Driver Interaction:
Mathematical Modeling and Numerical Methods

Von der Fakultät für Mathematik und Naturwissenschaften der RWTH Aachen University
zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften
genehmigte Dissertation

vorgelegt von

Amira El Amouri, M. Sc.

aus

Frechen

Berichter: Univ-Prof. Dr. rer. nat. Michael Matthias Herty
Univ-Prof. Dr. Simone Göttlich

Tag der mündlichen Prüfung: 23. Oktober 2025

In memory of my beloved father, Ali. He would have been so proud.

Abstract

The automotive industry is witnessing significant advancements in autonomous and assisted driving functions. However, a major challenge remains in ensuring these systems are accepted by drivers and effectively integrated into their driving routines. Hence, the necessity of a comprehensive framework becomes crucial to analyze and enhance the interaction between drivers and assisted driving functions.

In this thesis, we design and implement a driver interaction based framework for shared lateral driving. The research is divided into three main objectives: developing a methodology to capture the dynamics of shared control, constructing a driver interaction classifier, and modeling concepts for individualized driver support.

We introduce a novel driver-steering interaction model to analyze the interplay between driver and assistance system torques. The model allows a comprehensive analysis of the driver torque, the assistance torque, and their combined effect on the vehicle's steering behavior. To accurately represent these interactions, the driver-steering interaction model requires the formulation of a quadratic program. We apply the Varying Coefficient (VC) method to effectively formulate this QP.

We develop a driver interaction classifier based on a designed real-driving experimental framework. We identify and suggest suitable classification features: conflict and passivity, path consistency, adaptation and individual path pattern features. We categorize the identified driver interaction strategies into five distinct classes: adaptation, persistence, selective persistence, nonintervention and uncertainty. The classification is validated through a comparison with subjective expert assessments. Additionally, we apply Dynamic Mode Decomposition (DMD) to analyze the underlying dynamics for each class.

We present concepts for adapting the system behavior in real-time to account for the driver interaction strategies. We design a Model Predictive Control (MPC) system and conduct a performance analysis to ensure it aligns with desired system behavior for each class. We apply online active-set methods and interior point methods to solve the optimization problems in real-time. The framework is implemented with a focus on real-world applicability. Additionally, we suggest further concepts for adapting the system behavior based on independent components of the framework.

Zusammenfassung

Die Automobilindustrie erlebt erhebliche Fortschritte bei autonomen und assistierten Fahrfunktionen. Eine große Herausforderung bleibt jedoch, sicherzustellen, dass diese Systeme von den Fahrern akzeptiert und effektiv in ihre Fahrgewohnheiten integriert werden. Daher wäre ein umfassendes Framework notwendig, um die Interaktion zwischen Fahrern und assistierten Fahrfunktionen zu analysieren und zu optimieren.

Diese Arbeit präsentiert den Entwurf eines fahrerinteraktionsbasierten Frameworks für die geteilte Querführung eines Fahrzeugs. Drei wesentliche Forschungsziele werden betrachtet: die Entwicklung einer Methodik zur Erfassung der Dynamik der geteilten Querführung, die Konstruktion eines Fahrerinteraktionsklassifikators und die Modellierung von Konzepten für eine individualisierte Fahrerunterstützung.

Wir stellen ein mathematisches Fahrer-Lenkrad-Interaktionsmodell vor. Das Modell ermöglicht eine umfassende Analyse des Zusammenspiels zwischen Fahrer- und Assistenzsystem am Lenkrad. Dieses Modell erfordert die Formulierung eines quadratischen Programms. Wir wenden die Varying-Coefficient (VC) Methode an, um dieses quadratische Programm effektiv zu formulieren.

Wir entwickeln einen Fahrerinteraktionsklassifikator basierend auf einem realen Fahrversuch. Geeignete Klassifikationsmerkmale werden vorgeschlagen und untersucht: Konflikt und Passivität, Fahrlinienkonsistenz, Anpassung und individuelle Fahrlinienmuster.
Wir identifizieren fünf Fahrerinteraktionsstrategien: Anpassung, Beharrlichkeit, selektive Beharrlichkeit, Passivität und Unsicherheit. Die Klassifikationsergebnisse werden durch einen Vergleich mit subjektiven Experteneinschätzungen validiert. Zusätzlich wenden wir die Methode Dynamic Mode Decomposition (DMD) an, um die zugrundeliegenden Dynamiken für jede Klasse zu analysieren.

Wir präsentieren Konzepte zur Anpassung des Systemverhaltens in Echtzeit. Wir entwerfen eine Modellprädiktive Regelung (MPC) und führen verschiedene Performance-Tests durch, um sicherzustellen, dass das gewünschte Systemverhalten für jede der fünf Fahrerinteraktionsstrategien abgebildet werden kann. Wir wenden Online-Active-Set-Methoden und Innere-Punkt-Methoden an, um die Optimierungsprobleme in Echtzeit zu lösen. Das Framework wird mit einem Fokus auf die Anwendbarkeit in einem realen Fahrversuch implementiert. Zusätzlich schlagen wir weitere Konzepte zur Anpassung des Systemverhaltens basierend auf einzelnen Komponenten des Frameworks vor.

Disclaimer

Contents

List of Figures

List of Tables

Symbols and Abbreviations

Vehicle Dynamic Quantities

Symbol	Description	Unit
α_f	Front slip angle	rad
α_r	Rear slip angle	rad
β	Sideslip angle	rad
a_y	Lateral acceleration	$\mathrm{m\,s^{-2}}$
χ	Course angle	rad
δ	Steering angle	rad
J_z	Moment of inertia	$\mathrm{kg\,m^2}$
κ	Curvature	$\mathrm{m^{-1}}$
l_f	Distance from front axle to vehicle's center of gravity	m
l_r	Distance from rear axle to vehicle's center of gravity	m
m	Mass	kg
T_A	Assistance torque	Nm
T_S	Column torque	Nm
t	Time	s
ψ	Heading (yaw) angle	rad
v	Velocity	$\mathrm{m\,s^{-1}}$

Abbreviations

ACC	Active Cruise Control
ADAS	Advanced Driver Assistance System
DMD	Dynamic Mode Decomposition
DMDc	Dynamic Mode Decomposition with control
EPS	Electric Power Steering
IPPT	Individual Path Pattern Targeting
KKT	Karush-Kuhn-Tucker
LKA	Lane-Keeping Assistance
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
NN	Neural Network
QP	Quadratic Program
SbW	Steer-by-Wire
SVD	Singular Value Decomposition
SVM	Support Vector Machine
VC	Varying Coefficient

1 Introduction

1.1 Motivation and State of the Art

What is the first thing you do, when you get into your car and start the engine?

A. Adjust your seat.

B. Turn the radio on.

C. Turn the Lane-Keeping Assistance system off.

It is not unlikely that your choice would have been answer C. According to a Swiss study [2], only one in two drivers activates the lane departure assistant which intervenes actively in the steering task, and one in ten persons admits that they have never activated this system. But let us start again from the beginning. We are talking about Advanced Driver Assistance Systems (ADAS), popular features in modern vehicles, that are improving continuously. The automotive industry has increasingly focused on developing these intelligent technologies to enhance the driving experience of the driver and support him in different ways. Several types of ADAS have become widely recognized [3]: the cruise control maintains a constant speed and thus, reduces fatigue by allowing the driver to rest his foot while on long drives. The Adaptive Cruise Control (ACC) additionally increases the comfort by adapting the speed to the vehicle in front. Parking assistance systems help the driver to direct the vehicle into a parking space, and reduce possible damages on the vehicle during the parking task. Lane departure warning systems warn drivers of lane changes in the form of audible or visual alerts or even control vehicle steering to stay in the current lane, to reduce driver workload... But if we go back to the first question, one could ask: why are these systems not widely adopted by drivers, despite all of these benefits? We suppose that a possible explanation is the fact that some ADAS are designed for the average driver and do not take (or insufficiently take) the preferences and intentions of every single driver into consideration. Many assistance systems follow an objectively optimal target not adapted to the driver's expectation. Some lane departure assistants for example, intend to keep the vehicle in the lane by applying additional torque in a highly intrusive manner. This intervention can be perceived by drivers as disturbing or irritating, leading them to deactivate the system.

This gap between driver expectation and system capabilities poses a significant challenge for the automotive industry and makes it crucial to advance research on understanding of perception, expectation and acceptance of drivers towards ADAS. The first question that arises in this research is:

- What are the criteria that define the interaction between driver and assistance system? In other words, how can we characterize a positive or negative interaction between drivers and these systems?

To answer these questions, both industry and scientific research have primarily relied on subjective surveys. Either in driving simulator studies or in real world experiments, drivers were introduced to ADAS technologies and their feedback on the systems was gathered to evaluate their acceptance. The main considered criteria in the surveys were the driver attitude towards the system, the perceived usefulness, and subjective norms [4, 5, 6]. Further criteria were the driving experience and behavior [4, 7, 8], age, gender, and roadway environment [9, 10, 11], and accident data records [12]. While these criteria provide a snapshot of driver acceptance, they are not sufficient for a comprehensive assessment, as driver perception and preferences can vary over time. Hence, other studies additionally included a reevaluation of driver feedback from a long-term perspective to capture long-term changes in the interaction [6, 13]. Despite these different approaches, all of these studies still rely on subjective assessments, which are inherently limited due to the lack of a universal definition of the term *acceptance* (see [14]) and the influence of personal and contextual factors. Thus, as the automotive industry moves towards digitalization and automation, the use of objective approaches becomes essential in the analysis of the driver interaction with ADAS. However, fully objectifying this interaction remains challenging due to the complexity and variability of human behavior. Based on the literature reviewed by the author, there is only a limited number of sources that have attempted a completely objective approach. In [15], the authors conducted subjective assessments, but further employed data significance tests and sensitivity analysis to identify which objective parameters in the dataset influenced the subjective measures. Based on the objective parameters, the authors suggested an index for evaluating the driver acceptance of a lane change system [14]. A different objective approach can be found in [16, 17, 18, 19]. In contrast to the traditional focus on *acceptance*, the authors assessed the interaction between the driver and the vehicle more concretely by modeling the driver impedance. This involves analyzing the driver's resistance and responsiveness to the system's actions. Building on this approach, this thesis first considers to extend the modeling of the driver impedance to account for both automated and semi-automated scenarios.Furthermore, the necessity of long-term criteria and criteria for the overall driving experience has been highlighted in subjective assessments. Hence, this work aims to identify, model, and integrate these perspectives with objective criteria based only on the driver contribution to the driving task and on the dynamics of the vehicle. This requires modeling driver behavior *while* they interact with an active assistance system to provide a more comprehensive and accurate understanding of the interaction. After identifying the necessary criteria, the next question that arises is:

- How can driver interactions be systematically classified to improve their understanding and assessment? In other words, how can meaningful interaction strategies be identified and used for an effective driver classification based on objective criteria?

This question will be addressed in this thesis by analyzing driving data from a driving experiment, extracting relevant classification features, and identifying distinct driver interaction strategies. Using statistical and machine learning techniques, we develop a classification model that systematically categorizes these interactions. In the context of developing personalized driving assistance functions, the classification of driver interactions provides the foundation for adapting these systems to individual needs. This leads to the third question:

- How can the classification of driver interactions be used to configure personalized driving assistance functions that respond to individual driving behaviors and preferences?

In the context of driver assistance systems, different personalization approaches have been applied. A detailed literature review is presented in [20]. However, it focuses mainly on the technical side of personalization. For example, a driver can manually adjust his preferred safe distance configuration in the ACC by choosing between a number of pre-defined options. As driver preferences can vary in time, personalization should be considered a continuous process [21]. This perspective is also reflected in [16]. The authors integrated a continuously adaptable driver steering model into the control architecture of an assistance system and proved enhanced controller interventions. To support this approach, this thesis uses the computed driver interaction classifier as the basis for modeling and configuring a real-time capable Model Predictive Control (MPC). This enables the system to continuously adapt to the driver interaction and provide real-time personalization.

To address the challenge of increasing acceptance of driving assistance systems, this thesis integrates the answers to the presented key research questions above into a unified framework. The proposed shared control framework combines modeling the vehicle and driver dynamics, analyzing and classifying interaction patterns, and identifying methods for adapting the system behavior. To ensure the relevance and applicability of our findings, we incorporate data from real-world driving scenarios. This thesis focuses only on the lateral control aspect of the driver interaction, particularly the interaction with an intervening steering assistance system. Our goal is to propose methods to understand the driver interaction strategy in lateral shared control and lead to more intuitive steering system behavior.

1.2 Structure of this Thesis

This thesis has been developed as part of an industrial PhD program with Volkswagen AG. The main objective of this research is to address a key challenge in the automotive industry by analyzing the interaction between drivers and intervening steering assistance systems, to reduce the gap between driver expectation and system capabilities. The high-level foundational concepts explored in this thesis have led to the development of methodologies protected by patents and published European patent application from the author. These include [22], [23], [24], [25], [26], and [27].
The focus of this thesis is the scientific methods and approaches that contribute to the development and application of these concepts. We will analyze the different ways driver interact with an intervening steering assistance system. Hence, we will classify drivers based only on their contribution to the driving task and on the dynamics of the vehicle, while they share the control with a steering assistance system. We will design a framework for shared control and integrate the categorization in the control architecture of the assistance system to adapt its behavior to the individual driver strategy. The contributions of this thesis can be summarized as follows:

- Design of a shared control framework, where driver and assistance system continuously share the driving task in lateral control.
- Methodology for modeling the dynamics of the driver interaction and the vehicle during shared control.
- Identification of driver interaction strategies in shared control and construction of a classifier based on real driving data.
- Design concepts for adapting the control architecture to driver interaction strategies.

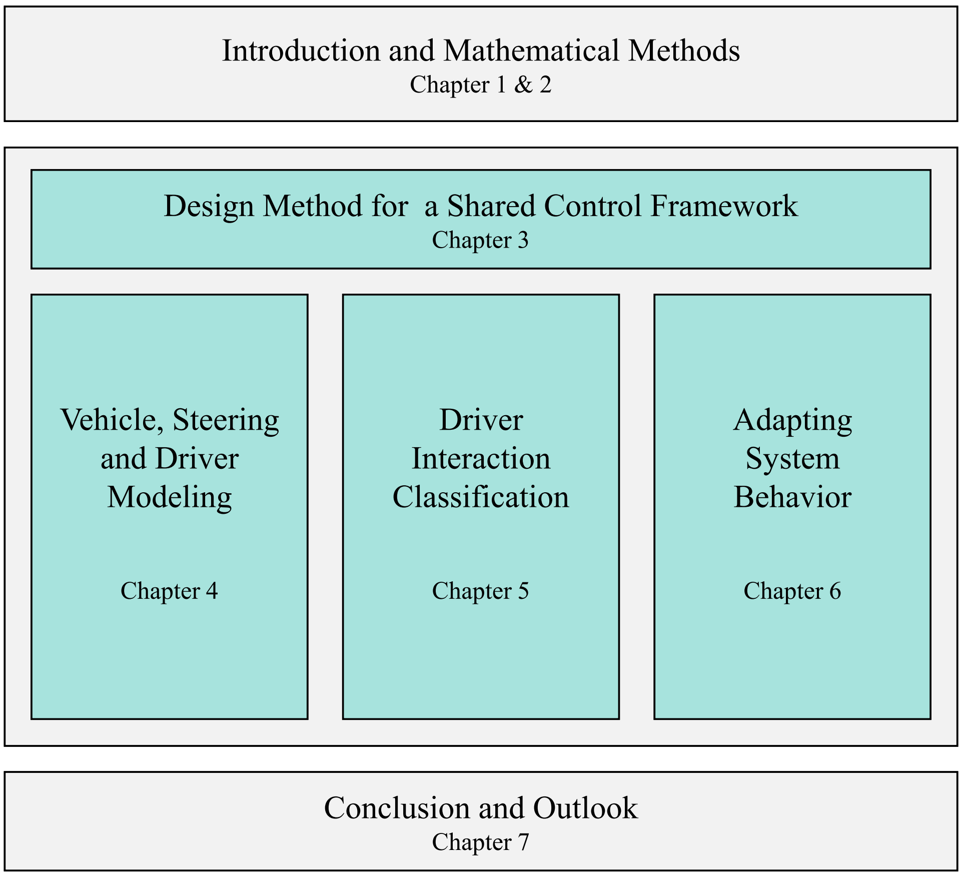

Introduction and Mathematical Methods
Chapter 1 & 2
Design Method for a Shared Control Framework
Chapter 3
Vehicle, Steering and Driver Modeling
Chapter 4
Driver Interaction Classification
Chapter 5
Adapting System Behavior
Chapter 6
Conclusion and Outlook
Chapter 7

Figure 1.1: Structure of this thesis

The outline of this thesis is illustrated in Figure 1.1. We start in Chapter 2 by giving an overview of the underlying theoretical basics of this thesis. It is structured around five key fields: Statistics, classification, dynamical systems, optimization and control, and model reduction. In Chapter 3, a shared control framework is presented. It forms the core of this work. We provide an overview of the framework's design and its main components. The following chapters will focus on each component individually, including the design, the modeling approach, and the contribution to the framework. In Chapter 4, we focus on modeling the dynamics of the system. We present the vehicle model, the steering mechanism models and we develop a driver-steering interaction model, that analyzes the driver torque at the steering wheel. In Chapter 5, we design a classifier, which is able to identify the driver interaction strategy of a driver when interacting with an intervening assistance system. The driver-steering interaction model and certain aspects of the classification process and the feature analysis have been previously published by the author as

- Amira El Amouri, Peter Markgraf, Samuel Schacher and Michael Herty. *Driver-Steering Interaction Model and Evaluation of Driver Interaction Strategies in Assisted Driving*, in IEEE Transactions on Intelligent Vehicles, May 2024, `https://doi.org/10.1109/TIV.2024.3407856`.
- Amira El Amouri. *Individual Path Pattern Targeting Algorithm for Personalized Shared Lateral Driving*. TechRxiv. December 2024. `https://doi.org/10.36227/techrxiv.173496546.60036140/v1`.

In Chapter 6, we design concepts for adapting the control architecture of a steering assistance system to the interaction strategy of the driver. Chapter 7 concludes the thesis and outlines potential future work.

2 Mathematical Methods

In this chapter, we introduce the mathematical methods that are fundamental to the modeling and analysis conducted throughout this thesis. This chapter is structured around five key fields: Statistics (in Section 2.1), classification (in Section 2.2), dynamical systems (in Section 2.3), optimization and control (in Section 2.4), and model reduction (in Section 2.5).

2.1 Statistics

2.1.1 Basic Measures

We start with recalling basic statistics measures mainly from [28].

Definition 2.1. Given the observations $x_1, x_2, \cdots, x_n$, with n being the sample size, the **mean value** is defined as

$$\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2.1}$$

Definition 2.2. For a discrete random variable with values $x_1, x_2, \cdots, x_n$ and corresponding probabilities $P(x_1), P(x_2), \cdots, P(x_n) \in [0,1]$, the **expected value** is

$$E\{\mathbf{x}\} = \mu_{\mathbf{x}} = \sum_{i=1}^{n} x_i \cdot P(x_i). \tag{2.2}$$

Definition 2.3. For observations $x_1, x_2, \cdots, x_n$, the **standard deviation** $\sigma_{\mathbf{x}}$ is given by

$$\sigma_{\mathbf{x}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \geq 0. \tag{2.3}$$

The standard deviation $\sigma_{\mathbf{x}}$ quantifies how much the observations vary or spread out around the mean value $\overline{\mathbf{x}}$. When the observations are concentrated around the mean value, the standard deviation takes on low values. When the observations are more spread out and significantly distant from the mean value, the standard deviation takes on high values.

Definition 2.4. The **standard score** for a specific observation $\mathbf{x}$ is computed by

$$\widetilde{\mathbf{x}} = \frac{\mathbf{x} - \mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}}, \tag{2.4}$$

where $\mu_{\mathbf{x}}$ is the expected value and $\sigma_{\mathbf{x}}$ the standard deviation of the dataset.

Definition 2.5. The **cross correlation function** of two discrete stationary processes X and Y is given by

$$R_{XY}(\tau) = E\{X_k Y_{k+\tau}\} = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} X_k Y_{k+\tau} = E\{X_{k-\tau} Y_k\}, \tag{2.5}$$

where E is the expected value operator. The cross correlation function is estimated by

$$\widehat{R}_{XY}(\tau) \approx R^n_{XY}(\tau) = \frac{1}{n} \sum_{k=1}^{n} X_k Y_{k+\tau}. \tag{2.6}$$

Depending on the sign of $\widehat{R}_{XY}(\tau)$, either a positive or negative correlation between X and Y can be indicated.

2.1.2 Varying Coefficient Method

The Varying Coefficient (VC) method is a method for solving time-varying problems that can handle constraints easily. It was introduced in [29]. The procedure is as follows: Given are snapshot vectors β_k and y_k and unknown coefficients x_k. The coefficients x_k are estimated by a standard linear regression:

$$y_k = \beta_k^T x_k + v_k, \qquad k = 1, \cdots, K, \tag{2.7}$$

where v_k denotes an error term to capture discrepancies due to measurements errors etc. The method assumes that the coefficients x_k change slowly over time:

$$x_{k+1} = x_k + u_k, \tag{2.8}$$

where u_k describes the rate of change of x_k.
The standard least square method minimizes the error $\sum_k v_k^2$, whereas VC aims to minimize the weighted sum $\sum_k v_k^2 + \sum_i \gamma_i \sum_k u_{k,i}^2$, where $u_{k,i}$ denotes the i-th entry of the vector u_k and $\gamma_i > 0$ are weight parameters.
We arrange the snapshots into matrices

$$\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_K \end{bmatrix}^T, \tag{2.9a}$$
$$\mathbf{y} = \begin{bmatrix} y_1 & \cdots & y_K \end{bmatrix}^T, \tag{2.9b}$$
$$\mathbf{u} = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix}^T, \tag{2.9c}$$
$$\mathbf{v} = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix}^T, \tag{2.9d}$$
$$\mathbf{B} = \operatorname{diag}\left(\beta_1^T, \cdots, \beta_K^T\right), \tag{2.9e}$$

and define the matrix $\mathbf{P}$ as

$$\mathbf{P} = \begin{bmatrix} -\mathbf{I}_n & \mathbf{I}_n & & & 0 \\ & -\mathbf{I}_n & \mathbf{I}_n & & \\ & & \ddots & \ddots & \\ 0 & & & -\mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \in \mathbb{R}^{(K-1)n \times Kn}, \tag{2.10}$$

with $\mathbf{I}_n$ denoting the identity matrix of order n.
Then, equations (2.7) and (2.8) lead to the model

$$\mathbf{y} = \mathbf{B}\mathbf{x} + \mathbf{v}, \tag{2.11a}$$
$$\mathbf{u} = \mathbf{P}\mathbf{x}. \tag{2.11b}$$

We write $\Gamma = \mathrm{diag}(\gamma_1, \cdots, \gamma_n)$ and define the matrix $\mathbf{G}$ as

$$\mathbf{G} = \mathbf{I}_{K-1} \otimes \Gamma, \tag{2.12}$$

where $\mathbf{I}_{K-1}$ denotes the identity matrix of order $K-1$ and $\otimes$ describes the Kronecker product, i.e., (2.12) becomes,

$$\mathbf{G} = \begin{bmatrix} \Gamma & & 0 \\ & \ddots & \\ 0 & & \Gamma \end{bmatrix} \in \mathbb{R}^{(K-1)n \times (K-1)n}. \tag{2.13}$$

The expression to be minimized to estimate x is given by

$$\mathbf{Q} = (\mathbf{y} - \mathbf{B}\mathbf{x})^T(\mathbf{y} - \mathbf{B}\mathbf{x}) + \mathbf{x}^T\mathbf{P}^T\mathbf{G}\mathbf{P}\mathbf{x}. \tag{2.14}$$

2.2 Classification

Classification is the systematic process of assigning objects into predefined categories based on training data. It involves identifying the patterns and relationships within the data and enables to predict the category for new instances based on their features. A common example of a classification task is an email spam filter [30]. Trained with several examples of *spam* and *non-spam* emails, it learns to predict whether a new email belongs to either category. This section introduces basic concepts of classification based on the references [30, 31, 32].

2.2.1 Basic Concepts

Definition 2.6. The distinct categories into which observations are classified are called **classes**.

Definition 2.7. We call **label** the output variable that the classification model aims to predict.

Definition 2.8. We denote a **feature** as a characteristic of an observation that serves as an input to the classification model. Features are used by the model to make predictions about the labels.

Definition 2.9. A **classifier** is the model used to assign labels to observations based on their features. Examples of classifiers include decision trees, support vector machines, and neural networks.

Definition 2.10. Given a data set $\mathbf{D} = \{(\mathbf{x}_k, y_k)\}_{k=1}^m$, where $\mathbf{x}_k$ is a vector containing the features of instance k and $y_k \in \{\pm 1\}$ is its corresponding class label. The data set is split into a **training** set $\mathbf{X}^{tr} = \{(\mathbf{x}_i, y_i)\}_{i\in\mathbf{I}}$ and a **test** set $\mathbf{X}^t = \{(\mathbf{x}_i, y_i)\}_{i\in\mathbf{J}}$, where $\mathbf{I} \cap \mathbf{J} = \emptyset$ and $|\mathbf{I} \cup \mathbf{J}| = m$. The training set is used to train the classification model and the test set to evaluate its performance.

Definition 2.11. We call **accuracy** an evaluation metric that measures the proportion of correctly classified instances out of the total instances. It is computed by

$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Total number of classifications}} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \in [0,1], \tag{2.15}$$

where TP is the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives.

Definition 2.12. **Precision** indicates the accuracy of the positive predictions made by the model. It is computed by

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \in [0,1]. \tag{2.16}$$

Definition 2.13. **Recall** is the ratio of positive instances that are correctly predicted by the classifier. It is computed by

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \in [0,1]. \tag{2.17}$$

Definition 2.14. The $\mathbf{F_1}$ **score** is an evaluation metric that combines recall and precision. It is given by

$$F_1 = \frac{2}{\dfrac{1}{\text{precision}} + \dfrac{1}{\text{recall}}} = \frac{\text{TP}}{\text{TP} + \dfrac{\text{FN}+\text{FP}}{2}} \in [0,1]. \tag{2.18}$$

Definition 2.15. A **confusion matrix** is a table that summarizes and visualizes the performance of a classification model by comparing its predictions to the actual targets. It is represented as

Definition 2.16. When the observation is classified into one of two classes, we call this type of classification **binary classification**. When the model predicts one of three or more classes, we call this type **multiclass classification**. Binary classifiers are expandable to multiclass problems, mainly by decomposing the problem with n classes, into multiple binary problems and training $n(n-1)/2$ classifiers, where each classifier f_{ij} is trained using the features from class i as positive examples, and the features from class j as negative examples, while disregarding all other classes.

Definition 2.17. In **k-fold cross-validation**, the training set $\mathbf{X}^{tr}$ is randomly divided into k (approximately) equal folds $X_1^{tr}, \cdots, X_k^{tr}$. In each iteration step of k-fold cross-validation, one fold X_i^{tr} is used for testing, while the remaining $k-1$ folds are used to train the classifier. This process is repeated k times, with each fold serving as the test set once. The performance achieved in each iteration is then averaged to determine the overall model performance.

		Predicted Condition	
	Total Population	Predicted Positive	Predicted Negative
Actual Condition	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

2.2.2 *Support Vector Machine*

Support Vector Machine (SVM) is a supervised learning algorithm used for classification. The term *supervised* indicates that SVM is trained with data that is already labeled with the correct class.
The main idea of SVM is to find an optimal hyperplane that maximally separates the different classes in the high-dimensional feature space. Figure 2.1 illustrates the idea of separating hyperplanes in a 2-dimensional feature space (x_1, x_2).

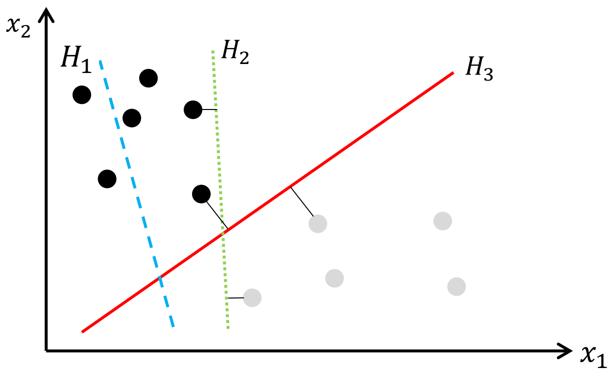


Figure 2.1: Illustration of separating hyperplanes in a 2-dimensional feature space (x_1, x_2). H_1 fails to separate the classes. H_2 separates the classes but with a small margin. H_3 optimal maximal-margin hyperplane.

The black and gray points represent two classes. The dashed line H_1 does not separate the classes, the dotted line H_2 separates them but is not maximal, and the solid line H_3 is the

optimal separating hyperplane.
Formally, SVM requires solving the optimization problem

$$\min_{\mathbf{w},b,\boldsymbol{\zeta}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{k=1}^{m}\zeta_k \tag{2.19a}$$

$$\text{subject to} \quad y_k(\mathbf{w}^T\varphi(\mathbf{x}_k)+b) \geq 1-\zeta_k, \tag{2.19b}$$

$$\zeta_k \geq 0, \tag{2.19c}$$

where $C > 0$ represents a penalty parameter of the error term, $\varphi(\cdot)$ denotes the kernel function and ζ_k is a slack variable that measures the degree of misclassification of sample k.

2.2.3 Neural Network

Neural Networks (NN) consist of interconnected layers: an input and an output layer, and at least one hidden layer in between. Each layer consists of neurons, that process input data and pass the result to the next layer. An illustration of this structure is shown in Figure 2.2.

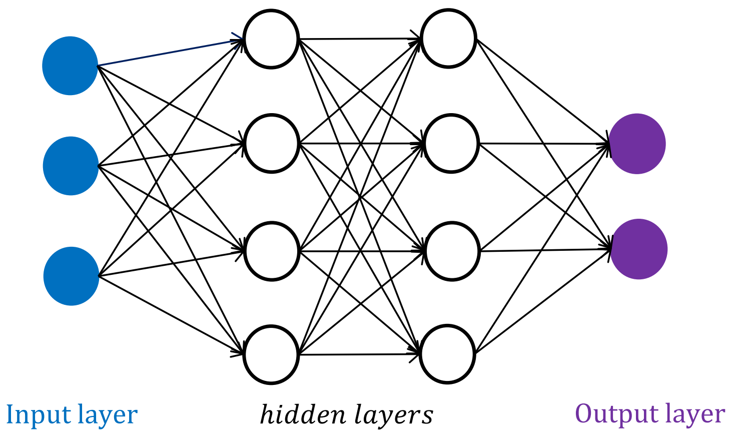


Figure 2.2: Illustration of a simple neural network architecture consisting of input, hidden, and output layers.

Let n_{i-1} be the number of neurons in layer $i-1$, then the value of the k-th neuron a_k^i in layer i is computed by

$$a_k^i = \varphi\left(\sum\nolimits_{j=1}^{n_{i-1}} \omega_{kj}^i a_j^{i-1} + b_k^i\right), \tag{2.20}$$

where φ is an activation function, ω_{kj}^i the weight connecting a_j^{i-1} to a_k^i, and b_k^i the bias term for a_k^i. An illustration of the computation of a single neuron is given in Figure 2.3
In a classification problem, the input layer receives the feature vector, the hidden and output layers process the signals received by the input layer, and the output layer outputs the corresponding class label.

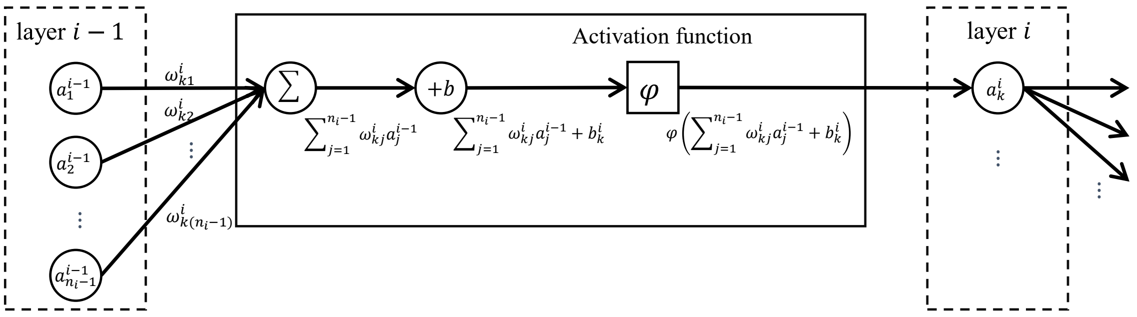

layer $i-1$
a_1^{i-1}
a_2^{i-1}
$a_{n_i-1}^{i-1}$
ω_{k1}^i
ω_{k2}^i
$\omega_{k(n_i-1)}^i$
Activation function
$\sum$
$+b$
φ
$\sum_{j=1}^{n_i-1} \omega_{kj}^i a_j^{i-1}$
$\sum_{j=1}^{n_i-1} \omega_{kj}^i a_j^{i-1} + b_k^i$
$\varphi\left(\sum_{j=1}^{n_i-1} \omega_{kj}^i a_j^{i-1} + b_k^i\right)$
layer i
a_k^i

Figure 2.3: Illustration of the computation of the k-th neuron in layer i.

2.3 Dynamical Systems

The introduction of dynamical systems goes back to the 19th century and to Henri Poincaré[1] [33]. The starting motivation were Newton[2]'s laws of motion and the study of time behavior of classical mechanical systems. Influenced by Poincaré's theory, David Birkhoff[3] published in 1927 his great work *Dynamical Systems*, which made him popular worldwide [34, 35]. Dynamical systems became a central aspect in studying various applications such as physics, mathematics, engineering, biology and chemistry. By the end of the 20th century, Ali H. Nayfeh[4] made an important contribution to the study of dynamical systems. He developed analysis techniques for nonlinear dynamics in the different application fields, from aerodynamics, ship motion and aircraft, to micro-electromechanical systems, computer software and nanotechnology [36]. His contributions became a groundwork in common daily life structures.
In this section, we will begin by defining key terms and provide the definition and types of dynamical systems, as well as the essential concepts in ensuring the well-posedness of these models. Unless otherwise noted, the information presented throughout this section is derived from references [1, 37, 38, 39, 40, 41].

1 Jules Henri Poincaré: April 29, 1854 (Nancy) – July 17, 1912 (Paris), French mathematician, theoretical physicist, engineer and philosopher of science

2 Sir Isaac Newton: December 25, 1642 (Woolsthorpe, Lincolnshire) – March 20, 1727 (London), English mathematician, physicist, astronomer, alchemist, and theologian.

3 George David Birkhoff: March 21, 1884 (Overisel) – November 12, 1944 (Cambridge), American mathematician

4 Ali Hasan Nayfeh: Dezember 21, 1933 (Tulkarem, Palestine) - March 27, 2017 (Amman), Palestinian-Jordanian mathematician, mechanical engineer and physicist

2.3.1 Basic Concepts

Definition 2.18. The **state** of a dynamic system is the set of all physical variables, that describe the system at a given point in time, allowing to predict the dynamic behavior of this system for any time.

Definition 2.19. We call **state variables** the minimum necessary set of physical variables required to completely determine the dynamic state of a physical system.

Definition 2.20. The **state vector x** of a dynamical system is the vector containing all n state variables $x_1, \cdots, x_n$ as its components.

Definition 2.21. The **state space** is the n-dimensional space, where each dimension corresponds to one of the state variables.

Definition 2.22. We call a continuous function $\mathbf{f} : \mathbf{K} \longrightarrow \mathbb{R}^n$ **locally Lipschitz** if at each $(t_1, \mathbf{x}_1) \in \mathbf{K}$, there exist a ball $\overline{\mathbf{B}}_r(\mathbf{x}_1)$, $\alpha > 0$ with $[t_1 - \alpha, t_1 + \alpha] \times \overline{\mathbf{B}}_r(\mathbf{x}_1) \subset \mathbf{K}$, and a constant $L = L(t_1, \mathbf{x}_1) > 0$, such that:

$$|\mathbf{f}(t,\mathbf{x}) - \mathbf{f}(t,\overline{\mathbf{x}})| \leq L(t_1,\mathbf{x}_1) \cdot |\mathbf{x} - \overline{\mathbf{x}}|, \quad \text{for} \quad \mathbf{x},\ \overline{\mathbf{x}} \in \overline{\mathbf{B}}_r(\mathbf{x}_1). \tag{2.21}$$

We call $\mathbf{f}$ a **globally Lipschitz** function, if the constant $L > 0$ does not depend of $(t_1, \mathbf{x}_1) \in \mathbf{K}$, i.e.,

$$|\mathbf{f}(t,\mathbf{x}) - \mathbf{f}(t,\overline{\mathbf{x}})| \leq L \cdot |\mathbf{x} - \overline{\mathbf{x}}|, \quad \text{for} \quad (t,\mathbf{x}),\ (t,\overline{\mathbf{x}}) \in \mathbf{K}. \tag{2.22}$$

Theorem 2.1. Let $\mathbf{K} \subset \mathbb{R}^n$ be an open set, $\mathbf{f} : \mathbf{K} \longrightarrow \mathbb{R}^n$ a locally Lipschitz continuous function, and to all $\mathbf{x}_0 \in \mathbf{K}$ there exists the solution $\mathbf{x}(t, \mathbf{x}_0)$ of the initial value problem

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}}(t) & = & \mathbf{f}(t,\mathbf{x}), \\ \mathbf{x}(t_0) & = & \mathbf{x}_0, \end{array} \right. \tag{2.23}$$

globally for all $t \in \mathbb{R}$. Then the function $\phi(t, \mathbf{x}_0) := \mathbf{x}(t, \mathbf{x}_0)$ with $\mathbf{x}_0 \in \mathbf{K}$ and $t \in \mathbb{R}$ is a **dynamical system**.

Theorem 2.2. Let $\mathbf{f}$ be a locally Lipschitz continuous function. Then, for an initial point $\mathbf{x}_0 \in \mathbf{K}$ the differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t)) \tag{2.24}$$

admits a **unique** solution defined on the maximal **existence** interval.

To characterize the dynamic behavior of linear systems, we briefly recall the concepts of eigenvalues, eigenvectors, and eigendecomposition.

Definition 2.23. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a square matrix. We call the scalar $\lambda \in \mathbb{C}$ and the vector $\mathbf{v} \in \mathbb{C}^n, \mathbf{v} \neq 0$ satisfying the eigenvalue equation

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}, \tag{2.25}$$

eigenvalue and corresponding **eigenvector** of $\mathbf{A}$ respectively.

Definition 2.24. The **Eigendecomposition** of a square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is given by the form

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}, \tag{2.26}$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is a square matrix containing the eigenvectors of $\mathbf{A}$ as columns, and Λ a diagonal matrix containing the eigenvalues $\lambda_{ii} = \lambda_i, i = 1, \cdots, n$, of $\mathbf{A}$ as diagonal entries. This decomposition exists if and only if $\mathbf{A}$ has n linearly independent eigenvectors, i.e., if $\mathbf{Q}$ is invertible.

2.3.2 State-Space Representation

A dynamical system is modeled by input, output and state variables. Let $u_1(t), \cdots, u_r(t)$ be r input variables, $y_1(t), \cdots, y_m(t)$ be m output variables and $x_1(t), \cdots, x_n(t)$ be n state variables. A general overview of a dynamical system is illustrated in Figure 2.4

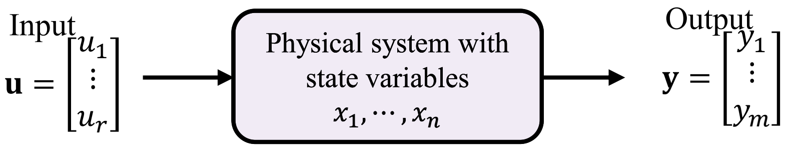

Input
$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_r \end{bmatrix}$
Physical system with state variables $x_1, \cdots, x_n$
Output
$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$

Figure 2.4: Illustration of a dynamical system with r input, n state and m output variables.

A state-space representation shows the full internal structure of a dynamical system. It is based on a series of first order differential equations (for time-continuous systems) as in (2.23) or difference equations (for time-discrete systems) and an output equation.
The differential (or difference) equations link the temporal changes of the state variables with the current values of the state variables and the input variables. This relationship describes the dynamic behavior of the system under analysis. The output equation represents the dependency of the output variables on the state and the input variables. For a time-continuous system, the state-space representation is given by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.27a}$$
$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.27b}$$

where

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} f_1(x_1(t), \cdots, x_n(t), u_1(t), \cdots, u_r(t)) \\ \vdots \\ f_n(x_1(t), \cdots, x_n(t), u_1(t), \cdots, u_r(t)) \end{bmatrix}, \tag{2.28}$$

and

$$\mathbf{g}(\mathbf{x}(t),\mathbf{u}(t)) = \begin{bmatrix} g_1(x_1(t),\cdots,x_n(t),u_1(t),\cdots,u_r(t)) \\ \vdots \\ g_m(x_1(t),\cdots,x_n(t),u_1(t),\cdots,u_r(t)) \end{bmatrix}. \tag{2.29}$$

For linear time-continuous systems, (2.27) becomes

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{2.30a}$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \tag{2.30b}$$

where the matrices **A**, **B**, **C** and **D** are interpreted as follows:

- **A** is the $n \times n$ state (or system) matrix, and describes the system's internal dynamics.
- **B** is the $n \times r$ input matrix, and describes the influence of the input to the system's state.
- **C** is the $m \times n$ output matrix, and maps the state vector to the output vector.
- **D** is the $m \times r$ feedthrough (or direct transmission) matrix, and describes the input's effect on the output. If this matrix is not zero, the system responds immediately (in an infinitely short time) to changes in the input vector **u**.

A block diagram of a linear dynamical system in state-space representation is illustrated in Figure 2.4

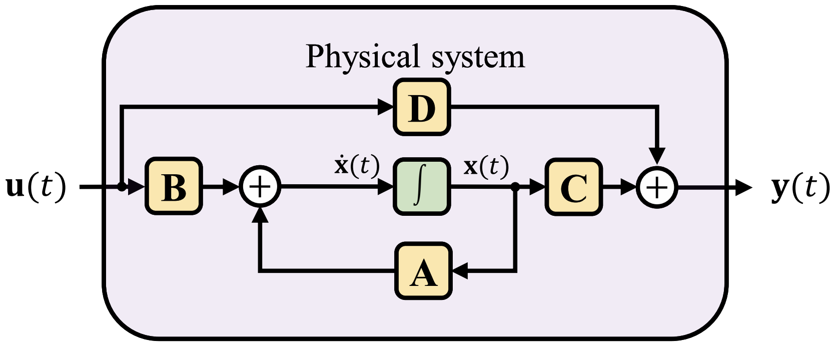


Figure 2.5: Block diagram of a linear system in state-space representation [1].

When implementing models in digital circuits or microprocessors, it is necessary to represent them in a time-discrete domain. In this context, the continuous signals are sampled at discrete time intervals t_k, $k = 0,1,2,\cdots$.
Let $\mathbf{x}_k = \mathbf{x}(t_k)$ for $t_k = k \cdot T$, then the discretized form of (2.27) becomes

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k,\mathbf{u}_k), \tag{2.31a}$$
$$\mathbf{y}_k = \mathbf{G}(\mathbf{x}_k,\mathbf{u}_k), \tag{2.31b}$$

and the linearized form becomes

$$\mathbf{x}_{k+1} = \widetilde{\mathbf{A}}\mathbf{x}_k + \widetilde{\mathbf{B}}\mathbf{u}_k, \tag{2.32a}$$
$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k, \tag{2.32b}$$

where $\widetilde{\mathbf{A}}$, $\widetilde{\mathbf{B}}$, $\mathbf{C}$ and $\mathbf{D}$ are obtained from the Jacobian of $\mathbf{F}$ and $\mathbf{G}$ with respect to $\mathbf{x}$ and $\mathbf{u}$. The discrete-time matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ are given by

$$\widetilde{\mathbf{A}} = e^{\mathbf{A}\cdot T}, \qquad \text{and} \qquad \widetilde{\mathbf{B}} = e^{\mathbf{A}\cdot T}\cdot\mathbf{B}. \tag{2.33}$$

A block diagram of a the time-discrete linear dynamical system in state-space representation is illustrated in Figure 2.6

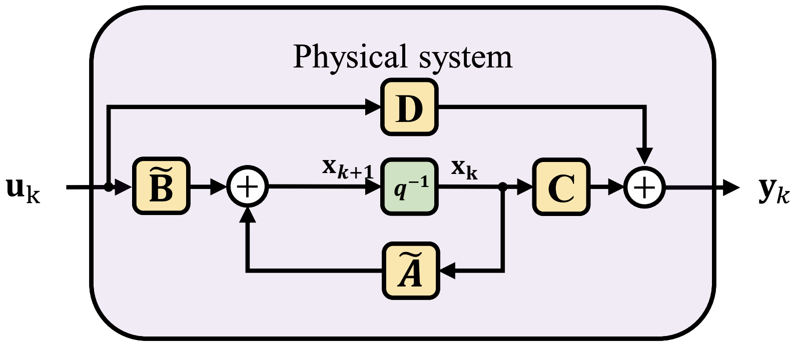


Figure 2.6: Block diagram of a discrete linear system in state-space representation [1].

2.3.3 Solution Method and Stability Analysis

Given a dynamical system of the form

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), t), \tag{2.34}$$

with an initial state $\mathbf{x}(t_0) = \mathbf{x}_0$. A first-order method to approximate the solution of (2.34) is the explicit Euler method. It iteratively computes the values of $\mathbf{x}$ at discrete time steps. Let h be the time step size, then for $k = 0, 1\cdots, n$, $n \in \mathbb{N}$, we compute

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\cdot f(\mathbf{x}_k, t_k). \tag{2.35}$$

Definition 2.25. In a dynamic system, an **equilibrium point** refers to a state where the system's variables remain unchanged over time and stay constant for future times.

Definition 2.26. Let $\mathbf{B}(\mathbf{x}, r)$ denote the open ball with radius r and center $\mathbf{x}$. An equilibrium point $\mathbf{x}^*$ is called **stable**, if for any $\varepsilon > 0$, there exists a $\delta > 0$ such that for all $\mathbf{x} \in \mathbf{B}(\mathbf{x}^*, \delta)$ and $t \geq 0$

$$\mathbf{x}\cdot t \in \mathbf{B}(\mathbf{x}^*, \varepsilon). \tag{2.36}$$

If $\lim_{t\to\infty}\mathbf{x}(t) = \mathbf{x}^*$, then $\mathbf{x}^*$ is called **asymptotically stable**.
If a system is disturbed around a stable equilibrium point, it eventually returns to its original location and remains there. We call the equilibrium point $\mathbf{x}^*$ **unstable**, if it is not stable.

Theorem 2.3. A *continuous* linear dynamical system $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$ is **asymptotically stable** if and only if all eigenvalues of $\mathbf{A}$ satisfy $\mathrm{Re}(\lambda_j) < 0$.

Theorem 2.4. A *discrete* linear dynamical system $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$ is **asymptotically stable** if and only if all eigenvalues of $\mathbf{A}$ satisfy $|\lambda_j| < 1$.

Example 2.1 (Vehicle Suspension System). We consider a simple vehicle suspension system consisting of a spring and a damper that connect the vehicle body to the wheels. An illustration is given in Figure 2.7.

The equilibrium point is where the vehicle is at rest, and the spring is neither compressed

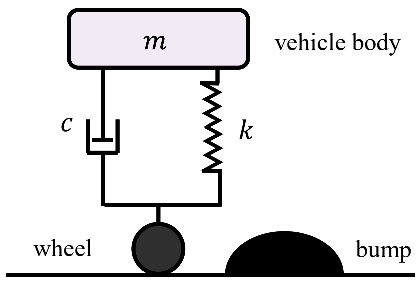


Figure 2.7: Illustration of the vehicle suspension system.

nor extended. We assume the vehicle hits a bump, i.e., an external force $F(t)$ acts on the suspension. The suspension system reacts by compressing the spring and activating the damper, aiming to return the vehicle to a stable position. The equation of motion of the system is given by

$$m\ddot{x} + c\dot{x} + kx = F(t), \tag{2.37}$$

where x describes the displacement of the vehicle body from its equilibrium position, $\dot{x}$ describes the velocity, $\ddot{x}$ is the acceleration, m is the mass of the vehicle body, c is the damping coefficient, and k is the spring constant.

Let $x_1 = x(t)$ and $x_2 = \dot{x}(t)$, then the system can be written in state-space representation as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \cdot F(t), \tag{2.38a}$$

$$y = x_1. \tag{2.38b}$$

The system is linear. To analyze its stability, we compute the eigenvalues by solving the characteristic equation

$$m\lambda^2 + c\lambda + k = 0. \tag{2.39}$$

This results in the eigenvalues

$$\lambda_1 = \frac{-c + \sqrt{c^2 - 4mk}}{2m} \quad \text{and} \quad \lambda_2 = \frac{-c - \sqrt{c^2 - 4mk}}{2m}, \tag{2.40}$$

and the following three cases:

- If $c^2 > 4mk$, we have $\lambda_1, \lambda_2 \in \mathbb{R}_{\leq 0}$. This indicates an overdamped system that returns slowly to equilibrium without oscillating.
- If $c^2 = 4mk$, we have $\lambda_1 = \lambda_2 \in \mathbb{R}$. This indicates a critically damped system that returns quickly to equilibrium without oscillating.
- If $c^2 < 4mk$, we have $\lambda_1, \lambda_2 \in \mathbb{C}$, with negative real parts. This indicates an underdamped system that oscillates while returning to equilibrium.

In all three cases, the real parts of the eigenvalues are negative. Thus, the system is stable, i.e., after hitting a bump, the vehicle will eventually return to its equilibrium position.

2.4 Optimization and Control

Optimization and control are fundamental concepts in engineering and applied mathematics [42]. The main goal in mathematical optimization is to find the best solution from a set of feasible solutions, that minimizes (or maximizes) an objective function subject to certain constraints. For a function $f : \mathbb{R}^n \to \mathbb{R}$ a general constrained continuous optimization problem is given by

$$\min_{\mathbf{x}} \; f(\mathbf{x}) \tag{2.41a}$$
$$\text{subject to} \quad c_i(\mathbf{x}) = 0, \quad i \in \mathcal{E}, \tag{2.41b}$$
$$c_i(\mathbf{x}) \geq 0, \quad i \in \mathcal{I}, \tag{2.41c}$$

where f is the nonlinear objective function and c_i, $i \in \mathcal{E} \cup \mathcal{I}$, are the constraints functions. To characterize the solution of (2.41), we introduce the necessary first-order optimality conditions, known as the Karush-Kuhn-Tucker conditions, in the following theorem.

Theorem 2.5. Let $\mathbf{x}^*$ be a local solution of (2.41), $\mathcal{A}$ be the active set defined by

$$\mathcal{A}(\mathbf{x}^*) = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(\mathbf{x}^*) = 0\}, \tag{2.42}$$

and let $\mathcal{L}$ be the Lagrangian function

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i \mathbf{c}_i(x). \tag{2.43}$$

Then there is a Lagrange multiplier λ^* with components λ_i^*, $i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions, called **Karush-Kuhn-Tucker (KKT)** conditions, are satisfied at $(\mathbf{x}^*, \lambda^*)$

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*) = \nabla f(\mathbf{x}^*) - \sum_{i \in \mathcal{A}(\mathbf{x}^*)} \lambda_i^* \nabla c_i(\mathbf{x}^*) = 0, \qquad \text{(stationarity)} \tag{2.44a}$$
$$c_i(\mathbf{x}^*) = 0, \ \forall i \in \mathcal{E}, \quad c_i(\mathbf{x}^*) \geq 0, \ \forall i \in \mathcal{I}, \qquad \text{(feasibility)} \tag{2.44b}$$
$$\lambda_i^* \geq 0, \quad \forall i \in \mathcal{I}, \qquad \text{(nonnegativity)} \tag{2.44c}$$
$$\lambda_i^* c_i(\mathbf{x}^*) = 0, \quad \forall i \in \mathcal{E} \cup \mathcal{I}. \qquad \text{(complementarity)} \tag{2.44d}$$

In the following sections, we will provide a mathematical introduction to quadratic programming, optimal control and model predictive control and the essential solution methods required for each approach.

2.4.1 Quadratic Programming

Quadratic Programming (QP) is a nonlinear optimization technique that plays a significant role in many application fields, such as engineering, finance, computer science and environmental science [43]. The QP optimization problem involves a quadratic objective function and linear constraints, and can be formulated as [44]

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}+\mathbf{c}^T\mathbf{x} \tag{2.45a}$$

$$\text{subject to} \quad \mathbf{a}_i^T\mathbf{x} = b_i, \quad i \in \mathcal{E}, \tag{2.45b}$$

$$\mathbf{a}_i^T\mathbf{x} \geq b_i, \quad i \in \mathcal{I}, \tag{2.45c}$$

where $\mathbf{Q}$ is an $n \times n$ symmetric matrix, $\mathbf{c}$ is an n-dimensional vector containing linear coefficients and $\mathbf{x}$ is the n-dimensional vector of decision variables. (2.45b) and (2.45c) are the linear optimization constraints, where $\mathbf{a}_i$, $i \in \mathcal{E} \cup \mathcal{I}$, are n-dimensional vectors, with $\mathcal{E}$ and $\mathcal{I}$ being the sets of indices for equality and inequality constraints respectively, and b_i are scalar constraint bounds. The QP problem (2.45) is called *convex*, if the matrix $\mathbf{Q}$ is positive semi definite.
The Lagrangian for the QP problem (2.45) can be written as

$$\mathcal{L}(\mathbf{x},\lambda,\mu) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}x+\mathbf{c}^T\mathbf{x}+\sum_{i\in\mathcal{E}}\lambda_i(\mathbf{a}_i^T\mathbf{x}-b_i)+\sum_{i\in\mathcal{I}}\mu_i(\mathbf{a}_i^T\mathbf{x}-b_i). \tag{2.46}$$

Following Theorem 2.5, for an optimal solution $\mathbf{x}^*$, the Karush-Kuhn-Tucker (KKT) conditions (2.44) become

$$\nabla_x\mathcal{L}(\mathbf{x}^*,\lambda^*,\mu^*) = \mathbf{Q}\mathbf{x}^*+\mathbf{c}+\sum_{i\in\mathcal{E}}\lambda_i^*\mathbf{a}_i+\sum_{i\in\mathcal{I}}\mu_i^*\mathbf{a}_i = 0 \qquad \text{(stationarity)} \tag{2.47a}$$

$$\mathbf{a}_i^T\mathbf{x}^* = b_i,\ \forall i \in \mathcal{E}, \quad \mathbf{a}_i^T\mathbf{x}^* \geq b_i,\ \forall i \in \mathcal{I}, \qquad \text{(feasibility)} \tag{2.47b}$$

$$\mu_i^* \geq 0, \quad \forall i \in \mathcal{I}, \qquad \text{(nonnegativity)} \tag{2.47c}$$

$$\mu_i^*(\mathbf{a}_i^T\mathbf{x}^*-b_i) = 0, \quad \forall i \in \mathcal{E}\cup\mathcal{I}. \qquad \text{(complementarity)} \tag{2.47d}$$

2.4.2 Active-Set Methods

One of the most widely used methods to solve QP problems are active-set methods [44]. The method operates by iteratively adjusting the set of active constraints and solving a series of equality-constrained subproblems. Algorithm 1 describes this approach.

Algorithm 1: Active-Set Method for Convex QP [44]

```
GIVEN a feasible initial state $\mathbf{x}_0$ and the subset $\mathcal{W}_0$ of active constraints at $\mathbf{x}_0$
FOR $k = 0, 1, 2, \cdots$
    Solve
        $$\min_{\mathbf{p}} \frac{1}{2}(\mathbf{x}_k + \mathbf{p})^T \mathbf{Q}(\mathbf{x}_k + \mathbf{p}) + \mathbf{c}^T(\mathbf{x}_k + \mathbf{p}),$$
        subject to $\mathbf{a}_i^T \mathbf{p} = 0, \ \forall i \in \mathcal{W}_k.$
    IF $\mathbf{x}_k + \mathbf{p}$ is feasible
        Compute Lagrange multipliers $\lambda_i$ and $\mu_i$ that satisfy
            $$\sum_{i \in \mathcal{E}} \mathbf{a}_i \mu_i + \sum_{i \in \mathcal{W}_k} \mathbf{a}_i \lambda_i = \mathbf{Q}\mathbf{x}_k + \mathbf{c}.$$
        IF $\lambda_i \geq 0, \forall i \in \mathcal{W}_k$
            STOP solution $\mathbf{x}^* = \mathbf{x}_k$.
        ELSE
            Set $j = \arg\min_{j \in \mathcal{W}_k} \lambda_j$.
            Set $\mathbf{x}_{k+1} = \mathbf{x}_k$.
            Set $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$.
    ELSE
        Compute $\alpha_k = \max \left\{ \alpha \in [0,1] \,\middle|\, \alpha \leq \frac{b_i - \mathbf{a}_i^T \mathbf{x}_k}{\mathbf{a}_i^T \mathbf{p}} \right\}$.
        Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}$.
        IF $\alpha_k < 1$, add the bounding constraint to $\mathcal{W}_{k+1}$.
ENDFOR
RETURN $\mathbf{x}^*$.
```

For application where the computational time for solving the QP is crucial or when multiple QPs are solved sequentially, an *online* active-set method is presented in [45]. It is based on the idea, that the active set remains unchanged from one QP to the next, which is a different approach from the conventional warm starting techniques. We briefly describe the approach of the online active-set method.
We consider the quadratic program QP($\mathbf{x}_0$) of the form

$$\text{QP}(\mathbf{x}_0): \qquad \min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T \mathbf{Q}\mathbf{w} + \mathbf{w}^T g(\mathbf{x}_0) \tag{2.49a}$$

$$\text{subject to} \quad \mathbf{G}\mathbf{w} \geq \mathbf{b}(\mathbf{x}_0), \tag{2.49b}$$

where $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $g(\mathbf{x}_0) = \mathbf{F}^T \mathbf{x}_0$ a linear function of $\mathbf{x}_0$ with $\mathbf{F} \in \mathbb{R}^{n \times n}$, $\mathbf{G} \in \mathbb{R}^{m \times n}$ the constraints coefficients of the inequality constraints, and $\mathbf{b}(\mathbf{x}_0) = \overline{\mathbf{b}} + \mathbf{E}\mathbf{x}_0$ an affine function of $\mathbf{x}_0$ with $\overline{\mathbf{b}} \in \mathbb{R}^m$ and $\mathbf{E} \in \mathbb{R}^{m \times n}$.
We assume, that QP($\mathbf{x}_0$) was solved for the initial state $\mathbf{x}_0$. Let $(\mathbf{w}^*, \lambda^*)$ be its computed optimal solution, with Lagrangian multiplier λ^*. For a new initial state $\mathbf{x}_0^{\text{new}}$, we compute the solution $(\mathbf{w}^*_{\text{new}}, \lambda^*_{\text{new}})$ by following Algorithm 2.

Algorithm 2: Online Active-Set Method [45]

```
GIVEN a solution $(\mathbf{w}^*, \lambda^*)$ of QP$(\mathbf{x}_0)$, a working set $\mathcal{W}$, and a new initial value $\mathbf{x}_0^{\text{new}}$.
Set $\tau_{\max} = 0$
WHILE $\tau_{\max} < 1$ DO
    1. $\Delta\mathbf{x}_0 := \mathbf{x}_0^{\text{new}} - \mathbf{x}_0$
       $\Delta g := g(\mathbf{x}_0^{\text{new}}) - g(\mathbf{x}_0) = \mathbf{F}^T \Delta\mathbf{x}_0$
       $\Delta\mathbf{b} := \mathbf{b}(\mathbf{x}_0^{\text{new}}) - \mathbf{b}(\mathbf{x}_0) = \overline{\mathbf{b}} + \mathbf{E}\Delta\mathbf{x}_0$
    2. Solve $\begin{bmatrix} \mathbf{Q} & \mathbf{G}^T \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{w}^* \\ \Delta\lambda^* \end{bmatrix} = \begin{bmatrix} -\Delta g \\ \Delta\mathbf{b} \end{bmatrix}$ for $(\Delta\mathbf{w}^*, \Delta\lambda^*)$
    3. $\tau_{\max}^{\Delta\mathbf{w}^*} \leftarrow \min\limits_{\substack{i \in \mathcal{I} \\ \mathbf{G}_i^T \Delta\mathbf{w}^* < \Delta b_i}} \dfrac{b_i(\mathbf{x}_0) - \mathbf{G}_i^T \mathbf{w}^*}{\mathbf{G}_i^T \Delta\mathbf{w}^* - \Delta b_i}$
       $\tau_{\max}^{\Delta\lambda^*} \leftarrow \min\limits_{\substack{i \in \mathcal{W} \\ \Delta\lambda_i < 0}} -\dfrac{\lambda_i^*}{\Delta\lambda_i}$
       $\tau_{\max} = \min\left\{1, \tau_{\max}^{\Delta\mathbf{w}^*}, \tau_{\max}^{\Delta\lambda^*}\right\}$
    4. $\tilde{\mathbf{x}}_0 \leftarrow \mathbf{x}_0 + \tau_{\max}\Delta\mathbf{x}_0$
       $\tilde{\mathbf{w}}^* \leftarrow \mathbf{w}^* + \tau_{\max}\Delta\mathbf{w}^*$
       $\tilde{\lambda}^* \leftarrow \lambda^* + \tau_{\max}\Delta\lambda^*$
    5. IF $\tau_{\max} = 1$
            Solution found $\mathbf{w}_{\text{new}}^* \leftarrow \tilde{\mathbf{w}}^*$, $\lambda_{\text{new}}^* \leftarrow \tilde{\lambda}^*$, $\mathcal{W}^{\text{new}} \leftarrow \mathcal{W}$
            RETURN $(\mathbf{w}_{\text{new}}^*, \lambda_{\text{new}}^*)$ and $\mathcal{W}^{\text{new}}$
            STOP!
       ELSEIF $\tau_{\max} = \tau_{\max}^{\Delta\lambda^*}$
            $\mathcal{W} \leftarrow \mathcal{W} \setminus \left\{ j \;\middle|\; \tau_{\max}^{\Delta\lambda^*} = -\dfrac{\lambda_j^*}{\Delta\lambda_j} \right\}$
       ELSEIF $\tau_{\max} = \tau_{\max}^{\Delta\mathbf{w}^*}$
            $\mathcal{W} \leftarrow \mathcal{W} \cup \left\{ j \;\middle|\; \tau_{\max}^{\Delta\mathbf{w}^*} = \dfrac{b_j(\mathbf{x}_0) - \mathbf{G}_j^T \mathbf{w}^*}{\mathbf{G}_j^T \Delta\mathbf{w}^* - \Delta b_j} \right\}$
    6. $\mathbf{x}_0 \leftarrow \tilde{\mathbf{x}}_0$ $\mathbf{w}^* \leftarrow \tilde{\mathbf{w}}^*$, $\lambda^* \leftarrow \tilde{\lambda}^*$
END WHILE
```

2.4.3 Optimal and Model Predictive Control

Optimal Control problems are a generalized form of variational problems by separating state and control variables and incorporating control constraints. According to [46], the introduction of optimal control problem goes back to 1950, when the maximum principle was proven by Lev Semyonovich Pontryagin[5] and his students, which provided the necessary optimality conditions for the control problems. Since then, optimal control was applied in various fields: in aerospace to address the aircraft minimum time to-climb problem [47], in economics and marketing to optimize the financing of firms and address the optimal advertising rate over time [48], in electric power systems to find the optimal torque and voltage control of a turbo alternator [49], in the automotive field to plan optimal trajectories [50]...

Let $\mathbf{x} \in \mathbb{R}^n$ be a state vector and $\mathbf{u} \in \mathbb{R}^m$ an input (or control) vector. Then, the general formulation of a continuous optimal control problem is given by [42]

$$\min_{\mathbf{u}} \; J(\mathbf{u}(t)) = \Phi\left(\mathbf{x}_f\right) + \int_{t_0}^{t_f} L\left(\mathbf{x}(t), \mathbf{u}(t)\right) \mathrm{d}t \tag{2.50a}$$

$$\text{subject to} \quad \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.50b}$$

$$g(\mathbf{x}_f) = 0, \tag{2.50c}$$

$$h(\mathbf{x}(t), \mathbf{u}(t)) \leq 0, \quad \forall t \in [t_0, t_f]. \tag{2.50d}$$

The cost function $J(\mathbf{u}(t))$ consists of the running cost $L\left(\mathbf{x}(t), \mathbf{u}(t)\right)$, which accumulates over time and penalizes undesirable states or control efforts, and of the terminal cost $\Phi\left(\mathbf{x}_f\right)$, that represents the penalty associated with the final state $\mathbf{x}_f$. The problem (2.50) finds the control input $\mathbf{u}(t)$ that minimizes the cost function $J(\mathbf{u}(t))$, and that transfers the system with the dynamics (2.50b) from the initial state $\mathbf{x}_0$ to the final state $\mathbf{x}_f = \mathbf{x}(t_f)$, taking into account the equality and inequality constraints (2.50c) and (2.50d) respectively.

A particular form of the optimal control problem (2.50) is the linear quadratic optimal control problem (2.51) for linear dynamic systems.

$$\min_{\mathbf{u}} \; J(\mathbf{u}(t)) = \mathbf{x}_f^T \mathbf{F}(t_f) \mathbf{x}_f + \int_{t_0}^{t_f} \left(\mathbf{x}(t)^T \mathbf{Q}(t) \mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}(t) \mathbf{u}(t) + 2\mathbf{x}(t)^T \mathbf{N}(t) \mathbf{u}(t)\right) \mathrm{d}t \tag{2.51a}$$

$$\text{subject to} \quad \dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.51b}$$

Here, $\mathbf{Q}$ is the state cost matrix, $\mathbf{R}$ the control cost matrix, $\mathbf{N}$ cross-term cost matrix and $\mathbf{F}$ the terminal cost matrix. If we assume that all matrices $\mathbf{A}, \mathbf{B}, \mathbf{Q}$ and $\mathbf{R}$ are constant and that the time horizon is infinite and starts at zero, we obtain the problem (2.52) referred to as the Linear Quadratic Regulator (LQR)

$$\min_{\mathbf{u}} \; J(\mathbf{u}(t)) = \frac{1}{2} \int_{t_0}^{t_f} \left(\mathbf{x}(t)^T \mathbf{Q}\, \mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}\, \mathbf{u}(t)\right) \mathrm{d}t \tag{2.52a}$$

$$\text{subject to} \quad \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0. \tag{2.52b}$$

5 Lev Semyonovich Pontryagin: September 3, 1908 (Moscow) – May 3, 1988 (Moscow), Russian mathematician known for his contributions to topology, algebra, and optimal control theory

The LQR (2.52) can be solved analytically, as it has the feedback

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t), \tag{2.53}$$

where

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}, \tag{2.54}$$

with $\mathbf{S}$ being the solution of the algebraic Ricatti equation

$$-\mathbf{SA} - \mathbf{A}^T\mathbf{S} + \mathbf{SBR}^{-1}\mathbf{B}^T\mathbf{S} - \mathbf{Q} = 0. \tag{2.55}$$

An effective strategy for control system design is Model Predictive Control (MPC) [51]. The basic idea of MPC is to predict the future behavior of the system and continuously adjust the applied control inputs to achieve a desired target. For this investigation, a series of optimization problems are formulated over a receding horizon. At each time step t, one optimal control problem is solved over a finite time interval to compute the optimal control inputs and corresponding states. We formulate the problem for discrete time steps in the interval $[t_k, t_{k+N-1}]$.

$$\min_{\substack{\mathbf{u}_k,\cdots,\mathbf{u}_{k+N},\\ \mathbf{x}_k,\cdots,\mathbf{x}_{k+N-1}}} J_{[t_k,t_{k+N-1}]} = \sum_{i=0}^{N-1} L\left(\mathbf{x}_{i+k}, \mathbf{u}_{i+k}\right) + \Phi\left(\mathbf{x}_{k+N-1}\right) \tag{2.56a}$$

$$\text{subject to} \quad \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \quad \mathbf{x}(t_k) = x_0, \tag{2.56b}$$

$$\mathbf{x}_{\min} \le \mathbf{x}_{k+i} \le \mathbf{x}_{\max}, \quad i = 0, \cdots, N \tag{2.56c}$$

$$\mathbf{u}_{\min} \le \mathbf{u}_{k+i} \le \mathbf{u}_{\max}, \quad i = 0, \cdots, N-1. \tag{2.56d}$$

The main difference between the (discretized) optimal control problem (2.50) and the MPC problem (2.56), is that once the optimization is computed, the first element of the optimal inputs, i.e. $\mathbf{u}_k$, is applied to the system. Then, the next optimal control problem is shifted in time by one step, i.e., we consider the time interval $[t_{k+1}, t_{k+N}]$ and solve (2.56) for $\mathbf{u}_{k+1}, \cdots, \mathbf{u}_{k+N+1}, \mathbf{x}_{k+1}, \cdots, \mathbf{x}_{k+N}$ and apply $\mathbf{u}_{k+1}$ to the system etc. This receding horizon in MPC allows the controller to update its predictions at each step. This makes it suitable for applications with moderate model uncertainty or disturbances [52].
However, the inter-dependency of the variables in the MPC problem introduces significant challenges. Each control input not only affects the immediate state but also influences future states and control actions. This coupling requires the optimization algorithm to account for the entire sequence of future states and inputs. Additionally, managing the constraints on both states and inputs is a complex task.

2.4.4 Interior-Point Methods

To ensure real-time feasibility and stability of the control system, primal-dual interior point methods have shown to be particularly effective. Many variants of primal-dual interior point methods exist [53]. In this thesis, we focus on a variant of Mehrotra's predictor-corrector method. We follow [54] to present the approach: let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and $g : \mathbb{R}^n \to \mathbb{R}$. Then given is the convex program

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{2.57a}$$
$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.57b}$$
$$g(\mathbf{x}) \leq 0, \tag{2.57c}$$

where $\mathbf{A}$ is a $p \times n$ matrix, $\mathbf{b}$ a p-dimensional vector, and $\mathbf{x}$ an n-dimensional vector.
Let $(\mathbf{x}, \lambda, \mu, \mathbf{s})$ be the *central path* for which the *relaxed* KKT optimality conditions of (2.57) hold, i.e.

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda, \mu) = \nabla_{\mathbf{x}} f(\mathbf{x}) + \mathbf{A}^T \lambda + \mathbf{G}(\mathbf{x})^T \mu = 0 \tag{2.58a}$$
$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{2.58b}$$
$$g(\mathbf{x}) + \mathbf{s} = 0, \tag{2.58c}$$
$$s_i \mu_i = \tau, \quad \forall i = 1, \cdots, m. \tag{2.58d}$$
$$(\mathbf{s}, \mu) > 0, \tag{2.58e}$$

where $\mathbf{G}$ is the $m \times n$ Jacobian of the inequality constraints g, $\mathbf{s} \geq 0$ is an m-dimensional slack variable for $g(\mathbf{x}) \leq 0$, and $\tau > 0$ is the path parameter. Hence, the optimal central path $(\mathbf{x}^*, \lambda^*, \mu^*, \mathbf{s}^*)$ is computed by iteratively solving (2.58) for $\tau \to 0$.
The average value of the complementarity condition (2.58d) is called *duality measure* and is given by

$$\gamma = \frac{\mathbf{s}^T \mu}{m}. \tag{2.59}$$

We set

$$\mathbf{S} = \operatorname{diag}(s_1, \cdots, s_m), \tag{2.60}$$

$$\mathbf{M} = \operatorname{diag}(\mu_1, \cdots, \mu_m), \tag{2.61}$$

and

$$\mathbf{H}(\mathbf{x}, \mu) = \nabla^2 f(\mathbf{x}) + \sum_{i=1}^{m} \mu_i \nabla^2 g_i(\mathbf{x}). \tag{2.62}$$

Let

$$\nu = \sigma \gamma \mathbf{1}, \tag{2.63}$$

be a parameter that allows to modify the search direction in a Newton-Step. Then, we approximate (2.58) linearly around $(\mathbf{x}_k, \lambda_k, \mu_k, \mathbf{s}_k)$ for an iteration step k. The primal-dual interior point method is summarized in Algorithm 3.

Algorithm 3: Primal-dual interior point method [54]

```
GIVEN initial values $(\mathbf{x}_0, \lambda_0, \mu_0, \mathbf{s}_0) > 0$ and a centering parameter $0 < \sigma < 1$.
FOR $k = 0, 1, 2, \cdots$ DO
    1. $\gamma_k \leftarrow \frac{\mathbf{s}_k^T \mu_k}{m}$
    2. SOLVE $\begin{bmatrix} \mathbf{H}(\mathbf{x}_k, \mu_k) & \mathbf{A}^T & \mathbf{G}(\mathbf{x}_k)^T & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ \mathbf{G}(\mathbf{x}_k) & 0 & 0 & \mathbb{I} \\ 0 & 0 & \mathbf{S} & \mathbf{M} \end{bmatrix} \cdot \begin{bmatrix} \Delta \mathbf{x}_k \\ \Delta \lambda_k \\ \Delta \mu_k \\ \Delta \mathbf{s}_k \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_k, \lambda_k, \mu_k) \\ \mathbf{A}\mathbf{x}_k - \mathbf{b} \\ g(\mathbf{x}_k) + \mathbf{s}_k \\ \mathbf{S}\mu_k - \nu_k \end{bmatrix}$
       for $(\Delta \mathbf{x}_k, \Delta \lambda_k, \Delta \mu_k, \Delta \mathbf{s}_k)$
    3. Choose the step size $h_k$ such that $s_{k+1}, \mu_{k+1} > 0$.
    4. $(\mathbf{x}_{k+1}, \lambda_{k+1}, \mu_{k+1}, \mathbf{s}_{k+1}) \leftarrow (\mathbf{x}_k, \lambda_k, \mu_k, \mathbf{s}_k) + h_k (\Delta \mathbf{x}_k, \Delta \lambda_k, \Delta \mu_k, \Delta \mathbf{s}_k)$
END FOR
```

The Mehrotra-Type predictor corrector method, introduced in [55], is a modified version of the primal-dual interior point method presented in Algorithm 3. The main idea is that the complete search direction is given by a combination of a *predictor* direction and a *corrector* direction within a primal-dual interior-point algorithm.
The predictor is given by solving step 2 for $\nu = 0$. The corrector is given by solving step 2 with the right hand side $\begin{bmatrix} 0 & 0 & 0 & -e \end{bmatrix}^T$, i.e., by performing an unmodified Newton-Step. Thus, the predictor-corrector approach requires solving the linear system in step 2 for $\nu = \sigma\gamma\mathbf{1} - e$ [54]. The method is illustrated in Algorithm 4

Algorithm 4: Mehrotra primal-dual interior point method [54]

```
GIVEN initial values $(\mathbf{x}_0, \lambda_0, \mu_0, \mathbf{s}_0) > 0$ and $\beta < 1$ (for numerical stability).
FOR $k = 0, 1, 2, \cdots$ DO
    1. $\gamma_k \leftarrow \frac{\mathbf{s}_k^T \mu_k}{m}$
    2. SOLVE $\begin{bmatrix} \mathbf{H}(\mathbf{x}_k, \mu_k) & \mathbf{A}^T & \mathbf{G}(\mathbf{x}_k)^T & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ \mathbf{G}(\mathbf{x}_k) & 0 & 0 & \mathbb{I} \\ 0 & 0 & \mathbf{S} & \mathbf{M} \end{bmatrix} \cdot \begin{bmatrix} \widetilde{\Delta\mathbf{x}}_k \\ \widetilde{\Delta\lambda}_k \\ \widetilde{\Delta\mu}_k \\ \widetilde{\Delta\mathbf{s}}_k \end{bmatrix} = - \begin{bmatrix} \nabla_\mathbf{x}\mathcal{L}(\mathbf{x}_k, \lambda_k, \mu_k) \\ \mathbf{A}\mathbf{x}_k - \mathbf{b} \\ g(\mathbf{x}_k) + \mathbf{s}_k \\ \mathbf{S}\mu_k \end{bmatrix}$
       for $\left(\widetilde{\Delta\mathbf{x}}_k, \widetilde{\Delta\lambda}_k, \widetilde{\Delta\mu}_k, \widetilde{\Delta\mathbf{s}}_k\right)$
    3. $h_k \leftarrow \max\left\{h \in [0,1] \,\middle|\, s_k + h\widetilde{\Delta s}_k \geq 0,\ \mu_k + h\widetilde{\Delta\mu}_k \geq 0\right\}$
    4. $\tilde{\gamma}_k \leftarrow (s_k + h_k\widetilde{\Delta s}_k)^T(\mu_k + h_k\widetilde{\Delta\mu}_k)/m$
    5. $\sigma_k \leftarrow (\tilde{\gamma}_k/\gamma_k)^3$
    6. SOLVE $\begin{bmatrix} \mathbf{H}(\mathbf{x}_k, \mu_k) & \mathbf{A}^T & \mathbf{G}(\mathbf{x}_k)^T & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ \mathbf{G}(\mathbf{x}_k) & 0 & 0 & \mathbb{I} \\ 0 & 0 & \mathbf{S} & \mathbf{M} \end{bmatrix} \cdot \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\lambda_k \\ \Delta\mu_k \\ \Delta\mathbf{s}_k \end{bmatrix} = - \begin{bmatrix} \nabla_\mathbf{x}\mathcal{L}(\mathbf{x}_k, \lambda_k, \mu_k) \\ \mathbf{A}\mathbf{x}_k - \mathbf{b} \\ g(\mathbf{x}_k) + \mathbf{s}_k \\ \mathbf{S}\mu_k - \sigma_k\tilde{\gamma}_k\mathbf{1} + \widetilde{\Delta\mathbf{s}}_k\widetilde{\Delta\mu}_k \end{bmatrix}$
       for $(\Delta\mathbf{x}_k, \Delta\lambda_k, \Delta\mu_k, \Delta\mathbf{s}_k)$
    7. $h_k \leftarrow \max\{h \in [0,1] \mid s_k + h\Delta s_k \geq 0,\ \mu_k + h\Delta\mu_k \geq 0\}$
    8. $(\mathbf{x}_{k+1}, \lambda_{k+1}, \mu_{k+1}, \mathbf{s}_{k+1}) \leftarrow (\mathbf{x}_k, \lambda_k, \mu_k, \mathbf{s}_k) + \beta h_k(\Delta\mathbf{x}_k, \Delta\lambda_k, \Delta\mu_k, \Delta\mathbf{s}_k)$
END FOR
```

The computational costs of the algorithm are dominated by solving the linear systems in step 2 and 6. However, the coefficient matrix is the same for both steps. Hence, the most critical factor is the factorization of this matrix in step 2. Let us consider the linear system

$$\begin{bmatrix} \mathbf{H}(\mathbf{x}, \mu) & \mathbf{A}^T & \mathbf{G}(\mathbf{x})^T & 0 \\ \mathbf{A} & 0 & 0 & 0 \\ \mathbf{G}(\mathbf{x}) & 0 & 0 & \mathbb{I} \\ 0 & 0 & \mathbf{S} & \mathbf{M} \end{bmatrix} \cdot \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\lambda \\ \Delta\mu \\ \Delta\mathbf{s} \end{bmatrix} = \begin{bmatrix} r_S \\ r_E \\ r_I \\ r_C \end{bmatrix}, \tag{2.64}$$

for different right-hand sides $\begin{bmatrix} r_S & r_E & r_I & r_C \end{bmatrix}^T$. By eliminating $\Delta\mu$ and $\Delta\mathbf{s}$ from the equation, by using

$$\Delta\mathbf{s} = \mathbf{M}^{-1}(r_C - \mathbf{S}\Delta\mu), \tag{2.65}$$

and

$$\Delta\mu = -\mathbf{S}^{-1}\mathbf{M}\left(r_I - \mathbf{M}^{-1}r_C - \mathbf{G}(\mathbf{x})\Delta\mathbf{x}\right), \tag{2.66}$$

the system (2.64) can be expressed in the compact form [54]

$$\begin{bmatrix} \varphi & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} r_S \\ r_E + \mathbf{G}(\mathbf{x})^T \mathbf{S}^{-1} \mathbf{M} r_I - \mathbf{G}(\mathbf{x})^T \mathbf{S}^{-1} r_c \end{bmatrix}, \tag{2.67}$$

where

$$\varphi = \mathbf{H}(\mathbf{x}, \mu) + \mathbf{G}(\mathbf{x})^T \mathbf{S}^{-1} \mathbf{M} \mathbf{G}(\mathbf{x}). \tag{2.68}$$

For practical applications, φ can be assumed as positive definite. This allows the elimination of $\Delta\mathbf{x}$ from the equation by using

$$\Delta \mathbf{x} = \varphi^{-1} \left(r_S - \mathbf{A}^T \Delta \lambda \right). \tag{2.69}$$

Hence, (2.67) can be rewritten as

$$\Lambda \Delta \lambda = \zeta, \tag{2.70}$$

where

$$\Lambda = \mathbf{A} \varphi^{-1} \mathbf{A}^T, \tag{2.71}$$

and

$$\zeta = -\left(r_E + \mathbf{G}(\mathbf{x})^T \mathbf{S}^{-1} \mathbf{M} r_I - \mathbf{G}(\mathbf{x})^T \mathbf{S}^{-1} r_C - \varphi^{-1} r_S \right). \tag{2.72}$$

Thus, solving (2.64) is equivalent to solving (2.70) and subsequently computing (2.65), (2.66) and (2.69). A standard method to solve this system is by using the Cholesky Decomposition (e.g., from [41]) given in Algorithm 5

Algorithm 5: Cholesky Decomposition $A = LL^T$

```
GIVEN $n$-dimensional positive definite matrix $A$, with entries $a_{ij}$, $i, j = 1, \cdots, n$.
FOR $k = 1, \cdots, n$ DO
    $l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{k,j}^2}$
    FOR $i = k + 1, \cdots, n$ DO
        $l_{i,k} = \left( A_{i,k} - \sum_{j=1}^{k-1} l_{i,j} l_{k,j} \right)$
    END FOR
END FOR
RETURN $L$ with entries $l_{i,j}$, $i, j = 1, \cdots, n$.
```

We aim to solve multistage control problems as in (2.56). Thus, we follow [54] to show the applicability of the Mehrotra primal-dual interior point method to a general multistage problem of the form

$$\min_{\mathbf{x}_k} \sum_{k=0}^{N} f_k(\mathbf{x}_k) \tag{2.73a}$$

$$\text{subject to} \quad g_k(\mathbf{x}_k) \leq 0, \qquad k = 0, \cdots, N, \tag{2.73b}$$

$$\mathbf{D}_0 \mathbf{x}_0 - \mathbf{c}_0 = 0, \tag{2.73c}$$

$$\mathbf{C}_{k-1} \mathbf{x}_{k-1} + \mathbf{D}_k \mathbf{x}_k - \mathbf{c}_k = 0, \quad k = 1, \cdots, N, \tag{2.73d}$$

The elements in (2.64) become

$$\mathbf{x} = [\mathbf{x}_0, \cdots, \mathbf{x}_N], \lambda = [\lambda_0, \cdots, \lambda_N], \mu = [\mu_1, \cdots, \mu_N], \mathbf{s} = [\mathbf{s}_1, \cdots, \mathbf{s}_N], \tag{2.74}$$

$$\nabla f(\mathbf{x}) = [\nabla f_0(\mathbf{x}_0), \cdots, \nabla f_N(\mathbf{X}_N)], \tag{2.75}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_0 & 0 & \cdots & \cdots & 0 \\ \mathbf{C}_0 & \mathbf{D}_1 & 0 & \cdots & \vdots \\ 0 & \mathbf{C}_1 & \mathbf{D}_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & \mathbf{C}_{N-1} & \mathbf{D}_N \end{bmatrix}, \tag{2.76}$$

$$\mathbf{G}(\mathbf{x}) = \begin{bmatrix} \nabla g_0(\mathbf{x}_0)^T & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \nabla g_N(\mathbf{x}_N)^T \end{bmatrix}, \tag{2.77}$$

$$\mathbf{b} = [\mathbf{c}_0, \cdots, \mathbf{c}_N], \tag{2.78}$$

$$g(\mathbf{x}) = [g_0(\mathbf{x}_0), \cdots, g_N(\mathbf{x}_N)] \tag{2.79}$$

and

$$\mathbf{H}(\mathbf{x}, \mu) = \begin{bmatrix} \mathbf{H}_0(\mathbf{x}_0, \mu_0) & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \mathbf{H}_N(\mathbf{x}_N, \mu_N) \end{bmatrix}, \tag{2.80}$$

where

$$\mathbf{H}_k(\mathbf{x}_k, \mu_k) = \nabla^2 f_k(\mathbf{x}_k) + \sum_{i=1}^{m} \mu_{ki} \nabla^2 g_{ki}(\mathbf{x}_k). \tag{2.81}$$

Let

$$\varphi_k = \mathbf{H}_k(\mathbf{x}_k, \mu_k) + \nabla g_k(\mathbf{x}_k)^T \mathbf{S}^{-1} \mathbf{M} \nabla g_k(\mathbf{x}_k). \tag{2.82}$$

Then, the coefficient matrix in (2.70) becomes

$$\Lambda = \begin{bmatrix} \Lambda_{0,0} & \Lambda_{0,1} & 0 & \cdots & 0 \\ \Lambda_{0,1}^T & \Lambda_{1,1} & \Lambda_{1,2} & \cdots & \vdots \\ 0 & \Lambda_{1,2}^T & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \Lambda_{N-1,N} \\ 0 & \cdots & 0 & \Lambda_{N-1,N}^T & \Lambda_{N,N} \end{bmatrix}, \tag{2.83}$$

with

$$\Lambda_{0,0} = \mathbf{D}_0 \varphi_0^{-1} \mathbf{D}_0^T, \tag{2.84a}$$

$$\Lambda_{k,k} = \mathbf{C}_{k-1} \varphi_{k-1}^{-1} \mathbf{C}_{k-1}^T + \mathbf{D}_k \varphi_k^{-1} \mathbf{D}_k^T, \qquad k = 1, \cdots, N, \tag{2.84b}$$

$$\Lambda_{k,k+1} = \mathbf{D}_k \varphi_k^{-1} \mathbf{C}_k^T, \qquad k = 0, \cdots, N-1. \tag{2.84c}$$

To calculate Λ efficiently, we first compute the Cholesky decomposition of $\varphi_k = L_k L_k^T$, then we solve

$$V_k L_k^T = \mathbf{C}_k, \tag{2.85a}$$
$$W_k L_k^T = \mathbf{D}_k, \tag{2.85b}$$

by matrix forward substitution. Thus, we obtain the rectangular factorization

$$\mathbf{C}_{k-1}\varphi_{k-1}^{-1}\mathbf{C}_{k-1}^T = V_{k-1}V_{k-1}^T, \tag{2.86a}$$
$$\mathbf{D}_k\varphi_k^{-1}\mathbf{D}_k^T = W_k W_k^T, \tag{2.86b}$$
$$\mathbf{D}_k\varphi_k^{-1}\mathbf{C}_k^T = W_k V_k^T. \tag{2.86c}$$

In a last step, we compute the lower Cholesky factor L_Λ

$$L_\Lambda = \begin{bmatrix} L_{0,0} & 0 & 0 & \cdots & 0 \\ L_{1,0} & L_{1,1} & 0 & \cdots & 0 \\ 0 & L_{2,1} & L_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & L_{N,N-1} & L_{N,N} \end{bmatrix}, \tag{2.87}$$

of Λ by solving

$$\Lambda_{0,0} = L_{0,0}L_{0,0}^T \tag{2.88a}$$
$$\Lambda_{k,k} - L_{k,k-1}L_{k,k-1}^T = L_{k,k}L_{k,k}^T, \qquad k = 1, \cdots, N, \tag{2.88b}$$

by Cholesky decomposition, and solving

$$\Lambda_{k,k+1} = L_{k,k}L_{k+1,k}^T, \quad k = 0, \cdots, N-1, \tag{2.89}$$

by matrix forward substitution.

2.5 Model Reduction

Many problems in engineering and science are defined in high-dimensional spaces and can not be solved directly by traditional numerical discretization techniques, such as finite elements or finite differences (see [41]). It is not without reason that Richard Bellman[6] used the expression “the curse of dimensionality” in the 1940s, when considering recursive decision making problems in dynamic programming [56]. We follow the explanatory example cited in [57], and consider a simple model with dimension $D = 30$ and $M = 10^3$ discretization nodes for each space. One can imagine that the numerical complexity of this problem is 10^{90}. This number is frighteningly large, when we think that some estimates imply that there are roughly 10^{86} elementary particles in the universe! High-dimensional models can appear in

6 Richard Bellman: August 29, 1920 (Brooklyn) – March 19, 1984 (Los Angeles), American applied mathematician

kinetic theories of complex fluids, social dynamics, economic systems, vehicular traffic flow phenomena, biological systems, quantum chemistry (see [57] and the references therein) and in machine learning and data mining problems [58].
Not only the high-dimensionality makes some problems intractable, also some constraints such as real-time capability, can easily make standard methods unusable or the required computational efforts incredibly huge. In the field of simulation-based engineering sciences, real-time control of complex systems is a challenging necessity, especially when it is about identifying malfunction and reconfiguration of malfunctioning.
The basic objective in model order reduction is to reproduce accurately the dynamics of a large scale system, using low-order structures and thus, reduce the computation complexity of the problem.
The methods and concepts we provide in the following, build on fundamental numerical techniques, for which we refer the reader to [41], [59] and [60].

Definition 2.27. A **surrogate model** is an approximation model, that reproduces the behavior of a system as closely as possible, while being computationally cheaper to evaluate.

Definition 2.28. The **conjugate transpose** of a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ is the matrix $\mathbf{A}^* \in \mathbb{C}^{n\times m}$, given by transposing $\mathbf{A}$ and taking the complex conjugate of each entry:

$$\mathbf{A}^* = \bar{\mathbf{A}}^T.$$

2.5.1 Singular Value Decomposition

The Singular Value Decomposition (SVD) of a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ is given by the form

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^*, \tag{2.90}$$

where $\mathbf{U} \in \mathbb{C}^{m\times m}$ and $\mathbf{V} \in \mathbb{C}^{n\times n}$ are orthogonal matrices and Σ the diagonal matrix

$$\Sigma := \operatorname{diag}(\sigma_1, \cdots, \sigma_p) \in \mathbb{R}^{m\times n}, \qquad p = \min\{m,n\},$$

with

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0. \tag{2.91}$$

Definition 2.29. Given a SVD of a matrix $\mathbf{A}$ with a diagonal matrix Σ as in (2.90). Then the diagonal entries σ_i, $i = 1, \cdots, p$ are called **singular values** of $\mathbf{A}$.

Definition 2.30. Let $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^*$ be a singular value decomposition of $\mathbf{A}$ with singular values $\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$, $p = \min\{m,n\}$. We define $\mathbf{A}^\dagger := \mathbf{U}\Sigma^\dagger\mathbf{V}^*$ with $\Sigma^\dagger := \operatorname{diag}(\sigma_1^{-1}, \cdots, \sigma_r^{-1}, 0, \cdots, 0)$. Then $\mathbf{A}^\dagger$ is called **pseudoinverse** of $\mathbf{A}$.

Let $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^*$ be the SVD of $\mathbf{A} \in \mathbb{C}^{n\times m}$ and $\sigma_1, \cdots, \sigma_p$ its singular values. Let $\sigma_1 \geq \cdots \geq \sigma_r$ be the $r < p$ largest singular values of $\mathbf{A}$.
We define $\tilde{\mathbf{U}} \in \mathbb{C}^{n\times r}$ and $\tilde{\mathbf{V}} \in \mathbb{C}^{m\times r}$ the matrices containing the first r columns of $\mathbf{U}$ and $\mathbf{V}$ respectively and $\tilde{\Sigma} := \operatorname{diag}(\sigma_1, \cdots, \sigma_r) \in \mathbb{C}^{r\times r}$. We call *truncated Singular Value Decomposition* (or *reduced SVD*) the decomposition

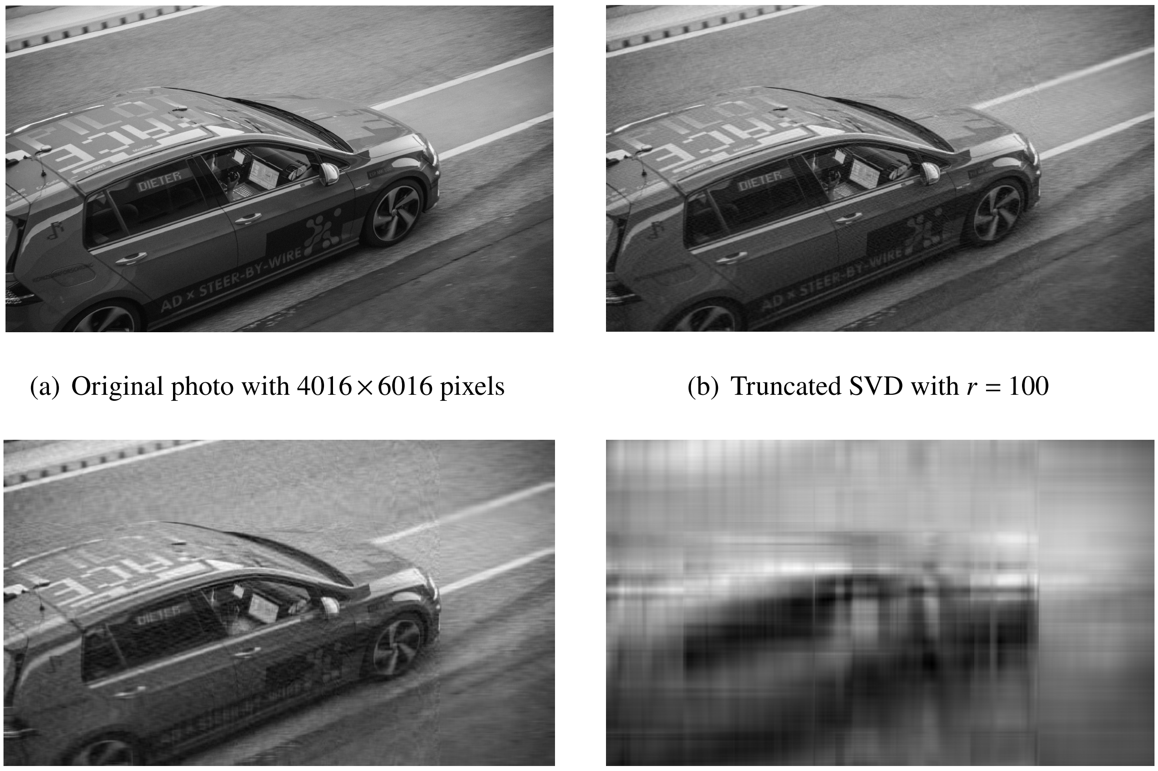
$$\mathbf{A} \approx \tilde{\mathbf{U}}\tilde{\Sigma}\tilde{\mathbf{V}}^*. \tag{2.92}$$

The parameter r is called *truncation value*.
In many applications large data matrices contains high-dimensional structures, but have dominant low dimensional patterns in the data. The truncated SVD performs a coordinate transformation from the high-dimensional to the low-dimensional pattern space. This plays an important role in the derivation of many model order reduction methods. To demonstrate the idea of the dimension reduction based on truncated SVD we consider the following example.

Example 2.2 (**Image Compression**). We consider a gray-scale photo of a vehicle illustrated in Figure 2.8(a) with 4016×6016 pixels. We describe the pixels of the picture as entries of a matrix $X \in \mathbb{R}^{n\times m}$, where $n = 4016$ and $m = 6016$ are the number of vertical and horizontal pixels, respectively. We compute the truncated SVD of the matrix X with different truncation values. The results are shown in Figure 2.8.
In Figure 2.8(b) the reconstructed image with truncation value $r = 100$ is remarkably close to the original, even though we are keeping only 2.49% of the eigenvalues. The picture in Figure 2.8(c) reconstructed with $r = 50$ eigenvalues, i.e. only 1.25% of the eigenvalues, shows a lower resolution of the picture. However, we can see the vehicle and its details. This means, that even with this small number of eigenvalues, the dominant pattern of the data are still identifiable. Figure 2.8(d) shows the vehicle reconstructed with a much smaller number of eigenvalues, only $r = 5$. Obviously, this approximation looses a lot of accuracy and precision. However, we can still recognize basic features of the vehicle in the picture.
This example illustrates that high-dimensional data can be explained by a few low-dimensional dominant patterns. The fact that, this identified patterns are extracted purely from data, makes the SVD a powerful technique in dimension reduction and serves as the basis for the derivation of many model order reduction methods.

2.5.2 Dynamic Mode Decomposition

Dynamic Mode Decomposition (DMD) was first introduced on 2008 in the fluid mechanics field by Schmid and Sesterhenn [61] and Schmid [62]. In general it can be understood as a modeling, prediction and control technique for dynamic systems. The method decomposes a complex system into simple spatiotemporal structures by identifying its low-order dynamics. The system state can be described by a superposition of empirically computed basis vectors, the POD modes, since in practice, the number of modes necessary to capture the gross behavior of a flow is often many orders of magnitude smaller than the state dimension of the system [63]. These reduced structures can be used for short-time future state prediction and



(a) Original photo with 4016 × 6016 pixels

(b) Truncated SVD with $r = 100$

(c) Truncated SVD with $r = 50$

(d) Truncated SVD with $r = 5$

Figure 2.8: Image compression of the original vehicle photo with 4016 × 6016 pixels using SVD with different truncation values $r \in \{100, 50, 5\}$.

control. The fact that the method is completely data driven and equation free made it gain popularity very quickly. It was successfully applied on various applications in different fields [63]: Erichson et al. applied DMD in the video processing field for background modeling [64]. Proctor et al. used DMD in discovering dynamic patterns from infectious disease data [65]. Kutz et al. cited different DMD applications in the fields of fluid dynamics, neuroscience and financial trading in his book [59]. Based on the standard DMD method, many alternative algorithms and DMD versions were derived: DMD with control, consistent DMD, compressed DMD, higher order DMD,... (see [63, 66, 67, 68, 69]).
In this section we will start with introducing the standard DMD algorithm following the presentation in [59]. We consider data collected from a dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, t), \tag{2.93}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector at time t and $\mathbf{f}$ the function describing the dynamics. Sampling the continuous-time dynamics from (2.93) every Δt in time, results in the discrete-time representation

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k), \tag{2.94}$$

where $\mathbf{x}_k = \mathbf{x}(k\Delta t)$ and $\mathbf{F}$ denotes the discrete-time flow map.
The state $\mathbf{x}$ is typically a quite large n-dimensional vector ($n \gg 1$). One can easily imagine

considering the discretization of a partial differential equation at a number of discrete spatial locations, or the total number of pixels in a given frame in video streams processing.
To construct or approximate solutions to the nonlinear equation (2.93) typically numerical methods are used, especially when the dynamic function $\mathbf{f}$ is unknown. However, the DMD method takes the equation-free perspective and approximates the dynamics based only on data measurements of the system, and uses this approximation to predict the future state.
The DMD method consists on approximating the system (2.93) by the approximate local linear dynamical system

$$\dot{\mathbf{x}} = \mathcal{A}\mathbf{x} \tag{2.95}$$

with initial condition $\mathbf{x}(0)$ and solution

$$\mathbf{x}(t) = \sum_{k=1}^{n} \phi_k \, \exp(\omega_k \, t) \, b_k = \Phi \exp(\Omega \, t) \, \mathbf{b}, \tag{2.96}$$

where ω_k and ϕ_k are the eigenvalues and eigenvectors of $\mathcal{A}$, and b_k the coordinates of $\mathbf{x}(0)$ in the eigenvector basis.
For the discrete-time system we sample (2.95) every Δt in time and get

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k, \tag{2.97}$$

where

$$\mathbf{A} = \exp(\mathcal{A}\Delta t). \tag{2.98}$$

Analogous to (2.96) the solution to the system (2.97) is given by

$$\mathbf{x}_k = \sum_{j=1}^{r} \phi_j \, \lambda_j^k \, b_j = \Phi \, \Lambda^k \, \mathbf{b}, \tag{2.99}$$

where $\mathbf{b}$ are the coefficients of the initial condition $\mathbf{x}_1$ in the eigenvector basis, so that $\mathbf{b} = \Phi^\dagger \mathbf{x}_1$, and λ_k and ϕ_k are the eigenvalues and eigenvectors of $\mathbf{A}$.

Definition 2.31. We call the eigenvectors ϕ_j, $j = 1, \cdots, r$, obtained by (2.99) **DMD modes**.

Definition 2.32. We call **snapshots** the solutions of an original time dependent equation system (ODEs or PDEs).

The DMD method is a data-driven approach. We collect snapshots $\mathbf{x}_k$ of the system at times t_k from $k = 1, 2, \cdots, m$, for a total of m measurement times.
The construction of the matrix $\mathbf{A}$ can be interpreted as solving the least-square minimization problem

$$\left\|\mathbf{x}_{k+1} - \mathbf{A}\mathbf{x}_k\right\|_2 \to \min \tag{2.100}$$

across all the snapshots.
To solve problem (2.100) we start with arranging the m snapshot sequences into two matrices $\mathbf{X}$ and $\mathbf{X}'$

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}, \tag{2.101a}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}. \tag{2.101b}$$

Then, we can rewrite (2.97) using the matrices from (2.101) as

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X}. \tag{2.102}$$

Definition 2.33. For a matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, the **Frobenius norm**, denoted $||\cdot||_F$, is defined by

$$\|\mathbf{X}\|_F = \sqrt{\sum_{j=1}^{n}\sum_{k=1}^{m} X_{jk}^2}.$$

Finding the Matrix $\mathbf{A}$ in equation (2.102), yields to solving the minimization problem

$$\|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F \to \min. \tag{2.103}$$

This means that the best-fit matrix $\mathbf{A}$, that solves (2.103), is given by

$$\mathbf{A} = \mathbf{X}'\mathbf{X}^\dagger. \tag{2.104}$$

Assuming that the data matrix $\mathbf{X}$ is high dimensional, i.e., the state dimension n is way larger than the number of snapshots m, yields to a high dimensional matrix $\mathbf{A}$. However, the rank of $\mathbf{A}$ is at most $m-1$. Instead of solving the problem for $\mathbf{A}$ directly, the DMD algorithm projects the data onto a low-rank subspace and solves for a low-dimensional matrix $\tilde{\mathbf{A}}$, without the need of explicitly computing $\mathbf{A}$.
Let n be the number of spatial points saved per time snapshot and m the number of snapshots. We arrange the snapshots into two matrices $\mathbf{X}$ and $\mathbf{X}'$ as in (2.101). The DMD algorithm is presented in Algorithm 6

Algorithm 6: DMD algorithm

1. Calculate $\mathbf{X} \approx \mathbf{U}\Sigma\mathbf{V}^*$ the reduced singular value decomposition of $\mathbf{X}$, with $\mathbf{U} \in \mathbb{C}^{n\times r}$, $\Sigma \in \mathbb{C}^{r\times r}$ and $\mathbf{V} \in \mathbb{C}^{m\times r}$, and with truncation value r.

2. Compute $\mathbf{A} = \mathbf{X}'\mathbf{V}\Sigma^{-1}\mathbf{U}^*$. For reasons of efficiency, instead compute $\widetilde{\mathbf{A}} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\Sigma^{-1}$ the $r \times r$ projection of the full matrix $\mathbf{A}$ onto POD modes.

3. Compute the eigendecomposition of $\widetilde{\mathbf{A}}$: $\widetilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\Lambda$, where the columns of $\mathbf{W}$ are eigenvectors and Λ is a diagonal matrix containing the corresponding eigenvalues λ_k.

4. The eigendecomposition of $\mathbf{A}$ may be reconstructed from $\mathbf{W}$ and Λ. The eigenvalues of $\mathbf{A}$ are given by Λ and the eigenvectors by columns of $\Phi = \mathbf{X}'\mathbf{V}\Sigma^{-1}\mathbf{W}$.

The DMD algorithm makes a low-rank approximation of the linear mapping that best approximates the nonlinear dynamics of the collected data of the system. Using the iteration

(2.97), a prediction of the future state of the system can be achieved for all time.
Many applications in the engineering and science field deal with dynamical systems with applied external control. In [68] Proctor et al. introduce the Dynamic Mode Decomposition with control (DMDc) algorithm. It takes advantage of the standard DMD algorithm and is additionally capable of distinguishing between the intern dynamics of the system and the effects of external forcing.
In addition to the state snapshots $\mathbf{X}$ and $\mathbf{X}'$ collected for DMD, we construct a matrix Υ containing input snapshots

$$\Upsilon = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_{m-1} \\ | & | & & | \end{bmatrix}. \tag{2.105}$$

Then, the controlled system can be written in terms of snapshot matrices as

$$\mathbf{X}' = \mathbf{A}\mathbf{X} + \mathbf{B}\Upsilon. \tag{2.106}$$

We distinguish two cases: either the matrix $\mathbf{B}$ is known or unknown. In [59] both cases are described in detail. Since in general, for complex systems without well-defined governing equations the operator $\mathbf{B}$ is at most estimated, we will focus on the general case, where both operators $\mathbf{A}$ and $\mathbf{B}$ are unknown.
We start with reformulating (2.106) as

$$\mathbf{X}' = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \Upsilon \end{bmatrix} =: \mathbf{G}\,\Omega. \tag{2.107}$$

The matrix $\mathbf{G}$ contains only unknown operators. The matrix Ω contains only data matrices. An analogy could be drawn between (2.102) and (2.107). Here, a best-fit solution of the operator $\mathbf{G}$ is sought, which solves the minimization problem

$$\left\| \mathbf{X}' - \mathbf{G}\Omega \right\|_F \to \min. \tag{2.108}$$

The DMDc algorithm is presented in Algorithm 7.

Algorithm 7: DMDc algorithm

```
1. Collect and construct the snapshot matrices $\mathbf{X}$ and $\mathbf{X}'$ as in (2.101) and $\Upsilon$ as in (2.105),
   and construct the matrix $\Omega = \begin{bmatrix} \mathbf{X} \\ \Upsilon \end{bmatrix}$.
2. Compute the reduced SVD of the input space $\Omega \approx \tilde{\mathbf{U}}\tilde{\Sigma}\tilde{\mathbf{V}}^*$, with truncation value $p$.
3. Compute the reduced SVD of the output space $\mathbf{X}' \approx \hat{\mathbf{U}}\hat{\Sigma}\hat{\mathbf{V}}^*$, with truncation value $r$ and
   $\hat{\mathbf{U}} \in \mathbb{R}^{n\times r}$, $\hat{\Sigma} \in \mathbb{R}^{r\times r}$, and $\hat{\mathbf{V}}^* \in \mathbb{R}^{r\times(m-1)}$.
4. Compute the approximation of the operator $\mathbf{G} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} \in \mathbb{R}^{n\times(n+q)}$
       $\tilde{\mathbf{A}} = \hat{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\Sigma}^{-1}\tilde{\mathbf{U}}_1^*\hat{\mathbf{U}} \in \mathbb{R}^{r\times r} \qquad \text{and} \qquad \tilde{\mathbf{B}} = \hat{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\Sigma}^{-1}\tilde{\mathbf{U}}_2^* \in \mathbb{R}^{r\times q}.$
   where $\tilde{\mathbf{U}}_1 \in \mathbb{R}^{n\times p}$, $\tilde{\mathbf{U}}_2 \in \mathbb{R}^{l\times p}$ and $\tilde{\mathbf{U}}^* = \begin{bmatrix} \tilde{\mathbf{U}}_1^* & \tilde{\mathbf{U}}_2^* \end{bmatrix}$.
5. Compute the eigendecomposition of $\tilde{\mathbf{A}}$: $\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\Lambda$.
6. The dynamic modes of $\mathbf{A}$ are given by $\Phi = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\Sigma}^{-1}\tilde{\mathbf{U}}_1^*\hat{\mathbf{U}}\mathbf{W}$.
```

3 Design Method for a Shared Control Framework

3.1 Related Work

To describe the interaction between human and machines, the term *Shared Control* was first used in 1978 by Sheridan and Verplank [70]. Sharing the control meant "... *both human and computer are active at the same time*". They also emphasized in their study, that there is a distinct difference between *sharing* and *trading* the control, which they defined by the computer being active at one time and the human at another. Since then, many studies have defined shared control in various ways. In [71] and [72], an overview of several definitions of shared control in the literature can be found: in the field of Brain-Computer-Interface (BCI), shared control refers to the continuous and immediate reflected input of both brain and computer [73]. In [74], shared control refers *only* to sharing the execution of the task, whereas the decision of which task to execute remains the human's responsibility. In [75], the term shared control is used to describe cooperative control in general. In [76], shared control refers to the fourth Level of Automation (LoA), i.e., the computer suggests a decision option and the human carries out the action.
In the context of automated vehicles, various studies examined shared control applications, especially for lane-keeping and lateral stability control, up to autonomous driving [77, 78, 79, 80, 81]. When the goal is to design a steering shared control system, that meets the driver's expectations and increases his acceptance and willingness to use the system, different design decisions are involved. On one hand, the steering mechanism plays a crucial role in defining the degrees of freedom in the driver's interaction with the steering wheel. In this context, two main steering mechanisms exist: coupled steering systems, where the steering wheel and the front axle are mechanically connected [19, 80, 82], and decoupled (steer-by-wire) systems, which rely on electronic controls [83, 84, 85]. On the other hand, the choice of the control method is essential. Especially Model Predictive Control (MPC) frequently emerges in the literature as a highly beneficial approach for steering assistance systems, particularly because it allows for personalization. In [78], an MPC-based shared control framework for lane-keeping is presented as a constrained optimization problem. The system automatically keeps the vehicle in its lane. Whenever the driver intervenes, the framework guarantees a smooth transition of control from the system to the driver. Other approaches integrate the driver's neuromuscular response at the steering wheel into the MPC design to improve driver's acceptance and comfort [17, 19, 86].
In this chapter, we present the design of a shared control framework, where the driver and the assistance system simultaneously control the vehicle's lateral motion. Through the haptic interface, i.e. the steering wheel, the interaction strategy of the driver is detected and used to adapt the behavior of the controller to meet his expectations of the system's intervention.

3.2 System Design Overview

We design a shared control system where a vehicle is controlled by a driver and a steering assistance system. The assistance system aims to lead the vehicle to follow a fixed path target. The driver can intervene at any time by taking control of the steering wheel. He can override the system, steer with the system, or allow it to take full control. An overview of the shared control framework is illustrated in Figure 3.1.

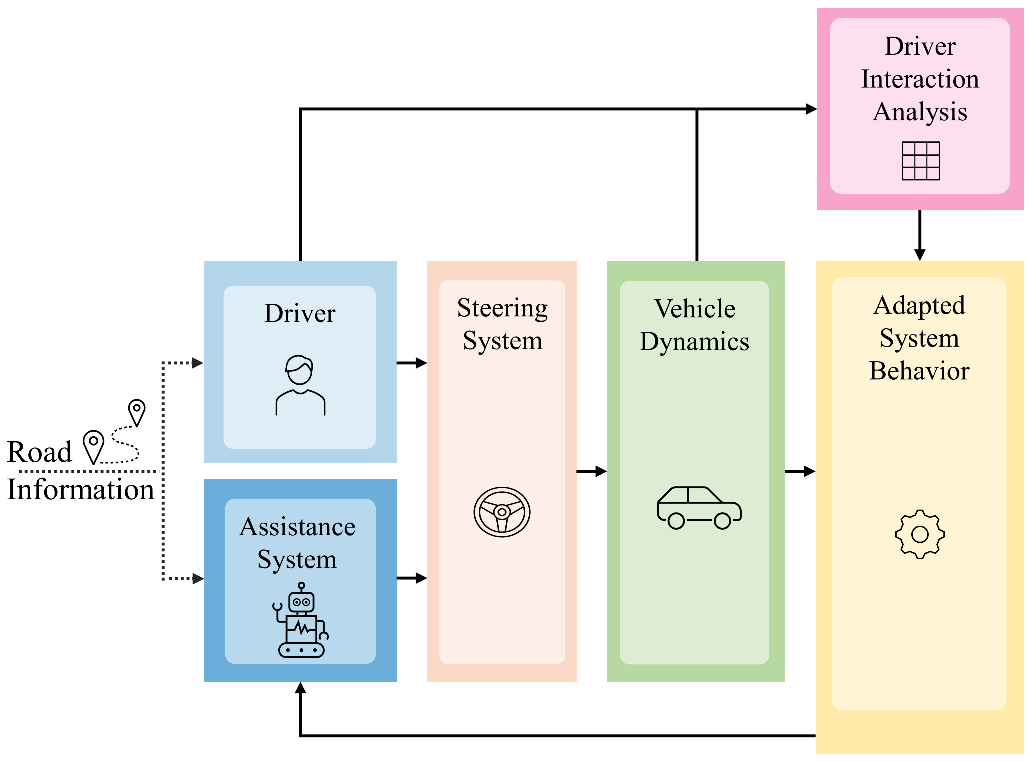


Figure 3.1: Overview of shared control framework

During the drive, the inputs of driver and assistance system are continuously collected as part of an interaction analysis. This analysis aims to understand and evaluate the interaction strategy chosen by the driver, *while* he shares the driving task with the assistance system. Once detected, the interaction strategy is used to adjust the assistance system behavior, to ensure a personalized driver support that matches driver's expectation of the system.

3.3 Dynamics Modeling

We design an intelligent vehicle system that allows sharing the lateral control between the driver and an assistance system. On one hand, a highly responsive and accurate steering system is needed to ensure that both the driver and the assistance system can influence the

vehicle's direction. On the other hand, an accurate modeling of the vehicle dynamics is necessary to accurately predict the vehicle's response to the steering inputs.
We assume that the vehicle motion can be modeled as a linear dynamical system, with the

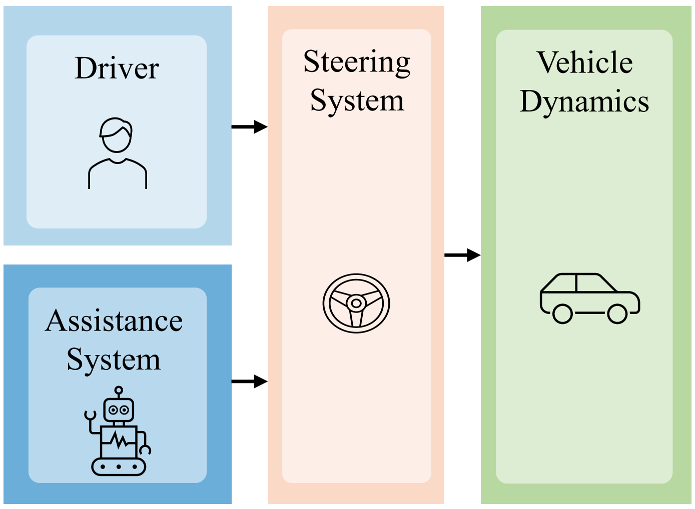


Figure 3.2: Overview of dynamical system including vehicle, driver and assistance system

time continuous state-space representation

$$\dot{\mathbf{x}}(t) = \tilde{\mathbf{A}}\mathbf{x}(t) + \tilde{\mathbf{B}}\, g(\mathbf{v}(t), \mathbf{w}(t)), \tag{3.1}$$

and its time discrete representation

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\, g(\mathbf{v}_k, \mathbf{w}_k), \tag{3.2}$$

where $\mathbf{x}_k = \mathbf{x}(t_k) \in \mathbb{R}^n$ represents the vehicle's state vector at time step t_k. The vectors $\mathbf{v} \in \mathbb{R}^m$ and $\mathbf{w} \in \mathbb{R}^m$ denote the two different control inputs of the driver and the assistance system. The function g combines the effect of both control inputs. To simplify the notation, we set

$$\mathbf{u} := g(\mathbf{v}, \mathbf{w}), \tag{3.3}$$

and rewrite (3.2) as

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\, \mathbf{u}_k. \tag{3.4}$$

The model is able to analyze and predict vehicle behavior.
In consideration of real-world sampled data and numerical simulation methods, all further analysis will be developed and evaluated in discrete-time.

3.4 Driver Interaction Design

To design the interaction between the driver and the assistance system, we propose to analyze this interplay based on the input data of the driver, the interventions of the assistance system and the vehicle state.

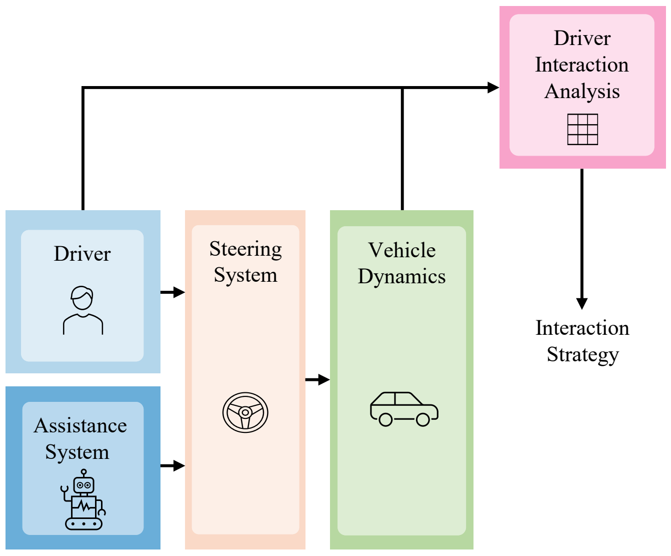


Figure 3.3: Overview of driver interaction analysis design with inputs and outputs.

We identify different interaction strategies of driver sharing the driving task with an assistance system. For this, we develop a classifier that can accurately categorize the interactions based on predefined criteria. The classifier will be trained using actual and historical data collected from real-driving scenarios. Once trained, the classifier can predict the current driver interaction strategy based on his recent driving history.
Let $\mathcal{C}$ be the classifier function that maps a history of states and control inputs $\mathcal{H}_k$, to a corresponding class $z_k \in \mathcal{Z}$ at time step k, where $\mathcal{Z}$ denotes the set of all available classes. We write z_k as

$$z_k = \mathcal{C}(\mathcal{H}_k) = \mathcal{C}(\mathbf{x}_{k-n_H}, \cdots, \mathbf{x}_{k-1}, \mathbf{u}_{k-n_H}, \cdots, \mathbf{u}_{k-1}), \tag{3.5}$$

where n_H is the length of the history window $\mathcal{H}_k$, and $\mathbf{x}_{k-n_H}, \cdots, \mathbf{x}_{k-1}$ and $\mathbf{u}_{k-n_H}, \cdots, \mathbf{u}_{k-1}$ are the states and control inputs respectively from time step $k - n_H$ to $k-1$.

3.5 Adapted System Behavior Design

The assistance system is designed in a way, that allows the adaptation to the intervention strategy of the driver. For this investigation a model predictive control problem is formulated to compute the suitable control strategy of the assistance system.

To adapt the control strategy to the driver's interaction, we integrate the output z_k of a classifier $\mathcal{C}(\mathcal{H}_k)$ into a cost function J_c, by determining positive definite weighting matrices

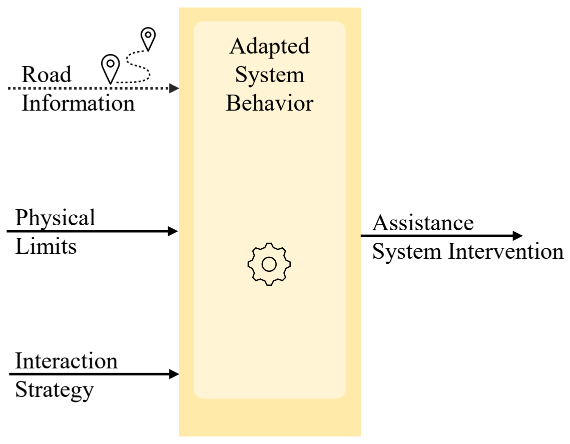


Figure 3.4: Overview of adapted system behavior design with inputs and outputs.

$Q(z_k)$, $R(z_k)$ and $Q_f(z_k)$ that depend on z_k and thus, allow the control strategy to account to each class. This leads to the classifier-dependent cost function

$$J_c = \sum_{k=0}^{N-1} \left(\mathbf{x}_k^T\, Q(z_k)\, \mathbf{x}_k + \mathbf{u}_k^T\, R(z_k)\, \mathbf{u}_k\right) + \mathbf{x}_N^T\, Q_f(z_k)\, \mathbf{x}_N, \tag{3.6a}$$

$$= \sum_{k=0}^{N-1} \left(\mathbf{x}_k^T\, Q(\mathcal{C}(\mathcal{H}_k))\, \mathbf{x}_k + \mathbf{u}_k^T\, R(\mathcal{C}(\mathcal{H}_k))\, \mathbf{u}_k\right) + \mathbf{x}_N^T\, Q_f(\mathcal{C}(\mathcal{H}_N))\, \mathbf{x}_N. \tag{3.6b}$$

To ensure the feasibility of the shared control system the vehicle state and the control inputs must obey physical constraints, to ensure reliable operation within the vehicle's physical limits. Additionally the vehicle must always remain within the road bounds.
Incorporating the driver's interaction strategy into the control algorithm requires classifier dependent state and control variables, i.e.,

$$\mathbf{x}_{\min}(z_k) \leq \mathbf{x}_k \leq \mathbf{x}_{\max}(z_k), \tag{3.7a}$$

$$\mathbf{u}_{\min}(z_k) \leq \mathbf{u}_k \leq \mathbf{u}_{\max}(z_k), \tag{3.7b}$$

where $\mathbf{u}_{\min}(z_k)$ and $\mathbf{u}_{\max}(z_k)$ represent the minimum and maximum allowable values for the classifier dependent control input $\mathbf{u}_k(z_k)$ and $\mathbf{x}_{\min}(z_k)$ and $\mathbf{x}_{\max}(z_k)$ the minimum and maximum allowable values for the system state $\mathbf{x}_k$.

3.6 Summary and Preview

A shared control framework for a vehicle that is managed by a driver and an assistance system is designed. The assistance system behavior is adjusted based on the analysis of the interaction between the two control inputs. An overview of the framework is illustrated in Figure 3.5. The controllers **v** and **w** simultaneously influence the system's dynamics. The inputs are analyzed and combined into a classifier function $\mathcal{C}$ that additionally depends on the system's state **x**. Based on this classifier, the control input **w** is adjusted to optimize the system's behavior.

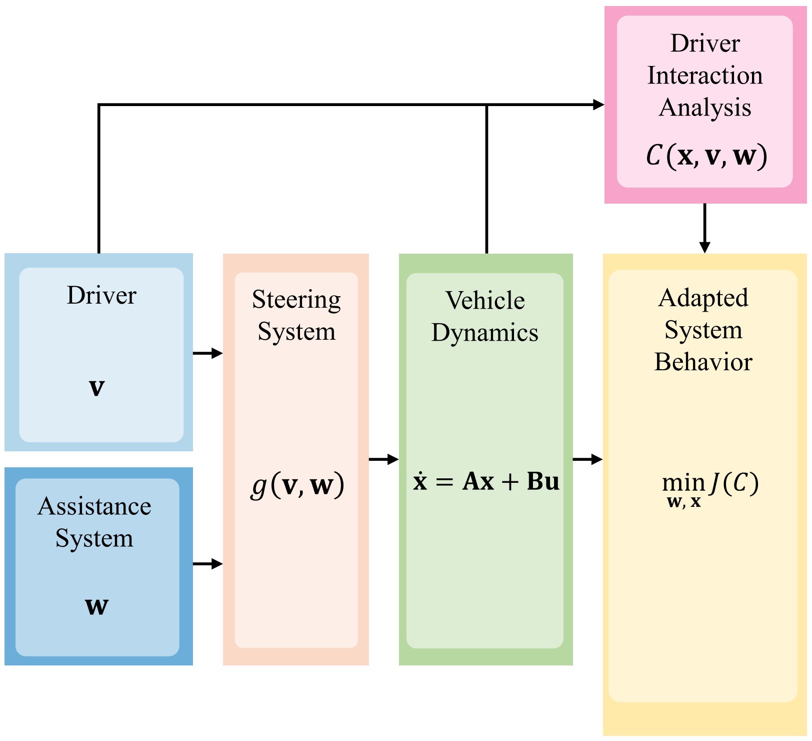

Driver Interaction Analysis
$C(\mathbf{x},\mathbf{v},\mathbf{w})$
Driver
$\mathbf{v}$
Assistance System
$\mathbf{w}$
Steering System
$g(\mathbf{v},\mathbf{w})$
Vehicle Dynamics
$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$
Adapted System Behavior
$\min_{\mathbf{w},\mathbf{x}} J(C)$

Figure 3.5: Overview of the system design

Considering (3.4), (3.6) and (3.7) the framework lead to the MPC formulation

$$\min_{\{(\mathbf{x}_k,\mathbf{u}_k)\}_{k=1}^{N}} \quad J_c(z_k) = \sum_{k=1}^{N} \left(\mathbf{x}_k^T\, Q(z_k)\, \mathbf{x}_k + \mathbf{u}_k^T\, R(z_k)\, \mathbf{u}_k\right) + \mathbf{x}_N^T\, Q_f(z_k)\, \mathbf{x}_N, \tag{3.8a}$$

$$\text{subject to} \quad \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\,\mathbf{u}_k, \tag{3.8b}$$

$$\mathbf{u}_k = g(\mathbf{v}_k, \mathbf{w}_k), \tag{3.8c}$$

$$\mathbf{x}_{\min} \le \mathbf{x}_k \le \mathbf{x}_{\max}, \tag{3.8d}$$

$$\mathbf{u}_{\min}(z_k) \le \mathbf{u}_k \le \mathbf{u}_{\max}(z_k). \tag{3.8e}$$

In the following chapters, the key components of the shared control framework will be explored in detail. Chapter 4 focus on the mathematical modeling of the vehicle and steering dynamics as well as the driver input. The introduction of the interaction strategies between driver and assistance system and the construction of the classifier is conducted in Chapter 5. In Chapter 6, concepts for the interaction strategy based behavior of the control system are presented.

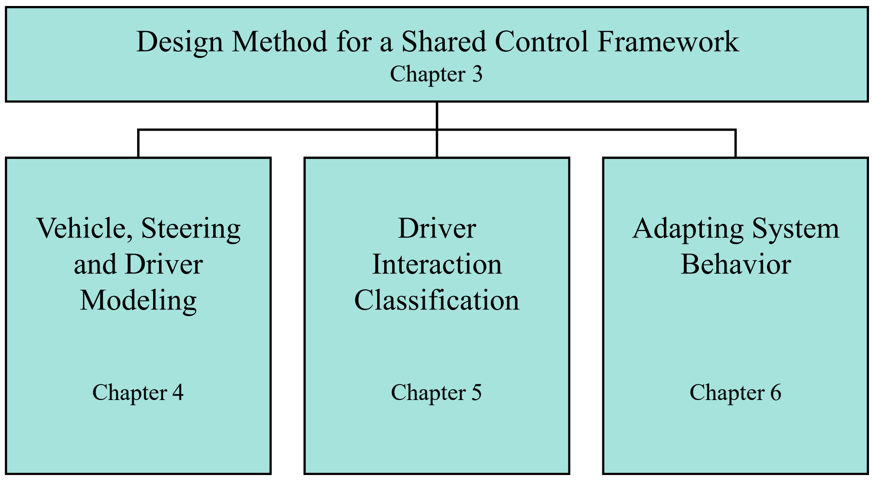

Design Method for a Shared Control Framework
Chapter 3
Vehicle, Steering and Driver Modeling
Chapter 4
Driver Interaction Classification
Chapter 5
Adapting System Behavior
Chapter 6

Figure 3.6: Key components of the shared control framework structured in the following chapters

4 Method for Vehicle, Steering and Driver Modeling

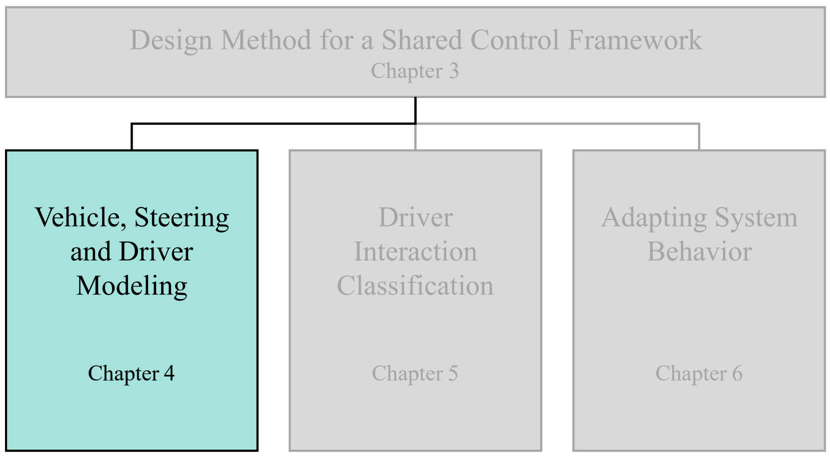

Design Method for a Shared Control Framework
Chapter 3
Vehicle, Steering and Driver Modeling
Chapter 4
Driver Interaction Classification
Chapter 5
Adapting System Behavior
Chapter 6

Figure 4.1: Scope of Chapter 4

In this chapter, we focus on the mathematical modeling of the various dynamics involved in the shared control framework. We start with presenting the vehicle model in Section 4.1. We choose to use a linear single-track model to capture the lateral vehicle motion. In Section 4.2, we briefly discuss the main concept of coupled and decoupled steering mechanisms. Finally, in Section 4.3, we present a driver-steering interaction model that captures the different torques involved at the steering wheel to model and analyze the driver torque. This model have been previously introduced and published by the author (El Amouri et al. [87]). The section mainly reiterates the methodologies detailed in the publication.

4.1 Vehicle Model

4.1.1 Single-Track Model

To analyze and predict vehicle behavior, we use a dynamic model for lateral vehicle motion known as single-track model, or bicycle model, with numerous references in the literature [88, 89, 90, 91]. The model describes the vehicle dynamics based on many simplifications, by mainly representing the front and rear tires as one single tire on each axle. Figure 4.2 illustrates a single-track model. The coordinate system (x, y) is located at the center of gravity of the vehicle, where x represents the longitudinal axis in the direction of the vehicle's movement, and y represents the lateral axis perpendicular to the direction of movement. The angle δ describes the steering angle on the front tire. It represents the angle between the direction of the front tire and the x-axis. The angles α_f and α_r are the front and rear slip angle, respectively. They describe the angle between the velocity vector v_f (respectively v_r) at the font (respectively rear) tire's contact patch and the direction the front (respectively rear) tire is pointing. The angle β represents the sideslip angle. It is the angle between the direction of the vehicle's movement at its center of gravity and the x-axis. For any chosen fixed reference, the angle χ represents the course angle, the angle between this reference and the direction of the vehicle's movement at its center of gravity. Then the heading (or yaw) angle ψ is defined as the sum

$$\psi = \chi + \beta. \tag{4.1}$$

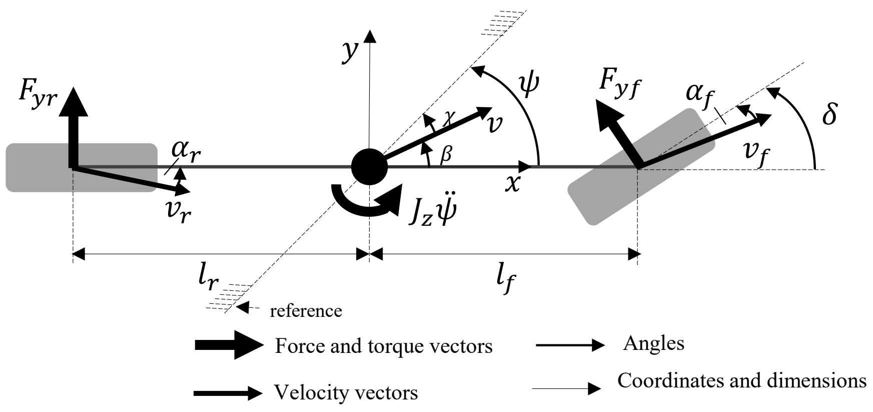


Figure 4.2: Single track model

The forces F_{yf} and F_{yr} represent the lateral tire forces of the front and rear tires respectively. They are assumed to be proportional to the front and rear slip angles α_f and α_r, respectively. Thus, they are given by

$$F_{yf} = -c_f \cdot \alpha_f, \tag{4.2}$$

and

$$F_{yr} = -c_r \cdot \alpha_r, \tag{4.3}$$

where c_f and c_r are called cornering stiffness of the front and rear tire respectively, and represent constant model parameters.

4.1.2 Equation of Motion

To derive the equations of the dynamical system we apply Newton's second law of linear motion along the y-axis [38]. Assuming a small steering angle ($\cos\delta = 1$) and a small sideslip angle (centrifugal force acts primarily in the y direction), yields to the equilibrium

$$F_{yf} + F_{yr} = m \cdot a_y, \tag{4.4}$$

where a_y represents the lateral acceleration of the vehicle at its center of gravity and $m > 0$ the vehicle's mass. It is given by the sum of the lateral acceleration $\dot{v}_y$ and the centripetal acceleration $v_x \dot{\psi}$, i.e.,

$$a_y = \dot{v}_y + v_x \dot{\psi}. \tag{4.5}$$

Inserting (4.2), (4.3) and (4.5) in (4.4) yields

$$-c_f \cdot \alpha_f - c_r \cdot \alpha_r = m \cdot (\dot{v}_y + v_x \dot{\psi}). \tag{4.6}$$

We assume that the vehicle moves forward with strictly positive longitudinal velocity, to ensure $v_x > 0$. Then, for small slip angles, α_f and α_r are written as

$$\alpha_f = \delta - \frac{v_y + l_f \dot{\psi}}{v_x}, \tag{4.7}$$

and

$$\alpha_r = \delta - \frac{v_y + l_r \dot{\psi}}{v_x}. \tag{4.8}$$

Then, (4.6) becomes

$$-c_f \cdot \left(\delta - \frac{v_y + l_f \dot{\psi}}{v_x}\right) - c_r \cdot \left(\delta - \frac{v_y + l_r \dot{\psi}}{v_x}\right) = m \cdot (\dot{v}_y + v_x \dot{\psi}). \tag{4.9}$$

Rearranging (4.9) to solve for $\dot{v}_y$, yields

$$\dot{v}_y = \frac{c_f + c_r}{m \cdot v_x} \cdot v_y + \left(\frac{-c_f l_f + c_r l_r}{m \cdot v_x} - v_x\right) \cdot \dot{\psi} - \frac{c_f}{m} \cdot \delta. \tag{4.10}$$

The equation for yaw dynamics is derived by applying Newton's second law of rotational motion on the vertical axis [38]:

$$J_z \cdot \ddot{\psi} = l_f \cdot F_{yf} - l_r \cdot F_{yr}, \tag{4.11}$$

where $J_z > 0$ is the moment of inertia of the vehicle and l_f (respectively l_r) is the distance between the front (respectively rear) tire and the center of gravity.
Inserting (4.2), (4.3), (4.7) and (4.8) in (4.11) and rearranging to solve for $\ddot{\psi}$ yields to

$$\ddot{\psi} = \frac{l_f \cdot c_f - l_r \cdot c_r}{J_z \cdot v_x} \cdot v_y + \frac{l_f^2 \cdot c_f - l_r^2 \cdot c_r}{J_z \cdot v_x} \cdot \dot{\psi} + \frac{-l_f \cdot c_f + l_r \cdot c_r}{J_z} \cdot \delta. \tag{4.12}$$

For lane keeping scenarios the vehicle is represented in terms of position and heading angle error with respect to a reference. We define the rate of change of the desired heading angle as

$$\dot{\psi}_{des} = v_x \cdot \kappa, \tag{4.13}$$

where κ represents the curvature of the road and is computed by

$$\kappa = \frac{1}{R}, \tag{4.14}$$

with R being the large constant radius of the road, the vehicle is traveling on.
Let $\Delta\psi$ be the heading angle error

$$\Delta\psi = \psi - \psi_{des}. \tag{4.15}$$

Then, $\Delta\dot{\psi}$ is given by

$$\Delta\dot{\psi} = \dot{\psi} - v_x \cdot \kappa. \tag{4.16}$$

Let Δd be the distance error describing the lateral deviation of the center of gravity from the reference. Using (4.5) and (4.15), we write

$$\Delta\dot{d} = v_y + v_x \Delta\psi. \tag{4.17}$$

To sum up, let $\mathbf{x}$ be the state vector of the dynamical system with

$$\mathbf{x} = \begin{bmatrix} \Delta d \\ \Delta\psi \\ \Delta\dot{\psi} \\ v_y \end{bmatrix}, \tag{4.18}$$

and let $\mathbf{u} = \delta$ be the control variable. Then, using (4.10), (4.12), (4.16) and (4.17), the controlled dynamical system for the vehicle model is given by

$$\dot{\mathbf{x}} = \widetilde{\mathbf{A}}\mathbf{x} + \widetilde{\mathbf{B}}\mathbf{u} + \widetilde{\mathbf{r}}, \tag{4.19}$$

where

$$\widetilde{\mathbf{A}} = \begin{bmatrix} 0 & v_x & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \dfrac{l_f \cdot c_f - l_r \cdot c_r}{J_z \cdot v_x} & \dfrac{l_f^2 \cdot c_f - l_r^2 \cdot c_r}{J_z \cdot v_x} \\ 0 & 0 & \dfrac{c_f + c_r}{m \cdot v_x} & \dfrac{-c_f l_f + c_r l_r}{m \cdot v_x} - v_x \end{bmatrix}, \tag{4.20}$$

$$\widetilde{\mathbf{B}} = \begin{bmatrix} 0 \\ 0 \\ \dfrac{-l_f \cdot c_f + l_r \cdot c_r}{J_z} \\ -\dfrac{c_f}{m} \end{bmatrix}, \tag{4.21}$$

and

$$\widetilde{\boldsymbol{r}} = \begin{bmatrix} 0 \\ -v_x \cdot \kappa \\ 0 \\ 0 \end{bmatrix}. \tag{4.22}$$

In a final step, we discretize the system (4.19) for a sample time T. This leads to the system

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{r}_k, \tag{4.23}$$

where

$$\mathbf{A} = \exp\left(\widetilde{\mathbf{A}}T\right), \tag{4.24}$$

$$\mathbf{B} = \widetilde{\mathbf{A}}^{-1}\left(\exp(\widetilde{\mathbf{A}}T) - \mathbf{I}\right)\widetilde{\mathbf{B}}, \tag{4.25}$$

and

$$\mathbf{r} = \widetilde{\mathbf{A}}^{-1}\left(\exp\left(\widetilde{\mathbf{A}}T\right) - \mathbf{I}\right)\widetilde{\mathbf{r}}. \tag{4.26}$$

4.2 Steering System Model

4.2.1 Coupled Steering Mechanism

We start by describing the mechanism of an EPS (Electric Power Steering) system [92]. The steering wheel presents the main haptic interface for the driver, providing essential feedback and control over the vehicle's direction. The driver applies a steering wheel torque T_D resulting in a steering wheel angle δ. The input from the steering wheel is transmitted through the steering column to the pinion shaft via the torsion bar, which has a stiffness coefficient $k_s > 0$. The torsion bar allows the steering column to twist relative to the pinion shaft. When there is a difference between the steering angle δ and the twist angle δ_{PS} on the pinion shaft, the static part of the column torque T_S is measured by a sensor. This can be approximated by

$$T_S \approx k_S \cdot (\delta - \delta_{PS}). \tag{4.27}$$

The pinion shaft converts rotational motion into translational motion on the rack. This translation is then converted into the rotation of the wheels at the wheel suspension. To reduce the driver's effort, steering assistance is provided by a standard assistant torque T_{EPS}^{std}, which is computed by the software controlled EPS. This assistant torque is converted into a rack force via the motor shaft, meaning both the pinion shaft and the motor shaft act on the rack simultaneously. In Figure 4.3, a simplified model of the steering mechanism is illustrated.

In addition to the standard assistant torque T_{EPS}^{std}, EPS is capable of generating an additional steering torque T_A^{EPS}. This additional torque allows the assistance system to fine-tune the steering target, providing more precise control and adaptability to various driving conditions, and thus, enhance the overall driving experience [93]. The system uses advanced sensors and control algorithms to adjust the steering assistance based on real-time data, such as vehicle speed and road conditions. This capability supports both assisted driving features, like lane-keeping assist and parking assist, and automated driving functions, enabling the vehicle to perform complex steering tasks with minimal driver input. The total EPS torque results in

$$T_{EPS} = T_{EPS}^{std} + T_A^{EPS}. \tag{4.28}$$

In coupled steering, the inputs of the driver and the assistance system are blended together. Let $b(\mathbf{v}, \mathbf{w})$ be a nonlinear blending function. Then $\mathbf{v}$ and $\mathbf{w}$ are coupled by

$$\mathbf{u} = g(\mathbf{v}, \mathbf{w}) = b(\mathbf{v}, \mathbf{w}) \cdot \mathbf{v} + (1 - b(\mathbf{v}, \mathbf{w})) \cdot \mathbf{w}. \tag{4.29}$$

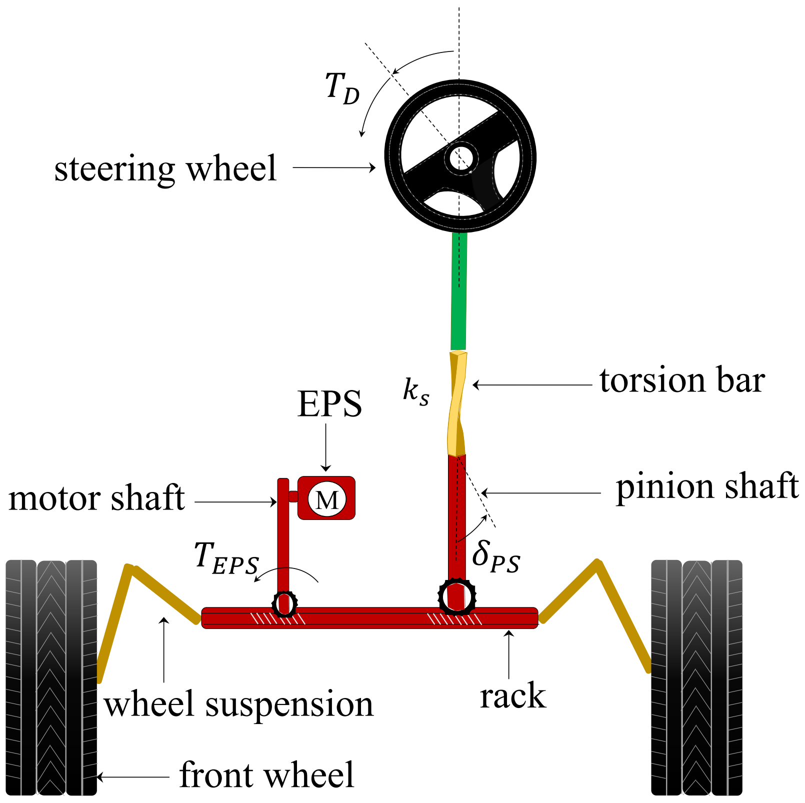


Figure 4.3: Electric Power Steering (EPS) mechanism. *© 2024, IEEE*

4.2.2 Decoupled Steering Mechanism

Decoupled steering, also known as Steer-by-Wire (SbW), is a steering mechanism where the steering wheel and the front wheels are not mechanically connected. Instead, this system relies on electronic controls to manage the steering [94]. Figure 4.4 illustrates the basic function of a SbW system. The driver uses the steering wheel to command his target steering angle. His torque input T_D is not transferred directly to the vehicle. It is transformed to a torque T_{EPS} by a control unit according to a predesignated function $u(T_D)$ [83]. The torque feedback $T_{A,f}$ the drivers perceives at the steering wheel is also provided by the control unit. In decoupled steering, the inputs of the driver and the assistance system do not necessarily interfere. The shared control input can be formulated as a weighted combination of both inputs, i.e.,

$$\mathbf{u} = g(\mathbf{v}, \mathbf{w}) = \omega_1 \cdot \mathbf{v} + \omega_2 \cdot \mathbf{w}, \tag{4.30}$$

where ω_1 and ω_2 are adjustable weights that determine the level of control each input has over the vehicle's steering.

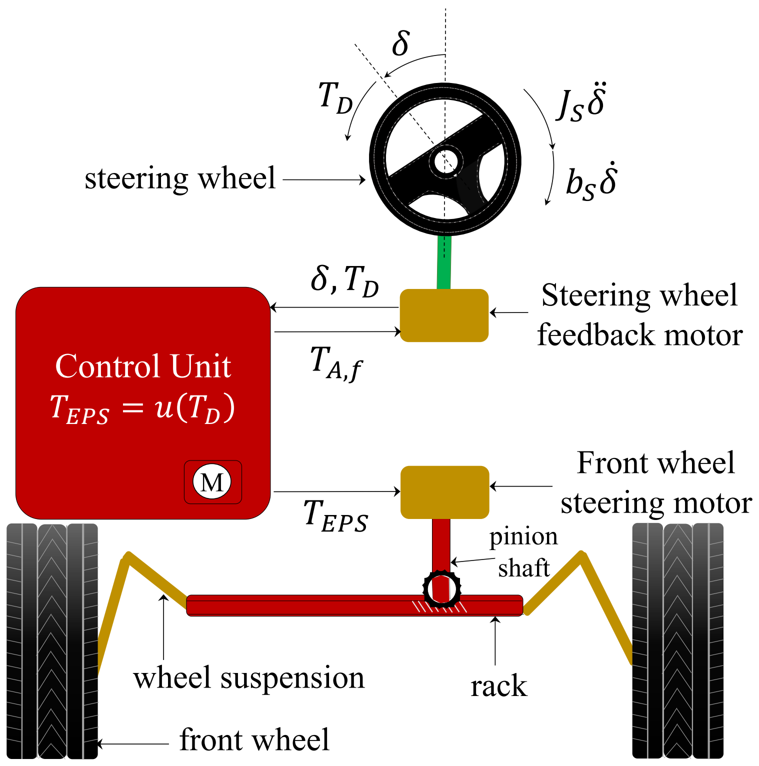


Figure 4.4: Decoupled steering mechanism

4.3 Driver-Steering Interaction Model

4.3.1 Torque Equilibrium

A driver-steering interaction model is presented to estimate and analyze the driver steering torque T_D. We start by converting the standard EPS torque T_A^{EPS} in (4.28) to the pinion shaft reference, resulting in an assistance torque T_A. Let T_r be the torque feedback on the steering wheel, resulting of the combined effect of the standard assistant torque T_{EPS}^{std} and tire-road interaction.

We consider a simplified free body diagram of the coupled steering mechanism and study two subsystems separately [19]:

- Subsystem 1: the steering wheel and the steering column
- Subsystem 2: the pinion shaft

An illustration is given in Figure 4.5.

The following torque equilibrium is established for Subsystem 1:

$$T_D - T_S - J_S\ddot{\delta} - b_S\dot{\delta} = 0, \tag{4.31}$$

where T_S is the column torque (4.27), $\dot{\delta}$ the steering angle velocity and $\ddot{\delta}$ the steering angle acceleration. J_S and b_S denote the steering wheel specific moment of inertia and damping

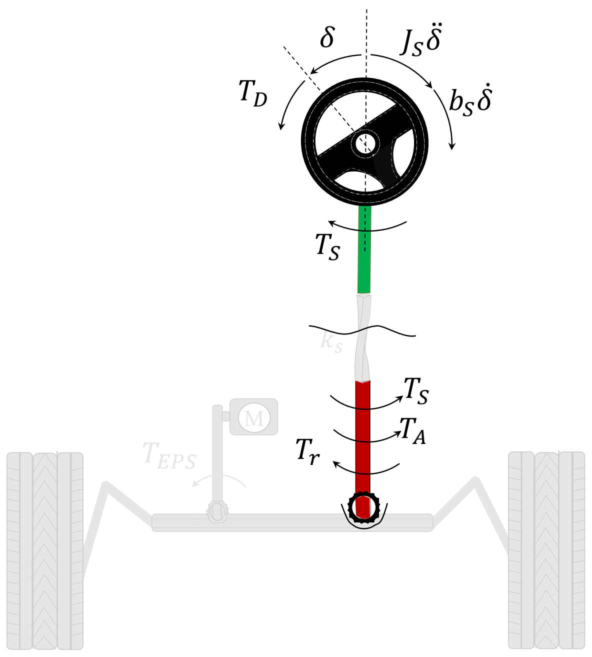


Figure 4.5: Section of steering wheel, steering column and pinion shaft. *© 2024, IEEE*

respectively and are assumed to be constant.
For Subsystem 2 the torque equilibrium is written as

$$T_S = T_r - T_A. \tag{4.32}$$

Inserting (4.32) in (4.31) leads to the equilibrium

$$T_A + T_D = T_r + J_S\ddot{\delta} + b_S\dot{\delta}. \tag{4.33}$$

This indicates that the driver and the assistance system cooperate to compensate the torque T_r resulting of tire-road interaction, the torque $J_S\ddot{\delta}$ countering the acceleration of the steering wheel and the torque $b_S\dot{\delta}$ overcoming the friction.

4.3.2 Tire-Road Interaction

The driver-steering interaction model have to account for both, fully and partially automation. Thus, we propose a novel modeling of the torque T_r resulting from tire-road interaction, by splitting the compensation of the tire-road effects between the driver and the assistance system, i.e.

$$T_r = T_{r,D} + T_{r,A}, \tag{4.34}$$

where $T_{r,D}$ is the part of T_r that is compensated by the driver and $T_{r,A}$ the part compensated by the assistance system. To ensure a well-defined modeling of T_r, additional constraints need to be specified. It is obvious that $T_{r,A}$ must fulfill

$$\begin{cases} 0 \leq T_{r,A} \leq T_A, & \text{for } T_A \geq 0, \\ T_A \leq T_{r,A} \leq 0, & \text{for } T_A < 0. \end{cases} \tag{4.35}$$

As for $T_{r,D}$, equation (4.32) implies

$$\begin{cases} 0 \leq T_{r,D} \leq T_S, & \text{for } T_S \geq 0, \\ T_S \leq T_{r,D} \leq 0, & \text{for } T_S < 0. \end{cases} \tag{4.36}$$

and (4.34) implies

$$\begin{cases} 0 \leq T_{r,D} \leq T_r, & \text{for } T_r \geq 0, \\ T_r \leq T_{r,D} \leq 0, & \text{for } T_r < 0. \end{cases} \tag{4.37}$$

Considering (4.32) and (4.34) and the inequalities (4.35), (4.36) and (4.37), the following conditions on $T_{r,D}$ are derived:

$$T_{r,D} = \begin{cases} T_r, & \text{for } T_r \cdot T_A < 0, \\ \max\{T_S, 0\}, & \text{for } T_r \geq 0, T_A \geq 0, \\ \min\{T_S, 0\}, & \text{for } T_r < 0, T_A < 0. \end{cases} \tag{4.38}$$

4.3.3 Driver Torque

The driver torque T_D is modeled as the sum of two parts [19]:

- *Activity torque*, denoted by T_D^s, the part of T_D that corresponds to the torque effort of the driver for steering the vehicle.
- *Conflict torque*, denoted by T_D^c, the part of T_D that corresponds to the torque generated by the driver's arms in order to counter the actions of the assistance system.

We write

$$T_D = T_D^s + T_D^c. \tag{4.39}$$

Activity Torque

The activity torque T_D^s is the part of the driver torque T_D exerted by the driver to steer the vehicle to the desired target. It corresponds to the complete driver torque, when the driver is driving manually. We assume that T_D^s is the sum of the torque performed by the driver to compensate the torque T_r resulting from tire-road interaction and the torque performed by the driver to achieve his target steering angle if no limitations through assistance or the road are provided. We denote this part T_D^δ. Thus, we write

$$T_D^s = T_{r,D} + T_D^\delta. \tag{4.40}$$

Inserting (4.39) and (4.40) in (4.31) yields to the equilibrium

$$T_D^\delta + T_{r,D} + T_D^c - T_S = J_S\ddot{\delta} + b_S\dot{\delta}. \tag{4.41}$$

To account for the case of fully manual driving, i.e. $T_A = 0$ and $T_D^c = 0$, (4.32) and (4.41) yield to the constraint

$$\begin{cases} 0 \le T_D^\delta \le J_S\ddot{\delta} + b_S\dot{\delta}, & \text{for } J_S\ddot{\delta} + b_S\dot{\delta} \ge 0, \\ J_S\ddot{\delta} + b_S\dot{\delta} \le T_D^\delta \le 0, & \text{for } J_S\ddot{\delta} + b_S\dot{\delta} < 0. \end{cases} \tag{4.42}$$

Conflict Torque

The conflict torque T_D^c is the part of the driver torque T_D that is generated *only* to counter the assistance torque T_A, i.e., the conflict torque never exceeds the assistance torque:

$$\begin{cases} 0 \le T_D^c \le -T_A, & \text{for } T_A \le 0, \\ -T_A \le T_D^c \le 0, & \text{for } T_A > 0. \end{cases} \tag{4.43}$$

Previous studies show that a conflict in objective of the driver and the assistance system while holding the steering wheel, can be modeled by a linear mass-damper-spring model [18]-[19], that refers to the neutral position of the steering wheel. We follow a similar approach to model T_D^c. However, in our approach, the model is not restricted to the neutral position of the steering wheel. It accounts for fully and partially automation, i.e., when the driver takes his hands from the steering wheel, the assistance system assumes control and aims to reach its steering angle target δ_A. Whenever the driver wants to set a different steering angle, the conflict torque T_D^c is applied, to achieve the steering angle $\delta_e = \delta_A - \delta$. This leads to a linear mass-damper-spring model that refers to the angle difference between the driver's and the system's target:

$$T_D^c = J_D\ddot{\delta}_e + b_D\dot{\delta}_e + k_D\delta_e. \tag{4.44}$$

(4.44) implies that the conflict torque T_D^c compensates the inertia moment $J_D\ddot{\delta}_e$, the damping torque $b_D\dot{\delta}_e$ and the stiffness torque $k_D\delta_e$, with $J_D, b_D, k_D \ge 0$.

4.3.4 Model Equation

To sum up, Figure 4.6 provides an illustration of the various modeling steps for the driver torque at the steering wheel, as outlined by the equations (4.33), (4.39), (4.40) and (4.44).

We integrate the equations of the previous section into one compact model equation describing the driver-steering interaction model. Inserting (4.44) in (4.41) leads to

$$\begin{bmatrix} \ddot{\delta}_e & \dot{\delta}_e & \delta_e & 1 \end{bmatrix} \cdot \begin{bmatrix} J_D \\ b_D \\ k_D \\ T_D^\delta \end{bmatrix} = J_s\ddot{\delta} + b_s\dot{\delta} + T_s - T_{r,D}. \tag{4.45}$$

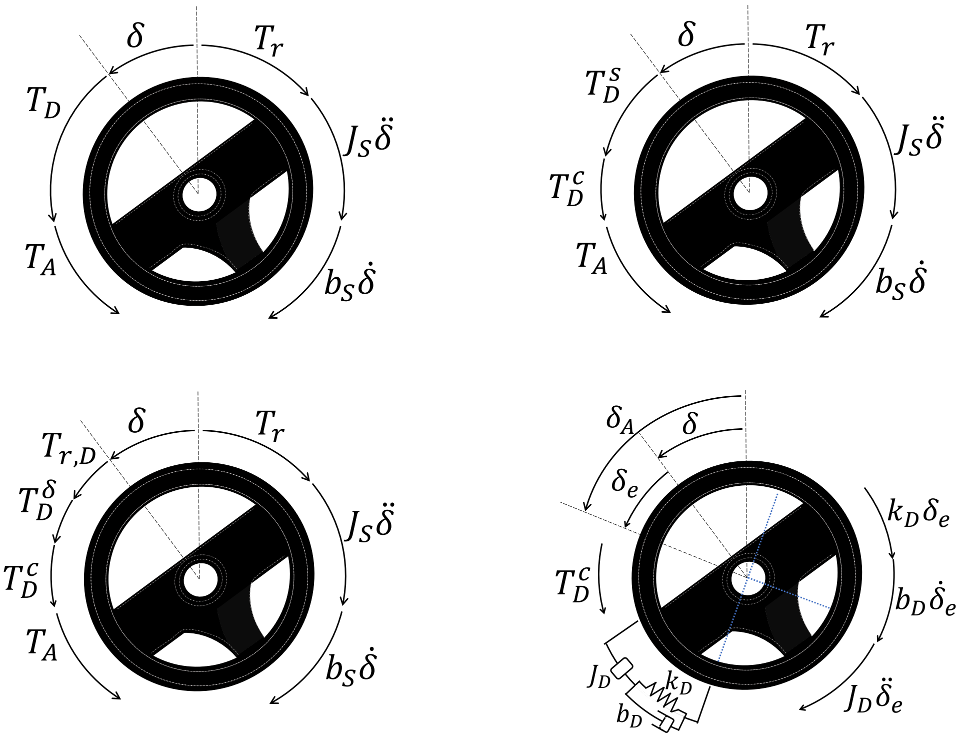


Figure 4.6: Modeling steps of the driver-steering interaction model: (a) Torque equilibrium at the steering wheel (4.33), (b) Modeling the driver torque T_D as the sum of the activity torque T_D^s and the conflict torque T_D^c (4.39), (c) Modeling the activity torque T_D^S as the sum of $T_{r,D}$ and T_D^δ (4.40), and (d) Modeling the conflict torque T_D^c with a linear mass-damper-spring model (4.44), © *2024, IEEE*

When the driver interacts with the assistance system, the parameter of the driver arm impedance J_D, b_D and k_D and the target steering angle torque T_D^δ vary. The identification of these coefficient allows a comprehensive analysis of the driver torque.

4.3.5 Discretization

The Varying-Coefficient (VC) method introduced in Section 2.1.2 is applied to the model (4.45). Let

$$y_k = J_S\ddot{\delta}_k + b_S\dot{\delta}_k + T_{S,k} - T_{r,D,k}, \tag{4.46a}$$

$$\beta_k = \begin{bmatrix}\ddot{\delta}_{e,k} & \dot{\delta}_{e,k} & \delta_{e,k} & 1\end{bmatrix}, \tag{4.46b}$$

$$x_k = \begin{bmatrix}J_{D,k} & b_{D,k} & k_{D,k} & T_{D,k}^\delta\end{bmatrix}^T. \tag{4.46c}$$

We compute $\mathbf{x}, \mathbf{y}, \mathbf{B}, \mathbf{P}$ and $\mathbf{G}$ as in (2.9a),(2.9b),(2.9e), (2.10) and (2.12) respectively, and write the vector $\mathbf{c}$ as

$$\mathbf{c} = -\mathbf{y}^T\mathbf{B}, \tag{4.47}$$

and the matrix $\mathbf{H}$ as

$$\mathbf{H} = \mathbf{B}^T\mathbf{B} + \mathbf{P}^T\mathbf{G}\mathbf{P}. \tag{4.48}$$

We consider (4.42), (4.43) and the definition of T_D^c in (4.44) and define the matrix $\mathbf{A}$ as

$$\mathbf{A} = \begin{bmatrix} & \widetilde{\mathbf{A}} & \\ \widetilde{\mathbf{N}} & & 0 \\ & \ddots & \\ 0 & & \widetilde{\mathbf{N}} \end{bmatrix}, \tag{4.49}$$

where

$$\widetilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_1^+ & & & 0 \\ & \mathbf{A}_2^+ & & \\ \mathbf{A}_1^- & & \ddots & \\ & \mathbf{A}_2^- & & \mathbf{A}_K^+ \\ & & \ddots & \\ 0 & & & \mathbf{A}_K^- \end{bmatrix}, \tag{4.50}$$

with

$$\mathbf{A}_k^+ = \begin{bmatrix} \ddot{\delta}_{e,k} & \dot{\delta}_{e,k} & \delta_{e,k} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{A}_k^- = -\mathbf{A}_k^+, \tag{4.51}$$

and

$$\widetilde{\mathbf{N}} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}. \tag{4.52}$$

Additionally, let

$$\mathbf{b} = \left[\mathbf{b}_{ub,1}, -\mathbf{b}_{lb,1}, \cdots, \mathbf{b}_{ub,K}, -\mathbf{b}_{lb,K}, \underbrace{0, \cdots, 0}_{3K} \right]^T, \tag{4.53}$$

where

$$\mathbf{b}_{ub,k} = \left[\max\{0, -T_{A,k}\} \quad \max\{0, J_S\ddot{\delta}_k + b_S\dot{\delta}_k\} \right]^T, \tag{4.54}$$

and

$$\mathbf{b}_{lb,k} = \left[-\min\{0, -T_{A,k}\} \quad \min\{0, J_S\ddot{\delta}_k + b_S\dot{\delta}_k\} \right]^T. \tag{4.55}$$

Then, $\mathbf{x}$ is computed by solving the quadratic program

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} \tag{4.56a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}. \tag{4.56b}$$

From the structure of $\mathbf{B}$ in (2.9e), $\mathbf{P}$ in (2.10) and $\mathbf{G}$ in (2.13) and from (4.48), it follows that $\mathbf{H}$ is symmetric positive definite, and hence, the QP in (4.56) is convex and has a unique solution.

5 Method for Driver Interaction Classification

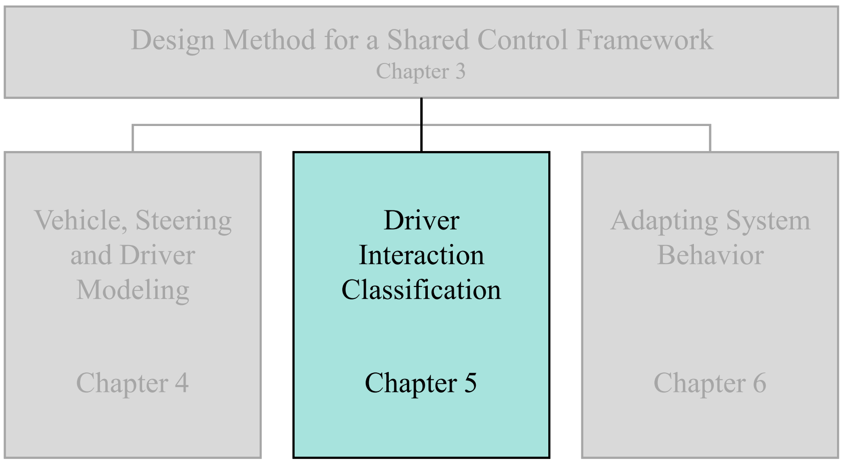

Design Method for a Shared Control Framework
Chapter 3
Vehicle, Steering and Driver Modeling
Chapter 4
Driver Interaction Classification
Chapter 5
Adapting System Behavior
Chapter 6

Figure 5.1: Scope of Chapter 5

In this chapter, we aim to design a classifier, which is able to identify the driver interaction strategy of a driver when interacting with an intervening assistance system. This chapter covers the design process of the classifier and is structured as follows: In Section 5.1, we start with a review of related works in the field of driver interaction classification. Then, Section 5.2 presents the experimental design and outlines the process of data collection and processing. In Section 5.3 and Section 5.4, we proceed to the feature analysis and the class identification. In Section 5.5, we present the implementation and validation process of the designed classifier. Finally, in Section 5.6, we apply Dynamic Mode Decomposition (DMD) to analyze the underlying dynamics for each class.
Certain aspects in this chapter have been previously published by the author (El Amouri et al. [87] and El Amouri [95]). Additionally, some methods and data processing steps were developed in a master's thesis under the author's supervision (Markgraf [96]). The presentation and analysis have been adapted and expanded for the purposes of this chapter.

5.1 Related Work

A classification of drivers *while* they interact with an intervening assistance system can be crucial for personalized assistance system design. Across many studies diverse terminology has been employed for different driver categorization. In [97], the authors considered four different driving styles: calm, moderate, aggressive and very aggressive. The driver style was integrated in the design of a lane keeping assist to enhance the personalization of the system.

The results showed reduced lane tracking errors. In [98], drivers were labeled based on their understanding of ADAS. They were classified in *skeptic*, those who do not trust the system and generally turn it off, *conscious*, those who are aware of the strengths and weaknesses of the system, and *enthusiast*, those who have excessive trust in the system and may even use it in a way that is not intended. In [99], the drivers were classified into three classes based on their interaction behavior in curves: *resistant*, those who steer too early or too late in the curve, *active*, those who steer actively *with* the system into the curve, and *passive*, those who completely leave the control to the system. In [100], different driving studies were conducted. The drivers were categorized based on their level of trust in the assistance system. The identified behavior ranges from being highly trusting and attentive to being inattentive or skeptical. The results approved that drivers choose different strategies when interacting with intervening ADAS.
In this chapter, we will introduce a new categorization of drivers based on their interaction with an intervening steering assistance system. We assess the interaction from an objective point view, based on real driving data, without relying on self-reported data from drivers. To the best of the author's knowledge, studies in this particular area of research are limited. In [15], it is shown that the subjective driver acceptance of a lane change system can be influenced by various objective parameters, such as the driver input, the dynamic vehicle behavior, and the driving environment. Based on theses parameters, the authors suggest an index for evaluating the driver acceptance of the lane change system [14]. Hence, we aim to design a classifier based only on drivers' contribution to the driving task and on the dynamics of the vehicle. This requires the definition of the driver classes as well as suitable features.

5.2 Experimental Design

5.2.1 Driving Demonstration

Over a twisty infield driving circuit, 58 drivers were instructed to drive a racing line in the research vehicle "Race Trainer" introduced in [99, 101]. The driving circuit is divided into track meters. One fixed optimal reference path was calculated in advance using a mathematical optimization that minimizes the driving time over the track

$$\min \quad t_{\text{end}} = \int_{s=0}^{s_{\text{end}}} \sigma(s)\,\mathrm{d}s, \tag{5.1}$$

where σ describes the mapping from spatial to temporal progression along the path, and hence, allows a space-based parametrization of the vehicle dynamics. It is computed by [102]

$$\sigma = \frac{\mathrm{d}t}{\mathrm{d}s} = \frac{1-\kappa d}{v\cos(\psi+\beta)}, \tag{5.2}$$

where κ is the curvature of the road as in (4.14), d the lateral deviation, v the velocity of the vehicle, β the sideslip angle and ψ the heading angle, as introduced in Section 4.1.1.
The optimal reference path and the driving circuit are illustrated in Figure 5.2.

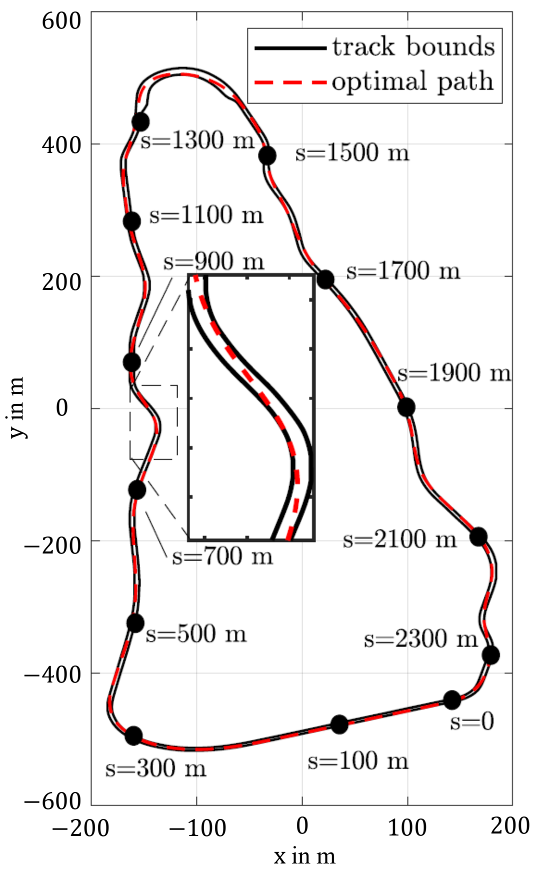


Figure 5.2: Full test track divided into track meters and optimal path target. *© 2024, IEEE*

In a first lap, the drivers were driven in a fully automated mode, which allowed them to get familiar with the track and the system's optimal reference path. The following laps were assisted by the system, which intervened actively in the driver actions. For this purpose, the test vehicle was equipped with an EPS based assistance system, that was able to operate additional assistance steering torque with different intensity levels, to lead the driver back to the reference line. If the driver were to take his hands off the steering wheel, this torque would be enough to stay on the racing line. Regarding longitudinal control, accelerating and braking is applied through separate interfaces: the accelerator and brake pedals. To guide the driver to the optimal speed, a minimum speed is recommended by the system, and for safety assurance, a maximum speed limit is enforced, varying based on the driver's experience. Thus, braking is implemented in a way that the system always considers the maximum between the driver's braking request and the system's braking request, i.e., the system can never apply less braking than the driver desires. Therefore, the interaction between the driver and the system in longitudinal control is limited and can not be handled in the same way as the haptic interaction in lateral control. As we aim to investigate the driver's interaction with the system, when it actively intervenes in his actions without any restrictions, we limit our focus in this work to the interaction in lateral control.
An experienced instructor observed the participants throughout without any intervention. He

subjectively categorized the drivers based on their driving skills and performance on the race track in three categories: beginner, intermediate and expert. The distribution of the drivers over the three categories is shown in Table 5.1.

Table 5.1: Expert Categorization of All Drivers © *2024, IEEE*

Category	Driver number	Percentage
beginner	1, 10, 13, 16, 19-23, 32, 33, 35, 36, 38-41, 48, 51, 52	34 %
intermediate	2-6, 8, 9, 14, 15, 17, 18, 24, 26, 27, 29, 30, 34, 44, 45, 47, 50, 53-58	47 %
expert	7, 11, 12, 25, 28, 31, 37, 42, 43, 46, 49	19 %

5.2.2 Data Collection and Processing

During the drive a wide range of data are collected, including inputs of the driver, physical quantities of the vehicle and position relative to the road. The data collection and processing steps presented here are based on the methodology developed in [96]. The data is recorded using the Measuring device RT3000 from OxTS[1]. This device features three accelerometers and three gyroscopes for measuring translational and rotational accelerations in the vehicle's fixed coordinate system, as well as a GPS receiver. The calculation of speed and position data is performed by the temporal integration of the measured accelerations, with systematic measurement deviations corrected using measured GPS data.
The data is recorded at a sampling rate of 200 Hz, corresponding to a sampling interval of 5 ms. Due to the high computational demands of the control algorithms, the recording of measurement data is paused at regular intervals. Whenever the time between two consecutive measurements exceeds the sampling interval of 5 ms, intermediate values are inserted and linearly interpolated. The interpolated data is filtered using a low-pass filter. The filter is configured with a passband cutoff frequency of 2 Hz and a stopband cutoff frequency of 5 Hz. The passband ripple is constrained to 1 dB, while the stopband attenuation is set to -60 dB. Since each driver has their own time scale, we transform all time dependent variables into a spatial representation, by aligning all the results along the fixed track meters in Figure 5.2. This ensures consistency in the comparison.

1 OxTS (Oxford Technical Solutions) provides advanced technology for precise vehicle measurements, including position, roll, pitch, and heading. The RT3000 series is highly accurate and widely used for vehicle dynamics testing. See: `https://www.oxts.com/`

5.3 Feature Analysis

5.3.1 Conflict and Passivity

Feature Exploration

A divergence between the assistance system path target and the driver's path target can lead to completely different reactions. While some drivers enforce their target and counteract, other drivers simply leave the control to the system. To describe both reactions, we need an indicator for the driver contribution (or non-contribution) to the driving task and an indicator for the driver counteraction.
For this investigation, we will take advantage of the driver torque modeling from Section 4.3. The activity and conflict torque will be considered as a basis for conflict and passivity features.

Feature Specification

To evaluate conflict and passivity, we extract features from the conflict torque T_D^c and the activity torque T_D^s. We assume that passivity is defined as the absence of activity torque. A summary of the considered features is given in Table 5.2

Table 5.2: Conflict and Passivity Features

Feature	Description
$\overline{\lvert T_D^c \rvert}$	Mean value of the conflict torque
$T_{D,\max}^c$	Maximal value of the conflict torque
$\widetilde{\Delta t}_c$	Absolute conflict duration
$\widetilde{\Delta t}_{c,rel}$	Relative conflict duration
$\widetilde{\Delta t}_{c,\max}$	Maximal duration of a single conflict phase
$\overline{\lvert T_D^s \rvert}$	Mean value of the activity torque
$T_{D,\max}^s$	Maximal value of the activity torque
$\widetilde{\Delta t}_p$	Total passivity duration
$\widetilde{\Delta t}_{p,\max}$	Maximal duration of a single passivity phase

Computation and Validation

To compute the conflict and activity features in Table 5.2, we first solve the driver-steering interaction model. The `quadprog` function from MATLAB's Optimization Toolbox [103] was applied to solve the quadratic program (4.56). The parameters used in the vehicle model and in the computation are listed in Table 5.3.

Table 5.3: Model and Computation Parameters © *2024, IEEE*

Param.	J_S	b_S	$\max\lvert T_A\rvert$	$\max\lvert T_S\rvert$	N	Δt
Value	0.03	0.3	3	8	100	0.005
Unit	kg m^2	Nm s^{-1}	Nm	Nm	-	s

Since each driver has their own time scale, the time dependent results are transformed into a spatial representation, by dividing the track into track meters and applying interpolation methods. Aligning all the results along the fixed track meters on the track, ensures consistency when comparing different drivers. The resulted parameters J_D, b_D and k_D are illustrated in Figure 5.3 for three different drivers over three sections of the track.

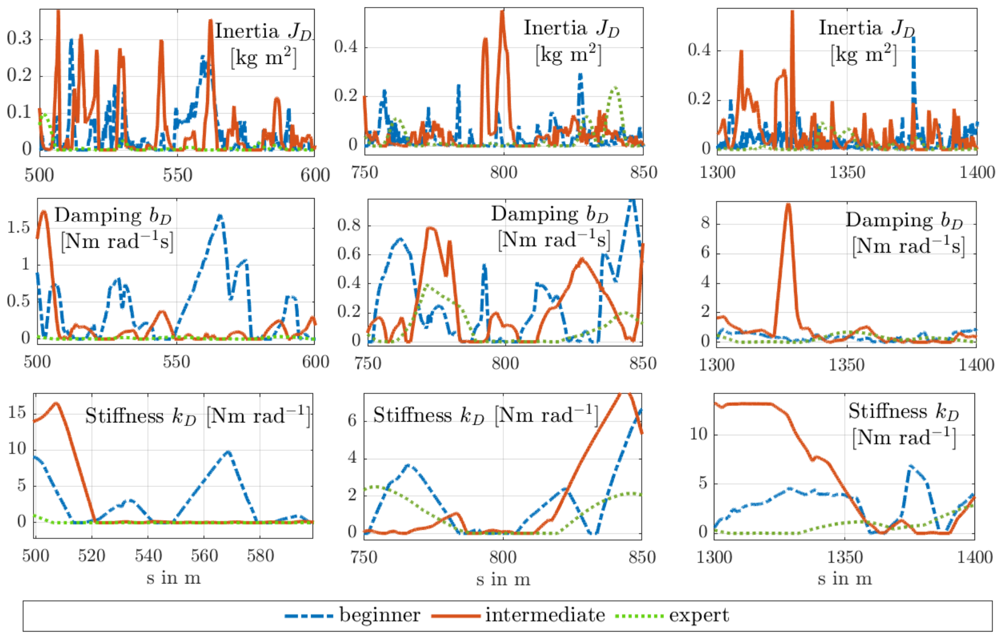


Figure 5.3: Resulted parameters J_D (top), b_D (middle) and k_D (bottom) estimated by the VC-method for three drivers over three different sections of the track. © *2024, IEEE*

The mean values of J_D, b_D and k_D for all 58 drivers over one full lap are illustrated in Figure 5.4.

The estimated parameters are used to compute the conflict torque T_D^c as in (4.44) and the activity torque T_D^s as in (4.40). The results for two different drivers are shown Figure 5.5.

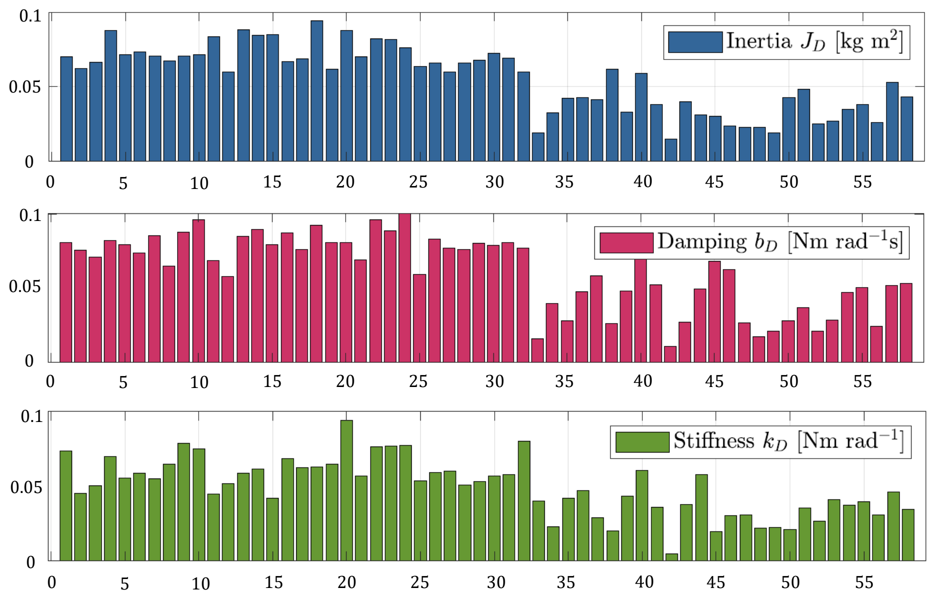


Figure 5.4: Mean values of the estimated parameters J_D (top), b_D (middle) and k_D (bottom) over one full lap for all 58 Drivers. © *2024, IEEE*

Driver 1 shows low conflict torque and higher activity torque values. This indicates an active driver, who is able to follow the system's target. Driver 2, who also applies high activity torque, is continuously in conflict with the system. In addition, his activity torque shows irregularities in the steering input, leaving the impression of overwhelm. The intended steering torques of the drivers align at multiple points. For both drivers, the driver-steering interaction model is able to capture the different dynamics.

We compare the resulted conflict torque with the results of a correlation analysis considering the two variables T_S and T_A. The results are shown in the two upper plots of Figure 5.5. The sign of the cross-correlation function $\hat{R}_{T_S T_A}$ changes depending on whether the column torque and the assistance torque are aligned or opposed. A negative sign indicates that the driver is applying torque against the assistance torque. However, this does not necessarily signify a conflict, as a passive driver can exert an opposing effect simply by holding the steering wheel, adding extra rotational damping. To address this, we experimentally derive a threshold value $\varepsilon_{\hat{R}}$ for the magnitude of $\hat{R}_{T_S T_A}$, based on expert evaluation, shown as the dashed horizontal line in the plots. We see that both our model and the correlation analysis yield consistent results when identifying conflict.

Although the correlation analysis has its limitations and can only highlight obvious conflicts, it still serves as a useful preliminary validation tool for the driver-interaction model, that can not only detect conflict but also identify it as a physical measure and is very sensitive to account for changes.

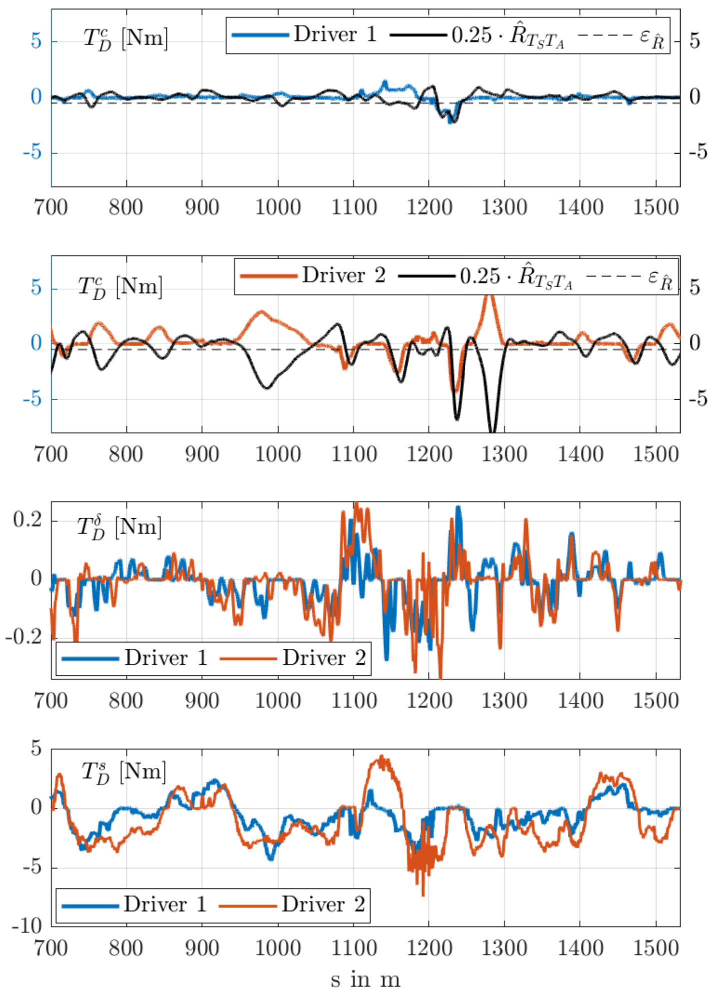


Figure 5.5: Resulted conflict torque (top two plots), desired torque (second from bottom) and activity torque (bottom) for two drivers. © *2024, IEEE*

5.3.2 Path Consistency

Feature Exploration

When watching the road ahead, a driver has an intuitive path target he aims to follow. While some drivers always aim to reproduce their own subjective *optimal* path, other drivers may alter their paths depending on the driving situation. Thus, in terms of an intervening assistance system, a path consistency analysis can indicate the persistence or the flexibility of a driver in following his target path.

Feature Specification

A driver who is able to consistently reproduce a similar path, shows similar values of the lateral distance d and the heading angle $\Delta\psi$ over the same track sections.
To evaluate the similarity of driven paths, the considered path consistency features are summarized in Table 5.4.

Table 5.4: Path Consistency Features

Feature	Description
σ_d	Standard deviation of lateral distance
$\max d$	Maximum of lateral distance
$\sigma_{\Delta\psi}$	Standard deviation of heading angle error
$\max \Delta\psi$	Maximum of heading angle error

5.3.3 Individual Path Pattern Targeting

Feature Exploration

Driver's target path can be influenced by the intensity of the assistance interventions, as they can accept the system up to their personal tolerance limit. In other words, drivers can *select* to counteract or to cooperate with the system depending on its configuration. To model this selectivity, we introduce the Individual Path Pattern Targeting Algorithm.

α-Coordinate System

We consider N points s_i, with distances $\Delta s_k = \|s_k - s_{k-1}\|_2$, $k = 2, \cdots, N$, defining a reference path on a track. The intersection of a normal to this reference line at a reference point s_k with the road bounds, defines the right and left bound points $(x_{r,k}, y_{r,k})$ and $(x_{l,k}, y_{l,k})$, respectively.
For every reference point s_k a new coordinate $\alpha_k \in [0,1]$ is defined. On the left bound point $(x_{r,k}, y_{r,k})$, we set $\alpha_k = 0$. On the right bound point $(x_{l,k}, y_{l,k})$, we set $\alpha_k = 1$. Thus, every position (x_k, y_k) on the road can be written as

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{r,k} \\ y_{r,k} \end{bmatrix} + \alpha_k \cdot \begin{bmatrix} x_{l,k} - x_{r,k} \\ y_{l,k} - y_{r,k} \end{bmatrix}, \tag{5.3}$$

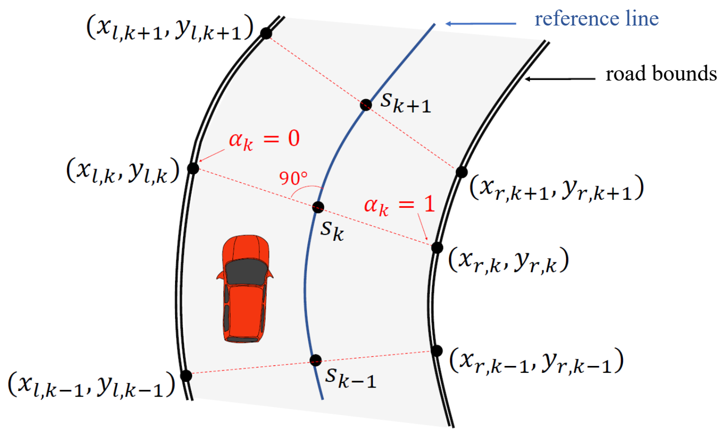


Figure 5.6: Reference line and α-coordinate system

with α_k its corresponding α-coordinate. We call this reference system *α-coordinate system*. A reference line in an α-coordinate system is illustrated in Figure 5.6.
To model a path line S_i with length N_i in the α-coordinate system, we define the vector

$$P_{S_i} = \begin{bmatrix}\alpha_1 & \cdots & \alpha_k & \cdots & \alpha_{N_i}\end{bmatrix}^T, \tag{5.4}$$

where $\alpha_k \in [0, 1]$ corresponds to the position point (x_k, y_k) at the reference point s_k, $k = 1, \cdots, N_i$.

Path Patterns

We present five different common path patterns: centering path pattern, shortening path pattern, curvature minimizing path pattern and track boundary path pattern, which includes both, the driving near the right and the driving near the left track side. In the following work, we will briefly discuss the path patterns and compute their optimal path lines in the α-coordinate system. The centering, shortening and curvature minimizing path patterns were introduced and referred to as driving modes in [104]. The presented derivation of these path lines mainly follows the methodology described therein. For more details, readers are directed to this source.

1. Track Boundary Path Pattern:
   Drivers tend to drive closer to the lane boundaries. Driving near the right side of the lane can make some drivers feel safer by avoiding oncoming traffic. On the other hand, driving near the left boundary provides a better view for overtaking opportunities. Although these driving patterns differ, they can be treated similarly from a mathematical perspective.
   We easily see that the optimal path lines in the α-coordinate system for the left and the right track boundary path pattern are given by

$$P_1 = \begin{bmatrix}0 & \cdots & 0\end{bmatrix}^T, \tag{5.5}$$

and

$$P_2 = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T, \tag{5.6}$$

respectively.

2. Centering Path Pattern:
   Drivers aim to keep their vehicle centered in the lane to maintain equal distance from both the left and right sides. This driving pattern is considered safety-focused.
   The center line is easily transformed into the α-coordinate system. Hence,

$$P_3 = \begin{bmatrix} 0.5 & \cdots & 0.5 \end{bmatrix}^T. \tag{5.7}$$

3. Shortening Path Pattern:
   The driving target is to reach the destination as quickly as possible. A shorter path can also lead to reduce fuel consumption for the vehicle.
   To compute the path with shortest length over the track we minimize the cost function

$$\sum_{k=1}^{N} \left\| \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} - \begin{bmatrix} x_k \\ y_k \end{bmatrix} \right\|_2^2, \tag{5.8}$$

for all position (x_k, y_k) on the road, where all (x_k, y_k) must remain within the road bounds. This constraint results in a computationally challenging problem in global coordinates. Thus, we transform the problem into the α-coordinate system. First, we define δ_{x_k}, δ_{y_k}, $\Delta x_{r,k}$ and $\Delta y_{r,k}$ as

$$\delta_{x_k} = \begin{bmatrix} x_{l,k+1} - x_{r,k+1} \\ -x_{l,k} + x_{r,k} \end{bmatrix}, \tag{5.9a}$$

$$\delta_{y_k} = \begin{bmatrix} y_{l,k+1} - y_{r,k+1} \\ -y_{l,k} + y_{r,k} \end{bmatrix}, \tag{5.9b}$$

$$\Delta x_{r,k} = x_{r,k+1} - x_{r,k}, \tag{5.9c}$$

$$\text{and} \quad \Delta y_{r,k} = y_{r,k+1} - y_{r,k}, \tag{5.9d}$$

respectively.
Let E_i be the $2 \times N$ matrix defined as

$$E_i = \begin{bmatrix} 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{5.10}$$

where only the i-th and $(i+1)$-th columns are non-zero.
We define the matrix H as

$$H = \sum_{i=1}^{N} E_i^T \left(\sum_{k=1}^{N} \delta_{x_k}^T \delta_{x_k} + \delta_{y_k}^T \delta_{y_k} \right) E_i, \tag{5.11}$$

and the vector c as

$$c = \sum_{i=1}^{N} \left(\sum_{k=1}^{N} 2\Delta x_{r,k} \delta_{x_k} + 2\Delta y_{r,k} \delta_{y_k} \right) E_i. \tag{5.12}$$

Using (5.3), (5.11), and (5.12), the problem in (5.8) can be reformulated as the following quadratic program depending only on α:

$$P_4 = \underset{\substack{0\le\alpha_k\le 1\\ k=1,\cdots,N}}{\arg\min}\ \frac{1}{2}\ \alpha^T H\alpha + c\alpha. \tag{5.13}$$

4. Curvature Minimizing Path Pattern:
Driving a trajectory with minimal curvature, enhances the comfort and smoothness perceived by the driver, as it reduces lateral accelerations. This driving pattern is also characterized by decreased steering effort for the driver.
To compute the path with minimal curvature over the track we minimize the cost function

$$\sum_{k=1}^{N} \left\| \begin{bmatrix} \frac{\mathrm{d}^2 x_k}{\mathrm{d}t^2} & \frac{\mathrm{d}^2 y_k}{\mathrm{d}t^2} \end{bmatrix}^T \right\|_2^2, \tag{5.14}$$

for all position (x_k, y_k) on the road, where all (x_k, y_k) must remain within the road bounds. we transform the problem into the α-coordinate system.
The main idea of the approach, is to model the path line as natural cubic splines composed of N control points. For control points $s_1, \cdots, s_N$ the natural cubic splines can be written as

$$S_i(t) = a_i + b_i t + c_i t^2 + d_i t^3, \tag{5.15}$$

where $t \in [0,1]$, $1 \le i \le N-1$ and a_i, b_i, c_i and d_i the coefficients to be solved. The second derivative of the natural cubic splines at each control point is given by

$$\left.\frac{\mathrm{d}^2 S(t)}{\mathrm{d}t^2}\right|_{t=0} = 6(K - LA^{-1}B)S = DS, \tag{5.16}$$

where

$$A = \begin{bmatrix} 2 & 1 & & & 0 \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ 0 & & & 1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 1 & & & 0 \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ 0 & & & -1 & 1 \end{bmatrix}, \tag{5.17}$$

$$K = \begin{bmatrix} -1 & 1 & & 0 \\ & \ddots & \ddots & \\ 0 & & -1 & 1 \end{bmatrix}, \quad L = \begin{bmatrix} 2 & 1 & & 0 \\ & \ddots & \ddots & \\ 0 & & 2 & 1 \end{bmatrix} \tag{5.18}$$

and

$$D = 6(K - LA^{-1}B). \tag{5.19}$$

Let

$$\Delta x_k = x_{l,k} - x_{r,k} \text{ and } \Delta y_k = y_{l,k} - y_{r,k}. \tag{5.20}$$

Then, we define the matrices ΔX and ΔY as

$$\Delta X = \mathrm{diag}(\Delta x_1, \cdots, \Delta x_N) \tag{5.21}$$

and

$$\Delta Y = \mathrm{diag}(\Delta y_1, \cdots, \Delta y_N), \tag{5.22}$$

and the vectors X_l and Y_l as the vectors containing all x- and y-components of the left bounds, i.e.,

$$X_l = \begin{bmatrix} x_{l,1} & \cdots & x_{l,N} \end{bmatrix}^T \tag{5.23}$$

and

$$Y_l = \begin{bmatrix} y_{l,1} & \cdots & y_{l,N} \end{bmatrix}^T. \tag{5.24}$$

Following (5.16), the second derivatives of the cubic splines $X(t)$ at the control points $x_1, \cdots, x_N$ and $Y(t)$ at the control points $y_1, \cdots, y_N$ become

$$\left(\left. \frac{\mathrm{d}^2 X(t)}{\mathrm{d}t^2} \right|_{t=0} \right)^2 = \alpha^T \Delta X^T D^T D \Delta X \alpha + 2X_l^T D^T D \Delta X \alpha + X_l^T D^T D X_l, \tag{5.25a}$$

$$\left(\left. \frac{\mathrm{d}^2 Y(t)}{\mathrm{d}t^2} \right|_{t=0} \right)^2 = \alpha^T \Delta Y^T D^T D \Delta Y \alpha + 2Y_l^T D^T D \Delta Y \alpha + Y_l D^T D Y_l. \tag{5.25b}$$

Let

$$Q = \Delta X^T D^T D \Delta X + \Delta Y^T D^T D \Delta Y, \tag{5.26}$$

and

$$f = 2X_l^T D^T D \Delta X + 2Y_l^T D^T D \Delta Y. \tag{5.27}$$

Then solving the problem in (5.14) is equivalent to solving

$$P_5 = \underset{\substack{0 \le \alpha_k \le 1 \\ k=1,\cdots,N}}{\arg\min}\ \alpha^T Q \alpha + f \alpha. \tag{5.28}$$

Table 5.5 summarizes the five paths $P_1, \cdots, P_5$ derived above

Table 5.5: Optimal Path Patterns

Notation	Description
P_1	Left Track Boundary Path Pattern
P_2	Right Track Boundary Path Pattern
P_3	Centering Path Pattern
P_4	Shortening Path Pattern
P_5	Curvature Minimizing Path Pattern

Following Tendency

The Following Tendency value $FT^j_{\mathcal{L}}$ describes whether a driver follows the optimal path pattern P_j at a specific section on the road.
Let P^* be the reference path and P_j the optimal paths in the α-coordinate system. We assume that $|\mathcal{L}|$ various intervention intensity levels can be adjusted in the assistance system configurations. Let $i \in I$ be the i-th time the same track section is passed and $l_i \in \mathcal{L}$ the corresponding intervention intensity level. Then, the driven path to evaluate is given by

$$P^d_{i,l_i} = \begin{bmatrix} \alpha^d_{i,l_i,1} & \cdots & \alpha^d_{i,l_i,N} \end{bmatrix}^T . \tag{5.29}$$

In a first step, we compute how *close* the driven path is to the optimal path patterns and the reference path. For two different paths x and y in the α-coordinate system, let $\Delta_{x\to y}$ be defined by the distance

$$\Delta_{x\to y} = \|x-y\|_2. \tag{5.30}$$

To allow a general evaluation of individual path pattern targeting, the Following Tendency value fulfills $FT^j_{\mathcal{L}} \in [0,1]$ and is defined as follows. For each path pattern P_j, $j \in \{1,\cdots,5\}$:

- Whenever the distance between the driven and the optimal path $\Delta_{P^d_{i,l_i}\to P_j}$ exceeds the distance between the reference and the optimal path $\Delta_{P^*\to P_j}$, we assume that the driver is not following this specific path pattern and we assign the value 0.
- Whenever the distance between the driven and the optimal path $\Delta_{P^d_{i,l_i}\to P_j}$ is equal or lower than the distance between the reference and the optimal path $\Delta_{P^*\to P_j}$, we assume that the driver is following this path pattern and we assign the value 1.
- Whenever the driven path is between the optimal and the reference path, we assign a value between 0 and 1, which increases linearly whenever the driven path approaches the optimal path.
- A driver only follows an individual path on a road section, if he shows this tendency each time he passes this section.

This approach can be summarized as follows. Let f^j_{i,l_i} be defined as the ratio of the distances $\Delta_{P^d_{i,l_i}\to P_j}$ and $\Delta_{P^*\to P_j}$, i.e.,

$$f^j_{i,l_i} = \frac{\Delta_{P^d_{i,l_i}\to P_j}}{\Delta_{P^*\to P_j}}, \tag{5.31}$$

and let $\overline{f}^j_{I,\mathcal{L}}$ be defined as

$$\overline{f}^j_{I,\mathcal{L}} = 1 - \frac{1}{|I|}\sum_{\substack{i\in I\\ l_i\in\mathcal{L}}} f^j_{i,l_i}. \tag{5.32}$$

Then the Following Tendency $FT^j_{\mathcal{L}}$ is given by

$$FT^j_{\mathcal{L}} = \begin{cases} 0, & \text{if} \quad \overline{f}^j_{I,\mathcal{L}} > 1 \text{ or } \max_{i \in I} f^j_{i,l_i} > 1, \\ \overline{f}^j_{I,\mathcal{L}}, & \text{if} \quad 0 \leq \overline{f}^j_{I,\mathcal{L}} \leq 1, \\ 1, & \text{if} \quad \overline{f}^j_{I,\mathcal{L}} < 0. \end{cases} \tag{5.33}$$

Increasing Tendency

The Following Tendency value introduced above, indicates if and how much a driver shows the tendency of following a specific path pattern, while driving with an intervening assistance system. We presume that the tendency of following a path pattern can only be confirmed, if it increases, when the assistance system intervention intensity decreases. Then, the driver has more freedom in controlling the driving task. To describe this characteristic, we introduce the Increasing Tendency value IT^j.
Let the set of intervention intensities of the assistance system be written as $\mathcal{L} = L_{\min} \cup L_{\max}$, where $L_{\min}$ contains the low intensity and $L_{\max}$ the high intensity levels. Let ω^+ and ω^- be the computed Following Tendencies

$$\omega^+ = FT^j_{L_{\max}}, \tag{5.34}$$

and

$$\omega^- = FT^j_{L_{\min}}, \tag{5.35}$$

respectively.
We suppose, that the tendency of following one of the optimal path patterns increases, when all the following conditions are fulfilled:

- The Following Tendencies are greater than 0 for high as well as for low assistance intervention intensities.
- The driven path with lower intervention intensity is nearer to the optimal path then to the reference path.
- The driven path with lower intervention intensity is nearer to the optimal path then the path with higher intervention intensity.
- The driven path is always nearer to the optimal path than to the reference path.

Then, the Increasing Tendency value IT^j is given by

$$IT^j = \begin{cases} 1, & \text{if } \omega^- - \omega^+ \geq 1, \\ \omega^- - \omega^+, & \text{if } 0 < \omega^- - \omega^+ < 1 \text{ or } \omega^- < \omega^+, \\ 0, & \text{if } \omega^+ \cdot \omega^- = 0 \text{ or } \max_{i \in I} f^j_{i,l_i} > 1. \end{cases} \tag{5.36}$$

Note that negative values of IT^j indicate a decreasing tendency.

Conflict Consideration

In shared control with fully or partially automation, we expect the vehicle to ensure a controlled passage through a curve, even when the driver remains passive on the steering wheel. However, the increased resistance in the steering system when passing through a curve may falsify the results of the Following and Increasing Tendencies, since it could be interpreted as a driver tendency of following an individual path pattern. To prevent this misinterpretation, these results are scaled using a weighting function. First, we introduce a conflict value $T_c \geq 0$, that describes the divergence rate between the driver input in the steering wheel and the assistance system steering. T_c can be modeled as a correlation value or as a driver conflict torque as in [87]. Given the conflict value T_c, we define the sigmoid weighting function

$$f_w(T_c) = 1 - \frac{1}{1 + e^{k(T_c+\beta)}}, \qquad k, \beta \in \mathbb{R}. \tag{5.37}$$

Multiplying the computed values FT^j and IT^j with f_w, ensures that the tendency of following an individual path pattern is only considered whenever the driver actively and sufficiently opposes the actions of the assistance system.

IPPT Algorithm

We summarize the previous steps into one concise algorithm: the Individual Path Pattern Targeting (IPPT) algorithm for shared lateral driving. For any driven path, where the driver shares the driving task with the assistance system, regardless different intensity levels, the IPPT algorithm is able to detect if the driver is targeting an individual path on a specific section of the road. We consider a driving assistance system, that is continuously intervening in lateral control, to guide the vehicle back to an optimal reference path, defined by N discrete reference points s_i, $i = 1, \cdots, N$. Let P_j be the optimal path patterns and P^* the reference path in the α-coordinate system. Let $\mathcal{L}$ be the set of various intervention intensity levels that can be adjusted in the assistance system, and $|I|$ the total number of times, the driver passes the same section of the road. We denote $P^d_{i,l} = [\alpha^d_{i,l,1}, \cdots, \alpha^d_{i,l,N}]^T$, $i \in I, l \in \mathcal{L}$, the driven path in the α-coordinate system. Then, the approach is outlined in Algorithm 8.

Feature Specification

The Following Tendency value $FT_{\mathcal{L}}$ and the Increasing Tendency value IT are the considered features for the individual path pattern analysis.

Algorithm 8: Individual Path Pattern Targeting (IPPT).

```
GIVEN
Optimal path patterns $P_j$, Reference path $P^*$, Driven path $P^d_{i,l}$, with
  $|P^d_{i,l}| = |P^*| = |P_j| = N$, $j \in \{1,\cdots,5\}$, $i \in I$, $n \leq N$, Intervention intensity set
  $\mathcal{L} = L_{\min} \cup L_{\max}$, weighting function $f_w(T_c) \geq 0$.
FOR $j \in \{1,\cdots,5\}$
    FOLLOWING TENDENCY
    1. $\Delta_{P^d_{i,l} \to P_j} \leftarrow \|P^d_{i,l} - P_j\|$ , $\Delta_{P^* \to P_j} \leftarrow \|P^* - P_j\|$ , $\Delta_{P^* \to P^d_{i,l}} \leftarrow \|P^* - P^d_{i,l}\|$
    2. $f^j_{i,l_i} \leftarrow \Delta_{P^d_{i,l_i} \to P_j} / \Delta_{P^* \to P_j}$
    3. $\overline{f}^j_{I,\mathcal{L}} \leftarrow 1 - \frac{1}{|I|} \sum_{\substack{i \in I \\ l_i \in \mathcal{L}}} f^j_{i,l_i}$
    4. $FT^j_{\mathcal{L}} \leftarrow \begin{cases} 0, & \text{if } \overline{f}^j_{I,\mathcal{L}} > 1 \text{ or } \max\limits_{i \in I} f^j_{i,l_i} > 1, \\ \overline{f}^j_{I,\mathcal{L}}, & \text{if } 0 \leq \overline{f}^j_{I,\mathcal{L}} \leq 1, \\ 1, & \text{if } \overline{f}^j_{I,\mathcal{L}} < 0. \end{cases}$
    5. $FT^j_{\mathcal{L}} \leftarrow f_w(T_c) \cdot FT^j_{\mathcal{L}}$
    INCREASING TENDENCY
    6. $\omega^+ \leftarrow FT^j_{L_{\max}}$, $\omega^- \leftarrow FT^j_{L_{\min}}$
    7. $IT^j \leftarrow \begin{cases} 1, & \text{if } \omega^- - \omega^+ \geq 1, \\ \omega^- - \omega^+, & \text{if } 0 < \omega^- - \omega^+ < 1 \text{ or } \omega^- < \omega^+, \\ 0, & \text{if } \omega^+ \cdot \omega^- = 0 \text{ or } \max\limits_{i \in I} f^j_{i,l_i} > 1. \end{cases}$
    8. $IT^j \leftarrow f_w(T_c) \cdot IT^j$
    RETURN $FT^j_{\mathcal{L}}$, $IT^j$
ENDFOR
```

5.3.4 Path Quality and Adaptation

Feature Exploration

Some drivers require time to become familiar with the system target and interventions. This time is an opportunity either to monitor the system closely and observe its action while it is in complete control, or to challenge it by testing and approaching its limits and limitations. Thus, it is crucial to consider the adaptation to the system over time.
For this investigation, let d and $\Delta\psi$ be the lateral distance and the heading angle error,

respectively, and let $\widetilde{d}$ and $\Delta\widetilde{\psi}$ be their standard scores. A good path quality is achieved whenever both $\widetilde{d}$ and $\Delta\widetilde{\psi}$ have lower absolute values. Thus, we introduce a cost function g_i describing the *quality* of the i-th driven path as

$$g_i = -\max\left\{|\widetilde{d}|, |\Delta\widetilde{\psi}|\right\}. \tag{5.38}$$

The adaptation to the system can be evaluated by comparing different driven paths over time. Let g_i and g_j describe the qualities of two different driven paths, where $j > i$ indicates that the corresponding path to g_j was driven later in time then the corresponding path to g_i. Then the driver adaptation value is given by

$$G = g_j - g_i, \tag{5.39}$$

for $j > i$. Thus, the sign of G indicates an improvement (positive sign) or a decline (negative sign) in following the assistance target.

Feature Specification

To evaluate the adaptation to the system, we consider the features summarized in Table 5.6

Table 5.6: Path Quality and Adaptation Features

Feature	Description
$\overline{G}$	Mean value of driver adaptation value over all driven paths
g_{end}	Quality of the last driven path
$\max\lvert\widetilde{d}\rvert$	Maximal absolute value of lateral distance standard score
$\max\lvert\Delta\widetilde{\psi}\rvert$	Maximal absolute value of heading angle error standard score

5.4 Driver Interaction Strategies

Five different driver interaction strategies were identified from the data: Adaptation, Persistence, Selective Persistence, Nonintervention and Uncertainty.

5.4.1 Adaptation

Drivers who choose the adaptation interaction strategy, showed willingness to adapt their driving line to follow the system target over time. These drivers avoided conflicts with the assistance system, i.e., whenever the assistance torque and the column torque opposed, they actively adjusted their steering actions to meet the system steering target. The driven line of these drivers approaches the optimal line over time. Drivers who were able to follow the optimal line from the beginning, are also included in this category.

5.4.2 Persistence

Drivers who choose the persistence interaction strategy have a clear path target they want to follow. They do not avoid conflicts with the system and assert their actions against the system. There is barely any deviation in their driven line from one lap to another, regardless of the assistance system intervention intensity.

5.4.3 Selective Persistence

Drivers who choose the selective persistence interaction strategy typically have their own path target they would follow. However, depending on the assistance intervention intensity, they would leave the control to the system in specific situations. Thus, these drivers are willing to address conflicts up to a certain limit but avoid engaging beyond that threshold.

5.4.4 Nonintervention

Drivers who choose the nonintervention interaction strategy let the system handle the driving task almost entirely. They take on a mostly passive role with minimal intervention.

5.4.5 Uncertainty

Drivers who choose the uncertainty interaction strategy are drivers who did not show any consistent pattern in their interaction with the system. They have neither approached the optimal line over time, nor showed a recognized clear path target, nor could they sustain a stable control of the steering wheel.

5.5 Classification

5.5.1 Classifier Design

We label each of the 58 drivers with his observed interaction strategy among the five interaction strategies: Adaptation, Persistence, Selective Persistence, Nonintervention and Uncertainty. Then, we split the data set of all labeled drivers into two sets: a training set X^{tr} containing 80% of the data, and a testing set X^{t} containing 20% of the data. We propose to utilize the MathWorks MATLAB Classification Learner App [105] with k-fold cross-validation to do the classification stage. This MATLAB resources allows to apply and compare the classification results of various classifiers. The workflow for training the classifier in the Classification Learner App is illustrated in Figure 5.7.

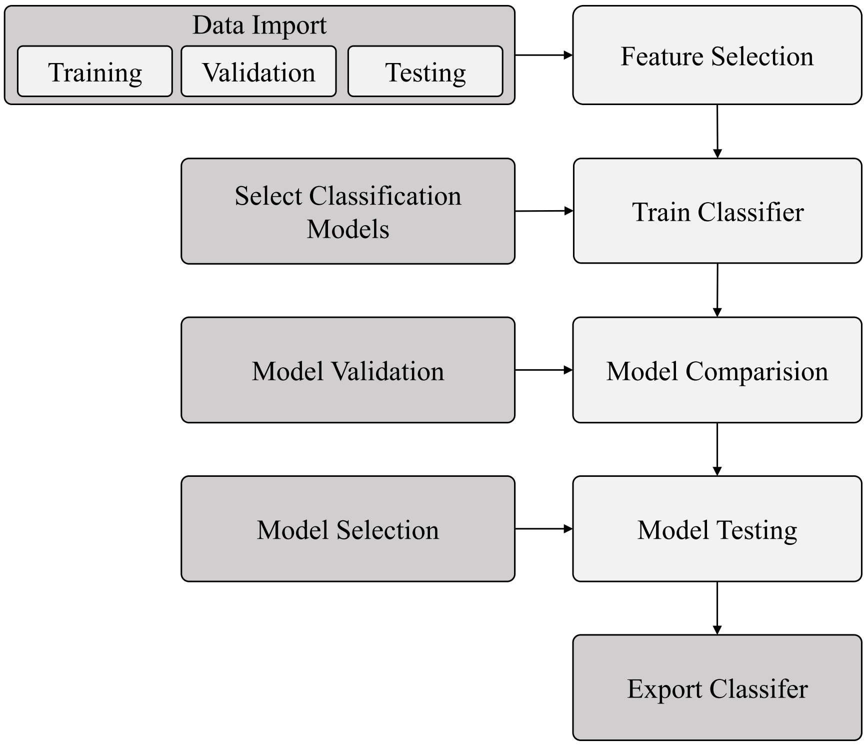


Figure 5.7: Workflow for training a classifier in the MATLAB Classification Learner App

5.5.2 Feature Selection

To assess the suitability of the developed features for the classifier design, various feature combinations will be investigated. We specify that all feature combinations will include the fixed features given in Table 5.7. All additional features are selected from one of two decision options, A or B, represented in Table 5.8.

Table 5.7: Fixed classification features included in all feature combinations

Feature	Fixed
Conflict	$\overline{\lvert T_D^c \rvert}$, $T^c_{D,\max}$, $\widetilde{\Delta t}_c$
Activity	$\overline{\lvert T_D^s \rvert}$, $T^s_{D,\max}$, $\widetilde{\Delta t}_p$
Path consistency	σ_d, $\sigma_{\Delta\psi}$
Individual path pattern	IT
Path quality and adaptation	$\overline{G}$, g_{end}

Table 5.8: Additional classification features chosen based on the decision between option A and B

Decision	Option A	Option B
1	Choosing only absolute durations $\widetilde{\Delta t}$	Choosing only relative durations $\widetilde{\Delta t}_{rel}$
2	Considering $\widetilde{\Delta t}_{c,\max}$ and $\widetilde{\Delta t}_{p,\max}$	Not considering $\widetilde{\Delta t}_{c,\max}$ and $\widetilde{\Delta t}_{p,\max}$
3	Considering the value FT	Not considering FT
4	Considering $\max\|d\|$, $\max\|\Delta\psi\|$, $\max\|\Delta\widetilde{d}\|$ and $\max\|\Delta\widetilde{\psi}\|$	Not considering $\max\|d\|$, $\max\|\Delta\psi\|$, $\max\|\Delta\widetilde{d}\|$ and $\max\|\Delta\widetilde{\psi}\|$
5	Computing the features over all driven paths	Computing the features only over difficult parts of the paths

Based on the fixed features in Table 5.7 and the five decisions to be made in Table 5.8, a total of 32 different feature vectors is constructed.

5.5.3 Results and Validation

We apply the MathWorks MATLAB Classification Learner App with k-fold cross-validation to the data set for 32 different combinations of the feature vector and choose the feature vector with the best accuracy, i.e. the vector who leads to the highest percentage of correctly classified observations. The training results with $k = 5$ folds for different SVM an NN techniques are summarized in Table 5.9.

Table 5.9: Classification Results. *© 2024, IEEE*

Classifier	Hyperparameters	Accuracy (Validation)	Accuracy (Test)
Linear SVM	linear kernel	76.1%	50.0 %
Quadratic SVM	quadratic kernel	69.6%	75.0 %
Cubic SVM	cubic kernel	**76.1**%	**75.0** %
Narrow NN	1st layer size = 10	**82.6**%	**83.3** %
Medium NN	1st layer size = 25	76.1%	66.7 %
Wide NN	1st layer size = 100	73.9%	83.3 %

We see that SVM with a cubic kernal achieves higher accuracy compared to linear and quadratic SVM. The highest overall accuracy is achieved in both validation and testing by the Narrow NN. These results are achieved with the feature vector containing the fixed features, option A for decision 2, 3 and 4, and option B for decision 1 and 5.
In Figure 5.8, the confusion matrix illustrates the performance of the narrow NN across the five classes for the test dataset (12 drivers). The majority of the drivers were classified correctly, only two drivers were misclassified, both into the Selective Persistence class. This

suggests a further refinement of the selected classification features especially to characterize this interaction strategy.

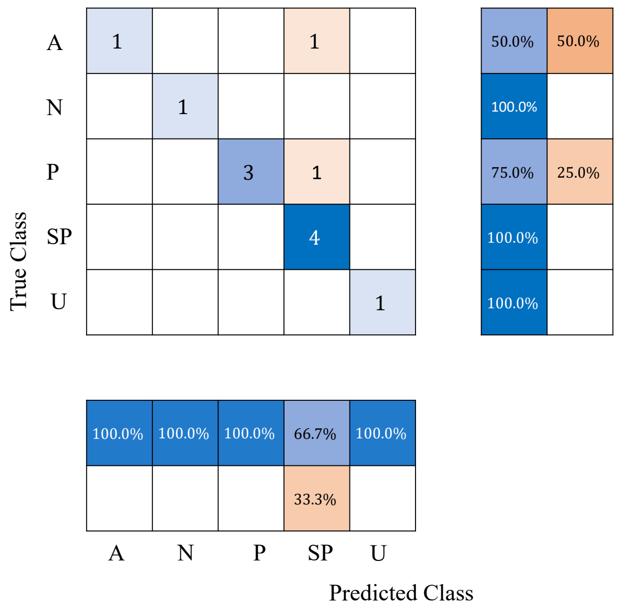


Figure 5.8: Confusion matrix of the narrow NN classification of the drivers (test data) into the classes Adaptation (A), Persistence (P), Selective Persistence (SP), Nonintervention (N) and Uncertainty (U). © *2024, IEEE*

The results confirm that drivers choose different strategies when interacting with an active assistance system, regardless of their driving experience in *common* driving. The majority of the drivers (43.1 %) chose the selective persistence strategy. This indicates that drivers' trust and their use of the system, depend on specific situations, especially on the intervention intensity of the system. However, the classification results confirm the need for further focus on these specific situations to extract enhanced features for this class.
Among all drivers, 24.1% chose the adaptation strategy and 12% the nonintervention strategy, and nearly 20% of both categories were experts. This shows that there exists a level of acceptance of opening up to new ways of manual driving. However, 13.8% of the drivers followed the persistence strategy. They did not allow to cooperate with the system to share the driving task. The assistance system in this case can be perceived as annoying and disturbing. On the other hand, a small part of the drivers (7%), categorized in the uncertainty strategy, did not show any clear pattern when interacting with the system. This could be explained by a lack of understanding the system target and being overwhelmed with its intervention. However, the classification rate for this category requires further investigation, since we did not implement features specifically for this class, due to the absence of clear patterns in this category.
To further validate the plausibility of the classification results, we compare our categorization with the subjective assessment of the instructor. The results are illustrated in Figure 5.9.
We see that no expert driver was classified in the uncertainty interaction strategy and that

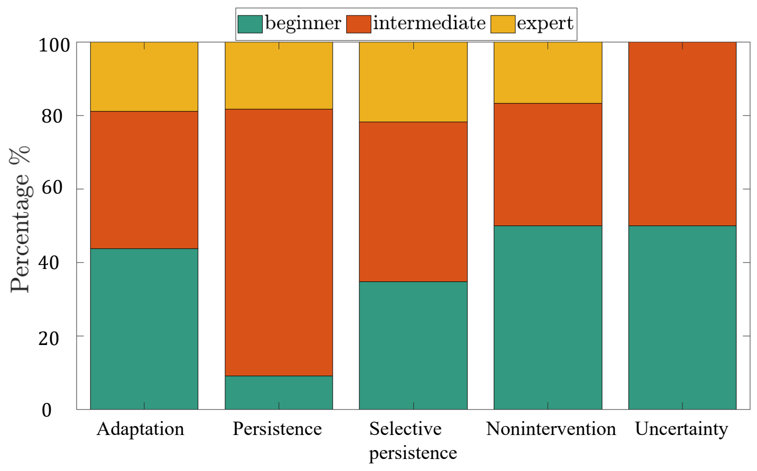


Figure 5.9: Comparison of the driver classification based on their interaction strategy with the subjective assessment. © *2024, IEEE*

the majority of the drivers in the nonintervention strategy are beginners. This observation appears to be plausible and in line with our expectations.

5.6 Interaction Analysis with DMD

For deeper insight into the dynamic interactions between the driver and the assistance system, we apply Dynamic Mode Decomposition with control (DMDc) to the classified driver data from the previous section. The main goal is to extract the modes that characterize the system's behavior for each driver class. We consider the vehicle states Δd (lateral deviation), $\Delta\psi$ (heading angle error), $\Delta\dot{\psi}$ (heading error rate) and v_y (lateral velocity). As control input, we consider the current steering angle δ. We arrange the data into snapshots and apply DMDc, following Algorithm 7.
First, we analyze the distribution of the modes across all drivers to understand the underlying dynamics. The results are illustrated in Figure 5.10.
Mode 1 is dominated by th lateral speed v_y, followed by the heading error rate $\Delta\dot{\psi}$. In contrast, the lateral deviation and the heading angle error's contributions to this mode are very low. The eigenvalues associated to this mode are real and range from 0.67 to 0.89 across all drivers. Hence, the dynamics decay at a moderate rate. This suggests that this mode is related to the lateral stability dynamics of the vehicle and the corrective behavior, as this gradually damp out as the system stabilizes.
Mode 2 and 3 are characterized by complex conjugate eigenvalues. The real parts of the eigenvalues are close to 1 and the imaginary parts are small (between -0.006 and 0.006). These modes are dominated by the lateral deviation Δd, followed by the heading angle error $\Delta\psi$, which are associated to the alignment of the vehicle to the reference path. This suggests that these modes represent the lateral path-tracking dynamics. The real parts of the eigenvalues shows that these modes are highly persistent. Hence, the lateral deviation and

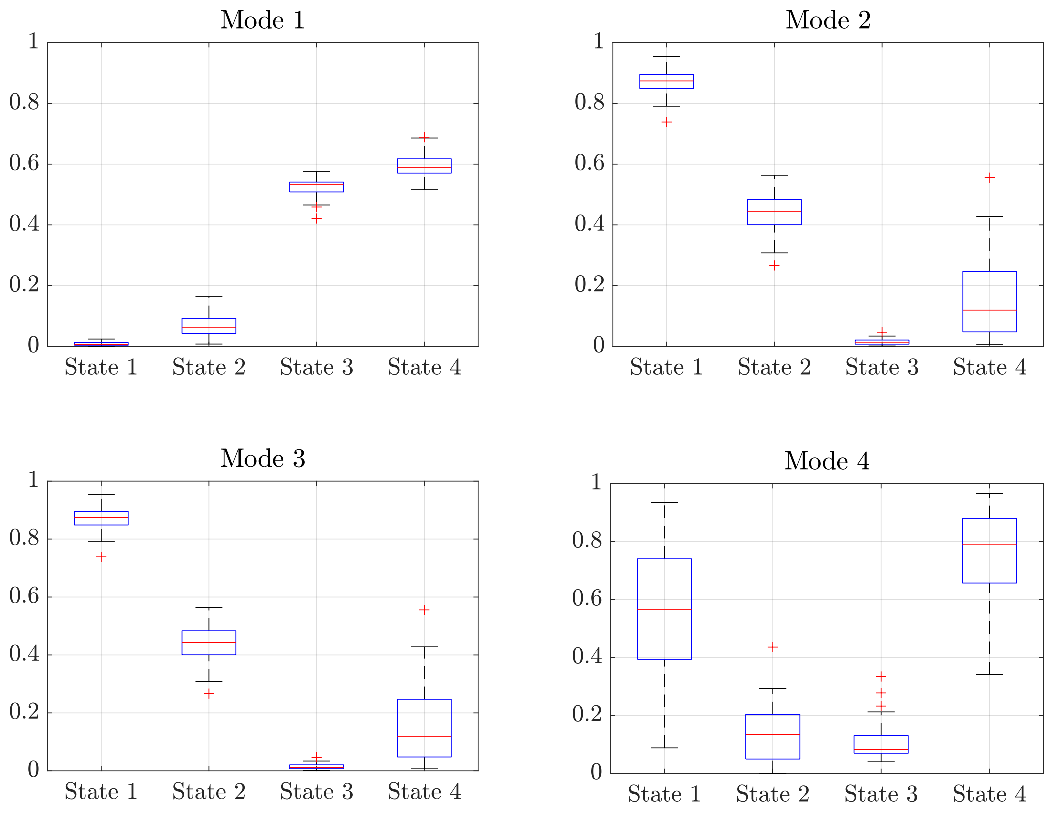


Figure 5.10: DMD modes distribution across all drivers. State 1: Δd, State 2: $\Delta\psi$, State 3: $\Delta\dot{\psi}$, State 4: v_y .

the heading angle error are gradually corrected without abrupt destabilizing actions. The small imaginary parts correspond to slow oscillations in the dynamics. This could reflect the interaction between the driver inputs and the assistance interventions.
Mode 4 shows more distribution across drivers compared to the other modes. The eigenvalues are real and close to 1. The lateral velocity and the lateral deviation are more dominant than the heading angle and rate. This indicates stable dynamics and a focus on maintaining steady lateral motion. The variability across the different drivers could be explained by a more generalized stabilization task of the dynamics compared to mode 2 and 3.
We analyze now the dynamic mode decomposition of the different driver classes. The distribution of the modes across the drivers with the persistence interaction strategy is illustrated in Figure 5.11.
Mode 1 aligns with the overall trends. This confirms that this mode focuses on the lateral stability regardless of the driver cooperation. For mode 2 and 3, we observe lower contribution of the lateral velocity v_y. This can be explained by the fact that the system prioritize correcting the high lateral deviations and heading angle errors, that are associated with this interaction strategy.
The distribution of the modes across drivers with the nonintervention interaction strategy is illustrated in Figure 5.12.

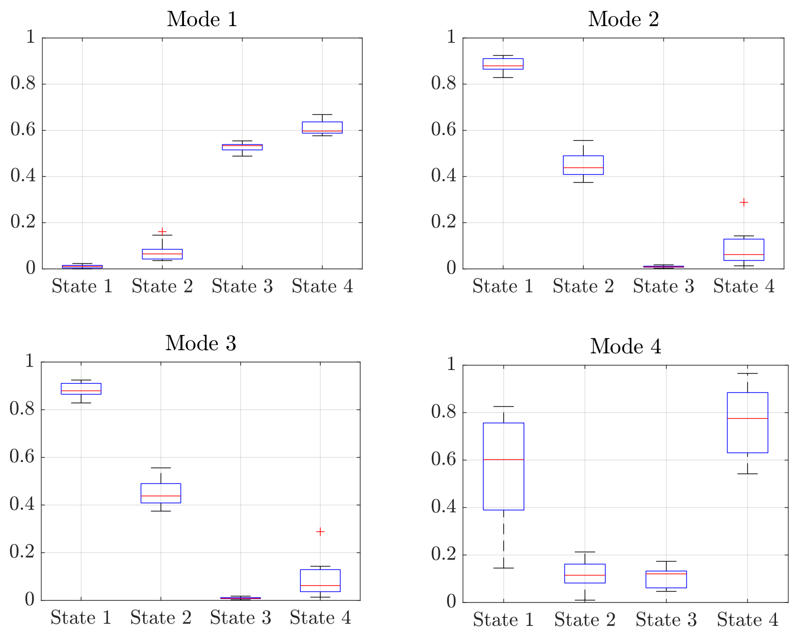


Figure 5.11: DMD modes distribution across drivers with the persistence interaction strategy. State 1: Δd, State 2: $\Delta\psi$, State 3: $\Delta\dot{\psi}$, State 4: v_y .

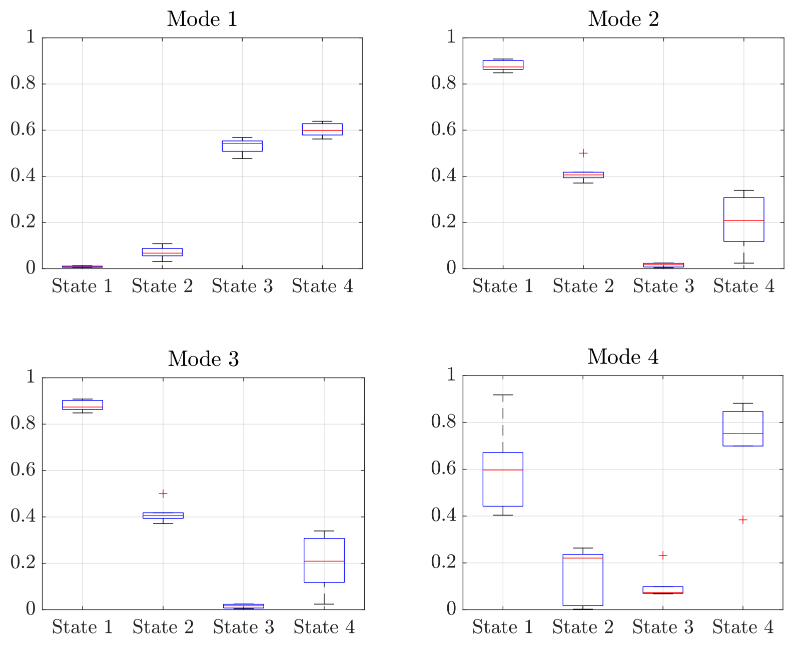


Figure 5.12: DMD modes distribution across drivers with the nonintervention interaction strategy. State 1: Δd, State 2: $\Delta\psi$, State 3: $\Delta\dot{\psi}$, State 4: v_y .

As expected, mode 1 aligns closely with the overall trend. The low contribution of Δd confirms that these drivers mainly rely on the system to maintain the path. For mode 2 and 3 we observe that the distribution of the lateral deviation, the heading angle error and the heading angle rate across the drivers shows only minimal variation. This confirms similar path-tacking dynamics, when the system is taking over the control. In contrast, mode 4 is not entirely invariant across this category. We argued that this mode is connected to the overall stability of the system. The variation across the drivers, could be explained by the individual differences, on how the drivers hold the steering wheel. Their grip or hand positioning could unconsciously contribute to the dynamics, even when they are not actively steering.
We apply DMDc to the data of the adaptation interaction strategy. The distribution is illustrated in Figure 5.13.

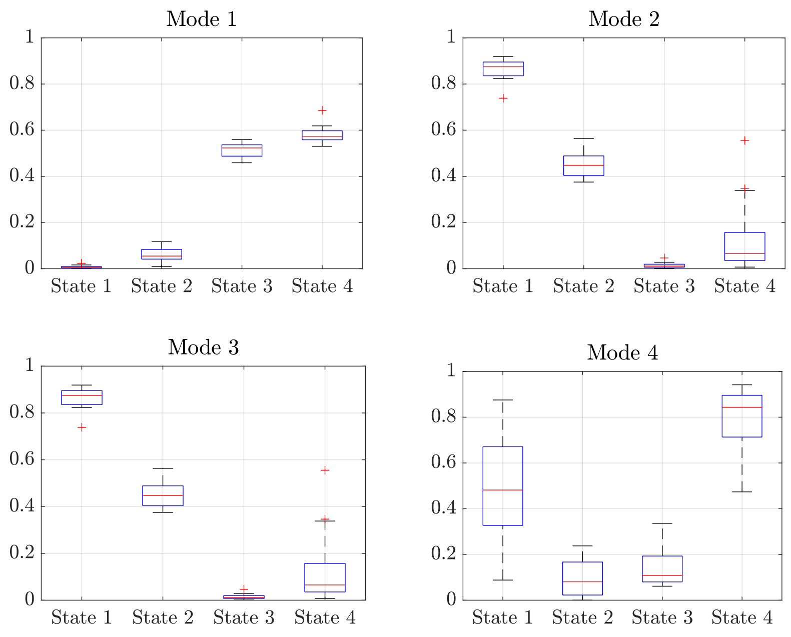


Figure 5.13: DMD modes distribution across drivers with the adaptation interaction strategy. State 1: Δd, State 2: $\Delta\psi$, State 3: $\Delta\dot{\psi}$, State 4: v_y .

The three first modes follow the overall trend. We remark a slightly lower influence of the lateral velocity to mode 2 and 3. This reflects a smoother path tracking without significant lateral deviation. Mode 4 suggests stable dynamics. The contribution of Δd shows variation across these drivers. This can be explained by the fact, that this category includes the drivers who adapt gradually to the system over time, as well as those, who tested or monitored the system in their first interaction.
Next, we consider the results for the selective persistence interaction strategy in Figure 5.14. The distribution of the modes can be seen as a combination of the results from the persistence and the nonintervention interaction strategy. The higher number of outliers we notice compared to the other categories, could indicate the variability of the interaction in this category, when drivers switch from persistence to nonintervention.

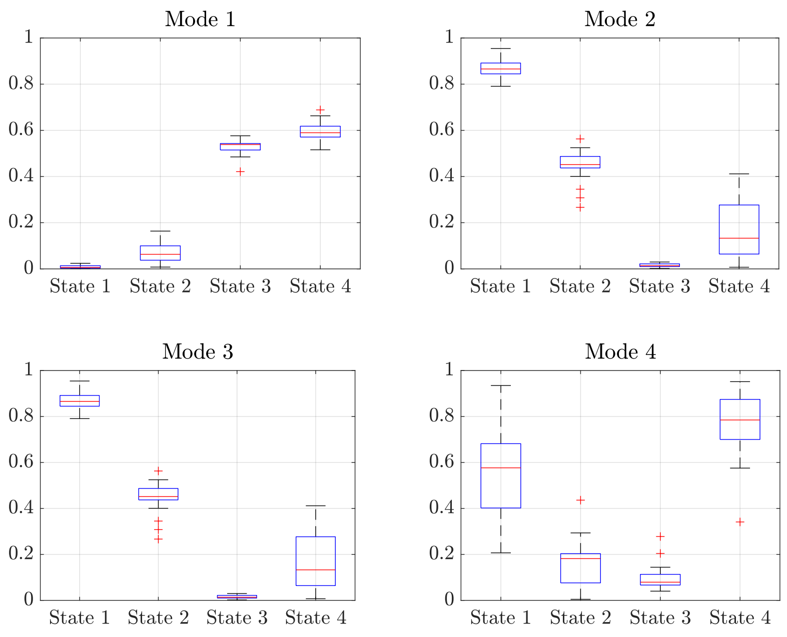


Figure 5.14: DMD modes distribution across drivers with the selective persistence interaction strategy. State 1: Δd, State 2: $\Delta\psi$, State 3: $\Delta\dot{\psi}$, State 4: v_y .

Finally, we consider two drivers classified in the uncertainty interaction strategy. The DMD modes are illustrated in Figure 5.15.

The main observation in the modes, is the lower contribution of the heading angle error in mode 2 and 3, despite the higher contribution of Δd. This could confirm the lack of understanding: when drivers only focus on one aspect of control, i.e., they try to keep the vehicle on the path (minimizing the lateral deviation), they may neglect the other elements of the task (the correct orientation).

The analysis of the dynamic modes across the different driver interaction strategies supports the validation of the classification results. The trends we observed in the modes and their dominant contribution states align with the expected behavior of each category. Especially, the path-tracking dynamics (mode 2 and 3) and the stability dynamics (mode 4) showed different characteristics across the classes. However, some modes showed variability and overlap with other classes, especially for the selective persistence interaction strategy we could not distinguish clear patterns. The data set may lack sufficient dynamic variation to fully capture all the interaction strategies. Additionally, the analysis of the uncertainty interaction strategy, for which we did not implement specific features, revealed one lack of understanding regarding how to balance lateral deviation minimization and heading angle correction. This insight could be valuable to identify patterns for this driver category. The application of DMD provided a method to validate the classification results by understanding the underlying dynamics. However, to strengthen the findings additional data across different driving scenarios and driver interaction strategies is necessary.

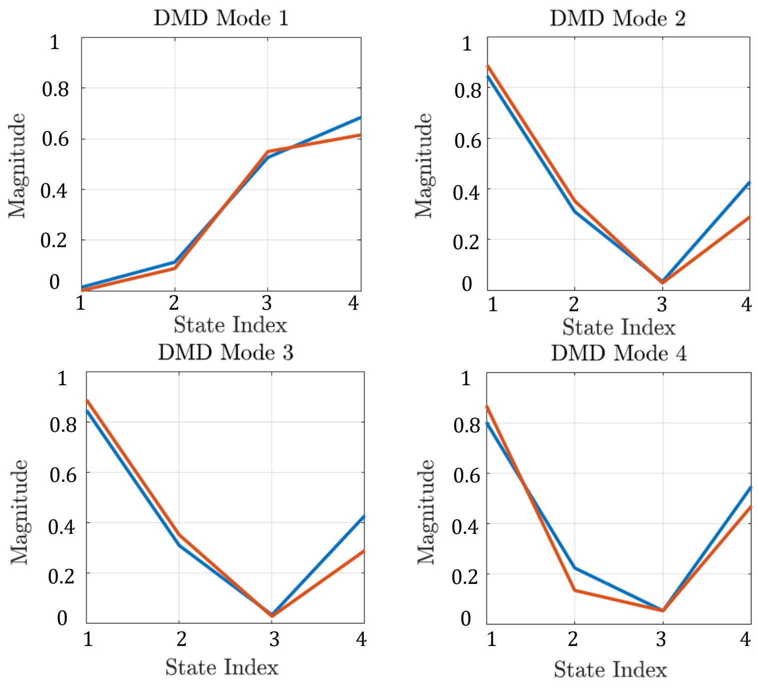


Figure 5.15: DMD modes with drivers with the uncertainty interaction strategy. State 1: Δd, State 2: $\Delta\psi$, State 3: $\Delta\dot{\psi}$, State 4: v_y .

6 Concepts for Adapting System Behavior

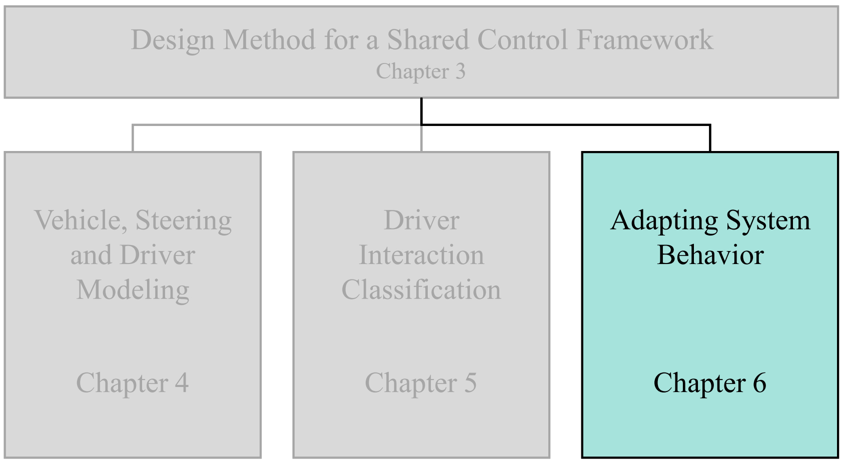

Design Method for a Shared Control Framework
Chapter 3
Vehicle, Steering and Driver Modeling
Chapter 4
Driver Interaction Classification
Chapter 5
Adapting System Behavior
Chapter 6

Figure 6.1: Scope of Chapter 6

In this chapter, we design concepts for adapting the control architecture of a steering assistance system to the interaction strategy of the driver. In Section 6.1, we design and formulate the cost function and the optimization constraints, and introduce the adjustable parameters of a Model Predictive Control (MPC). In Section 6.2, we map these adjustable parameters to the desired system behavior for each driver interaction strategy. Section 6.3 details the numerical solver essential for the framework. Next, Section 6.4 presents the workflow and architecture of the implemented framework. In Section 6.5, we focus on the real-time computation of the classification features. Then, Section 6.6 is dedicated to the performance analysis and results of the implemented MPC. Section 6.7 and Section 6.8 explore the real-world applicability of the framework and its limitations. Finally, in Section 6.9, we propose further concepts for adapting system behavior in shared control based on independent components of the designed framework.

6.1 Cost Function and Constraints Design

6.1.1 Key Objectives

In this section, we outline the main objectives the Model Predictive Control (MPC) aims to achieve within the context of the shared control framework presented in Chapter 3.
First, we briefly describe the general concept of how the MPC operates. Given a reference path over a track, whenever the vehicle deviates from this path, the MPC plans a corrective

trajectory to bring the vehicle back to the reference line. This involves continuously predicting the vehicle's future states and optimizing the control inputs to minimize the deviation from the desired path, taking into account the vehicle's current state, the driver input and the track information.

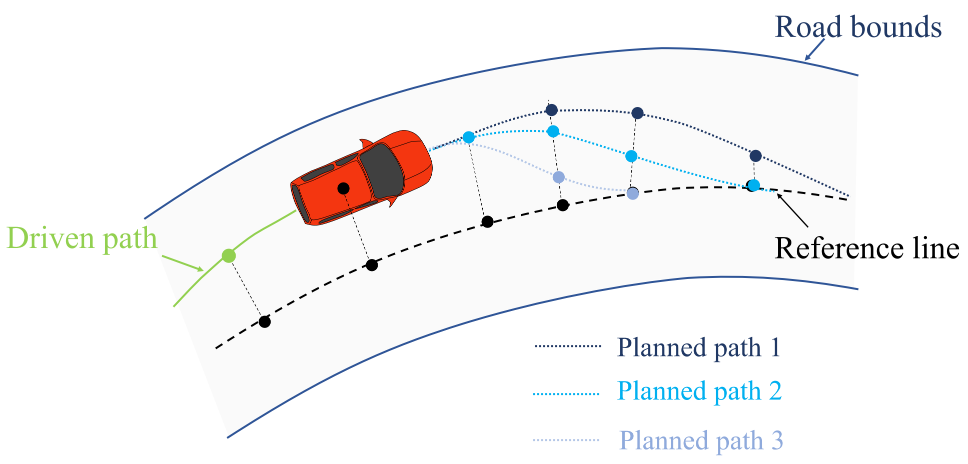


Figure 6.2: Illustration of three planned corrective paths.

Figure 6.2 illustrates three different planned paths to correct the vehicle's course when it deviates from the reference. Each path corresponds to a specific configuration of the cost function depending on the interaction strategy of the driver. To accurately model these configurations and achieve a safe and comfortable planned path, the MPC fulfills specific objectives.

Path Tracking and Orientation Accuracy

A fundamental objective of a lateral assistance system, is to ensure that the vehicle remains close to the reference path. Hence, minimizing the deviation from the reference line Δd as well as the heading angle error $\Delta\psi$ is crucial. For this objective, we define J_1 as

$$J_1 = \alpha \cdot \Delta d^2 + \beta \cdot \Delta\psi^2, \tag{6.1}$$

where α and β are weighting parameters. To account for the different driver interaction strategies, and thus also for drivers who follow their own path target, we propose to define the deviation from the reference as soft constraints with a penalty term. Thus, a slack variable is integrated into the cost function, to allow deviation from the reference.
Let Δd^f_{min} and Δd^f_{max} be the minimal and maximal allowed deviations of the front wheel from the reference line, and Δd^r_{min} and Δd^r_{max} be the minimal and maximal allowed deviations

of the rear wheel from the reference line, respectively. Then, the actual deviation from the reference line fulfills

$$\Delta d^f_{\text{min}} - s_1 \leq \Delta d + l_f \Delta\psi \leq \Delta d^f_{\text{max}} + s_1, \tag{6.2a}$$
$$\Delta d^r_{\text{min}} - s_2 \leq \Delta d - l_r \Delta\psi \leq \Delta d^r_{\text{max}} + s_2, \tag{6.2b}$$

where $s_1, s_2 \geq 0$ are slack variables integrated into the cost function J_2 as

$$J_2 = \mathbf{s}^T \mathbf{F}_s \mathbf{s}, \tag{6.3}$$

with $\mathbf{F}_s$ being a weighting matrix and $\mathbf{s} = [s_1 \;\; s_2]^T$.

Control Effort and Smoothness

To ensure a smooth control, steering angle and steering rate must be applied efficiently and gradually, i.e., abrupt steering inputs and rapid changes in steering direction are to be avoided. In the vehicle model (4.19), the steering angle is used as the control variable. To allow the MPC to directly control not only the steering angle but also its speed of change, we propose to add the steering angle δ into the state vector as an additional state and use the steering rate $\dot{\delta}$ as a control input instead. The considered state vector $\mathbf{y}$ for the MPC cost function becomes

$$\mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \Delta d \\ \Delta\psi \\ \dot{\psi} \\ v_y \\ \delta \end{bmatrix}. \tag{6.4}$$

Then, the steering rate $\dot{\delta}$ is used as a control input and integrated in the cost function J_3 with

$$J_3 = \dot{\delta}^T \mathbf{F}_{\dot{\delta}} \dot{\delta}, \tag{6.5}$$

where $\mathbf{F}_{\dot{\delta}}$ is a weighting matrix.

Boundary Compliance

To ensure a safe vehicle operation, the road bounds must be continuously monitored and respected. However, defining the road bounds as hard constraints may cause infeasibility of the optimization problem, especially if the driving scenario requires quick and dynamic changes. To account for the different driver interaction strategies, and thus also for drivers who require boundary flexibility to execute sharp turns, let $d^{f,l}_{\text{max}}$ and $d^{f,r}_{\text{max}}$ be the maximal allowed deviations to the left and right of the front wheel from the reference line, and $d^{r,l}_{\text{max}}$ and $d^{r,r}_{\text{max}}$ be the maximal allowed deviations to the left and right of the rear wheel from the reference, respectively. Then, we set $\Delta d^{f,w}_{\min} = w\ d^{f,l}_{\max}$, $\Delta d^{f,w}_{\max} = w\ d^{r}_{\max}$, $\Delta d^{r,w}_{\min} = w\ d^{r,l}_{\max}$ and $\Delta d^{r,w}_{\max} = w\ d^{r}_{\max}$, with $w \in [0,1]$, and rewrite (6.2) as

$$\Delta d^{f,w}_{\min} - s_1 \leq \Delta d + l_f \Delta\psi \leq \Delta d^{f,w}_{\max} + s_1, \tag{6.6a}$$

$$\Delta d^{r,w}_{\min} - s_2 \leq \Delta d - l_r \Delta\psi \leq \Delta d^{r,w}_{\max} + s_2. \tag{6.6b}$$

6.1.2 MPC Problem Formulation

Let $\mathbf{x}_k$ be the state vector introduced in (4.18) and $\mathbf{u}_k = \delta_k$ the steering angle, both at time step k. Then, the state vector of the MPC problem at time step k is

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}. \tag{6.7}$$

Cost Function

We define the 5×5 matrix $\mathbf{F}_{\mathbf{y},k}$ as

$$\mathbf{F}_{\mathbf{y},k} = \begin{bmatrix} \alpha & & & 0 & \\ & \beta & & & \\ & & 0 & & \\ & & & 0 & \\ & 0 & & & \zeta \end{bmatrix}, \tag{6.8}$$

and rewrite (6.1) as

$$J_1 = \mathbf{y}^T\ \mathbf{F}_{\mathbf{y}}\ \mathbf{y}. \tag{6.9}$$

Then, summing up (6.3), (6.5) and (6.9) at time step k leads to the total cost function

$$J_k = \frac{1}{2}\big(\mathbf{y}_k^T\ \mathbf{F}_{\mathbf{y},k}\ \mathbf{y}_k + \dot{\delta}_k^T \mathbf{F}_{\dot{\delta},k} \dot{\delta}_k + \mathbf{s}_k^T\ \mathbf{F}_{\mathbf{s},k}\ \mathbf{s}_k\big). \tag{6.10}$$

Constraints

The control variable $\dot{\delta}$ of the MPC problem is subject to box-constraints, with bounds

$$\dot{\delta}_{\min} \leq \dot{\delta} \leq \dot{\delta}_{\max}. \tag{6.11}$$

To formulate the inequality constraints, we set physical upper and lower bounds, $\delta_{\min}$ and $\delta_{\max}$, respectively, for the steering angle δ. This leads to

$$\mathbf{C}_{\mathbf{y}}\mathbf{y} + \mathbf{f}_{\mathbf{y}} \leq 0, \tag{6.12}$$

where

$$\mathbf{C_y} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \tag{6.13}$$

and

$$\mathbf{f_y} = \begin{bmatrix} -\delta_{\max} \\ \delta_{\min} \end{bmatrix}. \tag{6.14}$$

Then, we define the matrix $\mathbf{C_{ys}}$ as

$$\mathbf{C_{ys}} = \begin{bmatrix} 1 & l_f & 0 & 0 & 0 \\ -1 & -l_f & 0 & 0 & 0 \\ 1 & -l_r & 0 & 0 & 0 \\ -1 & l_r & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.15}$$

the matrix $\mathbf{C_s}$ as

$$\mathbf{C_s} = \begin{bmatrix} -1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}, \tag{6.16}$$

and the vector $\mathbf{f_s}$ as

$$\mathbf{f_s} = \begin{bmatrix} -\Delta d^{f,w}_{\max} \\ \Delta d^{f,w}_{\min} \\ -\Delta d^{r,w}_{\max} \\ \Delta d^{r,w}_{\min} \\ 0 \end{bmatrix}. \tag{6.17}$$

Additionally, we consider (6.6) and $\mathbf{s} \geq 0$. Then, this implies

$$\mathbf{C_{ys}y} + \mathbf{C_s s} + \mathbf{f_s} \leq 0. \tag{6.18}$$

Problem Formulation

Let N be the length of a prediction horizon and let $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{r}$ be defined as in (4.23). Then the MPC is formulated as

$$\begin{aligned}
\min_{\{(\mathbf{y}_k,\dot{\delta}_k)\}_{k=1}^{N}} \quad & J = \sum_{k=1}^{N} \frac{1}{2}\left(\mathbf{y}_k^T\,\mathbf{F}_{\mathbf{y},k}\,\mathbf{y}_k + \mathbf{F}_{\dot{\delta},k}\dot{\delta}_k^2 + \mathbf{s}_k^T\,\mathbf{F}_{\mathbf{s},k}\,\mathbf{s}_k\right) && \text{(6.19a)}\\
\text{subject to} \quad & \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\,\mathbf{u}_k + \mathbf{r}_k,\ k = 1,\cdots,N-1, && \text{(6.19b)}\\
& \mathbf{C_y y} + \mathbf{f_y} \leq 0, && \text{(6.19c)}\\
& \mathbf{C_{ys}y} + \mathbf{C_s s} + \mathbf{f_s} \leq 0, && \text{(6.19d)}\\
& \dot{\delta}_{\min} \leq \dot{\delta}_k \leq \dot{\delta}_{\max}. && \text{(6.19e)}
\end{aligned}$$

6.2 Interaction Strategy Based Behavior

In Section 6.1, we introduced the various parameters that can be adjusted within the optimization problem to adapt to the different driver interaction strategies. In addition, the choice of the type and configuration of the steering mechanism as introduced in Section 4.2 is crucial, to account for each strategy. Table 6.1 summarizes all the adjustable components. In this section, we will propose desired system behavior for each interaction strategy.

Table 6.1: Adjustable parameters for Optimization

Adjustable components	Description
N	Prediction horizon
α	Weight on lateral deviation Δd
β	Weight on heading angle error $\Delta\psi$
ζ	Weight on steering angle δ
$\mathbf{F_s}$	Weight matrix on slack variable for boundary compliance
$\mathbf{F}_{\dot{\delta}}$	Weight matrix for steering rate
w	Lateral deviation scaling factor
ω_1	Weight parameter for driver input in decoupled steering
ω_2	Weight parameter for system input in decoupled steering

6.2.1 Adaptation

For the adaptation strategy, moderate weights for path-tracking and orientation accuracy α and β are used. The penalty on the steering rate $\mathbf{F}_{\dot{\delta}}$ as well as the upper and lower bounds for $\dot{\delta}$ are also set to a moderate level, to allow smooth adjustments. To provide a smooth path for the driver, a medium prediction horizon N is chosen for the MPC. The use of slack variables is moderate to provide flexibility in adapting to the optimal line. For drivers with this interaction strategy, a coupled steering system is preferred, as they need the haptic feedback of the system's actions while they gradually adapt to its target.

6.2.2 Persistence

For the persistence interaction strategy, the input of the driver is prioritized. To avoid conflicts, the weights for the tracking accuracy are decreased. In contrast, the weights of the control input are increased to avoid strong corrective actions of the system and its lower and upper bounds are restricted. Concerning the lateral deviation, the driver gets higher scaling factors w, and thus, have more flexibility in using the whole width of the road. For this strategy, the system focuses on immediate corrections rather than future path tracking accuracy. Hence, a shorter prediction horizon N is chosen. To support the driver in following his own target, a coupled steering system is preferred.

6.2.3 Selective Persistence

The MPC configuration of the selective persistence strategy incorporates aspects of the configurations of the adaptation ans the persistence strategies. Depending on whether the driver takes over, the weights of path-tracking and orientation accuracy are adjusted: higher weights when the system is in control and lower when the driver is in control. The penalty on the steering rate $\mathbf{F}_{\dot{\delta}}$ is moderate to high to ensure smooth transitions between driver and system control. The bounds for $\dot{\delta}$ are flexible to accommodate varying levels of driver intervention. A medium prediction horizon N is chosen to balance immediate corrections with future path planning. The use of slack variables is higher to provide the necessary flexibility for the driver to take control when needed. A decoupled steering system is suitable to allow seamless transitions between driver and system control.

6.2.4 Nonintervention

For the nonintervention interaction strategy, the system's control over the vehicle is maximized and the need for driver intervention is minimized. Both the weights α and β are set to high values, to increase the path-tracking and orientation accuracy. In contrast, $\mathbf{F}_{\dot{\delta}}$ is minimized to reduce the penalty on the steering rate and the upper and lower bounds for $\dot{\delta}$ are relaxed, to enable higher control effort. As the system is primarily in control, the use of slack variables is minimized. In addition, the prediction horizon N can be maximized, to allow the MPC to plan further ahead. Concerning the steering mechanism, both a coupled as well as a decoupled steering system can be used for this driver category, with increased weights for the system input.

6.2.5 Uncertainty

To account for the unpredictability of the drivers with the uncertainty interaction strategy, a short prediction horizon N is chosen. In addition, the weights of the steering rate is increased to reduce abrupt corrective actions of the system. A higher use of slack variables is crucial, to provide the needed flexibility in correcting abrupt behavior. Decoupled steering is suitable for this interaction strategy. It helps to stabilize the vehicle by combining the system's corrective actions with the driver's inputs.

6.3 Numerical Solver for Real-Time Optimization

6.3.1 Quadratic Programming Solver

To efficiently solve real-time Quadratic Programs, we propose to apply the QP solver `qpOASES`[1] in its version 3.2.
`qpOASES` is a robust `C++` implementation of the online active set method presented in Algorithm 2. The software package is open-source and is available at [106]. A user manual is provided in [107]. According to [108], the `qpOASES` software provides the following features:

- It is written in an object-oriented manner with separate classes for different QP problem types: the class `QProblem` for single QP of the standard form (2.45), the class `QProblemB` for QP with only box-constraints, and the class `SQProblem` for QP with varying matrices.
- All matrix-vector operations are easy to adjust for the specific characteristics of the problem. The implemented linear algebra class `MATRIX` allows an easy switch between the specific linear algebra techniques for dense and sparse matrices. Within the system, dense matrices are stored in arrays, while sparse matrices are stored using either row-compressed or column-compressed formats.
- The implemented class `Options` allows an easy access to all the algorithmic details, such as the treatment of equality constraints, the choice of the initial working set, handling rounding errors and ill-conditioning...
- `qpOASES` offers various interfaces that enable easy integration with third-party software packages, such as MATLAB, Simulink, dSpace[2]... This integration capability makes it possible to implement and test the software package in real-world applications.

Let n be the number of variables and m the number of constraints, then, following [109], the computational complexity of `qpOASES` is analyzed based on one iteration in Algorithm 2:

- The computational complexities of the steps 1, 4, 5, and 6, grow linearly in n and m. Thus, it is of order $O(n+m)$.
- Due to the structure of the coefficient matrix in step 2, the computational complexity of this step is $O(n^2)$.
- The computational complexity of step 3 mainly depends on the matrix-vector product, and thus, it is of order $O(nm)$.

Therefore, the total computational complexity of a single iteration of `qpOASES` is of order $O(n^2+nm)$.

1 qpOASES: **q**uadratic **p**rogramming **O**nline **A**ctive **SE**t **S**trategy
2 dSpace (digital Signal processing and control engineering) provides tools for developing, testing and calibrating electronic control units (ECUs) across various engineering fields, such as automotive applications, aerospace and robotics. See `https://www.dspace.com/`

6.3.2 Interior Point Solver for MPC

To solve MPC problems, we propose to apply the interior point solver `LIPSOL`[3][110]. This solver is a `C++` implementation of the Mehrotra primal-dual interior point method (Algorithm 4) applied to a multistage problem, as discussed in Section 2.4.4. `LIPSOL` is based on the concepts and methodologies of the publicly available solver `FORCES` [111, 112]. `LIPSOL` provides the following features:

- It supports the general formulation of multistage problems of the form (2.73), which makes it suitable for most MPC and moving horizon estimation problems.
- It provides fast solutions, due to the efficient calculation of matrix structures (see Section 2.4.4).
- `LIPSOL` allows an easy parametrization of the necessary options for the considered problem: the problem dimensions (number of states, inputs, slacks...),the number of constraints (state box constraints, input box constraints, slack box constraints....), the sparsity properties...
- The solver requires matrices and vectors to be converted into one-dimensional arrays. For symmetric matrices, it is sufficient to convert only the lower triangular part.
- It offers Simulink and dSpace interfaces that enable easy integration and test for real-world applications.

6.4 Simulation Model Implementation

6.4.1 Overview

The shared control framework presented in this thesis, is tested through a simulation model, implemented within Simulink. Simulink is ideal for modular development. On one hand, its visual interface allows an easy decomposition of complex systems into different subsystems. On the other hand, it supports the integration of MATLAB functions and `C\C++` code. The overall architecture of the software is illustrated in Figure 6.3.
The architecture of the system mainly consists of six interconnected modules. The workflow begins with the *input module*, where the driver data is gathered. This data is then passed to the steering system and combined with the actual control input within the *blending module*. The resulted steering angle is routed to the *vehicle dynamics module*, where the system state is computed. Based on the system state, the *control module* determines the specific control actions required and routes this information to the *feedback module*. The feedback module then provides a new control input at the steering wheel. The parametrization of the control module depends on the information provided by the *classification module*, which collects data from both the input and vehicle dynamics modules. In the following sections, the implementation of each module will be discussed in detail.

3 LIPSOL: **L**inear **I**nterior **P**oint **SOL**ver

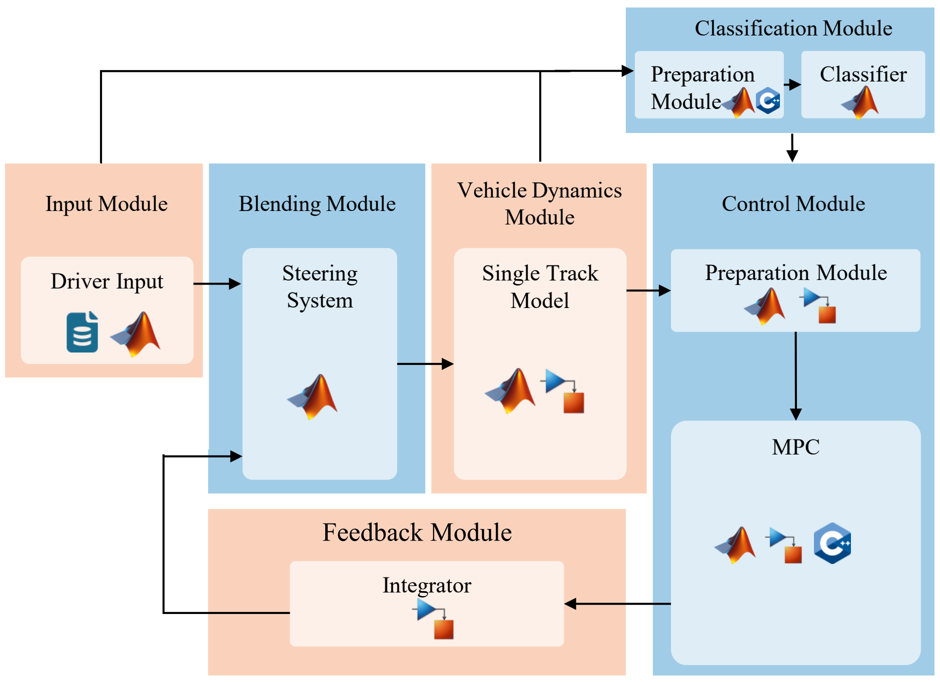


Figure 6.3: Overview of the interconnected software modules in the simulation model.

6.4.2 Input Module

The input module is designed to manage and configure the input data for the simulation. The input data is derived from recorded data of real driving scenarios. The data set includes information from two vehicles and over 50 different drivers that shared the driving task with an active assistance system over the same test track. The data is initially loaded offline and then converted into timetables. These timetables enable the data to be replayed in real-time in the simulation framework. This allows the simulation to closely mirror real driving conditions. In addition to the driving data, the Input module also provides the track information. This includes the pre-processing of road bounds and the provision of a fixed reference trajectory.

6.4.3 Blending and Vehicle Dynamics Modules

The blending module is implemented as a MATLAB function. As inputs, it gets the driver input, the actual control input, and the driver class. It blends the driver input with the control input and computes the combined steering angle, which is then provided as the output. The implemented blending functions are (4.29) and (4.30), where all parameters can be adjusted within the function block.

The vehicle dynamics module is implemented as a MATLAB function as well. It gets the computed steering angle as an input. Then, the differential equation (4.23) is formulated and solved, to compute the system states. To formulate the differential equation, the matrix **A** in (4.24) and the vectors **B** in (4.25) and **r** in (4.26) are computed by parameterizing the vehicle model and using (4.20),(4.21) and (4.22). The vehicle parameters m, l_f, l_r, c_f, c_r, J_z and κ are derived from experimental measurements of a real vehicle.
To solve (4.23), we apply the explicit Euler-Method in each iteration, i.e.,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T \cdot (\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{r}_k) \,. \tag{6.20}$$

Thus, $\mathbf{x}_{k+1}$ represents the new system state, which is then provided as the module's output.

6.4.4 Classification Module

The classification module consists of two main parts. A preparation part and the actual classification task. In the preparation part, input data for an adjustable time horizon is collected and stored. Then, the presented features in Section 5.3 are computed. Each analyzed feature is implemented as a separate MATLAB function. The consistency (Section 5.3.2), as well as the path quality and adaptation features (Section 5.3.4) have only a minimal impact on the computation time.
For the individual path pattern targeting features (Section 5.3.3), the optimal paths in Table 5.5 are computed offline, as we assume that the considered track for our analysis is fixed. Thus, the optimization problems (5.13) and (5.28) are solved in advance using the standard MATLAB quadratic programming solver `quadrprog`. For Algorithm 8, we assume that the number of available intervention intensities $|\mathcal{L}|$ and the considered number of times the same section is crossed $|I|$ are much smaller then the considered track steps for the classification N, i.e., $|\mathcal{L}|, |I| \ll N$. Hence, it is easy to see, that Algorithm 8 has a computational complexity of $O(N)$.
The computation of the conflict and activity features (Section 5.3.1) dominates the overall feature calculation time, since the quadratic program (4.56) is formulated and solved in each step. An illustration of the interface design of the conflict and activity features computation is given in Figure 6.4. In the preprocessing block, the necessary input data is used to construct the vectors y_k, β_k and x_k as in (4.46). Due to the sparsity structure of the vector **c**, the matrix **H**, the matrix **A** and the vector **b** in (4.47), (4.48), (4.49) and (4.53), respectively, the construction of those elements is implemented as `C++` code instead of MATLAB and integrated into the Simulink framework through S-functions. This results in significantly faster execution. To solve the resulting QP, we integrate the `qpOASES` quadratic programming solver presented in Section 6.3.1 into the classification module. It is loaded and compiled via its Simulink interface. To account for sparse matrices, we use its predefined class `SQProblem`. In a postprocessing block, the resulted optimization vector $\mathbf{x}^*$ is split into components to compute the conflict and activity time series (4.40) and (4.44). Then, the corresponding features introduced in Table 5.2 are calculated.
In the actual classification task, the pre-trained classifier presented in Section 5.5 is integrated into the module as a MATLAB function.

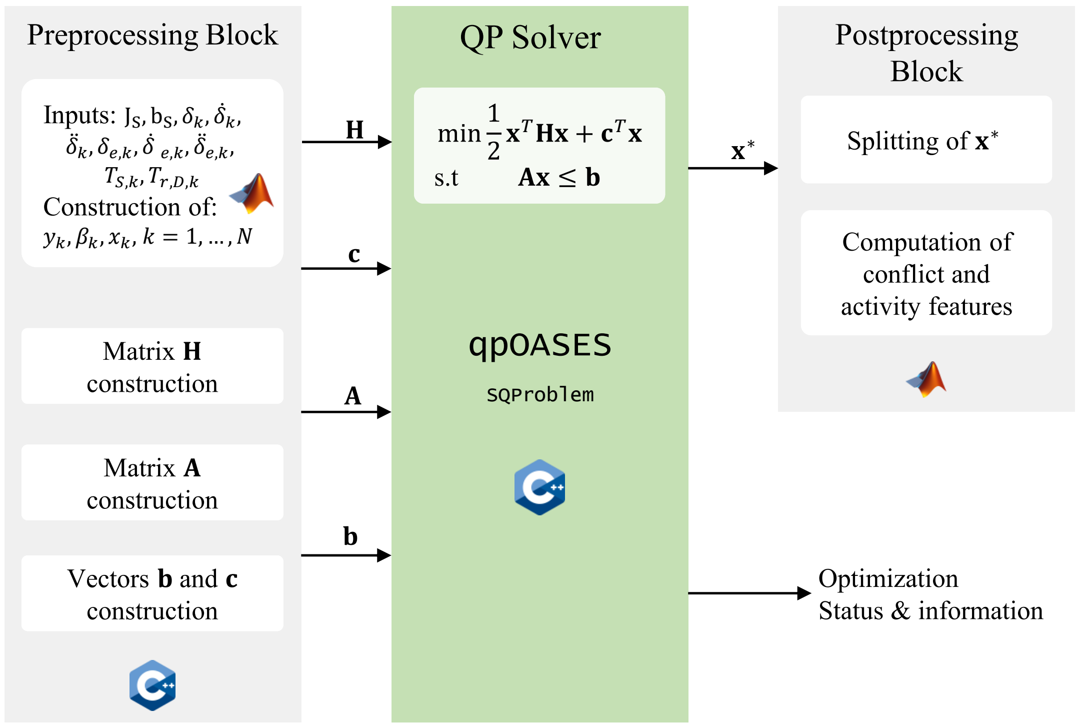

Preprocessing Block
Inputs: $\mathrm{J_S}, \mathrm{b_S}, \delta_k, \dot{\delta}_k, \ddot{\delta}_k, \delta_{e,k}, \dot{\delta}_{e,k}, \ddot{\delta}_{e,k}, T_{S,k}, T_{r,D,k}$
Construction of: $y_k, \beta_k, x_k, k = 1, \ldots, N$
Matrix **H** construction
Matrix **A** construction
Vectors **b** and **c** construction
H
c
A
b
QP Solver
$\min \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x}$
s.t $\mathbf{A}\mathbf{x} \leq \mathbf{b}$
qpOASES
SQProblem
$\mathbf{x}^*$
Postprocessing Block
Splitting of $\mathbf{x}^*$
Computation of conflict and activity features
Optimization Status & information

Figure 6.4: Illustration of the interface design for conflict and activity features computation.

6.4.5 Control and Feedback Modules

The control module is the part of the framework where the actual control variable for the system is computed. It consists of two main part. A preparation part and the MPC part. In the preparation part, the cost function (6.19a), as well as the constraints (6.19b),(6.19d) and (6.19e) are formulated and computed in a MATLAB function, based on the driver class. The MPC part consists of the solver `LIPSOL` introduced in Section 6.3.2. It is loaded and compiled via its Simulink interface. Once, the optimization problem is solved with `LIPSOL`, a steering rate vector $\dot{\delta}_1, \cdots, \dot{\delta}_N$ is provided as output.
In the feedback module, only the first component of the steering rate vector is considered, i.e. $\dot{\delta}_1$. This variable is integrated in time, to compute the corresponding steering angle δ. The integration is made by the built in Simulink Integrator based on a forward Euler integration. The computed steering angle is then routed to the next module.

6.5 Real-Time Feature Computation

In this section, we focus on the real-time computation of the classification features within the framework. The conflict and activity features dominate the computation time of the simulation framework, since they require solving the driver-steering interaction model presented in Section 4.3 in real-time. The dimension of the optimization problem (4.56) plays a crucial role in the required solver timing. Different window sizes for the optimization problem are tested and compared in terms of required computing time. The effects of the choice of the window size on the solver performance are summarized in Table 6.2.

Table 6.2: Effect of increasing the optimization dimensions on the solver performance.

Window Size	Average number of iterations	Average timing in ms	Convergence rate in %
3	3.47	0.02003	100
10	8.81	0.09985	100
20	25.28	0.81347	100
40	86	4.23704	100
50	92.28	5.95723	10.10
70	99.87	8.80751	0.13

Increasing the dimensions of the problem significantly effects the solver performance. The average number of required solver iterations increases gradually with increasing window size. For small window sizes (3 - 10), up to 10 iterations are required. For medium window sizes (20 - 40), this number even triples compared to the smaller dimensions. For larger dimensions the convergence rate decreases drastically. Even for the few times the solver converges, the average number of required iterations approaches the limit of 100. The average timing required by the solver shows a similar behavior. When real-time data is provided every 1 to 5 ms, we observe that window sizes exceeding 40 result in excessively high timing requirements.
We apply a regularization approach on the Hessian matrix to improve the solver's performance. `qpOASES` provides an automatic regularization scheme.The main idea is adding a regularization term controlled by a parameter λ to the Hessian matrix and solving the regularized problem. The results are summarized in Table 6.3.
The number of iterations and the computation time decreased significantly. Hence, the regularization term improved the stabilization of the optimization by penalizing large values in the solution. However, even with the regularization approach, the convergence rate for medium and large dimensions is not satisfactory. This aligns with the design of `qpOASES` which was originally optimized for small- to medium-scale quadratic programming problems. Hence, we consider the small to medium dimensions and illustrate the results of the optimization problem J_D, b_D, k_D and T_D^δ in Figure 6.5.
Three different window sizes are considered. For the lower dimensions ($n = 3$ and $n = 10$), the results are generally similar in shape. The key difference is the presence of oscillations and sharp peaks in the 3-dimensional case. These oscillations could indicate frequent adjustments as the solver attempts to converge and that the problem's structure has higher sensitivity at lower dimensions. In contrast, the results of the 10-dimensional case are notably smoother. With a larger window size, the solver has more historical data to rely on. This

Table 6.3: Effect of increasing the optimization dimensions on the solver performance with regularization parameter $\lambda = 0.01$.

Window Size	Average number of iterations	Average timing in ms	Convergence rate in %
3	1.08	0.00788	100
10	3.96	0.05794	100
20	8.06	0.33077	99.93
40	19.79	2.25685	93.62
50	46.45	4.05277	60.66
70	99.24	7.32291	0.76

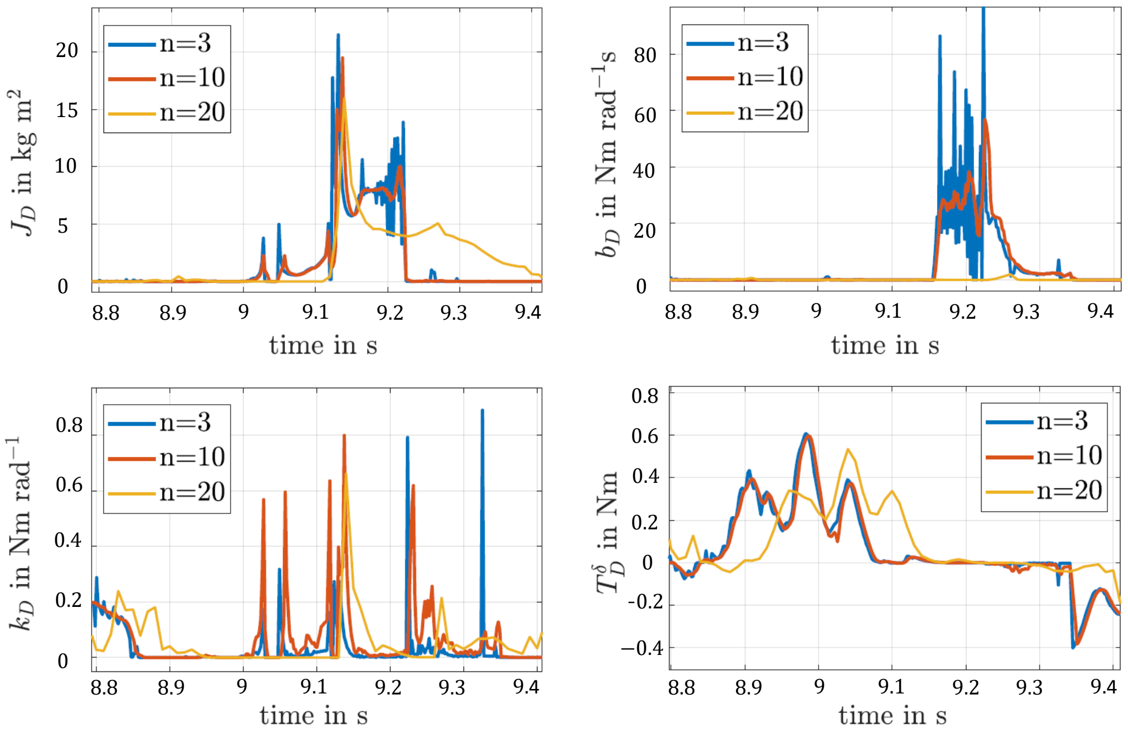


Figure 6.5: Resulted J_D, b_D, k_D and T_D^δ from the optimization problem (4.56) solved by `qpOASES` in real-time for different window sizes.

leads to a more stable solution. The smoother trajectory suggests that the additional time steps help to better capture the dynamics in the data. However, when the window size is increased to 20, there is a significant difference in the results. The general shape of the resulted plots remain similar. However, the timing of key features of the curves, such as peaks, is delayed. This is clearly observable in the bottom right plot illustrating T_D^δ. In the upper right plot of Figure 6.5 illustrating b_D, the solver does not exhibit these features at all. This slower response can be explained by the fact that with a larger window size, the solver gives more weight to a longer history of data, which could lead to a more gradual adjustment

to changes in the time series. As a result, the solver misses short-term dynamics. While the trajectory is smooth, it fails to capture all the finer details of the data.
We compute the corresponding conflict and activity torques, which form the basis for the conflict and activity features. The results are illustrated in Figure 6.6.

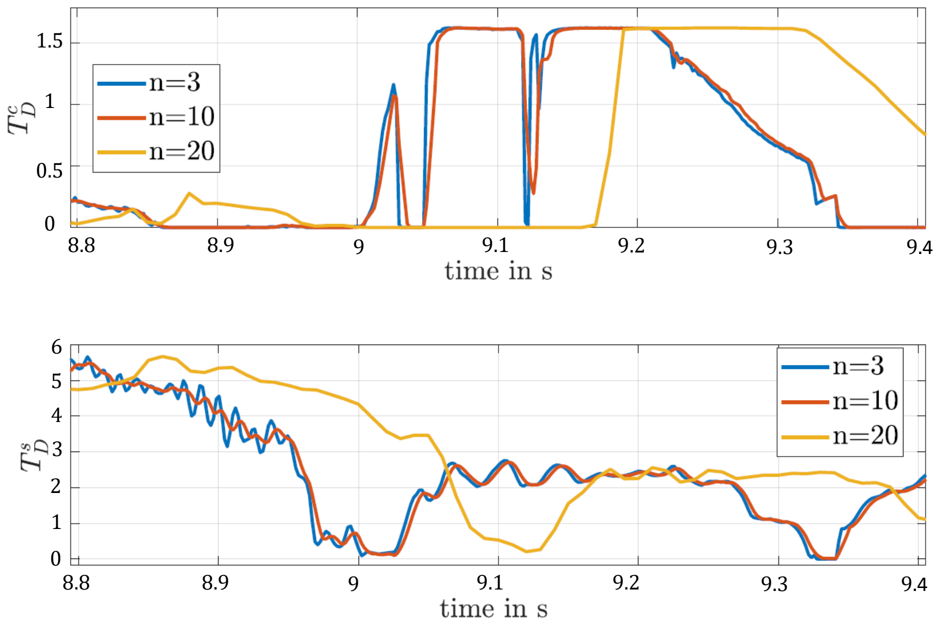


Figure 6.6: Resulted conflict and activity torques for different window sizes.

Obviously, the curves of the conflict and activity torque reflect the same characteristics as the original curves for the different window sizes. The key characteristics for the classification features are the magnitude and the duration of key features in the curves. Hence, to avoid both excessive oscillations and the loss of critical dynamics, the choice of a window size of 10 is optimal for our investigation.
As introduced in Section 6.4.4, the remaining classification features do not significantly effect the computation time of the classification module compared to the conflict and activity features. The consistency and adaptation features (Table 5.4 and Table 5.6) are based on the lateral deviation Δd and the heading angle error $\Delta\psi$. Both variables are vehicle states (see Section 4.1.2). They are provided in real-time from the input data module. Analogously, the individual path pattern features, resulted from Algorithm 8, are based on x- and y-coordinates that are also inputs provided in real-time. We compute the features for a 400-meter track section. The required computation time is summarized in Table 6.4.
To visualize the results of the consistency features, we apply the framework on the data of two different drivers. The drivers drive over the same track section for three times. Figure 6.7 illustrates the standard deviations of the lateral deviation σ_d (upper plot) and of the heading angle error $\sigma_{\Delta\psi}$ (bottom plot).
We select one driver to visualize his adaptation and individual path pattern features.

Table 6.4: Computation time of consistency, individual path pattern targeting and adaptation features over a 400-meter track section.

Classification feature	Computation time per step in ms
Consistency	0.003118
IPPT	0.000224
Adaptation	0.003085

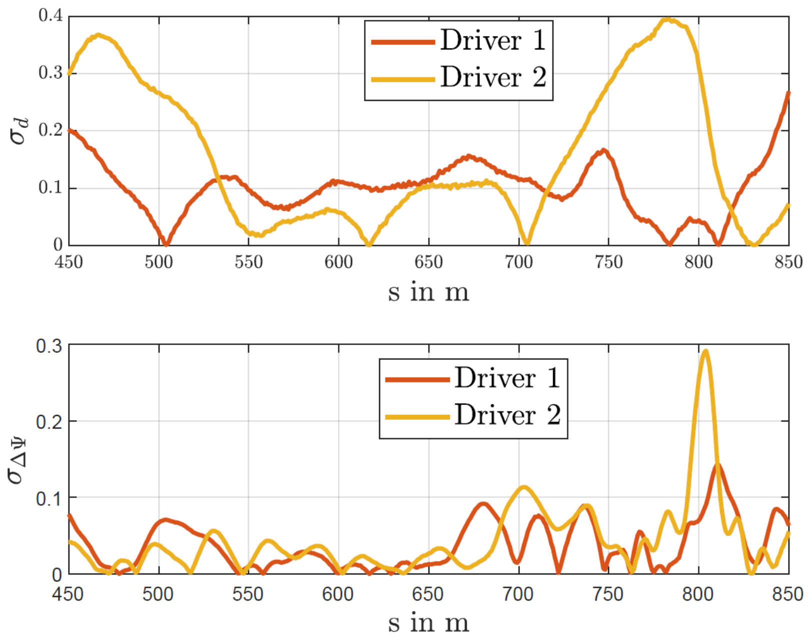


Figure 6.7: Real-time computed standard deviations of the lateral deviation σ_d (upper plot) and of the heading angle error $\sigma_{\Delta\psi}$ (bottom plot) for two different drivers on the same track section.

Figure 6.8 illustrates the adaptation features. The upper plot visualizes the computed quality function g_i introduced in (5.38) for two paths. The bottom plot visualizes the driver adaptation value (5.39) between the two paths. Figure 6.9 presents the individual path pattern targeting features: the Following and Increasing Tendency values for each path pattern.
All the three analyzed features require a comparison with historical data. Hence, each time a driver crosses a specific track section the key metrics (such as lateral deviation, heading angle error, path quality, (x, y)-coordinates...) are captured and stored in a memory structure. These metrics are then fed back into the simulation as reference data the next time the driver crosses the same track section. This feature is implemented using a memory block in Simulink. It presents a dynamic storage unit that holds relevant data from previous simulation iterations. The chosen size of the stored data in the memory block affects the computation time. Figure 6.10 illustrates this effect for the consistency function.

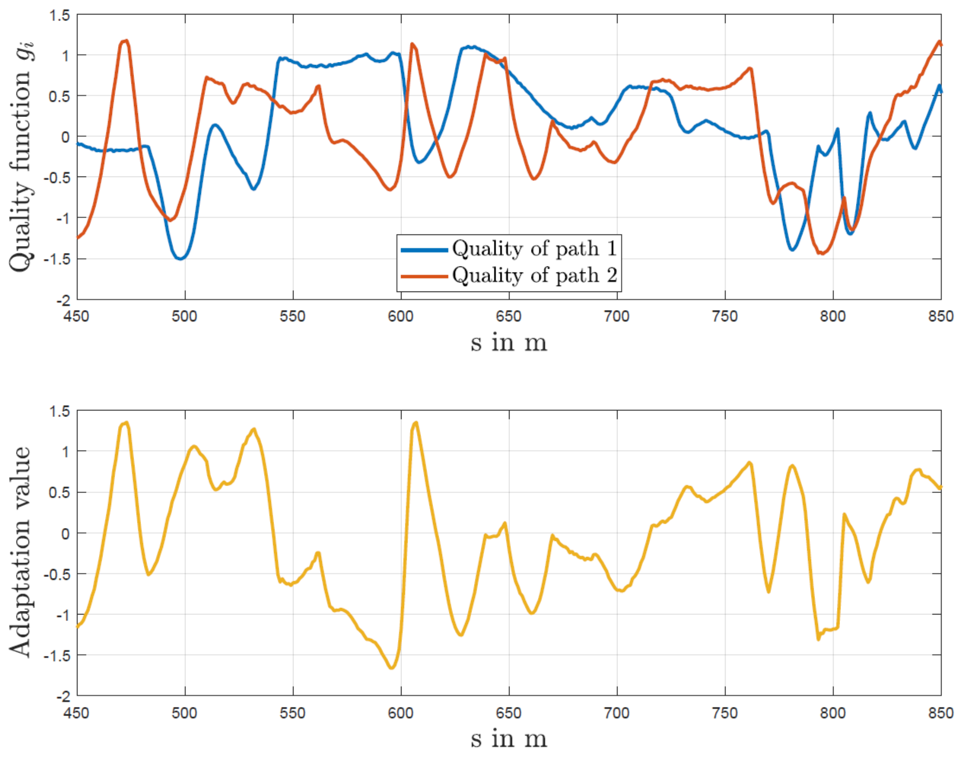


Figure 6.8: Real-time computed quality function g_i for two paths (upper plot) and corresponding driver adaptation value G (bottom plot).

The considered stored data for this feature corresponds to the lateral deviation and the heading angle error. The computation time increases linearly with increased size of stored data but remains within the same range (10^{-5}s), even with 2400 stored lap meters, which correspond to a full lap. For a sample time of $\Delta t = 0.01$s and an average speed of 19 ms^{-1}, this corresponds to approximately 17000 stored data point for each variable. However, to adapt the system behavior, we focus on driver behavior on particular road segments, such as curves or critical track sections. Hence, only segments of 50 up to 500 meters need to be stored. This ensures that the system is able to process large amounts of data for the different features without significant delays.

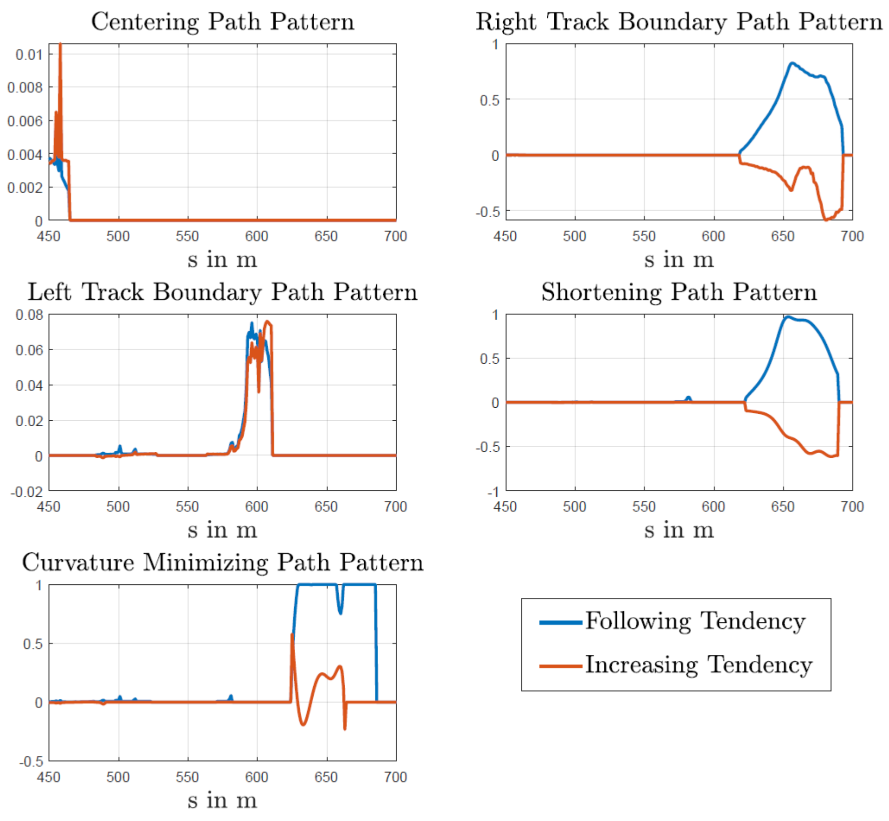


Figure 6.9: Real-time computed individual path pattern targeting features: Following and Increasing Tendency values.

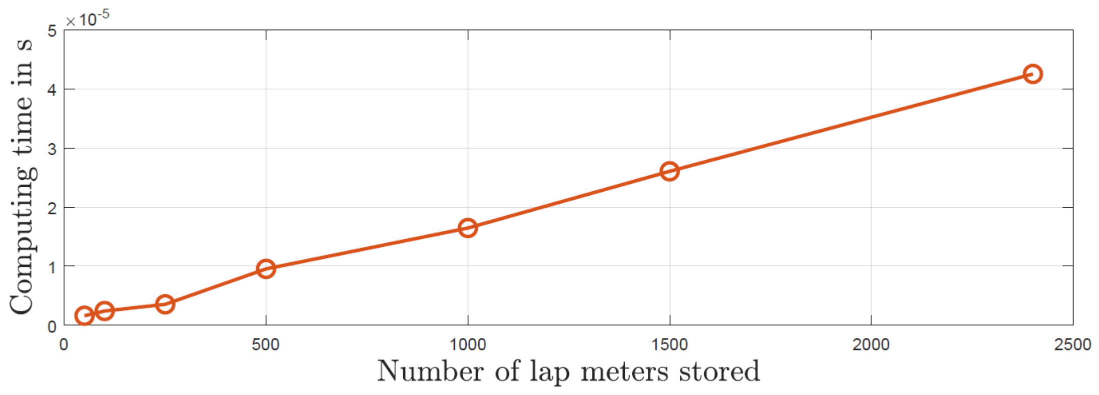


Figure 6.10: Effect of the stored data size on the computation time of the consistency feature function.

6.6 Trajectory Planning

6.6.1 *Performance Analysis*

To test the performance of the implemented MPC, we conduct experiments on the same test track presented in Section 5.2.1 with 2400 lap meters. To compute the maximal distances $d_{\max}^{f,l}$ and $d_{\max}^{f,r}$ between the reference line and the left and right bounds, respectively, we perform an orthogonal projection onto these boundaries, by solving an optimization problem that searches for the nearest point on the left and right boundaries in the direction of travel at each step. This is illustrated in Figure 6.11 for a section of the track.

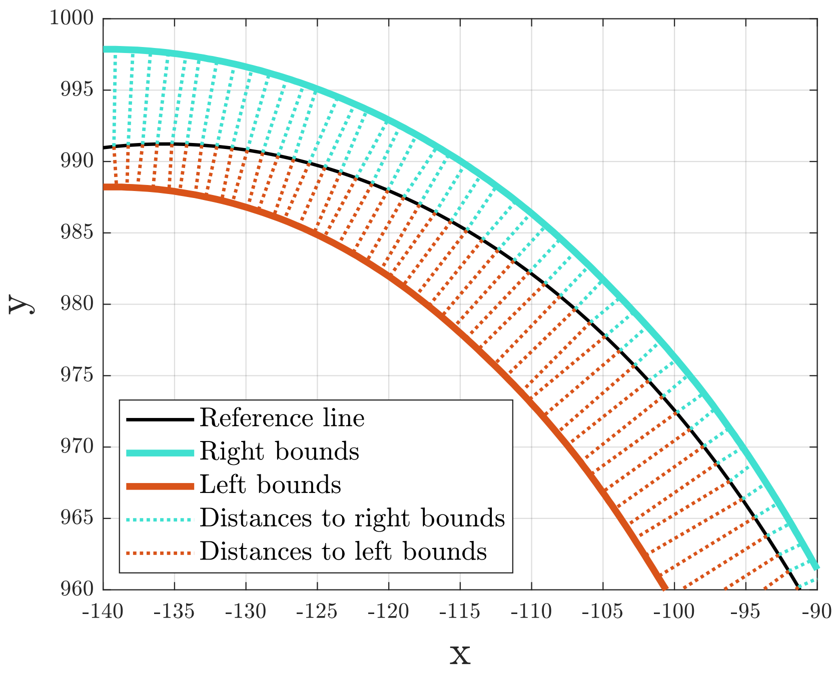


Figure 6.11: Track section with distances from reference line to left and right bounds.

The vehicle parameters used in the simulation model correspond to real vehicle data of a full-size car and are listed in Table 6.5. The vehicle speed is fixed to 15 ms^{-1}. Hence, a simulation time of 160 s corresponds to a full lap.
Let T be the total simulation time and Δt the discretization step. Then, the number of simulated time steps M is given by

$$M = \frac{T}{\Delta t}. \tag{6.21}$$

To compute the model accuracy, we consider the average lateral deviation

$$\overline{\Delta d} = \frac{1}{M} \sum_{k=1}^{M} |\Delta d_k^*|, \tag{6.22}$$

Table 6.5: Vehicle Model Parameters

Parameter	Value	Unit
m	2421	kg
l_f	1.60	m
l_r	1.53	m
c_f	136.62	kN rad^{-1}
c_r	195.29	kN rad^{-1}
J_z	3433.3	kg m^2
v_x	15	ms^{-1}

and the average heading angle error

$$\overline{\Delta\psi} = \frac{1}{M}\sum_{k=1}^{M} |\Delta\psi_k^*|. \tag{6.23}$$

Additionally, the performance of the MPC is evaluated based on the required solver time and the number of iterations needed. This information is provided by the solver after each step. To analyze the computation time for each component of the framework, we use the Simulink Profiler. This built-in application in Simulink allows to record the execution time not only for the entire model but also for individual blocks, functions, or subsystems.

We initially consider the case of fully automated driving, wherein the input and classification modules are excluded. In a first test, we analyze the effect of the discretization step Δt on the computation time and on the MPC performance with a fixed prediction horizon N. The considered time discretization steps are $\Delta t = 0.001$s, $\Delta t = 0.005$s and $\Delta t = 0.01$s. These are commonly used discretization values in vehicle control applications. Specifically, $\Delta t = 0.001$s is typically employed for scenarios involving fast vehicle dynamics or sharp maneuvers. On the other hand, $\Delta t = 0.01$s is generally applied in scenarios with smoother dynamics, with more gradually changing vehicle state. The results of this test are summarized in Table 6.6.

We observe that the performance and accuracy of the solver are slightly affected by the different Δt values. The required solver time is significantly reduced for $\Delta t = 0.005$s, with approximately one-eighth of the time required for the other discretization steps. The average number of solver iterations, as well as the average lateral deviation and heading angle error have comparable results for all three values. Additionally, the discretization steps slightly influence the required computation time for the feedback and vehicle dynamics modules, with the lowest computation time achieved for $\Delta t = 0.005$s. Hence, in the following analysis, we set the discretization step to $\Delta t = 0.005$s.

To improve the convergence of the solver, the choice of the initial barrier parameter γ_0 is crucial. We analyze the effect of different initial values on the solver performance. The results are summarized in Table 6.7.

The considered initial barrier parameters range from 1 to 10^8. As the barrier parameter

Table 6.6: Effect of different discretization steps Δt on the MPC performance for $N = 30$ over a full lap.

Δt	0.001	0.005	0.01
Average solver iterations	8.79	8.79	8.79
Maximum solver iterations	16	15	15
Minimum solver iterations	8	8	8
Average solver time per step in ms	8.6912e−02	1.0738e−02	9.1643e−02
Average lateral deviation $\overline{\Delta d}$	2.719e−02	2.718e−02	2.716e−03
Average heading angle error $\overline{\Delta \psi}$	7.261e−03	7.264e−03	7.268e−03
Control module computation time per step in ms	5.98e−01	5.18e−01	6.16e−01
Feedback module computation time per step in ms	5.16e−04	2.97e−04	4.06e−04
Vehicle dynamics module computation time per step in ms	4.37e−03	3.25e−03	5.06e−03

Table 6.7: Effect of the choice of the initial barrier parameter γ_0 on the solver performance for $N = 30$ and $\Delta t = 0.005$ over a full lap.

Barrier parameter γ_0	Average solver iterations	Maximum solver iterations	Minimum solver iterations	Average solver time per step in ms	Average duality gap value
10^0	8.52	13	6	0.079357	2.3951e−08
10^2	8.34	15	6	0.076368	9.5877e−08
10^4	8.78	15	8	0.079344	5.0128e−08
10^6	12.56	18	11	0.113650	5.5598e−08
10^8	15.17	22	13	0.139110	8.3081e−08

increases, there is a progressive increase in the solver's computation time. On the other hand, lower values of the barrier parameter tend to result in fewer iterations. This aligns with the expected behavior of interior-point methods. A larger initial value causes the solver to take larger steps early in the optimization process. However, more iterations are required to refine the solution. In contrast, smaller barrier parameter values lead to faster steps towards the optimal solution. This reduces the required number of iterations, but can compromise stability and accuracy. However, in this test, for all chosen initial values, the solver converges successfully and results in a comparable average duality gap value. Hence, to achieve a balance between convergence speed and solution quality, we set the initial barrier parameter to $\gamma_0 = 10^4$.

To account for different driver behavior, we analyze the effect of some of the adjustable parameters introduced in Table 6.1 on the solver performance. We start with varying the prediction horizon N. It describes the number of time steps, the MPC solver plans ahead. The duration of the prediction horizon is called the look-ahead time and is given by

$$T_N = N \cdot \Delta t. \tag{6.24}$$

We consider look-ahead times ranging from 25 ms up to 2.5 s and set the maximal number of solver iterations to 50. The results are summarized in Table 6.8.

Table 6.8: Effect of different prediction horizon lengths N on the solver performance for $\Delta t = 0.005$s and $\gamma_0 = 10^4$ over a full lap.

N	Look-ahead time T_N in s	Average solver iterations	Maximum solver iterations	Minimum solver iterations	Solver convergence rate in %	Average solver time per step in ms
5	0.025	47.96	50	8	5.36	0.096492
15	0.075	9.05	16	7	100	0.046866
30	0.15	8.78	15	8	100	0.079344
100	0.5	8.99	50	8	99.91	0.311236
300	1.5	9.71	50	8	99.29	0.999332
500	2.5	9.85	50	8	99.83	1.555695

For a short look-ahead time of 25 ms, the average solver iterations is 47.96. The solver is fast but converges only in 5.36% of the time. For medium look-ahead times between 75 ms and 150 ms, the solver performs well with an average of 9 iterations, and converges 100% of the time. For longer look-ahead times from 0.5 s up to 2.5 s, the solver becomes slower but still achieves over a 99% convergence rate. Figure 6.12 visualizes the solver's performance across different look-ahead times.
We see that a minimal prediction horizon (< 50ms) is insufficient for the solver to make accurate decisions. Analogously, longer look-ahead times (> 1.5 s) increase the computation time significantly (more than five times longer than medium look-ahead times). In contrast, the medium look-ahead times (75 – 500 ms) show a significant improvement in performance. Hence, this range appears to be the most efficient choice.
When there is no driver intervention, the MPC should perform in a similar way, regardless of whether the left and right bounds are tightened or relaxed. To demonstrate this behavior, we adjust the deviation scaling factor w and evaluate the MPC performance. The results are summarized in Table 6.9.
As expected, the solver converges in 100% of the time, regardless of how wide or tight the lateral deviation bounds are. However, a more challenging situation arises when the solver starts from an initial condition with a lateral deviation from the reference trajectory. We define the *tolerance recovery time*, denoted T_τ, the required time for the system to reduce the lateral deviation within the specified tolerance range $\pm\tau$ and maintain it within that range

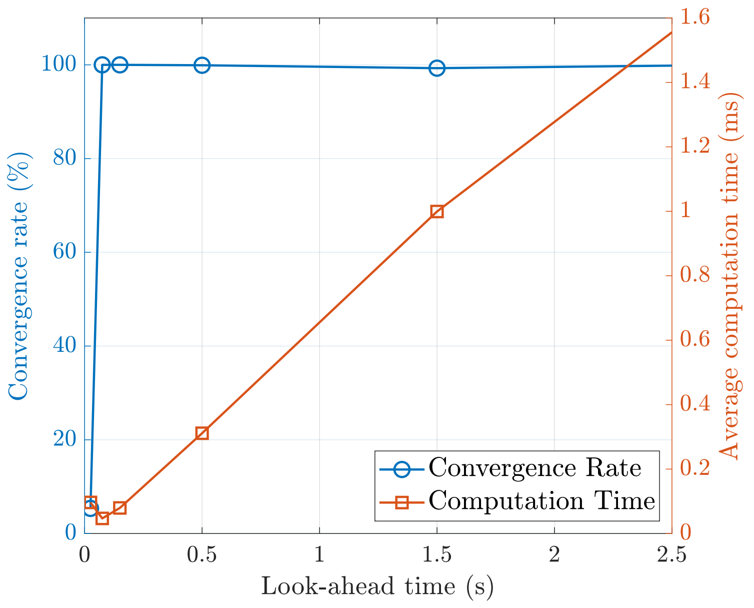


Figure 6.12: Solver's performance across different look-ahead times.

Table 6.9: Effect of different lateral deviation scaling factors w on the solver performance for $\Delta t = 0.005$s and $\gamma_0 = 10^4$ over a full lap.

w	Average solver iterations	Maximum solver iterations	Minimum solver iterations	Solver convergence rate in %	Average solver time per step in ms
1	8.78	15	8	100	0.079344
0.8	8.74	15	7	100	0.086500
0.6	8.72	15	7	100	0.080321
0.4	8.81	15	7	100	0.088219
0.2	8.88	15	7	100	0.089073
0.1	8.93	14	7	100	0.088719

consistently. We simulate the system for different initial lateral deviations Δd_0 and adjust the weight α in the cost function. Then, we compute the tolerance recovery time T_τ of the system for $\tau = 25$cm. The results are summarized in Table 6.10.
For small lateral deviations (< 75cm), the offset is reduced within maximal half a second for different values of α. For larger deviations (> 1m), the choice of the value of α has a significant effect on the tolerance recovery time. The higher the value of α, the shorter is the tolerance recovery time, i.e., the MPC prioritizes the minimizing of this deviation in the cost function over the other objectives. However, when a shorter tolerance recovery time is desired for a specific driving behavior, it is important that the rapid corrections of

Table 6.10: Effect of the lateral deviation weighting parameter α on the tolerance recovery time T_τ for different initial deviation values Δd_0 for $\tau = 25$cm.

	Tolerance recovery time T_τ in s			
Δd_0 in m	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$
0.5	0.48	0.25	0.21	0.21
0.75	0.53	0.27	0.26	0.26
1	1.34	0.58	0.44	0.44
1.5	2.20	0.91	0.65	0.64
2	2.66	1.01	0.88	0.83

the controller do not lead to overshooting or oscillations in the system. Hence, a desired tolerance recovery time should be coupled with penalizing aggressive control actions. We consider the case of an initial lateral deviation of 1.5 m and analyze the effect of penalizing the steering angle by adjusting the parameter ζ. The results are illustrated in Figure 6.13 All four simulations start at the initial state $\mathbf{y}_0 = [0.2, 1.5, 0, 0, -0.3]$. The upper plot in Figure 6.13 demonstrates the lateral deviation Δd over time. The bottom plot demonstrates the corresponding steering angle δ. For all four cases, the beginning of the curves indicate that the system prioritizes correcting the heading angle error early in the trajectory. For $\zeta = 10^{-6}$ and $\zeta = 10^{-5}$ the system achieves the tolerance range in $T_\tau \approx 0.5$s. However, the reduction of the lateral deviation is accompanied by an aggressive steering behavior. We see that the amplitude of the oscillation decreases between the first and second curve. For $\zeta = 10^{-4}$ this oscillatory behavior in the steering angle is not observable. The tolerance recovery time is longer ($T_\tau \approx 1$s) but the steering response remains stable. Increasing ζ to 10^{-3}, does not improve the performance. The system requires more time to reduce the heading angle error. Additionally, the sharp change in the steering angle indicates that the system is no longer operating smoothly and that the steering angle is possibly over-penalized. An other approach for the MPC to handle these large lateral deviations in a robuster way, is the introduction of local reference trajectories. The main idea, is to plan a new short reference path from the ego-position to the tolerance range. This new path is handed to the MPC as a new reference. Hence, the initial lateral deviation is perceived by the controller as effectively zero and the MPC focus on minimizing the deviations along this adjusted reference. This ensures a gradual return to the global reference trajectory without introducing oscillatory or abrupt behaviors. To account for different driver behaviors, the new planned reference should be always re-planned as long as the lateral deviations exceed the tolerance range. Hence, a computationally efficient approach is required. A simple and cost-effective solution is the generation of polynomial functions between the current position and the tolerance range in the curve-relative coordinate system, i.e., in terms of lateral deviation Δd and curvature κ relative to the reference. In [99], polynomial functions of order 7 have proven to be suitable

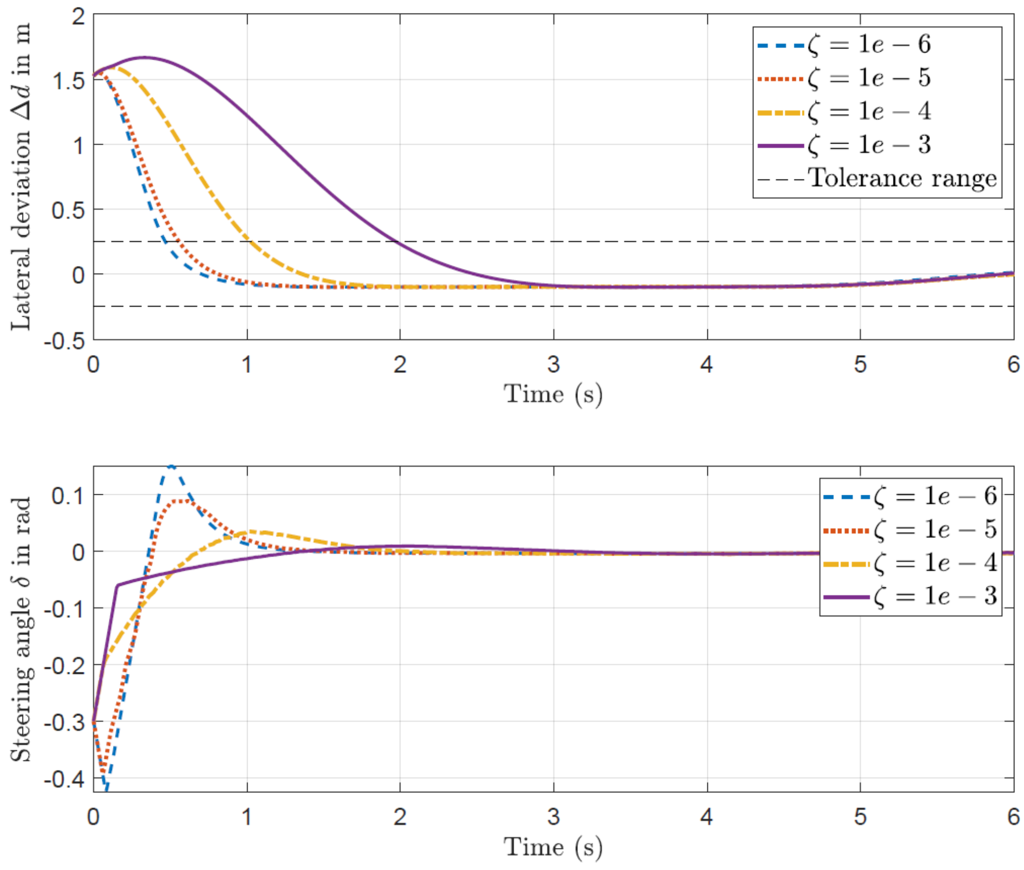


Figure 6.13: Effect of penalizing the steering angle on the lateral deviation for an initial lateral deviation Δd_0 = 1.5m.

curve structure for this application. Hence, let the new reference trajectory have the lateral deviation given by the 7th-order polynomial function

$$d(s) = a_7 s^7 + a_6 s^6 + a_5 s^5 + a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s^1 + a_0. \tag{6.25}$$

Let Δd_0 be the initial lateral deviation and $\Delta d_0', \Delta d_0'', \Delta d_0'''$ its first, second and third derivatives at an initial state s_0. And let Δd_fbe the desired final lateral deviation and $\Delta d_f', \Delta d_f'', \Delta d_f'''$its first, second and third derivatives at a final state s_f. Then the coefficients $a_0, ..., a_7$ can be simply computed by solving a linear system, or by defining a least squares minimization problem. Once, the polynomial function (6.25) is computed, the curvature is given by

$$\kappa(s) = \frac{d''(s)}{(1 + d'(s)^2)^{3/2}}. \tag{6.26}$$

We consider the example of an initial lateral deviation $\Delta d_0 = 1.5$m at $s_0 = 0$ and we aim to achieve the desired final lateral deviation $\Delta d_f = 0$ at $s_f = 50$. We consider three initial heading angle errors $\Delta\psi_0$. The computed polynomial function of the lateral deviation, as

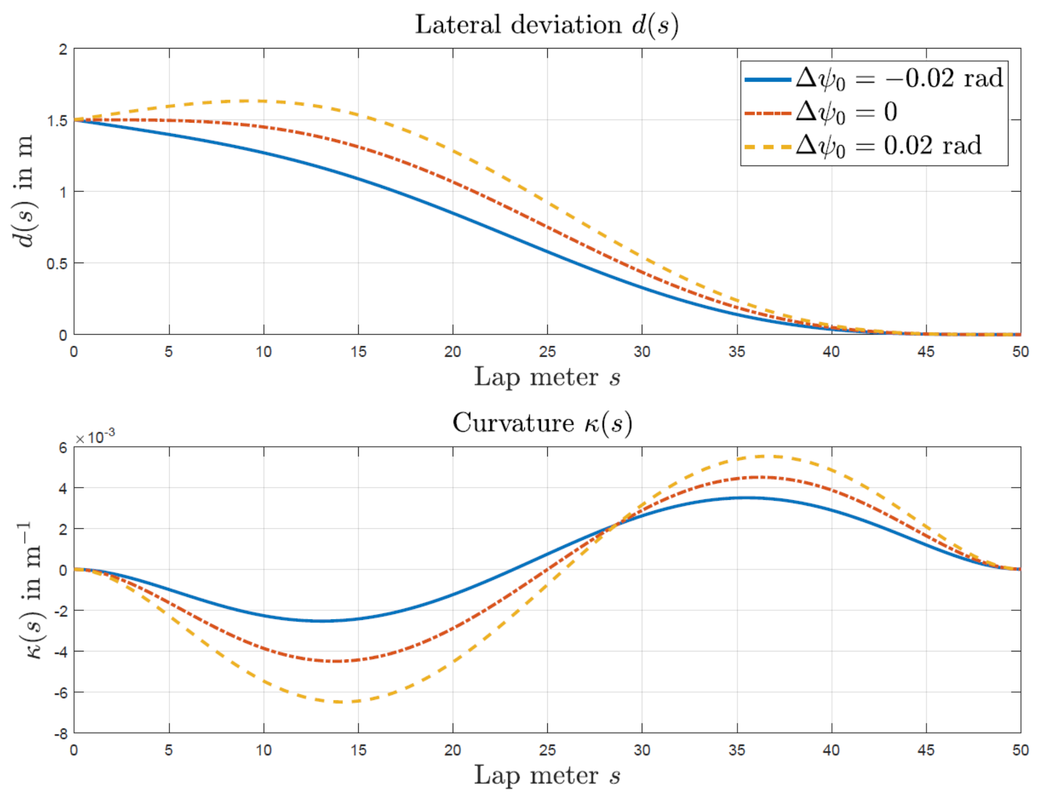


Figure 6.14: Computed reference path for the lateral deviation (upper plot) and its corresponding curvature (bottom plot) for three different initial heading angle errors.

well as its corresponding curvature are illustrated in Figure 6.14.
To reduce oscillations and aggressive controller behavior, the reference path at the initial lateral deviation Δd_0 is corrected by the computed polynomial function. Figure 6.15 shows the effect of correcting the reference trajectory on the resulted lateral deviation and the steering angle.
By correcting the reference, the MPC perceives the initial lateral deviation of 1.5 m as effectively zero and is able to handle the steering action in a smooth manner. The convergence rate of the solver is 100% in both cases. The required number of iterations for each case are shown in Figure 6.16
The solver requires significantly more iterations to handle the large lateral deviation with a maximum of 19 iterations. The corrected path reduces the number of iterations to an average of 13. This optimizes the computational efficiency of the solver.

6.6.2 Use Case Application

In the previous section, we analyzed the effect of various factors on the performance of the implemented control module. In this section, we apply suitable configurations to real driving data. Based on the desired system behavior discussed in Section 6.2, we apply different configurations on the MPC. For this, we consider three drivers classified based on their interaction strategy: nonintervention, uncertainty and adaptation. The applied configurations

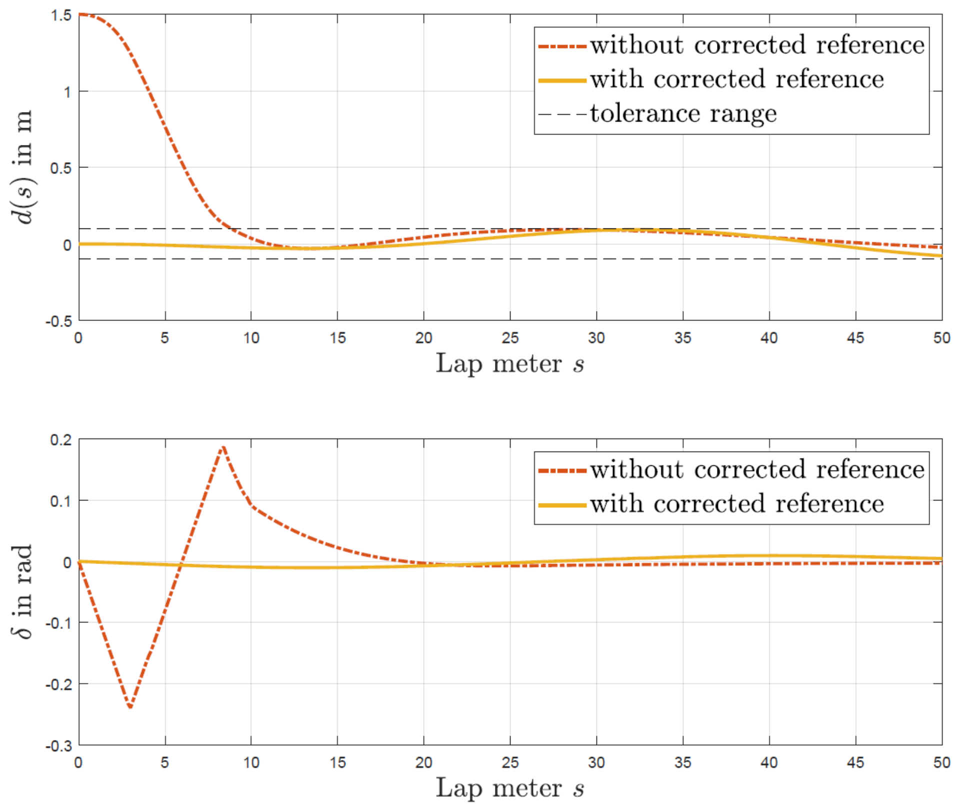


Figure 6.15: Effect of corrected reference trajectory on the MPC results: lateral deviation (upper plot) and steering angle (bottom plot).

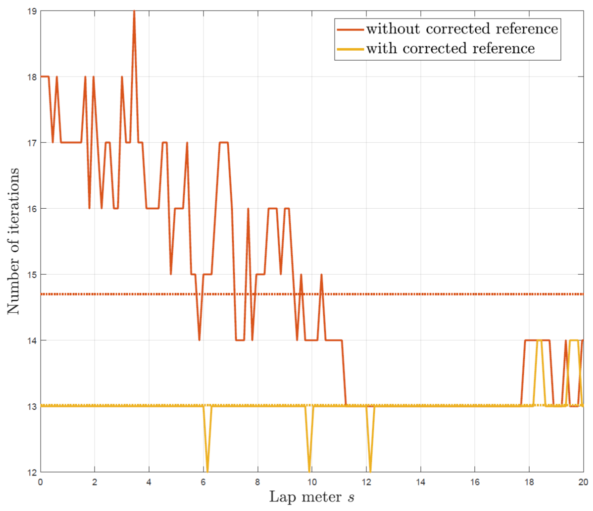


Figure 6.16: Effect of corrected reference trajectory on the number of iterations required by the solver.

for each driver interaction strategy are listed in Table 6.11.
We consider two scenarios: a straight path and a curved path. To visualize the adjusted

Table 6.11: Configuration of the MPC for different interaction strategies.

	Interaction strategy		
Configuration	Nonintervention	Uncertainty	Persistence
α	1e-03	1e-05	1e-04
β	1e-01	1e-03	1e-05
$\mathbf{F}_{\dot{\delta}}$	1e-05	15	1
$\mathbf{F}_{\mathbf{s}_{1,1}}$	1e-09	1e-11	1e-08
$\mathbf{F}_{\mathbf{s}_{2,2}}$	1e-09	1e-11	1e-08
w	0.1	0.8	1
$\dot{\delta}_{\text{min}}$	-1.75 rad	-1.57 rad	-0.61 rad
$\dot{\delta}_{\text{max}}$	+1.75 rad	+1.57 rad	+0.61 rad
N	30	15	30

reference on the track, we transform the simulation results from the curve-relative coordinates to the global (x,y)-coordinates using the transformation map

$$\begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} x_{\text{ref}}(s) \\ y_{\text{ref}}(s) \end{bmatrix} + d(s) \cdot \begin{bmatrix} -\sin\Delta\psi(s) \\ \cos\Delta\psi(s) \end{bmatrix}, \tag{6.27}$$

where $(x_{\text{ref}}, y_{\text{ref}})$ correspond to the global coordinate of the reference line.
We start with the nonintervention interaction strategy. Figure 6.17 illustrates the results for the straight path. The upper left plot shows the lateral deviation of the vehicle. It indicates that the MPC follows the desired path with a high precision. The upper right plot shows the heading angle error. Only small deviations are observed. Hence, the MPC is also effective in maintaining the vehicle's orientation. The bottom plot illustrates the transformed path into global coordinates. No significant deviations between the reference and the MPC path are observed.

Figure 6.18 illustrates the results for a curved path. Only small errors in the lateral deviation and the heading angle error are observable. When plotted in the global coordinate system, the vehicle's path closely follows the desired path with no significant deviations.

To simulate more realistic conditions, we must consider the possible perturbation that could occur when the driver holds the steering wheel, even when he is not actively steering. Hence, we introduce small random perturbations to the steering input. This is modeled using a combination of Gaussian noise and low-frequency oscillations. The Gaussian noise with a small variance, represents the random movements of the driver's hands. The low-frequency oscillations represent their natural movements. These oscillations are modeled using a sine wave with a low amplitude and frequency. The effect on the MPC results is illustrated in Figure 6.19.

The added perturbations in the steering input slightly degrade the MPC performance. Small increases in lateral deviation and heading angle error are observable. However, the lateral

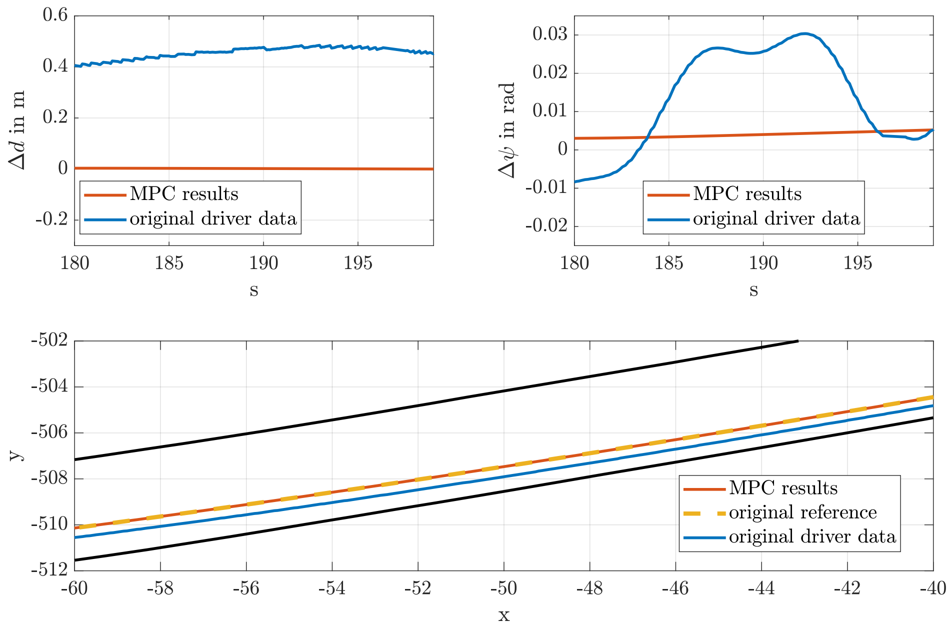


Figure 6.17: MPC results for nonintervention interaction strategy over a straight path.

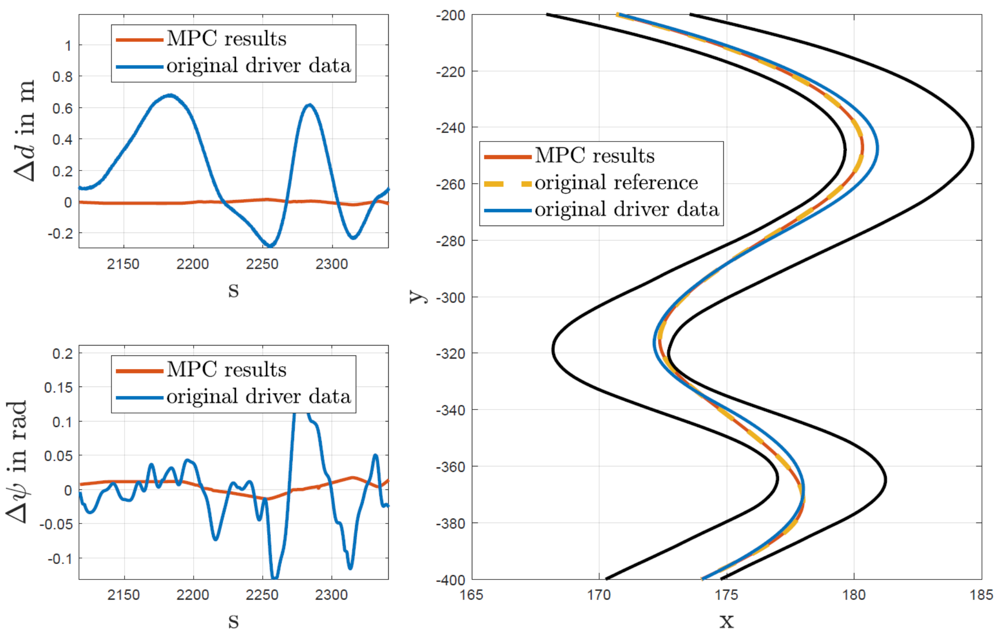


Figure 6.18: MPC results for nonintervention interaction strategy over a curved path.

deviations do not exceed 20 cm and the heading angle error ranges between -0.002 and 0.02 rad. Additionally, the solver converges over the full track with an average of only 12.57 iterations. Hence, the deviations remain controlled. This confirms the ability of the MPC to handle the perturbations in the steering wheel effectively for the nonintervention interaction

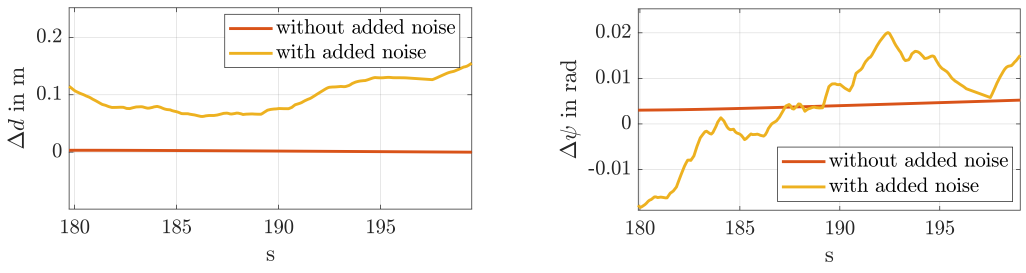


Figure 6.19: Effect of added noise to the MPC results.

strategy.
Now, we apply the MPC configuration to the uncertainty interaction strategy. The results for a straight and a curved path are illustrated in Figure 6.20 and Figure 6.21, respectively. The bounds for the lateral deviation are relaxed with scaling factor $w = 0.8$. The system achieves medium lateral deviations and heading angle errors.
The resulted steering angle with and without added noise, and the original driver steering

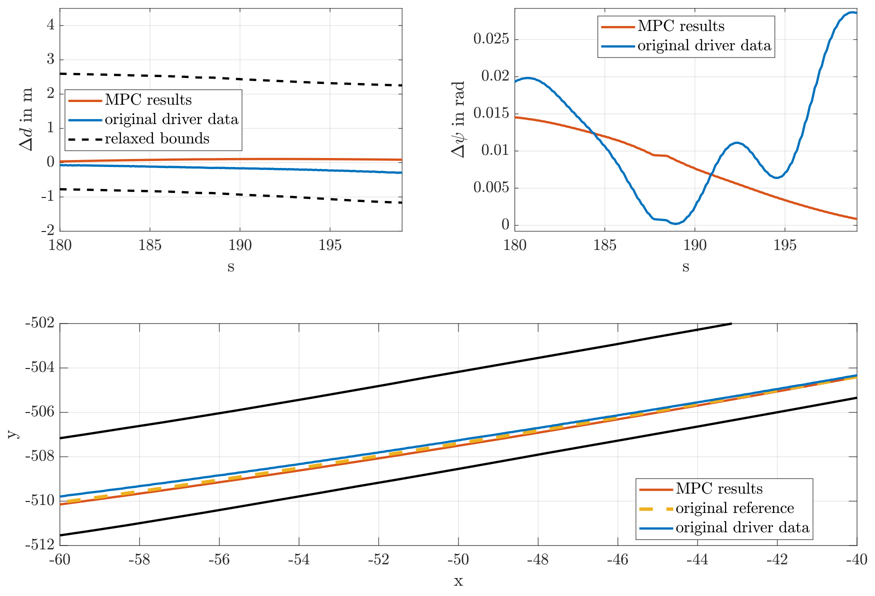


Figure 6.20: MPC results for uncertainty interaction strategy over a straight path.

angle are illustrated in Figure 6.22. The MPC tends to apply larger steering angle inputs, since the focus of this configuration is to achieve a smooth system behavior without abrupt corrective actions, rather than achieving accurate path tracking. The oscillations the steering input of the driver exhibits can be smoothed with the use of steer-by-wire. Blending the input with the MPC results could enable better integration between both actions.
Finally, we consider the persistence interaction strategy. The MPC configuration for this

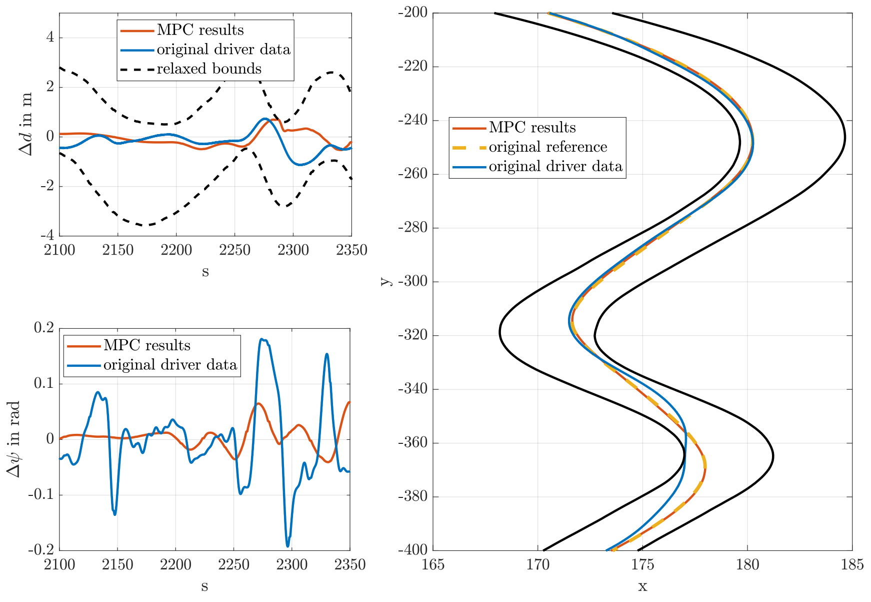


Figure 6.21: MPC results for uncertainty interaction strategy over a curved path.

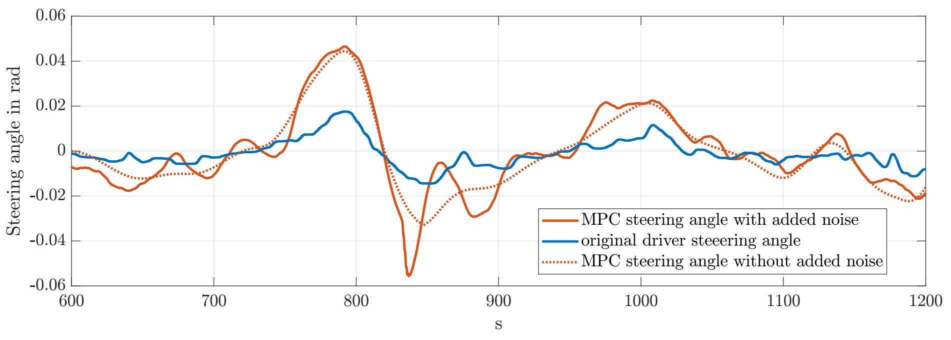


Figure 6.22: Comparison between MPC steering angle (with and without added noise) and original driver steering angle.

category leads to the results in Figure 6.23 and 6.24.

The bounds for the lateral deviation are relaxed to the maximum, i.e., the full width of the track can be used by the system. The input of the driver and the MPC steering angle are blended with scaling factors $\omega_1 = 0.3$ (driver) and $\omega_2 = 0.7$ (system). Over the straight line, the resulted path aligns more closely to the driver's behavior. However, in the curved path, the blending combination does not result in a significant improvement. Higher scaling factors for the driver led to the divergence of the solver.

The necessary configuration for the remaining interaction strategies (selective persistence and adaptation) can be seen as a combination of the use cases we demonstrated. The presented

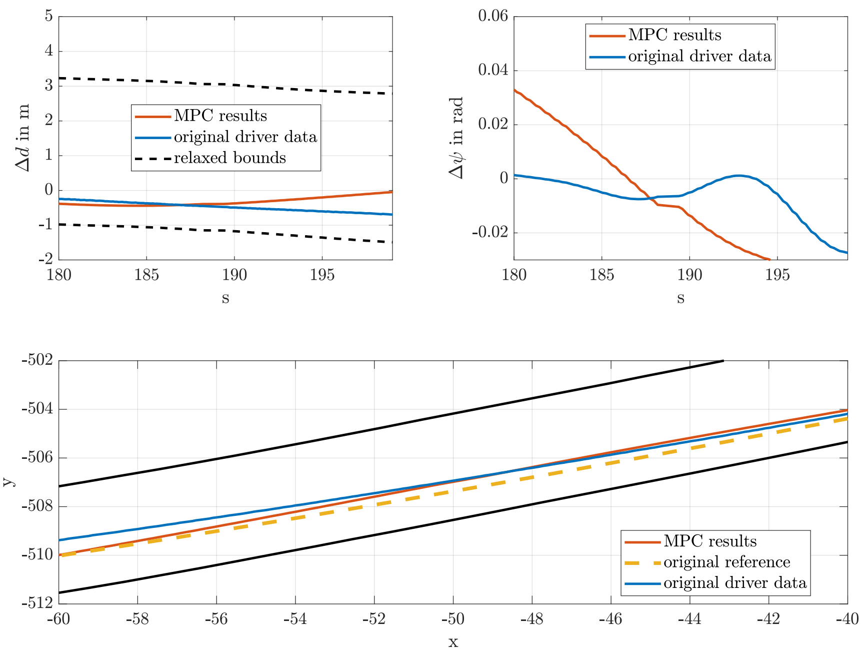


Figure 6.23: MPC results for persistence interaction strategy over a straight path.

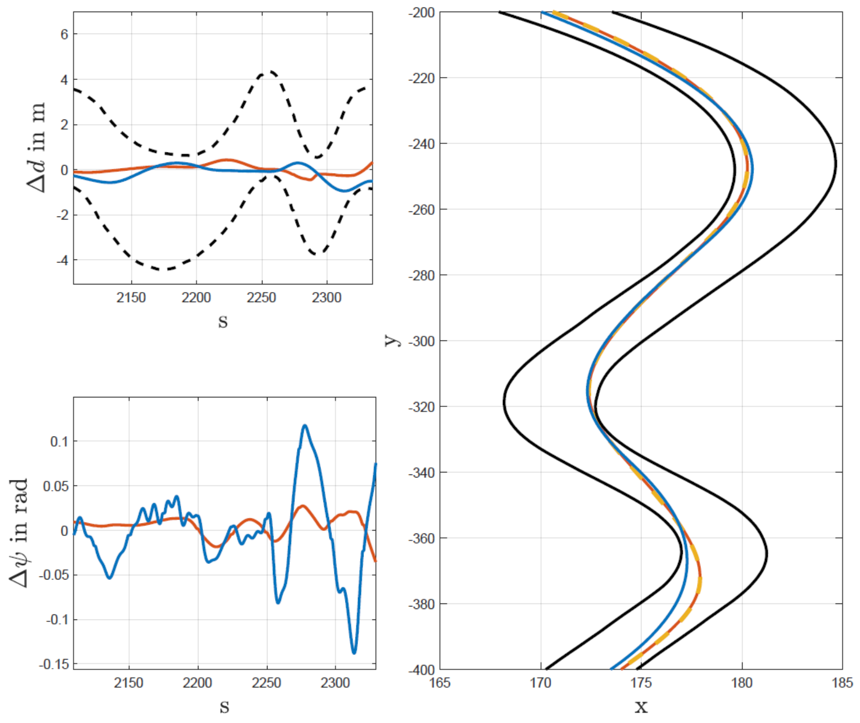


Figure 6.24: MPC results for persistence interaction strategy over a curved path.

results highlighted the flexibility of the implemented MPC and its ability to account for the different driving behavior.

6.7 Real-World Applicability

The implemented framework within Simulink is applicable for a real-world system. From a hardware and software perspective, all components are written either in MATLAB or in C++ code. Simulink allows the generation of C-code for these components, which can be directly deployed to a dSpace MicroAutoBox[4]. This allows the integration of the methods with a test vehicle for real-time testing. From a modeling perspective, the main components of the framework were built and tested based on real-driving data, i.e., main complexities of real-world driving conditions have already been considered. Additionally, the vehicle dynamics were modeled based on an accurate single track model. Despite the simplified representation in those models, they effectively capture the essential vehicle behavior. In fact, many series-production vehicle dynamics models are based on similar single track models. The steering-interaction model presented in Section 4.3, which forms one of the key components in the framework and a main finding in this thesis, has been individually tested and successfully validated in a real-world scenario. A validation of the full framework in a real-world application is beyond the scope of this thesis.

6.8 Limitations of the Framework

In this thesis, we designed and analyzed the potentials of the proposed shared control framework. However, several limitations must be noted. We proposed a new classification for the interaction in shared control driving. Especially the selective persistence category has not been considered in other studies to the best knowledge of the author. However, we noticed that the classifier misclassified drivers especially in this category. This suggests a further refinement of the classification features and a clear distinction from the adaptation and persistence category. Additionally, the uncertainty class was not analyzed in detail. In our dataset, we could not identify clear patterns to describe this category. However, addressing this category is crucial for increasing the acceptance of lateral assistance systems, particularly for drivers who feel overwhelmed by system interventions. To achieve this, more extensive data collection and analysis are necessary. This could enable the development of more robust and reliable classification features. In addition, all the classification features that require historical data, are implemented in a way, that the same track section is recognized based on the fixed lap meter. However, a more realistic approach would be the identification of similar curves and critical segments based on the track characteristics.
The suggested MPC also faces notable challenges. Due to the limited data base, the effect of rapidly changing classifier outputs on the solver performance could not be analyzed in this

4 dSpace MicroAutoBox: Real-time hardware platform, that can be used for different rapid control prototyping applications such as powertrain, chassis control, ADAS, electric drives control, x-by-wire... See: `https://www.dspace.com/en/pub/home/products/hw/micautob/microautobox2.cfm`

work. An other key issue is handling the large lateral deviations of drivers who significantly deviate from the reference trajectory. We employed a polynomial approach to avoid abrupt control actions. However, these polynomial references require further individualization to accommodate the driving behavior and driver preferences. While our focus in this thesis was the design and proof of concept of the framework, real-life implications must be considered. The suggested reference trajectories for the different driver classes need to be tested in real-life scenarios to validate their effectiveness. It is essential to collect feedback, either subjective from drivers or objective from the dataset, to determine if the classifier accurately detects changes in the interaction dynamics. This feedback helps refining the MPC configuration. Additionally, the overall acceptance of the system depends on how the driver (or passenger) perceives the driving experience. Perceptions of jerkiness or comfort are subjective, and their variation across the suggested driver categorization must be analyzed independently.
A key limitation remains the lack of a comprehensive dataset. While the framework was tested based on real driving data, the driving conditions were artificially controlled on a test track. Real-world traffic conditions present clearer and more varied challenges, such as accurately quantifying the lateral deviations drivers may face. Testing the framework in real traffic conditions with a larger dataset would provide a more comprehensive understanding of the system's performance and the specific challenges it needs to address.

6.9 Further Concepts for Adapting System Behavior

The shared control framework presented in this thesis offers a comprehensive solution for lateral shared driving. However, it is important to recognize that several components of the framework can be utilized independently. We suggest two further approaches to adapt system behavior based on two framework's components: the steering-interaction model and the IPPT algorithm.
The driver-steering interaction model provides a detailed driver analysis at each time step. The conflict torque reflects the driver's non-acceptance of assistance interventions, while the activity torque quantifies the driver's active input to the steering task. In [99], a framework for tuning ADAS was introduced. It suggests that two degree of freedom influence the driver perception: intervention dominance, which controls the strength of system interventions, and planning adaption, which determines the system's ability to adjust to the driver's behavior. The driver-steering interaction model aligns seamlessly with this framework. Specifically, the activity torque can be employed as a control parameter to tune the intervention dominance. An active driver, who exerts greater input to the steering task, requires less assertive assistance torque, whereas a passive driver benefits from stronger interventions. Conversely, the conflict torque serves as a control parameter for planning adaption. Higher conflict values indicate resistance to assistance, prompting the system to adapt more closely to the driver's target, while lower conflict values enable the system to maintain its original target. An illustration of the concept is given in Figure 6.25.
The IPPT algorithm can also be used independently for adapting assistance system behavior. The approach was published by the author in [95]. As presented in Algorithm 8, the

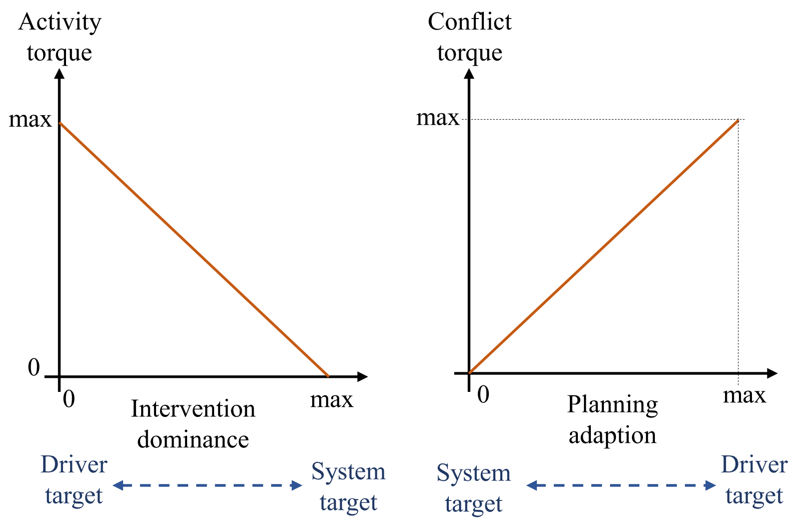


Figure 6.25: Concept for tuning an assistance system based on the driver-steering interaction model.

IPPT algorithm provides the Following and Increasing Tendency values (5.33) and (5.36), respectively, at each step. We define the combined value $\overline{T}^j$ as

$$\overline{T}^j = \sqrt{\overline{FT}^j_{\mathcal{L}} \cdot IT^j_{\geq 0}}, \tag{6.28}$$

where

$$\overline{FT}^j_{\mathcal{L}} = \frac{FT^j_{\mathcal{L}}}{\sum_{k=1}^{5} FT^k_{\mathcal{L}}}, \tag{6.29}$$

and where $IT^j_{\geq 0}$ considers only the positive part of IT^j.
The main idea is to find a new reference line in the α-coordinate system, that has the minimal distance from the resulted tendencies. Hence, the new reference line is computed by solving the optimization problem

$$\min_{0 \leq x_i \leq 1} \sum_i \sum_j \overline{T}^j_i \cdot \left\| x_i - \alpha^j_i, \overline{T}^j_i \right\|^2_2 \tag{6.30a}$$

$$+ \lambda_1 \cdot \sum_i (x_{i+1} - x_i)^2 \tag{6.30b}$$

$$+ \lambda_2 \cdot \sum_i (x_{i+2} - 2x_{i+1} - x_i)^2. \tag{6.30c}$$

The initial term of the cost function in (6.30) guarantees that the distance to the combined values $\overline{T}^j$ is minimized. The additional terms ensure a smooth transition between the points along the path.

7 Conclusion and Outlook

A big challenge in the development of autonomous and assisted driving functions within the automotive industry remains in ensuring their acceptance by drivers and the integration in their driving routines. In the context of this industrial PhD project, the main objective was to design and implement a driver interaction based framework for shared lateral driving to analyze and enhance the interaction between drivers and steering driver assistance functions. The research objectives of the thesis can be divided into the following three categories:

First, a methodology for capturing the different dynamics involved in the shared control were provided. While the vehicle dynamics were predicted based on a common single-track model, we proposed a new mathematical model, the driver-steering interaction model, to analyze the interaction between the driver and the assistance system while they share the steering task. The model allows a comprehensive analysis of the driver torque, the assistance torque, and their combined effect on the vehicle's steering behavior. To accurately represent these interactions, the driver-steering interaction model requires the formulation of a quadratic program. We applied the Varying Coefficient (VC) method to effectively formulate this QP.

Next, a driver interaction classifier was constructed based on a designed real-driving experimental framework. We identified and suggested suitable features for the classification. The driver-steering interaction model was used to compute conflict and passivity features. In addition, we defined metrics to describe path consistency and path quality. To further refine our analysis, we introduced the Individual Path Pattern Targeting (IPPT) algorithm, to identify specific patterns in the driven paths. We then introduced and categorized the identified driver interaction strategies into five distinct classes: adaptation, persistence, selective persistence, nonintervention and uncertainty. The classification was validated by a comparison with subjective expert assessments. Additionally, Dynamic Mode Decomposition (DMD) was applied to analyze the underlying dynamics for each class.

Finally, we presented concepts for adapting the system behavior in real-time based on the driver interaction strategy. We designed a Model Predictive Control (MPC) and conducted a performance analysis to ensure it aligns with the desired system behavior we suggested for each driver category. To achieve real-time optimization, we applied online active-set methods and interior point methods to solve the quadratic programs efficiently. The framework was implemented with a focus on real-world applicability, which makes it easily integrable into practical driving systems. In addition, we suggested further concepts for adapting the system behavior based on independent components of the framework.

However, several limitations were identified during the development and testing of this framework. We clearly outlined these limitations in the thesis. The lack of a comprehensive data set was a significant constraint. On one hand, it limited the classification process to accurately identify further features, especially for the selective persistence and the uncertainty interaction strategies. On the other hand, the MPC could not be tested on varying classifier outputs. The lack of direct feedback from an active driver or an accurate driver model limited the system's ability to adapt in real-time. Without an actual driver on board, the MPC could operate without considering the immediate responses and adjustments a human driver would make. This absence of real-time interaction means the MPC might behave in ways that are not fully aligned with the driver's intentions. Furthermore, the artificially controlled driving conditions restricted our understanding of the potential challenges the MPC might encounter in real-world scenarios. These controlled conditions do not fully capture the complexity and unpredictability of real-world driving. Although the current framework has demonstrated significant potential, future work should focus on addressing these limitations. Expanding the dataset to include a wider range of driving behaviors and conditions may improve the classifier's accuracy and the MPC's adaptability. Real-world testing is also imperative to validate the system's performance and identify the challenges that arise in dynamic environments. Finally, while this thesis focused primarily on lateral control, extending the framework to include longitudinal control is a crucial next step. Integrating longitudinal control would require a more comprehensive understanding of the vehicle's dynamics. For an improved understanding of driver interaction strategies, it is essential to develop algorithms that can identify patterns in longitudinal control. These algorithms will be crucial in analyzing how drivers manage speed and distance, and how these behaviors interact with the assistance system. These insights can improve the design of assistance systems and increase their use.

Acknowledgment

I am grateful to the friends and colleagues within the Volkswagen Group and beyond for their support and kindness, whether through words of encouragement, expressions of sympathy, or small acts of help, all of which helped me carry this thesis to completion.

I would like to thank Prof. Michael Herty for supervising this thesis. Pursuing an external PhD in Mathematics within an industrial environment nearly 500 km away from the university was not always straightforward. I am sincerely grateful for his continued commitment to the project through to its completion. I also wish to thank Prof. Simone Göttlich for taking the time to evaluate the thesis and for her contribution to the examination process.

Finally, this thesis would never have been accomplished without the presence, support, and encouragement of my husband Taha Yassine, my mother Saida and my siblings Sonya, Soufiane and Imen.

Eidesstattliche Erklärung

Amira El Amouri

erklärt hiermit, dass diese Dissertation und die darin dargelegten Inhalte die eigenen sind und selbstständig, als Ergebnis der eigenen originären Forschung, generiert wurden.

Hiermit erkläre ich an Eides statt:

1. Diese Arbeit wurde vollständig oder größtenteils in der Phase als Doktorand dieser Fakultät und Universität angefertigt.
2. Sofern irgendein Bestandteil dieser Dissertation zuvor für einen akademischen Abschluss oder eine andere Qualifikation an dieser oder einer anderen Institution verwendet wurde, wurde dies klar angezeigt.
3. Wenn immer andere eigene- oder Veröffentlichungen Dritter herangezogen wurden, wurden diese klar benannt.
4. Wenn aus anderen eigenen- oder Veröffentlichungen Dritter zitiert wurde, wurde stets die Quelle hierfür angegeben. Diese Dissertation ist vollständig meine eigene Arbeit, mit der Ausnahme solcher Zitate
5. Alle wesentlichen Quellen von Unterstützung wurden benannt.
6. Wenn immer ein Teil dieser Dissertation auf der Zusammenarbeit mit anderen basiert, wurde von mir klar gekennzeichnet, was von anderen und was von mir selbst erarbeitet wurde.
7. Teile dieser Arbeit wurden zuvor veröffentlicht und zwar in:
   - A. El Amouri, P. Markgraf, S. Schacher and M. Herty, "Driver-Steering Interaction Model and Evaluation of Driver Interaction Strategies in Assisted Driving," in IEEE Transactions on Intelligent Vehicles, May 2024, `https://doi.org/10.1109/TIV.2024.3407856`.
   - Amira El Amouri. "Individual Path Pattern Targeting Algorithm for Personalized Shared Lateral Driving." TechRxiv. December 23, 2024. `https://doi.org/10.36227/techrxiv.173496546.60036140/v1`.

Aachen, 23.10.2025

Bibliography

[1] Reiner Marchthaler and Sebastian Dingler. *Kalman-Filter*. 01 2017. ISBN 978-3-658-16727-1. doi: 10.1007/978-3-658-16728-8.

[2] Bureau de prévention des accidents BPA. Fahrerassistenzsysteme: Viele autofahrer schalten sie bewusst aus. March 2020. URL `https://www.agvs-upsa.ch/de/news/news-archiv/fahrerassistenzsysteme-viele-autofahrer-schalten-sie-bewusst-aus`.

[3] Adnan Shaout, Dominic Colella, and S.s Awad. Advanced driver assistance systems - past, present and future. *Proc. 7th Int. Computer Engineering Conf*, December 2011. doi: 10.1109/ICENCO.2011.6153935.

[4] Md Rahman. *Driver acceptance of advanced driver assistance systems and semi-autonomous driving systems*. PhD thesis, August 2016.

[5] Marcel Woide, Dina Stiegemeier, Stefan Pfattheicher, and Martin Baumann. measuring driver-vehicle cooperation: Development and validation of the human-machine-interaction-interdependence questionnaire (hmii). *Transportation Research Part F: Traffic Psychology and Behaviour*, 83:424–439, November 2021. doi: 10.1016/j.trf.2021.11.003.

[6] Fjollë Novakazi, Julia Orlovska, Lars-Ola Bligård, and Casper Wickman. Stepping over the threshold - linking understanding and usage of automated driver assistance systems (adas). *Transportation Research Interdisciplinary Perspectives*, 8, November 2020. doi: 10.1016/j.trip.2020.100252.

[7] Yuan Liao, Minjuan Wang, Lian Duan, and Fang Chen. Cross-regional driver-vehicle interaction design: An interview study on driving risk perceptions, decisions, and adas function preferences. *IET Intelligent Transport Systems*, 12, 04 2018. doi: 10.1049/iet-its.2017.0241.

[8] Federica Biassoni, Daniele Ruscio, and Maria Ciceri. Limitations and automation. the role of information about device-specific features in adas acceptability. *Safety Science*, 85:179–186, 06 2016. doi: 10.1016/j.ssci.2016.01.017.

[9] Joonwoo Son, Myoungouk Park, and B. Park. The effect of age, gender and roadway environment on the acceptance and effectiveness of advanced driver assistance systems. *Transportation Research Part F: Traffic Psychology and Behaviour*, 31, 05 2015. doi: 10.1016/j.trf.2015.03.009.

[10] Thierry Bellet, Jean-Christophe Paris, and Claude Marin-Lamellet. Difficulties experienced by older drivers during their regular driving and their expectations towards advanced driving aid systems and vehicle automation. *Transportation*

Research Part F: Traffic Psychology and Behaviour, 52:138–163, 01 2018. doi: 10.1016/j.trf.2017.11.014.

[11] Timo Guenthner. The moderating influence of life events on the acceptance of advanced driver assistance systems in aging societies. *Computers in Human Behavior Reports*, 7:100202, 05 2022. doi: 10.1016/j.chbr.2022.100202.

[12] Azra Habibovic and Johan Davidsson. Requirements of a system to reduce car-to-vulnerable road user crashes in urban intersections. *Accident; analysis and prevention*, 43:1570–80, 07 2011. doi: 10.1016/j.aap.2011.03.019.

[13] Kai Eckoldt, Martin Knobel, Marc Hassenzahl, and Josef Schumann. An experiential perspective on advanced driver assistance systems. *it - Information Technology*, 54: 165–171, August 2012. doi: 10.1524/itit.2012.0678.

[14] Chulwoo Moon. An objective evaluation method for driver/passenger acceptance of an autonomous driving system for lane changes. *Applied Sciences*, 13(17), 2023. ISSN 2076-3417. doi: 10.3390/app13179601. URL `https://www.mdpi.com/2076-3417/13/17/9601`.

[15] Chulwoo Moon, Youngseok Lee, Chang-Hyun Jeong, and Seibum Choi. Investigation of objective parameters for acceptance evaluation of automatic lane change system. *International Journal of Automotive Technology*, 19:179–190, 02 2018. doi: 10.1007/s12239-018-0017-0.

[16] Chouki Sentouh. Toward a shared lateral control between driver and steering assist controller. volume 43, pages 404–409, 8 2010. ISBN 9783902661944. doi: 10.3182/20100831-4-FR-2021.00071.

[17] Amirhossein Ghasemi, Paramsothy Jayakumar, and Brent Gillespie. Shared control architectures for vehicle steering. *Cognition, Technology and Work*, 21, November 2019. doi: 10.1007/s10111-019-00560-9.

[18] Yoshiyuki Tanaka, Naoki Yamada, Toshio Tsuji, and Takamasa Suetomi. Vehicle active steering control system based on human mechanical impedance properties of the arms. *Intelligent Transportation Systems, IEEE Transactions on*, 15:1758–1769, August 2014. doi: 10.1109/TITS.2014.2312458.

[19] Ziya Ercan, Ashwin Carvalho, M. Gokasan, and Francesco Borrelli. Modeling, identification, and predictive control of a driver steering assistance system. *IEEE Transactions on Human-Machine Systems*, PP:1–11, September 2017. doi: 10.1109/THMS.2017.2717881.

[20] Martina Hasenjäger and Heiko Wersing. Personalization in advanced driver assistance systems and autonomous vehicles: A review. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, 2017. doi: 10.1109/ITSC.2017.8317803.

[21] Gennaro Nicola Bifulco, Luigi Pariota, Fulvio Simonelli, and Roberta Di Pace. Development and testing of a fully adaptive cruise control system. *Transportation Research Part C: Emerging Technologies*, 29:156–170, 01 2013.

[22] Amira El Amouri, Peter Markgraf, and Samuel Schacher. Verfahren zum betreiben eines fahrerassistenzsystems für ein kraftfahrzeug, fahrerassistenzsystem und kraftfahrzeug. (DE 102022206361), January 2024. https://depatisnet.dpma.de/DepatisNet/depatisnet?action=bibdat&docid=DE102022206361A1.

[23] Amira El Amouri and Peter Markgraf. Verfahren und vorrichtung zur ermittlung der interaktion eines fahrers mit einem elektromechanischen lenksystem. (DE 102023200821), February 2024. https://depatisnet.dpma.de/DepatisNet/depatisnet?action=bibdat&docid=DE102023200821B3.

[24] Amira El Amouri and Samuel Schacher. Verfahren zum assistieren eines fahrers beim steuern eines kraftfahrzeugs sowie kraftfahrzeug. (DE 102023201074), October 2024. https://depatisnet.dpma.de/DepatisNet/depatisnet?action=bibdat&docid=DE102023201074B4.

[25] Amira El Amouri, Sevsel Gamze Kabil, Björn Mennenga, and Samuel Schacher. Verfahren zur steuerung eines fahrzeugs durch ein assistenzsystem, assistenzsystem und fahrzeug. (EP 24150620), July 2024. https://depatisnet.dpma.de/DepatisNet/depatisnet?action=bibdat&docid=EP000004406802A1.

[26] Amira El Amouri, Sevsel Gamze Kabil, Björn Mennenga, and Samuel Schacher. Method and device for autonomous movement of a vehicle in a variably optimized dynamic driving state. (US 202418424270), August 2024. https://depatisnet.dpma.de/DepatisNet/depatisnet?action=bibdat&docid=US020240253648A1.

[27] Amira El Amouri and Peter Markgraf. Verfahren und vorrichtung zur ermittlung der interaktion eines fahrers mit einem elektromechanischen lenksystem. (EP 2023087703), August 2024. https://depatisnet.dpma.de/DepatisNet/depatisnet?action=bibdat&docid=WO002024160453A1.

[28] Christian Heumann, Michael Schomaker, and Shalabh Shalabh. *Introduction to Statistics and Data Analysis*. 01 2016. ISBN 978-3-319-46160-1. doi: 10.1007/978-3-319-46162-5.

[29] Ekkehart Schlicht. Vc: a method for estimating time-varying coefficients in linear models. *Journal of the Korean Statistical Society*, 50, March 2021. doi: 10.1007/s42952-021-00110-y.

[30] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA, 2017. ISBN 978-1491962299.

[31] Jiawei Yang, Zeping Wu, Zhixiang Wang, Dequan Zhang, Wenjie Wang, Qian Wen, and Zhang Weihua. Enhanced anisotropic radius basis function metamodel based on recursive evolution latin hypercube design and fast k-fold cross-validation. *Structural and Multidisciplinary Optimization*, 66, 07 2023. doi: 10.1007/s00158-023-03597-7.

[32] Z.H. Zhou and S. Liu. *Machine Learning*. Springer Nature Singapore, 2021. ISBN 9789811519666. URL `https://books.google.de/books?id=Zd5hywEACAAJ`.

[33] Philip Holmes. Poincaré, celestial mechanics, dynamical-systems theory and "chaos". *Physics Reports*, 193:137–163, 09 1990. doi: 10.1016/0370-1573(90)90012-Q.

[34] George D. Birkhoff. *Dynamical systems*, volume 9; 1927; 305 pp; SoftcoverMSC: Primary 34;. 1927. ISBN 978-0-8218-1009-5.

[35] David Aubin. George david birkhoff 'dynamical systems (1927)'. *Landmark writings in western mathematics, 1640-1940, 871-881 (2005)*, 12 2005. doi: 10.1016/B978-044450871-3/50149-2.

[36] Ali Hasan Nayfeh. Franklin institute award, 2 2014, Retrieved 24 May 2022. URL `https://www.fi.edu/laureates/ali-hasan-nayfeh`.

[37] Jan Prüss and Mathias Wilke. *Gewöhnliche Differentialgleichungen und dynamische Systeme*. 01 2019. ISBN 978-3-030-12361-1. doi: 10.1007/978-3-030-12362-8.

[38] Anton Braun. *Dynamische Systeme: Modellierung mit den Methoden der Laplace-Transformation*. 01 2019. ISBN 978-3-658-18184-0. doi: 10.1007/978-3-658-18185-7.

[39] Stephen Lynch. *Dynamical Systems with Applications using MATLAB®*. 01 2004. ISBN 0817643214. doi: 10.1007/978-3-319-06820-6.

[40] Jürgen Adamy. *Nonlinear Systems and Controls*. 01 2024. ISBN 978-3-662-68689-8. doi: 10.1007/978-3-662-68690-4.

[41] Wolfgang Dahmen and Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. 01 2008. ISBN 978-3-540-76492-2. doi: 10.1007/978-3-540-76493-9.

[42] Pablo Pedregal. *Introduction to Optimization*. Springer New York, 2004. ISBN 978-0-387-21680-5. doi: 10.1007/b97412.

[43] Patricia Yagi, Erik Papa, and Miguel Cano. *A Systematic Literature Review on Quadratic Programming*, pages 739–747. 01 2023. ISBN 978-981-19-2396-8. doi: 10.1007/978-981-19-2397-5_66.

[44] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. 1 2006. ISBN 978-0-387-30303-1. doi: 10.1007/978-0-387-40065-5.

[45] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[46] Matthias Gerdts. *Optimal Control of Ordinary Differential Equations and Differential-Algebraic Equations*. 01 2006.

[47] James Longuski, José Guzmán, and John Prussing. *Optimal Control with Aerospace Applications*. 01 2014. ISBN 978-1-4614-8944-3. doi: 10.1007/978-1-4614-8945-0.

[48] Suresh Sethi. *Optimal Control Theory: Applications to Management Science and Economics*. 01 2021. ISBN 978-3-030-91744-9. doi: 10.1007/978-3-030-91745-6.

[49] G.S. Christensen, Mo El-Hawary, and Soliman Soliman. *Optimal Control Applications in Electric Power Systems*. 01 1987. ISBN 978-1-4899-2087-4. doi: 10.1007/978-1-4899-2085-0.

[50] Moritz Werling. *Optimale Trajektorien*, pages 1183–1204. 07 2024. ISBN 978-3-658-38485-2. doi: 10.1007/978-3-658-38486-9_43.

[51] Saša V. Raković and William Levine. *Handbook of Model Predictive Control*. 09 2018. ISBN 978-3-319-77488-6.

[52] Gjerrit Meinsma and Arjan Schaft. *A Course on Optimal Control*. 01 2023. ISBN 978-3-031-36654-3. doi: 10.1007/978-3-031-36655-0.

[53] Stephen Wright. Applying new optimization algorithms to model predictive control. 93, 06 1996.

[54] Alexander Domahidi. *Methods and tools for embedded optimization and control*. PhD thesis, ETH Zürich, 2013. Available at `http://dx.doi.org/10.3929/ethz-a-010010483`.

[55] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2:575–601, 11 1992. doi: 10.1137/0802028.

[56] RICHARD BELLMAN. *Dynamic Programming*. 08 2021. ISBN 9780691146683. doi: 10.2307/j.ctv1nxcw0f.

[57] Francisco Chinesta, Roland Keunings, and Adrien Leygue. *The proper generalized decomposition for advanced numerical simulations. A primer*. 1 2014. ISBN 978-3-319-02864-4. doi: 10.1007/978-3-319-02865-1.

[58] Dengpeng Huang, Jan Fuhg, Christian Weißenfels, and Peter Wriggers. A machine learning based plasticity model using proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 365:113008, 06 2020. doi: 10.1016/j.cma.2020.113008.

[59] J. Kutz, Steven Brunton, Bingni Brunton, and Joshua Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. 11 2016. ISBN 978-1-611974-49-2.

[60] Steven Brunton and J. Kutz. *Data-Driven Science and Engineering:Machine Learning, Dynamical Systems, and Control*. 01 2019. ISBN 9781108422093. doi: 10.1017/9781108380690.

[61] Peter Schmid and Jörn Sesterhenn. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656, 11 2008. doi: 10.1017/S0022112010001217.

[62] Peter Schmid. Dynamic mode decomposition of numerical and experimental data. 25: 249–259, 07 2010.

[63] Jonathan Tu, Clarence Rowley, Dirk Luchtenburg, Steven Brunton, and J. Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1, 11 2013. doi: 10.3934/jcd.2014.1.391.

[64] N. Erichson, Steven Brunton, and J. Kutz. Compressed dynamic mode decomposition for background modeling. *Journal of Real-Time Image Processing*, 16, 10 2019. doi: 10.1007/s11554-016-0655-2.

[65] Joshua Proctor and Philip Welkhoff. Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *International health*, 7:139–45, 03 2015. doi: 10.1093/inthealth/ihv009.

[66] Omri Azencot, Wotao Yin, and Andrea Bertozzi. Consistent dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18:1565–1585, 01 2019. doi: 10.1137/18M1233960.

[67] N. Erichson, Steven Brunton, and J. Kutz. Compressed dynamic mode decomposition for background modeling. *Journal of Real-Time Image Processing*, 16, 10 2019. doi: 10.1007/s11554-016-0655-2.

[68] Joshua Proctor, Steven Brunton, and J. Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15, 09 2014. doi: 10.1137/15M1013857.

[69] José Vega and Soledad Le Clainche. *Higher order dynamic mode decomposition*, pages 29–83. 01 2021. ISBN 9780128197431. doi: 10.1016/B978-0-12-819743-1.00009-4.

[70] Thomas Sheridan, W. Verplank, and T. Brooks. Human and computer control of undersea teleoperators. 12 1978.

[71] David Abbink, Tom Carlson, Mark Mulder, Joost de Winter, Farzad Aminravan, Tricia Gibo, and Erwin Boer. A topology of shared control systems—finding common ground in diversity. *IEEE Transactions on Human-Machine Systems*, 48:509–525, 09 2018. doi: 10.1109/THMS.2018.2791570.

[72] Joseba Sarabia, Mauricio Marcano, Joshué Pérez Rastelli, Asier Zubizarreta, and Sergio Diaz. A review of shared control in automated vehicles: system evaluation. *Frontiers in Control Engineering*, 3:1058923, 02 2023. doi: 10.3389/fcteg.2022.1058923.

[73] Thomas Sheridan, W. Verplank, and T. Brooks. Human and computer control of undersea teleoperators. 12 1978.

[74] David Kaber and Mica Endsley. The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task. *Theoretical Issues in Ergonomics Science*, 5:113–153, 03 2004. doi: 10.1080/1463922021000054335.

[75] Toshiyuki Inagaki. Adaptive automation: Sharing and trading of control. *The Proceedings of the Transportation and Logistics Conference*, 2001.10, 12 2001. doi: 10.1299/jsmetld.2001.10.79.

[76] Mica Endsley and David Kaber. Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics*, 42:462–92, 04 1999. doi: 10.1080/001401399185595.

[77] Michael Flad, Lukas Frohlich, and Soeren Hohmann. Cooperative shared control driver assistance systems based on motion primitives and differential games. *IEEE Transactions on Human-Machine Systems*, PP:1–12, 05 2017. doi: 10.1109/THMS.2017.2700435.

[78] Chunshi Guo, Chouki Sentouh, Jean-Christophe Popieul, and Jean-Baptiste Haue. Mpc-based shared steering control for automated driving systems. pages 129–134, 10 2017. doi: 10.1109/SMC.2017.8122590.

[79] Mingjun Li, Haotian Cao, Xiaolin Song, and Yanjun Huang. Shared control driver assistance system based on driving intention and situation assessment. *IEEE Transactions on Industrial Informatics*, 14:4982–4994, 08 2018. doi: 10.1109/TII.2018.2865105.

[80] Ryota Nishimura, Takahiro Wada, and Seiji Sugiyama. Haptic shared control in steering operation based on cooperative status between a driver and a driver assistance system. *Journal of Human-Robot Interaction*, 4:19–37, 12 2015. doi: 10.5898/4.3.Nishimura.

[81] Stefano Di Cairano and Ilya V Kolmanovsky. Automotive applications of model predictive control. *Handbook of model predictive control*, pages 493–527, 2019.

[82] Ryo Kondo, Takahiro Wada, and Kohei Sonoda. Use of haptic shared control in highly automated driving systems. 09 2019.

[83] Renjie Li, Yanan Li, Shengbo Li, Chaofei Zhang, Etienne Burdet, and Bo Cheng. Indirect shared control for cooperative driving between driver and automation in steer-by-wire vehicles. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–11, 08 2020. doi: 10.1109/TITS.2020.3010620.

[84] Biao Ma, Liu Yulong, Xiaoxiang Na, Yahui Liu, and Yiyong Yang. A shared steering controller design based on steer-by-wire system considering human-machine goal consistency. *Journal of the Franklin Institute*, 356:4397–4419, 04 2019. doi: 10.1016/j.jfranklin.2018.12.028.

[85] Franck Mars, Mathieu Deroo, and Jean-Michel Hoc. Analysis of human-machine cooperation when driving with different degrees of haptic shared control. *IEEE Transactions on Haptics*, 7, 07 2014. doi: 10.1109/TOH.2013.2295095.

[86] Ziya Ercan, Ashwin Carvalho, Eric Tseng, Metin Gökaşan, and Francesco Borrelli. A predictive control framework for torque-based steering assistance to improve safety in highway driving. *Vehicle System Dynamics*, 56:1–22, 06 2017. doi: 10.1080/00423114.2017.1337915.

[87] Amira El Amouri, Peter Markgraf, Samuel Schacher, and Michael Herty. Driver-steering interaction model and evaluation of driver interaction strategies in assisted driving. *IEEE Transactions on Intelligent Vehicles*, 10(1):3–12, 2025. doi: 10.1109/TIV.2024.3407856.

[88] Dieter Schramm, Manfred Hiller, and Roberto Bardini. *Vehicle Dynamics: Modeling and Simulation*. 01 2018. ISBN 978-3-662-54482-2. doi: 10.1007/978-3-662-54483-9.

[89] Massimo Guiggiani. *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*. 05 2018. ISBN 978-3-319-73219-0. doi: 10.1007/978-3-319-73220-6.

[90] R Rajamani. *Vehicle Dynamics and Control*. 01 2006. ISBN 0-387-26396-9. doi: 10.1007/0-387-28823-6.

[91] Hans-Peter Willumeit. *Modelle und Modellierungsverfahren in der Fahrzeugdynamik*. 01 1998. ISBN 978-3-663-12248-7. doi: 10.1007/978-3-663-12247-0.

[92] Peter Pfeffer and Manfred Harrer. *Lenkungshandbuch: Lenksysteme, Lenkgefühl, Fahrdynamik von Kraftfahrzeugen*. January 2013. ISBN 978-3-658-00976-2. doi: 10.1007/978-3-658-00977-9.

[93] Thomas Böhm. A matter of torque electric power steering system. *NXP Semiconductors*, pages 61–66, 2009. URL `https://www.nxp.com/docs/en/brochure/MatterofTorque`.

[94] Wanzhong Zhao. *Vehicle Steer-by-Wire System and Chassis Integration*. 01 2023. ISBN 978-981-19-4249-5. doi: 10.1007/978-981-19-4250-1.

[95] Amira El Amouri. Individual path pattern targeting algorithm for personalized shared lateral driving. *TechRxiv*, December 2024. doi: 10.36227/techrxiv.173496546.60036140/v1.

[96] Peter Markgraf. Identifikation eines datengetriebenen fahrermodells beim unterstützten fahren. Master's thesis, Otto-von-Guericke-Universität Magdeburg, Magdeburg, Germany, 2022.

[97] Jagat Rath, Chouki Sentouh, and Jean Popieul. Personalized lane keeping assist strategy: Adaptation to driving style. *IET Control Theory & Applications*, 13, 01 2019. doi: 10.1049/iet-cta.2018.5941.

[98] Fjollë Novakazi, Julia Orlovska, Lars-Ola Bligård, and Casper Wickman. Stepping over the threshold - linking understanding and usage of automated driver assistance systems (adas). *Transportation Research Interdisciplinary Perspectives*, 8, 11 2020. doi: 10.1016/j.trip.2020.100252.

[99] Samuel Schacher. *Das Mentorensystem Race Trainer - Konzept für ein semi-automatisches Fahrertraining*. PhD thesis, October 2019.

[100] Norah Neuhuber, Paolo Pretto, and Bettina Kubicek. Interaction strategies with advanced driver assistance systems. *Transportation Research Part F: Traffic Psychology and Behaviour*, 88:223–235, 07 2022. doi: 10.1016/j.trf.2022.05.013.

[101] Samuel Schacher and Rudibert King. Driver assistance systems teaching humans how to drive - the mentoring concept of the race trainer. *ATZ worldwide*, 121:60–63, 11 2019. doi: 10.1007/s38311-019-0130-3.

[102] Ingmar Gundlach and Ulrich Konigorski. Modellbasierte online-trajektorienplanung für zeitoptimale rennlinien. *at - Automatisierungstechnik*, 67:799–813, September 2019. doi: 10.1515/auto-2019-0032.

[103] The MathWorks Inc. Optimization toolbox version: 9.4 (r2022b), 2022.

[104] Weiyan Pan, Hui Chen, Wei Ran, Taokai Xia, Yosuke Nishimura, and Jianzhen Wang. A trajectory planning method for different drivers in the curve condition. 12 2021. doi: 10.4271/2021-01-7006.

[105] MathWorks. *MATLAB Classification Learner App*, 2024. Available at: `https://www.mathworks.com/help/stats/classification-learner-app.html`.

[106] H.J. Ferreau, A. Potschka, and C. Kirches. qpOASES webpage. `http://www.qpOASES.org/`, 2007–2015.

[107] Joachim Ferreau, Eckhard Arnold, H. Diedam, Boris Houska, A. Perrin, and T. Wiese. *qpOASES User's Manual*. 01 2009.

[108] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[109] Joachim Ferreau. *Model Predictive Control Algorithms for Applications with Millisecond Timescales*. PhD thesis, 01 2011.

[110] Frieder Gottmann. Lipsol interior point solver for model predictive optimisation, 2020. Unpublished software, Version 1.1, Volkswagen AG.

[111] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari. Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, pages 1–17, 2017.

[112] Alexander Domahidi and Juan Jerez. Forces professional. mbotech AG, `https://embotech.com/FORCES-Pro`, 2014–2019.