



BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS
DEPARTMENT OF CONTROL ENGINEERING AND INFORMATION TECHNOLOGY



HUN-REN INSTITUTE FOR COMPUTER SCIENCE AND CONTROL
SYSTEMS AND CONTROL LABORATORY

Adaptive trajectory-tracking control for small-scale autonomous ground vehicles

Scientific Conference of Students

Advisors:

Dr. Tamás Péni

Senior research fellow

HUN-REN SZTAKI – Systems and Control Lab

Prof. Dr. Roland Tóth

Senior research fellow / Full professor

HUN-REN SZTAKI – Systems and Control Lab

Eindhoven University of Technology – CS Group

Supervisor:

Dr. István Harmati

Associate professor

BME-VIK – IIT

BUDAPEST, 2023

Contents

1	Introduction	6
2	Preliminaries	9
2.1	Linear parameter-varying systems	9
2.2	Linear quadratic regulator	9
2.2.1	LQR for LTI systems	10
2.2.2	LQR for LPV systems	10
2.3	Stability and performance analysis of nonlinear systems	13
2.3.1	Stability of nonlinear systems	13
2.3.2	Lyapunov-functions	14
2.3.3	$\mathcal{L}_2$ -gain calculation of input-output systems	15
2.4	Gaussian Processes	16
2.4.1	Gaussian Process regression	16
2.5	Sparse Gaussian Process regression	17
3	Vehicle model	19
3.1	First principle model	19
3.2	Decoupling	20
3.3	GP augmentation	22
4	Trajectory tracking control	24
4.1	Trajectory tracking scenario	24
4.2	Control Architecture	24
4.3	Control design	25
5	Simulation study	27
5.1	Reference trajectories and error measures	27
5.2	Numerical simulations	28
5.2.1	Vehicle model and parameters	29
5.2.2	Trajectory tracking simulations	30
5.3	MuJoCo simulations	34

5.3.1	Simulation environment	34
5.3.2	Trajectory tracking simulations	34
6	Stability analysis	37
6.1	Closed-loop system	38
6.2	Iterative stability analysis	39
7	$\mathcal{L}_2$ gain analysis	44
7.1	System model	44
7.2	Simulation analysis	44
7.3	Iterative $\mathcal{L}_2$ gain computation	45
8	Conclusion and future work	48
	References	52

Abstract

Nowadays, autonomous mobile robots are starting to appear in various industrial applications, hence, further improvement of their capabilities is a subject of scientific research. Specifically, small-scale ground vehicles are an ideal platform for transportation and surveillance tasks, thanks to their robust and relatively simple mechanical structure. One essential prerequisite for reaching the general utilization of these vehicles is the development of control algorithms that ensure accurate motion and navigation. Research in recent years has resulted in many approaches to these problems. However, applied methods often neglect precise mathematical analysis of the closed-loop stability and performance properties or use approximations to apply classical methods of the linear control theory.

The main objective of this work is to develop trajectory-tracking algorithms that can be implemented on real systems and to propose methods that enable precise, quantitative analysis of the stability and performance properties of the controlled nonlinear vehicle behavior. First, a decoupled path-tracking algorithm is introduced. Here, the nonlinear dynamics are separated into two subsystems, which are complemented with a *Gaussian Process* based learning component to compensate for modeling errors. The subsystems are described in the *linear parameter varying* (LPV) framework, by which parameter-dependent state feedback controllers are synthesized. Then, a novel optimization-based analysis technique is presented, which can provide mathematically correct stability guarantees for arbitrary nonlinear closed-loop systems and give an accurate estimate for the induced $\mathcal{L}_2$ norm that describes the performance of the employed controller.

The results are demonstrated in simulations, first on the dynamical model of a 1/10 scale F1TENTH electric vehicle, then in a high-fidelity physical simulator, which is based on the digital twin model of the real car. These simulations are used to precisely evaluate the performance of the proposed methods in an environment where the vehicle dynamics are precisely known. Finally, future work in connection with the real-world implementation is outlined.

Kivonat

Napjainkban az autonóm mobil robotok számos alkalmazási területen jelennek meg, képességeik fejlesztése tudományos kutatások tárgyát képezi. Specifikusan, a kisméretű földi járművek egy ideális mobil platformot biztosítanak megfigyelési, illetve szállítási feladatok ellátására, köszönhetően a robusztus felépítésüknek és relatíve egyszerű mechanikai struktúrájuknak. Szélesebb körű alkalmazásuk egyik fontos feltétele olyan mozgásszabályozási algoritmusok kifejlesztése, amelyek lehetővé teszik a hatékony és precíz navigációt. Az utóbbi évek számos kiemelkedő tudományos eredményt hoztak ezen a területen. Ugyanakkor a bemutatott stratégiák a zárt kör stabilitási tulajdonságainak, illetve az irányítás minőségi jellemzőinek precíz matematikai elemzését sokszor elhanyagolják, vagy egyszerűsített, lineáris irányításelméleti módszerekkel, közelítő modelleken vizsgálják azt.

A dolgozat a fő célja adaptív, valós rendszerekhez igazodó pályakövető irányítási stratégiák fejlesztése, valamint olyan elemzési módszerek kidolgozása, amelyekkel a zárt kör stabilitás- és performancia tulajdonságaira precíz, könnyen számolható kvantitatív jellemzők határozhatók meg. Ennek érdekében először egy szétcsatolt irányítási stratégia kerül ismertetésre. Itt az összetett, nemlineáris járműmodellt két alrendszerre bontjuk, melyeket *Gauss folyamat* alapú tanuló komponensekkel egészítünk ki, a modellezési hibák kompenzálásához. Az alrendszereket *lineáris változóparaméterű* (LPV) alakban felírva, paraméterfüggő állapot visszacsatolást tervezünk. Ezt követően egy olyan újszerű optimalizáció alapú módszert ismertetünk, amellyel matematikailag korrekt módon igazolható a zárt kör stabilitása és pontos becslés adható az irányítás performanciáját jellemző indukált $\mathcal{L}_2$ normára.

Az eredmények először 1/10 méretarányú F1TENTH elektromos hajtású autó dinamikus modelljén, majd egy nagy megbízhatóságú fizikai szimulátorban implementált digitális ikermodellen kerülnek bemutatásra és validálásra, mivel ez lehetőséget nyújt a performancia hatékony mérésére az ideális rendszerleírás ismerete miatt. Végül a valós járműveken történő implementációhoz szükséges jövőbeli fejlesztési terveket ismertetjük.

1 Introduction

In recent years, the application of autonomous mobile robots has gained significant attention in industry and various research areas such as transportation, logistics and production support. Specifically, the utilization of car-like robots is beneficial, as they can provide an agile and versatile mobile platform that can be used in multiple different environments and changing operating conditions. However, to fully harness the capabilities of such vehicles, efficient motion control algorithms are required that can further improve the autonomous maneuvering capabilities of these vehicles.

An important aspect of maneuvering is trajectory tracking, which has been a widely researched topic in recent years [1]. The baseline techniques, still often utilized because of their simplicity, include the Pure Pursuit [2] and the Stanley [3] methods. These controllers are both derived from kinematic and geometric relations and can be easily implemented without expert knowledge of control system theory. However, as these algorithms do not consider dynamic effects, they cannot be used in agile maneuvering scenarios, where the vehicle operates closer to its physical limit. *Proportional integral derivative* (PID) approaches have also been utilized for path-tracking [4], but they also have significant drawbacks: these controllers are usually tuned "empirically" from measurements, or based on simplified linearized models of the vehicle dynamics. *Linear quadratic regulator* (LQR) [5] has also been utilized in both kinematic and dynamic contexts, however, linearizations in the control-oriented models quickly lead to performance degradation when the vehicle performs agile maneuvers and the effect of nonlinearities become more significant. The current state-of-the-art trajectory tracking algorithms are usually studied in the context of autonomous racing. They include model based approaches such as *sliding mode control* (SMC), *nonlinear model predictive control* (NMPC), dynamic inversion, and also learning based control solutions [6].

For model based techniques, either first principle modeling [7] or system identification [8] is utilized. The main advantage of first principle dynamic modeling is that the model states and parameters usually have clear physical interpretations. However, these physically derived vehicle models are highly nonlinear, complicated and difficult to tune for the task of control design, therefore, engineers often utilize linearization and simplification which results in inherent model mismatches. On the other hand, system identification techniques can provide a simpler linear structure, but we lose the physical interpretation and augmentation of models by preserving interpretability is in its infancy. For trajectory-tracking control both learning based identification approaches [9] and first principle modeling [10] have been successfully utilized in the MPC framework.

Another important aspect of control algorithms is the adaptability to changing environmental conditions. While system identification techniques can capture the changes in the model or the environment, the reidentification of first principle models is usually a time-consuming and cumbersome procedure. To tackle this problem, [11] also proposed an augmentation method, where states of the nominal vehicle dynamics have been aug-

mented with *Gaussian Processes* (GPs) to capture model mismatch. However, all these methods rely on MPC algorithms for trajectory tracking, instead of which, the less computationally demanding feedback-based solutions are still often favored [12].

It is also important to note that although the mathematically correct stability guarantees are one of the most important properties of control systems, the previously introduced works usually validate their results with simulations only. Other than that, the stability properties of the control algorithms are only proven with linearized and kinematic models, and the study of the complete, nonlinear closed-loop system is neglected. The main reason for this is the fact, that the examination of the closed loop of a dynamic vehicle model can be cumbersome because of the strong nonlinearities and the complexity of the model. However, in the context of mobile robotics mathematical guarantees of stability and performance are essential to provide safe and predictable operation.

This paper focuses on autonomous maneuvering in the context of autonomous mobile robotics. Therefore, we aim to develop trajectory-tracking algorithms that provide (a) robustness and adaptability to modeling uncertainties and changing environmental conditions; (b) straightforward, computationally efficient implementation; (c) guaranteed stability and performance in the operating range. While the previous references successfully tackle (a), they lack computational efficiency and straightforward implementation, due to online optimization of the MPC. Furthermore, formal verification of the closed-loop stability of trajectory-tracking controllers is often neglected or only analyzed with simplified linear model structures.

Therefore, to address the arising questions, the main contributions of this work are as follows:

- C1 Adaptive control design:** We propose a computationally efficient adaptive trajectory-tracking control algorithm for car-like mobile robots, based on the *linear parameter-varying* (LPV) framework and GPs. The tracking algorithm utilizes a decoupled approach, where nonlinear vehicle dynamics are decoupled into two subsystems corresponding to the longitudinal and lateral motions. The subsystems are derived in an LPV representation, by which parameter-dependent feedback controllers are designed. In order to adapt to varying environmental conditions and uncertain parameters, each subsystem is augmented with GPs to compensate for modeling errors and changing environmental conditions.
- C2 Nonlinear analysis:** Extending the results of [13], we present an optimization-based analysis technique, which can provide mathematically correct stability guarantees for arbitrary nonlinear closed-loop systems and give an accurate estimate for the induced $\mathcal{L}_2$ norm that describes the performance of the employed controller.
- C3 Implementation:** We demonstrate and evaluate the efficiency of the proposed control scheme of the digital twin model of an autonomous 1/10 scale electric F1TENTH vehicle [14] implemented in MuJoCo [15], a high-fidelity physical simulator.

The remainder of this work is organized as follows. Section 2 collects the theoretical foundations of this work, which are essential for the latter sections. Then, Section 3 overviews the utilized vehicle model and presents the decoupling and the GP augmentation of the nonlinear vehicle dynamics. It is followed by Section 4 which outlines our proposed learning based trajectory tracking approach. Finally, Section 5 presents the efficiency of our algorithm through simulation experiments. The stability and performance analysis of the nonlinear closed loop are presented in Section 6 and 7, respectively. Lastly, we summarize our results and future work in Section 8.

2 Preliminaries

The main purpose of this section is to collect the theoretical foundations of the methods and techniques used throughout the subsequent paragraphs. First, the *linear parameter-varying* (LPV) framework is introduced briefly, which provides a useful toolkit for nonlinear system analysis and control. Then, the *linear quadratic regulator* (LQR) problem is defined and extended for LPV systems. The next section overviews possible approaches to the analysis of nonlinear systems and outlines Lyapunov functions and their utilization for stability analysis. It is followed by the definition of the $\mathcal{L}_2$ gain, which is a quantifiable performance measure of nonlinear systems. Lastly, the brief theory of GP based function approximation is presented.

2.1 Linear parameter-varying systems

Throughout this work, the control-oriented models and the model-based control algorithms are derived within the *linear parameter-varying* (LPV) framework [16, 17]. The utilization of the LPV framework is beneficial, as it can describe complex, nonlinear behavior while preserving the advantageous properties of the linear model structure. The state-space representation of the LPV model structure can be expressed as

$$\dot{x}(t) = A(\rho(t))x(t) + B(\rho(t))u(t), \quad (2.1)$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, $u \in \mathbb{R}^{n_u}$ is the input vector. Furthermore, $\rho : \mathbb{R} \rightarrow \mathbb{R}^{n_\rho}$ is a function describing a time-varying signal, the so-called scheduling variable [17], while $A : \mathbb{R}^{n_\rho} \rightarrow \mathbb{R}^{n_x \times n_x}$ and $B : \mathbb{R}^{n_\rho} \rightarrow \mathbb{R}^{n_x \times n_u}$ are matrix functions describing the state and input matrices of the system. In LPV systems, we assume that the scheduling variable is known, i.e. $\rho(t)$ can be measured or calculated for any time t . For brevity, the time dependence of x , u and ρ will be dropped from the notation in the remainder of this work.

2.2 Linear quadratic regulator

The *linear quadratic regulator* (LQR) [18] is an optimal control technique that can be used to synthesize and tune state-feedback controllers intuitively. In this section, first, the LQR problem is introduced and solved for *linear time invariant* (LTI) systems. This synthesis technique is widely used and is built in many existing control related software tools such as Matlab [19] or the Python Control Systems Library [20]. However, to solve the LQR problem for LPV systems the standard Ricatti equation based solution cannot be used. Therefore, in the next part of this section, the LQR problem is extended for the LPV system by utilizing *semidefinite programming* (SDP) and a software implementation is also proposed.

2.2.1 LQR for LTI systems

First, in this section, we consider continuous-time LTI systems in the form of

$$\dot{x} = Ax + Bu, \quad (2.2)$$

where $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ are the state and input vectors of the system and $A \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_x \times n_u}$ are the system and input matrices, respectively. To stabilize (2.2) to the origin full state feedback can be applied with feedback matrix K which results in the following control law:

$$u = Kx, \quad (2.3)$$

where the value K is calculated by minimizing a quadratic cost function in the form of

$$J = \int_0^\infty (x^\top Qx + u^\top Ru) dt. \quad (2.4)$$

In (2.4) $Q \succeq 0$ and $R \succ 0$ are manually selected weighing matrices with the appropriate dimensions. Formally, the optimization problem can be formulated as

$$\min_K \int_0^\infty (x^\top Qx + u^\top Ru) dt, \quad (2.5a)$$

$$\text{subject to } \dot{x} = (A + BK)x. \quad (2.5b)$$

The optimal values for K that satisfies (2.5) can be calculated in closed-form as

$$K = -R^{-1}B^\top P, \quad (2.6)$$

where P also needs to satisfy the algebraic Ricatti equation:

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0. \quad (2.7)$$

Furthermore, it is important to note that the state-space representation (2.2) needs to satisfy the conditions of controllability, i.e.

$$M_c = \begin{bmatrix} B & \dots & A^i B & \dots & A^{n_x-1} B \end{bmatrix}, \quad (2.8)$$

controllability matrix needs to have full rank.

2.2.2 LQR for LPV systems

The main goal of this section is to extend the LQR control problem to the LPV framework based on [21]. While the Ricatti based LQR synthesis offers a convenient way for the controller design, it cannot be extended for LPV systems. However, the LQR problem can also be formulated with the help of *linear matrix inequalities* (LMIs), which solution can easily be extended to LPV systems.

Consider the LPV system in state space form introduced in (2.1):

$$\dot{x} = A(\rho)x + B(\rho)u, \quad (2.9)$$

where ρ is a n_ρ dimensional scheduling variable that belongs to the compact set

$$\Gamma = [\underline{\rho}_1, \bar{\rho}_1] \times [\underline{\rho}_2, \bar{\rho}_2] \times [\underline{\rho}_{n_\rho}, \bar{\rho}_{n_\rho}], \quad (2.10)$$

where $\underline{\rho}_i$ and $\bar{\rho}_i$ are the lower and upper bounds of the i -th element of the scheduling vector. The quadratic cost function of the LQR problem remains

$$J = \int_0^\infty (x^\top Qx + u^\top Ru) dt, \quad (2.11)$$

and the main objective of the control algorithm is to achieve asymptotic stability of the origin by applying full-state feedback in the form

$$u = K(\rho)x, \quad (2.12)$$

where the feedback matrix K can be dependent on the scheduling variable since it is assumed to be known. This results in the following closed-loop behavior:

$$\dot{x} = (A(\rho) + B(\rho)K(\rho))x. \quad (2.13)$$

Based on Lyapunov-theory, the asymptotic stability of the closed-loop system (2.13) can be verified by finding a Lyapunov function $V : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ that satisfies the following conditions:

$$1 : V(x) = 0 \iff x = 0, \quad (2.14a)$$

$$2 : V(x) \succ 0, \quad (2.14b)$$

$$3 : \dot{V}(x) \prec 0, \quad (2.14c)$$

for all x trajectories that satisfy (2.13). In this paper, we restrict ourselves to parameter-independent quadratic Lyapunov functions in the form of

$$V(x) = x^\top Px, \quad (2.15)$$

where $P \in \mathbb{R}^{n_x \times n_x}$ is the parameter matrix of V . Then, conditions (2.14a) and (2.14b) are satisfied if $P \succ 0$. Based on [22], the performance index from the cost (2.11) is substituted into (2.14c), which leads to

$$\dot{V}(x) < -(x^\top Qx + u^\top Ru) = -x^\top (Q + K^\top(\rho)RK(\rho))x. \quad (2.16)$$

Integrating (2.16) results in

$$V(x(\infty)) - V(x(0)) < -\int_0^\infty (x^\top Qx + u^\top Ru) dt, \quad (2.17)$$

from which if asymptotic stability holds, then $V(x(\infty)) = 0$ and an upper bound can be found for the LQR cost in the form of

$$\int_0^\infty (x^\top Qx + u^\top Ru) dt < V(0) = x(0)^\top Px. \quad (2.18)$$

Therefore, by minimizing this upper bound, a suboptimal solution for the LQR problem can be obtained. The expression $x(0)^\top Px(0)$ can be minimized for any arbitrary $x(0)$ by minimizing the trace of P , which leads to the following optimization problem:

$$\min_K \quad \text{trace}(P), \quad (2.19a)$$

$$\text{subject to} \quad P \succ 0, \quad (2.19b)$$

$$(2.16). \quad (2.19c)$$

However, (2.16) is a nonlinear inequality constraint that also contains x , so (2.19) cannot be solved directly. Therefore, (2.16) is converted to an LMI with the following steps. First, the derivative of the Lyapunov-function is substituted into (2.16) to obtain

$$x^\top (A(\rho) + B(\rho)K(\rho))^\top Px + x^\top P(A(\rho) + B(\rho)K(\rho))x < -x^\top (Q + K^\top(\rho)RK(\rho))x. \quad (2.20)$$

Multiplying (2.20) with $x^\top$ from the left and x from the right side renders a nonlinear matrix inequality as

$$(A(\rho) + B(\rho)K(\rho))^\top P + P(A(\rho) + B(\rho)K(\rho)) \prec -(Q + K^\top(\rho)RK(\rho)), \quad (2.21)$$

which is then multiplied with $X = P^{-1}$ from the left and the right side. After $Y(\rho) = K(\rho)X$ substitution, the resulting matrix inequality is

$$(A(\rho)X + B(\rho)Y(\rho))^\top + (A(\rho)X + B(\rho)Y(\rho)) \prec -(XQX + Y^\top(\rho)RY(\rho)), \quad (2.22)$$

which can be reorganized as

$$-(A(\rho)X + B(\rho)Y(\rho))^\top - (A(\rho)X + B(\rho)Y(\rho)) - (\mathcal{Q}X + \mathcal{R}Y(\rho))^\top (\mathcal{Q}X + \mathcal{R}Y(\rho)) \succ 0, \quad (2.23)$$

where the newly introduced $\mathcal{Q}$ and $\mathcal{R}$ matrices are defined as

$$\mathcal{Q} = \begin{bmatrix} Q^{\frac{1}{2}} \\ 0 \end{bmatrix}, \quad \mathcal{R} = \begin{bmatrix} 0 \\ R^{\frac{1}{2}} \end{bmatrix}. \quad (2.24)$$

By applying the Schur complement [23], (2.23) can be transformed into an LMI, namely

$$\begin{bmatrix} -(A(\rho)X + B(\rho)Y(\rho))^\top - (A(\rho)X + B(\rho)Y(\rho)) & (\mathcal{Q}X + \mathcal{R}Y(\rho))^\top \\ (\mathcal{Q}X + \mathcal{R}Y(\rho)) & I \end{bmatrix} \succ 0. \quad (2.25)$$

After the derivation of the LMI, an n -th order polynomial parametrization can be chosen for $Y(\rho)$ as

$$\begin{aligned} Y(\rho) = & Y_{1,0} + \rho_1 Y_{1,1} + \rho_1^2 Y_{1,2} + \cdots + \rho_1^n Y_{1,n} + \\ & Y_{2,0} + \rho_2 Y_{2,1} + \rho_2^2 Y_{2,2} + \cdots + \rho_2^n Y_{2,n} + \\ & \vdots \\ & Y_{n_\rho,0} + \rho_{n_\rho} Y_{n_\rho,1} + \rho_{n_\rho}^2 Y_{n_\rho,2} + \cdots + \rho_{n_\rho}^n Y_{n_\rho,n}, \end{aligned} \quad (2.26)$$

which is linear in the optimization variables $Y_{i,j}, i = 1, \dots, n_\rho, j = 0, \dots, n$. Lastly, after the parametrization, a discrete grid can be sampled from the scheduling space defined as

$$\mathbb{G} \in \Gamma, \quad (2.27)$$

where $\mathbb{G}$ is obtained such that it sufficiently covers Γ . Using this grid-based approach, the infinite set of LMIs in Γ can be relaxed to the finite set of matrix inequalities as

$$\begin{bmatrix} -(A(\rho)X + B(\rho)Y(\rho))^\top - (A(\rho)X + B(\rho)Y(\rho)) & (\mathcal{Q}X + \mathcal{R}Y(\rho))^\top \\ (\mathcal{Q}X + \mathcal{R}Y(\rho)) & I \end{bmatrix} \succ 0, \forall \rho \in \mathbb{G}. \quad (2.28)$$

Finally, with the LMI constraints (2.28), the SDP optimization can be formulated as

$$\max_K \quad \text{trace}(X), \quad (2.29a)$$

$$\text{subject to} \quad X \succ 0, \quad (2.29b)$$

$$(2.28). \quad (2.29c)$$

This optimization problem is formulated with CVXPY [24, 25], an open source modeling language for convex optimization in Python, and solved by MOSEK¹, a large-scale commercial optimization solver capable of handling SDPs.

In the end, the suboptimal feedback matrix can be calculated from the solution of (2.29) as

$$K(\rho) = Y(\rho)X^{-1}. \quad (2.30)$$

2.3 Stability and performance analysis of nonlinear systems

One of the most important questions in control theory is the closed-loop stability of controlled systems. Although the theory for the analysis of LTI is well-developed, nonlinear systems are still often analyzed through approximations to apply classical methods of the linear control theory. This section gives an overview of the stability and performance metrics used in this paper. First, the stability criterion is outlined. Then the $\mathcal{L}_2$ gain of input-output systems is outlined, which can serve as a quantitative measure of performance for nonlinear systems.

2.3.1 Stability of nonlinear systems

First, we briefly overview existing methods that can be utilized for analytic stability analysis.

A method used successfully in previous works is robust control theory, where nonlinearities are treated as disturbances and robust linear controllers are synthesized that can guarantee the stability of the original plants [26]. Other than that, there are also methods that can

¹<https://www.mosek.com>

only be utilized for systems with a predefined structure. Feedback linearization [27] is one of these techniques, most popularly applied for the control of robotic arms. However, an important condition for feedback linearization is that the system needs to have a relative degree equal to the number of states. Furthermore, it is sensitive to modeling uncertainties and external disturbances. This is a major limitation that reduces the applicability of this method.

Another method is the utilization of the LPV framework [16] for the analysis. Nonlinear systems can be embedded into LPV representations with different techniques (e.g. [17, 28, 29]) and the stability properties can be analyzed. However, performing these embedding methods often leads to more conservative representations of the original nonlinear systems, and formal verification of stability might not exist even if the original systems are stable. A relatively recent development in this field is universal shifted and incremental stability analysis techniques [30] developed specifically for avoiding the pitfalls of the LPV framework. It utilizes the preexisting tools of the LPV techniques and extends the stability conditions to reduce the conservativeness in the case of reference-tracking.

Lastly, we introduce the stability analysis concept based on Lyapunov-theory [31], which will also be used in the remainder of this work. Finding Lyapunov functions that satisfy a set of predefined conditions can be a powerful tool that is able to verify the stability of nonlinear systems. However, constructing such functions is often cumbersome, as analytical methods are hard to apply for systems with higher dimensions.

2.3.2 Lyapunov-functions

Consider an autonomous nonlinear system in state-space representation as

$$\dot{x} = f(x), \quad (2.31)$$

where, we assume w.l.o.g. that $x = 0$ is an equilibrium of the system. If the system has a nonzero equilibrium state, we can apply the following state transformation

$$\tilde{x} = x - x_{\text{eq}}, \quad (2.32)$$

where x_{eq} is an arbitrary equilibrium defined as $0 = f(x_{\text{eq}})$, and $\tilde{x}$ is often referred to as the error state. It is clear that for the error state, $\tilde{x} = 0$ defines an equilibrium point. The *domain of attraction* (DOA) associated with the equilibrium point x_{eq} is the set all initial states from which the trajectories of (2.31) converge to the equilibrium.

Let $\mathcal{X}$ be a closed subset of $\mathbb{R}^{n_x}$ that contains the origin of (2.31) in its interior. Moreover, let $\underline{\beta}, \beta, \bar{\beta} : [0, b) \rightarrow [0, \infty)$ be strictly increasing functions with $\underline{\beta}(0) = \beta(0) = \bar{\beta}(0) = 0$ and $b \in \mathbb{R}^+$ such that $\|x\| \leq b \forall x \in \mathcal{X}$. If there exists a continuously differentiable function $V : \mathcal{X} \rightarrow \mathbb{R}$ such that

$$\underline{\beta}(\|x\|) \leq V(x) \leq \bar{\beta}(\|x\|) \quad \forall x \in \mathcal{X}, \quad (2.33a)$$

$$\dot{V}(x) \leq -\beta(\|x\|) \quad \forall x \in \mathcal{X}, \quad (2.33b)$$

then V is called a Lyapunov function and system (2.31) is asymptotically stable on $\mathcal{X}$ [27]. Furthermore, the sublevel sets of the Lyapunov function defined as $\mathcal{X}_\alpha = \{x | V(x) \leq \alpha\}$, with $\alpha \in \mathbb{R}$ give an invariant subset of the DOA if $\mathcal{X}_\alpha \subset \mathcal{X}$.

However, unfortunately, there is no systematic procedure for finding Lyapunov functions and the corresponding domain of attractions for general nonlinear systems [32].

2.3.3 $\mathcal{L}_2$ -gain calculation of input-output systems

The $\mathcal{L}_2$ gain serves as a generalized measurement of energy transfer between the input and output of dynamical systems, motivated by the fundamentals of energy transfer within physical systems. It has been studied excessively for both linear and nonlinear systems to quantify performance measures which can be applied for stability and performance analysis and control design [31]. Next, we briefly present the formal definition of the $\mathcal{L}_2$ gain and its computation, based on [32].

First, we introduce the notion of dissipative dynamical systems. Consider an arbitrary nonlinear input-output system, defined as

$$\left. \begin{aligned} \dot{x} &= f(x, u) \\ z &= g(x, u) \end{aligned} \right\} \Sigma, \quad (2.34)$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, $u \in \mathbb{R}^{n_u}$ is the input vector and $z \in \mathbb{R}^{n_z}$ is the output vector, while $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathcal{X}$, and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_z}$ are nonlinear functions. System Σ is said to be dissipative with supply rate $s : \mathbb{R}^{n_u} \times \mathbb{R}^{n_z} \rightarrow \mathbb{R}$ if there exists a non-negative function $V : \mathcal{X} \rightarrow \mathbb{R}$ such that

$$V(x(t_0)) + \int_{t_0}^{t_1} s(u(t), z(t)) dt \geq V(x(t_1)), \quad (2.35)$$

for all $t_0 \leq t_1$ and all trajectories (x, z, u) which satisfy Σ . Next, consider a quadratic supply function in the form of

$$s(u, z) = \gamma^2 u^\top u - z^\top z, \quad (2.36)$$

where $\gamma \geq 0$. If the system Σ is dissipative with respect to this supply function, then

$$\dot{V}(x) \leq \gamma^2 u^\top u - z^\top z \quad (2.37)$$

holds along the solutions (x, z, u) . Furthermore, if u is taken from the set $\mathcal{L}_2$ of square integrable functions, i.e.

$$\|u\|_2^2 = \int_0^\infty u^\top(t) u(t) dt < \infty, \quad (2.38)$$

the state x and the output y are also square integrable functions and $\lim_{t \rightarrow \infty} x(t) = 0$. Therefore if (2.37) is integrated from $t = 0$ to ∞ , while (2.35) and (2.36) hold, then

$$\gamma^2 \|u\|_2^2 - \|z\|_2^2 \geq 0, \quad (2.39)$$

also holds, which can be reorganized as

$$\sup_{u \in \mathcal{L}_2} \frac{\|z\|_2}{\|u\|_2} \leq \gamma. \quad (2.40)$$

In (2.40), the left-hand side is the so-called $\mathcal{L}_2$ gain of the system Σ . As a consequence, the $\mathcal{L}_2$ gain is equivalent with the smallest $\gamma > 0$ for which (2.40) holds.

2.4 Gaussian Processes

Gaussian Processes (GPs) have been a popular choice for model augmentation [11, 33], as their nonparametric structure offers flexible utilization [34]. Furthermore, unlike other learning approaches such as *artificial neural networks* (ANNs), they also provide an uncertainty qualification next to the estimation, which can be employed for designing robust control algorithms [35]. Next, we briefly introduce GPs and sparse GPs.

2.4.1 Gaussian Process regression

To estimate a scalar unknown function $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ with a GP, let $X_N = [x_1^\top \cdots x_N^\top]^\top \in \mathbb{R}^{N \times n_x}$ denote the N number of inputs and $Y_N = [y_1 \cdots y_N]^\top \in \mathbb{R}^N$ the corresponding noisy output measurements of the data set $\mathcal{D}_N = \{X_N, Y_N\}$, generated by the following model:

$$y_i = f(x_i) + \epsilon_i, \quad i = 1, \dots, N, \quad (2.41)$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is an i.i.d. Gaussian noise. The core idea of GP based estimation of f is to consider that candidate estimates g belong to a GP, seen as a prior distribution. Then, using $\mathcal{D}_N$ and this prior, a predictive GP distribution of g is computed that provides an estimate of f in terms of its mean and describes the uncertainty of this estimate by its variance.

In terms of definition a scalar-valued *Gaussian Process* $\mathcal{GP} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ assigns to every point $x \in \mathbb{R}^{n_x}$ a random variable $\mathcal{GP}(x)$, such that, for any finite set $\{x_i\}_{i=1}^N \in \mathbb{R}^{n_x}$, the joint probability distribution of $\mathcal{GP}(x_1), \dots, \mathcal{GP}(x_N) \in \mathbb{R}$ is Gaussian. Due to this property, $g \sim \mathcal{GP}(m, \kappa)$ is fully determined by its mean m and covariance κ expressed as

$$m(x) = \mathbb{E}[g(x)], \quad (2.42a)$$

$$\kappa(x, \tilde{x}) = \mathbb{E}[g(x) - m(x)(g(\tilde{x}) - m(\tilde{x}))], \quad (2.42b)$$

This distribution describes our prior belief of the function space in which estimate of the the unknown function f is searched for. We assume that the prior mean is zero ($m(x) = 0$) and characterize covariance of the distribution with a *squared exponential* (SE) kernel, as it is a common choice for the estimation of smooth function:

$$\kappa_{\text{SE}}(x, \tilde{x}) = \sigma_f^2 \exp\left(-\frac{1}{2}(x - \tilde{x})^\top \Lambda^{-1}(x - \tilde{x})\right), \quad (2.43)$$

where $\sigma_f \in \mathbb{R}$ is a scaling factor and $\Lambda \in \mathbb{R}^{n_x \times n_x}$ is a positive definite and symmetric matrix that determines the smoothness of the candidate functions. Together σ_f , Λ and σ_ϵ in (2.41), are the so-called hyperparameters of the GP that can be tuned by maximizing the marginal likelihood computed from observed data [34].

Based on $\mathcal{D}_N$ and the prior $g \sim \mathcal{GP}(m, \kappa)$, the predictive distribution for $g(x_\star)$ at a test point $x_\star$ is the posterior $p(g(x_\star) | \mathcal{D}_N, x_\star) = \mathcal{N}(\mu(x_\star), \Sigma(x_\star))$ characterised by

$$\mu(x_\star) = K_N^\top(x_\star)(K_{NN} + \sigma_\epsilon^2 I)^{-1} Y, \quad (2.44a)$$

$$\Sigma(x_\star) = \kappa(x_\star, x_\star) - K_N^\top(x_\star)(K_{NN} + \sigma_\epsilon^2 I)^{-1} K_N(x_\star), \quad (2.44b)$$

where $[K_N(x_\star)]_i = \kappa(x_i, x_\star)$ and $[K_{NN}]_i = \kappa(x_i, x_i)$. Eq. (2.44a) characterizes the mean as the approximation of function f and (2.44b) is the variance, which gives the uncertainty of the approximation. Although the computation of (2.44) only requires elementary matrix multiplications, every training point is required for making predictions. Therefore in the case of large training datasets, the real-time implementation of GPs can be challenging.

2.5 Sparse Gaussian Process regression

One way to tackle the computational limitation of the traditional GP regression is to use *Sparse Gaussian Processes* (SGPs), where the number of data points used in the evaluation is limited to a fixed number of points.

The goal of the SGP is to find a set of pseudo inputs and outputs defined as $\mathcal{D}_M = \{X_M, Y_M\}$, where $X_M = [\hat{x}_1^\top, \dots, \hat{x}_M^\top]$ and $Y_M = [\hat{y}_1^\top, \dots, \hat{y}_M^\top]$ with $M \ll N$ that sufficiently describe $\mathcal{D}_N$.

The core philosophy of the SGP is to select the set $\mathcal{D}_M$ in a manner that minimizes the KL divergence between the GP trained on it and the GP trained on the initial $\mathcal{D}_N$ set. To find the proper $\mathcal{D}_M$ set, we utilize the variational free energy framework [36], in which a variational method is used that jointly selects the inducing inputs and the hyperparameters of the GP by maximizing a lower bound to the marginalized likelihood of the prediction. As a result, the pseudo inputs are fully determined by the hyperparameters, hence they are not part of the optimization. The full derivation can be found in [36].

After the training, the resulting posterior distribution after the training at an arbitrary test point $x_\star$ is $\mathcal{N}(\mu_p(x_\star), \Sigma_p(x_\star))$, where

$$\mu_p(x_\star) = K_M^\top(x_\star) K_{MM}^{-1} m_p, \quad (2.45a)$$

$$\begin{aligned} \Sigma_p(x_\star) = & \kappa(x_\star, x_\star) \\ & + K_M^\top(x_\star) (K_M^{-1} - K_{MM}^{-1} \mathcal{K}_p K_{MM}^{-1}) K_M(x_\star), \end{aligned} \quad (2.45b)$$

corresponding to a Nyström projection of the original GP to the pseudo inputs X_M , and where $[K_{MN}]_{i,j} = \kappa(\hat{x}_i, x_j)$, $i \in \mathbb{I}_1^M$, $j \in \mathbb{I}_1^N$ is the covariance matrix between the pseudo inputs and all training inputs, $[K_{MM}]_{i,j}$, $i, j \in \mathbb{I}_1^M$ is the covariance of the pseudo inputs

and $[K_M(x_*)]_i = \kappa(x_i, x_*)$. Furthermore, m_p and $\mathcal{K}_p$ are the mean and covariance of the approximation of the true posterior distribution, calculated as

$$m_p = \sigma_\epsilon^{-2} \mathcal{K}_p K_{MM}^{-1} K_{MN} Y \quad (2.46a)$$

$$\mathcal{K}_p = K_{MM} (K_{MM} + \sigma_\epsilon^{-2} K_{MN} K_{NM})^{-1} K_{MM}. \quad (2.46b)$$

This formulation reduces the $\mathcal{O}(N^3)$ computational complexity of the traditional GP to $\mathcal{O}(NM^2)$. Furthermore, it can also be recursively trained with high efficiency [37].

In the following, SGPs will be utilized as the learning component of the proposed trajectory tracking framework, which can efficiently adapt to modeling uncertainties and changing environmental conditions.

3 Vehicle model

This section gives an overview of the dynamic vehicle model utilized in this work. First, we introduce the first principle model. Then we decouple the nonlinear dynamics into two subsystems corresponding longitudinal and lateral motions. Finally, each subsystem is augmented with a GP based learning component to estimate the modeling uncertainties and the effect of changing environmental conditions on the dynamics.

3.1 First principle model

We model the trajectory-tracking vehicle with a dynamic single-track representation, as it has proven to be an optimal choice for describing the behavior of small-scale car-like vehicles [38, 39]. The model is depicted in Figure 3.1, and can be described with the following equations:

$$\dot{x} = v_\xi \cos(\varphi) - v_\eta \sin(\varphi), \quad (3.1a)$$

$$\dot{y} = v_\xi \sin(\varphi) + v_\eta \cos(\varphi), \quad (3.1b)$$

$$\dot{\varphi} = \omega, \quad (3.1c)$$

$$\dot{v}_\xi = \frac{1}{m} (F_\xi + F_\xi \cos(\delta) - F_{f,\eta} \sin(\delta) + mv_\eta \omega), \quad (3.1d)$$

$$\dot{v}_\eta = \frac{1}{m} (F_{r,\eta} + F_\xi \sin(\delta) + F_{f,\eta} \cos(\delta) - mv_\xi \omega), \quad (3.1e)$$

$$\dot{\omega} = \frac{1}{I_z} (F_{f,\eta} l_f \cos(\delta) + F_{f,\xi} l_f \sin(\delta) - F_{r,\eta} l_r), \quad (3.1f)$$

where $[x, y]$ is the position and φ is the orientation of the vehicle in the global coordinate frame. The states v_ξ and v_η denote the longitudinal and lateral velocity of the vehicle in a body fixed frame and ω is the yaw rate. Furthermore, the parameters of the model are the distance of the front and rear axis from the center of mass, denoted as l_f and l_r , respectively, the mass of the vehicle m and the inertia along the vertical axis I_z .

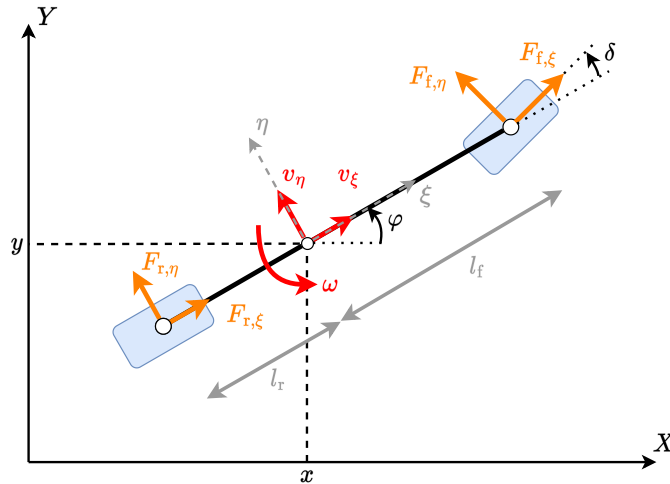


Figure 3.1: Dynamic single track vehicle model.

The longitudinal tire force F_ξ is determined by the following drivetrain model:

$$F_\xi = C_{m1}d - C_{m2}v_\xi - C_{m3}, \quad (3.2)$$

where C_{m1} , C_{m2} , C_{m3} are empirical drivetrain parameters and $d \in [-1, 1]$ is the motor input. This drivetrain assumes a first-order connection between a velocity and the motor input which can sufficiently describe the characteristics of DC and BLDC motors usually used in these vehicles. Lastly, the lateral tire forces are often calculated using a linear tire model as

$$F_{r,\eta} = C_r \arctan\left(\frac{-v_\eta + l_r\omega}{v_\xi}\right), \quad (3.3a)$$

$$F_{f,\eta} = C_f \arctan\left(\delta - \frac{v_\eta + l_f\omega}{v_\xi}\right), \quad (3.3b)$$

where C_f and C_r are cornering stiffness values of the front and rear tire respectively. Finally, the control inputs of the vehicle are the steering angle δ and the motor input d which a controller can directly actuate. The physical and empirical parameters of this first principle model can be determined by performing a set of identification experiments [38].

For the control design, it is convenient to transform the vehicle into a moving coordinate frame that is connected to the reference trajectory [38, 40]. The resulting model in the so-called curvilinear frame:

$$\dot{s} = (v_\xi \cos(\theta_e) - v_\eta \sin(\theta_e)) / (1 - c_c(s)e_s), \quad (3.4a)$$

$$\dot{e}_s = v_\xi \sin(\theta_e) + v_\eta \cos(\theta_e), \quad (3.4b)$$

$$\dot{\theta}_e = \omega - c_c(s)\dot{s}, \quad (3.4c)$$

where the newly introduced states are the position along the path s , the lateral deviation e_s and the heading error θ_e , while the lateral (v_η) longitudinal (v_ξ) and angular velocities (ω) are the same as in (3.1). Furthermore, $c_c(s)$ describes the curvature of the reference path at s . A big advantage of this first principle model is that all the states can be easily determined from physical measurements. Therefore, in the control design, we assume all states are available.

3.2 Decoupling

After the introduction of the full nonlinear model, the next step is to decouple the model into independent longitudinal and lateral subsystems for the model augmentation and the control design. This decoupling is beneficial as we can separate the complex nonlinear model into two subsystems corresponding to the longitudinal and lateral motion of the vehicle. Then for the subsystems, control algorithms can be synthesized individually, which results in a simpler but efficient structure.

From (3.4), the **longitudinal model** becomes

$$\dot{s} = \frac{v_\xi \cos(\theta_e) - v_\eta \sin(\theta_e)}{1 - c_c(s)e_s}, \quad (3.5a)$$

$$\begin{aligned} \dot{v}_\xi = \frac{1}{m} & \left((1 + \cos(\delta))(C_{m1}d - C_{m2}v_\xi - \text{sign}(v_\xi)C_{m3}) \right. \\ & \left. - C_f \left(\delta - \frac{v_\eta + l_f \omega}{v_\xi} \right) \sin(\delta) + mv_\eta \omega \right), \end{aligned} \quad (3.5b)$$

where the state vector $x_{\text{long}} = [s \ v_\xi]^\top$ describes the position along the path and the longitudinal velocity. Furthermore, θ_e heading error, e_s lateral deviation, v_η lateral velocity and δ steering angle are considered as varying parameters depending on the lateral subsystem. The reference input of the system is $x_{\text{long}}^{\text{ref}} = [s^{\text{ref}} \ v^{\text{ref}}]$, defined by the trajectory to track. To simplify the control design and analysis we separate the position and velocity states by introducing a virtual velocity generator that provides a position-adjusted virtual reference velocity denoted as v^r . As a result, the control objective is modified to achieve $v_\xi \rightarrow v^r$, and the system dimensions can be reduced as only the dynamics of v_ξ needs to be considered. Therefore, the system can be expressed as:

$$\begin{aligned} \underbrace{\dot{v}_\xi}_{\dot{\chi}_{\text{long}}} &= \underbrace{\frac{-C_{m2}(1 + \cos(\delta))}{m}}_{A_{\text{long}}(\delta)} \underbrace{v_\xi}_{\chi_{\text{long}}} + \underbrace{\frac{C_{m1}(1 + \cos(\delta))}{m}}_{B_{\text{long}}(\delta)} \underbrace{d}_{u_{\text{long}}} \\ &+ \underbrace{\frac{-C_{m3}(1 + \cos(\delta))}{m}}_{B_0(\delta)} \underbrace{\text{sign}(v_\xi)}_{w_0} + \underbrace{\frac{F_{f,\eta} \sin(\delta)}{m}}_{w_1} + v_\eta \omega \end{aligned} \quad (3.6)$$

where $A_{\text{long}}(\delta)$ and $B_{\text{long}}(\delta)$ are the parameter-varying state-transition and input matrices with scheduling variable δ , resulting in a linear parameter-varying (LPV) [17] representation. Furthermore, w_0 is a nonlinearity introduced by the dry friction of the drivetrain and w_1 is the effect of the lateral subsystem considered as disturbance.

Next, we consider the **lateral model**. The lateral behavior of the vehicle is described as

$$\dot{e}_s = v_\xi \sin(\theta_e) + v_\eta \cos(\theta_e), \quad (3.7a)$$

$$\dot{\theta}_e = \omega - c_c(s)\dot{s}, \quad (3.7b)$$

$$\dot{v}_\eta = \frac{1}{m} (F_{r,\eta} + F_\xi \sin(\delta) + F_{f,\eta} \cos(\delta) - mv_\xi \omega), \quad (3.7c)$$

$$\dot{\omega} = \frac{1}{I_z} (F_{f,\eta} l_f \cos(\delta) + F_{f,\xi} l_f \sin(\delta) - F_{r,\eta} l_r), \quad (3.7d)$$

where the state vector $x_{\text{lat}} = [e_s \ \theta_e \ v_\eta \ \omega]^\top$ consist of e_s lateral error, θ_e heading error, v_η lateral velocity and ω yaw rate, while v_ξ is considered a scheduling variable. To derive the LPV representation, we substitute the tire model into (3.7). Then, we use small angle approximations ($\sin(\alpha) \approx \alpha$, $\cos(\alpha) \approx 1$), neglect the longitudinal tire force ($F_\xi \approx 0$) and

approximate the path derived velocity as $\dot{s} \approx v_\xi$ which leads to the following model [41]:

$$\begin{bmatrix} \dot{e}_s \\ \dot{v}_\eta \\ \dot{\theta}_e \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 & v_\xi & 0 \\ 0 & -\frac{C_f+C_r}{mv_\xi} & 0 & -v_\xi - \frac{l_f C_f - l_r C_r}{mv_\xi} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{l_r C_r - l_f C_f}{I_z v_\xi} & 0 & -\frac{l_f^2 C_f + l_r^2 C_r}{I_z v_\xi} \end{bmatrix} \begin{bmatrix} e_s \\ v_\eta \\ \theta_e \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{C_f}{m} \\ 0 \\ \frac{l_f C_f}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ 0 \\ v_\xi \\ 0 \end{bmatrix} c_c(s), \quad (3.8)$$

where the path curvature is regarded as an external disturbance. Further simplification can be achieved by expressing the lateral model only in terms of the error variables and their derivatives. Let the second derivative of the (already linearized) lateral error be expressed as $\ddot{e}_s = \dot{v}_\eta + v_\xi \dot{\theta}_e$. Then, as proposed in [5], the error dynamics have the following form:

$$\begin{bmatrix} \dot{e}_s \\ \ddot{e}_s \\ \dot{\theta}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{C_f+C_r}{mv_\xi} & \frac{C_f+C_r}{m} & -\frac{l_f C_f + l_r C_r}{mv_\xi} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{l_r C_r - l_f C_f}{I_z v_\xi} & \frac{l_f C_f - l_r C_r}{I_z} & -\frac{l_f^2 C_f + l_r^2 C_r}{I_z v_\xi} \end{bmatrix} \begin{bmatrix} e_s \\ \dot{e}_s \\ \theta_e \\ \dot{\theta}_e \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{C_f}{m} \\ 0 \\ \frac{l_f C_f}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ \frac{l_r C_r - l_f C_f}{m} - 1 \\ 0 \\ -\frac{l_f^2 C_f + l_r^2 C_r}{I_z v_\xi} \end{bmatrix} c_c(s). \quad (3.9)$$

As discussed in [42], regulating both the heading and the lateral error to the origin results in poor tracking performance, as the two quantities cannot be simultaneously zero along the path if we assume perfect tracking. Therefore, we separate the lateral and the heading dynamics for the control design. We use the lateral error dynamics for feedback control design, and only regulate the heading error with a feedforward, as proposed in the so-called Stanley controller [3]. Furthermore, we also introduce a new state as the integral of the lateral error, i.e. $q = \int_0^t e_s dt$ to assure asymptotic convergence of e_s without offset. The final lateral model can be expressed as

$$\underbrace{\begin{bmatrix} \dot{q} \\ \dot{e}_s \\ \ddot{e}_s \end{bmatrix}}_{\chi_{\text{lat}}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\frac{C_f+C_r}{mv_\xi} \end{bmatrix}}_{A_{\text{lat}}(v_\xi)} \underbrace{\begin{bmatrix} q \\ e_s \\ \dot{e}_s \end{bmatrix}}_{\chi_{\text{lat}}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \frac{C_f}{m} \end{bmatrix}}_{B_{\text{lat}}} \delta + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \frac{l_r C_r - l_f C_f}{m} - 1 \end{bmatrix}}_{B_c(v_\xi)} \underbrace{c_c(s)}_{w_c} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{C_r+C_f}{m} & -\frac{l_f^2 C_f + l_r^2 C_r}{I_z v_\xi} \end{bmatrix}}_{B_2(v_\xi)} \underbrace{\begin{bmatrix} \theta_e \\ \dot{\theta}_e \end{bmatrix}}_{w_2}, \quad (3.10)$$

where the state vector $\chi_{\text{lat}} = [q \ e_s \ \dot{e}_s]^\top$ now only contains the lateral error its integral and derivative, and $A_{\text{lat}}(v_\xi)$ is the parameter varying state-transition matrix with scheduling signal v_ξ . Moreover, w_2 is used to lump together the unmodeled heading dynamics.

3.3 GP augmentation

After the nonlinear dynamics have been successfully decoupled, we can already design control algorithms for trajectory tracking as proposed in [38]. However, in order to make the control algorithms more robust to modeling uncertainties and external disturbances, we propose a learning based model augmentation method to capture and compensate for the model mismatch.

The model augmentation is done by additive, state and scheduling parameter dependent terms in the subsystems (3.6), (3.10). These terms are estimated by GPs and can be expressed as

$$\dot{\chi}_{\text{lat}} = A(v_{\xi})\chi_{\text{lat}} + B_{\text{lat}}\delta + B_c w_c + B_{\mathcal{GP}}\mathcal{GP}_{\text{lat}}(z_{\text{lat}}), \quad (3.11a)$$

$$\dot{\chi}_{\text{long}} = A(\delta)\chi_{\text{long}} + B(\delta)d + B_0(\delta)w_0 + \mathcal{GP}_{\text{long}}(z_{\text{long}}), \quad (3.11b)$$

where $\mathcal{GP}_{\text{long}}(z_{\text{long}})$ and $\mathcal{GP}_{\text{lat}}(z_{\text{lat}})$ denote the GPs linked to the lateral and longitudinal subsystems, respectively.

Based on the structure of the nominal path following model (3.4), we can observe that the first three equations only capture kinematic relationships. Therefore, we assume that modeling uncertainties only affect the velocity states v_{ξ} , v_{η} , ω , as proposed in [33]. As a result, the decoupling of the dynamics makes it possible to capture all uncertainties of the system with only two GPs in each subsystem, because the longitudinal model is one dimensional and $B_{\mathcal{GP}} = [0 \ 0 \ 1]^{\top}$ based on the previous assumption. Another important aspect of the GPs is the variables on which they are dependent on. Observing the original vehicle model (3.1), we can note that any change in the environmental conditions has an effect on the dynamics through the acting wheel forces. As these models depend on the velocity states (v_{ξ} , v_{η} , ω) and the inputs (d , δ) we choose these variables to construct the GPs. Furthermore, in the lateral case, the path curvature c_c is also added as a variable as it also has a significant effect on the lateral dynamics. Therefore the GP inputs are $z_{\text{long}} = [v_{\xi} \ v_{\eta} \ \omega \ \delta]^{\top}$ and $z_{\text{lat}} = [v_{\xi} \ v_{\eta} \ \omega \ d \ c_c]^{\top}$.

As we previously assumed that all the vehicle states are available, training inputs can be collected from the logged measurement data of driving experiments with the vehicle. Furthermore, the output of the training dataset is obtained by expressing $\mathcal{GP}_{\text{long}}(z_{\text{long}})$ and $\mathcal{GP}_{\text{lat}}(z_{\text{lat}})$ from (3.11a) and (3.11b), respectively.

Because of the large number of training points in the dataset, the utilization of SGPs is necessary to reduce the computation complexity of the evaluation. The resulting posterior distributions at arbitrary test points can be characterized by a normal distribution $\mathcal{N}(\mu(z^*), \Sigma(z^*))$, where the mean $\mu(z^*)$ provides an estimate of the additive term and the covariance $\Sigma(z^*)$ describes the uncertainty of the approximation. In both subsystems, $\mu(z^*)$ is compensated by a feedforward, while $\Sigma(z^*)$ can be utilized for fine-tuning GP by collecting additional training data in regions where the uncertainty is high [43].

4 Trajectory tracking control

This section introduces the adaptive control law, designed based on the decoupled dynamics augmented with the GPs. First, the trajectory tracking scenario and the parameterization of the reference trajectories are outlined. Then, after proposing the control architecture, we outline the design process of the control algorithms.

4.1 Trajectory tracking scenario

We aim to develop an adaptive trajectory tracking algorithm for autonomous car-like mobile robots that can ensure precise tracking in the presence of modeling uncertainties and changing environmental conditions. The path of the desired trajectory is expressed as a two-dimensional spline curve $\psi(s^{\text{ref}}(t))$, defined by coordinate functions $(x(s^{\text{ref}}(t)), y(s^{\text{ref}}(t)))$, where s^{ref} is a time domain signal defined as $s^{\text{ref}} : \mathbb{R} \rightarrow [0, L]$. The arc length of the full path is denoted as L , hence $s^{\text{ref}}(t)$ describes the desired vehicle position along the path at t . Both $x(s^{\text{ref}}(t))$, and $y(s^{\text{ref}}(t))$ are monotonic in $s^{\text{ref}}(t)$, furthermore $(x(0), y(0))$ and $(x(L), y(L))$ assign the endpoints of the curve. The speed profile $v^{\text{ref}}(s^{\text{ref}}(t)) = v^{\text{ref}}(t)$ along the trajectory is also given. These types of reference motion trajectories can be obtained by regular path planning algorithms.

4.2 Control Architecture

Based on the decoupled, GP augmented subsystems (3.11), two independent controllers are designed to achieve precise tracking of the reference trajectory. The longitudinal controller actuates the motor input d and is responsible for tracking the speed profile v^r , which is provided by a virtual velocity reference generator based on v^{ref} and the position error along the path. The lateral controller is used for tracking ψ by controlling the steering angle δ .

Using the nominal models in LPV form and the prediction of the GPs, we propose the following control architecture: we compensate the additive terms learned by the GPs by eliminating the predictive mean with a feedforward term. Then, based on the remaining nominal model structure in LPV form, we synthesize feedforward-feedback controllers to track the given reference. The overall control structure is depicted in Figure 4.1.

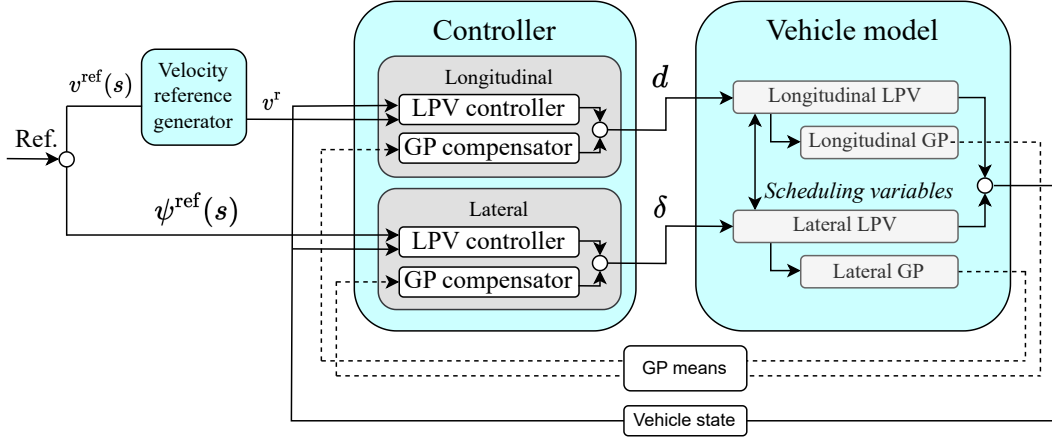


Figure 4.1: Proposed control architecture for trajectory tracking.

4.3 Control design

Based on the GP augmented subsystems we propose a feedforward-feedback control algorithm that stabilizes each subsystem and provides accurate tracking of the reference trajectories.

First, we construct the **longitudinal** control law as

$$d = d_{\text{ff}} + d_{\text{fb}}, \quad (4.1)$$

where the subscripts ff and fb denote the feedforward and feedback terms, respectively. The feedforward term can be expressed as

$$d_{\text{ff}} = B_{\text{long}}^{\dagger}(\delta)A_{\text{long}}(\delta)v^r + B_{\text{long}}^{\dagger}(\delta)B_{w_0}(\delta)w_0 - B_{\text{long}}^{\dagger}(\delta)\mu_{\text{long}}(z_{\text{long}}^*) \quad (4.2)$$

where $\dagger$ denotes the pseudoinverse. The term $B_{\text{long}}^{\dagger}(\delta)A_{\text{long}}(\delta)v^r$ provides the steady-state reference, while $B_{\text{long}}^{\dagger}(\delta)B_{w_0}(\delta)w_0$ is used to cancel out the dry friction of the drivetrain. Finally, with the GP augmentation proposed in Section 3.3, we can introduce the term $B_{\text{long}}^{\dagger}(\delta)\mu_{\text{long}}(z_{\text{long}}^*)$, which utilizes the predictive mean $\mu_{\text{long}}(z_{\text{long}})$. Applying the feedforwards to the subsystem, the remaining nominal LPV system is stabilized by a parameter varying state feedback law as

$$d_{\text{fb}} = K_{\text{long}}(\delta)(v_{\xi} - v^r), \quad (4.3)$$

where K_{long} feedback matrix can be synthesized by the LPV-LQR optimization proposed in Section 2.2.2, with scheduling parameters δ . Finally, the virtual velocity generator is constructed as

$$v^r = v^{\text{ref}} - k_v(s - s^{\text{ref}}), \quad (4.4)$$

where $k_v \in \mathbb{R}$ is the feedback gain of the position error. With this solution, we can achieve the tracking of the reference position, by generating so-called virtual reference velocities for the longitudinal system.

Next, we design the control law for the **lateral** subsystem, using the same feedforward-feedback approach:

$$\delta = \delta_{\text{ff}} + \delta_{\text{fb}}. \quad (4.5)$$

The proposed feedforward can be expressed as

$$\delta_{\text{ff}} = -\theta_e - B_{\text{lat}}^\dagger B_c w_c - B_{\text{lat}}^\dagger B_{\mathcal{GP}} \mu_{\text{lat}}(z_{\text{lat}}^*), \quad (4.6)$$

where $B_{\text{lat}}^\dagger B_c w_c$ and $B_{\text{lat}}^\dagger \mu_{\text{lat}}(z_{\text{lat}}^*)$ are used to cancel out the effect of the path curvature and the uncertainties estimated by the GP. Lastly, we introduce θ_e as a direct feedforward of the heading error, which is first proposed in the Stanley controller [3]. Similarly to the longitudinal controller, the full-state feedback law used for stabilization is based on the nominal system and can be expressed as

$$\delta_{\text{fb}} = K_{\text{lat}}(v_\xi) \chi_{\text{lat}} \quad (4.7)$$

constructed by the LPV-LQR synthesis with scheduling parameter v_ξ .

Collectively, we can summarize the control laws as

$$\delta = K_{\text{lat}}(v_\xi) \chi_{\text{lat}} - \theta_e - B_{\text{lat}}^\dagger B_c w_c - B_{\text{lat}}^\dagger B_{\mathcal{GP}} \mu_{\text{lat}}(z_{\text{long}}^*), \quad (4.8a)$$

$$d = K_{\text{long}}(\delta)(v_\xi - v^r) + B_{\text{long}}^\dagger(\delta) A_{\text{long}}(\delta) v^r + B_{\text{long}}^\dagger(\delta) B_{w_0}(\delta) w_0 - B_{\text{long}}^\dagger(\delta) \mu_{\text{long}}(z_{\text{long}}^*). \quad (4.8b)$$

Furthermore, using the covariance of the GP, we can also calculate a performance index for the controller. The uncertainty qualification of the GPs can be interpreted as white noise acting on the LPV system through input matrix $B_{\mathcal{GP}}$, characterized by covariance σ . Then as derived in [21], we can calculate the "worst-case" performance of the system as

$$\sigma = \text{trace}(B_{\mathcal{GP}} \bar{\Sigma} B_{\mathcal{GP}}^\top X^{-1}), \quad (4.9)$$

where $\bar{\Sigma}$ is the maximal covariance matrix of the corresponding GP. The resulting σ is then can serve as a quantifiable performance measure.

5 Simulation study

In this section, the proposed learning based adaptive control approach is evaluated. For this study, we consider the model of a 1/10 scale autonomous electric F1TENTH vehicle [14]. Before presenting the results, we outline the construction of the reference trajectories, which will be used throughout this section. Then, we first analyze the control performance in numerical simulations using the dynamic vehicle model introduced in Section 3, with model parameters identified in [38]. After the positive results of the numerical simulations, for a more precise analysis, we conduct experiments with MuJoCo [15], a high-fidelity physical simulator, where the digital-twin model of a real F1TENTH vehicle has been implemented.

5.1 Reference trajectories and error measures

The reference trajectories are constructed in a three-step process. First, we fit two dimensional 4th order Spline curves onto a set of predefined waypoints. As a result, we obtain $\tilde{\psi}(x(p), y(p))$ coordinate functions with $p \in [0, 1]$. However, the tracking algorithms require arc-length parametrization, as proposed in Section 4.1. Therefore in the next step, we numerically calculate the arc-length of the path, then reparameterize the splines of the coordinate functions to obtain $\psi(x(s^{\text{ref}}(t)), y(s^{\text{ref}}(t)))$, where $s^{\text{ref}} : \mathbb{R} \rightarrow [0, L]$ is a time domain function that describes the desired reference position along the path at t , with L denoting the total arc-length of the path. In the third step, we design $s^{\text{ref}}(t)$, by constructing a speed profile along the path, which is described as $v^{\text{ref}}(s^{\text{ref}}(t))$. The trajectory generation and parameter reparametrization have been implemented in Python, using the Scipy library [44].

From the trajectory definition, the tracking errors are calculated as follows. Based on the vehicle position (x, y) and orientation φ , we calculate the projection of the center of mass of the vehicle as

$$s_{\star} = \arg \min_s \left\{ ((x - x(s))^2 + (y - y(s))^2) \right\}. \quad (5.1)$$

With $s_{\star}$, we calculate the derivatives of the coordinate functions to obtain $x'(s_{\star})$, $y'(s_{\star})$, $x''(s_{\star})$ and $y''(s_{\star})$. Then, we can acquire the s_1 and z_1 unit vectors:

$$s_1(s_{\star}) = \frac{1}{\sqrt{x'^2(s_{\star}) + y'^2(s_{\star})}} [x'(s_{\star}), y'(s_{\star})]^{\top}, \quad (5.2a)$$

$$z_1(s_{\star}) = \frac{1}{\sqrt{x'^2(s_{\star}) + y'^2(s_{\star})}} [-y'(s_{\star}), x'(s_{\star})]^{\top}, \quad (5.2b)$$

$$(5.2c)$$

from which the path heading and the path curvature are

$$\theta_P(s_\star) = \arctan 2(y'(s_\star), x'(s_\star)), \quad (5.3a)$$

$$c_c(s_\star) = \frac{x''(s_\star)y'(s_\star) - x'(s_\star)y''(s_\star)}{\sqrt{x'^2(s_\star) + y'^2(s_\star)}^3} \quad (5.3b)$$

Finally, the tracking errors are

$$\theta_e = \varphi - \theta_P(s), \quad (5.4a)$$

$$e_s = [x - x(s_\star), y - y(s_\star)] \cdot z_1, \quad (5.4b)$$

$$v_e = v_\xi - v^r, \quad (5.4c)$$

where $v^r = v^{\text{ref}} - k_v(s_\star - s^{\text{ref}})$ as proposed in the decoupling in Section 3.2. With all the tracking errors obtained, we now move on to the experiments.

5.2 Numerical simulations

This section outlines the numerical simulations performed to analyze the control performance of the proposed controller. First, we outline the model and its parameters, as well as the simulation method. It is then followed by executing a series of experiments to present and compare the obtained results.

5.2.1 Vehicle model and parameters

We use the nonlinear dynamic single-track model, which has been outlined in Section 3. The differential equations governing the motion reiterated for clarity:

$$\dot{x} = v_\xi \cos(\varphi) - v_\eta \sin(\varphi), \quad (5.5a)$$

$$\dot{y} = v_\xi \sin(\varphi) + v_\eta \cos(\varphi), \quad (5.5b)$$

$$\dot{\varphi} = \omega, \quad (5.5c)$$

$$\begin{aligned} \dot{v}_\eta = \frac{1}{m} \left(C_r \left(\frac{-v_\eta + l_r \omega}{v_\xi} \right) + (C_{m1}d - C_{m2}v_\xi - \text{sign}(v_\xi)C_{m3})\sin(\delta) \right. \\ \left. + C_f \left(\delta - \frac{v_\eta + l_f \omega}{v_\xi} \right) \cos(\delta) - mv_\xi \omega \right), \end{aligned} \quad (5.5d)$$

$$\begin{aligned} \dot{v}_\xi = \frac{1}{m} \left((1 + \cos(\delta))(C_{m1}d - C_{m2}v_\xi - \text{sign}(v_\xi)C_{m3}) \right. \\ \left. - C_f \left(\delta - \frac{v_\eta + l_f \omega}{v_\xi} \right) \sin(\delta) + mv_\eta \omega \right), \end{aligned} \quad (5.5e)$$

$$\begin{aligned} \dot{\omega} = \frac{1}{I_z} \left(C_f \left(\delta - \frac{v_\eta + l_f \omega}{v_\xi} \right) l_f \cos(\delta) + (C_{m1}d - C_{m2}v_\xi - \text{sign}(v_\xi)C_{m3})l_f \sin(\delta) \right. \\ \left. - C_r \left(\frac{-v_\eta + l_r \omega}{v_\xi} \right) l_r \right), \end{aligned} \quad (5.5f)$$

The physical and empirical parameters of the model are collected in Table 5.1.

Table 5.1: Model parameters

Notation	Value	Unit
l_r	0.168	m
l_f	0.163	m
m	2.923	kg
I_z	0.0796	kg · m ²
C_{m1}	41.796	N
C_{m2}	2.0152	Ns·m ⁻¹
C_{m3}	0.4328	N
C_f	29.4662	N · rad ⁻¹
C_r	41.7372	N · rad ⁻¹

Finally, with the error measures (5.4a) and the control laws (4.8) we can construct the closed loop for the simulations. To simulate the model, we have implemented (5.5a) in Python and have used the 4th order Runge-Kutta method as the integration formula, with the sampling time $T_s = 0.01$ s.

5.2.2 Trajectory tracking simulations

With the simulation environment introduced, we now present the experiments. To demonstrate the adaptability of the proposed control algorithm to changing environmental conditions, we compare three scenarios, where the vehicle is required to track the same predefined lemniscate trajectory. In the first scenario, we use the nominal model with the parameters in Table 5.1 and the nominal control algorithm without the adaptive terms in the control law. Then, in the second scenario, we modify the parameters of the model to emulate different changed environmental conditions. This is done by modifying the tire model (C_f , C_r) and drivetrain parameters (C_{m1} , C_{m2} , C_{m3}), as these values are the most sensitive to the changes in the environment (e.g. a more slippery road results in smaller cornering stiffness values).

In the first scenario, since LPV-LQR (2.29) synthesis does not require the GPs to be trained, we use the nominal LPV feedback controller with $\mu_{\text{long}} = \mu_{\text{lat}} = 0$. The parameter-dependent feedback matrices are obtained by solving (2.29), with predefined weighting matrices and scheduling parameter bounds. For the lateral controller, the weighting matrices are chosen as

$$Q_{\text{lat}} = \text{diag}(1, 80, 0), \quad R_{\text{lat}} = 500,$$

which in practice means that we want to eliminate the error the most, while we scale the input such that we avoid saturation. For the scheduling variable of the lateral subsystem, we use the range $v_\xi \in [0, 2.5]$ as we want to operate the vehicle in this range. In the case of the longitudinal system, the weights

$$Q_{\text{long}} = 1, \quad R_{\text{long}} = 100,$$

while the scheduling variable is $\delta \in [-0.5, 0.5]$. Lastly, the gain of the reference velocity generator is $k_s = 0.1$. In both cases, the gains are obtained by examining simulations with the nominal model. As shown in the comparison in Figure 5.2, the nominal controller performs well if the motion dynamics of the vehicle are known precisely.

Next, we consider the second scenario, where the motion dynamics of the vehicle are heavily modified. This can be the result of several factors. First, simple changes in the environment such as the material of the ground have an obvious impact on the tire forces and the drivetrain. However, it is also important to note that some physical parameters of the cars (i.e. the inertia) are hard to obtain precisely and therefore there is an inherent model mismatch between the real vehicle and the model. Therefore, in this scenario, we make the following modifications:

$$\begin{aligned} C_f &= 10, & C_r &= 12.5, \\ C_{m1} &= 31, & C_{m2} &= 3.2, & C_{m3} &= 4.25. \end{aligned}$$

In practice, these parameters emulate a more slippery ground surface and slower drivetrain dynamics which can for example originate from a different wheel size used on the car. As

shown in Figure 5.2, these modifications had a significant effect on the control performance resulting in larger lateral deviations and steady-state error in the longitudinal velocity. As the main purpose of this work is precise navigation even in the case of large changes in the environmental conditions, this renders the nominal controller unsatisfactory. Therefore, the utilization of the adaptive terms is necessary.

For the third scenario, in order to utilize the proposed adaptive algorithm, the GPs require training data to estimate the model mismatch. The training data is obtained in two steps. First, we execute an initial training trajectory with the vehicle, depicted in Figure 5.1a. Then using the collected data, we generate a grid between the minimal and maximal bounds of each variable of the training input and we evaluate the GPs at each sample of the grid points. Then, based on the covariance classification of the GP, we can fine-tune the GPs by collecting additional training data where the uncertainty of the GPs is high. This second, fine-tuning trajectory is displayed in Figure 5.1b. After the additional training data has been successfully obtained, we retain the GPs with this extended dataset, to make the predictions more robust. In both cases, the GPs are trained by the VFE framework of Section 2.5, with $M = 20$ inducing points. The regression and the hyperparameter optimization have been implemented in GPyTorch [45]. As the complete number of data points $N > 1500$, the utilization of the SGPs is essential for computational efficiency.

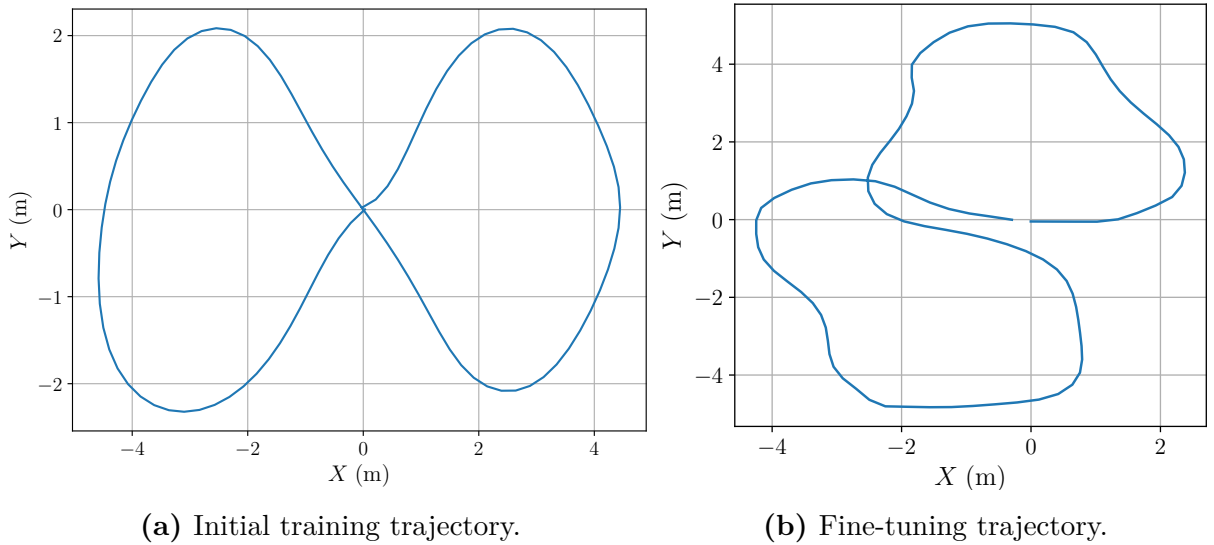


Figure 5.1: Training trajectories used for data collection.

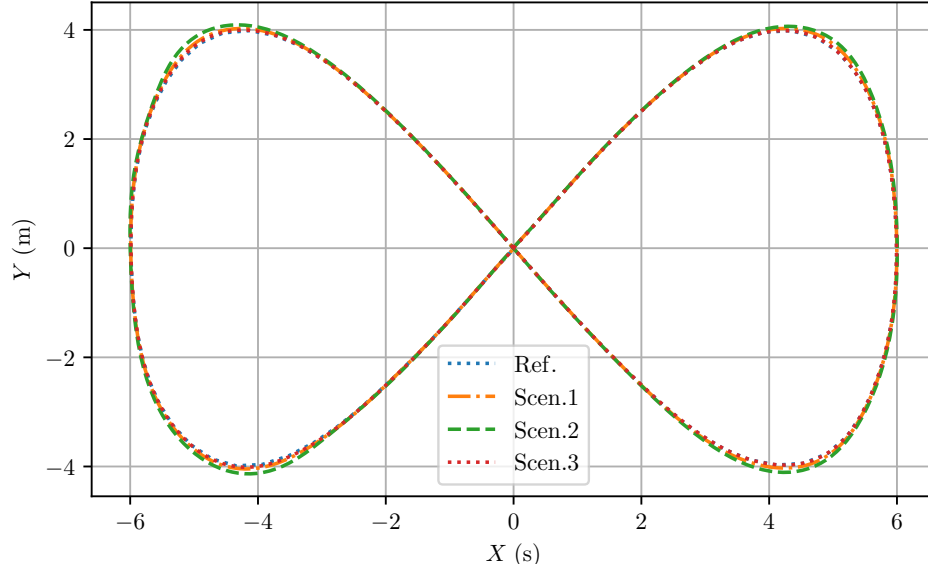
As shown in Table 5.2, this two-step approach is proven to be beneficial, as the uncertainties decreased after the second training, resulting in increased controller performance. Finally, the complete comparison of the three scenarios is displayed in Figure 5.2, while Table 5.3 collects the maximal error values. From the data, it is clear that the GP based learning components can successfully capture the model mismatch and the feedforward term of the controllers are able to effectively compensate for it.

Table 5.2: Comparison of controller performance and uncertainty qualifications.

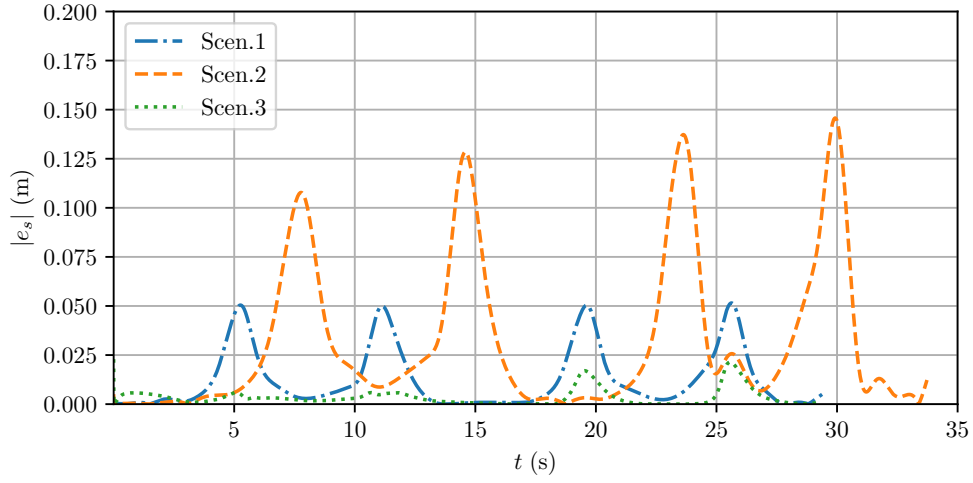
Training-trajectory	$\bar{\Sigma}_{\text{long}}$	$\bar{\Sigma}_{\text{lat}}$	σ_{long}	σ_{lat}
Initial	0.7672	1.0463	0.1206	18.7958
Fine-tuned	0.6258	0.9659	0.0984	17.3511

Table 5.3: Comparison between the maximal values of the tracking errors (*after the initial transient).

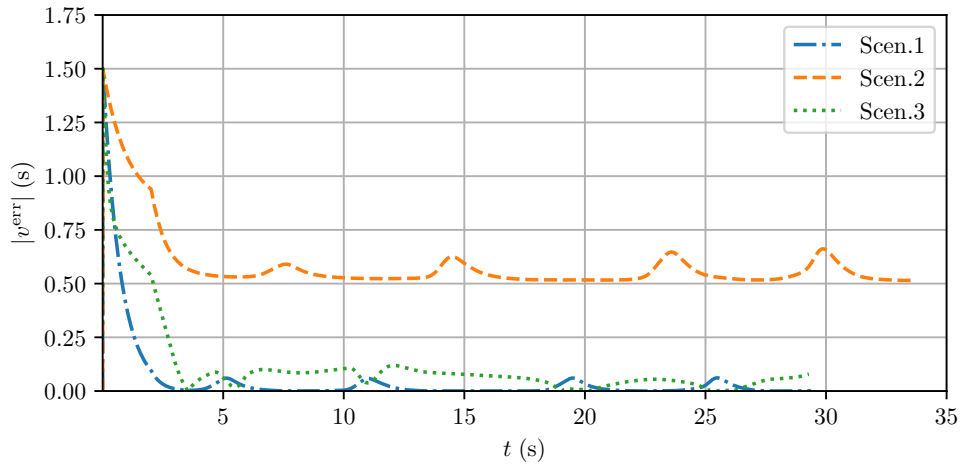
Scenario	$ e_s $	$ v^{\text{err}} ^{*}$
1	0.0608	0.1915
2	0.2109	0.3902
3	0.0625	0.0648



(a) Test trajectories.



(b) Lateral errors.



(c) Velocity errors.

Figure 5.2: Results of the numerical simulations.

5.3 MuJoCo simulations

After the numerical simulations, we further analyze the results on the digital twin model of a real F1TENTH vehicle implemented in the virtual test environment developed in MuJoCo [15].

5.3.1 Simulation environment

MuJoCo is an open-source physics engine developed by Google DeepMind. It is capable of simulating complex dynamical systems and friction effects and therefore is ideal for evaluating the control algorithms as the environmental conditions can be precisely altered. The core simulation engine is written in C++ which offers efficient and fast calculations. However, it also offers a high-level Python API which can be used to operate the main functionalities of the software. Therefore, in order to integrate well with the other components of this work, we have developed the simulation environment using this API.

The vehicle model has been constructed using the XML schema of the MuJoCo engine. The main parts of the vehicle, such as the wheels and the chassis have been modeled as rigid bodies with joints connecting them to each other. Furthermore, we have defined contact models between the wheels and the ground such that the model closely resembles the behavior of the real car. The final model is depicted in Figure 5.3.

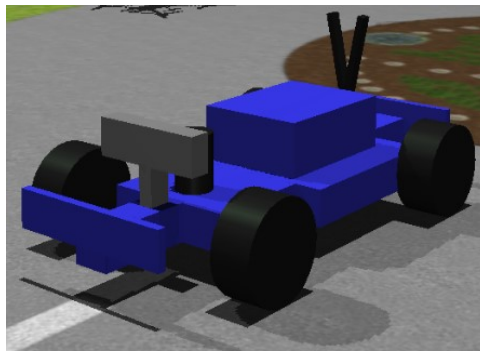


Figure 5.3: MuJoCo model of the F1TENTH vehicle.

5.3.2 Trajectory tracking simulations

Finally, in this section, we present the simulation results obtained from the MuJoCo environment. We perform the same series of experiments as proposed in Section 5.2.2. First, an initial test round with the nominal controller and car model is executed. Then, the model parameters are modified and the test round is performed to quantify the performance degradation of the controllers. Lastly, we collect data and train the GPs using the previously proposed two-step process to obtain the adaptive control law of the third scenario.

In the first simulation, we synthesize the nominal LPV-LQR controllers with the following gains:

$$\begin{aligned} Q_{\text{lat}} &= \text{diag}(0.1, 100, 0), & R_{\text{lat}} &= 500, \\ Q_{\text{long}} &= 1, & R_{\text{long}} &= 150, \end{aligned}$$

while the scheduling parameters remain in the same range as specified before: $v_\xi \in [0.5, 2]$, $\delta \in [-0.5, 0.5]$. These values are optimized by performing a series of experiments with the nominal model and controllers.

After the initial simulation with the nominal model, we modify the vehicle model in the MuJoCo environment as follows:

- We increase the radius of the wheels by 1.5 times its original size.
- We modify the contact forces between the wheel and the ground by halving the lateral and longitudinal friction coefficients to emulate more slippery roads.

Then, we analyze the performance of the nominal controller on this modified model. As Figure 5.4 shows, the tracking errors have increased, and precise tracking cannot be achieved without the adaptive terms.

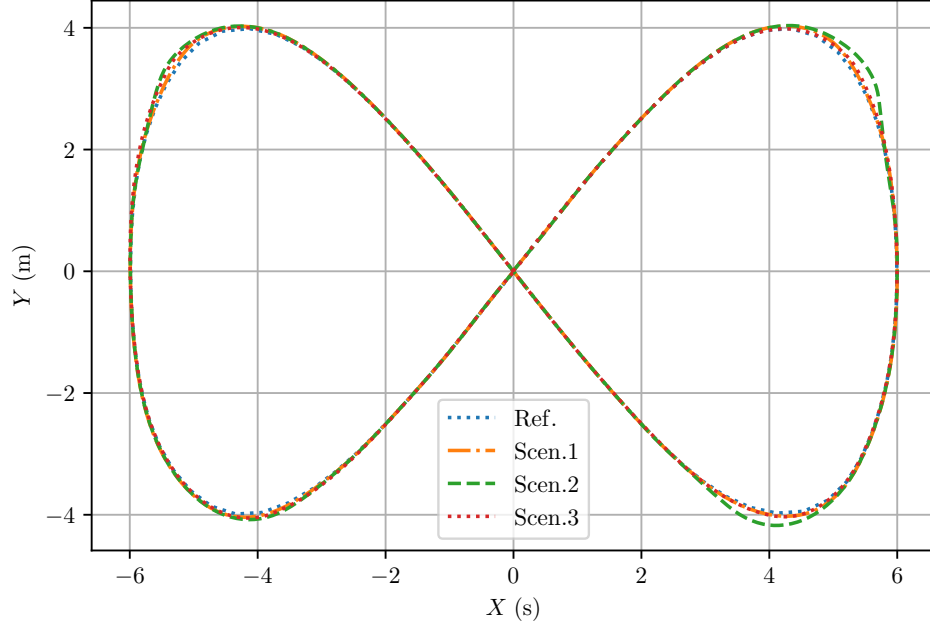
Therefore, we employ the adaptive terms of the control algorithms to compensate for the model mismatch. First, we collect measurement data by executing the training trajectories presented in Figure 5.1 of Section 5.2.2. As shown in Table 5.4, the two-step training approach is again proven to be beneficial, as the maximal covariance of the GPs decreases after the second round of training. The training method remains the same as proposed in the numerical simulation sections: we utilize the VFE framework with $M = 20$ inducing points. With the training completed, we execute the third simulation scenario to analyze the performance of the obtained adaptive controller. As Figure 5.4 and Table 5.5 show the adaptive control law can efficiently decrease the tracking errors and provide precise tracking of the reference trajectories.

Table 5.4: Comparison of controller performance and uncertainty qualifications.

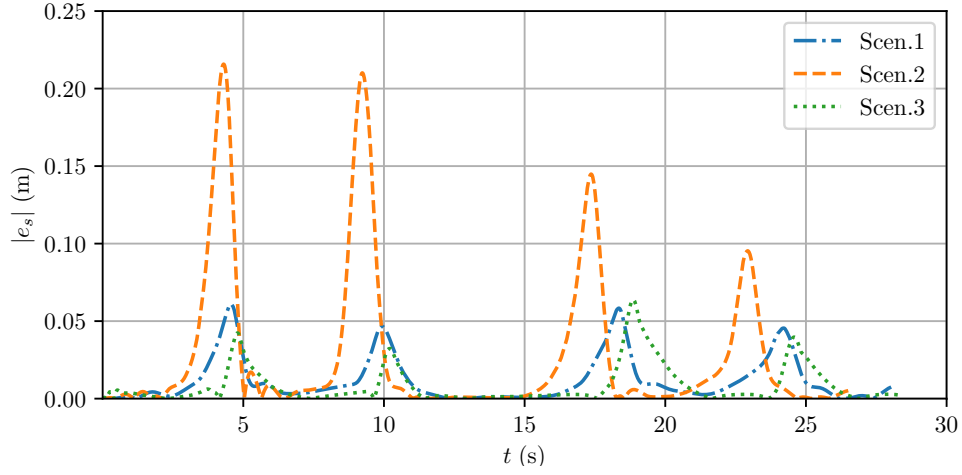
Training-trajectory	$\bar{\Sigma}_{\text{long}}$	$\bar{\Sigma}_{\text{lat}}$	σ_{long}	σ_{lat}
Initial	0.8497	2.0858	0.1336	33.5726
Fine-tuned	0.6178	1.7916	0.0971	31.7571

Table 5.5: Comparison between the maximal values of the tracking errors (*after the initial transient).

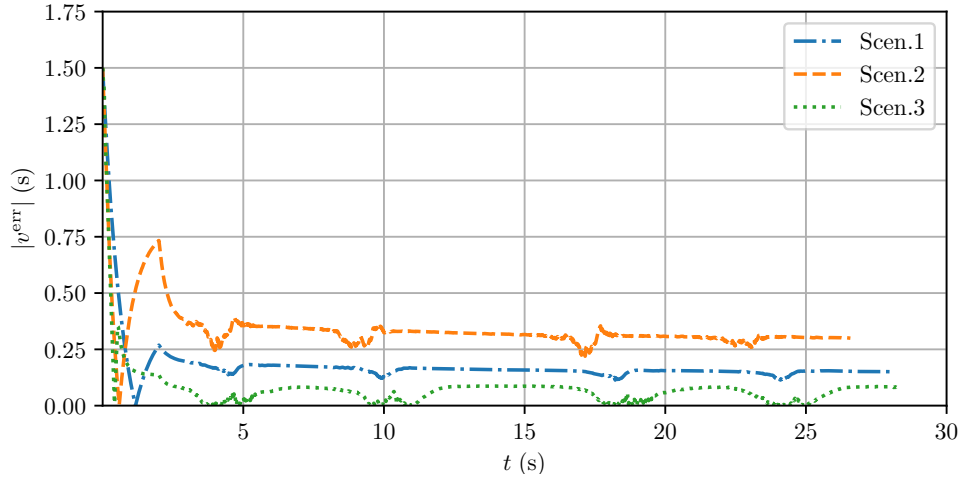
Scenario	$ e_s $	$ v^{\text{err}} ^*$
1	0.0515	0.0611
2	0.1459	0.6553
3	0.0208	0.1171



(a) Test trajectories.



(b) Lateral errors.



(c) Velocity errors.

Figure 5.4: Results of the numerical simulations.

6 Stability analysis

Finally, we conclude this paper with a precise mathematical analysis of the stability and performance properties of our proposed trajectory tracking solution. While each designed control algorithm can guarantee the stability of the used control-oriented model, it does not automatically imply the stability of the complete interconnected system. There are several reasons for this. First, the control algorithms are derived based on simplified models, where nonlinearities have been linearized and external effects have been neglected. Furthermore, the two subsystems that have been used for the control design are coupled due to varying parameters that depend on the states or inputs associated with the other model, which can also influence the stability of the system.

Therefore, in this section, the stability and the performance of the nonlinear closed-loop system are analyzed. During the stability analysis, we assume that the GP augmentation can efficiently estimate the model mismatch, therefore, we can compensate for it via the introduced feedforward. As a consequence, it is sufficient to analyze the nominal nonlinear vehicle model.

6.1 Closed-loop system

With the longitudinal and lateral controllers (4.8), and the path tracking model (3.4) we formulate the closed loop system as

$$\dot{q} = e_s, \quad (6.1a)$$

$$\dot{e}_s = v_\eta \cos(\theta_e) + v_\xi \sin(\theta_e), \quad (6.1b)$$

$$\dot{\theta}_e = \omega - c_c \frac{(v_\xi \cos(\theta_e) - v_\eta \sin(\theta_e))}{1 - c_c e_s}, \quad (6.1c)$$

$$\begin{aligned} \dot{v}_\eta = \frac{1}{m} \left(C_r \left(\frac{-v_\eta + l_r \omega}{v_\xi} \right) + (C_{m1}d - C_{m2}v_\xi - \text{sign}(v_\xi)C_{m3})\sin(\delta) \right. \\ \left. + C_f \left(\delta - \frac{v_\eta + l_f \omega}{v_\xi} \right) \cos(\delta) - m v_\xi \omega \right), \end{aligned} \quad (6.1d)$$

$$\begin{aligned} \dot{v}_\xi = \frac{1}{m} \left((1 + \cos(\delta))(C_{m1}d - C_{m2}v_\xi - \text{sign}(v_\xi)C_{m3}) \right. \\ \left. - C_f \left(\delta - \frac{v_\eta + l_f \omega}{v_\xi} \right) \sin(\delta) + m v_\eta \omega \right), \end{aligned} \quad (6.1e)$$

$$\begin{aligned} \dot{\omega} = \frac{1}{I_z} \left(C_f \left(\delta - \frac{v_\eta + l_f \omega}{v_\xi} \right) l_f \cos(\delta) + (C_{m1}d - C_{m2}v_\xi - \text{sign}(v_\xi)C_{m3})l_f \sin(\delta) \right. \\ \left. - C_r \left(\frac{-v_\eta + l_r \omega}{v_\xi} \right) l_r \right), \end{aligned} \quad (6.1f)$$

$$\delta = K_{\text{lat}}(v_\xi) [q \ e_s \ \dot{e}_s] - \theta_e - B_{\text{lat}}^\dagger B_c w_c, \quad (6.1g)$$

$$d = K_{\text{long}}(\delta)(v_\xi - v^r) + B_{\text{long}}^\dagger(\delta) A_{\text{long}}(\delta) v^r - B_{\text{long}}^\dagger B_{w_0}(\delta) w_0, \quad (6.1h)$$

which can also be written in a more compact form:

$$\dot{x} = f_{\text{cl}}(x, v^r, c_c), \quad (6.2)$$

where $x = [q \ e_s \ \theta_e \ v_\eta \ v_\xi \ \omega]^\top$ is the state vector, v^r is the reference velocity input and c_c is the varying curvature.

We show that for any constant $w \in \mathcal{W} = [c_c, \bar{c}_c] \times [v^r, \bar{v}^r]$ the vehicle will asymptotically converge to the reference trajectory determined by w . To verify such an asymptotic stability condition for each choice of w , we introduce the change of variables $\tilde{x} = x - x_{\text{eq}}(w)$ to turn the problem into the analysis of the equilibrium point $\tilde{x} = 0$ for the error state $\tilde{x}$, where the equilibrium is computed from $0 = f_{\text{cl}}(x_{\text{eq}}, w)$. The compact set of the error state-space in which asymptotic stability is analyzed is defined as $\tilde{\mathcal{X}} = [\underline{x}_1, \bar{x}_1] \times \cdots \times [\underline{x}_6, \bar{x}_6]$, where the lower and upper bar in the notation represents the minimal and maximal bounds, respectively. In practice if the proposed stability condition holds, we can guarantee that for any constant w , i.e. constant curvature and velocity reference the vehicle will asymptotically converge to this trajectory as depicted in Figure 6.1

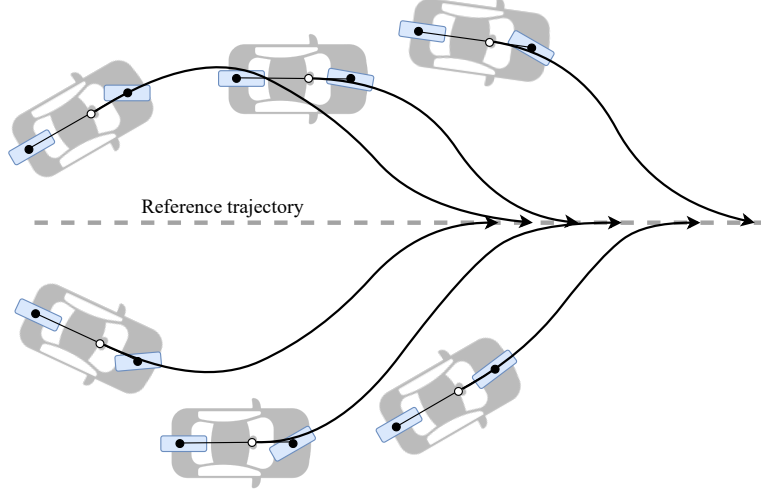


Figure 6.1: The stability concept investigated in this section.

6.2 Iterative stability analysis

We verify the stability of (6.2) by constructing a quadratic Lyapunov function as proposed in Section 2.3.2. To tackle the issue of finding Lyapunov functions in a systematic way, the following section proposes an iterative optimization based approach that can be used to prove the local stability of arbitrary nonlinear systems based on [13, 46].

The algorithm consists of two main parts: a learner and a verifier. First, the verifier solves a convex optimization problem to find a candidate function that satisfies the Lyapunov conditions (2.33) in a predefined operating region. Then, the verifier validates or rejects the candidate by trying to find counterexamples, where the conditions fail. If such counterexample exists, this information is incorporated into the learner and a new iteration starts.

First, we formulate the **learner** with which we verify asymptotic stability of (6.2), by constructing a Lyapunov function $V(\tilde{x}, w) : \tilde{\mathcal{X}} \times \mathcal{W} \rightarrow \mathbb{R}$ for (6.2). If for any fixed $w \in \mathcal{W}$, V is a positive definite and decrescent function in $\tilde{x}$ and its derivative along the trajectories of (6.2) is negative definite function on $\tilde{\mathcal{X}}$, then the origin is an asymptotically stable equilibrium of (6.2) under any $w \in \mathcal{W}$. By restricting ourselves to quadratic functions in the form of $V(\tilde{x}) = \tilde{x}^\top P \tilde{x}$, where $P \in \mathbb{R}^{6 \times 6}$, the Lyapunov conditions are satisfied if P is positive definite as a matrix, i.e., $P \succ 0$ and the following inequality holds for the derivative:

$$\underbrace{\tilde{f}_{\text{cl}}(\tilde{x}, w)^\top P \tilde{x} + \tilde{x}^\top P \tilde{f}_{\text{cl}}(\tilde{x}, w)}_{\dot{V}(\tilde{x}, w)} < -\epsilon \tilde{x}^\top \tilde{x}, \forall (\tilde{x}, w) \in \tilde{\mathcal{X}} \times \mathcal{W} \quad (6.3)$$

where $\tilde{f}_{\text{cl}}(\tilde{x}, w) = f_{\text{cl}}(\tilde{x} + x_{\text{eq}}(w), w)$. It is important to note that (6.3) is linear in P , which is an advantageous property that will be later exploited in the optimization. Furthermore, we assume that the compact region $\tilde{\mathcal{X}}$, where the stability is examined is symmetric. Then, we can normalize the states as $\chi = T\tilde{x}$ transformation, where $T = \text{diag}(\bar{x}_1, \dots, \bar{x}_{n_x})$. The

Lyapunov function of the normalized states is

$$V(\chi, w) = \chi^\top T P T \chi = \chi^\top \mathcal{P} \chi, \quad (6.4)$$

while the normalized state space is an n_x dimensional hypercube $\mathcal{C}^{n_x}$ after the transformation.

Next, we construct the optimization problem for the calculation of P . Essentially, our aim can be described as

$$\text{find } P \quad (6.5a)$$

$$\text{subject to } P \succ 0, \quad (6.5b)$$

$$\tilde{f}_{\text{cl}}^\top(\tilde{x}, w) P \tilde{x} + \tilde{x}^\top P \tilde{f}_{\text{cl}}(\tilde{x}, w) < 0, \quad \forall \tilde{x} \in \mathcal{X}. \quad (6.5c)$$

Unfortunately, (6.5) cannot be solved as (6.5c) indicates an infinite number of linear constraints. However, we can utilize a widely used technique from semidefinite optimization which is to introduce discrete grids $\mathbb{X} \in \tilde{\mathcal{X}}$ and $\mathbb{W} \in \mathcal{W}$ that relaxes (6.5) to a finite number of constraints. Note that it is important to sample $\mathbb{X}$ and $\mathbb{W}$ such that it sufficiently covers $\tilde{\mathcal{X}}$ and $\mathcal{W}$, which indicates a very large number of constraints as the number of grid points scales exponentially with the number of samples along each dimension of the state space. However, as the constraint is linear, a state-of-the-art numerical solver such as MOSEK is capable of handling it.

The other important limitation of (6.5) is the fact that by solving only a feasibility problem, we have no control over the shape and the values of P , which can cause numerical errors for the solver. Furthermore, another key property of stability is the invariant set. The invariant set can be defined as the largest subset of the Lyapunov function, which is contained in $\tilde{\mathcal{X}}$. If the initial value of the system is inside this set, then $\dot{V}(\tilde{x}) < 0$ condition guarantees that the state trajectories will never venture out of this set and will, in fact, converge to the origin.

To maximize invariant set, we denote the level set of (6.4) as $\mathcal{E} = \{\chi \mid \chi^\top \mathcal{P} \chi \leq 1\}$ and we also introduce the hypersphere $\mathcal{S} = \{x \mid x^\top \rho I x \leq 1\}$, where ρ^{-1} is the radius of $\mathcal{S}$. Then by prescribing $\mathcal{S} \in \mathcal{E}$, while minimizing ρ , we indirectly increase the invariant set. The $\mathcal{S} \in \mathcal{E}$ inclusion can be formulated as

$$\chi^\top \mathcal{P} \chi \leq \chi^\top \rho I \chi, \quad (6.6)$$

which can also be reformulated as an LMI:

$$\rho I - \mathcal{P} \succeq 0. \quad (6.7)$$

However, the level set also needs to remain in $\mathcal{C}^{n_x}$ which can be achieved by the following constraints [23]:

$$e_i^\top \mathcal{P} e_i \leq 1, \quad (6.8)$$

where $e_i = [0_{1 \times (i-1)}, 1, 0_{1 \times (n_x-i)}]^\top$.

Finally, the optimization problem of the learner can be formulated as

$$\min_{\rho, \mathcal{P}} \quad \rho, \quad (6.9a)$$

$$\text{s. t.} \quad \mathcal{P} \succ 0, \quad (6.9b)$$

$$e_i^\top \mathcal{P}^{-1} e_i \leq 1, \quad \forall i = 1, \dots, n_x, \quad (6.9c)$$

$$\rho I - \mathcal{P} \succeq 0, \quad (6.9d)$$

$$\tilde{f}_{\text{cl}}^\top(\tilde{x}, w) T^{-1} \mathcal{P} T^{-1} \tilde{x} + \tilde{x}^\top T^{-1} \mathcal{P} T^{-1} \tilde{f}_{\text{cl}}(\tilde{x}, w) < 0, \quad \forall \tilde{x} \in \tilde{\mathbb{X}}, w \in \mathbb{W}, \quad (6.9e)$$

which is an SDP with a total of $n_x + 2$ semidefinite constraints and n_x^N linear constraints, where N is the number of points sampled in each dimension of the state space. As the number of semidefinite constraints is small and only linear constraints scale exponentially, the optimal solvable for numerical solvers. Furthermore, in the case of large dimension systems, the *alternating direction method of multipliers* (ADMM) algorithm can also be used to decompose the optimization into subproblems. During this work the optimization is formulated with CVXPY [24, 25] and solved by MOSEK.

An important limitation of the learner is that it can only guarantee the Lyapunov conditions at the predefined discrete grid points of $\tilde{\mathcal{X}}$. However, there may exist an arbitrary $\tilde{x}_\star$ point in the state space where (6.3) is violated and asymptotic stability cannot be guaranteed. To address this issue, we introduce the **verifier**. The main objective of the verifier is to find $\tilde{x}_\star$ such that $\dot{V}(\tilde{x}_\star) \not\leq 0$. This can be achieved by finding the maximum of (6.3) with the following *nonlinear program* (NLP):

$$\max_{\tilde{x}, w} \quad \dot{V}(\tilde{x}, w)|_P := \tilde{f}^\top(\tilde{x}, w) P \tilde{x} + \tilde{x}^\top P \tilde{f}(\tilde{x}, w), \quad (6.10a)$$

$$\text{subject to} \quad \tilde{x} \in \tilde{\mathcal{X}}, \quad w \in \mathcal{W}, \quad (6.10b)$$

where P is the parameter of the Lyapunov function candidate obtained by (6.9). To solve this NLP, we utilized CasADi [47] paired with the Ipopt nonlinear solver.

If the objective function of the optimal solution is positive, (6.3) is violated and the candidate P is rejected. However, we can use the optimal state and disturbance values to add them to the discrete set $\mathbb{X}$ of (6.9). Then the learner can be re-executed with the augmented constraint set to provide a new candidate P that also complies with the previously obtained $\tilde{x}_\star$ and $w_\star$ optimal values. This iteration is then continued until (a) the feasibility problem in (6.9) becomes unfeasible, which means that a parameter-independent Lyapunov function cannot be found for the whole region; (b) the objective function of the NLP remains negative at the optimal solution, therefore the asymptotic stability of the system is proven. The full algorithm is outlined in Algorithm 1.

Algorithm 1 Iterative stability analysis

```
1: input:  $\mathbb{X}$  initial error state-space and  $\mathbb{W}$  generalized disturbance grids
2: loop
3:   solve (6.9) with  $\mathbb{X}$  and  $\mathbb{W}$  to obtain  $P$ 
4:   if (6.9) is unfeasible then
5:     return  $P = \emptyset$ 
6:   end if
7:   solve (6.10) with  $P$  to obtain  $\max_{\tilde{x}}(\dot{V}(\tilde{x}, w)|_P)$ 
8:   if  $\max(\dot{V}(\tilde{x}, w)|_P) \geq 0$  then
9:      $(\tilde{x}^*, w^*) = \operatorname{argmax}(\dot{V}(\tilde{x}, w)|_P)$ 
10:     $\mathbb{X} := \mathbb{X} \cup \{\tilde{x}^*\}$ ,  $\mathbb{W} := \mathbb{W} \cup \{w^*\}$ 
11:   else
12:     return  $P$ 
13:   end if
14: end loop
```

For the analysis, first, it is important to declare the lower and upper bound of the state space and the external reference. Based on measurements with the real vehicle, the following regions can be defined:

$$\begin{aligned} \tilde{x}_1 &\in [-0.2, 0.2], & \tilde{x}_2 &\in [-0.1, 0.1], \\ \tilde{x}_3 &\in [-0.5, 0.5], & \tilde{x}_4 &\in [-0.5, 0.5], \\ \tilde{x}_5 &\in [-0.3, 0.3], & \tilde{x}_6 &\in [-3.5, 3.5], \\ w_1 &\in [-1.44, 1.44], & w_2 &\in [0.5, 2], \end{aligned}$$

which defines $\tilde{\mathcal{X}}$ and $\mathcal{W}$. Initially, $\tilde{\mathcal{X}}$ and $\mathcal{W}$ are sampled equidistantly along each dimension at 7 points, with the exception of v^r as (6.2) is linear in it, therefore, its lower and upper bound is sufficient. After 13 iterations, we have obtained a valid Lyapunov function with

$$\mathcal{P} = \begin{bmatrix} 1.087 & 0.305 & 0.275 & 0.053 & 0.020 & 0.068 \\ 0.305 & 1.370 & 0.657 & 0.326 & 0.030 & 0.423 \\ 0.275 & 0.657 & 1.799 & 0.351 & -0.041 & 0.697 \\ 0.053 & 0.326 & 0.351 & 1.277 & 0.015 & -0.281 \\ 0.020 & 0.030 & -0.041 & 0.015 & 1.003 & 0.003 \\ 0.068 & 0.423 & 0.697 & -0.281 & 0.003 & 1.473 \end{bmatrix}, \quad (6.12)$$

that corresponds to the invariant set depicted in Figure 6.2, projected onto 2D planes. As asymptotic convergence to the origin can be guaranteed for any initial state that belongs to the invariant set, we can conclude that our controller is stable, which means that the controller is able to track the predefined reference trajectories effectively.

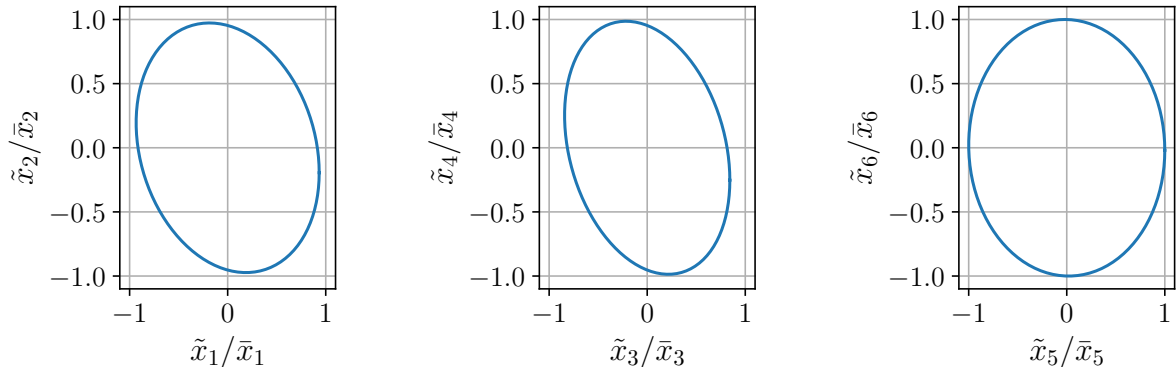


Figure 6.2: Invariant set of the Lyapunov function.

7 $\mathcal{L}_2$ gain analysis

In the previous section, the stability of the nonlinear system for arbitrary, but constant generalized disturbance. Therefore, this section further develops the analysis, by considering changes in the generalized disturbance and evaluating the tracking performance and disturbance rejection of the closed loop system. For this analysis we utilize $\mathcal{L}_2$ gain techniques introduced in Section 2.3.3.

First, we introduce the system model used in this section. Then, using numerical simulations, a lower bound for the induced $\mathcal{L}_2$ gain is estimated. Then, we give an iterative calculation method with the extension of the stability analysis algorithm proposed in Section 6.2 for the optimization-based computation of the upper bound of the induced $\mathcal{L}_2$ gain of the system between w generalized disturbance and the tracking errors.

7.1 System model

In (6.2), the nonlinear state space representation of the system has already been outlined. However, for the $\mathcal{L}_2$ gain computation, performance channels need to be defined. The performance outputs are calculated as

$$z = h(x, w), \quad (7.1)$$

where $z = [e_s \ v_\xi]^\top$. Therefore, the input-output system is defined as

$$\left. \begin{aligned} \dot{x} &= f_{\text{cl}}(x, w) \\ z &= Cx - Dw \end{aligned} \right\} \mathcal{S}, \quad (7.2)$$

where $x \in \mathcal{X}$ is the state vector, $w \in \mathcal{W}$ is the generalized disturbance vector. Note that for the analysis both the state and the disturbance input are centered such that $\tilde{v}_\xi = v_\xi - v_{\text{cent}}^r$ and $\tilde{v}^r = v^r - v_{\text{cent}}^r$ to achieve bounded and square-integrable w and $0 = f_{\text{cl}}(x, w)$ equilibrium. Furthermore, to characterize that v^r is only active in the low-frequency range, we augment system (7.2) with a first-order low-pass filter. As a result, the state space of the model is extended by one extra dimension of the filter and the disturbance can be eliminated from the output equation, i.e. $z = h(x, w = 0) = h(x)$.

7.2 Simulation analysis

Before calculating the induced $\mathcal{L}_2$ gain, simulation studies are carried out with system $\mathcal{S}$. These simulations can effectively illustrate the behavior of the system and can also give a lower bound estimate for the $\mathcal{L}_2$ gain.

During the simulation, various square-integrable functions were used as w to measure the z performance outputs. Subsequently, the squares of these signals were numerically integrated to calculate their quotient, which can be regarded as the lower bound for the

induced $\mathcal{L}_2$ gain. This measure can serve as a baseline when calculating the analytical solution in the latter sections of this work.

The maximal values of the signals have been characterized based on physical considerations as proposed in the previous sections, reiterated again for clarity. Therefore, the generalized disturbance is defined as $w \in \mathcal{W} = [-1.44, 1.44] \times [0.5, 2]$. The results of two simulations, where sinusoidal have been used as w , are displayed in Figure 7.1.

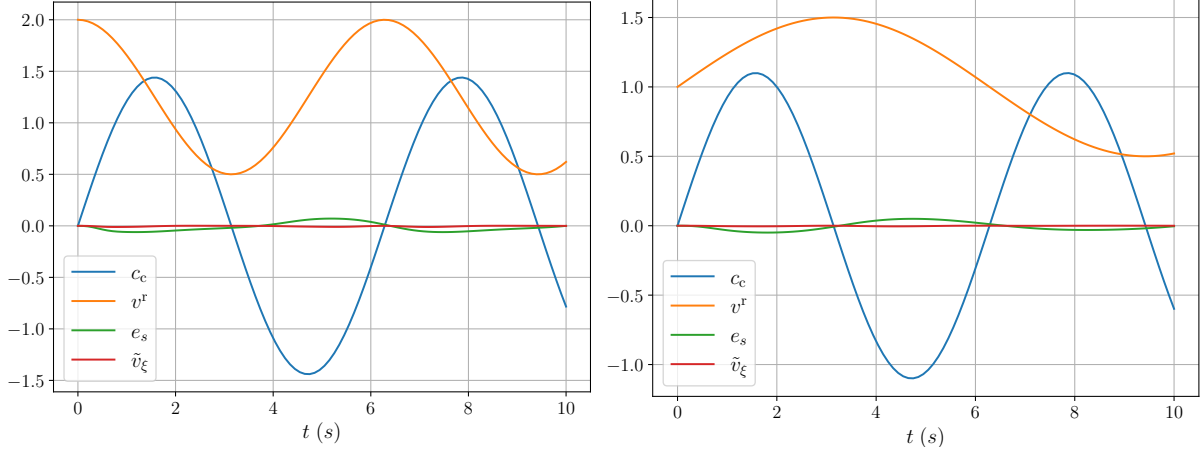


Figure 7.1: Numerical simulations for the lower estimation of the induced $\mathcal{L}_2$ gain.

After conducting the simulations, the lower bound estimation can be determined by selecting the maximum value among the aforementioned quotients:

$$\gamma_{\text{low}} = \max_{z, w} \frac{\|z\|_2}{\|w\|_2} = 0.0496. \quad (7.3)$$

where $\|\cdot\|_2$ denotes the finite time $\mathcal{L}_2$ norm.

7.3 Iterative $\mathcal{L}_2$ gain computation

After lower bound estimation by simulations, we propose an extension of the iterative method proposed in Algorithm 1 to utilize it for the calculation of the upper bound of the $\mathcal{L}_2$ gain of the system, based on the theory introduced in Section 2.3.3.

As stated in the theoretical introduction, the upper bound of $\mathcal{L}_2$ gain of a nonlinear input-output system can be estimated by the smallest γ for which the differential dissipation inequality (2.37) holds. Therefore, to formulate the optimization problem, the quadratic supply function and the state and output equations of system $\mathcal{S}$ from (7.2) are substituted back into the differential dissipation inequality to obtain

$$x^\top P f_{\text{cl}}(x, w) + f_{\text{cl}}^\top(x, w) P x \leq \gamma^2 w^\top w - (Cx - Dw)^\top (Cx - Dw). \quad (7.4)$$

Note that this equation is also linear in both P and γ^2 , therefore we can again construct a **learner** that can obtain a candidate P and γ . Formally, the learner solves the following

optimization

$$\min_{P, \gamma^2} \quad \gamma^2, \quad (7.5a)$$

$$\text{subject to} \quad P \succ 0, \quad (7.5b)$$

$$(7.4), \quad \forall x \in \mathbb{X}, \quad w \in \mathbb{W} \quad (7.5c)$$

where $\mathbb{X} \in \mathcal{X}$ and $\mathbb{W} \in \mathcal{W}$ are discrete sampled sets of the previously defined state and input space, respectively, resulting in the same grid-based computation approach proposed in Section 6.

After a candidate γ and P have successfully been obtained, the **verifier** validates them by maximizing a reorganized version of the differential dissipation inequality over the state and input space. The NLP of the verifier is formulated as

$$\max_{x, u} \quad J(x, u)|_{P, \gamma} := x^\top P f_{\text{cl}}(x, w) + f_{\text{cl}}^\top(x, w) P x - \gamma^2 w^\top w + h(x)^\top h(x), \quad (7.6a)$$

$$\text{s. t.} \quad x \in \mathcal{X}, \quad w \in \mathcal{W}. \quad (7.6b)$$

After solving the maximization, the value of the objective function is examined. If $\max(J(x, w)|_{P, \gamma}) > 0$ at the optimum, the candidate P and γ values are rejected and the corresponding $(x_\star, w_\star)$ values are added to the discrete sets $\mathbb{X}$, $\mathbb{W}$ respectively. Then, a new iteration starts with the learner using the extended $\mathbb{X}$ and $\mathbb{W}$ sets. The iterations are then repeated until (a) the optimization problem in (7.5) becomes unfeasible, which means that a finite $\mathcal{L}_2$ gain cannot be computed for the system with parameter independent quadratic storage functions; (b) the optimal value of the NLP objective function remains non-positive, and the $\mathcal{L}_2$ gain has been successfully computed. The whole iterative algorithm of the $\mathcal{L}_2$ gain computation is described in Algorithm 2.

Algorithm 2 Iterative $\mathcal{L}_2$ gain computation

- 1: **input:** $\mathbb{X}$ and $\mathbb{W}$ initial error state-space and generalized disturbance grids
 - 2: **loop**
 - 3: **solve** (7.5) with $\mathbb{X}$, $\mathbb{W}$ to obtain P , γ
 - 4: **if** (7.5) is unfeasible **then**
 - 5: **return** $\gamma = \emptyset$, $P = \emptyset$
 - 6: **end if**
 - 7: **solve** (7.6) with P , γ to obtain $\max(J(x, w)|_{P, \gamma})$
 - 8: **if** $\max(J(x, w)|_{P, \gamma})$ **then**
 - 9: $(x^\star, w^\star) = \text{argmax}(J(x, w)|_{P, \gamma})$
 - 10: $\mathbb{X} = \mathbb{X} \cup \{x^\star\}$, $\mathbb{W} = \mathbb{W} \cup \{w^\star\}$
 - 11: **else**
 - 12: **return** γ , P
 - 13: **end if**
 - 14: **end loop**
-

Like the stability analysis, both the learner and the verifier are implemented in Python, with the SDP formulated with CVXPY and the verifier in CasADi.

As the input of the algorithm, we use the previously declared $\mathbb{W}$ and $\mathbb{X}$ discrete sets, where each dimension is sampled equidistantly by 7 points initially. The algorithm required a total of 45 iterations to obtain the following upper bound of the induced $\mathcal{L}_2$ gain

$$\gamma_{\text{up}} = 0.104. \quad (7.7)$$

As Algorithm 2 has been able to find an upper bound for the induced $\mathcal{L}_2$ gain between the generalized disturbance and the tracking performance, we can prove that the tracking errors remain finite and therefore the closed loop system remains stable even for varying disturbance input. Furthermore, the $\mathcal{L}_2$ gain also acts as a performance indicator, therefore, we have also derived a quantitative indicator for the efficiency of the controller. As the induced $\mathcal{L}_2$ gain is proven to be small, we can conclude that the tracking performance is adequate.

It is important to remark that the exact value of the induced $\mathcal{L}_2$ gain is still unknown, as Sections 7.2 and 7.3 give the lower and upper estimates $\gamma_{\text{low}} \leq \mathcal{L}_2 \leq \gamma_{\text{up}}$. The conservativeness of the upper bound can also be reduced by the utilization of more complex, parameter dependent supply functions. However, in this specific case, the resulting upper bound has been deemed sufficient, and therefore increasing the computational complexity of the optimization of the learner is not worth the trade-off.

8 Conclusion and future work

In this paper, a learning based trajectory-tracking control algorithm with mathematically proven stability and performance guarantees has been proposed for small-scale car-like vehicle platforms, as an answer for increased demand in their general utilization.

After the introduction, we have first summarized the theoretical foundations of the LPV-LQR control design, nonlinear stability and performance analysis, as well as Gaussian Processes. It has been followed with the introduction of the vehicle model, which we have decoupled into separate subsystems corresponding to the longitudinal and lateral motion of the vehicle. Each subsystem has been augmented with a Gaussian Process based learning component to capture modeling uncertainties.

Based on the augmented model a computationally efficient feedforward-feedback controller has been proposed. The performance of the designed controller has been analyzed in numerical simulations as well as in a high-fidelity physical simulator. In both cases, our algorithm has proven to be effective in compensating for large modeling uncertainties and tracking the predefined reference trajectories.

After successful control design, the stability of the proposed control architecture has been thoroughly examined. First, the asymptotic stability of the nonlinear closed-loop system has been verified by a novel optimization based method that had constructed a quadratic Lyapunov function. Then to give quantitative measures for the tracking performance, the induced $\mathcal{L}_2$ gain of the generalized disturbance to the tracking errors has also been calculated. As the obtained $\mathcal{L}_2$ gain is not only finite but sufficiently small, we can provide stability guarantees even in the presence of nonconstant generalized disturbance and verify that the tracking performance of the controllers is adequate.

In the future, we plan to further expand the capabilities of the proposed adaptive trajectory-tracking method. Rapid identification of the model mismatch is planned to be implemented by recursively updating the sparse GPs online. Furthermore, based on the uncertainty qualification of the GPs, we aim to develop a systematic trajectory planning technique for training data collection to refine the estimation.

Acknowledgment

I would like to thank *Tamás Péni* and *Roland Tóth* at HUN-REN SZTAKI for their huge amount of help and professional support. I would also like to thank *István Harmati* from the Budapest University Technology and Economics for his supervision.

The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems. (RRF-2.3.1-21-2022-00002) and the ÚNKP-23-2-I-BME-171 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

References

- [1] Qiangqiang Yao et al. “Control Strategies on Path Tracking for Autonomous Vehicle: State of the Art and Future Challenges”. In: *IEEE Access* 8 (2020), pp. 161211–161222.
- [2] R. Craig Conlter. *Implementation of the Pursuit Path Tracking Algorithm*. Tech. rep. Pittsburgh, PA: Carnegie Mellon University, 1992.
- [3] Gabriel M. Hoffmann et al. “Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing”. In: *Proceedings of the 2007 American Control Conference*. 2007, pp. 2296–2301.
- [4] Riccardo Marino, Stefano Scalzi, and Mariana Netto. “Nested PID steering control for lane keeping in autonomous vehicles”. In: *Control Engineering Practice* 19.12 (2011), pp. 1459–1467.
- [5] Jarrod M Snider. *Automatic steering methods for autonomous automobile path tracking*. Tech. rep. Robotics Institute, Pittsburgh, PA, 2009.
- [6] Johannes Betz et al. “Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing”. In: *IEEE Open Journal of Intelligent Transportation Systems* 3 (2022), pp. 458–488.
- [7] Matthias Althoff, Markus Koschi, and Stefanie Manzing. “CommonRoad: Composable benchmarks for motion planning on roads”. In: *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 719–726.
- [8] Bernardo A. Hernandez Vicente, Sebastian S. James, and Sean R. Anderson. “Linear System Identification Versus Physical Modeling of Lateral–Longitudinal Vehicle Dynamics”. In: *IEEE Transactions on Control Systems Technology* 29.3 (2021), pp. 1380–1387.
- [9] Ugo Rosolia, Ashwin Carvalho, and Francesco Borrelli. “Autonomous racing using learning Model Predictive Control”. In: *Proceedings of the 2017 American Control Conference*. 2017, pp. 5115–5120.
- [10] Alexander Liniger, Alexander Domahidi, and Manfred Morari. “Optimization-based autonomous racing of 1:43 scale RC cars”. In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 628–647.
- [11] Juraj Kabzan et al. “Learning-Based Model Predictive Control for Autonomous Racing”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3363–3370.
- [12] Jonathan Becker et al. “Model- and Acceleration-based Pursuit Controller for High-Performance Autonomous Racing”. In: *Proceedings of the 2023 IEEE International Conference on Robotics and Automation*. 2023, pp. 5276–5283.
- [13] Shaoru Chen et al. “Learning Lyapunov Functions for Hybrid Systems”. In: *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. New York, NY, USA, 2021.

- [14] Matthew O’Kelly et al. “F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning”. In: *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*. Vol. 123. Proceedings of Machine Learning Research. PMLR, Dec. 2020, pp. 77–89.
- [15] Emanuel Todorov, Tom Erez, and Yuval Tassa. “MuJoCo: A physics engine for model-based control”. In: *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033.
- [16] Pascal den Boef, Pepijn B. Cox, and Roland Tóth. “LPVcore: MATLAB Toolbox for LPV Modelling, Identification and Control”. In: *IFAC-PapersOnLine* 54.7 (2021). 19th IFAC Symposium on System Identification SYSID 2021, pp. 385–390.
- [17] Roland Tóth. *Modeling and Identification of Linear Parameter-Varying Systems*. Springer Berlin, Heidelberg, 2010.
- [18] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*. New York: Wiley Interscience, 1972.
- [19] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States, 2022. URL: <https://www.mathworks.com>.
- [20] Sawyer Fuller et al. “The Python Control Systems Library (python-control)”. In: *Proceedings of the 60th IEEE Conference on Decision and Control*. IEEE. 2021, pp. 4875–4881.
- [21] Fen Wu. “Control of Linear Parameter Varying Systems”. PhD thesis. University of California at Berkeley, 1995.
- [22] Zhong Liu et al. “State Feedback Controller Design for Affine Parameter-Dependent LPV Systems”. In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 9760–9765.
- [23] Stephen Boyd et al. *Linear matrix inequalities in system and control theory*. SIAM, 1994.
- [24] Steven Diamond and Stephen Boyd. “CVXPY: A Python-embedded modeling language for convex optimization”. In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.
- [25] Akshay Agrawal et al. “A rewriting system for convex optimization problems”. In: *Journal of Control and Decision* 5.1 (2018), pp. 42–60.
- [26] Xingyu Zhou, Zejiang Wang, and Junmin Wang. “Automated Ground Vehicle Path-Following: A Robust Energy-to-Peak Control Approach”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.9 (2022), pp. 14294–14305.
- [27] Alberto Isidori. *Nonlinear Control Systems*. Springer London, 1995.
- [28] Hossam S. Abbas et al. “Embedding of Nonlinear Systems in a Linear Parameter-Varying Representation”. In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 6907–6913.

- [29] Patrick J.W. Koelewijn and Roland Tóth. *Automatic Grid-based LPV Embedding of Nonlinear Systems*. Tech. rep. Eindhoven University of Technology, 2021.
- [30] Patrick Jan Willem Koelewijn. “Analysis and Control of Nonlinear Systems with Stability and Performance Guarantees: A Linear Parameter-Varying Approach”. English. Proefschrift. Phd Thesis 1 (Research TU/e / Graduation TU/e). Electrical Engineering, Feb. 2023.
- [31] Arjan van der Schaft. *L₂-Gain and Passivity Techniques in Nonlinear Control*. Springer Cham, 2017.
- [32] Carsten Scherer and Siep Weiland. “Linear matrix inequalities in control”. In: *Lecture Notes, Dutch Institute for Systems and Control, Delft, The Netherlands* 3.2 (2000).
- [33] Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. “Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars”. In: *Proceedings of the 2018 European Control Conference*. 2018, pp. 1341–1348.
- [34] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005.
- [35] Péter Antal, Tamás Péni, and Roland Tóth. “Backflipping With Miniature Quadcopters by Gaussian-Process-Based Control and Planning”. In: *IEEE Transactions on Control Systems Technology* (2023), pp. 1–12.
- [36] Michalis Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*. Vol. 5. PMLR, Apr. 2009, pp. 567–574.
- [37] Yuhan Liu, Pengyu Wang, and Roland Tóth. “Learning For Predictive Control: A Dual Gaussian Process Approach”. In: *arXiv preprint arXiv:2211.03699* (2022).
- [38] Kristóf Floch. “Model-based motion control of the F1TENTH autonomous electrical vehicle”. Bachelor’s thesis. Budapest University of Technology and Economics, 2022. URL: <https://eprints.sztaki.hu/10544/>.
- [39] Fen Lin et al. “Path tracking of autonomous vehicle based on adaptive model predictive control”. In: *International Journal of Advanced Robotic Systems* 16.5 (2019).
- [40] D. Soetanto, L. Lapierre, and A. Pascoal. “Adaptive, non-singular path-following control of dynamic wheeled robots”. In: *Proceedings of the 42nd IEEE International Conference on Decision and Control*. Vol. 2. 2003, 1765–1770 Vol.2.
- [41] Ankit Gupta et al. “A Framework for Vehicle Lateral Motion Control With Guaranteed Tracking and Performance”. In: *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference*. 2019, pp. 3607–3612.
- [42] Chuan Hu et al. “Should the Desired Heading in Path Following of Autonomous Vehicles be the Tangent Direction of the Desired Path?” In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 3084–3094.

- [43] D. Gángó, T. Péni, and R. Tóth. “Learning Based Approximate Model Predictive Control for Nonlinear Systems”. In: *IFAC-PapersOnLine* 52.28 (2019), pp. 152–157.
- [44] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272.
- [45] Jacob R Gardner et al. “GPYtorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration”. In: *Advances in Neural Information Processing Systems*. 2018.
- [46] Shaoru Chen et al. “Learning Region of Attraction for Nonlinear Systems”. In: *Proceedings of the 2021 60th IEEE Conference on Decision and Control*. 2021, pp. 6477–6484.
- [47] Joel A E Andersson et al. “CasADi – A software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.