integrating and differentiating, as shown in Section 5.4.2. All three measurement data channels are then generated by uniformly sampling the reference trajectories in a time grid from $t = 0$ to $t = 50s$ with a sample time of $T_s = 0.1s$, and adding zero-mean, Gaussian noise with a standard deviation of $\sigma_d = 10$, $\sigma_v = 0.3$, $\sigma_a = 0.2$ for distance, velocity and acceleration, respectively.

5.5.2 Implementation details

We performed smoothing within a symmetric moving window and a central smoothing point as required for SaG. All algorithms (SaG, DeePLS, PKS and DeePKS) ran with a window size

Table 5.1: Data values defining piece-wise polynomial velocity profile

t_{start}	t_{end}	v_{start}	v_{end}
0	6	0	5
7	13	5	2
16	24	2	5
27	33	5	3
37	43	3	5
44	50	5	0

of $M = 4$, which means that SaG ran on $2M + 1 = 9$ samples, while DeePLS could exploit a number of $m_0^j = 27$ measurements within the same moving window. PKS and DeePKS were using the rightmost value(s) for update in a recursive manner. The polynomial order was set to $n = 5$. For DeePLS, we set the weights to the inverses of the variances, leading to constant weights $w_i = 1/\sigma_i^2$ over the entire measurement window for each measurement channel. The weighting matrix W was calculated using

$$W = \operatorname{diag}\left\{\left[\frac{1}{\sigma_0^2}, \ldots, \frac{1}{\sigma_0^2}, \frac{1}{\sigma_1^2}, \ldots, \frac{1}{\sigma_1^2}, \ldots, \frac{1}{\sigma_j^2}, \ldots, \frac{1}{\sigma_j^2}\right]\right\}. \tag{5.76}$$

To evaluate the statistics of the results, Normalized Root Mean Square Error (NRMSE) was calculated using

$$\text{NRMSE} = 1 - \frac{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|}{\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|}, \tag{5.77}$$

where $\bar{\boldsymbol{x}}$ denotes the mean value of $\boldsymbol{x}$. Values in Table 5.2 show mean values of NRMSE of static parts of the results over all Monte Carlo runs. Note that the simulation example shown was performed with uniformly sampled data only to make it comparable to SaG-smoothing, but similar improvements can be expected for non-uniformly sampled data, while it remains future work to find how the possibility of applying the proposed smoothing algorithms to non-uniformly sampled and sparse measurement data can be beneficial.

5.5.3 Results

Figures 5.2 and 5.3 show the results. Figure 5.2 serves to show an overview over the trajectories used in the experiment, while the smoothing results can be seen in Figure 5.3. The latter provides a zoom into the region marked by a rectangle in the corresponding trajectory of Figure 5.2.

We can see the ground truth together with the noisy measurement data, as well as the smoothed values obtained by various smoothing methods. SaG and PKS serve as baselines, to evaluate the proposed DeePLS and DeePKS algorithms.

First, in Figure 5.3 (top), a qualitative comparison is given, comparing ground truth trajectories with the estimation results from both SaG and the proposed DeePLS algorithm. Figure 5.3 (second row) shows the same comparison with PKS as a baseline together with the proposed DeePKS results. In all rows of Figure 5.3, we can see results for the distance on the left, the velocity in the center and the acceleration on the right hand side.

At the bottom of Figure 5.3, we illustrate the corresponding box plots for each of the three signal channels, comparing SaG, PKS, DeePLS and DeePKS. In addition, a quantitative comparison using NRMSE values, (both obtained from a series of 1000 Monte-Carlo runs),

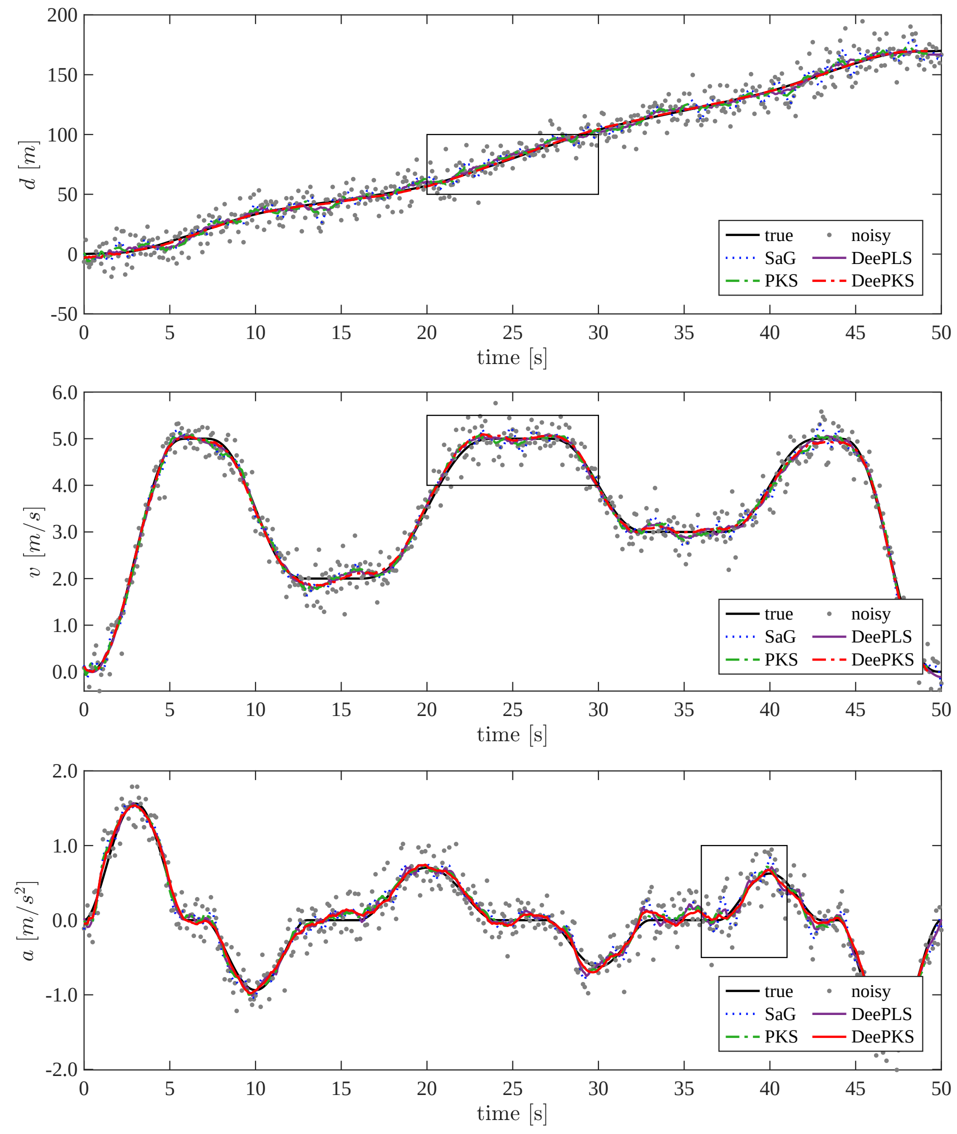

Figure 5.2: Overview of comparison of the true, noisy and estimated values for the proposed DeePLS and DeePKS algorithms with results obtained by the Savitzky-Golay (SG) and the PKS method. Find a zoom into the areas marked by rectangles in Figure 5.3.

is given in Table 5.2. We can observe that both DeePLS and DeePKS clearly outperform their non-derivative exploiting counterpart.

5.5.4 Efficiency analysis of DeePLS algorithm

We performed an analysis of computation times for parts of the DeePLS algorithm to investigate the effect of the proposed reduced order computation. The experiments were performed on an ASUS ZenBook i7-10510U running Windows 10 Home with an Intel(R) Core(TM) i7-10510U CPU @ 1.80 GHz (2.30 GHz TDP-up) with 16 GB RAM. The algorithms were implemented in MATLAB® 2019b. Although Lines 3-11 of Algorithm 2 could have been run outside the moving window iteration, since the kernel matrix in this example is time-invariant, we ran the full algorithm for the sole purpose of evaluating the computation times. We run 1000 episodes and calculated the averaged evaluation times for calculating lines 7-11 of Algorithm 2 as well as the evaluation time for line 10, while line 10 was calculated both using

Table 5.2: NRMSE (mean over 1000 MC runs) of the proposed DeePLS and DeePKS algorithm compared to Savitzky-Golay smoothing and PKS. Value of 1 means error free estimation.

NRMSE	SaG	PKS	DeePLS	DeePKS
$\boldsymbol{e}_x$	0.911	0.939	**0.953**	**0.983**
$\boldsymbol{e}_{\dot{x}}$	0.896	0.928	**0.943**	**0.955**
$\boldsymbol{e}_{\ddot{x}}$	0.847	0.894	0.885	**0.911**

the reduced order backwards substitution and using the full matrices. The average times for lines 7-10 were 11.18μs (full) and 13.88μs (reduced). It remains future work to investigate, how the overall gain of the reduced matrix calculation would be, if present, when generating C-code from the function.

As expected, the average execution time for line 10 of Algorithm 2 could be reduced, namely from 6.54μs, using full matrices, to 6.09μs with backward-substitution using reduced matrices $\tilde{Q}$ and $\tilde{R}$. An in depth investigation on the computational efficiency of all the proposed algorithms remains to future work.

5.6 Conclusion

5.6.1 Summary

Two novel smoothing algorithms were presented based on polynomial least squares, which are capable of exploiting the knowledge that measurement channels in time series data involve derivatives. We therefore named them Derivative-exploiting Polynomial Least Squares (DeePLS) and Derivative-exploiting Polynomial Kalman Smoother (DeePKS).

The DeePLS algorithm can be interpreted in various ways. First, it is a generalization of SaG-smoothing [SG64] to multi-channel time series data. Second, it can be seen as a physics-informed method to obtain optimal kernels for the time-series convolution, by which the smoothing can be calculated.

The DeePKS can be seen as a recursive version of Derivative-exploiting Polynomial Least Squares (DeePLS). Hence, it can be interpreted as a generalization of the PKS algorithm [Rho16], again to multi-channel time series data in the same fashion as before.

For DeePLS, the interpretation as a physics-informed method establishes connections between classical smoothing theory and modern ML algorithms. Thanks to the generic formulation of the algorithms, they can be applied to sparse and non-uniformly sampled measurement data. For DeePLS, this is valid as long as enough measurement points remain present within the moving window for the problem to remain well-defined. Hence, the algorithms can not only be used for smoothing and causal filtering but also as a re-sampling method. Additionally, differentiation of signals can be performed to obtain additional derivatives as latent states. In a simulation study, the performance was compared to the original SaG algorithm and its recursive counterpart PKS, and we demonstrated that both algorithms outperform their single-channel counterparts. This is achieved by leveraging the interrelation between measurement channels.

5.6.2 Discussion and future work

We want to first present a discussion on the proposed DeePLS algorithm. DeePLS is, at its core, based on weighted least squares regression and hence underlies the independent Gaussian noise assumption. This means, as a limitation, we do not expect the algorithm to be inherently robust to heavy-tailed measurement noise distributions and outliers. This is a known limitation for any least squares formulation, and it seems straightforward to extend the algorithm to robust estimation schemes, for example, similar to the one in [MS14], or the ones presented in Section A.9. Nevertheless, how this affects the mathematical properties of the solution and, as a result, its computational efficiency remains to be investigated. Still, we could see that thanks to performing the regression on an enhanced dataset by including the measurement channels with the derivative data, the algorithm is much more robust to outliers than when treating the channels individually, as in SaG and PKS.

We are looking forward to seeing the application of the proposed algorithm to real-world examples, for example, in the field of vehicle control and robotics. Applying the proposed algorithm as a way to obtain physics-informed kernels in convolutional neural network structures for time-series data involving derivatives seems a very promising path. Here, the properties of the DeePLS algorithm to allow irregularly sampled, sparse, and delayed measurement data can be especially beneficial. Another nice property of the algorithms is the possibility of providing additional, highly interpretable latent information, which are the higher-order derivatives of the dynamical system under observation.

Also, the algorithm seems to reduce variance without introducing much bias. This might explain why its application was improving the overall parameter estimation results in Chapter 6 more than the other smoothing algorithms used for comparison. An in-depth study of this behavior had to be left for future work.

Looking at the proposed DeePKS algorithm, we could observe that for the given scenario, DeePKS clearly outperforms PKS. This is due to the fact that the former could leverage the additional information given as the interrelation between measurement channels. Also, in the given example, both PKS and DeePKS outperformed their non-recursive counterparts. This likely will not be the case for any arbitrary trajectories, while a detailed analysis will have to remain for future work. To understand this more in detail, let us consider the following.

Both PKS and DeePKS are recursive algorithms that are based on the Kalman filter framework. The states to be estimated are the polynomial coefficients, and a propagation model is used to formulate the state model of the Kalman filter. One has to consider that the underlying state model expresses how, in an *absolute* time frame, *time invariant* polynomial coefficients have to be modified to consider a new, one-step forward center location. This means that, in reality, we aim to learn a constant set of absolute polynomial coefficients, building up memory in the form of prior knowledge. For trajectories which, again in an absolute time frame, are, in fact, described by a single polynomial and its time-derivatives, both algorithms will be able to asymptotically converge to the true value (assuming the Gaussian noise assumption also holds).

In the example we used to evaluate the algorithm, we constructed trajectories as a sequence of quintic polynomials. This means, in other words, that the polynomial coefficients are piecewise constant in an absolute time frame. This enables both recursive algorithms to outperform their moving horizon counterparts since the algorithms are actually able to learn the true parameters within each sequence. Since such sequences have been previously proposed to be realized for automated vehicles, we still considered such a scenario to evaluate the algorithms.

Future work remains to critically evaluate how the algorithms perform for manually driven vehicles, which, of course, do not follow such trajectories. Also, an interesting question is how far the performance can be increased by further exploiting knowledge about the driven trajectories. For example, if one knows that the vehicle was just commanded to follow a new sequence at a certain time, of which the coefficients are known, one could exploit that knowledge. At least, the covariance matrix could be reset each time to a high level of uncertainty when a new sequence was commanded. Alternatively, the prior could also be set to the known coefficient values. Exploiting such prior knowledge could increase the performance substantially. Nevertheless, if used for control purposes, as suggested in this work, it might also have a dangerous outcome. The smoother could potentially output values that are only closer to the commanded trajectory while the true trajectories diverge, and this could deteriorate the overall control performance. Unfortunately, we will have to leave it to future work to investigate such effects further.

5.6.3 Contribution

The contributions presented in this chapter can be listed as follows:

- We proposed two novel algorithms for smoothing and filtering of time series data, which we termed Derivative-exploiting Polynomial Least Squares (DeePLS) and Derivative-exploiting Polynomial Kalman Smoother (DeePKS).
- These algorithms can be seen as a generalization of SaG-smoothing and PKS-smoothing to multi-channel data and are based on local polynomial approximation of the data channels
- The main proposal is to exploit the knowledge of measurement channels being higher-order derivatives of a base channel.
- This enables an increase in smoothing performance and robustness compared to existing algorithms, which operate on single data channels only. Simulation studies were carried out to support this claim.
- The generic formulation of the algorithms further allows to incorporate irregularly sampled data as well as delayed and missing measurements
- For DeePLS, an interpretation of the algorithm was provided as a model-free but physically informed method to obtain optimal kernels for a convolution layer of a neural network structure. We believe this brings valuable insights for physics-informed machine learning applied to multi-channel time-series data.

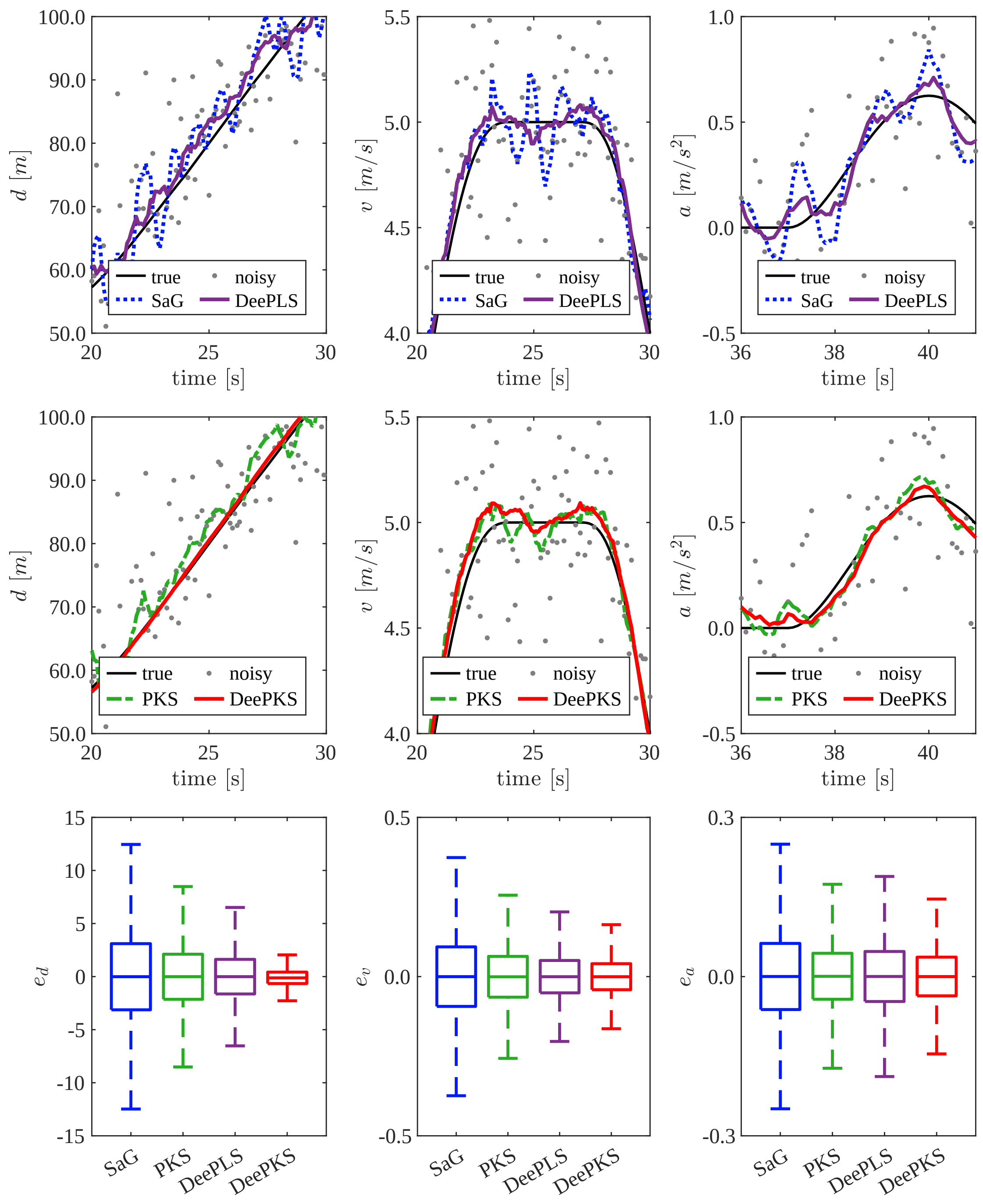

Figure 5.3: Zoom of the comparison of the true, noisy and estimated values for the proposed DeePLS and DeePKS algorithms with results obtained by the Savitzky-Golay (SG) and the PKS method. The areas shown in the top rows are depicted in Figure 5.2 as rectangles. Bottom: Box-plot statistics over 1000 Monte-Carlo simulations of estimation errors.

6 Combined Vehicle State and Parameter Estimation

"There is no such thing as absolute value in this world. You can only estimate what a thing is worth to you."

– Charles Dudley Warner

The content presented in this chapter builds on preliminary work that was previously published by the author of this thesis in Buechel and Knoll [BK16a], but contains substantial novel material, which is published for the first time within this thesis. The main contribution will be an estimation scheme for combined state and parameter estimation, which leverages the problem structure, is robust against measurement noise, is stable during phases of insufficient excitation, provides constrained parameter estimates, and shows improved performance compared to various state-of-the-art solutions. For further details and an overview of these contributions, please refer to Section 6.7.2.

6.1 Introduction

In this chapter, we propose a recursive method for the concurrent estimation of the true vehicle state together with the parameters of longitudinal vehicle motion. More in detail, with the state, we denote the vehicle's acceleration and speed and the tractive forces determined by the vehicle mass, the aerodynamic drag, and the rolling resistance parameters. The knowledge of the true values of these parameters is beneficial for several applications and especially useful for precise, offset-free, model-based motion tracking control, as we will present in Chapter 7. There, we also show how the performance of such a controller degrades under parametric uncertainty. Unfortunately, the vehicle state is (1) only observable through noisy measurements, and simple low-pass filtering will result in delays, which can deteriorate a controller's performance dramatically, which uses signals obtained in such a manner directly as inputs. Even more important, (2) the vehicle parameters might change over time.

While we want to give a formal problem definition in Section 6.3.1, at this point, we want to recall the (simplified) vehicle equation from (3.18):

$$m \cdot \dot{v} = F_{\text{tire}} - m \cdot g \cdot (\sin\varphi + C_{\text{r}} \cdot \cos\varphi) - C_{\text{d}} \cdot v^2. \qquad \text{(3.18 revisited)}$$

For a realistic approach, we have to consider the vehicle mass m, the coefficients of aerodynamic drag C_{d}, as well as rolling resistance C_{r} as time-varying values. For example, as stated by Kidambi, Harne, et al. [KH+14], the vehicle mass of a passenger car might vary by up to 50 % from ride to ride due to a changing number of passengers and cargo. For light trucks and cargo vans, the vehicle mass might even change by up to 400%. We see from (3.18) that if we want to calculate a desired tire force from a desired acceleration value, we will introduce an error that is directly proportional to the error between true and estimated vehicle mass. Hence, the quality of any feed-forward controller using such a model is strongly depending on a correct estimation.

Looking at the rolling resistance C_{r}, we might see changes over time due to different tire characteristics (e.g., changing winter to summer tires, different tire inflation pressures) or driving on varying surfaces like dirt roads or on snow. The National Highway Traffic Safety Administration (NHTSA) reports up to 60 % change in rolling resistance values of different tires and tire pressures [NHT09].

The aerodynamic drag, represented by the lumped coefficient C_{d}, might change due to modification in the aerodynamic area of the vehicle, for example, when driving with a mounted roof box. The installation of a roof box might increase the total drag by up to 30 % according to [AC+10]. Another reason for changes in the aerodynamic drag is when a vehicle joins a platoon, where, according to [Ala11], a reduction of up to 65 % is possible (see also Section 2.1.2).

This chapter is structured as follows: First, in the remainder of this introduction section, we want to provide some motivational material for the state and parameter estimation of the longitudinal vehicle dynamics. This includes the objectives and assumptions that lead to requirements for such an algorithm. Also, we will highlight the importance of vehicle mass estimation, in particular for a variety of tasks, besides enabling optimal control as demonstrated in Chapter 7. We then give an overview of existing solutions for the estimation of vehicle tractive forces parameters in Section 6.2. A formal description of the problem considered in this chapter is given in Section 6.3, together with an observability and identifiability analysis in Section 6.3.2 and discuss various solution approaches in Sections 6.3.3 and 6.3.4. Some preliminary investigations in Section 6.4 will serve to understand existing approaches with their shortcomings in detail, while we will already propose some countermeasures in experiments specifically designed for this purpose. In Section 6.5, we will present our proposed solution for the problem of combined state and parameter estimation, which will then be evaluated in a simulation study in Section 6.6. A discussion of the results is included. We then summarize and conclude this chapter in Section 6.7.

6.1.1 Motivation

Among the time-varying vehicle parameters m, C_{d}, and C_{r}, most impact is produced by the vehicle mass when used in a model-based, longitudinal vehicle motion control scheme. While the main motivation for vehicle mass estimation in this thesis is by nature the potential to substantially improve the tracking performance of a longitudinal controller, as discussed in detail in Chapter 7, a variety of other applications benefit from the knowledge of the true vehicle mass: For lateral motion control, the vehicle mass has an impact on the dynamic stability of the vehicle. Vehicle mass is directly related to tire normal forces, which have an influence on lateral and longitudinal tire force generation [Raj11]. As a consequence, active safety systems like emergency brake and collision avoidance assist, anti-lock braking systems, and stability control all benefit from precise knowledge of the current vehicle mass [WS+08].

It could be demonstrated by Carlson, Lohse-Busch, Diez, and Gibbs [CL+13] that for passenger cars, a 10 % increase in vehicle mass results in up to 4.1 % increase in power consumption. Neglecting the variation in vehicle mass will clearly lead to wrong predictions about the range of HEVs and Electric Vehicles (EVs) and also to sub-optimal battery man-

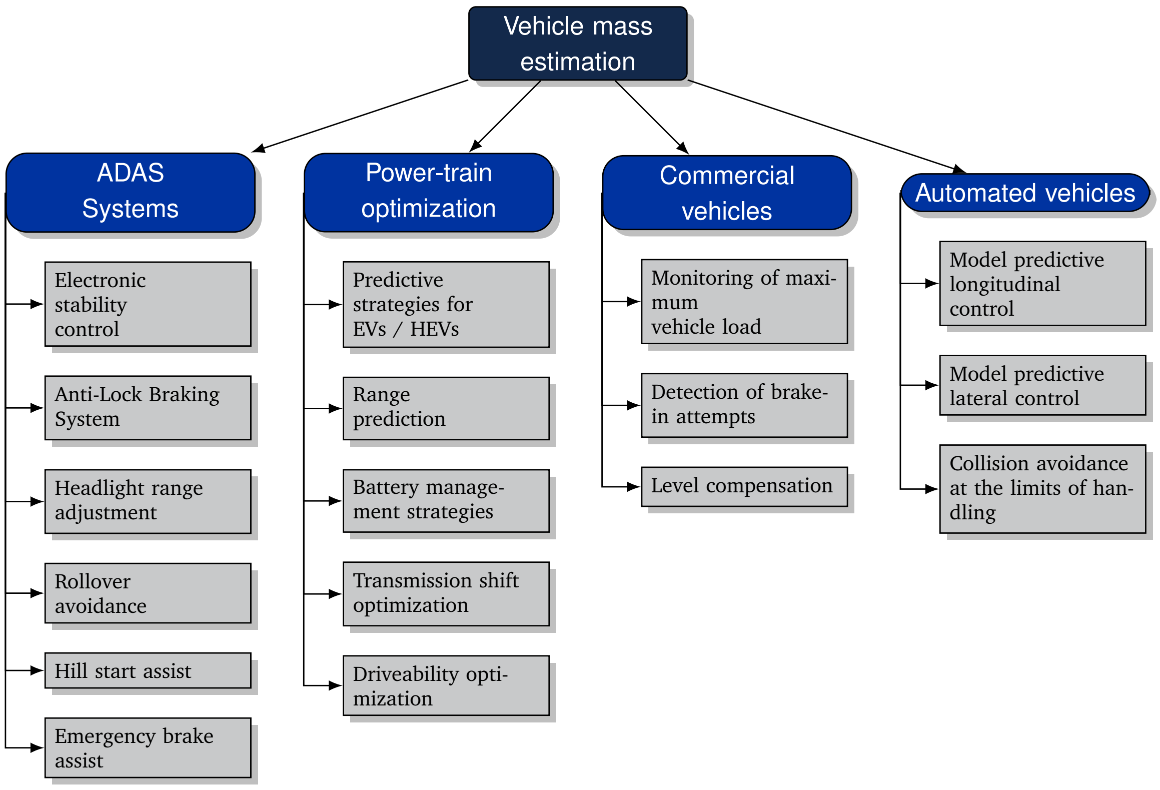
Vehicle mass estimation
ADAS Systems
Electronic stability control
Anti-Lock Braking System
Headlight range adjustment
Rollover avoidance
Hill start assist
Emergency brake assist
Power-train optimization
Predictive strategies for EVs / HEVs
Range prediction
Battery management strategies
Transmission shift optimization
Driveability optimization
Commercial vehicles
Monitoring of maximum vehicle load
Detection of brake-in attempts
Level compensation
Automated vehicles
Model predictive longitudinal control
Model predictive lateral control
Collision avoidance at the limits of handling

Figure 6.1: Example applications of an online vehicle mass estimator. Illustration inspired by Fechtner and Schmulling [FS17].

agement strategies. Energy consumption can be reduced by optimizing transmission shift strategies, which also benefit from a precise knowledge of the vehicle mass.

For commercial vehicles, other use cases for vehicle mass estimation have been reported. These include monitoring the maximum allowed vehicle load or even building a system to detect burglary incidents, which leads to a reduction of vehicle mass during parking.

For automated vehicles, vehicle mass estimation helps to optimize both longitudinal and lateral control. In combination with estimations about the center of gravity and the vehicle inertia, it can be used to improve planning and realize maneuvers at the limits of handling, for example, for racing applications [KG12], but also for collision avoidance exploiting lateral vehicle dynamics.

Inspired by the work of Fechtner and Schmulling [FS17], we give an overview of example applications, which all will benefit from a precise vehicle mass estimation in the illustration shown in Figure 6.1. Nevertheless, despite its importance, Kidambi, Harne, et al. from the University of Michigan, in cooperation with Ford Motor Company, stated in 2014: "While several [vehicle mass] estimation methods have been proposed in the literature, none have seen widespread adoption in current vehicle technologies despite their potential to significantly improve automotive controls" [KH+14].

6.1.2 Requirements

A vehicle state and parameter estimator should meet the following requirements (compare [FK+08]):

- Real-time capability: The algorithm needs to be simple enough to run in real-time without consuming too much computational load

- Accuracy: estimation of vehicle mass within a 3-5 % error
- High convergence rates, ideally to detect changes within a few seconds after a vehicle moves from standstill
- Low-cost solution: operation without additional sensors
- Provide estimation accuracy in terms of variance, which is beneficial for other components, e.g., a robust controller.
- Reliability for successful operation
- Robustness against disturbances and measurement outliers
- Robust estimation even in phases of insufficient excitation
- The solution should be simple to calibrate

As we will show in Section 6.2, not all of these requirements could be fulfilled with existing solutions. We focus on real-time capability, accuracy, and high convergence rates while providing a solution using sensor information as we expect it to be available in future automated vehicles. Another focus will lie on the robustness of the approach against outliers and phases of insufficient excitation.

6.1.3 Objective and assumptions

We want to provide combined estimates of the true vehicle state in terms of vehicle speed and acceleration together with the true vehicle parameters m, C_d and C_r (see 3.18) from noisy measurements. The estimator needs to provide estimated values at all times during vehicle operation in an online fashion. For input into the estimator, we consider the following information available from existing sensors, which we believe is a realistic assumption for AVs:

- Vehicle speed v (from odometer)
- Acceleration a (from Inertial Measurement Unit (IMU))
- Road slope φ (from IMU, possibly fused together with information from High Definition (HD)-map)
- Wheel torque estimations from engine T_we and brakes T_br (from PCM).

We assume that the available sensor readings will be distorted by non-gaussian measurement noise. The availability of a wheel torque estimation value holds for the assumption that our vehicle is equipped with a torque-based power-train control structure. A simplifying assumption is the absence of tire slip and wind. Hence, the remaining parameters in (3.18) which are left to be determined are:

- Vehicle mass m
- Aerodynamic drag C_d
- Rolling resistance C_r.

6.2 Related work vehicle state and parameter estimation

Various authors provided overviews of existing literature related to vehicle mass estimation and discussed different solution approaches. They include Fathy, Kang, and Stein [FK+08], Kidambi, Harne, et al. [KH+14] as well as Fechtner and Schmulling [FS17]. Inspired by these references, existing methods for vehicle mass estimation can be classified into the following categories, exploiting various effects that are influenced by the vehicle mass:

1. Estimation methods exploiting *longitudinal vehicle dynamics*.

2. Estimation methods exploiting *lateral / yaw dynamics*.

3. Other methods, including

    - *Power-train oscillations*, which are influenced by variations in vehicle mass.
    - Sensor based estimation exploiting *suspension dynamics*.
    - Sensor based estimation exploiting *tire pressure*.

We want to follow these categories to analyze the related work in more detail:

Methods exploiting longitudinal vehicle dynamics

The method exploiting longitudinal vehicle dynamics is the most commonly found in literature and will also be proposed for this thesis. Many of the approaches found in the literature are based on different assumptions about what is to be estimated and which signals are available to an estimation algorithm.

For example, Bae, Ryu, and Gerdes [BR+01a] presented an approach taking into consideration that road grad can be measured based on sensor readings of a dual antenna GNNS system. It does not provide any estimations for rolling resistance and aerodynamic drag parameters, nor does it provide an uncertainty value for the vehicle mass estimation. Vahidi, Stefanopoulou, and Peng [VS+03; VD+03; VS+05] proposed to apply recursive least squares with multiple forgetting to estimate road grade and vehicle mass in parallel. Again, no estimation of rolling resistance and aerodynamic drag is provided.

Various approaches were presented for the slightly different task of estimating road grade and vehicle mass simultaneously, but assuming both rolling resistance and aerodynamic drag as known quantities: First, Winstead and Kolmanovsky [WK05] presented an approach combining an Extended Kalman Filter with a model predictive controller. This is motivated by the fact that the performance of an Extended Kalman Filter (EKF) will degrade during phases of insufficient excitation. By taking estimation error covariance into account in the objective function of the MPC, the controller will not just track the desired velocity but also perform additional excitation around the desired trajectory, depending on the error covariance of the EKF. This has the advantage that it can increase the convergence rate of the estimation and also avoid insufficient excitation. A clear drawback of this approach is that the additional excitation has a negative impact on the tracking quality of the controller.

Another estimator of this category was presented by McIntyre, Ghotikar, et al. [MG+09]. They proposed to apply a two-stage estimation strategy to first determine a vehicle's mass using an adaptive least squares strategy and then freeze the vehicle mass and run a nonlinear, Lyapunov-based estimator to estimate road grade in the second stage.

Raffone [Raf13] presented a road slope and vehicle mass estimation for light commercial vehicles. They used a combination of linear Kalman filtering for road slope estimation and RLS with a forgetting factor for observing the vehicle mass.

An evaluation of different methods for vehicle mass and road grade estimation in parallel, which exploit longitudinal dynamics, is given in Kidambi, Harne, et al. [KH+14]. They propose to apply an accelerometer additionally to obtain better state estimations. Again, no estimation of rolling resistance and aerodynamic drag is provided.

Fathy, Kang, and Stein [FK+08] found that inertial forces dominate longitudinal vehicle forces and that drag, road grade induced, and rolling resistance forces affect vehicle dynamics only at low frequencies. This finding is leveraged to propose a "fuzzy supervisor" in order to determine which forces dominate vehicle dynamics. The supervisor is then used to switch between separate estimations of vehicle mass and road grade by a RLS algorithm. Also, here, no estimation of rolling resistance and aerodynamic drag is provided.

The author of this thesis proposed a solution in [BK16a], which can be seen as preliminary work to what will be presented in this chapter. The proposal was to make use of the full measurement data available in AVs, namely acceleration, speed, road grade, and the torque estimation at the tire. Further, a joint estimation scheme based on an extended Kalman Filter was proposed to provide an estimate of the vehicle mass together with state estimates of velocity and acceleration. Compared to the solution in this chapter, in [BK16a], no estimates of rolling resistance or drag coefficient were provided. The approach was non-standard from an algorithmic point of view since it allowed the incorporation of (one-step delayed) measurements of the first-order state derivative, in this case, vehicle acceleration, into the filter.

Except for the approach from Winstead and Kolmanovsky [WK05], none of the above-mentioned works properly discussed or addressed the topic of insufficient excitation and parameter wind-up or provided any observability studies. In 2016, Rhode published his thesis [Rho16] presenting robust and regularized algorithms for vehicle tractive force prediction and mass estimation. It compares various estimation algorithms in both numerical experiments and the application in a real vehicle. The algorithms are based on the Linear-in-Parameters formulation of the longitudinal vehicle equation; hence, no filtered vehicle state estimations were provided, and solely parameter estimation was performed. The algorithms considered include recursive least-squares and robust variants based on M-estimator schemes and Kalman filter variations, which have robust properties, like the M-Kalman filter and the regularized M-Kalman filter. Further, the effect of Errors-in-Variables in linear least-squares problems, together with various countermeasures, is discussed. Among them is the proposal for a recursive generalized total least squares algorithm with noise covariance estimation from [RH+16], as well as the method of instrumental variables is discussed. The problem of insufficient excitation is also mentioned. To overcome estimator wind-up, a method is proposed that combines the Stenlund-Gustafsson [SG02] anti-windup scheme with several other estimator variants.

The methods proposed by Rhode do not include any combined state and parameter estimation schemes. Nevertheless, he mentions performing pre-processing on vehicle speed and acceleration signals with several stand-alone filters, amongst them Butterworth filters with various cut-off frequencies as well as Savitzky-Golay (SaG) smoothing. Eventually, he reports using a novel "Polynomial Kalman Smoother (PKS)" algorithm, which can be regarded as a recursive version of SaG smoothing, see [RH+16]. PKS was applied in the context of generalized total least squares estimation and used to provide mean values for estimating the variance of the unknown measurement noise variances.

In distinction to [RH+16], Altmannshofer, Endisch, et al. [AE+16] achieved robustness against measurement outliers by assuming that the measurement noise is distributed according to a Student-t distribution, and they used real vehicle data to fit the parameters of the distribution. Based on this knowledge, they investigated a similar filter as the robust M-Kalman filters with Stenlund-Gustafsson anti-windup discussed in [RH+16]. Since no closed-form solution to the inverse problem is available under the assumption of Student-t distributed measurement noise, the problem was reformulated as a maximum-likelihood estimation problem and solved via Iteratively Reweighed Least Squares (IRLS). To reduce the computational load for potential deployment on an embedded system, they terminated the calculations after the first iteration. In addition, parameter constraints were enforced by using a complementary linear program. Contrary to [RH+16], they do not apply the method of Instrumental Variables in order to achieve convergence when Errors-in-Variables are present, while their offline validation result on real-world vehicle data showed a certain bias in the estimations. As a simplification, the longitudinal vehicle model was derived under the small angle assumption regarding road slopes. Further information is provided in his thesis [Bey19], most likely after he changed his name.

Xin, Xu, et al. [XX+22] applied RLS with multiple forgetting for online estimation of vehicles mass, also to improve a model predictive vehicle controller for optimal energy management of a fuel cell hybrid electric vehicle. While they found that such a scheme can substantially improve the controller's performance, they noticed their estimation algorithm to perform poorly in some scenarios, especially when trying to track a (quasi) continuously varying vehicle mass. Nevertheless, they left further investigation about the reasons and how to improve performance in future work. They did not mention the difficulty of vehicle mass estimation during phases of insufficient excitation.

Looking at very recent work on parameter estimation exploiting longitudinal motion, Korayem, Khajepour, and Fidan [KK+21b] gave a review on vehicle-trailer state and parameter estimation literature. While the study included many different parameter estimators, including, for example, ones for estimating roll angles using neural network-based approaches, the algorithms mentioned for trailer mass estimation were recursive least squares, extended Kalman filters, and dual Kalman filters. Some approaches processed Lidar point clouds to estimate hitch angles between vehicle and trailer using deep neural networks. While this is not the focus of this thesis, one could imagine that current and future road slope estimations could be performed or augmented by a similar approach.

Yu, Hou, Leng, and Huang [YH+22] suggested improving a conventional least squares approach with forgetting for vehicle mass estimation by combining it with a purely data-driven approach. The data-driven approach uses a neural network that is trained to output the correct vehicle mass depending on various inputs from simulated data. Since in the model-based approach, no countermeasures were taken against insufficient excitation, blending the neural network result using a fuzzy system approach helps to improve the result. No estimates for other parameters can be provided with this approach. The simulation results did not show good convergence of the approach.

Yuan and Song [YS22] recently proposed a modified EKF for vehicle state estimation with partial missing measurements caused by packet loss in sensor networks. We will also partly address the topic of fault tolerance against lost packets in a different way when introducing our proposal of the DeePLS and DeePKS smoothers in Chapter 5, while we will have to leave a comparison and deeper discussion of this topic to future work.

The proposed algorithm developed in Section 6.5 was mostly inspired by the work of Rhode, Hong, Hedrick, and Gauterin [RH+16] and Altmannshofer, Endisch, et al. [AE+16], but with several substantial differences, as will be discussed in detail in the remainder of this chapter.

Methods exploiting lateral / yaw vehicle dynamics

Some authors proposed to exploit the lateral and yaw vehicle dynamics. These include Best and Gordon [BG00], who proposed an Extended Kalman Filter on the lateral vehicle dynamics to estimate a variety of vehicle parameters.

Wenzel, Burnham, Blundell, and Williams [WB+04] and [WB+06] suggested the use of a dual Extended Kalman Filter setup to estimate vehicle states and parameters in parallel but separately on the lateral and yaw dynamics of the vehicle. They validated their approach in a simulation environment.

Huang and Wang [HW12] applies an adaptive compensator together with a Lyapunov-based control law based on the approach from [AG10] to a simplified bicycle model for tracking both vehicle speed and yaw rate. It is combined with an estimation of vehicle mass and yaw moment of inertia, which interacts with the control law to provide the excitation of the vehicle, which is necessary for fast convergence. The approach is applicable only to nonlinear systems that are affine in the parameters, and no estimation of rolling resistance and aerodynamic drag is provided.

Other methods

Fremd [Fre87] patented a method measuring first order *power-train oscillations* which are dominated by the vehicle mass. This involves the installation of a special measurement device. Kim and Ro [KR00] developed a multi-body dynamic model of a quarter-car suspension system and used a reduced-order model to identify its parameters. Rajamani and Hedrick [RH95] presented a methods *exploiting the suspension dynamics* to estimate the sprung mass. Both methods are based on the availability of additional suspension sensors. Chaklader [Cha00] patented a solution based on measuring the *tire pressure and temperature* of at least one tire in order to estimate the vehicle mass. It relies on sensor measurements of additional sensors.

Summary and discussion

To summarize, we found several methods for online vehicle mass estimation, of which the ones exploiting longitudinal vehicle dynamics dominate. Some approaches are based on the availability of sensors, which we regard as additional to what we expect to be available in the future AVs (including low-cost vehicles). This includes a tire pressure and temperature sensor or a measurement device to capture high-frequency power-train oscillations.

Most of the authors proposed applying a combined estimation of vehicle mass parallel to road grade. We believe that road grade will be available in AVs, even with a certain look-ahead from an HD map, and therefore do not consider such a solution necessary for the application in the future AVs.

On the methodical side, we found two algorithms predominantly used: EKF and RLS based methods, both based on a Linear-in-Parameters formulation. Kalman filter-based methods bring the advantage of additionally providing variance estimates.

We found two approaches [WK05; HW12] which linked the parameter estimation solutions to their control algorithms. Their objective was to enforce additional system excitation, which helps to increase convergence rates but comes at the cost of losing tracking performance. We find that the main objective of a controller should be to provide optimized tracking performance and that decisions about future trajectories of a vehicle should be solely taken within the trajectory planning module (on that level of granularity, on a higher level, a decision-making module of course also affects the vehicle's trajectories).

To the best of the author's knowledge, the only ones who suggested estimating the vehicle mass together with the tractive force parameters (coefficients of rolling resistance and aerodynamic drag) were Rhode [Rho16] as well as Altmannshofer (who seems to have changed his name to Beyer later on) [AE+16; Bey19]. Together with [WK05; HW12], they also addressed the topic of insufficient excitation and parameter estimator wind-up while suggesting different solutions.

Table 6.1: Overview on methods which have been proposed for vehicle mass and parameter estimation

	Method exploits					**Algorithm**			**Provides**				
Publication	**Long. dynamics**	**Lat. dynamics**	**Power-train oscillations**	**Suspension dynamics**	**Tire pressure**	**Recursive least squares**	**Ext. Kalman Filter**	**Other**	C_d, C_r **estimation**	**Vehicle state estimation**	**Uncertainty**	**No add. sensor required**	**Real world validation**
[BR+01a]	x	-	-	-	-	x	-	-	-	-	-	-	x
[VS+05]	x	-	-	-	-	x	-	-	-	-	-	x	x
[WK05]	x	-	-	-	-	-	x	-	-	-	x	x	-
[MG+09]	x	-	-	-	-	-	x	-	-	-	-	x	x
[Raf13]	x	-	-	-	-	x	-	-	-	-	-	x	-
[HW12]	x	x	-	-	-	-	-	x	-	-	-	x	-
[FK+08]	x	-	-	-	-	x	-	-	-	-	x	x	x
[BG00]	-	x	-	-	-	-	x	-	-	-	x	x	-
[WB+06]	-	x	-	-	-	-	x	-	-	-	x	x	-
[Fre87]	-	-	x	-	-	-	-	x	-	-	-	-	x
[KR00]	-	-	-	x	-	-	-	x	-	-		-	-
[RH95]	-	-	-	x	-	-	-	x	-	-	-	-	x
[Cha00]	-	-	-	-	x	-	-	-	-	-	-	-	-
[BK16a]	x	-	-	-	-	-	x	-	-	x	x	x	-
[AE+16]	x	-	-	-	-	x	-	-	x	-	x	x	x
[RH+16]	x	-	-	-	-	-	-	x	x	-	x	x	x
ours	x	-	-	-	-	-	-	x	**x**	**x**	**x**	**x**	-

An overview of the methods that were applied to vehicle mass estimation is given in Table 6.1. As shown in Table 6.1, to the best of the author's knowledge, the concurrent estimation of states *and* parameters (in the combination of vehicle mass, rolling resistance, coefficient, and aerodynamic drag coefficient), while also addressing the problem of parameter wind-up during insufficient excitation has not been proposed yet. This will be the case for the proposal we will discuss in the remainder of this chapter.

6.3 Problem formulation, analysis and solution approaches

In this section, we want to provide some preliminary results in order to be able to present the proposed approach in Section 6.5. For a general, application-agnostic introduction to state and parameter learning, please refer to Chapter A.

First, we want to provide a formal problem description for the combined state and parameter estimation problem based on the longitudinal vehicle motion equation, which includes the formulation of various mathematical interpretations. We will see that based on these different interpretations, distinct solution schemes can be formulated.

The first one is a reformulation of the vehicle motion equation as an equation, which is Linear-in-Parameters (Section 6.3.1). This interpretation allows, in general, the application of well-known recursive linear regression schemes, but we will see in Section 6.4.2 that the naive approach will not be able to cope with effects occurring under realistic assumptions. This method has already been applied in the context of vehicle state and parameter estimation in related literature, for example, in Rhode [Rho16] and Altmannshofer and Endisch [AE16]. One drawback of the approach is that it only provides a solution to the parameter estimation problem, while it is not directly possible to obtain filtered values of the vehicle states v and a at the current time step.

Another common formulation is the one as state space equation as an Ordinary Differential Equation (ODE) (see Section 6.3.1). Various Joint Estimation and Dual Estimation schemes are available based on this formulation. These allow us to provide estimates of both the states and the parameters concurrently.

Nevertheless, it is not straightforward to include measurements of the differential state a with a formulation based on ODEs. Therefore, we will present an approach that can be interpreted as a dual estimation scheme, which combines the method formulated as LiP with a state observer based on the ODE formulation in Section 6.5.

6.3.1 Problem formulations

We consider the longitudinal vehicle motion, which was already discussed earlier and can be modeled using the equation (3.18)

$$m \cdot \dot{v} = F_{\text{tire}} - m \cdot \mathrm{g} \cdot (\sin\varphi + C_{\mathrm{r}} \cdot \cos\varphi) - C_{\mathrm{d}} \cdot v^2. \tag{3.18 revisited}$$

We are given noisy measurements of the vehicle speed v and the acceleration $a = \dot{v}$. Additionally, although not explicitly appearing in (3.18), we can also assume to have noisy measurements of the traveled distance s, and we know per definition that $v = \dot{s}$.

Additionally, we are provided with the knowledge about wheel torque from engine $T_{\text{we,k}}$ and brakes $T_{\text{br,k}}$, as well as road grade information φ, which we can regard as known input and disturbance value, respectively, although this interpretation is not important for the estimation problem. We, therefore, define a known input vector u^e as

$$u_k^e = \begin{pmatrix} T_{\mathrm{we,k}} \\ T_{\mathrm{br,k}} \\ \varphi_k \end{pmatrix}, \tag{6.1}$$

where the superscript in u_k^e should denote that this is the input for the estimation problem, and hope that in this way, the reader will be easily able to distinguish it from the control input, which was denoted simply u without any superscript in other parts of this thesis.

The aim is to provide estimates $\hat{x}$ of the true states x together with the parameters $\hat{\theta}$, defined as

$$\theta = \begin{pmatrix} m & C_{\mathrm{d}} & C_{\mathrm{r}} \end{pmatrix}^{\mathsf{T}}, \tag{6.2}$$

with vehicle mass m, aerodynamic drag C_{d} and rolling resistance C_{r}, which we all assume to be time-varying.

As we will see, depending on the exact mathematical formulation, the quantities v, a, and s can be regarded as states, algebraic states, used to calculate virtual measurements or omitted from the formulation. The state vector x for which we want to obtain estimates should, as a minimal requirement, at least consist of the vehicle speed v in order to serve as feedback value for the predictive controller developed in Chapter 7 and Chapter 8. Ideally, we also want to additionally provide estimates of the true acceleration values a.

Looking at the evolution of the system over time, we are given noisy measurements $Y_t \doteq [y_0, y_1, \dots, y_k]$, as well as past and current inputs $U_t^e \doteq [u_0^e, u_1^e, \dots, u_k^e]$ at discrete points in time. For simplicity, we will assume these values to be uniformly sampled, with $t = kT_s$, but include a more general case in the discussions in Chapter 5.

Note that in the formulation of the dynamics above, we omitted any actuator dynamics, compared to the formulation in (3.75a), since we consider both wheel torques T_{we} and T_{br} together with the disturbance φ as *known* inputs for the purpose of state and parameter estimation.

Formulation as Linear-in-Parameters (LiP) equation

A simple, yet not straightforward formulation can be obtained by reordering (3.18) such that a Linear-in-Parameters model, as also discussed in Section 3.5.5, will be obtained. Such a model has the general form (see also (3.86):

$$y^{\bar{\theta}} = \bar{\theta}^{\mathsf{T}} \phi + v, \tag{6.3}$$

where in the problem under investigation, the (virtual) measurement, the vector of unknown parameters, and the feature vector are given as

$$y^{\bar{\theta}} = \frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{r_{\mathrm{eff}}} - a m_{\mathrm{I}} \tag{6.4a}$$

$$\bar{\theta} = [m \quad C_{\mathrm{d}} \quad m \cdot C_{\mathrm{r}}]^{\mathsf{T}} \tag{6.4b}$$

$$\phi = \begin{pmatrix} a + g \sin \varphi(\cdot) \\ v^2 \\ g \cos \varphi(\cdot) \end{pmatrix}, \tag{6.4c}$$

respectively. A detailed derivation can be found in Section 3.5.5. Note that this could only be established by using a modified parameter vector $\bar{\theta}$ instead of θ which contains $m \cdot C_{\mathrm{r}}$ instead of C_{r}. Note that the assumption of additive noise, as expressed above, does not hold since the regressor/feature vector ϕ is also subject to imperfect knowledge since it is composed of both v and a, which are accessible only through noisy measurements. More formally, both the virtual measurement $y_k^{\bar{\theta}}\left(u_k^e, y_k\right)$ and the feature vector $\phi\left(u_k^e, y_k\right)$ are functions of inputs and measurements.

Formulation as Ordinary Differential Equation (ODE)

A standard formulation for Ordinary Differential Equations (ODEs) can be of the following general form:

$$\dot{x}(t) = f\left(x(t), u^e(t), \theta\right) + \omega \tag{6.5a}$$
$$y(t) = h\left(x(t), u^e(t), \nu\right), \tag{6.5b}$$

and more specifically, it follows the ODE given in (3.66b),

$$\dot{v} = \frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{(\theta_1 + m_{\mathrm{I}})\, r_{\mathrm{eff}}} - \frac{\theta_1 g}{\theta_1 + m_{\mathrm{I}}}(\sin\varphi + \theta_3 \cos\varphi) - \frac{\theta_2}{\theta_1 + m_{\mathrm{I}}} \cdot v^2 \tag{6.6a}$$
$$y = v + \nu_v, \tag{6.6b}$$

with $\theta = [m, C_{\mathrm{r}}, C_{\mathrm{d}}]^{\mathsf{T}}$ and $u^e = [T_{\mathrm{we}}, T_{\mathrm{br}}]^{\mathsf{T}}$ and the process and measurement noise ω and ν, respectively. Note that in this formulation, the measurement equation is a function of the differential state x only, and measurements of algebraic states $\dot{x}$ cannot be easily incorporated.

6.3.2 Structural observability and identifiability analysis

We provide some additional background and definitions regarding observability and structural identifiability in Section A.10. As we will see from the investigations in the following subsections, the states and parameters of the longitudinal vehicle motion dynamics are locally observable and structurally identifiable. For the Linear-in-Parameters model, in case the vehicle drives at constant speeds, the identifiability property will be lost. In this case, by definition, the acceleration becomes zero.

In the following, we will first perform a structural observability and identifiability analysis as explained in Section A.10. Then, we will investigate the circumstances for the recursion derived from the Linear-in-Parameters form of the vehicle dynamics to be well-defined.

Analysis for the Ordinary Differential Equation model

We want to perform structural observability and identifiability analysis for the longitudinal vehicle system defined by the time-continuous system

$$\dot{x}(t) = \begin{pmatrix} \frac{T_{\mathrm{we}}-T_{\mathrm{br}}}{(m+m_{\mathrm{I}})\, r_{\mathrm{eff}}} + \frac{-C_{\mathrm{d}}}{m+m_{\mathrm{I}}} \cdot {x_1}^2 + \frac{-mg}{m+m_{\mathrm{I}}}(\sin\varphi + C_{\mathrm{r}}\cos\varphi) \\ \frac{-1}{\tau_{\mathrm{pwt}}} x_2 + \frac{1}{\tau_{\mathrm{pwt}}} u_1 \\ \frac{-1}{\tau_{\mathrm{br}}} x_3 + \frac{1}{\tau_{\mathrm{br}}} u_2. \end{pmatrix} \tag{6.7a}$$
$$y(t) = x(t) = \begin{pmatrix} v & T_{\mathrm{we}} & T_{\mathrm{br}} \end{pmatrix}^{\mathsf{T}}, \tag{6.7b}$$

with the known inputs $u_1 = T^{\mathrm{d}}_{\mathrm{we}}$, $u_2 = T^{\mathrm{d}}_{\mathrm{br}}$ and $u_3 = \varphi$.

We used the software tool STRIKE-GOLDD2 [VE+19], which is based on the Symbolic Toolbox of MATLAB®, to calculate the observability-identifiability matrix and perform the observability rank tests. We investigated several assumptions on the shape of the input variables, including constant inputs and ramp-shaped inputs. For the sake of brevity, and since the resulting extended observability-identifiability matrix is large enough to become unreadable when printed on a single page, we omit any intermediate results, but the usage of the toolbox is straightforward given the model equations. Table 6.2 shows various evaluations of the extended observability-identifiability matrix for different assumptions regarding the shape of the input signals. For simplicity, we show only a subset for $T_{\mathrm{br}} = 0$.

Table 6.2: Identifiability of parameters and observability of time-varying, known input depending on system excitation for model (6.7) with $T_{\mathrm{br}} = 0$.

Input trajectory		**Parameter identifiable**		
$\dot{T}_{\mathbf{we}}$	φ	m	$C_{\mathbf{d}}$	$C_{\mathbf{r}}$
$\dot{T}_{\mathrm{we}} = 0$	$\dot{\varphi} = 0$	✓	✓	✓
$\dot{T}_{\mathrm{we}} = 0$	$\dot{\varphi} \neq 0$	✓	✓	✓
$\dot{T}_{\mathrm{we}} \neq 0$	$\dot{\varphi} = 0$	✓	✓	✓
$\dot{T}_{\mathrm{we}} \neq 0$	$\dot{\varphi} \neq 0$	✓	✓	✓

We find that the model is Fully Input-State-Parameter Observable (FISPO): All its states are observable. All its parameters are locally structurally identifiable. Note that local structural identifiability does not automatically render practical identifiability. If we perform the same analysis for the model without input dynamics and neglecting any brake torque, we obtain the result that the parameters are only locally identifiable in case of a time-varying input $\dot{T}_{\mathrm{we}}$. This means that for constant wheel torque, practical unidentifiability might still occur.

Analysis for Linear-in-Parameters model

Now we want to perform a identifiability analysis using the Linear-in-Parameters model from (3.81) and Section 6.3.1, which is given as

$$\frac{T_{\mathrm{we}} - T_{\mathrm{br}}}{r_{\mathrm{eff}}} - a\, m_{\mathrm{I}} = [m \quad C_{\mathrm{d}} \quad m \cdot C_{\mathrm{r}}] \begin{pmatrix} a + g \sin\varphi \\ v^2 \\ g \cos\varphi \end{pmatrix},$$

with the regressor ϕ being

$$\phi = \begin{pmatrix} a + g \sin\varphi \\ v^2 \\ g \cos\varphi \end{pmatrix}.$$

In order for the inverse (regression) problem to be well-defined, we know that for a Full Information Estimation (FIE), we need the information matrix $\Phi\Phi^{\mathsf{T}}$ of the regressor $\Phi_N = [\phi_1, \phi_2, \ldots, \phi_N]$ to be invertible, which means that for a certain N, the matrix obtained by the sum

$$\Phi_N \Phi_N^{\mathsf{T}} = \sum_{i=1}^{N} \begin{pmatrix} (a_i + g\, \sin\varphi_i)^2 & {v_i}^2\, (a_i + g\, \sin\varphi_i) & g\, \cos\varphi_i\, (a_i + g\, \sin\varphi_i) \\ {v_i}^2\, (a_i + g\, \sin\varphi_i) & {v_i}^4 & g\, {v_i}^2 \cos\varphi_i \\ g\, \cos\varphi_i\, (a_i + g\, \sin\varphi_i) & g\, {v_i}^2 \cos\varphi_i & g^2 \cos{\varphi_i}^2 \end{pmatrix} \tag{6.8}$$

will have full rank. This is given in general for $N = 3$ unless we have that $a_i = 0$, which is, of course, the case for constant speed $v_i = v = const$. Then, the matrix of the sum (6.8) degenerates to

$$\Phi_N \Phi_N^{\mathsf{T}} = \sum_{i=1}^{N} \begin{pmatrix} (g\, \sin\varphi_i)^2 & v^2\, (g\, \sin\varphi_i) & g\, \cos\varphi_i\, (g\, \sin\varphi_i) \\ v^2\, (g\, \sin\varphi_i) & v^4 & g\, v^2 \cos\varphi_i \\ g\, \cos\varphi_i\, (g\, \sin\varphi_i) & g\, v^2 \cos\varphi_i & g^2 \cos{\varphi_i}^2 \end{pmatrix} \forall a_i = 0, \tag{6.9}$$

which has a rank < 2 and degenerates even further to rank $= 1$ if we assume the road grade to be constant:

$$\Phi_N \Phi_N^\mathsf{T} = \begin{pmatrix} 3\,g^2 \sin\varphi^2 & 3\,g\,v^2 \sin\varphi & 3\,g^2 \cos\varphi \sin\varphi \\ 3\,g\,v^2 \sin\varphi & 3\,v^4 & 3\,g\,v^2 \cos\varphi \\ 3\,g^2 \cos\varphi \sin\varphi & 3\,g\,v^2 \cos\varphi & 3\,g^2 \cos\varphi^2 \end{pmatrix}, \tag{6.10}$$

and this becomes obvious if we additionally set the road grade value to zero $\varphi = 0$:

$$\Phi_N \Phi_N^\mathsf{T} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3\,v^4 & 3\,g\,v^2 \\ 0 & 3\,g\,v^2 & 3\,g^2 \end{pmatrix} \forall a_i = 0, \varphi = 0. \tag{6.11}$$

For a recursive regression, the regressor needs to be Persistently exciting (PE) (see Definition A.10.4). This can be written as

$$0 < \alpha \mathbb{I} \leq$$
$$\sum_{i=j}^{j+S} \begin{pmatrix} (a_i + g \sin\varphi_i)^2 & {v_i}^2\,(a_i + g \sin\varphi_i) & g \cos\varphi_i\,(a_i + g \sin\varphi_i) \\ {v_i}^2\,(a_i + g \sin\varphi_i) & {v_i}^4 & g{v_i}^2 \cos\varphi_i \\ g \cos\varphi_i\,(a_i + g \sin\varphi_i) & g{v_i}^2 \cos\varphi_i & g^2 {\cos\varphi_i}^2 \end{pmatrix} \geq \alpha \mathbb{I} < \infty. \tag{6.12}$$

With all diagonal entries being positive for all real parameters, this can only be fulfilled if all diagonal entries are unequal to zero, and hence, all the following conditions are fulfilled

$$\sum_{i=1}^{N} a_i + g\,\sin\varphi_i \neq 0 \tag{6.13}$$

$$\sum_{i=1}^{N} v_i \neq 0 \tag{6.14}$$

$$\sum_{i=1}^{N} g^2 \cos{\varphi_i}^2 \neq 0. \tag{6.15}$$

The last condition is fulfilled for all realistic, small road-grade angles, the second only for nonzero vehicle speed over the entire sequence. The first condition above is clearly violated for $a_i = 0$ and $\varphi_i = 0$, but also, assuming constant road grade and acceleration over a short horizon, if $a = -gsin\varphi$.

This means that the Linear-in-Parameters estimation is not PE under the above-mentioned conditions, which includes phases of constant vehicle speeds at zero road grade.

6.3.3 Possible solution approaches

For application in a vehicle, we are interested in a solution that is able to run at reduced memory demand and provides immediate results in an embedded system.

Solutions based on the Linear-in-Parameters model

Recursive solutions exist for the LiP problem, like RLS or linear Kalman filtering. Nevertheless, the assumptions under which these algorithms are guaranteed to converge are quite restrictive in practice, and we want to investigate them in the remainder of this section. As a general advantage of the LiP approach, one can mention that the processing of all available measurement channels is directly supported.

Solutions based on the Ordinary Differential Equation model

The existing solutions based on the state space formulation of the problem can be separated by looking at different categories. Both joint estimation schemes and dual estimation approaches can be applied. Again, for implementation on an embedded device, recursive methods or moving horizon approaches, which operate on mini-batches of a fixed size, are required. The latter has the drawback that additional measurement data storage needs to be realized, which increases the memory footprint of the solution. We focus our investigations first on recursive solutions of the Kalman filter family. These algorithms are known to be a generalization of recursive least squares. Since we are dealing with a nonlinear system, we investigate Extended Kalman Filters (EKF), Sigma-Point Kalman filters, Square-root formulations of those, and particle filters as preferred methods.

We also looked into Moving Horizon Estimation, but these results are only discussed briefly in the future work section of this work since we found the benefit of the method is not worth the increased computational costs that come with the necessity to solve a nonlinear program at every time step. Methods from nonlinear model adaptive control also seemed interesting candidates but were found to be too restricted in terms of the assumptions on the structure of the nonlinearity to be applied to the given problem.

As a result, the discussion in the remainder of this section is mostly focused on Kalman filter-based methods, with the exception of the particle filter.

6.3.4 Challenges and shortcomings of existing approaches

Challenges for Linear-in-Parameters (LiP) estimation

We already learned in Section 6.3.2 that for our longitudinal vehicle model, under regular operation conditions, we cannot guarantee that the system is sufficiently excited for recursive parameter estimation to be well defined.

The assumptions under which recursive least squares methods and linear Kalman filters are guaranteed to converge for Linear-in-Parameters models are the following. First, only normally distributed (Gaussian) *output* noise is assumed. Second, the persistent excitation condition has to be fulfilled. Both assumptions do not hold in practice. As we will show next, Errors-in-Variables and non-normal measurement distributions can lead to bias and even unstable estimation. All of these are present in real-world measurements for the given problem.

General shortcomings of the LiP approach The estimation schemes formulated for the Linear-in-Parameters model only allow the estimation of the parameters in the equation but do not provide any filtering or smoothing of the vehicle states. In order to improve the performance of a control algorithm, one does not want to use the raw, noisy measurements directly as input to the control algorithm. For this reason, an additional mechanism is necessary to provide filtered estimates of the vehicle states.

As we will see in Section 6.4.3, applying filtering, or rather smoothing prior to the parameter estimation methods can also increase the performance of the estimators based on Linear-in-Parameters Kalman filters. Since a smoothed value in a causal setting per definition is delayed, we will propose to use an additional state filter in the method we will present in Section 6.5.3.

Another shortcoming is that for the LiP formulation, it was necessary to introduce the combined parameter θ, which only contains the product mC_r. Therefore, it is not possible to consider constraints on the parameter C_r directly.

Errors-in-Variables Assuming only Gaussian output noise for linear regression problems is equivalent to assuming that the feature vector is known perfectly at all times. Facing additional noise in the independent variables is known as the "Errors-in-Variables" problem [MM00; Söd07], which causes attenuation bias. We will demonstrate this effect in a simulation study in Section 6.4.2. To reduce this bias, various methods have been proposed. Among them, and also previously applied to (recursive) vehicle parameter estimation, is Recursive Generalized Total Least-Squares (RGTLS) [RB+14], as well as the method of Instrumental Variables (IV) (see Rhode [Rho16] and references therein, as well as the short summary of the method in Section A.8.6). Rhodes's comparison of methods dealing with Errors-in-Variables came to the conclusion that IV outperforms RGTLS for vehicle parameter estimation problems (see [Rho16], Section 5, Figure 5.10). This is why we decided to focus on the method of Instrumental Variables with Kalman Filtering in our analysis and did not further investigate Recursive Generalized Total Least Squares. The interested reader can find an overview and additional literature on the topic in [Rho16], Section 3.6.

Insufficient excitation We presented the term *persistent excitation* in Section A.10.4 and mentioned that this is the condition under which recursive linear regression is known to converge. Not surprisingly, *insufficient excitation* is present if the PE condition is not fulfilled. Then, the regression problem becomes ill-defined, and no convergence can be guaranteed.

To understand why insufficient excitation can lead to diverging parameter estimates, let us first provide some intuition about how Kalman filtering works under regular, persistently exciting conditions: In principle, two mechanisms are working against each other. First, the innovation drives the estimates toward the most probable solution. The Kalman Gain acts as a control mechanism to allow faster or slower adaptation and is proportional to the error covariance matrix. If the confidence in the prior knowledge is high, no or only little adaption takes place. How much this confidence is increased depends now on the information content of the incoming data. If the data is rich, which is under high excitation, the confidence of the estimator in its prior estimate becomes higher and would eventually become infinite. This means the information matrix becomes infinite, and the error covariance matrix becomes zero. With zero error covariance, we would have zero Kalman Gain, and the adaptation mechanism would stop forever. This would also mean, in the case of time-varying parameters, that no further learning and, hence, no tracking of the true parameters could take place. But, under time-varying parameters, the covariance matrix needs to be lower-bounded in order to allow tracking of upcoming parameter variations.

Now, the second mechanism, which is there to establish these lower bounds, is by adding parameter noise to the covariance estimate in each time step. So, looking at the evolution of the covariance or information matrix, rich incoming data leads to the effect of increased information, while the parameter noise increases the uncertainty by adding to the error covariance matrix.

Now, how does this mechanism change under insufficient excitation? Then, the uninformative incoming data does *not* lead to a decrease of the uncertainty. If the increase in uncertainty due to the parameter noise term is bigger, the error covariance matrix will grow without bounds as long as the system is insufficiently excited. This effect is called *estimation error covariance wind-up* and means that some of the eigenvalues of the covariance matrix grow to unacceptably large values [SG02]. The result is numerical problems and a high sensitivity to noise.

Since the Kalman filter for the parameter estimation problem can be seen as a generalized variant of linear recursive least squares, this effect is also happening in a similar fashion within the RLS family of algorithms.

Apart from different variable forgetting schemes that have been proposed for RLS, a countermeasure that can be applied to Kalman Filter formulations with scalar observations was presented by Stenlund and Gustafsson [SG02]. Apart from their original proposal to name it *Adaptive Kalman Filtering*, today it is commonly known, after the names of the originators, as *Stenlund-Gustafsson anti-windup scheme* (see Section A.8.5). The term adaptive in Stenlund-Gustafsson filtering is related to the process noise covariance matrix, which is controlled in an adaptive fashion, depending on the information content of the incoming data. Then, the parameter error covariance matrix can be driven to a desired value, which turns out to guarantee not only lower but also upper bounds, even during insufficient excitation. This desired value for the error covariance matrix is the tuning parameter of the Stenlund-Gustafsson (SG) scheme. A detailed analysis of the convergence properties and covariance bounds is given in Evestedt and Medvedev [EM05]. The topic of insufficient excitation within RLS and Kalman Filter (KF) schemes is still an area of active research; see, for example, Shin and Lee [SL20].

Again, the application of Stenlund-Gustafsson adaptation to the vehicle parameter estimation problem has already been previously proposed by [Rho16] and also [AE16].

We will also use the SG adaptation scheme in our studies, and the effect of the algorithm under lack of sufficient excitation will be demonstrated, again, the simulation study in Section 6.4.2. See also Figure 6.2 for the results.

Measurement outliers Standard recursive least-squares and Kalman Filter implementations are based on the assumption that only additive, zero-mean, and normally distributed measurement noise is present. This Gaussian noise assumption is violated in the presence of measurement outliers or if the probability density functions of the noise process follow distinct distributions.

In the context of vehicle parameter estimation as considered in this chapter, we know from investigations performed by Altmannshofer, Endisch, et al. [AE+16] that the measurement noise distribution for the Linear-in-Parameters model shows heavier tails than given by the Gaussian probability distribution. They suggested that the noise distribution of data obtained from vehicle experiments can be approximated as well as Student-t distributions [Stu08]. This distribution is similar to the Gaussian one but shows heavier tails, as presented in Section A.9.5.

As a countermeasure to the presence of outliers, both [RH+16] and [AE+16] suggested to apply robust M-estimation schemes [MP+21]. We presented M-estimation schemes in Section A.9 and discussed existing approaches, of which some resemble standard linear Kalman filters but with additional weight factors that depend on the loss function of the M-estimation scheme.

We will demonstrate in a simulation study in Section 6.4.2 how the presence of heavy-tailed noise can deteriorate the performance of recursive Linear-in-Parameters estimation and how robust M-estimation schemes can help to regain acceptable performance.

Challenges for Joint and Dual Estimation schemes

In general, the application of joint estimation schemes works by augmenting the state space with ordinary differential equations for constant parameters and is straightforward. Standard estimators can be used afterward, but the observability of the augmented system has to be given. For nonlinear systems, the observability is operation point dependent, and phases without observability might be reached. For the given problem, observability was investigated in Section 6.3.2. If standard Kalman filter-based estimators are used, one has to consider that they are designed under the Gaussian noise assumption, and only a certain robustness is given in the presence of outliers. Also, here, it is not straightforward to constrain the solution, and special care has to be taken. Due to the increased dimension of the augmented state space in joint estimation schemes, higher computational costs arise. For dual estimation approaches, one has to consider that the state and the parameter estimator cannot just be applied without further measures, and interdependence has to be considered.

Signal pre-processing for vehicle applications

As will be shown in the subsequent sections, the correct signal pre-processing method can have a beneficial impact, especially on the methods for the LiP approach. This has been suggested earlier for the problem of vehicle parameter estimation. For example, [RH+16] applied Savitzky-Golay smoothers and prosed a recursive version of it, which they called PKS smoother, to pre-process raw measurements of vehicle speed and acceleration.

Theoretically, with Kalman filters, applying signal filters prior to using the data within the Kalman filter is not necessary and might even deteriorate their performance under nominal measurement noise conditions. Nevertheless, as we will also demonstrate in the next sections, the effects of Errors-in-Variables can be reduced by doing so. Typically, each measurement channel is treated independently of the remaining measurements when applying these filtering or smoothing operations. Since we are dealing with distance, speed, and acceleration values, which per definition have an interrelation as one being the first-order derivative of the other, the following question arose: can we exploit this knowledge without imposing any further assumptions about the evolution of these quantities? The result of these investigations led to the proposed DeePLS algorithm, which is presented in Chapter 5, and its recursive counterpart, the DeePKS smoother.

In the following sections, we will investigate how combining the DeePLS algorithm can improve the parameter estimation quality of the problem under investigation.

6.4 Preliminary investigations

In this section, we want to present two simulation studies to analyze various estimators and pre-processing methods based on the LiP formulation. First, we want to explain some details on the simulation environment used for these studies.

6.4.1 Simulation environment 6.1

Simulation environment description

The purpose of this simulation environment is to create trajectories for the signals $x(t)$, $\Phi(t)$, $a(t)$, $v(t)$, and $\theta(t)$ which obey the Linear-in-Parameters equation

$$x(t) = \Phi\left(a(t), v(t), t\right)\theta(t). \tag{6.16}$$

Above, x is the state, θ a parameter vector, while $t = [1, 1e5]$ s. Φ is a regressor vector calculated from the trajectories of $a(t)$ and $v(t)$. All trajectories are then sampled at a sample time of $\Delta t = 1s$. This data will then be used as ground truth to calculate several noisy observations, which will then be used to compare the performance of various estimation algorithms.

This experiment was inspired by the one described in [Rho16], Section 3.5.6, but modified to be more similar to the longitudinal vehicle parameter estimation problem.

Calculation of parameter vector. We define a time-variant parameter vector θ as follows:

$$\theta(t) = \begin{cases} \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, & \text{if } t \leq 50000 \\ \begin{pmatrix} 1 & 2.5 & 3 \end{pmatrix}, & \text{otherwise.} \end{cases} \tag{6.17}$$

Calculation of acceleration and speed trajectories. We further calculated trajectories of acceleration $a(t)$, velocity $v(t)$, and in addition, although not directly needed for the calculation of Φ, the distance $d(t)$ according to

$$a(t) = \sin\left(\frac{2\pi t}{50}\right)\sin\left(\frac{\pi t}{50}\right) \tag{6.18}$$

$$v(t) = \frac{25\left(3\sin\left(\frac{\pi t}{50}\right) - \sin\left(\frac{3\pi t}{50}\right)\right)}{3\pi} + 12 \tag{6.19}$$

$$d(t) = \frac{40000\cos\left(\frac{\pi t}{100}\right)^6}{9\pi^2} - \frac{20000\cos\left(\frac{\pi t}{100}\right)^4}{3\pi^2} + 12t, \tag{6.20}$$

were the expressions for v and d were obtained by integration of the sinusoidal acceleration given in expression (6.19).

Calculation of regressor vector. Next, we obtained a time-varying regressor vector $\Phi(t)$ according to

$$\Phi(t) = \begin{pmatrix} v(t) + \nu_v & (a(t) + \nu_a)^2 & g \end{pmatrix}, \tag{6.21}$$

with the gravitational constant $g = 9.81$ and the trajectories for a and v as defined in (6.18). This formulation resembles the regressor of the Linear-in-Parameters formulation of the longitudinal vehicle equation given in (6.4), with zero road grade. ν_v and ν_a is measurement noise to simulate Errors-in-Variables and its definition will be scenario dependent. To calculate the ground truth, these are set to zero.

Calculation of noisy observations. We then calculate noisy observations for the measured output $y(t)$ The Observations $y(t)$ then calculated according to

$$y(t) = \Phi(t)\theta(t) + \nu_y, \tag{6.22}$$

where ν_y denotes output noise and will be drawn from distributions that are scenario-dependent.

Scenario 1. Only Gaussian output noise For Scenario 1, we use normally distributed output noise ν_y with $\sigma_y^2 = 0.02$ only, while ν_a, ν_v, ν_d are set to zero.

Scenario 2: Errors-in-Variables We also add Gaussian noise to the signals in the regressor vector, where the variances of the normal distribution were chosen to be $\sigma_y^2 = 0.02$ and $\sigma_{a,v,d}^2 = 0.005$.

Scenario 3: Insufficient Excitation To additionally simulate a phase of insufficient excitation, we calculate all signals in analogy to Scenario 2, but this time, we set

$$a(2500 \leq t \leq 7500) = 0 \tag{6.23}$$

$$v(2500 \leq t \leq 7500) = v(2499). \tag{6.24}$$

prior to the calculation.

Scenario 4: Measurement outliers In a last variation, which introduced outliers, instead of using normally distributed noises, we sample noise from a Student-t distribution for all noise values with $\nu_s = 3$ degrees of freedom.

6.4.2 Simulation study: Effect and countermeasures for challenges in LiP estimation

We want to demonstrate how the various challenges of Errors-in-Variables, insufficient excitation, and the presence of measurement outliers deteriorate the naive, recursive parameter estimation algorithms based on Linear-in-Parameters Kalman filtering. Also, how various countermeasures proposed in the literature help to overcome these challenges. For this purpose, we perform the following experiment.

Experiment 6.1

We run various simulations with Simulation Environment 6.1 described in Section 6.4.1. In this simulation environment, noisy trajectory data is generated, which is calculated based on the piece-wise constant parameter vector θ. We run all different scenarios described in simulation environment 6.1, for which different algorithms try to obtain the true parameter estimates. We quickly want to repeat the simulation scenarios which are simulated:

1. **Scenario 1:** Only Gaussian output noise is present
2. **Scenario 2:** Errors-in-Variables (additionally)
3. **Scenario 3:** Insufficient excitation (additionally)
4. **Scenario 4:** Measurement outliers (additionally)

For all four scenarios, we calculate estimates by using the four algorithms shown in Table 6.3.

Table 6.3: Overview over Algorithms and their ingredients used for Experiment 6.1.

Algorithm name	Instrumental Variables (IV)	Anti-windup (SG)	Robust Estimation (M)	Constraints (C)	Algorithm
KF	x	x	x	x	Algorithm 8
KF-IV	✓	x	x	x	Algorithm 9
SG-KF-IV	✓	✓	x	x	Algorithm 11
M-SG-KF-IV	✓	✓	✓	x	Algorithm 15

We obtain a 4×4 matrix of result plots, which can be observed in Figure 6.2.

Result discussion for Experiment 6.1

Results of Experiment 6.1 are shown in Figure 6.2. We see a 4×4 matrix of result plots with different challenges for the estimators in each column and different algorithms in each row. We will discuss each column in the following.

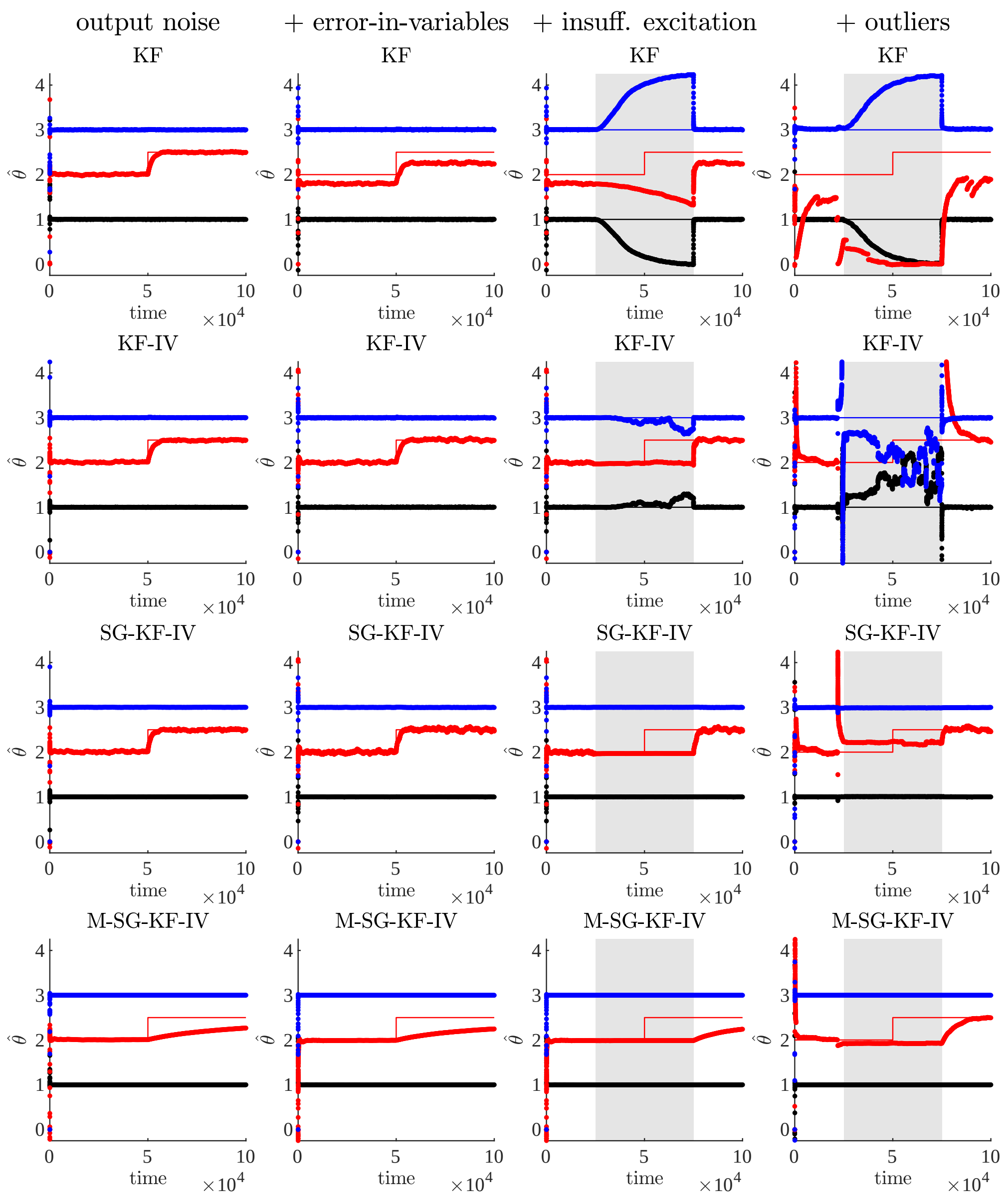

Figure 6.2: Comparison of simulation results of Experiment 6.1. Thin lines show true, bold lines show estimated values for θ_1 (black), θ_2 (red) and θ_3 (blue).

Gaussian output noise and standard Kalman Filter (KF) Figure 6.2, Column 1.
Top-left of Figure 6.2, we can observe how well a standard Kalman filter behaves when only Gaussian output noise is present. We can see that the parameter estimates converge quickly to the true values, and the filter is also capable of adjusting to the parameter step happening at $t = 5000$.

Errors-in-Variables and Instrumental Variables (IV) Figure 6.2, Column 2.
For the plot in Column 2, we added Errors-in-Variables and observed that the standard Kalman Filter is not capable of producing bias-free parameter estimates anymore. One possible countermeasure is shown below in the second subfigure from top and left. The entire second row shows results of an algorithm denoted by $KF-IV$, which uses Instrumental Variables to compensate the bias introduced by the Errors-in-Variables.

Insufficient excitation and anti-windup (SG) Figure 6.2, Column 3.
The results of column 3 correspond to Scenario 3, where a phase of insufficient excitation was present, which is marked by the gray area in the plots. Both the standard Kalman filter (KF) and the version with Instrumental Variables (KF-IV) suffer from the parameter-windup effect during the phase of insufficient excitation. This causes the filter to become unstable and diverge.

The third row in Figure 6.2 shows results of the SG-KF-IV Kalman filter (see Algorithm 11), which additionally uses the parameter noise adaptation scheme according to Stenlund-Gustafsson [SG02]. We can see that this mitigates the effect of insufficient excitation and eliminates divergence caused by the error-covariance wind-up. While during insufficient excitation, parameter learning cannot take place, tracking θ_2 to the new value at $t = 5000$ is impossible since not enough information is available to the algorithm. Nevertheless, as soon as the excitation is again rich enough, at $t = 7500s$, the algorithm quickly adapts to the modified parameter value.

Measurement outliers and robust M-estimation Figure 6.2, Column 4.
In the last column of Figure 6.2, results are shown with measurement noise containing outliers according to Scenario 4. One can see that with all algorithms discussed so far, in rows one to three, the effect of outliers cannot be mitigated. The bottom row of Figure 6.2 shows results of a robust M-Estimation algorithm (M-SG-KF-IV, see Algorithm 15).

Conclusion from Experiment 6.1 From the observations of Figure 6.2, we could conclude that the best choice for an estimator that can deal with all the challenges present in real-world estimation would be the robust M-estimation Stenlund-Gustafsson Kalman Filter with Instrumental Variables, which we denoted with the M-SG-KF-IV algorithm (see Algorithm 15) throughout this thesis. Remember that this algorithm is similar to the one proposed by [Rho16], p. 162, except that we did not use a Huber loss function but used an M-estimation scheme as used by [AE16], which applied a loss function derived from the Student-t distribution. So far, this observation is in line with the experiments performed in [Rho16], where he found this algorithm to also perform best.

Nevertheless, as we will see from the more detailed Monte-Carlo simulation study presented in Section 6.4.3, we will find that for this particular scenario, the method of Instrumental Variables is not robustly providing error covariance matrices which are positive definite, and therefor often shows diverging parameter estimates. Let us, therefore, have a closer look at these additional results before drawing early conclusions.

6.4.3 Simulation study: Effect and importance of signal pre-processing for recursive LiP estimation

The purpose of this study is to compare (1) different estimation strategies and algorithmic ingredients together with (2) various signal pre-processing methods and to evaluate their estimation performance. This is with regard to (a) robustness to Errors-in-Variables, (b) insufficient excitation, and (c) measurement outliers in the form of non-normal measurement noise.

Experiment 6.2

In this experiment, we also use Simulation Environment 6.1 described in Section 6.4.1, while the simulations are performed *only on Scenario 4.* As a reminder, Scenario 4 was configured to run with (a) Errors-in-Variables, (b) insufficient excitation, and (c) measurement outliers in the form of heavy-tailed measurement noise drawn from a Student-t distribution.

The experiment consists of performing a Monte-Carlo simulation using noise data obtained from a random generator with different seeds over a number of 100 simulations. This series of Monte-Carlo simulations runs over a broad variety of combinations of (1) different estimation algorithms and (2) different pre-processing methods, which will be listed next.

Overview of algorithms used for comparison A selection of the algorithms used for comparison and evaluation is shown in Table 6.4. Additionally, we also run variants of the same algorithms using the regular form of the covariance update, while the algorithms listed are given using the more robust Joseph's form (see Section A.9.6). Nevertheless, this resulted in the exact same result as with the corresponding algorithm, and therefore, we omit these results. Note that most of these algorithms and ingredients were also proposed and investigated by Rhode [Rho16]. While we re-implemented these algorithms for consistency in our own environment, we evaluated the results against the code examples provided by Rhode [Rho16] for download as supplementary material. An exception is the algorithm proposed by Altmannshofer, Endisch, et al. [AE+16] and M-SG-KF-IV-C, which is a combination of the algorithms by [Rho16] and [AE+16].

Table 6.4: Overview over Algorithms used for Experiment 6.2.

Algorithm name	Instrumental Variables (IV)	Anti-windup (SG)	Robust Estimation (M)	Constraints (C)	Algorithm
KF	x	x	x	x	Algorithm 8
KF-IV	✓	x	x	x	Algorithm 9
SG-KF	x	✓	x	x	Algorithm 10
SG-KF-IV	✓	✓	x	x	Algorithm 11
SG-KF-C	x	✓	x	✓	Algorithm 12
M-KF	x	x	✓	x	Algorithm 13
M-SG-KF	x	✓	✓	x	Algorithm 14
M-SG-KF-IV[†]	✓	✓	✓	x	Algorithm 15
M-SG-KF-C[‡]	x	✓	✓	✓	Algorithm 16
M-SG-KF-IV-C	✓	✓	✓	✓	Algorithm 17

[†] Modified from [Rho16], with Student-t loss function as in [AE+16]

[‡] Altmannshofer, Endisch, et al. [AE+16]

Overview of methods used for signal pre-processing We used the following smoothing and filtering algorithms prior to the estimation algorithms, while the pre-processed data was used as input to the estimators.

- Butterworth filter (BW) [But30]
- SaG-smoother [SG64]
- PKS-smoother [Rho16]

- DeePLS-smoother (see our proposed Algorithm 2)
- DeePKS-smoother (see our proposed Algorithm 3)

While the first ones are well known, the last two are the smoothing algorithms, which we will propose in Chapter 5, the DeePLS, and the DeePKS algorithm. The results are given in Table 6.5 and Figure 6.3, while a discussion can be found in Section 6.2.

Comments on Experiment 6.2

Although also a candidate for tackling the Errors-in-Variables problem, we did not consider evaluating any Recursive Generalized Total Least-Squares (RGTLS) approach since the existing algorithms lack robustness against outliers as reported in [Rho16], p. 159.

The choice of filtering and smoothing algorithms was originally motivated by Rhode [Rho16], who mentioned on page 135 that they used “a third order Butterworth filter with 1 Hz cut-off frequency” on all CAN signals in their publications [RG12] and [RB+14], where they applied a total least-squares approach. Nevertheless, he stated that for his final experiments, “all CAN signals pass a bank of independent PKS with the same configuration.” This is the reason we include the PKS method in our investigations. We will propose a generalization of PKS in Chapter 5. Since the proposed method of DeePLS smoothing is a generalization of SaG, the latter was also included.

Discussion of results for Experiment 6.2

The results of Experiment 6.2 can be found in Table 6.5 and a visualization of the error statistics of a selection of best-performing algorithm and pre-processing method is shown in Figure 6.3.

Discussion of Table 6.5 We find error statistics over all simulation runs in the form of the norm of the Root Mean Square Error (RMSE) of the parameter estimation errors vector. For each combination of the pre-processing method with an estimation algorithm, we show the median, minimum, and standard deviation of the norm of the root mean squared error. In the rightmost column, we further show an indicator if, over all runs, the error covariance matrix P within the Kalman filter variant was positive definite. We can observe that the latter was not the case for all estimation algorithm variants which used instruments (IV). The table is sorted, showing results with the lowest median RMSE on top.

We can observe that the best results, in terms of median RMSE, are the ones obtained using our proposed pre-processing smoothers DeePLS and DeePKS. At the bottom, methods involving Instrumental Variables IV perform worst, although the minimum values show that in some cases, good results can be obtained. These are examples only, as the one shown in the results of the previous Experiment 6.1, while the variance in these results is very high. We observed that the filters with IVs can be tuned such that specific examples achieve good results, but we did not manage to find a setting that was consistently good over all Monte-Carlo runs. Rhode already mentioned in [Rho16] p.182 that the method of Instrumental Variables is only applicable under certain conditions. One of them would be if the noise is auto-correlated, for example, when white noise runs through a Butterworth filter. Note that for this experiment, we did not use colored measurement noise but used noise drawn from a Student-t distribution. In any case, a more solid investigation of the exact conditions under which the method can be applied would be necessary for the application in a production environment. In summary, we found this method very difficult to apply.

Looking back at the top performers, we find it surprising that the SG-KF-C method performed best in this example. Only looking at Experiment 6.1, we would have expected that a robust variant with an M-Estimation scheme would perform best. Nevertheless, the results are very close, while the M-SG-KF-C filter, in second place, shows less in the standard deviation. This could be interpreted in that in exchange for performance, the robustness of the approach against different noise settings is traded. In any case, it seems that the proposed smoothers (DeePLS and DeePKS) are able to *robustly* improve the results for a variety of estimators.

We want to explain this by the capability of the algorithms to leverage measurements from more than one channel, which inherently makes them robust against outliers - even without explicitly considering that the noise patterns are non-normally distributed. This seems to have a positive effect on the parameter estimators and makes the need for robust parameter estimators (using the M-scheme) redundant. This can be seen very well in the graphical presentation of Figure 6.3.

Discussion of Figure 6.3 Figure 6.3 shows a selection of the best performing Kalman Filter variants over different pre-processing methods of a Monte-Carlo simulation performed over 100 runs from Experiment 6.2. More details of the same simulation study are provided in the numerical Table 6.5.

Looking at Figure 6.3, we can see Box - Charts with the error statistics of the norm of the RMSE of the parameter estimation errors. Results are grouped from left to right with estimators running on raw data (“none”), through a Butterworth filter (BW), a Savitzky-Golay (SaG) smoother and the proposed DeePLS-smoother. As expected from Experiment 6.1, without any pre-processing, the robust estimator variants (M) clearly outperform their non-robust counterparts. In this scenario, the effect of adding constraints is relatively low, but of course, in practice, this seems a useful way to be more robust against spurious outliers that might occur in reality. The results using Butterworth filters on all signals can be observed to show higher errors. This is most likely the case since they introduced an unequal lag into the measurements, leading to this effect.

With Savitzky-Golay (SaG) smoothers, we can already see an improvement for all shown estimators. We still see that the robust M-estimators outperform the non-robust versions, both in absolute errors as well as in the variance of the results.

The results obtained using the proposed DeePLS smoother clearly outperform all others. Also, the pre-processing obtained by the DeePLS smoother seems to robustly improve the results, such that the estimator, based only on the SG anti-windup, is able to outperform the others. This can also be seen when looking at numerical values presented in Table 6.5. For example, the best result obtained using the SaG-smoother is over 28 percent over the median error seen with DeePLS.

Conclusions from Experiment 6.2

We could see that pre-processing the values with smoothing algorithms prior to running the estimators can lead to substantial performance gains. The method that performed best for this scenario was the proposed Derivative-exploiting Polynomial Least Squares (DeePLS) method. It also seems the method is very robust to the heavy-tailed noise without any further algorithmic considerations. In this particular scenario, the recursive version of the proposed algorithm, DeePKS, also showed good performance but did not outperform our other method. This can most likely be explained by the fact that the trajectories in Experiment 6.2 are based on sine waves and, therefore, are difficult to reconstruct using a method suited for signals, which can be described as piece-wise constant polynomials.

Clearly, from the results obtained by this experiment, the method of Instrumental Variables (IV) seems not to be the method of first choice, at least not without further studies on the conditions when this method fails.

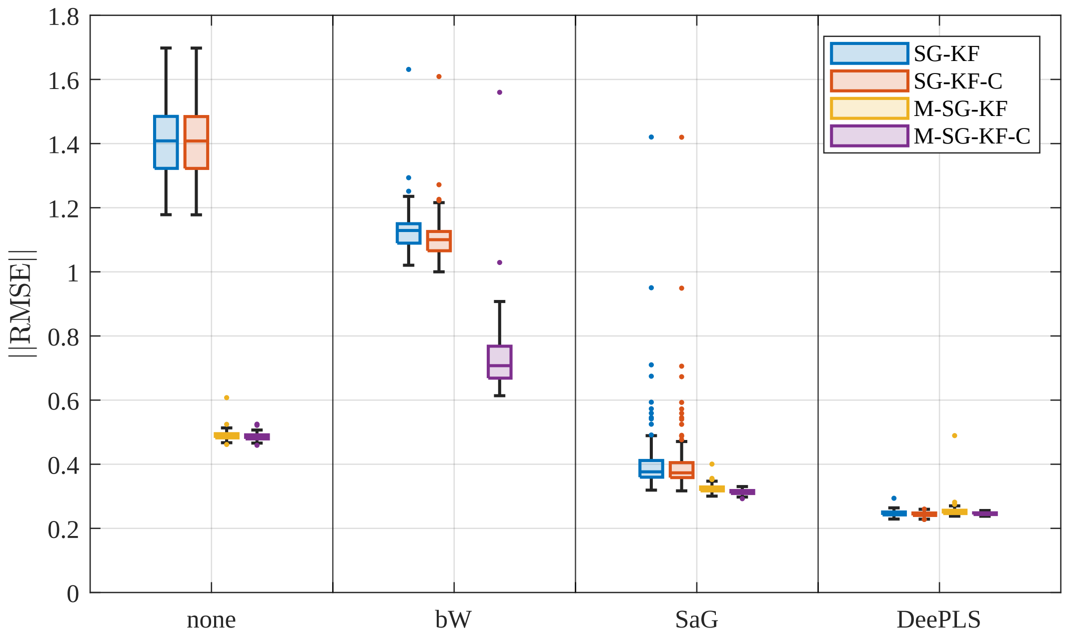

Figure 6.3: Comparison of various filtering methods and algorithms for the Linear-in-Parameters estimation problem of Experiment 6.2. Box-plots of error statistics (||RMSE||) over 100 Monte-Carlo simulations. We can observe how the proposed DeePLS smoother helps to reduce estimation bias and variance, performing best.

6.4.4 Simulation study: Comparison of existing Joint Estimation algorithms

In this simulation study, we want to evaluate and compare the performance of various joint estimation schemes, which allow for concurrent estimation of states and parameters. We identified the algorithms EKF, Unscented Kalman Filter (UKF), and Particle Filter (PF) as promising candidates for the vehicle state and parameter estimation problem. To evaluate the algorithms, we will perform simulations of the scenario and experiment described in the following Section 6.3.

Experiment 6.3

We run a comparison of the following five joint estimation algorithms

1. Extended Kalman Filter (EKF)
2. Unscented Kalman Filter (UKF)
3. Particle Filter (PF) with $N = 1e4$, $N = 1e5$ and $N = 1e6$ particles.

We created a ground truth of vehicle trajectories from Simulation Environment 6.2, which we use for recursive joint estimation of the parameters m, C_d, C_r, and to concurrently obtain a filtered vehicle speed value. The simulation scenario we use to evaluate the estimation algorithms consists of tracking a dynamic vehicle speed reference trajectory, which is based on velocity data defined by the Worldwide harmonized Light-duty vehicles Test Cycles (WLTC) Class 3 cycle, which is part of the Worldwide Harmonised Light Vehicle Test Procedure (WLTP) procedure defined by United Nations Economic Commission for Europe (UNECE) [UNE19]. The road slope is held constant at values of zero. Note that here, we use the unmodified reference trajectories as defined by the WLTP. For the particle filter, we performed a hyper-parameter sweep with different numbers of particles: $N = 1e4$, $N = 1e5$ and $N = 1e6$, since with lower numbers, no meaningful result could be obtained.

Simulation environment 6.2

The simulation environment is, in principle, identical to the one given in Simulation Environment 8.1, using parameters as given in Table 6.6. See Section 8.5.1 for further details. Here, the ground truth was created using the baseline PID controller to follow the desired trajectory, as described in Chapter 7. For the Unscented Kalman Filter, we used the same parameters as for the Extended Kalman Filter. The parameters specific for the UKF where chosen as $\alpha = 0.5$, $\beta = 2$ and $\kappa = 0$.

For the particle filters, we used the same state transition function with the augmented state as for the Kalman filters and added normally distributed process noise with a noise covariance matrix diagonal given as $[1e-5; 1e-7; 1e-7; 1e-8; 1e-6]^2$. To calculate the measurement likelihood function, the assumption of a multivariate normal error distribution with zero mean and variance one was taken. An overview of all the parameters for the estimators EKF, UKF and PF are shown in Table 6.6.

Implementation details of Simulation Environment 6.2

Simulation Environment 6.2 was realized in MATLAB® [Mat19b] R2019a. The sample time for both simulation and estimation algorithms was chosen to be 10 ms. In order to track the reference speed profiles, we used a standard PI controller with feed-forward, which was also used as a baseline in Chapter 7, where further implementation details can be found. We added noise to the measurements of vehicle speed and acceleration, with standard deviations of $\sigma_v = 0.03$ and $\sigma_a = 0.02$, respectively. The initial values for the estimated vehicle states were assumed to be known, with the vehicle starting at a standstill. The initial value of the parameter estimates $\hat{m}_0$ was set to 1400 kg instead of the true value of 1200kg. $\hat{C}_{\mathrm{d},0}$ was set to a value of 10 % above the true value of 0.015. $\hat{C}_{\mathrm{r},0}$ was set to a 20 % above the true value of 0.4262 used in the simulation. Please also note that we start the simulation at t=11s of the WLTP cycle, which is the time when the vehicle starts moving in the WLTP cycle.

Table 6.7: Simulation results: RMSE values of joint estimation algorithms. The UKF outperforms all alternative solutions.

RMSE	UKF	EKF	PF, N=1e4	PF, N=1e5	PF, N=1e6
v	**0.0055728**	0.0064713	0.3708564	0.6626081	0.0354357
m	**15.9673384**	15.9677569	19.0084108	18.2456309	17.7620933
C_d	**0.0105784**	0.0116569	0.0372842	0.0488737	0.0090376
C_r	**0.0002899**	0.0002935	0.0007469	0.0004780	0.0003290

Discussion of results for joint estimation comparison

Figure 6.4 shows a comparison of simulation results for the different joint estimation algorithms, namely the EKF, the UKF and three Particle Filters (PFs) with a varying number of particles N. The plot shows the average computation times (on a logarithmic scale) on the abscissa versus the norm of the Root Mean Square Errors (RMSE) of the estimated states and parameters on the ordinate.

Particle filter results One can observe that although the particle filter improves with a higher number of particles N, one cannot find an improvement over the less computationally intensive alternatives, the UKF and the EKF, which have considerably lower computation times. Quantitative details of the results can also be found in the detailed error statistics

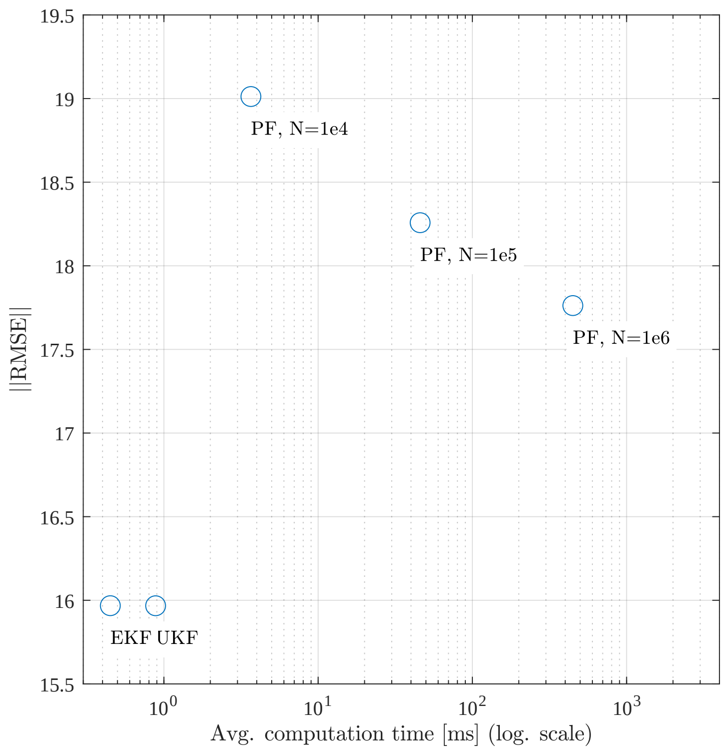

Figure 6.4: Results from Experiment 6.3. Comparison of simulation results for different joint estimation algorithms Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and particle filters (PF) with different sample sizes N. Plot shows average computation times (on a logarithmic scale) versus the norm of the Root Mean Square Errors (RMSE). Find a discussion in Section 6.2.

given in Table 6.7. We omit a presentation of qualitative results for the sake of brevity but want to mention that the particle filter with the lowest number of particles (N=1e4) is not capable of converging even to the vicinity of the true parameter solution.

Kantas, Doucet, et al. [KD+14] reports this behavior is a known problem when applying the particle filter to the parameter estimation problem, arising due to particle degeneracy. If an insufficient number of particles are present in the region of the true parameter combination, it is very unlikely that new particles are generated in that region. We tested several different re-sampling methods, which were reported to help mitigate this problem. These were the residual, stratified, and systematic methods, together with dynamically triggering the re-sampling, based on the number of effective particles as described in [LB+15]. As mentioned in Kantas, Doucet, et al. [KD+14], increasing the particles to a number of N=1e6 helps to partly overcome this problem. Due to the complexity of $\mathcal{O}(N)$ of the particle filter, this increases the computation time by a factor of 100. Please note that the experiment was performed only on a single run and not on a series of Monte Carlo simulations. This lies in the fact that one single run for the particle filters with N=1e4, N=1e5, and N=1e6 lasts more than 25 hours to simulate the WLTP cycle of 1800 seconds in real-world time. While this provides sufficient data for the evaluation of the computation times, only the tendency of the estimation quality becomes evident. Nevertheless, we did not find a Monte Carlo simulation would lead to any other insights for the following reasons. First, the particle filter seemed to be rather difficult to tune, at least for our application. An additional drawback is the stochastic nature of the Sequential Monte Carlo filter. This not only adds complexity to the tuning process but is also not a desirable property when using the filter output to adapt

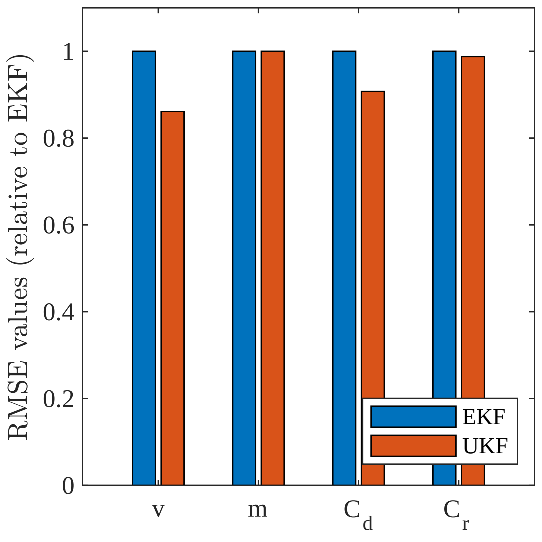

Figure 6.5: Relative comparison of estimation performance between EKF and UKF.

a model used in a closed-loop controller. Last but not least, finding the following statement in the preface of Rawlings, Mayne, and Diehl [RM+17], in the context of model predictive control, does not motivate to explore this path any further: "As with many sample-based procedures, however, it seems that all of the available sampling strategies in particle filtering do run into the curse of dimensionality. The low density of samples in a reasonably large-dimensional space (say N = 5) leads to inaccurate state estimates. For this reason, we omit further discussion of particle filtering in this edition."

Also, in a setting where the process and measurement noise is unimodal and close to normally distributed, the Unscented Kalman Filter can be seen as an "intelligent" particle filter (see [Sim06], p. 471), in which the particles are not randomly distributed.

Comparison between EKF and UKF Looking at the numerical results given in Table 6.7, we can calculate a relative improvement of the UKF compared to the EKF. This relative improvement is illustrated in a graphical way in Figure 6.5. We find the RMSE of the vehicle speed estimation reduced by over 13% and over 9% for the aerodynamic drag parameter C_d. Hence, the UKF clearly outperforms the EKF. This comes at the increased computational cost of a mean value of 0.88 ms for the UKF versus only 0.45 ms for the EKF in our experiments.

Result discussion As just discussed, the comparison between EKF and UKF shows improvements in performance when using the Unscented Kalman Filter. Another advantage of the Unscented Kalman filter lies in the reduced implementation effort since no a priori calculation of the Jacobians has to be provided. Another advantage could be the improved robustness in the presence of locally weakly unobservable states. [MO+09] showed this in their experiments but without giving further theoretical insights. The trade-off between performance and estimation quality clearly has to be discussed in an overall context for automotive engineering. One might want to consider that for the purpose of calculating the input to an adaptive nonlinear feedback controller, being able to run at a higher frequency than the controller is a desirable property. This has been reported to help stabilize the adaptive control loop, as mentioned in [GP17].

To conclude, the Kalman Filter variants show good performance at a very low computational cost. We also need to mention at this point that our implementation was not yet optimized for performance and that generating efficient C-code would be possible from our current implementation in MATLAB®. With this possibility in mind, we will further evaluate the joint UKF estimator in the experiments of Section 6.6.

6.4.5 Conclusion of results from preliminary investigations

We learned that although, at first glance, solutions based on the Linear-in-Parameters form of the longitudinal motion equation seem quite straightforward, many additional countermeasures are necessary to be able to provide a robust solution under realistic noise assumptions. These countermeasures have already been explored in the baseline solutions presented in Rhode [Rho16] and Altmannshofer, Endisch, et al. [AE+16], but their results can be substantially improved by using our proposed signal smoothing algorithm from Chapter 5.

Still, the joint estimation schemes similar to the one proposed in our own work in [BK16a] seem an interesting path to consider.

Clearly, one major disadvantage of the approach based on the LiP formulation is the shortcoming that it does not explicitly allow for estimation and filtering of the state value. This becomes especially important in the context of longitudinal tracking, an aspect that was not directly discussed in both references.

Another disadvantage of the approach is that one cannot directly estimate all three unknown parameters independently, since to obtain the equation in LiP form, we needed to perform a transformation which only allows to measure the combined parameter $C_\mathrm{r} \cdot m$. This also means that constraints can also only be applied to this combined parameter, which turns out to be a rather limiting factor when considering constraints. One advantage of the LiP-based estimators, at least when combined with the proposed DeePLS scheme, is the capability to converge at a very fast rate to the vicinity of the true parameters.

Therefore, we still want to consider both approaches in further evaluations. To obtain smooth estimations for the vehicle state as well, in the subsequent sections, we want to enhance the estimators based on the LiP formulation with another estimator solely for this purpose. We will present this solution in the following section.

6.5 Proposed solution for combined vehicle state and parameter estimation

As just mentioned in Section 6.4.5, we found that the solution based on the LiP formulation of the longitudinal motion equation has the capability to converge quickly to the vicinity of the true parameters. This was at least true if the estimator is combined with our proposed signal smoothing algorithm DeePLS, which we will derive in Chapter 5. To compensate for the shortcoming that with this approach, only parameters but no states can be estimated, we want to propose an enhancement, where we combine the LiP-based estimator with a conventional state estimator. We will propose to use an estimator of the Kalman filter family for this task also. Combining state and parameter estimators in separate algorithms is typically termed a *dual* filtering approach. It is well known that in order to achieve convergence of the solutions, one needs to consider the fact that the two problems are interlinked. Therefore, the algorithms need to be interlinked as well.

We want to stress at this point that the way we treat the problems is non-standard. We rather want to benefit from the best of the two worlds: the fast convergence properties of the parameter estimator based on the LiP approach and combine it with a state estimator based on a formulation of an ordinary differential equation. We are only aware of solutions for dual estimators, where both the parameter and the state estimator are based on a formulation as ODE.

The remainder of this section is organized as follows: In Subsection 6.5.1, we explain the overall architecture of the proposed approach. In Subsection 6.5.2, we derive an enhancement of the algorithm, on which we base the state estimator of the dual estimation scheme. In Subsections 6.5.3 and 6.5.4, we summarize two variants of the obtained algorithm, of which the first one is based on signal pre-processing using our DeePLS algorithm for smoothing, while the second uses DeePKS. Both of these smoothers will be derived as novel algorithms in Chapter 5. In Subsection 6.5.5, we want to critically discuss some of the limitations of the proposed approach.

6.5.1 Proposed dual estimation scheme

We propose a combined state and parameter estimation algorithm that has the following architecture.

Similar to dual estimation algorithms, we propose to use separate but interlinked algorithms for state and parameter estimation. The parameter estimator shall consist of the constrained M-estimation filter with adaptive process noise according to the Stenlund-Gustafsson anti-windup scheme, and hence be similar to the one proposed in [AE+16], but with the following differences: First, we use Joseph's form of the Kalman filter equations to improve numeric stability by making sure the covariance matrix remains positive and definite. Second, and most important, we want to feed the algorithm with the smoothed input values calculated by the DeePLS smoothing algorithm, which we develop in Chapter 5. We evaluated this algorithm already in Section 6.4.3 and found improved performance compared to the baseline solutions.

For the state estimation algorithm, we propose to apply a filter from the Kalman filter family but will have to derive an enhancement that allows us to consider the dependency of the current state of the parameter estimator. For this purpose, we choose to base these enhancements on a Square-root Central Difference Information Filter Square-root Central Difference Information Filter (SRCDIF) as originally developed by Liu, Worgotter, and Markelic [LW+12]. The reasons for this choice are as follows:

First, from our experiments comparing the EKF with the UKF in Section 6.4.4, we learned that the Sigma-Point filter showed improved performance at only a slightly increased computational cost. Second, when considering the dependency from the interlinked parameter estimator, we found numerical issues due to non-positive-definite error covariance matrices when using the standard UKF filter. The proposed SRCDIF was reported to prove stable and provide guarantees for the positive-definiteness of the error covariance matrices at reduced computational costs than other variants.

Next, we want to present a slight modification or enhancement to the original algorithm as proposed in [LW+12]. While the standard SRCDIF assumes that parameters of the state evolution function are known, in our formulation, we treat them as random Gaussian variables, of which we know the mean and the error covariance matrix. This allows the creation of the necessary link to the parameter estimator. Find the derivation of this algorithm in Section 6.5.2.

6.5.2 SRCDIF-PU: Modified Square-root central difference information filter with parameter uncertainty

Here we present a modified formulation of the SRCDIF algorithm [LW+12] presented in section A.6.5, which explicitly incorporates the dependency of time-varying parameters formulated as random variables $\theta_k \sim \mathcal{N}\left(\bar{\theta}, Q^\theta\right)$. Find motivation for this in the previous Section 6.5.3. The system under consideration is given as

$$x_k = f(x_{k-1}, u_{k-1}, \omega_{k-1}, \theta_{k-1}) \tag{6.25}$$

$$y_k = h(x_k) + \nu_k, \tag{6.26}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^{n_u}$, $\omega \in \mathbb{R}^{n_\omega}$ are the state, input and process noise vectors, and $y \in \mathbb{R}^{n_y}$, $\nu \in \mathbb{R}^{n_\nu}$ are the measurement/observation vector and measurement noise vector, respectively. We also assume the noises to be normally distributed as in (A.105). Note that the above system is less general than the one considered before since only additive measurement noise is present. Similar to section A.6.5, we define an augmented sigma-point matrix consisting of state vectors process noise vectors, but additionally, we explicitly augment the sigma-point matrix with parameter noise vectors $\mathcal{X}_i^\theta$:

$$\tilde{\mathcal{X}} = \begin{pmatrix} \mathcal{X} \\ \mathcal{X}^\omega \\ \mathcal{X}^\theta \end{pmatrix}. \tag{6.27}$$

Algorithm 1 from [LW+12] can then be enhanced as:

1. Initialize filter

$$\hat{x}_0^+ = \mathbb{E}[x_0] \tag{6.28}$$

$$\hat{S}_{x,0}^+ = \operatorname{chol}\left\{\mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^\mathsf{T}\right]\right\} \tag{6.29}$$

$$S_\omega = \sqrt{Q} = \operatorname{chol}\{Q\} \tag{6.30}$$

$$S_\nu = \sqrt{R} = \operatorname{chol}\{R\} \tag{6.31}$$

$$S_{\omega^\theta} = \sqrt{Q^\theta} = \operatorname{chol}\left\{Q^\theta\right\} \tag{6.32}$$

$$\tag{6.33}$$

2. For $k = 1, 2, \dots, \infty$ perform:

Generate sigma-points for prediction:

$$\tilde{x}_{k-1} = \begin{pmatrix} x_{k-1} \\ \bar{\omega} \\ \bar{\theta}_{k-1} \end{pmatrix} = \begin{pmatrix} x_{k-1} \\ 0 \\ \hat{\theta}_{k-1} \end{pmatrix} \tag{6.34}$$

$$\tilde{S}_{k-1} = \begin{pmatrix} S_{x_{k-1}} & 0 & 0 \\ 0 & S_\omega & 0 \\ 0 & 0 & S_{\omega^\theta} \end{pmatrix} \tag{6.35}$$

$$\tilde{\mathcal{X}}_{k-1} = \begin{pmatrix} \tilde{x}_{k-1} & \tilde{x}_{k-1} + h\tilde{S}_{k-1} & \tilde{x}_{k-1} - h\tilde{S}_{k-1} \end{pmatrix}. \tag{6.36}$$

Prediction equations:

$$\mathcal{X}_k^- = f\left(\mathcal{X}_{k-1}, u_{k-1}, \mathcal{X}_{k-1}^{\omega}, \mathcal{X}_{k-1}^{\theta}\right) \tag{6.37}$$

$$\hat{x}_k^- = \sum_{i=1}^{2n+1} w_i^{(m)} \mathcal{X}_{i,k}^- \tag{6.38}$$

$$A = \sqrt{w_2^{(c_1)}} \left(\mathcal{X}_{2:n+1,k}^- - \mathcal{X}_{n+2:2n+1,k}^-\right) \tag{6.39}$$

$$B = \sqrt{w_2^{(c_2)}} \left(\mathcal{X}_{2:n+1,k}^- + \mathcal{X}_{n+2:2n+1,k}^- - 2\mathcal{X}_{1,k}^-\right) \tag{6.40}$$

$$\hat{S}_{x_k}^- = \text{qr}\left\{\begin{pmatrix} A & B \end{pmatrix}\right\} \tag{6.41}$$

$$\iota_k^- = \left(\hat{S}_{x_k}^-\right)^{-\mathsf{T}} \left(\left(\hat{S}_{x_k}^-\right)^{-1} \hat{x}_k^-\right) = \left(\hat{S}_{x_k}^-\right)^{\mathsf{T}} \backslash \hat{S}_{x_k}^- \backslash \hat{x}_k^- \tag{6.42}$$

$$\hat{S}_{\iota_k}^- = \text{qr}\left\{\left(\hat{S}_{x_k}^-\right)^{-1} \mathbb{I}\right\} \tag{6.43}$$

Generate sigma-points for measurement update:

$$\tilde{\mathcal{X}}_k^y = \begin{pmatrix} \hat{x}_k^- & \hat{x}_k^- + h\hat{S}_{x_k}^- & \hat{x}_k^- - h\hat{S}_{x_k}^- \end{pmatrix} \tag{6.44}$$

Measurement update:

$$\mathcal{Y}_k^- = h\left(\tilde{\mathcal{X}}_k^y\right) \tag{6.45}$$

$$\hat{y}_k = \sum_{i=1}^{2n+1} w_i^{(m)} \mathcal{Y}_{i,k}^- \tag{6.46}$$

$$\hat{P}_{x_k \tilde{y}_k} = \sqrt{w_2^{(c_1)}} \hat{S}_{x_k}^- \left(\mathcal{Y}_{2:n+1,k}^- - \mathcal{Y}_{n+2:2n+1,k}^-\right)^{\mathsf{T}} \tag{6.47}$$

$$U = \left(\hat{S}_{x_k}^-\right)^{-\mathsf{T}} \left(\left(\hat{S}_{x_k}^-\right)^{-1} \hat{P}_{x_k \tilde{y}_k}\right) S_v^{-\mathsf{T}} \tag{6.48}$$

$$\hat{\iota}_k^+ = \hat{\iota}_k^- + U S_v^{-1} \left(y_k - \hat{y}_k^- + \hat{P}_{x_k \tilde{y}_k}^{\mathsf{T}} \hat{\iota}_k^-\right) \tag{6.49}$$

$$S_{\iota_k}^+ = \text{cholupdate}\left\{\hat{S}_{\iota_k}^-, U, +\right\} \tag{6.50}$$

In above algorithm, equations (6.38)-(6.50) are not affected by the changes. We can further extend the algorithm given in [LW+12] as presented above in order to calculate the state and covariance matrix as follows:

$$\hat{S}_{x_k}^+ = \text{qr}\left\{\left(S_{\iota_k}^+\right)^{-1} \mathbb{I}\right\} \tag{6.51}$$

$$\hat{P}_{x_k} = \hat{S}_{x_k}^+ \left(\hat{S}_{x_k}^+\right)^{\mathsf{T}} \tag{6.52}$$

$$\hat{x}_k^+ = \hat{P}_{x_k} \hat{\iota}_k^+. \tag{6.53}$$

6.5.3 Proposed DeePLS-Dual based state and parameter estimation

For concurrent estimation of states and parameters applied to longitudinal vehicle dynamics estimation, we propose Algorithm 4. It can be seen as a dual estimation algorithm, which combines the parameter estimator based on a LiP formulation, DeePLS-M-SG-KF-C, with the enhanced Square-root Central Difference Information Filter SRCDIF-PU. The parameter estimator is a robust and constrained Kalman Filter-based algorithm with an anti-wind-up mechanism, which is fed by the output of the proposed DeePLS smoother of Chapter 5.

In Algorithm 4, the 0-index for example in $\hat{\theta}_{0,k}$ and $P_{\theta_{0,k}}$ denotes this is the value given at the current center point of the moving window of the DeePLS algorithm (see Section 5.4.3).

Algorithm 4 Proposed DeePLS-Dual algorithm for combined state and parameter estimation.

```
Require: see requirements of DeePLS, M-SG-KF-C and SRCDIF-PU
1: Inputs:
       $< \boldsymbol{y}^0, \boldsymbol{t}^0 >, \dots, < \boldsymbol{y}^j, \boldsymbol{t}^j >$        ▷ as defined
   for Algorithm 2
2: $\hat{v}_{0,k}$, $\hat{a}_{0,k}$ ← from Algorithm DeePLS        ▷ as proposed in Algorithm 2
3: $y^{\theta}_{0,k}$, $\Phi_{0,k}$ ← from LiP formulation of long. veh. dynamics        ▷ see (6.4)
4: $\hat{\theta}_{0,k}$, $P_{\theta_{0,k}}$ ← from Algorithm M-SG-KF-C        ▷ as given in Algorithm 16
5: $S_{\omega^{\theta}} \leftarrow \text{chol}\{P_{\theta}\}$
6: $\hat{x}^+_k$, $P_{x,k}$ ← from Algorithm SRCDIF-PU        ▷ as proposed in Section 6.5.2
7: return $\hat{x}^+_k$, $P_{x,k}$, $\hat{\theta}_{0,k}$, $P_{\theta_{0,k}}$
```

6.5.4 Proposed DeePKS-Dual based state and parameter estimation

The DeePKS-based Dual state and parameter estimator as we propose in Algorithm 5 is identical to the algorithm presented in Section 6.5.3, except here we use the DeePKS smoother as a signal pre-processing unit.

Algorithm 5 Proposed DeePKS-Dual algorithm for combined state and parameter estimation.

```
Require: see requirements of DeePKS, M-SG-KF-C and SRCDIF-PU
1: Inputs:
       $< \boldsymbol{y}^0, \boldsymbol{t}^0 >, \dots, < \boldsymbol{y}^j, \boldsymbol{t}^j >$        ▷ as defined
   for Algorithm 3
2: $\hat{v}_{0,k}$, $\hat{a}_{0,k}$ ← from Algorithm DeePKS        ▷ as proposed in Algorithm 3
3: $y^{\theta}_{0,k}$, $\Phi_{0,k}$ ← from LiP formulation of long. veh. dynamics        ▷ see (6.4)
4: $\hat{\theta}_{0,k}$, $P_{\theta_{0,k}}$ ← from Algorithm M-SG-KF-C        ▷ as given in Algorithm 16
5: $S_{\omega^{\theta}} \leftarrow \text{chol}\{P_{\theta}\}$
6: $\hat{x}^+_k$, $P_{x,k}$ ← from Algorithm SRCDIF-PU        ▷ as proposed in Section 6.5.2
7: return $\hat{x}^+_k$, $P_{x,k}$, $\hat{\theta}_{0,k}$, $P_{\theta_{0,k}}$
```

6.5.5 Limitations of the proposed approach

As commented earlier, the parameter estimates for both Algorithm 4 and Algorithm 5 are provided only for the center points of the current moving window of the smoothing algorithms. While the *center points* of the smoothing algorithms could, in theory, also be set to the right-most value of the window, this would be identical to the current values. In practice, to obtain optimal smoothing results, the recommendation is to choose this to be the symmetric point of the window. We use the just-mentioned setting for all our experiments. Hence, typically, the parameter estimates would then be delayed by a number of m samples.

In the case of tracking time-invariant parameters θ, the delayed estimate is, in fact, identical to the non-delayed value. Also, for piece-wise constant parameters, they would be identical most of the time. Nevertheless, this could be of relevance if general time-varying parameters need to be tracked.

6.6 Evaluation of the proposed algorithms

We will evaluate the proposed combined state and parameter estimation algorithm in an experiment based on the simulation of a driving scenario described in Section 6.3. As a baseline, we will use the algorithm based on the Linear-in-Parameters model proposed by [AE+16], which we also evaluated in the experiments described earlier in the preliminary investigations in Section 6.4.3. We will use this algorithm both on raw measurements as well as on pre-processed data from the PKS-smoother proposed in Rhode [Rho16]. We then compare the estimation performance to the following algorithms proposed in this thesis: First, a joint estimation algorithm based on an Unscented Kalman Filter. A similar algorithm based on the joint estimation scheme was also proposed earlier in the author's publication in [BK16a]. We will provide details about this algorithm in Section 6.4.4, and denote it here as "UKF joint". Second, we compare the results to two variants of the proposed dual estimation algorithm from Section 6.5. The first uses pre-processed data resulting from DeePLS smoothing, which we will refer to as "DeePLS dual", and the second one from DeePKS smoothing. The latter will be termed "DeePKS dual". Remember that these algorithms can be seen as a combination of a Square-root central difference filter SRCDIF for the state estimation part and a robust Linear-in-Parameters estimation scheme for the parameter estimation part. The latter operates on data smoothed by the proposed smoothing algorithms presented in Chapter 5. The enhanced performance of this algorithm was already discussed in Section 6.2, which also explains the reasoning behind this choice of algorithms.

In this section, we first want to provide some details about the experiment used to evaluate the proposed algorithms before presenting and discussing the results in subsection 6.6.3.

6.6.1 Experiment description

Simulation environment 6.3

The simulation environment we use to evaluate the estimation algorithms consists of tracking a dynamic vehicle speed reference trajectory similar to what we used in Scenario 2 of Simulation Environment 7.1 and presented in Section 7.5.1.

There, the reference was based on velocity data defined by the WLTC Class 3 cycle, which is part of the WLTP procedure defined by UNECE. The scenario differs in that a different modification rule is applied to the original speed profile, which is given by the formula

$$v^d(t) = \begin{cases} 2.5 & \text{if } \left(v^d_{WLTC} < 2.5 \wedge t \in [100s, 1500s]\right) \\ & \vee\ (t \in [440s, 600s]) \vee\ (t \in [1000s, 1460s]) \\ v^d_{WLTC}(t) & \text{else,} \end{cases} \tag{6.54}$$

which basically means that except for the initial acceleration and the final deceleration phase, the minimum velocity is kept to a value of 2.5 [m/s^2]. Additionally to the modification from Chapter 7, in the sections between 100 and 1500, 440 and 600, as well as 1000 to 1460 seconds, the vehicle speed reference is also kept at 2.5 [m/s^2]. The reason for this modification is to add long-lasting stationary phases to the original cycle. Phases of constant velocity are challenging for the estimator since, with zero acceleration, insufficient excitation is present for the Linear-in-Parameters-based approach (see analysis in Section 6.3.2). For further details, like the re-sampling method used and the definition of a variable road grade, please refer to Section 7.5.1. Parameters used for Simulation Environment 6.3 can be found in Table 6.8. The vehicle speed and acceleration profiles obtained are illustrated in Figure 6.6.

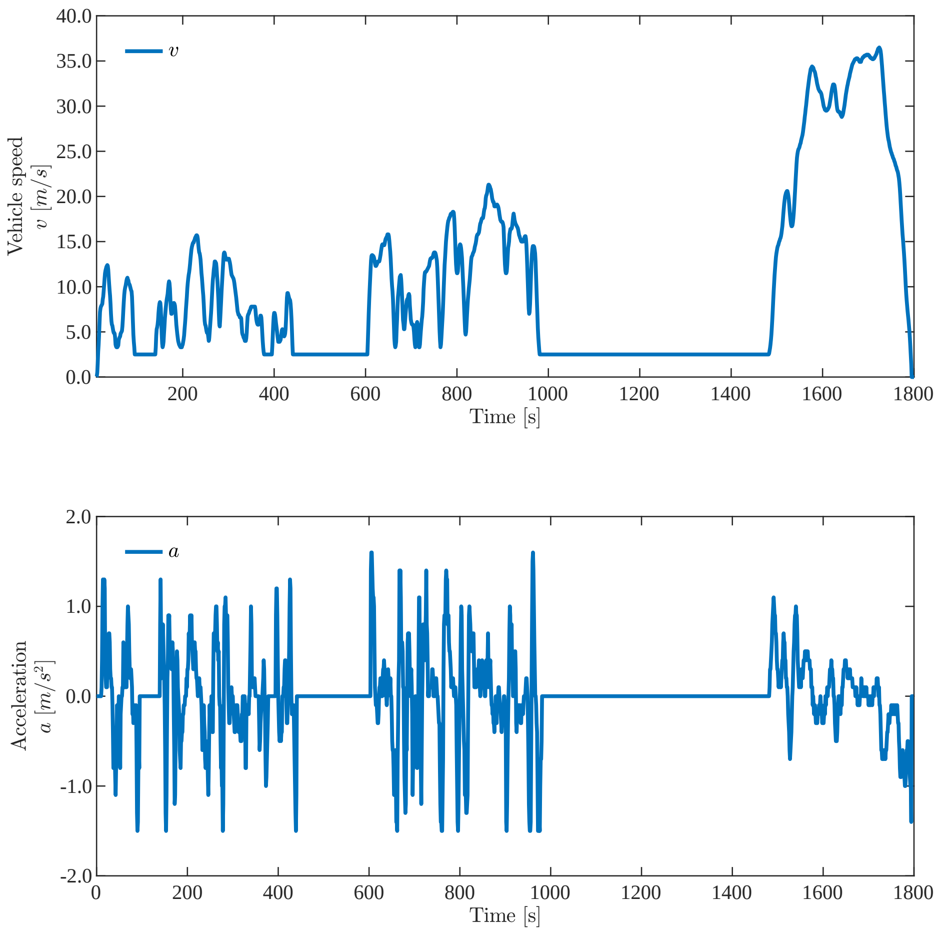

Figure 6.6: Vehicle speed and acceleration profiles of Simulation Environment 6.3, obtained as modification of a WLTC cycle as described in Section 6.3.

Experiment 6.4

For this experiment, we evaluate various combined state and parameter estimation algorithms and compare the results. We use data obtained from Simulation Environment 6.3, which is initially generated in a closed loop fashion using the ANMPC controller described in Section 7.5.1 together with the joint UKF estimator from Section 6.4.4 (UKF joint). The obtained ground truth trajectories are then used as a basis to create a variety of noisy trajectories for a Monte Carlo study by adding Gaussian noise obtained from a random number generator with different seeds. We then feed this data into different estimators as follows:

- M-SG-KF-C (see Algorithm 16)
- PKS + M-SG-KF-C (see Algorithm 16 and 1)
- DeePLS dual (see proposed Algorithm 4)
- DeePKS dual (see proposed Algorithm 5)
- UKF joint (as proposed in Section 6.4.4)

The M-SG-KF-C algorithm (see Algorithm 16) is used as a first baseline. For a second baseline, we also combined this algorithm with the PKS data pre-processing scheme presented in [Rho16], which we denote with “PKS + M-SG-KF-C”. Find results of this experiment in Section 6.6.3, in Figure 6.7, 6.8 and Figure 6.9.

6.6.2 Implementation details

In order to make the results as independent from hyper-parameter tuning as one can probably achieve, we used the same set of parameters for all the algorithms we compare wherever they exist. If different parameters are needed due to the type of the algorithm, we derived them from the same set of “meta-parameters”. What we mean by this is that, for example, we can derive the square root of the state covariance matrix S_x of the SRCDIF algorithm, which is used within the DeePLS-dual framework, from the initial state uncertainty standard deviation $\hat{x}_0^{std}$ by just building the diagonal matrix. The augmented state covariance matrix P_0 for the joint UKF algorithm, however, was calculated as $[\hat{x}_0^{std}, \hat{p}_0^{std}]^2$, also using the initial state uncertainty, but also the initial parameter uncertainty standard deviation $\hat{p}_0^{std}$. The only difference we had to make to get reasonable results was by setting the process noise standard deviation w_x^{std} to algorithm-specific values. This meta-parameter is used for the calculation of the augmented process noise covariance matrix of the joint UKF algorithm, as well as the square root of the process noise covariance matrix of the SRCDIF used in DeePLS-dual. In the CRAWKF algorithm, no such parameter exists since the true state is not event estimated by the nature of the algorithm. This choice can easily be justified since these parameters are, unlike the other ones, not directly motivated by problem formulation itself but rather algorithm-specific.

We perform a Monte Carlo Simulation over ten runs with a variation of measurement noise sampled from distributions with different random seeds. As a remark, we want to note that this evaluation takes around 20 hours on a high-performance laptop, and from undisclosed experiments on the first few minutes of the WLTP cycle, but with 50 Monte Carlo runs, we found that the error statistics are well covered already with this reduced number of 10 runs. The initial values for all parameter estimation algorithms were set to $m_0 = 1800$, $C_{\mathrm{d},0} = 0.8$ and $C_{\mathrm{r},0} = 0.018$, compared to the true values of $m = 1500$, $C_{\mathrm{d}} = 0.65$ and $C_{\mathrm{r}} = 0.015$. You can find a list of estimator parameters used in Experiment 6.4 in Table 6.9. In the following, we want to provide some further algorithm-specific details.

M-SG-KF-C settings The M-SG-KF-C algorithm (see Algorithm 16) is identical to the algorithm called CRAWKF in Altmannshofer, Endisch, et al. [AE+16], except that we use the numerically more stable calculation formula for the covariance update derived from Joseph’s form (as presented in Section A.9.6). This does not change the results of our experiments. The parameter constraints were set to obey $m \in [1000, 3000]$, $C_{\mathrm{d}} \in [0.1, 1]$ and $C_{\mathrm{r}} \in [0.012, 0.05]$. This leads per definition of the estimated parameter in the Linear-in-Parameters model of $\theta = [m, Cd, m \cdot Cr]^{\mathsf{T}}$ to $\theta_{max} = [3000, 1, 0.05]^{\mathsf{T}}$ and $\theta_{min} = [1000, 0.1, 0.0012]^{\mathsf{T}}$. Remember, one can only estimate the lumped parameter $m{\cdot}Cr$ using the LiP model, as discussed earlier. Also, M-SG-KF-C is only capable of providing parameter estimates but cannot provide any filtered values for the states, as discussed in Section 6.3.4

UKF joint settings To propagate the state dynamics within the UKF, we realized the integration of the general continuous-time state space description of the longitudinal vehicle dynamics as given in (3.69a), using a fourth-order Runge-Kutta scheme with $M = 5$ steps per interval as given in Algorithm 18 of Section B. We modeled the system dynamics with additive noise for both process and measurement noise.

DeePLS / PKS / DeePKS settings Please find a general description and derivation of all pre-processing algorithms, PKS, DeePKS, and DeePLS in Chapter 5. For this experiment, we run all algorithms in a smoothing setting, which means that the center point of the moving window, on which the algorithms operate, is the half window size of $M = 8$ samples plus one in the past from the current sample. This also means that we have to delay the parameter estimation by the same amount of samples. Both DeePLS and DeePKS run on the multi-channel data consisting of vehicle speed and acceleration, while for PKS, two separate banks of smoothers run on each channel. All parameters for the dual estimators, consisting of parameters for SRCDIF and the LiP estimator, can be found in Table 6.9.

6.6.3 Results

We can find results of a Monte Carlo simulation of Experiment 6.4 in Figure 6.7, 6.8 and Figure 6.9. Numerical results are given in Table 6.10. We calculated statistics of the residuals over time for the estimates of the vehicle mass parameter, the aerodynamic drag parameter, and the rolling resistance. For the proposed joint UKF algorithm (which can be seen as an extension of the algorithm proposed in [BK16a]) and the proposed dual estimators based on DeePLS and DeePKS, we also calculated errors of the state estimates for the vehicle speed v, while by nature of the baseline algorithm M-SG-KF-C proposed in [AE+16], no such estimate is provided.

Looking at the error statistics given in Table 6.10, we can see that the proposed DeePLS dual estimator clearly outperforms the other estimators. We can also see that adding PKS smoothing prior to the baseline algorithm M-SG-KF-C does not improve the results. The proposed multi-channel variant of PKS, namely the DeePKS algorithm, did also not help to improve the results in this scenario. For this reason, we focus on the rest of the visualizations on the remaining algorithms.

Looking at the residuals over time shown in Figure 6.7 first, we can observe that the proposed DeePLS dual estimator outperforms the other estimators. All three parameter estimates converge faster to the proximity of the true values compared to the baseline algorithm M-SG-KF-C [AE+16] and the joint UKF algorithm. Also, as can be seen from the Box-chart evaluation in Figure 6.9, not only the average and median RMSE values are smaller over the various runs of the Monte Carlo simulation, but the standard deviation and outliers are also reduced by the proposed algorithms. In terms of the variance of estimation results, the joint UKF performs best, while the DeePLS dual algorithm shows the lowest average and median RMSE values.

Table 6.10: Average RMSE values obtained in Experiment 6.4

	v	m	C_d	C_r	$\|\cdot\|$
DeePLS dual	**0.00426**	**3.80063**	**0.01383**	**0.00008**	**3.80066**
M-SG-KF-C	NaN	4.89674	0.01780	0.00029	4.89678
UKF joint	0.00783	6.37527	0.03120	0.00030	6.37536
PKS + M-SG-KF-C	NaN	10.56086	0.08369	0.00141	10.56119
DeePKS dual	0.00452	11.23993	0.08300	0.00144	11.24024

Result discussion

From the preliminary results in Section 6.4, one could already expect that the DeePLS based parameter estimator will outperform the other algorithms considered. We also see that adding PKS and its multi-channel counterpart is not achieving a better result. We believe this is due to the low-pass characteristics of the filters, which add bias to the following estimation. While the fastest convergence to the vicinity of the true parameters is achieved with the DeePLS-based dual estimator, we see that the Joint UKF solution also reliably converges to the true parameters. The baseline algorithms and the PKS-based estimators, nevertheless, show a certain bias, which becomes especially visible in the zoomed plot of Figure 6.8.

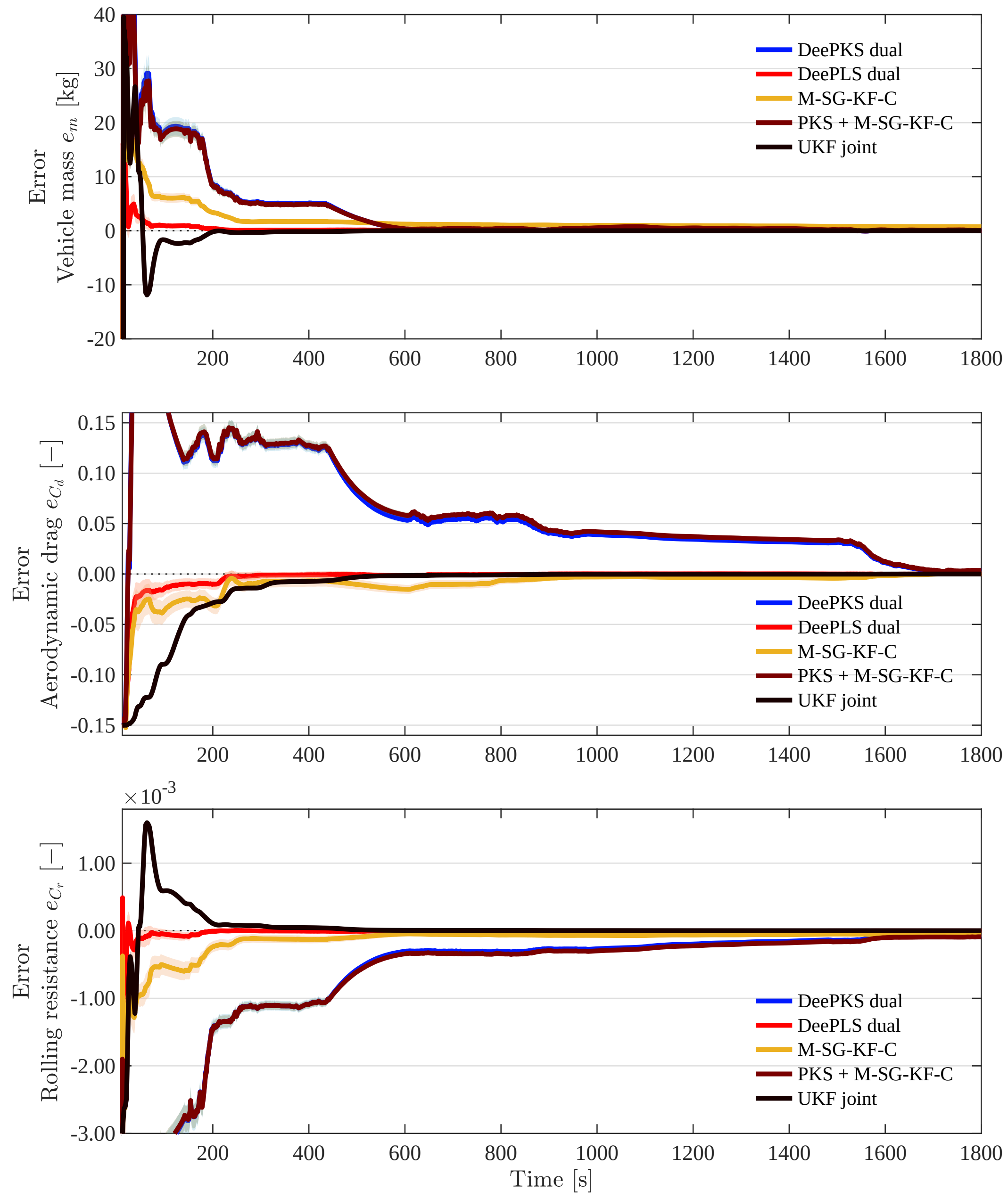

Figure 6.7: Simulation results of Experiment 6.4. Comparison of estimation of the proposed DeePLS dual estimator (red) and the joint UKF estimator (black) with other estimators. Dark-colored lines show average residuals over time (in seconds) of the parameter estimates for vehicle mass e_m (top), aerodynamic drag e_{C_d} (center), and rolling resistance e_{C_r} (bottom). Light-colored areas show confidence intervals as residuals $\pm\sigma$.

6.7 Conclusion

6.7.1 Summary and discussion

In this chapter, we discussed various options to realize a combined state and parameter estimator that is able to learn and track the parameters relevant to the longitudinal motion of the vehicle. These are the speed as system state and the longitudinal vehicle parameters of a variable vehicle mass, rolling resistance coefficient, and aerodynamic drag. We reviewed existing literature in this area, both from an application and an algorithmic point of view.

We are considering approaches based on the longitudinal vehicle motion model, which do not need additional vehicle sensors but can leverage the information from all sensors we assume to be present in automated vehicles. These are vehicle speed, acceleration, and road grade but also virtual sensors like the power-train and brake forces acting on the wheels.

We found two main categories especially suited for the task. First, methods based on a reformulation of the vehicle motion equation, leading to a Linear-in-Parameters model. This, in theory, allows the application of recursive least-squares methods, but our analysis has shown that this is only the case when neglecting the noise characteristics one needs to consider under realistic assumptions. Based on existing literature that also followed this approach, we conducted a series of experiments and found that special care is needed to create a solution that can be robustly applied. We found that applying the correct signal pre-processing methods can be the key to improving the quality of the estimators and producing robust results. Especially using the smoothing method, which was developed in Chapter 5 for this purpose, could help to make the estimator converge faster and more reliably to the true result.

One drawback of this first approach is the fact that it is not possible to obtain filtered values for the state concurrently with the parameter estimates. For this reason, we investigated two main paths. The first one is to use a formulation as an ordinary differential equation and perform joint estimation schemes, which augment the original state space and add the parameters as virtual states. The second one is a dual estimation scheme, where states and parameters are estimated in two interlinked but separate estimators. We found that we can leverage the advantages of the parameter estimation scheme based on the Linear-in-Parameters model and use this in combination with a state estimator to create a problem-specific dual estimation scheme.

To find out which algorithms are suited to solve the problem formulated as an ordinary differential equation, we performed another comparison of methods to find that the Sigma Point Kalman filter family offers both good convergence as well as computation times, which make an application feasible on an embedded system.

Based on these findings, two algorithms were compared to baseline approaches in further simulation experiments. First, a joint estimation scheme that uses an Unscented Kalman Filter to estimate states and parameters concurrently. Second, we developed a solution using a dual estimation scheme. The latter can be split into two parts: the parameter estimator and the state estimator. The parameter estimator uses smoothed data obtained from the proposed DeePLS algorithm in a pre-processing step, while the algorithm that performs the estimation is a robust, constrained Linear Kalman Filter with an anti-covariance-windup mechanism (M-SG-KF-C as in Algorithm 16). For the state estimator, we chose an enhanced version of a Square-root Central Difference Filter (SRCDIF) since this allows us to easily consider the interdependence of the estimation error covariance coming from the interlinked parameter estimator. Also, this filter is known for its guarantees of providing positive definite error covariance matrices, which otherwise would cause numerical instabilities of the overall solution.

The solution we found is real-time capable and provides accurate results in our simulation experiments, which converge, unlike the baseline algorithms, to the true parameter values, and to the vicinity in a reasonable time. Thanks to using a Bayesian approach, we additionally obtain estimates of the confidence of the solution. We considered constraints and robustness against disturbances.

6.7.2 Contribution

The work presented in this chapter partly builds on work published in the author's publication [BK16a] but extends this work significantly and includes substantial, previously unpublished work. A short summary of the main contributions presented in this chapter can be given as follows:

- Detailed review of related work on vehicle mass and tractive force parameter estimation in general and various algorithmic solutions in particular for methods based on the longitudinal vehicle motion model.
- Analysis regarding observability and persistence of excitation for different problem formulations
- Discussion and comparison of various possible solution approaches, including the one proposed by the author of this thesis in [BK16a]
- Detailed analysis of the effects of real-world conditions for Linear-in-Parameters estimation, with illustrative experimental results.
- Comparison and analysis of joint estimation schemes based on an ordinary difference equation formulation
- Proposal and evaluation of a novel hybrid dual estimation algorithm for combined state and parameter estimation, which builds on the DeePLS algorithm proposed in Chapter 5 and [Bue22], and outperforms all baseline algorithms found in the literature.

Table 6.5: Comparison of different combinations of parameter estimation methods with different pre-processing algorithms in Experiment 8.1. Table values show result statistics over 100 Monte-Carlo runs. Sorted for lowest median values of ||RMSE|| on top. Right: positive-definiteness of error covariance matrix P.

Method	**Preprocessing**	||RMSE||			$P > 0$
		median	min	std	
SG-KF-C	**DeePLS**[*]	**0.244**	0.228	0.007	✓
M-SG-KF-C	**DeePLS**[*]	**0.246**	0.238	**0.004**	✓
SG-KF	DeePLS[*]	0.246	0.229	0.008	✓
M-SG-KF	DeePLS[*]	0.250	0.238	0.025	✓
M-SG-KF-C	DeePKS[*]	0.266	0.249	0.047	✓
M-SG-KF	DeePKS[*]	0.266	0.251	0.044	✓
SG-KF-C	DeePKS[*]	0.271	0.246	0.014	✓
SG-KF	DeePKS[*]	0.271	0.248	0.015	✓
M-SG-KF-C	SaG	0.313	0.293	0.007	✓
M-SG-KF	SaG	0.323	0.301	0.013	✓
SG-KF-IV	none	0.324	0.164	35.315	x
SG-KF-IV	DeePLS	0.324	0.237	8.424	x
SG-KF-C	SaG	0.373	0.317	0.135	✓
SG-KF	SaG	0.376	0.319	0.134	✓
SG-KF-IV	SaG	0.383	0.319	0.904	x
SG-KF-IV	DeePKS	0.396	0.259	5.798	x
M-SG-KF-C[†]	none	0.484	0.459	0.011	✓
M-SG-KF	none	0.489	0.462	0.016	✓
M-SG-KF-C	PKS	0.512	0.403	0.209	✓
SG-KF-IV	PKS	0.513	0.322	3.301	x
KF-IV	none	0.519	0.147	104.141	x
KF-IV	DeePLS	0.532	0.439	9.494	x
KF	DeePLS	0.542	0.485	0.019	✓
M-SG-KF	PKS	0.551	0.405	0.218	✓
M-KF	DeePLS	0.608	0.573	0.017	✓
KF-IV	SaG	0.637	0.536	0.957	x
KF-IV	PKS	0.640	0.486	16.771	x
M-SG-KF-C	BW	0.707	0.614	0.113	✓
KF	SaG	0.709	0.619	0.132	✓
M-KF	SaG	0.732	0.688	0.016	✓
SG-KF	PKS	0.737	0.581	0.053	✓
SG-KF-C	PKS	0.745	0.580	0.055	✓
M-KF	PKS	0.761	0.633	0.268	✓
KF	DeePKS	0.771	0.729	0.029	✓
KF-IV	DeePKS	0.814	0.702	7.130	x
M-KF	DeePKS	0.814	0.784	0.017	✓
KF	PKS	0.876	0.788	0.043	✓
M-SG-KF-IV	SaG	0.967	0.294	21243001.164	x
SG-KF-C	BW	1.100	1.000	0.070	✓
SG-KF	BW	1.129	1.021	0.069	✓
KF	BW	1.194	1.064	0.076	✓
SG-KF-C	none	1.408	1.178	0.125	✓
SG-KF	none	1.408	1.178	0.126	✓
M-KF	none	1.426	1.390	0.017	✓
KF	none	1.777	1.685	0.056	✓
SG-KF-IV	BW	1.985	1.884	0.044	x
KF-IV	BW	2.008	1.892	0.041	x
M-KF	BW	4.172	2.752	0.449	✓
M-SG-KF-IV	BW	4.431	1.299	1036160.062	x
M-SG-KF	BW	5.179	2.376	0.673	✓
M-SG-KF-IV	DeePKS	330.480	0.262	85362.685	x
M-SG-KF-IV-C	PKS	422.181	0.379	168950.020	x
M-SG-KF-IV[‡]	PKS	472.599	0.321	722540.715	x
M-SG-KF-IV[‡]	none	626.096	0.236	852155.516	x

[†] as in Altmannshofer, Endisch, et al. [AE+16], [‡] as in Rhode [Rho16]
[*] ours, as proposed in Chapter 5

Table 6.6: Estimator parameter used in Experiment 6.3.

Symbol	Description	Value	Unit
φ	Road grade	0 / sinusoidal	[deg]
m	True vehicle mass	1200	[kg]
$\hat{m}_0$	Initial value of estimated vehicle mass	1400	[kg]
m_I	Mass resulting from power-train inertia	50	[kg]
η_pwt	Power-train efficiency	0.89	[-]
R_pwt	Power-train ratio	8.446	[-]
r_eff	Effective wheel radius	0.3	[m]
g	Gravitational constant	9.81	[N]
C_r	Rolling resistance coefficient	0.0150	[-]
C_d	Lumped aerodynamic drag coefficient	0.4262	[kg/m]
$\hat{C}_{\mathrm{r},0}$	Initial estimation of C_r	$0.0150 \cdot 1.1$	[-]
$\hat{C}_{\mathrm{d},0}$	Initial estimation of C_d	$0.4262 \cdot 1.2$	[kg/m]
T_s	Simulation sampling time	0.01	[s]
τ_e	Time constant of engine torque response	0.20	[s]
τ_br	Time constant of brake torque response	0.05	[s]
Q_0	(Initial) process noise covariance matrix	$q_{1,1} = (1e{-}5)^2$	$[(m/s)^2]$
		$q_{2,2} = (1e{-}7)^2$	$[(m/s^{-1})^2]$
		$q_{3,3} = (1e{-}7)^2$	$[kg^2]$
		$q_{4,4} = (1e{-}10)^2$	$[(kg/m)^2]$
		$q_{5,5} = (1e{-}8)^2$	$[-]$
R	Measurement noise covariance matrix	$r_{1,1} = (2e{-}2)^2$	$[(m/s)^2]$
		$r_{2,2} = (2e{-}2)^2$	$[(m/s^{-1})^2]$
P_0	Initial error covariance matrix	$p_{1,1} = (0.33)^2$	$[(m/s)^2]$
		$p_{2,2} = (0.33)^2$	$[(m/s^{-1})^2]$
		$p_{3,3} = (11)^2$	$[kg^2]$
		$p_{4,4} = (0.001)^2$	$[(kg/m)^2]$
		$p_{5,5} = (0.0001)^2$	$[-]$
α	Alpha parameter of UKF	0.5	[-]
β	Beta parameter of UKF	2.0	[-]
κ	Kappa parameter of UKF	0.0	[-]

Table 6.8: Simulation parameters for Simulation Environment 6.3

Symbol	Description	Value	Unit
T_s	Simulation sampling time	0.01	[s]
m	True vehicle mass	1500	[kg]
C_r	Rolling resistance coefficient	0.015	[-]
C_d	Lumped aerodynamic drag coefficient	0.65	[kg/m]
m_I	Mass equivalent from power-train inertia	40	[kg]
r_eff	Effective wheel radius	0.3	[m]
τ_e	Time constant of engine torque response	0.5	[s]
τ_br	Time constant of brake torque response	0.1	[s]
g	Gravitational constant	9.81	[N]

Table 6.9: Parameters used in Experiment 6.4

Parameter	Description	Value
Ts	Sample time	0.01
σ_v	Measurement noise std. dev. for v	0.03
σ_a	Measurement noise std. dev. for a	0.02
$\hat{x}_0$	Initial values	[0.1; 0; 0]
$\hat{p}_0$	Initial parameter estimation	[1700; 0.8; 0.018]
$\hat{x}_0^{std}$	Initial state uncertainty std. dev.	[1; 3e-4; 3e-04]/3
$\hat{p}_0^{std}$	Initial parameter uncertainty std. dev.	[35; 2.8e-3; 3.1e-4]/3
w_x^{std}	Process noise std. dev. UKF	[5e-6; 1e-5; 1e-5]
w_p^{std}	Parameter noise std. dev.	[1e-9; 1e-11; 1e-9]
v^{std}	Measurement noise std. dev.	[0.02; 1e-6; 1e-6]
$\operatorname{diag}\{P_0\}$	Augmented state covariance UKF	$[\hat{x}_0^{std}, \hat{p}_0^{std}]^2$
$\operatorname{diag}\{Q\}$	Process noise covariance UKF	$[w_x^{std}, w_p^{std}]^2$
$\operatorname{diag}\{R\}$	Measurement noise covariance UKF	$[v^{std}]^2$
α	UKF specific parameter	0.5
β	UKF specific parameter	2.0
κ	UKF specific parameter	0.0
w	DeePLS / DeePKS weights	$1/[\sigma_v; \sigma_a]^2$
N	Polynomial order for PKS / DeePLS / DeePKS	5
M	Half window size PKS / DeePLS / DeePKS	8
$\theta_{\max}$	Upper parameter limit for LiP estimation	[3000; 1; 150]
$\theta_{\min}$	Lower parameter limit for LiP estimation	[1000; 0.1; 12]
$\operatorname{diag}\{P_d\}$	Stenlund-Gustafsson “target” state covariance	[3, 1e-4, 3e-3]
$\operatorname{diag}\{P_\theta\}$	Parameter covariance M-SG-KF-C	[8e7, 16.2, 380880]/3
ν_s	Degrees of freedom of Student-t dist.	20
$\hat{\sigma}_s$	Scale parameter of Student-t dist.	5.47722557505166
r	Virtual meas. variance in M-SG-KF-C	1296
w_x^{std}	Process noise std. dev. SRCDIF	[8e-4; 1e-4; 1e-4]
$\operatorname{diag}\{S_x\}$	Square-root of state covariance SRCDIF	$\hat{x}_0^{std}$
$\operatorname{diag}\{S_{q_x}\}$	Square-root of process noise covariance SRCDIF	w_x^{std}
$\operatorname{diag}\{S_r\}$	Square-root of meas. noise covariance SRCDIF	v^{std}

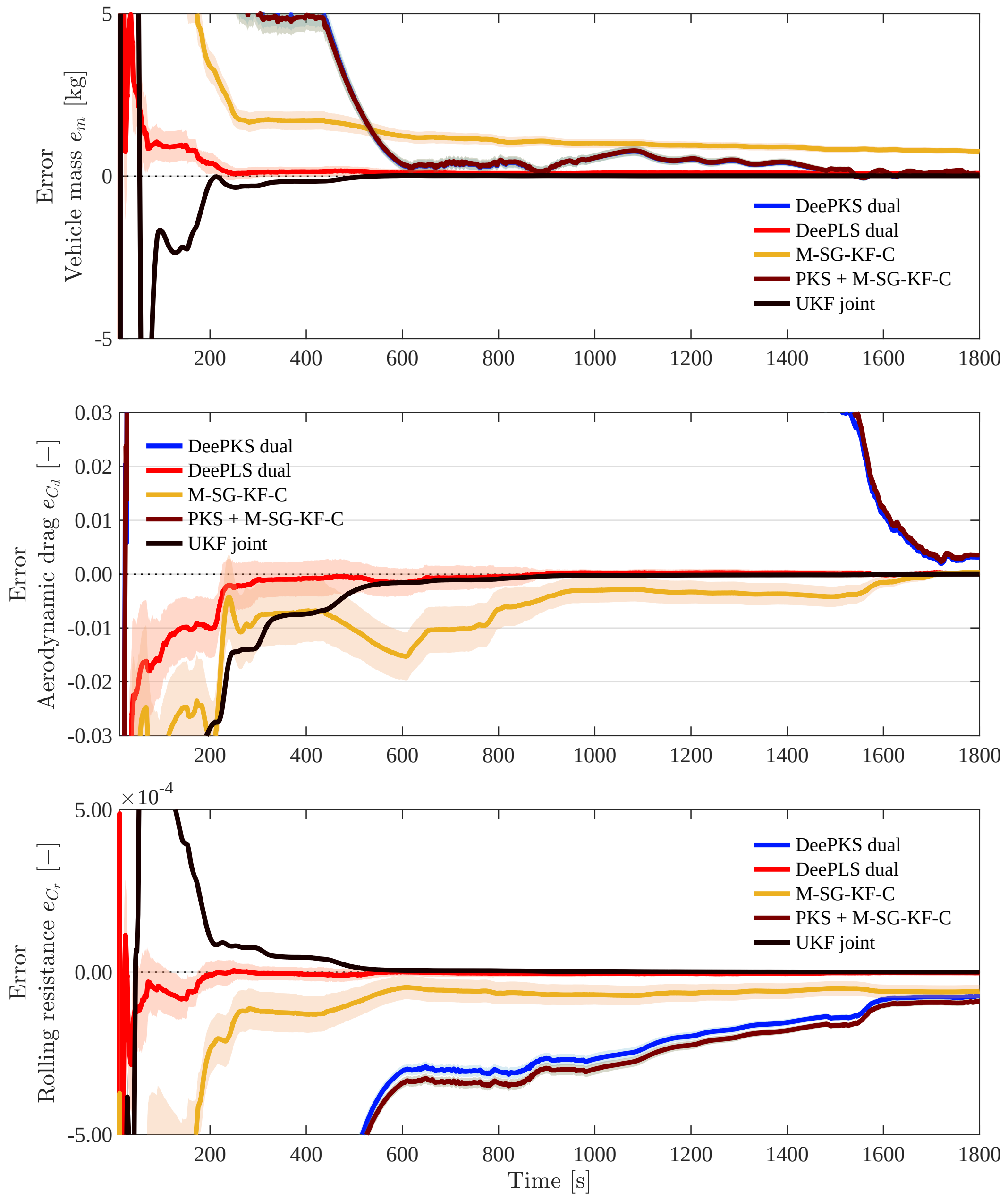

Figure 6.8: Zoomed version of Figure 6.7, with results of Experiment 6.4.

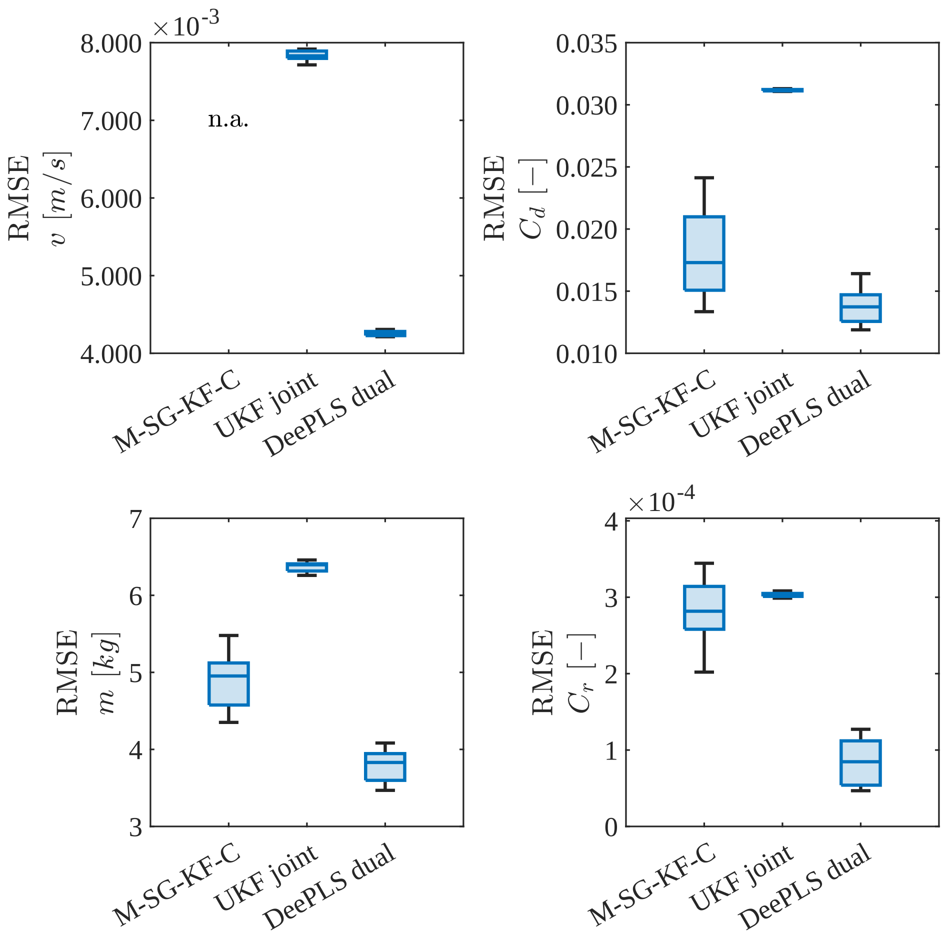

Figure 6.9: Visualization of the error statistics of Experiment 6.4. Box plot of RMSE values for errors of vehicle speed (top left), vehicle mass (bottom left), aerodynamic drag (top right), and rolling resistance (bottom right). For details, see Section 6.6.3.

7 Adaptive Nonlinear Model Predictive Control

"Utilizing a nonlinear approach and companion competencies makes it possible for people to move from good performance to great."

– John H. Zenger, *1931

In this chapter, a solution to the longitudinal vehicle motion tracking problem via Adaptive Nonlinear Model Predictive Control (ANMPC) is presented. The work introduced in this chapter builds on preliminary work that the author of this thesis published in Buechel and Knoll [BK16a] and Buechel and Knoll [BK16b], but includes substantial novel and previously unpublished material. Find an overview of the contributions within this chapter in Section 7.6.3.

7.1 Introduction

We then consider the problem of longitudinal vehicle motion tracking and develop a solution within the Model Predictive Control (MPC) framework. In the existing literature, lower-level control modules typically receive the desired acceleration at the *current point in time* as a control input from the higher-level module. We already discussed in Section 2.2 that when considering future automated vehicles, we assume that not only the reference point at the current point in time is available, but, in addition, a *reference motion trajectory* can be provided to the tracking unit by some higher-level control or motion planning module. By definition, this contains reference data with a certain *look-ahead* into the future. Additionally, *advance knowledge* regarding future disturbances arising from a time-varying road slope is assumed to be available. Contrary to most common implementations, we desire that our solution can exploit this information to calculate optimal control input trajectories while considering both input and state constraints. One very natural choice of control algorithms for such requirements is the one of Model Predictive Control (MPC), which can exploit knowledge about the system behavior utilizing a model-based approach based on the longitudinal vehicle motion dynamics.

This way, we aim to achieve optimal performance when tracking arbitrary reference trajectories by exploiting the nonlinear vehicle model. Nevertheless, we face a substantial difficulty if we aim to apply this approach in real vehicles: The model parameters, like vehicle mass and friction and drag coefficients, are known to be time-varying within a substantial parameter range (see Section 3.4). Under these circumstances, the tracking performance of a standard MPC controller will degrade, and offset-free tracking will not be possible without any additional ingredients.

Existing approaches propose to augment the system with a disturbance model to achieve asymptotic offset-free tracking. This disturbance model is typically implemented in a way such that all effects of the various unknown parameters are lumped together into a disturbance vector whose dimension is less than that of the unknown model parameters for various reasons. Unfortunately, this comes with an unavoidable degradation of the closed-loop behavior of the controller. Also, the knowledge of the true, physical system parameters remains hidden.

The reasons above are why we aim to explore a different approach: We use estimations of the true vehicle parameters, together with state estimations of measured but noisy state observations, and combine it with a nonlinear, model-predictive control scheme. The state- and parameter estimations will be calculated by the combined and dual state- and parameter observer, which is presented in Chapter 6. This enables the use of the true parameters also for other tasks, such as lateral control. As discussed in detail in Chapter 6, the observer presented therein can track the time-varying parameters under insufficient excitation, which occurs in phases of zero longitudinal vehicle acceleration.

In addition, we need to solve the control allocation problem while considering actuator-specific dynamics. The consideration of advance information, which is time-variant over the prediction horizon, differs from a standard approach in other existing MPC formulations. To incorporate all these requirements, a non-standard MPC formulation will be presented for nonlinear systems with actuator delays, which can achieve offset-free asymptotic tracking under certain assumptions.

Furthermore, we validate the proposed controller's feasibility, performance, and accuracy in longitudinal tracking tasks in various numerical experiments, demonstrating that the AN-MPC controller substantially outperforms a baseline controller. We will see that the proposed controller is able to keep the optimal performance of the nominal case, in which the model parameters are assumed to be known. Also, effects from actuator dynamics and lags and the nonlinear motion dynamics can be eliminated or at least reduced to a minimum.

The obtained control solution is easy to tune while offering a trade-off between tracking performance and reduced economic cost. It meets the requirements of a modular vehicle control architecture and can easily be extended to include additional operating modes. Also, optimal input *trajectories* are calculated instead of input set-values. This information is typically not exploited in lower-level control modules since it is unavailable from non-predictive control schemes or if a manual driver is in the loop. However, in a standard receding horizon scheme, only the first input at the current time step is used, and the rest of the information in the trajectories is discarded. Hence, it might be a substantial benefit for the overall vehicle performance if this additional information is kept and exploited, but further analysis in this direction exceeds the scope of this presentation.

This chapter is organized as follows: We first introduce some background on MPC in Section 7.2, followed by a selection of related work in Section 7.3. The latter includes an introduction to topic-related taxonomy. The proposed solution will then be presented in Section 7.4. Results based on simulation studies, which highlight the advantages of the approach, are given in Section 7.5. Section 7.6 finally gives a conclusion and outlook on future enhancements.

7.2 Model predictive control background

Model Predictive Control (MPC), also denoted Receding Horizon Control (RHC), is a control scheme that provides optimal control inputs by exploiting a system model. By planning several steps into the future and evaluating the cost of a given control trajectory, the optimal input sequence can be found by minimizing the cost of an Optimal Control Problem (OCP). This also allows incorporating advance information regarding known disturbances and a time-varying reference trajectory, while considering state and input constraints. In order to obtain a closed-loop controller, the controls are applied according to the *receding horizon principle*, which will be explained next.

7.2.1 Receding horizon principle

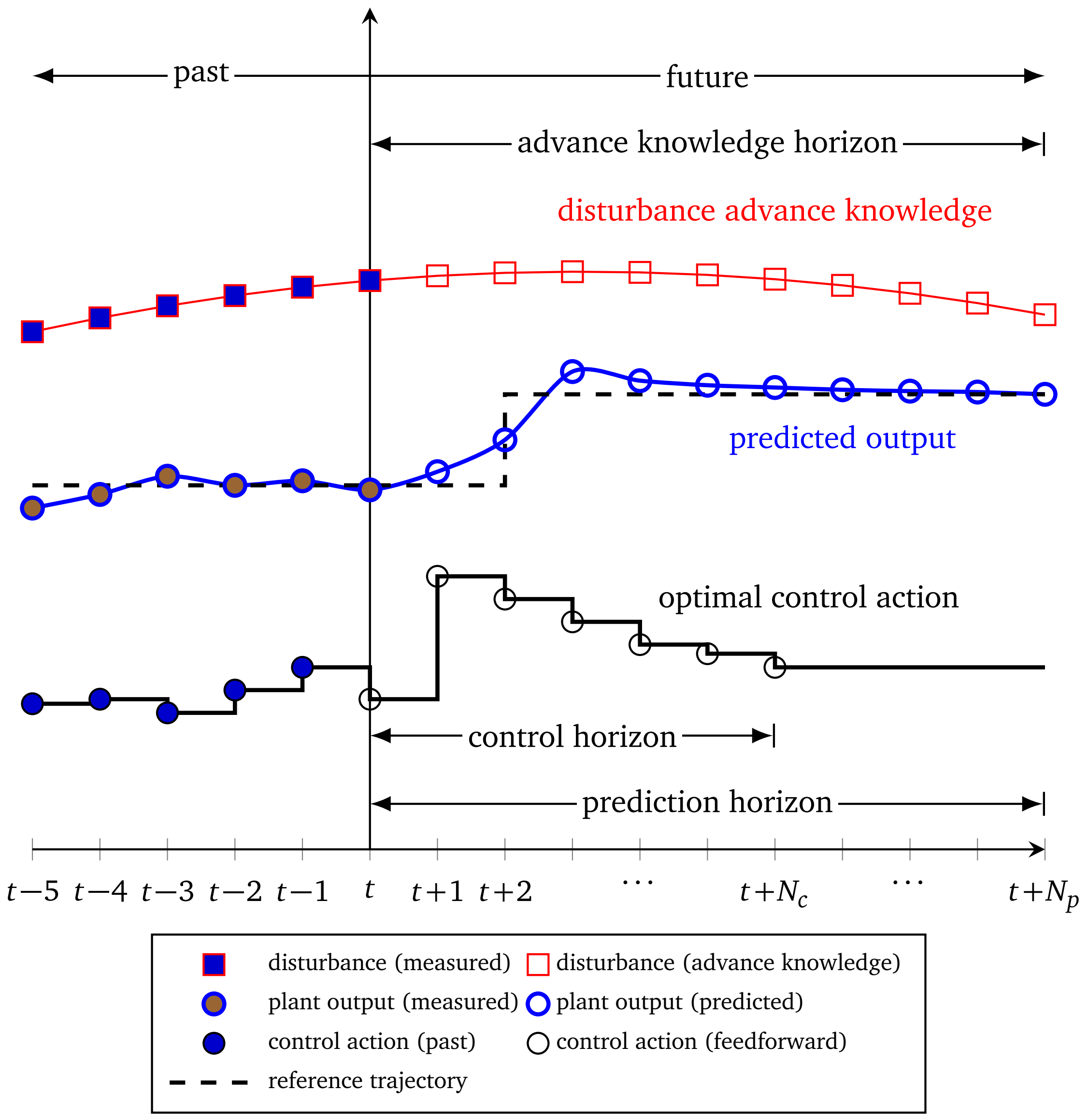
past
future
advance knowledge horizon
disturbance advance knowledge
predicted output
optimal control action
control horizon
prediction horizon
$t-5$ $t-4$ $t-3$ $t-2$ $t-1$ t $t+1$ $t+2$ $\cdots$ $t+N_c$ $\cdots$ $t+N_p$
disturbance (measured)
disturbance (advance knowledge)
plant output (measured)
plant output (predicted)
control action (past)
control action (feedforward)
reference trajectory

Figure 7.1: Receding horizon control scheme.

The solution of this Optimal Control Problem (OCP) could, in theory, be taken as an open loop control for the control horizon and fed to the plant. Due to inaccuracies in the model and unknown external or internal disturbances, the solution might diverge from the predicted one. In order to compensate for these errors, a feedback loop is introduced: Only the first control input of the solution to the OCP is applied, and then the algorithm repeats to find the optimal control sequence, this time based on the true new state rather than the predicted one. Since the prediction horizon, over which the optimal trajectories are found, remains constant but is moved forward at each time step, this is typically referred to as the *receding horizon principle*. An illustration of the receding horizon control principle is given in Figure 7.1.

7.2.2 Advantages of model predictive control

Several advantages arise from exploiting the knowledge of the plant behavior in MPC. Solving the OCP leads to optimal results for the nominal (error-free) system and, as long as the system model is accurate, also for the real plant. Here, optimal is with respect to a cost functional, defined over the prediction horizon.

Second, one of the most valuable properties of MPC for practical application is that both input and state constraints can be handled. Also, thanks to the predictive nature of the method, the incorporation of advance knowledge in the form of time-varying reference values and future disturbances is enabled. Note that this is not standard in common MPC formulations but will be exploited in our proposal.

Next, multi-variable control problems can be handled in this fashion without having to modify the control scheme. This includes problems in which the controls have a redundant effect on the plant. As such, MPC intrinsically provides a solution to the dynamic *control allocation problem*, under consideration of actuator dynamics. The control allocation problem was already discussed in Section 1.6. To summarize, we can list the following advantages arising from the application of MPC:

- *optimal solution* w.r.t. some given cost functional
- consideration of state and input *constraints*
- consideration of *advance knowledge*
- suited for *multi-variable* control problems
- directly provides a solution to the *control allocation problem* under consideration of actuator dynamics

7.2.3 Formal description of model predictive control

Next, we want to give a formal description of MPC and the inherent receding horizon principle. For the general, nonlinear, discrete-time system

$$x_{t+1} = f(x_t, u_t), \tag{7.1}$$

with $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$ and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \longrightarrow \mathbb{R}^{n_x \times n_u}$. We consider a (finite horizon, and hence truncated) cost function over the prediction horizon of N time steps:

$$J_N(x_0, U) \;=\; \sum_{k=0}^{N-1} l(x_k, u_k) + V_f(x_N), \tag{7.2}$$

with the input trajectory $U \doteq u_{t:t+N-1}$ and the stage cost $l(x,u)$. The terminal cost $V_f(x_N)$ aims to replace the value of the costs starting from $t+N$ to a desirably infinite horizon - since this is typically infeasible to compute for nonlinear problems. The aim is to find a solution at each time step $t = 0,1,2,\dots$ of the OCP, minimizing the cost according to the following expression (see for example [GP17]):

$$\begin{aligned} V_N(x_t) \doteq \min_U \quad & J_N(x_{k|t}, U) \\ \text{subject to} \quad & x_{k+1|t} = f(x_{k|t}, u_{k|t}) \quad \forall\, k \in \mathbb{N}_p, \\ & x_{0|t} = x_0, \\ & u_{k|t} \in \mathcal{U} \\ & x_{k|t} \in \mathcal{X} \quad\quad\quad\quad\quad\, \forall\, k \in \mathbb{N}_p \\ & x_{N|t} \in \mathcal{X}_f \end{aligned} \tag{7.3}$$

The control input and state constraint sets are denoted by $\mathcal{U}$ and $\mathcal{X}$, respectively. Generally, a terminal state constraint is necessary to achieve stability, achieved by the constraint set condition, which enforces the last state of the prediction horizon to lie within $\mathcal{X}_f$. Under certain assumptions, the terminal constraint can be omitted, as discussed in Section 7.3.6. Also, variations of the formulation given in (7.3) exist, as we will show in Section 7.3.8. The solution to the OCP (7.3) is the optimal input trajectory, denoted by U_t^*, with the first element $u_{0|t}^*(x_t)$. $u_{0|t}^*(x_t)$ is then applied as control input, which leads to the system evolving according to:

$$x_{t+1} = f(x_t, u_{0|t}^*(x_t)). \tag{7.4}$$

Repeatedly proceeding with this procedure in a receding horizon fashion results in a closed-loop system. In a particular case, the problem can be explicitly solved, and the resulting controller is known as the Linear Quadradic Regulator (LQR): If the system described by f is linear, this linear structure can be exploited. Additionally, the prediction horizon needs to be infinite, the reference trajectory needs to be constant, set to the origin, costs are quadratic, and no constraints are present.

7.3 Related work and taxonomy

We already listed previous work related to the application of longitudinal vehicle motion tracking control in Chapter 4. In this section, we want to highlight some important topics related to the background and theory of MPC.

It is out of scope to present a comprehensive survey on the broad literature about MPC. Instead, we refer to a variety of books and surveys available on the topic, like, for example, [HR08], Rawlings and Mayne [RM09], and Gruene and Pannek [GP17]. Mayne, Rawlings, Rao, and Scokaert [MR+00] published an excellent survey on constrained model predictive control. Nevertheless, this section aims to introduce some important concepts and terminology appearing within MPC literature, which helps to understand the proposal of this chapter. The presentation is done together with some references for further reading.

7.3.1 Brief history of MPC

MPC has its roots in the theory of optimal control, with foundations starting in the 1950-ties including the principle of dynamic programming Bellmann [Bel54] and the famous Pontryagin minimum principle Pontryagin, Boltyanskiĭ, and Mishchenko [PB+61]. Some authors believe it was further developed during the Apollo Lunar program in the 1960-ties [Rus13].

MPC was initially mostly found to be applied in process control. There, slow processes eased the application, which was more computationally intensive than other model-free control schemes, as, for example, PID control. It is often sufficient to achieve set-point regularization in process control, while constraint handling might be necessary.

First, results for stability were only available for linear time-invariant systems. Then, NMPC schemes were found to be stabilizing, using terminal constraints and terminal penalties. Later, conditions for stabilizing NMPC without terminal constraints and costs were established [Mac02; Ros03; GP11; GP17]. Solutions for *reference trajectory tracking* were only found later, especially for the nonlinear case [MM12].

7.3.2 Set-point regulation vs. trajectory tracking

The stage cost in MPC often is formulated as a quadratic cost function of the form:

$$\ell(x_k, u_k) = \|x_k - x_s\|_Q^2 + \|u_k - u_s\|_R^2, \tag{7.5}$$

where (x_s, u_s) is a control set-point. Depending on the control set-point used, control tasks can be divided into *set-point regulation* and *trajectory tracking*.

In *set-point regulation* problems, the aim is to design a controller that drives a system to a constant, time-invariant target set-point and which can stabilize it there. Since the origin can be arbitrarily defined, this set-point is usually set to the origin, hence $(x_s, u_s) = (0, 0)$ and $f(x_s, u_s) = 0$. The stage cost then typically has to penalize the distance of the current state to the origin, and the stage cost is reduced to:

$$\ell(x_k, u_k) = \|x_k\|_Q^2 + \|u_k\|_R^2. \tag{7.6}$$

Trajectory tracking MPC is the problem of tracking a time-variant set-point trajectory. Note that in this formulation, although x_s, u_s are in general time-variant, they are still considered as constants during the entire prediction horizon, so that the OCP is identical to solve than for the regulation problem. As discussed in [RV09], fewer applications consider the case of tracking time-varying references $(x_{s,k}, u_{s,k})$ which changes within the prediction horizon, since this increases the complexity for solving the OCP.

7.3.3 Advance knowledge for anticipative predictive control

If, in addition to tracking time-varying references, advance knowledge about future disturbances is considered, this further increases the parameter space. This is why current literature typically neglects such information, as discussed in Dughman and Rossiter [DR17]. The system dynamics then is described by

$$x_{t+1} = f(x_t, u_t, d_t), \tag{7.7}$$

and the cost functional is of the form $J_N(x_0, U, D)$, where $D \doteq d_{t:t+N_a-1}$ is the *known* disturbance or *advance knowledge* trajectory over the advance knowledge horizon N_a.

7.3.4 Adaptive MPC

Adaptive MPC seeks to improve system performance by updating the model online based on measurement data while calculating an optimal control policy for the nominal, adapted system.

Mayne, Rawlings, Rao, and Scokaert [MR+00] stated in the year 2000 that "although adaptation was one of the earliest motivations for model predictive control, a stabilizing adaptive model predictive controller for constrained systems has not yet been developed. A prior requirement for progress in this area, as in output-feedback model predictive control with state estimation is a good methodology for achieving robustness.".

[Joh11] gives a short introduction to NMPC in general, including adaptive NMPC. One of the main difficulties for obtaining general stability results for general nonlinear ANMPC is that, unlike with linear systems, the *separation principle* does not hold. The separation principle states that it is sufficient to prove stability separately for controller and observer to obtain overall stability when combining the two. Nevertheless, some results in this direction were presented. In 2009 Adetola, DeHaan, and Guay [AD+09] proposed a method for the adaptive model predictive control of constrained nonlinear systems and provided stability results. Unfortunately, it is restricted to nonlinear models of the form

$$\dot{x} = f(x,u) + g(x,u)\theta, \tag{7.8}$$

which is affine in the unknown parameter vector θ. Also, stability investigations did not include any presence of noise.

The more recent work of Vatankhah and Farrokhi [VF19] presents a framework for nonlinear adaptive model predictive control of constrained systems with offset-free tracking behavior using an adaptive neural network predictor. The control algorithm is restricted to single-input, single-output, and non-affine time-discrete systems with a NARX structure under disturbances. By nature, the method is not based on a first-principle model, and deriving results regarding the optimality of the approach is challenging to obtain. For more results on the stability of the combination of observers with NMPC, see Findeisen, Imsland, Allgöwer, and Foss [FI+03], Roset, Lazar, Heemels, and Nijmeijer; Messina, Tuna, and Teel [RL+06; MT+05] for an overview.

7.3.5 Other concepts in MPC

Robust MPC

"A controller is said to be robust when stability is maintained and performance specifications are met for a specified range of model variations and a class of noise signals" [BM99]. As with any robust control policy, robust MPC also aims to achieve robust performance and stability despite the presence of uncertainty. A drawback of robust MPC is that robust controllers generally do not adapt to changes in the plant, and "therefore their performance is limited by the quality of the model plus the uncertainty description initially available" [AD+09]. Nevertheless, robust *and* adaptive approaches exist. Until now, to the best of the author's knowledge, no general formulation for optimal adaptive and robust control for general nonlinear systems exists. Current approaches are typically restricted to a particular type of system and are also restricted in the way that they can only follow certain predefined, restricted dynamics, as, for example, in [PB+21].

While the combination of adaptive and robust control solutions is interesting, if not mandatory, for guarantees in automated driving, this thesis focuses solely on adaptive optimal control while taking measures to design the adaptation mechanism so that they remain robust, e.g., to measurement outliers. This will be discussed in Chapter 6, while we need to regard robust adaptive MPC as future work. Nevertheless, the reader shall be referred to [BM99] for an excellent survey about the origins of the field.

Economic NMPC

EMPC is an advance control scheme which is very promising. While traditional MPC relies on positive-definite cost functions, typically quadratic ones, to provide stability, economic MPC aims to relax this and introduce more generic cost functions. Rather than stabilizing the controlled system to a desired state or output, the focus is, e.g., on minimizing monetary costs, time, or energy - hence the name of the method, while keeping the system within given boundaries. Unfortunately, stability proofs for closed-loop performance can be complex in practice. Nevertheless, results exist and can be found in Ellis, Durand, and Christofides [ED+14] or Faulwasser, Grüne, and Müller [FG+18]. These include stability results for EMPC with and without terminal constraints and costs.

7.3.6 Stability of NMPC

All MPC applications rely on the solution of the OCP. This is why an essential condition when wanting to answer the question of whether a certain MPC controller is stabilizing is the question of *feasibility*.

Feasibility is only given if a solution to the OCP can be found at each time step. Additionally, this solution needs to be found in time in order to provide the control input for the next control step. More on feasibility can be found, e.g., in [GP17], p. 177. An important property is the one of *recursive feasibility*. Recursive feasibility is given if a system starts at a point contained in the set of feasible solutions, and the resulting state after applying the control (the first input from the solution of the OCP) still lies within the set of feasible solutions.

Since, for the case of set-point regulation, at any point in time, the target state is the same, establishing conditions for recursive feasibility is typically an easier task than for trajectory tracking MPC.

Looking at the timeline in the historical development of NMPC, according to [GP17], p. 115f, the first stability results using terminal constraints can be dated back to the 1980s. Different theoretical foundations exist for designing stabilizing nonlinear MPC; see, for example, Nicolao, Magni, and Scattolini [NM+98], Wan and Kothare [WK03], and Magni and Scattolini [MS06]. The first approaches to proof stability for NMPC used the concept of terminal constraints, while only later, schemes without terminal constraints were established. To the contrary, stabilizing NMPC without terminal constraints appeared much later, with results starting around 2001 [GP17; FG+18].

Stabilizing NMPC with Terminal Constraints

A sketch of the proof can be found, for example, in [FG+18], p.10. The idea of stabilizing NMPC with terminal constraints can be summarized as follows: If the infinite horizon problem was computationally tractable, one could establish proof that, for the set-point regulation problem, the system would be stabilized at infinite time. Since the horizon is truncated, we already introduced a terminal penalty for the remaining cost from the end of the horizon to infinity in (7.2). Instead of ensuring that the system arrives at the steady state origin at infinity, in a similar way, it is enough to ensure the system comes close enough to the origin. This is reflected in the terminal set $\mathcal{X}_f$. The informal proof can be given as follows: If the system arrives at a point close enough from the origin at the end of the prediction horizon, and additionally, a control law exists, which stabilizes any point within the terminal set to the origin, the MPC law is stabilizing. In other words, stability can be formally established by a sub-optimal but stabilizing control law of the form

$$u = k(x) \qquad \forall x \in \mathcal{X}_f. \tag{7.9}$$

Stability can then be proved for the case that V_N is a Lyapunov-Function. See [GP17], Chapter 5 for more detailed proof. As a remark, interestingly, the stability proof using terminal constraints includes a terminal control law $u = k(x)$, although this control law is actually never used to control the system.

Stabilizing NMPC without Terminal Constraints

This topic is treated, for example, in Faulwasser, Grüne, and Müller [FG+18] and Gruene and Pannek [GP17] or the references within. Not having to consider terminal constraints typically allows a significant speedup in the computation of the solution to the OCP. Additionally, adding terminal constraints might result in an (unnecessarily) small feasible set, depending on the prediction horizon. Various approaches have been proposed to guarantee the stability of NMPC without terminal constraints. Three main ideas are (taken from [FG+18] with modified references in brackets):

1. *Replace V_f by $V_f^\beta = \beta V_f$, with $\beta > 0$ sufficiently large, such that a suitable terminal constraint $\mathcal{X}_f$ is satisfied without being explicitly stated in the OCP, cf. [RM09].*
2. *Require that V_f is a global Control Lyapunov Function (CLF) [JH05].*
3. *Drop the terminal penalty ($V_f(x) = 0$), suppose specific bounds on the optimal value function, and require a sufficiently long prediction horizon [JY+01; GM+05; Gru09].*

Approach 3 means that under certain assumptions, it is possible to show that for a sufficiently long prediction horizon, NMPC is stabilizing without terminal constraints *and* costs.

7.3.7 Offset-free tracking and integral action

Offset is understood to be the steady-state tracking error to which a closed loop system converges when tracking an (asymptotically) constant reference value. Offset appears due to plant-model mismatches like un-modeled dynamics, uncertain parameters, and un-modeled disturbances.

Integral action is generally understood as a countermeasure to eliminate steady-state offset. Citing Ruscio [Rus13], we find that: “Standard MPC algorithms usually do not achieve integral action and there is one main reason for this. The answer is that integral action is not necessarily optimal.”

A recent survey about common methods for offset-free tracking is given by Jimoh, Kucukdemiral, Bevan, and Orukpe [JK+20], which were restricted to linear and time-discrete state space systems with deterministic disturbances. Several authors suggested achieving offset-free tracking by using a disturbance model together with an observer to estimate the disturbance state. Typically, the system dynamics is augmented with an estimated disturbance, which is modeled as constant (plus some additive white noise in some cases [MM12; JK+20]. Various formulations were presented, some of which have been later discovered to be equivalent [RR+09]. For more information, see the discussion in [PG+15].

An alternative to the approach mentioned above has been presented as *incremental* model-based MPC, also termed *velocity model approach* (see, e.g., [PR01], [PR03], [Rus13]). In this approach, the input change Δu_k instead of the input value u_k is computed by using the state change $\Delta x_k = x_k - x_{k-1}$ instead of the state value x_k. Augmenting the system with the offset value has been shown to enable offset-free tracking of constant references for Linear Time-Invariant (LTI) systems. See again the review [PG+15] and references therein. According to Werner [Wer21], existing velocity algorithms are all either limited to LTI systems or obtain a quasi-Linear Parameter-Varying (LPV) system by linearization.

Looking at nonlinear formulations, [VF19] stated only in 2019 that “although offset-free tracking in the MPC is not a new method, works related to the NMPC are limited”. Nevertheless, Morari and Maeder [MM12] is one example in which they extended the concepts of linear offset-free MPC to nonlinear systems.

All examined results had in common that they were designed to track a set point that remained constant throughout the entire prediction horizon, while most formulations were required to also calculate a target value. The latter can be seen as the steady-state value to which the system would settle under consideration of the constant, estimated disturbance ([PG+15], [MM12]).

We want to look closer at the approach for offset-free tracking of nonlinear systems given by Morari and Maeder [MM12]. Here, the nonlinear model is augmented with a disturbance model of the form (see [MM12] (8), with only slightly adapted notation):

$$\begin{aligned} x_{k+1} &= f_{\text{aug}}(x_k, d_k, u_k) \\ d_{k+1} &= d_k \\ y_k &= h_{\text{aug}}(x_k, d_k)\,. \end{aligned} \tag{7.10}$$

Note that in this formulation, d_k is a general *unknown* disturbance vector, contrary to the use in the rest of this thesis, where we used d to describe the known disturbance input. An observer of the form

$$\begin{aligned} \hat{x}_{k+1} &= f_{\text{aug}}(\hat{x}_k, \hat{d}_k, u_k) + \ell_x\left(y_{\phi,k} - h_{\text{aug}}(\hat{x}_k, \hat{d}_k)\right) \\ \hat{d}_{k+1} &= \hat{d}_k + \ell_d\left(y_{\phi,k} - h_{\text{aug}}(\hat{x}_k, \hat{d}_k)\right), \end{aligned} \tag{7.11}$$

was discussed “for clarity and due to its practical importance”. However, they also mentioned that “the arguments in this paper extend easily to other observers such as the Extended Kalman Filter.” In above equation, the measured output is denoted by $y_{\phi,k}$. It is important to mention that the results of this paper are derived under the assumption that both an asymptotically constant reference as well as disturbance is present, i.e.

$$\begin{aligned} r_k &\to r_\infty \\ d_{\phi,k} &\to d_{\phi,\infty}\,. \end{aligned} \tag{7.12}$$

An important ingredient for the output to asymptotically reach the target, meaning that $y_k \to r_\infty$, is that a *target* $(\bar{x}, \bar{u})$ is added in the cost function, which is defined as a solution to the equilibrium problem ([MM12], (10), again with adapted notation):

$$\begin{aligned} \bar{x} &= f_{\text{aug}}(\bar{x}, \hat{d}_k, \bar{u}) \\ r_k &= h_{\text{aug}}(\bar{x}, \hat{d}_k). \end{aligned} \tag{7.13}$$

Then, solving the following OCP in a receding horizon fashion was proven to lead to asymptotically offset-free tracking of an asymptotically *constant* reference *and* disturbance:

$$\begin{aligned} \min_{\substack{\bar{x},\bar{u} \\ u_0,\ldots,u_{N-1}}} \quad & F(x_N - \bar{x}) + \sum_{k=0}^{N-1} \ell(x_k - \bar{x}, u_k - \bar{u}) \\ \text{s.t.} \quad & x_0 = \hat{x}_t, \quad d_0 = \hat{d}_t, \\ & \bar{x} = f_{\text{aug}}(\bar{x}, d_0, \bar{u}), \\ & r_k = h_{\text{aug}}(\bar{x}, d_0), \\ & \bar{x} \in X, \quad \bar{u} \in U, \\ & x_{k+1} = f_{\text{aug}}(x_k, d_0, u_k), \quad k = 1, \ldots, N-1, \\ & x_{k+1} \in X, \quad u_k \in U, \quad k = 0, \ldots, N-1, \end{aligned} \tag{7.14}$$

with $\ell(0,0)=0$ and $F(0)=0$. Note that in this formulation, the solution to the equilibrium problem $\bar{x},\bar{u}$ is part of the OCP. Alternatively, instead of directly integrating the equilibrium problem into the optimization, it is mentioned that it is possible to solve this in a separate optimization problem. For details about the proof, the reader is referred to [MM12]. Here, we only want to mention, that it involves that the disturbance estimator is able to estimate the true plant output $y_{\phi,\infty}$ asymptotically error-free. For the linear case, it is then enough that the model has to be observable, which can be proven easily by well-known observability conditions. In the nonlinear case, some additional requirements have been presented. This includes that the observer is designed to be nominally error-free at steady-state, which means that no stationary solutions with nonzero estimation error exist (see [MM12], Assumption 4 and (30).)

Two main approaches have been discussed in [MM12] regarding the disturbance models. The first is the *pure output disturbance model*, with one disturbance state for every output variable:

$$\begin{aligned} x_{k+1} &= f(x_k,u_k) \\ d_{k+1} &= d_k \\ y_k &= h(x_k)+d_k, \end{aligned} \tag{7.15}$$

which implies that $\dim y = \dim d$. The second, and also rather simple method, is the pure input disturbance model. This was mentioned to often lead to better closed-loop performance while not suffering from observability issues when dealing with plants that contain integrators:

$$\begin{aligned} x_{k+1} &= f(x_k,u_k+d_k) \\ d_{k+1} &= d_k \\ y_k &= h(x_k), \end{aligned} \tag{7.16}$$

where they assumed that u, d, and y all to have the same dimension. They also mention that from the viewpoint of achieving offset-free tracking of asymptotically constant references, it is sufficient to choose d such that its dimension is not higher than the one of the output and that doing so would increase the complexity for the observer.

7.3.8 Solution strategies to the optimal control problem

Providing a solution to the OCP at each time step typically is a computationally expensive task, which makes it challenging to achieve the small sample times needed for fast and accurate control. While a vast increase in computational power could be observed over the last decades, it is still a limiting factor, especially for real-time embedded platforms found in automotive platforms.

That is why efficient solution strategies are required. In the following, we will discuss a few existing methods which have been proposed to solve the control problem more efficiently. Again, an in-depth introduction to this broad field would exceed the scope of this thesis, and the reader is referred to excellent introductions given, for example, by [Die16]. In general, three main approaches exist to solve the optimal control, which are (1) dynamic programming, (2) indirect approaches, and (3) direct approaches. The following summarizes the explanations given in Diehl, Bock, et al. [DB+06]:

Dynamic Programming uses Bellmann's principle of optimally [Bel57] to recursively compute a feedback control. For the continuous time case, this leads to the Hamilton-Jacobi-Bellmann equation, which is a Partial Differential Equation (PDE) in state space. This method usually suffers from Bellmann's "curse of dimensionality" and is therefore restricted to small-dimensional problems.

Indirect Methods use Pontryagin's maximimum principle [PB+61], which is a necessary condition. A Boundary Value Problem (BVP) can be derived in ODEs, which can be solved numerically. The approach follows a principle termed as "first optimize, then discretize". "The numerical solution of the BVP is mostly performed by shooting techniques or by collocation. The two major drawbacks are that the underlying differential equations are often difficult to solve due to strong nonlinearity and instability and that changes in the control structure, i.e., the sequence of arcs where different constraints are active, are difficult to handle: they usually require a completely new problem setup. Moreover, higher index DAEs arise on so-called singular arcs, which necessitate specialized solution techniques" [DB+06].

Direct Methods attack the original, infinite-dimensional problem by transforming it into a finite-dimensional Nonlinear Program (NLP). This paradigm has been termed as "first discretize, then optimize". One of the most important advantages of direct methods has been found to be that the approach facilitates consideration of inequality constraints [DB+06]. The original, continuous problem is typically approximated by a piece-wise constant parameterization of the controls to reduce complexity. This assumption aligns with the practice of realizing the controller on an embedded platform with a given sample time. The various direct methods might substantially differ in the way the state trajectory is handled. This leads to different strategies about how to transcribe the original problem into a nonlinear optimization problem of the general form:

$$\min J(x) \tag{7.17a}$$

$$g(x) = 0 \tag{7.17b}$$

$$h(x) \leq 0. \tag{7.17c}$$

Examples of such *transcription methods* are *direct single-shooting*, *direct multiple-shooting* as well as *direct collocation* methods. These transcription methods lead to a different size of the nonlinear optimization problem above. This results counter-intuitively to a different complexity since higher-dimensional problems can be much easier to solve.

Before shortly giving some essentials on the different methods, we want to look into how the resulting NLP can be solved. A solution is typically found by the method of Sequential Quadratic Programming (SQP) - an iterative method, solving local approximation of the original problem by quadratic programming. To do so, equality constraints are considered by formulation of a dual problem, which, informally, includes the "distance" to the equality constraint. The dual problem then becomes unconstrained but also has to minimize this "distance", also called *dual variables* or *Langrange multipliers*, named after the inventor of the method. Inequality constraints are more challenging to consider, as they can be active or non-active at a certain point, while the transitions at the boundaries are not continuous. By a generalization of the method of Lagrange multipliers, developed by Karush-Kuhn-Tucker (KKT), necessary first-order conditions, termed as "KKT conditions", can be derived. Details can be found in [Die16]. Again, a variety of iterative methods exists:

In the *active set method*, active and inactive constraints at the current points are identified to solve a subset of the problem, which only considers the active set. In this sub-problem, they are treated as equality constraints, which facilitates the calculation of the resulting quadratic sub-problem. Iteratively checking for all constraints and adding them, whenever necessary, to the active set eventually leads to a feasible solution to the problem. The method allows to warm-start the algorithm, which can be exploited when the quadratic program has to be solved as part of a sequential quadratic programming problem. However, the computational costs increase exponentially with the problem size [Won11].

The *interior-point method* is an alternative to the active set method for solving the underlying quadratic programs, which might face difficulties with the non-smoothness arising from the inequality conditions. The resulting conditions are replaced by smooth approximations, typically logarithmic barrier functions. This method has been found to be especially interesting for large problems with more than approximately 15 states ([Gon11]).

Single shooting

Single shooting is a sequential approach to solving the optimal control problem. Here, only the initial state x_0 and the control vector $U = u_{0:N-1}$ act as variables, while the other states are recursively eliminated by the system dynamics equation. The NLP formulated in this way has comparably few dimensions but might be challenging to solve due to a high degree of nonlinearity arising from the recursive forward propagation of the state.

Direct multiple shooting

A different approach is taken in *direct multiple shooting* [Boc84]. A higher-dimensional, nonlinear program is obtained by partitioning the prediction horizon in discrete intervals and treating each of the sub-intervals as boundary value problems. This means that not only the control vector U together with the initial state x_0 build the free variables, but the entire state trajectory $X = x_{0:N}$ is included as free variables at the so-called shooting nodes. Additional equality constraints have to be added to ensure the continuity of the trajectory. Informally stated, the end-states of each sub-interval have to match the initial state of their subsequent interval.

This leads to Quadratic Programs (QPs) as sub-problems which are typically “a very sparse linear system and can be solved at a cost linear with N.” ([Die16], p. 133.). Suppose the original problem is that of receding horizon control. In that case, the algorithm can be very effectively warm-started since the (time-shifted) state predictions of the last control should be very close to the ones of the optimal solution for the current problem.

Collocation methods

In direct collocation methods, the entire state trajectory is parameterized as piece-wise low-order polynomials, which are then included as decision variables in the NLP. This leads to even higher-dimensional but very sparse problems.

Real-time iteration

Real-time iteration [DS03] tackles a problem termed as *online dilemma*: Finding the solution to the OCP cannot be processed in zero time. While time passes by, until a solution is found, the system state moves from its current state (for which the solution is calculated) to a new state. This might lead to sub-optimal controls or in the worst case, even loss of stability. To further reduce the delay inherent with the computation time needed to solve the sequential quadratic program, in principle, “only one optimization iteration is performed per sampling instant, and the obtained estimate for the optimal solution is shifted suitably to allow overall fast convergence” [DS03]. Hence, only an approximation of the optimal control trajectory is found, but at a reduced computation time. In addition, the computation time of each cycle is divided into a short feedback phase and a more prolonged preparation phase. The preparation phase can be processed in the previous cycle before measurement feedback is available. This reduces the delay within each cycle until the control is available.

7.3.9 Available software tools

Various tools exist, partly as open-source software packages, which provide solutions to the numerical problems involved with solving OCPs. In the preliminary studies for this work [BK16a; BK16b], the nonlinear optimization routine *fmincon* was used, which is available within the software package MATLAB® [Mat16]. Another library, which was considered for the implementation of the controllers within this thesis, was ACADO Toolkit [HF+11]. It provides very efficient solver implementations based on code generation. Some efforts were made to use the latter in the present thesis, but it was incompatible with up-to-date software libraries. This was why the implementation was eventually done in another open-source tool named "CasADi". CasADi [AG+19] is, in its current form, a general-purpose tool for gradient-based numerical optimization with a strong focus on optimal control. At its core, it supports algorithmic differentiation (AD) using a syntax borrowed from computer algebra systems (CAS) and has interfaces to common languages like C++, Python, and MATLAB®/Octave.

It supports ODE/DAE integration and sensitivity analysis, nonlinear programming, and interfaces to other numerical tools, like QP solvers and NLP solvers. These are, for example, the SUNDIALS suite [HB+05] with its integrators CVodes and IDAS, or the open-source primal-dual interior-point solver Ipopt [WB06] for NLPs and qpOASES for QPs, along with some commercially distributed packages. It also supports the generation of C-code but for a limited selection of solvers, which themselves support this.

Another open-source library is MPCTools [RR16]. It provides interfaces to CasADi solvers for all components of MPC: the estimator, the target calculator, and the regulator.

Very recently, at the time of writing, the software library *acados* [VF+21] was released, which can be seen as the successor of the library ACADO Toolkit. It is "a collection of solvers for fast embedded optimization intended for fast embedded applications. Its interfaces to higher-level languages make it useful for quickly designing an optimization-based control algorithm by putting together different algorithmic components that can be readily connected and interchanged . . . The main features of acados are: efficient optimal control algorithms targeting embedded devices implemented in C ..., user-friendly interfaces to MATLAB® and Python, and compatibility with the modeling language of CasADi." Unfortunately, it is not yet fully available at the time when the implementation of this thesis work started; this could be worth a try for the authors' new academic projects, while at the time of reading, this information might be outdated already.

Another example is the commercially available solver Forces Pro [ZD+17a], for which temporary licenses seem to be available for free for academic purposes upon request. Both frameworks could be used to generate efficient code suited for running on embedded systems. They both have been reported to be up to a magnitude faster [ZD+17a] than Ipopt [WB06], which we used for our experiments within the CasADi framework.

7.3.10 The control allocation problem and solution approaches

We already gave a general and informal definition of the term *control allocation problem* in Section 1.6, and learned that we have to solve this problem whenever we have to distribute a desired total control effort among a set of redundant actuators. A more detailed discussion of this topic can be found, for example, in the dissertation of [Kir18], which is solely dedicated to this topic.

More formally, for a controller to be uniquely defined, one has to be able to find a unique mapping from a desired system output to the control input vector that drives the system. This means that if we consider the desired system output

$$y^d = f_y(x, u, t), \tag{7.18}$$

and the control problem consists of finding a control input that fulfills:

$$u = f_y^{-1}(x, y^d, t). \tag{7.19}$$

If the inverse mapping given above is not unique, we have an *over-actuated* or redundant system, and the inversion is, without considering further criteria, an ill-defined problem. Performing this non-unique inversion is called solving the *control allocation problem*. In its most simplistic form, it might be that the system can be rewritten in the form

$$y^d = f_y(x, \delta, t) \tag{7.20}$$

$$\delta = Bu, \tag{7.21}$$

where the effective, virtual control input δ to the system is given as a static, linear mapping Bu. Now, even if the mapping f_y is invertible, but not the matrix B, meaning that B^{-1} does not exist, no unique inversion exists. In other words, inputs are redundant whenever the input matrix $B \in \mathbb{R}^{n\times m}$ does not have full column rank, and so perturbations of u in null-space directions do not affect the systems dynamic, as discussed in Section A.3. Various methods have been proposed as solutions to the control allocation problem. These can be mainly categorized into *static* and *dynamic* control allocation methods.

One simple solution, which belongs to the group of static methods, would be to use the pseudo-inverse to calculate

$$u = B^{\dagger}\delta, \tag{7.22}$$

or the *weighted pseudo-inverse* in case one wants to enforce favored actuator positions (see Section A.3.2). A limitation is that these methods fail in case actuator constraints are present (see [Kir18], p. 14). Other engineering targets, such as energy minimization, pose additional difficulties.

An extension, which is only feasible in case a fixed hierarchy of actuator utilization can be specified, is *daisy chaining*. Here, redundant controls are applied, starting with controls that were given a high priority, and the lower-priority controls are only used in case the control target cannot be reached due to higher controls being saturated. No guarantee can be given that a feasible solution is found (see [Kir18], p. 16). Another iterative process taken into account is the method of *Redistributed Pseudo-Inverse*. Here, an initial solution is obtained by the method of the weighted pseudo-inverse. In case this solution led to constraint violations, the constraint-violating controls are set to their saturated values, and a new solution is calculated from this subset. The process continues until a feasible solution is found. Other possible methods include *direct allocation* (see [Kir18], p. 17) and *numerical optimization methods*. The latter has the advantage of the possibility of including additional cost terms, in order to favor, for example, solutions with reduced actuation energy while coming at increased computational costs.

Looking at *dynamic allocation* approaches, additional solutions exist for linear plants, including methods using integral action in the control allocation law as well as LQR laws (see [Kir18], 2.1.7 and 2.2, respectively).

Using MPC to solve the control allocation problem is known to be a very versatile method. It allows to *include actuator dynamics and constraints of each actor individually* while offering the possibility to formulate additional targets for selecting favored solutions, for example, reducing actuator energy. MPC also allows the direct integration of the control allocation problem into a control task. However, this comes at the expense of increased complexity regarding computational power and implementation effort.

An example of the application of model predictive control solely to solve the control allocation problem in heavy-duty vehicles is given in Sinigaglia, Tagesson, Falcone, and Jacobson [ST+16].

Hence, it is straightforward to solve the control allocation problem directly within a model predictive control scheme, not only dedicated to splitting the control effort to the redundant actuators but also to solve a higher-level control target. In our case, as we will see in the remainder of this section, this higher-level purpose will be longitudinal motion tracking.

7.4 Proposed solution

In this section, a proposal for a novel formulation of NMPC is presented, which is especially suited as a longitudinal motion tracking algorithm for automated road vehicles. We first present the proposed control architecture of AVs, in which the tracking controller is embedded. This is done in Section 7.4.1. Then, we give a formal description of the control problem in Section 7.4.2. Next, the proposed formulation of the adaptive, nonlinear model predictive control (ANMPC) strategy is presented in Section 7.4.3.

7.4.1 Proposed control architecture

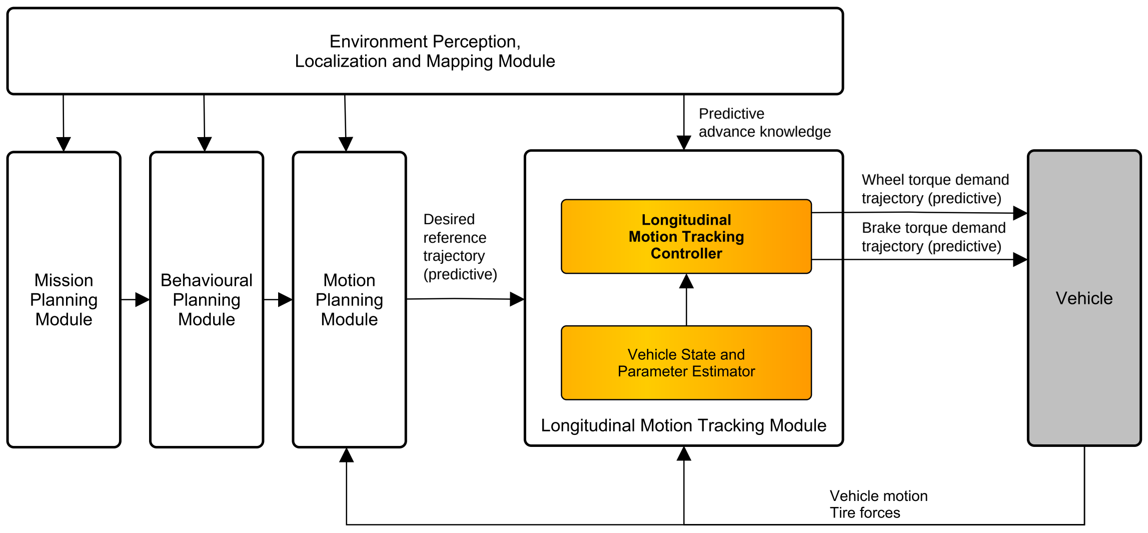
Environment Perception,
Localization and Mapping Module
Predictive
advance knowledge
Mission Planning Module
Behavioural Planning Module
Motion Planning Module
Desired reference trajectory (predictive)
Longitudinal Motion Tracking Controller
Vehicle State and Parameter Estimator
Longitudinal Motion Tracking Module
Wheel torque demand trajectory (predictive)
Brake torque demand trajectory (predictive)
Vehicle
Vehicle motion
Tire forces

Figure 7.2: Block diagram of the proposed longitudinal vehicle control architecture including the adaptive, model predictive control motion tracking module.

Figure 7.2 depicts the longitudinal motion tracking module as proposed, together with it's embedding within a general, control oriented system architecture of an automated vehicle. The proposal regarding the control architecture has already been presented within the author's publication [BK16a].

Let us look at Figure 7.2 in more detail. The longitudinal vehicle motion tracking module receives a time-varying reference trajectory from a motion planning control unit. This motion planning unit could itself be a model predictive controller, solving all sorts of driving scenario related control problems, like, for example the CACC problem. Alternatively, it could be a planning module in the sense that it computes trajectories in a feed-forward manner.

The perception module provides predictive advance knowledge about future road grade changes. They might come from map information, from sensory data (e.g., Lidar), or, even more likely, fused from different sources.

The output of the predictive longitudinal motion tracking module is two trajectories, which serve as an input to the power-train control module and the brake control module. By definition, these trajectories also contain predictive information that the just-mentioned lower-level modules can exploit.

In order to operate in a closed loop, the longitudinal motion tracking module also receives sensory feedback from the vehicle, which are noisy measurement readings of vehicle speed and acceleration, as well as (estimations) of the true tire forces resulting from the engine (or power-train) and the brakes.

Within the motion tracking controller, a *vehicle state and parameter estimator* acts as a data fusion module to provide filtered sensor information about vehicle speed and acceleration and estimations about the relevant vehicle parameter estimates. These are estimations of vehicle mass and the coefficients for rolling resistance and a (lumped) air resistance. The vehicle state and parameter estimator output is then leveraged in the vehicle dynamics model within the model predictive controller. Details about the estimator module can be found in Chapter 6, which can be seen as substantial extensions of the work presented in [BK16a].

7.4.2 Problem formulation

Plant model description

In this subsection, we want to present a nonlinear, continuous-time state-space form of the equations describing the longitudinal vehicle dynamics. In Section 3.5.2, we describe simplified equations for the longitudinal vehicle motion dynamics model in the form of a continuous-time system of ODEs in (3.66). For control purposes, we want to rewrite this in state space form while we transform it, solely for the purpose of giving some structure, into a system of the following form:

$$\dot{x}(t) = f\left(x(t), \theta(t)\right) + g\left(d(t), \theta(t)\right) + B \cdot u(t) \tag{7.23a}$$

$$y(t) = Hx(t). \tag{7.23b}$$

If we define the elements of the input vector $u(t) = [u_1, u_2]^{\mathrm{T}}$ as the demand values to power-train $u_1 = T_{\mathrm{we}}^d$ and brake $u_2 = T_{\mathrm{br}}^d$, the road slope angle as known disturbance $d = \varphi$, and the state vector x and the parameter vector θ as

$$x(t) = \begin{pmatrix} v(t) \\ T_{\mathrm{we}}(t) \\ T_{\mathrm{br}}(t) \end{pmatrix}, \quad \theta(t) = \begin{pmatrix} m(t) \\ C_{\mathrm{d}}(t) \\ C_{\mathrm{r}}(t) \end{pmatrix}, \tag{7.23c}$$

this will lead to the functions f and g and the matrices B and H given as:

$$f(x,\theta) = \begin{pmatrix} \dfrac{-\theta_2}{\theta_1 + m_\mathrm{I}} \cdot x_1^2 + \dfrac{x_2 - x_3}{\left(\theta_1 + m_\mathrm{I}\right) r_\mathrm{eff}} \\ \dfrac{-1}{\tau_\mathrm{pwt}} x_2 \\ \dfrac{-1}{\tau_\mathrm{br}} x_3 \end{pmatrix} \tag{7.23d}$$

$$g(d,\theta) = \begin{pmatrix} \dfrac{-\theta_1 g}{\theta_1 + m_\mathrm{I}} \left(\sin d + \theta_3 \cos d\right) \\ 0 \\ 0 \end{pmatrix} \tag{7.23e}$$

$$B \cdot u = \begin{pmatrix} 0 \\ \frac{1}{\tau_\mathrm{pwt}} \cdot u_1 \\ \frac{1}{\tau_\mathrm{br}} \cdot u_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ \frac{1}{\tau_\mathrm{pwt}} & 0 \\ 0 & \frac{1}{\tau_\mathrm{br}} \end{pmatrix} \cdot u \tag{7.23f}$$

$$H = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \tag{7.23g}$$

where we omitted time dependency this time for readability reasons. Remember that we presented this system already before in (3.74a) as semi-explicit DAE, where we performed a partitioning into system relevant states x^s and actuator states x^u and wrote:

$$\dot{x}^s = f^s(x^s, z) \tag{7.24a}$$

$$\dot{x}^u = A^u x^u + B^u u \tag{7.24b}$$

$$0 = g_\mathrm{DAE}\left(x^s, x^u, z, d, \hat{\theta}\right) \tag{7.24c}$$

$$y = h(x^s, z), \tag{7.24d}$$

where we had linear and stable actuator dynamics. We will use this representation to derive the proposed method for the target calculation within the proposed offset-free MPC approach.

Assumptions

The formulation of (7.23) was already done under several assumptions, which we want to summarize here. Further, we want to discuss additional assumptions which will be taken solely for control purposes and allows to formulate an OCP which can be efficiently solved in a model predictive scheme.

Consideration of unknown and time-varying parameters

In reality, most parameters that appear in (7.23), except the gravitational constant g, are time-varying. These are the effective tire radius r_eff, the time constants of power-train and brake τ_pwt and τ_br, respectively, as well as the parameters contained in the parameter vector θ. The mass equivalent resulting from the tire inertia m_I can be regarded as constant. To formulate the optimal control problem over a limited time horizon, we will assume that all the parameters mentioned, including θ, will not change substantially during the limited prediction horizon and treat them as time-invariant over this horizon.

Due to dynamic forces acting on the wheels, the effective wheel radius r_eff will increase in dependency on the rotational speed. We assume it would be possible to obtain the time-variant value using an approach similar to [CG05], which could then be used during each cycle. Nevertheless, for simulation purposes, we neglect entirely this influence and assume it is time-invariant.

The vehicle mass m is time-variant but generally only changes slowly while the vehicle is moving (e.g., due to fuel consumption). Nevertheless, it might happen that a vehicle loses load or that passengers jump onto the rolling vehicle. We assume that the parameter estimation module proposed in Chapter 6 can provide a meaningful estimation, which we will use in the model predictive controller, but treat the parameter value as time-invariant during the entire prediction horizon. The same applies to the other two parameters of the parameter vector θ, the aerodynamic drag coefficient C_d, and the rolling resistance C_r, which was the motivation behind this notation. Hence, estimations of the full parameter vector $\hat{\theta}$ are assumed available from the estimator.

The actuator delay coefficients of the power-train and brake, τ_e and τ_br, are also generally assumed to be constant. While these values might vary in reality, we assume that an empirically derived worst-case value is used in the tracking module. This clearly will lead to somewhat restrictive behavior, and investigations about how this can be avoided are left to future work.

Also, we assume to have knowledge about the maximal wheel torque values, which can be provided by the power-train and the brakes, and assume these values to be constant. In practice, again, these values will vary, depending on the operating point of an engine, for example, or the wear and tear of brakes. The same applies to the value of the minimum negative wheel torque value induced by the engine due to engine drag effects. All these values act as actuator limits within the tracking control module. For simplicity, we assume these values to be known and constant.

Consideration of advance knowledge

We assume that at each time t at which a control output should be calculated, for each discrete sampling point k in time, defined over a prediction horizon of length N_p into the future, a vehicle speed reference trajectory

$$Y_t^d = \begin{pmatrix} v_{t|t}^d & v_{t+1|t}^d & \cdots & v_{t+N_p|t}^d \end{pmatrix}, \tag{7.25}$$

as well as a vehicle acceleration trajectory:

$$Z_t^d = \begin{pmatrix} a_{t|t}^d & a_{t+1|t}^d & \cdots & a_{t+N_p|t}^d \end{pmatrix} \tag{7.26}$$

is available. In general, we do not pose any restrictions on the shape of these trajectories, except that they should lie within the feasible sets $v_k^d \in \mathcal{Y}$, $a_k^d \in \mathcal{Z}$ for the state and the algebraic state, respectively, and the acceleration values match the speed trajectory except by some round-off errors. In addition to many existing MPC formulations, we do not only consider reference values as time-varying over the prediction horizon but also further assume that a vector containing future road grade values, which can be seen as known disturbances D

$$D_t = \begin{pmatrix} \varphi_{t|t} & \varphi_{t+1|t} & \cdots & \varphi_{t+N_a|t} \end{pmatrix}, \tag{7.27}$$

given over the entire advance knowledge horizon, which we consider in this presentation to be of the same length as the prediction horizon $N_a = N_p$. We neglect the dependence of φ from the distance the vehicle travels and assume the values are obtained from a prediction of the given reference trajectory. Under the assumption that the true vehicle trajectory will not deviate very far from the reference, we will neglect the errors arising from this simplification. Nevertheless, the proposed framework could also be enhanced to include the distance traveled by the vehicle as another state, which would allow to correctly consider road grade information parameterized by distance rather than by time. Currently, this is left to future work.

As a note on the feasibility of the tracking control problem, we want to mention the following. If we require the controller to track a given trajectory without any tracking error at all times, restrictions have to be stated concerning the shape of the reference trajectory. Error-free tracking would be possible only under the assumption that v^d is Lipschitz continuous, with Lipschitz constant smaller than a certain value. This limit depends on the current road slope and all the vehicle parameters, as seen from (3.61), but also from the friction limits. Additionally, due to actuator delays, further restrictions regarding the second derivative of the reference trajectory would be necessary. We do not require that these restrictions be fulfilled by the reference trajectory if we allow the controller to deviate temporarily from the reference. For the work done in this thesis, we want to restrict ourselves only in a way that we assume the final value of the given reference trajectory is feasible, such that asymptotically offset-free tracking will be possible.

Objective

We want to find the control inputs u_1 and u_2 at each discrete time step over the control horizon in order to optimally track the vehicle speed reference value. In contrast, a trade-off between minimizing the tracking error, reducing the control input energy and achieving a comfortable vehicle motion should be possible by tuning. As mentioned, the controller should be capable of dealing with the unknown time-variant vehicle parameters vehicle mass, aerodynamic drag, and rolling resistance coefficient. Since the control matrix B in (7.23) is not square and full-rank, the above objective means that in addition, the controller has to solve the control allocation problem, as described in Section 7.3.10.

7.4.3 Proposed model predictive control formulation

Optimal control problem

To achieve the above objective, we formulate the following OCP according to the multiple-shooting paradigm presented in Section 7.3.8. The formulation was inspired by the work of [MM12]. This implies that the time-continuous formulation in (7.23) has to be discretized. Hence, the following OCP will be solved at each discrete time t to find the optimal control sequence $U = u_{0:N_p-1|t}$. Note that to reduce complexity, this could also be modified to consider only a reduced control sequence of length $N_c < N_p$ while keeping the controls constant at the remaining times over the prediction horizon, but for ease of presentation, we will omit this option in the following and use $N_c = N_p = N$. We formulate the optimal control problem, according to a multiple-shooting approach, as follows:

$$\begin{aligned}
\underset{U,X}{\text{minimize}} \quad & J(U,\bar{U},X,Y^d,D) \\
\text{subject to} \quad & x_{k+1} = \mathrm{f_d}(x_k,u_k,\hat{\theta}_k,d_k) \quad \forall\, k \in [0,N], \\
& x_0 = \widehat{x}_t, \\
& \widehat{\theta}_k = \widehat{\theta}_t, \\
& u_{-1} = u_{0|t-1}, \\
& u_k,\bar{u}_k \in \mathcal{U}, \qquad\qquad \forall\, k \in [0,N-1], \\
& x_k \in \mathcal{X}, \qquad\qquad\quad \forall\, k \in [0,N]
\end{aligned} \tag{7.28a}$$

Note that in this formulation, both the control matrix U and the state trajectory matrix $X = x_{0:N}$ are considered free variables. $\bar{U} = \bar{u}_{0:N}$ is the target input trajectory, see Section 7.4.3 for details. The system dynamics described by $x_{k+1} = \mathrm{f_c}(x_k,u_k,\hat{\theta}_k,d_k)$ here is the time-discretized version of the continuous-time system which can be written as:

$$\dot{x}^s = f_s(x^s, z) \tag{7.29}$$
$$\dot{x}^u = A^u x^u + B^u u \tag{7.30}$$
$$0 = g_{\text{DAE}}\left(x^s, x^u, z, d, \hat{\theta}\right) \tag{7.31}$$
$$y = h(x^s, z). \tag{7.32}$$

The arguments of the OCP are the optimal input and state trajectories U^* and X^*, where the first column of the matrix U^* will act as an input to the system in a receding horizon fashion.

Target input calculation

Above OCP in (7.28) depends on the target input trajectory $\bar{U}$, which we derive as follows: Basically, it is defined for any given references Y^d and Z^d, and, for any given parameter estimate $\hat{\theta}$, and has to fulfill the equilibrium of the actuator dynamics together with the algebraic constraint

$$0 = A^u \bar{x}_k^u + B^u \bar{u}_k \tag{7.33a}$$
$$0 = g_{\text{DAE}}\left(x_k^s, \bar{x}_k^u, z_k^d, d_k, \hat{\theta}_k\right) \tag{7.33b}$$
$$y_k^d = h\left(x_k^s, z_k^d\right). \tag{7.33c}$$

This can be interpreted as the solution to the system dynamics when neglecting actuator dynamics. For an asymptotically constant reference $y_k^d = const.$, the inputs obtained in such a way will be asymptotically identical to the optimal solution $u_\infty^* = \bar{u}_\infty$.

Note that the reason for formulating the above conditions using the system description of a semi-explicit DAE was not mandatory since one could transform it to a system of ODEs. Nevertheless, this offers a neat possibility to require that information about the target trajectory is used, which consists not only of the differential state x^d, but also the algebraic state z^d. Also note that to solve the above equations for the target input, a DAE solver is not necessarily required.

Objective function

In order to meet the control objectives mentioned in Section 7.4.2, we formulate the objective function J as

$$J\left(U, \bar{U}, X, \bar{x}, Y^d, D\right) = J_{\tilde{y}}\left(Y, Y^d\right) + J_{\tilde{u}}\left(U, \bar{U}\right) + J_{\Delta u}(U), \tag{7.34}$$

where output reference tracking is obtained by $J_{\tilde{y}}(Y, Y^d)$, which is the cost penalizing deviations $\tilde{y} = y_k - y_k^d$ from the time-variant output reference and defined as

$$J_{\tilde{y}}(Y, Y^d) = \sum_{k=1}^{N} \|y_k - y_k^d\|_{Q_k}^2 . \tag{7.35}$$

The cost terms related to the control inputs $J_{\tilde{u}}(U, \bar{U})$ and the control increments $J_{\Delta u}(U)$ are defined as:

$$J_{\tilde{u}}(U, \bar{U}) = \sum_{k=0}^{N-1} \|u_k - \bar{u}_k\|_{R_k}^2 \tag{7.36}$$
$$J_{\Delta u}(U) = \sum_{k=0}^{N-1} \|\Delta u_k\|_{S_k}^2 . \tag{7.37}$$

In $J_{\tilde{u}}$, the deviations $\tilde{u}_k = u_k - \bar{u}_k$ of the inputs to the target input values are penalized. With the cost term $J_{\Delta u}(U)$, smoother trajectories are achieved by penalizing squared differences of subsequent actuator values $\Delta u_k = u_k - u_{k-1}$.

Discussion of the proposed controller

The above formulation is similar to and inspired by the work of Morari and Maeder [MM12], as explained in detail in Section 7.3.7 but differs from their formulation in the following ways:

First, [MM12] only considers targets of both state and input $(\bar{x}, \bar{u})$, which were derived as the equilibrium regarding the full state vector x, which yields a single target which is then used for the entire prediction horizon. By restricting the equilibrium condition only to the actuator states x^s, we are enabled to calculate a time-variant target trajectory. This is possible by using the state-partitioned formulation as semi-explicit DAE, as given in (3.74a). Nevertheless, for the case of an asymptotically constant reference y^d as well as constant θ and d, the formulation becomes asymptotically identical to the one in [MM12], if we interpret the unknown parameter vector θ as unknown disturbance and ignore the presence of the known disturbance d (be aware of the different meaning of d in [MM12]).

Another distinction is that unlike [MM12], we did not directly include the equilibrium conditions (7.33) into the optimization problem. This possibility was also mentioned by [MM12], Section 6.2, and is due to the fact that we have to deal with an over-actuated system for which no unique solution to the actuator equilibrium conditions exists without further regularization terms. This could also be solved within the optimization as done by [MM12]. However, for the example of longitudinal vehicle dynamics, we found it easier to require that the solution of (7.33) additionally fulfills some static control allocation condition in order to avoid, as much as possible, situations in which both engine and brakes produce torques on the wheels which work against each other.

As another distinction, the formulation of the system dynamics as partitioned, semi-explicit DAE was chosen, which offers the possibility to formulate the actuator equilibrium conditions while also being able to calculate *time-variant* input targets over the prediction horizon. By doing so, we can exploit the fact that we are given a reference trajectory consisting of output Y^d and algebraic state Z^d.

A further difference is, as we see from the cost function that we additionally include a cost term that penalizes input differences, which is required to achieve a more comfortable behavior.

Note: it is not necessary to include additional terminal costs of the form $J_N(x_N, \bar{x})$ (as discussed in Section 7.3.6) to the objective function as typically found in other formulations [MM12]. Due to the fact that the actuator states x^u follow their independent, stable dynamics, and we penalize both the terminal output with $\|y_N - y_N^d\|_{Q_N}^2$ contained in $J_{\tilde{y}}(Y, Y^d)$, as well as the last input $\|u_{N-1} - \bar{u}_{N-1}\|_{R_{N-1}}$ together with actuator equilibrium condition (7.33), this is (asymptotically) equivalent to directly penalizing deviations of the full state vector within a separate terminal cost term. Of course, the formulation of [MM12] is for more general systems than those in our presentation.

Formulation for longitudinal vehicle tracking

In our example, the feasible set for states and inputs are defined as box constraints $\mathcal{X} \in [x_{\min}, x_{\max}]$ and $\mathcal{U} \in [u_{\min}, u_{\max}]$ with the following values

$$x_{\min} = [v_{\min}, T_{\text{we,drag}}, 0]^{\mathsf{T}} \tag{7.38}$$

$$x_{\max} = [v_{\max}, T_{\text{we,max}}, T_{\text{br,max}}]^{\mathsf{T}} \tag{7.39}$$

$$u_{\min} = [T_{\text{we,drag}}, 0]^{\mathsf{T}} \tag{7.40}$$

$$u_{\max} = [T_{\text{we,max}}, T_{\text{br,max}}]^{\mathsf{T}}. \tag{7.41}$$

We solve the target by the actuator equilibrium condition together with the algebraic constraint (7.33), which for the vehicle dynamics as given in (3.75a) yields for each time k:

$$0 = \begin{pmatrix} -\frac{1}{\tau_{\text{pwt}}} & 0 \\ 0 & -\frac{1}{\tau_{\text{br}}} \end{pmatrix} \bar{x}^u + \begin{pmatrix} \frac{1}{\tau_{\text{pwt}}} & 0 \\ 0 & \frac{1}{\tau_{\text{br}}} \end{pmatrix} \bar{u} \tag{7.42a}$$

$$0 = -\theta_2 \left(x_1^s\right)^2 + \frac{\bar{x}_2^u - \bar{x}_3^u}{r_{\text{eff}}} - (\theta_1 + m_{\text{I}})z - \theta_1 g(\sin d - \theta_3 \cos d) \tag{7.42b}$$

$$y^d = [x_1^{s,d}\ z^d]^{\mathsf{T}}, \tag{7.42c}$$

and we can eliminate the first and the last equation by noting that we assumed to be also given a reference acceleration denoted by z_k^d. The second equation yields

$$\bar{u}_k = \bar{x}_k^u, \tag{7.43}$$

which follows from $A^u = -B^u$, and is clear from the fact each actuator state follows its individual first-order dynamics. What remains is the equation for the algebraic constraint, omitting time indices for readability and from

$$0 = -\theta_2 \left(x_1^{s,d}\right)^2 + \frac{\bar{u}_1 - \bar{u}_2}{r_{\text{eff}}} - (\theta_1 + m_{\text{I}})z^d - \theta_1 g(\sin d - \theta_3 \cos d), \tag{7.44}$$

it is clear that this has no unique solution for $\bar{u}$. Instead, we first substitute $\bar{T}_w = \bar{u}_1 - \bar{u}_2$, for which we obtain a unique solution and will split the target wheel torque according to the torque split first introduced in Section 8.5.1.

$$\left.\begin{aligned} \bar{u}_1 &= \bar{T}_w \\ \bar{u}_2 &= 0, \end{aligned}\right\} \quad \bar{T}_w > T_{\text{we,drag}} \tag{7.45}$$

$$\left.\begin{aligned} \bar{u}_1 &= T_{\text{we,drag}} \\ \bar{u}_2 &= T_{\text{we,drag}} - \bar{T}_w \end{aligned}\right\} \quad \bar{T}_w \leq T_{\text{we,drag}}. \tag{7.46}$$

We want to emphasize that this way, we will get an entire, and *time-variant* target trajectory $\bar{U}_{0:N|t}$ at each time t.

Note that, although this is just a static control allocation procedure, but for the target $\bar{u}$ instead of the true input u. Like this, the model predictive controller is still capable of weighting to follow the static allocation against following the dynamic trajectory on the way toward the target value. Like this, whenever necessary, the model predictive controller will dynamically deviate from the static values and, therefore, achieve improved dynamic tracking compared to the purely static case.

Also, we can achieve asymptotic offset-free tracking while avoiding the case that both brakes and engine work against each other - except in transients close to the switching point.

As outlined in Sections 7.3.7 and 7.4.3, we could also integrate the solution for the input target $\bar{u}$ as a constraint within the optimization. Nevertheless, the solution above seemed more straightforward to achieve in that the controller avoids concurrently applying brakes against the engine torque. In any case, this could also be achieved with proper regularization, but has been left to future work.

7.5 Evaluation

7.5.1 Simulation environment 7.1

A simulation environment was implemented, which is based on the longitudinal vehicle dynamics (3.76) as given in Section 3.5.3. This model was also evaluated with real-world experiments in [RH+16; AE+16], which both found a good correlation between measurements and simulation. We realized two simulation scenarios, which we want to describe in detail in the following. Further implementation details are given in Section 7.5.1.

Scenario 1

The first simulation scenario aims to demonstrate the advantages and challenges of the proposed predictive controller compared to the baseline PI controller described in Section 7.5.1. The scenario consists of a piece-wise constant velocity reference, which the controllers have to track. Additionally, road grade changes are conducted, also in a step-wise fashion, resulting in a piece-wise constant shape of the disturbance, which aims to evaluate the step response as well as the disturbance response capability of the two controllers. The target reference can be seen as a discontinuous simplification of the one occurring in a low-speed scenario through a parking garage. As can be seen in Figure 7.3 (top), the vehicle should follow a speed profile with an initial value of 5 m/s, starting from a standstill. At time t = 10 s, the reference speed value is abruptly reduced to a value of 1 m/s (e.g., to drive around a corner). At t = 15 s, the vehicle enters a ramp, which results in the road grade changing from 0 to 0.15 rad until at t = 20 s, the vehicle leaves the ramp and continues driving on a horizontal plane. Another speed step to 5 m/s is carried out at t = 25 s, to remain constant until t = 30 s. Between t = 40 s and t = 45 s, the vehicle is simulated to be on a second ramp, only this time, the ramp is steeper with a gradient of $\varphi = 0.35$ rad. In a real driving scenario, such a discontinuous change of the road grade disturbance would not occur. Nevertheless, analyzing step responses brings valuable insights into the controller performance. The author's preliminary work published in [BK16b] also used a similar experiment.

We will conduct a variety of different experiments on this scenario: The first experiment aims to demonstrate the advantage of the proposed predictive control scheme compared to a baseline PI controller in a nominal setting (without any noise or parameter uncertainty). Details can be found in Section 7.1, and simulation results can be seen in Figure 7.3.

Experiment 7.2 and Experiment 7.3 demonstrate the robustness of both the baseline PI controller (see Section 7.2) and the non-adaptive NMPC controller (see Section 7.3) to wrong assumptions about the values of the vehicle model parameters, as well as the occurrence of measurement noise.

Experiment 7.4 demonstrates how the optimal behavior can be restored when combining the NMPC controller with the state and parameter estimator from Chapter 6 to an ANMPC, despite the previously unknown parameters and the occurrence of measurement noise.

Scenario 2

The second scenario we use to evaluate the proposed control algorithm consists of tracking a dynamic vehicle speed reference trajectory. The reference is based on velocity data defined by the WLTC Class 3 cycle, which is part of the WLTP procedure defined by UNECE [UNE19], where the cycle data is available for download. Two modifications to the original cycle data have been made. First, the desired vehicle speed profile was modified from the cycle data v^d_{WLTC}. Second, we added a variable road grade profile. The modifications are calculated as follows: The original speed profile is modified according to the formula

$$v^d(t) = \begin{cases} 2.5 & \text{if } v^d_{WLTC} < 2.5 \wedge t \in [100s, 1500s] \\ v^d_{WLTC}(t) & \text{else,} \end{cases} \tag{7.47}$$

which means that except the initial acceleration and the final deceleration phase, the minimum velocity is kept to a value of 2.5 [m/s^2]. The reason for this choice will be made clear and discussed in detail in Chapter 6. Please refer to Section 6.3 for more details.

In order to create trajectory data defined on a time grid with a higher resolution than one second of the original data provided by the WLTC cycle data mentioned above, we perform a re-sampling step using the interpolation algorithm *makima* with default parameters using the MATLAB® command *interp1*.

Additionally, we define a variable road grade, while the original cycle is defined on flat terrain only. We do this by first calculating an approximation of the distance s, which the vehicle travels when following the reference speed profile as a cumulative sum of the velocity. Next, we calculate a road grade profile according to the formula

$$\varphi = 0.2 \cdot \sin(0.0005 \cdot 2\pi s), \tag{7.48}$$

where s is the distance traveled. By modifying the WLTC cycle in the described manner, we are able to evaluate the proposed controller also in a challenging but realistic driving scenario while providing an easily reproducible evaluation cycle. See Figure 7.7 for an illustration of the resulting velocity and road grade profile.

Implementation details of Simulation Environment 7.1

Simulation Environment 7.1 was implemented in MATLAB®. The final results were produced using R2021a. In order to solve the DAE, we used the MATLAB® interface of the open-source framework for numerical optimization CasADi [AG+19], which integrates the DAE solver IDAS from the SUNDIALS suite [HB+05]. The values of all parameters used within the simulation can be found in Table 7.1.

Table 7.1: Simulation parameter

Symbol	Description	Value
T_s	Simulation sample time [s]	0.01
m	Vehicle mass [kg]	1500
C_d	Aerodynamic drag coefficient [kg/m]	0.65
C_r	Rolling resistance coefficient [-]	0.015
m_I	Mass equivalent resulting from power-train inertia [kg]	40
r_eff	Effective wheel radius [m]	0.3
$T_\mathrm{we,drag}$	Maximum negative power-train drag torque [Nm]	-300
$T_\mathrm{we,max}$	Maximum wheel torque induced by engine [Nm]	1600
$T_\mathrm{we,max}$	Maximum wheel torque induced by brakes [Nm]	1800
τ_pwt	Time constant of torque response [s]	0.5
τ_br	Time constant of brake response [s]	0.1
g	Gravitational constant [N]	9.81

NMPC controller implementation In order to realize the controller and solve the underlying OCP given in (7.28), the MATLAB® interface of the open-source framework for numerical optimization CasADi [AG+19] was used. For this purpose, we find a time discrete formulation from a fourth-order Runge-Kutta M-step approximation of the longitudinal vehicle dynamics (7.23) as defined in Section B, where a step size of $M = 1$ was used. CasADi was parameterized to use the interior-point method solver Ipopt [WB06] with default parameters, except the following:

- max_iter = 2000
- acceptable_tol = 1e-8
- acceptable_obj_change_tol = 1e-6.

The NMPC controller was running at a reduced sample time of $T_{\text{MPC}} = 0.1$ s. The controller parameters used are summarized in Table 7.2.

Table 7.2: Parameter settings for NMPC controller

Symbol	Description	Value
N_p	Prediction horizon	20
N_c	Control horizon	20
N_a	Advance knowledge horizon	20
Q_k	Speed deviation weight	50000
R_k	Input weight	diag([0.001, 0.05])
S_k	Input difference weight	diag([0.02, 0.02])
T_{MPC}	Controller sampling time [s]	0.1

The adaptive NMPC controller uses the state and parameter observer as described in Section 6.5.3 in order to provide the estimated values for $\hat{x}$ and $\hat{\theta}$. On a side note, we want to mention that an intermediate solution was implemented using the DAE solver IDAS from the SUNDIALS suite [HB+05], but since this led to substantially longer evaluation times without any notable improvements in control accuracy, which was why this solution was abandoned.

Baseline PI controller The baseline controller is implemented as a cascaded PI controller with a feed-forward part. The outer loop controls vehicle speed and calculates a set point for the inner loop based on vehicle acceleration. A feed-forward part that corrects for the known disturbances is included to actively consider the known disturbance. This leads to the calculation of a desired wheel torque value T_w^d according to

$$T_\text{w}^d = r_\text{eff}\left(a^d\,(m + m_\text{I}) + mg\,(\sin\varphi + C_\text{r}\cos\varphi) + C_\text{d}\cdot\left(v^d\right)^2\right) + u_\text{PI}, \tag{7.49}$$

where u_PI is the correction term, calculated as the output of the inner-loop PI-controller. Both inner and outer-loop controllers include an anti-windup scheme on the integral part. To solve the control allocation problem, the desired wheel torque is split into a power-train and a brake torque value identical to the static method described previously in Section 8.5.1. The PI controller was running at the same sampling time as the simulation with $T_s = 0.01$ s. The PI controller cascade was tuned to be robust to parameter changes, as can be seen in Experiment 2. This resulted in values $K_p^v = 2$ and $K_i^v = 0.3$ for the outer loop controller, and $K_p^a = 1$ and $K_i^a = 15$ for the inner loop of the cascade.

7.5.2 Experiment description

To evaluate the approach, we run various experiments with Simulation Environment 7.1, which contained the two simulation scenarios

- Scenario 1: Step response regarding reference $[v^d, a^d]^\mathsf{T}$, disturbance steps on φ,
- Scenario 2: A modified WLTP test cycle.

In these scenarios, we conducted various experiments, which will be described in more detail in Sections 7.1 to 7.5. Table 7.3 aims to provide an overview of the experiments discussed in this chapter.

Experiment	Description	Results
Experiment 7.1	Scenario 1, NMPC (nominal) vs. PI (nominal)	Figure 7.3
Experiment 7.2	Scenario 1, PI (nominal) vs. PI (realistic)	Figure 7.4
Experiment 7.3	Scenario 1, NMPC (nominal) vs. NMPC (realistic)	Figure 7.5
Experiment 7.4	Scenario 1, NMPC (nominal) vs. ANMPC (realistic)	Figure 7.6
Experiment 7.5	Scenario 2, PI (realistic) vs. ANMPC (realistic)	Figure 7.7, Table 7.5

Table 7.3: Overview of experiments discussed within this chapter. All experiments conducted in Simulation Environment 7.1.

Experiment 7.1

Experiment 7.1 was conducted in Scenario 1, which is described in Section 7.5.1, and compares the performance of a NMPC controller to the one of the baseline PI controller described in Section 7.5.1. We used the nominal model in the controllers, meaning that all model parameters were time-invariant and known to the controllers and that no measurement noise was present. Since perfect knowledge of the vehicle states and parameters is given, no state and parameter estimator is necessary for this experiment. The results can be seen in Figure 7.3 and a discussion can be found in Section 7.5.3.

Experiment 7.2

Experiment 7.2 serves to compare the performance of the baseline PI controller in Scenario 1, under both nominal and realistic assumptions. Nominal means that no measurement noise is present in the first simulation (PI 1), and the model parameters for the feed-forward part are perfectly known. The second time (labeled as PI 2 in Figure 7.4), the simulation was performed using noisy measurements and with wrong vehicle parameter values. As can be seen in the bottom subplot of Figure 7.4, the parameter estimations used in the controller were a value of $\widehat{m} = 1800$ instead of $m = 1500$ kg, $\widehat{C}_\mathrm{d} = 0.8$ instead of $C_\mathrm{d} = 0.65$ and $\widehat{C}_\mathrm{r} = 0.018$ instead of $C_\mathrm{r} = 0.015$, respectively. No filter was used, and the unfiltered, noisy measurement error was directly used as a control error.

Experiment 7.3

Experiment 7.3 aims to demonstrate that the predictive but non-adaptive NMPC controller has limited robustness against deviations between the real and the estimated vehicle parameter values. Non-adaptive here means that this experiment was not using the proposed parameter adaptation scheme from Chapter 6.

Hence, Experiment 7.3 is, in principle, a repetition of Experiment 7.2, only that instead of the PI controller, we run the (non-adaptive) NMPC controller in both a nominal setting (no noise, parameters known) and a realistic setting (noise present, parameters unknown). All estimated and real vehicle parameter values were also kept identical to before. Also, the unfiltered, noisy vehicle speed measurement was directly used in the controller. This is to highlight that a filter or state estimator is beneficial to achieve smooth performance. Results are shown in Figure 7.5.

Experiment 7.4

In Experiment 7.4, the simulation for Scenario 1 is performed twice to compare (1) the NMPC controller in the nominal setting (with perfect plant parameter knowledge and without measurement noise) to the response of the (2) adaptive ANMPC controller, which uses the state and parameter estimator from Section 6.5.3 in Chapter 6. The results are shown in Figure 7.6. The initial values for the parameter estimates were chosen to be identical to the ones already used in Experiment 7.2.

Experiment 7.5

In Experiment 7.5, we evaluate the results of a simulation of the modified WLTC cycle as described in Scenario 2. The performance of two controllers is compared: the baseline PI controller from Section 7.5.1, and the proposed ANMPC controller as described in Section 7.5.1. Results are shown in Figure 7.7 as well as Table 7.5, and a discussion of the results, including the evaluation of the tracking error performance and the control effort is found in Section 7.5.3.

7.5.3 Result discussion

General discussion of Experiments 7.1 to 7.4

Comparing the performance of the two controllers in Figure 7.3, which shows the results of Experiment 7.1, one can observe that the proposed NMPC outperforms the PI controller in various ways.

First, the predictive controller has less tendency to show over- and undershoot phenomena as a reaction to the reference speed changes, and only minor deviations from the reference speed value occur due to the sudden change in road grade, which causes a discontinuous disturbance. This can be explained by the observation that overshoots under the PI controller are mainly caused by the presence of integral action, which is not the case for the proposed predictive controller.

Second, we can observe a deviation from the reference speed when looking at the PI response at time $t = 40\,\mathrm{s}$, when entering the steep ramp. In comparison, the NMPC controller only shows a slight deviation around the times when the road grade changes happen. This can be explained by the following: Although the PI controller has a feed-forward part that immediately reacts when the road grade change is measured, the presence of actuator delay reduces the effect of this immediate action. This, together with the fact that the actuator runs into its saturation due to the engine's torque limit, results in a situation where the PI controller does not manage to fully recover from the disturbance until leaving the ramp.

This effect can be considered similar to the one that motorists might observe when driving an unknown car with a weaker engine as which they are used to onto a steep ramp. In such a situation, anticipated driving might prevent the engine from stalling: if the driver accelerates sufficiently before entering the ramp, this helps compensate for a delayed engine torque build-up.

The proposed NMPC shows such an anticipated behavior. Looking at the resulting speed for the NMPC controller, we can see that in all step responses, the vehicle starts to accelerate (or decelerate) in an anticipated fashion already before the step happens. Before entering the ramps, the vehicle has built up enough speed reserve to withdraw the disturbance. The vehicle can quickly recover from the disturbance and track the reference value again. This is especially true for the second and higher road grade disturbance.

Another observation one can make is that the speed trajectories show a non-symmetric behavior regarding speed increase and decrease: when decreasing speed, the gradients of the NMPC response are considerably lower than when accelerating. This is a result of the optimized power-train and brake split behavior obtained from solving the control allocation problem also in an optimal fashion. Since the weight to penalize the brake actuator was chosen to be higher than the one for the engine, the predictive controller tries to avoid braking. This can also be seen looking at the brake torque values at the bottom of Figure 7.3. This illustrates another advantage of the proposed predictive controller: it can be easily parameterized to avoid wear and tear of brakes.

Looking at the results of Experiment 7.2 in Figure 7.4, one can observe that the PI controller is widely robust against the deviations between true and estimated vehicle parameters. It is further very well known that the control error needs to be additionally filtered in some way to avoid noisy control output values, which would cause jerks and unnecessary actuator wear otherwise. Nevertheless, such a filter is left away in this presentation to highlight the fact that (1) filtering the state (= vehicle speed value) is necessary also for the baseline controller and (2) as a reminder that filtering the state typically comes along with additional delay and reduced controller dynamics. Hence, also a PI controller might benefit from the state and parameter estimator presented in Chapter 6.

In Experiment 7.3, we compared the NMPC controller in a nominal and a realistic setting with uncertain parameters. From the results in Figure 7.5, we can notice two main observations. First, the NMPC controller without parameter adaptation is prone to overshoots as a reaction to set-point changes. Second, and more concerning, in the phases during which the road grade differs from zero, the controller is not able to perform offset-free tracking. This is due to the fact that no integral action is present.

In Figure 7.6 of Experiment 7.4, one can observe that by combining the NMPC controller with the estimator, the optimal control performance of the nominal system can be retained. In the bottom subplot of Figure 7.6, the progress of the vehicle parameter estimates are displayed. Despite the presence of measurement noise and a deviation between the true vehicle parameter and their believed initial values, the parameter estimator is able to quickly converge to a region sufficiently close to the true values such that the influence on control performance becomes neglectable. A more detailed analysis of the estimator performance can be found in Section 6.5.3.

Tracking performance

To evaluate the tracking performance of the different controllers, we compare the RMSE values of the resulting velocity trajectories to the reference values. The results of this evaluation for both Scenario 1 and Scenario 2 from data obtained in Experiments 7.1 to 7.4 can be seen in Table 7.4.

For further analysis, we can interpret Experiments 7.1 to 7.4 in a way that we run Scenario 1 for all the controllers (PI, NMPC, ANMPC) both under nominal and realistic conditions (with and without the presence of measurement noise and wrong model parameter assumptions). One can observe that the model predictive controllers' tracking performance is generally much better than that of the baseline PI controller but degrades in a realistic setting without any adaptation. Combining the NMPC controller with the state and parameter estimator to make it adaptive keeps the performance close to optimal under realistic assumptions.

	Controller		
Scenario	**PI**	**NMPC**	**ANMPC**
Scenario 1 (nominal)	1.0536	0.7507	0.7507
Scenario 1 (realistic)	1.0547	0.7942	0.7508

Table 7.4: Root mean square error (RMSE) of the tracking errors for different controllers under nominal and realistic conditions obtained from simulation results of Scenario 1.

Looking at the results of Experiment 7.5, which are given in Figure 7.7 and Table 7.5, we can observe that the proposed controller is able to follow the reference trajectory of Scenario 2 very accurately. This can especially be seen looking at the third row of the figure, which shows a zoom in to the trajectories obtained in the part of the cycle which is marked by a rectangle in the first row, and also looking at the tracking error $e_v = v - v^d$ in the fourth row of the plot. We find that the adaptive, model-predictive controller is able to follow the reference with substantially less tracking error than the baseline controller: the resulting RMSE value of the proposed controller is even more than 90 % reduced compared to the baseline. This could be obtained even though the control effort is approximately equal, or even slightly less than with the baseline controller, as we will see in Section 7.5.3. The accurate tracking performance of the proposed controller is made possible thanks to the predictive nature of the controller, as well as using the estimates of the state and parameter estimator, which shows convergence to the true vehicle parameters, as one can see in the bottom of Figure 7.7.

Control effort

In order to evaluate the control effort the proposed control algorithm produces compared to the baseline controller, we performed simulations of the modified WLTC test cycle described in Scenario 2. We evaluate two simulation runs with: (1) the baseline PI controller, (2) the proposed ANMPC controller with the controller parameters as used in Experiments 7.1 to 7.4.

According to Pavlovic, Marotta, et al. [PM+16], in the case of a vehicle driven by an ICE, the average CO2 emissions one has to expect during a test cycle depends on the average power at the wheelbase [PM+16], according to the linear relation

$$CO_2 = k_v P(t) + D \tag{7.50}$$

where the power follows the equation

$$P(t) = F_{\text{tire}}(t) v(t)\,[W] \tag{7.51}$$

where the first equation is the Willans equation with the Willans coefficient k_v, and the $P(t)$ is the power at the wheelbase, depending on the tire force. The above relation is used within the WLTP framework to correct the test results for deviations from the target speed values [PM+16]. From the relationship we derived in (3.15), it becomes clear that the only source of fuel consumption is the engine gross torque at the wheel and that this would be even higher if the engine works against the brakes. So we want to look at the wheel torque induced by the engine, corrected by the engine drag torque as an indicator for fuel consumption, which is proportional to CO2, in the following. In reality, other factors besides fuel quantitiy also determine the engine's torque. Nevertheless, lacking a more sophisticated engine model, this can be used as a rough approximation, which can be used for a relative comparison. Therefore, we evaluate the average of the net wheel torque induced by the engine

$$T_{\text{fc}} = \text{mean}\left(T_{\text{we,k}} - T_{\text{we,drag,k}}\right) \tag{7.52}$$

over the entire driving cycle and compare the values to the result under the baseline controller. We find a reduction of at least 1 percent for a setting that is optimized for tracking performance, where the reduction of the RMSE of the tracking error is over 90 %. The results are summarized in Table 7.5.

Controller	**Avg. eng. torque**		**Tracking error (RMSE)**	
	abs.	**rel.**	**abs.**	**rel.**
PI	540.17	100 %	0.18423	100 %
ANMPC	535.59	99.15 %	0.01269	6.89 %

Table 7.5: Evaluation results for Experiment 7.5. The tracking error (RMSE) could be substantially reduced (-93 %), while the average gross engine torque as an indication of fuel consumption was also slightly less (-1 %)

One of the benefits of the proposed ANMPC approach is that the controller can easily be tuned to balance the weight parameters to either improve tracking performance (in exchange to control effort and allowing less smooth actuator commands) or to reduce control effort (at the expense of tracking performance)

Computation times

The execution time for solving the OCP of Experiment 7.5 on a Windows 10 Laptop with an Intel®Core™i7-10510U CPU running at 2.30 GHz and 16 GB RAM was in average 22.19 ms, with a standard deviation of 2.02 ms, a maximum value of 63.68 ms and a minimum value of 15.86 ms. These results were obtained from a typical simulation of Experiment 7.5, which we consider representative due to the large number of 180.000 samples. As mentioned, the implementation was performed using the MATLAB® interface to CasADi [AG+19], with Ipopt [WB06] as a solver. Considering that the controller sample time was 100 ms, and that one can expect a speedup by at least one magnitude from using an optimized solver according to [ZD+17a], one might expect that the implementation of the controller would be feasible also on a real-time platform. Further investigations in this direction have to be considered as future work and will be discussed more in detail in Section 9.3.2.

7.6 Conclusion

7.6.1 Summary

This chapter presented a proposal for a novel solution to the longitudinal vehicle motion tracking problem of automated vehicles. The proposed control scheme is an adaptive, nonlinear, model predictive controller (ANMPC), which, contrary to standard MPC formulations, is also able to incorporate advance knowledge about future values of both a time-varying reference speed, as well as a time-varying road grade. Most state-of-the-art predictive control schemes consider the reference constant over the entire prediction horizon, and do not include any known disturbances as input to the system dynamics. To retain optimal performance under parametric uncertainties, as another proposal, the NMPC controller is combined with the state and parameter estimator from Chapter 6, which makes the control scheme adaptive. This offers the possibility to achieve offset-free tracking when the estimator has converged to the true values. Several simulation studies have been carried out and showed promising results, which demonstrate that the proposed ANMPC controller is able to provide superior control performance compared to the baseline controller, despite the challenges arising from parametric uncertainties. Additionally, it could be shown how the control performance of a standard MPC formulation degrades under parametric uncertainty while the proposed controller achieves a behavior that is close to the optimal solution under nominal conditions. As another benefit, the proposed solution inherently solves the control allocation problem and, therefore, is able to calculate optimal actuator commands, which can be sent to the redundant actuators of the engine and brakes. Evaluations of the proposed controller in simulation studies of different driving scenarios showed promising results in order to further investigate towards a future embedded implementation and experiments with real vehicles.

7.6.2 Discussion

The proposed approach has several advantages compared to other control schemes for the purpose of longitudinal vehicle motion tracking. These advantages are:

- The proposed ANMPC controller is able to achieve optimal tracking behavior also in the presence of external but known disturbances. This is possible by including time-varying advanced information into a model predictive control scheme, which additionally allows the consideration of actuator delays, lags, and constraints.
- The proposed controller retains optimal and asymptotically offset-free trajectory tracking behavior even under parametric uncertainties. This can be achieved thanks to the combination of the controller with a robust state and parameter estimator.
- Thanks to the consideration of actuator delays and lags within the longitudinal motion tracking module, higher-level motion control design is facilitated, provided that the high-level controller provides predictive information.
- The model predictive control scheme inherently provides a solution to the control allocation problem, with the ability to consider specific actuator delays and lags for each redundant actuator.
- The approach is intuitively tunable for either high tracking performance or improved passenger comfort and economic performance by modifying the weights within the objective function.
- The proposed control module not only provides actuator commands given at the current time, but optimal actuator command trajectories are calculated over the full control horizon. This additional information can, in the future, be further exploited within lower-level actuator control modules, which themselves are designed in a predictive fashion. Although no investigation in such direction has been performed within this thesis, other results which were published recently [MW19] show this might have the potential to further reduce emissions (up to 13 % NOx reductions in the cited study) or energy consumption in future electric vehicles.
- An additional advantage might be the fact that not only the proposed longitudinal motion tracking controller but also control units designed for other tasks may benefit

from the available parameter estimates. Important examples are emergency brake assist functions or active safety assist functions, including lateral motion control at the limits of handling, which all heavily depend on correct vehicle mass estimates.

As there is “no free lunch”, the advantages can only be achieved at the expense of a computationally more expensive solution as well as increased implementation effort. Nevertheless, we already stated in the introduction that we assume it is reasonable to argue that in future vehicles, more powerful computing hardware will be available at reduced costs. This was also discussed in the work of, e.g., Di Natale and Sangiovanni-Vincentelli [DS10], Stahle, Mercep, Knoll, and Spiegelberg [SM+13], and Buechel, Frtunikj, et al. [BF+15] on recent developments on automotive hardware. This might alleviate the increased computational cost of the approach in the future. Also, much progress has been made in the last years, especially in the development of powerful and fast numeric solvers, together with novel ideas on how to reformulate the nonlinear program involved with solving the optimal control problem, and one can expect that these developments will all help to further reduce the computational effort needed for a solution. For a discussion about future work related to motion tracking, please refer to Section 9.3.2.

7.6.3 Contribution

The results in this chapter are based on previous work, which was already published in the author’s paper [BK16b]. In [BK16b], an ANMPC scheme was proposed, to the best of the author’s knowledge, for the first time as a solution to the longitudinal motion tracking control problem of automated vehicles. In [BK16b], the predictive controller was combined with the state and parameter estimator from another of the author’s publication [BK16a], which can be seen as the predecessor of the results from Chapter 6. For contributions related to the state and parameter estimator, the reader is referred to Section 6.7.2. We first want to review the contributions already published in the preliminary work before listing novelties appearing for the first time in this chapter. To the best of the author’s knowledge, in [BK16a] and [BK16b] we were the first ones to propose:

- a novel control architecture for automated vehicles that introduces a predictive longitudinal motion tracking unit. This enables motion tracking for various driving scenarios in a modular fashion
- a predictive tracking controller that not only considers time-variant references but also time-variant advance knowledge of future disturbances. Both are typically not included in other MPC formulations due to the increased complexity of the resulting optimal control problem.
- the combination of the NMPC scheme with the state and parameter estimator proposed in Section 6.5.3, which enables to achieve offset-free tracking under uncertain and time-varying vehicle model parameters. Compared to other offset-free tracking schemes, the resulting Adaptive Nonlinear Model Predictive Control (ANMPC) scheme retains the optimal dynamic behavior.

Subsequent investigations led to enhancements and improvements of the approach compared to the results presented in [BK16b], and are regarded as contributions that are presented for the first time within this thesis. These include:

- a novel tracking MPC formulation for a type of nonlinear system is provided, which can achieve offset-free tracking in combination with a state and parameter estimator.

This formulation presents a novel method for computing the target values, which are inherent in offset-free predictive control.

- the control allocation problem is additionally solved implicitly within the optimal control problem by reformulating it as a multi-output problem. This eliminates the need for a separate control allocation module dedicated to splitting the total desired wheel torque into the redundant actuators of engine or power-train torque and brake torque. Additionally, this bears the advantage that individual actuator delays and lags of both engine and brakes, as they appear in reality, can be considered with this approach.
- the reformulation of the original optimal control problem using a multiple-shooting approach, which leads to substantially reduced computation times needed for the numerical solution of the resulting NLP.
- a variety of simulation studies were carried out to demonstrate the capabilities of the proposed approach.
- an extensive discussion of background, related work, and possible future work is provided.

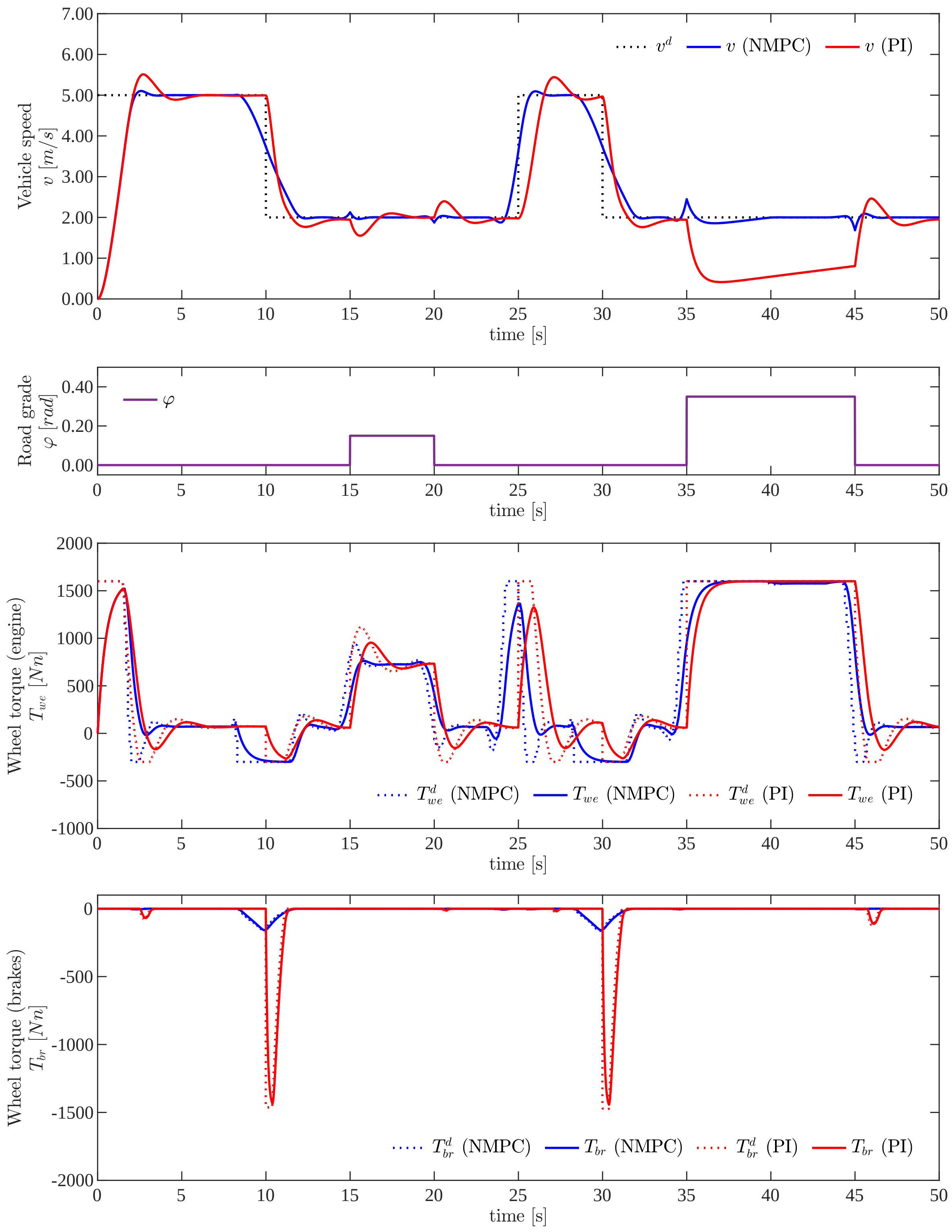

Figure 7.3: Simulation results of Experiment 7.1. Comparison between PI baseline controller and NMPC, both in a nominal setting. The predictive controller (NMPC) shows superior tracking capability compared to the baseline PI controller. Details are discussed in Section 7.5.3.

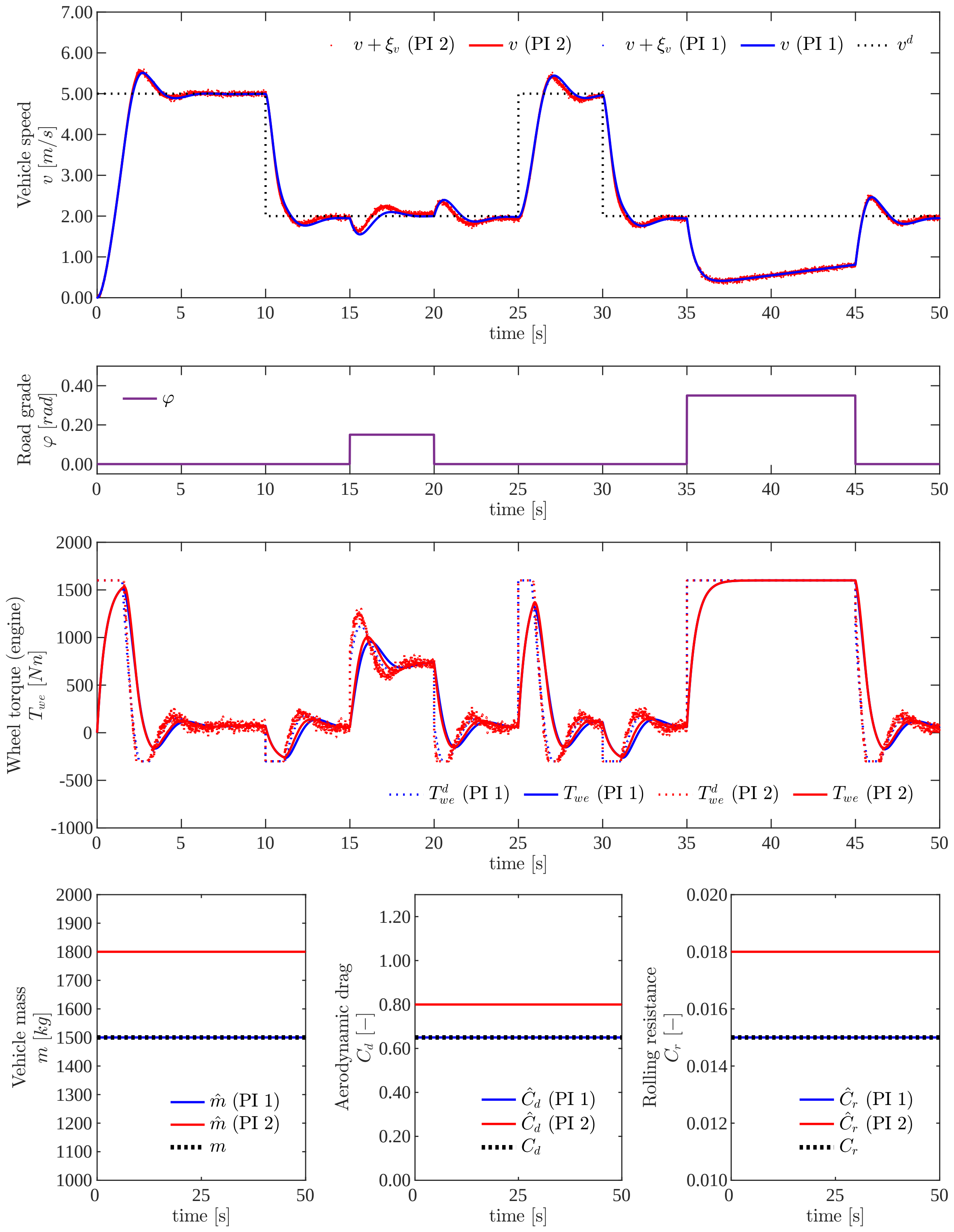

Figure 7.4: Simulation results of Experiment 7.2. Comparison of the baseline PI controller in a nominal (PI 1) and a realistic setting (PI 2). The PI controller shows a certain robustness against deviations between real and estimated vehicle parameter values. Details are discussed in Section 7.2.

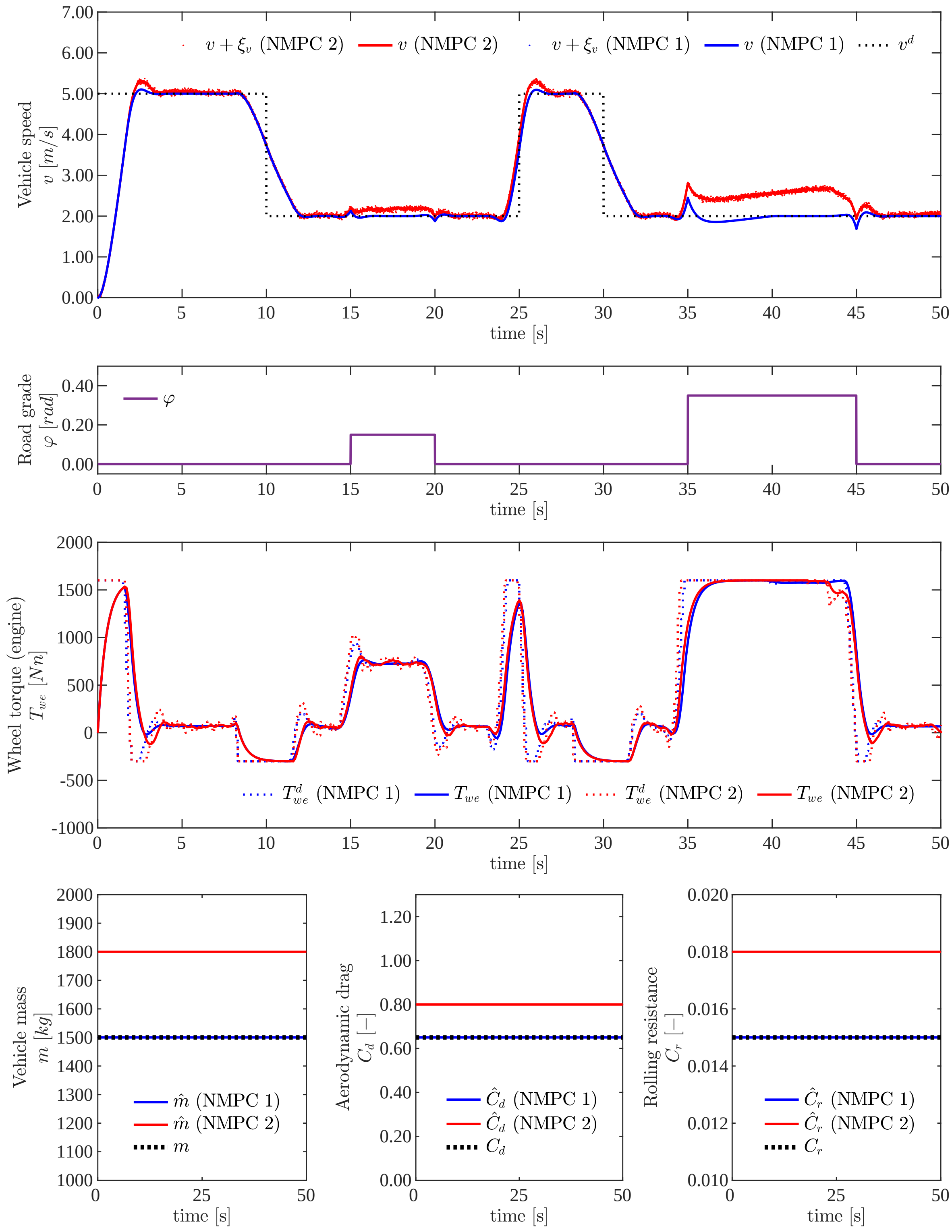

Figure 7.5: Simulation results of Experiment 7.3. Comparison of NMPC controller in a nominal (NMPC 1) and a realistic (NMPC 2) setting. Without further countermeasures, the NMPC controller is not robust against deviations between real and estimated vehicle parameter values, especially when high road grades are present.

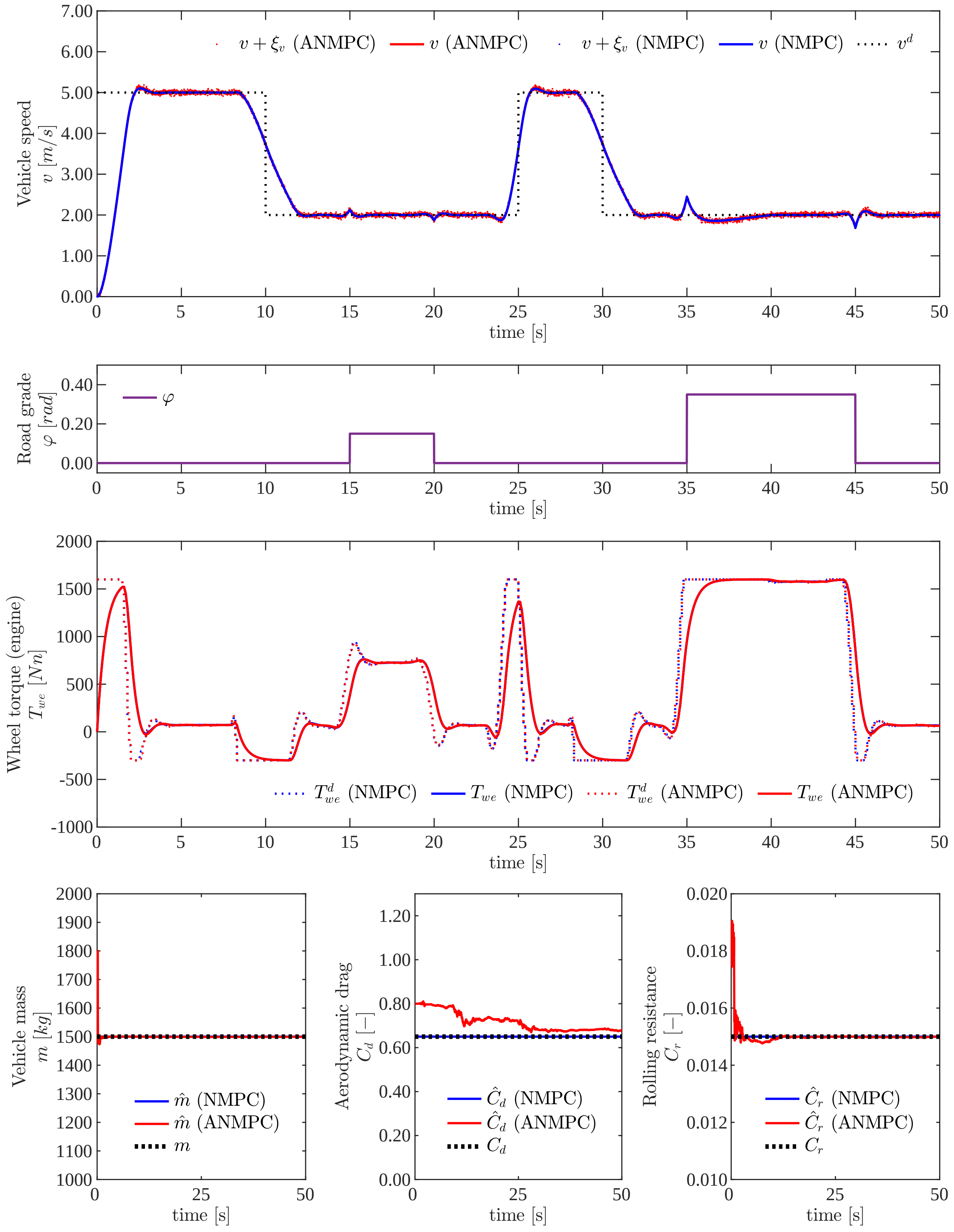

Figure 7.6: Simulation results of Experiment 7.4. Comparison of the nominal predictive controller (NMPC) and the proposed adaptive (ANMPC) controller. The latter achieves optimal performance despite the presence of deviations between values of real and estimated vehicle parameters, as well as measurement noise. Details are discussed in Section 7.4.

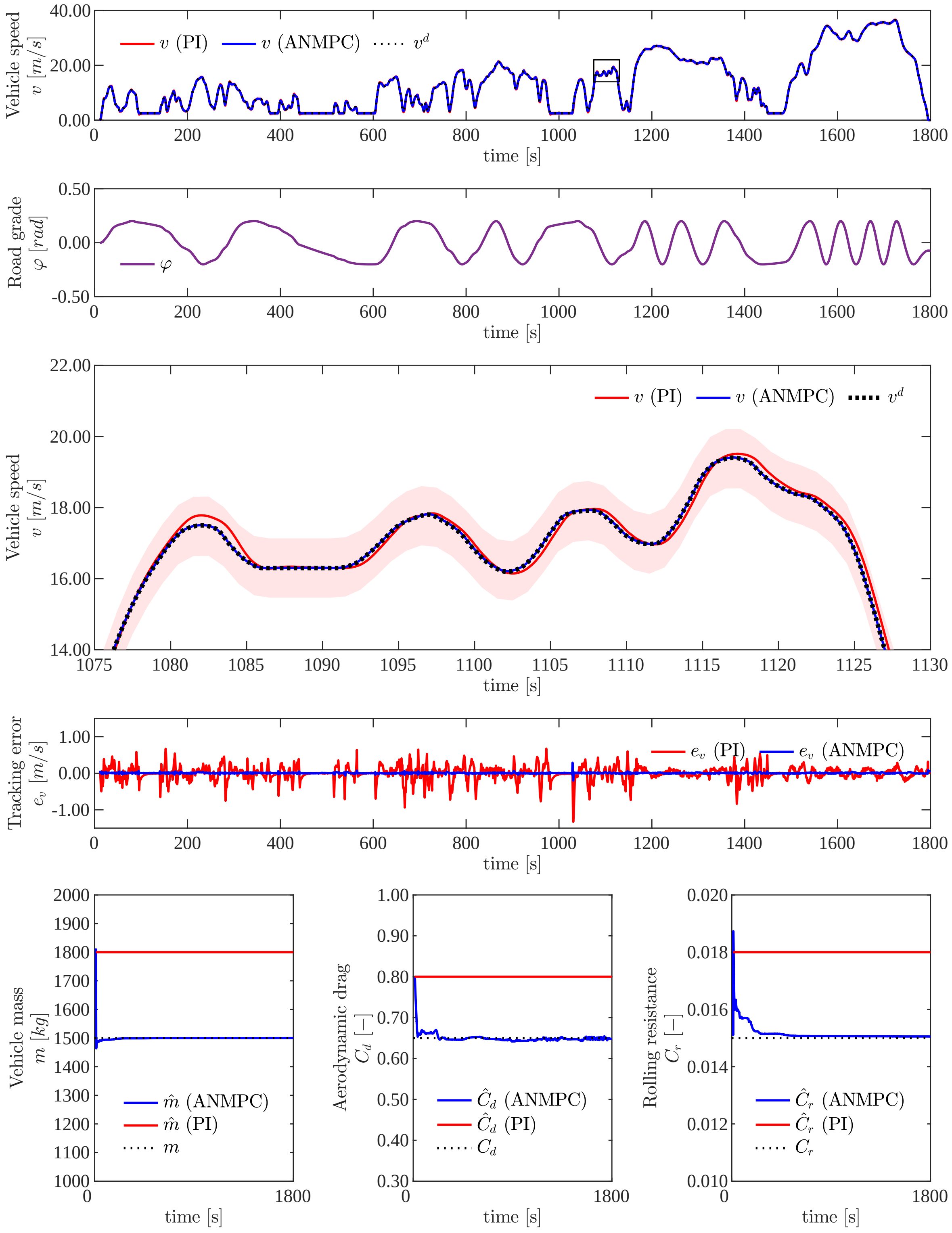

Figure 7.7: Simulation results of Experiment 7.5. The proposed adaptive controller ANMPC is able to follow the reference trajectory with increased performance compared to the baseline PI controller. The third subplot shows a zoom to the region marked by the rectangle in the first subplot.

8 Predictive Deep Reinforcement Learning Controller

"Good and evil, reward and punishment, are the only motives to a rational creature: these are the spur and reins whereby all mankind are set on work, and guided."

– John Locke, 1632 - 1704

Note: The main results discussed in this chapter were already presented in the author's publication "Deep reinforcement learning for predictive longitudinal control of automated vehicles" [BK18] during the Intelligent Transportation Systems Conference 2018 (Copyright ©2018, IEEE). In addition to these results, this chapter includes an introduction to reinforcement learning background, a detailed explanation of the selection process of the algorithm, which builds the foundation of the proposal in Buechel and Knoll [BK18], and a detailed discussion on the approach.

8.1 Introduction

The recent success of Reinforcement Learning (RL), especially deep reinforcement learning, motivates us to analyze its application to the predictive longitudinal vehicle motion tracking problem. This Chapter presents results which were partly presented in [BK18] after introducing the basics of reinforcement learning, its principles, and some of the main algorithmic concepts.

In 2017, the application of Deep reinforcement learning algorithms led to media attention when the programs "AlphaGo" and "AlhpaZero" defeated the human-master Go player Lee Sedol [SH+16] and won against the world champion chess program, Stockfish 8 [SH+17b]. It was perceived by the public as a breakthrough in artificial intelligence, mainly because the algorithms were able to teach themselves in self-play, only knowing the rules but without programming any prior knowledge. The solutions found by RL algorithms to the problems were highly creative and gained new insights into winning strategies in the well-studied problems of playing chess and Go.

While the algorithms applied for playing games are not directly applicable to real-world control problems, where continuous instead of discrete action and state spaces exist, continuous control problems were studied recently within the RL community. This led to significant progress in this subdomain [SB18]. There is evidence that RL algorithms are capable of finding a close-to-optimal solution compared to MPC approaches [EG+09].

One significant distinction lies in the RL approach to solving control problems: the agent learns by self-interaction with the environment. On the one hand, this property brings many advantages: the derivation of analytical models, and especially the determination of their parameters, becomes obsolete. This is a desirable property, not only for domains where the derivation of such models is complicated or the determination of parameters is impossible. The application of MPC and especially NMPC is a time-consuming task and requires highly qualified staff, not only for the design but also for the calibration of the controller. Hence, it would be interesting for the industry to find and calibrate controllers automatically using (standardized) reinforcement learning algorithms.

Recent studies [DR17] show that the computational complexity of solving the optimal control problem in the NMPC approach still poses challenges for real-world applications when long prediction horizons in combination with small sampling times arise.

Another drawback of model predictive control besides finding a useful model is the need to accurately determine the model parameters since parameter deviations might impact the controller's robustness. Estimating these parameters also might pose challenges, as we discuss in Chapter 6.

Looking at the problem under investigation in this thesis, we find that other researchers also started working in the direction of model-free longitudinal vehicle tracking in order to avoid the determination of model parameters [PD+17], although not utilizing reinforcement learning, but by using an ultra-local, linear model of the plant. As an alternative to the MPC approach presented in Chapter 7, we want to analyze the application of a sub-field of Deep reinforcement learning, namely model-free RL, to the predictive longitudinal vehicle motion tracking problem.

This chapter is organized as follows: we first provide the necessary RL background in Section 8.2. Then, we give an overview of related work in Section 8.3. Section 8.4 presents the proposed solution before evaluation details are given in Section 8.5. Results are presented and discussed in Section 8.6. The concluding Section 8.7 includes a list of the contributions presented in this chapter.

8.2 Reinforcement learning background

Before presenting our results, we want to give some background about RL, which is necessary to understand the proposed solution. A complete tutorial on RL is out of scope of this work. However, the interested reader is referred to the books *Reinforcement learning: an introduction* by Sutton and Barto [SB18] (the second edition following the original publication of 1998) and *Reinforcement learning : state-of-the-art* by Wiering and Otterlo [WO12]. The latter provides information about recent developments in special topics of RL besides giving an introduction. *Reinforcement learning and optimal control* Bertsekas [Ber19] is very much recommended for readers with a control systems background, as it widely uses the notation common within this field and builds many bridges between control systems and reinforcement learning communities. An excellent online tutorial can be found on the website of OpenAI [Ope]. This background section contains material from these sources, condensed to give the reader the building blocks of the algorithms used in the considerations and the proposed solution. It is deliberately held informal to focus on basic concepts rather than the mathematical background, which can be found in the above sources.

In the remainder of this section, we first want to introduce the taxonomy widely used in RL literature, but not without emphasizing differences and similarities to control literature. We then highlight the reinforcement learning principle and give an overview of different classes of reinforcement learning algorithms. Additionally, we will discuss some properties of different RL algorithms, which aim to help understand the contributions of this work.

8.2.1 Reinforcement learning and deep reinforcement learning

Reinforcement learning can be defined as a family of machine learning algorithms that seek solutions to the problem of sequential decision-making. Deep reinforcement learning combines reinforcement learning with function approximation using deep neural networks. We want to start with similarities between the control loop framework, which will be more familiar to the reader with a control systems background. We then want to formally define the framework of Markov Decision Processs (MDPs), which can be regarded as the mathematical foundation for most modern RL algorithms (see Figure 8.1). In closed-loop control, a con-

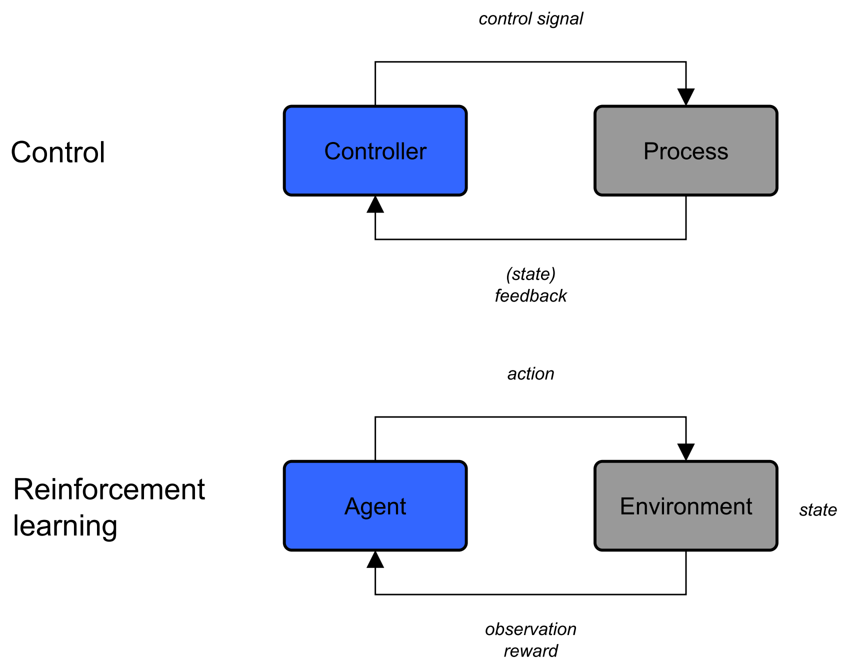

Figure 8.1: The principle of reinforcement learning and its similarity to a continuous control loop.

troller sends a control signal (typically denoted by u) to a process. The feedback is handed to the controller, which calculates the next action to apply. This feedback could be the control error, the measured output of the controlled variable, the full state vector in the case of a fully observable system, or a subset of it when controlling a partially observed system (Figure 8.1, top). In the case of MPC, the action is optimal with respect to the cost function as long as some assumptions on both model and environment hold.

In reinforcement learning, an *agent* replaces the controller. The agent performs an *action* on the process, usually named by the term *environment* (Figure 8.1, bottom). The action in RL literature is typically denoted by a. Since a in the context of vehicle control is already used for the physical quantity of acceleration, we will stick to the notation u for the action in the remainder of this thesis. The agent follows a certain *policy*, which is a mapping from *states* to actions. A policy might be stochastic (with a certain probability to perform a specific action) or deterministic. The underlying mathematical principals of RL are derived on the assumption that the process is a MDP (see Section 8.2.2). The state of the process will evolve depending on the action, and it can be observed via the observations. In addition to the observations, a reward signal is given to the agent. This reward might be influenced by the very last action performed (immediate reward), or by an action taken earlier (delayed reward). The agent's objective is to maximize the total reward it will receive in its lifetime.

8.2.2 Markov Decision Process

More formally, in a Markov Decision Process (MDP), an agent interacts with an *environment* E at each discrete time-step t. At each time t, the agent receives an *observation* o_t about the environment's *state* $x_t \in \mathcal{X}$. Again, we stick to the notation common in the control community rather than using the machine learning notation, where s is typically used to denote the state. According to Wiering and Otterlo, "A state is a unique characterization of that is important in a state of the problem that is modeled. For example, in chess, a complete configuration of board pieces of both black and white is a state" [WO12]. The environment E might be stochastic and is modeled with an initial state distribution $p(x_1)$. The observation might be the full state (fully observed) but also only contain parts of the state information (partially observed) while we focus on the former case. Also, a common setup for DRL is that the observation is a multi-variable state function, for example, an environment image.

Based on that observation, the agent then performs an *action* $u_t \in \mathcal{U}$. Actions are used to control the environment and are not limited to control signals but could rather also be some discrete decision like entering a door or not or moving a piece of chess to a certain position. The set of actions $\mathcal{U}(x)$ is the subset of all possible actions that can be performed in a certain state. By applying an action u_t in a state x_t, the system transitions from x_t to a new state $x_{t+1} \in \mathcal{X}$. This transition is based on the probability distribution over the set of possible transitions.

The *transition function* $T(x_t, u_t, x_{t+1})$ is defined as $T : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \rightarrow [0,1]$, which is the probability that, starting in state x_t and performing action u_t, the system will end up in state x_{t+1}. T is also called the *transition dynamics* of the MDP.

The agent then receives a scalar *reward* r_t, which is dependent on the reward function $R(x_t, u_t, x_{t+1})$, defined as $R : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \rightarrow \mathbb{R}$ (some other definitions exist, see [WO12], p12). We want to give a reward for the transition between states x_t and x_{t+1} given the action u_t. The reward can be looked at as the negative cost in optimal control. While in optimal control, one usually tries to minimize the cost, the agent aims to maximize the reward.

In an MDP, the state and reward depend only on the state of the process in the time step occurring immediately before, as well as the action taken in that state. Previously visited states do not influence the process.

$$p(x_{t+1}|x_t, u_t, x_{t-1}, u_{t-1}, \dots) = p(x_{t+1}|x_t, u_t) = T(x_t, u_t, x_{t+1}). \tag{8.1}$$

This substantial property is called the *Markov property*. It means that the state must include all necessary information to fully describe the effect of a particular action. For dynamical systems, this might be, for example, to include the acceleration value as a derivation of the velocity. We want to summarize the formal definition of an MDP as follows:

Definition 1. *A* ***Markov decision process*** *is a tuple* $\langle \mathcal{X}, \mathcal{U}, T, R \rangle$ *in which* $\mathcal{X}$ *is a set of states,* $\mathcal{U}$ *a set of actions,* T *a transition function defined as* $T : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \rightarrow [0,1]$ *and* R *a reward function defined as* $R : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \rightarrow \mathbb{R}$.

For many tasks, like playing chess, $\mathcal{X}$ and $\mathcal{U}$ are finite sets of states and actions. The *model* of the MDP is defined by T and R. Approximations of T and R are also referred to as models. If any of T and R are time-dependent, the MDP is *non-stationary*. Otherwise, it is *stationary*.

Episode

An episode, also called roll-out, is a trajectory τ_r, defined as a sequence of states and actions in the environment

$$\tau_\mathrm{r} = (x_0, u_0, x_1, u_1, \dots).$$

A task is called *episodic* if there exists one or more terminal or goal states in which the execution of any actions leads back to this same terminal state or the process ends. The game of chess, for example, is episodic. *Finite, fixed horizon* tasks run for a certain fixed time.

Policies

A policy is a function that provides the action in dependency of the state x. It can be either deterministic; then it is often denoted as μ and defined as $\mu : \mathcal{X} \to \mathcal{U}$. Stochastic policies are commonly denoted π and defined as a probability $\pi(x,u) = p(u|x)$ defined for discrete action spaces as $\pi : \mathcal{X} \times \mathcal{U} \to [0,1]$ such that for each x, it holds that $\pi(x,u) \geq 0$ and $\sum_{u \in \mathcal{U}} \pi(x,u) = 1$. In the case of continuous action spaces, π represents a Probability Density Function (PDF) on the action space, and the sum becomes $\int_{u \in \mathcal{U}} \pi(x,u)du = 1$.

Return and regret

The goal in RL is to find an agent that follows an optimal policy. We need to discuss more in detail what optimal means. We defined the reward as an immediate measure of how good it was to end up in the current state. Since we want to collect as much reward as possible in the long term, we need to define a measure of cumulative reward over time, which we name as *return* for a certain episode $G(\tau_\mathrm{r})$. One way that would work for episodic tasks is to define the optimal policy as the one which creates the most reward over a whole episode, and hence, building the sum over expected rewards during the whole episode length would be viable. However, for stochastic processes, we would have to keep repeating episodes over an infinite number of trials. For infinite tasks, we could build the sum over infinite steps, but this is very inconvenient since the sum would become infinite too. Hence, one method could be to build the sum of expected rewards only over a limited horizon of time steps into the future, and hence, optimal would be to perform the action that produces the most expected return over that horizon. This is the principle of receding horizon control. Another method to measure how much reward will be collected in the future is to define the *return* as

$$G(\tau_\mathrm{r}) = \sum_{t=1}^{\infty} \gamma^t r_t, \tag{8.2}$$

the infinite horizon discounted future reward. γ is the discount factor, with $0 \leq \gamma < 1$. This definition is widely used because of its nice mathematical property of giving a bounded infinite sum. A measure to compare a policy to the optimal one is to define the *regret* as the difference of the return of any policy π to the return gained due to following the optimal policy, denoted by π^*.

The goal of reinforcement learning

As already stated, the goal in RL is to find an agent that follows an optimal policy. We can now define more formally the expected return given a policy π in a stochastic world with

$$J(\pi) = \mathbb{E}[\tau_\mathrm{r} \sim \pi | G(\tau_\mathrm{r})]. \tag{8.3}$$

We want to maximize the expected return hence the optimization problem of RL can be seen as

$$\pi^* = \arg\max_{\pi} J(\pi), \tag{8.4}$$

with π^* being the optimal policy.

Value functions

A measure of how good it is to be in a certain state is the expected return for that state when following a certain policy. It is denoted with $V^{\pi}(x)$, the value of a state x under policy π, hence the expected return under policy π, defined as *the (state) value function*:

$$V^{\pi}(x) = \mathbb{E}[\tau_{\mathrm{r}} \sim \pi | G(\tau_{\mathrm{r}})] \, x_0 = x. \tag{8.5}$$

Another value function is the *action-value function*, which is the value of a policy when starting in state x, taking an arbitrary action u in that state, and only then following the policy π from that state on. It is defined as *the Q-function*:

$$Q^{\pi}(x,u) = \mathbb{E}[\tau_{\mathrm{r}} \sim \pi | G(\tau_{\mathrm{r}})] \, x_0 = x, u_0 = u. \tag{8.6}$$

The value functions under an agent acting according to the *optimal* policy π^*, are the *optimal value functions* $V^*(x) = V^{\pi^*}(x)$ and $Q^*(x,u) = Q^{\pi^*}(x,u)$.

If an agent knew the optimal values of all the successor states that can be reached from a given state, this information could be exploited to find the optimal actions. To exploit this information, it would be additionally required to have knowledge about the transition function, aka about to which successor state a given action would lead. The advantage of the action-value function is that it already incorporates the action into the value. This enables an agent, once it knows the action-value function, to consult it for the action-values of all possible actions in a given state. In order to act optimally, it suffices to take the action with the highest Q-value. Note: While the immediate reward r only measures how good a certain action is in a myopic sight of one step ahead, the value functions describe how good a policy is in the long run.

Another value function is the *Advantage function*, defined as the difference between action-value and the value function:

$$A^{\pi}(x,u) = Q^{\pi}(x,u) - V^{\pi}(x). \tag{8.7}$$

Exploration - Exploitation trade-off

In reality, the value functions are not known to the agent during reinforcement learning. If an agent wants to collect as much reward as possible, it needs to act to what in its *belief* is the optimal policy. This behavior *exploits* its knowledge about optimality. Since this policy was gained with limited knowledge, it will most likely be sub-optimal, and therefore, it needs to *explore* different, possibly better ways of acting. This inherently bares the risk of performing worse on the first run until a different strategy proves to be better. Hence, the agent needs to trade optimality by exploitation against potentially worse outcomes while exploring alternative policies.

Credit assignment

The case when reward is delayed brings a problem called the *credit assignment problem*. This is the difficulty of judging which action at which time led to the delayed reward. In this context, a related difficulty is when the reward function is *sparse*, meaning that, for example, no or zero reward is given most of the time, and only in some states, reward is achieved, for example, when successfully finishing a game.

8.2.3 Solving Markov Decision Processes

We stated in Section 8.2.2 that reinforcement learning aims to find the optimal policy that maximizes the return, which is the long-term reward we get in an environment. Many different algorithms have been developed to calculate the optimal policy. Most make use of a fundamental approach developed by Richard E. Bellmann called:

Bellmann's Principle of Optimality

"Principle of Optimality: an optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions."

– Richard E. Bellmann, 1954 [Bel54]

Bellmann equation

This principle can be expresses formally in the *Bellmann Equation* [Bel57], if we rewrite the value function for the discounted reward in a recursive manner:

$$\begin{aligned} V^{\pi}(x) =& \mathop{\mathbb{E}}_{\pi}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | x_t = x\right] \\ =& \mathop{\mathbb{E}}_{\pi}\left[r_t + \gamma V^{\pi}(x_{t+1}) | x_t = x\right] \\ =& \sum_{u_t} \pi(u_t|x_t) \sum_{x_{t+1}} T(x_t, \pi(x_t), x_{t+1}) \left[R(x_t, u_t, x_{t+1}) + \gamma V^{\pi}(x_{t+1})\right]. \end{aligned} \tag{8.8}$$

Similarly, this can also be done for the action-value function.

Temporal difference learning

The Principle of Optimality and the recursive formulation of the Bellmann equation have an important consequence for learning. This is, that the immediate reward already brings some information about the nature of the environment and the returns that can be achieved. This information can be recursively used to optimize the value function approximation according to the following rule:

$$V_{t+1}(x_t) = V_t(x_t) + \alpha\Big(r_t + \gamma V_t(x_{t+1}) - V_t(x_t)\Big), \tag{8.9}$$

with the learning rate α. The equation above goes back to [Sut88] and only uses one-step ahead experience and is called TD(0), with TD denoting *temporal difference* and (0) denoting the look-ahead further than the one-step temporal difference error. Other variants exist, using more steps along the environment. See, for example, Wiering and Otterlo [WO12], p. 37 for more information.

Q - learning

From the Bellmann equation for the action-value function, the Q-learning algorithm [Wat89] can be derived. It iteratively updates an estimation of the Q-function, according to the following update rule:

$$Q_{t+1}(x_t, u_t) = Q_t(x_t, u_t) + \alpha\Big(Q_t^* - Q_t(x_t, u_t)\Big), \tag{8.10}$$

again with a learning rate α. We can rewrite (8.10) using the Bellmann equation as

$$Q_{t+1}(x_t, u_t) = Q_t(x_t, u_t) + \alpha\Big(r_t + \gamma \max_u Q_t(x_{t+1}, u) - Q_t(x_t, u_t)\Big). \quad (8.11)$$

Here, the Q-function is updated each time in the direction of the optimal Q-function, approximated by the current reward plus the discounted maximum of the estimated Q-function evaluated at the successor state.

Classification of solution strategies for MDPs

Many more different solution strategies exist to solve MDPs, and it is out of scope to give a comprehensive review. Instead, we want to give some information on how to classify different families of these solution strategies. This provides a foundation to understand the decisions made for the choice of the proposed algorithm.

Using the Bellmann equation, in case the transition and reward functions are known, MDPs can be solved using Dynamic Programming [Bel57]. This is typically regarded as a planning problem. Reinforcement learning deals with the problem of unknown transition functions and the algorithms can be classified according to what is learned:

- the model
- the policy
- the value function / action-value function.

If a model of the transition function is learned, they are named *model-based*, otherwise *model-free* approaches. Another classification is into *value function based* versus *policy based* algorithms. The different approaches are not mutually exclusive, and many hybrid solutions exist. Policy-based algorithms directly store a representation of an approximation to the optimal policy. Value-based algorithms store a representation of the value (either the value or the action-value function). If only a policy and no value function are stored, the algorithms are often referred to as *direct policy search* or *actor-only* (see Wiering and Otterlo [WO12], p. 212).

In the intersection between value-based and policy-based algorithms, there is a family of combined methods that use both strategies. These are named *actor-critic* algorithms. Figure 8.2 illustrates the different families of algorithms for solving MDPs. The areas in grey depict the algorithms regarded as RL. Model-based approaches are commonly only regarded as reinforcement learning when the model is learned during the training phase. The area depicted in white represents algorithms for which both the transfer function and the reward function are known. These can be seen as planning problems, or when run in a closed loop, they fall into the category of classical control algorithms. Adaptive approaches of classical control, especially adaptive MPC, can be seen as borderline to reinforcement learning. In the following, we want to discuss each strategy in more in detail.

Model-based versus model-free reinforcement learning

Methods that exploit the knowledge about the transition dynamics and the reward function are called *model-based*. Once a model is available, the optimal action can be computed using dynamic programming, as done in optimal control. If we do not have full prior knowledge, a model of the system behavior has to be learned. In the control domain, this process is called system identification and is usually done prior to the controller acting on the plant. Supervised learning techniques are mainly applied when neural networks are used to approximate model behavior. This is often referred to as neuro-dynamic programming. As a

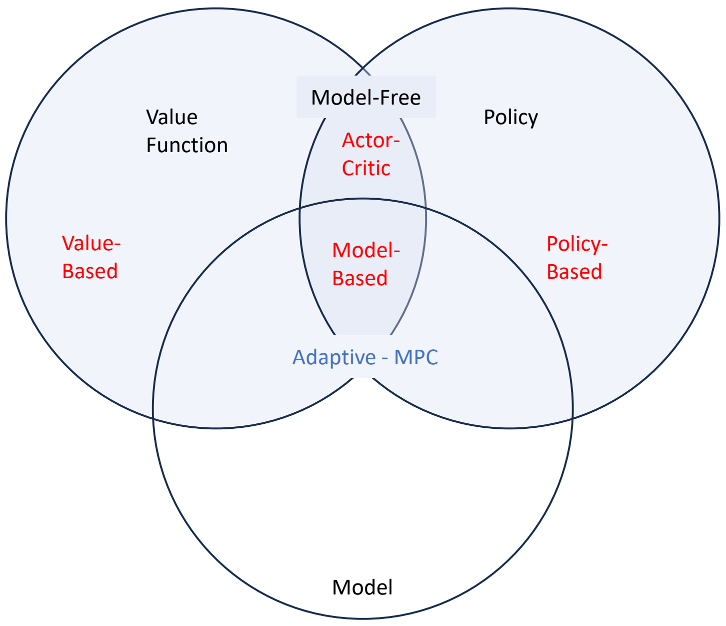

Figure 8.2: Model-based vs. model-free reinforcement learning. The structure of model-based RL is very similar to adaptive MPC if a neural network is used to represent the model of the environment, and the parameters of this model are modified over time. Illustration based on David Silver [Sil15].

distinction, we want to consider one property: in reinforcement learning, the agent typically learns while interacting with the environment. In neuro-dynamic programming, the model is learned during a system identification phase.

The methods that were derived for solving the optimal action can then be used over all domains. They typically have in common that they exploit the model to simulate the reward or cost that specific actions will produce without having visited these states. Contrary to model-based RL approaches, the agent can directly learn the optimal action using *model-free* reinforcement learning - without needing to identify the transition dynamics or its parameters.

Policy based algorithms

Policy-based algorithms directly store a representation of the policy π_θ for all possible states, with θ being the parameters of the policy. Through optimization of θ, the agent wants to learn the optimal policy. π_θ might be stored as a table or approximated using neural network function approximation.

Value function based algorithms

Instead of holding a representation of the policy, value function-based algorithms either learn a representation of the value function $V_\pi(x)$, approximated by $V_\theta(x)$, or the action-value function $Q_\pi(x,u)$, approximated by $Q_\theta(x,u)$. The latter is named *Q-learning*, as already presented in its basic form in (8.11). The action-value function can be directly exploited to select the action as an argument when maximizing the Q-value.

Actor-critic algorithms

Some algorithms store both a representation of the policy (actor) and the critic (which estimates a value function). The critic is then used to provide information on how (in which direction) the actor has to be updated during optimization.

On-policy versus off-policy RL

Another distinction is often made regarding how the policy is derived during learning. On-policy algorithms need to directly apply the action derived from the current policy, while off-policy algorithms can learn from a different policy. The property if a RL algorithm is learning on-policy or off-policy has important practical implications. One of them is the restriction of on-policy algorithms of needing to explore with its current belief of an optimal, stochastic policy - otherwise, no exploration will be possible. Off-policy algorithms allow the use of a deterministic policy, as often desired in continuous control, while exploration can be realized independently of the current policy. Another implication is that off-policy algorithms can perform a learning update with data collected at any time during training. This can be exploited in highly stochastic environments or when data collected recently is correlated, for example, using a concept called *Experience Replay* as used in Mnih, Kavukcuoglu, et al. [MK+13] (see Section 8.2.5).

Approximate dynamic programming and neuro-dynamic programming

In optimal control, dynamic programming is widely used to solve the problem of finding a controller that minimizes the cost function of a dynamical system over time. According to Sutton and Barto [SB18], "dynamic programming is widely considered the only feasible way of solving general stochastic optimal control problems. It suffers from what Bellman called *the curse of dimensionality*, meaning that its computational requirements grow exponentially with the number of state variables. However, it is still far more efficient and more widely applicable than any other general method."

Bertsekas and Tsitsiklis [BT96] originated the term "neuro-dynamic programming" to refer to the combination of dynamic programming and artificial neural networks. Another term currently in use is "approximate dynamic programming". One can also say that adaptive dynamic programming is a subset of or equivalent to reinforcement learning [LV09]. However, the distinct terms arose due to its origins in different research communities: While reinforcement learning is usually used in the machine learning community, approximate dynamic programming is rather used in the control systems community. Naturally, the algorithms and methods within the control systems group were typically derived along mathematically sound stability proofs. Consequently, the systems or environments under investigation usually need to follow some underlying assumptions. These assumptions may lie in that a mathematical description of the system under investigation is available in the first place or that this description falls into a specific type - for example, about linearity or structure of existing nonlinearity. This poses some restrictions. On the contrary, in the machine learning community, very often, one of the main assumptions is that the agent needs to learn to interact with the environment without any prior knowledge about the environment. This aims to be applicable for *any* future tasks and as a step to general Artificial Intelligence (AI) - but may suffer from lack of explainability of certain effects, for example, when no convergence is achieved during learning.

Online versus offline learning

The reinforcement learning setup is generally considered as *online learning*, meaning that either model, policy, or value functions are learned while the agent interacts with the environment. *Offline learning* typically means that the learning phase is performed after data collection previously happened - without the agent interacting with the environment. Some algorithms combine offline supervised learning with reinforcement learning approaches to improve the resulting behavior, create a better starting point, make the exploration phase feasible in real environments, or a combination of all. For an example, see Hwangbo, Lee, et al. [HL+19].

Properties of the environment

Another classification can be done regarding the nature of the state and action spaces of the environment. Many RL algorithms were developed to solve games and simple, academic grid-world examples. They typically can be perfectly modeled by a finite set of discrete states and a set of discrete actions that the agent can perform. For agents acting in the physical world, which intrinsically consists of an infinite set of continuous states, such a simplified model might not sufficiently describe the effects (although it is possible by discretizing the continuous state space). Also, more is needed to select from a discrete set of actions, for example, when an agent tries to solve the inverted pendulum problem. The ability to perform continuous actions (on the torque acting on the pendulum in this example) might be crucial to solve the problem. For discrete state and/or action spaces, representing the necessary functions within a RL setup in tabular data might be possible. If this becomes infeasible due to the vast number of states or because continuous state and/or action spaces are present, neural networks are typically used as representations since they offer the capability of interpolation on one hand or can reduce the storage needed for the parameters.

8.2.4 Metrics for reinforcement learning algorithms

Various metrics exist in order to judge how well an agent behaves. This is useful for benchmarking agents and algorithms or for judging the improvement of an agent during its learning phase. We want to list the most common and important ones in the following:

Performance

The *performance* of a reinforcement learning agent is directly linked to its objective of maximizing return. It is dependent on the definition of the reward and the return. While for the reasons given previously, the discounted return (8.2) might be used during learning, for evaluation, we might want to calculate the un-discounted return. Due to stochastic environments and maybe also stochastic policies, the performance can only be evaluated over a number of validation test runs, which leads to a metric of average performance and its variance. Special care has to be taken when evaluating performance in order to properly benchmark new algorithms since some experiments, especially in deep RL, might be highly sensitive to changes in hyper-parameters, as shown in Henderson, Islam, et al. [HI+18].

Sample complexity

Sample complexity describes how many actions an agent has to perform to learn the optimal policy. If sample complexity is low, an algorithm is called *sample efficient*. Again, due to the stochastic nature of many RL algorithms and the environments they are acting in, sample efficiency can only be measured as the statistics over a broad number of training runs.

Robustness

Robustness of an *algorithm* in RL is often referred to as the capability to learn successfully on a variety of problems without the need for hyper-parameter tuning. In contrast, *robustness of a policy* is the ability to perform close-to-optimal performance under environmental parameter changes. In the control systems community, robustness is typically understood as the ability to achieve a stable closed-loop behavior of a control system under (bounded) uncertainty in parameters or disturbances. In model-based control, the term might be often understood as *robustness to model mis-specification*.

8.2.5 Important reinforcement learning algorithms

It is out of scope to give a comprehensive overview even of the most important algorithms in a field where intense research is happening and ground-breaking new algorithms are published monthly. We still want to mention some prominent examples in the following, classifying according to the attributes described above. We further want to introduce some of the underlying concepts of these algorithms. This helps to understand the decision process for selecting a certain algorithm for the experiments on the system of investigation within this thesis.

Temporal difference algorithms

We already presented Temporal Difference (TD) learning in Section 8.2.3 and the basic Q-learning algorithm in Section 8.2.3. Q-learning also uses the temporal difference error to estimate the Q-function. An example of an extension of the Q-learning algorithm is Double Q-Learning (DQL) [Has10], which is an approach to tackle a problem arising in Q-learning, namely poor performance because of over-estimations of action values. It uses two Q-function estimates to concurrently estimate the maximum of the Q-value with less bias than in Q-learning with a single Q-function. An early algorithm in this group is SARSA [RN94], an extension of the Q-learning algorithm. Instead of building the maximum in the temporal difference error as in Q-learning, the target is built by using another Q-value dependent on a further action step. The name is derived from the tuple needed in the algorithm, generated by the sequence of state-action-reward-state-action. This makes it an on-policy algorithm. Although often seen in a discrete setting, it was initially developed primarily for continuous action spaces and using neural networks as function approximators.

Policy search algorithms

These methods search the optimal policy directly in the policy space. They can be further distinguished into policy gradient methods and gradient-free methods. Policy gradient methods build a noisy estimate of the gradient by sampling from the environment. The family of REINFORCE [Wil92] algorithms is also referred to as Monte Carlo policy-gradient, since it uses full Monte Carlo trajectories to build the gradient estimate. Since policy gradient methods are based on local search, they might get stuck in local optima; at least, they mostly lack theoretical proofs for convergence to global optima. Another family of policy search methods is gradient-free methods. They search for the optimal policy without using gradient information, but by "exploring in the parameter space rather than in the action space"

[MG+18]. Various methods exist, like, for example, Random Search [MG+18] or Genetic Algorithms [SM+17]. Hybrid solutions exist, which also explore parameter space but use the information sampled that way to compute gradients in policy space, like Evolution Strategies [SH+17a].

Model-based approaches

Probabilistic Inference for Learning COntrol (PILCO) [DR11] is a model-based approach to policy search, using Gaussian-Process models for the system dynamics. Maintaining the model allows us to compute the policy gradient analytically for efficient optimization of the policy parameters. PILCO is limited to episodic settings, while value-function-based methods offer the possibility to work in non-episodic environments. Normalized Advantage Functions (NAF) [GL+16] also learns a model but can be rather seen as a hybrid solution (see section 8.2.5).

Combination of model-based and temporal difference algorithms

Many algorithms were proposed on the intersection between model-based and temporal difference algorithms. A fundamental concept for this family of algorithms was presented and named DYNA by Sutton [Sut91]. It concurrently learns a model and performs updates in a model-free setting while additionally using the model to create hypothetical experiences, which are also used to update the value function.

Using deep neural networks for function approximation

The early algorithms were used on small-sized, discrete problems where functions could be represented in a tabular manner. Neural networks were applied to approximate policy or value functions to generalize on continuous state or action spaces. The first deep learning model to learn from pixel input date was probably presented with Deep Q-Network (DQN) [MK+13]. They trained a convolutional neural network to learn a Q-function and applied it to learning to play Atari computer games.

Recent extensions of the DQN algorithm are Dueling Deep Q-Networks (Dueling-DQN) [WS+16] and Double Deep Q-Network (Double-DQN) [VG+16]. The first aims to learn a value and an advantage function in parallel, combining them into a single (dueling) network architecture. The latter transforms the idea of Double Q-Learning DQL [Has10] to learning with deep function approximators. A similar evolution has happened with policy and value function-based methods, combining several approaches into new algorithms.

Delayed target networks

The DQN algorithm [MK+13] also made use of the concept of delayed target networks in order to stabilize learning. Instability arises from how the Q-function is updated in Q-learning. To calculate the update of the Q-function, the maximum value of the (same) Q-function is needed. Having the same value on both sides of the equation leads to instability. To overcome this, a separate copy of the target network is held, which is only updated in a delayed fashion. Various update schemes have been proposed.

Experience Replay

Experience replay most likely goes back to ideas presented in Lin [Lin92]. In combination with neural network approximation, experience replay addresses a difficulty when training neural network representations in Q-Learning with data collected in a local area of the state space. The weight changes have an unwanted effect on the global behavior of the network. This effect can be reduced with data samples taken from the whole area of the state space. For example, Neural Fitted Q-iteration (NFQ) [Rie05] is a particular variant of Q-Learning that combines Experience Replay with a method to speed up learning from [EG+05]. Many other algorithms exist that also adopted the idea of Experience Replay.

As a side note, we want to mention direct links to the work of Concurrent Learning Model Reference Adaptive Control by Chowdhary [Cho10], who works within the adaptive control systems community. Chowdhary could establish stability and convergence proofs for states *and* parameters of systems lacking sufficient excitation by using stored data concurrently with new data. The current limitations of this approach are that it only applies to systems with a specific structure and time-invariant parameters and that no measurement noise is included in the analysis of the stability proofs. A very interesting idea is the data selection scheme, which was proposed by Chowdhary. This method establishes guarantees for the information content in the replay memory to be sufficient for further learning [CJ11], which avoids catastrophic forgetting. See Section 9.3 for more information.

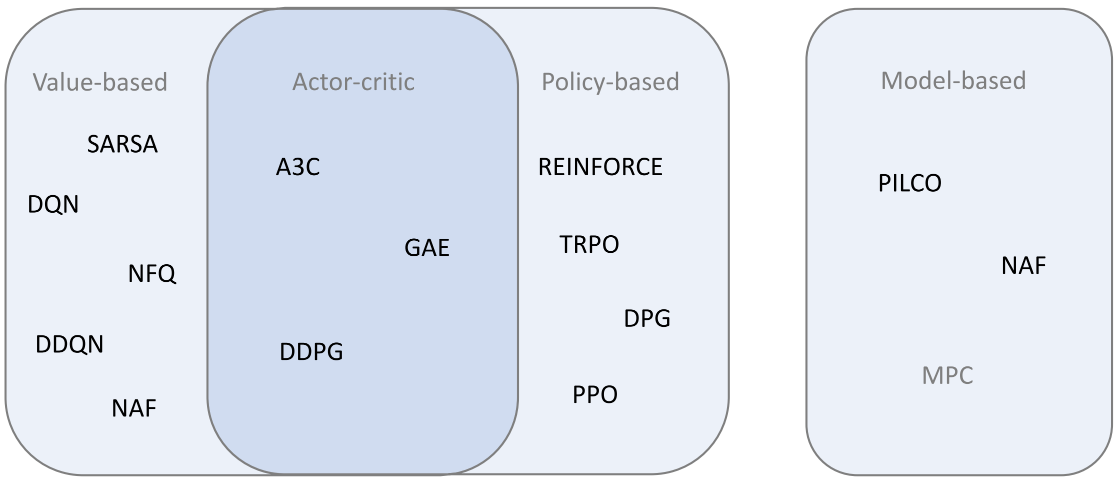

Figure 8.3: Overview over important existing RL methods for continuous control. Illustration inspired by David Silver [Sil15].

Risk awareness in policy improvement

Learning with policy-gradient-based algorithms might be unstable when parameter update steps are too big. This is often intuitively explained as falling from a cliff when hiking to a mountain top. In areas with highly changing gradients, it might be helpful to adjust the step size accordingly. Trust Region Policy Optimization (TRPO) [SL+15] does this with a constraint on the Kullback–Leibler divergence (also called relative entropy) on the step size at each iteration. While this can guarantee a monotonic policy improvement, it comes at a high computational cost. A similar idea with reduced computational complexity was proposed in Proximal Policy Optimization (PPO) [SW+17]. PPO has been found to perform very well on many tasks while being much more straightforward than TRPO, regarding both implementation effort and computational complexity.

Parallelization

Asynchronouos Advantage Actor-Critic (A3C) [MB+16] is an actor-critic algorithm that maintains both a policy and an estimate of the value function. The updates are done using an estimate of the advantage function exploiting asynchronous parallel actor-learners. This parallelization technique dramatically improves learning speed on multi-core CPUs.

Hybrid solutions

Normalized Advantage Functions (NAF) [GL+16] is a continuous variant of the Q-Learning algorithm. It uses the advantage function to improve learning. While the algorithm is off-policy, the original paper proposes incorporating a learned (local linear) model for on-policy imagination roll-outs and shows significant improvements in sample efficiency.

Generalized Advantage Estimation (GAE) [SM+15] is an actor-critic method that uses the advantage function for the critic and applies TRPO for the learning of both the actor and the critic network. The advantage function is learned using the TD(λ) temporal difference error, which provides a way to tune the bias-variance trade-off. Good results were achieved on various continuous control tasks. To learn more on TD(λ), see for example Sutton and Barto [SB18], p. 292.

Deep Deterministic Policy Gradient (DDPG) [LH+15] can be seen as a combination of the Deterministic Policy Gradient (DPG) algorithm [SL+14] with DQN. It makes use of learning a Q-function with experience replay and a target network to stabilize learning. It additionally uses Batch Normalization [IS15], a technique deployed to stabilize the learning of deep neural networks.

8.2.6 (Deep) reinforcement learning for continuous control

There are some special requirements for reinforcement learning algorithms when applied to continuous control problems.

Formal description

Continuous controllers act in continuous spaces; hence the state space is such that $X \in \mathbb{R}^{n_x}$ and the action space is in $\mathcal{U} \in \mathbb{R}^{n_u}$, with $n_x \in \mathbb{N}$ and $n_u \in \mathbb{N}$ the dimension of the state and the action space, respectively. The transition function T will be a probability density function (see also [WO12], p.209) such that

$$\int_{\mathcal{X}'} T(x_t, u_t, x_{t+1}) dx_{t+1} = p\big(x_{t+1} \in \mathcal{X}' | x_t = x \text{ and } u_t = u\big), \tag{8.12}$$

with $\mathcal{X}' \subseteq \mathcal{X}$. The transitions can also be written as time discrete, time-invariant, general nonlinear, and stochastic control systems with additive noise with the system dynamics function

$$x_{t+1} = T(x_t, u_t) + \omega_T(x_t, u_t), \tag{8.13}$$

with $T : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ being a deterministic transition function and $\omega_T(x, u)$ being a zero-mean (e.g., Gaussian) noise vector with the same size as the state vector. In the same manner, the reward function can also contain noise, such that

$$r_{t+1} = R(x_t, u_t, x_{t+1}) + \omega_R(x_t, u_t, x_{t+1}), \tag{8.14}$$

Table 8.1: Comparison of important (deep) reinforcement learning algorithms

Algorithm	Model-based	Model-free	on-policy	off-policy	Value based	Policy based	Actor-critic	Deterministic policy	State space	Action space	Comment
Q-learning	-	x	-	x	x	-	-	-	D	D	tabular learning
SARSA	-	x	x	-	x	-	-	*-	D	C	on-policy Q-learning
PILCO	x	-	x	-	-	x	-	x	C	C	GP dynamics model
REINFORCE		x	x	-	-	x	-	-	D,C	C	MC policy gradient
NFQ	-	x	-	x	x	-	-	x	D,C	D	Experience Replay + Fitted Q-iteration
DQN	-	x	-	x	x	-	-	x	D	D	Deep networks to play Atari
NAF	x	x	*	x	x	-	-	x	D,C	C	*on-policy imagination rollouts
TRPO	-	x	-	x	-	x	-	-	D,C	C	exact PG step-size limitation
PPO	-	x	-	x	-	x	-	-	D,C	C	efficient PG step-size limitation
GAE	-	x	x	-	-	-	x	-	D,C	C	Advantage actor-critic + TRPO
A3C	-	x	x	-	x	x	x	-	D,C	C	Advantage F. Parallelization CPU
DPG	-	x	-	x	-	x	-	x	D,C	C	Deterministic policy gradient
DDPG	-	x	-	x	-	-	x	x	D,C	C	DPG + DQN

with ω_R also being a zero mean noise term. If ω_T and ω_R are zero, the transition function is deterministic otherwise it is stochastic. The MDP is stationary if T and R are time-independent. Otherwise, it is non-stationary. Continuous control problems can only have continuous state spaces; the action space can be discrete or continuous.

Challenges with continuous control problems

Many reinforcement learning algorithms have been designed for problems with small finite state and action spaces. In such domains, policy and value functions can, for example, be represented as look-up tables. In the continuous control domain, one has to deal with infinite state spaces, and the control signals are within infinite action spaces. This poses a variety of challenges. Discretization of a continuous state space is often done in practice in order to transform a continuous space problem into a discrete one. A naive approach would be to discretize along all state space dimensions in equidistant quantization intervals. A more general approach is *tile coding*. The interested reader is referred to [WO12], p. 214. One major issue with tile coding is that the Markov property might be lost, which might have implications for the feasibility of the approaches for MDPs, for example, that convergence proofs for the MDP case do not hold anymore. While it might be feasible to discretize the state and action spaces for some problems, this inherently will lead to sub-optimal performance. Increasing the number of states by decreasing the discretization step size leads to the problem of the *curse of dimensionality* [Bel57] and often hinders the realization of discrete RL formulations for continuous problems. Function approximation plays an important role when dealing with continuous spaces. In order to apply reinforcement learning to real-world control problems, the agent has to conduct experience in the real world. It might be infeasible to run randomly initialized policies for safety reasons or to not destroy the plant. This poses challenges, especially for tasks and algorithms with high sample complexity. Running experiments in simulated environments before the real-world application requires a precise model of the environment or plant. Deviations between simulation and plant behavior might result in a policy that performs optimally in simulation but does not perform well in the real world. This is often referred to as *Simulation-to-Reality* gap or *Sim-to-Real* gap. Another challenge in continuous control is that the policy has to be computed in real-time. Hence, algorithms need to be computationally efficient. A more detailed overview of challenges is given in Andrew Bagnell [And14].

8.3 Related work

A vast amount of literature exists on continuous control applications in the context of robotics. Besides the examples already given in the background section 8.2.5, see Andrew Bagnell [And14] for a good overview. In this section, we want to give an overview of work done with applications of deep learning and RL in the automotive domain, with a focus on control applications. Again, it would be out of scope to give a comprehensive overview, mainly due to the topic's popularity in recent years. Please also note that this review was performed around 2019/2000 and hence does not contain more recent work. Next, we highlight work on reinforcement learning applied to the longitudinal vehicle motion control problem and further distinguish publications of the (predictive) tracking control problem, as investigated in this thesis. Instead, the interested reader is referred to the surveys from Elallid, Benamar, et al. [EB+22], Kalandyk [Kal21] and Farazi, Ahamed, Barua, and Zou [FA+20], of which the two last-mentioned included a reference to the author's paper [BK18].

8.3.1 Deep (reinforcement) learning for automated vehicle control

Some authors proposed to apply deep learning to directly learn vehicle controls based on raw sensor inputs in a so-called "end-to-end" fashion. These approaches recently gained much publicity by demonstrating impressive modern (deep) learning capabilities. Nevertheless, the overall behavior often shows sub-optimal behavior compared to more modular approaches based on rather "classical" decision-making algorithms. The vital question of incorporating traffic rules into such an approach is currently also open. Despite the recent publicity, probably the first known application of end-to-end learning (in a supervised manner) was demonstrated on the vehicle ALVINN by Pomerleau [Pom89] and goes back to 1989. A prominent and more recent example of the application of deep learning was presented by Bojarski, Del Testa, et al. [BD+16] in 2016. The solution was not based on RL; instead, a Convolutional Neural Network (CNN) was trained in a supervised manner to output the steering commands of the vehicle. Training data was recorded from manual test drives, containing the steering angle and the camera images of three cameras: one centered and two tilted to the left and right. The center camera was matched to the desired steering angle, while the tilted ones were used to incorporate a corrective input.

Chi and Mu [CM17] uses an architecture with recurrent neural networks (LSTM) to include inference on historical states. Bansal, Krizhevsky, and Ogale [BK+18] recently presented an approach called ChauffeurNet for higher-level decision-making/behavior generation of AVs. The examples mentioned above were all based on supervised deep learning. Looking at the approaches based on reinforcement learning, we find that Sallab, Abdou, Perot, and Yogamani [SA+17] surveyed recent advances in deep reinforcement learning and proposed a framework for an end-to-end deep RL pipeline for automated driving. Se-Young Oh, Jeong-Hoon Lee, and Doo-Hyun Choi [SJ+00] proposed a vision-based road-following approach similar to the one presented in [BK+18] but using reinforcement learning in combination with a lane edge feature extraction-based image processing algorithm.

Riedmiller, Montemerlo, and Dahlkamp [RM+07] showed in 2007 that it is possible to learn the steering task of an automated vehicle with reinforcement learning, purely conducted on a robotic vehicle and without prior simulation. They applied their NFQ algorithm [Rie05]. The agent was trained to receive a reward based on the difference between a given target trajectory and location data from a GPS sensor. Kendall, Hawke, et al. [KH+18] claimed to be the first example of deep reinforcement learning on-board of a real automated vehicle. They applied the DDPG algorithm to the lane-keeping task using visual input. Hyperparameter tuning was done in simulation before executing training episodes on an actual vehicle.

Williams, Wagener, et al. [WW+17] developed an information-theoretic MPC controller which exploits a model learned via model-based reinforcement learning. They applied it to make a model-sized car drift through the curve on a dirt track. Preceding work for the steering and acceleration control of an AV performing a power drift to backward park into a parking space used a method based on multiple LQRs [KP+10]. Reinforcement learning work with solutions to the decision-making problem includes Mirchevska, Blum, et al. [MB+17]. They apply a Fitted Q-iteration algorithm in combination with Extremely Randomized Trees as a function approximator to learn high-level decision-making in highway scenarios involving interactions with other road users. They proposed leaving low-level steering and acceleration commands to classic planning and control modules. They later proposed combining RL with formal verification methods in order to achieve safe exploration of the agent in [MP+18].

Wang, Chan, and de La Fortelle [WC+18] applied a deep Q-learning algorithm to the automated lane change problem. Hoel, Wolff, and Laine [HW+18] trained a deep Q-network agent to handle high-level speed and lane change decisions for a truck-trailer combination in a simulated environment. Xu, Tang, and Tomizuka [XT+18] recently combined deep reinforcement learning with an approach from classical robust control for the lateral control of a vehicle. Again, many researchers were and still are exploring RL applications in various fields of vehicle automation, and this list is far from comprehensive.

8.3.2 Reinforcement learning in other automotive applications

We also want to mention some work not (directly) related to *automated* vehicle control. The early work of Frost [Fro96] applied reinforcement learning to dynamic vehicle roll control in 1996. Bischoff, Nguyen-Tuong, et al. [BN+13] proposed to apply reinforcement learning for the position control of a throttle valve of a combustion engine. They applied an algorithm called PILCO, which is in the family of model-based policy search algorithms.

Various authors investigated applying (deep) RL for optimizing the energy management strategy of a hybrid vehicle. Both Lin, Wang, et al. [LW+14] and Qi, Wu, et al. [QW+16] applied the TD(λ) algorithm [Sut88] to the problem. Hu, Li, et al. [HL+18] proposed to use DQN [MK+13], while Liessner, Dietermann, and Bäker [LD+19; LS+19] applied the DDPG algorithm proposed in [LH+15]. Pietquin and Tango [PT12] used RL to learn from human interaction when calibrating a partial ADAS system to reduce the risk of forward collisions. Further publications can also be found in the recent survey paper by Elallid, Benamar, et al. [EB+22]. The work mentioned above only represents a tiny section of publications that appeared to date at the time of writing this section and is far from comprehensive, which would be out of the scope of this thesis.

8.3.3 Reinforcement learning for cruise control

The application of model-free policy gradient reinforcement learning to the Adaptive Cruise Control problem was investigated by Desjardins and Chaib-draa [DC11]. They studied using a discrete control policy and found that it results in oscillating behavior when used in Adaptive Cruise Control scenarios. They proposed to modify their algorithm to allow continuous actions in future work. Wei, Zou, et al. [WZ+18b] presented a supervised reinforcement learning (SRL)-based framework for longitudinal vehicle dynamics control of a CACC system. An interesting approach combining classical control with reinforcement learning was presented by Dai, Li, et al. [DL+05]. They suggested tuning fuzzy controllers using reinforcement learning and applied the method to longitudinal vehicle motion control in a Cruise Control scenario with a leading vehicle. In contrast to our work, aiming to incorporate predictive information in the approach, this is not possible with the suggested approach due to the nature of fuzzy controllers. Similarly, Wang, Xu, et al. [WX+14] used Least-squares Policy Iteration to tune the parameters of a PI controller. This approach to solving the Cruise Control problem was validated on an experimental vehicle. To adopt the problem of using a discrete action space, they conducted experiments prior to the training phase in order to find suitable candidates for combinations of proportional and integral coefficients. This reduced set of actions was then used in reinforcement learning. While leaving little room for optimization to the RL agent, this requires a considerable tuning effort. Again, due to the nature of a PI controller calculating the control signal, incorporating advance knowledge is impossible. In work presented in [WZ+15], they used supervised learning to learn an automated vehicle's upper level ACC controller.

Also, here, many researchers are active in this interesting field, and this list should be seen as an introduction rather than a comprehensive study. For further reading, the work done by Lin, McPhee, and Azad [LM+19], which compares deep reinforcement learning and MPC for adaptive cruise control, might be an exciting starting point. Again, the recent work of Elallid, Benamar, et al. [EB+22] includes a broad overview of existing publications.

8.3.4 RL for predictive reference tracking and vehicle motion control

Ng, Clark, and Huissoon [NC+08] applied Monte Carlo RL to tune the parameters of a gain-scheduling approach to the lower-level vehicle motion controller. Zhu, Dai, et al. [ZD+17b] applied a method based on a combination of Neural Dynamic Programming and Internal Model Control. The first was applied for the acceleration decision-making, while the latter was used for the acceleration tracking. To the best of the author's knowledge, no application of reinforcement learning for longitudinal vehicle motion tracking has been proposed before the author's publication in [BK18]. Especially a novelty of the proposed approach is the incorporation of advance knowledge about future disturbances and to allow arbitrary reference trajectories.

After the publication of the author's work in [BK18], [PR+19] proposed to apply a FIR filter within the critic network estimating the value function for linear systems with the aim to use it for linear longitudinal output control of a road vehicle. It addresses the problem of partial observability due to unobserved actuator dynamics.

After a personal meeting between the author and Puccetti in Munich in 2018 with discussions about the proposed method in [BK18], the work of [KW+19] from 2019 as well as Köpf, Puccetti, Rathgeber, and Hohmann from 2020 might have been inspired by my proposal. [KW+19] elaborated the idea to include predictive reference trajectories into Qlearning and contributed some very nice theoretical results on the convergence to arbitrary trajectories in a linear setting. In a pre-print version of [KW+19], it was also acknowledged that our work in [BK18] was the first to propose such a framework. Their results were further developed in [KP+20], where the application of the resulting method in a real vehicle was presented. Contrary to the author's work, they proposed a setting in which the agent remains also learning during deployment, which results in an adaptive controller for which they included stability investigations, although under the limiting assumption of a linear system.

Looking at suggested general frameworks for continuous predictive control, [SG16] proposed a model-free predictive controller based on RL. Their approach is limited to a control horizon of length one, which we consider insufficient for our application. It is realized by a fuzzy inference system, which is used together with Q-learning.

8.4 Proposed solution

This section will present the proposed solution after elaborating on the underlying assumptions and a formal problem description.

8.4.1 Assumptions

We assume that an automated vehicle's longitudinal motion tracking controller receives an arbitrary reference speed trajectory over a look-ahead over a prediction horizon of N_p samples from a planning module. In this section, we will name this tracking control module interchangeably by the term *agent*. The advance information of the expected future road slope values is also provided to the longitudinal motion tracking control module with the look-ahead of N_a samples. The actions calculated by the agent are wheel torque demand commands T^d_{we}, which are demanded by a power-train control module. We assume this module is available to the agent and splits the demand values into actuator commands for the power train and brake.

The reinforcement learning agent has the task of realizing trajectory tracking despite unknown actuator dynamics and unknown disturbances. These disturbances include external forces, like wind forces and changes in the rolling resistance, as well as internal ones, like actuator inaccuracies. The advance information containing both the reference trajectory and a known future disturbance is given to the control module, and the agent seeks to maximize the return by following an optimal policy that is able to compensate for actuator delays and optimally tracks the desired trajectory despite the disturbance. To investigate the feasibility of the approach, we want to assume that our environment is static.

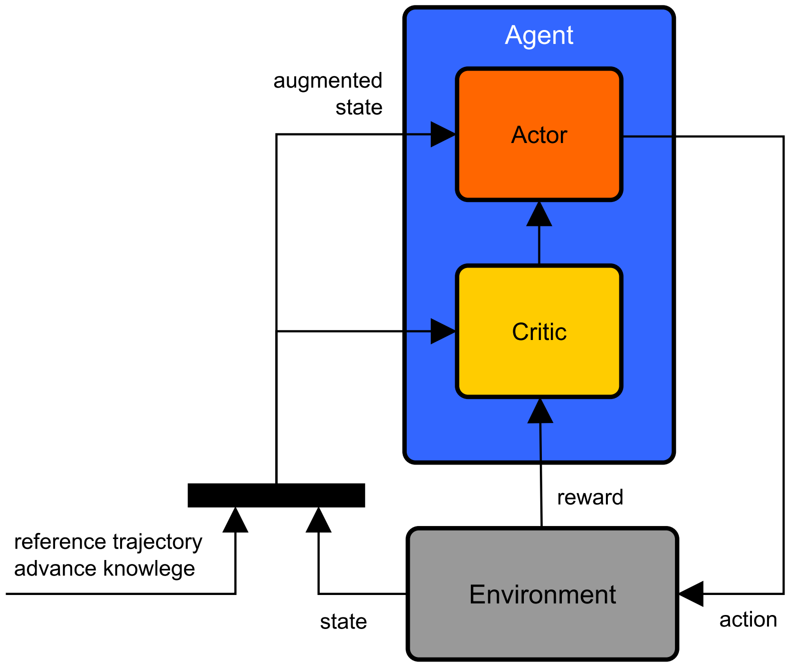

Figure 8.4: The Predictive reinforcement learning Controller architecture enables the incorporation of advance knowledge about desired reference values and expected disturbances. Figure reprinted with permission from the author's publication [BK18]. Copyright ©2018, IEEE.

8.4.2 Formal problem description

We formulate our control system as a Markov Decision Process to solve it via reinforcement learning. As a novelty, we incorporate the advance knowledge into the observation vector.

General formulation of predictive control agent with incorporated advance knowledge

We assume the environment provides access to fully observable state information given by the state vector $x \in \mathcal{X}$. The agent acts by the bounded control $u \in \mathcal{U}$. The target is to track arbitrary but bounded reference values $\rho \in \mathcal{P} \subset \mathcal{X}$, while ρ might contain full reference state information or only at least one of the states. We assume the environment is (output) controllable under the control u. The agent has further access to future reference values given in the reference trajectory vector

$$\rho_{t:t+N_p} = [\rho_t; \rho_{t+1}; \ldots; \rho_{t+N_p}],$$

containing information from the target values ρ for at least one of the environment states at all time steps over the prediction horizon N_p. We can then use $\rho_{t:t+N_p}$ to calculate a trajectory tracking error vector $e_{t:t+N_p}$ according to

$$e_{t:t+N_p} = [e_t; e_{t+1}; \ldots; e_{t+N_p}],$$

with $e_i = \rho_i - x_t$. Additionally, the advance knowledge vector

$$\alpha_{t:t+N_a} = [\alpha_t; \alpha_{t+1}; \ldots; \alpha_{t+N_a}],$$

containing predictive information about known disturbances acting on the environment is assumed to be known over the advance knowledge horizon N_a. In order to incorporate advance knowledge and to let the agent learn a predictive control policy, we compose an augmented observation vector o_t as a concatenation of the state vector x_t, the elements of the trajectory tracking error $e_{t:t+N_p}$ and the advance knowledge vector $\alpha_{t:t+N_a}$ to

$$o_t = [x_t; e_{t:t+N_p}; \alpha_{t:t+N_a}]. \tag{8.15}$$

We further assume that the trajectory tracking problem is feasible and the prediction and advance knowledge horizons are long enough for the agent to stabilize the system. The dimension of the observation vector can be calculated according to

$$\dim(o) = \dim(x) + N_p + N_a. \tag{8.16}$$

We denote a controller using predictive information as described above as Predictive Reinforcement Learning Controller with incorporated Advance Knowledge (PRLC-A).

PRLC-A for longitudinal vehicle motion tracking

We define the vehicle state vector as $x_t = [v_t, a_t]$, with the vehicle's acceleration a_t and speed v_t at time t. We assume to receive a speed reference trajectory with desired speed values over a prediction horizon N_p

$$v^d_{t:t+N_p} = [v^d_t, v^d_{t+1}, \dots, v^d_{t+N_p}],$$

from a higher-level longitudinal motion planning module. We use this information to calculate our trajectory tracking error $e_{t:t+N_p}$ according to

$$e_{t:t+N_p} = [e_t, e_{t+1}, \dots, e_{t+N_p}]^\mathsf{T},$$

where errors at times i are calculated as $e_i = v^d_i - v_t$. Instead of directly using v_d, we calculate the deviation of desired speed values from the actual vehicle speed at each time step over the prediction horizon. We further include advance knowledge information about current and future road slope φ in the advance knowledge vector $\alpha_{t:t+N_a}$:

$$\alpha_{t:t+N_a} = [\varphi_t, \varphi_{t+1}, \dots, \varphi_{t+N_a}].$$

This results according to (8.15) in an observation vector

$$o_t = [v_t, a_t, e_t, e_{t+1}, \dots, e_{t+N_p}, \varphi_t, \varphi_{t+1}, \dots, \varphi_{t+N_a}]^\mathsf{T},$$

while for simplicity we keep the advance knowledge horizon $N_a = N_p$ identical to the prediction horizon. According to (8.16), the length of our observation vector is then given by:

$$\dim(o) = \dim(x) + 2 \cdot N_p = 2 + 2 \cdot N_p.$$

If we want to include predictive information over two seconds with a sample time of 100 ms, this will, as an example, result in an observation vector dimension of

$$\dim(o) = 42 \text{ for } N_p = 20.$$

Action definition

We let our agent perform a one-dimensional action $u_t \in \mathbb{R}^1$ at each time t, which is defined as the desired wheel torque

$$u_t = T^\mathrm{d}_\mathrm{we},$$

which we split into torque demand values for engine and brake according to (8.28) as described in Chapter 7.

Reward function

Our main objective is to minimize the tracking error between reference speed trajectory and resulting vehicle speed for all time steps. So, with a perfect controller, the norm of the difference between the actual and the desired value, given with

$$|v_t^d - v_t|,$$

would be zero for all t (assuming that our reference trajectory is smooth and feasible). While in classical control, quadratic costs are prevalent because they have mathematical properties that can be exploited in many algorithms, Engel and Babuska [EB14] found that quadratic costs have a negative impact on the steady-state error behavior in reinforcement learning. This finding can be explained by the small gradient in the proximity of the origin of quadratic functions. They proposed to use the ℓ_1-norm instead.

We additionally want to penalize high actuator values and, therefore, add a term dependent on u_t. We define the resulting reward as

$$r(x_t, u_t) = -\left(w_{v,r} \cdot |v_t^d - v_t| + w_{u,r} \cdot |u_t|\right), \tag{8.17}$$

with the weighting factors $w_{v,r}$ and $w_{u,r}$ as tunable hyper-parameters to penalize tracking errors and high action values, respectively.

Objective

We previously defined the return in (8.2) as the finite sum of discounted future reward given by:

$$G(\tau_r) = \sum_{t=1}^{N} \gamma^t r_t, \tag{8.18}$$

with the discount factor $\gamma \in [0, 1]$. We now want to use a reinforcement learning algorithm in order to learn a policy $\pi(x)$, which maximizes the expected return $\mathbb{E}(G)$ from the start state. We want this policy to be deterministic in order to be able to deploy it as a controller with reproducible results in deterministic environments, while it may also be stochastic.

8.4.3 Learning and deployment setup

In supervised machine learning, splitting the available data into different sets is common to perform training, validation and test. During training, the networks' performance is validated on the validation data set, not used for training. This helps to detect over-fitting tendencies and tune hyper-parameters. The final network performance is then tested on the test data set.

In RL, especially with on-policy algorithms, the collection of data samples is profoundly connected to learning, and the task to solve is considered as given. This is why, typically, no such separation is present. Hence, in RL, only a split between training and test time is common. In a standard RL setup, the agent interacts during training time with the environment through exploration and exploitation, while the collected data samples are used to update the agent's policy, model, value function, or all of the above. Different formulations exist in which a separation between the exploration, learning, and application phase is made (see, for example [WO12], Chapter 2.2, "The batch reinforcement learning problem", p48ff). The training setup we want to use for our reinforcement learning agent is similar to the growing batch reinforcement learning problem described there but with some additions.

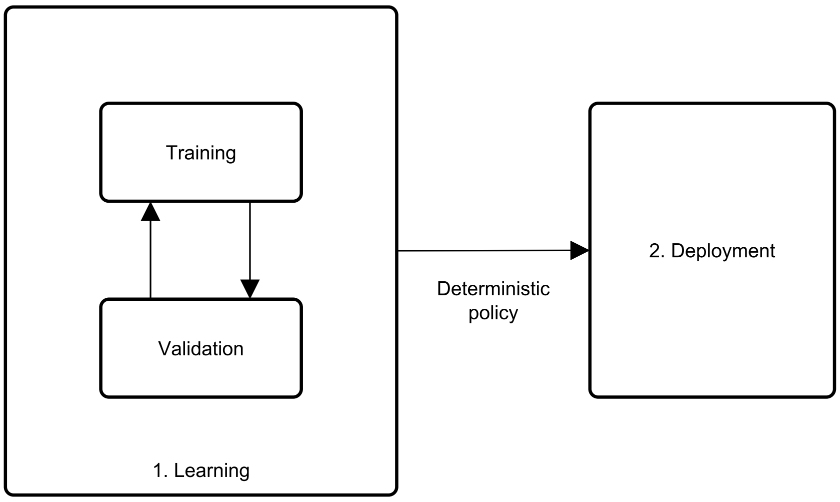

Figure 8.5: Overview of the learning and deployment phase. During learning, various iterations between training and validation are performed. At the end of the learning phase, the parameters of the policy network are frozen, and the deterministic policy will be deployed as a closed-loop controller in the vehicle.

For this feasibility study, we want to distinguish between a learning phase and a deployment (or application) phase (Figure 8.5). See Section 8.7.2 for a discussion about the benefits, drawbacks, and implications of this design choice. During the learning phase, the policy network parameters are updated, while in the deployment phase, the previously trained policy network is used without any further learning.

For the setup of *predictive* reinforcement learning, we suggest further introducing a separation of the learning phase into two parts: A training and a validation setup, similar to best practice in supervised learning. We already learned in Section 8.4.2 that we augment the agents' observations with predictive information. This information can be seen as external input defining the state of the environment. Since we included the reference trajectories in our observations, they have a profound impact on the rewards the agents see and what follows, as in the states visited by the agent. Figure 8.6 should help visualize how the predictive information is an external source to the observations the agent sees during learning. In the training phase, we allow the engineer to choose arbitrary reference trajectory sequences. The agent is trained and improved in a reinforcement learning manner, having to perform actions to explore the environment. As another distinction of the training phase, stochasticity is added into the exploration process by adding exploration noise to the deterministic actions from the policy network. This is necessary for sufficient exploration and common practice for deterministic policy learning. A noise generator block symbolically represents this in Figure 8.6. The training phase is terminated after a given number of training steps.

After a training phase has ended, the performance is evaluated in a validation phase. To make the a comparison of the agents performance possible, two things are necessary. First, no action or exploration noise is added. Second, a fixed set of validation reference trajectories is used instead of arbitrary reference trajectories. Various iterations between the training and validation phases happen during the learning process. With each iteration, the improvement of the agent can be determined in the validation phase. When a saturation of the learning curve has happened, the learning phase is stopped, and the agent with the best performance is selected.

Then, the previously learned agent is used in a deployment setup. With policy or actor-critic agents, the policy can be extracted from the agent in order to used it as a closed loop controller (see Figure 8.7). This has the advantage that inference of a policy network is

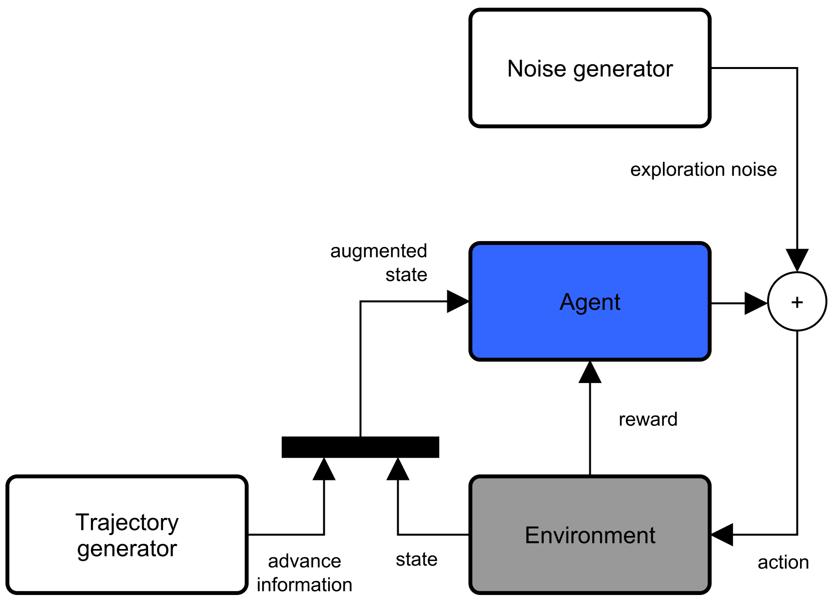

Figure 8.6: Illustration of the training phase as depicted in Fig. 8.5. A trajectory generator provides the reference and advance knowledge trajectories, and action noise is added to a deterministic policy for exploration.

computationally cheap and, as we will show in section 8.6.1, widely independent of the prediction horizon. We want to refer to Section 8.7.2 for a detailed discussion about this design choice's benefits, drawbacks, and implications.

8.4.4 Design of the excitation signal

We just learned that in the suggested learning setup, the engineer has to choose reference trajectories with advance information which are used during the training phase. We are not aware of any existing literature bringing answers to this question published before our own work in [BK18]. As we want to show later, the choice of reference trajectories during the training phase plays a crucial role for learning performance.

One intuitive choice of reference trajectories during the training phase would be to train our agent on the same scenario we want to evaluate. One might also believe that our agent will learn to perform well on this single scenario quickly but perform poorly on a different one which lies outside the state and action space visited during learning due to extrapolation effects. This would be similar to over-fitting in supervised learning. In order to perform well in the whole state space, we also need to visit these states during learning. Covering the whole state space during vehicle testing might be a challenge on test tracks. Considering to perform the training on vehicle test beds gives additional possibilities to the design of advance knowledge trajectories.

To design the trajectories of the reference and the known disturbance, we suggest applying a method from nonlinear dynamic system identification as used in Nelles [Nel01], which is broadly used in the domain of internal combustion engine identification, and propose to use Amplitude-modified Pseudo-Random Binary Sequence (APRBS) [VM+05; DZ11] for the training phase. Next, we want to learn more about this method and the signal type involved.

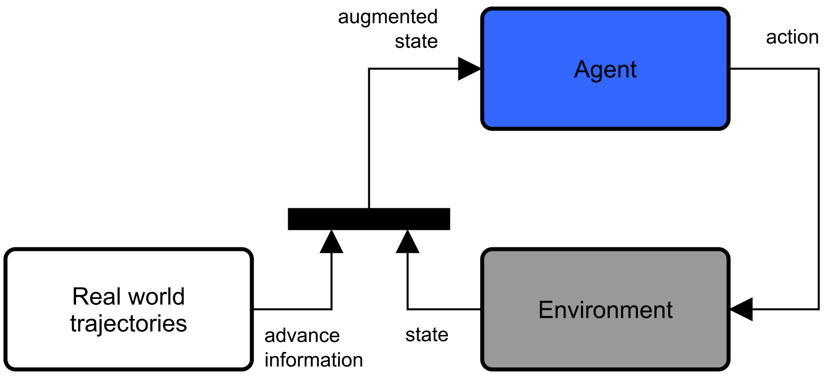

Figure 8.7: Illustration of the validation and application phase as depicted in Fig. 8.5. The reference and advance knowledge trajectories are either validation trajectories or real world data coming from a planning module and a map. No exploration noise is added.

APRBS signals

Amplitude-modified Pseudo-Random Binary Sequences (APRBSs) is a sequence of steps with arbitrary amplitudes and a varying step length. They are derived by creating sequences of variable length from PRBS signals. PRBSs have the property to be generated by a deterministic algorithm, while the statistical behavior is that of a true random sequence in the sense that the elements are uncorrelated. That is the reason for the name *pseudo-random*. PRBS signals can be calculated using linear feedback shift registers (see Sung and Lee [SL03]).

The output of these registers can then be used to trigger a true random number generator, for example, with uniform distribution, to determine the amplitude of the APRBS signal. Details about the calculation of APRBS reference signals are given in Deflorian and Zaglauer [DZ11]. An example of an APRBS originating from a PRBS is given in Figure 8.8.

8.4.5 Considerations for algorithm selection

Many different RL algorithms exist and new ones are presented regularly. This makes the selection of the best algorithm for the predictive longitudinal motion vehicle tracking task challenging. We discussed some important RL algorithms in Section 8.2.5 without the focus of applicability to the longitudinal tracking task. In this subsection, we want to discuss some consideration which led to the selection of the algorithm which was used in [BK18].

Among the various RL algorithmic families, we want to first discuss model-based approaches. One of the key problems of model-based RL is that models introduce model bias; that is, they believe that the learned model is correct and exploit that information, and this leads to sub-optimal behavior. Learning of continuous models has been found intractable in the past for many applications. For example, H. van Hasselt stated: “[. . .] models of continuous MDPs quickly become intractable to solve, making explicit approximations of these less useful” [WO12], p 242.

Exploiting a model might be computationally costly. Although a model can be, as long as it is good enough, beneficial in order to execute planning steps, this comes at a high cost. This cost is the same as we face in MPC algorithms and depends on the prediction horizon. Since we want to achieve long prediction horizons for stability and performance reasons, a way to decrease the computational cost involved is precisely what we try to reduce by using RL as an alternative method.

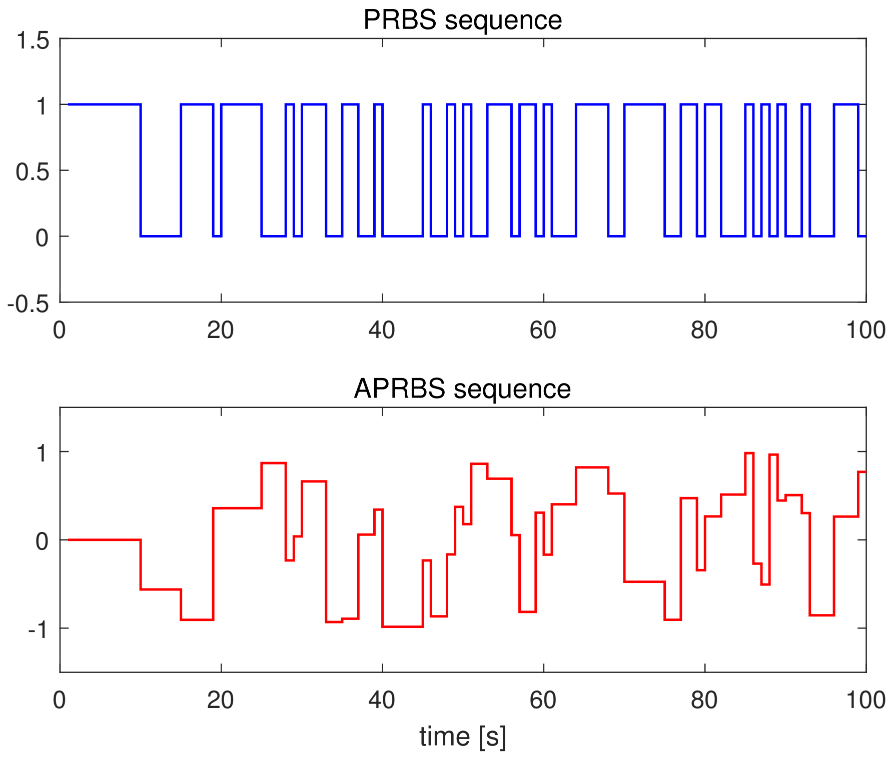

Figure 8.8: Illustration of an Amplitude modified Pseudo Random Binary Sequence (APRBS, below), generated from the Pseudo Random Binary Sequence (PRBS, above). Amplitudes are uniformly distributed random numbers.

Still, PILCO (see section 8.2.5), which learns a Gaussian Process Model, has been successfully demonstrated on continuous control tasks. However, these were all low-dimensional problems, and Nagabandi, Kahn, Fearing, and Levine found that "the most high-dimensional task demonstrated with PILCO [. . .] has 11 dimensions" [NK+17]. Since we want to incorporate advance knowledge over a long time horizon, we have an increased augmented observation space size, which could possibly be challenging for PILCO.

The planning steps within model-based RL frameworks are very similar to the NMPC approach. Also, one question arises when considering a model-based approach for the given problem: why should a model and the model structure be learned at all, when the underlying physics is mostly well known and *available*? But then, the problem reduces to learning the model parameters, and this is exactly what is investigated in the remaining chapters of this thesis. These were the thoughts which led to the decision of rather investigating a model-free approach, but of course, no final answer can be given to the question of which approach will be best.

If we look further into various model-free approaches, we consider the following: In order to realize small sample times, we desire short computation times in order to find the optimal action at each time step. This can be easily realized by maintaining an efficiently computable representation of a policy function. A (deep) neural network representation has this property, and since it stores an approximation of the policy function, it can be directly used in continuous action spaces.

Among the approaches to learning the policy parameters, we find that policy gradient methods, actor-critic approaches, and evolutionary algorithms can be applied. A benchmark of various recent RL algorithms for continuous control was published in Duan, Chen, Schulman, and Abbeel [DC+16] in 2016. According to the authors, the algorithms that performed best over several tasks are the Truncated Natural Policy Gradient (TNPG) algorithm [DC+16], TRPO [SL+15], as well as the DDPG algorithm [LH+15]. Classical algorithms, like REINFORCE [Wil92], suffered from convergence to local optima. Among these, we prefer to obtain a representation of a deterministic policy during learning since we want our controller behavior to be deterministic. Although this could also be realized from stochastic representations and evaluating the mean value, we consider it beneficial to have a deterministic function directly since this is more efficient to store.

8.4.6 Proposed algorithm

Following the considerations above, we want to build our algorithm on basis of DDPG [LH+15]. It is model-free, can learn in an off-policy fashion and belongs to the family of actor-critic networks. It uses Deep Neural Networks (DNNs) as approximators for actor and critic, which enables to learn policies in high-dimensional and continuous state and action spaces. For completeness, we want to provide a short version of the explanation of the algorithm from the original paper [LH+15] in the following before presenting the algorithm with our modifications: DDPG is based on Q-learning and hence uses a representation of the action-value function $Q(x,u)$ which, for a deterministic policy $\mu(x)$ and using the Bellmann equation, can be expressed as

$$Q^{\mu}(x_t,u_t) = \mathop{\mathbb{E}}_{r_t,x_{t+1}\sim E}[r(x_t,u_t)+\gamma Q^{\mu}(x_{t+1},\mu(x_{t+1}))], \tag{8.19}$$

with $r_t, x_{t+1} \sim E$ denoting that r, x are sampled from the environment. Since Q^{μ} is only dependent on the environment but not on the policy, it can be learned off-policy, taking actions from a distinct behavior policy β. The action-value function Q is approximated by a DNN with parameters θ^{Q}, which can be updated, minimizing the mean square error given as the loss, sampled from the behavior policy

$$L(\theta^{Q}) = \mathop{\mathbb{E}}_{x_t\sim\rho^{\beta},u_t\sim\beta,r_t\sim E}\left[\left(Q\left(x_t,u_t|\theta^{Q}\right)-y_t\right)^2\right], \tag{8.20}$$

where

$$y_t = r(x_t,u_t)+\gamma Q\left(x_{t+1},\mu\left(x_{t+1}|\theta^{Q}\right)\right), \tag{8.21}$$

and ρ^{β} is the state visitation distribution of the behavior policy β as in [LH+15], (4) and (5). In order to stabilize learning, DDPG applies an idea from [MK+13] and uses replay buffers and a separate target network which calculates y_t. Replay buffers act to randomly choose samples out of the history of collected tuples of starting state, action, target state and reward. This incorporates the idea of letting the algorithm see independent and identically distributed samples, an assumption of the underlying Markov Chain theory, which does not necessarily hold in dynamic environments. The actor policy $\mu(x|\theta^{\mu})$ is updated using:

$$\nabla_{\theta^{\mu}}J \approx \mathop{\mathbb{E}}_{x_t\sim\rho^{\beta}}\left[\nabla_u Q(x,u|\theta^{Q})|_{x=x_t,u=\mu(x_t)}\nabla_{\theta_{\mu}}\mu(x|\theta^{\mu})|_{x=x_t}\right], \tag{8.22}$$

as in [LH+15], (6). This was proven to be the policy gradient in [SL+14]. In order to make the learning components less sensitive to state information of different units and ranges across different environments, a technique called batch normalization [IS15] is applied. To stabilize Q-learning, copies of the actor and critic network are created and used for calculating the target values (Equation 8.21). The weights of these target networks are updated to slowly track the learned networks according to:

$$\theta^{Q'} \leftarrow \tau_Q\theta^{Q}+(1-\tau_Q)\theta^{Q'} \tag{8.23}$$

$$\theta^{\mu'} \leftarrow \tau_{\mu}\theta^{\mu}+(1-\tau_{\mu})\theta^{\mu'}, \tag{8.24}$$

with $\tau_Q, \tau_\mu \ll 1$ being the Polyak-Ruppert averaging factors. The proposed algorithm is given in Algorithm 6) and was published by the author of this thesis in [BK18]. It is an enhancement to the DDPG algorithm [LH+15], modified to include the generation of APRBS reference trajectories for each episode during learning

Algorithm 6 Enhanced DDPG algorithm for tracking control (modified from [LH+15]).

```
1: Randomly initialize critic network $Q(x,u|\theta^Q)$ and actor network $\mu(x|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
2: Initialize target network $Q'$ and $\mu'$ with weights
3:        $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
4: Initialize replay buffer $\mathcal{B}$
5: for episode = 1, M do
6:     Initialize a random process $\mathcal{N}$ for action exploration
7:     Create reference trajectory signal using APRBS Generator
8:     Receive initial observation state $x_1$
9:     for t = 1, T do
10:        Select action $u_t = \mu(s_t|\theta^\mu) + \mathcal{N}$ according to current policy and exploration noise
11:        Execute action $u_t$ and observe reward $r_t$ and new state $x_{t+1}$
12:        Store transition $(x_t, u_t, r_t, x_{t+1})$ in $\mathcal{B}$
13:        Sample a random mini-batch of $N$ transitions $(x_i, u_i, r_i, x_{i+1})$ from $\mathcal{B}$
14:        Set $y_i = r_i + \gamma Q'(x_{i+1}, \mu'(x_{i+1}|\theta^{\mu'})|\theta^{Q'})$
15:        Update critic by minimizing the loss:
16:       $L(\theta^Q) = \frac{1}{N}\sum_i (y_t - Q(x_t, u_t|\theta^Q))^2$
17:        Update the actor policy using the sampled policy gradient:
18:       $\nabla_{\theta^\mu} J \approx$
19:       $\frac{1}{N}\sum_i \nabla_u Q(x,u|\theta^Q)|_{x=x_i,u=\mu(x_i)} \nabla_{\theta_\mu}\mu(x|\theta^\mu)|_{x_i}$
20:        Update the target networks:
21:       $\theta^{Q'} \leftarrow \tau_Q \theta^Q + (1-\tau_Q)\theta^{Q'}$
22:       $\theta^{\mu'} \leftarrow \tau_\mu \theta^\mu + (1-\tau_\mu)\theta^{\mu'}$
23:    end for
24: end for
```

8.5 Evaluation details

To validate the proposed controller, we used real-world data from a driving scenario in a parking garage for the road grade profile to create a realistic disturbance profile. We also created demand velocity trajectories based on these real-world experiments and used these as references to feed a simulation environment. The idea was to emulate that the vehicle should follow a speed profile with different speed levels coming from a planning module, in which higher speeds are requested for longer straight, open passages, and speed is reduced to drive around corners or in tight passages while steep ramps are traversed. The resulting speed and road grade profile is illustrated in Figure 8.13.

8.5.1 Simulation environment 8.1

We use the time-discrete vehicle dynamics model as presented in the author's publication [BK18], which is equivalent to the time-discrete longitudinal vehicle motion model as derived in Section 3.5.4, but using a slightly different notation. Also, the model additionally includes a significantly simplified, stationary power-train model, which is reduced to the power-train efficiency and -ratio given by $T_{we,k} = \eta_{pwt}\, i_{pwt}\, T_{e,k}$. This was done to achieve meaningful values for engine and drag torques. In order to stay consistent with the rest of the formulae in this thesis, we do not restate the identical versions as they were described in [BK18]. Nevertheless, all equations should be mathematically equivalent between this chapter and the ones given in [BK18]; the results shown later are a repetition of the ones already published therein. All the experiments were conducted for this feasibility study under the assumption that no measurement noise or other disturbances, like time-varying parameters, were present. Note that since this has to be expected within a potential real-world application, we do not pose this restriction in our analysis in the remaining chapters of this thesis. Find a summary of the parameters used for all the experiments in this chapter in Table 8.2.

Table 8.2: Simulation parameters used for the experiments presented in Section 8.6

Symbol	Description	[Unit]
m	Vehicle mass [kg]	2000
m_{I}	Mass resulting from power-train inertia [kg]	50
η_{pwt}	Power-train efficiency [-]	0.89
i_{pwt}	Power-train ratio [-]	8.446
r_{eff}	Effective wheel radius [m]	0.3
$T_{\mathrm{e,drag}}$	Maximum negative engine drag torque [Nm]	-20
g	Gravitational constant [N]	9.81
C_{r}	Rolling resistance coefficient [-]	0.015
C_{d}	Aerodynamic drag coefficient	0.4262 [kg/m]
N	Prediction horizon	20
T_{s}	Sampling time for simulation and controllers [s]	0.05
τ_{e}	Time constant of engine torque response [s]	0.15
τ_{br}	Time constant of brake torque response [s]	0.05

Input substitution The formulation as a multi-input problem with the action space defined as

$$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} T^d_{\mathrm{we}} \\ T^d_{\mathrm{br}} \end{pmatrix}, \tag{8.25}$$

with the desired wheel torque values induced by the engine or power-train T^d_{we} and by the brake T^d_{br} poses additional complexity within a RL framework, since the control allocation problem needs to be solved by the agent additionally to the tracking problem. From an economic point of view, we want to avoid having brakes applied unless the vehicle needs to decelerate. Otherwise, the engine must work against the brakes, increasing fuel consumption and abrasion. Clearly, the agent must consider penalties for mutually applying the brakes and commanding power-train torque at the same time to learn that this is an undesired behavior. We want to introduce the following input substitution to solve the control allocation problem already within the problem formulation and reduce complexity for the RL framework. This solution was originally also proposed within a model-predictive control framework in the author's publication [BK16b] and was re-used in [BK18]: Derived from the relation that in steady state operation, it will be that

$$T_{\mathrm{we}} - T_{\mathrm{br}} = T^d_{\mathrm{we}} - T^d_{\mathrm{br}} \qquad \forall \quad \dot{T}^d_{\mathrm{we}}, \dot{T}^d_{\mathrm{br}} = 0. \tag{8.26}$$

We define

$$\bar{u} = T^d_{\mathrm{we}} - T^d_{\mathrm{br}}, \tag{8.27}$$

and want to achieve that the brake is only applied if the negative drag torque from the power-train is not sufficient to achieve deceleration:

$$\left.\begin{aligned} T^{\mathrm{d}}_{\mathrm{br}} &= 0, \\ T^{\mathrm{d}}_{\mathrm{we}} &= \bar{u} \end{aligned}\right\} \quad \bar{u} > T_{\mathrm{we,drag}} \tag{8.28a}$$

$$\left.\begin{aligned} T^{\mathrm{d}}_{\mathrm{br}} &= T_{\mathrm{we,drag}} - \bar{u} \\ T^{\mathrm{d}}_{\mathrm{we}} &= T_{\mathrm{we,drag}} \end{aligned}\right\} \quad \bar{u} \leq T_{\mathrm{we,drag}} \tag{8.28b}$$

with

$$(-)T^{\max}_{\mathrm{br}} \leq \bar{u} \leq T^{\max}_{\mathrm{we}}. \tag{8.28c}$$

Like this, we can achieve to drive the vehicle dynamics to any state given by $\bar{u}$ between the maximum brake torque and the maximum wheel torque induced by power-train. As a drawback of this solution, we should consider limiting certain trajectories to the slower power-train dynamics in case we still want to have positive torque values on the wheel.

Note that the equations above are mathematically equivalent to the ones stated in [BK18], but a different formulation appears due to using T_{we} instead of T_{e}, involving the need for less compact formulae due to the transformation.

8.5.2 Implementation details

The longitudinal vehicle dynamics model, according to the description in Section 8.5.1, was implemented as OpenAI gym [BC+16] environment in Python. The RL agent was trained with a modified baseline implementation from [DH+17] (in Python and Tensorflow), where the modifications are given according to Algorithm 6. The network structure for actor and critic networks are shown in Figure 8.9 and Figure 8.10, respectively. DNNs for actor and critic contained 1 input layer and 2 hidden layers with Rectified Linear Units (RELUs) [GB+11] and a tangens-hyperbolicus layer for the output. We used 300 and 200 units in the first and second hidden layer, respectively. The training was done with the following hyper-parameter settings (see also Table 8.3): We used an Adam optimizer [KB14] for parameter learning, with learning rates of 10^{-4} for the actor-network and 10^{-3} for the critic network.

As proposed in the original DDPG paper [LH+15], we included L_2 weight decay of 10^{-2} for the critic network $Q(x,u)$, and the initialization of the final layer weights and biases were sampled from a uniform distribution $\left[-3\times10^{-3}, 3\times10^{-3}\right]$. The other layers were initialized from uniform distributions $\left[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}\right]$, where f is the number of inputs (fan-in) to the layer.

Remark 3. *We would recommend trying the Xavier initialization [GB10] or the Kaimin initialization [HZ+15] in case of using RELUs for future experiments, especially when using deeper networks. See also information provided in this link:[Lin] for a short introduction.*

We used mini-batch sizes of 64 and a replay buffer size of 10^6 for training. The discount factor was $\gamma = 0.99$ and for the soft target updates we set $\tau_Q = \tau_\mu = 0.01$.

Table 8.3: Hyper-parameters used for Reinforcement learning

Hyperparameter	**Actor Network**	**Critic Network**
Learning rate	10^{-4}	10^{-3}
L_2 weight decay	–	10^{-2}
discount factor γ	0.99	
soft target updates τ_Q, τ_μ	0.01	
Minibatch sizes	64	
Replay buffer size	10^6	
Action noise	Gaussian, $\sigma = 0.02$	

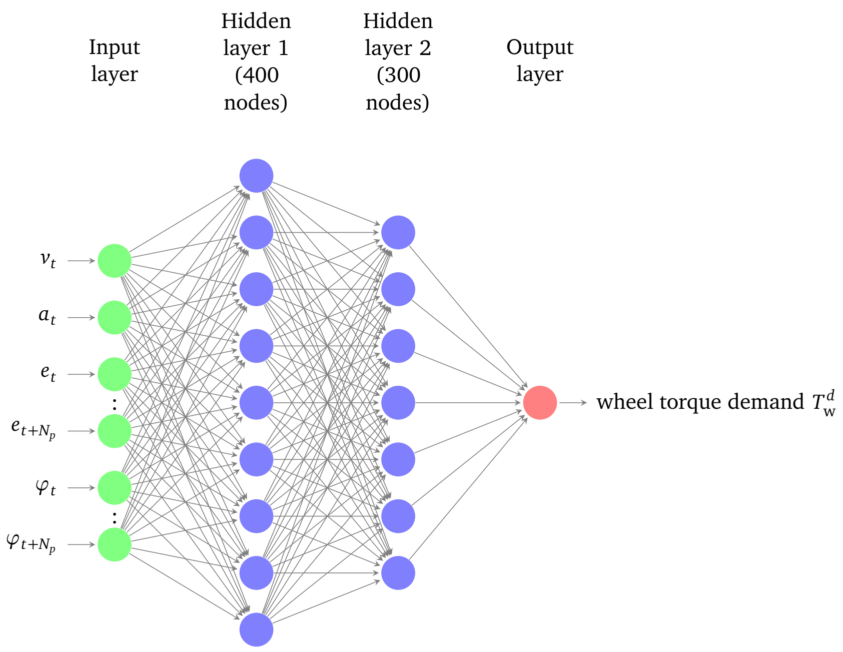
Input layer
Hidden layer 1 (400 nodes)
Hidden layer 2 (300 nodes)
Output layer
v_t
a_t
e_t
e_{t+N_p}
φ_t
φ_{t+N_p}
wheel torque demand T_w^d

Figure 8.9: Schematic structure of Actor network. The number of inputs depends on the length of the prediction horizon.

Exploration noise

In order to allow exploration, we applied action noise during the training phase of the learning process. We achieved good results with Gaussian noise with $\sigma = 0.02$, since learning performance with parameter noise as proposed by [PH+18] did not show good learning performance, but this finding was not analyzed any further.

Remark 4. *Action smoothing is recommended for direct learning in the real world since Gaussian noise makes the actuators jitter at high frequency, potentially damaging the hardware. Thus, temporally correlating the exploration, for example, by using Ornstein-Uhlenbeck action noise [UO30], might be important.*

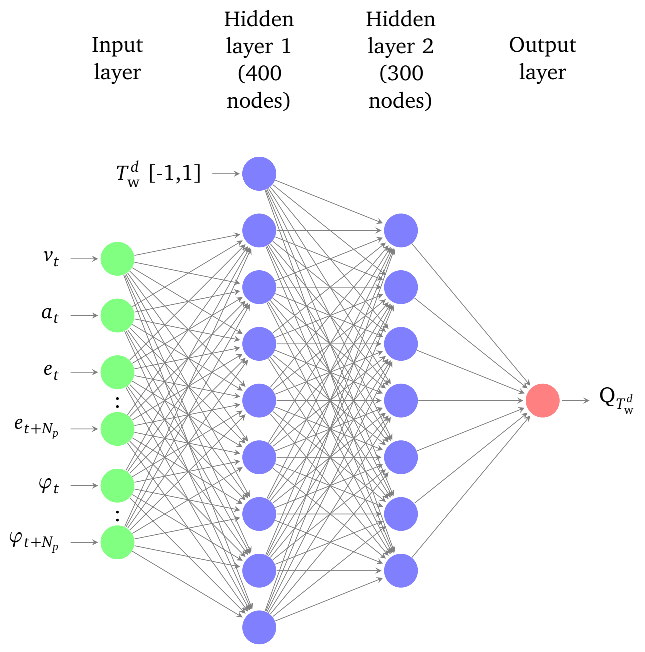

Figure 8.10: Schematic structure of critic network. The number of inputs depends on the length of the prediction horizon. The action value is directly fed into the first hidden layer.

8.6 Results

8.6.1 Comparison of learning speed

We evaluate the learning performance by training a DDPG agent with reference and advance knowledge information from the real-world data set from Figure 8.13. In other words, we let our agent learn by repeatedly driving through a simulated parking garage, and evaluate the performance on the same driving scenario by calculating the evaluation return. We then perform training runs on APRBS sequences and evaluate the parking garage data set as before, and compare the results. Figure 8.11 shows the comparison of the learning curves as a mean over 20 runs with a total of 1M steps with 95 % confidence intervals. The learning speed of our approach using APRBS signals is considerably better than when training on the evaluation data set.

8.6.2 Training using APRBS sequence

We can see a typical APRBS sequence of steps of random length and vehicle speed demand values in Figure 8.12. One can see that the PRLC-A controller learns well to generate smooth transitions to the step demand values, thanks to the predictive nature. Perturbations on the acceleration signal are due to exploration noise, which is necessary during training. Once we observe the agent's performance has converged, we stop training and use the generated policy in a closed-loop manner without any action noise.

8.6.3 Comparison of computation times

A comparison of computation times to calculate the control action of the NMPC approach from [BK16b] with the PRLC-A controller is given in Table 8.4. Times shown are mean

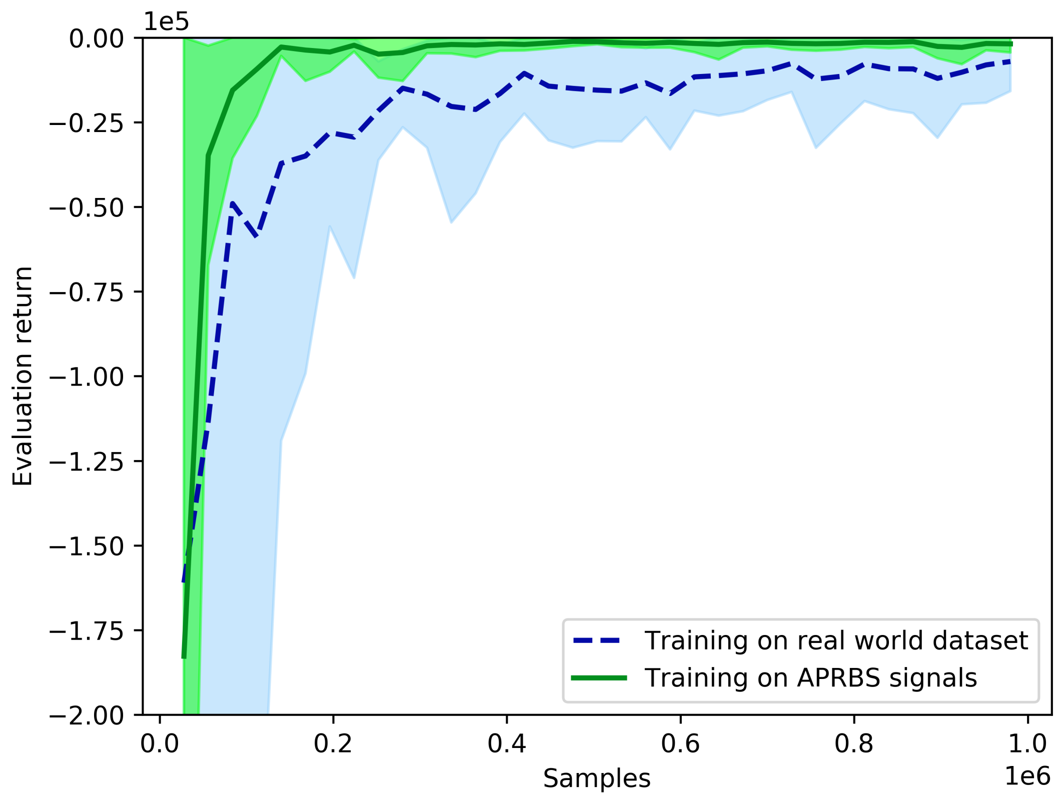

Figure 8.11: Comparison of learning speed when the agent trains on different data sets. The learning performance when learning on APRBS signals, as we propose, is considerably better. Figure reprinted with permission from the author's publication [BK18]. Copyright ©2018, IEEE.

values over 2500 cycles and evaluated on a Laptop with an Intel Core i7-6700HQ CPU with 2.6GHz and 16GB RAM. The NMPC controller was implemented in MATLAB® and the PRLC-A was inference only with Tensorflow on a CPU, so both were not optimized for fast execution. We want to emphasize that a direct and fair comparison of the resulting computation times between the two methods could not be done for several reasons: First, the implementation was done in different languages (MATLAB®, Phyton). Second, various ways exist to formulate the nonlinear program when solving the NMPC problem. For the comparison which was done in [BK18], a single-shooting approach was implemented, which is well known to lead to longer computation times than, for example, a multiple-shooting approach, as demonstrated in the results of Chapter 7.

Also, the development of efficient numeric algorithms to solve the NMPC problem is an area of active research. Torrisi [Tor17], for example, recently presented a method to solve a similar NMPC formulation, which requires $\mathcal{O}\big(N(n_x^2 + n_x n_u)\big)$ Floating-Point Operations (FLOPs) (see p.8). At the same time, they cited an active-set method which requires $\mathcal{O}\big((N n_u)^2\big)$ FLOPs, and an interior-point method exploiting sparsity of the MPC which requires $\mathcal{O}\big(N(n_x^3 + n_x^2 n_u)\big)$ FLOPs (see p.55). Above, N denotes the prediction horizon of the NMPC controller, and n_x and n_u are the state and input dimensions, respectively.

Therefore, we want to list the results from the author's original paper [BK18] regarding the execution times, but instead of providing absolute values, we show percentages of the computation times *relative* to the result with $N = 10$, which was the smallest prediction horizon used. The message of Table 8.4 is that we find the execution time of the PRLC-A controller to be almost insensitive to increased prediction horizons - which is a desirable

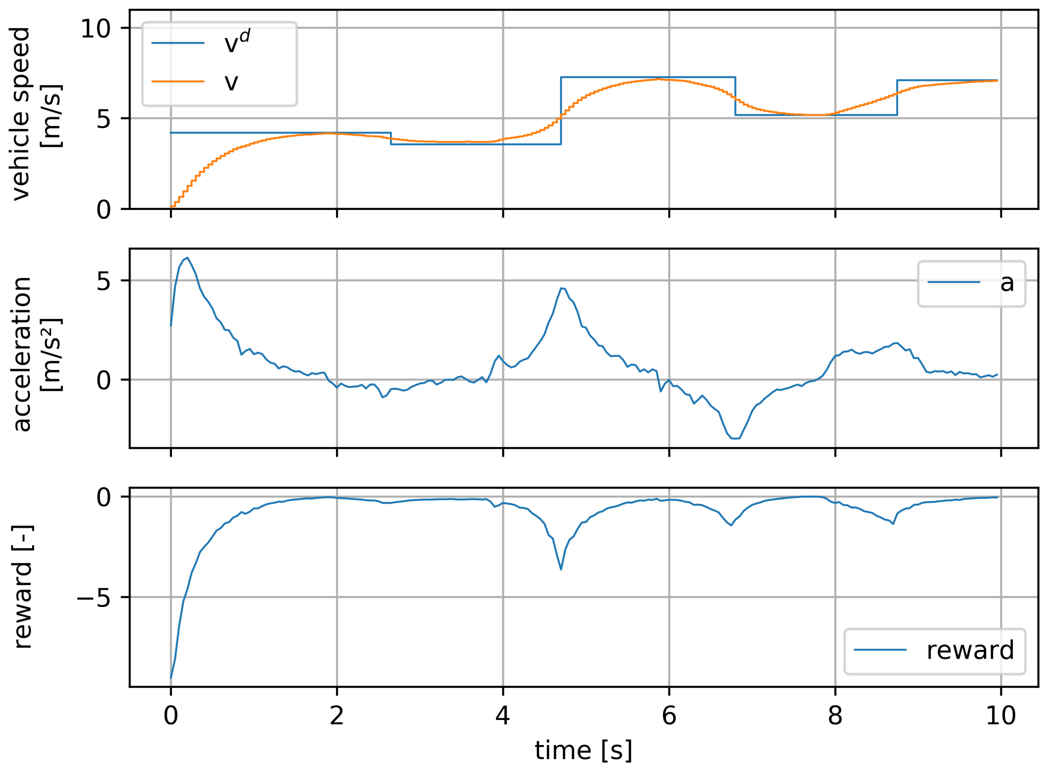

Figure 8.12: Vehicle speed response during learning. The predictive policy learns a smooth response behavior to a sequence of short APRBS step responses. The perturbations on the acceleration a are due to action noise and only occur during learning. Please consider the different time-scale compared to Figures 8.13 and 8.14. Figure reprinted with permission from the author's publication [BK18]. Copyright ©2018, IEEE.

property to realize long prediction horizons.

Table 8.4: Evaluation of relative computation times of PRLC-A compared to NMPC for different prediction horizons. Values given are relative compared to the time obtained for a horizon $N = 10$ for each algorithm.

	Prediction horizon		
Controller	$N = 10$	$N = 15$	$N = 20$
PRLC-A	100.00 %	100.00 %	100.09 %
NMPC	100.00 %	154.99 %	218.60 %

8.6.4 Evaluation of PRLC-A controller

We then evaluated the performance of a resulting controller, which was trained on APRBS signals in comparison to other existing control solutions. First, a standard PI controller was tuned to aperiodic step responses in flat terrain and the relevant speed range to avoid uncomfortable oscillations. Second, the NMPC controller from [BK16b] was evaluated, which includes identical advance knowledge to calculate the optimal control command.

We evaluate the controllers on the parking garage data set from Figure 8.13. Figure 8.14 shows the same evaluation but includes the comparison data. It shows a zoom to the most

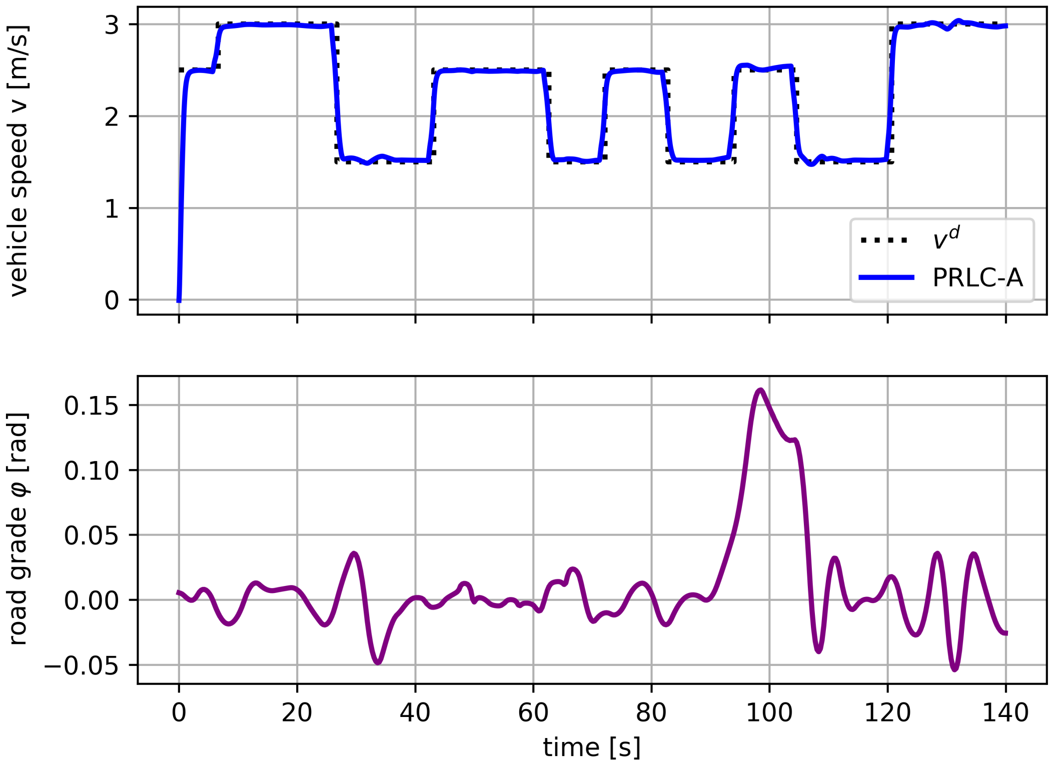

Figure 8.13: Evaluation of a predictive reinforcement learning controller in a real-world example. The reference speed trajectories are taken from a driving scenario in a parking garage. The Predictive reinforcement learning Controller with incorporated advance knowledge about future road grade changes (PRLC-A)performs well in tracking the desired vehicle speed v^d. Figure reprinted with permission from the author's publication [BK18]. Copyright ©2018, IEEE.

critical part between seconds 90 and 115 of the data set when right after climbing up the ramp at increased speed, the vehicle is requested to slow down in order to safely pass a narrow passage at a gate.

We can observe that the PI controller has difficulties coping with the disturbance due to the ramp and produces a big undershoot when decelerating. The PRLC-A controller performs close to the optimal solution of the NMPC controller.

8.7 Conclusion

8.7.1 Summary

It could be demonstrated that it is possible to incorporate advance information about future disturbances and achieve a predictive, reinforcement learning-based tracking controller. The predictive longitudinal vehicle motion tracking problem served as an example to empirically demonstrate the capability of such an approach.

A novelty appearing in the proposed predictive RL setting is the need to design both the trajectories to be tracked and the information about future disturbances that are given to the agent during learning. This appears when learning in a simulated environment, a common setting for training reinforcement learning agents.

A significant finding was that the advance information design has a substantial impact

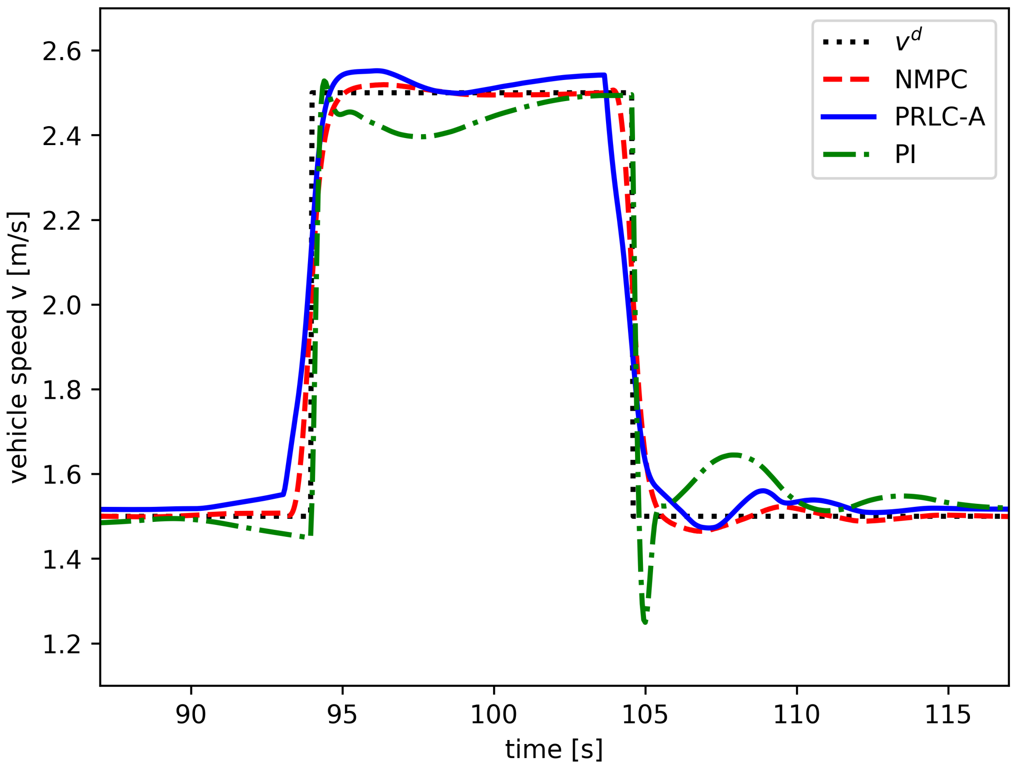

Figure 8.14: Comparison of tracking performances of the proposed PRLC-A controller with results from a PI controller and a NMPC controller. The Predictive reinforcement learning Controller shows a performance close to the optimal solution of the NMPC. Figure reprinted with permission from the author's publication [BK18]. Copyright ©2018, IEEE.

not only on the learning speed but also on the resulting tracking performance. The proposed method to use APRBS signals to design the trajectories in such a setting was experienced to substantially improve the sample efficiency during the learning process.

Simulation studies were performed to compare a trained reinforcement learning agent to conventional controllers, namely a PI-control scheme and a nonlinear, model-predictive controller. The simulation studies revealed that RL agents achieved close-to-optimal performance compared to the tracking performance of a model predictive controller and performed better than a PI-controller. Comparing the computational costs of different agents with the ones achieved by a model predictive control scheme, it could be empirically shown that inferring the policy of a RL agents scales well with the prediction horizon used in the controller. While the computational cost of MPC schemes is known to increase, in many cases prohibitively, with the prediction horizon, the neural network inference times were observed to remain widely constant. This makes the proposed approach interesting, especially for applications where large prediction horizons in combination with small sample times are desired or needed. Nevertheless, many obstacles still have to be considered for a practical implementation of the approach. These obstacles will be discussed in detail in the following Section 8.7.2.

8.7.2 Discussion

While the investigations showed that it is, in principle, possible to let an RL agent learn predictive tracking behavior, we already mentioned in Section 8.7.1 that there still seem to be many obstacles in the way in order to successfully apply such a reinforcement learning based control scheme to real-world applications.

One of the challenges to face is that learning rates were observed to be considerably slow. Learning performance in general, which by design is the result of a stochastic process, showed a significantly high variance between the training runs. Also, an untrained agent will likely not be able to track any desired trajectory for a considerably long duration. This has a wide-reaching impact. First, it seems out of reach to perform the learning phase directly in a real vehicle, not without any modifications and enhancements regarding networks and learning algorithm used. With data obtained from a real vehicle, training times necessary for convergence will rather increase due to noise and other stochastic effects present in reality. While one could think of moving the training phase onto a vehicle test bench instead of real roads for safety reasons, this option is very costly and might introduce some modeling errors due to the simulated road - vehicle interaction. Nevertheless, applying transfer learning techniques might facilitate application from simulation to test-bed and then to road.

Still, the difficulties arising when RL based solutions are ported from simulation environments to real-world applications are well known within the machine learning community and commonly termed “simulation to reality gap”, often shorthanded by *Sim2Real* - gap. This poses challenges to the application under investigation, too.

A consequence of the high variance observed between different training runs is that the engineer faces challenges when choosing the best agent. At least, time- and resource-consuming evaluation runs must be performed.

One potentially prohibiting observation was that in some cases, agents started to diverge during training. This observation went in hand with the fact that learning curves generally had a high variance. This effect is also known as catastrophic forgetting and appears with deep neural networks as nonlinear function approximators, which are generally not proven to converge to the optimal solution.

At this point, it is worth discussing another observation made when performing the experiments described in Section 8.6. We already learned that the design of reference trajectories and advance knowledge information given to an agent during learning has a high impact on the sample efficiency and the controller performance. Contrary to first intuition, using identical trajectories during learning and during evaluation test runs did not lead to an “over-fitting” effect: My expectation was that one might observe reasonably good evaluation performance under such a setting. At the same time, the agents eventually might perform much worse on different, previously unseen reference- and advance knowledge trajectories. Instead, as demonstrated, the agent’s performance was reasonably better on *any* given trajectory when using the proposed APRBS signals during training. While these findings were not elaborated on in more detail, we do not want to leave them unmentioned. Retrospectively, this might be intuitively explained by the fact that in order to be able to learn, it is important that the information content provided during the learning process is rich enough. This goes in line with the substantial findings from Chapter A and Chapter 6, that poor excitation during learning does not only prohibit learning but also might lead to diverging learning behavior if no special algorithmic precautions are put in place.

At the time of performing the experiments with RL, the observation of this sometimes catastrophic learning behavior, together with the fact that deterministic behavior under a deterministic environment is a desirable property of any controller in safety-relevant applications, led the author to propose a learning setting as described earlier in Section 8.4.3, with a separation between training phase and deployment phase: During the learning phase, the policy network parameters are updated, while in the deployment phase, the previously trained policy network is used without any further learning. However, using previously trained agents in a deployment phase appears to have further consequences. While such a setting might be desirable when considering safety and homologation aspects due to its deterministic behavior, this entirely removes any adaptive component from the controller. While this might not be restrictive for time-invariant systems, for the given use-case of longitudinal vehicle motion tracking, we stated that one of the major challenges is that both the vehicle mass and the driving resistances are expected to change considerably during regular vehicle operation. This means that an agent's performance would need to be robust enough to such changes.

Unfortunately, some preliminary investigations on the robustness of policies learned with the method described in [BK18], which are not discussed further within this thesis, showed considerably low robustness to parameter variations as well as to external disturbances. As a consequence, while the proposed learning scheme looks viable at first glance or for other use cases, retrospectively, a practical application in the proposed setting to the use-case under investigation seems, at least at present, out of reach. This is why the focus of the discussion in the remainder of this thesis is put to an alternative solution, which will be presented in Chapters 6 and 7. Some more comments on alternatives to this approach are discussed in the discussion of possible future work which can be found in Chapter 9. A discussion including a summary table comparing the proposed RL approach to a model-predictive control scheme can be found in Section 9.2.2.

8.7.3 Contribution

In this section, a summary of the contributions from the work described in this chapter is provided. Note that the main contributions of this chapter were previously published in the author's publication [BK18]. Nevertheless, in addition, we thoroughly discussed the choice of algorithm based on a broad review of literature available for continuous control. Of course, one has to consider that in such a fast-moving field like DRL, where new and sometimes even ground-braking algorithms are proposed on a monthly, if not weekly, basis, this review will most likely already be outdated at the time a reader will read this thesis. Nevertheless, describing the decision process that led to the proposed algorithm might help researchers in the future-facing similar questions. The work published in [BK18] was the first to propose using (deep) reinforcement learning to solve automated vehicles' longitudinal vehicle motion tracking problem.

Looking at the algorithmic side of the work presented in this chapter, the major contribution might be the proposition to include advance knowledge information into the RL framework to enable a predictive tracking controller setting. This enables the agent to learn truly predictive policies. To the best of the author's knowledge, no prior work existed before [BK18] to use RL in such a setting. This was later also done by Puccetti, Rathgeber, and Hohmann [PR+19] and his collaborators [KW+19], which might have been inspired by a personal meeting between Puccetti and the author of this thesis prior to their publications. This was partly acknowledged in their first pre-print version of [KW+19].

In addition, some empirical findings related to this proposal can be considered as further contributions. For example, the need for designing advance knowledge trajectories that the agent experiences during learning arises as a novelty in the proposed setup. Here, we could show that the design of advance knowledge trajectories used during training has a major impact on the sample efficiency of the algorithm. Hence, another proposal was to apply a Design of Experiments (DOE) method known from systems identification literature, namely using APRBS signals to design the trajectories that need to be tracked during training in simulation. This could reduce training times by orders compared to tracking references taken from a “real world” data set. To summarize, the main contributions of the work presented in this chapter can be listed as follows:

- A discussion of existing literature on RL algorithms which are suited for continuous control tasks in general and to solve the longitudinal vehicle motion tracking problem specifically.
- Proposition to apply reinforcement learning to learn a controller for predictive longitudinal motion tracking of automated vehicles. This was published in the author’s work [BK18]. To the best of the author’s knowledge, the application of RL in such a setting was novel at the time of publishing.
- Proposition to incorporate predictive information including advance knowledge about future disturbances, into the framework of DRL. This was published in the author’s work [BK18].
- Suggesting a method for designing the reference and advance knowledge trajectories to be experienced by the agent during learning. This suggestion is empirically shown to improve both sample efficiency and the tracking performance of the resulting agent. This was published in the author’s work [BK18].

9 Conclusion

"The scientific man does not aim at an immediate result. He does not expect that his advanced ideas will be readily taken up. His work is like that of the planter—for the future. His duty is to lay the foundation for those who are to come, and point the way. He lives and labors and hopes."

– Nikola Tesla, Serbo-Austrian inventor,
Co-alumnus Graz University of Technology, 1856 – 1943

9.1 Summary

We investigated predictive methods for longitudinal tracking of automated vehicles, where knowledge about future trajectories can be incorporated to improve control performance. First, we analyzed a model-free approach based on Reinforcement Learning (RL). We showed that, although not typically done in standard RL formulations, one can obtain predictive behavior by including the advance knowledge about future trajectories into the reward function. This posed the question of how to construct trajectories during the interactive learning phase of a reinforcement learning agent, and we proposed an approach that led to improved learning behavior.

As a second approach to achieve predictive tracking, we chose to investigate the performance of a control algorithm within the Model Predictive Control (MPC) framework. In various simulation studies, we showed that the performance of MPC under nominal conditions, where perfect knowledge about the model parameters is assumed, substantially outperformed the one achieved by a baseline controller based on Proportional-Integral (PI) control. However, under realistic assumptions, where vehicle parameters are not known a priori, the baseline controller only showed negligible performance losses, while standard model predictive approach formulations deteriorated substantially. To achieve a result without sacrificing much of the potential performance gains of the nominal case, we suggest combining the nonlinear model predictive tracking controller with a concurrent state and parameter estimator to achieve an adaptive control solution. Due to the importance of the state and parameter estimator, we dedicate a substantial portion of this work to an in-depth analysis of both existing and proposed solutions. We found that the performance of estimators can be substantially improved by performing specific signal smoothing prior to running the estimation algorithms. This insight led to the development of two novel smoothing algorithms based on polynomial smoothing, which outperform existing solutions. This is achieved by including prior knowledge about the interrelation of signals being higher-order derivatives of each other, but without having to add any further knowledge about the process that generates the signals.

To summarize, in this work, we could find answers to the following research questions already posted in the introduction of this thesis. We want to briefly address each of these questions and add some new questions whose answers could be found in this thesis.

Can optimal longitudinal trajectory tracking of time-varying references be achieved for automated vehicles, even in the presence of (1) actuator delays, (2) unknown and time-variant vehicle parameters, and (3) measurement noise? We demonstrated that we could improve tracking performance by incorporating information about future road grade changes and considering time-varying reference trajectories. Using a NMPC scheme, we can consider actuator delays and directly solve the control allocation problem within the tracking module. Thanks to the combination of the controller with a combined state and parameter estimator, we achieve a solution that quickly converges to the optimal one even under parameter uncertainty, time-varying parameters, and measurement noise.

Can modern deep reinforcement learning be successfully applied to the problem of predictive motion tracking in general, and longitudinal vehicle motion tracking in particular? We demonstrated that reinforcement learning could be applied to obtain a vehicle motion-tracking controller. However, many obstacles still need to be addressed to apply the technology safely to real vehicles. We will address the most important ones in the discussion in Section 9.2.2.

Can predictive information about future disturbances be incorporated into the reinforcement learning framework? As a novelty, we also showed that it is possible to incorporate the information on future disturbances and desired trajectories into a reinforcement learning control scheme.

How does a reinforcement learning controller perform compared to the solution obtained by model predictive control? The performance of the proposed adaptive and predictive control scheme was compared to the results, which can be achieved by a model-free RL approach. Under some simplifying assumptions, the reinforcement learning agent can obtain a performance close to the optimal model predictive solution. This can be achieved at a fraction of the computational cost, especially considering long prediction horizons. Nevertheless, the assumptions are somewhat restrictive, and a detailed discussion will be provided in Section 9.2.2.

How robust is a model predictive controller to changes in the model due to time-varying vehicle parameters, and can offset-free tracking still be achieved without sacrificing performance? By combining the model predictive controller with a state and parameter estimator, we achieve an adaptive, nonlinear model predictive controller ANMPC, which can eliminate the tracking offset arising from imperfect measurements of the true state and lacking knowledge about the vehicle parameters.

How and under which conditions can we learn the true unknown and time-varying parameters in an online recursive fashion? We investigated the conditions under which observability or persistency of excitation is achieved. This was done based on different problem formulations which can be used for parameter learning. We found that insufficient excitation is given when trying to follow a constant vehicle speed trajectory, posing difficulties for parameter learning within a Linear-in-Parameters (LiP) framework. We investigated various countermeasures and obtained a robust estimator scheme allowing for combined state and parameter estimation.

The questions above were already posed in the introduction in Section 1.2.1. In addition, new questions arose during this thesis. Find open questions in Section 9.3, which is dedicated to future work. Some answers could be found, which are as follows:

How does the choice of reference trajectories used during training of a reinforcement learning agent that considers predictive information impact sample efficiency? We could show that in the proposed setting, the shape of the reference and advance knowledge trajectories has an impact on the learning rates of the reinforcement learning agent. We then proposed a method that is capable of substantially improving sample efficiency.

Can the performance of parameter and state estimation be improved using signal pre-processing methods like smoothing and filtering? By design, Bayesian state and parameter estimators are able to deal with measurement errors. Nevertheless, if the underlying assumptions are violated, and for example, Errors-in-Variables are present, the performance can degrade substantially. In such cases, additional smoothing and filtering prior to running the estimators can improve (or restore) performance. This insight led to the development of a novel signal smoothing algorithm we presented in Chapter 5.

Can we exploit knowledge about general physical properties of signals in filtering and smoothing algorithms without the need for an explicit physical model? Moreover, can incorporating such knowledge help to improve smoothing performance? We discovered a novel smoothing algorithm, which can be interpreted as a generalization of the well-known SaG smoother and has improved smoothing capabilities in the setting where multiple noisy measurement channels are available, which are derivatives of a base channel. This is the case in the given problem, where we deal with noisy vehicle speed and acceleration measurements. We show that the estimation results can be improved substantially when operating a proposed estimation algorithm on the values obtained from this smoother.

9.2 Discussion

While we have already discussed many findings related to the various topics within each of the preceding chapters, and the reader is invited to read these sections to get an overall picture, here we want to discuss some more overarching aspects. This includes a discussion about the comparison between model predictive control and model-free reinforcement learning for the problem under investigation. In addition, we want to provide an overview of the building blocks of the proposed learning control architecture. This overview is rather generic and can be understood as an outlook on future work for control systems with learning capabilities.

9.2.1 Architecture for learning control

In this subsection, we want to depict and analyze some essential elements of our proposed learning control architecture, which can (1) learn without catastrophic forgetting and (2) provide offset-free tracking. This is an attempt to help future researchers understand some crucial elements that might be needed to build future artificial intelligence systems successfully. We can find an illustration of the proposed architecture in Figure 9.1.

This architecture was partly motivated by recent work of LeCun [LeC22]. However, this might not be obvious initially and is thought of as a starting point for discussing lessons learned during this thesis. As shown in Figure 9.1, a central element naturally occurring

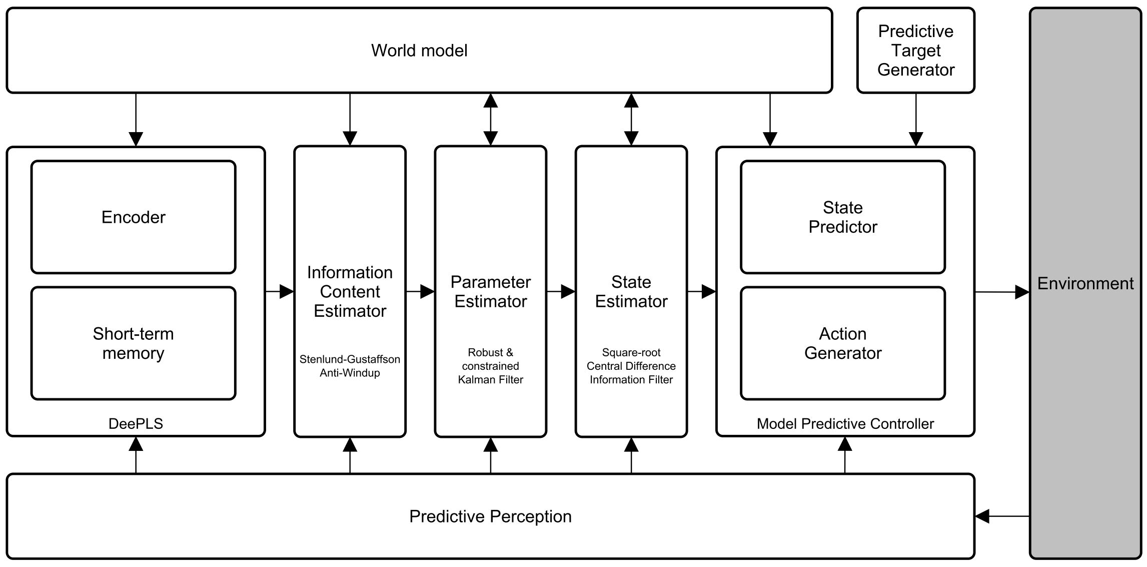
World model
Predictive Target Generator
Environment
Encoder
Short-term memory
DeePLS
Information Content Estimator
Stenlund-Gustaffson Anti-Windup
Parameter Estimator
Robust & constrained Kalman Filter
State Estimator
Square-root Central Difference Information Filter
State Predictor
Action Generator
Model Predictive Controller
Predictive Perception

Figure 9.1: Proposed control and learning architecture depicting some ingredients that might be necessary for any artificial intelligence control module.

in model predictive control is a world model. To optimally interact within its environment, an agent (or model predictive controller) leverages this world model in a state predictor to generate predictive actions. A perception module (bottom) provides predictive information to all other modules. This might not only be observations directly given by the environment but also from stored information and extrinsic sources, which one could interpret as long-term memory. In the case of our tracking control module, we use predictive information in the form of advance knowledge, or future disturbances and hand this to the predictive control module. We also store past observations of past noisy measurements. The moving window smoothing approach used in the DeePLS algorithm (left) can also be seen as short-term memory, on which an encoder operates to obtain richer information than contained solely within the raw measurements. The module also leverages information from the world model, which in our case is the interrelation between measurement channels, to increase the information content produced by the encoder. As such, one can see this as a physically informed module.

The output of the encoder (in our case, the DeePLS smoother) is first fed into an information content estimator. While in our case, this can hardly be recognized as an own module, this might be a crucial part necessary for the overall mechanism to work. In the proposed estimator, the Information Content Estimator is present in the form of the Stenlund-Gustafsson noise covariance adaptation. Essentially, this mechanism extracts the information content of the current sample relevant for the Linear-in-Parameters Kalman filter and adjusts its learning rate by adjusting the process noise covariance matrix. As discussed in detail in Chapter 6, this is essential to avoid parameter wind-up, leading to catastrophic forgetting, and lets the parameter estimator learn only during phases in which rich information content is observed, and prohibits learning in phases of insufficient excitation. Such a mechanism is necessary for learning and adapting the parameters of a world model, which we assume will always be over-parameterized.

The parameter estimator module can leverage knowledge from the world model and act to adjust its parameters. It should be robust and ideally also able to consider constraints, which in a physical world will always be present, and disregarding them could potentially lead to facing singularities. We realized the parameter estimator as an adaptive (using Stenlund-Gustafsson), robust, and constrained Kalman filter.

The next module in our learning-based control architecture is the state estimator, which extracts the current state information. In our case, this was realized by the SRCDIF algorithm.

The MPC module then uses the state information provided by the previous modules and the future target provided by a predictive target generator. We assumed the latter to be existing in the form of a planning module for automated vehicles. Actions are generated by minimizing a cost function of predicted trajectories versus desired trajectories. It might be important to note that for achieving offset-free results, one had to include the target costs into this cost function, which are obtained by imagination of the actions necessary in infinity during steady state operation.

9.2.2 Comparison between RL and MPC approach

One question that might interest the practitioner and the scientist could be: "Which approach should be chosen for similar tracking control problems, or in general? Reinforcement learning or model predictive control? What are the advantages and drawbacks of each approach?". We try to answer these questions in this subsection and provide an overview in a table that compares reinforcement learning control to NMPC and can be found in Table 9.1. This table was partly inspired by a similar comparison, which was given for linear time-invariant systems in Görges [Gör17].

One benefit of any model-free approach is that there is no need to derive a model from first principles to realize a controller. While this is a considerable advantage for problems where a first principles model cannot be derived easily, this was possible for the given problem of longitudinal vehicle control. This benefit alone does not justify applying a reinforcement learning-based approach, mainly since other model-free control algorithms exist, like any PID-based controller. Also, as soon as stability considerations are of interest, one should start deriving a model to be able to perform an analysis. Not including any physically based model also comes with additional drawbacks. One drawback is that not having a model dramatically reduces the interpretability of results. Another one is that knowledge about model parameters can often be exploited for various applications, as shown in the example of the vehicle mass in Section 6.1.1. For state-of-the-art reinforcement learning, where (at least a significant portion) of the training typically happens within a simulation environment, one might argue that in order to provide the simulation environment, knowledge about the environment model needs to exist prior to the application of the approach. The question, why not directly leverage this knowledge in a model-based approach, clearly seems legitimate.

Another advantage of the model predictive approach is that considering constraints is directly supported within the framework, although it might be computationally more expensive. While through the incorporation of barrier functions into the reward function, one might be able to achieve sufficient performance within the RL framework. This is neither straightforward nor can strong guarantees be achieved.

Table 9.1: Comparison of model predictive control to reinforcement learning for continuous control

Property	**MPC**	**RL**	**Comments**
First principles model	–	+	Not required for training of model-free RL variants
Estimation of true physical parameters	++	– –	MPC: in combination with parameter estimation. Cannot be incorporated in model-free RL
Online adaptation to time-varying parameters	+	-	only during RL training, difficult with safety-critical systems
Optimality criteria	+	++	Delayed reward possible in RL
Incorporation of advance knowledge	++	+	Possible in RL with proposed approach
Input / action constraint handling	++	+	RL: only constant time-invariant
State constraint handling	++	– –	Difficult to obtain
Training time	n.a.	– –	
Execution time	–	++	Policy inference in RL independent of time horizon
Interpretability	++	– –	
Offset-free tracking	++	–	Hard to obtain in RL
Stability	mature	immature	
Robustness	mature	immature	

So why would one want to use reinforcement learning instead of model-based predictive control? One benefit of the RL based approach is the reduced computation time during inference, which we also empirically proved to be vastly independent of long prediction horizons. This makes reinforcement learning particularly attractive if long prediction horizons are necessary. Another interesting property seems to be the possibility that in the future, a "one serves all" solution approach might be possible, thanks to the universal function approximator property of the neural networks learning the policy. Still, as of today, many problem-dependent ingredients have to be added to apply the approach successfully. Therefore, more is needed to achieve this vision.

One of the main drawbacks of the reinforcement learning-based approach seems not to be obvious at first. Although RL is an interactive learning procedure, one might think that it would be straightforward to achieve online adaptation to account for an environment or system with time-varying parameters. This does not hold true for many reasons. First, the before-mentioned advantage of an RL agent being able to perform inference on its policy network only holds for the setting where an already trained agent is deployed for control purposes. Including the training also during deployment would substantially increase the computational cost. Furthermore, the most prohibitive factor when wanting to do so is that with today's RL methods, no guarantees can be given about the performance of such a solution. This is especially difficult since, for further learning, any RL agent will find itself in the exploration-exploitation dilemma. Some work in this direction already exists mainly under the term of Adaptive Dynamic Programming (see for example [ZZ+13] and references therein). However, under prohibitively restrictive assumptions, and we want to refer to Section 9.3 for this discussion. A long path in this promising direction seems to be ahead of future researchers. These will have to consider stability, robustness, and feasibility considerations when performing learning directly on real systems to avoid the Sim2Real gap (see also [HL+19] about the Sim2Real gap).

Advances can probably be made best when closing the gap between the different research communities of machine learning on the one hand and the control community on the other. Future solutions should be capable of incorporating problem-specific model knowledge. Additionally, they should allow learning and adaptation of model parameters and unmodeled dynamics, covering effects one cannot model with reasonable effort. Another interesting path is the one of learning cost functions, which is typically referred to as inverse reinforcement learning. This could be helpful in many cases where cost functions cannot be defined that easily but could be inferred from demonstrations.

At this point, it might be of interest to note that other authors have also found that "model-free RL is extremely sample-inefficient" and that "reward is not enough" [LeC22]. Hence, reinforcement learning might only provide a generic solution to some problems in the future. Nevertheless, RL provides many valuable findings, and investigating the problem of machine learning and artificial intelligence from its point of view will help to understand many difficulties, challenges, and solution approaches.

9.3 Future Work

"The best way to predict the future is to invent it."

– Alan Curtis Kay, American computer scientist, *1940

With each solution found while researching this thesis, numerous new questions emerged. Solutions could already be found for some of them, but many still remain unanswered. We want to list the most important ones while providing at least a direction into where one could start searching for answers. While most of the interesting answers can probably be found at the intersection of different research fields, we still want to categorize them according to the main topics covered within this work.

9.3.1 Longitudinal vehicle motion tracking specific topics

Vehicle model and assumptions It would be interesting to validate the proposed approach in a real vehicle. We found that such an evaluation is only possible for a research institute in cooperation with a car manufacturer since more documentation of the sensor readings, especially the virtual sensors, is needed to perform this task with a research prototype vehicle. This is typically considered as protected know-how and not provided to third parties. Also, a modified controller interface would be necessary, but this is hindered by the fact that this would impact the road approval for safe road operation, which is necessary by legal bodies. Hence, real vehicle experiments had to be left to future work.

Next, evaluating the proposed controller and the other higher- and lower-level modules in an automated vehicle would be particularly interesting. This could also be performed in simulation, provided the other control modules are available. In this context, the improvements we expect to see in scenarios like smart intersections could be evaluated.

One motivation to propose the predictive controller architecture was to increase energy efficiency while keeping emissions low, at least for ICE engines. The full potential of this proposal can only be evaluated in a real vehicle and evaluating the interplay with a power-train and engine controller. As already stated in Section 2.2, after the author's work [BK16a] and [BK16b] was published, a study performed by Ma and Wang [MW19] could demonstrate NOx reductions of 13 % by using model predictive engine control on a known driving cycle. The expectation would be to see similar improvement also on *unkown* velocity profiles when combined with the proposed approach of this thesis. Such a combination would be straightforward, thanks to the predictive nature of the proposed tracking module.

It is further of interest to test the robustness of the approach under some additional conditions, which had to be left out in the course of this thesis. These include the presence of wind as an external and unknown disturbance, the presence of heteroscedastic noise, offsets (aka non-zero mean noise) on the torque estimates coming from the power-train module, and plant/model mismatch. Concerning the vehicle model, a further extension could be to include curve resistance forces when driving in curves, as in Radke [Rad13], p 15, and in [Rho16].

Our analysis was based on the assumption that the effect of wind speed and the resulting forces can be neglected. Enhancing the equations and adding wind speed as an additional unknown parameter in the proposed approach is, in principle, possible. Although we did not show these results, we also investigated the observability conditions for a model under the constant parameter assumptions and found this system to be observable. Nevertheless, wind gusts and forces from wind produced by overtaking and oncoming vehicles will show substantial time-varying behavior. It remains open work to investigate how these forces can be estimated or how the resulting disturbances will impose difficulties for robust control. Looking at vehicle-to-vehicle and vehicle-to-infrastructure communication, one could also consider incorporating knowledge from other information sources to improve control. Here, future wind disturbance prediction could be based on meteorological data, which contains spectral densities of wind phenomena, as, for example, given in [Huc08] p. 291.

One effect that could be easily handled by the MPC approach is the presence of actuator lags, which will be present not only because of communication delays between the modules in a real vehicle. Nevertheless, while constant lags could be handled, considering time-varying lags might be challenging.

Also, while in principle, the proposed estimator was designed to track time-varying parameters, investigations had to be left to future work, which brings more insights into this approach's limits. While slowly time-varying or piece-wise constant parameters are quite easily tracked by the proposed estimator, it is still challenging to correctly track the reduction in the aerodynamic coefficient when, for example, approaching the preceding vehicle in platooning scenarios. This is especially limited due to the poor conditioning of the estimator under noise. Here, adding prior knowledge and modeling the reduction in aerodynamic drag based on the distance to the lead vehicle might be feasible.

Another extension could be to correctly model the dependency of road grade within the prediction model from a distance traveled rather than time. So far, it was regarded as sufficient to parameterize the road grade on time, assuming a constant velocity. Suppose this simplifying assumption would prove to be insufficient in real word tests. In that case, one could easily modify the prediction model, but bearing in mind that this would increase the order of the control problem. Nevertheless, including both the traveled distance and the jerk in the equations for the system dynamics would also have other advantages. First, this would enable the formulation of additional state constraints. Like this, one could not only penalize high jerk values indirectly by including the control output differences in the cost function but also directly limit them through state constraints. However, one has to consider that adding additional state constraints increases the computational burden of solving the OCP. Second, there is a possibility of running the controller in a mode in which, for example, traveled distance is the output - which certainly is of interest for the Smart Intersection scenario explained earlier.

Possible enhancements One interesting future path could include prior knowledge about changing parameters. Let us illustrate what we mean by this. The Bayesian filters used in the proposed approach allow the incorporation of prior knowledge in the form of initial error covariance matrices. The choice of these parameters has a considerable impact on the performance of the estimators. For constant parameters, the error covariance will then decrease during learning. We learned that this also decreases the ability to track time-varying parameters or a new parameter value for piece-wise constant parameters. In some instances, one could incorporate knowledge about changing parameters by resetting the covariance matrices to higher values. For example, information like if the vehicle came to a stop and doors were opened will increase the probability that one has to face an immediate change in the true vehicle mass due to passengers and loads entering or leaving the vehicle. On the other hand, it is improbable that the vehicle mass changes abruptly while the vehicle is moving with doors closed. One could also add dependencies on geospatial information, for example, when a vehicle is detected to leave a paved road, increasing the probability that the rolling resistance will change. This could be derived from vehicle GPS information together with map information. Many interesting scenarios could be considered, and solutions could be coded or learned by leveraging artificial intelligence models.

9.3.2 Predictive and adaptive (learning) control

The obtained results motivate to continue and perform further investigations in various directions. This can be to improve the proposed controller further, as well as develop an improved algorithmic foundation for similar control problems.

Unified framework Which unified framework allows the investigation of more general nonlinear systems in the presence of process and measurement noise, which enters in a non-additive manner and also allows the integration (uncertain) knowledge about future disturbances? The recently published work from Powell [Pow19] seems to be a promising starting point in this direction, trying to unify concepts from stochastic optimal control and that of reinforcement learning and trying to bring together existing work from “15 different research communities”.

Adaptive dynamic programming Adaptive dynamic programming (ADP) can be seen as analogous to reinforcement learning but includes rigorous theoretical investigations about convergence and stability. For more details, see, for example [ZZ+13], who stated that “in the field of ADP, a function approximation structure is used to approximate the solution of the Hamilton-Jacobi-Bellman (HJB) equation. The approximate optimal control policy is obtained by using the offline iteration algorithm or the online update algorithm.” Examples of ADP are found in Lewis and Vrabie [LV09], Wang, Liu, and Wei [WL+12] and Wang, Zhao, and Cheng [WZ+18a].

Unfortunately, the discipline still seems immature in that solutions found are typically very restrictive about the systems investigated, especially since these systems are often considered deterministic. For tracking problems, another very restrictive assumption is, for example in Wang, Liu, and Wei [WL+12] “Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach", that the reference trajectory is generated by an exo-system, that means

$$r_{k+1} = \Phi(r_k), \tag{9.1}$$

which does not allow for generic trajectory tracking, as discussed in this thesis. Nevertheless, this is an area of promising active research, where, for example, in Köpf, Ramsteiner, et al. [KR+20] and his recently published thesis [Köp22], some of the restrictions, as mentioned earlier, were already addressed.

Stability considerations While this thesis focussed on the general feasibility and the performance comparison of predictive reinforcement learning with predictive control while trying to do so under realistic assumptions, rigorous proofs of the stability of the nonlinear model predictive controller in combination with the online state estimator are still pending. We deliberately performed investigations in this thesis that are not restricted to concepts that per se allow for stability proofs. We, therefore, included approaches for which a stability proof might be more difficult. We wanted to determine the potential benefits of such an approach, and our results motivate performing control theoretic investigations for the proposed setting. Currently, no existing work for the unrestricted problem is known to the author, as considered within this thesis.

While literature exists, as discussed in Section 7.3.6, which investigates the question of stability of model predictive control schemes, the necessary assumptions for those proofs, to the best of the author's knowledge, do not hold in the problem under investigation. Many stability proofs for NMPC have been presented only for the nominal system, which assumes that no parametric uncertainty nor measurement noise is present. While literature exists investigating the robustness of MPC against these uncertainties, they are typically dedicated to proving that the system response will stay within a specific region around the nominal system. Here, we wanted to consider realistic assumptions regarding the nonlinearity of the system and the measurement noise distributions, including arbitrary reference trajectories and disturbances, while aiming for an optimal solution. The assumption of arbitrary reference trajectories could be restricted to lie in the space of the ones parameterized by, for example, smooth polynomials (see also [KP+20]).

Looking at existing literature, offset-free tracking schemes with rigorous proof for nonlinear MPC have been presented, for example, in [MM12], but were limited to asymptotically stationary references and disturbances. Also, the proof was provided under a deterministic setting without any measurement noise. Another restriction was that only asymptotic stability behavior was investigated, while global closed-loop stability was assumed apriori. They also mentioned that while some robust output feedback formulations exist for linear systems, extending these methods to general nonlinear systems is generally difficult. Next, the approach's disturbance observer must be designed so that observability is always given. In order to achieve this, the authors suggested reducing the dimension of the observed disturbances to be smaller than the system output. While this seems valuable in order to prove stability for constant references, such a formulation yields sub-optimal behavior during transients, and the system's true parameters remain unknown. Also, the approach does not allow for the inclusion of advance knowledge - which is one of the main proposals of this work.

Looking at stabilizing MPC with infeasible reference trajectories, very recently published work is the one of Batkovic, Ali, Falcone, and Zanon [BA+21]. When tracking infeasible trajectories under certain terminal conditions, they proved Input-to-State stability for Linear Time-Variant (LTV) systems. Additionally, they proved that the stability results could be extended for sub-optimal terminal conditions, where the controlled system is stabilized around a neighborhood of the optimal trajectory.

This might be another valuable starting point for further investigations. Also here, the *simultaneous estimation and control* approach, as seen in [CH16; CH17b] could be another promising starting point while combining with ideas from [CJ10b] could provide solutions also for insufficiently excited systems. Clearly, establishing rigorous stability results for general nonlinear systems with time-varying parameters and unmodeled dynamics is an important research direction, and the lack of existing results which are valid under fully realistic assumptions is alarming.

Compliance with safety requirements The longitudinal motion control unit of automated vehicles is a safety-relevant component and must fulfill specific safety requirements to be deployed in production vehicles. Various challenges are posed by this need, which was regarded as out of the scope of this work.

For the proposed model-predictive control scheme, the solver used to solve the NLP must be ASIL-D compliant. To the best of the author's knowledge, no such solver is available that fulfills this requirement for general nonlinear NMPC. This is challenging for many reasons. First, the optimal solution is rarely found with iterative algorithms, especially under run-time restrictions. The impact of the deviation of the solution found to the optimal one is hard to consider. Moreover, with NMPC, the algorithm might get stuck in local minima. Stability proofs of MPC typically rely on the assumption that the solver can find the optimal solution of the OCP. Therefore, the interplay between the disciplines of numerical methods and system theory has to be considered. Hence, this is an essential topic for future research. One paper we want to mention in this context is the one of Diehl and Schlöder [DS03], where the real-time iteration scheme was proposed. This is a scheme with which timing guarantees for embedded systems can be obtained.

Explicit NMPC One solution method which seems very promising is Explicit MPC. Explicit MPC controllers might be easier to validate for meeting ASIL-D requirements since all possible solutions of the OCP are calculated offline in advance and stored in look-up tables [GF+18]. This also means that the numeric calculation does not have to be performed during run-time, which can dramatically speed up computation. Various authors have been proposing the application of explicit linear MPC (e.g., [Joh04] and recently [GF+18] for lateral vehicle control). Results for explicit NMPC also exists and can be found in [BB+16a; Joh04; ZC+16a; NB+08; TM+18; CD+17].

One open question arising from this thesis work is whether it will be possible to include general time-varying advance information in an explicit NMPC scheme. The curse of dimensionality will likely make such an approach very memory-demanding.

Persistence of excitation in machine learning Armed with the knowledge that in adaptive control, persistence of excitation Persistence of Excitation (PoE) plays a crucial role in parameter convergence, this can be exploited in general machine learning problems. Since any machine learning problem can be interpreted as a parameter estimation problem, one needs to make sure that persistence of excitation is also given. This can be used in many different ways. One similar work already following such ideas was just recently published, [SS+21], demonstrating neural network robustness in the sense of withstanding adversarial attacks when a PoE motivated learning scheme is applied for gradient descent learning. Another example is the work of Karg, Köpf, Braun, and Hohmann [KK+21a], which is dedicated to constructing excitation signals that are provably persistently exciting when resulting from transformations by polynomials. This helps Adaptive Dynamic Programming algorithms, which use polynomial function approximators.

Since neural network parameters of certain types of networks can also be learned using Kalman filters (and derivates) [MW01], another interesting direction can be to apply a Stenlund-Gustafsson type anti-windup scheme for these filters for neural network parameter learning. This could also lead to substantially more robust learning when non-sufficiently exciting data is present. Nevertheless, this method is restricted to small networks unless Ensemble methods are used.

Improving computational efficiency In order to further reduce the computational costs of the approach, various possibilities exist which have not been investigated yet. First, the costs of solving the optimal control problem most likely can be further reduced by reformulation of the control problem using a direct collocation method [HP86; DB+06]. Here, an approximation of the problem is parameterized using polynomials, so the integration of the system dynamics is indirectly solved within the optimization problem. This leads to a higher dimensional but sparse problem, which may be faster to solve numerically.

Another possibility would be to formulate the problem using the real-time iteration scheme [Die01; DS03; DB+06; GZ+20], which divides the solution to the problem into a phase that can be pre-computed before the information is available for the full problem. This reduces the effect in practical applications that controls are only available with a specific time lag originating from the time necessary to compute the solution. Also, results from the previous time steps are exploited to warm-start the algorithm, which may result in a solution that is sufficiently accurate only after the first iteration of the numerical optimization of the underlying NLP.

Recently, as already stated in Section 7.3.9, a variety of fast solvers for optimal control problems have been released, which have been reported to be up to a magnitude faster [ZD+17a] than Ipopt [WB06], which we used for our experiments within the CasADi framework. Apart from these, many other solution approaches exist for solving optimal control problems efficiently, which is also worth investigating for the given problem. See, for example, the overview and comparison presented in Biral, Bertolazzi, and Bosetti [BB+16b] and Diehl [Die16].

Deep model predictive control Very recent work published by the group of Girish Chowdhary, who proposed the framework of Concurrent learning within the Model Reference Adaptive Control (MRAC) scheme, combined these techniques with deep neural network learning. In April 2021, they published the paper named *Deep model predictive control for a class of nonlinear systems* [MG+21]. This is an auspicious research direction, combining system knowledge with neural network-based learning of unmodeled system dynamics. Unfortunately, the class of nonlinear systems for which a stability proof could be established is very restrictive with:

$$x_{k+1} = f(x_k) + g(x_k)(u_k + h(x_k)), \tag{9.2}$$

where both $f(x_k)$ and $g(x_k)$ are *known* functions, while the learning and adaptation only work on the matched uncertain and *constant* function $h(x_k)$. Unfortunately, these requirements are not given in the problem under investigation in this thesis.

9.3.3 State and parameter learning

Estimation schemes based on Differential Algebraic Equations (DAEs) Since the longitudinal vehicle model can also be described in the form of a semi-explicit DAE, one could also extend the joint estimation approach to use a variation of Kalman Filters, which are based on DAEs instead of ODEs, like the one proposed in [BR+01b] and [JK+07], [KR+10] or an

Unscented Kalman Filter variant as in [KB+12]. Formulations exist which can also incorporate the algebraic state as a measurement. It remains to be evaluated if this could improve the joint estimation solution such that it even outperforms the one achieved by the proposed dual estimation approach.

Reinforcement learning While we found model-free reinforcement learning to be the more promising approach in this thesis work, in hindsight, model-based RL might have some essential advantages, the most prominent being the increased sample efficiency. Nevertheless, one might ask why, if a model is used at all, one should not also include as much information about the environment as is available to be able to further exploit this knowledge. Hence, it seems consequent to incorporate knowledge about model equations derived from first principles. While this leads to a solution of an adaptive model-based approach, as proposed, many hybrid solutions are possible. For example, the advantage of a policy network regarding inference times clearly could be leveraged. One possibility would be to train a policy network from data provided by MPC. This can be done either offline in the sense of transfer learning or online-adaptive and could lead to feasible approximate controls in cases that permit the computationally expensive calculation of the solution of the optimal control problem. One has to consider that this approach is essentially similar to the one of Explicit MPC, which we already discussed in Section 9.3.2, when talking about safety requirements. A combination of reinforcement learning with NMPC can be used to tune the parameters of an MPC controller via RL. This was, for example, proposed by [GZ20].

Apart from the question of whether RL as such can provide a competitive solution for tracking control problems, many improvements can be obtained to the proposed RL solution. Many new algorithms for RL are presented on a regular basis. Among them are algorithms that can improve sample efficiency and are also suited for continuous control tasks, including SAC [HZ+18] and TD3 [HZ+18].

Concurrent learning approach Another framework considering the case of insufficient excitation was the proposed one of Chowdhary and Johnson [CJ10a] and the subsequent developments in [CJ11; MC+12; CY+13; CW+12; CM+13] on concurrent learning control, which combines parameter estimation in the context of MRAC. Unfortunately, we also found the underlying assumptions too restricted to apply to our problem under investigation. In concurrent learning control, despite insufficient excitation, parameter learning is realized by enhancing the input to the estimator by additional data batches stored in memory. By carefully selecting the data points kept in the memory batch, the convergence of the learning scheme has been proven under certain assumptions. This approach is very similar to the practice of experience replay, which was introduced in the RL community.

Some RL algorithms, like for example DDPG, which was also used, in an enhanced form, for the investigations in Chapter 8, also maintain mini-batches of previous data samples in memory. Learning is performed using a mixture of old and recent data points. Most importantly, the concurrent learning scheme provided theoretical answers about the conditions under which learning is still successful: in principle, this is the case as long as the information content within the memory increases. Hence, carefully selecting and updating the data samples in the memory by monitoring the information content increases the learning rate. The proposed criterion was to maximize the minimum singular value of the information matrix in the data memory. Unfortunately, current restrictive assumptions of the approach prohibited its application to the problem under investigation. The approach has numerous restrictions: first, and quite obviously, learning from old data samples is only meaningful as long as the information in the memory is up to date. This means one can only learn constant parameters, while this thesis aimed to investigate a system with time-varying parameters. Nevertheless, this assumption could be relaxed for the vehicle mass if mass reductions arising from fuel consumption are not present in electric vehicles. Then, it would be sufficient to model the vehicle mass as piece-wise constant. Second, and even more hindering, is that the concurrent learning model reference adaptive control approach was presented only for a very limiting class of nonlinear systems with a certain structure, which is not given in the problem under investigation. Third, as the name suggests, the approach relies on a reference model whose output is tracked by the adaptive control scheme. Apart from the fact that it is hard to find a meaningful reference model, this leads to sub-optimal performance. Another restriction is that the approach was initially derived for the deterministic case only.

Nevertheless, extensions have been proposed recently, and the idea of concurrent learning, not necessarily restricted to the MRAC framework, seems a beneficial one when wanting to learn from insufficiently excited data. Some investigations of combining concurrent learning into the framework of Moving Horizon Estimation (MHE) have been carried out within this thesis but are still ongoing, and this might bring some interesting results. It seems that a lot can be similarly explored in the future as done by [MG+21], considering more general nonlinear models, time-varying parameters, and non-normal measurement noise distributions.

Moving Horizon Estimation Our proposed approach for state and parameter estimation could be enhanced to the one of nonlinear Moving Horizon Estimation (MHE) ([RR+01], [HR05] and a survey in [AB20]). For example, [ZF+13] proposed to apply MHE for the estimation of road-tire friction parameters of automated vehicles. Moving horizon estimators can be seen as the inverse problem of MPC, operating on a moving window of measurement data. They use past control actions as additional inputs to find the initial state at the beginning of the measurement horizon by solving a nonlinear program at each iteration. One advantage of MHE, in analogy to MPC, is that state constraints can be easily considered.

One difficulty with this approach is that to produce a convergent solution, one needs to find a reasonable estimation of the *arrival cost*. This is typically done by penalizing the state estimation at the horizon's beginning with the difference to an estimated value of a converging approximate solution. Often, Kalman filters are used to provide this estimated state for the arrival cost. Hence, MHE does not help directly to stabilize the estimator but will only be able to improve on the solution found by the arrival cost estimator. Another advantage is that since the arrival cost estimator can estimate past states, smoothing instead of filtering is possible, which may improve the solution. For this reason, an estimator based on MHE can be directly combined with the proposed parameter estimator, which also provides values lying in the past. Another advantage of MHE is that state constraints (and parameter constraints in joint estimation schemes) can be easily incorporated. How much the improvements by such an approach justify the additional computational costs needs to be validated. We already implemented a version of MHE during the work of this thesis without finding significant improvements, but we are facing substantial difficulties in overcoming numerical instabilities during phases of insufficient excitation, where the optimization problem becomes ill-defined. A method suited for the case of insufficient excitation was proposed in [SJ11], but it could not resolve these issues for our control problem. For MHE, a converging arrival cost estimator is mandatory, at least for horizons short enough to be feasible for real-time implementation. We found that the small performance gain does not rectify the effort and the high computational cost involved, especially since the time needed to solve the nonlinear program comes on top of the time needed for the arrival cost estimator. With many prospering developments in numerical computation for both MPC and MHE, we find the Moving Horizon approaches still an interesting path to follow for combined state and estimation problems.

Formal convergence guarantees without overly restrictive assumptions How can we formally guarantee parameter convergence for a nonlinear observer while tracking reference trajectories, which include phases of insufficient excitation?

One challenge for rigorous proofs in the context of combined learning and control is that the separation principle does not hold for nonlinear systems. This means more is needed to provide stability proofs for the controller and the observer separately. Answers might be found in a dual framework that combines estimation and tracking, like the ones investigated in *simultaneous estimation and control*, see for example [CH16; CH17b]. The advantage of this approach is that a stability proof can be established more easily when treating learning and control in a unified framework. Unfortunately, it seems that the proofs could only be established so far under the restrictive assumption that the system is always observable.

Probably one of the most important aspects to be investigated in the future is the question of rigorous stability proofs, which also hold under realistic assumptions in the presence of non-normal measurement noise. One difficulty when establishing proofs in probabilistic settings seems to be that these are often under the assumption of a particular given measurement error distribution, while in practice, the true distribution often cannot be determined a priory, or it might change over time due to drift. Another one is that parameters in such frameworks are typically assumed to be random variables drawn from a particular distribution. In most cases, these distributions are of infinite support, which contradicts the knowledge that a physical parameter is bounded and, for example, can only be a positive real number. Other distributions with finite support are not easily mathematically tractable. New approaches should be explored that allow for uncertainty but do not need to be of a strictly probabilistic nature, which often makes a solution intractable. Looking into suggestions proposed by LeCun [LeC22], although for different applications and reasons, abandoning probabilistic approaches could be a starting point for such a journey.

A Background State and Parameter Estimation

"Never memorize something that you can look up."

– Albert Einstein

This chapter presents existing literature and aims to provide the necessary mathematical foundation for the algorithms derived and used for learning and tracking the vehicle parameters in Chapter 6. The content was widely collected from different textbook sources, of which we want to mention Simon [Sim06], Thrun [Thr02] and Brunton and Kutz [BK17], as well as the dissertation of Merwe [Mer04]. First, we want to show different views on the state and parameter learning or estimation problem, like the formulation as joint and dual estimation problems. We do so in Section A.1.

Next, we want to introduce optimal solutions methods to the linear regression problem as an important foundation for many more advance solution techniques, like recursive and stochastic methods like the Kalman Filter. We also discuss underlying assumptions and necessary conditions for convergence of these algorithms, and introduce the terminology of observability and identifiability, together with formal methods to investigate, if one is able to learn the parameters in an online, recursive fashion. We then discuss alternative formulations of the existing algorithms, which not only leads to additional insights, but also provides some less common but (numerically) more robust and more computationally efficient existing solutions. In Section A.9, we talk about existing methods which are robust against measurement outliers. By deriving optimal solutions (or approximations to optimal solutions) to above problems, we want to enable the reader to understand differences and similarities between these methods, and hence allow to easily see connections between them.

What will *not* be covered in this review, is the method of Generalized least squares, since this was applied to vehicle parameter estimation in [RB+14] but found to be less accurate in [Rho16]. Methods using Expectation Maximization (EM) [DL+77] will also not be covered in a broader manner, while the algorithm of Iteratively Reweighed Least Squares can be seen as a special case of EM [DL+77].

A.1 State and parameter estimation problem

First we want to give an informal description of the state and parameter estimation problem:

Problem 2. *Given a system/environment which was influenced at each time by measured system/environment inputs, we want to reproduce all unknown and immeasurable characteristic states and parameters of that system/environment by looking at the evolution in time of what we can observe from that system/environment. Characteristic states and parameters are the ones which are needed to exactly calculate the future evolution of the system/environment, assuming that all future inputs and disturbances to that system would be known.*

We can further reduce this problem to general nonlinear systems, for which a description is available in the form:

$$\Sigma_d : \begin{cases} x_k &= f(x_{k-1}, u_{k-1}, \theta_{k-1}, \omega_{k-1}) \\ y_k &= h(x_k, u_k, \theta_k, \nu_k). \end{cases} \tag{A.1}$$

System Σ_d in equation (A.1) describes the (time discrete, nonlinear) evolution of states x_k at discrete times $k \in \mathbb{N}^+$, in dependency of states x_{k-1} at the previous time step, previous inputs u_{k-1} and parameters θ_{k-1}, of which we assume the latter (for the moment) to be unknown. Additionally, we assume the evolution of the system states is corrupted by disturbance ω_{k-1}. The second equation in (A.1) provides the relation between observations y and system states, inputs and parameters, while the observations are corrupted by disturbances ν.

We want to find estimates of states and parameters, denoted by $\hat{x}_t$ and $\hat{\theta}_t$, which are as close as possible to the true states and parameters. By defining the estimation errors

$$\epsilon_x = (\hat{x}_k - x_k) \tag{A.2}$$

$$\epsilon_\theta = (\hat{\theta}_k - \theta_k), \tag{A.3}$$

we can reformulate this objective into: we want to find estimates such that estimations errors are as close as possible to zero. We can further make this statement more general by defining a stage cost function dependent on the estimation errors of the form

$$\ell_k(\epsilon_{x,k}, \epsilon_{\theta,k}), \tag{A.4}$$

and minimize some cost functional of the form

$$J_{[1:k]}(\ell_1, \ell_2, \dots, \ell_k). \tag{A.5}$$

Now we are able to state the following problem:

Problem 3. *Given the system description Σ_d and vectors containing information about past inputs $u_{[1:k-1]}$ and past and current observations $y_{[1:k]}$, we want to find the evolution of state estimates $\hat{x}_t$ and parameters $\hat{\theta}_t$ for all $t \in [1:k]$ such that the cost $J_{[1:k]}$ related to estimation errors ϵ_x and ϵ_θ is minimized. .*

Clearly, to facilitate (or even allow) a solution to Problem 3, we might want to add some assumptions. These can be of the following types:

1. If information about the evolution of system parameters θ is available, we want to exploit this information.
2. Since the disturbances ω and ν for now are assumed to be unknown, we want to add some assumptions about them.
3. If functions f and h are known and of a simpler, for example linear form, we want to exploit this knowledge

Further, the solutions will depend on the formulation of the stage cost ℓ and the cost functional J_k. We want to discuss different possibilities more in detail. There are two main approaches in the literature of how to solve Problem 3. The first is termed *joint estimation* and the second *dual estimation*. In *joint estimation*, the parameters are considered as additional states and a state space equation for their evolution is formulated. Then, the two state vectors can be combined into an *augmented state vector*, which allows to solve the two problems concurrently based on the augmented state equation. On the contrary, *dual estimation* looks at the problem as two separate, but linked problems: one of estimating the evolution of states, and another of estimating the parameters of the system under investigation. In the following sections, we want to look at these possibilities more in detail.

A.1.1 Joint estimation problem

If we consider the evolution of system Σ_d in equation (A.1) and additionally assume to have some knowledge about the evolution of parameters θ in time of the form

$$\theta_k = g(x_{k-1}, u_{k-1}, \theta_{k-1}, \omega^{\theta}_{k-1}), \tag{A.6}$$

which is a state space description of the "parameter states" θ, we can combine the system state and the parameter states evolution into one *augmented state*, which we define as the vector $\tilde{x} = [x \quad \theta]^{\mathsf{T}}$, and rewrite the system equations as one combined equation

$$\tilde{x}_k = \tilde{f}(\tilde{x}_{k-1}, u_{k-1}, \tilde{\omega}_{k-1}), \tag{A.7}$$

eliminating the parameters from the system equation. This approach reduces the combined estimation problem to a state estimation problem. In the case that our unknown parameters are constant, this facilitates the procedure. We could then write

$$\theta_k = \theta_{k-1} + \omega^{\theta}_{k-1}, \tag{A.8}$$

to express the fact that the parameters are constant. The *parameter noise* ω^{θ} , as the non-zero additive disturbances ω^{θ} are often termed, also allows for some parameter drift. We can further augment the error vector to write

$$e = \begin{pmatrix} \epsilon_x \\ \epsilon_\theta \end{pmatrix} = \begin{pmatrix} \hat{x}_k - x_k \\ \hat{\theta}_k - \theta_k \end{pmatrix} = \hat{\tilde{x}}_k - \tilde{x}_k, \tag{A.9}$$

and perform a combined state and parameter estimation by running an observer on the augmented state $\tilde{x}_k$.

A.1.2 Dual estimation problem

The dual estimation approach differs from the joint estimation approach in that the state and parameter estimation are considered as two separate, but linked problems. While erroneously one could believe it is sufficient to just run one estimator for the states and another estimator for the parameters, inter-dependencies have to be taken into account. Also here, we start with modeling the parameter evolution in state space form, for example as a constant parameter model as in (A.8), but together with an observation model in the form

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \omega^{\theta}_{k-1} \tag{A.10}$$

$$y_k = h(x_k, u_k, \hat{\theta}_k, \nu_k). \tag{A.11}$$

Now in order to combine results from a state estimator running in parallel, using its state estimate $\hat{x}_k$, this leads to

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \omega_{k-1}^{\theta} \tag{A.12}$$

$$y_k = h(\hat{x}_k, u_k, \theta_k, v_k). \tag{A.13}$$

where one has to consider the fact that $\hat{x}_k$ itself is a function of the previous state estimate. This will be discussed more in detail in Section A.8.3, where the Dual EKF is presented.

A.1.3 Parameter estimation problem

If we do not have any unknown states in the sense that states follow some (known) dynamic state evolution, but only observations following some known dependency from parameters and independent variables, we talk about a pure parameter estimation problem. Formally, we write for this general regression problem:

$$\theta_k = \theta_{k-1} + \omega_{k-1}^{\theta} \tag{A.14}$$

$$y_k = h(\phi_k, \theta_k, v_k), \tag{A.15}$$

which is similar to the problems above, but we wrote the measurement equation (or regression equation) in dependency from the independent regressor variable ϕ to highlight the fact that we do not expect it to follow any dynamic evolution. If the regression function is linear of the form:

$$y_k = \phi_k^{\mathsf{T}} \theta + v_k, \tag{A.16}$$

the parameter estimation problem results in a linear regression problem (see section A.3).

A.1.4 Deterministic and stochastic estimation problem

Looking at the disturbances ω and v, we can (assuming the inputs u are deterministic) distinguish between the system Σ_d being *deterministic* and *stochastic*. If the disturbances are deterministic and known, we can treat them as inputs and the problem reduces to a pure deterministic state and parameter estimation problem. If they are deterministic but unknown, we additionally have to solve the *input estimation* problem. If the disturbances are (at least partly) stochastic, we need to solve a *stochastic estimation problem*. One additional helpful assumption for stochastic systems is that the noisy disturbances are zero mean, that is, for $t \to \infty$

$$\frac{1}{k}\sum_{t=1}^{k} v_k \to 0 \tag{A.17}$$

$$\frac{1}{k}\sum_{t=1}^{k} \omega_k \to 0. \tag{A.18}$$

That means that, given an infinitely large history of inputs and observations, we could use this property and instead of directly trying to estimate the states and parameters, we could calculate estimations of the disturbances by letting their means be as close as possible to zero. We could then use the estimates of disturbances at each time step to reconstruct the evolution of states and parameters. We can reformulate this for the realistic case of estimation over a finite history of past inputs and observations by formulating stage costs

$$\ell_k^{v,\omega}(v_k, \theta_k), \tag{A.19}$$

and aiming to minimize some cost functional of the form

$$J^{\nu,\omega}_{[1:k]}(\ell^{\nu,\omega}_1, \ell^{\nu,\omega}_2, \dots, \ell^{\nu,\omega}_k). \tag{A.20}$$

Alternatively, we can use probabilistic formulations to find optimal estimations, assuming that some knowledge about the statistical distribution of the random variables ν and ω is given:

$$\nu_k \sim \rho^{\nu} \tag{A.21}$$
$$\omega_k \sim \rho^{\omega}, \tag{A.22}$$

denoting that ν and ω are drawn from the PDFs ρ^{ν} and ρ^{ω}, respectively.

A.2 Solution approaches to state and parameter estimation

We learned in the previous section A.1 that one way of looking at the state and parameter estimation problem is aiming to minimize some cost functional, which is depending on the estimation error. We also mentioned that another way of formulating the problem is by taking a probabilistic view on the matter. Many different solution approaches have been proposed in the literature. Depending on the assumptions, optimal solutions can be found in some sense, and in certain cases, starting from different viewpoints can lead to the same algorithms.

In this section, we want to give a quick overview of the most common approaches, which helps to understand the algorithms used in this thesis, and of course it would be far beyond the scope of this manuscript to provide a comprehensive overview.

A.2.1 Minimum mean squared error estimation

A common choice for defining the stage costs in optimization problems are sums of quadratic functions. These are easy to compute and result in positiv (semi-) definite and convex cost functionals. In the deterministic case, we can formulate the estimation errors as deterministic variables, and want to minimize the mean squared estimation error

$$MSE = \frac{1}{k}\sum_{t=1}^{k} e_k^{\mathsf{T}} e_k = \frac{1}{k}\sum_{t=1}^{k} \left\{(\hat{\tilde{x}}_k - \tilde{x}_k)^{\mathsf{T}}(\hat{\tilde{x}}_k - \tilde{x}_k)\right\} = \frac{1}{k}\left\|\hat{\tilde{x}}_k - \tilde{x}_k\right\|^2. \tag{A.23}$$

Minimizing the Mean Squared Error (MSE) is the same as minimizing

$$\min MSE = \min \left\{ J_{[1:k]} = \sum_{t=1}^{k} \operatorname{tr}\left\{e_k e_k^{\mathsf{T}}\right\} = \sum_{t=1}^{k} e_k^{\mathsf{T}} e_k = \sum_{t=1}^{k} \left\{(\hat{\tilde{x}}_k - \tilde{x}_k)^{\mathsf{T}}(\hat{\tilde{x}}_k - \tilde{x}_k)\right\} \right\}. \tag{A.24}$$

For problems with stochastic disturbances, the estimations (and hence the estimation errors) become random variables, and we have to take the expectations of the above, and with

$$\operatorname{tr}\left\{\mathbb{E}\left[e_k e_k^{\mathsf{T}}\right]\right\} = \operatorname{tr}\left\{\operatorname{Cov}\left[\hat{\tilde{x}}_k\right]\right\}, \tag{A.25}$$

we can also minimize the trace of the covariance matrix of the state estimation. This is an intuitive optimization criterion, since we want our estimates to have small variances. As we will show in the next sections, minimizing the squared errors also results from a probabilistic viewpoint under the assumption of Gaussian error and noise distributions, and since these are commonly used in practice, the resulting family of algorithms and their underlying mathematical properties are of great importance.

A.2.2 Probabilistic approach

The observed errors ϵ_k can also be viewed as random variables, drawn from a certain probability distribution $p(\epsilon|w)$, parameterized by some parameter vector w. In order to make the presentation less abstract, let us directly assume that the probability distribution follows a normal distribution $p = \mathcal{N}(\epsilon_k|\mu, \sigma^2)$. For independent and identically distributed errors, we can write for the probability of a certain error observation with M samples (see [Bis06], p.26):

$$p\big(\epsilon|\mu, \sigma^2\big) = \mathcal{L}(\mu, \sigma^2) = \prod_{k=1}^{M} \mathcal{N}(\epsilon_k|\mu, \sigma^2) \tag{A.26}$$

The expression on the right is called the *likelihood function* of the normal distribution. It expresses how probable the observed errors are, given the parameters μ, σ^2. Given some observations, we can now infer the parameters of the probability distribution by *maximizing the likelihood function*. This results in the sample mean and the sample standard deviation of the dataset [Bis06], p.27. On the other hand, we might want to find an estimation of the value of another parameter θ, given a model $\epsilon(\theta)$. Again, we can do so by maximizing the likelihood, this time with respect to the parameter θ.

If, as above, the estimation is based solely on the observations which we made, this is termed a *frequentist* approach. This is in contrast to a *Bayesian approach*, named after the mathematician Thomas Bayes (1701–1761), and aims at incorporating knowledge available prior to the inference process. In the Bayesian setting, the uncertainty in the estimated variables is expressed through a probability distribution over θ. Then, by using Bayes' Theorem

$$p_{X|Y}(x|y) = \frac{p_{Y|X}(y|x)}{p_Y(y)} \times p_X(x), \tag{A.27}$$

which is of the form:

$$\text{posterior} \propto \text{likelihood} \times \text{prior}. \tag{A.28}$$

a posterior distribution $p_{X|Y}(x|y)$ of the random variable X can be computed as a function of the observation y and the prior $p_X(x)$. Like this, prior knowledge about the estimated parameters can be additionally incorporated. This can be advantageous but can rise difficulties as well, for example that it might be hard to find informative priors which are also mathematically convenient. A prominent example of an estimator based on the Bayesian approach is the Kalman Filter (see Section A.6).

A.2.3 Maximum likelihood estimation and M-Estimation

Another possibility to derive estimators for state and parameter estimation is to maximize the likelihood within probabilistic frameworks. Since this is typically used in connection with the desire to make estimators more robust against outliers, which produce non-Gaussian error distributions, we will treat this topic within Section A.9, which is entirely dedicated to robust estimation

A.3 Linear regression

A.3.1 Solution of a system of linear equations as an estimation problem

Many complex and nonlinear estimation problems can be broken down into eventually solving systems of linear equations, and therefor we want to use this as a starting point. The problem is finding a solution to the system of linear equations in matrix form, given by [1]

$$y = \Phi\theta, \tag{A.29}$$

where $y \in \mathbb{R}^m$ is a known vector of results or observations, and $\Phi \in \mathbb{R}^{m\times n}$ is a known data matrix. A solution is given by a vector of unknown parameters $\theta \in \mathbb{R}^n$. Solution strategies are well known results from fundamental linear algebra, but since they are essential for the understanding of more complex algorithms, we want to repeat these results in detail. We can distinguish between mainly three cases:

Exact solution

Under the conditions that Φ is a squared matrix of size n, and it is invertible, we can calculate an exact solution by expressing θ explicitly using the matrix inverse and writing

$$\theta = \Phi^{-1}y. \tag{A.30}$$

The condition that Φ is invertible can also be expressed by the rank condition

$$\operatorname{rank}\{\Phi\} = n, \tag{A.31}$$

or that matrix Φ is of full rank since $\dim(\Phi) = n$. This is equal of saying that we need n equations for solving for n variables, and these equations need to be linearly independent. If we have $n = 3$, we could then think of each equation giving a plane in three dimensional space, and if none of the three planes is parallel, we have a unique intersection point.

Under-determined case

In case $\operatorname{rank}\{\Phi\} < n$, the system is under-determined and there is not enough information to provide a unique solution. This can happen if Φ is a rectangular matrix with less than n rows, or some rows are linearly dependent. In the above three dimensional example, if we only had two intersecting planes, and others either missing or parallel to one of the two, we had missing information and the resulting intersection would be a line. Hence, all (infinite numbered) solutions on this line were the solution space of the systems of equations.

If we want to provide one of these infinite solutions, we need to add further assumptions as a selection criterion. A valid and often used selection criterion is to add a cost function dependent on Φ to find a unique solution, while the resulting optimization problem based on that cost function could be given as

$$\hat{\theta} = \operatorname{argmin} \|\theta\|^2 \tag{A.32}$$

$$\text{s.t. } y = \Phi\theta, \tag{A.33}$$

minimizing the norm of θ. Adding additional criteria on penalizing the parameter values in some form is termed regularization, and many approaches exist, but we do not further discuss this here, since they are known to introduce bias into the estimation.

[1] Many textbooks write this in $Ax = b$ notation

Over-determined case

In case $\operatorname{rank}\{\Phi\} > n$, the system is over-determined and it is not possible to provide a solution which exactly fulfills all given equations. Nevertheless, if the reason for having more equations than unknowns is the presence of (possibly Gaussian) noise in a variety of noisy measurements, we can reformulate the problem as

$$y = \Phi\theta + \nu, \tag{A.34}$$

and find the solution which explains the data with the minimum (in a 2-norm sense) error

$$\hat{\theta} = \operatorname{argmin} \|\nu\|^2 \tag{A.35}$$

$$\text{s.t. } y - \Phi\theta, \tag{A.36}$$

which is equivalent to writing

$$\hat{\theta} = \operatorname{argmin} \|y - \Phi\theta\|^2. \tag{A.37}$$

We will discuss the solution of this case in Section A.3.2.

A.3.2 Linear least squares estimation

Let us assume now that we obtained problem (A.29) by taking noisy measurements from a signal which was produced by linear process

$$y_k = \phi_k^{\mathsf{T}}\theta + \nu_k, \tag{A.38}$$

then we will be able to put all measurements taken up to time k into the observation vector $\boldsymbol{y} = [y_1, y_2, \dots y_k]$ containing k available measurements, and the regressor matrix Φ by stacking the regressor vectors $\Phi = [\phi_1; \phi_2, \dots; \phi_k]$, then we will be able to write

$$\boldsymbol{y} = \Phi\theta + \nu, \tag{A.39}$$

with the observation matrix Φ, the state vector θ and with zero mean noise in the vector ν. We want to find the minimum mean squared error solution for the parameter vector $\theta \in \mathbb{R}^n$. This can be done by minimizing the error vector

$$e = \nu = \boldsymbol{y} - \Phi\theta, \tag{A.40}$$

which is the same as "minimizing the noise" (in an ℓ_2-norm sense) leading to

$$\min J = \min\ \|e\|^2 = (\boldsymbol{y} - \Phi\theta)^{\mathsf{T}}(\boldsymbol{y} - \Phi\theta). \tag{A.41}$$

We expand

$$(\boldsymbol{y} - \Phi\theta)^{\mathsf{T}}(\boldsymbol{y} - \Phi\theta) = \boldsymbol{y}^{\mathsf{T}}\boldsymbol{y} - \theta^{\mathsf{T}}\Phi^{\mathsf{T}}\boldsymbol{y} - \boldsymbol{y}^{\mathsf{T}}\Phi\theta + \theta^{\mathsf{T}}\Phi^{\mathsf{T}}\Phi\theta, \tag{A.42}$$

and solve above optimization by building the derivative with respect to $\hat{\theta}$ and setting the result to zero

$$\frac{\partial}{\partial\theta}J = 0 - \boldsymbol{y}^{\mathsf{T}}\Phi - \boldsymbol{y}^{\mathsf{T}}\Phi + 2\hat{\theta}^{\mathsf{T}}\Phi^{\mathsf{T}}\Phi = 0, \tag{A.43}$$

and solving for $\hat{\theta}$

$$-2\Phi^{\mathsf{T}}\boldsymbol{y} = -2\Phi^{\mathsf{T}}\Phi\hat{\theta} \tag{A.44}$$

$$\hat{\theta} = (\Phi^{\mathsf{T}}\Phi)^{-1}\Phi^{\mathsf{T}}\boldsymbol{y}. \tag{A.45}$$

The matrix product $\Phi^\dagger = (\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}$ is called the left "Moore-Penrose pseudo inverse" (see for example also in [Sim06], p. 81) and it is important to understand the details of this equation. The pseudo-inverse only exists if $(\Phi^\mathsf{T}\Phi)^{-1}$ is invertible. This is only the case if Φ is full rank, and $n \leq k$, meaning that it has linearly independent columns. In other words, if at least as many (linearly independent) measurement error equations are available, as there are unknown variables contained in θ. The linear least squares approach described here is also termed *ordinary least squares*. Since we had to build a batch of measurements and regressor vectors to formulate above problem, it has also been termed *batch linear least squares* before.

In order to realize the calculation of the ordinary least squares solution in (A.45), different approaches are used [Lee12]. These include (i) the Cholesky factorization of $\Phi^\mathsf{T}\Phi$, (ii) the QR-factorization of Φ and (iii) the Singular Value Decomposition (SVD).

A.3.3 Weighted linear least squares

In the ordinary least squares solution, we found the parameters by minimizing the sum of squared errors, which is the ℓ_2-norm of the error vector. This can be generalized by finding a minimum to a weighted ℓ_2-norm instead, which transforms the cost function to

$$\min\ ||e||_W^2 = \min J = (\boldsymbol{y} - \Phi\theta)^\mathsf{T} W (\boldsymbol{y} - \Phi\theta), \tag{A.46}$$

where W is a symmetric, positive semi-definite weighting matrix. Similar to the derivation above, the solution can then be shown to be [PT+07]

$$(\Phi^\mathsf{T} W \Phi)^{-1}\Phi^\mathsf{T} W \boldsymbol{y}. \tag{A.47}$$

As a special case, if $W = diag(w)$ is a diagonal matrix with the entries of a weight vector w, then the cost function is

$$||e||_W^2 = \sum w_i e_i^2. \tag{A.48}$$

A.3.4 Alternative solution methods to the linear least-squares problem

In the following, we will present the following alternative solution methods to the linear least-squares problem: The QR-decomposition, which we will use extensively in the derivation of the DeePLS algorithm in Chapter 5, and it is also used as efficient method within the SRCDIF algorithm, see Section A.6.5. The latter also makes use of the Efficient Least Squares method, which can be found in Section A.3.4.

QR - decomposition

An alternative method for computing the least-squares solution of $\boldsymbol{y} = \Phi\theta + \nu$ is by using the QR-decomposition. In this orthogonal projection method, the feature matrix $\Phi \in \mathbb{R}^{m\times n}$ is decomposed as

$$\Phi = QR, \tag{A.49}$$

where $Q \in \mathbb{R}^{m\times m}$ is an orthogonal matrix and $R \in \mathbb{R}^{m\times n}$ is an upper triangular matrix such that

$$\Phi = Q\begin{pmatrix} R_1 \\ 0 \end{pmatrix}. \tag{A.50}$$

An orthogonal matrix Q is one for which $QQ^\mathsf{T} = Q^\mathsf{T}Q = \mathbb{I}$ holds, which means that $Q^\mathsf{T} = Q^{-1}$. The matrix $R_1 \in \mathbb{R}^{m\times m}$ above is a square right triangular matrix. Let us define the following operator for the QR-decomposition:

$$QR = \operatorname{qr}\{\cdot\}. \tag{A.51}$$

It can be shown that the solution for $\hat{\theta}$ for the underdetermined case ($m < n$) can be calculated as

$$\hat{\theta} = Q \begin{pmatrix} \left(R_1^\mathsf{T}\right)^{-1} y \\ 0 \end{pmatrix}, \tag{A.52}$$

as for the over-determined case ($m > n$) as

$$\hat{\theta} = R_1^{-1}\left(Q_1^\mathsf{T} y\right). \tag{A.53}$$

In the last equation above, Q_1 is an $m \times n$ matrix, which contains only the first n columns of Q. In both cases, the solution can be obtained efficiently without inverting the matrix R_1 by using forward substitution (or backward substitution, respectively).

A very common algorithm to obtain the QR-decomposition is the Householder-algorithm, which can be computed with a complexity (in terms of floating point multiplications) of $\frac{2}{3}n^3 + n^2 + \frac{1}{3}n - 2 = O\left(n^3\right)$

In order to solve the weighted least-squares problem, we minimize the cost function as previously given in (A.46), we can find the solution as

$$Q_1 R_1 = W^{\frac{1}{2}}\Phi \tag{A.54}$$

$$\hat{\theta} = R_1^{-1}\left(Q_1^\mathsf{T}\left(W^{\frac{1}{2}} y\right)\right), \tag{A.55}$$

where again Q_1, R_1 are obtained from an economy-sized QR-decomposition. This can easily be shown if the weighted normal equations are rewritten as

$$\left(W^{\frac{1}{2}}\Phi\right)^\mathsf{T}\left(W^{\frac{1}{2}}\Phi\right)\hat{\theta} = \left(W^{\frac{1}{2}}\Phi\right)^\mathsf{T}\left(W^{\frac{1}{2}} y\right), \tag{A.56}$$

where the square root of the weighting matrix exists since W is positive semi-definite [PT+07].

Efficient least squares

Efficient least squares solves the linear equation $Px = b$ by using the Cholesky factor S (satisfying $P = SS^\mathsf{T}$), resulting in the expression $x = S^{-\mathsf{T}}\left(S^{-1}b\right)$. This reduces the computational complexity of the least-squares operation to $\mathcal{O}\left(n^2\right)$ [LW+12].

A.4 Recursive linear regression

A.4.1 From full information estimation to recursive estimation

We defined our cost functional above in (A.20) to include stage costs starting from time $t = 1$ until the current time $t = k$, which we silently expressed in the subscript of $J_{[1:k]}$. Using all this available information in an estimator is commonly termed as Full Information Estimation (FIE). Instead, we could also divide the data into two separate batches, one running from time $t = 1$ to time $t = k - N$, and another running from $t = k - N + 1$ to $t = k$. Minimizing the two cost functions resulting from this action would not change the optimization problem since we defined the running costs to depend only on current values at time t:

$$J_{[1:k]}(\ell_1, \ell_2, \ldots, \ell_k) = J_{[1:k-N]} + J_{[k-N+1:k]}. \tag{A.57}$$

The cost $J_{[1:k-N]}$ in equation (A.57) is termed *arrival cost*. A recursive estimation scheme, which at the current time only solves the optimization problem of minimizing $J_{[k-N+1:k]}$ over the horizon of the last N samples is called Moving Horizon Estimation (MHE).

If the horizon length is reduced to $N = 1$, and the estimator recursively operates only on the current data at time $t = k$, the estimator scheme is called *recursive*.

A.4.2 Recursive Linear Least-Squares estimation

For the purpose of computational inexpensive online estimation, recursive versions of the linear least-squares algorithm were proposed. The recursive least-squares algorithm can be given as follows:

$$\hat{\theta}_k = \hat{\theta}_{k-1} + K_k \left(y_k - \phi_k^\top \hat{\theta}_{k-1} \right) \tag{A.58}$$

with a time-varying adaptive gain K_k, calculated as a function of the covariance matrix P_k. The dynamics of P_k, or rather its inverse Υ_k can generally be given by

$$\Upsilon_k = \Lambda_k \Upsilon_{k-1} + \phi_k \phi_k^\top. \tag{A.59}$$

The matrix $\Upsilon_k = P_k^{-1}$ is usually termed the *information matrix* or the precision matrix, while Λ is the *forgetting matrix* (see also section A.6.4). From (A.59), we can see that the information matrix, and hence the information content, is increased by the term given as the matrix $\phi_k \phi_k^\top$. This formulation therefor is useful to get an intuition about the fact that this term determines the increase in information about the system. Let us memorize this term as we will find in when discussing conditions, under which RLS will converge.

Different formulations exist which are mathematically identical, but many times they are based on the covariance matrix formulation instead. For lecture notes on the topic, including proofs that the recursive version is equivalent to the batch processing version, see Islam and Bernstein [IB19]. There, a Recursive Least Squares (RLS) version with the forgetting is provided in which the gain K is given as

$$K_k = P_k \phi_k \left(\lambda \mathbb{I} + \phi^\top P_k \phi \right)^{-1} \tag{A.60}$$

and a covariance matrix update rule as

$$P_{k+1} = \frac{1}{\lambda} P_k - \frac{1}{\lambda} P_k \phi_k \left(\lambda \mathbb{I} + \phi_k^\top P_k \phi_k \right)^{-1} \phi_k^\top P_k. \tag{A.61}$$

As a side note, in [IB19], ϕ was defined as transposed vector. The reason for a forgetting factor $\lambda < 1$ is that the algorithm maintains its ability to identify the plant parameters and this is the case also for time-varying parameters. In order for RLS to converge, necessary conditions were provided in [JJ+82]. An important condition is the notion of *persistent excitation* (see Section A.10.4).

While standard RLS uses a constant forgetting factor λ, variants with variable forgetting factors have been proposed in order to improve convergence for the estimation of time-variant parameters. While standard recursive least squares is a fundamental algorithm for estimation and adaptive control, variable-rate forgetting schemes are still an area of active research. For an example, in which also the convergence of RLS schemes is discussed, see [BG+20].

A.5 From least squares to recursive Bayesian state estimation

Until now, ordinary least squares and recursive least squares dealt with the problem of estimating *parameters*, of which we know nothing about their evolution in time but observe measurements, which are known to follow the relation

$$y = \phi^{\mathsf{T}}\theta + \nu. \tag{A.62}$$

A more general problem is the one of obtaining estimates of states x, where we assume that x follows a certain (known) dynamics, but again we are only able to observe noisy observations, while we assume to know the relation between the state x and the observation y. Next, we will explain this concept for the linear, deterministic case, which leads to the well known Luenberger Observer. Then, we will formulate a more general, stochastic problem, for which the Kalman filter is the optimal solution. This aims to present the connection of the two concepts. The Kalman filter will then be discussed in detail in Section A.6.

A.5.1 The Luenberger Observer

The Luenberger observer is an algorithm able to provide convergent estimates of the full state of a linear state-space system given only the system outputs. Let us consider the system

$$x_k = Fx_{k-1} + Bu_{k-1} \tag{A.63a}$$
$$y_k = Hx_k, \tag{A.63b}$$

of which F, B and H are known matrices. The Luenberger observer can be viewed as a state-space controller, which drives the output error

$$\epsilon_k = y_k - \hat{y}_k \tag{A.64}$$

to zero. In the equation above, $\hat{y}_k$ is the output of the calculation

$$\hat{x}_k = F\hat{x}_{k-1} + Bu_{k-1} \tag{A.65a}$$
$$\hat{y}_k = H\hat{x}_k, \tag{A.65b}$$

where $\hat{x}$ denotes an estimate of the unknown state vector. The aim is to find a “controller” that drives the system error to zero, which is given if we find a matrix K such that

$$\epsilon_{k+1} = \mathcal{K}\epsilon_k, \tag{A.66}$$

where $\mathcal{K}$ needs to be a stabilizing matrix and hence $\mathcal{K}$ needs to have poles within the unit circle for asymptotic stability. This can be realized by adding a corrective term to the system dynamics equations as

$$\hat{x}_k = F\hat{x}_{k-1} + Bu_{k-1} + K\epsilon_k \tag{A.67a}$$
$$\hat{y}_k = H\hat{x}_k. \tag{A.67b}$$

It can be shown that with this choice, the error dynamics results in

$$\epsilon_{k+1} = \mathcal{K}\epsilon_k = (F - KC)\epsilon_k, \tag{A.68}$$

and we are able to choose K, given that the system is observable, to place the poles of $\mathcal{K}$. Note that the Luenberger observer above was derived from a deterministic setting without any measurement noise.

A.5.2 The state estimation problem of stochastic dynamical systems

If we additionally consider additive, Gaussian noise both on the process and the measurement equation of system (A.63), this leads to the linear, time-discrete and time-invariant system of the form

$$\Sigma : \begin{cases} x_k & = F x_{k-1} + B u_{k-1} + \omega_{k-1} \\ y_k & = H x_k + v_k \\ \omega_k & \sim \mathcal{N}(0, Q_k) \\ v_k & \sim \mathcal{N}(0, R_k) \\ P_0 & = \mathbb{E}\left[\left(x_0 - \hat{x}_0^-\right)\left(x_0 - \hat{x}_0^-\right)^\mathsf{T}\right]. \end{cases} \tag{A.69}$$

The disturbances are known to be white, zero mean and uncorrelated noise drawn from the normal distributions with the process noise covariance matrix Q and the measurement noise covariance matrix R. It is further assumed that the system parameters in the matrices F, G and H are known.

Additionally, in order to derive state estimation algorithm, in a Bayesian setting, one needs to provide some kind of *prior knowledge* about the system. Above, this is given in the form of that the state estimation error covariance matrix P_0 at initialization time is assumed to be known. Given (A.69), one is able to derive an optimal observer gain in each step. The resulting equations are the ones of the linear Kalman filter, as we will show in Section A.6.

A.6 The Kalman Filter family

The Kalman Filter is known to be - in many senses - the optimal recursive solution to the stochastic, linear, time-discrete state estimation problem for the system given in (A.69) [Sim06]. It is named after Rudolph Emil Kalman, who published his seminal paper R. E. Kalman. “A new approach to linear filtering and prediction problems”. In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45. Since then, the Kalman filter and its various derivations found themselves to be the solution to many real world problems. By today, more than 1.420.000 search results for the term “Kalman” within academic papers can be found in Google Scholar and no less than 14 million search results are given by the search engine Google. The original paper proposed the problem formulation in today’s well known state space formulation, and derived the Kalman filter equations by using orthogonality properties. Many approaches are known to lead to the same results for the linear Kalman filter equations (see for example Simon [Sim06] or Thrun [Thr02]). In this section, we want to derive Kalman Filter equations from different viewpoints.

As already stated in Section A.5.1, one can view the Kalman Filter as an adaptive version of a Luenberger observer. If we consider additive, Gaussian noise both on the process and the measurement equation as given in equation (A.69), the Kalman filter can be seen as a way to optimally (in a least-squares sense) choose an adaptive observer gain K_k.

Also, the Kalman Filter can be proven to be a generalized variant of Recursive Least-Squares (see for example Rhode [Rho16], p.91).

In this section, we want to start by introducing the linear, time-discrete Kalman Filter in Section A.6.1. Next, we want present the Extended Kalman Filter [SS+62] in Section A.6.2, which is an extension for systems following nonlinear dynamics. The Extended Kalman Filter applies the linear Kalman Filter after linearizing a nonlinear system in its operating point.

For highly nonlinear transition functions, the Extended Kalman Filter might have poor performance because of errors in the propagation of the covariances due to linearization effects. The Unscented Kalman Filter can reduce these errors by performing a so called unscented transformation of the covariances. This is done by propagating so-called Sigma points which have the same mean and covariance and are normally distributed. Like this, the true covariance can be recovered since these Sigma points undergo the full nonlinear transformation, while the EKF only performs a linearized transformation of the covariance. But, as a drawback, this comes a increased computational cost. If even more points, so called particles, are used to propagate arbitrary probability density functions, this leads to the so called particle filter. Variants for time-continuous, hybrid and time-discrete systems exist.

A.6.1 Linear Kalman Filter

First, we want to provide the steps to realize the linear, discrete-time Kalman Filter. A derivation of the equations can be found in Simon [Sim06], pp 124-129.

Let us consider the system Σ given in (A.69). We can only perceive the true states x_k of the system by observing the state outputs y_k with added measurement noise. We have only limited knowledge about the state transitions due to process noise affecting the state propagation. It is assumed that the system parameters in the matrices F, B and H are known, together with the process noise covariance matrix Q and the measurement noise matrix R. The Kalman Filter can then be used to obtain apriori state estimations $\hat{x}^-$ and a posteriori state estimations $\hat{x}^+$ at each time step with the following algorithm.

1. Initialize filter

$$\hat{x}_0^+ = \mathbb{E}[x_0] \tag{A.70}$$

$$\hat{P}_0^+ = \mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^\mathsf{T}\right] \tag{A.71}$$

2. For $k = 1, 2, \dots$ perform:

   Measurement update:

   Calculate a posteriori state estimate:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - H\hat{x}_k^-) \tag{A.72}$$

   Calculate estimation error covariance:

$$P_k^+ = P_k^- - K_k H P_k^- = (\mathbb{I} - K_k H)P_k^- \tag{A.73}$$

   or using Joseph's form

$$= (\mathbb{I} - K_k H)P_k^-(\mathbb{I} - K_k H)^\mathsf{T} + K_k R_k K_k^\mathsf{T} \tag{A.74}$$

   Compute (optimal) Kalman gain:

$$K_k = P_k^- H^\mathsf{T}\left(H P_k^- H^\mathsf{T} + R_k\right)^{-1} \tag{A.75}$$

   Time update:

   Calculate a priori state estimate:

$$\hat{x}_{k+1}^- = F\hat{x}_k^+ + Bu_k \tag{A.76}$$

   Calculate a priori estimation error covariance:

$$P_{k+1}^- = FP_k^+F^\mathsf{T} + Q_k. \tag{A.77}$$

In the different expressions for the estimation error covariance P, the last expression (A.74) is the computationally more expensive, but also more robust Joseph's form. It always provides a positive definite result, and it remains valid also for other than the optimal Kalman gain.

A.6.2 Extended Kalman Filter

The Extended Kalman Filter is obtained by applying a linear Kalman Filter to a linearized nonlinear system. It does not provide an optimal filter as the Kalman Filter does in the linear case. Nevertheless, it is widely applied in engineering solutions due to its recursive nature and the resulting computational efficiency. We want to provide the background similar to Simon [Sim06], pp 407-409, but using our own notation. The time-discrete, nonlinear system and measurement equations are given as follows:

$$x_k = f(x_{k-1}, u_{k-1}, \omega_{k-1}) \tag{A.78}$$

$$y_k = h(x_k, \nu_k) \tag{A.79}$$

$$\omega_k \sim \mathcal{N}(0, Q_k) \tag{A.80}$$

$$\nu_k \sim \mathcal{N}(0, R_k), \tag{A.81}$$

with state vector $x_k \in \mathbb{R}^n$, the system dynamics function f, the system input u and process noise ω drawn from a multivariate normal distribution $\mathcal{N}$ determined by the process noise covariance matrix $Q \in \mathbb{R}^{n \times n}$. Further, we have the measurement equation given with the observations vector $y_k \in \mathbb{R}^m$, the measurement function h and measurement noise ν_k the measurement noise covariance matrix $R \in \mathbb{R}^{m \times m}$.

The EKF is then obtained by linearizing the above equations around the system state. Therefor, a Taylor series expansion of the state equation is performed around the a posteriori state estimation $\hat{x}_{k-1}^+$ and ω_{k-1}. The measurement equation is linearized around the a priori state estimation $\hat{x}_k^-$ and $\nu_k = 0$.

It can be shown that, after performing the Taylor series expansion, the system and measurement equations can be rewritten with the linear, time-discrete equations

$$x_k = F_k x_{k-1} + B_k \tilde{u}_{k-1} + \tilde{\omega}_{k-1} \tag{A.82}$$

$$y_k = H_k x_k + z_k + \tilde{\nu}_k \tag{A.83}$$

$$\tilde{\omega}_k \sim \mathcal{N}\left(0, L_k Q_k L_k^\mathsf{T}\right) \tag{A.84}$$

$$\tilde{\nu}_k \sim \mathcal{N}\left(0, M_k R_k M_k^\mathsf{T}\right) \tag{A.85}$$

with the matrices F, H, L, M given by the Jacobians as in (A.90) and (A.91) and the known signals

$$\tilde{u}_k = f_k(\hat{x}_k^+, u_k, 0) - F_k \hat{x}_k^+ \tag{A.86}$$

$$z_k = h_k(\hat{x}_k^-, 0) - H_k \hat{x}_k^-. \tag{A.87}$$

Then, the discrete-time Extended Kalman Filter algorithm is as follows:

1. Initialize filter

$$\hat{x}_0^+ = \mathbb{E}[x_0] \tag{A.88}$$

$$\hat{P}_0^+ = \mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^\mathsf{T}\right] \tag{A.89}$$

2. For $k = 1, 2, \ldots$ perform:

   Compute Jacobians F_k, L_k, H_k, M_k

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}^+, u_{k-1}}, \qquad L_k = \left.\frac{\partial f}{\partial \omega}\right|_{\hat{x}_{k-1}^+, u_{k-1}} \tag{A.90}$$

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k^-}, \qquad M_k = \left.\frac{\partial h}{\partial \nu}\right|_{\hat{x}_k^-} \tag{A.91}$$

Perform time update of state estimate and estimation error covariance:

$$\hat{x}_k^- = f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, 0) \tag{A.92}$$
$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^\mathsf{T} + L_{k-1}Q_{k-1}L_{k-1}^\mathsf{T} \tag{A.93}$$

Compute Kalman gain K:

$$K_k = P_k^-H_k^\mathsf{T}\left(H_kP_k^-H_k^\mathsf{T} + M_kR_kM_k^\mathsf{T}\right)^{-1} \tag{A.94}$$

Perform measurement update of state estimate

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - H_k\hat{x}_k^-) \tag{A.95}$$

Calculate estimation error covariance:

$$P_k^+ = (\mathbb{I} - K_kH_k)P_k^-(\mathbb{I} - K_kH_k)^\mathsf{T} + K_kR_kK_k^\mathsf{T}. \tag{A.96}$$

Note that for noise terms entering the equations in an additive manner, the Jacobians M and L are the identity matrix.

A.6.3 Sigma-Point Kalman Filters

The *Extended* Kalman Filter approximates a nonlinear function by linearizing in each operating point and works with linearized transformations of probability density functions. As long as the linearized transformation of means and covariances are approximately equal to the nonlinear transformations, it provides sufficient results. The *Unscented Kalman Filter* is a method to reduce the error obtained in the Extended Kalman Filter due to linearization effects, by performing a nonlinear transformation on single points instead of an entire probability density function. These points, called sigma-points, are chosen so that they have the same mean and covariance than the one which needs to be transformed. This is called an *unscented transformation*. The interested reader is referred to [Sim06], pp. 433-446 for an overview with detailed description and derivation of the UKF equations. The UKF was originally proposed by Julier and Uhlmann[JU97] with modifications in [Jul02]. Merwe [Mer04] provides detailed analysis and presented variants including an improved version, introducing a recursive square-root formulation to avoid the computation of the matrix square-root operation as in the basis versions.

Unscented transformation

The principle behind the UKF is the unscented transformation [JU97]: Consider that a random vector $x \in \mathbb{R}^n$ with mean $\bar{x}$ and covariance P^x is transformed through a nonlinear function $y = g(x)$, with $g : \mathbb{R}^n \to \mathbb{R}^m$. We therefore consider a set of so called *Sigma points*, which are a set of $2n + 1$ weighted samples:

$$\mathcal{S}_i = \{w_i, \chi_i\}, \quad i \in 1, \ldots, p, \qquad p = 2n + 1 \tag{A.97}$$

The sigma-points are propagated through the function:

$$y_i = g(\chi_i). \tag{A.98}$$

We can then calculate the weighted average of the transformed Sigma points.

$$\bar{y} = \sum_{i=0}^{p} w_i y_i, \qquad \sum_{i=0}^{p} w_i = 1, \tag{A.99}$$

as well as the covariance by the weighted outer product of the transformed points:

$$P_{y_i} = \sum_{i=0}^{p} w_i (y_i - \bar{y}_i)(y_i - \bar{y}_i)^{\mathsf{T}} \tag{A.100}$$

Note that the calculation for both the sigma-points and the weights are defined in a deterministic manner such that they fully capture the mean and covariance of the original random variable x. In order to match this requirement, the sigma-points have been previously defined in different ways. Before we show the *scaled* unscented transformation [Jul02], let us directly provide a general system description for which we demonstrate the UKF.

This choice of sigma-points ensures that the first and second moment of the distribution with the covariance matrix P is preserved:

$$\hat{\mathbf{x}}_n = \frac{1}{2n_x} \sum_{i=1}^{2n_x} \mathbf{x}_n^i \tag{A.101}$$

$$P_{x_n} = \frac{1}{2n_x} \sum_{i=1}^{2n_x} \left(x_n^i - \hat{x}_n\right)\left(x_n^i - \hat{x}_n\right)^{\mathsf{T}}. \tag{A.102}$$

Note that only the mean and covariance (first and second order moments) are preserved.

Uncented Kalman filter

Here we want to present the UKF Julier and Uhlmann [JU97] and Julier [Jul02] similarly to the presentations in Simon [Sim06] and Merwe [Mer04]. We consider the nonlinear, time discrete system dynamics as follows:

$$x_k = f(x_{k-1}, u_{k-1}, \omega_{k-1}) \tag{A.103}$$

$$y_k = h(x_k, u_k, \nu_k), \tag{A.104}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^{n_u}$, $\omega \in \mathbb{R}^{n_\omega}$ are the state, input and process noise vectors, and $y \in \mathbb{R}^{n_y}$, $\nu \in \mathbb{R}^{n_\nu}$ are the measurement / observation vector and measurement noise vector, respectively. As with the Kalman filter, we consider Gaussian process and measurement noise

$$\omega_k \sim \mathcal{N}(0, Q_k) \tag{A.105}$$

$$\nu_k \sim \mathcal{N}(0, R_k). \tag{A.106}$$

Note that both the process noise ω can enter the system dynamics $f : \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\omega} \to \mathbb{R}^n$ and the measurement noise term ν can enter $h : \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\nu} \to \mathbb{R}^{n_y}$ in a non-additive fashion which allows for a more general description. To take this into account, we can define an augmented state vector as

$$\tilde{x} \triangleq \begin{pmatrix} x \\ \omega \\ \nu \end{pmatrix}, \tag{A.107}$$

with dimension $n = n_x + n_\omega + n_\nu$. The expectation of this augmented state at time $k-1$ given the information available at $k-1$ is

$$\tilde{x}_{k-1|k-1} = \tilde{x}_{k-1}^{+} \doteq \begin{pmatrix} x_{k-1|k-1} \\ \mathbf{0}^{n_\nu \times 1} \\ \mathbf{0}^{n_\omega \times 1} \end{pmatrix}, \tag{A.108}$$

and its augmented state covariance matrix is composed from the covariance matrices as follows:

$$\tilde{P}_{k-1|k-1} = \begin{pmatrix} P_{k-1|k-1} & \mathbf{0}^{n_x \times n_\omega} & \mathbf{0}^{n_x \times n_\nu} \\ \mathbf{0}^{n_\omega \times n_x} & Q_{k-1} & P^{\omega\nu}_{k-1} \\ \mathbf{0}^{n_\nu \times n_x} & P^{\nu\omega}_{k-1} & R_{k-1} \end{pmatrix}, \tag{A.109}$$

where $P^{\nu\omega} = P^{\omega\nu} = 0$ might be zero in case of uncorrelated noises. The above values are initialized prior to starting the algorithm with

$$\hat{x}_0^+ = \mathbb{E}[x_0] \tag{A.110}$$

$$P^+_{x,0} = \mathbb{E}\Big[\|x_0 - \hat{x}_0^+\|^2 \Big]. \tag{A.111}$$

From this information, one can calculate a matrix containing a set of sigma-points as follows:

$$\tilde{\mathcal{X}}_{i,k-1|k-1} \doteq \begin{cases} \tilde{x}^+, & i = 1 \\ \tilde{x}^+ + \gamma S_i, & i = 2, \ldots, n+1 \\ \tilde{x}^+ - \gamma S_i, & i = n+2, \ldots, 2n+1 \end{cases} \tag{A.112}$$

where S_i is the i-th column of the matrix

$$S = \sqrt{\tilde{P}}, \tag{A.113}$$

and $\sqrt{\tilde{P}}$ is the square root of the matrix $\tilde{P}$ with the following properties:

$$\sqrt{\tilde{P}}^{\mathsf{T}} \sqrt{\tilde{P}} = \tilde{P}. \tag{A.114}$$

which can be robustly and efficiently calculated by using the *Cholesky decomposition*. In (A.112), γ is a scaling parameter calculated as:

$$\gamma = \sqrt{n + \lambda}, \quad \lambda = \alpha^2(n + \kappa) - n, \tag{A.115}$$

where α and κ are tuning parameters (see section A.6.3). In addition to (A.112), we calculated a set of weights which are chosen such that

$$\begin{aligned} w_1^m &= \frac{\lambda}{n+\lambda} & & i = 1 \\ w_1^c &= \frac{\lambda}{n+\lambda} + \left(1 - \alpha^2 + \beta\right) & & i = 1 \\ w_i^m &= w_i^c = \frac{1}{2(n+\lambda)} & & i = 2, \ldots, 2n+1 \end{aligned} \tag{A.116}$$

where w^m denotes weights for the mean and w^c stands for weights used to calculate the covariance matrices in the subsequent steps.

The sigma-points defined by the augmented matrix $\tilde{\mathcal{X}}$ now contain the sigma-points for states and noises:

$$\tilde{\mathcal{X}} = \begin{pmatrix} \mathcal{X} \\ \mathcal{X}^\omega \\ \mathcal{X}^\nu \end{pmatrix}. \tag{A.117}$$

In order to perform the prediction step of the UKF, we now propagate all sigma-points through the system dynamics:

$$\mathcal{X}_k^- = f\left(\mathcal{X}_{k-1}, u_{k-1}, \mathcal{X}^\omega_{k-1}\right) \tag{A.118}$$

$$\mathcal{Y}_k = h\left(\mathcal{X}_{k-1}, u_{k-1}, \mathcal{X}^\nu_{k-1}\right), \tag{A.119}$$

from which we can obtain an approximation of the predicted means as weighted averages

$$\hat{x}_k^- \approx \sum_{i=1}^{2n+1} w_i^m \mathcal{X}_{i,k} \tag{A.120}$$

$$\hat{y}_k \approx \sum_{i=1}^{2n+1} w_i^m \mathcal{Y}_{i,k}, \tag{A.121}$$

and the covariance matrices from

$$P_{x_k x_k}^- \approx \sum_{i=1}^{2n+1} w_i^c \left(\mathcal{X}_k^- - \hat{x}_k^-\right)\left(\mathcal{X}_k^- - \hat{x}_k^-\right)^\mathsf{T} \tag{A.122}$$

$$P_{\tilde{y}_k \tilde{y}_k} \approx \sum_{i=1}^{2n+1} w_i^c \left(\mathcal{Y}_{i,k} - \hat{y}_k\right)\left(\mathcal{Y}_{i,k} - \hat{y}_k\right)^\mathsf{T} \tag{A.123}$$

$$P_{x_k \tilde{y}_k} \approx \sum_{i=1}^{2n+1} w_i^c \left(\mathcal{X}_k^- - \hat{x}_k^-\right)\left(\mathcal{Y}_{i,k} - \hat{y}_k\right)^\mathsf{T}, \tag{A.124}$$

where P^{xy} is the cross-covariance matrix. To perform the measurement update step of the Kalman filter, we now calculate the (optimal) Kalman gain matrix different from the standard Kalman filter as

$$K_k = P_{x_k \tilde{y}_k} \left(P_{\tilde{y}_k \tilde{y}_k}\right)^{-1}, \tag{A.125}$$

while the posterior estimation of $\hat{\hat{x}}$ and the state covariance update steps are similar to the standard formulas:

$$\hat{\hat{x}}_k^+ = \hat{\hat{x}}^- + K_k \left(y_k - \hat{y}_k\right), \tag{A.126}$$

$$P_{x_k x_k}^+ = P_{x_k x_k}^- - K_k P_{\tilde{y}_k \tilde{y}_k} K_k^\mathsf{T}. \tag{A.127}$$

Note that the covariance matrix $P_{\tilde{y}_k \tilde{y}_k}$ is used above. We can substitute the expression given above for the optimal Kalman gain into (A.127), which yields

$$\begin{aligned}
P_{x_k x_k}^+ &= P_{x_k x_k}^- - P_{x_k \tilde{y}_k} \left(P_{\tilde{y}_k \tilde{y}_k}\right)^{-1} P_{\tilde{y}_k \tilde{y}_k} \left(P_{x_k \tilde{y}_k} \left(P_{\tilde{y}_k \tilde{y}_k}\right)^{-1}\right)^\mathsf{T} && \text{(A.128)}\\
&= P_{x_k x_k}^- - P_{x_k \tilde{y}_k} \left(P_{x_k \tilde{y}_k} \left(P_{\tilde{y}_k \tilde{y}_k}\right)^{-1}\right)^\mathsf{T} && \text{(A.129)}\\
&= P_{x_k x_k}^- - P_{x_k \tilde{y}_k} \left(P_{\tilde{y}_k \tilde{y}_k}\right)^{-1} P_{x_k \tilde{y}_k}^\mathsf{T} && \text{(A.130)}\\
&= P_{x_k x_k}^- - P_{x_k \tilde{y}_k} \Upsilon_{\tilde{y}_k \tilde{y}_k} P_{x_k \tilde{y}_k}^\mathsf{T}. && \text{(A.131)}
\end{aligned}$$

where the expression (A.130) follows from symmetry of the covariance matrix $P_{\tilde{y}_k \tilde{y}_k}$ and using the information matrix form $\Upsilon_{\tilde{y}_k \tilde{y}_k} = \left(P_{\tilde{y}_k \tilde{y}_k}\right)^{-1}$ in (A.131). In the same fashion, one can rewrite (A.126) using the cross-covariance form of the Kalman gain as

$$\hat{\hat{x}}_k^+ = \hat{\hat{x}}^- + P_{x_k \tilde{y}_k} \left(P_{\tilde{y}_k \tilde{y}_k}\right)^{-1} \left(y_k - \hat{y}_k\right). \tag{A.132}$$

We will use these alternative expressions later when deriving the information form of the sigma-point filters.

Some comments on the Uncented Kalman Filter

In (A.115), we must choose $\kappa \geq 0$ in order to guarantee that the covariance matrix remains positive definite. The choice of this value is regarded as non critical, and typically a good default value is reported to be $\kappa = 0$ [Mer04].

The value α needs to be in the range of $0 \leq \alpha \leq 1$, and controls the distance of the sigma-points to the mean value. To avoid the sampling of non-local effects of strong nonlinearities, a recommendation is to choose it to be a very small number [Mer04], p. 56.

$\beta \geq 0$ is a non-negative weighting term and can be used to incorporate knowledge of higher order moments of the given prior distribution. The optimal choice for Gaussian priors was given as $\beta = 2$ [Mer04], but one needs to take care that this statement only holds under certain assumptions, while including knowledge about the transforming function could lead to different results [Jul02].

Regarding implementation effort, one can benefit that the UKF does not require the calculation of the Jacobians. But instead, three additional tuning parameters have to be chosen (the three unscented transformation parameters α, β and κ). The computational complexity is slightly higher than for the EKF, since it requires the computation of a matrix square-root at each time step.

Sterling interpolation and the Central Difference Filter

Apart from the unscented Kalman filters, another derivative-less Kalman filter family has been proposed using Sterling interpolation. According to Merwe [Mer04], p. 62, both [NP+00] and [IX00] independently derived such formulations, which they called *divided difference filter* and *central difference filter*.

As outlined more in detail in Merwe [Mer04], p. 62, instead of calculating a Taylor series expansion, on can approximate a nonlinear function over a certain interval using (second order) *Sterling polynomial interpolation*. This is given by:

$$f(x) = f(\bar{x}) + \tilde{D}_{\nabla_x f} + \frac{1}{2!}\tilde{D}^2_{\nabla_x f}. \tag{A.133}$$

Here, $\tilde{D}_{\nabla_x f}$ and $\tilde{D}^2_{\nabla_x f}$ are the first and second order central divided difference operators acting on $f(x)$, which are given (for the scalar case) as:

$$\tilde{D}_{\nabla_x f} = (x - \bar{x})\frac{f(\bar{x}+h) - f(\bar{x}-h)}{2h} \tag{A.134}$$

$$\tilde{D}^2_{\nabla_x f} = (x - \bar{x})^2\frac{f(\bar{x}+h) - f(\bar{x}-h) - 2g(\bar{x})}{h^2}. \tag{A.135}$$

Here, h is the interval length or central difference step size and $\bar{x}$ is the prior mean of x around which the expansion is done. As stated in Merwe [Mer04], p. 62, "one can thus interpret the Sterling interpolation formula as a Taylor series wehre the analytical derivatives are replaced by central divided differences." In order to apply above formulas for the multivariate case, the following transformation is necessary:

$$z = S_x^{-1}x \tag{A.136}$$

$$\tilde{f}(z) \doteq f(S_x z) = f(x), \tag{A.137}$$

where S_x is the Cholesky factor of the covariance matrix of x, P_x, such that $P_x = S_x S_x^\mathsf{T}$. This transformation leads to a decoupling of the variables in x, leading to mutually uncorrelated individual components of z:

$$P_z = \mathbb{E}\left[(z - \bar{z})(z - \bar{z})^\mathsf{T}\right] = \mathbb{I}. \tag{A.138}$$

For the sake of brevity, further derivation of the Central Difference Kalman Filter (CDKF) is ommited at this point, and the interested reader is referred to [Mer04], pp. 62 - 68, where a neat discussion is presented.

A.6.4 Information Filter

An alternative way of calculating the Kalman filter equations is in the so called *information form*. Here, instead of propagating the error covariance matrix P forward in time, its inverse $\Upsilon \doteq P^{-1}$ is used. Various approaches can be taken to derive the time and measurement update equation of the Kalman filter in information form. One is by applying the matrix inversion lemma to represent the Kalman filter equations using *information parameters* instead of the moment parameters. As shown in Simon [Sim06], one can write for the measurement update equation of the covariance matrix P:

$$\left(P_k^+\right)^{-1} = \left(P_k^-\right)^{-1} + H_k^\mathsf{T} R_k^{-1} H_k, \tag{A.139}$$

and it is easy to see that by substituting the definition of the information matrix this yields

$$\Upsilon_k^+ = \Upsilon_k^- + H_k^\mathsf{T} R_k^{-1} H_k. \tag{A.140}$$

Applying the matrix inversion lemma to the time-update equation (A.77) for the covariance matrix P yields:

$$\left(P_{k+1}^-\right)^{-1} = \left(F_k P_k^+ F_k^\mathsf{T} + Q_k\right)^{-1} \tag{A.141}$$

$$\Upsilon_{k+1}^- = Q_k^{-1} - Q_k^{-1} F_k \left(\Upsilon_k^+ + F_k^\mathsf{T} Q_k^{-1} F_k\right)^{-1} F_k^\mathsf{T} Q_k^{-1}. \tag{A.142}$$

The following summarizes the full information filter algorithm containing time and measurement update equations:

1. Initialize filter

$$\hat{x}_0^+ = \mathbb{E}[x_0] \tag{A.143}$$

$$\hat{\Upsilon}_0^+ = \left\{\mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^\mathsf{T}\right]\right\}^{-1} \tag{A.144}$$

2. For $k = 1, 2, \ldots$ perform:

   Time update:

   Calculate a priori state estimate:

$$\hat{x}_k^- = F_{k-1} \hat{x}_{k-1}^+ + G_{k-1} u_{k-1} \tag{A.145}$$

   Calculate a priori information matrix:

$$\Upsilon_k^- = Q_{k-1}^{-1} - Q_{k-1}^{-1} F_{k-1} \left(\Upsilon_{k-1}^+ + F_{k-1}^\mathsf{T} Q_{k-1}^{-1} F_{k-1}\right)^{-1} F_{k-1}^\mathsf{T} Q_{k-1}^{-1} \tag{A.146}$$

   Measurement update:

   Calculate information update:

$$\Upsilon_k^+ = \Upsilon_k^- + H_k^\mathsf{T} R_k^{-1} H_k \tag{A.147}$$

   Compute Kalman gain:

$$K_k = \left(\Upsilon_k^+\right)^{-1} H_k^\mathsf{T} R_k^{-1} \tag{A.148}$$

   Calculate a posteriori state estimate:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \left(y_k - H_k \hat{x}_k^-\right). \tag{A.149}$$

Note that in case of constant matrices R and Q, the matrix inversions appearing in above equations can be calculated during the initialization process.

Information space representation

One can also transform the Kalman filter equations into a *information space representation*, which is based on the *information state vector* $\iota \doteq \Upsilon x$ instead of the state vector x. A derivation can be found for example in [Mut98], p. 27. The resulting information state vector update formulas then take the form (for the measurement update)

$$\hat{\iota}_k^+ = \hat{\iota}_k^- + H_k^{\mathsf{T}} R^{-1} y_k \tag{A.150}$$
$$\hat{\iota}_k^+ = \hat{\iota}_k^- + i_k, \tag{A.151}$$

where the information state contribution i_k from an observation y_k is defined as

$$i_k = H_k^{\mathsf{T}} R^{-1} y_k. \tag{A.152}$$

Similarly, the associated information matrix $I_k \doteq H_k^{\mathsf{T}} R_k^{-1} H_k$ is defined such that the measurement update of the information matrix becomes a simple sum between the a priori information plus the additional information given from a measurement:

$$\Upsilon_k^+ = \Upsilon_k^- + I_k. \tag{A.153}$$

Information filter for parameter estimation

If we consider the dynamics of constant parameters driven by Gaussian noise, we can write

$$\theta_k = \theta_{k-1} + \omega_\theta \tag{A.154}$$

with the dynamics matrix being the unity matrix. Then the equations from the Kalman and information filter become:

$$P_k^- = P_{k-1}^+ + Q_{k-1} \tag{A.155}$$
$$\Upsilon_k^- = Q_{k-1}^{-1} - Q_{k-1}^{-1}\left(\Upsilon_{k-1}^+ + Q_{k-1}^{-1}\right)^{-1} Q_{k-1}^{-1} \tag{A.156}$$

or alternatively

$$\Upsilon_k^- = \Upsilon_{k-1}^+ - \Upsilon_{k-1}^+\left(\Upsilon_{k-1}^+ + Q_{k-1}^{-1}\right)^{-1} \Upsilon_{k-1}^+ \tag{A.157}$$
$$\Upsilon_k^+ = \Upsilon_k^- + H_k^{\mathsf{T}} R_k^{-1} H_k \tag{A.158}$$
$$K_k = \left(\Upsilon_k^+\right)^{-1} H_k^{\mathsf{T}} R_k^{-1} \tag{A.159}$$
$$\hat{x}_k^- = \hat{x}_{k-1}^+ \tag{A.160}$$
$$\hat{x}_k^+ = \hat{x}_k^- + K_k\left(y_k - H_k \hat{x}_k^-\right). \tag{A.161}$$

From (A.157) and (A.158) we get

$$\Upsilon_k^+ = \underbrace{\left(\mathbb{I} - \Upsilon_{k-1}^+\left(\Upsilon_{k-1}^+ + Q_{k-1}^{-1}\right)^{-1}\right)}_{\doteq \Lambda_{k-1}} \Upsilon_{k-1}^+ + H_k^{\mathsf{T}} R_k^{-1} H_k, \tag{A.162}$$

with the *forgetting matrix* Λ. Above equation shows the evolution of the information matrix, which is driven by the measurements as

$$\Upsilon_k^+ = \Lambda_{k-1}\Upsilon_{k-1}^+ + H_k^{\mathsf{T}} R_k^{-1} H_k \tag{A.163}$$
$$\Upsilon_k^+ = \Lambda_{k-1}\Upsilon_{k-1}^+ + I_k. \tag{A.164}$$

We can also write

$$\Lambda_{k-1} := \mathbb{I} - \Upsilon^+_{k-1}\left(\Upsilon^+_{k-1} + Q^{-1}_{k-1}\right)^{-1} \tag{A.165}$$
$$= \mathbb{I} - \Upsilon^+_{k-1}\left(\mathbb{I} - P^+_{k-1}\left(P^+_{k-1} + Q_{k-1}\right)^{-1}\right)P^+_{k-1} \tag{A.166}$$
$$= P^+_{k-1}\left(P^+_{k-1} + Q_{k-1}\right)^{-1}, \tag{A.167}$$

where we derived the last equations by applying above matrix inversion lemma again. This representation makes it easy to see that for $Q_{k-1} = 0$, $\Lambda_{k-1} = \mathbb{I}$, which means that in this case, no forgetting (and therefor no reduction of information), takes place. Since, unless we can and want to influence the trajectory of the system, the information gain $I_k = H_k^\mathsf{T} R_k^{-1} H_k$ is solely given by the nature of the system evolution. What remains as a design parameter, is the *parameter noise* covariance matrix Q.

A.6.5 Square-root formulations

Square-root forms of Kalman filters and their derivates have been originally developed to avoid numerical problems on embedded computers with limited numerical precision. The basic idea is to predict and update the square-root of the covariance matrix, instead of the covariance itself. This was shown to dramatically improve the numerical properties of the solution under certain circumstances [LW+12].

More in detail, the idea of square-root filtering is to decompose the covariance matrix P into the matrix square-root such that $P = SS^\mathsf{T}$. If then, instead of propagating P directly, its square-root S is propagated, one can make sure that the resulting covariance matrix will always be positive semi-definite [Sim06]. The Cholesky Matrix Square Root Algorithm [Sim06], p. 160 can be used to calculate a possible solution for S from a given P. The reason for gaining precision by working with the matrix square-root is that the condition number of the matrix P is the square of the condition number of the matrix S, and hence one achieves twice the precision when calculating the matrix inverse on a computer with limited numerical precision, or when the elements of the covariance matrix have very different numerical ranges. The latter appears frequently when dealing with different variable scaling, for example due to the use of different units.

Various square-root forms of different filter types have been proposed since the 1960s, which all follow this basic idea. They include square-root forms of the Kalman filter (see section A.6.5), of sigma-point filters (see for example [Mer04]) and of information filters (see section A.6.5).

In the following subsections, we want to shortly present some ideas and information about the different filter types of square-root filtering. This mostly aims to help the reader categorize different filter methods. Nevertheless, a profound treatment of square-root forms of all filters would be out of scope of this presentation.

Square-root form of the Kalman filter

According to [Sim06], p. 158, the square-root form of the Kalman filter was most likely originally proposed by Potter [PS63] and used in the Apollo manned mission.

A profound treatment of square-root forms of the standard, linear Kalman filter would be out of scope of this presentation, but the interested reader is referred to [Sim06], section 6.3. At this point, we only want to point out that various algorithms have been proposed to calculate the update of the square-root of the covariance matrix. Since the matrix square-root is not unique, these algorithms might find different results. Besides Potter's algorithm [PS63], alternatives include algorithms based on orthogonal transformations, like the Householder's algorithm, the Gram-Schmidt algorithm, modified Gram-Schmidt algorithm or the Givens transformation (see [Sim06], p. 163 and references therein).

Square-root sigma-point information filtering

In section A.6.5, we already learned some background information about square-root Kalman filters. The main idea behind this form of Kalman filter is to propagate the matrix square-root of the error covariance matrix instead of the covariance itself, which results in higher numerical robustness and precision. This idea can be applied to different variants of Kalman filters, and instead of propagating the square-root of the error covariance matrix, for example, one can propagate the square-root of the information matrix. Here, we want to present the square-root form of sigma-point information filters as proposed by Liu, Worgotter, and Markelic [LW+12]. This variant is essentially a combination of information filtering and the square-root formulation, while for the covariance calculation, sigma-points are generated. As with the original sigma-point algorithm family, the generation of sigma-points can be either done via the unscented transform or by using Stirling's interpolation, resulting in the central difference form of a sigma-point filter. In that sense, besides the properties of the original sigma-point algorithms, using a square-root formulation inherits the increased numerical properties of the square-root algorithms. Again, these are improved numerical accuracy, higher precision and preservation of symmetry. In their paper [LW+12], they showed that the square-root central difference filter also preserves the positive-definiteness of the information matrix, and hence this filter is preferable to the square-root unscented information filter as well all the non-square root versions. As in the original paper, we will refer to the square-root version of the sigma-point information filters as Square-root Unscented Information Filter (SRUIF) and SRCDIF.

Square-root central difference information filter (SRCDIF)

As outlined in [LW+12], the SRCDIF applies three powerful matrix factorization techniques, namely

- the QR-decomposition
- the Cholesky Factor update
- and Efficient least-squares

Details on the QR-decomposition can be found in Section A.3.4. The Cholesky Factor update [Mat19a] is an efficient method to calculate the Cholesky factor

$$\mathrm{chol}\left\{A + xx^{\mathsf{T}}\right\} = \mathrm{cholupdate}\left\{R, x, +\right\}, \tag{A.168}$$

from the Cholesky factor $R = \mathrm{chol}\{A\}$. Efficient least-squares solves the linear equation $Px = b$ by using the Cholesky factor S (satisfying $P = SS^{\mathsf{T}}$), resulting in the expression $x = S^{-\mathsf{T}}\left(S^{-1}b\right)$. This reduces the computational complexity of the least-squares operation to $\mathcal{O}\left(n^2\right)$ [LW+12].

Here we summarize the SRCDIF as presented in [LW+12], but with slightly modified notation for consistency. The system considered in [LW+12] is given as

$$x_k = f\left(x_{k-1}, u_{k-1}, \omega_{k-1}\right) \tag{A.169}$$

$$y_k = h(x_k) + \nu_k, \tag{A.170}$$

where $x \in \mathbb{R}^{n}_{x}$, $u \in \mathbb{R}^{n_u}$, $\omega \in \mathbb{R}^{n_\omega}$ are the state, input and process noise vectors, and $y \in \mathbb{R}^{n_y}$, $\nu \in \mathbb{R}^{n_\nu}$ are the measurement / observation vector and measurement noise vector, respectively. We also assume the noises to be normally distributed as in (A.105). Note that above system is less general than the one considered for the UKF since only additive measurement noise is present.

Similar to the equations in section A.6.3, we define an augmented sigma-point matrix consisting of state vectors and process noise vectors:

$$\tilde{\mathcal{X}} = \begin{pmatrix} \mathcal{X} \\ \mathcal{X}^{\omega} \end{pmatrix} \in \mathbb{R}^{n \times 2n+1}, \tag{A.171}$$

where $n = n_x + n_\omega$ is the dimension of the augmented state vector. Algorithm 1 from [LW+12] can then be rewritten as:

1. Initialize filter

$$\hat{x}_0^+ = \mathbb{E}[x_0] \tag{A.172}$$
$$\hat{S}_{x_0}^+ = \operatorname{chol}\left\{\mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^{\mathsf{T}}\right]\right\} \tag{A.173}$$
$$S_\omega = \sqrt{Q} = \operatorname{chol}\{Q\} \tag{A.174}$$
$$S_\nu = \sqrt{R} = \operatorname{chol}\{R\} \tag{A.175}$$

2. For $k = 1, 2, \ldots, \infty$ perform:

Generate sigma-points for prediction:

$$\tilde{x}_{k-1} = \begin{pmatrix} x_{k-1} \\ \bar{\omega} \end{pmatrix} \tag{A.176}$$
$$\tilde{S}_{k-1} = \begin{pmatrix} S_{x_{k-1}} & 0 \\ 0 & S_{\omega_{k-1}} \end{pmatrix} \tag{A.177}$$
$$\tilde{\mathcal{X}}_{k-1} = \begin{pmatrix} \tilde{x}_{k-1} & \tilde{x}_{k-1} + h\tilde{S}_{k-1} & \tilde{x}_{k-1} - h\tilde{S}_{k-1} \end{pmatrix}. \tag{A.178}$$

Prediction equations:

$$\mathcal{X}_k^- = f(\mathcal{X}_{k-1}, u_{k-1}, \mathcal{X}_{k-1}^{\omega}) \tag{A.179}$$
$$\hat{x}_k^- = \sum_{i=1}^{2n+1} w_i^{(m)} \mathcal{X}_{i,k}^- \tag{A.180}$$
$$A = \sqrt{w_2^{(c_1)}} \left(\mathcal{X}_{2:n+1,k}^- - \mathcal{X}_{n+2:2n+1,k}^-\right) \tag{A.181}$$
$$B = \sqrt{w_2^{(c_2)}} \left(\mathcal{X}_{2:n+1,k}^- + \mathcal{X}_{n+2:2n+1,k}^- - 2\mathcal{X}_{1,k}^-\right) \tag{A.182}$$
$$\hat{S}_{x_k}^- = \operatorname{qr}\left\{\begin{pmatrix} A & B \end{pmatrix}\right\} \tag{A.183}$$
$$\iota_k^- = \left(\hat{S}_{x_k}^-\right)^{-\mathsf{T}} \left(\left(\hat{S}_{x_k}^-\right)^{-1} \hat{x}_k^-\right) \tag{A.184}$$
$$\hat{S}_{\iota_k}^- = \operatorname{qr}\left\{\left(\hat{S}_{x_k}^-\right)^{-1} \mathbb{I}\right\} \tag{A.185}$$

Generate sigma-points for measurement update:

$$\tilde{\mathcal{X}}_k^y = \begin{pmatrix} \hat{x}_k^- & \hat{x}_k^- + h\hat{S}_{x_k}^- & \hat{x}_k^- - h\hat{S}_{x_k}^- \end{pmatrix} \tag{A.186}$$

Measurement update:

$$\mathcal{Y}_k^- = h\left(\tilde{\mathcal{X}}_k^y\right) \tag{A.187}$$

$$\hat{y}_k = \sum_{i=1}^{2n+1} w_i^{(m)} \mathcal{Y}_{i,k}^- \tag{A.188}$$

$$\hat{P}_{x_k\tilde{y}_k} = \sqrt{w_2^{(c_1)}} \hat{S}_{x_k}^- \left(\mathcal{Y}_{2:n+1,k}^- - \mathcal{Y}_{n+2:2n+1,k}^-\right)^\mathsf{T} \tag{A.189}$$

$$U = \left(\hat{S}_{x_k}^-\right)^{-\mathsf{T}} \left(\left(\hat{S}_{x_k}^-\right)^{-1} \hat{P}_{x_k\tilde{y}_k}\right) S_\nu^{-\mathsf{T}} \tag{A.190}$$

$$\hat{\iota}_k^+ = \hat{\iota}_k^- + U S_\nu^{-1} \left(y_k - \hat{y}_k^- + \hat{P}_{x_k\tilde{y}_k}^\mathsf{T} \hat{\iota}_k^-\right) \tag{A.191}$$

$$S_{\iota_k}^+ = \text{cholupdate}\left\{\hat{S}_{\iota_k}^-, U, +\right\}. \tag{A.192}$$

In above algorithm, $\mathbb{I}$ is the identity matrix of appropriate size. The weighting parameters $w_i^{(m)}$, $w_i^{(c1)}$ and $w_i^{(c2)}$ are calculated as follows [LW+12]:

$$\begin{aligned} w_1^m &= \frac{h^2 - n}{h^2} & i &= 1 \\ w_i^m &= \frac{1}{2h^2} & i &= 2, \dots, 2n+1 \\ w_i^{(c_1)} &= \frac{1}{4h^2} & i &= 2, \dots, 2n+1 \\ w_i^{(c_2)} &= \frac{h^2 - 1}{4h^2} & i &= 2, \dots, 2n+1 \end{aligned} . \tag{A.193}$$

Note that different from the paper, we used 1-indexing in the sums and definitions above, which is also used in MATLAB®. The scaling parameter h is the scalar central difference interval size, which defines the spread of the distribution of the sigma-points. The optimal value for h was reported in [Mer04], p. 68, to be the square root of the kurtosis of the prior distribution of $z = S_x^{-1} x$. For Gaussian distributions, this leads to an optimal value of $h_{\text{opt}} = \sqrt{3}$ [Mer04; LW+12].

A.7 The particle filter

As an alternative to Kalman filtering, we quickly introduce the background to particle filtering techniques. We will perform a comparison between Kalman filter techniques and the particle filter in Chapter 6. We only want to roughly highlight the idea behind particle filtering here for the sake of brevity, but a full tutorial on particle filtering can be found, for example, in Simon [Sim06]. The particle filter algorithm works as follows [Sim06]. Suppose our system is given as

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, \omega_{k-1}) \tag{A.194}$$

$$y_k = h_k(x_k, \nu_k) \tag{A.195}$$

with ω_k and ν_k being independent white noise processes with known PDFs. The particle filter then subsequently performs the following steps:

1. Randomly generate a number of N particles, distributed according to the assumed known PDF of the initial state $p(x_0)$, denoted by $x_{0,i}^+ (i = 1, ...N)$.

2. For $k = 1, 2, \ldots$ do

   (a) Propagation step: propagate the particles to obtain a priori particles $x_{0,i}^{-}$ by using the known process equation and the known PDF of the process noise

   $$x_{k,i}^{-} = f_{k-1}\left(x_{k-1,i}^{+}, u_{k-1}, \omega_{k-1}^{i}\right) \qquad (i = 1, \ldots, N) \tag{A.196}$$

   with each ω_{k-1}^{i} noise vector is randomly generated on the basis of the known PDF of ω_{k-1}.

   (b) Compute the relative likelihood q^i of each particle, given the measurement observation y_k. This can be done by evaluating the PDF $p(y_k|x_{k,i}^{-})$ on basis of the measurement equation and the PDF of the measurement noise.

   (c) Then normalize the relative likelihoods so that the sum of all equals to one:

   $$q_i = \frac{q_i}{\sum_{j=1}^{N} q_j} \tag{A.197}$$

   (d) Resampling step: generate a set of a posteriori particles $x_{k,i}^{+}$ on basis of the obtained relative likelihoods

   (e) Now, given the set of a posteriori particles which are distributed according to the PDF $p\left(x_k|y_k\right)$, any statistical measure can be calculated, for example the mean and covariance.

A variety of variants exist, especially regarding the details of how the resampling step is performed. An overview is given in [LB+15].

A.8 Parameter estimation using Kalman filters

A.8.1 Parameters only estimation

We already outlined the problem formulation for the parameters-only estimation in state space form in Section A.1.3 as

$$\theta_k = \theta_{k-1} + \omega_{k-1}^{\theta} \tag{A.198}$$
$$y_k = h(\phi_k, \theta_k, \nu_k). \tag{A.199}$$

The application of the Extended Kalman filter or Uncented Kalman filter to this problem is straightforward, one only has to use the parameter vector θ_k in place of the state vector x_k. For measurement equations which are Linear-in-Parameters, by adding the assumption of constant parameters, the linear Kalman Filter can be applied on the system

$$\theta_k = \theta_{k-1} + \omega_{k-1}^{\theta} \tag{A.200}$$
$$y_k = \phi_k^{\mathsf{T}} \theta_k + \nu_k. \tag{A.201}$$

The formulation of the linear Kalman filter equations for this special case is given in Algorithm 8. Note that thanks to the parameter noise ω^{θ}, this formulation also allows to track time-varying parameters to certain degree, since this will keep the parameter estimation error covariance matrix lower bounded.

A.8.2 Enforcing parameter constraints

The assumption of parameters being Gaussian random variables might be, for many real world applications, somewhat contradictory to the true range of physical values which parameters can have. For example, parameter estimates being negative might often be physically impossible. Therefore, various methods to deal with enforcing parameter constraints within parameters-only Kalman Filtering have been proposed.

The naive solution of just saturating parameters to certain box constraints can cause instability of the Kalman Filter algorithm, since optimality conditions are then violated. A survey on methods to overcome this situation is given in Simon [Sim10], and other solutions were presented for example in Rao, Rawlings, and Lee [RR+01] and Kandepu, Imsland, and Foss [KI+08]. Altmannshofer, Endisch, et al. [AE+16] applied the method proposed by Timmons, Chizeck, et al. [TC+97], which we will shortly present in the following along the lines of Altmannshofer, Endisch, et al. The algorithm is derived for the Linear-in-Parameters problem. We consider linear equality and inequality constraints, which can be formulated as the linear matrix inequality

$$L\hat{\theta} \leq c, \tag{A.202}$$

where $L = [\mathbb{I}, -\mathbb{I}]^{\mathsf{T}}$, and $c = [\theta_{\text{max}}, -\theta_{\text{min}}]^{\mathsf{T}}$. The minimization of an original unconstrained cost function

$$J(\hat{\theta}) = \frac{1}{2}(y - \Phi\theta)^{\mathsf{T}}(y - \Phi\theta) \tag{A.203}$$

can be written as constrained problem using the Lagrangian

$$\mathcal{L}(\hat{\theta}, \eta) = J(\hat{\theta}) - \eta^{\mathsf{T}}\left(c - L\hat{\theta}\right), \tag{A.204}$$

where η is the vector of Lagrangian variables. This optimization problem has to satisfy the Karush-Kuhn-Tucker conditions, which in this case yields

$$c - L\hat{\theta} \geq 0 \tag{A.205a}$$
$$\eta \geq 0 \tag{A.205b}$$
$$\eta^{\mathsf{T}}\left(c - L\hat{\theta}\right) = 0 \tag{A.205c}$$
$$-\Phi^{\mathsf{T}}\left(y - \Phi\hat{\theta}\right) + L^{\mathsf{T}}\eta = 0. \tag{A.205d}$$

Solving (A.205d) for $\hat{\theta}$ yields

$$\hat{\theta}_c = \left(\Phi^{\mathsf{T}}\Phi\right)^{-1}\Phi^{\mathsf{T}}y - \left(\Phi^{\mathsf{T}}\Phi\right)^{-1}L^{\mathsf{T}}\eta \tag{A.206}$$
$$= P\Phi^{\mathsf{T}}y - PL^{\mathsf{T}}\eta \tag{A.207}$$
$$= \hat{\theta} - PL^{\mathsf{T}}\eta, \tag{A.208}$$

for the constrained solution $\hat{\theta}_c$ using the fact that $\left(\Phi^{\mathsf{T}}\Phi\right)^{-1} = P$ and that the unconstrained solution is given as $\hat{\theta} = P\Phi^{\mathsf{T}}y$. Above, both the unconstrained solution $\hat{\theta}$ and P can be used from the current solution found by a recursive algorithm. The Lagrangian variable η can be found by solving a complementary linear program, for which additional slack variables μ are introduced, to solve

$$\mu = c - L\hat{\theta}_c \geq 0, \tag{A.209}$$

which yields

$$\mu = LPL^{\mathsf{T}}\eta + c - L\hat{\theta} \tag{A.210a}$$
$$\mu \geq 0 \tag{A.210b}$$
$$\eta \geq 0 \tag{A.210c}$$
$$\mu^{\mathsf{T}}\eta = 0. \tag{A.210d}$$

To solve this linear complementarity program efficiently, Lemke's pivoting algorithm as found in Almqvist [Alm19] can be used, as also suggested by Altmannshofer, Endisch, et al. [AE+16].

A.8.3 Combined state and parameter estimation using Kalman Filters

Joint estimation using Kalman filters

Both Kalman Filter and particle filter frameworks can be used to estimate unknown system parameters together with the system states in a joint estimation framework, which we presented in Section A.1.1. According to Simon [Sim06], the first one to apply this method using a Kalman Filter might have been Kopp and Orford [KO63]. The application of the various filters is straightforward, since the state vector x in the Kalman Filter equations only has to be substituted by the augmented state vector $\tilde{x}$.

Dual Extended Kalman Filter

As outlined in Section A.1.2, the dual estimation problem finds estimates both for states and parameters of a system given as

$$x_k = f(x_{k-1}, u_{k-1}, \theta_{k-1}, \omega_{k-1}) \tag{A.211}$$

$$y_k = h(x_k, u_k, \nu_k), \tag{A.212}$$

and a parameter model given in state space form as

$$\theta_k = \theta_{k-1} + \omega^{\theta}_{k-1} \tag{A.213}$$

$$y^{\theta}_k = g(x_k, u_k, \theta_k, \nu^{\theta}_k). \tag{A.214}$$

Again, we assume independent white Gaussian noise processes for all the noise terms above. As proposed by Wan and Nelson [WN97] and further elaborated in Nelson [Nel00] and Haykin [Hay01] we can solve the dual estimation problem using EKFs.

In principle, the method works by concurrently running two separate Kalman filters for the state and parameter estimation. Nevertheless, special care has to be taken when deriving the Jacobians used in the Taylor expansion of the EKF equations. Since the filters are connected, there is an inter-dependency between the state estimation value and the parameter estimation value used for obtaining the state estimation. This becomes evident, if we re-write above system for the parameter evolution as

$$\theta_k = \theta_{k-1} + \omega^{\theta}_{k-1} \tag{A.215}$$

$$y^{\theta}_k = g(f(x_{k-1}, u_{k-1}, \theta_{k-1}, \omega_{k-1}), u_k, \theta_k, \nu^{\theta}_k). \tag{A.216}$$

Here, we replaced the value for the state x_k by its function representation. We obtain the following equations for the two EKFs: Definition of the Jacobians:

$$\begin{aligned}
\hat{F}_k &= \left.\frac{\mathrm{d}f\left(x_k, u_k, \hat{\theta}^-_k, \omega_k\right)}{\mathrm{d}x_k}\right|_{x_k=\hat{x}^+_k} & \hat{L}_k &= \left.\frac{\mathrm{d}f\left(x_k, u_k, \hat{\theta}^-_k, \omega_k\right)}{\mathrm{d}\omega_k}\right|_{\omega_k=\bar{\omega}} \\
\hat{H}^x_k &= \left.\frac{\mathrm{d}h\left(x_k, u_k, \hat{\theta}^-_k, \nu_k\right)}{\mathrm{d}x_k}\right|_{x_k=\hat{x}^-_k} & \hat{M}^x_k &= \left.\frac{\mathrm{d}h\left(x_k, u_k, \hat{\theta}^-_k, \nu_k\right)}{\mathrm{d}\nu_k}\right|_{\nu_k=\bar{\nu}} \\
\hat{H}^{\theta}_k &= \left.\frac{\mathrm{d}g\left(\hat{x}^-_k, u_k, \theta, \nu^{\theta}_k\right)}{\mathrm{d}\theta}\right|_{\theta=\hat{\theta}^-_k} & \hat{M}^{\theta}_k &= \left.\frac{\mathrm{d}g\left(\hat{x}^-_k, u_k, \theta, e_k\right)}{\mathrm{d}\nu^{\theta}_k}\right|_{\nu^{\theta}_k=\bar{\nu}^{\theta}_k},
\end{aligned} \tag{A.217}$$

where the *recursive calculation of the total derivative* of the weight matrix has to be considered, and following the chain rule results in:

$$\begin{aligned}\hat{H}_k^\theta &= \left.\frac{\mathrm{d}g\left(\hat{x}_k^-,u_k,\theta\right)}{\mathrm{d}\theta}\right|_{\theta=\hat{\theta}_k^-} = \\ \frac{\mathrm{d}g\left(\hat{x}_k^-,u_k,\theta\right)}{\mathrm{d}\theta} &= \frac{\partial g\left(\hat{x}_k^-,u_k,\theta\right)}{\partial\theta} + \frac{\partial g\left(\hat{x}_k^-,u_k,\theta\right)}{\partial\hat{x}_k^-}\frac{\mathrm{d}\hat{x}_k^-}{\mathrm{d}\theta} \\ \frac{\mathrm{d}\hat{x}_k^-}{\mathrm{d}\theta} &= \frac{\partial f\left(\hat{x}_{k-1}^+,u_{k-1},\theta\right)}{\partial\theta} + \frac{\partial f\left(\hat{x}_{k-1}^+,u_{k-1},\theta\right)}{\partial\hat{x}_{k-1}^+}\frac{\mathrm{d}\hat{x}_{k-1}^+}{\mathrm{d}\theta} \\ \frac{\mathrm{d}\hat{x}_{k-1}^+}{\mathrm{d}\theta} &= \frac{\mathrm{d}\hat{x}_{k-1}^-}{\mathrm{d}\theta} - K_{k-1}^x\frac{\mathrm{d}g\left(\hat{x}_{k-1}^-,u_{k-1},\theta\right)}{\mathrm{d}\theta}.\end{aligned} \tag{A.218}$$

Theoretically, the dependency of K_{k-1} on the parameter estimation would have to be considered, but practically, this dependence is known to be small and hence can be neglected [Nel00; Hay01]. In case that the observation equation $g(\cdot) = Cx_k$, the total derivative results to [Nel00] p. 102:

$$\begin{aligned}\hat{H}_k^\theta &= \left.\frac{\mathrm{d}g\left(\hat{x}_k^-,u_k,\theta\right)}{\mathrm{d}\theta}\right|_{\theta=\hat{\theta}_k^-} = \\ \frac{\mathrm{d}g\left(\hat{x}_k^-,u_k,\theta\right)}{\mathrm{d}\theta} &= C\frac{\mathrm{d}\hat{x}_k^-}{\mathrm{d}\theta} = \frac{\partial f\left(\hat{x}_{k-1}^+,u_{k-1},\theta\right)}{\partial\theta} + \frac{\partial f\left(\hat{x}_{k-1}^+,u_{k-1},\theta\right)}{\partial\hat{x}_{k-1}^+}\frac{\mathrm{d}\hat{x}_{k-1}^+}{\mathrm{d}\theta} \\ \frac{\mathrm{d}\hat{x}_{k-1}^+}{\mathrm{d}\theta} &= \left(\mathbb{I}-K_{k-1}^xC\right)\frac{\mathrm{d}\hat{x}_{k-1}^-}{\mathrm{d}\theta} + \frac{\partial K_{k-1}^x}{\partial\theta}\left(y_k^\theta - C\hat{x}_{k-1}^-\right),\end{aligned} \tag{A.219}$$

where $\dfrac{\partial K_{k-1}^x}{\partial\theta} \approx 0$ is typically neglected.

Summary of Dual EKF equations We can summarize the Dual EKF steps are as follows: Initialization:

$$\begin{aligned}\hat{\theta}_0^+ &= \mathbb{E}[\theta_0], \quad P_{\tilde{\theta},0}^+ = \mathbb{E}\left[\left(\theta_0-\hat{\theta}_0^+\right)\left(\theta_0-\hat{\theta}_0^+\right)^\mathsf{T}\right] \\ \hat{x}_0^+ &= \mathbb{E}[x_0], \quad P_{\tilde{x},0}^+ = \mathbb{E}\left[\left(x_0-\hat{x}_0^+\right)\left(x_0-\hat{x}_0^+\right)^\mathsf{T}\right].\end{aligned} \tag{A.220}$$

Computation: for $k = 1,2,\ldots$ calculate: Time update for parameter filter:

$$\hat{\theta}_k^- = \hat{\theta}_{k-1}^+ \tag{A.221}$$

$$P_{\tilde{\theta},k}^- = P_{\tilde{\theta},k-1}^+ + Q^\theta \tag{A.222}$$

Time update for state filter:

$$\hat{x}_k^- = f\left(\hat{x}_{k-1}^+,u_{k-1},\hat{\theta}_k^-,\bar{\omega}\right) \tag{A.223}$$

$$P_{\tilde{x},k}^- = \hat{F}_{k-1}P_{\tilde{x},k-1}^+\hat{F}_{k-1}^\mathsf{T} + \hat{L}_{k-1}Q\hat{L}_{k-1}^\mathsf{T} \tag{A.224}$$

Measurement update for state filter:

$$K_k^x = P_{\tilde{x},k}^-\hat{H}_k^\mathsf{T}\left[\hat{H}_kP_{\tilde{x},k}^-\hat{H}_k^\mathsf{T} + \hat{M}_kR\hat{M}_k^\mathsf{T}\right]^{-1} \tag{A.225}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k^x\left[y_k - h\left(\hat{x}_k^-,u_k,\hat{\theta}_k^-,\bar{v}\right)\right] \tag{A.226}$$

$$P_{\tilde{x},k}^+ = P_{\tilde{x},k}^- - K_k^xR_k\left(K_k^x\right)^\mathsf{T} \tag{A.227}$$

Measurement update for parameter filter:

$$K_k^\theta = P_{\tilde{\theta},k}^- \left(\hat{H}_k^\theta\right)^\mathsf{T} \left[\hat{H}_k^\theta P_{\tilde{\theta},k}^- \left(\hat{H}_k^\theta\right)^\mathsf{T} + \hat{M}_k^\theta R^\theta \left(\hat{M}_k^\theta\right)^\mathsf{T}\right]^{-1} \tag{A.228}$$

$$\hat{\theta}_k^+ = \hat{\theta}_k^- + K_k^\theta \left[y_k^\theta - h\left(\hat{x}_k^-, u_k, \hat{\theta}_k^-, \bar{e}\right)\right] \tag{A.229}$$

$$P_{\tilde{\theta},k}^+ = P_{\tilde{\theta},k}^- - K_k^\theta P_{\tilde{z},k} \left(K_k^\theta\right)^\mathsf{T}. \tag{A.230}$$

A.8.4 One-step Kalman filter equations

Alternatively, the prediction and correction step of the classical Kalman filter equations can be merged into single equations. This will be useful for the derivation of the Stenlund-Gustafsson filter in Section A.8.5. As derived more in detail in [Sim06], p. 131, one can combine the covariance propagation formula (A.77) with the posterior covariance update formula (A.73) and write for the *a priori* state error covariance update:

$$\begin{aligned} P_{k+1}^- &= F_k \left(P_k^- - K_k H_k P_k^-\right) F_k^\mathsf{T} + Q_k \\ &= F_k P_k^- F_k^\mathsf{T} - F_k K_k H_k P_k^- F_k^\mathsf{T} + Q_k \\ &= F_k P_k^- F_k^\mathsf{T} - F_k P_k^- H_k^\mathsf{T} \left(H_k P_k^- H_k^\mathsf{T} + R_k\right)^{-1} H_k P_k^- F_k^\mathsf{T} + Q_k \end{aligned} \tag{A.231}$$

where for the last equation, the identity $K_k = P_k^- H_k^\mathsf{T} \left(H_k P_k^- H_k^\mathsf{T} + R_k\right)^{-1}$ for the optimal Kalman gain as given in (A.75) was used. This last equation in (A.231) is termed a *time-discrete matrix Ricatti equation*. Similar, the following equations can be derived for the *a posteriori* state and error covariance matrix:

$$\begin{aligned} \hat{x}_k^+ &= (\mathbb{I} - K_k H_k)\left(F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1}\right) + K_k y_k \\ P_k^+ &= (\mathbb{I} - K_k H_k)\left(F_{k-1}P_{k-1}^+ F_{k-1}^\mathsf{T} + Q_{k-1}\right). \end{aligned} \tag{A.232}$$

For a system with unity dynamics $F_k = \mathbb{I}$ and no input, as given in parameter estimation problems, this simplifies to:

$$\begin{aligned} P_{k+1}^- &= (\mathbb{I} - K_k H_k) P_k^- + Q_k \\ &= P_k^- - K_k H_k P_k^- + Q_k \\ &= P_k^- - P_k^- H_k^\mathsf{T} \left(H_k P_k^- H_k^\mathsf{T} + R_k\right)^{-1} H_k P_k^- + Q_k. \end{aligned} \tag{A.233}$$

and

$$\begin{aligned} \hat{x}_k^+ &= (\mathbb{I} - K_k H_k)\,\hat{x}_{k-1}^+ + K_k y_k \\ P_k^+ &= (\mathbb{I} - K_k H_k)\left(P_{k-1}^+ + Q_{k-1}\right). \end{aligned} \tag{A.234}$$

Alternatively, we can instead also combine (A.74) and (A.77) to get *Josephs form of the one-step Kalman filter equation*:

$$P_{k+1}^- = F_k \left((\mathbb{I} - K_k H_k) P_k^- (\mathbb{I} - K_k H_k)^\mathsf{T} + K_k R_k K_k^\mathsf{T}\right) F_k^\mathsf{T} + Q_k. \tag{A.235}$$

A.8.5 Stenlund-Gustafsson anti-windup scheme

A modification of the Kalman filter for the least-squares parameter estimation problem is the Stenlund-Gustafsson anti-windup scheme originally proposed in Stenlund and Gustafsson [SG02]. It can be applied to parameter estimation problems which are Linear-in-Parameters and have a scalar observation. The algorithm can be interpreted as a process-noise adaptation scheme for Kalman filtering. Consider a scalar, Linear-in-Parameters observation of a constant parameter vector with the observation

$$y_k = \phi_k^\mathsf{T}\theta + \nu, \tag{A.236}$$

and a constant parameter model, but with some additive, white Gaussian noise

$$\theta_{k+1} = \theta_k + \omega_{\theta,k}. \tag{A.237}$$

Let us recall the Kalman filter equations for recursive estimation of the parameter vector θ: The parameter update is given by

$$\hat{\theta}_{k+1} = \hat{\theta}_k + K_k\left(y_k - \phi_k^\mathsf{T}\hat{\theta}_k\right) \tag{A.238}$$

with the optimal (in the minimum *a posteriori* parameter error covariance matrix sense [EM05]) Kalman gain

$$K_k = \frac{P_k^- \phi_k}{r_k + \phi_k^\mathsf{T} P_k^- \phi_k}. \tag{A.239}$$

The principle of the Stenlund-Gustafsson algorithm becomes obvious, by looking at the covariance matrix update formula, given in the *one-step update form*, which we derived in Section A.8.4. For the parameter estimation problem, for which the system dynamic matrix is the identity matrix, and a scalar observation, meaning that $R_k = r_k \in \mathbb{R}^1$, the discrete Ricatti equation from (A.233) becomes

$$P_{k+1}^- = P_k^- - \frac{P_k^- \phi_k \phi_k^\mathsf{T} P_k^-}{r_k + \phi_k^\mathsf{T} P_k^- \phi_k} + Q_k. \tag{A.240}$$

In the parameter estimation problem, the parameter covariance matrix Q can be seen as a design parameter. From (A.240), it can easily be seen that by choosing the parameter noise covariance matrix as suggested by Stenlund-Gustafsson [SG02]:

$$Q_k = \frac{P_d \phi_k \phi_k^\mathsf{T} P_d}{r_k + \phi_k^\mathsf{T} P_d \phi_k} \tag{A.241}$$

results in a system with the equilibrium point at the design matrix P_d. Further details and a profound discussion on convergence properties for the Stenlund-Gustafsson filter can be found in [EM05] and [ME09].

A.8.6 Instrumental Variables Kalman Filter

In many cases, the assumption, that in the measurement equation

$$y_k = \phi_k^\mathsf{T}\theta_k + \nu_k \tag{A.242}$$

only additive Gaussian output noise ν is present, does not hold in practice. Rather, one is faced with additional noise in the independent variable, such that

$$y_k = \left(\phi_k + \nu_{\phi,k}\right)^\mathsf{T}\theta_k + \nu_k. \tag{A.243}$$

This is known as the "Errors-in-Variables" problem [MM00; Söd07], which causes attenuation bias. To reduce this bias, various methods have been proposed. One method to mitigate these unwanted effects is the one of Instrumental Variabless (IVs) (see for example [PS12] and references therein), which was already proposed in the context of vehicle mass estimation [RH+16; Rho16]. See a discussion on alternative solutions to this more general problem in Section 6.3, and a simulation study which demonstrates the effect of both Errors-in-Variables and the method of IV in Section 6.4.2. Here, for the sake of brevity, we only want to quickly introduce the method in the context of Kalman Filtering for parameter estimation.

The method of Instrumental Variables introduces data variables called instruments, which need to be uncorrelated with the errors in the feature vector, but correlated with the true regressors. Delayed measurements of the regressors have been successfully used in the past to serve as instruments. For the method to work, the Errors-in-Variables needs to be independent and zero mean, and the regressors need to be changing slowly between measurement samples [MM00]. A realization of the method can be achieved within the Kalman Filtering framework by modifying the calculation of the Kalman Gain (A.75), (see Section A.6) to

$$K_k = \left(P_k^- \, \Phi_k^{\mathrm{IV}}\right)\left(r + \Phi^\mathsf{T} \, P_k^- \, \Phi_k^{\mathrm{IV}}\right)^{-1}, \tag{A.244}$$

where Φ_k^{IV} are the instruments, and typically just delayed values of $\Phi_k^{\mathrm{IV}} = \Phi_{k-n_{iv}}$ are used. We will use this technique in combination with other methods in the Kalman-like algorithms KF-IV (Algorithm 9), SG-KF-IV (Algorithm 11), M-SG-KF-IV (Algorithm 15).

A.9 Robust estimation

While both RLS and Kalman filtering where derived under the assumption of zero-mean, Gaussian noise, in practice, measurement distributions are often non-Gaussian since measurements include outliers. To take these effects into account, various robust estimation schemes have been developed. One possibility to take outliers into account is to assume noise distributions which are different from the Gaussian, and show heavier tails. Algorithms can the be derived, for example by maximum likelihood estimation schemes under the assumption that noise distributions are known. Unfortunately, typically no closed form solutions exist for such cases, and iterative algorithms have to be applied, with the drawback of increased computational load while only providing approximate solutions. Similar to the duality of a Gaussian noise assumption and the application of a quadratic cost function, alternatively to assuming a certain noise distribution with a known PDF, one can set a certain cost function instead. Here, any cost function which will penalize higher errors less than the quadratic function can be seen as a valid instrument to put less weight on outliers, and hence create a more robust behavior with respect to outliers. A general family of robust estimation schemes is M-estimation, which we will explain next in Section A.9.2. This builds the foundation for the derivation of the equations for a special member of this family, which is based on the assumption that measurement noise is distributed according to the Student-t distribution. We will use this algorithm in our proposed solution presented in Section 6.5. An interesting fact is that recursive M-estimators, solved using iteratively re-weighed least squares (of which only the first iteration is performed), can be seen as a general form of a Kalman filter, if a weight parameter is included. Using different cost functions to derive the algorithms then only have an influence on this weight parameter. This will be shown in Section A.9.6. For Gaussian noise, which is equivalent to squared losses, the resulting weight is the unit weight, and the algorithm is reduced to standard Kalman Filtering. This means that the M-estimator based on Kalman Filtering can be seen as a generalization of the Linear Kalman Filter.

A.9.1 Maximum likelihood estimation

A maximum likelihood estimate is the point where the likelihood function is maximized, and hence the derivative of the likelihood function with respect to the estimated parameter becomes zero. For independent and identically distributed observations / residuals e_i, the maximum likelihood estimate $\hat{\theta}$ satisfies

$$\hat{\theta} = \arg\max L(e,\theta) = \arg\max_{\theta}\left(\prod_{i=1}^{n} p\left(e_i,\theta\right)\right), \tag{A.245}$$

or, equivalently,

$$\hat{\theta} = \arg\min \ell(e,\theta) = \arg\min_{\theta}\left(\sum_{i=1}^{n} -\log\left(p\left(e_i,\theta\right)\right)\right). \tag{A.246}$$

The last expression is equivalent since taking the logarithm of an objective function does not change the location of the extremum, but has the nice property that the product of the terms becomes a sum after taking the logarithm. We used the likelihood function $L(e,\theta)$ and the log-likelihood function $\ell(e,\theta)$ in the expressions above, while the function p denotes the probability density function of a given probability distribution.

A.9.2 M-estimation

As outlined in section A.2.3, least squares regression can be seen as a special case of a more general class of estimation algorithms, namely Maximum Likelihood Estimation (MLE). An even broader class of estimation algorithms than maximum likelihood estimation is termed *M-estimation*. M-Estimators minimize the value of an objective function, which sums a number of m outcomes of a (nonlinear) weight function ρ, dependent on residuals e_i:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{m} \rho(e_i). \tag{A.247}$$

In M-estimation, the residuals are assumed to be models which are linear in the parameters of the form $e_i = \phi^{\mathsf{T}}\theta$. In the special case that the weighting function ρ is the negative log-likelihood of a probability distribution, the M-estimator becomes a maximum likelihood estimator.

For M-estimators, more general functions with certain properties can be used instead of the ones derived from true probability and likelihood functions. A very popular one is the Huber loss function, proposed originally by Huber [Hub73]. A recent survey of functions used in robust estimation can be found in [MP+21]. If derived from true probability density functions, densities which have heavier tails than the Gaussian distribution help to down-weight outliers, as can be seen from the next sections.

A.9.3 Iteratively re-weighted least squares

Finding a solution to the maximum likelihood problem in general involves the solution of a nonlinear function and no closed form algebraic expression can be derived. Such problems can be solved by iteratively re-weighted least squares, which we want to shortly present here.

The parameters minimizing the objective function as given in (A.247) fulfill the following equality, obtained by finding the root of the derivative of (A.247):

$$\sum_{i=1}^{n} \psi(e_i)\,\phi_i = 0, \tag{A.248}$$

with the *influence function* defined as $\psi = \frac{\partial \rho}{\partial e}$. Defining weights as $w_i(e_i) = \frac{\psi(e_i)}{e_i}$, this can be rewritten as

$$\sum_{i=1}^{n} w_i e_i \phi_i = \sum_{i=1}^{n} w_i \left(y_i - \phi_i^{\mathsf{T}}\theta\right)\phi_i = 0, \tag{A.249}$$

which is equivalent to solving a weighted least squares problem of the form

$$\min_{\theta} \sum_{i=1}^{n} w_i^2 e_i^2. \tag{A.250}$$

Since the weights $w_i(e_i(\theta))$ are depending on the unknown parameters, this can only be solved in a iterative manner, and the algorithm is termed Iteratively Re-weighted Least Squares (IRLS). At each iteration, the weights have to be updated and an updated parameter can be calculated using

$$\theta_{i+1} = \left(\Phi^{\mathsf{T}} W_i \Phi\right)^{-1} \Phi^{\mathsf{T}} W_i y, \tag{A.251}$$

with the regressor matrix Φ composed of rows of vectors ϕ^{T}, and the diagonal weight matrix $W = \operatorname{diag}\{w_i\}$. As shown for example in [Gre84], the iteratively re-weighted least squares algorithm can be derived from performing Newton-Raphson steps at each iteration. A more general derivation along this lines looks like the following: Let us aim to find the maximum of the log-likelihood function $\ell(\theta)$

$$\ell(e(\theta)) = \sum_{i=1}^{m} -\log\left(p\left(e(\theta)\right)\right), \tag{A.252}$$

which is a function of the residuals $e(\theta)$, which is again a function of the unknown parameter vector θ. We need to find θ such that the gradient of the likelihood fulfills

$$\nabla \ell(e(\theta)) = J_\ell = \frac{\partial}{\partial \theta}\ell = \left(\frac{\partial e}{\partial \theta}\right)^{\mathsf{T}} \frac{\partial \ell}{\partial e} = D^{\mathsf{T}}\alpha = 0, \tag{A.253}$$

where the matrix $D = \frac{\partial e}{\partial \theta}$ and a vector $\alpha = \frac{\partial \ell}{\partial e}$, as well as the Jacobian J_ℓ were introduced. As shown in [Gre84], a Newton-Raphson step for an iterative solution would be to solve the system of linear equations:

$$-H_\ell(\theta)\left(\theta_{i+1} - \theta_i\right) = J_\ell = D^{\mathsf{T}}\alpha, \tag{A.254}$$

with the Hessian H_ℓ. Expanding H_ℓ yields

$$\frac{\partial}{\partial \theta}\left(\left(\frac{\partial e}{\partial \theta}\right)^{\mathsf{T}} \frac{\partial \ell}{\partial e}\right) = \sum \frac{\partial \ell}{\partial e_i} \frac{\partial^2 e_i}{\partial \theta \partial \theta^{\mathsf{T}}} + \left(\frac{\partial e_i}{\partial \theta}\right)^{\mathsf{T}} \frac{\partial^2 \ell}{\partial e \partial e^{\mathsf{T}}} \left(\frac{\partial e_i}{\partial \theta}\right). \tag{A.255}$$

A general strategy to simplify the expression of the Hessian is by approximating the terms on the right by their expectations at the current parameter values (also denoted *Fisher scoring* [Gre84]). Then, the Hessian can be written as

$$H_\ell(\theta) = D^{\mathsf{T}} W D, \tag{A.256}$$

where the weight matrix $W = \mathbb{E}\left[\frac{\partial \ell}{\partial e}\left(\frac{\partial \ell}{\partial e}\right)^{\mathsf{T}}\right]$ was introduced. The Newton step can then be rewritten as

$$\theta_{i+1} = \theta_i - (D^{\mathsf{T}} W D)^{-1} D^{\mathsf{T}} \alpha \tag{A.257}$$

$$\theta_{i+1} = (D^{\mathsf{T}} W D)^{-1} D^{\mathsf{T}} W \underbrace{\left(D\theta_i - W^{-1}\alpha\right)}_{z}. \tag{A.258}$$

In the last expression, matrices where factored out in order to see that this is the solution to a weighted least squares problem of the from

$$\hat{\theta} = \arg\min_{\theta} (W^{-1}\alpha + D(\theta - \theta^*)^{\mathsf{T}} W (W^{-1}\alpha + D(\theta - \theta^*). \tag{A.259}$$

In the case that the residuals are given by $e = y - \Phi\theta$, then $D = -\Phi$. For independent residuals, the weighting matrix results into a diagonal matrix, and the weighted least squares problem simplifies with the diagonal entries w_i of the weight matrix to:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{m} w_i (z_i - \phi_i^{\mathsf{T}} \theta)^2, \tag{A.260}$$

while the Newton step is found as the solution to

$$\Phi^{\mathsf{T}} W \Phi (\theta_{i+1} - \theta_i) = \Phi^{\mathsf{T}} \frac{\partial \ell}{\partial e}. \tag{A.261}$$

A.9.4 Maximum likelihood estimation for normal distribution

If we want to minimize the residuals e_k of a linear model given by $e_k = y_k - \phi_k^{\mathsf{T}} \theta$ under the assumption that the residuals follow a Gaussian normal distribution $p_n(e)$, given by the PDF

$$p_n(e) = \frac{1}{\sigma_n \sqrt{2\pi}} \exp\left(-\frac{(e - \mu_n)^2}{2\sigma_n^2}\right). \tag{A.262}$$

Taking the negative logarithm for the normal PDF yields

$$-\log p_n(e) = -\log\left(\frac{1}{\sigma_n \sqrt{2\pi}}\right) + \left(\frac{1}{2\sigma_n^2}\right)(e - \mu_n)^2 = C_1 + C_2 (e - \mu_n)^2. \tag{A.263}$$

Since both the additive and multiplicative constants do not alter the location of an extremum of the above function, the maximum likelihood estimation under the assumption of normally distributed residuals with zero mean, can easily be seen equivalent to minimizing the square of the residuals:

$$\hat{\theta}_{\text{MLE},\mathbb{N}} = \arg\min_{\theta} \sum (e_i)^2, \qquad e \sim \mathcal{N}(0, \sigma_n). \tag{A.264}$$

A.9.5 Maximum likelihood estimation for Student-t distribution

Maximum likelihood estimation based on the Students-t distribution was already used in the context of vehicle and parameter estimation in Altmannshofer, Endisch, et al. [AE+16]. Since our proposed method will include, in parts, this method, we want to shortly present some of the background, following the lines of Altmannshofer, Endisch, et al. [AE+16].

The Student-t distribution was originally developed to represent “the frequency distribution of standard deviations of samples drawn from a normal population” [Stu08]. An interesting side fact is that the original paper was published by William Sealy Gosset under the pseudonym "Student", to hide the fact that his employer, the Guinnes Brewery, was employing statisticians [Box81]. The PDF of the non-standardized Student-t distribution, is given, for the uni-variate case as

$$p_S(e(\theta)) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{1}{\sigma_S^2 \nu\pi}\right)^{1/2}\left[\frac{\nu+\left(\frac{e_i(\theta_i)-\mu_S}{\sigma_S}\right)^2}{\nu}\right]^{-\frac{\nu+1}{2}}, \tag{A.265}$$

where Γ represents the Gamma function, and ν is an additional parameter which describes the distribution, termed the degrees of freedom. The Student-t distribution is similar in shape than the normal distribution, but has heavier tails. This leads to maximum likelihood estimators based on Student-t being more robust to outliers. Taking the logarithm of the probability density function yields

$$-\log\left(p_S(e(\theta)\right) = C_1 + C_2 \log\left[\frac{\nu+\left(\frac{e_i(\theta_i)-\mu_S}{\sigma_S}\right)^2}{\nu}\right]. \tag{A.266}$$

The maximum likelihood estimate for the Student-t distribution $e \sim \mathcal{D}_{\text{Student-t}}(0, \sigma_S, \nu)$ becomes the result of an M-estimator:

$$\hat{\theta}_{\text{MLE,St}} = \arg\min_{\theta} \sum \rho\left(e_i\left(\theta_i\right)\right), \tag{A.267}$$

with

$$\rho\left(e\left(\theta\right)\right) = \log\left[\frac{\nu+\left(\frac{e(\theta)}{\sigma_S}\right)^2}{\nu}\right]. \tag{A.268}$$

This is a nonlinear optimization problem, which can be solved, for example, using IRLS (see section A.9.3). The influence function $\psi = \partial\rho/\partial e$ then takes the form

$$\psi\left(e\left(\theta\right)\right) = \frac{2\,e}{\sigma_S^2\,\nu + e^2}, \tag{A.269}$$

and the weights $w_i = \psi(e_i)/e_i$ for IRLS (see Section A.9.3) are calculated as

$$w_i = \frac{2}{\sigma_S^2\,\nu + e_i^2}. \tag{A.270}$$

Note that the variance of the Student-t distribution, although σ_S is a scale parameter, is *not* given by σ_S^2, but defined (only for $\nu > 2$) as follows:

$$\text{Var}(X_{\sim\text{St}}) = \sigma_S^2 \frac{\nu}{\nu-2}. \tag{A.271}$$

A.9.6 M-estimation using Kalman filtering techniques

As we showed in Section A.9.2, the M-estimation problem can be solved by an iterated, reweighted least squares algorithm. A recursive version of weighted least squares estimation in the context of M-estimation was previously proposed in [ZC+00], while they termed their algorithm *a recursive least M-Estimate (RLM) adaptive filter for robust filtering*. Based on this, [RH+16] presented an algorithm in a form similar to the least-squares formulation

Kalman filter and proposed its application to estimate longitudinal vehicle parameters. In other words, [ZC+00] used a formulation based on a forgetting factor λ, while [RH+16] used a formulation using parameter noise covariance Q to achieve a forgetting. Additionally, they proposed to set Q to a value equivalent to the one proposed by Stenlund and Gustafsson [SG02] in order to avoid wind-up under lack of excitation. We want to shortly derive the equations for the recursive, Kalman-like M-estimator in the following. Remember that the iterated re-weighted least squares solution to the M-estimation problem minimizes

$$J(e) = \sum \rho(e_i), \tag{A.272}$$

which is equivalent to minimizing

$$J(e) = \sum e_i^\mathsf{T} w(e_i) e_i. \tag{A.273}$$

The regular Kalman filter for a constant state model, as it is the case for parameter estimation, seeks to minimize a squared cost function $J(e) = \sum e_i^\mathsf{T} e_i$, but uses a regularized cost function to incorporate prior knowledge instead:

$$J_{KF} = \|x - x_0\|^2_{P_0^{-1}} + \|e\|^2_{R^{-1}}. \tag{A.274}$$

The weights are inverses of the state error covariance matrix P_0 and the measurement noise covariance matrix R. Since the cost function for the M-estimator is a weighted sum in the first place, one has to additionally consider this weight in the error terms to derive the equations for the robust Kalman filter:

$$J_{KF} = \|x - x_0\|^2_{P_0^{-1}} + \|e\|^2_{R^{-1}W}. \tag{A.275}$$

If we define a new modified inverse weight matrix $\tilde{R}^{-1} = R^{-1}W$, the cost function becomes identical to the one for the Kalman filter. We therefor can use the same equations as for the Kalman filter, but have to replace the measurement covariance matrix R with $\tilde{R} = W^{-1}R$. The Kalman gain is then given by

$$K_k = P_k^- \phi \left(\tilde{R} + \phi^\mathsf{T} P_k^- \phi \right)^{-1}, \tag{A.276}$$

which can we rewritten in terms of R and the (scalar) weight w_k as

$$K_k = w_k P_k^- \phi \left(R + w_k \phi^\mathsf{T} P_k^- \phi \right)^{-1}, \tag{A.277}$$

where we factored out the weight parameter. The discrete Ricatti equation will then have the form:

$$P_{k+1}^- = P_k^- - K_k \phi_k^\mathsf{T} P_k^- + Q_k \tag{A.278}$$

$$= P_k^- - w_k P_k^- \phi \left(R + w_k \phi_k^\mathsf{T} P_k^- \phi \right)^{-1} \phi_k^\mathsf{T} P_k^- + Q_k. \tag{A.279}$$

The full algorithm, including Stenlund-Gustafsson adaptation, is given as follows: Considering the system

$$\theta_k = \theta_{k-1} + \tilde{\omega}_{k-1} \tag{A.280}$$

$$y_k = \phi_k^\mathsf{T} \theta_k + \tilde{\nu}_k. \tag{A.281}$$

The robust, Kalman filter like M-estimation procedure with Stenlund-Gustafsson noise covariance adaptation works as follows:

1. Initialize filter

$$\hat{\theta}_0 = \mathbb{E}[x_0] \tag{A.282}$$

$$P^- = \mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^\mathsf{T}\right] \tag{A.283}$$

$$\hat{P}_d = \text{user defined} \tag{A.284}$$

2. For $k = 1, 2, \ldots$ perform:

$$e_k = \left(y_k - \phi_k^\mathsf{T}\hat{\theta}_{k-1}\right) \tag{A.285}$$

$$w_k = \frac{\psi(e_k)}{e_k}, \quad \text{depending on influence function} \tag{A.286}$$

$$K_k = \left(w_k P_k^- \phi_k\right)\left(R_k + w_k \phi_k^\mathsf{T} P_k^- \phi_k\right)^{-1} \tag{A.287}$$

$$\hat{\theta}_k = \hat{\theta}_{k-1} + K_k e_k \tag{A.288}$$

$$Q_k = w_k P_d \phi \left(R + w_k \phi_k^\mathsf{T} P_d \phi\right)^{-1} \phi_k^\mathsf{T} P_d \tag{A.289}$$

$$P_{k+1}^- = (\mathbb{I} - K_k \phi_k) P_k^- + Q_k. \tag{A.290}$$

The algorithm above is identical to the one given in [RH+16], if one rewrites the matrix inverse in the form of a scalar denominator, as possible for the scalar case.

Proposed modification of error covariance update in M-estimation algorithm We want to introduce a slight modification to the error covariance update in equation (A.290) of the M-estimation algorithm presented in Section A.9.6. The modification can be derived showing that one can split the one-step update into

$$P_k^+ = (\mathbb{I} - K_k \phi_k) P_k^- \tag{A.291}$$

$$P_{k+1}^- = P_k^+ + Q_k, \tag{A.292}$$

and alternatively use the update formula known as *Joseph's form*, see [Sim06] and (A.74). In order to adapt this to the given problem of M-estimation, we again have to replace the measurement covariance matrix R with $\tilde{R} = W^{-1}R$, which yields:

$$P_k^+ = (\mathbb{I} - K_k H_k) P_k^- (\mathbb{I} - K_k H_k)^\mathsf{T} + K_k \tilde{R}_k K_k^\mathsf{T} \tag{A.293}$$

$$= (\mathbb{I} - K_k H_k) P_k^- (\mathbb{I} - K_k H_k)^\mathsf{T} + K_k \frac{1}{w_k} R_k K_k^\mathsf{T}. \tag{A.294}$$

Using *Joseph's form* is known to guarantee a positive definite covariance matrix as a result (see [Sim06]). This formulation is, for the optimal Kalman Gain, mathematically identical to the version used in (A.290), but numerically more stable. We introduced this formulation in an attempt to make Kalman Filtering using Instrumental Variables more stable, where we found a tendency of the error covariance matrix to become indefinite. A more detailed discussion can be found in Section 6.4.3. Nevertheless, as we will see there, this could not solve the difficulties we faced when the error covariance matrix did not stay positive definite when applying the method of Instrumental Variables IV.

A.10 Observability, identifiability and persistent excitation

The reconstruction of the full state and parameter information from lower-dimensional output data is only possible under certain conditions, which we want to summarize here. Next, we want to clarify some taxonomy and provide background and definitions.

A.10.1 State observability

The notion of (state) *observability* is widely used to express the conditions to reconstruct the *state* information from input-output data of dynamical systems. A system is observable only if there is a unique relationship between the input-output behavior and the system state. If only a subspace of the full state space information can be observed, a system is generally not observable, and some of the states remain unobservable.

A.10.2 Detectability

A slightly weaker definition than observablility is the notion of *detectability*. A system is said to be detectable, when all the unobservable states are stable, or in other words, if all unstable states are observable (for a more formal definition, see for example Grimm, Messina, Tuna, and Teel [GM+05]). A special case exists, for which the unobservable states are stable. Then it might be sufficient to ignore the unobservable states, and we might be able to stabilize the system without knowledge of these stable substates. This is commonly termed *detectability*. The concept of observability was introduced by Rudolf E. Kalman [Kal60a] for linear systems together with its dual *controllability*. Detectability can be regarded as the dual of stabilizability in control systems theory.

A.10.3 Identifiability

When analyzing systems with unknown parameters, the concept of *parameter identifiability* aims to answer the question if the system parameters can be estimated by observing the system output. The mathematical property of *structural identifiability* was introduced to describe, if it is theoretically possible to determine the true value of a parameter from observations of the model output [CD80; God99; VE+19]. While structural identifiability does not take into account limitations which may arise from data quantity or quality, practical identifiability also accounts for these effects and aims to describe if a system, given certain input-output data, is parameter identifiabel in practice.

Identifiability can be analyzed for example by augmenting the state by the unknown parameters and performing an observability analysis. As noted for example by [VE+19] and references therein, for general nonlinear systems, structural identifiability does not only depend on the system dynamics but also on:

- specific values of initial conditions
- the system output definition
- the type of perturbations introduced to the system by its input signals

The last point is related to the notion of *persistent excitation*, which is discussed in Section A.10.4. Necessary and sufficient conditions for observability and identifiablility were provided by various authors, probably starting from Kalman [Kal60a] for linear systems, Walter [Wal82] for nonlinear discrete and continuous system, and Isidori [Isi85] for nonlinear continuous systems. An extension to nonlinear DAEs can be found in Gerdin [Ger06].

In order to provide necessary conditions for observability and identifiability, we consider the general description of a nonlinear and time-continuous system Σ in state space form given as

$$\Sigma : \begin{cases} \dot{x}(t) & = f(x(t), u(t), w(t), \theta)) \\ y(t) & = h(x(t), \theta) \\ x_0 & = x(t_0, \theta), \end{cases} \tag{A.295}$$

where f and h are analytic vector functions, $\theta \in \mathbb{R}^q$ is the vector of system parameters, $u(t) \in \mathbb{R}^r$ the input vector of known inputs (which might include known disturbances together with control inputs), $w(t) \in \mathbb{R}^s$ the input vector of unknown inputs or disturbances, $x(t) \in \mathbb{R}^n$ the state variable vector and $y(t) \in \mathbb{R}^m$ the output vector. For the purpose of jointly estimating the parameters θ, system Σ will be enhanced by $\tilde{x} = (x, \theta)$ and $\dot{\theta} = 0$. Note that for the purpose of determining the structural identifiability, we only consider a deterministic system without any process and measurement noise. The identifiability of the augmented system can be determined by calculating the rank of the observability-identifiabilty matrix $\mathcal{O}_I(\tilde{x})$, see for example [VE+19], which is defined as

$$\mathcal{O}_I(\tilde{x}) = \begin{pmatrix} \frac{\partial}{\partial \tilde{x}} h(\tilde{x}) \\ \frac{\partial}{\partial \tilde{x}} (\mathcal{L}_f h(\tilde{x})) \\ \frac{\partial}{\partial \tilde{x}} (\mathcal{L}_f^2 h(\tilde{x})) \\ \vdots \\ \frac{\partial}{\partial \tilde{x}} (\mathcal{L}_f^{n+q-1} h(\tilde{x})) \end{pmatrix}, \tag{A.296}$$

with $\mathcal{L}_f h(\tilde{x})$ being the Lie derivative of h with respect to f, defined by:

$$\mathcal{L}_f h(\tilde{x}) = \frac{\partial h(\tilde{x})}{\partial \tilde{x}} f(\tilde{x}, u), \tag{A.297}$$

and higher order Lie derivatives being recursively calculated as:

$$\mathcal{L}_f^2 h(\tilde{x}) = \frac{\partial \mathcal{L}_f h(\tilde{x})}{\partial \tilde{x}} f(\tilde{x}, u), \tag{A.298}$$

$$\vdots \tag{A.299}$$

$$\mathcal{L}_f^i h(\tilde{x}) = \frac{\partial \mathcal{L}_f^{i-1} h(\tilde{x})}{\partial \tilde{x}} f(\tilde{x}, u). \tag{A.300}$$

For constant inputs $u(t) = u$, $\mathcal{O}_I(\tilde{x})$ this yields to

$$\mathcal{O}_I(\tilde{x}) = \begin{pmatrix} \frac{\partial}{\partial \tilde{x}} y(t) \\ \frac{\partial}{\partial \tilde{x}} \dot{y}(t) \\ \frac{\partial}{\partial \tilde{x}} \ddot{y}(t) \\ \vdots \\ \frac{\partial}{\partial \tilde{x}} y^{n+q-1}(t) \end{pmatrix}, \tag{A.301}$$

and the rank test of the Observability-Identifiability Condition (OIC) can be directly applied:

Theorem 1 (Observability-Identifiability Condition (OIC, see [VE+19], Section D, Theorem 2)). *If the system Σ given by (A.295) with constant input u satisfies rank $\{\mathcal{O}_I(\tilde{x}_0)\} = n + q$, with $\mathcal{O}_I(\tilde{x}_0)$ given by (A.301), then it is observable and (at least locally) identifiable in a neighborhood $N(\tilde{x}_0)$ of $\tilde{x}_0$.*

Villaverde, Evans, Chappell, and Banga [VE+19] recently proposed advances in nonlinear system identification and offer a methodology of determining whether a certain time-varying input is sufficiently exciting to guarantee structural identifiability by considering if the time derivative of this input must be non-zero. In their proposal, they consider extended Lie derivatives to correctly consider the effect of time-varying inputs, and assesses whether an observability-identifiability matrix has full rank. Previous methods might have failed and wrongly classified a identifiable model as unidentifiable, if a time-varying input is required and a constant input would not excite the system dynamics sufficiently. The extended Lie derivative in case of time-varying inputs $u(t)$, defined for output functions which do not directly depend on the input ($h(\tilde{x})$ and not $h(\tilde{x},u)$) by:

$$\mathcal{L}_f h(\tilde{x}) = \frac{\partial h(\tilde{x})}{\partial \tilde{x}} f(\tilde{x},u), \tag{A.302a}$$

$$\mathcal{L}_f^i h(\tilde{x}) = \frac{\partial \mathcal{L}_f^{i-1} h(\tilde{x})}{\partial \tilde{x}} f(\tilde{x},u) + \sum_{j=0}^{j=i-2} \frac{\partial \mathcal{L}_f^{i-1} h(\tilde{x})}{\partial u^{(j)}} u^{(j+1)}, \tag{A.302b}$$

with $u^{(j)}$ and $u^{(j+1)}$ denoting the j^{th} and the $(j+1)^{th}$ derivatives of the input $u(t)$. This preserves the correspondence between output derivatives and Lie derivatives and allows to perform the rank test with conditions for the derivatives of the input variable. For example, if $\mathcal{O}_I(\tilde{x})$ has full rank for $\dot{u}(t)=0$, a constant input is sufficient for structural identifiability. If $\mathcal{O}_I(\tilde{x})$ does not have full rank for $\dot{u}(t)=0$ but for $\dot{u}(t)\neq 0, \ddot{u}(t)=0$, a input ramp is necessary. An extension for additionally considering the estimation of unknown inputs is given in [VT+19]. They also provide a software tool written in the language MATLAB®, called Strike-Goldd2, which is licensed under the GNU General Public License v3 (GPLv3) [VE+19]. We use Strike-Goldd2 [VT+19] for examining the structural identifiabililty properties of the longitudinal vehicle motion dynamics in Section 6.3.2.

A.10.4 Conditions for convergence of recursive least squares

Typical assumptions for recursive least squares include the measurement noise to be independent, zero-mean and Gaussian, perfect knowledge of the feature vector ϕ as well as the input signal to be persistently excited. Before giving formal definitions of what persistently exciting inputs are, we want to give some intuition for the concept.

Persistent excitation

From the information update equation in (A.59), we could see that the term $\phi_k \phi_k^\mathsf{T}$ is the reason for the information matrix to increase in recursive least squares. Comparing this to the batch version solution to the over-determined linear least squares regression problem in (A.45), we learned that one is able to obtain a solution by calculating the Moore-Penrose Pseudo Inverse $\Phi^\dagger = (\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}$, which is only well defined if the matrix $(\Phi^\mathsf{T}\Phi)^{-1}$ is invertible, and in other words, has full rank. Similar to that, various definitions have been proposed for necessary conditions on a regressor signal in the context of recursive least squares estimation. Examples can be found for example in Gaudio [Gau20] for the continuous time case, and in Johnstone, Johnson, Bitmead, and O. Anderson [JJ+82] and Shin and Lee [SL20] for the time-discrete case, where the former defines:

Definition 2. *([JJ+82]). We say that the vector sequence $\{\phi_k\}$ is* ***persistently exciting (PE)*** *if* *for some constant integer S and all j there exist positive constants α and β such that*

$$0 < \alpha \mathbb{I} \leq \sum_{i=j}^{j+S} \phi_i \phi_i^{\mathsf{T}} \leq \beta \mathbb{I} < \infty. \tag{A.303}$$

B

Overview of Algorithms

"Pure mathematics is, in its way, the poetry of logical ideas."

– Albert Einstein

Here we want to present an overview of some of the most important existing estimation algorithms used in this thesis. Please note that this list is not comprehensive, since some of the algorithms, especially the ones proposed in this thesis, can be found directly within each chapter.

Linear Kalman Filter The linear Kalman Filter (KF) can be found in Algorithm 7.

Algorithm 7 Linear Kalman Filter [Kal60a].

```
Require: $F, B, H$
1: Inputs:
       $\hat{x}^+_{k-1}, P^+_{k-1}, Q_{k-1}, R_k, u_{k-1}, y_k$
2: $P^-_k = FP^+_{k-1}F^\mathsf{T} + Q_{k-1}$                          ▷ A priori estimation error covariance (A.77)
3: $K_k = P^-_k H^\mathsf{T} \left(HP^-_k H^\mathsf{T} + R_k\right)^{-1}$        ▷ (optimal) Kalman Gain (A.75)
4: $\hat{x}^-_k = F\hat{x}^+_{k-1} + Bu_{k-1}$                         ▷ A priori state estimate
5: $\hat{x}^+_k = \hat{x}^-_k + K_k(y_k - H\hat{x}^-_k)$                ▷ A posteriori state estimate
6: $P^+_k = P^-_k - K_kHP^-_k = (\mathbb{I} - K_kH)P^-_k$
       $= (\mathbb{I} - K_kH)P^-_k(\mathbb{I} - K_kH)^\mathsf{T} + K_kR_kK_k^\mathsf{T}$   ▷ Estimation error covariance in standard or Joseph's form
7: return $\hat{x}^+_k, P^+_k$
```

Kalman Filter version for parameters-only estimation (KF) For the parameters-only estimation problem, the following state space formulation can be used:

$$\theta_k = \theta_{k-1} + \omega^\theta_{k-1} \tag{B.1}$$

$$y^\theta_k = \phi^\mathsf{T}_k\theta + \nu_k. \tag{B.2}$$

Applying the linear Kalman Filter to this system yields the Kalman Filter for parameters-only estimation, as given in Algorithm 8.

Algorithm 8 KF: Kalman Filter for parameters-only estimation [Kal60a].

```
Require: $r$
1: Inputs:
       $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$, $Q_k$
2: $e_k = y_k^\theta - \Phi_k^\mathsf{T} \hat{\theta}_k$                              ▷ Prediction error
3: $K_k = \left(P_k^- \Phi_k\right)\left(r + \Phi^\mathsf{T} P_k^- \Phi_k\right)^{-1}$                    ▷ Kalman Gain
4: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_k e_k$                    ▷ Parameter estimate correction
5: $P_{k+1}^- = \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right) P_k^- \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right)^\mathsf{T} + K_k r K_k^\mathsf{T}$   ▷ Covariance update, Joseph's form
6: $P_{k+1}^- = P_k^+ + Q_k$                 ▷ A priori estimation error covariance
7: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Kalman Filter with Instrumental Variables (KF-IV) For the Kalman Filter with Instrumental Variables, we used instruments Φ_k^{IV} such that $\Phi_k^{\mathrm{IV}} = \Phi_{k-n_{iv}}$ with a delay of n_{iv} samples.

Algorithm 9 KF-IV: Kalman Filter with Intrumental Variables. Modified from [Rho16].

```
Require: $r$
1: Inputs:
       $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$, $\Phi_k^{\mathrm{IV}}$, $Q_k$
2: $e_k = y_k^\theta - \Phi_k^\mathsf{T} \hat{\theta}_k$                              ▷ Prediction error
3: $K_k = \left(P_k^- \Phi_k^{\mathrm{IV}}\right)\left(r + \Phi^\mathsf{T} P_k^- \Phi_k^{\mathrm{IV}}\right)^{-1}$               ▷ Kalman Gain for IV
4: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_k e_k$                    ▷ Parameter estimate update
5: $P_k^+ = \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right) P_k^- \left(\mathbb{I} - K_k \left(\Phi_k^{\mathrm{IV}}\right)^\mathsf{T}\right)^\mathsf{T} + K_k r K_k^\mathsf{T}$   ▷ modified Joseph's form for IV
6: $P_{k+1}^- = P_k^+ + Q_k$                 ▷ A priori estimation error covariance
7: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Kalman Filter with Stenlund-Gustafsson (SG) Process Noise Covariance Adaptation (SG-KF) The Kalman Filter with Stenlund-Gustafsson (SG) process noise covariance adaptation is provided in Algorithm 10.

Kalman Filter with Stenlund-Gustafsson (SG) Process Noise Covariance Adaptation and Instrumental Variables (SG-KF-IV) The Stenlund-Gustafsson Kalman Filter with Intrumental Variables is can be found in Algorithm 11.

Kalman Filter with Stenlund-Gustafsson (SG) Process Noise Covariance Adaptation with Contraints (SG-KF-C) The Kalman Filter with Stenlund-Gustafsson (SG) process noise covariance adaptation which additionally considers constraints is provided in Algorithm 12.

Robust M-Estimation Kalman Filter (M-KF) A robust version of the M-Estimation Kalman Filter is given in Algorithm 13.

Algorithm 10 SG-KF: Stenlund-Gustafsson Kalman Filter. Modified from [Rho16].

```
Require: $P_d$, $r$, $\nu_s$, $\sigma_s$
 1: Inputs:
        $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$
 2: $e_k = y_k^\theta - \Phi_k^\mathsf{T} \hat{\theta}_k$                                    ▷ Prediction error
 3: $K_k = \left(P_k^- \Phi_k\right)\left(r + \Phi^\mathsf{T} P_k^- \Phi_k\right)^{-1}$          ▷ Kalman Gain
 4: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_k e_k$                                   ▷ Parameter estimate correction
 5: $Q_k = \left(P_d \Phi_k \Phi_k^\mathsf{T} P_d\right)\left(r + \Phi_k^\mathsf{T} P_d \Phi_k\right)^{-1}$     ▷ SG cov. adaptation [SG02]
 6: $P_k^+ = \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right) P_k^- \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right)^\mathsf{T} + K_k r K_k^\mathsf{T}$   ▷ Covariance update, Joseph's form
 7: $P_{k+1}^- = P_k^+ + Q_k$                                  ▷ A priori estimation error covariance
 8: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Algorithm 11 SG-KF-IV: Stenlund-Gustafsson Kalman Filter with Instrumental Variables. Modified from [Rho16].

```
Require: $P_d$, $r$, $\nu_s$, $\sigma_s$
 1: Inputs:
        $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$, $\Phi_k^{\mathrm{IV}}$
 2: $e_k = y_k^\theta - \Phi_k^\mathsf{T} \hat{\theta}_k$                                    ▷ Prediction error
 3: $K_k = \left(P_k^- \Phi_k^{\mathrm{IV}}\right)\left(r + \Phi^\mathsf{T} P_k^- \Phi_k^{\mathrm{IV}}\right)^{-1}$       ▷ Kalman Gain for IV
 4: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_k e_k$                                   ▷ Parameter estimate update
 5: $Q_k = \left(P_d \Phi_k \Phi_k^\mathsf{T} P_d\right)\left(r + \Phi_k^\mathsf{T} P_d \Phi_k\right)^{-1}$     ▷ SG cov. adaptation [SG02]
 6: $P_k^+ = \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right) P_k^- \left(\mathbb{I} - K_k \left(\Phi_k^{\mathrm{IV}}\right)^\mathsf{T}\right)^\mathsf{T} + K_k r K_k^\mathsf{T}$   ▷ modified Joseph's form for IV
 7: $P_{k+1}^- = P_k^+ + Q_k$                                  ▷ A priori estimation error covariance
 8: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

M-Estimation Stenlund-Gustafsson Kalman Filter (M-SG-KF) The M-Estimation Stenlund-Gustafsson Kalman Filter as given in Algorithm 14 combines robust M-Estimation based on the Student-t distribution as proposed by [AE+16] with Stenlund-Gustafsson anti-windup [SG02].

M-Estimation Stenlund-Gustafsson Kalman Filter with Instrumental Variables (M-SG-K-F-IV) Find the M-Estimation Stenlund-Gustafsson Kalman Filter with Intrumental Variables (M-SG-KF-IV) in Algorithm 15. It combines robust M-Estimation, with the anti-windup mechanism from [SG02] together with Instrumental Variables. We use a modified variant which uses Joseph's form for the covariance update, which is essentially a combination to the one proposed by [Rho16] and the weight derived from the Student-t distribution as given in [AE+16].

M-Estimation Stenlund-Gustafsson Kalman Filter with Constraints For the M-Estimation Stenlund-Gustafsson Kalman Filter with Constraints as in Algorithm 16, we used the projection method suggested by Altmannshofer, Endisch, et al. [AE+16], which is realized by Lemke's pivoting algorithm as found in Almqvist [Alm19]. The Linear Complementary

Algorithm 12 SG-KF-C: Stenlund-Gustafsson Kalman Filter with Contraints. Modified from [AE+16].

```
Require: $P_d$, $r$, $c = [\theta_{\max}, \theta_{\min}]^\mathsf{T}$, $L = [\mathbb{I}, -\mathbb{I}]^\mathsf{T}$
 1: Inputs:
        $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$
 2: $e_k = y_k^\theta - \Phi_k^\mathsf{T}\hat{\theta}_k$                                  ▷ Prediction error
 3: $K_k = \left(P_k^- \Phi_k\right)\left(r + \Phi^\mathsf{T} P_k^- \Phi_k\right)^{-1}$       ▷ Kalman Gain
 4: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_k e_k$                         ▷ Parameter estimate correction
 5: $Q_k = \left(P_d \Phi_k \Phi_k^\mathsf{T} P_d\right)\left(r + \Phi_k^\mathsf{T} P_d \Phi_k\right)^{-1}$   ▷ SG cov. adaptation [SG02]
 6: $P_k^+ = \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right) P_k^- \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right)^\mathsf{T} + K_k r K_k^\mathsf{T}$   ▷ Covariance update, Joseph's form
 7: $q_k = c - L\hat{\theta}_k^+$          ▷ Offset for Linear Complementarity Problem (LCP)
 8: if $q_k < 0$ then
 9:     $M_k = L P_k^+ L^\mathsf{T}$                                       ▷ Matrix for LCP
10:     $\eta = \mathrm{LCPSolve}(M, q)$                                  ▷ Solution of LCP
11:     $\hat{\theta}_{k+1} = \hat{\theta}_k^- - P_k^+ L^\mathsf{T} \eta$          ▷ Constrained parameter estimate
12: end if
13: $P_{k+1}^- = P_k^+ + Q_k$                       ▷ A priori estimation error covariance
14: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Program was solved using the MATLAB® function provided by Almqvist [Alm19].

M-Estimation Stenlund-Gustafsson Kalman Filter with Instrumental Variables and constraints (M-SG-KF-IV-C) Find the M-Estimation Stenlund-Gustafsson Kalman Filter with Intrumental Variables (M-SG-KF-IV) and constraints in Algorithm 17. It combines robust M-Estimation, with the anti-windup mechanism from [SG02] together with Instrumental Variables. Additionally, constraints are enforced in the fashion as used by [AE+16]. We use a modified variant which uses Joseph's form for the covariance update, which is essentially a combination to the one proposed by [Rho16] and the weight derived from the Student-t distribution as given in [AE+16].

Runge-Kutta method For completeness, we also want to provide the Runge-Kutta algorithm [Run95]. With the general continuous-time state space description of the longitudinal vehicle dynamics as given in (3.69a), we can approximate the integration using a fourth order Runge-Kutta algorithm as described in Algorithm 18.

Algorithm 13 M-KF: M-Estimation Kalman Filter. Modified from [Rho16].

```
Require: $r$, $\nu_s$, $\sigma_s$
1: Inputs:
       $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$
2: $e_k = y_k^\theta - \Phi_k^\mathsf{T}\hat{\theta}_k$    ▷ Prediction error
3: $w_k = 2/\left(\nu_s\sigma_s^2 + e_k^2\right)$    ▷ Approx. weight for Student-t M-estimator
4: $K_k = \left(w_k \cdot P_k^- \, \Phi_k\right)\left(r + w_k \cdot \Phi^\mathsf{T} \, P_k^- \, \Phi_k\right)^{-1}$    ▷ M - Kalman Gain
5: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_k e_k$    ▷ Parameter estimate update
6: $P_k^+ = \left(\mathbb{I} - K_k\Phi_k^\mathsf{T}\right)P_k^-\left(\mathbb{I} - K_k\Phi_k^\mathsf{T}\right)^\mathsf{T} + K_k\frac{r}{w_k}K_k^\mathsf{T}$    ▷ modified Joseph's form
7: $P_{k+1}^- = P_k^+ + Q_k$    ▷ A priori estimation error covariance
8: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Algorithm 14 M-SG-KF: M-Estimation Stenlund-Gustafsson Kalman Filter. Modified from [AE+16].

```
Require: $P_d$, $r$, $\nu_s$, $\sigma_s$
1: Inputs:
       $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$
2: $e_k = y_k^\theta - \Phi_k^\mathsf{T}\hat{\theta}_k$    ▷ Prediction error
3: $w_k = 2/\left(\nu_s\sigma_s^2 + e_k^2\right)$    ▷ Approx. weight for Student-t M-estimator
4: $K_k = \left(w_k \cdot P_k^- \, \Phi_k\right)\left(r + w_k \cdot \Phi^\mathsf{T} \, P_k^- \, \Phi_k\right)^{-1}$    ▷ Kalman Gain
5: $\hat{\theta}_{k+1}^- = \hat{\theta}_k + K_k e_k$    ▷ Parameter estimate correction
6: $Q_k = \left(w_k \cdot P_d\Phi_k\Phi_k^\mathsf{T}P_d\right)\left(r + w_k \cdot \Phi_k^\mathsf{T}P_d\Phi_k\right)^{-1}$    ▷ SG cov. adaptation [SG02]
7: $P_k^+ = \left(\mathbb{I} - K_k\Phi_k^\mathsf{T}\right)P_k^-\left(\mathbb{I} - K_k\Phi_k^\mathsf{T}\right)^\mathsf{T} + K_k\frac{r}{w_k}K_k^\mathsf{T}$    ▷ modified Joseph's form
8: $P_{k+1}^- = P_k^+ + Q_k$    ▷ A priori estimation error covariance
9: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Algorithm 15 M-SG-KF-IV: M-Estimation Stenlund-Gustafsson Kalman Filter with Intrumental Variables. Modified from [Rho16].

```
Require: $P_d$, $r$, $\nu_s$, $\sigma_s$
1: Inputs:
       $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$, $\Phi_k^\mathrm{IV}$
2: $e_k = y_k^\theta - \Phi_k^\mathsf{T}\hat{\theta}_k$    ▷ Prediction error
3: $w_k = 2/\left(\nu_s\sigma_s^2 + e_k^2\right)$    ▷ Approx. weight for Student-t M-estimator. As in [AE+16]
4: $K_k = \left(w_k \cdot P_k^- \, \Phi_k^\mathrm{IV}\right)\left(r + w_k \cdot \Phi^\mathsf{T} \, P_k^- \, \Phi_k^\mathrm{IV}\right)^{-1}$    ▷ Kalman Gain for IV
5: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_k e_k$    ▷ Parameter estimate update
6: $Q_k = \left(w_k \cdot P_d\Phi_k\Phi_k^\mathsf{T}P_d\right)\left(r + w_k \cdot \Phi_k^\mathsf{T}P_d\Phi_k\right)^{-1}$    ▷ SG cov. adaptation [SG02]
7: $P_k^+ = \left(\mathbb{I} - K_k\Phi_k^\mathsf{T}\right)P_k^-\left(\mathbb{I} - K_k\left(\Phi_k^\mathrm{IV}\right)^\mathsf{T}\right)^\mathsf{T} + K_k\frac{r}{w_k}K_k^\mathsf{T}$    ▷ modified Joseph's form for IV
8: $P_{k+1}^- = P_k^+ + Q_k$    ▷ A priori estimation error covariance
9: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Algorithm 16 M-SG-KF-C: M-Estimation Stenlund-Gustafsson Kalman Filter with Constraints. Modified from [AE+16].

```
Require: $P_d,\ r,\ \nu_s,\ \sigma_s,\ c = [\theta_{\max}, \theta_{\min}]^\mathsf{T},\ L = [\mathbb{I}, -\mathbb{I}]^\mathsf{T}$
 1: Inputs:
        $y_k^\theta,\ \hat{\theta}_k,\ P_k^-,\ \Phi_k$
 2: $e_k = y_k^\theta - \Phi_k^\mathsf{T} \hat{\theta}_k$                                    ▷ Prediction error
 3: $w_k = 2 / \left( \nu_s \sigma_s^2 + e_k^2 \right)$                     ▷ Approx. weight for Student-t M-estimator
 4: $K_k = \left( w_k \cdot P_k^- \ \Phi_k \right) \left( r + w_k \cdot \Phi^\mathsf{T} \ P_k^- \ \Phi_k \right)^{-1}$          ▷ Kalman Gain
 5: $\hat{\theta}_{k+1}^- = \hat{\theta}_k + K_k e_k$                         ▷ Parameter estimate correction
 6: $Q_k = \left( w_k \cdot P_d \Phi_k \Phi_k^\mathsf{T} P_d \right) \left( r + w_k \cdot \Phi_k^\mathsf{T} P_d \Phi_k \right)^{-1}$     ▷ SG cov. adaptation [SG02]
 7: $P_k^+ = \left( \mathbb{I} - K_k \Phi_k^\mathsf{T} \right) P_k^- \left( \mathbb{I} - K_k \Phi_k^\mathsf{T} \right)^\mathsf{T} + K_k \frac{r}{w_k} K_k^\mathsf{T}$    ▷ modified Joseph's form
 8: $q_k = c - L \hat{\theta}_k^+$                                        ▷ Offset for LCP
 9: if $q_k < 0$ then
10:     $M_k = L P_k^+ L^\mathsf{T}$                                    ▷ Matrix for LCP
11:     $\eta = \mathrm{LCPSolve}(M, q)$                                ▷ Solution of LCP
12:     $\hat{\theta}_{k+1} = \hat{\theta}_k^- - P_k^+ L^\mathsf{T} \eta$              ▷ Constrained parameter estimate
13: end if
14: $P_{k+1}^- = P_k^+ + Q_k$                          ▷ A priori estimation error covariance
15: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Algorithm 17 M-SG-KF-IV-C: M-Estimation Stenlund-Gustafsson Kalman Filter with Intrumental Variables and constraints. Modified and combined from [Rho16] and [AE+16].

```
Require: $P_d$, $r$, $\nu_s$, $\sigma_s$, $c = [\theta_{\max}, \theta_{\min}]^\mathsf{T}$, $L = [\mathbb{I}, -\mathbb{I}]^\mathsf{T}$
 1: Inputs:
        $y_k^\theta$, $\hat{\theta}_k$, $P_k^-$, $\Phi_k$, $\Phi_k^{\mathrm{IV}}$
 2: $e_k = y_k^\theta - \Phi_k^\mathsf{T} \hat{\theta}_k$    ▷ Prediction error
 3: $w_k = 2/\left(\nu_s \sigma_s^2 + e_k^2\right)$    ▷ Approx. weight for Student-t M-estimator. As in [AE+16]
 4: $K_k = \left(w_k \cdot P_k^- \, \Phi_k^{\mathrm{IV}}\right)\left(r + w_k \cdot \Phi^\mathsf{T} \, P_k^- \, \Phi_k^{\mathrm{IV}}\right)^{-1}$    ▷ Kalman Gain for IV
 5: $\hat{\theta}_{k+1} = \hat{\theta}_k + K_k e_k$    ▷ Parameter estimate update
 6: $Q_k = \left(w_k \cdot P_d \Phi_k \Phi_k^\mathsf{T} P_d\right)\left(r + w_k \cdot \Phi_k^\mathsf{T} P_d \Phi_k\right)^{-1}$    ▷ SG cov. adaptation [SG02]
 7: $P_k^+ = \left(\mathbb{I} - K_k \Phi_k^\mathsf{T}\right) P_k^- \left(\mathbb{I} - K_k \left(\Phi_k^{\mathrm{IV}}\right)^\mathsf{T}\right)^\mathsf{T} + K_k \frac{r}{w_k} K_k^\mathsf{T}$    ▷ modified Joseph's form for IV
 8: $q_k = c - L\hat{\theta}_k^+$    ▷ Offset for LCP
 9: if $q_k < 0$ then
10:     $M_k = L P_k^+ L^\mathsf{T}$    ▷ Matrix for LCP
11:     $\eta = \mathrm{LCPSolve}(M, q)$    ▷ Solution of LCP
12:     $\hat{\theta}_{k+1} = \hat{\theta}_k^- - P_k^+ L^\mathsf{T} \eta$    ▷ Constrained parameter estimate
13: end if
14: $P_{k+1}^- = P_k^+ + Q_k$    ▷ A priori estimation error covariance
15: return $\hat{\theta}_{k+1}, P_{k+1}^-$
```

Algorithm 18 Runge-Kutta of 4th order with M steps per sampling interval T_s [Run95].

```
 1: $\Delta t = T_s/M$
 2: $x^+ = x_k$
 3: for j = 1, M do
 4:     $k_1 = f(x^+, u_k, d_k, \theta_k, w_k)$
 5:     $k_2 = f(x^+ + \Delta t/2 k_1, u_k, d_k, \theta_k, w_k)$
 6:     $k_3 = f(x^+ + \Delta t/2 k_2, u_k, d_k, \theta_k, w_k)$
 7:     $k_4 = f(x^+ + \Delta t k_3, u_k, d_k, \theta_k, w_k)$
 8:     $x^+ = x^+ \Delta t/6\left(k_1 + 2k_2 + 2k_3 + k_4\right)$
 9: end for
10: $x_{k+1} = x^+$
```

List of Acronyms

5G	5th Generation Telecommunication Standard 20–22
A3C	Asynchronouos Advantage Actor-Critic 189
ACC	Adaptive Cruise Control 4, 11, 12, 18, 20, 50–53, 58, 193
ADAS	Advanced Driver Assistance System 1, 4, 5, 8, 10, 11, 20, 26, 49, 52, 54, 59, 193
ADP	Adaptive dynamic programming 223
ADS	Automated Driving System 10
AHS	Automated Highway System 17
AI	Artificial Intelligence 184
ANMPC	Adaptive Nonlinear Model Predictive Control 5, 9, 122, 135, 136, 141, 150, 158, 161–167, 172, 173, 216, 288
APRBS	Amplitude-modified Pseudo-Random Binary Sequence 199, 200, 203, 207, 209, 211, 212, 214
ARX	Auto Regressive with Exogenous Input 55
AV	Automated Vehicle 10, 54, 57, 59, 90, 92, 94, 150, 192
AVCS	Advanced Vehicle Control System 17
BVP	Boundary Value Problem 146
CACC	Cooperative Adaptive Cruise Control 4, 11, 12, 51, 52, 150, 193
CC	Cruise Control 4
CDKF	Central Difference Kalman Filter 250
CLF	Control Lyapunov Function 143
CMU	Carnegie Mellon University 18, 19
CNN	Convolutional Neural Network 191
DAE	Differential-algebraic equation 41–44, 148, 152, 155, 156, 159, 160, 226, 270
DARPA	Defense Advanced Research Projects Agency 18, 19
DDPG	Deep Deterministic Policy Gradient 189, 192, 193, 202, 203, 205, 207, 227
DeePKS	Derivative-exploiting Polynomial Kalman Smoother 6, 64, 76, 78–83, 85, 86, 93, 104, 110, 111, 117, 120, 121, 124, 287, 293
DeePLS	Derivative-exploiting Polynomial Least Squares 6, 64, 72–76, 78–83, 85, 86, 93, 104, 110, 111, 116, 117, 119–121, 123–127, 218, 239, 287, 288, 293

DNN	Deep Neural Network 202, 205
DOE	Design of Experiments 213
Double-DQN	Double Deep Q-Network 187
DPG	Deterministic Policy Gradient 189
DQL	Double Q-Learning 186, 187
DQN	Deep Q-Network 187, 189, 193
DRL	Deep Reinforcement Learning 6, 178, 213, 214
Dueling-DQN	Dueling Deep Q-Networks 187
E/E	Electric / Electronic 20, 21, 287
ECM	Engine Control Module 35, 40, 50, 55
EGR	Exhaust Gas Recirculation 35
EKF	Extended Kalman Filter 91, 93, 94, 101, 112, 113, 115, 117, 234, 244, 245, 250, 259, 260, 288
EMPC	Economic Model Predictive Control 52, 142
ESC	Electronic Stability Control 18
EV	Electric Vehicle 88
FIE	Full Information Estimation 99, 240
FIR	Finite Impulse Response 63, 194
FLOP	Floating-Point Operation 208
GAE	Generalized Advantage Estimation 189
GNNS	Global Navigation Satellite System 18, 91
HCU	Hybrid Control Unit 40
HD	High Definition 90, 94
HEV	Hybrid Electric Vehicle 4, 58, 88
ICE	Internal Combustion Engine 16, 25, 26, 33, 35, 40, 164, 222
IMU	Inertial Measurement Unit 90
IRLS	Iteratively Re-weighted Least Squares 265, 267
IV	Instrumental Variables 102, 106, 109–111, 262, 263, 269, 276, 277, 279, 281
KF	Kalman Filter 103, 275
KKT	Karush-Kuhn-Tucker 146
LCP	Linear Complementarity Problem 278, 280, 281
LiP	Linear-in-Parameters 6, 46, 96, 100, 101, 104, 116, 119, 120, 123, 124, 217
LPV	Linear Parameter-Varying 143
LQ	Linear Quadradic 55
LQR	Linear Quadradic Regulator 139, 149, 192
LTI	Linear Time-Invariant 143
LTV	Linear Time-Variant 224
MaaS	Mobility-as-a-Service 23
MDP	Markov Decision Process 177, 178, 182, 190, 191, 200

MHE	Moving Horizon Estimation 228, 241
ML	Machine Learning 62, 82
MLE	Maximum Likelihood Estimation 264
MPC	Model Predictive Control 6, 12, 49, 51, 52, 55, 56, 58, 59, 91, 135–144, 148, 149, 152, 153, 165–167, 175–177, 182, 192, 193, 200, 208, 211, 215, 219, 222, 224, 225, 227, 228
MRAC	Model Reference Adaptive Control 226–228
MSE	Mean Squared Error 235
NAF	Normalized Advantage Functions 187, 189
NFQ	Neural Fitted Q-iteration 188, 192
NHTSA	National Highway Traffic Safety Administration 88
NLP	Nonlinear Program 146–148, 168, 225, 226
NMPC	Nonlinear Model Predictive Control 52, 56, 59, 140–143, 150, 158, 161–163, 166, 167, 171, 176, 201, 207–211, 216, 219, 224, 225, 227, 288, 290, 291
NN	Neural Network 62
NRMSE	Normalized Root Mean Square Error 80
OCP	Optimal Control Problem 137–140, 142–145, 147, 148, 152, 154, 155, 159, 165, 223, 225
ODD	Operational Desing Domain 20
ODE	Ordinary Differential Equation 41, 42, 45, 47, 96, 98, 117, 146, 148, 151, 155, 226
PCC	Predictive Cruise Control 4, 11, 12, 53
PCM	Power-train Control Module 40, 90
PDE	Partial Differential Equation 145
PDF	Probability Density Function 179, 235, 256, 257, 263, 266, 267
PE	Persistently exciting 100, 102
PF	Particle Filter 112, 113
PI	Proportional-Integral 50, 53, 55, 158, 160–164, 169, 173, 193, 210, 211, 215, 288, 290
PID	Proportional-Integral-Derivative 50–52, 54–56, 59, 113, 140, 219
PILCO	Probabilistic Inference for Learning COntrol 187, 192, 201
PKS	Polynomial Kalman Smoother 6, 63, 64, 69, 76–83, 85, 86, 92, 104, 109, 110, 121, 123–125, 287
PoE	Persistence of Excitation 225
PPO	Proximal Policy Optimization 188
PRBS	Pseudo-Random Binary Sequence 55, 200
PRLC-A	Predictive Reinforcement Learning Controller with incorporated Advance Knowledge 196, 207, 208, 210
QP	Quadratic Program 147, 148

RACE	Robust and Reliant Automotive Computing Environment for Future eCars 20, 21, 287
RELU	Rectified Linear Unit 205
RGTLS	Recursive Generalized Total Least-Squares 102, 110
RHC	Receding Horizon Control 137
RL	Reinforcement Learning 6, 9, 49, 59, 175–177, 179, 180, 182–186, 191–194, 197, 200, 201, 204, 210–216, 219–221, 227
RLS	Recursive Least Squares 52, 91–94, 100, 102, 103, 241, 263
RMPC	Robust Model Predictive Control 53
RMSE	Root Mean Square Error 110, 111, 115, 124, 163, 164, 291
SAE	Society of Automotive Engineers 10, 12
SaG	Savitzky-Golay 62–65, 67, 69, 73, 75, 78–80, 82, 83, 85, 92, 109–111, 217
SG	Stenlund-Gustafsson 103, 106, 109, 111
SI	Spark Ignition 57
SQP	Sequential Quadratic Programming 146
SRCDIF	Square-root Central Difference Information Filter 117–119, 121, 123, 124, 126, 219, 239, 254
SRUIF	Square-root Unscented Information Filter 254
SVD	Singular Value Decomposition 239
TCU	Transmission Control Unit 40
TD	Temporal Difference 186
TNPG	Truncated Natural Policy Gradient 201
TRPO	Trust Region Policy Optimization 188, 189, 202
UKF	Unscented Kalman Filter 112, 113, 115–117, 122–125, 246–248, 250, 255, 288, 291
UNECE	United Nations Economic Commission for Europe 112, 121, 158
V2X	Vehicle to-Everything 21, 22
WLTC	Worldwide harmonized Light-duty vehicles Test Cycles 112, 121, 122, 158, 159, 162, 164, 288
WLTP	Worldwide Harmonised Light Vehicle Test Procedure 112, 113, 121, 158, 161, 164

List of Figures

List of Tables

List of Algorithms

List of Symbols

$\alpha_{\mathbf{w}}$	Side slip angle [rad]
$\delta_{\mathbf{w}}$	Toe-in angle [rad]
$\delta_{\mathbf{w}}$	Wheel angle relative to vehicle axis [rad]
$\eta_{\mathbf{pwt}}$	Power-train efficiency [-]
$\mu_{\mathbf{br}}$	Brake friction coefficient [-]
$\nu_{\mathbf{g}}$	Transmission efficiency [-]
$\omega_{\mathbf{e}}$	Engine speed [s^{-1}]
$\omega_{\mathbf{w}}$	Wheel speed [s^{-1}]
$\psi_{\mathbf{a}}$	Air approach angle [rad]
$\rho_{\mathbf{air}}$	air density [kg m^{-3}
$\tau_{\mathbf{br}}$	Time constant of brake torque response [s]
$\tau_{\mathbf{e}}$	Time constant of engine torque response [s]
φ	Road slope (Road plane elevation angle λ in ISO8855) [deg]
a	Vehicle acceleration [m s^{-2}]
$A_{\mathbf{br}}$	Effective brake pad area [m^2]
$A_{\mathbf{v}}$	Frontal vehicle area [m^2]
$C_{\mathbf{d}}$	Lumped aerodynamic drag coefficient, $C_\mathrm{d} = \frac{1}{2} \cdot \rho_\mathrm{air} \cdot c_\mathrm{x} \cdot A_\mathrm{v}$
$C_{\mathbf{r}}$	Rolling resistance coefficient [-]
$C_{\mathbf{s}}$	Cornering stiffness [-]
$C_{\mathbf{s}}$	Cornering stiffness [N/deg]
$c_{\mathbf{x}}$	Longitudinal aerodynamic drag coefficient [-]
d	Vehicle distance travelled $d(t) = \int_{t0}^{t} v$
F_aero	Aerodynamic drag force [Nm]
$F_{\mathbf{cr}}$	Cornering resistance [Nm]
$F_{\mathbf{c}}$	Centrifugal force [Nm]

$F_{\mathbf{pitch}}$	Gravity force [Nm]
$F_{\mathbf{r,f}}$	Rolling resistance tire force, front wheels [Nm]
$F_{\mathbf{r,r}}$	Rolling resistance tire force, rear wheels [Nm]
$F_{\mathbf{s}}$	Lateral tire force [Nm]
$F_{\mathbf{tire,f}}$	Longitudinal tire force, front wheels [Nm]
$F_{\mathbf{tire,r}}$	Longitudinal tire force, rear wheels [Nm]
$F_{\mathbf{xy}}$	Force acting on information given in subscript [Nm]
$F_{\mathbf{z}}$	Vertical tire force (lumped) [Nm]
g	Gravitational constant; $g = 9.81$ [N]
$I_{\mathbf{e}}$	Engine inertia [kg m^2]
$i_{\mathbf{g}}$	Transmission gear ratio [-]
$I_{\mathbf{w}}$	Inertia of all wheels [kg m^2]
m	Vehicle mass [kg]
$m_{\mathbf{I}}$	Mass equivalent resulting from wheel inertia [kg]
$p_{\mathbf{a}}$	Ambient pressure [bar]
$p_{\mathbf{br}}$	Brake pressure [Pa]
$r_{\mathbf{br}}$	Effective brake disc radius [m]
$r_{\mathbf{eff}}$	Effective wheel radius [m]
$R_{\mathbf{pwt}}$	Power-train ratio [-]
$r_{\mathbf{v}}$	Curve radius of the vehicle [m]
$r_{\mathbf{v}}$	Vehicle turning radius [m]
$R_{\mathbf{air}}$	Specific gas constant for idealized air, $_{\text{air}}$ =287.058 [Jkg^{-1}K^{-1}]
$T_{\mathbf{amb}}$	Ambient temperature [deg C]
$T_{\mathbf{br}}$	Brake torque [Nm]
$T_{\mathbf{drag}}$	Maximum negative engine drag torque [Nm]
$T_{\mathbf{e}}$	Engine net torque [Nm]
$T_{\mathbf{e}}^{d}$	Engine torque demand value [Nm]
$T_{\mathbf{g}}$	Gearbox torque [Nm]
$T_{\mathbf{we}}$	Wheel torque induced by engine [Nm]
$T_{\mathbf{we}}^{d}$	Wheel torque incuded by engine, demand value [Nm]
$T_{\mathbf{w}}$	Wheel torque [Nm]

$T_{\mathbf{w}}^{d}$	Wheel torque demand value [Nm]
v	Vehicle speed [m/s]
v^{d}	Desired vehicle speed demand value [m/s]
$v_{\mathbf{a}}$	Air approach velocity [m/s], see [Hak18]
$v_{\mathbf{wind}}$	Wind speed [m/s]

References

[AB20] A. Alessandri and G. Battistelli. “Moving horizon estimation: Open problems, theoretical progress, and new application perspectives”. In: *International Journal of Adaptive Control and Signal Processing* 34.6 (2020), pp. 703–705. ISSN: 10991115. DOI: 10.1002/acs.3127 (cit. on p. 228).

[AC+10] F. Alam, H. Chowdhury, H. Moria, and S. Watkins. “Effects of Vehicle Add-Ons on Aerodynamic Performance Effects of Vehicle Add-Ons on Aerodynamic Performance”. In: *The Proceeding of the 13th Asian Congress of Fluid Mechanics (ACFM2010)*. January. 2010 (cit. on p. 88).

[AD+09] V. Adetola, D. DeHaan, and M. Guay. “Adaptive model predictive control for constrained nonlinear systems”. In: *Systems and Control Letters* 58.5 (2009), pp. 320–326. ISSN: 01676911. DOI: 10.1016/j.sysconle.2008.12.002. URL: http://dx.doi.org/10.1016/j.sysconle.2008.12.002 (cit. on p. 141).

[AE+16] S. Altmannshofer, C. Endisch, J. Martin, M. Gerngross, R. Limbacher, M. Gernground, and R. Limbacher. “Robust estimation of vehicle longitudinal dynamics parameters”. In: *IEEE Intelligent Vehicles Symposium, Proceedings* (2016), pp. 566–571. DOI: 10.1109/IVS.2016.7535443 (cit. on pp. 46, 92–95, 103, 109, 116, 117, 121, 123, 124, 128, 158, 258, 259, 266, 277–281).

[AE16] S. Altmannshofer and C. Endisch. “Robust vehicle mass and driving resistance estimation”. In: *Proceedings of the American Control Conference* 2016-July.2 (2016), pp. 6869–6874. ISSN: 07431619. DOI: 10.1109/ACC.2016.7526754 (cit. on pp. 46, 96, 103, 108).

[AG+19] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. “CasADi: a software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11.1 (2019). ISSN: 18672957. DOI: 10.1007/s12532-018-0139-4 (cit. on pp. 148, 159, 165).

[AG10] V. Adetola and M. Guay. “Performance Improvement in Adaptive Control of Linearly Parameterized Nonlinear Systems”. In: *IEEE Transactions on Automatic Control* 55.9 (2010), pp. 2182–2186 (cit. on p. 94).

[AL+15] B. Alrifaee, Y. Liu, and D. Abel. “ECO-cruise control using economic model predictive control”. In: *2015 IEEE Conference on Control and Applications, CCA 2015 - Proceedings* (2015), pp. 1933–1938. DOI: 10.1109/CCA.2015.7320892 (cit. on p. 52).

[Ala11] A. Alam. “Fuel-efficient distributed control for heavy duty vehicle platooning”. PhD thesis. KTH Royal Institute of Technology, 2011, pp. 1–9. ISBN: 9789175010953. URL: http://www.diva-portal.org/smash/record.jsf?pid=diva2:447050 (cit. on pp. 22, 39, 88).

[Alm19] A. Almqvist. *A pivoting algorithm solving linear complementarity problems*. 2019. URL: https://de.mathworks.com/matlabcentral/fileexchange/41485-a-pivoting-algorithm-solving-linear-complementarity-problems (cit. on pp. 259, 277, 278).

[AM12] H. A. Asl and J. Mcphee. “Modeling Torque Converter Characteristics in Automatic Drivelines : Lock-up Clutch and Engine Braking Simulation”. In: *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. ASME, 2012, pp. 1–11 (cit. on p. 33).

[Ami16] A. Amiraslani. “Differentiation matrices in polynomial bases”. In: *Mathematical Sciences* 10.1-2 (2016), pp. 47–53. ISSN: 22517456. DOI: 10.1007/s40096-016-0177-x (cit. on p. 72).

[And14] J. Andrew Bagnell. “Reinforcement Learning in Robotics: A Survey”. In: *Springer Tracts in Advanced Robotics* 97 (2014), pp. 9–67. ISSN: 1610742X. DOI: 10.1007/978-3-319-03194-1_2. arXiv: 9605103 [cs] (cit. on p. 191).

[AR+15] M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, W. Huber, and N. Kaempchen. “Experience, Results and Lessons Learned After Over 2 Years of Automated Driving on Germany’ s Highways”. In: *IEEE Intelligent Transportation Systems Magazine* (2015) (cit. on p. 20).

[AV11] B. Asadi and A. Vahidi. “Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time”. In: *IEEE Transactions on Control Systems Technology* 19.3 (2011), pp. 707–714. ISSN: 10636536. DOI: 10.1109/TCST.2010.2047860 (cit. on pp. 52, 56).

[AZ+15] T. C. Au, S. Zhang, and P. Stone. “Autonomous intersection management for semiautonomous vehicles”. In: *Routledge Handbook of Transportation* (2015), pp. 88–104. DOI: 10.4324/9781315756684 (cit. on p. 23).

[BA+21] I. Batkovic, M. Ali, P. Falcone, and M. Zanon. “Model Predictive Control with Infeasible Reference Trajectories”. In: (2021), pp. 1–8. arXiv: 2109.04846. URL: http://arxiv.org/abs/2109.04846 (cit. on p. 224).

[Bar04] A. Bartono. “Fahrzeuglängsführung im Niedergeschwindigkeitsbereich”. PhD thesis. Aachen: TU Braunschweig, 2004. ISBN: 3-8322-2367-3 (cit. on pp. 51, 56, 60).

[BB+16a] F. A. Bayer, F. D. Brunner, M. Lazar, M. Wijnand, and F. Allgower. “A tube-based approach to nonlinear explicit MPC”. In: *2016 IEEE 55th Conference on Decision and Control, CDC 2016* Cdc (2016), pp. 4059–4064. DOI: 10.1109/CDC.2016.7798884 (cit. on p. 225).

[BB+16b] F. Biral, E. Bertolazzi, and P. Bosetti. “Notes on Numerical Methods for Solving Optimal Control Problems”. In: *IEEJ Journal of Industry Applications* (2016) (cit. on p. 226).

[BC+07] M. Buechel, W. Carnochan, F. M. Ewen, A. Gallacher, E. Martini, M. H. Wellers, M. Büchel, W. Camochan, F. M. Ewen, A. Gallacher, M. Buechel, W. Carnochan, F. M. Ewen, A. Gallacher, E. Martini, and M. H. Wellers. “Smart calibration - Successful Combination of Process, Know-How and Tools”. In: *MTZ worldwide* 68.2 (2007), pp. 22–24. ISSN: 2192-9114. DOI: 10.1007/BF03226808. URL: http://link.springer.com/10.1007/BF03226808 (cit. on p. 40).

[BC+12a] A. Broggi, P. Cerri, M. Felisa, M. C. Laghi, L. Mazzei, and P. P. Porta. “The VisLab Intercontinental Autonomous Challenge: an extensive test for a platoon of intelligent vehicles”. In: *International Journal of Vehicle Autonomous Systems* 10.3 (2012), pp. 147–164. ISSN: 1471-0226. DOI: 10.1504/IJVAS.2012.051250. URL: http://www.inderscience.com/link.php?id=51250 (cit. on p. 19).

[BC+12b] C. Buckl, A. Camek, G. Kainz, C. Simon, L. Mercep, H. Stähle, and A. Knoll. “The software car: Building ICT architectures for future electric vehicles”. In: *Proceedings of the IEEE International Electric Vehicle Conference (IEVC)*. 2012. ISBN: 9781467315623. DOI: 10.1109/IEVC.2012.6183198 (cit. on p. 21).

[BC+16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. *OpenAI Gym*. 2016. arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540 (cit. on p. 205).

[BD+16] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. “End to End Learning for Self-Driving Cars”. In: *arXiv preprint* (2016). ISSN: 1938-7228. arXiv: 1604.07316. URL: http://arxiv.org/abs/1604.07316 (cit. on pp. 23, 191).

[Bel54] R. E. Bellmann. “The theory of dynamic programming”. In: *Summer Meeting of the American Mathematical Society* (1954) (cit. on pp. 139, 181).

[Bel57] R. E. Bellmann. *Dynamic Programming*. Republishe. Princeton University Press, 1957. ISBN: 9780691146683 (cit. on pp. 145, 181, 182, 191).

[Ber19] D. P. Bertsekas. *Reinforcement learning and optimal control*. 1st editio. Athena Scientific, 2019, p. 373. ISBN: 9781886529397 (cit. on p. 176).

[Bey19] S. Beyer. “Robuste Parameterschätzung für Elektrofahrzeuge”. PhD thesis. Universität der Bundeswehr München, 2019 (cit. on pp. 93, 94).

[BF+09] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield. “Little Ben: The Ben Franklin Racing Team’s entry in the 2007 DARPA Urban Challenge”. In: *Springer Tracts in Advanced Robotics*. 2009. ISBN: 9783642039904 (cit. on pp. 54, 58).

[BF+15] M. Buechel, J. Frtunikj, K. Becker, S. Sommer, C. Buckl, M. Armbruster, C. Klein, A. Marek, A. Zirkler, and A. Knoll. “An automated electric vehicle prototype showing new trends in automotive architectures”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Las Palmas, Gran Canaria, Spain: IEEE, 2015. DOI: 10.1109/ITSC.2015.209. URL: https://ieeexplore.ieee.org/document/7313301 (cit. on pp. 7, 20, 21, 167).

[BG+20] A. L. Bruce, A. Goel, and D. S. Bernstein. “Convergence and consistency of recursive least squares with variable-rate forgetting”. In: *Automatica* 119.3 (2020). ISSN: 00051098. arXiv: 2003.02737 (cit. on p. 241).

[BG00] M. C. Best and T. J. Gordon. “Combined state and parameter estimation of vehicle handling dynamics”. In: *Proceedings of of the 5th International Symposium on Advanced Vehicle Control (AVEC)* (2000) (cit. on pp. 93, 95).

[BG08] S. Björnander and L. Grunske. “Adaptive Cruise Controllers – A Literature Review”. In: August (2008) (cit. on p. 50).

[BH+17] M. Buechel, G. Hinz, F. Ruehl, H. Schroth, C. Gyoeri, and A. Knoll. “Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. 2017. ISBN: 9781509048045. DOI: 10.1109/IVS.2017.7995917. URL: https://ieeexplore.ieee.org/document/7995917 (cit. on p. 7).

[BI+09] M. Buehler, K. Iagnemma, and S. Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer, 2009. ISBN: 9783642039904 (cit. on p. 19).

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Vol. 4. 4. 2006. ISBN: 978-1-4939-3843-8 (cit. on p. 236).

[BK+18] M. Bansal, A. Krizhevsky, and A. Ogale. “ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst”. In: (2018), pp. 1–20. DOI: arXiv: 1812.03079v1. arXiv: 1812.03079. URL: http://arxiv.org/abs/1812.03079 (cit. on p. 192).

[BK16a] M. Buechel and A. Knoll. “A parameter estimator for a model based adaptive control scheme for longitudinal control of automated vehicles”. In: *Proceedings of the 9th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)* 49.15 (2016), pp. 181–186. DOI: https://doi.org/10.1016/j.ifacol.2016.07.729. URL: https://www.sciencedirect.com/science/article/pii/S2405896316310059 (cit. on pp. 5–7, 9, 20, 26, 40, 56, 60, 87, 92, 95, 116, 121, 124, 127, 135, 148, 150, 151, 167, 222).

[BK16b] M. Buechel and A. Knoll. “An adaptive nonlinear model predictive controller for longitudinal motion of automated vehicles”. In: *Proceedings of the IEEE Conference on Control Applications (CCA)*. IEEE, 2016, pp. 103–108. ISBN: 978-1-5090-0755-4. DOI: 10.1109/CCA.2016.7587829. URL: https://ieeexplore.ieee.org/document/7587829 (cit. on pp. 5, 7, 9, 26, 56, 60, 135, 148, 158, 167, 204, 207, 209, 222).

[BK17] S. L. Brunton and J. N. Kutz. *Data Driven Science & Engineering*. 2017 (cit. on p. 231).

[BK18] M. Buechel and A. Knoll. “Deep reinforcement learning for predictive longitudinal control of automated vehicles”. In: *Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2391–2397. DOI: 10.1109/ITSC.2018.8569977. URL: https://ieeexplore.ieee.org/document/8569977 (cit. on pp. 6, 7, 9, 175, 191, 194, 195, 199, 200, 203–205, 208–211, 213, 214).

[Ble13] F. Bleimund. “2013-05-19_notes to polynominal shift matrix.pdf”. 2013 (cit. on p. 68).

[BM99] A. Bemporad and M. Morari. “Robust model predictive control: A survey”. In: *Robustness in identification and control*. London: Springer London, 1999, pp. 207–226. DOI: 10.1007/BFb0109870. URL: http://control.ethz.ch/%20http://link.springer.com/10.1007/BFb0109870 (cit. on p. 141).

[BN+13] B. Bischoff, D. Nguyen-Tuong, T. Koller, H. Markert, and A. Knoll. *Learning Throttle Valve Control Using Policy Search*. Berlin Heidelberg: Springer, 2013, pp. 49–64. ISBN: 978-3-642-40988-2. DOI: 10.1007/978-3-642-40988-2_4 (cit. on p. 192).

[Boc84] H. G. Bock. “A multiple shooting algorithm for direct solution of optimal control algorithms”. In: *IFAC Proceedings Volumes* 17.2 (1984), pp. 1603–1608. ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)61205-9. URL: http://dx.doi.org/10.1016/S1474-6670(17)61205-9 (cit. on p. 147).

[Bon21] M. Bonnet. *Lecture notes on numerical linear algebra*. Palaiseau, 2021. URL: https://hal.archives-ouvertes.fr/hal-03321502/document (cit. on p. 71).

[Box81] J. F. Box. “Gosset, Fisher, and the t Distribution”. In: *The American Statistician* 35.2 (May 1981), p. 61. ISSN: 00031305. DOI: 10.2307/2683142. URL: https://www.jstor.org/stable/2683142?seq=1%20https://www.jstor.org/stable/2683142?origin=crossref (cit. on p. 267).

[BR+01a] H. S. Bae, R. H. Ryu, and J. C. Gerdes. “Road grade and vehicle parameter estimation for longitudinal control using GPS”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2001, 166–171 ST –Road grade and vehicle parameter est (cit. on pp. 91, 95).

[BR+01b] V. M. Becerra, P. D. Roberts, and G. W. Griffiths. “Applying the extended Kalman filter to systems described by nonlinear differential-algebraic equations”. In: *Control Engineering Practice* 9.3 (2001), pp. 267–281. ISSN: 09670661. DOI: 10.1016/S0967-0661(00)00110-6 (cit. on p. 226).

[Bra59] L. de Branges. “The Stone-Weierstrass theorem”. In: *Proceedings of the American Mathematical Society* 10.5 (1959), pp. 822–824. ISSN: 0002-9939. DOI: 10.1090/S0002-9939-1959-0113131-7. URL: https://www.ams.org/proc/1959-010-05/S0002-9939-1959-0113131-7/ (cit. on pp. 62, 64).

[Bro91] F. Broqua. “Cooperative Driving: Basic Concepts and a First Assessment of "Intelligent Cruise Control" Strategies”. In: *DRIVE Conference (1991: Brussels, Belgium). Advanced telematics in road transport. Vol. II*. 1991 (cit. on p. 18).

[Bro99] A. Broggi. *Automatic vehicle guidance: the experience of the ARGO autonomous vehicle*. 1999. ISBN: 9810237200 (cit. on p. 18).

[BS+19] M. Buechel, M. Schellmann, H. Rosier, T. Kessler, and A. Knoll. “Fortuna: Presenting the 5G-connected automated vehicle prototype of the project PROVIDENTIA”. In: *Preprint*. 2019. DOI: 10.13140/RG.2.2.24402.91842 (cit. on pp. 7, 20, 22).

[BT07] F. Bu and H. S. Tan. “Pneumatic brake control for precision stopping of heavy-duty vehicles”. In: *IEEE Transactions on Control Systems Technology* 15.1 (2007), pp. 53–64. ISSN: 10636536. DOI: 10.1109/TCST.2006.883238 (cit. on p. 57).

[BT96] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996, p. 491. ISBN: 978-1886529106 (cit. on p. 184).

[Bue22] M. Buechel. “Physics-informed kernels for convolutional smoothing of multi-channel time-series data”. In: *Preprint*. 2022, pp. 1–13. DOI: DOI:10.13140/RG.2.2.31636.17289 (cit. on pp. 6, 7, 61, 127).

[But30] S. Butterworth. “On the Theory of Filter Amplifiers”. In: *Wireless Engineer* 7.6 (1930), pp. 536–541 (cit. on p. 109).

[CA+11] F. Cabello, A. Acuña, P. Vallejos, M. E. Orchard, and J. R. del Solar. “Design and Validation of a Fuzzy Longitudinal Controller Based on a Vehicle Dynamic Simulator”. In: *Proceedings of the 9th IEEE International Conference on Control and Automation (ICCA)*. Santiago de Chile, 2011. ISBN: 9788578110796. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3 (cit. on p. 51).

[Car70] K. H. F. Cardew. “The Automatic Steering of Vehicles: An Experimental System Fitted to a DS 19 Citroen Car”. In: *RRL report ; LR 340* (1970). URL: https://trid.trb.org/view/641347 (cit. on p. 18).

[CD+09] S. Choi, B. D’Andrea-Novel, M. Fliess, H. Mounier, and J. Villagra. “Model-free control of automotive engine and brake for Stop-and-Go scenarios”. In: *2009 European Control Conference (ECC)*. 2009, pp. 3622–3627. ISBN: 9783952417393. DOI: 10.23919/ecc.2009.7074962 (cit. on p. 56).

[CD+17] A. Chakrabarty, V. Dinh, M. J. Corless, A. E. Rundell, S. H. Zak, and G. T. Buzzard. “Support Vector Machine Informed Explicit Nonlinear Model Predictive Control Using Low-Discrepancy Sequences”. In: *IEEE Transactions on Automatic Control* 62.1 (Jan. 2017), pp. 135–148. ISSN: 0018-9286. DOI: 10.1109/TAC.2016.2539222. URL: http://ieeexplore.ieee.org/document/7428832/ (cit. on p. 225).

[CD80] C. Cobelli and J. J. DiStefano. “Parameter and structural identifiability concepts and ambiguities: a critical review and analysis.” In: *The American journal of physiology* 239.1 (1980), pp. 7–24. ISSN: 00029513. DOI: 10.1152/ajpregu.1980.239.1.R7 (cit. on p. 270).

[CD95] S.-B. Choi and P. Devlin. “Throttle and Brake Combined Control for Intelligent Vehicle Highway Systems”. In: 1995. DOI: 10.4271/951897. URL: https://www.sae.org/content/951897/ (cit. on pp. 50, 54).

[CG05] C. Carlson and J. Gerdes. “Consistent nonlinear estimation of longitudinal tire stiffness and effective radius”. English. In: *IEEE Transactions on Control Systems Technology* 13.6 (2005), pp. 1010–1020. ISSN: 1063-6536 (cit. on pp. 37, 152).

[CH16] D. A. Copp and J. P. Hespanha. *Addressing Adaptation and Learning in the Context of Model Predictive Control With Moving-Horizon Estimation*. Elsevier Inc., 2016, pp. 187–209. ISBN: 9780128054376. DOI: 10.1016/B978-0-12-805246-4.00006-9. URL: http://dx.doi.org/10.1016/B978-0-12-805246-4.00006-9 (cit. on pp. 225, 229).

[CH17a] H. Chehardoli and M. R. Homaeinezhad. “Stable control of a heterogeneous platoon of vehicles with switched interaction topology, time-varying communication delay and lag of actuator”. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 231.22 (2017), pp. 4197–4208. ISSN: 20412983. DOI: 10.1177/0954406217709491 (cit. on p. 58).

[CH17b] D. A. Copp and J. P. Hespanha. “Simultaneous nonlinear model predictive control and state estimation”. In: *Automatica* 77.August 2018 (2017), pp. 143–154. ISSN: 00051098. DOI: 10.1016/j.automatica.2016.11.041 (cit. on pp. 225, 229).

[Cha00] A. C. D. Chaklader. “Vehicle weight and cargo load determination using tire pressure, US Patent, US09567472”. In: (2000). URL: https://patents.google.com/patent/US6449582B1/en (cit. on pp. 94, 95).

[Cho10] G. Chowdhary. “Concurrent learning for convergence in adaptive control without persistency of excitation”. PhD thesis. Georgia Institute of Technology, 2010. ISBN: 9781424477456. DOI: 10.1109/CDC.2010.5717148 (cit. on p. 188).

[CJ10a] G. Chowdhary and E. Johnson. “Concurrent learning for convergence in adaptive control without persistency of excitation”. In: *Proceedings of the IEEE Conference on Decision and Control* December (2010), pp. 3674–3679. ISSN: 01912216. DOI: 10.1109/CDC.2010.5717148 (cit. on p. 227).

[CJ10b] G. Chowdhary and E. N. Johnson. “Concurrent Learning for Convergence in Adaptive Control without Persistency of Excitation”. In: *49th IEEE Conference on Decision and Control*. 4. 2010, pp. 3674–3679 (cit. on pp. 62, 225).

[CJ11] G. Chowdhary and E. Johnson. “A singular value maximizing data recording algorithm for concurrent learning”. In: *Proceedings of the American Control Conference* (2011), pp. 3547–3552. ISSN: 07431619. DOI: 10.1109/acc.2011.5991481 (cit. on pp. 188, 227).

[CL+06] D. Corona, M. Lazar, B. De Schutter, and M. Heemels. “A Hybrid MPC Approach to the Design of a Smart Adaptive Cruise Controller”. In: *Proceedings of the IEEE International Conference on Control Applications (CCA)* (2006), pp. 231–236. ISSN: 01912216. DOI: 10.1109/CACSD-CCA-ISIC.2006.4776651 (cit. on p. 51).

[CL+12] S. Chakraborty, M. Lukasiewycz, C. Buckl, S. Fahmy, N. Chang, S. Park, Y. Kim, P. Leteinturier, and H. Adlkoferl. “Embedded Systems and Software Challenges in Electric Vehicles”. In: *In Design, Automation and Test in Europe (DATE)*. 2012. DOI: 10.1109/DATE.2012.6176508 (cit. on p. 21).

[CL+13] R. ’. Carlson, H. Lohse-Busch, J. Diez, and J. Gibbs. “The Measured Impact of Vehicle Mass on Road Load Forces and Energy Consumption for a BEV, HEV, and ICE Vehicle”. In: *SAE International Journal of Alternative Powertrains* 2.1 (2013), pp. 2013–01–1457. ISSN: 2167-4205. DOI: 10.4271/2013-01-1457. URL: http://papers.sae.org/2013-01-1457/ (cit. on p. 88).

[Clu04] O. I. Club. *1958 Chrysler Auto-Pilot Brochure*. 2004. URL: http://www.imperialclub.com/Yr/1958/58AutoPilot/index.htm (cit. on p. 16).

[CM+13] G. Chowdhary, M. Mühlegg, J. P. How, and F. Holzapfel. “Concurrent Learning Adaptive Model Predictive Control”. In: *Advances in Aerospace Guidance, Navigation and Control* (2013), pp. 29–47. DOI: 10.1007/978-3-642-38253-6_3 (cit. on p. 227).

[CM17] L. Chi and Y. Mu. “Deep Steering: Learning End-to-End Driving Model from Spatial and Temporal Visual Cues”. In: (2017), pp. 1–12. arXiv: 1708.03798. URL: http://arxiv.org/abs/1708.03798 (cit. on p. 192).

[CNN03] CNN. *CNN.com - Toyota unveils car that parks itself - Sep. 1, 2003*. 2003. URL: http://edition.cnn.com/2003/TECH/ptech/09/01/toyota.prius.reut/index.html (visited on 08/30/2018) (cit. on p. 18).

[Con18] Continental. *Milestones: The AD Timeline | Blog | 2025AD - The Automated Driving Community*. 2018. URL: https://www.2025ad.com/latest/milestones-the-ad-timeline/ (visited on 08/29/2018) (cit. on p. 17).

[CW+12] G. Chowdhary, T. Wu, M. Cutlerz, N. K. Üre, and J. P. How. “Experimental results of concurrent learning adaptive controllers”. In: *AIAA Guidance, Navigation, and Control Conference 2012* (2012), pp. 1–14. DOI: 10.2514/6.2012-4551 (cit. on p. 227).

[CW+18] N. Chen, M. Wang, T. Alkim, and B. van Arem. “A Robust Longitudinal Control Strategy of Platoons under Model Uncertainties and Time Delays”. In: *Journal of Advanced Transportation* 2018 (2018), pp. 1–13. ISSN: 0197-6729. DOI: 10.1155/2018/9852721 (cit. on p. 58).

[CY+13] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson. “Concurrent learning adaptive control of linear systems with exponentially convergent bounds”. In: *International Journal of Adaptive Control and Signal Processing* 27.4 (2013), pp. 280–301. ISSN: 08906327. DOI: 10.1002/acs.2297. URL: http://doi.wiley.com/10.1002/acs.2297 (cit. on p. 227).

[DAF21] DAF. *DAF – Predictive Cruise Control*. 2021. URL: https://www.daf.at/-/media/files/document-library/infosheets/comfort-and-safety/pcc/69793-daf-pcc-de.pdf (cit. on p. 11).

[Dan12] Dang. *Handbook of Intelligent Vehicles*. 2012, pp. 1459–1477. ISBN: 978-0-85729-084-7. DOI: 10.1007/978-0-85729-085-4. URL: http://link.springer.com/10.1007/978-0-85729-085-4 (cit. on pp. 49, 50).

[DAR04] DARPA. *The Grand Challenge for Autonomous Vehicles*. 2004. URL: https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles (visited on 08/30/2018) (cit. on p. 18).

[Dav13] L. C. Davis. “The effects of mechanical response on the dynamics and string stability of a platoon of adaptive cruise control vehicles”. In: *Physica A: Statistical Mechanics and its Applications* 392.17 (2013), pp. 3798–3805 (cit. on p. 35).

[DB+06] M. Diehl, H. G. Bock, H. Diedam, P.-b. Wieber, M. Diehl, H. G. Bock, H. Diedam, P.-b. W. Fast, and D. Multiple. “Fast Direct Multiple Shooting Algorithms for Optimal Robot Control”. In: *Fast motions in biomechanics and robotics*. Berlin, Heidelberg: Springer, 2006, pp. 65–93 (cit. on pp. 145, 146, 226).

[DC+16] Y. Duan, X. Chen, J. Schulman, and P. Abbeel. “Benchmarking Deep Reinforcement Learning for Continuous Control”. In: *arXiv* (2016). arXiv: 1604.06778v2 (cit. on p. 201).

[DC11] C. Desjardins and B. Chaib-draa. “Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 1248–1260. ISSN: 1524-9050. DOI: Doi10.1109/Tits.2011.2157145 (cit. on p. 193).

[DH+17] P. Dhariwal, C. Hesse, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. *OpenAI Baselines*. https://github.com/openai/baselines. 2017 (cit. on p. 205).

[Dic02] E. Dickmanns. “The development of machine vision for road vehicles in the last decade”. In: *Proceedings of the IEEE Intelligent Vehicle Symposium (IV)*. Vol. 1. Versailles: IEEE, 2002, pp. 268–281. ISBN: 0-7803-7346-4. DOI: 10.1109/IVS.2002.1187962. URL: http://ieeexplore.ieee.org/document/1187962/ (cit. on p. 18).

[Die01] M. Diehl. “Real-time optimization for large scale nonlinear processes”. PhD thesis. University Heidelberg, 2001. DOI: https://doi.org/10.11588/heidok.00001659 (cit. on p. 226).

[Die16] M. Diehl. *Lecture Notes on Numerical Optimization*. 2016. URL: https://cdn.syscop.de/publications/Diehl2016.pdf (cit. on pp. 145–147, 226).

[DJ+09] M. Djordjevic, A. Jankovic, and B. Jeremic. “Rolling resistance as the risk factor for fuel consumption”. In: *International Journal of Vehicle Systems Modelling and Testing* 4.3 (2009), pp. 185–200 (cit. on p. 39).

[DL+05] X. Dai, C.-k. K. Li, S. Member, A. B. Rad, S. Member, A. B. Rad, and S. Member. "An Approach to Tune Fuzzy Controllers Based on Reinforcement Learning for Autonomous Vehicle Control". In: *IEEE Transactions on Intelligent Transportation Systems* 6.3 (2005), pp. 285–293. ISSN: 15249050. DOI: 10.1109/TITS.2005.853698 (cit. on p. 193).

[DL+77] A. P. Dempster, N. M. Laird, and D. B. Rubin. " Maximum Likelihood from Incomplete Data Via the EM Algorithm ". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. DOI: 10.1111/j.2517-6161.1977.tb01600.x (cit. on p. 231).

[DO+02] S. Drakunov, U. Ozguner, P. Dix, and B. Ashrafi. "ABS control using optimum search via sliding modes". In: *IEEE Transactions on Control Systems Technology* 3.1 (2002), pp. 79–85. ISSN: 10636536. DOI: 10.1109/87.370698 (cit. on p. 36).

[DR11] M. P. Deisenroth and C. E. Rasmussen. "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: *Icml* (2011), pp. 465–472. URL: http://eprints.pascal-network.org/archive/00008310/ (cit. on p. 187).

[DR17] S. S. Dughman and J. A. Rossiter. "The Feasibility of Parametric Approaches to Predictive Control when Using Far Future Feed Forward Information". In: *Proceedings of the IEEE International Conference on Control and Automation (ICCA)* 1 (2017), pp. 1101–1106. ISSN: 19483457. DOI: 10.1109/ICCA.2017.8003215 (cit. on pp. 140, 176).

[DS03] M. Diehl and J. P. Schlöder. "Nominal Stability of the Real-Time Iteration Scheme for Nonlinear Model Predictive Control". In: (2003) (cit. on pp. 147, 225, 226).

[DS10] M. Di Natale and A. L. Sangiovanni-Vincentelli. "Moving from Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools". In: *Proceedings of the IEEE* 98.4 (2010), pp. 603–620. ISSN: 0018-9219. DOI: 10.1109/JPROC.2009.2039550 (cit. on p. 167).

[DZ11] M. Deflorian and S. Zaglauer. "Design of Experiments for Nonlinear Dynamic System Identification". In: *18th IFAC World Congress* (2011). DOI: 10.4108/icst.simutools.2012.247734. URL: http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac11-proceedings/data/html/papers/1502.pdf (cit. on pp. 199, 200).

[EB+22] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani. "A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving". In: *Journal of King Saud University - Computer and Information Sciences* 34.9 (2022), pp. 7366–7390. ISSN: 22131248. DOI: 10.1016/j.jksuci.2022.03.013. URL: https://doi.org/10.1016/j.jksuci.2022.03.013 (cit. on pp. 191, 193).

[EB14] J. M. Engel and R. Babuska. "On-line Reinforcement Learning for Nonlinear Motion Control: Quadratic and Non-quadratic Reward Functions". In: *IFAC Proceedings Volumes (IFAC-PapersOnline)* 19 (2014), pp. 7043–7048. ISSN: 14746670. DOI: 10.3182/20140824-6-ZA-1003.02042 (cit. on p. 197).

[EC+16] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff. “The Grand Cooperative Driving Challenge 2016: boosting the introduction of cooperative automated vehicles”. In: *IEEE Wireless Communications* 23.4 (2016), pp. 146–152. ISSN: 1536-1284. DOI: 10.1109/MWC.2016.7553038. URL: http://ieeexplore.ieee.org/document/7553038/ (cit. on p. 19).

[ED+14] M. Ellis, H. Durand, and P. D. Christofides. “A tutorial review of economic model predictive control methods”. In: *Journal of Process Control* 24.8 (2014), pp. 1156–1178. ISSN: 09591524. DOI: 10.1016/j.jprocont.2014.03.010 (cit. on p. 142).

[EG+05] D. Ernst, P. Geurts, and L. Wehenkel. *Tree-Based Batch Mode Reinforcement Learning Louis Wehenkel*. Tech. rep. 2005, pp. 503–556 (cit. on p. 188).

[EG+09] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel. “Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem”. In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 39.2 (2009), pp. 517–529. ISSN: 1083-4419. DOI: 10.1109/TSMCB.2008.2007630 (cit. on p. 175).

[Elb15] M. Elbanhawi. “In the Passenger Seat : Investigating Ride Comfort Measures in Autonomous Cars”. In: *IEEE Intelligent Transportation Systems Magazine* 3 (2015), pp. 4–17 (cit. on p. 24).

[EM05] M. Evestedt and A. Medvedev. *Stationary Behavior of an Anti-Windup Scheme for Recursive Parameter Estimation Under Lack of Excitation*. Vol. 38. 1. IFAC, 2005, pp. 107–112. DOI: 10.3182/20050703-6-CZ-1902.00018 (cit. on pp. 103, 262).

[ES15] J. Eriksson and L. Svensson. “Tuning for ride quality in autonomous vehicle”. PhD thesis. 2015, p. 61 (cit. on pp. 24, 25).

[Esk12] A. Eskandarian. “Handbook of intelligent vehicles”. In: *Handbook of Intelligent Vehicles* 1-2 (2012), pp. 1–1599. DOI: 10.1007/978-0-85729-085-4 (cit. on p. 51).

[FA+20] N. P. Farazi, T. Ahamed, L. Barua, and B. Zou. “Deep Reinforcement Learning and Transportation Research: A Comprehensive Review”. In: (2020), pp. 1–60. arXiv: 2010.06187. URL: http://arxiv.org/abs/2010.06187 (cit. on p. 191).

[FG+18] T. Faulwasser, L. Grüne, and M. A. Müller. “Economic nonlinear model predictive control”. In: *Foundations and Trends in Systems and Control* 5.1 (2018), pp. 1–98. ISSN: 23256826. DOI: 10.1561/2600000014 (cit. on pp. 142, 143).

[FI+03] R. Findeisen, L. Imsland, F. Allgöwer, and B. A. Foss. “Output feedback stabilization of constrained systems with nonlinear predictive control”. In: *International Journal of Robust and Nonlinear Control* 13.3-4 (2003), pp. 211–227. ISSN: 10498923. DOI: 10.1002/rnc.814. URL: https://onlinelibrary.wiley.com/doi/10.1002/rnc.814 (cit. on p. 141).

[FJ13] M. Fliess and C. Join. “Model-free control”. In: *International Journal of Control* 86.12 (2013), pp. 2228–2252. ISSN: 00207179. DOI: 10.1080/00207179.2013.810345. arXiv: arXiv:1305.7085v2 (cit. on p. 55).

[FK+08] H. K. Fathy, D. Kang, and J. L. Stein. “Online vehicle mass estimation using recursive least squares and supervisory data extraction”. In: *Proceedings of the American Control Conference* (2008), pp. 1842–1848. ISSN: 07431619. DOI: 10.1109/ACC.2008.4586760 (cit. on pp. 89, 91, 92, 95).

[Fle18] Fleetnews. *Legislation puts brakes on Audi's Level 3 autonomous technology | Manufacturer News*. 2018. URL: https://www.fleetnews.co.uk/news/manufacturer-news/2018/02/15/legislation-puts-brakes-on-audi-s-level-3-autonomous-technology (visited on 08/30/2018) (cit. on p. 20).

[FM67] G. E. Forsythe and C. B. Moler. *Computer solution of linear algebraic systems*. Prentice-Hall, 1967 (cit. on p. 72).

[FM91] R. E. Fenton and R. J. Mayan. "Automated Highway Studies at The Ohio State University - An Overview". In: *IEEE Transactions on Vehicular Technology*. Vol. 40. 1991 (cit. on p. 17).

[FP04] A. Ferrara and P. Pisu. "Minimum Sensor Second-Order Sliding Mode Longitudinal Control of Passenger Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 5.1 (2004), pp. 20–32. ISSN: 15249050. DOI: 10.1109/TITS.2004.825080 (cit. on p. 51).

[Fre87] R. Fremd. *Apparatus for measuring the mass of a motor vehicle, US Patent No. 4656876A*. 1987 (cit. on pp. 94, 95).

[Fro96] G. Frost. "Dynamic vehicle roll control using reinforcement learning". In: *UKACC International Conference on Control. Control '96*. Vol. 1996. IEE, 1996, pp. 1107–1112. ISBN: 0 85296 666 0. DOI: 10.1049/cp:19960708. URL: https://digital-library.theiet.org/content/conferences/10.1049/cp%7B%5C_%7D19960708 (cit. on p. 192).

[FS17] H. Fechtner and B. Schmulling. "Survey of current vehicle mass estimators in the context of future mobility". In: *EEE International Conference on Intelligent Transportation Engineering, (ICITE)* (2017), pp. 78–82. DOI: 10.1109/ICITE.2017.8056885 (cit. on pp. 89, 91).

[FT+12] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Muller-Bessler, and B. Huhnke. "Up to the limits: Autonomous Audi TTS". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)* (2012), pp. 541–547. DOI: 10.1109/IVS.2012.6232212. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6232212 (cit. on pp. 19, 57).

[Gar60] K. Gardels. "Automatic Car Controls for Electronic Highways". In: *General Motors Research Laboratories, Warren, MI* (1960). URL: https://trid.trb.org/view/488645 (cit. on p. 17).

[Gau20] J. E. Gaudio. "Fast Learning and Adaptation in Control and Machine Learning". PhD thesis. Massachusetts Institute of Technology, 2020 (cit. on p. 272).

[GB+11] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf (cit. on p. 205).

[GB10] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010* 9 (2010), pp. 249–256 (cit. on p. 205).

[Geh00] O. Gehrling. "Automatische Längs- und Querführung einer Lastkraftwagenkolonne". PhD thesis. Universität Stuttgart, 2000, p. 233 (cit. on p. 54).

[Ger06] M. Gerdin. “Local Identifiability and Observability of Nonlinear Differential-Algebraic Equations”. In: *IFAC Proceedings Volumes* 39.1 (2006), pp. 802–807. ISSN: 14746670. DOI: 10.3182/20060329-3-au-2901.00126 (cit. on p. 270).

[GF+18] A. Gupta, P. Falcone, and E. Engineering. “Low-Complexity Explicit MPC Controller for Vehicle Lateral Motion Control”. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. 2018, pp. 2839–2844. ISBN: 9781728103228 (cit. on p. 225).

[GG+19] G. Gunter, D. Gloudemans, R. E. Stern, S. McQuade, R. Bhadani, M. Bunting, M. L. D. Monache, R. Lysecky, B. Seibold, J. Sprinkle, B. Piccoli, and D. B. Work. “Are commercially implemented adaptive cruise control systems string stable?” In: (2019). arXiv: 1905.02108. URL: http://arxiv.org/abs/1905.02108 (cit. on pp. 11, 58).

[GK+15] A. Ghasemi, R. Kazemi, and S. Azadi. “Exact stability of a platoon of vehicles by considering time delay and lag”. In: *Journal of Mechanical Science and Technology* 29.2 (2015), pp. 799–805. ISSN: 1738494X. DOI: 10.1007/s12206-015-0142-x (cit. on p. 58).

[GK+18] J. Guanetti, Y. Kim, and F. Borrelli. “Control of connected and automated vehicles: State of the art and future challenges”. In: *Annual Reviews in Control* 45.March (2018), pp. 18–40. ISSN: 13675788. DOI: 10.1016/j.arcontrol.2018.04.011. URL: https://doi.org/10.1016/j.arcontrol.2018.04.011 (cit. on pp. 12, 49, 59).

[GL+12] A. Geiger, M. Lauer, F. Moosmann, B. Ranft, H. Rapp, C. Stiller, and J. Ziegler. “Team AnnieWAY’s Entry to the 2011 Grand Cooperative Driving Challenge”. In: *IEEE Transactions on Intelligent Transportation Systems* September (2012), pp. 1008–1017 (cit. on p. 54).

[GL+16] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. “Continuous Deep Q-Learning with Model-based Acceleration”. In: (2016). arXiv: 1603.00748. URL: http://arxiv.org/abs/1603.00748 (cit. on pp. 187, 189).

[GM+05] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel. “Model predictive control: For want of a local control Lyapunov function, all is not lost”. In: *IEEE Transactions on Automatic Control* 50.5 (2005), pp. 546–558. ISSN: 00189286. DOI: 10.1109/TAC.2005.847055 (cit. on pp. 143, 270).

[God99] K. Godfrey. “Identification of parametric models from experimental data [Book Review]”. In: *IEEE Transactions on Automatic Control* 44.12 (Dec. 1999), pp. 2321–2322. ISSN: 0018-9286. DOI: 10.1109/TAC.1999.811220. URL: http://ieeexplore.ieee.org/document/811220/ (cit. on p. 270).

[Gon11] J. Gondzio. “Interior Point Methods 25 Years Later”. In: *European Journal of Operational Research* (2011), pp. 1–33 (cit. on p. 147).

[Gör17] D. Görges. “Relations between Model Predictive Control and Reinforcement Learning”. In: *IFAC-PapersOnLine* 50.1 (2017), pp. 4920–4928. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.747. URL: https://doi.org/10.1016/j.ifacol.2017.08.747 (cit. on p. 219).

[GP11] L. Grüne and J. Pannek. “Nonlinear Model Predictive Control”. In: (2011), pp. 43–67. DOI: 10.1007/978-0-85729-501-9. URL: http://link.springer.com/10.1007/978-0-85729-501-9 (cit. on p. 140).

[GP17] L. Gruene and J. Pannek. *Nonlinear model predictive control - Theory and Algorithms*. Second edi. Springer, 2017. ISBN: 978-3-319-46023-9. DOI: DOI10.1007/978-3-319-46024-6 (cit. on pp. 115, 139, 140, 142, 143).

[Gre84] P. Green. “Iteratively Reweighted Least Squares for Maximum Likelihood Estimation, and some Robust and Resistant Alternatives”. In: *Journal of Royal Statistical Society* 2.46 (1984), pp. 149–192. URL: https://www.jstor.org/stable/2345503 (cit. on p. 265).

[Gru09] L. Gruene. “Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems”. In: *SIAM Journal on Control and Optimization* 48.2 (2009), pp. 1206–1228. DOI: 10.1137/070707853. URL: http://www.siam.org/journals/ojsa.php (cit. on p. 143).

[GW+97] S. Germann, M. Würtenberger, and R. Isermann. “Modellgestützte Verfahren zur parameteradaptiven Regelung der Fahrzeuglängsdynamik”. In: *At-Automatisierungstechnik* 45.2 (1997), pp. 84–92. ISSN: 01782312. DOI: 10.1524/auto.1997.45.2.84 (cit. on p. 36).

[GZ+20] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl. “From linear to nonlinear MPC: bridging the gap via the real-time iteration”. In: *International Journal of Control* 93.1 (2020), pp. 62–80. ISSN: 13665820. DOI: 10.1080/00207179.2016.1222553 (cit. on p. 226).

[GZ20] S. Gros and M. Zanon. “Data-driven economic NMPC using reinforcement learning”. In: *IEEE Transactions on Automatic Control* 65.2 (2020), pp. 636–648. ISSN: 15582523. DOI: 10.1109/TAC.2019.2913768. arXiv: 1904.04152 (cit. on p. 227).

[Hak18] K.-L. Haken. *Grundlagen der Kraftfahrzeugtechnik*. München: Carl Hanser Verlag GmbH & Co. KG, 2018. ISBN: 978-3446454125. DOI: 10.3139/9783446455702.fm (cit. on pp. 28, 29, 297).

[Har21] O. Harkegard. *QCAT*. 2021. URL: https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/4609/versions/1/previews/QCAT/doc/intro.html (cit. on p. 13).

[Has10] H. V. Hasselt. “Double Q-learning”. In: *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Curran Associates, Inc., 2010, pp. 2613–2621. URL: http://papers.nips.cc/paper/3964-double-q-learning.pdf (cit. on pp. 186, 187).

[Hay01] S. Haykin. *Kalman filtering and neural networks*. Ed. by S. Haykin. John Wiley & Sons, 2001. ISBN: ISBN 0-471-22154-6 (cit. on pp. 259, 260).

[HB+05] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. “{SUNDIALS}: Suite of nonlinear and differential/algebraic equation solvers”. In: *ACM Transactions on Mathematical Software (TOMS)* 31.3 (2005), pp. 363–396 (cit. on pp. 148, 159, 160).

[HB+17] G. Hinz, M. Buechel, F. Diehl, G. Chen, and A. Kraemmer. “Designing a far-reaching view for highway traffic scenarios with 5G-based intelligent infrastructure”. In: *8. Tagung Fahrerassistenzsysteme TUEV - Sued*. 2017. URL: https://mediatum.ub.tum.de/doc/1421303/1421303.pdf (cit. on pp. 7, 22).

[HE+17] G. Hinz, J. Eichinger, M. Buechel, and A. Knoll. “Proactive video-based use of telecommunications technologies in innovative motorway scenarios”. In: *5th Fachgespraech Inter-Vehicle Communication*. 2017. URL: http://fg-ivc.car2x.org/proceedings-fg-ivc-2017.pdf (cit. on pp. 7, 22).

[Hel10] E. Hellström. “Look-ahead Control of Heavy Vehicles”. PhD thesis. Linkoeping University, 2010 (cit. on p. 53).

[Hes72] R. Hesse. “Cabinentaxi: A Personal Public Transport System”. In: *Transportation* 1.3 (1972), pp. 321–328. URL: https://doi.org/10.1007/BF00150319 (cit. on p. 18).

[HF+11] B. Houska, H. J. Ferreau, and M. Diehl. “ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization”. In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312 (cit. on p. 148).

[HI+18] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. “Deep Reinforcement Learning that Matters”. In: *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. New Orleans, Louisiana, USA, 2018. arXiv: 1709.06560. URL: http://arxiv.org/abs/1709.06560 (cit. on p. 185).

[HJ+00] K. J. Hunt, T. a. Johansen, J. Kalkkuhl, H. Fritz, and T. Göttsche. “Speed Control Design for an Experimental Vehicle Using a Generalized Gain Scheduling Approach”. In: *Control* 8.3 (2000), pp. 381–395. ISSN: 10636536. DOI: 10.1109/87.845870 (cit. on p. 55).

[HK+85] K. Hamabe, K. Kitoh, H. Ogata, and T. Kobayashi. “An Estimation of Aerodynamic Drag Coefficient of a Passenger Car by Coast-down Tests in Windy Environments”. In: *JSAE Review* 16 (1985), pp. 114–120. URL: http://www.sciencedirect.com/science/journal/03894304 (cit. on p. 39).

[HL+18] Y. Hu, W. Li, K. Xu, T. Zahid, F. Qin, and C. Li. “Energy Management Strategy for a Hybrid Electric Vehicle Based on Deep Reinforcement Learning”. In: *Applied Sciences* 8.2 (2018), p. 187. ISSN: 2076-3417. DOI: 10.3390/app8020187. URL: http://www.mdpi.com/2076-3417/8/2/187 (cit. on p. 193).

[HL+19] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. “Learning agile and dynamic motor skills for legged robots”. In: (2019), pp. 1–14. ISSN: 2470-9476. DOI: 10.1126/scirobotics.aau5872. URL: http://robotics.sciencemag.org/ (cit. on pp. 185, 221).

[HP86] C. R. Hargraves and S. W. Paris. “Direct trajectory optimization using nonlinear programming and collocation”. In: *Astrodynamics Conference, 1986* 10.4 (1986). DOI: 10.2514/6.1986-2000 (cit. on p. 226).

[HR05] E. L. Haseltine and J. B. Rawlings. “Critical evaluation of extended Kalman filtering and moving-horizon estimation”. In: *Industrial and Engineering Chemistry Research* 44.8 (2005), pp. 2451–2460. ISSN: 08885885. DOI: 10.1021/ie034308l (cit. on p. 228).

[HR08] B. Huang and K. Ramesh. “Dynamic Modeling, Predictive Control and Performance Monitoring”. In: *Dynamic Modeling, Predictive Control and Performance Monitoring* (2008). DOI: 10.1007/978-1-84800-233-3 (cit. on p. 139).

[HT+94] J. K. Hedrick, I. M. Tomizuka, and P. Varaiya. “Control Issues in Automated Highway Systems”. In: *IEEE Control Systems* 14.6 (1994), pp. 21–32. ISSN: 1066033X. DOI: 10.1109/37.334412 (cit. on pp. 49, 54).

[Hub73] P. J. Huber. “Robust Regression: Asymptotics, Conjectures and Monte Carlo”. In: *The Annals of Statistics* 1.5 (Sept. 1973), pp. 1403–1433. ISSN: 0090-5364. DOI: 10.1214/aos/1176342503 (cit. on p. 264).

[Huc08] W.-H. Hucho. *Aerodynamik des Automobils*. Ed. by W.-H. Hucho. Vol. 5. Wiesbaden: Vieweg+Teubner Verlag, 2008. ISBN: 978-3-663-09218-6. DOI: 10.1007/978-3-663-09217-9. URL: http://link.springer.com/10.1007/978-3-663-09217-9 (cit. on pp. 39, 222).

[HW+18] C.-J. Hoel, K. Wolff, and L. Laine. "Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning". In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. 2018, pp. 2148–2155. ISBN: 9781728103228. DOI: arXiv:1803.10056v2. arXiv: 1803.10056. URL: http://arxiv.org/abs/1803.10056 (cit. on p. 192).

[HW12] X. Huang and J. Wang. "Adaptive vehicle planar motion control with fast parameter estimation". In: *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2012), pp. 5034–5039. ISSN: 01912216. DOI: 10.1109/CDC.2012.6426997 (cit. on pp. 94, 95).

[HW14] M. Horn and D. Watzenig. *Automated Driving*. 2014, pp. 1–16. DOI: 10.1002/9781118354179.auto023. URL: http://doi.wiley.com/10.1002/9781118354179.auto023 (cit. on p. 49).

[HZ+15] K. He, X. Zhang, S. Ren, and J. Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2015 Inter. 2015, pp. 1026–1034. ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.123. arXiv: 1502.01852 (cit. on p. 205).

[HZ+18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *35th International Conference on Machine Learning, ICML 2018* 5 (2018), pp. 2976–2989. arXiv: 1801.01290 (cit. on p. 227).

[IB19] S. A. U. Islam and D. S. Bernstein. "Recursive Least Squares for Real-Time Implementation". In: *IEEE Control Systems Magazine* 39.3 (2019), pp. 82–85. ISSN: 1941000X. DOI: 10.1109/MCS.2019.2900788 (cit. on p. 241).

[IS15] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv preprint* (2015). ISSN: 0717-6163. DOI: 10.1007/s13398-014-0173-7.2. arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167 (cit. on pp. 189, 202).

[Ise06] R. Isermann. *Fahrdynamikregelung*. ATZ/MTZ Fachbuch, 2006 (cit. on p. 35).

[Isi85] A. Isidori. "Nonlinear Control Systems: An Introduction". In: *Nonlinear Control Systems: An Introduction*. 1985 (cit. on p. 270).

[ISO11] ISO. *ISO - ISO 8855:2011 - Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary*. 2011. URL: https://www.iso.org/standard/51180.html (cit. on p. 27).

[IX+93] P. Ioannou, Z. Xu, S. Eckert, D. Clemons, and T. Sieja. "Intelligent cruise control: theory and experiment". In: *Proceedings of 32nd IEEE Conference on Decision and Control*. 1993. DOI: 10.1109/cdc.1993.325521 (cit. on p. 50).

[IX00] K. Ito and K. Xiong. "Gaussian filters for nonlinear filtering problems". In: *IEEE Transactions on Automatic Control* 45.5 (May 2000), pp. 910–927. ISSN: 00189286. DOI: 10.1109/9.855552. URL: http://ieeexplore.ieee.org/document/855552/ (cit. on p. 250).

[JH05] A. Jadbabaie and J. Hauser. “On the stability of receding horizon control with a general terminal cost”. In: *IEEE Transactions on Automatic Control* 50.5 (2005), pp. 674–678. ISSN: 00189286. DOI: 10.1109/TAC.2005.846597 (cit. on p. 143).

[JJ+82] R. Johnstone, C. Johnson, R. Bitmead, and B. O. Anderson. “Exponential convergence of recursive least squares with exponential forgetting factor”. In: *1982 21st IEEE Conference on Decision and Control*. IEEE, Dec. 1982, pp. 994–997. DOI: 10.1109/CDC.1982.268295. URL: http://ieeexplore.ieee.org/document/4047398/ (cit. on pp. 241, 272).

[JK+07] J. B. Jørgensen, M. R. Kristensen, P. G. Thomsen, and H. Madsen. “New extended Kalman filter algorithms for stochastic differential algebraic equations”. In: *Lecture Notes in Control and Information Sciences* 358.1 (2007), pp. 359–366. ISSN: 01708643. DOI: 10.1007/978-3-540-72699-9_29 (cit. on p. 226).

[JK+20] I. A. Jimoh, I. B. Kucukdemiral, G. Bevan, and P. E. Orukpe. “Offset-free Model Predictive Control: A Study of Different Formulations with Further Results”. In: *2020 28th Mediterranean Conference on Control and Automation, MED 2020* December (2020), pp. 671–676. DOI: 10.1109/MED48518.2020.9183056 (cit. on p. 143).

[Joh04] T. A. Johansen. “Approximate explicit receding horizon control of constrained nonlinear systems”. In: *Automatica* 40.2 (2004), pp. 293–300. ISSN: 00051098 (cit. on p. 225).

[Joh11] T. A. Johansen. “Introduction to nonlinear model predictive control and moving horizon estimation”. In: *Topics on Constrained and Nonlinear Control*. 1. 2011. Chap. Introducti, pp. 1–53. ISBN: 978-80-968627-4-0. URL: https://iam.chtf.stuba.sk/%7B~%7Dfikar/nil11/nil-tbook-p.pdf%7B%5C#%7Dpage=201 (cit. on p. 141).

[JU97] S. J. Julier and J. K. Uhlmann. “New extension of the Kalman filter to nonlinear systems”. In: *Signal Processing, Sensor Fusion, and Target Recognition VI* 3068 (1997), p. 182. ISSN: 0277786X. DOI: 10.1117/12.280797 (cit. on pp. 246, 247).

[Jul02] S. J. Julier. “The scaled unscented transformation”. In: *Proceedings of the American Control Conference* 6.2 (2002), pp. 4555–4559. ISSN: 07431619. DOI: 10.1109/acc.2002.1025369 (cit. on pp. 246, 247, 250).

[JY+01] A. Jadbabaie, J. Yu, and J. Hauser. “Unconstrained receding-horizon control of nonlinear systems”. In: *IEEE Transactions on Automatic Control* 46.5 (2001), pp. 776–783. ISSN: 00189286. DOI: 10.1109/9.920800 (cit. on p. 143).

[Kai74] Kai-Ching Chu. “Optimal dencentralized regulation for a string of coupled systems”. In: *IEEE Transactions on Automatic Control* 19.3 (1974), pp. 243–246. ISSN: 0018-9286. DOI: 10.1109/TAC.1974.1100538. URL: http://ieeexplore.ieee.org/document/1100538/ (cit. on pp. 10, 58).

[Kal21] D. Kalandyk. “Reinforcement learning in car control : A brief survey”. In: *2021 Selected Issues of Electrical Engineering and Electronics (WZEE)* (2021), pp. 1–8. DOI: 10.1109/WZEE54157.2021.9576838 (cit. on p. 191).

[Kal60a] R. Kalman. “On the general theory of control systems”. In: *IFAC Proceedings Volumes* 1.1 (1960), pp. 491–502. ISSN: 14746670. DOI: 10.1016/s1474-6670(17)70094-8. URL: http://dx.doi.org/10.1016/S1474-6670(17)70094-8 (cit. on pp. 270, 275, 276).

[Kal60b] R. E. Kalman. “A new approach to linear filtering and prediction problems”. In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45 (cit. on p. 243).

[KB+12] S. C. Kadu, M. Bhushan, and K. Roy. *Continuous Discrete Unscented Kalman Filtering for Nonlinear Differential Algebraic Equations Systems*. Vol. 31. July. Elsevier B.V., 2012, pp. 940–944. DOI: 10.1016/B978-0-444-59506-5.50019-5. URL: http://dx.doi.org/10.1016/B978-0-444-59506-5.50019-5 (cit. on p. 227).

[KB+19] T. Kessler, J. Bernhard, M. Buechel, K. Esterle, P. Hart, D. Malovetz, M. Truong Le, and A. Knoll. “Bridging the gap between open source software and vehicle hardware for autonomous driving”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019. DOI: 10.1109/IVS.2019.8813784. URL: https://ieeexplore.ieee.org/document/8813784/ (cit. on p. 7).

[KB14] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). arXiv: arXiv:1412.6980,. URL: https://arxiv.org/abs/1412.6980 (cit. on p. 205).

[KD+14] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin. “On Particle Methods for Parameter Estimation in State-Space Models”. In: 30.3 (2014), pp. 328–351. DOI: 10.1214/14-STS511. arXiv: 1412.8695 (cit. on p. 114).

[KG12] K. Kritayakirana and J. C. Gerdes. “Autonomous vehicle control at the limits of handling”. In: *International Journal of Vehicle Autonomous Systems* 10.4 (2012), p. 271. ISSN: 1471-0226. DOI: 10.1504/IJVAS.2012.051270. URL: http://www.inderscience.com/link.php?id=51270 (cit. on pp. 19, 57, 89).

[KH+14] N. Kidambi, R. L. Harne, Y. Fujii, G. M. Pietron, and K. W. Wang. “Methods in Vehicle Mass and Road Grade Estimation”. In: *SAE International Journal of Passenger Cars - Mechanical Systems* 7.3 (2014), pp. 2014–01–0111. ISSN: 1946-4002. DOI: 10.4271/2014-01-0111. URL: http://papers.sae.org/2014-01-0111/ (cit. on pp. 88, 89, 91, 92).

[KH+18] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. “Learning to Drive in a Day”. In: (2018). DOI: arXiv:1807.00412v1. arXiv: 1807.00412. URL: http://arxiv.org/abs/1807.00412 (cit. on pp. 23, 192).

[KI+08] R. Kandepu, L. Imsland, and B. A. Foss. “Constrained state estimation using the unscented kalman filter”. In: *2008 Mediterranean Conference on Control and Automation - Conference Proceedings, MED’08* (2008), pp. 1453–1458. DOI: 10.1109/MED.2008.4602001 (cit. on p. 258).

[Kir18] M. Kirchengast. “Control Allocation Techniques for Redundantly Actuated Systems”. PhD thesis. Graz University of Technology, 2018, p. 209 (cit. on pp. 148, 149).

[KK+21a] P. Karg, F. Köpf, C. A. Braun, and S. Hohmann. “Excitation for Adaptive Optimal Control of Nonlinear Systems in Differential Games”. In: (2021), pp. 1–8. arXiv: 2105.02260. URL: http://arxiv.org/abs/2105.02260 (cit. on p. 225).

[KK+21b] A. H. Korayem, A. Khajepour, and B. Fidan. “A Review on Vehicle-Trailer State and Parameter Estimation”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2021), pp. 5993–6010. ISSN: 15580016. DOI: 10.1109/TITS.2021.3074457 (cit. on p. 93).

[KN+09] T. V. Keulen, G. Naus, and B. D. Jager. "Predictive Cruise Control in Hybrid Electric Vehicles". In: *World Electric Vehicle Journal* 3 (2009), pp. 494–504 (cit. on p. 52).

[KO63] R. E. Kopp and R. J. Orford. "Linear regression appied to system identification for adaptive control". In: *AIAA Journal* 1.10 (1963), pp. 2300–2306. ISSN: 0001-1452. DOI: 10.2514/3.2056. URL: http://arc.aiaa.org/doi/10.2514/3.2056 (cit. on p. 259).

[Köp22] F. Köpf. "Adaptive Dynamic Programming : Solltrajektorienfolgeregelung und Konvergenzbedingungen". In: (2022) (cit. on p. 224).

[Kot82] A. J. Kotwicki. "Dynamic models for torque converter equipped vehicles". In: *SAE Technical Papers* (1982), pp. 1595–1609. ISSN: 26883627. DOI: 10.4271/820393 (cit. on p. 33).

[KP+10] J. Z. Kolter, C. Plagemann, D. T. Jackson, A. Y. Ng, and S. Thrun. "A Probabilistic Approach to Mixed Open-loop and Closed-loop Control , with Application to Extreme Autonomous Driving". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 839–845. ISBN: 9781424450404 (cit. on p. 192).

[KP+20] F. Köpf, L. Puccetti, C. Rathgeber, and S. Hohmann. "Reinforcement Learning for Speed Control with Feedforward to Track Velocity Profiles in a Real Vehicle". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC 2020* (2020). DOI: 10.1109/ITSC45102.2020.9294541 (cit. on pp. 194, 224).

[KR+10] R. Kumar Mandela, R. Rengaswamy, S. Narasimhan, and L. N. Sridhar. "Recursive state estimation techniques for nonlinear differential algebraic systems". In: *Chemical Engineering Science* 65.16 (2010), pp. 4548–4556. ISSN: 00092509. DOI: 10.1016/j.ces.2010.04.020. URL: http://dx.doi.org/10.1016/j.ces.2010.04.020 (cit. on p. 226).

[KR+20] F. Köpf, S. Ramsteiner, L. Puccetti, M. Flad, and S. Hohmann. "Adaptive dynamic programming for model-free tracking of trajectories with time-varying parameters". In: *International Journal of Adaptive Control and Signal Processing* 34.7 (2020), pp. 839–856. ISSN: 10991115. DOI: 10.1002/acs.3106. arXiv: 1909.07239 (cit. on p. 224).

[KR00] C. Kim and P. I. Ro. "Reduced-order modelling and parameter estimation for a quarter-car suspension system". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 214.8 (2000), pp. 851–864. ISSN: 09544070. DOI: 10.1243/0954407001527907 (cit. on pp. 94, 95).

[KW+19] F. Köpf, J. Westermann, M. Flad, and S. Hohmann. "Reinforcement-Learning-based Adaptive Optimal Control for Arbitrary Reference Tracking". In: *arXiv preprint*. June 2019. arXiv: 1906.05085. URL: https://arxiv.org/abs/1906.05085v1 (cit. on pp. 194, 213).

[LB+05] R. Leithgoeb, M. Buechel, M. Bollig, and F. Henzinger. "Methodology for efficient calibration of model based ECU structures". In: *1st International Symposium on Development Methodology*. Wiesbaden, 2005, pp. 195–208 (cit. on p. 40).

[LB+15] T. Li, M. Bolić, and P. M. Djurić. "Resampling Methods for Particle Filtering: Classification, implementation, and strategies". In: *IEEE Signal Processing Magazine* 32.3 (2015), pp. 70–86. ISSN: 10535888. DOI: 10.1109/MSP.2014.2330626 (cit. on pp. 114, 257).

[LC+16] S. Lefèvre, A. Carvalho, and F. Borrelli. “A learning-based framework for velocity control in autonomous driving”. In: *IEEE Transactions on Automation Science and Engineering* 13.1 (2016), pp. 32–42. ISSN: 15455955. DOI: 10.1109/TASE.2015.2498192 (cit. on p. 53).

[LD+19] R. Liessner, A. M. Dietermann, and B. Bäker. “Safe Deep Reinforcement Learning Hybrid Electric Vehicle Energy Management”. In: Springer, Cham, Jan. 2019, pp. 161–181. DOI: 10.1007/978-3-030-05453-3_8. URL: http://link.springer.com/10.1007/978-3-030-05453-3%7B%5C_%7D8 (cit. on p. 193).

[LeC22] Y. LeCun. *A Path Towards Autonomous Machine Intelligence*. 2022. URL: https://openreview.net/forum?id=BZ5a1r-kVsf (cit. on pp. 218, 221, 229).

[Lee12] D. O. Q. Lee. “Numerically efficient methods for solving least squares problems”. In: (2012), pp. 1–15 (cit. on p. 239).

[Lee17] J.-h. Lee. “Control Oriented Modeling of Dynamic Torque Converter System”. In: *2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)* (2017), pp. 1–5. DOI: 10.1109/ICMSAO.2017.7934860 (cit. on p. 33).

[LH+02] X.-y. Lu, J. K. Hedrick, Xiao-Yun Lu, J. K. Hedrick, and M. Drew. “ACC/CACC - control design, stability and robust performance”. In: *Proceedings of the 2002 American Control Conference* (2002), pp. 4327–4332. DOI: 10.1109/ACC.2002.1025325. URL: http://ieeexplore.ieee.org/document/1025325/ (cit. on p. 51).

[LH+15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. “Continuous Control with Deep Reinforcement Learning”. In: *arXiv preprint arXiv:1509.02971* (2015), pp. 1–14. ISSN: 1935-8237. DOI: 10.1561/2200000006. arXiv: 1509.02971. URL: http://arxiv.org/abs/1509.02971 (cit. on pp. 189, 193, 202, 203, 205).

[Li16] Y. Li. “Modelling and measurement of transient torque converter characteristics”. PhD thesis. Chalmers Universitiy of Technology, 2016 (cit. on p. 33).

[Lin] Link. *Towardsdatascience*. URL: https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79 (cit. on p. 205).

[Lin92] L.-J. Lin. “Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching”. In: *Reinforcement Learning*. Boston: Springer US, 1992, pp. 69–97. DOI: 10.1007/978-1-4615-3618-5_5 (cit. on p. 187).

[LJ+15] S. E. Li, Z. Jia, K. Li, and B. Cheng. “Fast Online Computation of a Model Predictive Controller and Its Application to Fuel Economy Oriented Adaptive Cruise Control”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.3 (2015), pp. 1199–1209. ISSN: 1524-9050. DOI: 10.1109/TITS.2014.2354052 (cit. on p. 52).

[LL+11] S. Li, K. Li, R. Rajamani, and J. Wang. “Model predictive multi-objective vehicular adaptive cruise control”. In: *IEEE Transactions on Control Systems Technology* 19.3 (2011), pp. 556–566. ISSN: 10636536. DOI: 10.1109/TCST.2010.2049203 (cit. on p. 52).

[LM+19] Y. Lin, J. McPhee, and N. L. Azad. “Comparison of Deep Reinforcement Learning and Model Predictive Control for Adaptive Cruise Control”. In: (2019), pp. 1–9. arXiv: 1910.12047. URL: http://arxiv.org/abs/1910.12047 (cit. on p. 193).

[Lom] N. Lomas. *Tesla Autopilot*. Techcrunch. URL: https://techcrunch.com/2016/10/17/tesla-told-to-stop-using-misleading-term-autopilot-in-germany/ (visited on 04/29/2019) (cit. on p. 20).

[LP99] C.-Y. Liang and H. Peng. "Optimal Adaptive Cruise Control with Guaranteed String Stability". In: *Vehicle System Dynamics* 32.4-5 (1999), pp. 313–330. ISSN: 0042-3114. DOI: 10.1076/vesd.32.4.313.2083 (cit. on p. 50).

[LS+19] R. Liessner, C. Schroer, A. Dietermann, and B. Bäker. "Deep Reinforcement Learning for Advanced Energy Management of Hybrid Electric Vehicles". In: (Jan. 2019), pp. 61–72. URL: http://www.scitepress.org/PublicationsDetail.aspx?ID=rfkxMQOLYgU%7B%5C%%7D3D (cit. on p. 193).

[LT+17] Y. C. Lin, H. L. T. Thi, and C. H. Wang. "Adaptive neuro-fuzzy predictive control for design of adaptive cruise control system". In: *Proceedings of the 2017 IEEE 14th International Conference on Networking, Sensing and Control, ICNSC 2017* (2017), pp. 767–772. DOI: 10.1109/ICNSC.2017.8000187 (cit. on p. 51).

[LV09] F. L. Lewis and D. Vrabie. "Reinforcement learning and adaptive dynamic programming for feedback control". In: *IEEE Circuits and Systems Magazine* 9.3 (2009), pp. 32–50. ISSN: 1531636X. DOI: 10.1109/MCAS.2009.933854 (cit. on pp. 184, 223).

[LW+12] G. Liu, F. Worgotter, and I. Markelic. "Square-root sigma-point information filtering". In: *IEEE Transactions on Automatic Control* 57.11 (2012), pp. 2945–2950. ISSN: 00189286. DOI: 10.1109/TAC.2012.2193708 (cit. on pp. 117–119, 240, 253–256).

[LW+14] X. Lin, Y. Wang, P. Bogdan, N. Chang, and M. Pedram. "Reinforcement learning based power management for hybrid electric vehicles". In: *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*. 2014, pp. 33–38. ISBN: 978-1-4799-6277-8. URL: http://dl.acm.org/citation.cfm?id=2691365.2691372 (cit. on p. 193).

[LY+05] J. Luo, K. Ying, and J. Bai. "Savitzky–Golay smoothing and differentiation filter for even number data". In: *Signal Processing* 85.7 (July 2005), pp. 1429–1434. ISSN: 01651684. DOI: 10.1016/j.sigpro.2005.02.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0165168405000654 (cit. on pp. 62, 63).

[MA+90] L. D. Metz, C. K. Akouris, C. S. Agney, and M. C. Clark. "Moments of Inertia of Mounted and Unmounted Passenger Car and Motorcycle Tires". In: *SAE International* (1990). DOI: 10.4271/900760. URL: https://www.sae.org/content/900760/ (cit. on p. 37).

[Mac02] J. Maciejowski. *Predictive control: with constraints*. 2002 (cit. on p. 140).

[Mad78] H. H. Madden. "Comments on the Savitzky-Golay Convolution Method for Least-Squares Fit Smoothing and Differentiation of Digital Data". In: *Analytical Chemistry* 50.9 (1978), pp. 1383–1386. ISSN: 15206882. DOI: 10.1021/ac50031a048 (cit. on p. 63).

[Man80] T. A. Manteuffel. *The Weighted Linear Least Squares Problem : An Interval Analysis Approach to Rank Determination*. Albuquerque, 1980. URL: https://www.osti.gov/servlets/purl/5047753 (cit. on p. 74).

[Mat16] MathWorks Inc. *MATLAB R2016a*. 2016 (cit. on p. 148).

[Mat19a] MathWorks Inc. *cholupdate*. 2019. URL: mathworks.com/help/matlab/ref/cholupdate.html (cit. on p. 254).

[Mat19b] MathWorks Inc. *MATLAB R2019b*. 2019. URL: https://de.mathworks.com/products/matlab.html (cit. on p. 113).

[Mat19c] MathWorks Inc. *MATLAB sgolay*. 2019. URL: https://de.mathworks.com/help/signal/ref/sgolay.html (cit. on pp. 64–67).

[Mau74] J. P. Maury. “The ARAMIS PRT System”. In: *SAE Technical Paper* 740143 (1974) (cit. on p. 18).

[May99] T. Mayer. *Modellierung und Regelung des autarken Hybridfahrzeugs*. Ed. by H. U. Verlag. Herbert Utz Verlag, 1999 (cit. on p. 35).

[MB+16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. “Asynchronous Methods for Deep Reinforcement Learning”. In: (2016). arXiv: 1602.01783. URL: http://arxiv.org/abs/1602.01783 (cit. on p. 189).

[MB+17] B. Mirchevska, M. Blum, L. Louis, J. Boedecker, and M. Werling. *Reinforcement Learning for Autonomous Maneuvering in Highway Scenarios*. Walting, 2017 (cit. on p. 192).

[MC+12] M. Mühlegg, G. Chowdhary, and E. N. Johnson. “Concurrent learning adaptive control of linear systems with noisy measurements”. In: *AIAA Guidance, Navigation, and Control Conference 2012* (2012), pp. 1–13. DOI: 10.2514/6.2012-4669 (cit. on p. 227).

[MC07] J. J. Martinez and C. Canudas-de-Wit. “A safe longitudinal control for adaptive cruise control and stop-and-go scenarios”. In: *IEEE Transactions on Control Systems Technology* 15.2 (2007), pp. 246–258 (cit. on p. 53).

[MD+18] L. Menhour, B. D’Andrea-Novel, M. Fliess, D. Gruyer, and H. Mounier. “An Efficient Model-Free Setting for Longitudinal and Lateral Vehicle Control: Validation Through the Interconnected Pro-SiVIC / RTMaps Prototyping Platform”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.2 (2018), pp. 461–475. ISSN: 15249050. DOI: 10.1109/TITS.2017.2699283. arXiv: 1705.03216 (cit. on p. 56).

[ME09] A. Medvedev and M. Evestedt. “Elementwise decoupling and convergence of the Riccati equation in the SG algorithm”. In: *Automatica* 45.6 (2009), pp. 1524–1529. ISSN: 00051098. DOI: 10.1016/j.automatica.2009.02.010. URL: http://dx.doi.org/10.1016/j.automatica.2009.02.010 (cit. on p. 262).

[Mer04] R. V. D. Merwe. “Sigma-point Kalman filters for probabilistic inference in dynamic state-space models”. PhD thesis. Oregon Health & Science University, 2004, p. 378. URL: http://www.cslu.ogi.edu/publications/ps/merwe04.pdf (cit. on pp. 231, 246, 247, 249, 250, 253, 256).

[MG+09] M. L. McIntyre, T. J. Ghotikar, A. Vahidi, X. Song, and D. M. Dawson. “A two-stage Lyapunov-based estimator for estimation of vehicle mass and road grade”. In: *IEEE Transactions on Vehicular Technology* 58.7 (2009), pp. 3177–3185 (cit. on pp. 91, 95).

[MG+18] H. Mania, A. Guy, and B. Recht. “Simple random search provides a competitive approach to reinforcement learning”. In: *arXiv preprint* (2018), pp. 1–22. arXiv: 1803.07055. URL: https://arxiv.org/pdf/1803.07055v1.pdf (cit. on p. 187).

[MG+21] P. K. Mishra, M. V. Gasparino, A. E. B. Velsasquez, and G. Chowdhary. *Deep model predictive control for a class of nonlinear systems*. 2021. arXiv: 2104.07171. URL: http://arxiv.org/abs/2104.07171 (cit. on pp. 226, 228).

[MH+90] D. McMahon, J. K. Hedrick, and S. E. Shladover. “Vehicle Modelling and Control for Automated Highway Systems”. In: *American Control Conference*. San Diego, CA, USA, 1990 (cit. on p. 54).

[MK+13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. “Playing Atari with Deep Reinforcement Learning”. In: *arXiv preprint* (2013), pp. 1–9. ISSN: 0028-0836. DOI: 10.1038/nature14236. arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602 (cit. on pp. 184, 187, 193, 202).

[MM+09] S. Moon, I. Moon, and K. Yi. “Design, tuning, and evaluation of a full-range adaptive cruise control system with collision avoidance”. In: *Control Engineering Practice* 17.4 (2009), pp. 442–455. ISSN: 09670661. DOI: 10.1016/j.conengprac.2008.09.006 (cit. on p. 50).

[MM+18] S. M. Mohtavipour, M. Mollajafari, and A. Naseri. “A guaranteed-comfort and safe adaptive cruise control by considering driver’s acceptance level”. In: *International Journal of Dynamics and Control* (2018). ISSN: 21952698. DOI: 10.1007/s40435-018-0500-5. URL: https://doi.org/10.1007/s40435-018-0500-5 (cit. on p. 51).

[MM00] B. Matei and P. Meer. “General method for errors-in-variables problems in computer vision”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2 (2000), pp. 18–25. ISSN: 10636919. DOI: 10.1109/cvpr.2000.854727 (cit. on pp. 102, 262, 263).

[MM12] M. Morari and U. Maeder. “Nonlinear offset-free model predictive control”. In: *Automatica* 48.9 (2012), pp. 2059–2067. ISSN: 00051098. DOI: 10.1016/j.automatica.2012.06.038. URL: http://dx.doi.org/10.1016/j.automatica.2012.06.038 (cit. on pp. 140, 143–145, 154, 156, 224).

[MO+09] G. Marafioti, S. Olaru, and M. Hovd. “State estimation in nonlinear model predictive control, unscented kalman filter advantages”. In: *Lecture Notes in Control and Information Sciences*. Vol. 384. 2009, pp. 305–313. ISBN: 9783642010934. DOI: 10.1007/978-3-642-01094-1_25 (cit. on p. 115).

[MP+18] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker. “High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning”. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. 2018, pp. 2156–2162. ISBN: 9781728103228 (cit. on p. 192).

[MP+21] D. Q. de Menezes, D. M. Prata, A. R. Secchi, and J. C. Pinto. “A review on robust M-estimators for regression analysis”. In: *Computers and Chemical Engineering* 147 (2021), p. 107254. ISSN: 00981354. DOI: 10.1016/j.compchemeng.2021.107254. URL: https://doi.org/10.1016/j.compchemeng.2021.107254 (cit. on pp. 103, 264).

[MR+00] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814. ISSN: 00051098. DOI: 10.1016/S0005-1098(99)00214-9 (cit. on pp. 139, 141).

[MS06] L. Magni and R. Scattolini. “Stabilizing decentralized model predictive control of nonlinear systems”. In: *Automatica* 42.7 (2006), pp. 1231–1236 (cit. on p. 142).

[MS14] S. V. Menon and C. S. Seelamantula. “Robust Savitzky-Golay filters”. In: *2014 19th International Conference on Digital Signal Processing*. 1. IEEE, Aug. 2014, pp. 688–693. ISBN: 978-1-4799-4612-9. DOI: 10.1109/ICDSP.2014.6900752. URL: http://ieeexplore.ieee.org/document/6900752/ (cit. on pp. 62, 63, 83).

[MS17a] L. Marina and A. Sandu. “Deep Reinforcement Learning for Autonomous Vehicles - State of the Art”. In: *Bulletin of the Transilvania University of Brasov* 10.2 (2017). arXiv: 1701.08878. URL: http://arxiv.org/abs/1701.08878 (cit. on p. 23).

[MS17b] K. D. Mishra and K. Srinivasan. “On-line Identification of a Torque Converter Model”. In: *IFAC-PapersOnLine* 50.1 (2017), pp. 4763–4768. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2017.08.952. URL: https://doi.org/10.1016/j.ifacol.2017.08.952 (cit. on p. 33).

[MT+05] M. J. Messina, S. E. Tuna, and A. R. Teel. “Discrete-time certainty equivalence output feedback: Allowing discontinuous control laws including those from model predictive control”. In: *Automatica* 41.4 (2005), pp. 617–628. ISSN: 00051098. DOI: 10.1016/j.automatica.2004.11.015 (cit. on p. 141).

[Mut98] A. G. O. Mutambara. *Decentralized estimation and control for multisensor systems*. CRC press, 1998. ISBN: 9780849318658 (cit. on p. 252).

[MV+15] C. M. Martinez, E. Velenis, D. Tavernini, B. Gao, and M. Wellers. “Modelling and estimation of friction brake torque for a brake by wire system”. In: *2014 IEEE International Electric Vehicle Conference, IEVC 2014* December (2015). DOI: 10.1109/IEVC.2014.7056105 (cit. on p. 36).

[MW01] R. V. D. Merwe and E. Wan. “Efficient derivative-free Kalman filters for online learning”. In: *Esann* April (2001), pp. 205–210 (cit. on p. 226).

[MW19] Y. Ma and J. Wang. “Predictive Control for NOx Emission Reductions in Diesel Engine Vehicle Platoon Application”. In: *IEEE Transactions on Vehicular Technology* PP.x (2019), pp. 1–1. ISSN: 0018-9545. DOI: 10.1109/TVT.2019.2914062. URL: https://ieeexplore.ieee.org/document/8703069/ (cit. on pp. 26, 166, 222).

[MY08] S. Moon and K. Yi. “Human driving data-based design of a vehicle adaptive cruise control algorithm”. In: *Vehicle System Dynamics* 46.8 (2008), pp. 661–690. ISSN: 00423114. DOI: 10.1080/00423110701576130 (cit. on p. 50).

[MY09] A. Murayama and M. Yamakita. “Speed Control of Vehicles with Variable Valve Lift Engine by Nonlinear MPC”. In: *ICROS-SICE International Joint Conference*. Vol. 1. 2009, pp. 4128–4133. ISBN: 9781479977864 (cit. on p. 56).

[NB+08] G. Naus, R. van den Bleek, J. Ploeg, B. Scheepers, R. van de Molengraft, and M. Steinbuch. “Explicit MPC design and performance evaluation of an ACC Stop-&-Go”. In: *2008 American Control Conference*. IEEE, 2008, pp. 224–229. ISBN: 978-1-4244-2078-0. DOI: 10.1109/ACC.2008.4586495. URL: http://ieeexplore.ieee.org/document/4586495/ (cit. on pp. 52, 225).

[NC+08] L. Ng, C. M. Clark, and J. P. Huissoon. “Reinforcement learning of adaptive longitudinal vehicle control for dynamic collaborative driving”. In: *IEEE Intelligent Vehicles Symposium, Proceedings* (2008), pp. 907–912. DOI: 10.1109/IVS.2008.4621222 (cit. on p. 193).

[Nel00] A. T. Nelson. “Nonlinear Estimation and Modeling of Noisy Time-Series by Dual Kalman Filtering Methods”. PhD thesis. Oregon Graduate Institute, 2000 (cit. on pp. 259, 260).

[Nel01] O. Nelles. *Nonlinear System Identification*. Heidelberg: Springer, 2001. ISBN: 9783642086748 (cit. on p. 199).

[NG+07] J. E. Naranjo, C. González, R. García, and T. de Pedro. “Cooperative throttle and brake fuzzy control for ACC+Stop&Go maneuvers”. In: *IEEE Transactions on Vehicular Technology* 56.4 I (2007), pp. 1623–1630. ISSN: 00189545. DOI: 10.1109/TVT.2007.897632 (cit. on p. 51).

[NHT09] NHTSA. *NHTSA Tire Fuel Efficiency Consumer Information Program Development : Phase 2 – Effects of Tire Rolling Resistance Levels on Traction, Treadwear, and Vehicle Fuel Economy*. Springfield, Virginia, 2009 (cit. on p. 88).

[NJ+12] J. E. Naranjo, F. Jiménez, O. Gómez, and J. G. Zato. “Low level control layer definition for autonomous vehicles based on fuzzy logic”. In: *Intelligent Automation and Soft Computing* 18.4 (2012), pp. 333–348. ISSN: 2326005X. DOI: 10.1080/10798587.2012.10643247 (cit. on p. 55).

[NK+17] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. “Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning”. In: (Aug. 2017). arXiv: 1708.02596. URL: http://arxiv.org/abs/1708.02596 (cit. on p. 201).

[NM+98] G. de Nicolao, L. Magni, and R. Scattolini. “Stabilizing nonlinear receding horizon control via a nonquadratic terminal state penalty”. In: *CESA’96 IMACS Multiconference: computational engineering in systems applications* (1998), pp. 185–187 (cit. on p. 142).

[Nor40] Norman Bel Geddes. *Magic Motorways*. Read Books Ltd, 1940. URL: https://books.google.de/books?isbn=1446545776 (cit. on p. 17).

[NP+00] M. Nørgaard, N. K. Poulsen, and O. Ravn. “New developments in state estimation for nonlinear systems”. In: *Automatica* 36.11 (Nov. 2000), pp. 1627–1638. ISSN: 00051098. DOI: 10.1016/S0005-1098(00)00089-3. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109800000893 (cit. on p. 250).

[Ope] OpenAI. *Welcome to Spinning Up in Deep RL! — Spinning Up documentation*. URL: https://spinningup.openai.com/en/latest/index.html (cit. on p. 176).

[Orf10] S. J. Orfanidis. *Introduction to signal processing*. 2010, pp. 427–453. ISBN: 0-13-209172-0. URL: https://www.ece.rutgers.edu/%7B~%7Dorfanidi/intro2sp/ (cit. on pp. 63, 64, 66, 67, 76).

[Osh65] R. Oshima. “Control System for Automobile Driving”. In: *Proceedings of the IFAC Symposium Tokyo 1965* (1965) (cit. on p. 18).

[PB+21] K. Pereida, L. Brunke, and A. P. Schoellig. “Robust adaptive model predictive control for guaranteed fast and accurate stabilization in the presence of model errors”. In: *International Journal of Robust and Nonlinear Control* 31.18 (2021), pp. 8750–8784. ISSN: 10991239. DOI: 10.1002/rnc.5712 (cit. on p. 141).

[PB+61] L. S. Pontryagin, V. G. Boltyanskiĭ, and E. F. Mishchenko. “The mathematical theory of optimal processes”. In: *Gosudarstv. Izdat. Fiz.-Mat. Lit.* Moscow, 1961 (cit. on pp. 139, 146).

[PB+99] M. Persson, F. Botling, E. Hesslow, and R. Johansson. “Stop and go controller for adaptive cruise control”. In: *IEEE International Conference on Control Applications*. 1999 (cit. on p. 50).

[PD+17] P. Polack, B. D'Andréa-Novel, M. Fliess, A. de La Fortelle, and L. Menhour. "Finite-Time Stabilization of Longitudinal Control for Autonomous Vehicles via a Model-Free Approach". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 12533–12538. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.2191. arXiv: 1704.01383. URL: http://arxiv.org/abs/1704.01383 (cit. on pp. 56, 60, 176).

[PG+15] G. Pannocchia, M. Gabiccini, and A. Artoni. "Offset-free MPC explained: Novelties, subtleties, and applications". In: *IFAC-PapersOnLine* 48.23 (2015), pp. 342–351. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.11.304. URL: http://dx.doi.org/10.1016/j.ifacol.2015.11.304 (cit. on pp. 143, 144).

[PH+18] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz. "Parameter Space Noise for Exploration". In: *ICLR*. 2018, pp. 1–18. arXiv: 1706.01905. URL: http://arxiv.org/abs/1706.01905 (cit. on p. 206).

[PM+16] J. Pavlovic, A. Marotta, B. Ciuffo, S. Serra, G. Fontaras, K. Anagnostopoulos, S. Tsiakmakis, V. Arcidiacono, S. Hausberger, and G. Silberholz. "Correction of Test Cycle Tolerances: Evaluating the Impact on CO2 Results". In: *Transportation Research Procedia* 14.December (2016), pp. 3099–3108. ISSN: 23521465. DOI: 10.1016/j.trpro.2016.05.250. URL: http://dx.doi.org/10.1016/j.trpro.2016.05.250 (cit. on p. 164).

[Pom89] D. Pomerleau. "Alvinn: An autonomous land vehicle in a neural network". In: *Advances in Neural Information Processing Systems 1* (1989), pp. 305–313 (cit. on pp. 18, 23, 191).

[Pow19] W. B. Powell. "From Reinforcement Learning to Optimal Control: A unified framework for sequential decisions". In: (2019). arXiv: 1912.03513. URL: http://arxiv.org/abs/1912.03513 (cit. on p. 223).

[PR+19] L. Puccetti, C. Rathgeber, and S. Hohmann. "Actor-Critic Reinforcement Learning for Linear Longitudinal Output Control of a Road Vehicle". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, NZ: IEEE, 2019, pp. 2907–2913. ISBN: 9781538670231 (cit. on pp. 194, 213).

[PR01] G. Pannocchia and J. Rawlings. "The velocity algorithm LQR: a survey". In: *Texas-Wisconsin Modeling and Control Consortium* September (2001), pp. 1–21. URL: http://jbrwww.che.wisc.edu/tech-reports/twmcc-2001-01.pdf (cit. on p. 143).

[PR03] G. Pannocchia and J. B. Rawlings. "Disturbance models for offset-free model-predictive control". In: *AIChE Journal* 49.2 (2003), pp. 426–437. DOI: 10.1002/aic.690490213 (cit. on p. 143).

[PS+12] J. Ploeg, S. Shladover, H. Nijmeijer, and N. van de Wouw. "Introduction to the Special Issue on the 2011 Grand Cooperative Driving Challenge". In: *IEEE Transactions on Intelligent Transportation Systems* 13.3 (2012), pp. 989–993. ISSN: 1524-9050. DOI: 10.1109/TITS.2012.2210636. URL: http://ieeexplore.ieee.org/document/6266747/ (cit. on p. 19).

[PS12] R. Pintelon and J. Schoukens. *System identification - a frequency domain approach*. 2012. ISBN: 9780470640371 (cit. on p. 262).

[PS63] J. Potter and R. Stern. "Statistical filtering of space navigation measurements". In: *Proc. AIAA Guidance Control Conference* (1963) (cit. on p. 253).

[PT+07] W. H. Press, S. a. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Vol. 1. 2007, p. 1262. ISBN: 0521880688. URL: https://www.amazon.com/Numerical-Recipes-3rd-Edition-Scientific/dp/0521880688 (cit. on pp. 239, 240).

[PT12] O. Pietquin and F. Tango. "A reinforcement learning approach to optimize the longitudinal behavior of a Partial Autonomous Driving Assistance System". In: *Frontiers in Artificial Intelligence and Applications*. Vol. 242. IOS Press, 2012, pp. 987–992. ISBN: 9781614990970. DOI: 10.3233/978-1-61499-098-7-987 (cit. on p. 193).

[QT+15] W. Qiu, Q. U. Ting, Y. U. Shuyou, G. U. O. Hongyan, and C. Hong. "Autonomous Vehicle Longitudinal Following Control Based On Model Predictive Control". In: *Proceedings of the 34th Chinese Control Conference*. Hangzhou, 2015, pp. 8126–8131 (cit. on pp. 52, 55).

[QW+16] X. Qi, G. Wu, K. Boriboonsomsin, M. J. Barth, and J. Gonder. "Data-Driven Reinforcement Learning–Based Real-Time Energy Management System for Plug-In Hybrid Electric Vehicles". In: *Transportation Research Record: Journal of the Transportation Research Board* 2572.January (2016), pp. 1–8. ISSN: 0361-1981. DOI: 10.3141/2572-01 (cit. on p. 193).

[Rad13] T. Radke. "Energieoptimale Laengsfuehrung von Kraftfahrzeugen durch Einsatz vorausschauender Fahrstrategien". PhD thesis. Karlsruher Institut fuer Technologie (KIT), 2013. ISBN: 9783731500698. URL: http://dx.doi.org/10.5445/KSP/1000035819 (cit. on pp. 53, 222).

[Raf13] E. Raffone. "Road slope and vehicle mass estimation for light commercial vehicle using linear Kalman filter and RLS with forgetting factor integrated approach". In: *16th International Conference on Information Fusion* (2013), pp. 1167–1172 (cit. on pp. 91, 95).

[Raj11] R. Rajamani. *Vehicle dynamics and control*. Springer, 2011. ISBN: 0-387-26396-9 (cit. on pp. 27, 29, 35, 49, 50, 88).

[RB+12] A. Reschka, J. R. Böhmer, F. Saust, B. Lichte, and M. Maurer. "Safe, dynamic and comfortable longitudinal control for an autonomous vehicle". In: *IEEE Intelligent Vehicles Symposium, Proceedings* (2012), pp. 346–351. DOI: 10.1109/IVS.2012.6232159 (cit. on p. 57).

[RB+14] S. Rhode, F. Bleimund, and F. Gauterin. *Recursive generalized total least squares with noise covariance estimation*. Vol. 19. 3. IFAC, 2014, pp. 4637–4643. ISBN: 9783902823625. DOI: 10.3182/20140824-6-za-1003.01568. URL: http://dx.doi.org/10.3182/20140824-6-ZA-1003.01568 (cit. on pp. 6, 63, 64, 68, 70, 102, 110, 231).

[RG12] S. Rhode and F. Gauterin. "Vehicle mass estimation using a total least-squares approach". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* Iv (2012), pp. 1584–1589. DOI: 10.1109/ITSC.2012.6338638 (cit. on p. 110).

[RH+16] S. Rhode, S. Hong, J. K. Hedrick, and F. Gauterin. "Vehicle tractive force prediction with robust and windup-stable Kalman filters". In: *Control Engineering Practice* 46 (2016), pp. 37–50. ISSN: 09670661. DOI: 10.1016/j.conengprac.2015.10.002. URL: http://dx.doi.org/10.1016/j.conengprac.2015.10.002 (cit. on pp. 92, 93, 95, 103, 104, 158, 263, 267–269).

[RH95] R. Rajamani and J. K. Hedrick. “Adaptive Observers for Active Automotive Suspensions: Theory and Experiment”. In: *IEEE Transactions on Control Systems Technology* 3.1 (1995), pp. 86–93. DOI: 10.1109/87.370713 (cit. on pp. 29, 94, 95).

[Rho16] S. Rhode. “Robust and Regularized Algorithms for Vehicle Tractive Force Prediction and Mass Estimation”. PhD thesis. Karlsruher Institut für Technologie KIT, 2016 (cit. on pp. 46, 63, 68–70, 78, 82, 92, 94, 96, 102–104, 108–110, 116, 121, 123, 128, 222, 231, 243, 263, 276–279, 281).

[Rie05] M. Riedmiller. “Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method”. In: *Machine Learning: ECML 2005*. Ed. by J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 317–328. ISBN: 978-3-540-31692-3. DOI: 10.1007/11564096_32. URL: papers3://publication/uuid/9B068F19-A3AD-43B0-BD3C-2219189F765D (cit. on pp. 188, 192).

[RL+06] B. J. Roset, M. Lazar, W. P. Heemels, and H. Nijmeijer. “A stabilizing output based nonlinear model predictive control scheme”. In: *Proceedings of the IEEE Conference on Decision and Control* (2006), pp. 4627–4632. ISSN: 01912216. DOI: 10.1109/cdc.2006.377814 (cit. on p. 141).

[RM+07] M. Riedmiller, M. Montemerlo, and H. Dahlkamp. “Learning to Drive a Real Car in 20 Minutes”. In: *2007 Frontiers in the Convergence of Bioscience and Information Technologies*. IEEE, 2007, pp. 645–650. ISBN: 978-0-7695-2999-8. DOI: 10.1109/FBIT.2007.37. URL: http://ieeexplore.ieee.org/document/4524181/ (cit. on p. 192).

[RM+17] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd Editio. Vol. 197. Nob Hill Publishing, LLC, 2017. ISBN: 978-0975937730 (cit. on p. 115).

[RM09] J. Rawlings and D. Mayne. *Model predictive control: Theory and design*. Nob Hill Pub, Llc, 2009. ISBN: 978-0975937709 (cit. on pp. 139, 143).

[RN94] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*. Volume 37. September. University of Cambridge, Department of Engineering Cambridge, England, 1994 (cit. on p. 186).

[Roh21] B. Rohrer. *End to End Machine Learning*. 2021. URL: https://end-to-end-machine-learning.teachable.com/ (visited on 04/09/2022) (cit. on p. 63).

[Ros03] J. Rossiter. *Model-Based Predictive Control: A Practical Approach*. CRC Press, 2003, p. 344. ISBN: 0203503961. URL: https://books.google.com/books?hl=en%7B%5C&%7Dlr=%7B%5C&%7Did=owznQTI-NqUC%7B%5C&%7Dpgis=1 (cit. on p. 140).

[RR+01] C. V. Rao, J. B. Rawlings, and J. H. Lee. “Constrained linear state estimation - A moving horizon approach”. In: *Automatica* 37.10 (2001), pp. 1619–1628. ISSN: 00051098. DOI: 10.1016/S0005-1098(01)00115-7 (cit. on pp. 228, 258).

[RR+09] M. R. Rajamani, J. B. Rawlings, and S. J. Qin. “Achieving state estimation equivalence for misassigned disturbances in offset-free model predictive control”. In: *AIChE Journal* 55.2 (2009), pp. 396–407. ISSN: 00011541. DOI: 10.1002/aic.11673. URL: https://onlinelibrary.wiley.com/doi/10.1002/aic.11673 (cit. on p. 143).

[RR16] M. Risbeck and J. Rawlings. *MPCTools: Nonlinear model predictive control tools for CasADi (Octave interface)*. 2016. URL: https://bitbucket.org/rawlings-group/octave-mpctools (cit. on p. 148).

[RS+99] J. Rossiter, J. Schuurmans, and B. Grinnell. "Optimal use of advance knowledge in the presence of constraints". In: *IFAC Proceedings Volumes* 32.2 (1999), pp. 1856–1861. ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)56315-6. URL: https://www.sciencedirect.com/science/article/pii/S1474667017563156 (cit. on p. 12).

[Run95] C. Runge. "Ueber die numerische Auflösung von Differentialgleichungen". In: *Mathematische Annalen* 46.2 (1895), pp. 167–178. ISSN: 00255831. DOI: 10.1007/BF01446807 (cit. on pp. 278, 281).

[Rus13] D. D. Ruscio. "Model Predictive Control with Integral Action: A simple MPC algorithm". In: *Modeling, Identification and Control* 34.3 (2013), pp. 119–129. ISSN: 03327353. DOI: 10.4173/mic.2013.3.2 (cit. on pp. 139, 143).

[RV09] J. Rossiter and G. Valencia-Palomo. "Feed Forward Design in MPC". In: *Control Conference (ECC), 2009 European* (2009), pp. 5–10. ISSN: ' (cit. on p. 140).

[RZ02] R. Rajamani and C. Zhu. "Semi-autonomous adaptive cruise control systems". In: *IEEE Transactions on Vehicular Technology* 51.5 (2002), pp. 1186–1192. ISSN: 00189545. DOI: 10.1109/TVT.2002.800617 (cit. on p. 50).

[SA+04] M. Sühling, M. Arigovindan, P. Hunziker, and M. Unser. "Multiresolution moment filters: Theory and applications". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 484–495. ISSN: 10577149. DOI: 10.1109/TIP.2003.819859 (cit. on pp. 62, 63).

[SA+17] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani. "Deep Reinforcement Learning Framework for Autonomous Driving". In: *Electronic Imaging* (2017), pp. 70–76. arXiv: arXiv:1704.02532v1. URL: https://arxiv.org/pdf/1704.02532.pdf (cit. on p. 192).

[SAE00] SAE International. *Hydrodynamic Drive Test Code*. 2000. URL: https://www.sae.org/standards/content/j643%7B%5C_%7D200005/ (cit. on p. 33).

[SAE16] SAE. *SAE Document J3016 - Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*. 2016. URL: https://www.sae.org/standards/content/j3016%7B%5C_%7D201609/ (cit. on p. 12).

[SAE18] SAE. *SAE Document J3016 - Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*. 2018. URL: https://www.sae.org/standards/content/j3016%7B%5C_%7D201806/ (cit. on p. 10).

[SB18] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. Second Edi. Vol. 9. 5. Cambridge, MA: The MIT Press, 2018, p. 1054. ISBN: 9780262039246. DOI: 10.1109/TNN.1998.712192. arXiv: 1603.02199. URL: http://incompleteideas.net/book/RLbook2018.pdf (cit. on pp. 175, 176, 184, 189).

[SC+13] S. Sommer, A. Camek, K. Becker, C. Buckl, A. Zirkler, L. Fiege, M. Armbruster, G. Spiegelberg, and A. Knoll. "RACE: A Centralized Platform Computer Based Architecture for Automotive Applications". In: *Vehicular Electronics Conference (VEC) and the International Electric Vehicle Conference (IEVC)*. IEEE, 2013. DOI: 10.1109/IEVC.2013.6681152 (cit. on p. 21).

[SC+19] E. Solano-Araque, G. Colin, G. M. Cloarec, A. Ketfi-Cherif, and Y. Chamaillard. “Determining vehicle acceleration from noisy non-uniformly sampled speed data for control purposes”. In: *IFAC-PapersOnLine* 52.5 (2019), pp. 66–71. ISSN: 24058963. DOI: 10.1016/j.ifacol.2019.09.011 (cit. on pp. 62, 64).

[SC13] S. Solyom and E. Coelingh. “Performance limitations in vehicle platoon control”. In: *IEEE Intelligent Transportation Systems Magazine* 5.4 (2013), pp. 112–120. ISSN: 19391390. DOI: 10.1109/MITS.2013.2272174 (cit. on p. 58).

[Sch11] R. Schafer. “What Is a Savitzky-Golay Filter? [Lecture Notes]”. In: *IEEE Signal Processing Magazine* 28.4 (July 2011), pp. 111–117. ISSN: 1053-5888. DOI: 10.1109/MSP.2011.941097. URL: http://ieeexplore.ieee.org/document/5888646/ (cit. on pp. 62–65).

[Sch17] M. Schaedler. “Development of a Method to Evaluate Longitudinal Motion Control for Automated Vehicles Entwicklung einer Methode zur Evaluierung von Längsdynamikregelungen für automatisierte Fahrzeuge Wissenschaftliche Arbeit zur Erlangung des Grades Master of”. Master Thesis. Technical University of Muncih, 2017 (cit. on pp. 24, 25).

[SG02] B. Stenlund and F. Gustafsson. “Avoiding windup in recursive parameter estimation”. In: *Preprints of reglermöte 2002* (2002), pp. 148–153 (cit. on pp. 92, 102, 103, 108, 261, 262, 268, 277–281).

[SG16] H. Shah and M. Gopal. “Model-free predictive control of nonlinear processes based on reinforcement learning”. In: *IFAC-PapersOnLine* (2016). ISSN: 24058963. DOI: 10.1016/j.ifacol.2016.03.034 (cit. on p. 194).

[SG64] A. Savitzky and M. J. E. Golay. “Smoothing and Differentiation of Data by Simplified Least Squares Procedures.” In: *Analytical Chemistry* 36.8 (1964), pp. 1627–1639. ISSN: 0003-2700. DOI: 10.1021/ac60214a047. URL: https://pubs.acs.org/doi/abs/10.1021/ac60214a047 (cit. on pp. 6, 61–64, 75, 78, 82, 109).

[SH+01] D. Swaroop, J. K. Hedrick, and S. B. Choi. “Direct adaptive longitudinal control of vehicle platoons”. In: *IEEE Transactions on Vehicular Technology* 50.1 (2001), pp. 150–161 (cit. on p. 51).

[SH+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antono-glou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M.-l. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (2016), pp. 484–489. ISSN: 14764687. DOI: 10.1038/nature16961. arXiv: 1610.00633. URL: http://dx.doi.org/10.1038/nature16961 (cit. on p. 175).

[SH+17a] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. “Evolution Strategies as a Scalable Alternative to Reinforcement Learning”. In: *arXiv preprint* (2017), pp. 1–13. arXiv: 1703.03864. URL: http://arxiv.org/abs/1703.03864 (cit. on p. 187).

[SH+17b] D. Silver, T. Hubert, J. Schrittwieser, I. Antono-glou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm”. In: *arXiv preprint* (2017), pp. 1–19. ISSN: 23289503. DOI: 10.1002/acn3.501. arXiv: 1712.01815. URL: http://arxiv.org/abs/1712.01815 (cit. on p. 175).

[Shl95] S. E. Shladover. “Review of the State of Development of Advanced Vehicle Control Systems (AVCS)”. In: *Vehicle System Dynamics* 24.24 (1995), pp. 6–7. DOI: 10.1080/00423119508969108 (cit. on pp. 17, 18, 22).

[Sil15] D. Silver. *UCS Course on Reinforcement Learning*. 2015. URL: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html (cit. on pp. 183, 188).

[Sim06] D. Simon. *Optimal State Estimation Kalman, H, and Nonlinear Approaches*. Wiley, 2006 (cit. on pp. 115, 231, 239, 243–247, 251, 253, 254, 256, 259, 261, 269).

[Sim10] D. Simon. “Kalman filtering with state constraints: a survey of linear and nonlinear algorithms”. In: *IET Control Theory & Applications* 4.8 (2010), p. 1303. ISSN: 17518644. DOI: 10.1049/iet-cta.2009.0032 (cit. on p. 258).

[SJ+00] Se-Young Oh, Jeong-Hoon Lee, and Doo-Hyun Choi. “A new reinforcement learning vehicle control architecture for vision-based road following”. In: *IEEE Transactions on Vehicular Technology* 49.3 (2000), pp. 997–1005. ISSN: 00189545. DOI: 10.1109/25.845116. URL: http://ieeexplore.ieee.org/document/845116/ (cit. on p. 192).

[SJ11] D. Sui and T. A. Johansen. “Moving horizon observer with regularisation for detectable systems without persistence of excitation”. In: *International Journal of Control* 84.6 (2011), pp. 1041–1054. ISSN: 00207179. DOI: 10.1080/00207179.2011.589081 (cit. on p. 228).

[SK+01] D. Swaroop, J. Karl Hedrick, and S. B. Choi. “Direct adaptive longitudinal control of vehicle platoons”. In: *IEEE Transactions on Vehicular Technology* 50.1 (2001), pp. 150–161. ISSN: 00189545. DOI: 10.1109/25.917908 (cit. on p. 22).

[SK+10] Y. Sakai, M. Kanai, and M. Yamakita. “Torque Demand Control by Nonlinear MPC for Speed Control of Vehicles with Variable Valve Lift Engine”. In: *Proceedings of the IEEE Multi-Conference on Systems and Control*. 2010, pp. 494–499. ISBN: 9783902661722. DOI: 00164 (cit. on p. 56).

[SK99] A. G. Stefanopoulou and I. Kolmanovsky. “Analysis and control of transient torque response in engines with internal exhaust gas recirculation”. In: *IEEE Transactions on Control Systems Technology* 7.5 (1999), pp. 555–566. DOI: 10.1109/87.784419 (cit. on p. 35).

[SL+14] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. “Deterministic Policy Gradient Algorithms”. In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (2014), pp. 387–395. ISSN: 1938-7228 (cit. on pp. 189, 202).

[SL+15] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. “Trust Region Policy Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015, pp. 1889–1897. DOI: 10.1063/1.4927398. arXiv: 1502.05477. URL: http://arxiv.org/abs/1502.05477 (cit. on pp. 188, 202).

[SL03] S. W. Sung and J. H. Lee. “Pseudo-random binary sequence design for finite impulse response identification”. In: *Control Engineering Practice* 11.8 (2003), pp. 935–947. ISSN: 09670661. DOI: 10.1016/S0967-0661(03)00035-2 (cit. on p. 200).

[SL20] H. S. Shin and H. I. Lee. “A New Exponential Forgetting Algorithm for Recursive Least-Squares Parameter Estimation”. In: *arXiv* April (2020). ISSN: 23318422. arXiv: 2004.03910 (cit. on pp. 103, 272).

[SM+13] H. Stahle, L. Mercep, A. Knoll, and G. Spiegelberg. "Towards the Deployment of a Centralized ICT Architecture in the Automotive Domain". In: *2nd Mediterranean Conference on Embedded Computing (MECO)* (2013), pp. 66–69. DOI: 10.1109/MECO.2013.6601320 (cit. on p. 167).

[SM+15] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. "High - Dimensional Continuous Control Using Generalized Advantage Estimation". In: (2015). arXiv: 1506.02438. URL: http://arxiv.org/abs/1506.02438 (cit. on p. 189).

[SM+17] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune. "Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning". In: *arXiv preprint* (2017). arXiv: 1712.06567. URL: http://arxiv.org/abs/1712.06567 (cit. on p. 187).

[SN+14] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Hoogendoorn. *Using Cooperative Adaptive Cruise Control (CACC) to Form High-Performance Vehicle Streams*. Tech. rep. June. 2014, p. 54. URL: http://escholarship.org/uc/item/3m89p611 (cit. on pp. 11, 59).

[SO+11] P. Shakouri, A. Ordys, D. S. Laila, and M. Askari. "Adaptive Cruise Control System: Comparing Gain-scheduling PI and LQ Controllers". In: *18th IFAC World Congress*. Vol. 18. PART 1. Milano, 2011, pp. 12964–12969. ISBN: 9783902661937. DOI: 10.3182/20110828-6-IT-1002.02250 (cit. on p. 55).

[SO+12] P. Shakouri, A. Ordys, and M. R. Askari. "Adaptive cruise control with stop&go function using the state-dependent nonlinear model predictive control approach". In: *ISA Transactions* 51.5 (2012), pp. 622–631. ISSN: 00190578. DOI: 10.1016/j.isatra.2012.05.001. URL: http://dx.doi.org/10.1016/j.isatra.2012.05.001 (cit. on p. 52).

[Söd07] T. Söderström. "Errors-in-variables methods in system identification". In: *Automatica* 43.6 (2007), pp. 939–958. ISSN: 00051098. DOI: 10.1016/j.automatica.2006.11.025 (cit. on pp. 102, 262).

[SP+04] P. Seiler, A. Pant, and K. Hedrick. "Disturbance propagation in vehicle strings". In: *IEEE Transactions on Automatic Control* 49.10 (2004), pp. 1835–1841. ISSN: 00189286. DOI: 10.1109/TAC.2004.835586 (cit. on pp. 22, 23).

[SS+21] K. Sridhar, O. Sokolsky, I. Lee, and J. Weimer. "Improving Neural Network Robustness via Persistency of Excitation". In: (2021). arXiv: 2106.02078. URL: http://arxiv.org/abs/2106.02078 (cit. on p. 225).

[SS+62] G. L. Smith, S. F. Schmidt, and L. A. McGee. *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration, 1962 (cit. on p. 243).

[SS15] M. Sivak and B. Schoettle. "Motion Sickness in Self-Driving Vehicles". In: April (2015), pp. 1–15 (cit. on pp. 16, 24).

[ST+16] A. Sinigaglia, K. Tagesson, P. Falcone, and B. Jacobson. "Coordination of motion actuators in heavy vehicles using Model Predictive Control Allocation". In: *IEEE Intelligent Vehicles Symposium, Proceedings* 2016-Augus.Iv (2016), pp. 590–596. DOI: 10.1109/IVS.2016.7535447 (cit. on p. 150).

[Stu08] Student. "The Probable Error of a Mean". In: *Biometrika* 6.1 (1908), p. 1. ISSN: 00063444. DOI: 10.2307/2331554 (cit. on pp. 103, 266).

[Sut88] R. S. Sutton. “Learning to predict by the methods of temporal differences”. In: *Machine Learning* 3.1 (1988), pp. 9–44. ISSN: 0885-6125. DOI: 10.1007/BF00115009. URL: http://link.springer.com/10.1007/BF00115009 (cit. on pp. 181, 193).

[Sut91] R. S. Sutton. “Dyna, an integrated architecture for learning, planning, and reacting”. In: *ACM SIGART Bulletin* 2.4 (1991), pp. 160–163. ISSN: 01635719. DOI: 10.1145/122344.122377 (cit. on p. 187).

[SW+15] R. Schmied, H. Waschl, R. Quirynen, and M. Diehl. “Nonlinear MPC for Emission Efficient Cooperative Adaptive Cruise Control”. In: *IFAC - PapersOnLine* 48.2014 (2015), pp. 160–165. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.11.277 (cit. on p. 52).

[SW+17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal Policy Optimization Algorithms”. In: (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347 (cit. on p. 188).

[TC+97] W. D. Timmons, H. J. Chizeck, F. Casas, V. Chankong, and P. G. Katona. “Parameter-Constrained Adaptive Control”. In: *Industrial & Engineering Chemistry Research* 36.11 (Nov. 1997), pp. 4894–4905. ISSN: 0888-5885. DOI: 10.1021/ie9606597. URL: https://pubs.acs.org/doi/10.1021/ie9606597 (cit. on p. 258).

[Tee48] R. R. Teetor. *Speed Control Device for Resisting Operation of the Accelerator*. 1948. URL: https://patents.google.com/patent/US2519859 (cit. on p. 17).

[Tes15] Tesla. *Ihr Autopilot ist da | Tesla Deutschland*. 2015. URL: https://www.tesla.com/de%7B%5C_%7DDE/blog/your-autopilot-has-arrived?redirect=no (visited on 08/30/2018) (cit. on p. 20).

[TH+10] C. C. Tsai, S. M. Hsieh, and C. T. Chen. “Fuzzy longitudinal controller design and experimentation for adaptive cruise control and stop&Go”. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 59.2 (2010), pp. 167–189. ISSN: 09210296. DOI: 10.1007/s10846-010-9393-z (cit. on p. 51).

[The04] The Automotor Journal. *The Automotor Journal*. 1904. URL: https://archive.org/details/TheAutomotorJournalFirstHalf1904 (cit. on pp. 16, 17).

[Thr02] S. Thrun. “Probabilistic robotics”. In: *Communications of the ACM* 45.3 (2002), pp. 52–57. ISSN: 00010782. DOI: 10.1145/504729.504754 (cit. on pp. 231, 243).

[TK+01] S. Tsugawa, S. Kato, K. Tokuda, T. Matsui, and H. Fujii. “A cooperative driving system with automated vehicles and inter-vehicle communications in Demo 2000”. In: *IEEE Intelligent Transportation Systems Conference (ITSC)*. 2001. ISBN: 0-7803-7194-1. DOI: 10.1109/ITSC.2001.948783 (cit. on p. 18).

[TK+17] V. Turri, Y. Kim, J. Guanetti, K. H. Johansson, and F. Borrelli. “A model predictive controller for non-cooperative eco-platooning”. In: *Proceedings of the American Control Conference* (2017), pp. 2309–2314. ISSN: 07431619. DOI: 10.23919/ACC.2017.7963297 (cit. on p. 52).

[TM+06] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. “Stanley: The robot that won the DARPA Grand Challenge”. In: *Journal of Field Robotics* 23.9 (2006), pp. 661–

692. ISSN: 15564959. DOI: 10.1002/rob.20147. URL: http://doi.wiley.com/10.1002/rob.20147 (cit. on pp. 19, 54).

[TM+18] D. Tavernini, M. Metzler, P. Gruber, and A. Sorniotti. “Explicit Nonlinear Model Predictive Control for Electric Vehicle Traction Control”. In: *IEEE Transactions on Control Systems Technology* PP (2018), pp. 1–14. ISSN: 1558-0865. DOI: 10.1109/TCST.2018.2837097 (cit. on p. 225).

[Tor17] G. Torrisi. “Low-Complexity Numerical Methods for Nonlinear Model Predictive Control”. PhD thesis. ETH Zürich, 2017. URL: https://doi.org/10.3929/ethz-a-010025751 (cit. on p. 208).

[TS+16] R. Tachet, P. Santi, and S. Sobolevsky. “Revisiting street intersections using slot-based systems”. In: *PLoS one* 11.3 (2016). DOI: 10.1371/journal.pone.0149607 (cit. on p. 23).

[TS85] T. Takagi and M. Sugeno. “Fuzzy Identification of Systems and Its Applications to Modeling and Control”. In: *IEEE Transactions on Systems, Man and Cybernetics* SMC-15.1 (1985), pp. 116–132. ISSN: 21682909. DOI: 10.1109/TSMC.1985.6313399 (cit. on p. 55).

[Tur66] L. R. Turner. *Inverse of the Vandermonde Matrix with applications*. Ohio, 1966 (cit. on p. 65).

[TY+79] S. Tsugawa, T. Yatabe, T. Hirose, and S. Matsumoto. “An automobile with artificial intelligence”. In: *Proceedings of the 6th international Joint Conference on Artificial Intelligence (IJCAI)*. Tokyo: IJCAI, 1979, pp. 893–895. ISBN: 0934613478. URL: https://dl.acm.org/citation.cfm?id=1623117 (cit. on p. 18).

[UA+08] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. “Autonomous Driving in Urban Environments: Boss and the Urban Challenge”. In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466 (cit. on pp. 54, 58).

[UNE19] UNECE. *WLTP Test cycle data*. 2019. URL: https://unece.org/DAM/trans/doc/2012/wp29grpe/WLTP-DHC-12-07e.xls (cit. on pp. 112, 158).

[UO30] G. E. Uhlenbeck and L. S. Ornstein. “On the Theory of the Brownian Motion”. In: *Physical Review* 36.5 (1930), pp. 823–841. ISSN: 0031899X. DOI: 10.1103/PhysRev.36.823 (cit. on p. 206).

[VD+03] A. Vahidi, M. Druzhinina, A. G. Stefanopoulou, and H. Peng. “Simultaneous Mass and Time-Varying Grade Estimation for Heavy-Duty Vehicles”. In: *Proceedings of the American Control Conference (ACC)*. Denver, Colorado, 2003. ISBN: 0-7918-3713-0. DOI: 10.1115/IMECE2003-43848. URL: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1591351 (cit. on p. 91).

[VE+19] A. F. Villaverde, N. D. Evans, M. J. Chappell, and J. R. Banga. “Input-dependent structural identifiability of nonlinear systems”. In: *IEEE Control Systems Letters* 3.2 (2019), pp. 272–277. ISSN: 24751456. DOI: 10.1109/LCSYS.2018.2868608 (cit. on pp. 98, 270–272).

[VF+21] 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”. In: *Mathematical Programming Computation* (2021). ISSN: 1867-2949. DOI: 10.1007/s12532-021-00208-8 (cit. on p. 148).

[VF19] B. Vatankhah and M. Farrokhi. “Nonlinear Adaptive Model Predictive Control of Constrained Systems with Offset-Free Tracking Behavior”. In: *Asian Journal of Control* 21.5 (2019), pp. 2232–2244. ISSN: 19346093. DOI: 10.1002/asjc.1655 (cit. on pp. 141, 144).

[VG+16] H. Van Hasselt, A. Guez, and D. Silver. “Deep Reinforcement Learning with Double Q-Learning”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016. URL: www.aaai.org (cit. on p. 187).

[VK+82] H. Vogel, R. Kohlhaas, and R. von Baumgarten. “Dependence of Motion Sickness in Automobiles on the Direction of Linear Acceleration”. In: *European Journal of Applied Physiology* (1982) (cit. on p. 24).

[VM+05] M. Vogels, E. Martini, K. Gschweitl, P. Mathis, H. Altenstrasser, and M. Buechel. “Dynamic Powertrain Calibration: Using Transient DoE and Modelling Techniques”. In: *Design of Experiments (DoE): In: Engine Development II, Haus der Technik Fachbuch* 49 (2005) (cit. on p. 199).

[VS+03] A. Vahidi, A. G. Stefanopoulou, and H. Peng. “Experiments for Online Estimation of Heavy Vehicle’s Mass and Time-Varying Road Grade”. In: *ASME 2003 International Mechanical Engineering Congress and Exposition* (2003), pp. 451–458 (cit. on p. 91).

[VS+05] A. Vahidi, A. Stefanopoulou, and H. Peng. “Recursive least squares with forgetting for online estimation of vehicle mass and road grade: theory and experiments”. In: *Vehicle System Dynamics* 43.1 (2005), pp. 31–55. ISSN: 0042-3114. DOI: 10.1080/00423110412331290446. URL: http://www.tandfonline.com/doi/abs/10.1080/00423110412331290446 (cit. on pp. 91, 95).

[VT+19] A. Villaverde, N. Tsiantis, and J. Banga. “Full observability and estimation of unknown inputs, states, and parameters of nonlinear biological models”. In: *Journal of the Royal Society Interface* 16.156 (2019) (cit. on p. 272).

[WA+04] J. Wang, L. Alexander, and R. Rajamani. “Friction Estimation on Highway Vehicles using Longitudinal Measurements”. In: *Journal of Dynamic Systems, Measurement, and Control* 126.2 (2004), pp. 265–275 (cit. on p. 38).

[Wal82] E. Walter. *IdentifiabiIity of State Space Models*. Springer, 1982. ISBN: 978-3-642-61823-9 (cit. on p. 270).

[Wat89] C. J. C. H. Watkins. “Learning from delayed rewards”. PhD thesis. Kings College, 1989 (cit. on p. 181).

[WB+04] T. A. Wenzel, K. J. Burnham, M. V. Blundell, and R. A. Williams. “Approach to vehicle state and parameter estimation using extended Kalman filtering”. In: *Proceedings of the International Symposium on Advanced Vehicle Control (AVEC)*. 2004 (cit. on p. 94).

[WB+06] T. A. Wenzel, K. J. Burnham, M. V. Blundell, and R. A. Williams. “Dual extended Kalman filter for vehicle state and parameter estimation”. In: *Vehicle System Dynamics* 44.2 (Feb. 2006), pp. 153–171. ISSN: 0042-3114. DOI: 10.1080/00423110500385949. URL: http://www.tandfonline.com/doi/abs/10.1080/00423110500385949 (cit. on pp. 94, 95).

[WB06] A. Wächter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming* 106.1 (2006), pp. 25–57. ISSN: 00255610. DOI: 10.1007/s10107-004-0559-y (cit. on pp. 148, 159, 165, 226).

[WC+18] P. Wang, C.-Y. Chan, and A. de La Fortelle. “A Reinforcement Learning Based Approach for Automated Lane Change Maneuvers”. In: (Apr. 2018). arXiv: 1804.07871. URL: http://arxiv.org/abs/1804.07871 (cit. on p. 192).

[WD+00] G. R. Widmann, M. K. Daniels, L. Hamilton, L. Humm, B. Riley, J. K. Schiffmann, D. E. Schnelker, and W. H. Wishon. “Comparison of Lidar-Based and Radar-Based Adaptive Cruise Control Systems”. In: *SAE Technical Paper Series* 1.345 (2000). DOI: 10.4271/2000-01-0345 (cit. on p. 50).

[Weba] Weblink. *Scania Lowliner R 490 - ES-GE Nutzfahrzeuge GmbH*. URL: https://www.es-ge.de/scania-r490-lowliner/ (visited on 07/15/2021) (cit. on p. 38).

[Webb] Weblink. *Technische Daten Ford Mondeo 1.6 TDCi ECOnetic Turnier (85 kW / 116 PS), 6-Gang Handschaltung (seit Februar 2015) - AutoKlicker*. URL: https://www.autoklicker.de/autokatalog/ford-mondeo-16-tdci-econetic-technische-daten13213.html (visited on 07/15/2021) (cit. on p. 38).

[Webc] Weblink. *Top-Liste maximale Zuladung: Autos, die richtig wegpacken können | AUTO MOTOR UND SPORT*. URL: https://www.auto-motor-und-sport.de/reise/top-liste-maximale-zuladung-autos-die-richtig-wegpacken-koennen/ (visited on 07/15/2021) (cit. on p. 38).

[Wer10] M. Werling. “Ein neues Konzept für die Trajektoriengenerierung und -stabilisierung in zeitkritischen Verkehrsszenarien”. PhD thesis. Karlsruhe Institut für Technologie (KIT), 2010. ISBN: 9783866446311 (cit. on pp. 56, 57, 79).

[Wer21] H. Werner. “A Velocity Algorithm for Nonlinear Model Predictive Control”. In: *IEEE Transactions on Control Systems Technology* 29.3 (2021), pp. 1310–1315. DOI: 10.1109/TCST.2020.2979386 (cit. on p. 143).

[WH19a] F. Walz and S. Hohmann. “Model predictive longitudinal motion control for low velocities on known road profiles”. In: *Vehicle System Dynamics* 0.0 (2019), pp. 1–19. ISSN: 0042-3114. DOI: 10.1080/00423114.2019.1618880. URL: https://doi.org/00423114.2019.1618880 (cit. on p. 56).

[WH19b] F. Walz and S. Hohmann. “On model-based longitudinal feed-forward motion control for low velocities on known road profiles”. In: *Vehicle System Dynamics* 57.8 (2019), pp. 1126–1142. ISSN: 17445159. DOI: 10.1080/00423114.2018.1498111. URL: https://doi.org/10.1080/00423114.2018.1498111 (cit. on p. 56).

[Wik18] Wikipedia. *Cruise Control (wikipedia)*. 2018. URL: https://en.wikipedia.org/wiki/Cruise%7B%5C_%7Dcontrol (visited on 05/30/2018) (cit. on p. 17).

[Wik20] Wikipedia. *Automobile Drag Coefficient*. 2020. URL: https://en.wikipedia.org/wiki/Automobile%7B%5C_%7Ddrag%7B%5C_%7Dcoefficient (visited on 07/14/2020) (cit. on p. 39).

[Wil92] R. J. Williams. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Machine Learning* 8.3 (1992), pp. 229–256. ISSN: 15730565. DOI: 10.1023/A:1022672621406 (cit. on pp. 186, 202).

[Wis68a] D. A. Wisner. *Speed control for automotive vehicles*. 1968. URL: https://patents.google.com/patent/US3511329A/en (cit. on p. 17).

[Wis68b] D. A. Wisner. *Speed control for motor vehicles*. 1968. URL: https://patents.google.com/patent/US3570622A/en (cit. on p. 17).

[WJ+22] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. “Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems”. In: *ACM Computing Surveys* 1.1 (Mar. 2022), pp. 1–35. ISSN: 0360-0300. DOI: 10.1145/3514228. URL: https://dl.acm.org/doi/10.1145/3514228 (cit. on p. 62).

[WK03] Z. Wan and M. V. Kothare. “Efficient scheduled stabilizing model predictive control for constrained nonlinear systems”. In: *International Journal of Robust and Nonlinear Control* 346.August 2002 (2003), pp. 331–346 (cit. on p. 142).

[WK05] V. Winstead and I. Kolmanovsky. “Estimation of road grade and vehicle mass via model predictive control”. In: *Proceedings of the IEEE Conference on Control Applications (CCA)* (2005), pp. 1588–1593. ISSN: 1085-1992. DOI: 10.1109/CCA.2005.1507359 (cit. on pp. 91, 92, 94, 95).

[WL+12] D. Wang, D. Liu, and Q. Wei. “Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach”. In: *Neurocomputing* 78.1 (2012), pp. 14–22. ISSN: 09252312. DOI: 10.1016/j.neucom.2011.03.058. URL: http://dx.doi.org/10.1016/j.neucom.2011.03.058 (cit. on p. 223).

[WN97] E. A. Wan and A. T. Nelson. “Dual Kalman filtering methods for nonlinear prediction, smoothing, and estimation”. In: *Advances in Neural Information Processing Systems* 1 (1997), pp. 793–799. ISSN: 10495258 (cit. on p. 259).

[WO12] M. Wiering and M. van. Otterlo. *Reinforcement learning : state-of-the-art*. Springer, 2012, p. 638. ISBN: 9783642276446 (cit. on pp. 176, 178, 181, 182, 189, 191, 197, 200).

[Won08] J. Wong. *Theory of ground vehicles*. John Wiley & Sons, 2008. ISBN: 978-0470170380. URL: https://books.google.de/books?isbn=0470170387 (cit. on p. 38).

[Won11] E. Wong. “Active-Set Methods for Quadratic Programming”. PhD thesis. University of California, 2011 (cit. on p. 146).

[WS+08] J. Wang, J. Steiber, and B. Surampudi. “Autonomous ground vehicle control system for high-speed and safe operation”. In: *Proceedings of the American Control Conference* (2008), pp. 218–223. ISSN: 07431619. DOI: 10.1109/ACC.2008.4586494 (cit. on p. 88).

[WS+13] J. Wang, Z. Sun, X. Xu, D. Liu, J. Song, and Y. Fang. “Adaptive Speed Tracking Control for Autonomous Land Vehicles in All-Terrain Navigation: An Experimental Study”. In: *Journal of Field Robotics* 30.1 (2013), pp. 102–128 (cit. on pp. 53, 55).

[WS+16] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Frcitas. “Dueling Network Architectures for Deep Reinforcement Learning”. In: *33rd International Conference on Machine Learning, ICML 2016*. Vol. 4. International Machine Learning Society (IMLS), 2016, pp. 2939–2947. ISBN: 9781510829008. arXiv: 1511.06581 (cit. on p. 187).

[WS+18a] F. Walz, T. Schucht, J. Reger, and S. Hohmann. “Shaft Torque and Backlash Estimation for Longitudinal Motion Control of All-Wheel-Drive Vehicles”. In: *2018 IEEE Conference on Control Technology and Applications (CCTA)* (2018), pp. 1434–1440 (cit. on p. 40).

[WS+18b] F. Walz, T. Schucht, J. Reger, and S. Hohmann. “Shaft Torque and Backlash Estimation for Longitudinal Motion Control of All-Wheel-Drive Vehicles”. In: *2018 IEEE Conference on Control Technology and Applications, CCTA 2018* (2018), pp. 1434–1440. DOI: 10.1109/CCTA.2018.8511096 (cit. on p. 56).

[WT13] C. Wuthishuwong and A. Traechtler. “Vehicle to infrastructure based safe trajectory planning for autonomous intersection management”. In: *13th International Conference on ITS Telecommunications (ITST)* (2013), pp. 175–180. ISSN: 1687-1499. DOI: 10.1109/ITST.2013.6685541 (cit. on p. 23).

[WW+17] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. “Information theoretic MPC for model-based reinforcement learning”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017, pp. 1714–1721. ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989202. URL: http://ieeexplore.ieee.org/document/7989202/ (cit. on p. 192).

[WX+14] J. Wang, X. Xu, D. Liu, Z. Sun, and Q. Chen. “Self-learning Cruise Control using Kernel-based Least Squares Policy Iteration”. In: *IEEE Transactions on Control Systems Technology* 22.3 (2014), pp. 1078–1087. ISSN: 10636536. DOI: 10.1109/TCST.2013.2271276 (cit. on p. 193).

[WZ+15] B. Wang, D. Zhao, C. Li, and Y. Dai. “Design and implementation of an adaptive cruise control system based on supervised actor-critic learning”. In: *2015 5th International Conference on Information Science and Technology, ICIST 2015*. Institute of Electrical and Electronics Engineers Inc., Oct. 2015, pp. 243–248. ISBN: 9781479974894. DOI: 10.1109/ICIST.2015.7288976 (cit. on p. 193).

[WZ+18a] B. Wang, D. Zhao, and J. Cheng. “Adaptive cruise control via adaptive dynamic programming with experience replay”. In: *Soft Computing* 23.12 (2018), pp. 1–14. ISSN: 14337479. DOI: 10.1007/s00500-018-3063-7. URL: https://doi.org/10.1007/s00500-018-3063-7 (cit. on pp. 53, 223).

[WZ+18b] S. Wei, Y. Zou, T. Zhang, X. Zhang, and W. Wang. “Design and Experimental Validation of a Cooperative Adaptive Cruise Control System Based on Supervised Reinforcement Learning”. In: *Applied Sciences* 8.7 (2018), pp. 2076–3417. ISSN: 2076-3417. DOI: 10.3390/app8071014. URL: http://www.mdpi.com/2076-3417/8/7/1014 (cit. on p. 193).

[XD+08] L. Xiao, S. Darbha, and F. Gao. “Stability of string of adaptive cruise control vehicles with parasitic delays and lags”. In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* (2008), pp. 1101–1106. DOI: 10.1109/ITSC.2008.4732532 (cit. on pp. 25, 35).

[XG10] L. Xiao and F. Gao. “A comprehensive review of the development of adaptive cruise control systems”. In: *Vehicle System Dynamics* 48.10 (2010), pp. 1167–1192. ISSN: 0042-3114. DOI: 10.1080/00423110903365910. URL: http://www.tandfonline.com/doi/abs/10.1080/00423110903365910 (cit. on pp. 18, 38, 50).

[XG11] L. Xiao and F. Gao. “Practical String Stability of Platoon of Adaptive Cruise Control Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 1184–1194. ISSN: 1524-9050. DOI: 10.1109/TITS.2011.2143407. URL: http://ieeexplore.ieee.org/document/5782989/ (cit. on pp. 11, 58).

[XI94] Z. Xu and P. Ioannou. “Adaptive Throttle Control for Speed Tracking”. In: *Vehicle System Dynamics* 23.1 (1994), pp. 293–306. ISSN: 17445159. DOI: 10.1080/00423119408969063 (cit. on p. 56).

[XT+18] Z. Xu, C. Tang, and M. Tomizuka. “Zero-shot Deep Reinforcement Learning Driving Policy Transfer for Autonomous Vehicles based on Robust Control”. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. 2018, pp. 2865–2871. ISBN: 9781728103228 (cit. on p. 192).

[XX+22] W. Xin, E. Xu, W. Zheng, H. Feng, and J. Qin. “Optimal energy management of fuel cell hybrid electric vehicle based on model predictive control and on-line mass estimation”. In: *Energy Reports* 8 (2022), pp. 4964–4974. ISSN: 23524847. DOI: 10.1016/j.egyr.2022.03.194. URL: https://doi.org/10.1016/j.egyr.2022.03.194 (cit. on pp. 52, 93).

[YH+22] Z. Yu, X. Hou, B. Leng, and Y. Huang. “Mass estimation method for intelligent vehicles based on fusion of machine learning and vehicle dynamic model”. In: *Autonomous Intelligent Systems* 2.1 (2022). DOI: 10.1007/s43684-022-00020-8. URL: http://dx.doi.org/10.1007/s43684-022-00020-8 (cit. on p. 93).

[YS22] H. Yuan and X. Song. “A Modified EKF for Vehicle State Estimation With Partial Missing Measurements”. In: *IEEE Signal Processing Letters* 29 (2022), pp. 1594–1598 (cit. on p. 93).

[ZC+00] Y. Zou, S. C. Chan, and T. S. Ng. “Recursive Least M-estimate (RLM) adaptive filter for robust filtering in impulse noise”. In: *IEEE Signal Processing Letters* 7.11 (2000), pp. 324–326. ISSN: 10709908. DOI: 10.1109/97.873571 (cit. on pp. 267, 268).

[ZC+16a] H. Zhang, A. Chakrabarty, R. Ayoub, G. T. Buzzard, and S. Sundaram. “Sampling-based explicit nonlinear model predictive control for output tracking”. In: *2016 IEEE 55th Conference on Decision and Control, CDC 2016* Cdc (2016), pp. 4722–4727. DOI: 10.1109/CDC.2016.7798989 (cit. on p. 225).

[ZC+16b] M. Zhu, H. Chen, and G. Xiong. “A Model Predictive Speed Tracking Control Approach for Autonomous Ground Vehicles”. In: *Mechanical Systems and Signal Processing* (2016), pp. 1–15. ISSN: 08883270. DOI: 10.1016/j.ymssp.2016.03.003 (cit. on p. 52).

[ZD+13] J. Ziegler, T. Dang, U. Franke, H. Lategahn, P. Bender, M. Schreiber, T. Strauss, N. Appenrodt, C. G. Keller, E. Kaus, C. Stiller, and R. G. Herrtwich. “Making Bertha Drive - An Autonomous Journey on a Historic Route”. In: *IEEE Intelligent Transportation Systems Magazine* 11.4 (2013), pp. 1–10. ISSN: 1939-1390 (cit. on pp. 19, 54).

[ZD+17a] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari. “FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs”. In: *International Journal of Control* 93.1 (2017), pp. 13–29. ISSN: 13665820. DOI: 10.1080/00207179.2017.1316017. URL: https://www.tandfonline.com/doi/abs/10.1080/00207179.2017.1316017 (cit. on pp. 148, 165, 226).

[ZD+17b] Q. Zhu, B. Dai, Z. Huang, Z. Sun, and D. Liu. “An Adaptive Longitudinal Control Method for Autonomous Follow Driving Based on Neural Dynamic Programming and Internal Model Structure”. In: *International Journal of Advanced Robotic Systems* 14.6 (2017), pp. 1–13. ISSN: 17298814. DOI: 10.1177/1729881417740711 (cit. on pp. 53, 193).

[ZD12] X. Zhongpu and Z. Dongbin. “Hybrid Feedback Control of Vehicle Longitudinal Acceleration”. In: *Chinese Control Conference (CCC)* (2012), pp. 7292–7297. ISSN: 19341768 (cit. on pp. 53, 55).

[ZF+13] M. Zanon, J. V. Frasch, and M. Diehl. “Nonlinear Moving Horizon Estimation for combined state and friction coefficient estimation in autonomous driving”. In: *2013 European Control Conference, ECC 2013* (2013), pp. 4130–4135. DOI: 10.23919/ecc.2013.6669832 (cit. on p. 228).

[ZH+14] D. Zhao, Z. Hu, Z. Xia, C. Alippi, Y. Zhu, and D. Wang. “Full-range Adaptive Cruise Control Based on Supervised Adaptive Dynamic Programming”. In: *Neurocomputing* 125.2014 (2014), pp. 57–67. ISSN: 09252312. DOI: 10.1016/j.neucom.2012.09.034 (cit. on p. 53).

[ZW+13] D. Zhao, B. Wang, and D. Liu. “A supervised Actor-Critic approach for adaptive cruise control”. In: *Soft Computing* 17.11 (2013), pp. 2089–2099. ISSN: 14327643. DOI: 10.1007/s00500-013-1110-y (cit. on p. 53).

[ZZ+13] H.-G. Zhang, X. Zhang, Y.-H. Luo, and J. Yang. “An Overview of Research on Adaptive Dynamic Programming”. In: *Acta Automatica Sinica* 39.4 (Apr. 2013), pp. 303–311. ISSN: 18741029. DOI: 10.1016/S1874-1029(13)60031-2. URL: https://linkinghub.elsevier.com/retrieve/pii/S1874102913600312 (cit. on pp. 220, 223).