Technische Universität München
TUM School of Computation, Information and Technology

Gaussian Processes in Control: Performance Guarantees through Efficient Learning

Armin Lederer

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

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr.-Ing. Klaus Diepold

Prüfer*innen der Dissertation:

1. Prof. Dr.-Ing. Sandra Hirche
2. Prof. Dr. Andreas Krause
3. Prof. Dr. Colin Jones

Die Dissertation wurde am 22.02.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 24.05.2023 angenommen.

"There is nothing more practical than a good theory."

Kurt Lewin

Preamble

This thesis summarizes my research conducted at the Chair of Information-oriented Control (ITR) at the Technical University of Munich (TUM). I am very thankful for all the amazing people that I had the pleasure of working with and the support I have received from them during this time.

First of all, I am truly grateful to my doctoral advisor, Prof. Sandra Hirche, who has supported me ever since my Bachelor's thesis, allowing me to explore research fields along my interests and find a topic I love working on. Her critical comments and questions have pushed me to deepen my understanding of control and have brought me valuable inspiration.

I would also like to thank Prof. Andreas Krause for having me as a visiting guest researcher in the Learning and Adaptive Systems Group at ETH Zurich. I have greatly benefited from his machine learning perspective and the fruitful scientific discussions, which have given me plenty of ideas for research.

A special thanks goes out to all my colleagues at ITR and LAS for the endless discussions, comments on my papers, help with experiments and the enjoyable time. In particular, I am grateful to Thomas Beckers and Jonas Umlauft for introducing me to Gaussian processes during my Bachelor's and Master's thesis. I have greatly benefited from their continued support and experiences in my research. Over the years, ITR has become a second home for me, not just because of the amount of time I have spent at the lab, but because of people like Alex Capone, Petar Bevanda, Jan Brüdigam, Pablo Budde gen. Dohmann, Samuel Tesfazgi, Robert Lefringhausen and Xiaobing Dai. The joint kicker matches and lunches, late night discussions and last-minute paper submissions, as well as social evenings have really made my time at ITR memorable. I am indebted to Dr. Stefan Sosnowski, Miruna Werkmeister and Ulrike Scholze for always having an open door and having my back when it comes to administrative and teaching issues. I am also thankful to Mojmír Mutný, Mohammad Reza Karimi and Sebastian Curi for being so welcoming and making my time in Zurich enjoyable despite the many weekends in the office.

I have had the pleasure to work with many outstanding students and much of my work would not have been possible without their help. I particularly want to thank Alejandro Ordóñez Conejo, Korbinian Maier, Kang Lin, Wenxin Xiao, Markus Kessler, Marcel Dißemond, Ralf Römer and Azra Begzadić, who have all pushed me to give my best and payed me back with incredible results.

Finally, I want to thank everyone back from home. I am thankful to my friends and everyone in my Ju-jitsu club for keeping me grounded and reminding me that there is more in life than research. I cannot express my gratitude towards my brother Patrick and my parents. They have always been there for me, provided me with emotional support and have been essential in getting this thesis finished.

Acknowledgments

This work was supported by the EU Seventh Framework Programme FP7/2007-2013 within the ERC Starting Grant "Control based on Human Models" (con-humo), grant agreement no. 337654, by the European Union's Horizon 2020 research and innovation programme within the ERC Consolidator Grant "Safe data-driven control for human-centric systems" (CO-MAN), grant agreement no. 864686, and the German National Scholarship Foundation.

Abstract

For a long time, control engineering research has focused on problems for which accurate models are available. In the last decade, this has started to change, e.g., because of an increasing interest in biomedical applications and autonomous robots remotely operating in unknown and dynamically changing environments. Due to the lack of precise models, designing control strategies for these systems can be difficult. This often prevents achieving a high control performance.

Machine learning offers great promise to overcome this limitation by inferring models from data generated during the operation of a system. In particular, for control applications requiring performance and safety guarantees, Gaussian process (GP) regression exhibits many beneficial properties, such as its data efficiency and strong theoretical foundations. This has led to multiple approaches for deriving and applying learning-based control laws based on GP models. However, many challenges remain unaddressed, which prevents the widespread usage of GP models in control. The existing theory on performance guarantees for GP-based control is restricted to specific control design techniques and does not generalize. Moreover, the impact and importance of training data for control performance are not clear, such that little foundation of current data sampling strategies exists. In addition to these theoretical problems, practical issues such as a high computational complexity and a lack of scalability limit the applicability of GP-based control laws in real-world systems.

In this thesis, we address these challenges by developing a framework that covers all relevant aspects for analyzing and implementing learning-based control laws based on GP models. We begin our analysis with GP models themselves and derive a Bayesian prediction error bound. Moreover, we demonstrate how all necessary parameters of this bound can be effectively determined. Based on this model accuracy guarantee, we investigate the performance of a large class of GP-based tracking control laws and prove different probabilistic tracking error bounds. In order to formalize the relationship between control performance guarantees and available training data, we propose a data density measure that is tailored to GPs. By utilizing this measure, we can derive decay rates of tracking error bounds with increasing data set sizes and illustrate the state dependency of data density requirements for good control performance. This insight is used to develop data sampling schemes for offline and online learning with GP models, which can guarantee arbitrarily high tracking accuracies.

In order to enable the practical applicability of the proposed strategies, we develop a computationally efficient approximation for GP-based online learning. By aggregating multiple, locally active GP models, updates and predictions can be efficiently computed. Furthermore, error bounds from exact GP regression are proven to straightforwardly extend under weak assumptions on the aggregation method. In addition to online learning, we show that the aggregation of GP predictions enables distributed GP models in multi-agent systems. For achieving this, we perform the aggregation using a consensus algorithm. This allows the straightforward extension of prediction error bounds, which is exploited to derive accuracy guarantees for a cooperative tracking control law employing the distributed GP model. Finally, we address the problem of implementing GP-based control on real-world systems using such approximations and present architectures which help to mitigate remaining practical limitations such as high memory requirements. The effectiveness of the presented architectures for learning-based control with GP models is demonstrated in several robotics experiments and simulations.

Zusammenfassung

Lange Zeit hat sich die regelungstechnische Forschung auf Probleme konzentriert, für die genaue Modelle verfügbar sind. In den letzten zehn Jahren begann sich dies zu ändern, z. B. aufgrund des zunehmenden Interesses an biomedizinischen Anwendungen und autonomen Robotern, die in abgelegenen, unbekannten und sich dynamisch verändernden Umgebungen arbeiten. Das Fehlen präziser Modelle für solche Systeme erschwert im Allgemeinen den Reglerentwurf, so dass oft keine hohe Performanz erzielt werden kann.

Maschinelles Lernen verpsricht diese Einschränkung zu überwinden, indem es Modelle aus Daten ableitet, die während des Betriebs eines Systems erzeugt werden. Insbesondere für regelungstechnische Anwendungen, die Performanz- und Sicherheitsgarantien erfordern, weist die Gauß-Prozess (GP) Regression viele vorteilhafte Eigenschaften auf, wie z. B. ihre Dateneffizienz und ihr solides theoretisches Fundament. Dies hat zu zahlreichen Ansätzen für die Herleitung und Anwendung lernfähiger Regelungsgesetze auf der Grundlage von GP-Modellen geführt. Allerdings gibt es noch viele ungelöste Probleme, die den weit verbreiteten Einsatz von GP-Modellen in der Regelungstechnik verhindern. Die bestehende Theorie zu Performanzgarantien für GP-basierte Regelung ist auf spezifische Techniken zum Reglerentwurf beschränkt und lässt sich nicht verallgemeinern. Darüber hinaus sind die Auswirkungen und die Bedeutung der Trainingsdaten für die Regelungsperformanz nicht klar, so dass es kaum eine Grundlage für die aktuellen Strategien zur Datenerzeugung gibt. Zusätzlich zu diesen theoretischen Limitationen schränken praktische Probleme wie die hohe Rechenkomplexität und die mangelnde Skalierbarkeit die Anwendbarkeit von GP-basierten Regelgesetzen in realen Systemen ein.

In dieser Arbeit gehen wir diese Herausforderungen an, indem wir einen Rahmen entwickeln, der alle relevanten Aspekte für die Analyse und Implementierung von lernenden Regelgesetzen auf der Basis von GP-Modellen abdeckt. Wir beginnen unsere Analyse mit den GP-Modellen selbst und leiten eine Bayessche Modellfehlerschranke her. Darüber hinaus zeigen wir, wie alle notwendigen Parameter dieser Schranke effektiv bestimmt werden können. Auf der Grundlage dieser Modellgenauigkeitsgarantie untersuchen wir die Leistung einer großen Klasse von GP-basierten Folgeregelungsgesetzen und beweisen verschiedene probabilistische Folgefehlerschranken. Um die Beziehung zwischen den Performanzgarantien der Regelung und den verfügbaren Trainingsdaten zu formalisieren, schlagen wir ein Datendichtemaß vor, das auf GPs zugeschnitten ist. Mit Hilfe dieses Maßes können wir Abnahmeraten für Folgefehlerschranken mit zunehmender Datensatzgröße herleiten und die Zustandsabhängigkeit der Anforderungen an die Datendichte für eine gute Reglerperformanz veranschaulichen. Diese Einsicht wird genutzt, um Datensampling-Schemata für das Offline- und Online-Lernen mit GP-Modellen zu entwickeln, die beliebig hohe Folgegenauigkeiten garantieren können.

Um die praktische Anwendbarkeit der vorgeschlagenen Strategien zu ermöglichen, entwickeln wir eine rechnerisch effiziente Approximation für GP-basiertes Online-Lernen. Durch die Aggregation mehrerer, lokal aktiver GP-Modelle können Aktualisierungen und Model Evaluierungen effizient berechnet werden. Darüber hinaus wird bewiesen, dass sich die Fehlerschranken der exakten GP-Regression unter schwachen Annahmen an die Aggregationsmethode einfach erweitern lassen. Zusätzlich zum Online-Lernen zeigen wir, dass die Aggregation von GP-Prädiktionen verteilte GP-Modelle in Multiagentensystemen ermöglicht. Um dies zu erreichen, führen wir die Aggregation mit Hilfe eines Konsensalgorithmus durch. Dies ermöglicht eine unkomplizierte Erweiterung der Prädiktionsfehlerschranken, die genutzt

wird, um Genauigkeitsgarantien für ein kooperatives Folgeregelungsgesetz abzuleiten, das das verteilte GP-Modell verwendet. Schließlich befassen wir uns mit dem Problem der Implementierung einer GP-basierten Regelung auf realen Systemen unter Verwendung solcher Approximationen und stellen Architekturen vor, die dazu beitragen, die verbleibenden praktischen Einschränkungen, wie z. B. den hohen Speicherbedarf, zu mildern. Die Effektivität der vorgestellten Architekturen für die lernbasierte Steuerung mit GP-Modellen wird in mehreren Robotikexperimenten und Simulationen demonstriert.

Contents

1. Introduction

Control theory and engineering deal with the fundamental problem of forcing a system to behave in a desired way [1]. This abstract definition includes many practical problems ranging from the simple temperature regulation in a refrigerator [2] to the complex synchronization of electric power systems [3]. For enforcing the desired behavior, a rule, the so-called control law, needs to be defined, which maps measurements to the actions applied to a system. The control law is classically designed using a model of the considered system, which can be accurately determined using physics principles in many applications such as industrial machinery and chemical processes [4, 5, 6]. While this approach has paved the road for various technological advancements in the past [7], the necessity of such first principle models also poses a considerable limitation. This has become increasingly apparent with the recent interest in applications such as biomedical engineering and autonomous robots.

For example, robotic rehabilitation has received growing attention due to the need for therapeutic treatment of people with neurological disorders caused, e.g., by stroke or spinal injuries [8, 9]. Since the exoskeletons used for robotic rehabilitation are often constructed with series-elastic actuators in order to exhibit a compliant behavior [10], describing them mathematically is complicated and accurate models are challenging to obtain in practice. This lack of a precise mathematical description is even amplified when the exoskeleton is attached to a patient: humans are notoriously challenging to model due to the absence of principled methods to explain them, their change of behavior over time, and the variation between individual persons [11]. Due to this absence of precise models for human-robot systems, a high control performance can often not be achieved, and even the safety of the human is potentially endangered.

While humans are probably one of the most prominent examples for the absence of precise models, robotic systems are also strongly affected. Due to the growing computational power of low-cost processors, robots such as underwater, aerial, and wheeled vehicles operate increasingly autonomously. The dynamical behavior of these systems is well understood in nominal operating conditions, but complex, unmodeled effects can occur when leaving them, e.g., when a drone flies an aggressive maneuver [12] or when an autonomous car drives near the limits of tire adhesion [13]. Additionally, when autonomous robots are employed in remote, outdoor areas, unknown environmental effects like wind or water currents can have a significant, time-varying effect on the robot behavior [14, 15]. Just as in robotic rehabilitation, this lack of an accurate mathematical system description poses a critical problem for the design of control laws, which is a potential threat to the safe execution of remote tasks with autonomous robots.

These examples demonstrate the need for precise mathematical descriptions of complex and possibly dynamically changing system behaviors in control design. Since data is continuously generated during the operation of technical systems, such system models can be

obtained by employing techniques from supervised machine learning, which has led to the specification of learning-based system identification in control systems as a key research and innovation challenge [16]. The probably most popular approach for supervised learning is deep learning with artificial neural networks [17]. Although it has seen tremendous success in applications with access to precise simulation environments such as games [18, 19], applying these results to practical control scenarios has proven difficult. This is due to several attributes of deep learning, which are not ideal for control problems. Deep neural networks generally need to be trained using large amounts of data, such that even for simple control problems, hundreds of thousands of training samples are used [20]. Collecting such data sets from real-world systems takes a long time and is expensive, which severely limits the applicability of deep learning in control. In addition, state-of-the-art training methods for deep learning do not admit sequential updates, which prevents a fast adaptation of deep learning-based control laws to changing environmental conditions. Finally, the derivation of certifiable performance guarantees for deep learning-based control laws is challenging, such that the safety of real-world systems is a relevant concern. This is particularly problematic because safety plays a vital role in the practical application of learning-based algorithms [21].

Many of these weaknesses of deep learning are avoided by employing Gaussian process (GP) [22] models in control laws. GPs are generally considered to be data-efficient and allow us to learn models for simple control problems already from a few dozen training samples [23]. Moreover, their explicit expressions for model evaluations allow the straightforward derivation of exact, iterative model updates [24], and their potential for non-parametric regression can make GP models universal approximators [25]. Finally, they provide a measure of model uncertainty along with their predictions of the system behavior. This has led to a frequent application in learning-based control experiments in recent years [26, 27, 28, 29], which is complemented by a theoretical analysis of many control design approaches with GP models [30, 31, 32, 33, 34]. Motivated by these previous works, this thesis considers the problem of deriving a framework for learning-based control with GP models, which covers all relevant aspects for analyzing and implementing GP-based controllers. In the remainder of this chapter, we first illustrate open challenges for control with GP models in Section 1.1. Our contributions addressing these challenges and the structure of this thesis are outlined in Section 1.2.

1.1. Challenges in Gaussian Process-Based Control

Despite the growing interest in GP models from theoretical and applied control perspectives, research on GP-based control is only at an early stage, and many questions still need to be answered. A particularly important open question revolves around GP-based control laws for tracking problems, in which a dynamical system has to be steered along a reference trajectory. Examples for this type of problem include underwater, aerial, and wheeled robots following a desired path.

In order to enable the theoretically supported application of learning-based controllers with GP models in real-world trajectory tracking problems, we believe that solutions for the following open problems are essential.

Challenge 1. *How can we ensure data-dependent learning error bounds for GP models?*

In order to safely employ a model in control laws, we must be able to trust it. This is

not a crucial problem in classical identification methods with parametric models, for which it suffices to derive accuracy guarantees for inferred model parameters. When GP models with their potentially non-parametric structure are employed, this approach is not possible since the prediction accuracy can depend heavily on the available data in the proximity of a test point. Therefore, the data-driven nature of GP models necessitates the derivation of learning error bounds, which can reflect locally varying model accuracies.

Challenge 2. *How can we analyze the tracking accuracy for a broad class of control laws based on Gaussian process models?*

While tracking error bounds have been derived for many GP-based tracking control laws as a guarantee for their control performance, the current theory is specific for certain control design techniques and does not generalize. Therefore, every new control design requires a new theoretical analysis. This prohibits the combination of GP models with novel control techniques and drains research resources from crucial questions beyond the straightforward performance analysis. Therefore, generally applicable theoretical guarantees for a wide range of tracking control laws are necessary when GP models are employed in the control loop.

Challenge 3. *How do training samples impact the certifiable control performance, and how do we get valuable data in closed-loop control?*

A crucial promise of learning-based control approaches is the improvement of the performance with growing data sets. However, this conjecture has barely been investigated theoretically, and the general relationship between available training data and certifiable control performance is largely unknown. As a consequence, approaches for offline data generation and online learning with GP models are employed in literature, for which little theoretical justification exists. Therefore, we need to establish a direct relationship between the locally available training data density and its impact on control performance guarantees. This insight is necessary to develop efficient data generation and online learning approaches that fulfill the conjectured improvement guarantees.

Challenge 4. *How can we reduce the computational complexity to enable online learning and distributed learning in multi-agent systems while retaining accuracy guarantees?*

The large number of theoretically advantageous properties of GP regression is accompanied by a severe practical limitation: the computational complexity of GP regression scales cubically with the number of training samples. This prevents the exact inference of GP models from large data sets and causes computation times for iterative model updates, which are often significantly larger than the sampling time of practically found control loops. Moreover, the closed-form expression for GP regression provides a huge advantage in its theoretical analysis, but it does not directly admit a distributed computation. This limits the scalability of exact GP models to large-scale multi-agent systems since a centralized computation of model evaluations is required. Since learning error bounds are crucial, as discussed for Challenge 1, this explains the need for computationally efficient GP approximations, which enable online and distributed learning while retaining model accuracy guarantees.

Challenge 5. *Which adaptations of derived GP-based control laws are necessary for an implementation in real-world systems but keep theoretical guarantees?*

While suitable assumptions allow proving many theoretical properties of learning-based control laws, the relevant condition for the practical usefulness of GP-based controllers is usually the validity of these guarantees in real-world experiments. This requires the additional consideration of practical restrictions, e.g., due to real sensors, actuators, and computational infrastructure. Depending on the available hardware, a direct implementation of theoretically derived GP-based control laws is possible, but it might be accompanied by undesired negative effects. Therefore, modifications of the implementation architecture of GP-based control laws must be investigated in order to determine beneficial structures for practical limitations while retaining theoretical guarantees.

1.2. Main Contributions and Outline

In this thesis, we address these challenges of learning-based tracking control with GP models by deriving a framework that spans the complete relevant process necessary for implementing a GP-based controller on a real-world system. Therefore, our analysis starts with the control design and data collection strategies, continues with tracking accuracy guarantees, and finishes with practically implementable learning-based control architectures with GP models. Each chapter of this thesis is devoted to one of the challenges described in Section 1.1 as outlined in the following.

Chapter 2: Gaussian Process Regression and Prediction Error Bounds We begin our analysis by investigating different approaches for the derivation of GP model error bounds in order to address Challenge 1. For this purpose, we first provide an intuitive introduction to GP-based machine learning with a particular focus on illustrating GP models from different angles. By deriving non-parametric GP models from Bayesian linear regression, we obtain a weight-space view on GP regression. We show that this perspective allows the straightforward extension of error bounds from parametric regression problems to learning error guarantees for general GP models. Furthermore, we exploit the interpretation of GPs as a distribution over models to derive Bayesian regression error bounds, which exploit the regularity of GP sample functions and tail bounds for the Gaussian distribution. The results in this chapter have been partially published in [35, 36].

Chapter 3: Tracking Control with Gaussian Process Models Concerned with Challenge 2, we derive novel tracking error bounds for two scenarios. First, we consider the special case of linear systems with nonlinear input perturbations, which are compensated in control using GP models. We show that this setting allows the formulation of tracking error bounds in the form of dynamical systems, with the GP error bound taking the role of the input. While this linear systems scenario is very effective for the theoretical analysis, it causes a restriction to linear feedback control laws. Therefore, we additionally analyze general, nonlinear dynamical systems for which control laws are designed using the GP mean function. We show that this design approach, which follows the ideas of the certainty equivalence principle, admits the derivation of error bounds by analyzing the temporal decay of tracking error proxies. Due to the generality of our analysis, it is applicable to a wide range of practically employed nonlinear control laws. The results in this chapter have been partially published in [36].

Chapter 4: Learning for Control with Arbitrary Accuracy Guarantees In order to address Challenge 3, we establish a direct relationship between tracking error bounds for GP-based control and the density of available training data. We achieve this by first investigating the dependency of GP error bounds on the training data. This analysis yields a novel measure for the local density of data, which is specific to each GP model. Based on the proposed density measure, we derive asymptotic guarantees for the GP model accuracy with growing data set sizes. In combination with previously derived tracking error bounds, these results directly ensure the improvement of control performance with additional training data. Moreover, the structure of our data density measure allows the analysis of local training data requirements for achieving desired tracking accuracies. We exemplarily investigate this dependency for a control law revealing that training data in the proximity of the reference trajectory is potentially insufficient for ensuring a high tracking accuracy using GP-based control. In order to obtain suitable training data from control systems, we develop multiple data sampling schemes for learning in closed-loop systems. By exploiting the gained insights about training data requirements, we show that the proposed schemes can ensure desired tracking accuracy guarantees by design. The results in this chapter have been partially published in [36, 37].

Chapter 5: Efficient Learning via Gaussian Process Model Aggregation As a solution approach for Challenge 4, we propose the utilization of model aggregation schemes for enabling GP-based online learning and distributed learning in multi-agent systems. First, we focus on the problem of performing online GP model updates and evaluations at rates necessary for closed-loop control systems. To address this problem, we propose an iterative approach for constructing tree structures with GP models in their leaf nodes. By expanding the tree whenever necessary, the number of training samples in each GP model can be kept bounded, which allows fast model updates. Moreover, our method ensures that only a few of the GP models must be evaluated for computing predictions, which in turn guarantees a low computational complexity. Due to the structure of commonly used aggregation schemes, the proposed GP-based online learning method directly inherits model error bounds from exact GP regression. Therefore, we can safely employ it in learning-based control applications and keep tracking accuracy guarantees derived for exact GP models. In order to obtain scalable GP models for multi-agent systems, we propose a consensus-based model aggregation approach. Due to the strong theoretical foundation of consensus algorithms, the error bounds for aggregated models straightforwardly extend to the distributed aggregation. We exploit these guarantees to employ the consensus-based GP model aggregation in a cooperative tracking control law, which we prove to provide tracking accuracy guarantees. The results in this chapter have been published in [38, 39].

Chapter 6: Architectures for Practical Control with Gaussian Processes Since Challenge 5 is concerned with the implementation of GP-based control laws on real-world systems, we propose and investigate different learning-based control architectures. We begin our analysis with a synchronous architecture corresponding to the direct realization of theoretically derived controllers. Due to the limitations of real sensors, we determine the effect of measurement disturbances on the derived performance guarantees and illustrate their effect in simulations. The real-world applicability of the synchronous architecture is demonstrated in an experiment with a robotic manipulator. Since the synchronous evaluation of GP models

and other components of control laws can be challenging, we also propose an asynchronous architecture. This architecture computationally decouples GP predictions from the rest of the control loop, such that time-consuming operations on the GP model do not slow down the overall control loop. While this potentially causes delayed GP predictions in the control law, we prove the deterioration caused by the delay to be bounded and illustrate the advantages in comparison to the synchronous architecture in simulations. The straightforward applicability of the asynchronous architecture is demonstrated in a human-robot interaction experiment involving several participants. Finally, we propose a networked architecture that alleviates the escalating memory requirements of online learned GP models. By performing a reachability analysis based on the derived tracking accuracy guarantees, relevant data for the GP model can be determined. Therefore, the irrelevant data can be regularly sent to the cloud via a network connection, from where it can be fetched if it is eventually needed again on the local device. This allows the enforcement of resource constraints on the local system, while GP model accuracy guarantees are unaffected. The effectiveness of this networked online learning control architecture with GP models is demonstrated in a simulation of a robotic exoskeleton attached to a human, which illustrates practical benefits in comparison to the synchronous architecture. The results in this chapter have been partially published in [40, 41].

Chapter 7: Conclusion and Outlook on Future Research Directions Finally, this thesis is concluded by a review of the presented results. Moreover, important challenges for research on control with GP models are presented, and the first approaches for solving remaining open problems are discussed.

Gaussian Process Regression and Prediction Error Bounds

2.

Gaussian process (GP) regression [22], also referred to as Kriging [42], is a supervised machine learning method whose basic ideas can be dated back at least as far as the 1940s with works from Wiener [43] and Kolmogorov [44] on time series analysis. As the name indicates, its fundamental component is a GP, which can be interpreted as a generalization of the normal distribution. Since GPs naturally arise in many problems, their statistical properties have been thoroughly investigated and are generally well understood [45]. Moreover, they admit the straightforward application of Bayesian principles, such that GPs can serve as the basis for various probabilistic machine learning techniques [46].

When applied to regression problems, GPs can be used to infer a model $\mu : \mathbb{R}^{d_z} \to \mathbb{R}$ of an unknown function $f : \mathbb{R}^{d_z} \to \mathbb{R}$ from data. For this purpose, we assume to have access to a data set $\mathbb{D} = \{(\boldsymbol{z}^{(n)}, y^{(n)} = f(\boldsymbol{z}^{(n)}) + \epsilon^{(n)})\}_{n=1}^{N}$ consisting of $N \in \mathbb{N}$ sample pairs $(\boldsymbol{z}^{(n)}, y^{(n)})$, where $\boldsymbol{z}^{(n)} \in \mathbb{R}^{d_z}$ are training inputs and $y^{(n)} \in \mathbb{R}$ are training targets. The training targets are assumed to be measurements of the unknown function $f(\cdot)$ perturbed by observation noise $\epsilon^{(n)} \in \mathbb{R}$. While this standard regression problem can be solved using many approaches ranging from a simple least squares fitting of a polynomial [47] to complex techniques such as deep learning [17], GP regression exhibits the advantage of providing a measure of uncertainty along with the model $\mu(\cdot)$. This explicit uncertainty representation enables the derivation of probabilistic error bounds as formalized in the following.

Definition 2.1. *A model $\mu(\cdot)$ of an unknown function $f(\cdot)$ admits a uniform error bound $\eta : \mathbb{R}^{d_z} \to \mathbb{R}_{0,+}$ with probability of at least $1-\delta$, $\delta \in (0, 1]$, on the domain $\mathbb{S} \subset \mathbb{R}^{d_z}$ if*

$$\mathbb{P}\Big(|\mu(\boldsymbol{z}) - f(\boldsymbol{z})| \leq \eta(\boldsymbol{z}), \quad \forall \boldsymbol{z} \in \mathbb{S}\Big) \geq 1 - \delta. \tag{2.1}$$

It is important to note that the probability in (2.1) must hold jointly for all $\boldsymbol{z} \in \mathbb{S}$. Therefore, any approach employing the GP model on the domain $\mathbb{S}$ can be analyzed using deterministic methods accounting for the model error $\eta(\cdot)$, and the resulting guarantees will hold with probability $1-\delta$. This is particularly useful in control, as it enables the application of robust control techniques [48].

In the remainder of this chapter, we present different approaches for deriving probabilistic error bounds. We start by introducing the fundamentals of GP regression in Section 2.1, where we put a particular focus on different perspectives on regression problems. Based on a classical weight space view, we demonstrate how parametric error bounds for regularized regression can be straightforwardly extended to non-parametric GP regression in Section 2.2. In Section 2.3, we take a more function space-oriented perspective and derive Bayesian uniform error bounds. The chapter is concluded by a discussion of the results in Section 2.4.

2.1. Fundamentals of Gaussian Process Regression

While GP regression is a term often used to refer to supervised learning using a specific set of equations, it actually comprises different formulations and perspectives. This is due to the flexibility of GPs, which are defined, according to [22], as follows.

Definition 2.2. *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

This definition merely requires a joint Gaussian distribution between random variables, but it does not impose any restrictions on the description of a GP. Therefore, GPs can be represented in different forms, which provide a variety of insights into regression and can offer problem-specific advantages.

In the remainder of this section, we review the fundamentals of Gaussian process regression. Based on a parametric description of GPs, we present computationally efficient equations for regression in finite-dimensional feature spaces in Section 2.1.1. Using the kernel trick [49], we transform these equations in order to enable non-parametric regression in possibly infinite-dimensional feature spaces in Section 2.1.2. In Section 2.1.3, we introduce a function space perspective on GP regression, which allows the decomposition of learned models into additive and multiplicative components. We outline how GPs can be employed for regression of multi-dimensional training targets in Section 2.1.4 before we briefly introduce approaches for hyperparameter tuning in Section 2.1.5.

2.1.1. Parametric Regression in Feature Spaces

When performing GP regression in finite-dimensional feature spaces, which is commonly known as Bayesian linear regression [47], we assume that the unknown function $f(\cdot)$ has the form

$$f(\boldsymbol{z}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{z}), \tag{2.2}$$

where $\boldsymbol{\phi} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_w}$ is a potentially high dimensional, non-linear feature map and $\boldsymbol{w} \in \mathbb{R}^{d_w}$ are the corresponding weights. Moreover, we place a Gaussian prior $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_w^0, \boldsymbol{\Sigma}_w^0)$ on the weights $\boldsymbol{w}$ with mean $\boldsymbol{\mu}_w^0 \in \mathbb{R}^{d_w}$ and covariance matrix $\boldsymbol{\Sigma}_w^0 \in \mathbb{R}^{d_w \times d_w}$. Due to this prior distribution, it immediately follows that function evaluations $f(\boldsymbol{z})$, $f(\boldsymbol{z}')$ at inputs $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{R}^{d_z}$ are jointly Gaussian distributed. Thus, the conditions of Definition 2.2 are satisfied, and $f(\cdot)$ follows a GP distribution. In order to ensure that any collection of random variables $y^{(n)}$, $n = 1, \ldots, N$, also satisfies the conditions of Definition 2.2, we make the following assumption on the distribution of the observation noise.

Assumption 2.1. *The independent and identically distributed (i.i.d.) observation noise $\epsilon^{(n)}$ follows a Gaussian distribution with mean 0 and variance σ_{on}^2, i.e., $\epsilon^{(n)} \sim \mathcal{N}(0, \sigma_{\text{on}}^2)$ for all $n \in \mathbb{N}$.*

Since the relationship between $f(\boldsymbol{z})$ and $\boldsymbol{z}$ is deterministic for fixed weights $\boldsymbol{w}$, this assumption implies that $p(\boldsymbol{y}|\boldsymbol{Z}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^T \boldsymbol{\Phi}(\boldsymbol{Z}), \sigma_{\text{on}}^2 \boldsymbol{I}_N)$, where we concatenate all training input and target samples into $\boldsymbol{Z} = [\boldsymbol{z}^{(1)} \ \cdots \ \boldsymbol{z}^{(N)}]$ and $\boldsymbol{y} = [y^{(1)} \ \cdots \ y^{(N)}]^T$, respectively.

Moreover, we use the shorthand notation $\boldsymbol{\Phi}(\boldsymbol{Z}) = [\boldsymbol{\phi}(\boldsymbol{z}^{(1)}) \ \cdots \ \boldsymbol{\phi}(\boldsymbol{z}^{(N)})]$. This allows us to directly employ Bayes' law

$$p(\boldsymbol{w}|\boldsymbol{Z}, \boldsymbol{y}) = \frac{p(\boldsymbol{w}, \boldsymbol{y}|\boldsymbol{Z})}{p(\boldsymbol{y}|\boldsymbol{Z})} = \frac{p(\boldsymbol{y}|\boldsymbol{Z}, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{Z})}, \tag{2.3}$$

where $p(\boldsymbol{y}|\boldsymbol{Z})$ is merely a normalization constant. Therefore, all involved distributions are Gaussian, which results in a Gaussian posterior distribution $p(\boldsymbol{w}|\boldsymbol{Z}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, whose mean $\boldsymbol{\mu}_w$ and variance $\boldsymbol{\Sigma}_w$ can be straightforwardly derived as [47]

$$\boldsymbol{\mu}_w = \boldsymbol{\Sigma}_w \left(\left(\boldsymbol{\Sigma}_w^0\right)^{-1} \boldsymbol{\mu}_w^0 + \frac{1}{\sigma_{\text{on}}^2} \boldsymbol{\Phi}(\boldsymbol{Z}) \boldsymbol{y} \right), \tag{2.4}$$

$$\boldsymbol{\Sigma}_w = \left(\left(\boldsymbol{\Sigma}_w^0\right)^{-1} + \frac{1}{\sigma_{\text{on}}^2} \boldsymbol{\Phi}(\boldsymbol{Z}) \boldsymbol{\Phi}^T(\boldsymbol{Z}) \right)^{-1}. \tag{2.5}$$

These equations require the inversion of $d_w \times d_w$ dimensional matrices, such that this form of GP regression exhibits a computational complexity of $\mathcal{O}(d_w^3)$. Therefore, the complexity of parameter inference only depends on the number of unknown weights, but it is independent of the number of training samples N.

When the training samples are generated and processed sequentially, this complexity can even be reduced through iterative model updates. For this purpose, we consider the posterior distribution after $N-1$ training samples[1] as prior, such that Bayes' law yields

$$p(\boldsymbol{w}|\boldsymbol{Z}_N, \boldsymbol{y}_N) = \frac{p(y^{(N)}|\boldsymbol{z}^{(N)}, \boldsymbol{w}) p(\boldsymbol{w}|\boldsymbol{Z}_{N-1}, \boldsymbol{y}_{N-1})}{p(y^{(N)}|\boldsymbol{z}^{(n)})}. \tag{2.6}$$

Therefore, we obtain, analogously to (2.4) and (2.5), a posterior mean vector and a covariance matrix

$$\boldsymbol{\mu}_w^N = \boldsymbol{\Sigma}_w^N \left(\left(\boldsymbol{\Sigma}_w^{N-1}\right)^{-1} \boldsymbol{\mu}_w^{N-1} + \frac{1}{\sigma_{\text{on}}^2} \boldsymbol{\phi}(\boldsymbol{z}^{(N)}) y^{(N)} \right), \tag{2.7}$$

$$\boldsymbol{\Sigma}_w^N = \left(\left(\boldsymbol{\Sigma}_w^{N-1}\right)^{-1} + \frac{1}{\sigma_{\text{on}}^2} \boldsymbol{\phi}(\boldsymbol{z}^{(N)}) \boldsymbol{\phi}^T(\boldsymbol{z}^{(N)}) \right)^{-1}. \tag{2.8}$$

While this formulation does not yet provide a direct computational advantage, it should be noted that for given matrices $\left(\boldsymbol{\Sigma}_w^{N-1}\right)^{-1}$ and $\boldsymbol{\Sigma}_w^N$, the mean computation (2.7) has quadratic complexity $\mathcal{O}(d_w^2)$. Moreover, $\left(\boldsymbol{\Sigma}_w^N\right)^{-1}$ can be updated with quadratic complexity based on (2.8) since $\boldsymbol{\phi}(\boldsymbol{z}^{(N)})$ is a vector. Therefore, the computation of the inverse on the right side of (2.8) is the only operation causing a cubic complexity. This inversion can be avoided by applying Corollary A.2, commonly known as the Sherman-Woodbury-Morrison formula, which results in

$$\boldsymbol{\Sigma}_w^N = \boldsymbol{\Sigma}_w^{N-1} - \frac{\boldsymbol{\Sigma}_w^{N-1} \boldsymbol{\phi}(\boldsymbol{z}^{(N)}) \boldsymbol{\phi}^T(\boldsymbol{z}^{(N)}) \boldsymbol{\Sigma}_w^{N-1}}{\sigma_{\text{on}}^2 + \boldsymbol{\phi}^T(\boldsymbol{z}^{(N)}) \boldsymbol{\Sigma}_w^{N-1} \boldsymbol{\phi}(\boldsymbol{z}^{(N)})}. \tag{2.9}$$

This formula can be computed in a quadratic number of operations since it merely requires matrix additions, matrix-vector, and vector-vector products. Therefore, it enables computationally efficient online model updates with a complexity of $\mathcal{O}(d_w^2)$.

[1] Whenever necessary for clarity due to a varying number of training samples, we use a superscript to indicate the number of training samples.

2.1.2. Non-Parametric Regression using the Kernel Trick

While parametric GP regression can be performed computationally efficiently, its flexibility is inherently limited by the necessity of explicitly specifying features $\boldsymbol{\phi}(\cdot)$. As this limitation does not arise from the need for fixed features, but rather the restriction to a finite number of them [22], we present a reformulation of the regression equations (2.4) and (2.5), which allows tractable inference in infinite-dimensional, implicitly defined feature spaces.

For this purpose, we start by considering the posterior distribution of the unknown function $f(\cdot)$ directly, which is immediately obtained from (2.2), (2.4), and (2.5) as $p(f(\boldsymbol{z})|\boldsymbol{z}, \boldsymbol{Z}, \boldsymbol{y}) = \mathcal{N}(\mu(\boldsymbol{z}), \sigma^2(\boldsymbol{z}))$, where the mean function $\mu : \mathbb{R}^{d_z} \to \mathbb{R}$ and the variance function $\sigma^2 : \mathbb{R}^{d_z} \to \mathbb{R}_{0,+}$ are defined as

$$\mu(\boldsymbol{z}) = \boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w \left(\boldsymbol{\Sigma}_w^0\right)^{-1} \boldsymbol{\mu}_w^0 + \frac{1}{\sigma_{\text{on}}^2}\boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{Z})\boldsymbol{y}, \tag{2.10}$$

$$\sigma^2(\boldsymbol{z}) = \boldsymbol{\phi}^T(\boldsymbol{z}) \left(\left(\boldsymbol{\Sigma}_w^0\right)^{-1} + \frac{1}{\sigma_{\text{on}}^2}\boldsymbol{\Phi}(\boldsymbol{Z})\boldsymbol{\Phi}^T(\boldsymbol{Z})\right)^{-1} \boldsymbol{\phi}(\boldsymbol{z}). \tag{2.11}$$

Due to Lemma A.1, often referred to as the Woodbury matrix inversion lemma, we can equivalently express (2.11) as

$$\sigma^2(\boldsymbol{z}) = \boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w^0\boldsymbol{\phi}(\boldsymbol{z}) - \boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w^0\boldsymbol{\Phi}(\boldsymbol{Z}) \left(\sigma_{\text{on}}^2\boldsymbol{I}_N + \boldsymbol{\Phi}^T(\boldsymbol{Z})\boldsymbol{\Sigma}_w^0\boldsymbol{\Phi}(\boldsymbol{Z})\right)^{-1} \boldsymbol{\Phi}^T(\boldsymbol{Z})\boldsymbol{\Sigma}_w^0\boldsymbol{\phi}(\boldsymbol{z}), \tag{2.12}$$

which only depends on inner products of features $\boldsymbol{\phi}(\cdot)$. Therefore, we can exploit the kernel trick, which bases on the idea of avoiding an explicit feature representation through an implicit encoding in a kernel [49]. This can be achieved by defining a kernel $k(\boldsymbol{z}, \boldsymbol{z}') = \boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w^0\boldsymbol{\phi}(\boldsymbol{z}')$, which yields

$$\sigma^2(\boldsymbol{z}) = k(\boldsymbol{z}, \boldsymbol{z}) - \boldsymbol{k}^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2\boldsymbol{I}_N\right)^{-1} \boldsymbol{k}(\boldsymbol{z}), \tag{2.13}$$

where the Gram matrix $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ has elements $K_{i,j} = k(\boldsymbol{z}^{(i)}, \boldsymbol{z}^{(j)})$, $i, j = 1, \ldots, N$ and the kernel vector $\boldsymbol{k}(\boldsymbol{z}) \in \mathbb{R}^N$ is defined through $k_i(\boldsymbol{z}) = k(\boldsymbol{z}, \boldsymbol{z}^{(i)})$, $i = 1, \ldots, N$.

For the first summand of the posterior mean function (2.10), we can proceed analogously, such that we obtain

$$\boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w \left(\boldsymbol{\Sigma}_w^0\right)^{-1} \boldsymbol{\mu}_w^0 = \boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\mu}_w^0 - \boldsymbol{k}^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2\boldsymbol{I}_N\right)^{-1} \boldsymbol{\Phi}^T(\boldsymbol{Z})\boldsymbol{\mu}_w^0 \tag{2.14}$$

due to the kernel trick [49] and Lemma A.1. The second summand in (2.10) can be reformulated using Corollary A.1, known as the matrix push-through lemma, which results in

$$\frac{1}{\sigma_{\text{on}}^2}\boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{Z})\boldsymbol{y} = \boldsymbol{k}^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2\boldsymbol{I}_N\right)^{-1} \boldsymbol{y}. \tag{2.15}$$

Therefore, we can equivalently express the posterior mean function (2.10) as

$$\mu(\boldsymbol{z}) = \mu^0(\boldsymbol{z}) + \boldsymbol{k}^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2\boldsymbol{I}_N\right)^{-1} (\boldsymbol{y} - \boldsymbol{\mu}^0), \tag{2.16}$$

where we define the prior mean function as $\mu^0(\boldsymbol{z}) = \boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\mu}_w^0$ and the elements of the mean data vector $\boldsymbol{\mu}^0 \in \mathbb{R}^N$ as $\mu_i^0 = \mu^0(\boldsymbol{z}^{(i)})$, $i = 1, \ldots, N$.

Since the explicit dependency on the features $\boldsymbol{\phi}(\cdot)$ is removed in the reformulated expressions (2.13) and (2.16), these formulas allow to completely base GP regression on kernels $k(\cdot,\cdot)$. The kernels can be defined by directly exploiting the kernel trick, which leads to dot product kernels such as polynomial covariance functions

$$k_{\text{poly}}(\boldsymbol{z},\boldsymbol{z}') = \left(\sigma_0^2 + \boldsymbol{z}^T\boldsymbol{\Sigma}_w^0\boldsymbol{z}\right)^p, \tag{2.17}$$

where $\sigma_0^2 \in \mathbb{R}_{0,+}$ can be considered a hyperparameter and $p \in \mathbb{R}_+$ denotes the degree of the polynomial. Moreover, there exists a wide range of kernels that implicitly define an infinite number of features. Probably the most commonly used among them are squared exponential (SE) kernels with automatic relevance determination (ARD)

$$k_{\text{SE}}(\boldsymbol{z},\boldsymbol{z}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{d_z}\frac{(z_i - z_i')}{2l_i^2}\right) \tag{2.18}$$

and Matérn class ARD kernels

$$k_{1/2}(\boldsymbol{z},\boldsymbol{z}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{d_z}\frac{|z_i - z_i'|}{l_i}\right), \tag{2.19}$$

$$k_{3/2}(\boldsymbol{z},\boldsymbol{z}') = \sigma_f^2\left(1+\sqrt{3}\sum_{i=1}^{d_z}\frac{|z_i - z_i'|}{l_i}\right)\exp\left(-\sqrt{3}\sum_{i=1}^{d_z}\frac{|z_i - z_i'|}{l_i}\right), \tag{2.20}$$

$$k_{5/2}(\boldsymbol{z},\boldsymbol{z}') = \sigma_f^2\left(1+\sqrt{5}\sum_{i=1}^{d_z}\frac{|z_i - z_i'|}{l_i} + \frac{5}{3}\sum_{i=1}^{d_z}\frac{|z_i - z_i'|^2}{l_i^2}\right)\exp\left(-\sqrt{5}\sum_{i=1}^{d_z}\frac{|z_i - z_i'|}{l_i}\right), \tag{2.21}$$

where the length scales $l_i \in \mathbb{R}_{0,+}$ and the signal variances $\sigma_f^2 \in \mathbb{R}_{0,+}$ are considered hyperparameters. Automatic relevance determination refers here to the fact that input dimensions i become irrelevant as the corresponding length scale l_i goes to ∞. Since ARD requires a comparatively high number of hyperparameters, these kernels can also be found without it, which simply means that a single length scale parameter is used for all input dimensions, i.e., $l_i = l_j$ for all $i,j = 1,\ldots,d_z$.

It is important to note that the complexity of GP regression differs between the feature representation (2.10), (2.11) and the kernel-based formulation (2.13), (2.16). While the direct inversion of $\boldsymbol{\Sigma}_w$ causes a cubic complexity in the number of features d_w, i.e., $\mathcal{O}(d_w^3)$, the complexity of (2.13), (2.16) depends cubically on the number of training samples N, i.e., $\mathcal{O}(N^3)$. Since the number of training samples N is often significantly larger than the number of features d_w, this means that it is usually advantageous to perform regression directly in the feature space using (2.10), (2.11). This is the case, e.g., for polynomial kernels. However, it is not always possible to revert the kernel trick for performing regression in the feature space. When the kernel defines infinite-dimensional features, the Morrison matrix inversion lemma cannot be used due to its restriction to finite-dimensional matrices, such that a feature space equivalent of (2.13), (2.16) cannot be straightforwardly obtained. This implies, for example, that SE and Matérn class kernels do not admit a finite, parametric representation in the form of (2.10) and (2.11) such that we refer to them as non-parametric kernels in the following. Therefore, both formulations (2.10), (2.11) and (2.13), (2.16) have a practical relevance.

Remark 2.1. *Although it is not possible to exactly revert the kernel trick, for every positive definite kernel $k(\cdot,\cdot)$, there exists a finite-dimensional feature vector $\boldsymbol{\phi}(\cdot)$ with sufficiently*

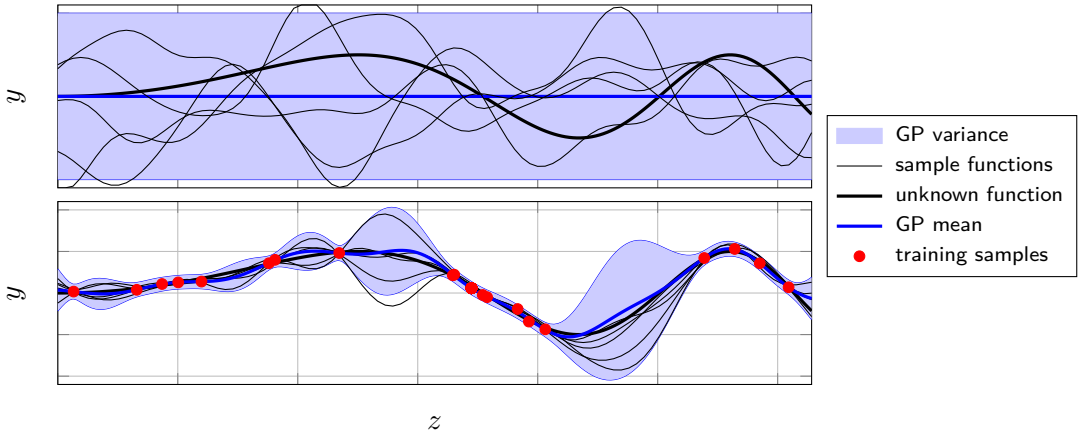

Figure 2.1.: Top: Prior GP distribution for a SE kernel together with sample functions. Bottom: GP distribution conditioned on training samples together with sample functions. The variance is illustrated through 3 times standard deviation intervals.

large dimension d_w, such that $\boldsymbol{\phi}^T(\cdot)\boldsymbol{\phi}(\cdot)$ is arbitrarily close to $k(\cdot,\cdot)$ [49, Proposition 2.11]. Therefore, (2.13), (2.16) *can be approximated arbitrarily well using* (2.10), (2.11), *which can be exploited to obtain computationally efficient approximations to non-parametric GP regression [50, 51].*

2.1.3. A Function Space View for Model Decomposition

While the weight space view on GP regression presented in Section 2.1.2 allows an intuitive interpretation as linear regression in potentially infinite-dimensional feature spaces, it does not provide insight into how structural properties of unknown functions can be represented in kernels. Therefore, we present a function space-oriented perspective on GP regression in this section, which focuses particularly on additive and multiplicative structures.

The function space view on GP regression is based on considering a GP as a distribution over functions as illustrated in Fig. 2.1. This is commonly denoted as $f(\cdot) \sim \mathcal{GP}(\mu^0(\cdot), k(\cdot,\cdot))$, which is simply a shorthand notation for

$$\mu^0(\boldsymbol{z}) = \mathbb{E}[f(\boldsymbol{z})], \tag{2.22}$$

$$k(\boldsymbol{z},\boldsymbol{z}') = \mathbb{E}[(f(\boldsymbol{z}) - \mu^0(\boldsymbol{z}))(f(\boldsymbol{z}') - \mu^0(\boldsymbol{z}'))] \tag{2.23}$$

together with the statement of a joint Gaussian distribution of $f(\boldsymbol{z})$ and $f(\boldsymbol{z}')$ for all $\boldsymbol{z},\boldsymbol{z}'$ as required by Definition 2.2. Therefore, the kernel is often referred to as covariance function. Due to these definitions, we can write down the joint distribution of training data and the function value $f(\boldsymbol{z})$ at an arbitrary test input $\boldsymbol{z}$ as

$$\begin{bmatrix} \boldsymbol{y}|\boldsymbol{Z} \\ f(\boldsymbol{z})|\boldsymbol{z} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}^0 \\ \mu^0(\boldsymbol{z}) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N & \boldsymbol{k}(\boldsymbol{z}) \\ \boldsymbol{k}^T(\boldsymbol{z}) & k(\boldsymbol{z},\boldsymbol{z}) \end{bmatrix} \right). \tag{2.24}$$

Since the posterior distribution of $f(\boldsymbol{z})$ can be obtained by conditioning on $\boldsymbol{y}$, we can determine it using standard properties of Gaussian distributions [22] resulting in

$$f(\boldsymbol{z})|\boldsymbol{z},\boldsymbol{Z},\boldsymbol{y} \sim \mathcal{N}(\mu(\boldsymbol{z}), \sigma^2(\boldsymbol{z})), \tag{2.25}$$

where the posterior mean $\mu(\boldsymbol{z})$ and variance $\sigma^2(\boldsymbol{z})$ are defined as

$$\mu(\boldsymbol{z}) = \mu^0(\boldsymbol{z}) + \boldsymbol{k}^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N\right)^{-1} (\boldsymbol{y} - \boldsymbol{\mu}^0), \tag{2.26}$$

$$\sigma^2(\boldsymbol{x}) = k(\boldsymbol{z}, \boldsymbol{z}) - \boldsymbol{k}^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{k}(\boldsymbol{z}). \tag{2.27}$$

Therefore, this approach directly leads to the non-parametric formulation of GP regression in (2.13), (2.16).

The advantages of this function space perspective become apparent when considering an unknown function with an additive structure $f(\cdot) = f_1(\cdot) + f_2(\cdot)$, where $f_1, f_2 : \mathbb{R}^{d_z} \to \mathbb{R}$ can be arbitrary functions. Since we know that two different functions are involved, we can put an individual GP prior on each function, i.e., $f_i(\cdot) \sim \mathcal{GP}(\mu_i^0(\cdot), k_i(\cdot, \cdot))$, with mean $\mu_i^0 : \mathbb{R}^{d_z} \to \mathbb{R}$ and covariance function $k_i : \mathbb{R}^{d_z} \times \mathbb{R}^{d_z} \to \mathbb{R}^{0,+}$, $i = 1, 2$. This allows us to consider the joint distribution of $f_i(\boldsymbol{z})$ and training targets $\boldsymbol{y}$ analogously to (2.24), which can be straightforwardly obtained as

$$\begin{bmatrix} \boldsymbol{y}|\boldsymbol{Z} \\ f_i(\boldsymbol{z})|\boldsymbol{z} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}^0 \\ \mu_i^0(\boldsymbol{z}) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N & \boldsymbol{k}_i(\boldsymbol{z}) \\ \boldsymbol{k}_i^T(\boldsymbol{z}) & k_i(\boldsymbol{z}, \boldsymbol{z}) \end{bmatrix}\right) \tag{2.28}$$

under the assumption of independence of $f_1(\cdot)$ and $f_2(\cdot)$, i.e., $\mathbb{E}[(f_1(\boldsymbol{z}) - \mu_1^0(\boldsymbol{z}))(f_2(\boldsymbol{z}') - \mu_2^0(\boldsymbol{z}'))] = 0$ [52]. Note that we use $k(\cdot, \cdot) = k_1(\cdot, \cdot) + k_2(\cdot, \cdot)$ and $\mu^0(\cdot) = \mu_1^0(\cdot) + \mu_1^0(\cdot)$ for all induced variables and functions without index i in the sequel (e.g., $\boldsymbol{K}$) while other variables and functions are defined via $k_i(\cdot, \cdot)$ and $\mu_i^0(\cdot)$ (e.g., $\boldsymbol{k}_i(\cdot)$). Due to the joint distribution (2.28), the posterior of $f_i(\cdot)$ can again be obtained using properties of Gaussian distributions, which results in the conditional distribution $f_i(\boldsymbol{z})|\boldsymbol{z}, \boldsymbol{Z}, \boldsymbol{y} \sim \mathcal{N}(\mu_{\text{add},i}(\boldsymbol{z}), \sigma_{\text{add},i}^2(\boldsymbol{z}))$ with

$$\mu_{\text{add},i}(\boldsymbol{z}) = \mu_i^0(\boldsymbol{z}) + \boldsymbol{k}_i^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N\right)^{-1} (\boldsymbol{y} - \boldsymbol{\mu}^0), \tag{2.29}$$

$$\sigma_{\text{add},i}^2(\boldsymbol{x}) = k_i(\boldsymbol{z}, \boldsymbol{z}) - \boldsymbol{k}_i^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{k}_i(\boldsymbol{z}). \tag{2.30}$$

Similarly, we can consider a function with multiplicative structure $f(\cdot) = f_1(\cdot) f_2(\cdot)$, where we assume $f_1(\cdot)$ to be unknown, while $f_2(\cdot)$ is known. Therefore, we can put a GP prior on $f_1(\cdot)$, i.e., $f_1(\cdot) \sim \mathcal{GP}(\mu_1^0(\cdot), k_1(\cdot, \cdot))$. This allows us to proceed analogously to the additive function structure leading to the posterior distribution $f_1(\boldsymbol{z})|\boldsymbol{z}, \boldsymbol{Z}, \boldsymbol{y} = \mathcal{N}(\mu_{\text{mult},i}(\boldsymbol{z}), \sigma_{\text{mult},i}^2(\boldsymbol{z}))$ with

$$\mu_{\text{mult},1}(\boldsymbol{z}) = \mu_1^0(\boldsymbol{z}) + \boldsymbol{k}_1^T(\boldsymbol{z}) \boldsymbol{F}_2 \left(\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N\right)^{-1} (\boldsymbol{y} - \boldsymbol{\mu}^0), \tag{2.31}$$

$$\sigma_{\text{mult},1}^2(\boldsymbol{x}) = k_1(\boldsymbol{z}, \boldsymbol{z}) - \boldsymbol{k}_1^T(\boldsymbol{z}) \boldsymbol{F}_2 \left(\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{F}_2 \boldsymbol{k}_1(\boldsymbol{z}), \tag{2.32}$$

where $\boldsymbol{F}_2 \in \mathbb{R}^{N \times N}$ is a diagonal matrix with elements $F_{2,n,n} = f_2(\boldsymbol{z}^{(n)})$, $n = 1, \ldots, N$. These examples clearly illustrate the advantages of the function space view, which admits the straightforward inference of individual components of an unknown function.

2.1.4. Multi-Output Gaussian Process Regression

In many problems, it is necessary to learn a multi-dimensional function $\boldsymbol{f} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_f}$. Probably the most common approach to address this problem using GPs is by considering a GP prior on each element $f_i(\cdot)$ of $\boldsymbol{f}(\cdot)$ individually, i.e., $f_i(\cdot) \sim \mathcal{GP}(\mu_i^0(\cdot), k_i(\cdot, \cdot))$

with mean $\mu_i^0 : \mathbb{R}^{d_z} \to \mathbb{R}$ and covariance function $k_i : \mathbb{R}^{d_z} \times \mathbb{R}^{d_z} \to \mathbb{R}^{0,+}$, $i = 1, \dots, d_f$. This allows to employ (2.26), (2.27) d_f times, once for each target dimension, and concatenate the resulting mean and variance functions into vectors $\boldsymbol{\mu}(\cdot) = [\mu_1(\cdot) \ \cdots \ \mu_{d_f}(\cdot)]^T$ and $\boldsymbol{\sigma}^2(\cdot) = [\sigma_1^2(\cdot) \ \cdots \ \sigma_{d_f}^2(\cdot)]^T$. Since the complexity of this approach scales linearly with the dimensionality of $\boldsymbol{f}(\cdot)$, it merely increases the total complexity to $\mathcal{O}(d_f N^3)$. However, it is not capable of exploiting possible correlations between the target dimensions to increase the data efficiency.

This limitation can be overcome by employing multi-dimensional kernel functions, which can be defined specifically for a given problem, e.g., via differential operators applied to scalar kernels [53]. Moreover, the approach of individual priors can be interpreted as a diagonal kernel matrix, which can be straightforwardly extended through multi-dimensional kernels of the form $\boldsymbol{H}\mathrm{diag}\big([k_1(\cdot,\cdot) \ \cdots \ k_{d_f}]\big)\boldsymbol{H}^T$, where $k_i(\cdot)$, $i = 1, \dots, d_f$ are arbitrary scalar kernel functions and $\boldsymbol{H} \in \mathbb{R}^{d_f \times d_f}$ are arbitrary matrices. This concept, known as coregionalization [54], leads to kernel structures consisting of multiplicative and additive elements, such that individual model components can be inferred analogously to (2.29)-(2.32) [55]. Therefore, the methods derived in this dissertation straightforwardly extend to these models. However, the increased data efficiency achieved by considering correlations between target dimensions generally requires the inversion of $d_f N \times d_f N$-dimensional matrices, such that the complexity of multi-output GP regression increases to $\mathcal{O}(d_f^3 N^3)$. Therefore, data efficiency and computational complexity have to be carefully balanced when employing GPs in multi-dimensional regression problems.

2.1.5. Hyperparameter Optimization

Although GP regression admits the closed-form expressions (2.26), (2.27) for the posterior mean and variance, these equations rely on the assumption of a known prior mean function $\mu^0(\cdot)$, kernel $k(\cdot,\cdot)$ and observation noise variance σ_{on}^2. However, $\mu^0(\cdot)$ and $k(\cdot,\cdot)$ often depend on parameters in practice, e.g., the length scales l_i and the signal variance σ_f^2 for SE and Matérn class kernels. Since these parameters can have a non-negligible impact on the regression performance, they are usually considered hyperparameters together with the observation noise variance σ_{on}^2.

While it is sometimes possible to choose hyperparameters based on prior knowledge about the unknown function [56], they usually have to be inferred from data. This can be done with methods such as cross-validation [42] and log-pseudo likelihood maximization [57], but probably the most common approach is log-likelihood maximization [22]. For this technique, we concatenate all hyperparameters into a vector $\boldsymbol{\vartheta} \in \mathbb{R}^{d_\vartheta}$ and note that the marginal likelihood

$$p(\boldsymbol{y}|\boldsymbol{Z}) = \int p(\boldsymbol{y}|f, \boldsymbol{Z}) p(f|\boldsymbol{Z}) \mathrm{d}f \tag{2.33}$$

is Gaussian. Moreover, its logarithm can be straightforwardly derived as [22]

$$\begin{aligned} \log\left(p(\boldsymbol{y}|\boldsymbol{Z})\right) = & \\ -\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu}^0)^T\left(\boldsymbol{K}+\sigma_{\mathrm{on}}^2\boldsymbol{I}_N\right)^{-1}(\boldsymbol{y}-\boldsymbol{\mu}^0) - \frac{1}{2}\log\left(\det\left(\boldsymbol{K}+\sigma_{\mathrm{on}}^2\boldsymbol{I}_N\right)\right) - \frac{N}{2}\log(2\pi), & \end{aligned} \tag{2.34}$$

which allows an intuitive interpretation. The first term represents the data fit, which becomes small if the prior mean evaluated at the training samples $\boldsymbol{\mu}^0$ is similar to the measured

targets $\boldsymbol{y}$. The second term can be considered a complexity measure, which penalizes, e.g., large observation noise variances σ_{on}^2, while the last term is a constant. Therefore, the log-likelihood (2.34) is a well-suited criterion for tuning the hyperparameters of GP regression. This results in the optimization problem

$$\boldsymbol{\vartheta}^* = \arg\max_{\boldsymbol{\vartheta} \in \mathbb{R}^{d_\vartheta}} \log\left(p(\boldsymbol{y}|\boldsymbol{Z})\right), \tag{2.35}$$

which is known as log-likelihood maximization. Despite the general non-convexity of this problem, it is commonly solved using gradient-based numerical optimization techniques [22].

2.2. Uniform Error Bounds for Learning in Reproducing Kernel Hilbert Spaces

Since uniform error bounds cannot be derived for arbitrary functions, it is necessary to restrict the admissible class of unknown functions $f(\cdot)$. A common choice for this restriction is a function space defined through the kernel used for regression, which is known as reproducing kernel Hilbert space (RKHS). This space contains a large class of functions for many kernels, e.g., all analytic functions on a compact set for SE kernels [58], such that it does not pose a severe limitation in practice.

Regression error bounds for functions in an RKHS have been investigated for a long time, initially going back to the field of scattered data approximation with radial basis functions [59, 60, 61]. These works consider interpolation, i.e., regression without observation noise, and derive deterministic error bounds for a certain class of kernels. The derived bounds depend on the so-called power function, which is equivalent to the GP posterior standard deviation [62]. Therefore, these results directly transfer to GP regression without observation noise. Extending the approaches from scattered data interpolation to noisy observations leads to the concept of kernel ridge regression [63], which formulates the regression as a regularized optimization problem. This problem has received considerable attention, and various error bounds have been derived, e.g., in dependence on $\mathcal{L}_2$ covering numbers [64, 65]. In particular, it is possible to recover the structure of interpolation error bounds and express the regression error bound in terms of the power function for bounded observation noise [66]. Since the results of kernel ridge regression are identical to the GP posterior mean under certain assumptions [22], these error bounds immediately extend to GP regression. When GP regression is directly considered for the derivation of uniform error bounds, information-theoretic approaches can be used [67, 68]. These bounds exhibit the advantage of admitting a straightforward asymptotic analysis, while comparatively tight finite data guarantees can also be obtained [69]. Moreover, these bounds can be flexibly adapted to different noise distributions by employing tail bounds.

In this section, we demonstrate the straightforward derivation of such information-theoretic uniform error bounds based on the weight space perspective on GP regression. Hence, we start by presenting an approach for deriving uniform error bounds for parametric GP regression in Section 2.2.1. In Section 2.2.2, show how these results can be generalized to the non-parametric GP regression formulation using bounds on the norm of the unknown function in the RKHS.

2.2.1. Guarantees through Prior Parameter Bounds

While the set of admissible functions is clearly defined through the features $\boldsymbol{\phi}(\cdot)$ for parametric GP regression as

$$\mathbb{H}_0 = \{f(\cdot) : \exists \boldsymbol{w} \in \mathbb{R}^{d_w} \text{ such that } f(\cdot) = \boldsymbol{w}^T \boldsymbol{\phi}(\cdot)\}, \tag{2.36}$$

the weights $\boldsymbol{w}$ of an unknown function $f(\cdot) = \boldsymbol{w}^T\boldsymbol{\phi}(\cdot)$ can be arbitrarily large without any further restrictions. This is problematic since for $w_i \to \infty$, no practically useful uniform error bound can be obtained. Therefore, we require the following bound on the unknown weights $\boldsymbol{w}$ and their prior distribution $\mathcal{N}(\boldsymbol{\mu}_w^0, \boldsymbol{\Sigma}_w^0)$.

Assumption 2.2. *The standardized prior error is bounded by a known constant $\bar{w} \in \mathbb{R}_{0,+}$, i.e.,*

$$\left\| \left(\boldsymbol{\Sigma}_w^0\right)^{-1} \left(\boldsymbol{\mu}_w^0 - \boldsymbol{w}\right) \right\| \leq \bar{w}. \tag{2.37}$$

This assumption does not just pose an upper bound on the weights $\boldsymbol{w}$, but it also takes into account the parameters of the prior distribution. This is achieved through the use of the inverse covariance matrix $(\boldsymbol{\Sigma}_w^0)^{-1}$ as scaling factor, which ensures that a large difference between the prior mean $\boldsymbol{\mu}_w^0$ and the unknown weights $\boldsymbol{w}$ does not cause a large bound $\bar{w}$ as long as the prior uncertainty is sufficiently large. Therefore, Assumption 2.2 allows small bounds $\bar{w}$ even when little knowledge about the true weights $\boldsymbol{w}$ is known a priori.

In order to derive a uniform error bound $\eta(\cdot)$ ensuring

$$\mathbb{P}\Big(|\mu(\boldsymbol{z}) - f(\boldsymbol{z})| \leq \eta(\boldsymbol{z}), \quad \forall \boldsymbol{z} \in \mathbb{S}\Big) \geq 1 - \delta \tag{2.1 revisited}$$

for parametric GP regression based on Assumption 2.2, we make use of the fact that $f(\cdot) \in \mathbb{H}_0$ allows us to express the concatenated training targets as $\boldsymbol{y} = \boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{Z}) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = [\epsilon^{(1)} \ \cdots \ \epsilon^{(N)}]^T$. This formulation directly leads to the following result.

Lemma 2.1. *Consider an unknown function $f(\cdot) \in \mathbb{H}_0$ and a prior Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_w^0, \boldsymbol{\Sigma}_w^0)$ on the weights $\boldsymbol{w}$, such that Assumption 2.2 is satisfied. Then, the model $\mu(\cdot) = \boldsymbol{\mu}_w^T\boldsymbol{\phi}(\cdot)$ admits a uniform error bound*

$$\eta(\boldsymbol{z}) = \bar{w} \left\|\boldsymbol{\Sigma}_w \boldsymbol{\phi}(\boldsymbol{z})\right\| + \frac{1}{\sigma_{\text{on}}^2} \left\|\boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{Z})\right\| \|\boldsymbol{\epsilon}\| \tag{2.38}$$

with probability 1 on $\mathbb{R}^{d_z}$, where $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ are defined via (2.4) and (2.5), respectively.

Proof. Due to the structure of the unknown function $f(\cdot)$ and the definition of the posterior mean $\boldsymbol{\mu}_w$ in (2.4), we have

$$\boldsymbol{\mu}_w = \boldsymbol{\Sigma}_w \left(\left(\boldsymbol{\Sigma}_w^0\right)^{-1} \boldsymbol{\mu}_w^0 + \frac{1}{\sigma_{\text{on}}^2} \boldsymbol{\Phi}(\boldsymbol{Z}) \left(\boldsymbol{\Phi}^T(\boldsymbol{Z})\boldsymbol{w} + \boldsymbol{\epsilon}\right) \right). \tag{2.39}$$

Since $\boldsymbol{\mu}_w^T\boldsymbol{\phi}(\cdot) = \boldsymbol{\phi}^T(\cdot)\boldsymbol{\mu}_w$, the triangle inequality yields

$$\begin{aligned} \|\boldsymbol{\mu}_w^T\boldsymbol{\phi}(\boldsymbol{z}) - f(\boldsymbol{z})\| \leq & \left| \boldsymbol{\phi}^T(\boldsymbol{z}) \left(\boldsymbol{\Sigma}_w \left(\left(\boldsymbol{\Sigma}_w^0\right)^{-1} \boldsymbol{\mu}_w^0 + \frac{1}{\sigma_{\text{on}}^2} \boldsymbol{\Phi}(\boldsymbol{Z})\boldsymbol{\Phi}^T(\boldsymbol{Z})\boldsymbol{w} \right) - \boldsymbol{w} \right) \right| \\ & + \frac{1}{\sigma_{\text{on}}^2} \left\|\boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{Z})\right\| \|\boldsymbol{\epsilon}\|. \end{aligned} \tag{2.40}$$

The first term can be reformulated since (2.5) implies

$$\frac{1}{\sigma_{\text{on}}^2}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{Z})\boldsymbol{\Phi}^T(\boldsymbol{Z}) = \boldsymbol{I}_{d_w} - \boldsymbol{\Sigma}_w\left(\boldsymbol{\Sigma}_w^0\right)^{-1}. \tag{2.41}$$

Substituting this expression in (2.40) results in

$$\|\boldsymbol{\mu}_w^T\boldsymbol{\phi}(\boldsymbol{z}) - f(\boldsymbol{z})\| \leq \left|\boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w\left(\boldsymbol{\Sigma}_w^0\right)^{-1}\left(\boldsymbol{\mu}_w^0 - \boldsymbol{w}\right)\right| + \frac{1}{\sigma_{\text{on}}^2}\left\|\boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{Z})\right\|\|\boldsymbol{\epsilon}\|, \tag{2.42}$$

such that Assumption 2.2 yields the uniform error bound (2.38). □

This lemma provides the intuitive result that a decrease in the posterior covariance $\boldsymbol{\Sigma}_w$ generally reduces the uniform error bound (2.38). However, Lemma 2.1 cannot be directly used in practice since it still depends on the unknown observation noise realizations $\boldsymbol{\epsilon}$. In order to overcome this limitation, we can employ additional assumptions on the observation noise, such as a Gaussian distribution, which is exploited in the following corollary.

Corollary 2.1. *Consider an unknown function $f(\cdot) \in \mathbb{H}_0$ and a prior Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_w^0, \boldsymbol{\Sigma}_w^0)$ on the weights $\boldsymbol{w}$, such that Assumption 2.2 is satisfied. Moreover, assume that the noise $\epsilon^{(n)}$, $n = 1, \ldots, N$, satisfies Assumption 2.1. Then, for every $\delta \in (0, 1)$, the model $\mu(\cdot) = \boldsymbol{\mu}_w^T\boldsymbol{\phi}(\cdot)$ admits a uniform error bound*

$$\eta(\boldsymbol{z}) = \bar{w}\,\|\boldsymbol{\Sigma}_w\boldsymbol{\phi}(\boldsymbol{z})\| + \sqrt{2\sqrt{N\log\left(\frac{1}{\delta}\right)} + 2\log\left(\frac{1}{\delta}\right) + N}\left\|\boldsymbol{\phi}^T(\boldsymbol{z})\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{Z})\right\| \tag{2.43}$$

with probability $1-\delta$ on $\mathbb{R}^{d_z}$, where $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ are defined via (2.4) and (2.5), respectively.

Proof. The squared norm of a N-dimensional normal random vector follows a chi-square distribution with N degrees of freedom. Hence. we have $\frac{\|\boldsymbol{\epsilon}\|^2}{\sigma_{\text{on}}^2} \sim \chi_N^2$. Due to [70], this guarantees a bounded norm

$$\|\boldsymbol{\epsilon}\|^2 \leq \left(2\sqrt{N\log\left(\frac{1}{\delta}\right)} + 2\log\left(\frac{1}{\delta}\right) + N\right)\sigma_{\text{on}}^2. \tag{2.44}$$

with probability of at least $1-\delta$, such that (2.43) immediately follows from Lemma 2.1. □

This result demonstrates that tail bounds can be directly employed to obtain a practically useful uniform error bound from Lemma 2.1. The probabilistic nature of these tail bounds is directly inherited by the error bound, such that its size is closely linked to the specified confidence parameter δ. Therefore, (2.43) exhibits the intuitive behavior of a growing error bound with higher reliability.

While probabilistic uniform error bounds are the best achievable guarantees for Gaussian noise, it is possible to obtain deterministic guarantees using different noise assumptions. For this purpose, we consider deterministically bounded noise similarly as in [66].

Assumption 2.3. *The observation noise is bounded by a known constant $\bar{\epsilon} \in \mathbb{R}_+$, i.e., $|\epsilon^{(n)}| \leq \bar{\epsilon}$ for all $n = 1, \ldots, N$.*

Since the deterministic bound $\bar{\epsilon}$ admits a straightforward bound for $\|\boldsymbol{\epsilon}\|$, we obtain the following uniform error bound.

Corollary 2.2. *Consider an unknown function $f(\cdot) \in \mathbb{H}_0$ and a prior Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_w^0, \boldsymbol{\Sigma}_w^0)$ on the weights $\boldsymbol{w}$, which satisfies Assumption 2.2. Moreover, assume that the noise $\epsilon^{(n)}$, $n = 1, \ldots, N$, satisfies Assumption 2.1. Then, for every $\delta \in (0, 1)$, the model $\mu(\cdot) = \boldsymbol{\mu}_w^T \boldsymbol{\phi}(\cdot)$ admits a uniform error bound*

$$\eta(\boldsymbol{z}) = \bar{w} \left\| \boldsymbol{\Sigma}_w \boldsymbol{\phi}(\boldsymbol{z}) \right\| + \frac{\sqrt{N}\bar{\epsilon}}{\sigma_{\text{on}}^2} \left\| \boldsymbol{\phi}^T(\boldsymbol{z}) \boldsymbol{\Sigma}_w \boldsymbol{\Phi}(\boldsymbol{Z}) \right\| \tag{2.45}$$

with probability 1 on $\mathbb{R}^{d_z}$, where $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ are defined via (2.4) and (2.5), respectively.

Proof. This result is a direct consequence of Lemma 2.1 and the fact that $\|\boldsymbol{\epsilon}\| \leq \sqrt{N}\bar{\epsilon}$ under Assumption 2.3. □

Therefore, Lemma 2.1 allows a flexible derivation of uniform error bounds for parametric GP regression.

2.2.2. From Parameter Bounds to Reproducing Kernel Hilbert Space Norms

While Assumption 2.2 is limited to parametric GP regression, Corollary 2.1 and Corollary 2.2 clearly demonstrate the flexibility with respect to different types of observation noise, which is enabled through the weight space perspective on GP regression. Therefore, we extend these results to non-parametric GP regression in this section.

Mercer's theorem [71] plays a key role in this analysis. Given a positive definite kernel $k(\cdot, \cdot)$, Mercer's theorem guarantees the existence of orthogonal features $\phi_i : \mathbb{R}^{d_z} \to \mathbb{R}$, $i = 1, \ldots, \infty$, such that we can express the kernel as

$$k(\boldsymbol{z}, \boldsymbol{z}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{z}) \phi_i(\boldsymbol{z}') \tag{2.46}$$

with some $\lambda_i \in \mathbb{R}$ and $\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{S} \subset \mathbb{R}^{d_z}$ in a compact set $\mathbb{S}$. Note that we can simply choose $\lambda_i = 0$ for $i > d_w$ when we have a parametric kernel, such that (2.46) can be interpreted as a guaranteed reversal of the kernel trick requiring possibly infinitely many features. This motivates a generalization of the set of admissible functions via

$$\mathbb{H}_0^{\mathbb{S}} = \{ f(\cdot) : \ \exists d_w \in \mathbb{N}, \boldsymbol{w} \in \mathbb{R}^{d_w} \text{ such that } f(\boldsymbol{z}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{z}), \forall \boldsymbol{z} \in \mathbb{S} \}, \tag{2.47}$$

which is identical to (2.36) for parametric kernels. The set $\mathbb{H}_0^{\mathbb{S}}$ is limited to functions expressible through an arbitrarily large but finite number of features d_w, which is in contrast to the requirement of possibly infinitely many features to represent the kernel $k(\cdot, \cdot)$ using (2.46). Therefore, we define an inner product $< f_1(\cdot), f_2(\cdot) >_k = \sum_{i=1}^{\infty} \frac{w_{1,i} w_{2,i}}{\lambda_i}$, where $w_{1,i}$ and $w_{2,i}$ denote the weights corresponding to functions $f_1(\cdot)$ and $f_2(\cdot)$. This allows us to consider the completion of $\mathbb{H}_0^{\mathbb{S}}$ under the norm $\|f(\cdot)\|_k = \sum_{i=1}^{\infty} \frac{w_i^2}{\lambda_i}$ induced by the inner product $< \cdot, \cdot >_k$, which we denote as $\mathbb{H}_k^{\mathbb{S}} = \overline{\mathbb{H}}_0^{\mathbb{S}}$. Due to the definition as a completion, the set $\mathbb{H}_k^{\mathbb{S}}$ contains functions with an infinite number of features that exhibit a sufficiently fast weight decay [22]. Hence, it is capable of reflecting the expressivity of non-parametric kernels.

Since $\mathbb{H}_k^{\mathbb{S}}$ is equipped with an inner product and complete by construction, it is a Hilbert space. Moreover, the orthogonality of features implies that for every function $f(\cdot) \in \mathbb{H}_k$, it

holds that $< k(\cdot, \boldsymbol{z}), f(\cdot) >_k = f(\boldsymbol{z})$ for all $\boldsymbol{z} \in \mathbb{S}$. Therefore, the kernel has the so-called reproducing property, which leads to $\mathbb{H}_k^{\mathbb{S}}$ being referred to as reproducing kernel Hilbert space (RKHS). It should be noted that the RKHS is unambiguously connected to its kernel and vice versa [72], even though this might not be visible from the presented derivation.

While it is generally not apparent from the kernel how the corresponding RKHS looks, it often comprises a large class of functions. Moreover, many properties of this relationship have been thoroughly investigated, such that the direct transfer of kernel attributes like differentiability to functions in the RKHS is well-known [22]. This has allowed the explicit formulation of the RKHS for certain kernels. For example, the RKHS of SE kernels corresponds to the space of analytic functions [58], while Matérn kernels induce Sobolev spaces as RKHS [62]. Even though the RKHS cannot correspond to the space of continuous functions [73], these examples clearly illustrate the large variety of functions admitted by the restriction to an RKHS.

In order to derive uniform error bounds similarly as in Lemma 2.1, the restriction to an RKHS itself is again not sufficient. However, the RKHS allows a direct extension of Assumption 2.2 through its norm.

Assumption 2.4. *The unknown function $f(\cdot)$ has a bounded norm in the RKHS $\mathbb{H}_k^{\mathbb{S}}$ attached to the kernel $k(\cdot, \cdot)$, i.e., $\|f(\cdot)\|_k \leq \Gamma$ for some $\Gamma \in \mathbb{R}_+$.*

Since it is possible to derive lower bounds that are non-decreasing with a growing number of data points, an upper bound Γ can be estimated with a sufficiently large number of training samples [74]. Moreover, the RKHS norm is strongly connected to the spectral properties of kernels. This can be illustrated particularly well for the following class of kernels, which includes SE and Matérn class ARD kernels.

Definition 2.3. *A kernel is called stationary if it is only a function of the difference of its arguments, i.e., $k(\boldsymbol{z}, \boldsymbol{z}') = k(\boldsymbol{z} - \boldsymbol{z}')$ with a slight abuse of notation. If it is only a function of the normed difference, i.e., $k(\boldsymbol{z}, \boldsymbol{z}') = k(\|\boldsymbol{z} - \boldsymbol{z}'\|)$, it is referred to as isotropic kernel.*

Since stationary kernels effectively depend only on a single argument, we can apply the Fourier transform $\mathcal{F}[\cdot]$ to them. As shown in [62], this allows us to formulate the RKHS norm of a function $f(\cdot)$ as

$$\|f(\cdot)\|_k^2 = \frac{1}{(2\pi)^{\frac{d_{\boldsymbol{z}}}{2}}} \int \frac{|\mathcal{F}[f(\cdot)](\boldsymbol{\omega})|^2}{\mathcal{F}[k(\cdot)](\boldsymbol{\omega})} \mathrm{d}\boldsymbol{\omega}. \tag{2.48}$$

Therefore, a bounded RKHS norm corresponds to a sufficiently fast decaying spectrum of the unknown function $f(\cdot)$. In fact, this allows us to pose an assumption on the spectrum of $f(\cdot)$, which guarantees a bounded RKHS norm. Note that a restriction to band-limited functions gives rise to Paley-Wiener spaces [75], which are RKHSs themselves. In these spaces, the RKHS norm can be shown to correspond to the $\mathcal{L}_2$ norm, such that it can directly be estimated from data [76].

Based on Assumption 2.4, we can generalize Lemma 2.1 to non-parametric kernels, as shown in the following proposition.

Proposition 2.1. *Consider an unknown function $f(\cdot) \in \mathbb{H}_k^{\mathbb{S}}$ satisfying Assumption 2.4 and a prior Gaussian process $\mathcal{GP}(0, k(\cdot, \cdot))$. Then, the posterior mean function $\mu(\cdot)$ defined in* (2.26) *admits a uniform error bound*

$$\eta(\boldsymbol{z}) = \beta\sigma(\boldsymbol{z}) \tag{2.49}$$

with probability 1 *on the compact set* $\mathbb{S} \subset \mathbb{R}^{d_z}$*, where* $\sigma(\cdot)$ *is defined via* (2.27) *and*

$$\beta = \Gamma + \frac{1}{\sigma_{\text{on}}} \sqrt{\boldsymbol{\epsilon}^T \boldsymbol{K} (\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N)^{-1} \boldsymbol{\epsilon}}. \tag{2.50}$$

Proof. In order to prove this proposition, we proceed analogously to the proof of Lemma 2.1 by employing the triangle inequality to obtain

$$|f(\boldsymbol{z}) - \mu(\boldsymbol{z})| \leq \left| f(\boldsymbol{z}) - \boldsymbol{k}^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N \right)^{-1} \boldsymbol{f} \right| + \left| \boldsymbol{k}^T(\boldsymbol{z}) \left(\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N \right)^{-1} \boldsymbol{\epsilon} \right|, \tag{2.51}$$

where $\boldsymbol{f}$ denotes the noise-free evaluations of the unknown function $f(\cdot)$ and $\boldsymbol{\epsilon}$ is the concatenation of all noise realizations. Due to [68], the right side of this inequality can be upper bounded, such that

$$|f(\boldsymbol{z}) - \mu(\boldsymbol{z})| \leq \|f(\cdot)\|_k \sigma(\boldsymbol{z}) + \frac{\sigma(\boldsymbol{z})}{\sigma_{\text{on}}} \sqrt{\boldsymbol{\epsilon}^T \boldsymbol{K} (\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N)^{-1} \boldsymbol{\epsilon}} \tag{2.52}$$

holds for all $\boldsymbol{z} \in \mathbb{S}$. Then, the uniform error bound (2.49) directly follows from Assumption 2.4. □

This result shows that the posterior standard deviation $\sigma(\cdot)$ takes over the role of the covariance matrix $\boldsymbol{\Sigma}_w$ in (2.38), while the RKHS norm bound Γ replaces the prior parameter error bound $\bar{w}$. Even though the observation noise now appears in a quadratic expression, it still admits the same procedure for getting a practically computable error bound presented in Section 2.2.1. For example, the following result can be obtained when dealing with bounded noise $\epsilon^{(n)}$.

Corollary 2.3. *Consider an unknown function* $f(\cdot) \in \mathcal{H}_k^{\mathbb{S}}$ *satisfying Assumption 2.4 and a prior Gaussian process* $\mathcal{GP}(0, k(\cdot,\cdot))$*. Moreover, assume that the noise* $\epsilon^{(n)}$*,* $n = 1, \ldots, N$*, satisfies Assumption 2.3. Then, the posterior mean function* $\mu(\cdot)$ *defined in* (2.26) *admits a uniform error bound* (2.49) *for*

$$\beta = \Gamma + \frac{\sqrt{N}\bar{\epsilon}}{\sigma_{\text{on}}} \sqrt{\|\boldsymbol{K} (\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N)^{-1}\|} \tag{2.53}$$

with probability 1 *on the compact set* $\mathbb{S} \subset \mathbb{R}^{d_z}$*.*

Proof. This result directly follows from Proposition 2.1 and the fact that $\|\boldsymbol{\epsilon}\| \leq \sqrt{N}\bar{\epsilon}$ due to Assumption 2.3 □

While this approach can also be applied to i.i.d. Gaussian noise [69], more sophisticated concentration inequalities considering the gram matrix $\boldsymbol{K}$ can admit tighter uniform error bounds [68]. Furthermore, the class of admissible observation noise distributions can be generalized as formalized in the following assumption.

Assumption 2.5. *The observation noise* $\epsilon^{(n)}$ *is i.i.d. sub-Gaussian with scaling constant* $\tilde{\sigma}_{\text{on}} \in \mathbb{R}_{0,+}$*, i.e.,*

$$\forall s \in \mathbb{R}: \quad \mathbb{E}\left[\exp\left(s\epsilon^{(n)}\right)\right] \leq \exp\left(\frac{s^2 \tilde{\sigma}_{\text{on}}^2}{2}\right). \tag{2.54}$$

This assumption effectively requires the tails of the observation noise distribution to behave similar to a normal distribution or decay even faster. This can also be seen in the equivalent characterization of sub-Gaussian distributions through the following concentration inequality [77]

$$\mathbb{P}\left(|\epsilon^{(n)}| \geq s\right) \leq 2\exp\left(-\frac{s^2}{2\tilde{\sigma}_{\mathrm{on}}^2}\right). \tag{2.55}$$

Therefore, sub-Gaussian distributions can be interpreted as a generalization of the normal distribution, including it as a special case.

Using this assumption, the following uniform error bound, which can be found in a similar form in [68, 69], can be derived as an extension of Proposition 2.1.

Theorem 2.1. *Consider an unknown function $f(\cdot) \in \mathcal{H}_k^{\mathbb{S}}$ satisfying Assumption 2.4 and a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$. Moreover, assume that the observation noise satisfies Assumption 2.5. Then, for every $\delta \in (0,1)$, the posterior mean function $\mu(\cdot)$ defined in* (2.26) *admits a uniform error bound* (2.49) *for*

$$\beta = \Gamma + \frac{\sqrt{2}\tilde{\sigma}_{\mathrm{on}}}{\sigma_{\mathrm{on}}}\sqrt{\frac{1}{2} + \log\left(\sqrt{\det\left(\boldsymbol{I}_N + \frac{1}{\sigma_{\mathrm{on}}^2}\boldsymbol{K}\right)}\right) + \log\left(\frac{1}{\delta}\right)} \tag{2.56}$$

with probability $1-\delta$ on the compact set $\mathbb{S} \subset \mathbb{R}^{d_z}$.

Proof. Since the assumptions of Proposition 2.1 are satisfied, we directly obtain (2.50), for which it remains to bound the second summand. To achieve this, note that

$$\boldsymbol{K}(\boldsymbol{K}+\sigma_{\mathrm{on}}^2\boldsymbol{I}_N)^{-1} = \frac{1+\frac{1}{N}}{\sigma_{\mathrm{on}}^2}\boldsymbol{K}\left(\frac{1+\frac{1}{N}}{\sigma_{\mathrm{on}}^2}\boldsymbol{K} + \left(1+\frac{1}{N}\right)\boldsymbol{I}_N\right)^{-1}. \tag{2.57}$$

Therefore, we define $\tilde{\boldsymbol{K}} = \frac{1+\frac{1}{N}}{\sigma_{\mathrm{on}}^2}\boldsymbol{K}$, such that it is straightforward to show

$$\begin{aligned}\tilde{\boldsymbol{K}}\left(\tilde{\boldsymbol{K}} + \left(1+\frac{1}{N}\right)\boldsymbol{I}_N\right)^{-1} &\leq \left(\tilde{\boldsymbol{K}} + \frac{1}{N}\boldsymbol{I}_N\right)\left(\tilde{\boldsymbol{K}} + \left(1+\frac{1}{N}\right)\boldsymbol{I}_N\right)^{-1} &(2.58)\\ &\leq \left(\left(\tilde{\boldsymbol{K}} + \frac{1}{N}\boldsymbol{I}_N\right)^{-1} + \boldsymbol{I}_N\right)^{-1}. &(2.59)\end{aligned}$$

Since the observation noise is independent of each other and the training inputs, this allows us to apply [68, Theorem 1], which guarantees

$$\boldsymbol{\epsilon}^T\left(\left(\tilde{\boldsymbol{K}}+\frac{1}{N}\boldsymbol{I}_N\right)^{-1}+\boldsymbol{I}_N\right)^{-1}\boldsymbol{\epsilon} \leq 2\tilde{\sigma}_{\mathrm{on}}^2\left(\log\left(\sqrt{\det\left(\left(1+\frac{1}{N}\right)\boldsymbol{I}_N+\tilde{\boldsymbol{K}}\right)}\right)+\log\left(\frac{1}{\delta}\right)\right). \tag{2.60}$$

Therefore, we obtain

$$\boldsymbol{K}(\boldsymbol{K}+\sigma_{\mathrm{on}}^2\boldsymbol{I}_N)^{-1} \leq 2\tilde{\sigma}_{\mathrm{on}}^2\left(\log\left(\sqrt{\det\left(\boldsymbol{I}_N+\frac{1}{\sigma_{\mathrm{on}}^2}\boldsymbol{K}\right)}\right)+\frac{N}{2}\log\left(1+\frac{1}{N}\right)+\log\left(\frac{1}{\delta}\right)\right) \tag{2.61}$$

by employing the definition of $\tilde{\boldsymbol{K}}$. Substituting this expression into (2.50) and noting that $\log(1+1/N) \leq 1/N$ finally proves (2.56). □

Due to the use of a dedicated concentration inequality for GP regression, this result offers a crucial advantage compared to other error bounds presented in this section: it does not have an explicit dependency on the number of training samples N. While it still implicitly depends on N through the determinant, information-theoretic methods can be used to derive kernel-specific bounds. This often yields a logarithmic growth with increasing number of training samples N for the scaling factor β in (2.56) [67]. Therefore, this bound enables an effective analysis of the asymptotic behavior of the error bound.

Remark 2.2. *Although not stating it explicitly, we have assumed a zero prior mean function $\mu^0(\cdot)$ in this section. This does, however, pose no practical restriction because we can always consider the prior error $\tilde{f}(\cdot) = f(\cdot) - \mu(\cdot)$ as an unknown function which we want to infer with GP regression. Thereby, all results immediately extend to the case of a non-zero prior mean function $\mu^0(\cdot)$.*

2.3. Bayesian Uniform Error Bounds

While the weight space view on GP regression allows the straightforward derivation of statistical uniform error bounds, the resulting guarantees for functions in reproducing kernel Hilbert spaces are agnostic of the probabilistic nature of GPs [67]. This leads to the problem that there is little to no justification that hyperparameter tuning methods relying on the probabilistic GP formulation, e.g., log-likelihood maximization, also improve error bounds. In order to overcome this limitation, we derive Bayesian uniform error bounds in this section, whose guarantees are inherently tied to the GP prior distribution.

Point-wise Bayesian error bounds for GP regression directly follow from tail bounds for Gaussian distributions [77], which has led to their wide-spread usage in control [78, 79, 80]. Due to their highly local nature, these bounds provide only theoretical value when the GP model is evaluated at a finite number of test points. While this is sufficient in some scenarios, e.g., when evaluated at a grid [67], it does not allow continuous domains as required for uniform error bounds. In order to overcome this limitation, additional assumptions on the continuity of the unknown function are necessary [81], such that interpolation between grid points can extend point-wise bounds to uniform error bounds. However, computing these bounds generally requires additional knowledge about the unknown function, e.g., a Lipschitz constant.

In this section, which is based on our work [35, 36], we straightforwardly derive a Bayesian uniform error bound by extending point-wise error bounds on a virtual grid to a compact domain. This approach allows us to recover the structure of RKHS-based uniform error bounds through a suitable choice of the virtual grid constant. Moreover, the Bayesian perspective can be viewed as a direct consequence of the function space view on GP regression, which enables us to prove uniform error bounds for additive and multiplicative model components. Finally, we show Hölder continuity for the GP mean $\mu(\cdot)$, standard deviation $\sigma(\cdot)$ and the unknown function $f(\cdot)$ itself, whose constants only depend on the prior GP distribution, such that the proposed uniform error bounds can be directly computed without any further assumptions.

The remainder of this section is structured as follows. In Section 2.3.1, the Bayesian uniform error bound for unstructured kernels is derived, which is extended to additive and multiplicative structures in Section 2.3.2. The Hölder continuity parameters of the posterior

mean $\mu(\cdot)$ and standard deviation $\sigma(\cdot)$ are derived in Section 2.3.3. In Section 2.3.4, a Lipschitz constant for the unknown function $f(\cdot)$ is shown, which follows from classical sample path properties of Gaussian processes.

2.3.1. Continuity-Based Error Bounds for Non-Parametric Regression

While the weight space view directly leads to the RKHS $\mathbb{H}_k^{\mathbb{S}}$ as the set of admissible functions, this restriction is not necessary from a Bayesian perspective. This is because GPs already induce such a hypothesis space via their prior. Therefore, assuming that the prior GP is suitable for the unknown function as formalized in the following assumption is sufficient for the derivation of Bayesian uniform error bounds.

Assumption 2.6. *The unknown function $f(\cdot)$ is a sample from a Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$.*

This assumption, which has similarly been used in, e.g., [67, 81], has a twofold implication. On the one hand, it specifies the admissible functions for regression via the space of sample functions, which depends on the employed kernel $k(\cdot,\cdot)$. For example, it is straightforward to see that polynomial kernels can be used to learn polynomial functions of the same degree. Moreover, it is well known that the sample space of GPs with SE kernel contains all continuous functions with probability one [58]. Therefore, choosing a suitable kernel for ensuring that the unknown function lies in the space of sample functions is usually not a challenging problem in practice. On the other hand, Assumption 2.6 induces a weighting between possible sample functions due to the GP probability density. Since we base the derivation of the uniform error bound on this weighting, an unknown function $f(\cdot)$ with low prior probability density would lead to sets $\{f'(\cdot) : |f'(\boldsymbol{z}) - \mu(\boldsymbol{z})| \leq \eta(\boldsymbol{z})\}$ with a high probability under the GP prior, even though they do not contain the unknown function $f(\cdot)$. Hence, the true function $f(\cdot)$ should have a high probability density under the GP prior. This can be efficiently achieved in practice using suitable kernel tuning methods, e.g., [82], or via a re-calibration of the probability distribution after training [83]. Therefore, ensuring a suitable prior distribution is not a severe limitation, such that Assumption 2.6 is not restrictive in practice.

Since the prior Gaussian process induces a probability distribution for each point in a compact set $\mathbb{S}$, we can discretize this set and exploit standard tail bounds for Gaussian distributions to obtain point-wise error bounds [67]. If all involved functions are continuous, we can straightforwardly extend these point-wise guarantees to a continuous domain $\mathbb{S}$. Therefore, we make the following assumption.

Assumption 2.7. *The posterior mean function $\mu(\cdot)$, the standard deviation $\sigma(\cdot)$, and the unknown function $f(\cdot)$ are Hölder continuous with order $p_\mu, p_\sigma, p_f \in \mathbb{R}_+$ and coefficients $L_\mu, L_\sigma, L_f \in \mathbb{R}_+$, respectively, on the compact set $\mathbb{S} \subset \mathbb{R}^{d_z}$.*

Hölder continuity of a function $f(\cdot)$ on a compact set $\mathbb{S}$ requires the existence of constants $p_f \in \mathbb{R}_+$, $L_f \in \mathbb{R}_+$ such that

$$|f(\boldsymbol{z}) - f(\boldsymbol{z}')| \leq L_f \|\boldsymbol{z} - \boldsymbol{z}'\|^{p_f} \quad \forall \boldsymbol{z}, \boldsymbol{z}' \in \mathbb{S}. \tag{2.62}$$

Therefore, Hölder continuity can be interpreted as a generalization of Lipschitz continuity, coinciding with it for $p_f = 1$[2]. Note that we only assume Hölder continuity at this point for ease of understanding, but we provide conditions guaranteeing the satisfaction of Assumption 2.7 in Section 2.3.3 and Section 2.3.4.

Based on these assumptions, it is straightforward to show the following result.

Proposition 2.2. *Consider an unknown function $f(\cdot)$, a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ satisfying Assumption 2.6, and training data $\mathbb{D}$ with observation noise $\epsilon^{(n)}$, $n = 1, \ldots, N$, fulfilling Assumption 2.1. If Assumption 2.7 holds, then, for every $\delta \in (0,1)$ and $\tau \in \mathbb{R}_+$, the posterior mean function $\mu(\cdot)$ defined in (2.26) admits a uniform error bound*

$$\eta(\boldsymbol{z}) = \tilde{\beta}\sigma(\boldsymbol{z}) + L_\mu \tau^{p_\mu} + L_f \tau^{p_f} + \tilde{\beta} L_\sigma \tau^{p_\sigma} \tag{2.63}$$

with probability $1-\delta$ on the compact set $\mathbb{S} \subset \mathbb{R}^{d_z}$, where

$$\tilde{\beta} = \sqrt{2\log\left(\frac{M(\tau, \mathbb{S})}{\delta}\right)} \tag{2.64}$$

and $M(\tau, \mathbb{S})$ denotes the τ-covering number of $\mathbb{S}$, i.e.,

$$M(\tau, \mathbb{S}) = \underset{\mathbb{S}_\tau \subset \mathbb{R}^{d_z}}{\arg\min} |\mathbb{S}_\tau| \tag{2.65}$$

$$\text{such that } \forall \boldsymbol{z} \in \mathbb{S}, \exists \boldsymbol{z}' \in \mathbb{S}_\tau : \|\boldsymbol{z} - \boldsymbol{z}'\| \leq \tau \tag{2.66}$$

Proof. We exploit the continuity properties of the posterior mean, variance and the unknown function to prove the probabilistic uniform error bound by exploiting the fact that for every grid $\mathbb{S}_\tau$ with $|\mathbb{S}_\tau|$ grid points and

$$\max_{\boldsymbol{z} \in \mathbb{S}} \min_{\boldsymbol{z}' \in \mathbb{S}_\tau} \|\boldsymbol{z} - \boldsymbol{z}'\| \leq \tau \tag{2.67}$$

it holds with probability of at least $1 - |\mathbb{S}_\tau| \mathrm{e}^{-s/2}$ that [67]

$$|f(\boldsymbol{z}) - \mu(\boldsymbol{z})| \leq \sqrt{s}\sigma(\boldsymbol{z}) \quad \forall \boldsymbol{z} \in \mathbb{S}_\tau. \tag{2.68}$$

Choose $s = 2\log\left(\frac{|\mathbb{S}_\tau|}{\delta}\right)$, then

$$|f(\boldsymbol{z}) - \mu(\boldsymbol{z})| \leq \sqrt{2\log\left(\frac{|\mathbb{S}_\tau|}{\delta}\right)}\sigma(\boldsymbol{z}) \quad \forall \boldsymbol{z} \in \mathbb{S}_\tau \tag{2.69}$$

holds with probability of at least $1-\delta$. Due to Hölder continuity of $f(\boldsymbol{z})$, $\mu(\boldsymbol{z})$ and $\sigma(\boldsymbol{z})$ we obtain

$$\min_{\boldsymbol{z}' \in \mathbb{S}_\tau} |f(\boldsymbol{z}) - f(\boldsymbol{z}')| \leq L_f \tau^{p_f} \quad \forall \boldsymbol{z} \in \mathbb{S}, \tag{2.70}$$

$$\min_{\boldsymbol{z}' \in \mathbb{S}_\tau} |\mu(\boldsymbol{z}) - \mu(\boldsymbol{z}')| \leq L_\mu \tau^{p_\mu} \quad \forall \boldsymbol{z} \in \mathbb{S}, \tag{2.71}$$

$$\min_{\boldsymbol{z}' \in \mathbb{S}_\tau} |\sigma(\boldsymbol{z}) - \sigma(\boldsymbol{z}')| \leq L_\sigma \tau^{p_\sigma} \quad \forall \boldsymbol{z} \in \mathbb{S}. \tag{2.72}$$

[2]For notational simplicity, L_f denotes the Lipschitz constant of f if we do not explicitly state the order p_f of Hölder continuity.

Moreover, the minimum number of grid points satisfying (2.67) is given by the covering number $M(\tau, \mathbb{S})$. Hence, we obtain

$$|f(\boldsymbol{z}) - \mu(\boldsymbol{z})| \leq \sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)}\sigma(\boldsymbol{z}) + L_\mu\tau^{p_\mu} + L_f\tau^{p_f} + L_\sigma\tau^{p_\sigma}\sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)} \quad (2.73)$$

for all $\boldsymbol{z} \in \mathbb{S}$ with probability $1-\delta$, which concludes the proof. □

The virtual grid constant τ used in (2.63) balances the error due to the interpolation and the inherent uncertainty measured by the posterior standard deviation $\sigma(\cdot)$. Therefore, the uncertainty-independent terms in (2.63) can be made arbitrarily small by choosing a sufficiently fine virtual grid. This, in turn, increases the covering number $M(\tau, \mathbb{S})$ and thus the scaling $\tilde{\beta}$ of the posterior standard deviation $\sigma(\cdot)$. However, this scaling depends merely logarithmically on τ such that even poor Hölder constants L_μ, L_σ and L_f can be easily compensated by small virtual grid constants τ.

While the flexibility of choosing arbitrary values for the virtual grid constant τ in Proposition 2.2 can be advantageous for some theoretical derivations, it also complicates the practical application of (2.63) due to an additional tuning parameter. Moreover, the existence of a constant offset, which is independent of $\sigma(\cdot)$, generally reduces the interpretability of (2.63). In order to mitigate these weaknesses, we couple τ with the posterior standard deviation $\sigma(\cdot)$ as shown in the following theorem.

Theorem 2.2. *Consider an unknown function $f(\cdot)$, a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ satisfying Assumption 2.6, and training data $\mathbb{D}$ with observation noise $\epsilon^{(n)}$, $n = 1, \ldots, N$, fulfilling Assumption 2.1. If Assumption 2.7 holds, then, for every $\delta \in (0,1)$ and $\tau \in \mathbb{R}_+$ with*

$$\delta \leq \frac{1}{\exp(\frac{1}{2})}, \quad (2.74)$$

$$L_\mu\tau^{p_\mu} + L_f\tau^{p_f} + L_\sigma\tau^{p_\sigma} \leq \min_{\boldsymbol{z}\in\mathbb{S}} \sigma(\boldsymbol{z}), \quad (2.75)$$

the posterior mean function $\mu(\cdot)$ defined in (2.26) admits a uniform error bound (2.49) for

$$\beta = 2\sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)}\sigma(\boldsymbol{z}) \quad (2.76)$$

with probability $1-\delta$ on the compact set $\mathbb{S} \subset \mathbb{R}^{d_z}$.

Proof. It directly follows from Proposition 2.2 that (2.76) ensures a uniform error bound (2.49) if

$$\sigma(\boldsymbol{z}) \geq \frac{L_\mu\tau^{p_\mu} + L_f\tau^{p_f}}{\sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)}} + L_\sigma\tau^{p_\sigma} \quad (2.77)$$

holds for all $\boldsymbol{z} \in \mathbb{S}$. Since the covering number $M(\tau, \mathbb{S})$ is trivially lower bounded by 1 for all τ, it follows that

$$2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right) \geq 2\log\left(\frac{1}{\delta}\right) \geq 1, \quad (2.78)$$

where the second inequality follows from (2.74). Therefore, (2.75) guarantees (2.77), which concludes the proof. □

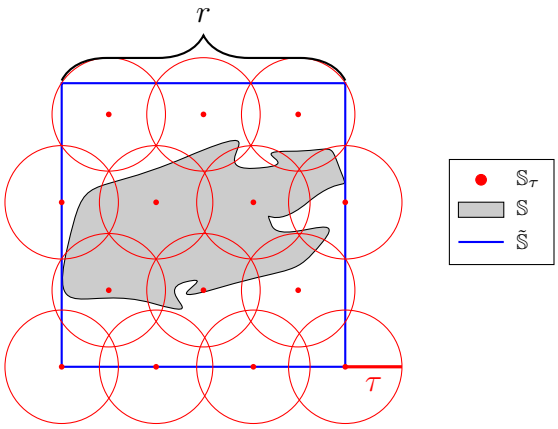

Figure 2.2.: Illustration of the derivation of an upper bound for the covering number $M(\tau, \mathbb{S})$.

While this theorem cannot completely resolve the problem of unspecified values for τ, the constraint (2.75) can be straightforwardly employed for determining one, e.g., by employing a line search to find the maximum value of τ satisfying (2.75). Moreover, if $\mu(\cdot)$, $\sigma(\cdot)$ and $f(\cdot)$ admit the same order of Hölder continuity, i.e., $p_f = p_\mu = p_\sigma$, condition (2.75) significantly simplifies, such that we can directly choose

$$\tau = \left(\frac{\min\limits_{\boldsymbol{z} \in \mathbb{S}} \sigma(\boldsymbol{z})}{L_\mu + L_\sigma + L_f} \right)^{\frac{1}{p_f}} . \tag{2.79}$$

Note that the upper bound for δ in (2.74) does not pose a practically restrictive condition since we are usually interested in small values $\delta \approx 0$.

Due to the absence of a constant offset, the Bayesian uniform error bound in Theorem 2.2 depends linearly on the posterior standard deviation $\sigma(\cdot)$. Therefore, it exhibits the same structure as the RKHS-based results in Proposition 2.1, Corollary 2.3 and Theorem 2.1 for a fixed data set, such that subsequential results derived using RKHS-based bounds directly transfer to the Bayesian setting and vice versa. Moreover, (2.76) can be straightforwardly computed in practice given the parameters of Hölder continuity since it only depends on the covering number $M(\tau, \mathbb{S})$. While the computation of exact covering numbers is a difficult problem for general sets $\mathbb{S}$, it can be easily upper bounded for Euclidean spaces as illustrated in Fig. 2.2. For this reason, we over-approximate the set $\mathbb{S}$ through a d-dimensional hypercube $\tilde{\mathbb{S}}$ with edge length r. Then, the covering number of $\tilde{\mathbb{S}}$ is bounded by [84]

$$M(\tau, \tilde{\mathbb{S}}) \leq \left(1 + \sqrt{d_z} \frac{\max\limits_{\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{S}} \|\boldsymbol{z} - \boldsymbol{z}'\|_\infty}{2\tau} \right)^{d_z} , \tag{2.80}$$

where $\|\cdot\|_\infty$ denotes the infinity norm. By construction, this is also a bound for the covering number of $\mathbb{S}$, i.e.,

$$M(\tau, \mathbb{S}) \leq \left(1 + \frac{\sqrt{d_z} \max\limits_{\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{S}} \|\boldsymbol{z} - \boldsymbol{z}'\|_\infty}{2\tau} \right)^{d_z} . \tag{2.81}$$

Therefore, (2.76) can be directly evaluated given the parameters of Hölder continuity. Note that similar approximations are also possible for other topological spaces such as the special Euclidean group SE(3) [85].

2.3.2. Uniform Error Bounds for Function Components

Since the Bayesian uniform error in Theorem 2.2 is completely based on the GP posterior distribution, we can directly apply it to posterior distributions of component functions as derived in Section 2.1.3. This is illustrated for additive functions in the following corollary.

Corollary 2.4. *Consider an unknown function $f(\cdot) = f_1(\cdot) + f_2(\cdot)$, prior Gaussian processes $\mathcal{GP}(0, k_i(\cdot,\cdot))$, $i = 1, 2$, satisfying Assumption 2.6, and training data $\mathbb{D}$ with observation noise $\epsilon^{(n)}$, $n = 1, \ldots, N$, fulfilling Assumption 2.1. If Assumption 2.7 individually holds for the component functions $f_1(\cdot)$, $f_2(\cdot)$ and the corresponding mean and variance functions defined in* (2.29),(2.30)*, then, for every $\delta \in (0, 1)$ and $\tau_i \in \mathbb{R}_+$ satisfying* (2.74) *and*

$$L_{\mu_i}\tau_i^{p_{\mu_i}} + L_f\tau_i^{p_{f_i}} + L_{\sigma_i}\tau_i^{p_{\sigma_i}} \leq \min_{\boldsymbol{z}\in\mathbb{S}} \sigma_i(\boldsymbol{z}), \tag{2.82}$$

the posterior mean functions $\mu_{\text{add},i}(\cdot)$, $i = 1, 2$, defined in (2.29) *admit uniform error bounds*

$$\eta_i(\boldsymbol{z}) = \beta_i \sigma_{\text{add},i}(\boldsymbol{z}) \tag{2.83}$$

with probability $1 - \delta$ on the compact set $\mathbb{S} \subset \mathbb{R}^{d_z}$ for

$$\beta_i = 2\sqrt{2\log\left(\frac{M(\tau_i, \mathbb{S})}{\delta}\right)}. \tag{2.84}$$

Proof. The uniform error bounds $\eta_i(\cdot)$, $i = 1, 2$ immediately follow from [52]

$$f_i(\boldsymbol{z})|\boldsymbol{z}, \boldsymbol{Z}, \boldsymbol{y} \sim \mathcal{N}(\mu_{\text{add},i}(\boldsymbol{z}), \sigma^2_{\text{add},i}(\boldsymbol{z})), \tag{2.85}$$

which allows a proof analogously to Theorem 2.2. □

It can be directly seen that the uniform error bound for an additive component only depends on the posterior mean $\mu_{\text{add},i}(\cdot)$ and the posterior standard deviation $\sigma_{\text{add},i}(\cdot)$ of the function $f_i(\cdot)$. Therefore, it naturally follows that condition (2.82) on the virtual grid constant τ_i also depends purely on the Hölder continuity parameters of these functions.

When the function has a multiplicative structure with a known component, we can analogously obtain the following error bound.

Corollary 2.5. *Consider an unknown function $f(\cdot) = f_1(\cdot)f_2(\cdot)$ with known function $f_2(\cdot)$, a prior Gaussian process $\mathcal{GP}(0, k_1(\cdot,\cdot))$, satisfying Assumption 2.6, and training data $\mathbb{D}$ with observation noise $\epsilon^{(n)}$, $n = 1, \ldots, N$, fulfilling Assumption 2.1. If Assumption 2.7 individually holds for the component function $f_1(\cdot)$ and the corresponding mean and variance function defined in* (2.31),(2.32)*, then, for every $\delta \in (0, 1)$ and $\tau_1 \in \mathbb{R}_+$ satisfying* (2.74) *and* (2.82)*, the posterior mean function $\mu_{\text{mult},1}(\cdot)$ defined in* (2.31) *admits a uniform error bound*

$$\eta_i(\boldsymbol{z}) = \beta_i \sigma_{\text{mult},i}(\boldsymbol{z}) \tag{2.86}$$

with probability $1-\delta$ *on the compact set* $\mathbb{S} \subset \mathbb{R}^{d_z}$ *for*

$$\beta_i = 2\sqrt{2\log\left(\frac{M(\tau_i, \mathbb{S})}{\delta}\right)}. \tag{2.87}$$

Proof. The uniform error bound $\eta_1(\cdot)$ immediately follows from [52]

$$f_1(\boldsymbol{z})|\boldsymbol{z}, \boldsymbol{Z}, \boldsymbol{y} \sim \mathcal{N}(\mu_{\text{mult},1}(\boldsymbol{z}), \sigma^2_{\text{mult},1}(\boldsymbol{z})), \tag{2.88}$$

which allows a proof analogously to Theorem 2.2. □

2.3.3. Hölder Continuity of Mean and Variance Functions

While the RKHS and its norm reflect the smoothness properties of the admissible unknown functions in the RKHS-based uniform error bounds, the Hölder continuity parameters take on this role in the Bayesian formulation. Therefore, it stands to reason that these parameters also inherit smoothness properties of the kernel $k(\cdot,\cdot)$. In order to formally show this, we define Hölder continuity of a kernel $k(\cdot,\cdot)$ with order $p_k \in \mathbb{R}_+$ and coefficient $L_k \in \mathbb{R}_+$ on a compact set $\mathbb{S}$ as

$$|k(\boldsymbol{z}, \boldsymbol{z}') - k(\boldsymbol{z}, \boldsymbol{z}'')| \leq L_k \|\boldsymbol{z}' - \boldsymbol{z}''\|^{p_k} \quad \forall \boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}'' \in \mathbb{S}. \tag{2.89}$$

Then, it is straightforward to show that Hölder continuity with order p_k and coefficient L_k is directly inherited from the kernel $k(\cdot,\cdot)$ due to the linear dependence of the GP mean function $\mu(\cdot)$ on the kernel vector $\boldsymbol{k}(\cdot)$.

Lemma 2.2. *Consider a prior GP* $\mathcal{GP}(0, k(\cdot,\cdot))$ *defined through a Hölder continuous kernel* $k(\cdot,\cdot)$ *with order* p_k *and coefficient* L_k*. Then, the posterior mean* $\mu(\cdot)$ *is Hölder continuous with order* p_k *and coefficient*

$$L_\mu \leq L_k \sqrt{N} \left\| (\boldsymbol{K} + \sigma^2_{\text{on}} \boldsymbol{I}_N)^{-1} \boldsymbol{y} \right\|. \tag{2.90}$$

Proof. The norm of the difference between the posterior mean $\mu(\boldsymbol{z})$ evaluated at two different points is given by

$$\|\mu(\boldsymbol{z}) - \mu(\boldsymbol{z}')\| = \left\| (\boldsymbol{k}(\boldsymbol{z}) - \boldsymbol{k}(\boldsymbol{z}')) (\boldsymbol{K} + \sigma^2_{\text{on}} \boldsymbol{I}_N)^{-1} \boldsymbol{y} \right\|. \tag{2.91}$$

Due to the Cauchy-Schwarz inequality and the Lipschitz continuity of the kernel we obtain

$$\|\mu(\boldsymbol{z}) - \mu(\boldsymbol{z}')\| \leq L_k \sqrt{N} \left\| (\boldsymbol{K} + \sigma^2_{\text{on}} \boldsymbol{I}_N)^{-1} \boldsymbol{y} \right\| \|\boldsymbol{z} - \boldsymbol{z}'\|^{p_k}, \tag{2.92}$$

which proves Lipschitz continuity of the mean $\mu(\boldsymbol{z})$. □

Due to the quadratic dependence of the posterior variance $\sigma^2(\cdot)$ on the kernel vector $\boldsymbol{k}(\cdot)$, the posterior standard deviation does not admit a derivation of the Hölder continuity parameters analogous to Lemma 2.2. However, we can exploit the relationship of the posterior standard deviation with a kernel-induced metric [86], which yields the following result for general Hölder continuous kernels $k(\cdot,\cdot)$.

Lemma 2.3. *Consider a prior GP $\mathcal{GP}(0, k(\cdot,\cdot))$ defined through a Hölder continuous kernel $k(\cdot,\cdot)$ with order p_k and coefficient L_k. Then, the posterior standard deviation $\sigma(\cdot)$ is Hölder continuous with order $p_k/2$ and coefficient*

$$L_\sigma \leq \sqrt{2L_k}. \tag{2.93}$$

Proof. The difference between two different evaluations of the posterior standard deviation is bounded by

$$|\sigma(\boldsymbol{z}) - \sigma(\boldsymbol{z}')| \leq d_k(\boldsymbol{z}, \boldsymbol{z}') \tag{2.94}$$

as shown in [86], where the kernel metric is defined as

$$d_k(\boldsymbol{z}, \boldsymbol{z}') = \sqrt{k(\boldsymbol{z}, \boldsymbol{z}) + k(\boldsymbol{z}', \boldsymbol{z}') - 2k(\boldsymbol{z}, \boldsymbol{z}')}. \tag{2.95}$$

Due to Hölder continuity of the kernel, we have

$$d_k(\boldsymbol{z}, \boldsymbol{z}') \leq \sqrt{2L_k \|\boldsymbol{z} - \boldsymbol{z}'\|^{p_k}}, \tag{2.96}$$

which concludes the proof. □

While this lemma provides suitable parameters for Proposition 2.2 and Theorem 2.2, it ensures Hölder continuity with a smaller order, which leads to a slower decay of the constant coefficient in Lemma 2.1. Under the additional restriction to stationary kernels, this weakness can be overcome, as shown in the following corollary.

Corollary 2.6. *Consider a prior GP $\mathcal{GP}(0, k(\cdot,\cdot))$ defined through a continuously differentiable kernel $k(\cdot,\cdot)$. Then, the posterior standard deviation $\sigma(\cdot)$ is Hölder continuous with order $p_k = 1$ and coefficient*

$$L_\sigma(\tau) = \sup_{\boldsymbol{z},\boldsymbol{z}' \in \mathbb{S}} \sqrt{\frac{1}{2k(\boldsymbol{0}) - 2k(\boldsymbol{z} - \boldsymbol{z}')}} \left\| \nabla k(\boldsymbol{z} - \boldsymbol{z}') \right\|. \tag{2.97}$$

Proof. For stationary kernels, we can express the kernel metric as

$$d_k(\boldsymbol{z}, \boldsymbol{z}') = d_k(\boldsymbol{z} - \boldsymbol{z}') = \sqrt{2k(\boldsymbol{0}) - 2k(\boldsymbol{z} - \boldsymbol{z}')}. \tag{2.98}$$

The simplified kernel metric is only a function of $\boldsymbol{z} - \boldsymbol{z}'$, such that the supremum of the norm of the derivative of $d_k(\cdot,\cdot)$ with respect to $\boldsymbol{z} - \boldsymbol{z}'$ is the Lipschitz constant of $\sigma(\cdot)$. This derivative directly follows from the chain rule of differentiation as

$$\nabla d_k(\boldsymbol{z} - \boldsymbol{z}') = \sqrt{\frac{1}{2k(\boldsymbol{0}) - 2k(\boldsymbol{z} - \boldsymbol{z}')}} \nabla k(\boldsymbol{z} - \boldsymbol{z}'), \tag{2.99}$$

which concludes the proof. □

While computing the coefficient L_σ requires the computation of a supremum in general, this optimization problem can be straightforwardly solved analytically for specific kernel choices. For example, it immediately follows from L'Hôpital's rule that

$$L_\sigma = \sigma_f \left\| \begin{bmatrix} \frac{1}{l_1} \\ \vdots \\ \frac{1}{l_d} \end{bmatrix} \right\| \tag{2.100}$$

for a SE ARD kernel. Since many practically employed kernels are Lipschitz continuous, which is identical to Hölder continuity with order $p_k = 1$, Corollary 2.6 consequently ensures that $p_\sigma = p_\mu = 1$. Therefore, it only remains to ensure Lipschitz continuity of the unknown function $f(\cdot)$ in order to define τ using (2.79).

2.3.4. Probabilistic Lipschitz Constants for Sample Functions

Due to the importance of Hölder continuity with $p_f = 1$ for employing (2.79), we focus on this special case, such that L_f corresponds to a Lipschitz constant. In order to derive a probabilistic Lipschitz constant L_f of the unknown function $f(\cdot)$ from the prior GP distribution, we exploit the fact that the derivative of a GP is again a GP. Therefore, Lipschitz constants can be obtained by adapting results from the well-studied theory of suprema of GPs. This yields the following lemma, which is based on the metric entropy criterion [87].

Lemma 2.4. *Consider a GP $\mathcal{GP}(0, k(\cdot,\cdot))$ with a continuously differentiable covariance function $k(\cdot,\cdot)$ and let L_k denote its Hölder continuity coefficient for order $p_k = 1$ on the compact set $\mathbb{S}$ which is included in a cube with edge length r. Then, the expected supremum of a sample function $f(\cdot)$ of this GP satisfies*

$$\mathbb{E}\left[\sup_{\boldsymbol{z}\in\mathbb{S}} f(\boldsymbol{z})\right] \leq 12\sqrt{6d_z}\max\left\{\max_{\boldsymbol{z}\in\mathbb{S}}\sqrt{k(\boldsymbol{z},\boldsymbol{z})}, \sqrt{rL_k}\right\}. \tag{2.101}$$

Proof. We prove this lemma by making use of the metric entropy criterion for the sample continuity of GPs [87]. This criterion allows to bound the expected supremum of a sample function $f(\cdot)$ by

$$\mathbb{E}\left[\sup_{\boldsymbol{z}\in\mathbb{S}} f(\boldsymbol{z})\right] \leq \int_0^{\max_{\boldsymbol{z}\in\mathbb{S}}\sqrt{k(\boldsymbol{z},\boldsymbol{z})}} \sqrt{\log(N_k(\varrho,\mathbb{S}))}\mathrm{d}\varrho, \tag{2.102}$$

where $N_k(\varrho,\mathbb{S})$ is the ϱ-packing number of $\mathbb{S}$ with respect to the kernel metric (2.95). Instead of bounding the ϱ-packing number, we bound the $\varrho/2$-covering number, which is known to be an upper bound of the packing number. The covering number can be easily bounded by transforming the problem of covering $\mathbb{S}$ with respect to the metric $d_k(\cdot,\cdot)$ into a coverage problem in the original metric of $\mathbb{S}$. For this reason, define

$$\psi(\varrho') = \sup_{\substack{\boldsymbol{z},\boldsymbol{z}'\in\mathbb{S}\\ \|\boldsymbol{z}-\boldsymbol{z}'\|_\infty\leq\varrho'}} d_k(\boldsymbol{z},\boldsymbol{z}'), \tag{2.103}$$

which is continuous due to the continuity of the covariance kernel $k(\cdot,\cdot)$. Consider the inverse function

$$\psi^{-1}(\varrho) = \inf\{\varrho' > 0 :\ \psi(\varrho') > \varrho\}. \tag{2.104}$$

Continuity of $\psi(\cdot)$ implies $\varrho = \psi(\psi^{-1}(\varrho))$. In particular, this means that we can guarantee $d_k(\boldsymbol{z},\boldsymbol{z}') \leq \frac{\varrho}{2}$ if $\|\boldsymbol{z}-\boldsymbol{z}'\| \leq \psi^{-1}(\frac{\varrho}{2})$. Due to this relationship it is sufficient to construct a uniform grid with grid constant $2\psi^{-1}(\frac{\varrho}{2})$ in order to obtain a $\varrho/2$-covering net of $\mathbb{S}$. Furthermore, the cardinality of this grid is an upper bound for the $\varrho/2$-covering number, such that we obtain

$$N_k(\varrho,\mathbb{S}) \leq \left\lceil \frac{r}{2\psi^{-1}(\frac{\varrho}{2})} \right\rceil^{d_z}. \tag{2.105}$$

Due to the Hölder continuity of the covariance function, we can bound $\psi(\cdot)$ by $\psi(\varrho') \leq \sqrt{2L_k\varrho'}$. Hence, the inverse function satisfies

$$\psi^{-1}\left(\frac{\varrho}{2}\right) \geq \left(\frac{\varrho}{2\sqrt{2L_k}}\right)^2 \tag{2.106}$$

and consequently

$$N_k(\varrho, \mathbb{S}) \leq \left(1 + \frac{4rL_k}{\varrho^2}\right)^{d_z} \tag{2.107}$$

holds, where the ceil operator is resolved through the addition of 1. Substituting this expression in the metric entropy bound (2.102) yields

$$\mathbb{E}\left[\sup_{\boldsymbol{z}\in\mathbb{S}} f(\boldsymbol{z})\right] \leq 12\sqrt{d_z} \int_0^{\max_{\boldsymbol{z}\in\mathbb{S}} \sqrt{k(\boldsymbol{z},\boldsymbol{z})}} \sqrt{\log\left(1+\frac{4rL_k}{\varrho^2}\right)}\mathrm{d}\varrho. \tag{2.108}$$

As shown in [88] this integral can be bounded by $\sqrt{6}\max\{\max_{\boldsymbol{z}\in\mathbb{S}}\sqrt{k(\boldsymbol{z},\boldsymbol{z})}, \sqrt{rL_k}\}$, which concludes the proof. □

While Lemma 2.4 provides a bound merely for the expected supremum of a sample function, a high probability bound for the supremum can be obtained using the Borell-TIS inequality [89]. This is shown in the following result.

Lemma 2.5. *Consider a GP $\mathcal{GP}(0, k(\cdot,\cdot))$ with a continuously differentiable covariance function $k(\cdot,\cdot)$ on the compact set $\mathbb{S}$ which is included in a cube with edge length r. Then, with probability $1-\delta_L$ the supremum of a sample function $f(\cdot)$ of this GP is bounded by*

$$f_{\sup}(\delta_L, k(\cdot,\cdot), r) = \sqrt{2\log\left(\frac{1}{\delta_L}\right)} \max_{\boldsymbol{z}\in\mathbb{S}} \sqrt{k(\boldsymbol{z},\boldsymbol{z})} + 12\sqrt{6d_z}\max\left\{\max_{\boldsymbol{z}\in\mathbb{S}}\sqrt{k(\boldsymbol{z},\boldsymbol{z})}, \sqrt{rL_k}\right\}. \tag{2.109}$$

Proof. We prove this lemma by exploiting the wide theory of concentration inequalities to derive a bound for the supremum of the sample function $f(\cdot)$. We apply the Borell-TIS inequality [89], which ensures for arbitrary $c \in \mathbb{R}_{0,+}$ that

$$P\left(\sup_{\boldsymbol{z}\in\mathbb{S}} f(\boldsymbol{z}) - \mathbb{E}\left[\sup_{\boldsymbol{z}\in\mathbb{S}} f(\boldsymbol{z})\right] \geq c\right) \leq \exp\left(-\frac{c^2}{2\max_{\boldsymbol{z}\in\mathbb{S}} k(\boldsymbol{z},\boldsymbol{z})}\right). \tag{2.110}$$

Due to Lemma 2.4 and the fact that continuous differentiability on a compact set implies Hölder continuity with $p_k = 1$, we can directly bound $E[\sup_{\boldsymbol{z}\in\mathbb{S}} f(\boldsymbol{z})]$. Therefore, the lemma follows from substituting (2.101) in (2.110) and choosing $c = \sqrt{2\log\left(1/\delta_L\right)}\max_{\boldsymbol{z}\in\mathbb{S}}\sqrt{k(\boldsymbol{z},\boldsymbol{z})}$. □

Since the derivatives of sample functions from GPs with sufficiently smooth kernels are sample functions of the derivative GPs [90], Lemma 2.5 directly allows to compute a high probability Lipschitz constant for the unknown function $f(\cdot)$ from the prior GP distribution. This is summarized in the following Theorem.

Theorem 2.3. *Consider a GP $\mathcal{GP}(0, k(\cdot,\cdot))$ defined through the covariance kernel $k(\cdot,\cdot)$ with continuous partial derivatives up to the fourth order and partial derivative kernels*

$$k^{\partial i}(\boldsymbol{z}, \boldsymbol{z}') = \frac{\partial^2}{\partial z_i \partial z_i'} k(\boldsymbol{z}, \boldsymbol{z}') \quad \forall i = 1, \ldots, d_z. \tag{2.111}$$

Then, a sample function $f(\cdot)$ of the GP is almost surely continuous on $\mathbb{S}$ and with probability of at least $1 - \delta_L$,

$$L_f \leq \hat{L}_f = \left\| \begin{bmatrix} f_{\sup}(\delta_L/2d_z, k^{\partial 1}(\cdot,\cdot), r) \\ \vdots \\ f_{\sup}(\delta_L/2d_z, k^{\partial d}(\cdot,\cdot), r) \end{bmatrix} \right\| \tag{2.112}$$

for $f_{\sup}(\cdot,\cdot,\cdot)$ defined in (2.109).

Proof. Continuity of the sample function $f(\boldsymbol{z})$ follows directly from [90, Theorem 5]. Furthermore, this theorem guarantees that the derivative functions $\frac{\partial}{\partial x_i} f(\boldsymbol{z})$ are samples from derivative Gaussian processes with covariance functions $k^{\partial i}(\boldsymbol{z}, \boldsymbol{z}')$. Therefore, we can apply Lemma 2.5 to each of the derivative processes and obtain with probability of at least $1 - \frac{\delta_L}{d}$

$$\sup_{\boldsymbol{z} \in \mathbb{S}} \left| \frac{\partial}{\partial z_i} f(\boldsymbol{z}) \right| \leq f_{\sup}(\delta_L/2d_z, k^{\partial i}(\cdot,\cdot), r). \tag{2.113}$$

Applying the union bound over all partial derivative processes $i = 1, \ldots, d_z$ finally yields the result. □

Since many practically employed kernels such as, e.g., the squared exponential and the Matern $5/2$, satisfy the required smoothness assumption of Theorem 2.3, it does not pose a severe restriction. Therefore, this theorem allows us to straightforwardly determine high-probability Lipschitz constants for the unknown function $f(\cdot)$, which can be directly used in Proposition 2.2 and Theorem 2.2, while barely requiring additional assumptions.

2.4. Discussion

This chapter provides a concise overview of GP regression focused on relevant aspects for this dissertation, but we do not claim that we even come close to presenting all the facets of Gaussian processes. In fact, many general aspects of GPs are not even mentioned, such as their combination with linear operators [91]. Furthermore, extensions with high relevance to the identification of models for control are not introduced, e.g., constrained GP regression [92] or the inference of GP models with control-theoretic properties such as stability [93, 94]. Therefore, we refer the interested reader to one of the many resources on Gaussian processes, e.g., [22, 62, 95].

To the best of our knowledge, such a resource does not exist for uniform GP error bounds. Even though many of the results on RKHS-based error bounds have been previously stated in a similar form [68, 69], we believe this is the first time that they are introduced completely using the weight space view, which allows an intuitive interpretation by establishing the connection to regularized linear regression. It is important to note that RKHS-based uniform error bounds have frequently been employed heuristically in early approaches for safe control

design, e.g., by simply using a constant scaling factor 2 for the GP standard deviation [96]. While this often seems to work, it has caused concerns about theoretical guarantees for safety-critical control applications, such that a more rigorous application of error bounds has started to emerge in the field of control theory in recent years [69]. However, the correct application strongly hinges on the knowledge of an upper bound for the RKHS norm of the unknown function, which is a controversially discussed assumption. While we have highlighted ideas to determine such bounds on the RKHS norm, this is still a largely unsolved problem in general, and it is unclear whether the RKHS norm bound assumption can indeed be verified using data. However, alternative approaches aiming for error bounds not relying on RKHS theory have been proposed, which might offer a solution to this issue [97].

While a similar problem holds for the prior GP distribution in our novel Bayesian error bounds in principle, it is less severe due to the existence of calibration methods [82, 83]. Therefore, obtaining a suitable prior distribution is usually not a crucial challenge. However, Bayesian error bounds provide guarantees only with respect to the prior distribution. This means that they do not allow a statistical interpretation in the sense that for $N_D \in \mathbb{N}$ different data sets of the same function, at most δN_D resulting GP models violate the error bound in expectation. It is straightforward to see that this kind of guarantee would only hold if the function would be sampled from the GP prior for every data set. Due to this interpretation, Bayesian error bounds also do not admit deterministic guarantees similar to Corollary 2.3. Therefore, both RKHS-based and Bayesian uniform error bounds have their limitations in practice, such that they should be chosen based on application-specific requirements.

Tracking Control with Gaussian Process Models

3.

Since Gaussian process regression exhibits a high flexibility and data efficiency, it is frequently employed for inferring models of unknown dynamical systems, which can be used to design control laws through a wide variety of approaches [98, 99]. A particular advantage of GP models for this application is their explicit uncertainty representation, which can be used to tune the behavior of control laws towards curiosity [100, 101] or cautiousness [102, 103]. Moreover, the existence of uncertainty-based uniform error bounds allows the derivation of theoretical guarantees for controllers designed using GP models, which is particularly important in safety-critical problems [21].

Due to the high practical relevance, we specifically consider the control problem of tracking a known reference trajectory $\boldsymbol{x}_{\mathrm{ref}} : \mathbb{R}_{0,+} \to \mathbb{R}^{d_x}$ with the state trajectory $\boldsymbol{x} : \mathbb{R}_{0,+} \to \mathbb{R}^{d_x}$ generated by an unknown or partially known dynamical system. Since we cannot achieve exact tracking without knowledge of the system dynamics in general, we are interested in quantifying the tracking accuracy. Due to the probabilistic guarantees from uniform error bounds for GP models, this leads to the following definition of suitable tracking error bounds.

Definition 3.1. *A closed-loop dynamical system admits a probabilistic tracking error bound* $\upsilon : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$*, if there exists a set* $\mathbb{X}_0 \subset \mathbb{X}$ *and a probability* $\delta \in (0, 1]$*, such that*

$$\mathbb{P}\left(\|\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{ref}}(t)\| \leq \upsilon(t),\ \forall t \geq 0\right) \geq 1 - \delta \tag{3.1}$$

is satisfied for all $\boldsymbol{x}(0) \in \mathbb{X}_0$*.*

Note that this definition does not require the global existence of a tracking error bound, but a non-empty set of initial conditions $\mathbb{X}_0$ is sufficient. This restriction to a local guarantee is necessary since the uniform error bounds presented in Chapter 2 are generally valid merely on a compact domain $\mathbb{S}$.

In this chapter, we derive tracking error bounds satisfying Definition 3.1 for two scenarios. First, we consider the special case of linear systems with a certain nonlinear perturbation for which a GP model is learned. As shown in Section 3.1, this allows the formulation of a linear differential equation, whose solution is an upper bound for the tracking error. In order to generalize these results, we consider stabilizing control laws designed using the certainty equivalence principle in 3.2. Based on a Lyapunov approach, we derive general tracking error bounds, which we show to be straightforwardly computable in practice.

3.1. Compensating Nonlinear Perturbations in Linear Control Systems

Although GP regression generally yields nonlinear models, it can be applied to linear control systems in various forms. One of the probably most straightforward approaches relies on a linearization of the GP mean function at a reference point, such that optimal and robust controllers can be synthesized using classical $\mathcal{H}_\infty$ design methods [104]. Since the derivative of a GP is a GP itself [90], the probability distribution can be considered in such an optimization-based design, which allows the derivation of Bayesian stability guarantees [105]. The knowledge of a linear system structure can also be directly encoded in the Gaussian process prior through a linear kernel, such that optimal controllers can be efficiently designed using scenario optimization [106]. A more sophisticated kernel design can even enable the computation of optimal controllers without inferring a system model by learning the cost surface of linear quadratic controllers [107]. Finally, GPs can be used to learn perturbations of a nominal linear system. This allows the derivation of stability guarantees both for time-dependent [108] and state-dependent nonlinearities [30, 109]. While these methods generally perform well, the derived guarantees are usually agnostic of the local model uncertainty but instead rely on a worst-case error perspective over the state space. Therefore, a local improvement of the GP model accuracy does not necessarily lead to an improvement of the guarantees.

We address this issue by deriving a novel, uncertainty-dependent error bound for tracking control systems, in which a GP model is used to compensate for an unknown nonlinear perturbation. Such nonlinearities, which can be considered a form of matched uncertainty [48], can be found in a wide range of applications ranging from underwater vehicles, where unmodeled hydrodynamic forces due to currents can appear [14], to physical human-robot interaction, where humans introduce generally unknown torques [10]. We formulate the error bound as a stable linear dynamical system, which is forced by the uniform error bound evaluated along the reference trajectory. Thereby, the local uncertainty is taken into account. Moreover, we demonstrate that improved bounds can be shown under additional assumptions on the dynamics, which we exemplarily illustrate for approximate feedback linearization. The applicability and effectiveness of the derived results are demonstrated in a simulation.

The remainder of this section, which is based on [36], is structured as follows. Section 3.1.1 formalizes the considered problem setting by specifying the system structure and the employed control law. A bound for general linear systems is derived in Section 3.1.2 before we show the improved bounds for approximately feedback linearized systems in Section 3.1.3. Finally, the derived theoretical results are evaluated on a numerical example in Section 3.1.4.

3.1.1. Problem Setting

We consider single-input linear dynamical systems with nonlinear input perturbation of the form[1]

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}(u + f(\boldsymbol{x})) \tag{3.2}$$

[1] Throughout this document, we drop explicit dependencies on the time t in differential equations for notational simplicity.

with initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0 \in \mathbb{X} \subseteq \mathbb{R}^{d_x}$ and scalar control input $u : \mathbb{R}_{0,+} \to \mathbb{U} \subseteq \mathbb{R}$. The matrix $\boldsymbol{A} \in \mathbb{R}^{d_x \times d_x}$ and vector $\boldsymbol{b} \in \mathbb{R}^{d_x}$ are assumed to be known, while we consider $f : \mathbb{X} \to \mathbb{R}$ to be an unknown nonlinearity. This system structure covers a wide range of practical systems and can represent, e.g., systems controlled via approximate feedback linearization [31] or backstepping controllers for certain classes of dynamics [110]. Note that we merely consider the restriction to single-input systems for notational convenience, but our derived results can be easily generalized to multi-input dynamics.

The considered task is to track a bounded reference trajectory $\boldsymbol{x}_{\text{ref}} : \mathbb{R}_{0,+} \to \mathbb{R}^{d_x}$ with the state $\boldsymbol{x}(t)$. In order to enable the accurate tracking of the reference trajectory $\boldsymbol{x}_{\text{ref}}(\cdot)$, we restrict ourselves to references of the form

$$\dot{\boldsymbol{x}}_{\text{ref}} = \boldsymbol{A}\boldsymbol{x}_{\text{ref}} + \boldsymbol{b}r_{\text{ref}}, \tag{3.3}$$

where $r_{\text{ref}} : \mathbb{R}_{0,+} \to \mathbb{R}$ is a reference signal. Moreover, we require the following assumption on $\boldsymbol{A}$ and $\boldsymbol{b}$.

Assumption 3.1. *The pair $(\boldsymbol{A}, \boldsymbol{b})$ is controllable.*

This assumption is a common requirement in linear systems theory since it guarantees that a linear dynamical system can be stabilized [1, 111]. It is satisfied by many linear systems and linear dynamics obtained through nonlinear control techniques such as feedback linearization, such that it is generally unrestrictive in practice.

Based on Assumption 3.1, we can employ a linear control law

$$u(t) = -\boldsymbol{\theta}^T(\boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t)) + r_{\text{ref}}(t) - \hat{f}(\boldsymbol{x}(t)) \tag{3.4}$$

for tracking the reference trajectory $\boldsymbol{x}_{\text{ref}}(\cdot)$, where $\boldsymbol{\theta} \in \mathbb{R}^{d_x}$ is a control gain vector and $\hat{f} : \mathbb{X} \to \mathbb{R}$ is a model of the unknown nonlinear perturbation $f(\cdot)$. In order to obtain this model, we assume to learn it from measurements $(\boldsymbol{z}^{(n)} = \boldsymbol{x}^{(n)}, y^{(n)} = f^{(n)} + \epsilon^{(n)})$, $n = 1, \ldots, N$, with observation noise $\epsilon^{(n)}$ using Gaussian process regression as explained in Section 2.1. Therefore, we can use $\hat{f}(\boldsymbol{x}) = \mu(\boldsymbol{x})$ in the control law (3.4), such that we can quantify the error of the control input $\hat{f}(\boldsymbol{x})$ compared to the exact but unknown perturbation $f(\boldsymbol{x})$ using uniform error bounds. This is formalized in the following assumption.

Assumption 3.2. *The error between the GP mean function $\mu(\cdot)$ and the unknown function $f(\cdot)$ is uniformly bounded by a function $\eta(\cdot)$ on a compact domain $\mathbb{S} \subset \mathbb{X}$ with probability $1 - \delta$.*

Uniform error bounds satisfying this assumption can be directly obtained using, e.g., Theorem 2.1 under Assumptions 2.4 and 2.5, or Theorem 2.2 under Assumptions 2.1, 2.6 and 2.7. Therefore, Assumption 3.2 is merely used as a placeholder to straighten the presentation.

Due to the employment of the model $\hat{f}(\cdot) = \mu(\cdot)$ in the controller (3.4), the nonlinearity $f(\cdot)$ is generally not completely compensated. This leads to closed-loop dynamics of the tracking error $\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t)$ given by

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta})\boldsymbol{e} + \boldsymbol{b}(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})), \tag{3.5}$$

where $\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta}) = \boldsymbol{A} - \boldsymbol{b}\boldsymbol{\theta}^T$. In order to simplify the presentation in the subsequent sections, we make the following assumption on the diagonalizability of the matrix $\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta})$.

Assumption 3.3. *The matrix $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ has distinct eigenvalues.*

Since the pair $(\boldsymbol{A}, \boldsymbol{b})$ is assumed to be controllable, the eigenvalues of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ are a design choice depending solely on the control gain vector $\boldsymbol{\theta}$. Therefore, this assumption can be easily satisfied using control design techniques such as pole placement [111], such that it is not restrictive in practice. Note that all results in subsequent sections can be generalized to non-diagonalizable matrices using Jordan blocks [112], which underlines the role of Assumption 3.3 for straightening the exposition.

Since (3.5) does generally not converge to 0 even for matrices $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ with negative eigenvalues, we can only aim to guarantee that the state $\boldsymbol{x}$ remains in a neighborhood of the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$. Hence, we consider the problem of quantifying the tracking accuracy using probabilistic error bounds as formalized in Definition 3.1.

3.1.2. General Linear Tracking Control Systems

Due to the structure of the error dynamics (3.5), the tracking error crucially depends on the eigenvalues of the matrix $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$. This can be straightforwardly shown using Assumption 3.3, which allows the computation of the eigendecomposition $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix consisting of the eigenvalues of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$. Based on this decomposition, a dynamic bound for the tracking error $\boldsymbol{e}(\cdot)$ can be derived inspired by the comparison principle [113], as shown in the following theorem.

Theorem 3.1. *Consider a linear system* (3.2) *satisfying Assumption 3.1, which is controlled by* (3.4) *with control gains $\boldsymbol{\theta}$ satisfying Assumption 3.3. Assume that a Gaussian process is used to learn a model $\hat{f}(\cdot) = \mu(\cdot)$ of $f(\cdot)$, such that Assumption 3.2 holds on a compact set $\mathbb{S} \subset \mathbb{X}$ with a uniform error bound $\eta(\cdot)$ which admits a Lipschitz constant L_η. Then, the closed-loop system* (3.5) *admits a probabilistic tracking error bound $\upsilon(\cdot)$, if $\mathbb{B}_{\upsilon(t)}(\boldsymbol{x}_{\mathrm{ref}}(t)) \subset \mathbb{S}$ holds for all $t \in \mathbb{R}_{0,+}$, where $\upsilon(\cdot) = \xi(\cdot)$ is the solution of the linear dynamical system*

$$\dot{\xi} = \left(\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) + L_\eta \|\boldsymbol{U}\| \|\boldsymbol{U}^{-1}\boldsymbol{b}\|\right) \xi + \|\boldsymbol{U}\| \|\boldsymbol{U}^{-1}\boldsymbol{b}\| \eta(\boldsymbol{x}_{\mathrm{ref}}) \tag{3.6}$$

with initial condition $\xi(0) = \|\boldsymbol{U}\| \|\boldsymbol{U}\boldsymbol{e}(0)\|$.

Proof. Due to the error dynamics in (3.5), its solution is given by

$$\boldsymbol{e}(t) = \mathrm{e}^{\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})t}\boldsymbol{e}(0) + \int_0^t \mathrm{e}^{\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})(t-t')}\boldsymbol{b} f_e(t')\mathrm{d}t', \tag{3.7}$$

where $f_e(t) = f(\boldsymbol{x}(t)) - \mu(\boldsymbol{x}(t))$. Therefore, we directly obtain

$$\|\boldsymbol{e}(t)\| \leq \|\mathrm{e}^{\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})t}\boldsymbol{e}(0)\| + \int_0^t \|\mathrm{e}^{\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})(t-t')}\boldsymbol{b}\| |\bar{f}_e(t')| \mathrm{d}t', \tag{3.8}$$

where $\bar{f}_e(t)$ can be any function such that $|f_e(t)| \leq \bar{f}_e(t)$. Using the eigendecomposition of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$, it can be directly seen that

$$\|\mathrm{e}^{\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})t}\boldsymbol{b}\| \leq \|\boldsymbol{U}\| \|\boldsymbol{U}^{-1}\boldsymbol{b}\| \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))t}. \tag{3.9}$$

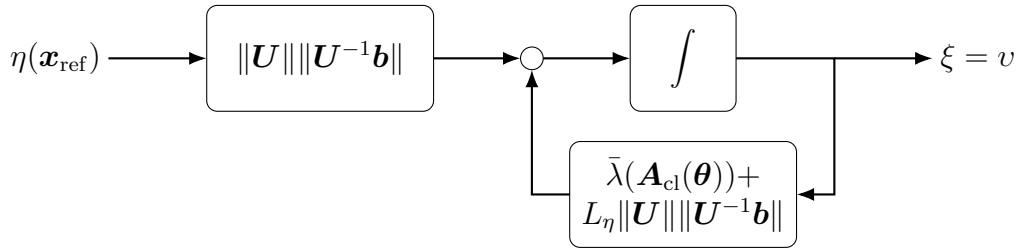

Figure 3.1.: Illustration of the tracking error bound for linear systems with input perturbations as a dynamical system forced by the uniform GP error bound.

Hence, we obtain

$$\|\boldsymbol{e}(t)\| \leq \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{e}(0)\|\mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))t} + \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\| \int_0^t \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))(t-t')}|f_e(t')|\mathrm{d}t'. \tag{3.10}$$

The right-hand side of this inequality is again the solution of a differential equation such that $\|\boldsymbol{e}(t)\| \leq \tilde{\xi}$ for

$$\dot{\tilde{\xi}} = \bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))\tilde{\xi} + \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|\bar{f}_e \tag{3.11}$$

with $\tilde{\xi}(0) = \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{e}(0)\|$. It remains to derive a bound $\bar{f}_e(t)$ for $|f_e(t)|$ in (3.11). Due to Assumption 3.2, it holds that $|f_e(t)| \leq \eta(\boldsymbol{x}(t))$ for all $\boldsymbol{x}$ in $\mathbb{S}$ with probability of at least $1-\delta$. Moreover, we have $\eta(\boldsymbol{x}(t)) \leq \eta(\boldsymbol{x}_{\mathrm{ref}}(t)) + L_\eta\|\boldsymbol{e}(t)\|$ due to Lipschitz continuity of $\eta(\cdot)$ guaranteed by Assumption 3.2. Therefore, it follows that

$$\dot{\tilde{\xi}} \leq \left(\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) + L_\eta\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|\right)\tilde{\xi} + \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|\eta(\boldsymbol{x}_{\mathrm{ref}}), \tag{3.12}$$

which concludes the proof. □

Since $\eta(\boldsymbol{x}_{\mathrm{ref}}(t))$ can be directly computed at any time instant, determining the tracking error bound using Theorem 3.1 simply requires simulating the linear dynamical system (3.6) as illustrated in Fig. 3.1. This can be straightforwardly done for a given time horizon in contrast to similar prior approaches [31, 35], where the uniform error bound needs to be determined at the actual system state $\boldsymbol{x}(t)$. In order to achieve this improved practical applicability, additional requirements on the stability of the linear dynamics described by $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ are necessary. It is obvious that $\upsilon(\cdot)$ is unbounded if the linear dynamics (3.6) are unstable, such that eventually $\mathbb{S} \subset \mathbb{B}_{\upsilon(t)}(\boldsymbol{x}_{\mathrm{ref}}(t))$ holds for sufficiently large t, i.e., the set inclusion is reversed. Therefore, (3.6) effectively poses the upper bound

$$\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) < -L_\eta\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|. \tag{3.13}$$

on the eigenvalues of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$, which effectively corresponds to a lower bound on the magnitude of admissible control gains $\boldsymbol{\theta}$. Note that except for (3.13), $\mathbb{B}_{\upsilon(t)}(\boldsymbol{x}_{\mathrm{ref}}(t)) \subset \mathbb{S}$ does usually not cause severe restrictions in practice, but merely requires the choice of a sufficiently large set $\mathbb{S}$ for a given set of initial conditions $\mathbb{X}_0$.

While Theorem 3.1 provides an accurate bound for the tracking error depending on the local data density, it is sometimes beneficial to have a time-independent tracking error bound. Therefore, we bound the maximum tracking error along the reference trajectory, as shown in the following proposition.

Proposition 3.1. *Consider a linear system* (3.2) *satisfying Assumption 3.1, which is controlled by* (3.4) *with control gains* $\boldsymbol{\theta}$ *satisfying Assumption 3.3. Assume that a Gaussian process is used to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *of* $f(\cdot)$*, such that Assumption 3.2 holds on a compact set* $\mathbb{S} \subset \mathbb{X}$ *with a uniform error bound* $\eta(\cdot)$ *which admits a Lipschitz constant* L_η*. Then, for* $\mathbb{X}_0 = \{\boldsymbol{x}_{\mathrm{ref}}(0)\}$*, the closed-loop system admits a probabilistic tracking error bound* $\upsilon(\cdot) = \bar{\upsilon}$*, if* (3.13) *is satisfied and* $\mathbb{B}_{\bar{\upsilon}}(\boldsymbol{x}_{\mathrm{ref}}(t)) \subset \mathbb{S}$ *holds for all* $t \in \mathbb{R}_{0,+}$*, where*

$$\bar{\upsilon} = -\frac{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) + L_\eta\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|} \sup_{t\geq 0} \eta(\boldsymbol{x}_{\mathrm{ref}}(t)). \tag{3.14}$$

Proof. It immediately follows from (3.7) that

$$\|\boldsymbol{e}(t)\| \leq \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\| \int_0^t \mathrm{e}^{\left(\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) + L_\eta\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|\right)(t-t')}\mathrm{d}t' \sup_{0\leq t'\leq t} \eta(\boldsymbol{x}_{\mathrm{ref}}(t')). \tag{3.15}$$

Since the integral can be straightforwardly calculated, we obtain

$$\sup_{t\geq 0} \|\boldsymbol{e}(t)\| \leq -\frac{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\| \sup_{t\geq 0} \eta(\boldsymbol{x}_{\mathrm{ref}}(t))}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) + L_\eta\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}, \tag{3.16}$$

which concludes the proof. □

Note that the restriction to a zero initial condition is only considered to simplify the derivation, but the extension to non-zero initial conditions is straightforward. Moreover, the additional requirement of Lipschitz continuity for the uniform error bound $\eta(\cdot)$ is not restrictive in practice since the error bounds derived in Chapter 2 depend linearly on the GP standard deviation $\sigma(\cdot)$. This directly allows us to apply, e.g., Corollary 2.6, in order to compute the necessary Lipschitz constant L_η. Therefore, the assumptions of Proposition 3.1 are not significantly more restrictive than those of Theorem 3.1.

3.1.3. Approximately Feedback Linearized Systems

While Theorem 3.1 and Proposition 3.1 are applicable to a wide class of linear systems, it is possible to derive tighter tracking error bounds with additional assumptions on the matrix $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$. We exemplarily demonstrate this for systems (3.2) with a structure

$$\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ -k_c\tilde{\theta}_1 & -k_c\tilde{\theta}_2 - \tilde{\theta}_1 & \cdots & -k_c - \tilde{\theta}_{d_x-1} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \tag{3.17}$$

where the control gain vector is defined as $\boldsymbol{\theta} = k_c[\tilde{\boldsymbol{\theta}}^T \ 1]^T + [0 \ \tilde{\boldsymbol{\theta}}^T]^T$ for a Hurwitz vector $\tilde{\boldsymbol{\theta}} \in \mathbb{R}^{d_x-1}$ and a scalar gain $k_c \in \mathbb{R}_+$. Systems (3.2) with this structure can be obtained when approximately feedback linearizing nonlinear dynamics [31], such that matrices $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ of the form (3.17) can be frequently found.

Due to the structure of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$, we can perform a coordinate transform and define $\tilde{e}_1(t) = \tilde{\boldsymbol{\theta}}^T \tilde{\boldsymbol{e}}_2(t) + e_{d_x}(t)$, $\tilde{\boldsymbol{e}}_2(t) = [e_1(t) \ \cdots \ e_{d_x-1}(t)]^T$. In these new coordinates, the dynamics (3.2) can be equivalently expressed as

$$\dot{\tilde{e}}_1 = -k_c \tilde{e}_1 + f(\boldsymbol{x}) - \mu(\boldsymbol{x}), \tag{3.18}$$

$$\dot{\tilde{\boldsymbol{e}}}_2 = \tilde{\boldsymbol{A}} \tilde{\boldsymbol{e}}_2 + \tilde{\boldsymbol{b}} \tilde{e}_1 \tag{3.19}$$

where

$$\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}}) = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ -\tilde{\theta}_1 & -\tilde{\theta}_2 & \cdots & -\tilde{\theta}_{d-1} \end{bmatrix}, \qquad \tilde{\boldsymbol{b}} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{3.20}$$

Therefore, we can represent the part of the dynamics affected by the model error $f(\cdot) - \mu(\cdot)$ via $\tilde{e}_1(\cdot)$, while $\tilde{\boldsymbol{e}}_2(\cdot)$ captures the internal dynamics. This decoupled perspective admits an effective analysis of the tracking error, as shown in the following theorem.

Theorem 3.2. *Consider a linear system* (3.2) *with system structure* (3.17)*, which is controlled by* (3.4) *with control gains* $\boldsymbol{\theta}$ *satisfying Assumption 3.3. Assume that a Gaussian process is used to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *of* $f(\cdot)$*, such that Assumption 3.2 holds on a compact set* $\mathbb{S} \subset \mathbb{X}$ *with a uniform error bound* $\eta(\cdot)$ *which admits a Lipschitz constant* L_η*. Then, the closed-loop system* (3.5) *admits a probabilistic tracking error bound* $\upsilon(\cdot)$*, if* $\mathbb{B}_{t)}(\boldsymbol{x}_{\mathrm{ref}}(t)) \subset \mathbb{S}$ *holds for all* $t \in \mathbb{R}_{0,+}$*, where* $\upsilon(\cdot)$ *is defined through*

$$\upsilon(t) = \sqrt{\xi_1^2(t) + (1 + \|\tilde{\boldsymbol{\theta}}\|)\xi_2^2(t)} \tag{3.21}$$

$$\dot{\boldsymbol{\xi}} = \begin{bmatrix} L_\eta - k_c & L_\eta(1 + \|\tilde{\boldsymbol{\theta}}\|), \\ \|\boldsymbol{U}\| \boldsymbol{U}^{-1} \tilde{\boldsymbol{b}}\| & \bar{\lambda}(\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}})) \end{bmatrix} \boldsymbol{\xi} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \eta(\boldsymbol{x}_{\mathrm{ref}}) \tag{3.22}$$

with initial condition $\xi_1(0) = |\tilde{e}_1(0)|$, $\xi_2(0) = \|\boldsymbol{U}\| \|\boldsymbol{U}^{-1} \tilde{\boldsymbol{e}}_2(0)\|$.

Proof. Due to the definition of $\tilde{\boldsymbol{e}}_2$, it can be directly seen that

$$\|\boldsymbol{e}(t)\|^2 = e_d^2(t) + \|\tilde{\boldsymbol{e}}_2(t)\|^2. \tag{3.23}$$

Moreover, if follows immediately from the definition of $\tilde{e}_1(t)$ that $e_d(t) = \tilde{e}_1(t) - \tilde{\boldsymbol{\theta}} \tilde{\boldsymbol{e}}_2(t)$, such that we obtain the bound

$$|e_d(t)| \leq |\tilde{e}_1(t)| + \|\tilde{\boldsymbol{\theta}}\| \|\tilde{\boldsymbol{e}}_2(t)\|. \tag{3.24}$$

Substituting this expression into (3.23) yields

$$\|\boldsymbol{e}(t)\| \leq |\tilde{e}_1(t)| + (1 + \|\tilde{\boldsymbol{\theta}}\|) \|\tilde{\boldsymbol{e}}_2(t)\|. \tag{3.25}$$

Therefore, it remains to obtain bounds for $|\tilde{e}_1(t)|$ and $\|\tilde{\boldsymbol{e}}_2(t)\|$. Due to the dynamics of $\tilde{e}_1(\cdot)$ in (3.18), its solution is given by

$$\tilde{e}_1(t) = \tilde{e}_1(0) \mathrm{e}^{-k_c t} + \int_0^t \mathrm{e}^{-k_c(t-t')} f_e(t') \mathrm{d}t', \tag{3.26}$$

where $f_e(t) = f(\boldsymbol{x}(t)) - \mu(\boldsymbol{x}(t))$. Therefore, we directly obtain

$$|\tilde{e}_1(t)| \leq |\tilde{e}_1(0)|\mathrm{e}^{-k_c t} + \int_0^t \mathrm{e}^{-k_c(t-t')}\bar{f}_e(t')\mathrm{d}t', \tag{3.27}$$

where $\bar{f}_e(t)$ can be any function such that $|f_e(t)| \leq \bar{f}_e(t)$. It is straightforward to see that the right-hand side of this inequality is the solution of the differential equation

$$\dot{\xi}_1 = -k_c\xi_1 + \bar{f}_e \tag{3.28}$$

with initial condition $\xi_1(0) = |\tilde{e}_1(0)|$, such that we obtain the bound $|\tilde{e}_1(t)| \leq \xi_1(t)$. In order to derive a bound for $\|\tilde{\boldsymbol{e}}_2(t)\|$ we proceed analogously. The norm along the trajectory $\tilde{\boldsymbol{e}}_2(\cdot)$ can straightforwardly be bounded by

$$\|\tilde{\boldsymbol{e}}_2(t)\| \leq \|\mathrm{e}^{\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}})t}\tilde{\boldsymbol{e}}_2(0)\| + \int_0^t \|\mathrm{e}^{\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}})(t-t')}\tilde{\boldsymbol{b}}\||\tilde{e}_1(t')|\mathrm{d}t' \tag{3.29}$$

Using the eigendecomposition of $\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}}) = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$, it can be directly seen that

$$\|\mathrm{e}^{\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}})t}\tilde{\boldsymbol{b}}\| \leq \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\tilde{\boldsymbol{b}}\|\mathrm{e}^{\bar{\lambda}(\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}}))t}. \tag{3.30}$$

Hence, we obtain

$$\|\tilde{\boldsymbol{e}}_2(t)\| \leq \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\tilde{\boldsymbol{e}}_2(0)\|\mathrm{e}^{\bar{\lambda}(\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}}))t} + \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\tilde{\boldsymbol{b}}\|\int_0^t \mathrm{e}^{\bar{\lambda}(\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}}))(t-t')}\xi_1(t')\mathrm{d}t' \tag{3.31}$$

by exploiting the upper bound $\xi_1(t)$ of $|\tilde{e}_1(t)|$. The right-hand side of this inequality is again the solution of a differential equation such that $\|\tilde{\boldsymbol{e}}_2(t)\| \leq \xi_2(t)$ for

$$\dot{\xi}_2 = \bar{\lambda}(\tilde{\boldsymbol{A}}(\tilde{\boldsymbol{\theta}}))\xi_2 + \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\tilde{\boldsymbol{b}}\|\xi_1 \tag{3.32}$$

with initial condition $\xi_2(0) = \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\tilde{\boldsymbol{e}}_2(0)\|$. It remains to derive a bound $\bar{f}_e(t)$ for $|f_e(t)|$ in (3.28). Due to Assumption 3.2, it holds that $|f_e(t)| \leq \eta(\boldsymbol{x}(t))$ with probability of at least $1-\delta$. Moreover, we have $\eta(\boldsymbol{x}(t)) \leq \eta(\boldsymbol{x}_{\mathrm{ref}}(t)) + L_\eta\|\boldsymbol{e}(t)\|$ due to Lipschitz continuity of $\eta(\cdot)$ guaranteed by assumption. Due to the upper bound $\xi_2(t)$ of $\|\tilde{\boldsymbol{e}}_2(t)\|$, it follows from (3.25) that

$$|f_e(t)| \leq \eta(\boldsymbol{x}_{\mathrm{ref}}(t)) + L_\eta(\xi_1(t) + (1 + \|\tilde{\boldsymbol{\theta}}\|)\xi_2(t)). \tag{3.33}$$

Defining the right handside expression as $\bar{f}_e(t)$, substituting it into (3.28), and writing (3.28), (3.32) in matrix-vector form yields (3.22). Finally, due to (3.25), we have $\|\boldsymbol{e}(t)\|^2 \leq \xi_1^2(t) + (1 + \|\tilde{\boldsymbol{\theta}}\|)^2\xi_2^2(t)$, which concludes the proof. □

While Theorem 3.1 requires solving a one-dimensional differential equation, the decoupled approach employed in Theorem 3.2 yields a two-dimensional system describing the tracking error bound $\upsilon(\cdot)$ as illustrated in Fig. 3.2. This allows a more easily interpretable investigation of the dependence of (3.22) on the parameters of the matrix $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$. For example, we can directly see that a large value of k_c is advantageous for small tracking error bounds, which is an intuitive result considering the fact that k_c corresponds to the control gain when (3.17) is obtained using feedback linearization [31]. Therefore, Theorem 3.2 demonstrates that the dedicated derivation of tracking error bounds $\upsilon(\cdot)$ for specific systems following the ideas of Theorem 3.1 can yield improved results.

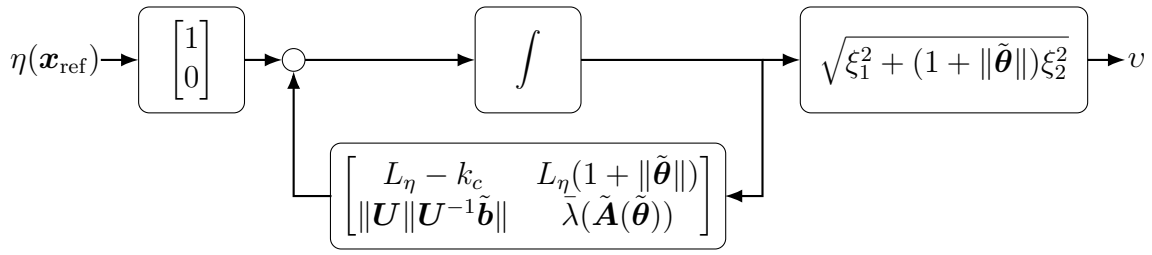

Figure 3.2.: Illustration of the tracking error bound for feedback linearized systems with input perturbations as a dynamical system forced by the uniform GP error bound.

3.1.4. Numerical Evaluation

For evaluating the time-varying tracking error bound, we consider a nonlinear dynamical system

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = f(\boldsymbol{x}) + g(\boldsymbol{x})u, \tag{3.34}$$

where $f(\boldsymbol{x}) = 1 - \sin(2x_1) + {}^1\!/_{(1+\exp(-x_2))}$ and $g(\boldsymbol{x}) = 20\,(1 + {}^1\!/_2 \sin\,({}^{x_2}\!/_4))$, which is a marginal variation of the system considered in [31]. Assuming exact knowledge of $g(\cdot)$, we can approximately feedback linearize this system and apply a control law (3.4) with a gain vector $\boldsymbol{\theta} = k_c[\tilde{\boldsymbol{\theta}}^T\ 1]^T + [0\ \tilde{\boldsymbol{\theta}}^T]^T$ as introduced in Section 3.1.3. This yields a two-dimensional system of the form (3.2) with $\boldsymbol{A}$ and $\boldsymbol{b}$ defined in (3.17). In order to demonstrate the effect of the training data distribution on the tracking error bound $\upsilon(\cdot)$, we use a uniform grid over $[0\ 3] \times [-4\ 4]$ with 25 points and $\sigma_{\mathrm{on}}^2 = 0.01$ as training data set, such that half of the considered state space $\mathbb{X} = [-5\ 5]^2$ is not covered by training data. A SE ARD kernel is employed for Gaussian process regression, and the hyperparameters are optimized using likelihood maximization. Furthermore, we employ Proposition 2.2 with $\tau = 0.01$, $\delta = 0.01$ and $L_f = 2$ for computing the uniform error bound $\eta(\cdot)$. The task is to track the circular reference trajectory $x_{\mathrm{ref},1}(t) = 2\sin(t)$ with state x_1, which leads to the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(t) = [2\sin(t)\ 2\cos(t)]^T$. We aim to achieve this using $\tilde{\theta} = 20$ and $k_c = 50$, which can be shown to satisfy condition (3.13).

Snapshots of the resulting trajectory together with visualizations of the tracking error bounds obtained using Theorem 3.1 are illustrated in Fig. 3.3. When the GP standard deviation $\sigma(\boldsymbol{x}_{\mathrm{ref}}(t))$ is large, the tracking error bound $\upsilon(t)$ starts to increase, such that it reaches its maximum just before the system enters the region with low standard deviation. Afterward, the feedback controller reduces the tracking error until the standard deviation starts to increase again. This leads to the minimum tracking error bound illustrated on the left of Fig. 3.3.

This effect can also be seen in the observed tracking error as illustrated in Fig. 3.4, which has its peaks at times when the tracking error bound $\upsilon(t)$ is large. Therefore, the tracking error bound $\upsilon(\cdot)$ reflects the behavior of the observed error $\|\boldsymbol{e}(t)\|$ well, even though it is rather conservative. The sources of this conservatism can be easily investigated by determining the bound obtained when using the true model error $|f(\boldsymbol{x}_{\mathrm{ref}}(t)) - \mu(\boldsymbol{x}_{\mathrm{ref}}(t))|$ as input in (3.6). It is clearly visible that even with the knowledge of the true prediction error, the tracking error bound exhibits some conservatism due to the linearization around

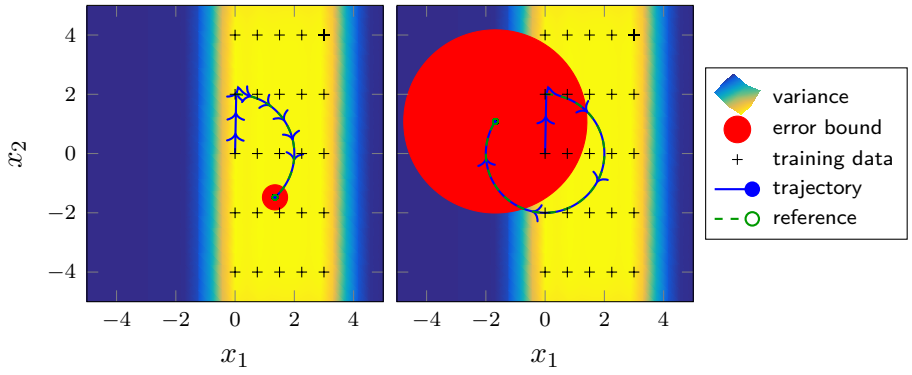

Figure 3.3.: Snapshots of the reference trajectory and simulated trajectory together with the illustration of the tracking error bound $v(\cdot)$. Low posterior standard deviations lead to significantly smaller tracking error bounds.

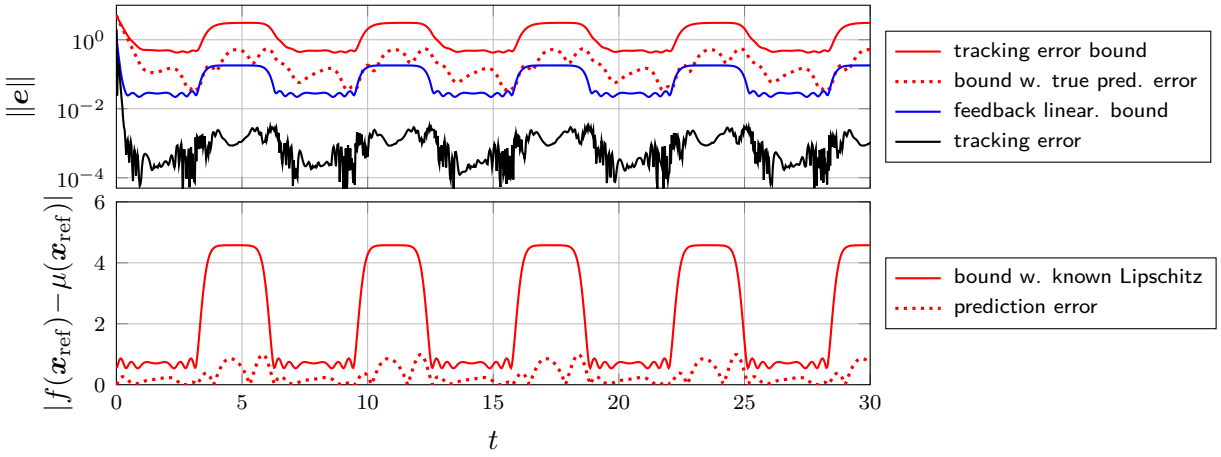

Figure 3.4.: Top: Tracking error bounds computed using (3.6) and (3.21) with different prediction error bounds as inputs in comparison to the observed tracking error. Bottom: Prediction error bounds in comparison to the true model error.

the reference trajectory $\boldsymbol{x}_{\text{ref}}(t)$. By employing the dedicated tracking error bound for feedback linearization in Theorem 3.2, this weakness can be partially mitigated. The remaining conservatism is a consequence of the prediction error bound $\eta(\boldsymbol{x}_{\text{ref}}(t))$ as visualized at the bottom of Fig. 3.4. Even though this bound reflects the availability of data well, it needs to capture the probabilistic worst case and is therefore considerably larger than the actual prediction error $|f(\boldsymbol{x}_{\text{ref}}(t))-\mu(\boldsymbol{x}_{\text{ref}}(t))|$. This leads to the fact that the tracking error bound $v(\cdot)$ conservatively reflects the behavior of the observed tracking error $\|\boldsymbol{e}(t)\|$.

3.2. Certainty Equivalence Approaches for Lyapunov-Based Control Design

While the results in the previous section provide effective tracking error bounds, they are inherently limited to linear control systems. Therefore, nonlinear tracking control laws based on GP models must be employed in order to achieve a flexible applicability to a broad

class of continuous-time systems. These control laws can be designed using a variety of methods. Even though there are approaches based on methods such as control contraction metrics [114], the vast majority of design techniques for GP-based nonlinear control laws rely on Lyapunov stability theory [113]. This theory can be directly applied using control Lyapunov functions, such that second order cone programs need to be solved online for determining control inputs [115]. The required Lyapunov functions can be chosen based on classical control principles [32], but also admit a design exploiting the GP variance [102]. While the control Lyapunov function allows a straightforward implementation, solving an optimization problem online is computationally expensive in general.

This issue is avoided by using closed-form control laws. Such controllers exist for a broad class of systems and can be designed using feedback linearization [31, 33], computed torque control [26, 34], sliding mode control [116], and backstepping [110]. The design approach for these controllers is usually fundamentally based on the equivalence principle [117], i.e., they are designed as if the GP model would correspond to the true system dynamics, even though robustness can be additionally increased. Due to the existence of uniform error bounds for the GP model, the nominal stability guarantees obtained through the certainty equivalent design can be extended to the unknown system in a weaker form under some assumptions. This allows the straightforward derivation of tracking error bounds for specific control laws, which depend on the GP error bound.

Due to the widespread employment of certainty equivalent controllers based on GP models, we propose a novel, unified approach for deriving tracking error bounds. The required analysis is inspired by Lyapunov stability theory[2], which is based on the sub-level sets of Lyapunov candidates defined as follows.

Definition 3.2. *A Lyapunov candidate is a continuous function $V : \mathbb{R}^{d_x} \to \mathbb{R}_{0,+}$, such that there exist class $\mathcal{K}$ functions $\alpha_1, \alpha_2 : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$ satisfying*

$$\alpha_1(\|\boldsymbol{x}\|) \leq V(\boldsymbol{x}) \leq \alpha_2(\|\boldsymbol{x}\|) \quad \forall \boldsymbol{x} \in \mathbb{R}^{d_x}. \tag{3.35}$$

Due to the upper and lower bound by monotonically increasing functions $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$, respectively, Lyapunov candidates can be used to measure the distance of $\boldsymbol{x} \in \mathbb{X}$ to the origin $\mathbf{0}$. This allows us to investigate the behavior of tracking errors through the derivative of Lyapunov candidates. We exploit this by bounding the Lyapunov candidate derivative using uniform error bounds for GP models, which allows the computation of tracking error bounds by solving an optimization problem. Using additional continuity assumptions, this approach can be extended into a time-varying tracking error bound depending on the local model error. These general approaches can be directly applied to specific control laws, which we demonstrate for the example of feedback linearization to illustrate the practicality of the derived theory.

The remainder of this section is structured as follows. The problem setting, together with the assumption on the certainty equivalent controller design, is formalized in Section 3.2.1. A general approach for deriving tracking error bounds based on subsets of Lyapunov candidates is presented in Section 3.2.2. In Section 3.2.3, the general approach is slightly restricted to admit the derivation of tracking error bounds, which depend on the local GP error bound. Finally, the theoretical results are illustrated for feedback linearizing control laws in Section 3.2.4.

[2] A short introduction to Lyapunov stability theory can be found in Appendix A.2, but is not necessary for the comprehension of this section.

3.2.1. Problem Setting

We consider a general, nonlinear dynamical system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \tag{3.36}$$

where $\boldsymbol{f} : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^{d_x}$ is an unknown, continuous function, $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^{d_x}$ denotes the state and $\boldsymbol{u} \in \mathbb{U} \subseteq \mathbb{R}^{d_u}$ are the control inputs. The task is to track a bounded reference trajectory $\boldsymbol{x}_{\text{ref}} : \mathbb{R}_{0,+} \to \mathbb{X}$ with the state $\boldsymbol{x}(t)$.

In order to be able to execute this task, we assume to have a data set $\mathbb{D} = \{\boldsymbol{z}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$, $N \in \mathbb{N}$, where we define $\boldsymbol{z} = [\boldsymbol{x}^T \ \boldsymbol{u}^T]^T$, such that we can train a multi-output GP model with mean $\boldsymbol{\mu} : \mathbb{R}^{d_x+d_u} \to \mathbb{R}^{d_x}$ and variance $\boldsymbol{\sigma}^2 : \mathbb{R}^{d_x+d_u} \to \mathbb{R}^{d_x}$ as discussed in Section 2.1.4. Based on the GP model, the goal is the development of a controller $\boldsymbol{\pi} : \mathbb{R}^{d_x} \times \mathbb{R}_{0,+} \to \mathbb{U}$ with accuracy guarantees. While this is a challenging problem for general nonlinear dynamical systems (3.36) even with exact knowledge of the function $\boldsymbol{f}(\cdot, \cdot)$, many suitable nonlinear control design methods exist under additional assumptions, e.g., feedback linearization, sliding mode control or backstepping [113]. Since the derivation of tracking error bounds for such control laws is usually based on Lyapunov stability theory, we avoid a restriction to specific control design methods through the following assumption.

Assumption 3.4. *Given a GP mean function $\boldsymbol{\mu}(\cdot)$, the design method provides a continuous control law $\boldsymbol{\pi}(\cdot, \cdot)$, such that there exists a Lyapunov candidate $V : \mathbb{R}^{d_x} \to \mathbb{R}_{0,+}$ satisfying*

$$\dot{V}_{\text{nom}}(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t)) = \nabla^T V(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t))(\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}, t)) - \dot{\boldsymbol{x}}_{\text{ref}}(t)) < 0, \quad \forall \boldsymbol{x} \neq \boldsymbol{x}_{\text{ref}}(t), t \in \mathbb{R}_{0,+}, \tag{3.37}$$

where we define $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}, t)) = \boldsymbol{\mu}([\boldsymbol{x}^T \ \boldsymbol{\pi}^T(\boldsymbol{x}, t)]^T)$ with a slight abuse of notation.

It is important to note that this assumption does not require knowledge of the unknown dynamics $\boldsymbol{f}(\cdot, \cdot)$, but only depends on the GP mean $\boldsymbol{\mu}(\cdot)$. Therefore, it merely requires a sufficient flexibility of the nonlinear control design method, which is capable of dealing with the non-parametric structure of GP models. In the past few years, this flexibility has been shown for many classical nonlinear controller design methods such as backstepping [110], computed torque control [34], feedback linearization [31], and sliding mode control [116], such that Assumption 3.4 is not restrictive in practice.

Since control design methods satisfying Assumption 3.4 rely solely on the GP mean function $\boldsymbol{\mu}(\cdot)$, they are ignorant of potential model errors. This perspective on control design is known as the certainty equivalence approach [117] and directly gives rise to a nominal dynamical system

$$\dot{\boldsymbol{x}} = \boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x})). \tag{3.38}$$

While it is straightforward to check that (3.37) ensures a stable tracking of the reference $\boldsymbol{x}_{\text{ref}}(\cdot)$ with the nominal system (3.38), this does not imply guarantees for the unknown system (3.36). Therefore, we consider the problem of deriving probabilistic tracking error bounds as formalized in Definition 3.1 for certainty equivalent control design approaches satisfying Assumption 3.4 based on uniform error bounds for GP models as stated in Assumption 3.2.

3.2.2. General Lyapunov-Based Tracking Error Bounds

In order to derive probabilistic tracking error bounds for general closed-loop systems

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x})), \tag{3.39}$$

we bound the impact of model errors on the Lyapunov condition (3.37) using uniform error bounds for GP models. While this approach allows to straightforwardly determine if a positive Lyapunov derivative $\dot{V}(\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{ref}}(t))$ at a given state $\boldsymbol{x}(t)$ cannot occur, this information alone does not allow us to decide if $\|\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{ref}}(t)\| \leq \upsilon(t)$ or even determine $\upsilon(\cdot)$. This is due to the fact that Lyapunov theory does not allow statements for specific states but crucially relies on (sub)-level sets of the Lyapunov candidate $V(\cdot)$. Therefore, we determine a probabilistic tracking error bound for system (3.39) by analyzing the sub-level sets $\mathbb{V}_c(t) = \{\boldsymbol{x} \in \mathbb{X} : V(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) \leq c\}$, $c \in \mathbb{R}_{0,+}$, which results in the following theorem.

Theorem 3.3. *Consider a dynamical system* (3.36) *and a Gaussian process model with mean* $\boldsymbol{\mu}(\cdot)$*, such that Assumption 3.2 holds for each* $\mu_i(\cdot)$*,* $i = 1, \ldots, d_x$*, with uniform error bound* $\eta_i(\cdot)$ *on a compact set* $\mathbb{S} = \mathbb{S}_x \times \mathbb{S}_u$ *with* $\mathbb{S}_x \subset \mathbb{X}$*,* $\mathbb{S}_u \subset \mathbb{U}$*. Moreover, assume that a controller* $\boldsymbol{\pi}(\cdot, \cdot)$ *satisfying Assumption 3.4 is used to track the bounded reference trajectory* $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$*. If* $\mathbb{V}_{\bar{V}}(t) \subset \mathrm{int}(\mathbb{S}_x)$ *holds for all* $t \in \mathbb{R}_{0,+}$ *and* $\boldsymbol{\pi}(\boldsymbol{x}, t) \in \mathrm{int}(\mathbb{S}_u)$ *holds for all* $\boldsymbol{x} \in \mathbb{V}_{\bar{V}}(t)$*,* $t \in \mathbb{R}_{0,+}$*, where*

$$\bar{V} = \max_{\boldsymbol{x} \in \mathbb{S}, t \in \mathbb{R}_{0,+}} V(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) \tag{3.40}$$

$$\text{such that } \dot{V}_{\mathrm{nom}}(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) + |\nabla^T V(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t))| \boldsymbol{\eta}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}, t)) \geq 0, \tag{3.41}$$

then, the closed-loop system (3.39) *admits a probabilistic tracking error bound* $\upsilon(\cdot) = \bar{\upsilon}$*, where* $\bar{\upsilon} = \alpha_1^{-1}(\bar{V})$*.*

Proof. In order to prove this theorem, we employ the Lyapunov candidate $V(\cdot)$ available from Assumption 3.4 and investigate its temporal derivative, which is given by

$$\dot{V}(\boldsymbol{e}) = \nabla^T V(\boldsymbol{e}) \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x})). \tag{3.42}$$

Due to Assumption 3.2, we can bound this expression on the compact set $\mathbb{S}$ by

$$\dot{V}(\boldsymbol{e}) \leq \dot{V}_{\mathrm{nom}}(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) + |\nabla^T V(\boldsymbol{e})| \boldsymbol{\eta}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x})). \tag{3.43}$$

Note that $V(\boldsymbol{e}(\cdot))$ is continuous due to the continuity of $V(\cdot)$ and the fact that $\boldsymbol{e}(\cdot)$ is a solution to a differential equation. Therefore, it is straightforward to see that

$$\dot{V}(\boldsymbol{e}) \leq 0 \quad \forall \boldsymbol{e} \in \{\boldsymbol{e} \in \mathbb{R}^{d_x} : V(\boldsymbol{e}) = c\} \tag{3.44}$$

for any $c \in \mathbb{R}_+$ implies that $V(\boldsymbol{e}(t)) \leq c$ for $V(\boldsymbol{e}(0)) \leq c$ since $V(\cdot)$ is a scalar function. This immediately implies that a bound for $V(\boldsymbol{e}(\cdot))$ can be obtained by considering (3.44) as a constraint, which yields the optimization problem

$$\inf_{c \in \mathbb{R}_+} c \tag{3.45}$$

$$\text{such that } \dot{V}(\boldsymbol{e}) \leq 0 \quad \forall \boldsymbol{e} \in \{\boldsymbol{e} \in \mathbb{R}^{d_x} : V(\boldsymbol{e}) = c\}. \tag{3.46}$$

If we know a value c satisfying (3.44), a local optimum of this problem can be obtained via the optimization problem

$$\max_{\boldsymbol{e}\in\mathbb{R}^{d_x}} V(\boldsymbol{e}) \tag{3.47}$$

$$\text{such that } \dot{V}(\boldsymbol{e}) \geq 0 \tag{3.48}$$

$$V(\boldsymbol{e}) \leq c, \tag{3.49}$$

which employs $\boldsymbol{e}$ as optimization variable. When substituting $\dot{V}(\boldsymbol{e})$ by (3.43), $\boldsymbol{x}$ and t explicitly occur in the constraint, such that we need to consider them as optimization variables instead of $\boldsymbol{e}$. This leads to the optimization problem

$$\sup_{\boldsymbol{x}\in\mathbb{X},t\in\mathbb{R}_{0,+}} V(\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t)) \tag{3.50}$$

$$\text{such that } \dot{V}_{\mathrm{nom}}(\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t)) + |\nabla^T V(\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t))|\boldsymbol{\eta}(\boldsymbol{x},\boldsymbol{\pi}(\boldsymbol{x},t)) \geq 0 \tag{3.51}$$

$$V(\boldsymbol{e}) \leq c. \tag{3.52}$$

It is straightforward to see that if (3.40), (3.41) results in a solution $\bar{V}$ such that $\mathbb{V}_{\bar{V}}(t) \subset \mathrm{int}(\mathbb{S}_x)$ holds for all $t \in \mathbb{R}_{0,+}$, $\bar{V}$ is identical to the solution of (3.50) for a sufficiently large value c such that $\mathbb{V}_c(t) \subset \mathrm{int}(\mathbb{S}_x)$. Therefore, $\bar{V}$ upper bounds $V(\boldsymbol{e}(\cdot))$. Finally, the probabilistic tracking error bound follows from (3.35), which results in $\upsilon(\cdot) = \alpha_1^{-1}(\bar{V})$ for all $\boldsymbol{x}(0) \in \mathbb{X}_0 = \mathbb{V}_{\bar{V}}(0)$. □

Similar as in Section 3.1, a restriction to a compact set $\mathbb{S}$ is required in Theorem 3.3 in order to ensure the validity of uniform error bounds for the GP model. While this local analysis can lead to the problem that no valid tracking error bound $\upsilon(\cdot)$ is obtained by solving (3.40), this is usually merely an artifact stemming from a misspecification of $\mathbb{S}$. By choosing a sufficiently large set $\mathbb{S}$, e.g., via an iterative enlargement until $\mathbb{V}_{\bar{V}}(t) \subset \mathrm{int}(\mathbb{S})$ is satisfied, this limitation can often be overcome in a practically easily implementable way.

Due to the derivation of the optimization problem (3.40), it generally results in a locally optimal tracking error bound $\upsilon(\cdot)$, whose tightness depends on the choice of $\mathbb{S}$. Moreover, the solution of (3.40) can generally not be determined analytically, but it can be straightforwardly approximated using numerical optimization methods. If additional knowledge about the structure of the Lyapunov candidate and the dynamics is known, it is possible to significantly simplify (3.41). This can enable the straightforward derivation of closed-form expressions for $\bar{V}$ when an upper bound for $\boldsymbol{\eta}(\cdot)$ is available [31, 34, 110], but generally increases the conservatism of the probabilistic tracking error bound $\upsilon(\cdot)$. Therefore, a trade-off between quality and computational complexity for determining the tracking error bound must be found in practice.

Finally, it is important to point out that the first term in (3.41) is negative due to the certainty equivalent controller design considered in Assumption 3.4. Therefore, we can generally improve the tracking accuracy guarantees by reducing the model uncertainty represented by the uniform error bound $\boldsymbol{\eta}(\cdot)$. However, (3.41) highlights that we do not need the same model accuracy everywhere: when the uniform error bound is small enough to ensure the satisfaction of (3.41) at a state $\boldsymbol{x}$, an improvement of the model accuracy at this state does not improve the guaranteed tracking performance. This effect is investigated in more detail in Section 4.2.

In order to demonstrate the applicability of Theorem 3.3, we employ it in the example of a feedback linearizing controller designed using a GP mean function. For this purpose, we consider a dynamical system of the form

$$\dot{x}_1 = x_2, \qquad \cdots \qquad \dot{x}_{d_x-1} = x_{d_x}, \qquad \dot{x}_{d_x} = f(\boldsymbol{x}, u), \tag{3.53}$$

where $f : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ is an unknown, continuous function. Moreover, we assume that the unknown function $f(\cdot, \cdot)$ has a control-affine structure, i.e., $f(\boldsymbol{x}, u) = \tilde{f}(\boldsymbol{x}) + g(\boldsymbol{x})u$ for unknown functions $\tilde{f} : \mathbb{R}^{d_x} \to \mathbb{R}$ and $g : \mathbb{R}^{d_x} \to \mathbb{R}$. To allow exact tracking, the reference trajectory is defined as $\boldsymbol{x}_{\mathrm{ref}}(t) = [r(t)\ \dot{r}(t)\ \cdots\ \frac{\mathrm{d}^{d_x-1} r(t)}{\mathrm{d}t^{d_x-1}}]^T$ for a d_x times continuously differentiable function $r : \mathbb{R}_{0,+} \to \mathbb{R}$.

We reflect the structure of the function $f(\cdot)$ in the GP model using a composite kernel $k(\boldsymbol{z}, \boldsymbol{z}') = k_{\tilde{f}}(\boldsymbol{x}, \boldsymbol{x}') + u k_g(\boldsymbol{x}, \boldsymbol{x}') u'$ [31], where $\boldsymbol{z} = [\boldsymbol{x}^T\ u]^T$. Therefore, it directly follows from Section 2.1.3 that we can recover individual mean functions for $\tilde{f}(\cdot)$ and $g(\cdot)$ from measurements of $f(\cdot)$ via

$$\mu_{\tilde{f}}(\boldsymbol{z}) = \boldsymbol{k}_{\tilde{f}}^T(\boldsymbol{z}_1) \left(\boldsymbol{K} + \sigma_{\mathrm{on}}^2 \boldsymbol{I}_N \right)^{-1} \boldsymbol{y}, \tag{3.54}$$

$$\mu_g(\boldsymbol{z}) = \boldsymbol{k}_g^T(\boldsymbol{z}_1) \boldsymbol{U} \left(\boldsymbol{K} + \sigma_{\mathrm{on}}^2 \boldsymbol{I}_N, \right)^{-1} \boldsymbol{y} \tag{3.55}$$

under the assumption of prior means equal to 0, where the diagonal matrix $\boldsymbol{U} \in \mathbb{R}^{N \times N}$ is composed of control input measurements, i.e., $U_{n,n} = u^{(n)}$. This model allows the definition of a certainty equivalent control law

$$\pi(\boldsymbol{x}, t) = \frac{1}{\mu_g(\boldsymbol{x})} \left(\pi_{\mathrm{lin}}(\boldsymbol{x}, t) - \mu_f(\boldsymbol{x}) \right), \tag{3.56}$$

where $\pi_{\mathrm{lin}} : \mathbb{X} \to \mathbb{R}_{0,+}$ is a linear tracking controller. In order to ensure that this controller is well-defined, we make the following assumption.

Assumption 3.5. *The GP mean function $\mu_g(\cdot)$ is positive, i.e., $\mu_g(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \mathbb{S}$.*

This assumption, which can be commonly found when GP models are employed in feedback linearizing control laws [31, 32, 37], can be easily ensured by a suitable choice of GP hyperparameters [31, Proposition 1]. Moreover, the existence of a uniform error bound immediately implies that this assumption can be guaranteed using sufficiently large data sets if the true function $g(\cdot)$ is positive. Therefore, it is generally not restrictive.

Defining the linear tracking controller as

$$\pi_{\mathrm{lin}}(\boldsymbol{x}, t) = \frac{\mathrm{d}^{d_x} r(t)}{\mathrm{d}t^{d_x}} - k_c \begin{bmatrix} \tilde{\boldsymbol{\theta}}^T & 1 \end{bmatrix} (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) - \tilde{\boldsymbol{\theta}}^T \begin{bmatrix} x_2 - x_{\mathrm{ref},2}(t) \\ \vdots \\ x_{d_x} - x_{\mathrm{ref},d_x}(t) \end{bmatrix}, \tag{3.57}$$

where $\tilde{\boldsymbol{\theta}} \in \mathbb{R}^{d_x - 1}$ are Hurwitz coefficients, and assuming the reference exhibits the form (3.3), we can express the closed-loop dynamics of (3.53) as

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}) + \boldsymbol{b}(f(\boldsymbol{x}) - \mu_f(\boldsymbol{x}) + (g(\boldsymbol{x}) - \mu_g(\boldsymbol{x}))\pi(\boldsymbol{x})) + \dot{\boldsymbol{x}}_{\mathrm{ref}}, \tag{3.58}$$

where $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) \in \mathbb{R}^{d_x \times d_x}$ and $\boldsymbol{b} \in \mathbb{R}^{d_x}$ are defined in (3.17) and $\boldsymbol{\theta} = k_c[\tilde{\boldsymbol{\theta}}^T\ 1]^T + [0\ \tilde{\boldsymbol{\theta}}^T]^T$. This formulation directly separates the model into a nominal component given by the first summand and a model error described by the second summand. Therefore, it allows the straightforward application of Theorem 3.3, which yields the following result.

Corollary 3.1. *Consider a dynamical system* (3.53)*, prior Gaussian processes* $\mathcal{GP}(0, k_{\tilde{f}}(\cdot,\cdot))$*,* $\mathcal{GP}(0, k_g(\cdot,\cdot))$ *with Lipschitz continuous kernels such that Assumption 2.6 is satisfied, and training data* $\mathbb{D}$ *with observation noise* $\epsilon^{(n)}$*,* $n = 1, \ldots, N$*, fulfilling Assumption 2.1. Moreover, assume that a controller* (3.56) *with model* $\mu_g(\cdot)$ *satisfying Assumption 3.5 is used to track the bounded reference trajectory* $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$*. Let the symmetric, positive definite matrix* $\boldsymbol{P} \in \mathbb{R}^{d_x \times d_x}$ *denote the solution to the Lyapunov equation* $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})^T \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = -\boldsymbol{I}$ *for* $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ *defined in* (3.17)*. If* $\mathbb{V}_{\bar{V}}(t) \subset \mathrm{int}(\mathbb{S})$ *holds for all* $t \in \mathbb{R}_{0,+}$*, where*

$$\bar{V} = \max_{\boldsymbol{x} \in \mathbb{S}, t \in \mathbb{R}_{0,+}} V(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) \tag{3.59}$$

$$\text{such that } \|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)\|^2 \leq |(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}| \left(\eta_{\tilde{f}}(\boldsymbol{x}) + \frac{\eta_g(\boldsymbol{x})}{\mu_g(\boldsymbol{x})} \left| \pi_{\mathrm{lin}}(\boldsymbol{x}, t) - \mu_{\tilde{f}}(\boldsymbol{x}) \right| \right) \tag{3.60}$$

with $\eta_{\tilde{f}}(\cdot)$ *and* $\eta_g(\cdot)$ *defined through the combination of Corollary 2.4 and Corollary 2.5, then, the closed-loop system* (3.58) *admits a probabilistic tracking error bound* $v(\cdot) = \bar{v}$*, where* $\bar{v} = \sqrt{\bar{V} / \underline{\lambda}(\boldsymbol{P})}$*.*

Proof. Due to Lemma 2.2, Lemma 2.3 and Theorem 2.3, Hölder continuity of $\mu_{\tilde{f}}(\cdot)$, $\sigma_{\tilde{f}}(\cdot)$, $\tilde{f}(\cdot)$, $\mu_g(\cdot)$, $\sigma_g(\cdot)$ and $g(\cdot)$ is guaranteed. Therefore, probabilistic uniform error bounds for $\tilde{f}(\cdot)$ and $g(\cdot)$ are guaranteed through the combination of Corollary 2.4 and Corollary 2.5. Moreover, it is obvious from the closed-loop dynamics (3.58) that $V(\boldsymbol{e}) = \boldsymbol{e}^T \boldsymbol{P} \boldsymbol{e}$ is a Lyapunov candidate satisfying Assumption 3.4 with $\nabla V(\boldsymbol{e}) = 2\boldsymbol{P}\boldsymbol{e}$. Hence, it directly follows from Theorem 3.3 that

$$\bar{V} = \max_{\boldsymbol{x} \in \mathbb{S}, t \in \mathbb{R}_{0,+}} V(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) \tag{3.61}$$

$$\text{such that } \|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)\|^2 \leq 2|(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}| \left(\eta_{\tilde{f}}(\boldsymbol{x}) + \eta_g(\boldsymbol{x}) \, |\pi(\boldsymbol{x}, t)| \right) \tag{3.62}$$

induces a probabilistic tracking error bound. Since $\mu_g(\cdot)$ is positive by assumption, we can simplify this optimization problem to (3.59) using the definition of $\pi(\cdot,\cdot)$ in (3.56). Finally, the tracking error bound $\bar{v}$ follows from $\underline{\lambda}(\boldsymbol{P})\|\boldsymbol{e}\|^2 \leq V(\boldsymbol{e})$. □

This corollary demonstrates that Theorem 3.3 admits the straightforward derivation of a simple constraint (3.60) by exploiting the additional structure provided by the restriction to feedback linearizing controllers. Moreover, it provides a helpful insight into the impact of the model accuracy. In order to see this, assume that the quotient $\eta_g(\boldsymbol{x}) / \mu_g(\boldsymbol{x})$ in (3.60) is upper bounded by a constant $c \in \mathbb{R}$. Then, we can exploit the definition of $\pi_{\mathrm{lin}}(\cdot,\cdot)$ to show that

$$-|\boldsymbol{e}^T|(\hat{\boldsymbol{A}}^T \boldsymbol{P} + \boldsymbol{P}\hat{\boldsymbol{A}}|\boldsymbol{e}| \leq 2|\boldsymbol{e}^T \boldsymbol{P}\boldsymbol{b}| \left(\eta_{\tilde{f}}(\boldsymbol{x}) + \eta_g(\boldsymbol{x}) \left| \frac{\mathrm{d}^{d_x} r(t)}{\mathrm{d}t^{d_x}} - \mu_{\tilde{f}}(\boldsymbol{x}) \right| \right), \tag{3.63}$$

where

$$\hat{\boldsymbol{A}} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ (c-1)k_c\tilde{\theta}_1 & (c-1)(k_c\tilde{\theta}_2 + \tilde{\theta}_1) & \cdots & (c-1)(k_c + \tilde{\theta}_{d_x - 1}) \end{bmatrix}. \tag{3.64}$$

Therefore, the model error in $g(\cdot)$ effectively causes a control gain scaling by a factor $1-c$. If this factor is positive, we can simply increase the control gain k_c to reduce the norm of $\boldsymbol{P}\boldsymbol{b}$ for $\boldsymbol{P}$ defined via $\hat{\boldsymbol{A}}^T\boldsymbol{P}+\boldsymbol{P}\hat{\boldsymbol{A}}=-\boldsymbol{I}$. Hence, an arbitrary small tracking error bound can be achieved with sufficiently large control gain k_c if the uniform error bound $\eta_g(\cdot)$ is sufficiently small compared to $\mu_g(\cdot)$.

Note that the right side of (3.63) grows linearly with the error $\boldsymbol{e}$ when bounded kernels $k_f(\cdot,\cdot)$ and $k_g(\cdot,\cdot)$ are employed, while the left side increases quadratically. Furthermore, the Bayesian uniform error bound in Theorem 2.2 depends logarithmically on the extension of $\mathbb{S}$. Therefore, ${}^{\eta_g(\boldsymbol{x})}/_{\mu_g(\boldsymbol{x})}<1$ ensures that it is always possible to find a sufficiently large set $\mathbb{S}$ for determining a valid tracking error bound $\upsilon(\cdot)$, which underlines the statement that the necessary restriction to a compact set $\mathbb{S}$ in Theorem 3.3 is often not a severe limitation.

3.2.3. Linearization-Based Time-Varying Accuracy Guarantees

While Theorem 3.3 allows a flexible application to a wide class of dynamical systems, the resulting tracking error bounds $\upsilon(\cdot)$ exhibit the same size globally along the whole trajectory. In fact, a small region around one point along the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$ with high uniform error bound $\boldsymbol{\eta}(\cdot)$ can cause a large tracking error bound along the whole trajectory. Therefore, (3.40) can only partially reflect the local model accuracy. In the following, we show that this limitation can be overcome using additional assumptions, such as a further restriction of the admissible Lyapunov candidates.

Assumption 3.6. *The design method provides a continuous control law $\boldsymbol{\pi}(\cdot,\cdot)$, such that there exists a twice differentiable Lyapunov candidate $V:\mathbb{R}^{d_x}\to\mathbb{R}_{0,+}$ satisfying*

$$a_1\|\boldsymbol{e}\|^2\leq V(\boldsymbol{e})\leq a_2\|\boldsymbol{e}\|^2, \tag{3.65}$$

$$\dot{V}_{\mathrm{nom}}(\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t))\leq -a_3\|\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t)\|^2,\quad \forall \boldsymbol{x}\in\mathbb{X}, t\in\mathbb{R}_{0,+}. \tag{3.66}$$

This assumption effectively requires exponential stability of the nominal closed-loop system [113], which can be ensured for control laws such as computed torque [34] and feedback linearization [31]. Therefore, it is not overly restrictive in practice.

Based on Assumption 3.6, we can linearize the Lyapunov derivative condition (3.41) around the reference trajectory. This allows us to express the tracking error bound through a differential equation, as shown in the following theorem.

Theorem 3.4. *Consider a dynamical system* (3.36) *and a Gaussian process model with mean $\boldsymbol{\mu}(\cdot)$, such that Assumption 3.2 holds for each $\mu_i(\cdot)$, $i=1,\ldots,d_x$, with Lipschitz continuous uniform error bound $\eta_i(\cdot)$ on a compact set $\mathbb{S}=\mathbb{S}_x\times\mathbb{S}_u$ with $\mathbb{S}_x\subset\mathbb{X}$, $\mathbb{S}_u\subset\mathbb{U}$. Moreover, assume that a controller $\boldsymbol{\pi}(\cdot,\cdot)$ satisfying Assumption 3.6 is used to track the bounded reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$. If*

$$a_3>L_\eta L_{\nabla V}(1+L_\pi), \tag{3.67}$$

$\mathbb{V}_{\xi(t)}(t)\subset\mathrm{int}(\mathbb{S}_x)$ holds for all $t\in\mathbb{R}_{0,+}$ and $\boldsymbol{\pi}(\boldsymbol{x},t)\in\mathrm{int}(\mathbb{S}_u)$ holds for all $\boldsymbol{x}\in\mathbb{V}_{\xi(t)}(t)$, $t\in\mathbb{R}_{0,+}$, where $\xi(\cdot)$ is the solution to the differential equation

$$\dot{\xi}=\frac{-a_3+L_\eta L_{\nabla V}(1+L_\pi)}{a_2}\xi+\frac{L_{\nabla V}\|\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t))\|}{\sqrt{a_1}}\sqrt{\xi} \tag{3.68}$$

for initial condition $\xi(0) = V(\boldsymbol{x}(0) - \boldsymbol{x}_{\mathrm{ref}}(0)) > 0$ *and* $\boldsymbol{z}_{\mathrm{ref}}(\cdot) = [\boldsymbol{x}_{\mathrm{ref}}^T(\cdot)\ \boldsymbol{\pi}^T(\boldsymbol{x}_{\mathrm{ref}}(\cdot), \cdot)]^T$*, then, the closed-loop system* (3.39) *admits a probabilistic tracking error bound* $\upsilon(t) = \sqrt{\xi(t)/a_1}$.

Proof. Since the assumptions of Theorem 3.3 are satisfied, it holds that

$$\dot{V}(\boldsymbol{e}) \leq \dot{V}_{\mathrm{nom}}(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) + \|\nabla V(\boldsymbol{e})\| \|\boldsymbol{\eta}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}, t))\|. \tag{3.69}$$

Moreover, Assumption 3.6 implies that

$$\begin{aligned} \dot{V}(\boldsymbol{e}) &\leq -a_3 \|\boldsymbol{e}\|^2 + \|\nabla V(\boldsymbol{e})\| \|\boldsymbol{\eta}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}, t))\| && (3.70) \\ &\leq -a_3 \|\boldsymbol{e}\|^2 + L_{\nabla V} \|\boldsymbol{\eta}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}, t))\| \|\boldsymbol{e}\| && (3.71) \end{aligned}$$

where the second line follows from differentiability of $\nabla V(\cdot)$ on a compact set and the fact that (3.65) requires $\nabla V(\mathbf{0}) = \mathbf{0}$. Since $\boldsymbol{\eta}(\cdot)$ is Lipschitz continuous, we can linearize $\boldsymbol{\eta}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}, t))$ around $\boldsymbol{x}_{\mathrm{ref}}(t)$, which results in

$$\dot{V}(\boldsymbol{e}) \leq \left(-a_3 + L_{\nabla V} L_\eta (1 + L_\pi)\right) \|\boldsymbol{e}\|^2 + L_{\nabla V} \|\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t))\| \|\boldsymbol{e}\|. \tag{3.72}$$

Due to (3.67), we can exploit (3.65) to bound this expression completely in terms of $V(\boldsymbol{e})$ yielding

$$\dot{V}(\boldsymbol{e}) \leq \frac{-a_3 + L_\eta L_{\nabla V}(1 + L_\pi)}{a_2} V(\boldsymbol{e}) + L_{\nabla V} \|\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t))\| \sqrt{\frac{V(\boldsymbol{e})}{a_1}}. \tag{3.73}$$

Therefore, it directly follows from the comparison lemma [113, Lemma 3.4] that $V(\boldsymbol{e}(t))$ is bounded by $\xi(t)$ defined through the differential equation (3.68), such that the probabilistic tracking error bound $\upsilon(\cdot)$ can be obtained by employing the lower bound in (3.65). Finally, note that stationary points of the Lyapunov candidate $V(\cdot)$ do not allow us to draw conclusions about the transient behavior of the tracking error $\boldsymbol{e}(\cdot)$, such we need can only ensure a bound for initial conditions $\boldsymbol{e}(0) \neq \mathbf{0}$ [118]. □

Condition (3.67) ensures the boundedness of $\xi(\cdot)$, which is necessary due to the restriction of uniform error bounds for GP models to compact sets $\mathbb{S}$. While a_3 can often be increased by increasing control gains in practice, this also causes a growth of L_π. Therefore, the satisfaction crucially relies on a sufficiently slowly varying uniform error bound $\boldsymbol{\eta}(\cdot)$, which can be locally achieved using a sufficiently high data density in a neighborhood of the reference trajectory. Therefore, this assumption poses a relevant restriction, but it can be satisfied through a careful design of the GP model.

Since (3.68) only depends on the uniform error bound along the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$, it can be straightforwardly solved using numerical integration methods in practice. The uniform error bound $\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t))$ can hereby be considered as an external input forcing a stable dynamical system. Therefore, the tracking error bound converges to a stationary point if $\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t))$ is almost constant but can also capture transient behavior if it changes. This interpretation allows a straightforward computation of an upper bound for $\upsilon(\cdot)$ by considering the worst case error bound $\sup_{t \in \mathbb{R}_{0,+}} \|\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t))\|$ as a constant input to (3.72), such that we obtain

$$\dot{V}(\boldsymbol{e}) \leq 0 \quad \forall \boldsymbol{e} : \|\boldsymbol{e}\| \geq \frac{L_{\nabla V} \sup\limits_{t \in \mathbb{R}_{0,+}} \|\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t))\|}{a_3 - L_{\nabla V} L_\eta (1 + L_\pi)}. \tag{3.74}$$

Thereby, we directly obtain the constant tracking error bound

$$\|\boldsymbol{e}(t)\| \leq \frac{\sqrt{a_2} L_{\nabla V} \sup\limits_{t \in \mathbb{R}_{0,+}} \|\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t))\|}{\sqrt{a_1}(a_3 - L_{\nabla V} L_\eta (1 + L_\pi))} \tag{3.75}$$

from (3.65). When the trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$ is periodic or has a finite length, this supremum can be easily computed numerically, such that (3.75) can be efficiently determined.

In order to demonstrate the practical applicability of Theorem 3.4, we consider again the example of a feedback linearizing controller designed using a GP mean function. While we assume a system of the form (3.53), we slightly simplify the scenario by restricting it to known functions $g(\cdot)$. Therefore, we only have to learn a GP model $\mu(\cdot)$ of the unknown function $\tilde{f}(\cdot)$, where we drop the index $\tilde{f}$ for notational simplicity. This allows us to define the controller by slightly adapting (3.56), which yields

$$\pi(\boldsymbol{x}, t) = \frac{1}{g(\boldsymbol{x})} \left(\pi_{\mathrm{lin}}(\boldsymbol{x}, t) - \mu(\boldsymbol{x}) \right). \tag{3.76}$$

Since this controller requires the reciprocal value of $g(\boldsymbol{x})$, we need to adapt 3.5 to ensure well-defined control inputs. This leads to the following assumption.

Assumption 3.7. *The function $g(\cdot)$ is known and positive, i.e., $g(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \mathbb{X}$.*

Using this assumption, it directly follows that the closed-loop dynamics of system (3.53) controlled by (3.76) with a reference trajectory of the form (3.3) are given by

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)) + \boldsymbol{b}(\tilde{f}(\boldsymbol{x}) - \mu(\boldsymbol{x})) + \dot{\boldsymbol{x}}_{\mathrm{ref}}(t). \tag{3.77}$$

As the uniform error bound for the unknown function $\tilde{f}(\cdot)$ does not depend on the control input u, it is possible to increase the decay rate a_3 without a potential increase of the linearization error. This allows the derivation of a straightforward condition to ensure (3.67) as shown in the following corollary.

Corollary 3.2. *Consider a dynamical system* (3.53) *with function $g(\cdot)$ satisfying Assumption 3.7, a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ with Lipschitz continuous kernel such that Assumption 2.6 is satisfied, and training data $\mathbb{D}$ with observation noise $\epsilon^{(n)}$, $n = 1, \ldots, N$, fulfilling Assumption 2.1. Moreover, assume that a controller* (3.76) *is used to track the bounded reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$. Let the symmetric, positive definite matrix $\boldsymbol{P} \in \mathbb{R}^{d_x \times d_x}$ denote the solution to the Lyapunov equation $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})^T \boldsymbol{P} + \boldsymbol{P} \boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = -\boldsymbol{I}$ for $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ defined in* (3.17)*. If*

$$\|\boldsymbol{P}\boldsymbol{b}\| < \frac{1}{2L_\eta}, \tag{3.78}$$

and $\mathbb{V}_{\xi(t)}(t) \subset \mathrm{int}(\mathbb{S})$ holds for all $t \in \mathbb{R}_{0,+}$, where

$$\xi(t) \leq \frac{2\sqrt{\bar{\lambda}(\boldsymbol{P})} \|\boldsymbol{P}\boldsymbol{b}\| \max\limits_{t' \in [0,t]} \|\boldsymbol{\eta}(\boldsymbol{z}_{\mathrm{ref}}(t'))\|}{(1 - 2\|\boldsymbol{P}\boldsymbol{b}\| L_\eta)}, \tag{3.79}$$

then, the closed-loop system (3.77) *admits a probabilistic tracking error bound $\upsilon(t) = \xi(t) / \sqrt{\underline{\lambda}(\boldsymbol{P})}$.*

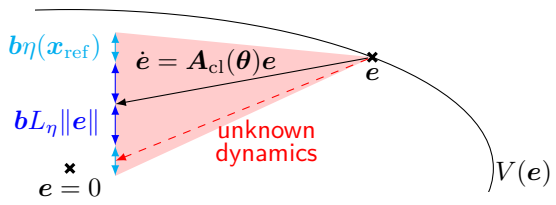

Figure 3.5.: Illustration of the linearization-based Lyapunov approach for feedback linearizing control laws: The linearization around the reference causes an additional linear error term in the Lyapunov derivative.

Proof. It is trivial to see that for $V(\boldsymbol{e}) = \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e}$, such that we have $a_1 = \underline{\lambda}(\boldsymbol{P})$ and $a_2 = \bar{\lambda}(\boldsymbol{P})$ in (3.65). Moreover, due to Corollary 3.1, we have

$$\dot{V}(\boldsymbol{e}) \leq -\|\boldsymbol{e}\|^2 + 2|\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{b}|\eta(\boldsymbol{x}), \tag{3.80}$$

such that $a_3 = 1$ holds in (3.66). Since this expression is independent of the control law, the linearization of the control is not necessary, and we can simply set $L_\pi = 0$. Finally, it is straightforward to see that

$$\|\nabla 2\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{b}\| = \|\boldsymbol{P}\boldsymbol{b}\|, \tag{3.81}$$

such that (3.78) and (3.79) immediately follow from Theorem 3.4 and (3.75). □

Since the linearization of the control law is not necessary due to Assumption 3.7, Corollary 3.2 relies on a decoupling of error sources as illustrated in Fig. 3.5. While the nominal Lyapunov component $-\|\boldsymbol{e}\|^2$ is guaranteed to be negative and thus $\dot{\boldsymbol{e}} = \boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta})\boldsymbol{e}$ points towards the inside of the Lyapunov sub-level set, the error bound along the reference trajectory $\eta(\boldsymbol{x}_{\text{ref}}(t))$ and the linearization error $L_\eta\|\boldsymbol{e}(t)\|$ cause an opposite effect. As long as the nominal component is dominating, a bounded tracking error can be ensured.

This straightforward approach immediately results in condition (3.78). In this condition, the matrix $\boldsymbol{P}$ is the parameter we can change since its values depend on the dynamics matrix $\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta})$. Due to the structure of $\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta})$ defined in (3.17), it is possible to achieve arbitrarily small values $\|\boldsymbol{P}\boldsymbol{b}\|$ through suitably high control gains $\boldsymbol{\theta}$. Therefore, (3.78) effectively restricts the admissible values of $\boldsymbol{\theta}$, such that the convergence to the reference trajectory is fast enough compared to the rate of change of the uniform error bound represented by L_η. While this can cause the inapplicability of Theorem 3.4 to systems that admit a probabilistic tracking error bound according to Theorem 3.3, it significantly simplifies the computation. Therefore, it provides a useful instrument when the computationally efficient evaluation of a probabilistic tracking error bound is required.

Remark 3.1. *It is important that* (3.79) *can only partially reflect the temporal behavior of the tracking error bound since it does not decrease if* $\|\boldsymbol{\eta}(\boldsymbol{z}_{\text{ref}}(\cdot))\|$ *reduces over time. Despite this limitation, it yields a practically useful error bound* $\upsilon(\cdot)$*, in particular for finite time intervals* $[t_1, t_2]$ *for* $t_1, t_2 \in \mathbb{R}_{0,+}$*. If it holds that* $\|\boldsymbol{e}(t_2)\| \leq \tilde{\xi}(t_2)/\sqrt{\underline{\lambda}(\boldsymbol{P})}$*, where*

$$\tilde{\xi}(t) \leq \frac{2\sqrt{\bar{\lambda}(\boldsymbol{P})}\|\boldsymbol{P}\boldsymbol{b}\|\|\boldsymbol{\eta}(\boldsymbol{z}_{\text{ref}}(t))\|}{(1 - 2\|\boldsymbol{P}\boldsymbol{b}\|L_\eta)}, \tag{3.82}$$

then, we can apply the Corollary 3.2 for another time interval $[t_2, t_3]$ for $t_3 \in \mathbb{R}_{0,+}$. Since the transfer of sub-level set guarantees to the tracking error bound $\upsilon(\cdot)$ is often conservative, this property is frequently satisfied in practice. Therefore, a sequential application of Corollary 3.2 can provide a practical method to reflect a decreasing tracking error bound. This approach leads to the heuristic tracking error bound $\tilde{\upsilon}(\cdot) = \tilde{\xi}(\cdot)/\sqrt{\underline{\lambda}(\boldsymbol{P})}$ for infinitesimally small time intervals.

3.2.4. Numerical Evaluation

We demonstrate the effectiveness of the derived theoretical results by applying them to the example in Section 3.1.4. Conforming with the notation in this section, this means we consider a system

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = \tilde{f}(\boldsymbol{x}) + g(\boldsymbol{x})u, \tag{3.83}$$

with $\tilde{f}(\boldsymbol{x}) = 1 - \sin(2x_1) + 1/(1+\exp(-x_2))$ and $g(\boldsymbol{x}) = 20\,(1 + 1/2 \sin(x_2/4))$ controlled by a feedback linearizing controller. For the beginning, we assume that both $\tilde{f}(\cdot)$ and $g(\cdot)$ are unknown and put a prior GP distribution $\mathcal{GP}(0, k_{\tilde{f}}(\cdot,\cdot))$ and $\mathcal{GP}(20, k_g(\cdot,\cdot))$ with SE kernels $k_{\tilde{f}}(\cdot,\cdot)$, $k_g(\cdot,\cdot)$ on them. Note that the prior mean for $g(\cdot)$ does not affect the theoretical results as discussed in Remark 2.2. We generate $N_x = 49$ training samples for the states $\boldsymbol{x}$ on a uniform grid over $[-2.5, 2.5]^2$ and $N_u = 2$ training samples for the control inputs u on a grid over $[0, 5]$. Training targets are determined for all combinations of training states and control inputs, which leads to a data set of size $N = 98$. The hyperparameters of the GPs are determined using log-likelihood maximization as explained in Section 2.1.5. Based on the resulting GP model, we define a feedback linearizing control law (3.56) with tracking controller (3.57), where we use $kc = 30$ and $\theta = 5$ to track a reference $\boldsymbol{x}_{\mathrm{ref}}(t) = [r(t)\ \dot{r}(t)]^T$ for $r(t) = 2\sin(t)$. For computing the tracking error bound in Corollary 3.1, we employ a uniform error bound based on Theorem 2.2 with $\delta = 0.01$. The required Lipschitz constants for the GP mean and standard deviation functions are computed numerically, while the Lipschitz constants for $\tilde{f}(\cdot)$ and $g(\cdot)$ are determined using Theorem 2.3. Moreover, the optimization problem (3.59) is solved by discretizing the state space using a uniform grid over $[-2.5, 2.5]^2$ with $N_{\mathrm{test},x} = 10000$ and the time interval $[0, 2\pi]$ with $N_{\mathrm{test},t} = 100$ uniformly spaced samples.

A snapshot of the resulting Lyapunov derivative bound (3.43) is illustrated in Fig. 3.6. Due to the uniform grid data, the uniform error bound is almost constant, such that the derivative bound is dominated by the quadratic growth of the Lyapunov candidate $V(\boldsymbol{e}) = \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e}$. This leads to merely a small sub-level set $\mathbb{V}_{\bar{V}}(t)$ around the reference position with possibly positive Lyapunov derivative bound. Due to the strong elliptical shape of this set, the resulting tracking error bound is rather conservative.

This can also be seen when comparing the tracking error bound $\upsilon(\cdot)$ obtained from Corollary 3.1 to the actually observed error as depicted in Fig. 3.7. The tracking error bound holds during the complete simulation but is rather conservative. This conservatism is caused by the shape of the sub-level set because the value of the Lyapunov candidate itself comes quite close to the bound $\bar{V}$ defined in (3.59). Therefore, the Lyapunov-based approach for deriving tracking error bounds provides a high flexibility but requires suitable Lyapunov candidates to achieve tight tracking error bounds.

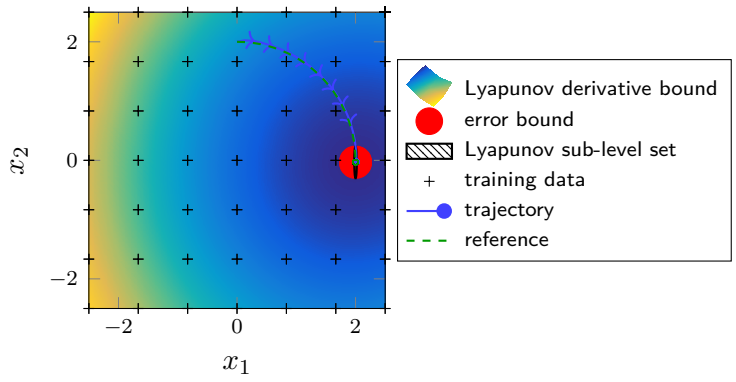

Figure 3.6.: Snapshot of the Lyapunov derivative bound and the tracking error bound for feedback linearization with a GP model resulting from Corollary 3.1.

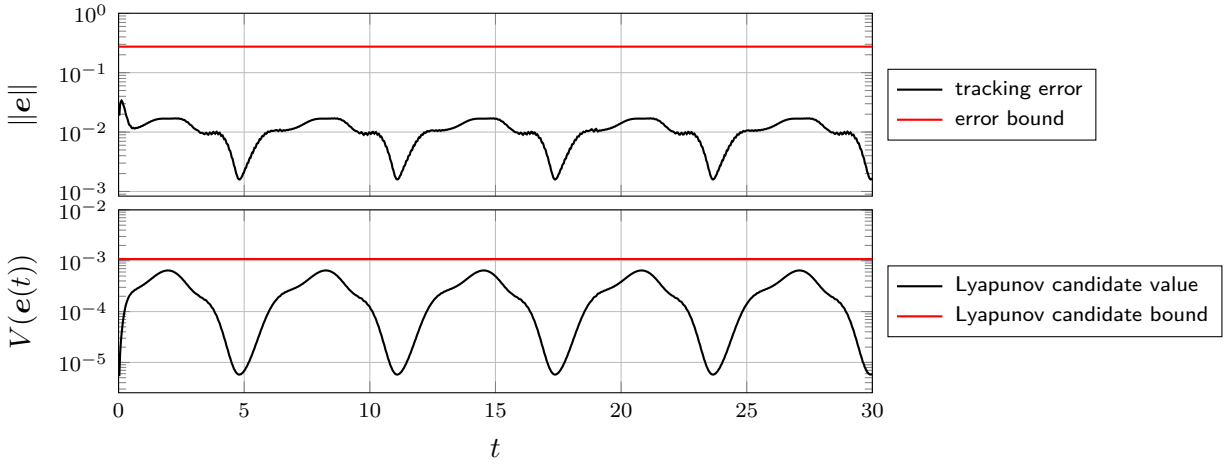

Figure 3.7.: Top: Lyapunov-based error bound computed using Corollary 3.1 in comparison to the true error. Bottom: Value of the Lyapunov candidate along trajectories together with its bound (3.59).

In addition to the general Lyapunov-based tracking error bound relying on Theorem 3.3, we also evaluate the computationally more efficient, linearization-based approach in Theorem 3.4. For this purpose, we consider the same setup as before but assume that $g(\cdot)$ is known. Therefore, we can employ (3.76) as feedback linearizing control law with a tracking controller (3.57), where we use $k_c = 100$ and $\theta = 1$. Moreover, we change the training data set of states $\boldsymbol{x}$ to a uniform grid $[0, 2.5] \times [-2.5, 2.5]$ in order to illustrate the effect of the training data density on the tracking error bound.

Snapshots of the heuristic error bound $\tilde{\xi}(\cdot)/\sqrt{\underline{\lambda}(\boldsymbol{P})}$ based on (3.82) for this setting are depicted in Fig. 3.8. When the GP variance is high, the uniform error bound is large, which results in a high value for the heuristic error bound $\tilde{\xi}(\cdot)/\sqrt{\underline{\lambda}(\boldsymbol{P})}$. Thereby, a direct relationship between data density and tracking error can be established.

As discussed in 3.1, this relationship can also be partially transferred to the certifiable tracking error bound in Corollary 3.2. For this purpose, (3.79) is evaluated for time intervals $[t, t + \Delta t]$. At the end of each interval, it is checked if the actual error is sufficiently small to admit the computation of (3.79) for the following time interval. This procedure is repeated until the end of the simulation time. The resulting error bounds for small time intervals with

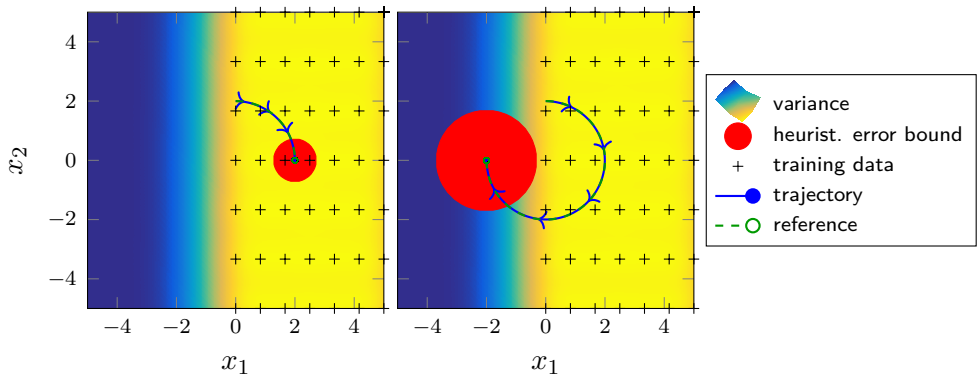

Figure 3.8.: Snapshots of the Lipschitz-based heuristic tracking error bound adapted from Corollary 3.2 in different uncertainty regimes. The tracking error bound is strongly related to the GP variance.

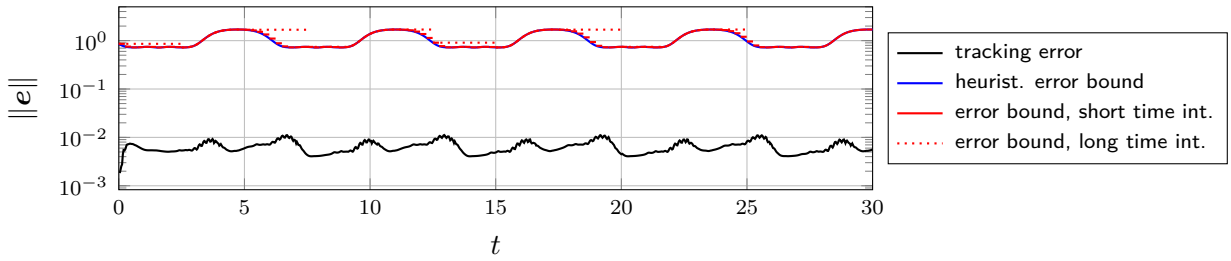

Figure 3.9.: Lipschitz-based tracking error bound (3.2) for different time interval lengths together with the limit case for infinitesimally small intervals (3.82) as heuristic error bound $\tilde{\xi}(\cdot)/\sqrt{\underline{\lambda}(\boldsymbol{P})}$.

$\Delta t = 0.25$ and large intervals with $\Delta t = 2.5$ are illustrated in Fig. 3.9. It can be clearly seen that large time intervals can sometimes lead to a significant increase in conservatism. The tracking error bound based on (3.79) for the short time interval is barely distinguishable from the heuristic, which emphasizes the practical strength of the heuristic (3.82) as a simple approximation. Overall, it must be noted that the computational efficiency obtained from the linearization-based approach in Theorem 3.4 comes at the cost of a significantly increased conservatism compared to the general approach of Theorem 3.3. Therefore, both approaches offer practical advantages, such that the suitable tracking error bound must be chosen dependent on the application requirements.

3.3. Discussion

The tracking error bounds in this chapter are applicable to a broad class of systems, but they are generally restricted to controllers designed for nominal dynamics defined through the GP mean function. Since the presented results are based on uniform error bounds for GP models, they can still be applied to certain forms of robust controllers, e.g., [26]. However, they are clearly not applicable to probabilistic design approaches as considered, e.g., in [105]. In general, there are many other approaches to design control laws for continuous-time systems using GP regression, such as black-box optimization [119] and sampling-based methods [120]. Moreover, other control goals such as optimality [121] and safety [81, 122] can be addressed,

too. Therefore, we are convinced that our results are an essential step towards a more unified theory for the analysis of GP-based control laws, but they are certainly only the beginning of a holistic perspective.

The approach based on linear systems theory offers a remarkably straightforward and effective analysis. While we only exploit this for the derivation of tracking error bounds, it can be similarly used for the investigation of safety conditions [123]. Moreover, it can be straightforwardly extended to scenarios other than input uncertainties, although this requires more sophisticated control laws than the direct nonlinearity compensation considered in Section 3.1. However, this flexible applicability to different problems comes at the price of a necessary restriction to systems behaving almost linearly.

For general dynamical systems, directly solving the differential equation is usually impossible, such that we need to represent relevant properties through a suitable proxy function. Lyapunov candidates are such proxies for tracking errors, which we exploit for different error bounds in Section 3.2. These ideas can be straightforwardly extended to safety concepts using control barrier functions [124], although it becomes necessary to explicitly take the GP model error into account. Due to the analysis using a proxy function, a broad class of nonlinear dynamical systems can be analyzed, but often the resulting guarantees are rather conservative. This is caused by the incapability of proxies to describe the full dynamic behavior as illustrated for the Lyapunov candidate in 3.2.4. In addition to this conservatism, proxies generally increase the complexity of the analysis, such that numerical methods are often required for solving the arising problems exactly. By employing suitable approximations, this weakness can be partially mitigated in practice. Therefore, theoretical properties of general dynamical systems with GP-based control laws can be effectively analyzed, but multiple practical challenges must be addressed.

Learning for Control with Arbitrary Accuracy Guarantees

4.

When employing Gaussian process regression for inferring a model of an unknown function, the model accuracy crucially depends on the available training data. In general, the relationship between the learning error and the training data is well understood [125], and the informativeness of training data with respect to the global approximation error can be effectively bounded using information-theoretic concepts [67]. However, such results have a global nature over a considered compact domain, which is agnostic of specific model accuracy requirements for the derivation of control performance guarantees. Therefore a control-oriented analysis of the training data requirements beyond uniform model approximation is necessary, similarly as in GP-based optimization [126].

On the one hand, this requires a fundamental understanding of how the training data locally affects uniform error bounds for GP regression, such that we can develop approaches to reduce it where necessary. On the other hand, we need to know where a high model accuracy is necessary to ensure high control performance guarantees. Such insight is necessary to develop effective learning strategies to ensure arbitrarily high tracking accuracy guarantees for GP-based control laws by generating suitable training data. Therefore, it is critical to achieving one of the key promises of data-driven control: certifiably improving the performance of control techniques using measurements of the controlled system.

In this section, we address these challenging problems by first investigating the local data dependency of uniform error bounds in Section 4.1. Based on the results derived in this section, we investigate the explicit data dependency of tracking accuracy guarantees in Section 4.2. In Section 4.3, the gained insights are used to develop several approaches for data generation in closed-loop control, which guarantee tracking error bounds for GP-based control. The chapter concludes by discussing the derived theoretical results in Section 4.4.

4.1. Data Dependency of Uniform Error Bounds

For investigating the data dependency of uniform error bounds for GP regression, we can make use of the structure of the bounds in Theorem 2.1 and Proposition 2.2, which allows us to decouple the dependency into two core components. On the one hand, the uniform error bounds directly depend on the training data, e.g., through the Gram matrix in Theorem 2.1 or simply the number of samples in Proposition 2.2. For RKHS based approaches, this dependency is well understood [68] and can be bounded through the maximum information gain [67]. In contrast, such an asymptotic relationship has not yet been established for Bayesian uniform error bounds and requires a suitable choice for the virtual grid constants.

On the other hand, the GP standard deviation, which is a core component of the presented uniform error bounds in Chapter 2, crucially depends on the training inputs. Bounds for the

standard deviation can be straightforwardly obtained for noise-free GP regression using results for kernel-based interpolation. By exploiting spectral properties of kernels, an effective analysis of the asymptotic behavior is possible [127], but such approaches are not suited to bound the GP standard deviation for fixed data sets. Moreover, classical results for scattered data approximation can be applied due to the equivalence of the posterior variance and the power function [62]. Therefore, classical results [59, 61] as well as newer findings [128, 129] can be directly used for noise-free GP regression. However, it is generally not clear how these results can be generalized to regression with noisy observations. Bounds for the case of general GP regression have mostly been derived as intermediate results. For example, a standard deviation bound for GPs with isotropic kernels has been shown in the context of Bayesian optimization [130] while bounds for general kernels have been investigated within the analysis of average learning curves [131, 132] and experimental design [133]. Although these bounds are well-suited for low data regimes, they fail to capture the asymptotic behavior. Therefore, upper bounds on the GP standard deviation are missing, which allow us to describe the learning behavior over the whole range of training data densities.

In order to address this lack of theoretical understanding about the relationship between training data and uniform error bounds, we derive an explicit dependency of the uniform error bound on a measure for the training data density. For this purpose, we propose a kernel-based measure to evaluate the training data density, whose flexibility we demonstrate by exemplarily illustrating it for squared exponential (SE), Matérn class and linear kernels. Moreover, we show that our considered uniform error bounds directly depend on this data density measure. This allows us to prove vanishing regression errors with growing data density using RKHS-based and Bayesian uniform error bounds.

The remainder of this section, which is based on our work [36], is structured as follows. The problem setting is formalized in Section 4.1.1. In Section 4.1.2, a novel bound for the asymptotic behavior of our Bayesian uniform error bounds in the limit of infinitely many training samples is derived, and an analogous result for RKHS-based bounds is restated. Since these results retain their dependency on the GP standard deviation, we develop a measure for the density of training data, which reflects the data requirements encoded through kernels, and prove a vanishing GP variance with growing data density in Section 4.1.3. Finally, we demonstrate the intuitiveness of the proposed density measure by exemplarily bounding it in terms of Euclidean distances for SE, Matérn class, and linear kernels in Section 4.1.4.

4.1.1. Problem Setting

Based on the probabilistic uniform error bounds $\eta(\cdot)$ derived in Chapter 2, our goal is the derivation of conditions on the training data $\mathbb{D}$ which ensure that the error of GP regression stays below a desired value $\bar{\eta} \in \mathbb{R}_+$. This can be formulated as the problem of finding a measure of data density $h : \mathbb{S} \to \mathbb{R}_+$, which is capable of capturing the dependency of the error bound $\eta(\cdot)$ on the data distribution. Since the error bounds $\eta(\cdot)$ derived in Chapter 2 strongly depend on the GP standard deviation $\sigma(\cdot)$, a suitable density measure $h(\cdot)$ must reflect the data dependency induced by the kernel $k(\cdot, \cdot)$. Moreover, the existence of a lower bound $\underline{h} \in \mathbb{R}_{0,+}$ of the density measure $h(\cdot)$ for a data set $\mathbb{D}$ must guarantee the implication

$$h(\boldsymbol{z}) \geq \underline{h} \qquad \Rightarrow \qquad \eta(\boldsymbol{z}) \leq \bar{\eta}(\underline{h}) \tag{4.1}$$

for a bounding function $\bar{\eta} : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$. As observation noise complicates the extraction of model knowledge from training data, the function $\bar{\eta}(\cdot)$ can change with the observation

noise variance σ_{on}^2, but it must vanish for an infinite data density regardless of it, i.e.,

$$\lim_{\underline{h} \to \infty} \bar{\eta}(\underline{h}) = 0. \tag{4.2}$$

Since these conditions ensure that arbitrary small uniform error bounds $\bar{\eta}(\underline{h})$ can be achieved using a data set $\mathbb{D}$ with sufficiently high density $h(\cdot)$, we consider the problem of deriving a data density measure $h(\cdot)$ along with the bounding function $\bar{\eta}(\cdot)$ satisfying (4.1) and (4.2) in this section.

4.1.2. Asymptotic Bounds for the Learning Error

In order to derive a data density measure $h(\cdot)$ satisfying (4.1) and (4.2), it is necessary to analyze the asymptotic behavior of the uniform error bound considered in Section 2.2 and Section 2.3 for $N \to \infty$. In order to limit the complexity of this analysis, we consider a constant GP standard deviation $\sigma(\cdot)$ in this subsection, such that we can focus on other data dependencies and remaining unspecified parameters. Moreover, we restrict our analysis to the stochastic noise scenario, e.g., Theorem 2.1 and Proposition 2.2, due to its admissibility for both RKHS-based and Bayesian uniform error bounds.

Due to the explicit dependency of the RKHS-based uniform error bounds in Section 2.2.2 on the kernel matrix, their asymptotic behavior is determined by the employed kernel $k(\cdot, \cdot)$. This is exemplarily shown for linear and SE ARD kernels in the following result, which is a direct consequence of [67, 68].

Theorem 4.1. *Consider an unknown function $f(\cdot) \in \mathbb{H}_k^{\mathbb{S}}$ satisfying Assumption 2.4 and a prior Gaussian process $\mathcal{GP}(0, k(\cdot, \cdot))$. Moreover, assume that the observation noise satisfies Assumption 2.5. Then, for every $\delta \in (0, 1)$, the posterior mean function $\mu(\cdot)$ defined in (2.26) admits a uniform error bound with probability $1 - \delta$ on the compact set $\mathbb{S} \subset \mathbb{R}^{d_z}$, whose asymptotic behavior*

1. *for linear kernels is given by*

$$\eta(\boldsymbol{z}) \in \mathcal{O}\left(\left(1 + \sqrt{\log\left(\frac{1}{\delta}\right)} + \sqrt{d_z \log(N)}\right) \sigma(\boldsymbol{z})\right); \tag{4.3}$$

2. *for SE ARD kernels is given by*

$$\eta(\boldsymbol{z}) \in \mathcal{O}\left(\left(1 + \sqrt{\log\left(\frac{1}{\delta}\right)} + (\log(N))^{\frac{d_z+1}{2}}\right) \sigma(\boldsymbol{z})\right). \tag{4.4}$$

Proof. Due to [68, Lemma 3], it holds that

$$\gamma_N \geq \log\left(\sqrt{\det\left(\boldsymbol{I}_N + \frac{1}{\sigma_{\mathrm{on}}^2} \boldsymbol{K}\right)}\right), \tag{4.5}$$

where γ_N is the maximum information gain defined as

$$\gamma_N = \max_{\boldsymbol{Z} \in \mathbb{R}^{d_z \times N} \in \mathbb{S}^N} I(\boldsymbol{y}_Z, \boldsymbol{f}_Z) \tag{4.6}$$

for $\boldsymbol{y}_Z = \boldsymbol{f}_Z + \boldsymbol{\epsilon}$ with $f_{Z,i} = f(\boldsymbol{x}_i)$ and $\boldsymbol{\epsilon}$ the concatenation of observation noise realizations. The function $I : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}_{0,+}$ is the mutual information, which is given by

$$I(\boldsymbol{y}_Z, \boldsymbol{f}_Z) = \frac{1}{2} \log(\det(\boldsymbol{I}_N + \sigma_{\text{on}}^2 \boldsymbol{K}_Z)) \tag{4.7}$$

for Gaussian random vectors $\boldsymbol{y}_Z$ and $\boldsymbol{f}_Z$. Using (4.5), Theorem 2.1 guarantees a probabilistic uniform error bound

$$\eta(\boldsymbol{z}) \leq \left(\Gamma + \frac{\sqrt{2}\tilde{\sigma}_{\text{on}}}{\sigma_{\text{on}}} \sqrt{\gamma_N + 1 + \log\left(\frac{1}{\delta}\right)} \right) \sigma(\boldsymbol{z}). \tag{4.8}$$

Finally, the result follows from upper bounds on the information gain γ_N derived in [67, Theorem 5], which results in (4.3) and (4.4). □

This result shows that an arbitrarily small uniform error bound $\eta(\boldsymbol{z})$ can be achieved if the GP standard deviation $\sigma(\boldsymbol{z})$ decays fast enough. The required decrease rate is reciprocal logarithmic, regardless if a SE ARD or a linear kernel is used. However, the choice of the kernel determines how the dimension of the training inputs affects the increase rate of the scaling factor for $\sigma(\cdot)$.

While the RKHS-based uniform error bound in Theorem 4.1 depends explicitly on the kernel matrix, the Bayesian bound in Proposition 2.2 depends on the unspecified parameter τ. Moreover, the required parameters of Hölder continuity are partially data-dependent. As shown in the following theorem, the degree of freedom offered by τ allows to compensate for the asymptotic behavior of Hölder coefficients.

Theorem 4.2. *Consider an unknown function $f(\cdot)$, a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ defined through a kernel $k(\cdot,\cdot)$ with continuous partial derivatives up to the fourth order such that Assumption 2.6 is satisfied, and training data $\mathbb{D}$ with observation noise $\epsilon^{(n)}$, $n = 1, \ldots, N$, fulfilling Assumption 2.1. Then, for every $\delta \in (0,1)$ and $\tau \in \mathcal{O}(1/N^2)$, the posterior mean function $\mu(\cdot)$ defined in* (2.26) *admits a probabilistic uniform error bound asymptotically behaving as*

$$\eta(\boldsymbol{z}) \in \mathcal{O}\left(\sqrt{d_z \log\left(\frac{N d_z}{\delta}\right)} \left(\sigma(\boldsymbol{z}) + \frac{1}{N} \right) \right). \tag{4.9}$$

Proof. Due to Proposition 2.2 it holds that

$$\eta(\boldsymbol{z}) = \sqrt{2 \log\left(\frac{M(\tau, \mathbb{S})}{\delta}\right)} \sigma(\boldsymbol{z}) + L_\mu \tau^{p_\mu} + L_f \tau^{p_f} + L_\sigma \tau^{p_\sigma} \sqrt{2 \log\left(\frac{M(\tau, \mathbb{S})}{\delta_\eta}\right)} \tag{4.10}$$

with probability of at least $1 - \delta_\eta$ for $\delta_\eta \in (0,1)$. A trivial bound for the covering number can be obtained by considering a uniform grid over the cube containing $\mathbb{X}$. This approach leads to

$$M(\tau, \tilde{\mathbb{S}}) \leq \left(1 + \sqrt{d_z} \frac{\max\limits_{\boldsymbol{z}, \boldsymbol{z}' \in \mathbb{S}} \|\boldsymbol{z} - \boldsymbol{z}'\|_\infty}{2\tau} \right)^{d_z}. \tag{4.11}$$

Therefore, we have

$$\sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)}\sigma(\boldsymbol{z}) \in \mathcal{O}\left(\sqrt{d_z \log\left(\frac{d_z}{\tau\delta_\eta}\right)}\sigma(\boldsymbol{z})\right) \tag{4.12}$$

for $N \to \infty$ and $\tau \to 0$. In order to derive a bound for the parameters of Hölder continuity, we employ the assumed differentiability of the kernel $k(\cdot,\cdot)$, which implies its Lipschitz continuity on the compact set $\mathbb{S}$. Moreover, it guarantees the existence of a probabilistic Lipschitz constant L_f for the unknown function $f(\cdot)$ on $\mathbb{S}$ due Theorem 2.3, such that $p_f = 1$ and

$$L_f\tau \in \mathcal{O}\left(\sqrt{d_z \log\left(\frac{d_z}{\delta_L}\right)}\tau\right) \tag{4.13}$$

holds with probability $1-\delta_L$ for $\delta_L \in (0,1)$ for $\tau \to 0$. We can similarly derive a bound for the Hölder coefficient of the GP mean $\mu(\cdot)$. The continuous differentiability of the kernel $k(\cdot,\cdot)$ implies its Lipschitz continuity on $\mathbb{S}$, i.e., $p_k = 1$. Furthermore, Lemma 2.2 ensures that the Hölder coefficient, which is the Lipschitz constant of $\mu(\cdot)$, is given by

$$L_\mu \leq L_k\sqrt{N}\left\|(\boldsymbol{K}+\sigma_{\text{on}}^2\boldsymbol{I}_N)^{-1}\boldsymbol{y}\right\|. \tag{4.14}$$

Since the Gram matrix $\boldsymbol{K}$ is positive semidefinite and $f(\cdot)$ is bounded by some $\bar{f}$ due to Lipschitz continuity and a compact domain $\mathbb{X}$, we can bound $\|(\boldsymbol{K}+\sigma_{\text{on}}^2\boldsymbol{I}_N)^{-1}\boldsymbol{y}\|$ by

$$\begin{aligned}\left\|(\boldsymbol{K}+\sigma_{\text{on}}^2\boldsymbol{I}_N)^{-1}\boldsymbol{y}\right\| &\leq \frac{\|\boldsymbol{y}\|}{\underline{\lambda}(\boldsymbol{K}+\sigma_{\text{on}}^2\boldsymbol{I}_N)} \\ &\leq \frac{\sqrt{N}\bar{f}+\|\boldsymbol{\epsilon}\|}{\sigma_{\text{on}}^2},\end{aligned} \tag{4.15}$$

where $\boldsymbol{\epsilon}$ is a vector of N i.i.d. zero mean Gaussian random variables with variance σ_{on}^2. Therefore, it follows that $\frac{\|\boldsymbol{\epsilon}\|^2}{\sigma_{\text{on}}^2} \sim \chi_N^2$. Due to [70], with probability of at least $1-\delta_\epsilon$, $\delta_\epsilon \in (0,1)$, we have

$$\|\boldsymbol{\epsilon}\|^2 \leq \left(2\sqrt{N\log\left(\frac{1}{\delta_\epsilon}\right)}+2\log\left(\frac{1}{\delta_\epsilon}\right)+N\right)\sigma_{\text{on}}^2. \tag{4.16}$$

Hence, the Lipschitz constant of the posterior mean function $\mu(\cdot)$ satisfies with probability of at least $1-\delta_\epsilon$

$$L_\mu\tau \in \mathcal{O}\left(N\tau\log\left(\frac{1}{\delta_\epsilon}\right)\right). \tag{4.17}$$

For the GP standard deviation $\sigma(\cdot)$, Hölder continuity directly follows from Lemma 2.3 with order $p_\sigma = 1/2$ and coefficient

$$L_\sigma = \sqrt{2L_k}, \tag{4.18}$$

such that we directly obtain

$$L_\sigma \tau^{\frac{1}{2}} \sqrt{2 \log\left(\frac{M(\tau, \mathbb{S})}{\delta_\eta}\right)} \in \mathcal{O}\left(\sqrt{d_z \tau \log\left(\frac{d_z}{\tau \delta_\eta}\right)}\right). \tag{4.19}$$

Finally, choose $\delta_\eta = \delta_L = \delta_\epsilon = \delta/3$ and $\tau \in \mathcal{O}(1/N^2)$ to compensate for the linear growth of (4.17) with respect to the number samples N, which directly results in (4.9). □

Due to the linear dependency of the bound for the Lipschitz constant L_μ on the number of training samples, the virtual grid constant must decay faster than $\mathcal{O}(1/N)$. This, in turn, leads to a logarithmic growth of the scaling factor of $\sigma(\boldsymbol{z})$, such that a reciprocal logarithmic decrease of the GP standard deviation becomes necessary for achieving a vanishing uniform error bound. Therefore, the decay requirements for Bayesian and RKHS-based error bounds are similar, although the specific behavior is slightly different.

4.1.3. Asymptotic Bounds for the Posterior Variance

In order to compensate for the growth of the scaling factor of $\sigma(\boldsymbol{z})$ in Theorem 4.1 and Theorem 4.2, a sufficiently fast decay of the standard deviation $\sigma(\boldsymbol{z})$ must be ensured. Therefore, we investigate the behavior of the posterior variance $\sigma^2(\boldsymbol{z})$ depending on the training data density of an input data set $\mathbb{D}^z = \{\boldsymbol{z}^{(n)}\}_{n=1}^N$. The starting point of this analysis is the following lemma, which provides a straightforward upper bound for the posterior variance $\sigma^2(\boldsymbol{z})$.

Lemma 4.1. *Consider a GP trained using a data set with input training samples $\mathbb{D}^z$. Then, the posterior variance is bounded by*

$$\sigma^2(\boldsymbol{z}) \leq \frac{\sigma_{\text{on}}^2 k(\boldsymbol{z}, \boldsymbol{z}) + N \Delta k(\boldsymbol{z})}{N \max\limits_{\boldsymbol{z}' \in \mathbb{D}^z} k(\boldsymbol{z}', \boldsymbol{z}') + \sigma_{\text{on}}^2}, \tag{4.20}$$

where

$$\Delta k(\boldsymbol{z}) = k(\boldsymbol{z}, \boldsymbol{z}) \max_{\boldsymbol{z}' \in \mathbb{D}^z} k(\boldsymbol{z}', \boldsymbol{z}') - \min_{\boldsymbol{z}' \in \mathbb{D}^z} k^2(\boldsymbol{z}', \boldsymbol{z}). \tag{4.21}$$

Proof. Since $\boldsymbol{K} + \sigma_{\text{on}}^2 \boldsymbol{I}_N$ is a positive definite, quadratic matrix, it follows that

$$\sigma^2(\boldsymbol{z}) \leq k(\boldsymbol{z}, \boldsymbol{z}) - \frac{\|\boldsymbol{k}(\boldsymbol{z})\|^2}{\bar{\lambda}(\boldsymbol{K}) + \sigma_{\text{on}}^2}. \tag{4.22}$$

Applying the Gershgorin theorem [134], the maximal eigenvalue is bounded by

$$\bar{\lambda}(\boldsymbol{K}) \leq N \max_{\boldsymbol{z}' \in \mathbb{D}^z} k(\boldsymbol{z}', \boldsymbol{z}'). \tag{4.23}$$

Furthermore, due to the definition of $\boldsymbol{k}(\boldsymbol{z})$ we have

$$\|\boldsymbol{k}(\boldsymbol{z})\|^2 \geq N \min_{\boldsymbol{z}' \in \mathbb{D}^z} k^2(\boldsymbol{z}', \boldsymbol{z}). \tag{4.24}$$

Therefore, $\sigma^2(\boldsymbol{z})$ can be bounded by

$$\sigma^2(\boldsymbol{z}) \leq k(\boldsymbol{z}, \boldsymbol{z}) - \frac{N \min\limits_{\boldsymbol{z}' \in \mathbb{D}^z} k^2(\boldsymbol{z}', \boldsymbol{z})}{N \max\limits_{\boldsymbol{z}' \in \mathbb{D}^z} k(\boldsymbol{z}', \boldsymbol{z}') + \sigma_{\text{on}}^2}. \tag{4.25}$$

Finally, the proof follows from the definition of $\Delta k(\boldsymbol{z})$. □

This theorem does not pose any restriction on the employed kernel but strongly depends on the particular choice of kernel. Therefore, it can be difficult to interpret. However, it can be significantly simplified for specific kernels, as shown in the following corollary for stationary covariance functions.

Corollary 4.1. *Consider a GP with stationary kernel and input training samples $\mathbb{D}^z$. Then, the posterior variance is bounded by*

$$\sigma^2(\boldsymbol{z}) \leq k(0) - \frac{\min\limits_{\boldsymbol{z}' \in \mathbb{D}^z} k^2(\boldsymbol{z} - \boldsymbol{z}')}{k(0) + \frac{\sigma_{\mathrm{on}}^2}{N}}. \tag{4.26}$$

Proof. The proof follows directly from Lemma 4.1 and the fact that $\max_{\boldsymbol{z}' \in \mathbb{D}^z} k(\boldsymbol{z}', \boldsymbol{z}') = k(\mathbf{0})$ since the kernel is stationary. □

In this special case of Lemma 4.1, which has been previously stated, e.g., in [130], the kernel induces a notion of proximity, where the absence of training inputs $\boldsymbol{z}'$ with $k(\boldsymbol{z}-\boldsymbol{z}') \approx 0$ leads to a large bound for the posterior variance $\sigma^2(\boldsymbol{z})$. Therefore, this corollary shows that it is desirable to have data close to the test point $\boldsymbol{z}$ as measured by $k(\cdot)$ for stationary kernels.

Since Lemma 4.1 and Corollary 4.1 still consider the full input data set $\mathbb{D}^z$, a single sample with $k(\boldsymbol{z}', \boldsymbol{z}) \approx 0$ can practically lead to the trivial bound $\sigma^2(\boldsymbol{z}) \lesssim k(\boldsymbol{z}, \boldsymbol{z})$. This is clearly an undesired behavior for a bound since it would imply that additional data can potentially increase the GP variance bound. In order to avoid this effect, we make use of an important property of GP posterior variances, which is the fact that $\sigma^2(\boldsymbol{z})$ is non-increasing with the number of training samples N [135]. Therefore, we can consider subsets of $\mathbb{D}^z$ to compute the GP variance bounds in Lemma 4.1 and Corollary 4.1, which exclude these training samples with a negative effect on the bound. Due to the importance of $\Delta k(\boldsymbol{z})$ for these bounds, we make use of the following subset

$$\mathbb{K}_h(\boldsymbol{z}) = \{\boldsymbol{z}' \in \mathbb{D}^z : k^2(\boldsymbol{z}, \boldsymbol{z}) \leq k^2(\boldsymbol{z}', \boldsymbol{z}') \leq \frac{1}{h} + k^2(\boldsymbol{z}', \boldsymbol{z})\} \tag{4.27}$$

for this purpose, where $h \in \mathbb{R}_+$. It can be easily seen that considering only the subset $\mathbb{K}_h(\boldsymbol{z}) \subset \mathbb{D}^z$ in (4.21) ensures

$$k(\boldsymbol{z}, \boldsymbol{z}) \max_{\boldsymbol{z}' \in \mathbb{K}_h(\boldsymbol{z})} k(\boldsymbol{z}', \boldsymbol{z}') - \min_{\boldsymbol{z}' \mathbb{K}_h(\boldsymbol{z})} k^2(\boldsymbol{z}', \boldsymbol{z}) \leq \frac{1}{h}. \tag{4.28}$$

Since the consideration of a subset of $\mathbb{D}^z$ also reduces the number of considered training samples in (4.20), we trade-off the size of $\mathbb{K}_h(\boldsymbol{z})$ and the ensured value for $\Delta k(\boldsymbol{z})$ by defining h using the following optimization problem

$$h(\boldsymbol{z}) = \max_{h \in \mathbb{R}_+} h \tag{4.29}$$

$$\text{such that } |\mathbb{K}_h(\boldsymbol{z})| \geq h\sigma_{\mathrm{on}}^2 k(\boldsymbol{z}, \boldsymbol{z}). \tag{4.30}$$

It can easily be seen that $h(\boldsymbol{z})$ is well-defined since the optimization problem is always feasible for $h \to 0$. Moreover, it can be directly used as a measure of data density, as shown in the following proposition.

Proposition 4.1. *Consider a Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ defined by the kernel $k(\cdot,\cdot)$. If $k(\boldsymbol{z},\boldsymbol{z}) \neq 0$, the posterior standard deviation at $\boldsymbol{z}$ satisfies*

$$\sigma(\boldsymbol{z}) \leq \sqrt{\frac{2}{h(\boldsymbol{z})k(\boldsymbol{z},\boldsymbol{z})}}, \tag{4.31}$$

such that it behaves as $\sigma(\boldsymbol{z}) \in \mathcal{O}(1/\sqrt{h(\boldsymbol{z})})$.

Proof. By exploiting the fact that the posterior variance $\sigma^2(\boldsymbol{z})$ is non-increasing with the number of training samples N [135] and considering only samples inside the set $\mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})$ for the computation of the posterior standard deviation, we obtain

$$\sigma^2(\boldsymbol{z}) \leq \frac{\sigma_{\text{on}}^2 k(\boldsymbol{z},\boldsymbol{z}) + |\mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})|\Delta k(\boldsymbol{z})}{|\mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})| \max\limits_{\boldsymbol{z}' \in \mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})} k(\boldsymbol{z}',\boldsymbol{z}') + \sigma_{\text{on}}^2} \tag{4.32}$$

due to Lemma 4.1. Since $\boldsymbol{z}' \in \mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})$ implies $k(\boldsymbol{z}',\boldsymbol{z}') \geq k(\boldsymbol{z},\boldsymbol{z})$, we can simplify this expression to

$$\sigma^2(\boldsymbol{z}) \leq \frac{\sigma_{\text{on}}^2}{|\mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})|} + \frac{\Delta k(\boldsymbol{z})}{k(\boldsymbol{z},\boldsymbol{z})}. \tag{4.33}$$

Moreover, it can be straightforwardly checked that the restriction to $\mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})$ implies $\Delta k(\boldsymbol{z}) \leq 1/h(\boldsymbol{z})$, which yields

$$\sigma^2(\boldsymbol{z}) \leq \frac{\sigma_{\text{on}}^2}{|\mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})|} + \frac{1}{h(\boldsymbol{z})k(\boldsymbol{z},\boldsymbol{z})}. \tag{4.34}$$

Since $|\mathbb{K}_{h(\boldsymbol{z})}(\boldsymbol{z})|$ is lower bounded by $h(\boldsymbol{z})\sigma_{\text{on}}^2 k(\boldsymbol{z},\boldsymbol{z})$ by definition, we obtain

$$\sigma^2(\boldsymbol{z}) \leq \frac{2}{h(\boldsymbol{z})k(\boldsymbol{z},\boldsymbol{z})}, \tag{4.35}$$

which directly implies $\sigma(\boldsymbol{z}) \in \mathcal{O}(1/\sqrt{h(\boldsymbol{z})})$, concluding the proof. □

It can be clearly seen that $h(\boldsymbol{z})$ is a measure of data density that is highly specific for each particular GP and, therefore, capable of reflecting the requirements on good data distributions posed by the employed kernel $k(\cdot,\cdot)$. Moreover, it immediately follows from Theorem 4.2 that a sufficiently fast growth of $h(\boldsymbol{z})$ guarantees a vanishing error bound $\eta(\boldsymbol{z})$, e.g., $h(\boldsymbol{z}) \notin \mathcal{O}(\log(N))$ for the Bayesian uniform error bound in Proposition 2.2. Therefore, $h(\cdot)$ satisfies the requirements posed on a suitable measure of data density in Section 4.1.1.

4.1.4. Conditions for Specific Kernels

The high flexibility of Proposition 4.1 allows its application to GPs with arbitrary kernels but comes at the price of a difficult interpretation. However, when we fix a specific kernel, it is often possible to derive more accessible and intuitive subsets contained in $\mathbb{K}_h(\boldsymbol{z})$, as shown in the following lemma for linear, squared exponential and Matérn class kernels.

Lemma 4.2. *Geometrically interpretable subsets of $\mathbb{K}_h(\boldsymbol{z})$ defined in (4.27) are given by*

1. *the set*

$$\tilde{\mathbb{K}}_h^c(\boldsymbol{z}) = \Big\{\boldsymbol{z}' \in \mathbb{D}^z : \|\boldsymbol{z}'\|^2(\|\boldsymbol{z}'\|^2 - c\|\boldsymbol{z}\|^2) \leq \frac{1}{h}, \quad \|\boldsymbol{z}\| \leq \|\boldsymbol{z}'\|, \left|\boldsymbol{z}^T\boldsymbol{z}'\right| \geq c\|\boldsymbol{z}\|\|\boldsymbol{z}'\|\Big\} \subset \mathbb{K}_h(\boldsymbol{z}) \tag{4.36}$$

   with any $c \in (0,1)$ *for linear kernels;*

2. *the Euclidean ball*

$$\mathbb{B}_{\sqrt{1/2L_{\partial k}\sigma_f^2 h}}(\boldsymbol{z}) = \left\{\boldsymbol{z}' \in \mathbb{D}^z \colon \|\boldsymbol{z} - \boldsymbol{z}'\| \leq \sqrt{\frac{1}{2L_{\partial k}\sigma_f^2 h}}\right\} \subset \mathbb{K}_h(\boldsymbol{z}) \tag{4.37}$$

   for isotropic SE kernels $k_{\mathrm{SE}}(\cdot,\cdot)$ *and Matérn kernels* $k_{3/2}(\cdot,\cdot)$, $k_{5/2}(\cdot,\cdot)$, *where* $\sigma_f^2 = k(\boldsymbol{z},\boldsymbol{z})$.

Proof. Due to the definition of the linear kernel, we have the identity

$$k^2(\boldsymbol{z}',\boldsymbol{z}') - k^2(\boldsymbol{z}',\boldsymbol{z}) = \|\boldsymbol{z}'\|^4 - (\boldsymbol{z}^T\boldsymbol{z}')^2. \tag{4.38}$$

For ${}^{|\boldsymbol{z}^T\boldsymbol{z}'|}/_{(\|\boldsymbol{z}\|\|\boldsymbol{z}'\|)} \geq c$, we therefore obtain

$$k^2(\boldsymbol{z}',\boldsymbol{z}') - k^2(\boldsymbol{z}',\boldsymbol{z}) \leq \|\boldsymbol{z}'\|^2\left(\|\boldsymbol{z}'\|^2 - c\|\boldsymbol{z}\|^2\right). \tag{4.39}$$

Finally, the first inequality in (4.27) yields the requirement

$$k^2(\boldsymbol{z},\boldsymbol{z}) = \|\boldsymbol{z}\|^4 \leq \|\boldsymbol{z}'\|^4 = k^2(\boldsymbol{z}',\boldsymbol{z}'), \tag{4.40}$$

which concludes the first part of the proof. For the second part of the proof, we exploit the continuous differentiability of the considered kernels together with the fact that their derivative at $\boldsymbol{z} - \boldsymbol{z}' = \boldsymbol{0}$ is 0. Therefore, we have

$$k(\boldsymbol{z} - \boldsymbol{z}') \geq \sigma_f^2 - L_{\partial k}\|\boldsymbol{z} - \boldsymbol{z}'\|^2. \tag{4.41}$$

where $L_{\partial k} \in \mathbb{R}_+$ is the Lipschitz constant of the kernel derivative. Using this lower bound, we obtain

$$k^2(\boldsymbol{0}) - k^2(\boldsymbol{z} - \boldsymbol{z}') \leq 2L_{\partial k}\sigma_f^2\|\boldsymbol{z} - \boldsymbol{z}'\|^2 - L_{\partial k}^2\|\boldsymbol{z} - \boldsymbol{z}'\|^4, \tag{4.42}$$

which we can simplify to

$$k^2(\boldsymbol{0}) - k^2(\boldsymbol{z} - \boldsymbol{z}') \leq 2L_{\partial k}\sigma_f^2\|\boldsymbol{z} - \boldsymbol{z}'\|^2 \tag{4.43}$$

due to the non-negativity of the norm. Therefore, $\|\boldsymbol{z} - \boldsymbol{z}'\|^2 \leq {}^{h}/_{2L_{\partial k}\sigma_f^2}$ implies $|k^2(\boldsymbol{z},\boldsymbol{z}) - k^2(\boldsymbol{z},\boldsymbol{z}')| \leq h$. Since $k(\boldsymbol{z},\boldsymbol{z}) = k(\boldsymbol{z}',\boldsymbol{z}')$ for isotropic kernels, the first inequality is always satisfied, concluding the proof.

□

Figure 4.1.: Illustration of the set $\mathbb{K}_h(\boldsymbol{z})$ and geometrically simple subsets for a linear and a SE kernel.

This lemma illustrates the flexibility of quantifying the data density using $\mathbb{K}_h(\boldsymbol{z})$. While this set can be inner-approximated by a ball for Matérn and SE kernels as illustrated in Fig. 4.1, it looks more like segments of a sphere for linear kernels. Since we can easily determine the volume of such simple geometrical structures, Lemma 4.2 enables the derivation of a straightforward relationship between the sampling distributions and data density $h(\boldsymbol{z})$. For example, when training samples in $\mathbb{D}^z$ are generated by drawing from a uniform distribution, the number of points in a Euclidean ball is proportional to the volume of the ball, i.e., $\mathbb{B}_h(\boldsymbol{z}) \propto {}^{N}/_{h^{d_z}}$. Therefore, it follows from (4.30) that $h(\boldsymbol{z}) \in \mathcal{O}(N^{1/d_z+1})$ for SE or Matérn kernels with uniformly drawn input training samples. This in turn implies that $\sigma(\boldsymbol{z}) \in \mathcal{O}({}^{1}/_{N^{1/2d_z+2}})$ due to Proposition 4.1 [136, 137]. Hence, we obtain for the Bayesian error bound due to Theorem 4.2 that

$$\eta(\boldsymbol{z}) \in \mathcal{O}\left(\sqrt{\frac{d_z \log(N d_z)}{N^{1/1d_z+1}}}\right), \tag{4.44}$$

where we neglect the dependency on δ for notational convenience. This demonstrates the flexibility and effectiveness of the derived formalism for bounding the asymptotic decay of the prediction error bounds $\eta(\boldsymbol{z})$ presented in this section.

4.2. The Role of Data for Control-Theoretic Guarantees

Since the results in the previous section provide insights into the effect of training data on uniform error bounds for GP regression, it immediately arises the question about the implications of these results on the guaranteed control performance of learning-based control with GP models. While a global model accuracy perspective [125] allows to directly draw conclusions using the results in Chapter 3, such an approach is task-agnostic and known to be potentially sub-optimal for problems beyond modeling [67]. Therefore, a control-oriented analysis of the role of training data is generally necessary.

This is a problem that has received comparatively little attention in research. In the control-oriented literature, it is frequently assumed that there simply exists a training data set, without even considering the problem of finding out how such a data set should look like. This often leads to data sets in simulations that are chosen based on global model accuracy considerations [34, 109] or ease of practical realizability [26, 138]. Due to this lack

of understanding of the relationship between data distributions and control performance, random sampling-based approaches have recently been employed to estimate the effect of data on learning-based control systems [139]. Using a receding horizon formulation, these approaches can be extended to implicitly reflect the relevance of future training samples online [140, 141]. Even though such approaches can admit a rigorous theoretical analysis, they are computationally expensive, such that their application is usually limited to small-scale problems. Moreover, they do not provide direct insight into the interrelation between training data and control performance. Therefore, understanding the control performance's data dependency is usually limited to a simple qualitative relationship: the more data, the better the control performance [30, 109].

In order to improve the understanding of the interrelation between training data and control performance guarantees, we exemplarily investigate conditions for the training data distribution to ensure a desired error bound. First, this analysis is executed for the case of linear systems with input perturbations. We show that data is required only in proximity to the reference trajectory due to the structure of the tracking error bound for this example and derive the asymptotic convergence rate of the tracking error bound in dependency on the data density. While the required optimal data distribution is apparent for this example, it cannot be trivially seen for the Lyapunov-based tracking error bounds. Therefore, we analyze the training distribution requirements depending on the Lyapunov functions, which leads to an intuitive quantity for determining the suitability of training data distribution for control tasks specified through a Lyapunov candidate. Thereby, we show that the ideal training data distribution generally depends on the considered control task.

The remainder of this section is structured as follows. The problem setting is specified in Section 4.2.1. In Section 4.2.2, which is based on [36], the decay rate of the tracking error bound is derived for linear systems with input perturbation compensated using a GP model. Requirements on the training data distribution for data-efficient learning are shown for a feedback linearizing control law with quadratic Lyapunov function in Section 4.2.3. This section is comprised of adapted results from our prior work [37]. Finally, the presented insights are illustrated on simulation examples in Section 4.2.4.

4.2.1. Problem Setting

Due to the construction of the control laws in Sections 3.1 and 3.2, they achieve exact tracking if the uniform error bound $\eta(\cdot)$ is equal to zero. However, this is merely a sufficient condition, and it cannot be straightforwardly seen how the probabilistic uniform error bound must behave to ensure a desired upper bound $\bar{e} \in \mathbb{R}_+$ for a probabilistic tracking error bound $\upsilon(\cdot)$, i.e.,

$$\upsilon(t) \leq \bar{e}. \tag{4.45}$$

The requirements on the training data density $h(\cdot)$ are even more unclear, such that the role of training data $\mathbb{D}$ for control-theoretic guarantees is generally poorly understood for control based on GP models. In order to increase the understanding of this relationship, we consider the problem of deriving conditions for the training data density measured based on $h(\cdot)$, such that (4.45) can be achieved. This means that we want to find lower bounds $\underline{h} : \mathbb{S} \to \mathbb{R}_{0,+}$ for the data density $h(\cdot)$ such that the implication

$$h(\boldsymbol{z}) \geq \underline{h}(\boldsymbol{z}) \qquad \Rightarrow \qquad \upsilon(t) \leq \bar{e} \tag{4.46}$$

holds. This problem is exemplarily investigated for nonlinearity compensation as presented in Section 3.1 and the feedback linearizing control law in Section 3.2. Hence, the specific problem settings of these sections apply to the corresponding subsections in the following.

4.2.2. Asymptotic Tracking Error Bound

We first investigate the data dependency of the tracking error bounds derived in Section 3.1. Therefore, we follow the problem setting described in Section 3.1.1 and consider the example of a linear dynamical system

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}(u + f(\boldsymbol{x})), \tag{3.2 revisited}$$

which is perturbed by an unknown nonlinearity $f(\cdot)$, such that a GP model $\hat{f}(\cdot) = \mu(\cdot)$ is employed to compensate it using a control law

$$u(t) = -\boldsymbol{\theta}^T(\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{ref}}(t)) + r_{\mathrm{ref}}(t) - \hat{f}(\boldsymbol{x}(t)). \tag{3.4 revisited}$$

This results in the closed-loop error dynamics

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})\boldsymbol{e} + \boldsymbol{b}(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})), \tag{3.5 revisited}$$

where $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \boldsymbol{A} - \boldsymbol{b}\boldsymbol{\theta}^T$. Due to Proposition 3.1, a probabilistic tracking error bound $v(\cdot)$ can be computed in closed-form in this scenario, which allows the direct observation of a linear relationship between $\bar{v}$ and $\sup_{t\geq 0} \eta(\boldsymbol{x}_{\mathrm{ref}}(t))$. Since this implies that the GP standard deviation $\sigma(\cdot)$ along the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$ is dominating the behavior of the uniform error bounds in Theorem 2.1 and Proposition 2.2, it directly follows from Proposition 4.1 that we can also restrict the analysis of the data density $h(\cdot)$ to the reference trajectory. By considering only the worst case density, this leads to the value $\underline{h}_{\mathrm{ref}} = \inf_{t\geq 0} h(\boldsymbol{x}_{\mathrm{ref}}(t))$ as a measure of data quality. As exemplarily shown in the following result for uniform error bounds obtained from Proposition 2.2, $\underline{h}_{\mathrm{ref}}$ allows the straightforward derivation of asymptotic conditions on the data distribution to achieve (4.45).

Theorem 4.3. *Consider a system* (3.2) *satisfying Assumption 3.1, to which a control law* (3.4) *with gains* $\boldsymbol{\theta}$ *satisfying Assumption 3.3 is applied to track a bounded reference trajectory* $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$*. Assume that a Gaussian process* $\mathcal{GP}(0, k(\cdot,\cdot))$ *with Lipschitz continuous, stationary kernel* $k(\cdot,\cdot)$ *is used to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *of the unknown, Lipschitz continuous function* $f(\cdot)$*, such that Assumptions 2.1 and 2.6 are satisfied. Then, for* $\mathbb{X}_0 = \{\boldsymbol{x}_{\mathrm{ref}}(0)\}$*, ensuring* (4.45) *asymptotically requires at most*

$$\underline{h}_{\mathrm{ref}} \in \mathcal{O}\left(\frac{1}{\bar{e}^2}\right), \tag{4.47}$$

$$-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) \in \mathcal{O}\left(\sqrt{\log\left(\frac{N}{\bar{e}}\right)}\right). \tag{4.48}$$

Proof. Choose $\boldsymbol{\theta}$ such that

$$\kappa = \frac{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|\sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)}} \tag{4.49}$$

is constant and (3.13) is satisfied. Due to Proposition 3.1 and Proposition 2.2, this implies the existence of a probabilistic tracking error bound $\upsilon(\cdot) = \bar{\upsilon}$ on a suitable compact set $\mathbb{S}$, where

$$\bar{\upsilon} = -\frac{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) + L_\eta\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|} \sup_{t\geq 0} \eta(\boldsymbol{x}_{\mathrm{ref}}(t)). \tag{4.50}$$

It directly follows from the definition of $\eta(\cdot)$ in (2.63) that its Lipschitz constant is given by

$$L_\eta = L_\sigma \sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)} \tag{4.51}$$

where L_σ is the Lipschitz constant of the GP standard deviation $\sigma(\cdot)$, which is guaranteed to exist due to Corollary 2.6. Choose $\tau \in \mathbb{R}_+$ such that

$$\tau \leq \frac{1}{L_\mu + L_f + L_\sigma}\sqrt{\frac{2}{\underline{h}_{\mathrm{ref}} k(\mathbf{0},\mathbf{0})}} \tag{4.52}$$

holds and $\delta \in (0,1)$ satsifying (2.74). Then, we have

$$\sup_{t\geq 0} \eta(\boldsymbol{x}_{\mathrm{ref}}(t)) \leq 2\sqrt{\frac{2}{\underline{h}_{\mathrm{ref}} k(\boldsymbol{x},\boldsymbol{x})}}\sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)} \tag{4.53}$$

due to Proposition 4.1, such that we obtain

$$\bar{\upsilon} = -\frac{2\sqrt{2}}{(\kappa + L_\sigma)\sqrt{\underline{h}_{\mathrm{ref}} k(\boldsymbol{x},\boldsymbol{x})}}. \tag{4.54}$$

Due to the choice of $\boldsymbol{\theta}$ resulting in a constant value of κ and the fact that L_σ can be globally bounded independent of the data set due to Corollary 2.6, we can solve (4.54) for $\underline{h}_{\mathrm{ref}}$ to determine the asymptotic behavior

$$\underline{h}_{\mathrm{ref}} \in \mathcal{O}\left(\frac{1}{\bar{\upsilon}^2}\right). \tag{4.55}$$

Since the required tracking accuracy in (4.45) can be guaranteed if $\bar{\upsilon} = \bar{e}$, this directly leads to (4.47) and thereby concludes the first part of the proof. It remains to determine the necessary asymptotic behavior of $-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))$. For this purpose, note that

$$\frac{1}{\tau} \in \mathcal{O}\left(\frac{N}{\bar{e}}\right) \tag{4.56}$$

due to (4.47) and $L_\mu \in \mathcal{O}(N)$ as shown the proof of Theorem 4.2. As κ is constant, (4.48) immediately follows from (4.49) and (4.12). □

While suitably small eigenvalues of the dynamics matrix $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ are sufficient to allow the application of Proposition 3.1, the necessary condition (3.13) depends on the Lipschitz constant L_η. This Lipschitz constant, in turn, grows with the data density $\underline{h}_{\mathrm{ref}}$ because the scaling factor of the posterior standard deviation $\sigma(\cdot)$ in Proposition 2.2 does so for τ defined in (4.52). Therefore, constant eigenvalues of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ are not sufficient, but they have

to become smaller with vanishing desired error bound $\bar{e}$. This requirement is not severe in practice since the eigenvalues of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ are often designed through a lower level controller as discussed after Assumption 3.1, such that arbitrarily small eigenvalues $\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))$ can be achieved. Moreover, if the number of training samples N grows at most polynomially with $\underline{h}$ as ensured, e.g., for the case of SE or Matérn kernels with uniformly distributed training data discussed in Section 4.1.4, this implies the requirement $-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) \in \mathcal{O}(\sqrt{\log(1/\bar{e})})$. Therefore, the necessary increase rate of $-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))$ is merely log-hyperbolic, which is a significant improvement compared to a reduction of the tracking error completely through the choice of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ and without any model. This is shown in the following lemma.

Lemma 4.3. *Consider a system* (3.2) *with Lipschitz continuous nonlinearity* $f(\cdot)$ *satisfying Assumption 3.1, to which a control law* (3.4) *with gains* $\boldsymbol{\theta}$ *satisfying Assumption 3.3 with* $\hat{f}(\cdot) = 0$ *is applied to track a bounded reference trajectory* $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$*. Then, for* $\mathbb{X}_0 = \{\boldsymbol{x}_{\mathrm{ref}}(0)\}$*, ensuring* (4.45) *asymptotically requires at most*

$$-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) \in \mathcal{O}\left(\frac{1}{\bar{e}}\right). \tag{4.57}$$

Proof. Due to Lipschitz continuity of $f(\cdot)$, it is upper bounded by a constant $\bar{f}$ on a suitable compact set $\mathbb{S}$. Therefore, it follows from (3.11) with $\bar{f}_e(\cdot) = \bar{f}$ that the tracking error is upper bounded by

$$\bar{v} = -\frac{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|\bar{f}}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))} \tag{4.58}$$

Setting $\bar{e} = \bar{v}$ and solving for $\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))$ leads to (4.57). □

Remark 4.1. *In order to streamline the presentation, we have used several simplifications in this subsection. Although we have restricted the analysis of the data dependency to the Bayesian uniform error bound, an equivalent result can be derived based on error bounds obtained from Theorem 2.1. Moreover, we have dropped the explicit dependencies on the dimension* d_x *of the state space and the probability* δ *of violating the tracking error bound since the dependency on the data density* $\underline{h}_{\mathrm{ref}}$ *is the focus of this section.*

4.2.3. Lyapunov-Based Quality Assessment

While the result in Section 4.2.2 relies on a closed-form expression for the tracking error bound, we can also analyze the data dependency of results such as Theorem 3.3. In order to demonstrate this, we follow the problem setting described in Section 3.2.1 with the particular example of a system

$$\dot{x}_1 = x_2, \qquad \cdots \qquad \dot{x}_{d_x-1} = x_{d_x}, \qquad \dot{x}_{d_x} = f(\boldsymbol{x}, u), \tag{3.53 revisited}$$

with dynamics defined by $f(\boldsymbol{x}, u) = \tilde{f}(\boldsymbol{x}) + g(\boldsymbol{x})u$, such that a feedback linearizing controller

$$\pi(\boldsymbol{x}, t) = \frac{1}{\mu_g(\boldsymbol{x})}\left(\pi_{\mathrm{lin}}(\boldsymbol{x}, t) - \mu_f(\boldsymbol{x})\right), \tag{3.56 revisited}$$

is employed for tracking a reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$. Therefore, we can employ Corollary 3.1, which allows us to obtain a probabilistic tracking error bound by solving (3.59).

Due to the structure of the optimization constraint (3.60), it admits a straightforward decoupling of model error sources. Therefore, it can be directly seen that

$$\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t)\|^2 \geq |(\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}|\eta_{\tilde{f}}(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in \mathbb{S}, t \in \mathbb{R}_{0,+}, \quad (4.59)$$

$$\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t)\|^2 \geq |(\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}|\frac{\eta_g(\boldsymbol{x})}{\mu_g(\boldsymbol{x})}\left|\pi_{\mathrm{lin}}(\boldsymbol{x},t)-\mu_{\tilde{f}}(\boldsymbol{x})\right| \qquad \forall \boldsymbol{x} \in \mathbb{S}, t \in \mathbb{R}_{0,+} \quad (4.60)$$

guarantees 0 to be the only solution of (3.59), which implies exact tracking.

Since (4.59) and (4.60) independently depend on the individual uniform error bounds $\eta_{\tilde{f}}(\cdot)$ and $\eta_g(\cdot)$, respectively, it also makes sense to analyze the data requirements separately. This can be achieved by adapting the data density measure $h(\cdot)$ to capture the behavior of the component standard deviations $\sigma_{\tilde{f}}(\cdot)$ and $\sigma_g(\cdot)$ instead of the overall GP standard deviation $\sigma(\cdot)$. Since the derivation of $h(\cdot)$ relies on $\Delta k(\cdot)$ defined in (4.21), we consider the analogous function

$$\Delta k_{\tilde{f}}(\boldsymbol{x}) = k_{\tilde{f}}(\boldsymbol{0})\left(k_{\tilde{f}}(\boldsymbol{0}) + k_g(\boldsymbol{0}) \max_{[\boldsymbol{x}^T,u]^T \in \mathbb{D}} u^2\right) - \min_{[(\boldsymbol{x}')^T,u']^T \in \mathbb{D}} k_{\tilde{f}}^2(\boldsymbol{x}-\boldsymbol{x}') \quad (4.61)$$

for the standard deviation $\sigma_{\tilde{f}}(\cdot)$ resulting from stationary kernels $k_{\tilde{f}}(\cdot,\cdot)$ and $k_g(\cdot,\cdot)$. Similarly as in (4.27), this allows us to define the sets

$$\mathbb{K}_h^{\tilde{f}}(\boldsymbol{x}) = \left\{\boldsymbol{z}' = \begin{bmatrix}(\boldsymbol{x}')^T & u'\end{bmatrix}^T \in \mathbb{D} : u' \leq \sqrt{\frac{1}{2hk_{\tilde{f}}(\boldsymbol{0})k_g(\boldsymbol{0})}}, k_{\tilde{f}}^2(\boldsymbol{0}) - k_{\tilde{f}}^2(\boldsymbol{x}-\boldsymbol{x}') \leq \frac{1}{2h}\right\}, \quad (4.62)$$

which include only training data points with $\Delta k_{\tilde{f}}(\boldsymbol{x}) \leq h$. Based on these sets, we can finally measure the data density using

$$h_{\tilde{f}}(\boldsymbol{x}) = \max_{h \in \mathbb{R}_+} h \quad (4.63)$$

$$\text{such that } |\mathbb{K}_h^{\tilde{f}}(\boldsymbol{x})| \geq h\sigma_{\mathrm{on}}^2 k_{\tilde{f}}(\boldsymbol{x},\boldsymbol{x}). \quad (4.64)$$

By construction, $h_{\tilde{f}}(\cdot)$ behaves analogously to $h(\cdot)$ and guarantees a posterior variance bound similar to Proposition 4.1. However, note that this requires an additional constraint on the control in (4.62) compared to (4.27). This is due to the intuitive property that only training samples with control inputs close to zero allow the precise identification of $\tilde{f}(\cdot)$, such that only training samples with $u = 0$ can be contained in $\mathbb{K}_h^{\tilde{f}}(\boldsymbol{x})$ for $h \to \infty$. Thereby, $h_{\tilde{f}}(\cdot)$ retains the beneficial property that it is reciprocal to the GP variance $\sigma^2(\cdot)$. This property is exploited in the following proposition to describe the requirements on the spatial distribution of training data for ensuring (4.59).

Proposition 4.2. *Assume that $k_{\tilde{f}}(\cdot,\cdot)$ and $k_g(\cdot,\cdot)$ defining $\eta_{\tilde{f}}(\cdot)$, $\eta_g(\cdot)$, $\mu_{\tilde{f}}(\cdot)$ and $\mu_g(\cdot)$ in* (4.59) *are stationary kernels. Then,* (4.59) *can be satisfied using training data with a density satisfying*

$$\sqrt{h_{\tilde{f}}(\boldsymbol{x})} \propto \sqrt{\underline{h}_{\tilde{f}}(\boldsymbol{x})} = \max_{t \in \mathbb{R}_{0,+}} \frac{|(\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}|}{\|\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t)\|^2} \qquad \forall \boldsymbol{x} \in \mathbb{S}. \quad (4.65)$$

Proof. It is straightforward to see that

$$\eta_{\tilde{f}}(\boldsymbol{x}) \leq \max_{t \in \mathbb{R}_{0,+}} \frac{\|\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t)\|^2}{|(\boldsymbol{x}-\boldsymbol{x}_{\mathrm{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}|} \tag{4.66}$$

ensures the satisfaction of (4.59). The uniform error bound $\eta_{\tilde{f}}(\cdot)$ following from a combination of Corollary 2.4 and Corollary 2.5 is proportional to the GP standard deviation $\sigma_{\tilde{f}}(\boldsymbol{z})$, i.e., $\eta_{\tilde{f}}(\boldsymbol{z}) \propto \sigma_{\tilde{f}}(\boldsymbol{z})$. By extending Lemma 4.1, the GP standard deviation $\sigma_{\tilde{f}}(\boldsymbol{z})$ can be bounded by

$$\sigma_{\tilde{f}}^2(\boldsymbol{x}) \leq \frac{\sigma_{\mathrm{on}}^2}{|\mathbb{K}_{h_{\tilde{f}}(\boldsymbol{x})}(\boldsymbol{x})|} + \frac{\Delta k_{\tilde{f}}(\boldsymbol{x})}{\max\limits_{\boldsymbol{z}' \in \mathbb{K}^{\tilde{f}}_{h_{\tilde{f}}(\boldsymbol{x})}(\boldsymbol{x})} k(\boldsymbol{z},\boldsymbol{z})} \tag{4.67}$$

Since kernels are positive definite, we simplify the bound to

$$\sigma_{\tilde{f}}^2(\boldsymbol{x}) \leq \frac{\sigma_{\mathrm{on}}^2}{|\mathbb{K}_{h_{\tilde{f}}(\boldsymbol{x})}(\boldsymbol{x})|} + \frac{\Delta k_{\tilde{f}}(\boldsymbol{x})}{k_{\tilde{f}}(\boldsymbol{x},\boldsymbol{x})}, \tag{4.68}$$

such that the definition of $h_{\tilde{f}}(\boldsymbol{x})$ yields

$$\sigma_{\tilde{f}}^2(\boldsymbol{x}) \leq \frac{2}{h_{\tilde{f}}(\boldsymbol{x}) k_{\tilde{f}}(\boldsymbol{0})}. \tag{4.69}$$

Therefore, training data with a density satisfying (4.65) can ensure the existence of an error bound $\eta_{\tilde{f}}(\boldsymbol{x})$ satisfying (4.66), which concludes the proof. □

This proposition shows that it is not necessary to obtain uniformly distributed training data for reducing the uncertainty about $\tilde{f}(\cdot)$ in order to satisfy (4.59) and consequently to ensure a high tracking accuracy. Due to the quadratic growth of the nominal Lyapunov derivative with the tracking error reflected by the left side of (4.59), the uniform GP error bound $\eta_{\tilde{f}}(\cdot)$ can increase linearly with the distance from the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$. This increase directly admits the quadratically decaying data requirement ensured by Proposition 4.2. Note that this is a very intuitive requirement since the linear tracking controller (3.57) becomes dominant in the feedback linearizing control law (3.56) for large tracking errors in comparison to the model $\mu_{\tilde{f}}(\cdot)$. Therefore, the GP model $\mu_{\tilde{f}}(\cdot)$ can be more imprecise and consequently does not need as many training samples far away from the reference trajectory.

For the standard deviation $\sigma_g(\cdot)$, we can proceed analogously and consider

$$\Delta k_g(\boldsymbol{x}) = k_g(\boldsymbol{0}) \left(k_{\tilde{f}}(\boldsymbol{0}) + k_g(\boldsymbol{0}) \max_{[\boldsymbol{x}^T,u]^T \in \mathbb{D}} u^2 \right) - \min_{[(\boldsymbol{x}')^T,u']^T \in \mathbb{D}} k_g^2(\boldsymbol{x}-\boldsymbol{x}')(u')^2. \tag{4.70}$$

Constructing a set that ensures $\Delta k_g(\boldsymbol{x}) \leq h$ for all contained samples is more complicated since control inputs appear in maximization and minimization queries. Therefore, we define $\bar{u} = \max_{[\boldsymbol{x}^T,u]^T \in \mathbb{D}} |u|$, such that we can replace the minimization by a constraint on the minimum values of the control inputs parameterized as $\underline{u} = {}^{\bar{u}\alpha(h)}/_{1+\alpha(h)}$, where $\alpha : \mathbb{R}_{0,+} \to$

$\mathbb{R}_{0,+}$ can be an arbitrary class $\mathcal{K}_\infty$ function. These definitions lead to the set

$$\mathbb{K}^g_{h,\bar{u}}(\boldsymbol{x}) = \Bigg\{ \boldsymbol{z}' = \begin{bmatrix} (\boldsymbol{x}')^T & u' \end{bmatrix}^T \in \mathbb{D} \colon \tag{4.71}$$

$$c\bar{u} \leq |u'|, k_g^2(\mathbf{0}) - c^2 k_g^2(\boldsymbol{x} - \boldsymbol{x}') \leq \frac{\frac{1}{h} - k_{\tilde{f}}(\mathbf{0}) k_g(\mathbf{0})}{\bar{u}^2}, c = \frac{\alpha(h)}{\alpha(h)+1} \Bigg\}.$$

Based on this set, we can define a measure of data density via the optimization problem

$$h_g(\boldsymbol{x}) = \max_{h \in \mathbb{R}_+} h \tag{4.72}$$

$$\text{such that } |\mathbb{K}^{g,\bar{u}}_h(\boldsymbol{x})| \geq h \sigma^2_{\text{on}} k_g(\boldsymbol{x}, \boldsymbol{x}) \bar{u}^2. \tag{4.73}$$

This function is again constructed in a way to ensure a bound for the GP variance $\sigma_g^2(\cdot)$ similar to Proposition 4.1. While the training samples with large control input amplitude are excluded for $h_{\tilde{f}}(\cdot)$, only those are considered for $h_g(\cdot)$. Therefore, the data density measure $h_g(\cdot)$ reflects the intuitive behavior that only large control input amplitudes allow the accurate identification of $g(\cdot)$. Note, however, that $h_g(\cdot)$ alone is not capable of capturing the full data dependency because even for $\bar{u} \to \infty$, $h_g(\boldsymbol{x})$ is upper bounded by $1/k_{\tilde{f}}(\mathbf{0}) k_g(\mathbf{0})$. Due to the consideration of $\bar{u}^2$ in (4.73), it is still possible to straightforwardly show a vanishing posterior variance following the ideas of Proposition 4.1. Therefore, this limitation is not a critical issue, such that we can employ $h_g(\cdot)$ to describe conditions on the spatial distribution of training data for ensuring (4.60).

Proposition 4.3. *Assume that $k_{\tilde{f}}(\cdot,\cdot)$ and $k_g(\cdot,\cdot)$ defining $\eta_{\tilde{f}}(\cdot)$, $\eta_g(\cdot)$, $\mu_{\tilde{f}}(\cdot)$ and $\mu_g(\cdot)$ in* (4.60) *are stationary kernels. If $g(\cdot)$ is lower bounded by $\underline{g} \in \mathbb{R}_+$ and $|f(\cdot)|$ is upper bounded by $\bar{f} \in \mathbb{R}_+$, then,* (4.60) *can be satisfied using training data with a density satisfying*

$$\sqrt{h_g(\boldsymbol{x})} \propto \sqrt{\underline{h}_g(\boldsymbol{x})} = \frac{\|(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}\| \left(|\pi_{\text{lin}}(\boldsymbol{x}, t)| + \bar{f} + \sqrt{\frac{1}{h_{\tilde{f}}(\boldsymbol{x})}} \right) + \|\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t)\|^2}{\|\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t)\|^2}. \tag{4.74}$$

Proof. Since we are interested in the spatial distribution of the data, assume that we have sufficiently many samples to ensure $\eta_g(\boldsymbol{x}) \leq \underline{g}$. Then, the GP mean $\mu_g(\boldsymbol{x})$ is lower bounded by $\underline{g} - \eta_g(\boldsymbol{x})$, which implies that (4.59) is satisfied if

$$\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t)\|^2 \geq |(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}| \left(|\pi_{\text{lin}}(\boldsymbol{x}, t)| + \bar{f} + \eta_{\tilde{f}}(\boldsymbol{x}) \right) \frac{\eta_g(\boldsymbol{x})}{\underline{g} - \eta_g(\boldsymbol{x})}. \tag{4.75}$$

By solving this inequality for $\eta_g(\boldsymbol{x})$, we obtain the condition

$$\eta_g(\boldsymbol{x}) \leq \frac{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t)\|^2 \underline{g}}{|(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t))^T \boldsymbol{P}\boldsymbol{b}| \left(|\pi_{\text{lin}}(\boldsymbol{x}, t)| + \bar{f} + \eta_{\tilde{f}}(\boldsymbol{x}) \right) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t)\|^2}, \tag{4.76}$$

such that we can proceed analogously to the proof of Proposition 4.2. Therefore, we have

$$\sigma_g^2(\boldsymbol{x}) \leq \frac{2}{h_g(\boldsymbol{x}) k_g(\mathbf{0}) \bar{u}^2} \tag{4.77}$$

due to the definition of $h_g(\boldsymbol{x})$, such that training data with a density satisfying (4.74) can ensure the existence of an error bound $\eta_g(\boldsymbol{x})$ satisfying (4.76), which concludes the proof. □

While the data density requirement $\underline{h}_{\tilde{f}}(\cdot)$ quickly decreases away from the reference trajectory, it remains almost constant for the identification of $g(\cdot)$. Ignoring $\bar{f}$ and $\sqrt{1/h_{\tilde{f}}(\boldsymbol{x})}$, this directly follows from the linear increase of the tracking controller $\pi_{\text{lin}}(\boldsymbol{x}, t)$ with the distance to the reference trajectory. Therefore, an increase of the required data density $\underline{h}_g(\cdot)$ only occurs close to states $\boldsymbol{x}_{\text{ref}}(t)$, when the constant term $\bar{f}$ becomes dominant. This is an expectable behavior of the necessary data density since the ratio between the GP mean $\mu_g(\cdot)$ and the uniform error bound $\eta_g(\cdot)$ can be interpreted as a state-dependent reduction of the control gain k_c. Bounding the mean using a constant, this immediately leads to the requirement of a flat data distribution, which underlines the high interpretability of the data density specification $\underline{h}_g(\cdot)$.

Due to the intuitiveness of the data density measures, we employ them to analyze the suitability of training data distributions for ensuring low tracking errors. For this purpose, we can simply compare the required densities $\underline{h}_{\tilde{f}}(\cdot)$ and $\underline{h}_g(\cdot)$ with the corresponding actual densities $h_{\tilde{f}}(\cdot)$ and $h_g(\cdot)$. Since the requirements are formulated only up to constant factors, this can be effectively realized using the log-ratios

$$\rho_{\tilde{f}}(\boldsymbol{x}) = \log\left(\frac{\underline{h}_{\tilde{f}}(\boldsymbol{x})}{h_{\tilde{f}}(\boldsymbol{x})}\right), \tag{4.78}$$

$$\rho_g(\boldsymbol{x}) = \log\left(\frac{\underline{h}_g(\boldsymbol{x})}{h_g(\boldsymbol{x})}\right), \tag{4.79}$$

such that proportional factors merely appear as constant offsets. Therefore, the density log-ratios $\rho_{\tilde{f}}(\cdot)$ and $\rho_g(\cdot)$ allow a straightforward analysis of data distributions, e.g., $\rho_g(\boldsymbol{x}) > \rho_g(\boldsymbol{x}')$ for states $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{X}$ implies that more training samples with large control input amplitude should be added around $\boldsymbol{x}$ to come closer to a good distribution. This information can be exploited when designing active learning methods and closed-loop data generation approaches to reduce the requirements on the number N of training samples necessary to achieve desired tracking error bounds $\bar{e}$.

4.2.4. Numerical Evaluation

In order to illustrate the role of training data for tracking accuracy guarantees, we investigate two examples in this section. First, we demonstrate the asymptotic error decay for a linear system with unknown input perturbation before we investigate the Lyapunov-based requirements for suitable training distributions.

Asymptotic Error Decay

In order to illustrate the dependency of the tracking error bound $\upsilon(\cdot)$ on the data density $\underline{h}_{\text{ref}}$ derived in Theorem 4.3, we consider the same setting as in Section 3.1.4, i.e., a linear system with

$$\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta}) = \begin{bmatrix} 0 & 1 \\ -k_c\tilde{\theta} & -k_c - \tilde{\theta} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.80}$$

perturbed by a nonlinear function $f(\boldsymbol{x}) = 1 - \sin(2x_1) + 1/(1+\exp(-x_2))$. Since we are interested in the dependency on the data density, we use grids with different grid constants defined

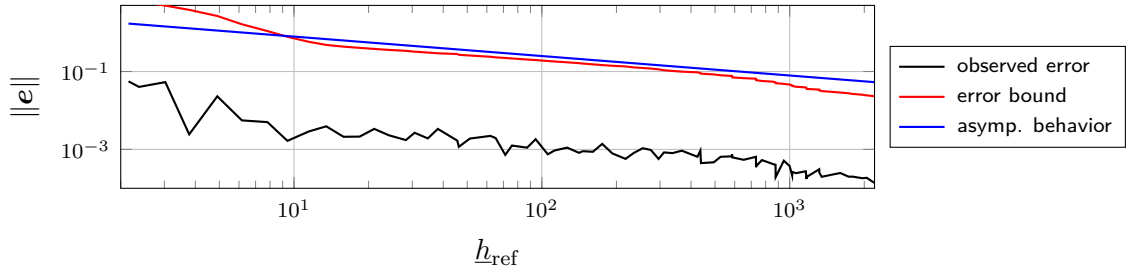

Figure 4.2.: Comparison of the observed tracking error, the tracking error bound and its guaranteed asymptotic decay rate for growing data densities.

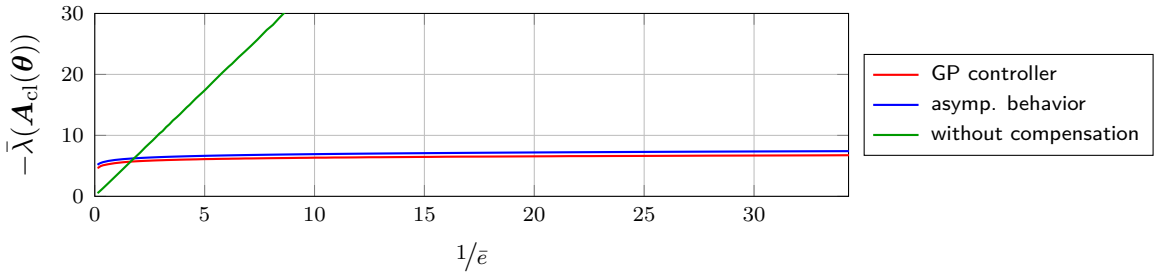

Figure 4.3.: Maximum eigenvalue $\bar{\lambda}(\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta}))$ necessary to ensure a given tracking error bound $\bar{e}$ when learning a control law using equidistant grids in comparison to a pure feedback controller without compensation of nonlinearities.

on $[-4, 4]^2$ as training data sets, such that they cover the whole relevant domain. Due to the varying size of the training data set, we determine τ by finding the maximum value satisfying (4.52) using a line search. We set $k_c = \theta$, such that we can compute a gain k_c ensuring $\kappa = 10$ in (4.49) for the obtained value of τ.

The resulting tracking errors $\|\boldsymbol{e}\|$ and bounds $\bar{v}$ obtained with Theorem 3.1 for different data densities $\underline{h}_{\text{ref}}$ are illustrated in Fig. 4.2. Moreover, the asymptotic decay rate of $\bar{v}$ following from Theorem 4.3 is depicted. It can be clearly seen that the asymptotic decay rate closely reflects the actual decay rate of the error bound $\bar{v}$. Analogously to Section 3.1.4, the tracking error bound is rather conservative, but the observed error $\|\boldsymbol{e}\|$ exhibits a decay rate with high similarity to its bound $\bar{v}$. Despite this conservatism, the necessary maximum eigenvalues $\bar{\lambda}(\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta}))$ for ensuring a low desired tracking error bound $\bar{e}$ with such training data are significantly larger than without a controller compensating the nonlinearity as depicted in Fig. 4.3. Due to the linear growth required by Lemma 4.3, the necessary eigenvalues $-\bar{\lambda}(\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta}))$ without a GP model quickly exceed the requirements for the learning-based approach. Hence, $\underline{h}_{\text{ref}}$ effectively captures the asymptotic data dependency for nonlinearity compensation, such that the advantages of the learning-based approach can be demonstrated.

Dependency of Tracking Accuracy Guarantees on the Data Distribution

For illustrating the benefits of a spatial analysis of the data distribution as proposed in Section 4.2.3, we investigate the example in Section 3.2.4 again. Therefore, we consider a nonlinear dynamical system

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = \tilde{f}(\boldsymbol{x}) + g(\boldsymbol{x})u, \qquad \text{(3.83 revisited)}$$

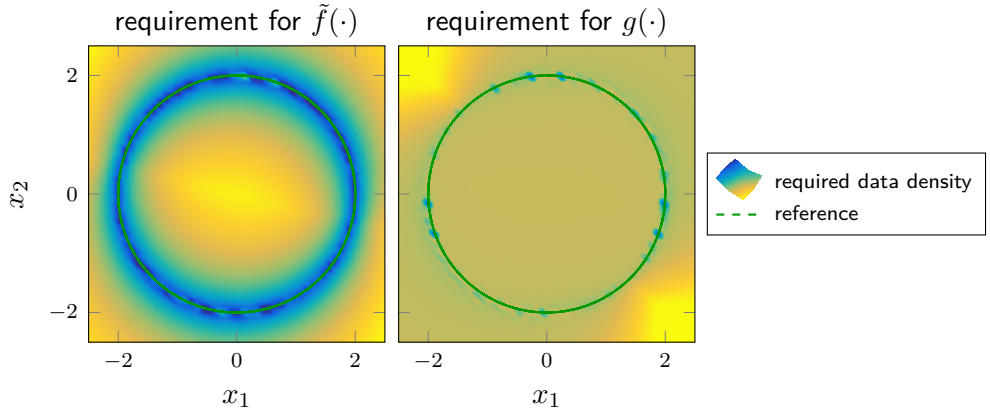

Figure 4.4.: Required data densities $\underline{h}_{\tilde{f}}(\cdot)$ and $\underline{h}_g(\cdot)$ for systems of the form (3.83) controlled via GP-based feedback linearization.

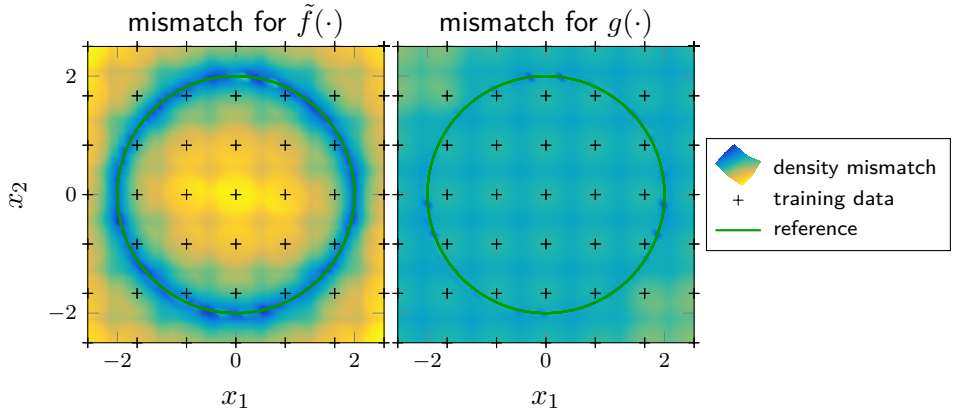

Figure 4.5.: Training data density mismatch measured by the density log-ratios $\rho_{\tilde{f}}(\cdot)$ and $\rho_g(\cdot)$ for systems of the form (3.83) controlled via feedback linearization based on a GP model trained on grid data.

where $\tilde{f}(\boldsymbol{x}) = 1 - \sin(2x_1) + {}^1\!/_{(1+\exp(-x_2))}$ and $g(\boldsymbol{x}) = 20\,(1 + {}^1\!/_2 \sin\,({}^{x_2}\!/_4))$.

Based on the prior GP used in Section 3.2.4, we can directly compute the required data densities $\underline{h}_{\tilde{f}}(\cdot)$ and $\underline{h}_g(\cdot)$ using Proposition 4.2 and Proposition 4.3, respectively. The result, which is depicted in Fig. 4.4, exhibits exactly the behavior discussed in Section 4.2.3. The necessary data density for inferring a sufficiently accurate model of $\tilde{f}(\cdot)$ increases strongly towards the reference trajectory, while it is almost constant for learning a suitable model for $g(\cdot)$. Merely very close to the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$, a significant increase of $\underline{h}_g(\cdot)$ can be observed. Finally, we can see a small directional dependency of the required data densities $\underline{h}_{\tilde{f}}(\cdot)$ and $\underline{h}_g(\cdot)$, which is caused by a large difference of the eigenvalues of $\boldsymbol{P}$.

When using the training data employed in Section 3.2.4, which is basically a grid over states and control inputs, it clearly does not satisfy these density requirements. This can be directly seen when computing the density log-ratios $\rho_{\tilde{f}}(\cdot)$ and $\rho_g(\cdot)$ as illustrated in Fig. 4.5. While we can observe slight declines in the density between training samples, the grid generally achieves almost constant densities $h_{\tilde{f}}(\cdot)$ and $h_g(\cdot)$. Thereby, it satisfies the requirements of a good data set for learning $g(\cdot)$, while it does not exhibit the increased density $h_{\tilde{f}}(\cdot)$ around the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$, which is required for achieving a sufficiently accurate

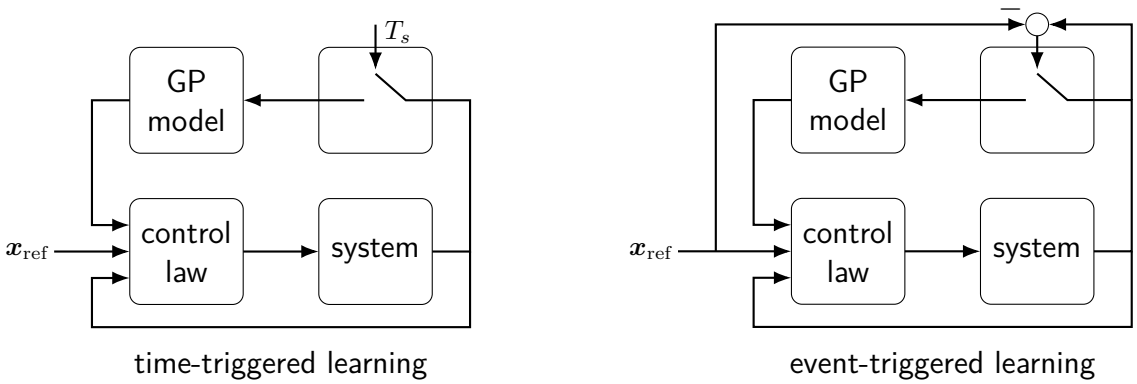

Figure 4.6.: Illustration of time-triggered and event-triggered learning with Gaussian process models: in time-triggered learning, data is sampled periodically with a sampling time T_s, while in event-triggered learning, training data is generated when the tracking error exceeds a threshold.

model of $\tilde{f}(\cdot)$. When generating additional training data for the GP model, we can take this into account, e.g., by generating 49 equally spaced samples with control input $u = 0$ along the reference trajectory $\boldsymbol{x}_{\text{ref}}(\cdot)$. These additional samples result in a reduction of the probabilistic tracking error bound $\bar{v}$ determined using Corollary 3.1 from 0.2740 to 0.1977. If we simply augmented the training data set with more grid samples with control input 0, we would merely achieve a reduction to 0.2107. Therefore, the insights gained from the density log-ratios $\rho_{\tilde{f}}(\cdot)$ and $\rho_g(\cdot)$ cause an improvement of 20%, which demonstrates the practical advantages of choosing training data-dependent on the control task.

Remark 4.2. *Due to the dependency of $h_{\tilde{f}}(\cdot)$ and $h_g(\cdot)$ on the control inputs of training samples, the almost constant densities crucially rely on the fact that the training data grid contains samples with high and small control input for each training state. Therefore, a uniform grid over the state space does not directly imply approximately constant data densities $h_{\tilde{f}}(\cdot)$ and $h_g(\cdot)$ in general.*

4.3. Closed-Loop Data Generation for Tracking Accuracy Guarantees

While we investigate the dependency of tracking error bounds on the data distribution in Section 4.2, the problem of actually obtaining suitable training data is not addressed. Since the system dynamics constrain the generation process, a straightforward approach is the sampling of closed-loop data during the regular operation of the system. If this data can be processed in real-time, it can be directly employed for online updates of GP models, such that an adaptive control behavior can be realized [33]. This behavior is commonly achieved in a time-triggered fashion [142, 143], where training samples are measured at regularly spaced time instances as illustrated on the left side of Fig. 4.6. Despite the simplicity of this approach, the interrelation between control and learning has not been analyzed for this approach to the best of our knowledge, such that merely tracking error bounds depending on the a priori unknown GP standard deviation can be guaranteed [144]. This weakness

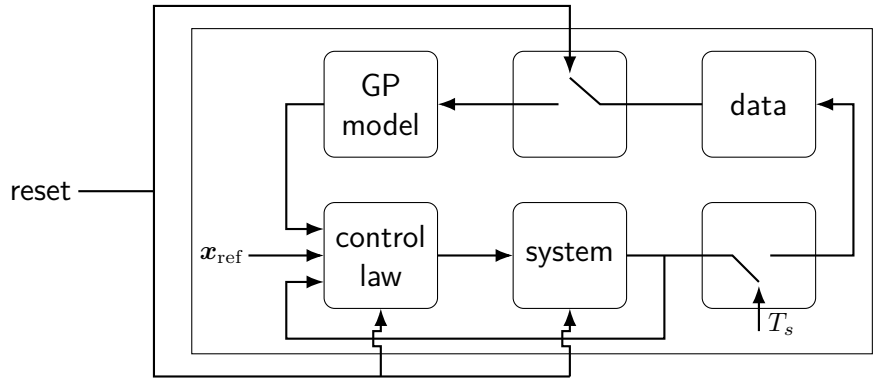

Figure 4.7.: Illustration of episodic learning with GP models: Data is collected and stored while the control law is executed. A reset signal is regularly sent, starts the updating of the GP model, and restarts the system by bringing it back to the initial state.

can be overcome using event-triggered learning [145], which transfers concepts commonly employed in networked control methods [146] to the online learning problem. The key idea behind event-triggered learning is to update the model only when needed as indicated, e.g., by the tracking error, which is illustrated on the right side of Fig. 4.6. By exploiting uniform error bounds and GP variance bounds, this strategy enables the straightforward derivation of probabilistic tracking accuracy guarantees for feedback linearizing [31] and backstepping control laws [147]. However, the triggering conditions in these approaches are highly specific for the considered controllers, such that a generalization of these results is not obvious.

When the data cannot be processed online, a model needs to be learned episodically through an iteration between applying the control law for generating data and inferring a GP model offline as illustrated in Fig. 4.7. This approach has received a high attention in the context of optimization-based controller tuning [96, 148], which can be shown to provide data-dependent performance guarantees due to the close relationship to Bayesian optimization [67, 126]. While these guarantees can be extended to model-based reinforcement learning [86, 149], they strongly rely on the solved optimization problems, such that they do not generalize to a broader class of control techniques. Therefore, these results are not applicable to model-based control laws, such that the episodic learning of GP models for model-based control remains an open problem.

We address this lack of generalizable data generation approaches for learning GP models by proposing three different strategies which provide tracking error guarantees:

1. We derive error bounds for time-triggered online learning control, which relate the tracking accuracy to the sampling time. To the best of our knowledge, this is the first result for time-triggered learning with GPs.

2. We present a straightforward triggering condition, which we prove to guarantee bounded tracking error and a bounded time difference between two GP model updates.

3. We develop an episodic approach for learning GP models, which ensures an arbitrarily high desired tracking accuracy within a finite number of episodes.

Even though these results are exemplarily derived for linear systems with unknown input perturbation, they straightforwardly generalize to Lyapunov-based tracking error bounds.

Therefore, we present a flexible framework for GP-based online and offline learning with performance guarantees.

The remainder of this section is structured as follows. In Section 4.3.1, the formal problem setting is stated. The time-triggered online learning strategy for inferring GP models for control is analyzed in Section 4.3.2. In Section 4.3.3, tracking error bounds for the event-triggered online learning method are shown. The episodic approach for offline inference of GP modelsis presented in Section 4.3.4. Finally, the different closed-loop data generation strategies are illustrated in numerical simulations in Section 4.3.5.

4.3.1. Problem Setting

In order to develop closed-loop data generation approaches for achieving an arbitrary tracking accuracy, we consider again the setting in Section 3.1. This means that the system dynamics are given by

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}(u + f(\boldsymbol{x})), \tag{3.2 revisited}$$

where $f : \mathbb{X} \to \mathbb{R}$ is an unknown, scalar perturbation of the linear system. Moreover, a control law of the form

$$u(t) = -\boldsymbol{\theta}^T(\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{ref}}(t)) + r_{\mathrm{ref}}(t) - \hat{f}(\boldsymbol{x}(t)). \tag{3.4 revisited}$$

is employed for compensating the unknown nonlinearity, such that the closed-loop error dynamics

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})\boldsymbol{e} + \boldsymbol{b}(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})) \tag{3.5 revisited}$$

are obtained, where $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \boldsymbol{A} - \boldsymbol{b}\boldsymbol{\theta}^T$. While a given data set $\mathbb{D}$ is implicitly assumed for inferring a GP model $\mu(\cdot)$ in the previous sections, this set must usually be generated by taking measurements of the system (3.2) during closed-loop control in practice. Therefore, we need to jointly address the problem of sampling data of the form $(\boldsymbol{x}(t), y = f(\boldsymbol{x}(t)) + \epsilon)$ for learning a model $\mu(\cdot)$ and tracking a reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$ with the state $\boldsymbol{x}(t)$ by compensating the nonlinearity $f(\cdot)$ using a control law (3.4) with $\hat{f}(\cdot) = \mu(\cdot)$. This problem manifests in the following two forms:

- Online learning problem: If we gather and process the data online, we obtain a time-variant data set $\mathbb{D}(t)$ and thereby a time-variant control law with model $\hat{f}(\cdot) = \mu(\cdot)$. The data can be collected at fixed times or arbitrary sampling instances. We consider the special case that the model update does not take any time, such that the improved model is available immediately after the training data has been sampled.

- Episodic learning problem: When the model can only be updated offline, we need to iterate between applying the control law (3.4) with a time-invariant GP model $\hat{f}(\cdot) = \mu(\cdot)$ to the system and updating the GP model $\mu(\cdot)$ offline using measurements of the so-called roll-out trajectory $\boldsymbol{x}(\cdot)$. For simplicity, we consider the special case of the same reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$ and execution time T during all roll-outs.

In this section, we address these problem manifestations by developing algorithms for guaranteeing a desired, possibly arbitrarily tracking error bound $\bar{e}$.

Algorithm 4.1. Time-triggered online learning for GP-based control

1: Initialize GP model with $N = \lceil h\sigma_{\text{on}}^2 k(\boldsymbol{x}_{\text{ref}}(0), \boldsymbol{x}_{\text{ref}}(0))\rceil$ training samples at $\boldsymbol{x}_{\text{ref}}(0)$
2: **while** $t < T$ **do**
3: Apply control input defined by GP-based control law (3.4)
4: **if** $t/T_s \in \mathbb{N}$ **then**
5: Take measurement $\boldsymbol{x}^{(N+1)}$, $y^{(N+1)} = f(\boldsymbol{x}^{(N+1)}) + \epsilon^{(N+1)}$, $N \leftarrow N+1$
6: Update GP mean $\mu(\cdot)$ in the control law (3.4)
7: **end if**
8: **end while**

Remark 4.3. *While we restrict our analysis to the case of a linear system with input perturbation, the proposed algorithms can be straightforwardly extended to more general nonlinear systems using the Lyapunov-based approach for certainty equivalent controllers presented in Section 3.2. In fact, all theoretical guarantees derived for the tracking error $\boldsymbol{e}(\cdot)$ can be equivalently derived for Lyapunov candidates $V(\cdot)$. Therefore, the restriction to systems of the form* (3.2) *is merely employed to simplify the presentation.*

4.3.2. Time-Triggered Learning

We first consider the online learning scenario in which training data can be processed online, but we restrict the time of the data measurements to fixed time instances nT_s, where $T_s \in \mathbb{R}_+$ denotes the sampling time. Therefore, we generate a time-variant data set

$$\mathbb{D}(t) = \left\{\boldsymbol{x}^{(n)} = \boldsymbol{x}(nT_s), y^{(n)} = f(\boldsymbol{x}^{(n)}) + \epsilon^{(n)}\right\}_{n=1}^{\left\lfloor \frac{t}{T_s} \right\rfloor}, \tag{4.81}$$

such that the GP model is updated using a new training pair at times $t/T_s \in \mathbb{N}$. The resulting online learning procedure, which we refer to as time-triggered learning, is summarized in Algorithm 4.1.

Due to the definition of the training inputs $\boldsymbol{x}^{(n)}$ using the states $\boldsymbol{x}(nT_s)$, the distance between them can be analyzed via properties of the trajectory $\boldsymbol{x}(\cdot)$. For linear control systems (3.2), it directly follows from the following lemma that this distance depends linearly on the sampling time T_s.

Lemma 4.4. *Consider a system* (3.2) *satisfying Assumption 3.1, to which a control law* (3.4) *is applied to track a reference trajectory with bounded derivative $\dot{\boldsymbol{x}}_{\text{ref}}(\cdot)$. Assume that a Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ is used to learn a model $\hat{f}(\cdot) = \mu(\cdot)$, such that Assumption 3.2 holds on a compact set $\mathbb{S} \subset \mathbb{X}$ with a uniform error bound $\eta(\cdot)$. Then, for all $\boldsymbol{x} \in \mathbb{S}$, the state derivative $\dot{\boldsymbol{x}}(t)$ is bounded by*

$$\|\dot{\boldsymbol{x}}(t)\| \leq F = \|\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta})\| \sup_{\boldsymbol{x}\in\mathbb{S}, t\in\mathbb{R}_{0,+}} \|\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}(t)\| + \max_{\boldsymbol{x}\in\mathbb{S}} \eta_0(\boldsymbol{x}) + \sup_{t\in\mathbb{R}_{0,+}} \|\dot{\boldsymbol{x}}_{\text{ref}}(t)\| \tag{4.82}$$

with probability $1-\delta$, where $\eta_0(\cdot)$ denotes the error bound without any training samples.

Proof. It is straightforward to see that the state dynamics of the closed-loop system are given by

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_{\text{ref}} + \boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta})\boldsymbol{e} + \boldsymbol{b}(f(\boldsymbol{x}) - \mu(\boldsymbol{x})). \tag{4.83}$$

We can choose the parameters of the uniform error bound, such that it holds for the GP model $\mu(\cdot)$ used in the control law (3.4), and apply the same parameters for computing $\eta_0(\cdot)$. Since the GP standard deviation $\sigma(\cdot)$ is non-increasing with the number of training samples N, this implies that $\eta_0(\cdot)$ is also a uniform error bound for the model $\mu(\cdot)$. Then, (4.82) immediately follows from (4.83), a bounded derivative of the reference trajectory and the restriction to a compact set $\mathbb{S}$. □

Because of the restriction to a compact set $\mathbb{S}$, the boundedness of $\dot{\boldsymbol{x}}(\cdot)$ is a direct consequence of the boundedness of $\dot{\boldsymbol{x}}_{\mathrm{ref}}(\cdot)$. In fact, we do not necessarily need to determine the supremum over $\mathbb{S}$ and $\mathbb{R}_{0,+}$ in (4.82). To see this, note that we only need to consider the maximum difference between $\boldsymbol{x}(t)$ and $\boldsymbol{x}_{\mathrm{ref}}(t)$. Therefore, we can obtain the tighter bound

$$\|\dot{\boldsymbol{x}}(t)\| \leq F = \|\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})\|\bar{\upsilon} + \max_{\boldsymbol{x}\in\mathbb{S}} \eta_0(\boldsymbol{x}) + \sup_{t\in\mathbb{R}_{0,+}} \|\dot{\boldsymbol{x}}_{\mathrm{ref}}(t)\| \tag{4.84}$$

if we have a probabilistic tracking error bound $\upsilon(\cdot) = \bar{\upsilon}$. Furthermore, the function $\eta_0(\cdot)$ is constant for stationary kernels, which allows the easy computation of (4.83).

Based on F, it is straightforward to see that $\|\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n+1)}\| \leq FT_s$ holds. Therefore, we can upper bound the uniform error bound $\eta(\boldsymbol{x}(nT_s))$ after collecting n training samples by

$$\eta_{T_s} = \inf_{N\in\mathbb{N}} \max_{\mathbb{D}^x\subset\mathbb{X},\boldsymbol{x}\in\mathbb{S}} \eta_{\mathbb{D}^x}(\boldsymbol{x}) \tag{4.85}$$

$$\text{such that } \|\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n+1)}\| \leq FT_s, \forall n = 1\ldots, N-1 \tag{4.86}$$

$$\mathbb{D}^x = \{\boldsymbol{x}^{(1)} = \boldsymbol{x}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}. \tag{4.87}$$

By fixing the number N of past sampling times, the error bounds defined in Theorem 2.1 and Proposition 2.2 only depend on the GP standard deviation $\sigma(\cdot)$. Then, no optimization is necessary for stationary kernels since we can simply determine the direction of the fastest decay of the kernel, e.g., the smallest length scale l_i for ARD SE and Matérn class kernels, and align the samples $\boldsymbol{x}^{(n)}$ in this direction with a distance FT_s. The GP standard deviation $\sigma(\boldsymbol{x})$ obtained for these training samples defines η_{T_s}. When the uniform error bound $\eta(\cdot)$ is Lipschitz continuous, the value η_{T_s} immediately implies a probabilistically bounded tracking error using a time-triggered online learning control law (3.4) as shown in the following theorem.

Theorem 4.4. *Consider a system* (3.2) *satisfying Assumption 3.1, to which a control law* (3.4) *is applied to track a reference trajectory with bounded derivative* $\dot{\boldsymbol{x}}_{\mathrm{ref}}(\cdot)$. *Assume that a Gaussian process* $\mathcal{GP}(0, k(\cdot,\cdot))$ *with stationary kernel* $k(\cdot,\cdot)$ *is employed to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *online using Algorithm 4.1, such that Assumption 3.2 holds on a compact set* $\mathbb{S} \subset \mathbb{X}$ *with a uniform error bound* $\eta(\cdot)$ *which admits a Lipschitz constant* L_η. *If* $\mathbb{B}_{\bar{\upsilon}}(\boldsymbol{x}_{\mathrm{ref}}(t)) \in \mathbb{S}$ *holds for all* $t \in [0, T]$, $T \in \mathbb{R}_{0,+}$, *where*

$$\bar{\upsilon} = -\frac{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}\left(L_\eta F T_s + \eta_{T_s}\right), \tag{4.88}$$

then, Algorithm 4.1 ensures a probabilistic tracking error bound $\upsilon(\cdot) = \bar{\upsilon}$ *during the time interval* $[0, T]$ *for initial states* $\mathbb{X}_0 = \{\boldsymbol{x}_{\mathrm{ref}}(0)\}$.

Proof. Analogous to (3.10), we can bound the tracking error by

$$\|\boldsymbol{e}(t)\| \leq \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\| \int_0^t \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))(t-t')}\eta(\boldsymbol{x}(t'))\mathrm{d}t'. \tag{4.89}$$

Since Lemma 4.4 ensures a bounded state derivative, we can linearize the uniform error bound around a training sample, such that we obtain

$$\eta(\boldsymbol{x}(t)) \leq \eta(\boldsymbol{x}(0)) + L_\eta F T_s. \tag{4.90}$$

Due to the definition of η_{T_s} and the initialization of the GP model in Algorithm 4.1, it follows that

$$\|\boldsymbol{e}(t)\| \leq \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\| \int_0^t \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))(t-t')} \left(L_\eta F T_s + \eta_{T_s}\right) \mathrm{d}t'. \tag{4.91}$$

Computing this integral yields (4.88), which concludes the proof. □

Since the uniform error bound $\eta(\cdot)$ only holds for a constant data set $\mathbb{D}$ with probability $1-\delta$, we have to combine multiple of them to deal with a time-variant data set $\mathbb{D}(t)$. Similarly to previous works [67], we do this using the union bound, such that we need to sum up the probabilities δ. However, this implies that we can only consider a finite number of model updates N and consequently a finite time interval $[0, T]$ using a constant probability $\delta = \tilde{\delta}/N$ during each time interval $[(n-1)T_s, T_s]$, $n = 1, \ldots, N$. Since this restriction admits arbitrarily large final times T, it is not severe in practice. Moreover, it can be circumvented in principle to allow an infinite number of updates N [67], but this leads to a growing uniform error bound η_{T_s} over time and thereby to an increase of (4.88).

For fixed time intervals $[0, T]$, the probabilistic tracking error bound (4.88) exhibits an intuitive behavior. On the one hand, a decrease of the sampling time T_s reduces the possible increase of the uniform error bound during two sampling instances. On the other hand, it directly diminishes the uniform error bound η_{T_s} by ensuring a higher data density. In fact, it immediately follows from a combination of Proposition 4.1 with Theorem 4.2 or Theorem 4.1 that the tracking error converges to 0 for vanishing sampling time T_s. Since the effect of the time delay can effectively not be compensated by the eigenvalues of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ due to the definition of F in (4.82), this shows that the sampling time T_s is the crucial parameter affecting the probabilistic tracking error bound (4.88).

4.3.3. Event-Triggered Learning

While using a fixed sampling time T_s allows us to ensure arbitrarily small tracking error bounds $\bar{e}$ using sufficiently high sampling rates, it generally causes large amounts of data which require significant computational resources to be processed online. In order to mitigate this issue, the training data can be generated on a need basis by adapting the sampling scheme to the realized trajectory $\boldsymbol{x}(\cdot)$. This can be achieved through an event-triggered sampling strategy, where the sampling times t_n are defined via conditions for the tracking error $\boldsymbol{e}(t)$. For the linear system (3.2), we employ the triggering condition

$$t_{n+1} = \inf_{t > t_n} t \tag{4.92}$$

$$\text{such that } \|\boldsymbol{U}^{-1}\boldsymbol{e}(t)\| = \frac{\bar{e}}{\|\boldsymbol{U}\|} \tag{4.93}$$

Algorithm 4.2. Event-triggered learning for GP-based control

```
Initialize GP prior
while t < T do
    Apply control input defined by GP-based control law
    if ‖ẽ(t)‖ = ē/‖U‖ then
        Take measurement x^(N+1), y^(N+1) = f(x^(N+1)) + ε^(N+1), N ← N + 1
        Update GP mean μ(·) in the control law
    end if
end while
```

for determining sampling times t_n, which leads to the event-triggered online learning approach outlined in Algorithm 4.2. Since each training sample is measured when the tracking error $\boldsymbol{e}(t)$ is high, the accuracy of the GP mean function $\mu(\cdot)$ is improved whenever necessary. Thereby, a time-varying data set

$$\mathbb{D}(t) = \left\{ \boldsymbol{x}^{(n)} = \boldsymbol{x}(t_n), y^{(n)} = f(\boldsymbol{x}^{(n)}) + \epsilon^{(n)} \right\}_{n=1}^{N_{\mathrm{trig}}(t)} \tag{4.94}$$

is created, where $N_{\mathrm{trig}}(t)$ denotes the number of triggering instances occurring up to time t.

As the accuracy guaranteed by an additional training sample can be quantified through the uniform error bound $\eta(\cdot)$ and Proposition 4.1, the behavior of a tracking error bound after a triggered measurement can be straightforwardly analyzed. This allows us to provide the following guarantees for the tracking error $\boldsymbol{e}(t)$ and the inter-event times $t_n - t_{n-1}$ achieved by Algorithm 4.2.

Theorem 4.5. *Consider a system* (3.2) *satisfying Assumption 3.1, to which a control law* (3.4) *is applied to track a reference trajectory with bounded derivative* $\dot{\boldsymbol{x}}_{\mathrm{ref}}(\cdot)$. *Assume that a Gaussian process* $\mathcal{GP}(0, k(\cdot,\cdot))$ *is employed to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *online using Algorithm 4.2, such that Assumption 3.2 holds on a compact set* $\mathbb{S} \subset \mathbb{X}$ *with a uniform error bound* $\eta(\cdot)$ *which admits a Lipschitz constant* L_η. *If* $\mathbb{B}_{\bar{e}}(\boldsymbol{x}_{\mathrm{ref}}(t)) \in \mathbb{S}$ *holds for all* $t \in [0, T]$, $T \in \mathbb{R}_{0,+}$, *and* $\tilde{\kappa} < -1$ *for*

$$\tilde{\kappa} = -1 - \frac{-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}{L_\eta F} \left(\frac{-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))\bar{e}}{\|\boldsymbol{U}^{-1}\boldsymbol{b}\|\|\boldsymbol{U}\|} - \eta_1 \right), \tag{4.95}$$

where η_1 *denotes the maximal uniform error bound at state* $\boldsymbol{x}$ *of a GP model whose only training input is* $\boldsymbol{x}$, *then, Algorithm 4.2 ensures a probabilistic tracking error bound* $\upsilon(\cdot) = \bar{e}$ *during the time interval* $[0, T]$ *for initial states* $\mathbb{X}_0 = \{\boldsymbol{x}_{\mathrm{ref}}(0)\}$. *Moreover, the inter-event time is lower bounded by*

$$\underline{T}_s = \frac{\tilde{\kappa} - W_0\left(\tilde{\kappa}\mathrm{e}^{\tilde{\kappa}}\right)}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}, \tag{4.96}$$

where $W_0 : [-1/\mathrm{e}, \infty) \to [-1, \infty)$ *denotes the primary branch of the Lambert W function.*

Proof. In order to prove this theorem, we need to ensure that the uniform error bounds hold for all GP models generated by Algorithm 4.2. Therefore, we choose $\delta = {}^{\tilde{\delta}\underline{T}_s}/_{T}$, $\tilde{\delta} \in (0, 1)$ for determining the uniform error bound η_1, such that the union bound guarantees the

joint satisfaction of η_1 with probability $1-\tilde{\delta}$ for ${}^{T}/_{\underline{T}_s}$ GP models with different data sets. Moreover, we perform a change of variables and define $\tilde{\boldsymbol{e}} = \boldsymbol{U}^{-1}\boldsymbol{e}$. Similarly as in the proof of Theorem 4.4, we can linearize the error bound around a sampling state. By denoting the corresponding time as $t = 0$ for notational convenience, it immediately follows from (3.10) and (4.91) that

$$\|\tilde{\boldsymbol{e}}(t)\| \leq \|\tilde{\boldsymbol{e}}(0)\| \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))t} + \|\boldsymbol{U}^{-1}\boldsymbol{b}\| \int_0^t \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))(t-t')} \left(L_\eta F t' + \eta_1\right) \mathrm{d}t' \tag{4.97}$$

holds until the next measurement is taken, where $\|\dot{\boldsymbol{x}}(t)\| \leq F$ follows from Lemma 4.4. This integral can be straightforwardly determined, such that exploiting the fact that $\|\tilde{\boldsymbol{e}}(0)\| \leq {}^{\bar{e}}/_{\|\boldsymbol{U}\|}$ at the triggering time results in

$$\begin{aligned} &\|\tilde{\boldsymbol{e}}(t)\| \leq \\ &\frac{\bar{e}}{\|\boldsymbol{U}\|}\mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))t} + \frac{\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}\left(\eta_1\left(\mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))t}-1\right) + L_\eta F\left(-t + \frac{1}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}\left(\mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))t}-1\right)\right)\right). \end{aligned} \tag{4.98}$$

Due to (4.95), it follows that

$$c = \frac{\bar{e}}{\|\boldsymbol{U}\|} + \|\boldsymbol{U}^{-1}\boldsymbol{b}\|\left(\frac{L_\eta F}{\bar{\lambda}^2(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))} + \frac{\eta_1}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}\right) > 0. \tag{4.99}$$

Therefore, we have

$$\|\tilde{\boldsymbol{e}}(t)\| \leq c\mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))t} + \frac{\bar{e}}{\|\boldsymbol{U}\|} - c - \frac{L_\eta F\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}t. \tag{4.100}$$

Since $\tilde{\kappa} > 1$, this bound is decreasing at $t = 0$, which directly implies that $\|\tilde{\boldsymbol{e}}(t)\| < {}^{\bar{e}}/_{\|\boldsymbol{U}\|}$ for some time interval $t \in (0, T_s)$. Since a new measurement is triggered as soon as $\|\tilde{\boldsymbol{e}}(t)\| = {}^{\bar{e}}/_{\|\boldsymbol{U}\|}$, this ensures $\|\tilde{\boldsymbol{e}}(t)\| \leq {}^{\bar{e}}/_{\|\boldsymbol{U}\|}$ and consequently $\|\boldsymbol{e}\| \leq \|\boldsymbol{U}\|\|\tilde{\boldsymbol{e}}\| = \bar{e}$ for all $t \in [0, T]$ if not more than ${}^{T}/_{\underline{T}_s}$ update events are triggered. The time T_s between two update events can be lower bounded by solving (4.100) for t when $\|\tilde{\boldsymbol{e}}(t)\| = {}^{\bar{e}}/_{\|\boldsymbol{U}\|}$. This yields the equality

$$0 = \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))t} - 1 - \frac{L_\eta F\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{c\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}t, \tag{4.101}$$

which can be directly solved for t via the Lambert W function resulting in (4.96). Hence, no more than ${}^{T}/_{\underline{T}_s}$ different models are used, which concludes the proof. □

Since the time $t_n - t_{n-1}$ between sampling events can be arbitrarily large, we can generally not ensure that the uniform error bound $\eta(\cdot)$ for GP regression decreases below η_1 at sampling times t_n. This makes it necessary to restrict the admissible error bounds $\bar{e}$ in Algorithm 4.2 using condition $\tilde{\kappa} < -1$ in order to ensure that the tracking error $\boldsymbol{e}(t)$ actually decreases after the trigger event. The value $\tilde{\kappa}$ measures the distance of the desired error bound $\bar{e}$ to the minimum admissible value

$$\bar{e}_{\min} = \frac{\|\boldsymbol{U}^{-1}\boldsymbol{b}\|\|\boldsymbol{U}\|\eta_1}{-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}, \tag{4.102}$$

for which continuous triggering would be necessary. Therefore, (4.96) exhibits the intuitive behavior that for $\bar{e} \to \bar{e}_{\min}$, it potentially becomes necessary to sample more than once at the same time instance t_n, i.e., $\underline{T}_s \to 0$.

Algorithm 4.3. Episodic learning with arbitrary accuracy guarantees

1: Initialize GP prior, $i \leftarrow 0$
2: Compute $\bar{v}_0$ using (3.14) for τ satisfying (4.52)
3: **while** $\bar{v}_i > \bar{e}$ **do**
4: $\quad i \leftarrow i + 1$
5: $\quad$ Apply the controller (3.4) to the system (3.2) and collect training samples
6: $\quad$ Choose $\boldsymbol{\theta}$ such that (4.104) holds
7: $\quad$ Find T_s such that (4.105) is satisfied for a set $\mathbb{D}_N^{T_s}$ of collected samples
8: $\quad$ Determine $\mu_{T_s}(\cdot)$ and $\sigma_{T_s}^2(\cdot)$ based on the data set $\mathbb{D}_N^{T_s}$
9: $\quad$ Compute $\bar{v}_i$ using (3.14) for τ satisfying (4.52)
10: **end while**

4.3.4. Episodic Learning

When online learning is not possible due to computational restrictions, we have to update the GP model offline. Therefore, we develop an episodic approach for generating training data sets in this section. For simplicity, we consider a constant sampling time $T_s \in \mathbb{R}_+$ during each episode with execution time $T_p \in \mathbb{R}_+$, which yields data sets of the form

$$\mathbb{D}_N^{T_s} = \left\{ (\boldsymbol{x}(nT_s), f(\boldsymbol{x}(nT_s)) + \epsilon^{(n)}) \right\}_{n=0}^{N_p}, \tag{4.103}$$

where $N_p = \lfloor 1 + {}^{T_p}/_{T_s} \rfloor$ denotes the number of training samples gathered during one episode. Therefore, the tracking error bound $\bar{v}$ from one episode immediately provides guarantees for the training data of the next episode. We exploit this by adjusting the sampling time T_s and the maximum eigenvalue $\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))$ as demonstrated in Algorithm 4.3 in order to ensure a sufficiently small error bound for the next episode. An index T_s in the posterior standard deviation $\sigma_{T_s}(\cdot)$ emphasizes this dependency on the sampling time. As shown in the following theorem, this approach guarantees the termination of Algorithm 4.3 after a finite number of iterations.

Theorem 4.6. *Consider a system* (3.2) *satisfying Assumption 3.1, to which a control law* (3.4) *with control gains* $\boldsymbol{\theta}$ *satisfying Assumption 3.3 is applied to track a bounded reference trajectory* $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$. *Assume that a Gaussian process* $\mathcal{GP}(0, k(\cdot,\cdot))$ *with Lipschitz continuous, stationary kernel* $k(\cdot,\cdot)$ *is used to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *of the unknown, Lipschitz continuous function* $f(\cdot)$, *such that Assumptions 2.1 and 2.6 are satisfied. If* $\boldsymbol{\theta}$ *and* T_s *are chosen such that*

$$-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) \geq 2\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\| \sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)} \left(\frac{8\sqrt{L_{\partial k}}}{c} + L_\sigma\right) \tag{4.104}$$

$$\max_{0 \leq t \leq T_p} \sigma_{T_s}^2(\boldsymbol{x}_{\mathrm{ref}}(t)) \leq 16 L_{\partial k} \bar{v}_{i-1}^2 \tag{4.105}$$

holds in every episode for $c < 1$, *Algorithm 4.3 ensures a probabilistically bounded tracking error* $\bar{e}$ *and terminates after at most*

$$N_E = \left\lceil \frac{\log\left(4\bar{e}\sqrt{L_{\partial k}}\right) - \log\left(\sqrt{k(\mathbf{0})}\right)}{\log(c)} \right\rceil \tag{4.106}$$

episodes with probability of at least $1 - N_E\delta$.

Proof. It immediately follows from Proposition 3.1 and Proposition 2.2 that

$$\bar{\upsilon} = -\frac{2\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) + L_\eta\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}\sqrt{2\log\left(\frac{M(\tau,\mathbb{S})}{\delta}\right)}\max_{0\leq t\leq T_p}\sigma_{T_s}(\boldsymbol{x}_{\mathrm{ref}}(t)) \tag{4.107}$$

for τ satisfying (4.52). Due to (4.104), this yields

$$\bar{\upsilon}_0 = \frac{c\sqrt{k(\mathbf{0})}}{4\sqrt{L_{\partial k}}}, \tag{4.108}$$

$$\bar{\upsilon}_{i+1} = \frac{c}{4\sqrt{L_{\partial k}}}\max_{0\leq t\leq T_p}\sigma_{T_s}(\boldsymbol{x}_{\mathrm{ref}}(t)), \tag{4.109}$$

where the index i is used to denote the episode. Due to (4.105), this directly implies the recursion

$$\bar{\upsilon}_{i+1} \leq c\bar{\upsilon}_i \tag{4.110}$$

with probability $1-\delta$ in each episode. Therefore, Algorithm 4.3 ensures a probabilistic tracking error bound $\bar{e}$ after at most N_E episodes with probability $1-N_E\delta$ due to the union bound. □

Due to the exponential decay of the tracking error bound $\bar{\upsilon}$ ensured by Theorem 4.6, Algorithm 4.3 quickly terminates. This comes at the price of higher requirements (4.104) on the eigenvalues of $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})$ compared to the online learning scenarios presented in Sections 4.3.2 and 4.3.3. However, the difference is small, and it is indeed straightforward to see that $-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})) \propto 1/\sqrt{\log(\bar{e})}$ is sufficient to compensate the effect of an increasing scaling factor in the uniform error bound $\eta(\cdot)$ for all polynomially growing data sets. Therefore, this requirement is still significantly lower compared to ensuring the tracking error bound $\bar{e}$ without learning as guaranteed by Lemma 4.3.

While Theorem 4.6 provides requirements on the sampling time T_s, the condition (4.105) cannot be computed before the controller is applied to the system. However, it can easily be verified a posteriori, such that we can simply downsample the obtained data to the necessary sampling time T_s. The required maximum sampling rate can be bounded using the following proposition.

Proposition 4.4. *Consider a system* (3.2) *satisfying Assumption 3.1, to which a control law* (3.4) *with control gains* $\boldsymbol{\theta}$ *satisfying Assumption 3.3 is applied to track a bounded reference trajectory* $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$*. Assume that a Gaussian process* $\mathcal{GP}(0, k(\cdot,\cdot))$ *with an isotropic SE kernel* $k(\cdot,\cdot)$ *is used to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *of the unknown, Lipschitz continuous function* $f(\cdot)$*, such that Assumptions 2.1 and 2.6 are satisfied. Then, the sampling time* T_s *required by condition* (4.105) *in Algorithm 4.3 is bounded by*

$$T_s \geq \underline{T}_s = \frac{16L_{\partial k}\bar{e}^3}{\sigma_{\mathrm{on}}^2\max_{0\leq t\leq T_p}\|\dot{\boldsymbol{x}}(t)\|}. \tag{4.111}$$

Proof. We prove this proposition by deriving a value of T_s which satisfies (4.105) Due to Proposition 4.1, (4.105) is guaranteed to hold if $\underline{h} \geq 1/(8L_{\partial k}\sigma_f^2\bar{\upsilon}_{i-1}^2)$. Set $h = 1/(8L_{\partial k}\sigma_f^2\bar{\upsilon}_{i-1}^2)$. Then, it follows from Lemma 4.2 that

$$\mathbb{B}_{2\upsilon_{i-1}}(\boldsymbol{x}_{\mathrm{ref}}(t)) \subset \mathbb{K}_h(\boldsymbol{x}_{\mathrm{ref}}(t)). \tag{4.112}$$

The Euclidean ball around $\boldsymbol{x}_{\mathrm{ref}}(t)$ on the left-hand side can be inner bounded by a Euclidean ball with half the radius around the actual trajectory, i.e.,

$$\mathbb{B}_{\bar{v}_{i-1}}(\boldsymbol{x}(t)) \subset \mathbb{B}_{2\bar{v}_{i-1}}(\boldsymbol{x}_{\mathrm{ref}}(t)). \tag{4.113}$$

The smaller Euclidean ball has a diameter of $\bar{v}_{i-1}$, and the actual trajectory passes through its center. Moreover, the distance between two samples can be bounded by $T_s \max_{0\leq t\leq T_p} \|\dot{\boldsymbol{x}}(t)\|$, where $\|\dot{\boldsymbol{x}}(t)\|$ is bounded due to Lemma 4.4. This allows us to bound the number of points in $\mathbb{K}_h(\boldsymbol{x}_{\mathrm{ref}}(t))$ by

$$|\mathbb{K}_h(\boldsymbol{x}_{\mathrm{ref}}(t))| \geq |\mathbb{B}_{\bar{v}_{i-1}}(\boldsymbol{x}(t))| \geq \frac{2\bar{v}_{i-1}}{T_s \max_{0\leq t\leq T_p} \|\dot{\boldsymbol{x}}(t)\|}. \tag{4.114}$$

For $\underline{h} \geq h$, it must hold that

$$\frac{2v_{i-1}}{T_s \max_{0\leq t\leq T_p} \|\dot{\boldsymbol{x}}(t)\|} \geq h\sigma_{\mathrm{on}}^2 k(\mathbf{0}) = \frac{\sigma_{\mathrm{on}}^2}{8L_{\partial k} v_{i-1}^2} \tag{4.115}$$

due to (4.30). This inequality can be ensured to hold by setting

$$T_s = \frac{16L_{\partial k}\bar{v}_{i-1}^3}{\sigma_{\mathrm{on}}^2 \max_{0\leq t\leq T_p} \|\dot{\boldsymbol{x}}(t)\|}, \tag{4.116}$$

which concludes the proof. □

This result clearly shows that the necessary sampling rate for ensuring a desired tracking error bound $\bar{e}$ converges to 0 when $\bar{e}$ vanishes. Due to the restriction to SE kernels, the required rate in (4.111) can be quantified and easily computed, e.g., by bounding the state derivative $\dot{\boldsymbol{x}}(t)$ using Lemma 4.4. Therefore, Proposition 4.4 allows us to effectively determine the maximally required sampling rate for implementing Algorithm 4.3.

4.3.5. Numerical Evaluation

We illustrate the different data generation strategies by considering the setting in Section 3.1.4 again. Therefore, we have a linear closed-loop system described by

$$\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \begin{bmatrix} 0 & 1 \\ -k_c\tilde{\theta} & -k_c - \tilde{\theta} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.80 revisited}$$

which is perturbed by the nonlinear input perturbation $f(\boldsymbol{x}) = 1 - \sin(2x_1) + 1/(1+\exp(-x_2))$. If not explicitly specified otherwise, the control gains are defined as $k_c = 10$ and $\tilde{\theta} = 5$.

We first demonstrate the performance dependency of the time-triggered online learning in Algorithm 4.1 on the sampling time T_s. Next, we evaluate the event-triggered online learning using Algorithm 4.2 and investigate the behavior of the inter-event times caused by different specified error bounds $\bar{e}$. Finally, the effectiveness of the episodic offline learning approach outlined in Algorithm 4.3 is presented.

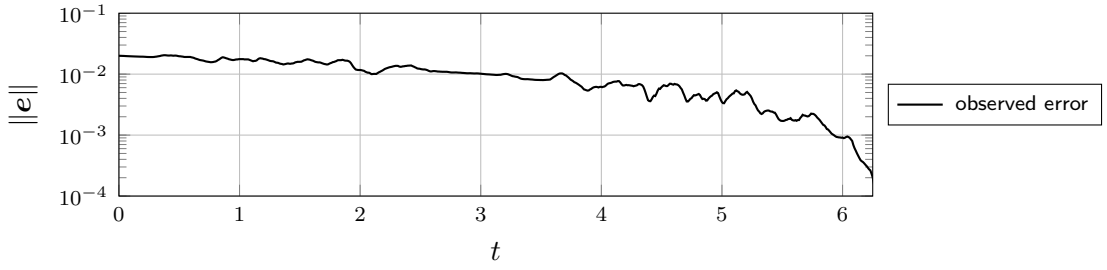

Figure 4.8.: Examplary evolution of the tracking error norm $\|\boldsymbol{e}(\cdot)\|$ with time-triggered online learning as outlined in Algorithm 4.1.

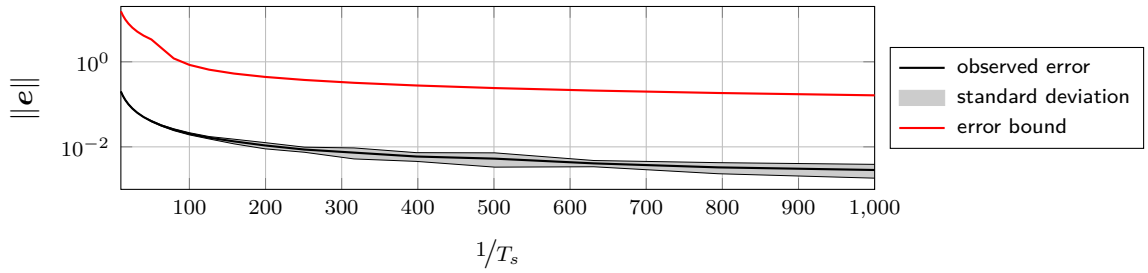

Figure 4.9.: Maximum tracking error and its bound $\bar{v}$ in dependency of the sampling time T_s for time-triggered learning as outlined in Algorithm 4.1.

Time-triggered Online Learning

For determining the probabilistic tracking error bound using Theorem 4.4, we iterate between computing F using (4.84) and evaluating (4.88) until convergence is reached. The required optimization problem (4.85) for obtaining η_{T_s} is approximated by fixing the number of considered training samples to $N = 100$, such that a closed-form solution can be straightforwardly seen for the employed ARD SE kernel.

A resulting exemplary evolution of the tracking error norm $\|\boldsymbol{e}(\cdot)\|$ is depicted in Fig. 4.8. Due to the large length scales $l_1 = 2.04$ and $l_2 = 0.86$ in the ARD SE kernel (2.18), the tracking error slowly decreases over time. In particular, the periodic reference trajectory causes a significant increase of the training data density towards the end of the period time $T_p = 2\pi$, which results in a clearly visible reduction of the tracking error. While this effect is not considered by the probabilistic tracking error bound in Theorem 4.4, it is straightforward to extend the derived theory to reflect the data density increase with periodic reference trajectories $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$.

Due to the randomness in training samples caused by the observation noise, the dependency of the worst-case error on the sampling time T_s is investigated by averaging over 10 simulation runs. As illustrated in Fig. 4.9, the resulting curve decreases with the sampling frequency $1/T_s$ with a similar rate as the probabilistic tracking error bound $\bar{e}$. The variation caused by the randomness is marginal, as visualized by the shaded area, which reflects the 3 times standard deviation. Therefore, the time-triggered online learning strategy described in Algorithm 4.1 exhibits the behavior guaranteed by Theorem 4.4 and enables the effective reduction of the tracking error norm $\|\boldsymbol{e}(\cdot)\|$ by decreasing the sampling time T_s.

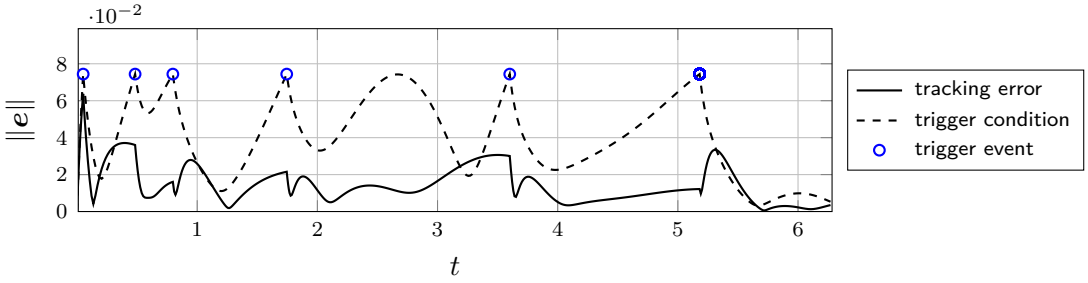

Figure 4.10.: Examplary evolution of the tracking error norm $\|\boldsymbol{e}(t)\|$ with event-triggered online learning as outlined in Algorithm 4.2.

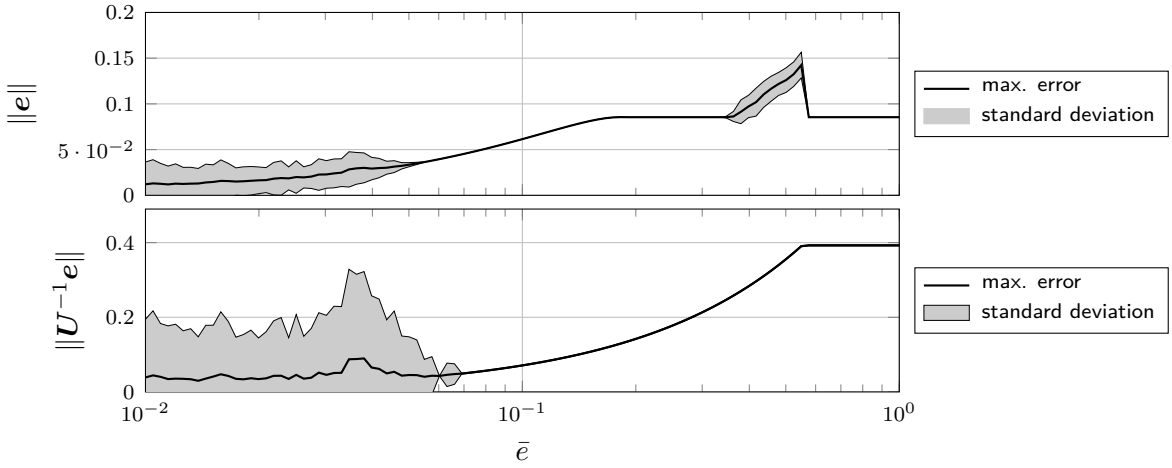

Figure 4.11.: Maximum tracking error in dependency of the prescribed error $\bar{e}$ for event-triggered learning as outlined in Algorithm 4.2.

Event-triggered Online Learning

While the time-triggered learning approach generates a continuous stream of data, a desired tracking error bound $\bar{e}$ can be ensured with a higher data efficiency using the event-triggered learning approach outlined in Algorithm 4.2, which is illustrated for $\bar{e} = 0.10$ in Fig. 4.10. Whenever the value $\|\boldsymbol{U}^{-1}\boldsymbol{e}\|$ reaches the threshold $\bar{e}/\|\boldsymbol{U}\| = 0.074$, a training sample is generated and the model is updated. After these trigger events, symbolized with blue circles, the tracking error decreases, such that overall only a small number of GP updates must be computed to ensure the desired error bound $\bar{e}$.

This behavior can be observed for a wide range of prescribed error bounds $\bar{e}$ as depicted in Fig. 4.11. Due to the randomness of training samples, this figure shows again average curves for 100 simulation runs. When the specified error $\bar{e}$ is large, no training data is necessary, such that the curves for $\|\boldsymbol{e}(\cdot)\|$ and $\|\boldsymbol{U}^{-1}\boldsymbol{e}\|$ are constant. While the bound $\|\boldsymbol{U}^{-1}\boldsymbol{e}(\cdot)\|$ exhibits an almost continuous decrease for smaller values $\bar{e}$ with practically no variation, the actual tracking error $\|\boldsymbol{e}(\cdot)\|$ suffers from a steep increase when GP updates start. However, this is accompanied by a jump in the standard deviation, which indicates that this is a consequence of the observation noise, which causes a performance deterioration of the GP model with very few training samples in a few simulation runs. This artifact vanishes as $\bar{e}$ becomes smaller and quickly disappears, such that both curves decrease for the remaining considered range of error bounds $\bar{e}$. Interestingly, the standard deviation starts to increase

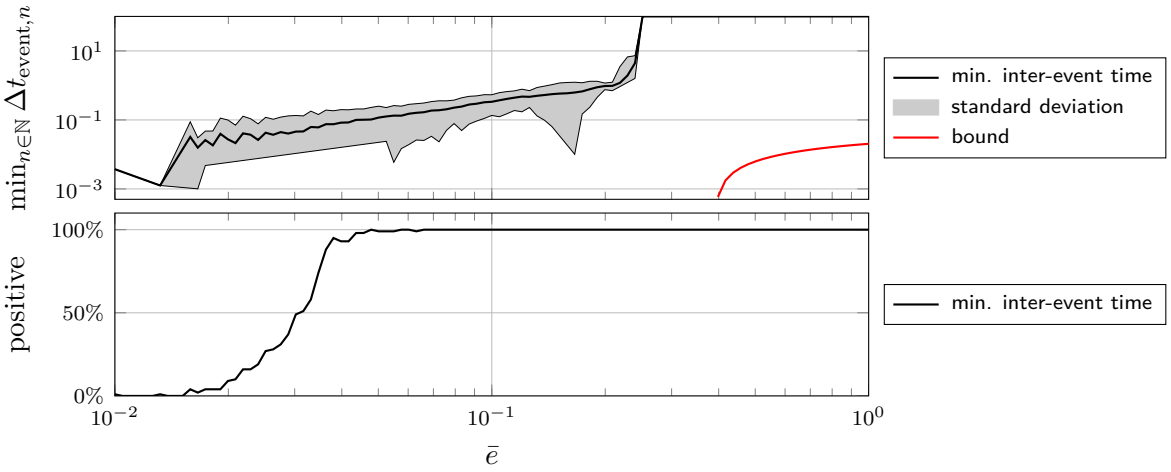

Figure 4.12.: Top: Minimum inter-event time $\min_{n\in\mathbb{N}} \Delta t_{\text{event},n}$ together with its bound in dependency of the prescribed tracking error $\bar{e}$ for event-triggered learning as described in Algorithm 4.2. Bottom: Percentage of non-zero minimum inter-event times depending on the prescribed tracking error $\bar{e}$.

for small values $\bar{e}$ in both plots.

This behavior can be easily explained when looking at the minimum inter-event times illustrated in Fig. 4.12. While the average of positive minimum inter-event times exhibits an approximately constant decay rate, the theoretical error bound does not ensure this. Due to the conservatism of the theoretical analysis, this is not a problem for comparatively large prescribed error bounds $\bar{e}$, but it eventually leads to minimum inter-event times becoming 0 in some simulation runs, as illustrated at the bottom of Fig. 4.12. This occurs precisely around the values of $\bar{e}$ for which the standard deviation increase in Fig. 4.11 can be observed. Therefore, the triggered GP updates cannot ensure a sufficient model accuracy increase, which leads to a potential growth of the tracking error. Since this does not happen in all simulation runs due to the randomness of training data, this causes the variation in the observed tracking accuracy. This demonstrates that Theorem 4.5 describes a practically relevant effect by providing a lower bound for the certifiable tracking accuracy, even though the theoretically guaranteed bound is conservative.

Episodic Learning

If we need to process the training data offline due to computational constraints, the episodic learning strategy outlined in Algorithm 4.3 can be employed. For realizing this approach, we set $k_c = \tilde{\theta}$ and choose k_c such that $c = 0.95$ holds in every iteration. A high-frequency data set with sampling time $3 \cdot 10^{-4}$ is generated in every episode, such that a line search can be used to determine the maximum value of T_s satisfying (4.105).

The tracking error bounds obtained from Algorithm 4.3 with these parameters are exemplarily illustrated for several different episodes in Fig. 4.13. Due to the constant sampling time, the training data density along the reference is very similar within an episode, which directly leads to the minor variations in the tracking error bound over time. Moreover, it can be seen that the decrease of the tracking error bound $\upsilon(\cdot)$ is significantly larger during the first few episodes before it slows down. This becomes even clearer when plotting the

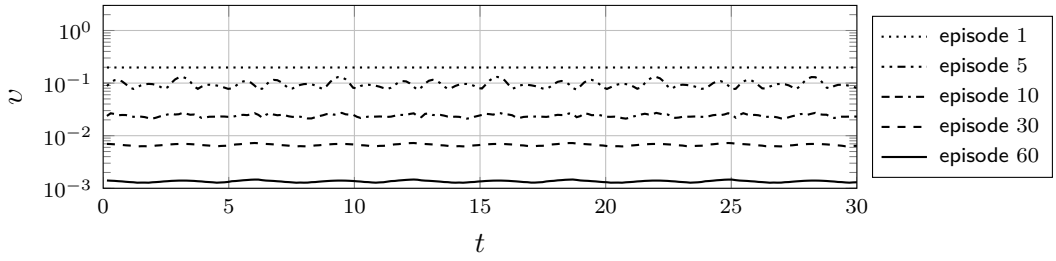

Figure 4.13.: Tracking error bounds $v(t)$ for different episodes of Algorithm 4.3.

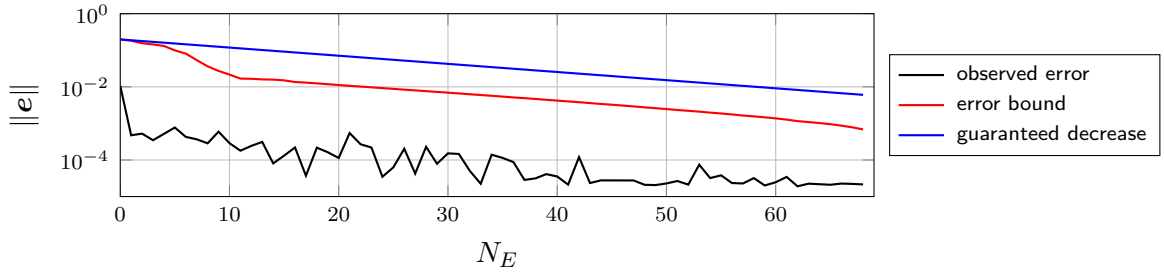

Figure 4.14.: Decay rate of the tracking error bound $\bar{v}$ and the observed tracking error norm $\|\boldsymbol{e}\|$ resulting from Algorithm 4.3.

behavior of the error bound over the number of episodes as depicted in Fig. 4.14. During the first 10 episodes, the error bound $\bar{v}$ decays faster than the guaranteed rate of $c^{N_E}\bar{v}_0$, which is guaranteed by Theorem 4.6. This can be attributed to the fact that even a single additional data point reduces the posterior variance more than required for (4.105) at the beginning. Once a sufficiently large number of additional training samples is necessary to ensure (4.105), this inaccuracy is overcome, and the error bound $\bar{v}$ closely follows the guaranteed decrease rate. In fact, while being rather conservative similar to the previous simulations, the tracking error bound $\bar{v}$ even reflects the behavior of the actually observed tracking error $\|\boldsymbol{e}\|$ accurately after 10 episodes.

Note that this unexpected fast decay at the beginning has no influence on the required maximum eigenvalues $\bar{\lambda}(\boldsymbol{A}_{\boldsymbol{\theta}})$ as depicted at the top of Fig. 4.15. While smaller eigenvalues are required for the episodic approach compared to the asymptotic analysis in Section 4.2.4, the maximum eigenvalue $\bar{\lambda}(\boldsymbol{A}_{\boldsymbol{\theta}})$ used in Algorithm 4.3 closely follow the expected $\mathcal{O}(1/\sqrt{\log(\bar{e})})$ behavior. Moreover, it can be directly seen that Algorithm 4.3 offers a significant advantage over a direct reduction of the tracking error bound using the maximum eigenvalue $\bar{\lambda}(\boldsymbol{A}_{\boldsymbol{\theta}})$ without compensation of the nonlinearity. Note that the sampling time T_s necessary to achieve this behavior quickly decays as illustrated at the bottom of Fig. 4.15. However, since it stays significantly larger than its theoretical bound $\underline{T}_s$, it remains in magnitudes that can be realized in practice. Therefore, Algorithm 4.3 provides an effective method for generating data, such that a GP model can be trained offline to ensure an arbitrary tracking error $\bar{e}$.

4.4. Discussion

While the presented approach for analyzing the impact of training data on the derived uniform error bounds in Section 4.1 is applicable to a wide range of kernels in principle, it is

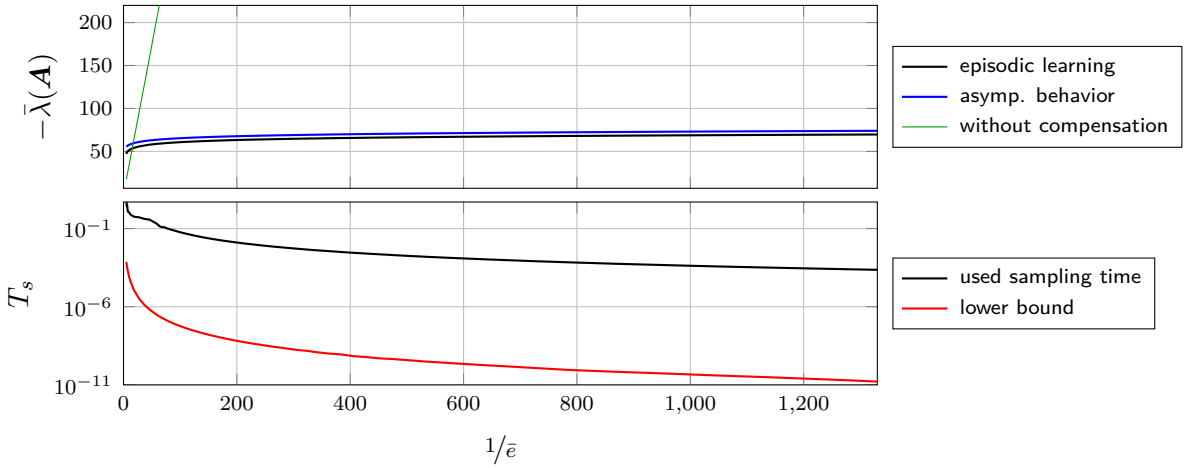

Figure 4.15.: Top: Maximum eigenvalue $\bar{\lambda}(\boldsymbol{A})$ necessary to ensure a given tracking error bound $\bar{e}$ when learning a control law using Algorithm 4.3 in comparison to a pure feedback controller without compensation of nonlinearities. Bottom: Employed sampling time T_s together with its lower bound $\underline{T}_s$ for given error bounds $\bar{e}$.

particularly well-suited for stationary covariance functions. This is clearly visible from the analysis in Section 4.1.4, which shows that the global data dependency of linear kernels can only be partially reflected by the derived density measure $h(\cdot)$. Since stationary kernels are predominantly used in GP models for control applications at the moment, this weakness of our approach is not critical for our addressed problems. Therefore, we leave the improvement of our results for non-stationary kernels for future research, e.g., by investigating the data dependency of the parametric error bounds derived in Section 2.2.1.

Due to the benefits of our analysis for stationary kernels, the derived bounds for the posterior variance have a highly local nature, which allows the investigation of individual test inputs through the training samples in its proximity. This local dependency fits excellently to the requirements of the performance guarantees for the tracking accuracy derived in Chapter 3, which depend on the local model accuracy. Therefore, the locality of the density measure $h(\cdot)$ enables the restriction of the training data analysis along the reference trajectory in Section 4.2.2 and a state-dependent derivation of data density requirements in Section 4.2.3. For the latter example, the provided analysis illustrates an intuitive insight that has independently been posed as a condition for suitable training samples in [150]: a high control input magnitude is necessary to ensure a small prediction error of the model $\mu_g(\cdot)$ for the function $g(\cdot)$ when merely data of the control-affine function $f(\boldsymbol{x}) = \tilde{f}(\boldsymbol{x}) + g(\boldsymbol{x})u$ are available. Therefore, Section 4.2 presents an effective approach for relating desired tracking accuracies to requirements on the training data. It should be noted, however, that the posterior variance bound in Lemma 4.1, which is the foundation of our analysis, is generally not tight. When the global model accuracy on a compact domain is of concern, it is well-known that information-theoretic concepts allow improved bounds [67]. The combination of such information-theoretic concepts with our control-oriented analysis remains an open problem for future research, which potentially can be addressed by transferring ideas from safe Bayesian optimization such as [151].

The insights gained from the analysis in Section 4.2 lead to the development of effective

methods for ensuring a desired tracking accuracy using GP-based learning in Section 4.3. While the derived guarantees for the presented learning control approaches are rather conservative, they allow the valuable conclusion that we can achieve arbitrary tracking accuracies through sufficiently high sampling rates, both in offline and online learning. The proposed strategies can be straightforwardly realized in practice and have been successfully demonstrated in real-world applications [26, 27, 152]. Therefore, our derived results in Section 4.2 provide an important theoretical foundation for practically employed and highly relevant learning approaches. While they can be easily extended beyond linear systems with unknown input perturbation, e.g., by employing the Lyapunov approach proposed in Section 3.2, they are limited to unknown dynamic components depending only on the system state, but not the control inputs. This is due to the necessity of active exploration to excite more general unknown functions, which directly follows from the results of Section 4.2.3. Hence, the development of suitable learning and data generation approaches for general unknown components in dynamical systems remains an open problem.

Efficient Learning via Gaussian Process Model Aggregation

5.

While Gaussian processes have many beneficial theoretical properties for control, as demonstrated in the previous chapters, their practical application is generally challenging. This is mainly due to the computational complexity of both their predictions and model updates, which strongly grows with the number of training samples. While there exist many approaches to enable the learning of models from large data sets [153], the problem of developing suitable methods for learning in control loops has not specifically been addressed.

Control applications are a challenging problem for developing GP-based learning methods in many different aspects. Control loops often run at high sampling rates [154], such that data needs to be processed quickly and sequentially, and models used in control laws need to be evaluated with low latency. Furthermore, control systems are often composed of multiple individual agents, which usually have the capability to communicate with each other [155]. Since this implies that data is scattered across physically separated computing units, extracted model knowledge needs to be efficiently shared between agents for the effective control of multi-agent systems. Finally, many control problems require performance guarantees in order to ensure the safe operation of systems [156]. Therefore, model accuracy guarantees such as probabilistic uniform error bounds must exist for practical GP-based learning in control loops.

We address this challenging problem of developing learning methods based on GP regression for control systems using the aggregation of individual GP models [157]. The structure of common aggregation methods can be generally expressed as

$$\tilde{\mu}(\boldsymbol{z}) = \psi_\mu \left(\sum_{m \in \mathbb{M}} \omega_m \boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{z}), \sigma_m^2(\boldsymbol{z})) \right), \tag{5.1}$$

$$\tilde{\sigma}^2(\boldsymbol{z}) = \psi_\sigma \left(\sum_{m \in \mathbb{M}} \omega_m \boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{z}), \sigma_m^2(\boldsymbol{z})) \right), \tag{5.2}$$

where ω_m are weighting factors, $\mathbb{M}$ denotes the index set of the individual models and $\psi_\mu, \psi_\sigma : \mathbb{R}^{d_\psi} \to \mathbb{R}$ and $\boldsymbol{\psi}_\omega : \mathbb{R}^2 \to \mathbb{R}^{d_\psi}$ are nonlinear functions. For example, a mixture of GP experts approach [158] corresponds to

$$\boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{z}), \sigma_m^2(\boldsymbol{z})) = \begin{bmatrix} \mu_m(\boldsymbol{z}) \\ \sigma_m^2(\boldsymbol{z}) + \mu_m^2(\boldsymbol{z}) \end{bmatrix}, \tag{5.3}$$

$$\psi_\mu(\boldsymbol{\psi}_\omega) = \psi_{\omega,1}, \tag{5.4}$$

$$\psi_\sigma(\boldsymbol{\psi}_\omega) = \psi_{\omega,1} - \psi_{\omega,1}^2 \tag{5.5}$$

for $\boldsymbol{\psi}_\omega \in \mathbb{R}^2$, which is often used in the form of a mixture of explicitly localized experts [159], see, e.g., [142, 160]. Similarly, the generalized product of GP experts approach [161] can be

obtained by choosing

$$\boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{z}), \sigma_m^2(\boldsymbol{z})) = \begin{bmatrix} \mu_m(\boldsymbol{x})\sigma_m^{-2}(\boldsymbol{x}) \\ \sigma_m^{-2}(\boldsymbol{x}) \end{bmatrix}, \tag{5.6}$$

$$\psi_\mu(\boldsymbol{\psi}_\omega) = \frac{\psi_{\omega,1}}{\psi_{\omega,2}}, \tag{5.7}$$

$$\psi_\sigma(\boldsymbol{\psi}_\omega) = \frac{1}{\psi_{\omega,2}}. \tag{5.8}$$

Based on such aggregation schemes, we propose a computationally efficient method for GP-based online learning in control loops in Section 5.1. In this section, which is based on [38], we also show that probabilistic uniform error bounds derived for exact GP regression straightforwardly transfer to many practically used aggregation schemes. In order to enable the application of GP regression in multi-agent systems with a distributed computing infrastructure, we propose a consensus-based aggregation in Section 5.2. We demonstrate the improved data efficiency of this distributed learning approach by employing it in a cooperative control law and derive probabilistic tracking error bounds. These results are based on [39]. Finally, this chapter is concluded by a discussion in Section 5.3.

5.1. Computationally Efficient Online Learning with Error Bounds

In order to practically realize the online learning control approaches presented in Section 4.3, predictions and model updates must typically be performed in real time due to the fast evolution of many physical processes. These applications include the control of autonomous cars [162], unmanned aerial vehicles [163], robotic manipulators [164], combustion engines [165], and many others, where update rates in the magnitude of 10^2 Hz to 10^4 Hz are required. In the case of predictive control schemes, where possible future trajectories are inferred and evaluated, multiple predictions are made for a single control command, requiring prediction rates that are orders of magnitudes higher [166].

Since the computational complexity of updates and predictions for GP models grows strongly with the number of training points, many approximations have been developed to enable the employment in real-time applications. Deterministic training conditional approximations [164, 167] and inducing point methods [168, 169] can speed up predictions, while variational inference approaches for streaming data [170] allow fast model updates and exhibit a beneficial performance-complexity trade-off compared to stochastic variational inference [171]. However, error bounds from exact GP inference as derived in Chapter 2 do not extend to these methods, which prevents their usage in applications with performance guarantee requirements. Even though finite feature approximations of kernels [51] are advantageous in this regard and yield constant update and prediction complexities, safety guarantees require an impractically high number of features [172]. Therefore, there is a clear lack of methods that allow updates and predictions in real-time for applications with performance guarantee requirements.

The main contribution of this section is a novel, computationally efficient, GP-based method for real-time predictions and model updates in applications with performance guarantee requirements, called locally growing random tree of GPs (LoG-GP). Based on aggre-

gation schemes (5.1), we propose an iterative random division of individual models, which results in a random tree as computation graph and guarantees a logarithmic complexity of model updates. In order to reduce the complexity of predictions, the number of necessary individual GP evaluations is limited through the application of locally active models. We prove that uniform error bounds from exact GP inference directly carry over to the proposed method, such that it can be used in applications with performance guarantee requirements. This is demonstrated through the application of LoG-GPs for event-triggered online-learning control. In a comparison on real-world data sets and a control simulation, the superior computational efficiency is demonstrated while providing comparable regression performance to state-of-the-art methods.

The remainder of this section is structured as follows. Due to the extensive literature on GP approximations for reducing the computational complexity, a more detailed overview of GP-based online learning approaches is provided in Section 5.1.1. In Section 5.1.2, the problem setting for GP-based online learning with high update and prediction rates is presented. The LoG-GP approach relying on the aggregation of locally active models is proposed in Section 5.1.3. The computational complexity of the proposed method is analyzed in Section 5.1.4 before probabilistic uniform error bounds for GP regression are extended to aggregated predictions in Section 5.1.5. The beneficial computational efficiency in comparison to state-of-the-art methods is demonstrated in Section 5.1.6. Finally, the straightforward applicability in online learning control with performance guarantees is demonstrated in Section 5.1.7.

5.1.1. Existing Approaches for Gaussian Process-based Online Learning

Although scalability is a major issue of exact Gaussian process regression, a wide variety of methods has been developed in recent years to overcome this problem. An extensive overview of these methods can be found in [153]. Following the classification of methods introduced in this survey article, we distinguish between global and local approximations of GPs for online learning.

Global Approximations

Global GP approximations comprise, by far, the largest group of online learning methods. Among the most common approaches are sparse approximations [173], which aim to reduce the computational complexity of the inverse of the kernel matrix. While there exist approaches exploiting the structural properties of Gram matrices for ensuring a high computational efficiency [174, 175], approximations of the prior and posterior GP distributions are the most common approaches for online learning. The deterministic training conditional (DTC) approximation is a particularly widespread prior approximation since it achieves a constant complexity of predictions by heuristically choosing an active subset from the training data set. For determining the active subset, various methods have been proposed with different complexities [164, 167, 176, 177]. However, the constant complexity of predictions comes at the price of a linear complexity of updates. Fully independent training conditional (FITC) and partially independent training conditional (PITC) approximations follow a similar idea for approximating the prior but use arbitrary inducing points to compress

the information of the original training data. The flexibility of choosing arbitrary inducing points can be used to construct a suitable covering of the input domain online, e.g., through a prior design [168], online clustering techniques [169, 178], or by selecting training inputs as inducing points [179]. While this can be advantageous regarding regression performance, it generally does not positively affect the computational complexity. Therefore, these approximations are best suited for offline training and online predictions but cannot be applied when fast updates are necessary as in, e.g., event-triggered learning.

Subset of regressor approaches are a form of prior approximation, which are rather depreciated in big data problems but have demonstrated to be very successful in online learning problems. The idea of these approaches lies in the approximation of the kernel, such that the complexity of the kernel inverse is reduced. This can be achieved directly via a compactification of covariance functions [180, 181], such that sparsity in the Cholesky factors is ensured. More popularly, finite feature maps are constructed, which allow to approximately express the kernel as a scalar product. This procedure results in constant update and prediction complexities, which only depend on the number of features. For determining the features, different approaches exist. When prior data is available, meta-training can be used to fit features to the training data using neural networks [182] or least squares [183]. Conversely, without any offline data, random trigonometric features with strong theoretical guarantees can be easily determined using Bochner's theorem [184], such that the method is often referred to as sparse spectrum GP [50, 51]. In contrast to most other methods, sparse spectrum GPs directly inherit many theoretical properties from exact Gaussian process regression due to Bochner's theorem [185]. Moreover, when numerical integration is used for obtaining the feature maps instead of random sampling, uniform error bounds can be extended from exact GP inference [172, 186]. However, these bounds often require a practically intractable number of features. Moreover, these methods are known to suffer from overfitting [187], and their posterior variances are overconfident [153]. Furthermore, the posterior mean and variance will be periodic functions, such that the variance might collapse far from any training samples [188], leading to overconfident predictions.

Posterior approximations of GPs do not approximate the prior distribution but instead aim at minimizing the difference between the approximate and exact posterior GP distributions. The most common posterior approximation is the variational free energy [189], which can be efficiently optimized using stochastic optimization methods [171, 190]. Although these approaches can be applied in online learning problems with streaming data in principle, they are usually unsuited for this task, as discussed in [170]. The reasons for this are manifold. First, the optimization methods have the underlying assumption that data is uniformly randomly subsampled into mini-batches. While streaming data can often be aggregated into mini-batches, the data is rarely drawn i.i.d. from the input distribution. Moreover, the data should typically be passed to the optimizer multiple times, which typically cannot be satisfied with streaming data due to computational constraints. Finally, typically only a single gradient step is performed for every mini-batch. Since data cannot be revisited, this causes a risk of forgetting old data. In order to overcome these issues, [170] proposed a posterior approximation for streaming data, which allows online predictions and online updates in mini-batches. While this algorithm achieves a good regression performance, the limitation to mini-batches can be prohibitive in applications such as event-triggered learning, where the update must be performed after every new sample.

Local Approximations and their Relationship to the Proposed Approach

The number of local GP approximations for online learning is significantly lower than for global approximations, but they are frequently used in practical applications. Among the most straightforward approaches are naive local models, which adapt the used data set to the input. This can be achieved, e.g., through a windowing approach [191] or by choosing the data subset based on task-specific information theoretic metrics [27]. Although these approaches achieve a constant update and prediction complexity and typically work well in applications where a local model is sufficient, they frequently suffer from discontinuous mean functions and merely local validity of the models, which prevents their usage in applications such as model predictive control or model-based reinforcement learning. Mixture of experts approaches overcome this issue by composing a global model of multiple locally active Gaussian process experts. While mixture of experts approaches have been initially proposed to address the challenge of multi-modal data [158], explicitly localized models have led to great success in online learning [24, 142, 160]. Since the number of local models and their respective regions in the input domain are not known a priori, they are typically adapted to the streaming data online. The resulting prediction performance crucially depends on the parameters controlling this domain clustering behavior. In order to avoid an excessive number of data points per local model, data points are typically added and removed according to an information criterion. When this happens too often, the regression performance can suffer. However, if too many local models are generated, the computation time increases due to a linear dependency of the update and prediction complexity on the number of models. The trade-off between computation time and prediction performance depends on a few crucial parameters, which are hard to tune, particularly in online learning problems with streaming data. Therefore, the application of mixtures of explicitly localized experts in real-time learning problems is often challenging.

In order to overcome the issues of mixtures of explicitly localized GP experts, our approach employs trees for defining the computing architecture. This idea goes back to [161], where a generalized product of experts approach is used for aggregating individual GP models. In the original approach, the data is split into multiple subsets by constructing a ball-tree [192], which is an efficient method for representing models and allows fast queries of individual leaves of the tree. Although each node of the ball-tree contains a separate GP model and all models are generally evaluated for the prediction of a test point, only evaluating models along the branch assigned to a test point has been investigated, too. Similar ideas have been used in [193], where the tree is employed primarily as a computation graph. In contrast to the ball-tree, a k-d tree is recursively constructed from a batch of data until a prescribed number of leaves containing all the individual GP models is reached. While these approaches exploit methods for the explicit localization of models, this idea is dropped in [157]. Instead, the Bayesian Committee Machine proposed in [194] is adapted for aggregating the predictions of Gaussian process models, such that higher importance is put on models with low posterior variance. In order to address consistency concerns of Bayesian Committee Machines, different extensions of this approach have been proposed [195, 196]. Although these approaches can scale GPs to millions of training samples, this is mostly achieved through parallelization, but the asymptotic complexity of predictions typically remains linear in the number of individual GP models. In online learning problems, this can become problematic since the overhead of parallelization becomes significant when only a single prediction is computed. Moreover, an efficient online construction of the tree computing structure as well as error bounds for the

predictions as required for safety-critical applications have not been investigated.

In contrast to the existing GP approximations using trees as computation graphs, we focus on an iterative construction of the computation tree by growing a random tree with streaming data. In fact, the proposed method is very general and includes many existing methods as special cases, e.g., the k-d [193] and ball-tree constructions [161] can be seen as deterministic cases for batch data. Although the idea of adapting the density and extension of local models to the data distribution has been inherent in aggregation schemes with localized models, most of the proposed approaches require the data in advance for clustering the data points into the leaves, such that they cannot handle streaming data. Moreover, the primary motivation behind localized models in existing methods lies in an improvement of the regression performance. All models, even those far from a test input, are typically evaluated for the overall prediction, which leads to a linear computational complexity in the number of models. We overcome this issue in a principled way by exploiting the tree structure and locality in each layer of the tree to limit the number of individual models which need to be evaluated at a test point. Each model can only be active for prediction in regions in which it also has a positive probability of receiving training samples. This ensures a good prediction performance, while at the same time, active models can be determined very efficiently using recursive tree search algorithms. The graph generated by our approach can be interpreted as a random splitting tree [197], which enables a straightforward formal analysis of the computational complexity of model updates and predictions. Therefore, our proposed method is tailored for online learning of streaming data as needed in event-triggered and time-triggered approaches for the inference of GP models.

5.1.2. Problem Setting

We consider the real-time Gaussian process regression problem, in which the objective is to iterate between updating the GP model $\mu(\cdot)$ of the unknown function $f(\cdot)$ based on sequentially arriving training pairs $(\boldsymbol{z}^{(i)}, y^{(i)})$, $i = 1, \dots, \infty$ and evaluating $\mu(\cdot)$ at arbitrary test points $\boldsymbol{z}$. This setting is encountered, e.g., in the time-triggered learning approach in Section 4.3.2 and when performing event-triggered GP updates as proposed in Section 4.3.3.

Due to the continuous stream of training samples, the data needs to be processed fast. In order to understand the computational bottlenecks of exact GP regression, we reformulate the expressions for the GP mean and variance in (2.26) and (2.27), respectively, into a more efficient form. For this purpose, we define

$$\boldsymbol{L}^{\boldsymbol{K}} = \text{cholesky}(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}), \tag{5.9}$$

$$\boldsymbol{\alpha} = (\boldsymbol{L}^{\boldsymbol{K}})^T \setminus (\boldsymbol{L}^{\boldsymbol{K}} \setminus \boldsymbol{y}), \tag{5.10}$$

where \" denotes the forward and backward substitution, respectively, and $\boldsymbol{K}$ denotes the Gram matrix induced by the kernel and training inputs. From this definition, it is easy to see that $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ operations are required for the computation of $\boldsymbol{L}^{\boldsymbol{K}}$ and $\boldsymbol{\alpha}$ [22], respectively, which can be interpreted as the complexity of inference. The posterior GP distribution $p(f(\boldsymbol{z})|\boldsymbol{z}, \boldsymbol{Z}, \boldsymbol{y}) = \mathcal{N}(\mu(\boldsymbol{z}), \sigma^2(\boldsymbol{z}))$ at a test point $\boldsymbol{z}$ can then be computed using

$$\mu(\boldsymbol{z}) = \boldsymbol{k}(\boldsymbol{z})\boldsymbol{\alpha}, \tag{5.11}$$

$$\sigma^2(\boldsymbol{z}) = k(\boldsymbol{z}, \boldsymbol{z}) - \boldsymbol{v}^T\boldsymbol{v}, \qquad \boldsymbol{v} = \boldsymbol{L}^{\boldsymbol{K}} \setminus \boldsymbol{k}^T(\boldsymbol{z}). \tag{5.12}$$

Therefore, $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ calculations are necessary for posterior mean and variance evaluations, respectively, which can be seen as the complexity of predictions with GP models.

Although it is possible to reduce the complexity of model updates to $\mathcal{O}(N^2)$ when samples are added online to the training data set[1], i.e., $\boldsymbol{Z}_{N+1} = [\boldsymbol{Z}_N \ \boldsymbol{z}^{(N+1)}]$, $\boldsymbol{y}_{N+1} = [\boldsymbol{y}_N^T \ y^{(N+1)}]^T$, by employing rank one updates to obtain $\boldsymbol{L}_{N+1}^{\boldsymbol{K}}$ from $\boldsymbol{L}_N^{\boldsymbol{K}}$ [24], the streaming data quickly accumulates to large data sets in real-time learning problems. Hence, the quadratic complexity in each model update step becomes computationally prohibitive, such that the total number of training samples for straightforward inference is roughly limited to 10^4 training samples in practice on today's machines [157]. While this weakness can be effectively overcome by employing model aggregation schemes of the form (5.1), (5.2) to combine predictions of multiple GP models trained on smaller subsets of the training data [142, 157], such approaches do not deal with the specific difficulties of applying GPs to real-time learning problems. The complexity of computing predictions still grows linearly with the number of individual models, and the assignment of streaming data to individual models is often not investigated [157] or becomes inefficient for large data sets and requires further approximations [142]. Moreover, the existence of probabilistic uniform error bounds for these methods is unclear, such that their usage in real-time learning for safety-critical applications remains a challenge. Therefore, we consider the problem of developing a computationally efficient, aggregation-based online learning method using Gaussian process models, which allows the derivation of uniform error bounds.

5.1.3. Locally Growing Random Trees of Gaussian Processes

In order to address the limitations of existing GP aggregation methods, we develop an efficient alternative for an iterative data distribution to individual models such that predictions are computed based on local data, allowing both updates and predictions with logarithmic complexity. Starting from a single, global model, local models are iteratively generated by dividing existing models. This is efficiently performed by sampling the model, to which each training sample is assigned, from localizing random distributions. Thereby, we locally grow a random tree of GP models.

In detail, we iteratively construct a model by starting with a single data set $\mathbb{D}_1 = \emptyset$. This data set constitutes the root node 1 of a rooted tree T, as depicted in Fig. 5.1. Incoming training data is added to the data set $\mathbb{D}_1$, and each new data point can be efficiently included into the single GP model (5.10) using rank one updates, which exhibit quadratic complexity [24]. When the number of training samples reaches a prescribed threshold $\bar{N}$, we extend the tree T by growing leaf nodes $1, \ldots, K$, $K \in \mathbb{N}_+$ with data sets $\mathbb{D}_2, \ldots, \mathbb{D}_{K+1}$ as children of the root node 1, as shown in the center of Fig. 5.1. In order to distribute the data efficiently to the sets $\mathbb{D}_2$, ..., $\mathbb{D}_{K+1}$, we define a function $\boldsymbol{p}^1 : \mathbb{R}^d \to [0,1]^K$, $\sum_{k=1}^K p_k^1(\boldsymbol{z}) = 1$ for all $\boldsymbol{z} \in \mathbb{R}^d$, which returns the probability of an assignment of a point $\boldsymbol{z} \in \mathbb{R}^d$ to the sets $\mathbb{D}_{k+1}$, $k = 1, \ldots, K$, i.e., $P(\boldsymbol{z} \in \mathbb{D}_{k+1}) = p_k^1(\boldsymbol{z})$. We determine the probabilities $\boldsymbol{p}^1(\boldsymbol{z})$ for each data pair $(\boldsymbol{z}, y)$ in $\mathbb{D}_1$, and sample the child nodes n from the corresponding discrete probability distributions. After the data set division, we compute the local GP models (5.10) for all data sets, which generally has a complexity of $\mathcal{O}(\bar{N}^3)$ [22].

After the initial data set division, we continue to assign the streaming data pairs $(\boldsymbol{z}, y)$

[1] As in Chapter 2, we use a superscript N to indicate the number of training samples a value depends on whenever necessary for clarity.

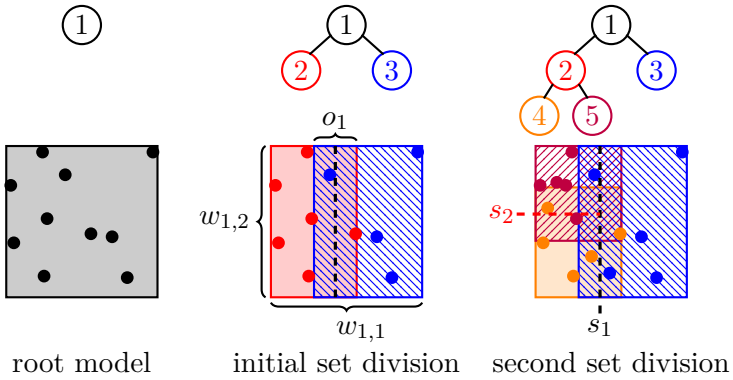

Figure 5.1.: Iterative model tree construction and corresponding layout of the input space for $K = 2$: active regions and training samples belonging to the same node are depicted in the same color.

Algorithm 5.1. Updating of a LoG-GP with K-ary tree T using data $(\boldsymbol{z}, y)$

Start at root node of T
while Current node n is not a leaf **do**
    randomly draw a child node n' from $\boldsymbol{p}^n(\boldsymbol{z})$
    $n \leftarrow n'$
end while
if $|\mathbb{D}_n| = \bar{N}$ **then**
    Generate K children at current node n
    for each $(\boldsymbol{z}', y') \in \mathbb{D}_n$ **do**
        Randomly assign pair $(\boldsymbol{z}', y')$ to a child n' by sampling from $\boldsymbol{p}^n(\boldsymbol{z}')$
        Update local GP model of node n' using (5.10)
    end for
    randomly draw a child node n' from $\boldsymbol{p}^n(\boldsymbol{z})$
    $n \leftarrow n'$
end if
Assign pair $(\boldsymbol{z}, y)$ to current node n
Update local GP model of node n using (5.10)

to the sets $\mathbb{D}_2, \ldots, \mathbb{D}_{K+1}$ by sampling from the discrete distributions with parameters $\boldsymbol{p}^1(\boldsymbol{z})$. When either of the sets $\mathbb{D}_1, \ldots, \mathbb{D}_{K+1}$ reaches its data capacity limit $\bar{N}$, we define a new function $\boldsymbol{p}^{k+1}(\cdot)$, $k = 1, \ldots, K$, such that it induces the conditional probabilities given the parent node, e.g., $P(\boldsymbol{z} \in \mathbb{D}_{K+2} | \boldsymbol{z} \in \mathbb{D}_2) = p_1^2(\boldsymbol{z})$. Based on this conditional probability, we repeat the division process as explained for the root node. Therefore, we add another level to the random tree as shown on the right-hand side of Fig. 5.1. For further training data assignment, it is necessary to iteratively determine a branch of the tree using random transitions based on the discrete distributions with probability parameters $\boldsymbol{p}^n(\boldsymbol{z})$ until a leaf node is reached, as outlined in Algorithm 5.1.

Note that the nodes n, which are not leaves, contain neither data nor a local GP model after the data set division but instead encode the structure of the data distribution to individual data sets using the discrete distributions $\boldsymbol{p}^n(\cdot)$. Therefore, $\boldsymbol{p}^n(\cdot)$ are crucial design choices of the LoG-GP approach. Intuitively, they should be chosen such that the data is dis-

tributed equally to all children in order to generate a balanced tree, and this condition indeed guarantees a logarithmic growth in complexity for the random tree construction as shown in Section 5.1.4. A trivial example of a probability distribution satisfying this requirement is the discrete uniform distribution, i.e., $p_k^n(\boldsymbol{z}) = 1/K$, which can be seen as the sequential version of the commonly used random assignment in batch aggregation methods [161, 157].

Although the random tree construction in Algorithm 5.1 reduces the complexity of updates, it barely affects the complexity of predictions since the direct evaluation of (5.1) and (5.2) with $\mathbb{M}$ denoting the set of leaf nodes still exhibits a linear complexity in the number of training samples. In order to achieve a low complexity of predictions as well, we propose to enforce a small number of active models at each input using the remaining design parameters ω_m. Since a typical condition for these parameters requires their sum to equal one, a straightforward choice is to set them equal to the marginal probabilities $P(\boldsymbol{z} \in \mathbb{D}_m)$ of the leaf nodes, i.e., $\omega_m = P(\boldsymbol{z} \in \mathbb{D}_m)$. The marginal probabilities of a leaf m with depth h^m can be determined by multiplying the conditional probabilities on the branch $\mathbb{A}_m = \{(s_1^m, b_1^m), \ldots, (s_{h^m}^m, b_{h^m}^m)\}$, where $s_1^m = 1$, $b_i^m = 1, \ldots, K$ denotes the child index of the subsequent node and s_i^m denotes the sequence of nodes before reaching leaf m. Therefore, we can express the marginal probability of a leaf as

$$\omega_m = \prod_{i=1}^{h^m} p_{s_i^m}^{b_i^m}(\boldsymbol{z}). \tag{5.13}$$

It is straightforward to see that the computation of a marginal probability requires only local information of nodes along the branch but is independent of other branches. This independence of the branches is the keystone for a reduction of the computational complexity of predictions: a conditional probability $p_{s_i^m}^{b_i^m}(\boldsymbol{z}) = 0$ allows to omit determining the following conditional probabilities $p_{s_j^m}^{b_j^m}(\boldsymbol{z})$, $j > i$, since $\omega_m = 0$ holds regardless of their values. Together with the structure of (5.1), (5.2), this allows to spare the computation of individual GP predictions with a zero conditional probability on the branch, which can be efficiently exploited through recursive tree search algorithms as depicted in Algorithm 5.2.

Due to this property, the conditional probabilities $\boldsymbol{p}^n(\boldsymbol{z})$ effectively control the computational complexity of predictions: when only a few children of every node can have a positive conditional probability $p_k^n(\boldsymbol{z}) > 0$, the recursion can often stop early, and only a few individual GP predictions have to be performed. Thus, the maximum number of active children with $p_k^n(\boldsymbol{z}) > 0$ should be kept small in each node n in order to achieve a low computational complexity. This in turn induces a notion of proximity of points $\boldsymbol{z}$, in which two points $\boldsymbol{z}, \boldsymbol{z}'$ can be considered close to each other if $p_k^n(\boldsymbol{z}) > 0$ and $p_k^n(\boldsymbol{z}') > 0$. Hence, the conditional probabilities can be considered as localizing probability functions.

A simple class of conditional probabilities $\boldsymbol{p}^n(\boldsymbol{z})$ inducing spatial locality are saturating linear functions

$$\xi_k^n(\boldsymbol{z}) = \begin{cases} 0 & \text{if } x_{j_k^n} < s_k^n - \frac{o_k^n}{2} \\ \frac{x_{j_k^n} - s_k^n}{o_k^n} + \frac{1}{2} & \text{if } s_k^n - \frac{o_k^n}{2} \leq x_{j_k^n} \leq s_k^n + \frac{o_k^n}{2} \\ 1 & \text{if } s_k^n + \frac{o_k^n}{2} < x_{j_k^n}, \end{cases} \tag{5.14}$$

where j_k^n defines a splitting dimension, s_k^n denotes the position of the splitting hyperplane, and o_k^n is the overlapping ratio, which determines the size of the region in which two individual

Algorithm 5.2. Prediction recursion for a LoG-GP called for a node n with test input $\boldsymbol{z}$

if Current node n is a leaf **then**
  Compute $\mu_n(\boldsymbol{z})$, $\sigma_n^2(\boldsymbol{z})$ using (5.12), (5.11)
  return $\hat{\boldsymbol{\psi}}_\omega = \boldsymbol{\psi}_\omega(\mu_n(\boldsymbol{z}), \sigma_n^2(\boldsymbol{z}))$, $\hat{\omega} = 1$
else
  $\boldsymbol{\Psi}_\omega \leftarrow [\,]$, $\boldsymbol{\omega} \leftarrow [\,]$
  for all child nodes $n' \in \{i = 1, \ldots, K : p_i^n(\boldsymbol{z}) > 0\}$ **do**
    Determine $\hat{\boldsymbol{\Psi}}_\omega, \hat{\boldsymbol{\omega}}$ by calling the predict recursion for child node n'
    $\boldsymbol{\Psi}_\omega \leftarrow [\boldsymbol{\Psi}_\omega \quad \hat{\boldsymbol{\Psi}}_\omega]$, $\boldsymbol{\omega} \leftarrow [\boldsymbol{\omega} \quad p_j^n(\boldsymbol{z})\hat{\boldsymbol{\omega}}]$
  end for
  if n is root node **then**
    return $\tilde{\mu}(\boldsymbol{z})$, $\tilde{\sigma}^2(\boldsymbol{z})$ based on (5.1), (5.2)
  else
    return $\hat{\boldsymbol{\Psi}}_\omega = \boldsymbol{\Psi}_\omega$, $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}$
  end if
end if

models have a non-zero probability. The interpretation of these parameters is illustrated in Fig. 5.1. Based on (5.14), the conditional probabilities can be defined as

$$p_k^n(\boldsymbol{z}) = \begin{cases} \xi_k^n(\boldsymbol{z}) \prod_{j=1, j\neq k}^{K-1} (1 - \xi_j^n(\boldsymbol{z})) & k < K \\ \prod_{j=1}^{K-1} (1 - \xi_j^n(\boldsymbol{z})) & k = K. \end{cases} \tag{5.15}$$

This parameterization allows straightforward heuristics for choosing the parameters j_k^n, s_k^n, o_k^n of the saturating linear functions $\xi_k^n(\cdot)$, such that the goal of equal division of existing training sets during the updating step as motivated before can be approximately achieved, too. For example, one option to choose j_k^n is the maximum spread of the inputs $\boldsymbol{z}$ in the individual sets $\mathbb{D}_n$, a simple choice for s_k^n is the mean in the dimensions j_k^n, and the overlapping ratio o_k^n can be designed as a constant fraction of the spread. As demonstrated in [198], other computationally efficient approaches also often lead to satisfying results, and a PCA-based definition of the parameters is straightforward as well [199]. Therefore, it is easily possible to achieve the goal of a balanced tree and the desired limitation of the active number of children in each node.

Remark 5.1. *While the proposed approach can be used in combination with other regression techniques as a meta framework, the improvement in computational efficiency can be significantly smaller. However, locally growing random trees have the potential to improve the performance of many regression methods in non-stationary problems, where localization methods have been shown to be useful. Since this problem is not the focus of our work, we leave the combination of the proposed approach with other regression methods for future research.*

5.1.4. Complexity Guarantees

In this section, we formalize the intuitive conditions for achieving low computational complexities discussed in the previous sections. In order to define the meaning of an approxi-

mately equal splitting of data in nodes, we introduce the following assumption, which poses a condition on the relationship between the conditional probabilities $\boldsymbol{p}^n(\cdot)$ and the probability density $q(\boldsymbol{z})$ of the input training data.

Assumption 5.1. *There exists a constant $c_1 \in \mathbb{R}_+$, such that the conditional assignment probabilities $\boldsymbol{p}^n(\boldsymbol{z})$ satisfy*

$$c_1 \leq \int_{\mathbb{R}^d} q(\boldsymbol{z}) \; p_{s_{h_m}^m}^{b_{h_m}^m}(\boldsymbol{z}) \; H\left(\prod_{i=1}^{h^m-1} p_{s_i^m}^{b_i^m}(\boldsymbol{z})\right) \mathrm{d}\boldsymbol{z} \tag{5.16}$$

for all leaves $m \in \mathbb{M}$ with depths h^m and branches $\mathbb{A}_m$, where $H : \mathbb{R} \to \{0, 1\}$ denotes the unit step function.

The right-hand side of (5.16) corresponds to the marginal conditional probability that a training sample is assigned to leaf m, given the prior assignment to node $s_{h_m}^m$. Therefore, this assumption prevents nodes from never receiving training data. In practice, this can easily be achieved through a data-based design of the conditional probabilities as outlined in Section 5.1.3.

Based on Assumption 5.1, it is straightforward to analyze the complexity of LoG-GP updates using the theory of random split trees [197]. A random split tree T is defined through the parameters K, $\bar{N}$, s_0, s_1, $\boldsymbol{p}$ and N. The parameter N describes the number of balls in the tree, while $\bar{N}$ denotes the maximum number of balls in a node of the tree. The number of children of each node is given by K. Each internal (non-leaf) node has s_0 balls, while each leaf node has at least s_1 balls. The split probability is described by $\boldsymbol{p}$. The distribution of balls is done iteratively. Starting at the tree, a ball is assigned to a child by drawing from the random distribution $\boldsymbol{p}$ until a leaf node is reached. If this leaf has already reached its capacity, then the tree is extended, and s_1 balls are assigned to each child. The remaining $\bar{N}+1-Ks_1-s_0$ balls are finally assigned according to the random distribution $\boldsymbol{p}$.

It can be clearly seen that the tree construction of LoG-GPs is identical to that of a random split tree with $s_0 = 0$, $s_1 = 0$ and input dependent $\boldsymbol{p}^n(\boldsymbol{z})$. Due to Assumption 5.1, this allows us to bound the height of the tree in LoG-GPs by the height of a random split tree with $c_1 \leq p_i \leq 1 - Kc_1$ for all $i = 1, \ldots, K$. This is exploited to bound the complexity of updates in LoG-GPs in the following theorem.

Theorem 5.1. *The update of a LoG-GP with conditional assignment probabilities $\boldsymbol{p}^n(\cdot)$ satisfying Assumption 5.1 requires $\mathcal{O}_p(\log(N))$ computations.*

Proof. The tree of LoG-GPs is a random split tree in the sense of [197]. Assumption 5.1 ensures that in any split of the tree, data is distributed approximately equally to both sides in the sense that no child gets all the data almost surely. Hence, it follows from [197, Theorem 1] that the height of the tree, i.e., the maximum depth of any leaf, grows logarithmically in probability. Moreover, it is trivial to see that the updating complexity of LoG-GPs depends linearly on the height of the tree, which proves the result. □

In order to bound the complexity of predictions using LoG-GPs as well, an additional assumption on the maximum number of children with positive conditional probabilities along a branch is necessary. This is formalized as follows.

Assumption 5.2. *There exist constants $c_2, c_3 \in \mathbb{R}_+$, such that the conditional assignment probabilities $\boldsymbol{p}^n(\cdot)$ satisfy*

$$\sum_{i=1}^{h^m} H\left(1 - p_{s_i^m}^{b_i^m}(\boldsymbol{z})\right) H\left(p_{s_i^m}^{b_i^m}(\boldsymbol{z})\right) \leq \log(c_2 h^m + c_3) \tag{5.17}$$

for all leaves $m \in \mathbb{M}$ with depths h^m and branches $\mathbb{A}_m$.

Since this condition can individually be checked for every branch during the generation of a new layer, it can directly be included in the specification of the conditional probabilities $\boldsymbol{p}^n(\cdot)$ during the generation of a new layer, e.g., through the adaptation of the overlapping ratio o_m in (5.14). Therefore, this assumption is not restrictive in practice. In combination with Assumption 5.1, it allows the following bound on the computational complexity of predictions.

Theorem 5.2. *Mean and variance predictions of LoG-GPs with conditional assignment probabilities $\boldsymbol{p}^n(\cdot)$ satisfying Assumptions 5.1 and 5.2 require $\mathcal{O}_p(\log^2(N))$ computations.*

Proof. It is trivial to see that the prediction of a single branch requires $\mathcal{O}(h^m)$ operations, such that Theorem 5.1 guarantees a complexity of $\mathcal{O}_p(\log(N))$ for a single branch. Moreover, the number of leaves m in the tree of a LoG-GP depends linearly on the number of training samples N, i.e., $|\mathbb{M}| \in \mathcal{O}(N)$. Therefore, we have to show that only $\mathcal{O}_p(\log(N))$ leaves m have a positive marginal probability ω_m and must be evaluated. It immediately follows from Assumption 5.2 that no more than $K^{\log(c_2 h^m + c_3)}$ leaves can be active, which implies that the number of active leaves behaves as $\mathcal{O}(h^m)$. Therefore, a prediction requires $\mathcal{O}((h^m)^2)$, which concludes the proof using Theorem 5.1. □

Remark 5.2. *Although the maximum number of samples $\bar{N}$ has a strong impact on the computation time, it merely acts as a constant factor on the asymptotic complexities. Therefore, we drop it for clarity of presentation.*

5.1.5. Uniform Regression Error Bounds

Since LoG-GP predictions rely on the aggregation of exact GP mean functions, it is straightforward to extend the probabilistic uniform prediction error bounds derived in Section 2.2 and Section 2.3. For this extension, we require the following assumption on the admissible aggregation structures.

Assumption 5.3. *The distributed GP mean can be expressed as*

$$\tilde{\mu}(\boldsymbol{z}) = \sum_{m \in \mathbb{M}} w_m(\boldsymbol{z}) \mu_m(\boldsymbol{z}), \tag{5.18}$$

where $w_i : \mathbb{R}^d \to \mathbb{R}_{0,+}$ is a weighting function satisfying $\sum_{m \in \mathbb{M}} w_m(\boldsymbol{z}) = 1$, $\forall \boldsymbol{z} \in \mathbb{R}^d$.

It can be trivially checked that both the mixture of experts (5.3), (5.4) and the generalized product of experts approach (5.6), (5.6) in combination with weights ω_m following from the LoG-GP approach (5.13) satisfy this condition. Hence, this assumption does not severely restrict the class of possible approaches for aggregating GP predictions but allows to sum up the individual uniform error bounds for the mean functions $\mu_m(\cdot)$, which is the core idea in the following theorem.

Theorem 5.3. *Consider an unknown function $f(\cdot)$, a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ satisfying Assumption 2.6, and training data sets $\mathbb{D}_m$ with observation noise $\epsilon^{(n)}$, $n = 1,\dots,N$, such that Assumption 2.1 individually holds for all data sets $\mathbb{D}_m$. If Assumptions 2.7 and 5.3 hold, then, for every $\delta \in (0,1)$ and $\tau \in \mathbb{R}_+$, the aggregated mean function $\tilde{\mu}(\cdot)$ defined in (5.1) admits a uniform error bound*

$$\eta(\boldsymbol{z}) = \sum_{m\in\mathbb{M}} w_m(\boldsymbol{z}) \left(\tilde{\beta}\sigma_m(\boldsymbol{z}) + L_f\tau^{p_f} + L_{\mu_m}\tau^{p_\mu} + \tilde{\beta}L_{\sigma_m}\tau^{p_\sigma}\right) \tag{5.19}$$

with probability $1-\delta$ on the compact set $\mathbb{S} \subset \mathbb{R}^{d_z}$, where

$$\tilde{\beta} = \sqrt{2\log\left(\frac{M(\tau,\mathbb{S})|\mathbb{M}|}{\delta}\right)}. \tag{5.20}$$

Proof. Since the assumptions of Proposition 2.2 are satisfied, it holds with probability of at least $1 - {}^{\delta}/_{|\mathbb{M}|}$ for each individual model that

$$|f(\boldsymbol{z}) - \mu_m(\boldsymbol{z})| \leq \tilde{\beta}\sigma_m(\boldsymbol{z}) + L_f\tau^{p_f} + L_{\mu_m}\tau^{p_\mu} + \tilde{\beta}L_{\sigma_m}\tau^{p_\sigma} \tag{5.21}$$

with $\tilde{\beta}$ defined in (5.20). Moreover, we have

$$\begin{aligned} |f(\boldsymbol{z}) - \tilde{\mu}(\boldsymbol{z})| &= \left| f(\boldsymbol{z}) - \sum_{m\in\mathbb{M}} w_m(\boldsymbol{z})\mu_m(\boldsymbol{z}) \right| && (5.22)\\ &= \left| \sum_{m\in\mathbb{M}} w_m(\boldsymbol{z})(f(\boldsymbol{z}) - \mu_m(\boldsymbol{z})) \right| && (5.23)\\ &\leq \sum_{m\in\mathbb{M}} w_m(\boldsymbol{z})|f(\boldsymbol{z}) - \mu_m(\boldsymbol{z})|, && (5.24) \end{aligned}$$

where the second line follows from Assumption 5.3 and the third line follows from the triangle inequality. Finally, the result follows from the union bound, such that (5.21) holds jointly for all $m \in \mathbb{M}$, which concludes the proof. □

This theorem straightforwardly extends Proposition 2.2 to LoG-GP models requiring no additional conditions except for Assumption 5.3. Due to the assumed structure of the admissible aggregations, (5.19) effectively constitutes the sum of individual GP error bounds, such that we need to consider the Hölder coefficients and the standard deviations of individual GP models. This structure can also be obtained for RKHS-based uniform error bounds, as shown in the following theorem.

Theorem 5.4. *Consider an unknown function $f(\cdot) \in \mathbb{H}_k^{\mathbb{S}}$ satisfying Assumption 2.4 and a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$. Moreover, assume that the observation noise of each data set $\mathbb{D}_m$ individually satisfies Assumption 2.5. Then, for every $\delta \in (0,1)$, the posterior mean function $\mu(\cdot)$ defined in (5.1) using an aggregation scheme satisfying Assumption 5.3 admits a uniform error bound*

$$\eta(\boldsymbol{z}) = \beta \sum_{m\in\mathbb{M}} w_m(\boldsymbol{z})\sigma_m(\boldsymbol{z}) \tag{5.25}$$

with probability $1-\delta$ on the compact set $\mathbb{S} \subset \mathbb{R}^{d_z}$, where

$$\beta = \Gamma + \frac{\sqrt{2}\tilde{\sigma}_{\text{on}}}{\sigma_{\text{on}}} \sqrt{\frac{1}{2} + \log\left(\sqrt{\det\left(\boldsymbol{I}_N + \frac{1}{\sigma_{\text{on}}^2}\boldsymbol{K}\right)}\right) + \log\left(\frac{|\mathbb{M}|}{\delta}\right)}. \tag{5.26}$$

Proof. Since the assumptions of Theorem 2.1 are satisfied, we have for each local model that

$$|f(\boldsymbol{z}) - \mu_m(\boldsymbol{z})| \leq \beta \sigma_m(\boldsymbol{z}) \tag{5.27}$$

with probability of at least $1 - \delta/|\mathbb{M}|$ for β defined in (2.56). By exploiting (5.24) and applying the union bound, we finally obtain the result analogously to the proof of Theorem 5.3. □

These theorems demonstrate that extensions of the results in Sections 2.2 and 2.3 can be straightforwardly obtained for LoG-GPs. Due to the structural equivalence of the uniform error bounds for exact GP models and LoG-GPs, this allows the direct application of LoG-GPs while retaining theoretical guarantees proven in the previous chapters.

5.1.6. Evaluation on Real-World Data

In order to demonstrate the computational efficiency and the prediction performance of LoG-GPs[2], we compare them to several state-of-the-art GP approximations for online learning and evaluate the performance on three real-world data sets. As an example for a relatively small data set, we employ the SARCOS data [22] containing 44484 samples of the dynamics of a robotic manipulator ($d = 21$), a widely used example for comparing GP approximations. Moreover, we employ the buzz in social media data set [200], which consists of 583250 samples with $d = 77$ features, and the individual household electric power consumption data set [201] composed of 2048380 measurements with $d = 11$. The data is preprocessed following [202].

The LoG-GP is evaluated using mixture of experts (MoE), generalized product of experts (gPoE) and robust Bayesian committee machine (rBCM) aggregations and conditional probabilities (5.15). We compare to incremental sparse spectrum Gaussian processes (ISSGP) [51], local Gaussian processes [142], streaming sparse GPs (SSGP) [170], and the robust Bayesian committee machine [157] using the implementation of [196]. The following parameters are used in these methods:

- LoG-GP: each node n has $K = 2$ children. The probabilities $\boldsymbol{p}^n(\cdot)$ are defined through (5.14), (5.15), where j_k^n is the dimension of the maximum spread of the local data set, s_k^n is the mean of the data in this dimension and

$$o_k = \frac{\max_{\boldsymbol{z},\boldsymbol{z}' \in \mathbb{D}_k} \|\boldsymbol{z} - \boldsymbol{z}'\|}{100(\frac{h^n}{10} + 1)} \tag{5.28}$$

  with h^n being the height of node n. Moreover, each local model can contain a maximum of $\bar{N} = 100$ training samples. For the MoE and gPoE aggregation schemes, we use (5.13), while we define

$$\omega_m = \frac{\log(k(\boldsymbol{z},\boldsymbol{z})) - \log(\sigma_m^2(\boldsymbol{z}))}{2} \prod_{i=1}^{h^m} p_{s_i^m}^{b_i^m}(\boldsymbol{z}). \tag{5.29}$$

  for the rBCM aggregation, where the first factor is the differential entropy between the prior and the posterior distribution as proposed in [157].

- ISSGP: as suggested in [51], we use $D = 200$ random Fourier features. Hence, the kernel matrix has a size of 400×400.

[2]Matlab and Python code is online available at `https://gitlab.lrz.de/online-GPs/LoG-GPs`.

- local GPs[3]: the maximum number of training samples of a local model is set to $\bar{N} = 100$. Moreover, a threshold of 0.9 is used to determine if a new model is generated.
- SSGP[4]: the method is used with 100 inducing points, but no online optimization of hyperparameters and inducing points to reduce computational complexity. Since the method does not allow iterative updates, we update it using mini-batches of size 300 as proposed in the original publication [170]. This means that we determine the prediction error on 300 training samples before we update the model using these data samples. This method is implemented in Python.
- rBCM[5]: the maximum number of samples in a local model is set to $\bar{N} = 100$. The data is randomly distributed to the local models. Since the method does not allow iterative updates, we recompute the model each time after observing 1000 new samples using the data observed up to this time.

All GPs use an ARD squared exponential kernel, and the hyperparameters are fitted using the first 1000 training samples for all methods in order to ensure that poor hyperparameters are excluded throughout all simulations. This is done using the well-documented MATLAB internal routine whenever possible. For the Python implementation of SSGPs, hyperparameter optimization is based on the GPflow toolbox[6] [203]. The computations for all methods are performed on a single computation unit. Note that the evaluation of the local models of LoG-GPs can be parallelized similarly as proposed by [157] by distributing the active models to multiple computation units. The data used for hyperparameter optimization is added to the GP approximations before they are evaluated in a sequential setting, in which we iterate between prediction for an input $\boldsymbol{z}^{(i)}$ and update of a model using $(\boldsymbol{z}^{(i)}, y^{(i)})$. All simulations are executed on a cluster computer with Intel(R) Core(TM) i9-9900X CPU and 128GB DDR4 RAM. The code is run using MATLAB R2019a when not stated differently.

As performance metrics, we use the average prediction and update times. Moreover, we evaluate the regression performance based on the standardized mean squared error (SMSE) and the mean standardized log loss (MSLL) [22] in a sequential interpretation, e.g.,

$$\mathrm{SMSE}_k = \frac{\sum_{i=1}^{k} (\tilde{\mu}_{k-1}(\boldsymbol{z}^{(k)}) - y^{(k)})^2}{k s_y^2}, \tag{5.30}$$

where s_y^2 denotes the empirical variance of the targets $y^{(i)}$ and $\tilde{\mu}_{k-1}(\boldsymbol{z}^{(k)})$ the prediction after observing $k-1$ training samples.

The resulting computation times averaged over 20 runs are depicted in Fig. 5.2. These simulation results, which are additionally summarized in Table 5.1, show that computation times of LoG-GPs are more strongly varying compared to existing methods, which is a consequence of the continuously changing size of the local models. Moreover, the logarithmic growth of the computation time is not visible since the overlapping ratio o_k was chosen small, such that the average number of active models is almost constant as illustrated in Fig. 5.3 for the MoE-LoG-GP trained on the SARCOS data set. Since the depth of the tree is practically

[3]The code is available at `https://www.ias.informatik.tu-darmstadt.de/Miscellaneous/Miscellaneous`.
[4]The code is available at `https://github.com/thangbui/streaming_sparse_gp`.
[5]The code is available at `https://github.com/LiuHaiTao01/GRBCM`.
[6]The code is available at `https://github.com/GPflow/GPflow`.

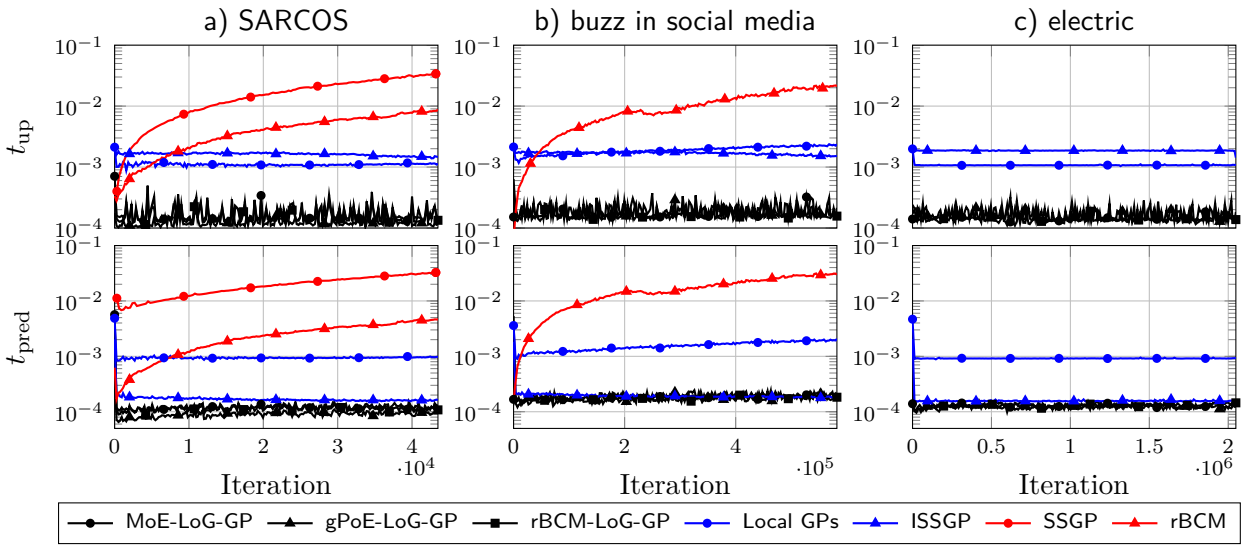

Figure 5.2.: Plots of average update time t_{up} (top) and the average prediction time t_{pred} (bottom) on a) SARCOS b) buzz in social media and c) electric data sets. Due to the high computation times, the SSGP could only be applied to the SARCOS, while the rBCM could not be evaluated on the electric data set. Computation times of LoG-GP approaches are more noisy than those of existing methods due to the strongly varying size of local models. However, they are generally smaller, in particular for computing model updates.

not relevant, almost constant computation times can be achieved with LoG-GPs, which is in strong contrast to batch methods such as SSGPs or the rBCM leading to a quickly growing complexity. While state-of-the-art real-time learning methods such as ISSGPs can yield a similar prediction rate, LoG-GP approaches significantly outperform all existing methods in terms of model update times. Therefore, these simulation results clearly demonstrate the computational advantages of LoG-GP approaches.

Remark 5.3. *Note that the difference in the used programming languages has an effect on the absolute computation time but not on the behavior. While the computation time for updates of LoG-GPs in a Python implementation increases to approximately* 5 ms *on the SARCOS data set, it does practically not change with a growing number of training samples. Therefore, LoG-GPs remain advantageous compared to SSGP regardless of the employed programming language.*

Table 5.2 displays the standardized mean square error and mean standardized log loss averaged over 20 simulation runs together with the corresponding standard deviations, whose evolution over the number of training samples is illustrated in Fig. 5.4. These results clearly show that LoG-GP approaches outperform state-of-the-art online learning methods regarding the overall prediction error for data sets with medium and high dimensional input domains such as the SARCOS and buzz in social media data. For low dimensional input domains such as in the house electric data set, ISSGPs provide slightly better performance. The weaker performance of ISSGPs for high dimensional input domains is a direct consequence of the strong dependence of the kernel approximation error on the input dimension as discussed in Section 5.1.1.

Table 5.1.: Average update and prediction times in ms with the corresponding standard deviation in brackets for the SARCOS, buzz in social media and electric data sets. LoG-GP methods outperform state-of-the-art methods for streaming data regarding the computation time: updates are up to 10 times faster, and they achieve state-of-the-art prediction rates. SSGP and rBCM are greyed out as they allow only batch updates.

COMPUTATION TIME (ms)	SARCOS		BUZZ		ELECTRIC	
	t_{pred}	t_{up}	t_{pred}	t_{up}	t_{pred}	t_{up}
MoE-LoG-GP	**0.12 (0.09)**	0.17 (0.21)	0.19 (0.09)	0.18 (0.20)	**0.12 (0.05)**	**0.15 (0.15)**
gPoE-LoG-GP	**0.12 (0.08)**	**0.16 (0.21)**	0.19 (0.10)	0.18 (0.20)	0.13 (0.50)	0.16 (0.16)
rBCM-LoG-GP	**0.12 (0.08)**	**0.16 (0.20)**	**0.17 (0.08)**	**0.17 (0.15)**	0.13 (0.05)	0.16 (0.15)
ISSGP	0.17 (0.03)	1.6 (0.27)	0.19 (0.03)	1.7 (0.28)	0.16 (0.02)	1.9 (0.07)
LOCAL GPs	0.94 (0.11)	1.1 (0.18)	1.5 (0.66)	1.9 (0.67)	1.1 (0.03)	0.91 (0.03)
SSGP	19 (7.48)	16 (9.69)	> 20	> 20	> 20	> 20
rBCM	2.5 (1.4)	4.3 (2.6)	17 (9.1)	10 (6.6)	> 20	> 10

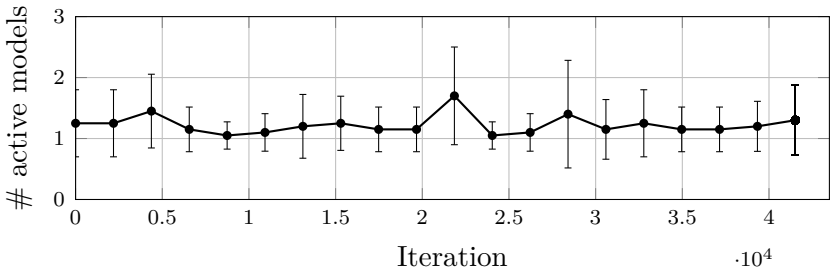

Figure 5.3.: The low computation times for predictions are achieved by a low number of active models, which is on average less than 2 for the MoE-LoG-GP on the SARCOS data set. The comparatively high standard deviation of the number of active models illustrated by the error bars contributes to noisy prediction times of LoG-GPs. The observed maximum number of active models is 12 on the SARCOS data set.

The poor performance of local GPs, which is significantly worse than originally presented by [142], is a result of not tuning the threshold for generating new models for each data set. While tuning this parameter could improve the performance, this would conflict with the principle of online learning: for tuning the parameter, a significant amount of training data is necessary, while the online learning paradigm assumes little or even no data in advance. Moreover, tuning must be done by hand, which is time-consuming. Therefore, we choose the value 0.9 for the threshold empirically such that many local models are generated, yet not too many to keep the computation time tractable.

Table 5.2 shows that LoG-GP approaches result in the best MSLL values, merely marginally outperformed by ISSGPs for the first half of the electric data set as illustrated in Fig. 5.4. This emphasizes the high reliability of the epistemic uncertainty estimate provided by LoG-GP approaches, which is crucial in safety-critical applications due to the dependence of uniform error bounds on the posterior standard deviations. Therefore, even a small improvement over existing methods is highly beneficial in practice.

While it might be surprising that rBCMs provide poor predictive distributions as indicated by the high MSLL values, this effect has already been observed in [196], where it is

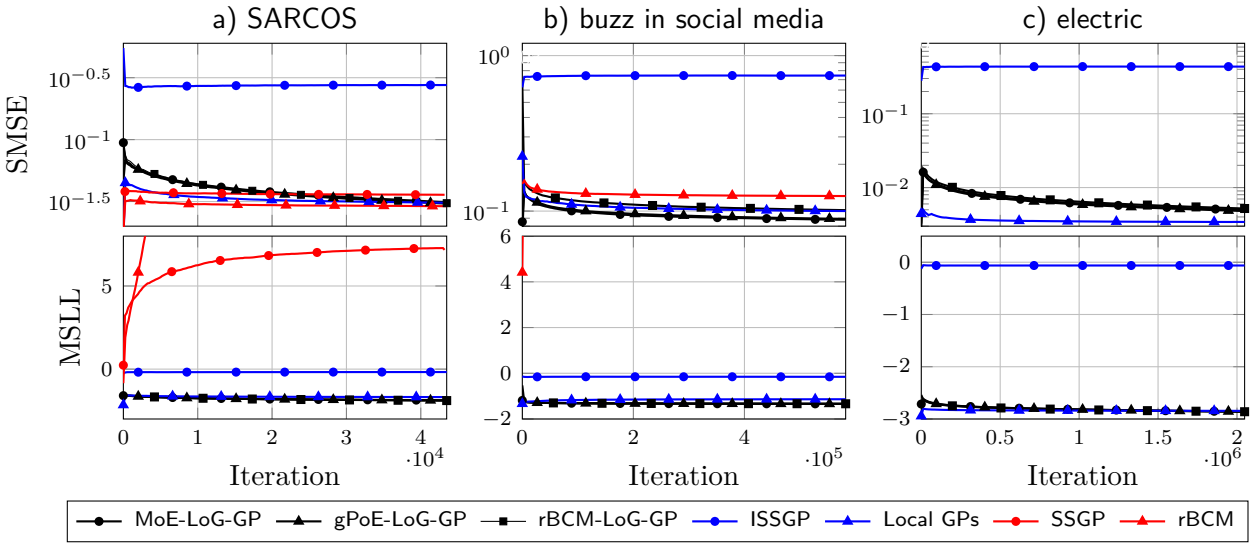

Figure 5.4.: Plots of the SMSE (top) and MSLL (bottom) on a) SARCOS, b) buzz in social media, and c) electric data sets. Due to the high computation times, the SSGP could only be applied to the SARCOS, while the rBCM could not be evaluated on the electric data set. LoG-GP approaches achieve at least a comparable performance to existing methods with slight advantages in high dimensional problems.

Table 5.2.: Average standardized mean squared error and average mean standardized log loss with the corresponding deviation in brackets for the SARCOS, buzz in social media, and electric data sets. LoG-GP approaches show advantages in problems with medium and high dimensional input domains, while ISSGPs exhibit advantageous performance on low dimensional problems.

	SARCOS		BUZZ		ELECTRIC	
	SMSE ($\cdot 10^{-3}$)	MSLL	SMSE ($\cdot 10^{-3}$)	MSLL	SMSE ($\cdot 10^{-3}$)	MSLL
MoE-LoG-GP	31.3 (0.85)	−1.87 (0.02)	**88.0 (12.9)**	**−1.34 (0.02)**	5.0 (0.56)	**−2.86 (0.02)**
gPoE-LoG-GP	**30.3 (0.75)**	−1.88 (0.02)	89.2 (10.4)	**−1.34 (0.02)**	4.8 (0.60)	−2.86 (0.03)
rBCM-LoG-GP	30.7 (0.85)	**−1.89 (0.02)**	101.4 (27.0)	−1.33 (0.04)	5.2 (0.44)	−2.86 (0.03)
ISSGP	30.9 (1.4)	−1.68 (0.05)	100.1 (16.0)	−1.14 (0.11)	**3.4 (0.24)**	−2.84 (0.03)
Local GPs	276.0 (64.5)	−0.18 (0.07)	744.0 (35.9)	−0.15 (0.05)	450.9 (87.9)	−0.03 (0.15)
SSGP	35.9 (2.4)	7.17 (2.61)	—	—	—	—
rBCM	29.0 (1.3)	78.2 (10.8)	124.9 (42.4)	375 (11.7)	—	—

shown that the rBCM asymptotically becomes overconfident. Similarly, the relatively weak performance of the SSGP can be explained by the slight differences in the simulation setup we use. While the inducing points and hyperparameters are updated in every mini-batch in the original publication [170], we refrain from doing so in order to achieve the lowest possible computation times.

Note that the additional online adaptation of the hyperparameters with SSGPs does not change the general observation of a beneficial regression performance of LoG-GP compared to SSGPs as demonstrated in Fig. 5.5. For a fair comparison, we perform a single gradient step for log-likelihood maximization in LoG-GPs after a data point is added, and the performance is compared using a pre-training of hyperparameters with 1000 and 100 data points. It can be clearly seen that for 1000 pre-training samples, the MSLL of SS-

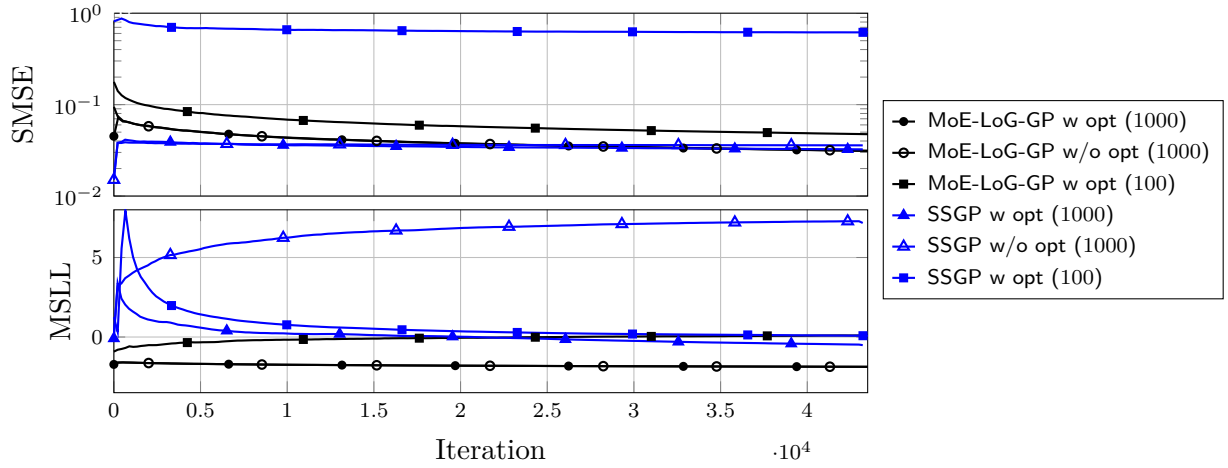

Figure 5.5.: Online hyperparameter optimization has a weak impact on the SMSE. The MSLL of the SSGP improves through hyperparameter optimization, but it remains significantly worse than the MSLL of LoG-GP approaches. When the number of pre-training samples is reduced to 100, both approaches suffer from a deteriorated prediction performance.

GPs benefits enormously from the online hyperparameter optimization, while the SMSE is barely affected. The regression accuracy and the quality of the predictive distributions of LoG-GPs remain almost unchanged. Moreover, both methods exhibit an inferior regression performance when the number of pre-training samples is reduced to 100 despite the online hyperparameter training. Overall, LoG-GPs still yield lower SMSE and MSLL values with online hyperparameter optimization, but the gap between SSGPs and LoG-GPs becomes smaller. Therefore, the advantageous regression performance of LoG-GPs is not limited to scenarios with fixed hyperparameters.

Remark 5.4. *Since LoG-GPs have the same principled structure as distributed GPs proposed by [157], the hyperparameter optimization approach can be employed to optimize the hyperparameters of LoG-GPs. If a separate process is spawned to perform the required optimization after batches of data while data is continuously added to the LoG-GP, existing hyperparameter optimization approaches can be executed online without a crucial impact on the computational complexity. This approach allows a batch adaptation of hyperparameters similar to SSGPs.*

5.1.7. Application to Event-Triggered Learning Control

In order to demonstrate the straightforward applicability of LoG-GPs for control with accuracy guarantees, we employ them in the event-triggered online learning approach for compensating an unknown nonlinearity presented in Section 4.3.3. For this purpose, we consider once more the example setting in Section 3.1.4, such that we have a linear closed-loop system described by

$$\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \begin{bmatrix} 0 & 1 \\ -k_c\tilde{\theta} & -k_c - \tilde{\theta} \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad \text{(4.80 revisited)}$$

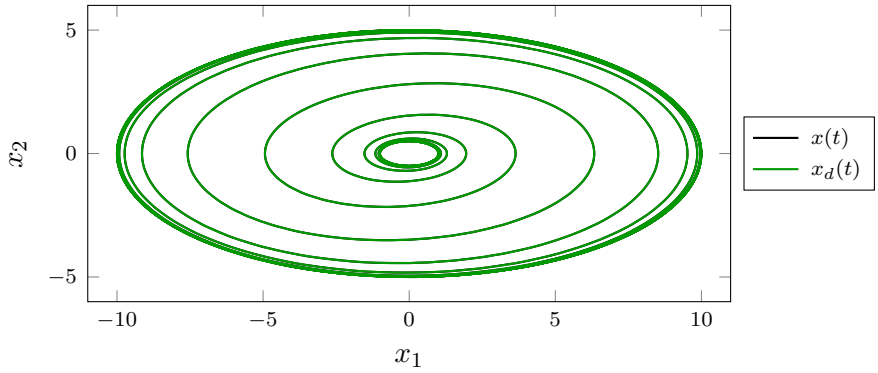

Figure 5.6.: The system properly tracks the desired outwards spiral trajectory after an initial transient phase.

which is perturbed by the nonlinear input perturbation $f(\boldsymbol{x}) = 1 - \sin(2x_1) + {}^{1}/_{(1+\exp(-x_2))}$. The control gains are chosen as $k_c = 10$ and $\tilde{\theta} = 1$ and define the reference trajectory $\boldsymbol{x}_{\mathrm{ref}}(\cdot)$ as an outwards spiral by setting

$$x_{\mathrm{ref},1}(t) = \left(1 + \frac{9}{1 + \exp(-0.1(t - 100))}\right) \sin(0.5t), \tag{5.31}$$

$$x_{\mathrm{ref},2}(t) = \dot{x}_{\mathrm{ref},1}(t), \tag{5.32}$$

$$r = \dot{x}_{\mathrm{ref},2}(t). \tag{5.33}$$

For online learning, we employ a MoE-LoG-GP with $\bar{N} = 100$, $K = 2$ and conditional probabilities $\boldsymbol{p}^n(\cdot)$ defined through (5.15). In the local GP models, an ARD squared exponential kernel is used, whose hyperparameters are fixed and manually chosen as $\sigma_f = 5$ and $l_1 = l_2 = 0.5$. The observation noise standard deviation is set to $\sigma_{\mathrm{on}} = 10^{-3}$.

Since real-world continuous-time control systems are run on computers with finite processing power, controllers must be applied in a sampled-data sense in practice. Our simulation reflects this by running the control loop with 1kHz rate and using zero-order hold digital to analog conversion. The triggering condition (4.92) with $\bar{e} = 7.5 \cdot 10^{-3}$ is sampled with the same rate, such that the event-triggered model updates effectively cause a time-triggered learning scheme in the worst case of permanent triggering. Therefore, data efficiency is ensured whenever the desired tracking error bound $\bar{e}$ is satisfied. Moreover, a probabilistically bounded tracking error directly follows from Theorem 4.4 and, e.g., Theorem 5.3, as well as the continuity of the closed-loop dynamics with a LoG-GP model. Note, however, that this tracking error bound can generally be larger than $\bar{e}$ used in the trigger condition (4.92), even though the difference can be made arbitrarily small for $\bar{e}$ satisfying the conditions of Theorem 4.5 by increasing the rate of the control loop. Hence, this simple approach combines the theoretical advantages of event-triggered and time-triggered online learning in a practical method.

To demonstrate the computational advantages of LoG-GPs in such an application, we compare it to an exact GP model as used in Section 4.3.5. The resulting trajectory for learning with a LoG-GP model during the time interval $t \in [0\mathrm{s}, 200\mathrm{s}]$ is illustrated in Fig. 5.6, and it can be seen that it closely follows the reference trajectory. This is a result of a frequent triggering of the learning event at the beginning until $\approx 12\mathrm{s}$ as illustrated in Fig. 5.7. At this time, the system state has converged to the reference trajectory, and sufficient data of the

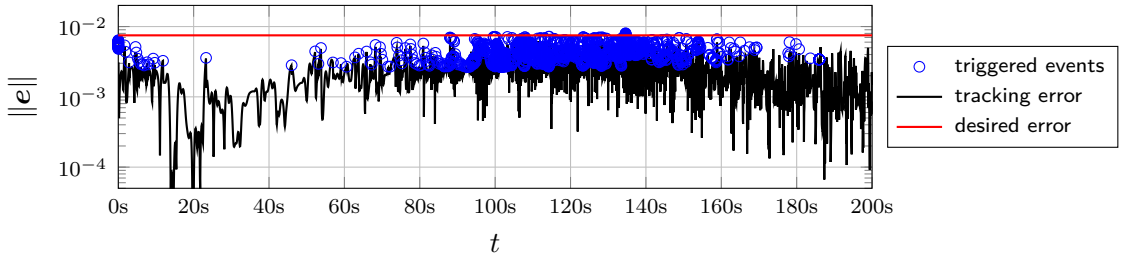

Figure 5.7.: Update events are frequently triggered at the beginning and when changing the radius of the reference trajectory in order to ensure the desired tracking error $\bar{e}$. Once a stationary behavior has been reached, very few events are triggered.

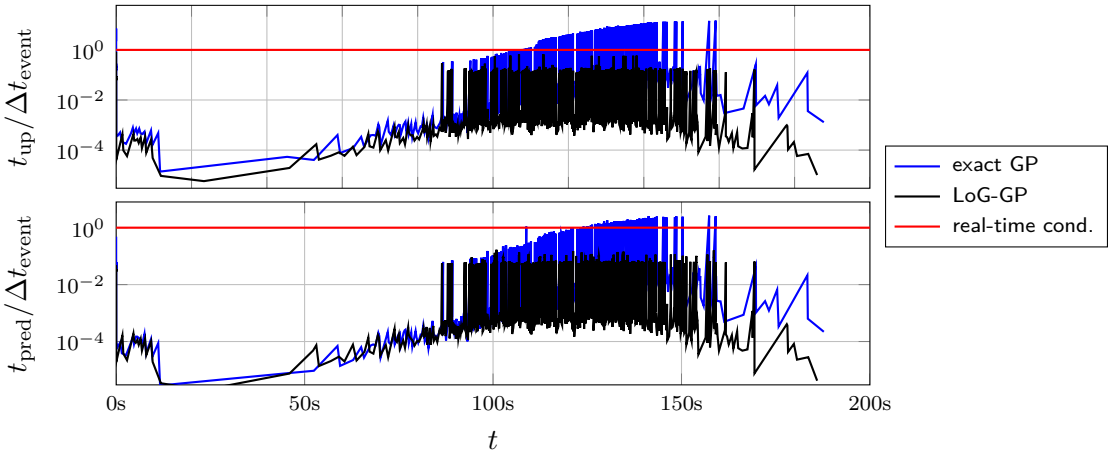

Figure 5.8.: The ratio between training/prediction and inter-event time keeps growing for the exact GP and eventually exceeds the real-time threshold 1, while it stagnates after some time for the LoG-GP despite a slight growth in the added training samples (951 vs. 1662).

inner circular motion has been collected. Afterward, barely any new data is sampled until the radius of the reference starts to increase at $\approx$ 45s. Then, the model is again frequently updated until the learning has finished after 185s. Most importantly, it can be observed that the stationary tracking error is upper bounded by $8 \cdot 10^{-3}$, while most of the time, the desired error bound $\bar{e}$ used for defining the triggering condition (4.92) is satisfied.

While the exact GP only requires 951 training samples in this approach, the update event is triggered 1662 times for the MoE-LoG-GP. However, this slight reduction in data efficiency is necessary in order to meet the real-time constraints as depicted in Fig. 5.8. In contrast to the exact GP model, where the prediction and update times t_{pred} and t_{up} start to exceed the inter-event time Δt_{event} after $\approx$ 125s and $\approx$ 110s, respectively, LoG-GPs remain fast enough to satisfy this condition. Therefore, LoG-GPs can enable the straightforward application of Gaussian processes in control problems with performance guarantees under real-time computation constraints.

5.2. Data-Efficient Learning for Cooperative Control of Multi-Agent Systems

While the LoG-GP approach exploits the aggregation of local models for GP-based online learning with low computational complexity, it is limited to tree-structured computation architectures. However, in many large-scale systems ranging from energy networks [204] over vehicle platooning [205] to swarms of autonomous robots [206], more general computation graphs are required to deal with available communication topologies.

Several learning techniques based on GP regression have been proposed in recent years to enable distributed learning in such multi-agent systems. Addressing the issues faced when GPs are scaled to large data sets, early approaches only distribute the inference in local models to multiple agents but employ a central coordinator for their aggregation [157]. This limitation is overcome by computing various summary statistics for the individual GPs, which can be combined using consensus protocols [207, 208]. Through an online optimization of hyperparameters and summary statistics, this can even be extended for implementing online learning [143]. While these distributed GP approaches have shown strong empirical performances, probabilistic uniform error bounds similar to those for exact GP regression have not been derived.

Therefore, independent GP models in each agent are commonly used for the control of multi-agent systems. For example, a distributed model predictive control method, which allows to cooperatively solve optimal control problems with Gaussian process models as dynamics, is developed in [209]. In [210], GP models are employed to design control laws for formation control, while a similar approach is used for flocking control in [211]. Even though these approaches follow cooperative control principles, they employ merely local data of each agent [209, 210, 211], such that an agent's performance crucially depends on its local data set. While this limitation can be mitigated by aggregating predictions with direct neighbors [212], such an approach does not transfer the cooperative control paradigm to model learning problems. Therefore, model knowledge is generally not disseminated through the communication network when control performance guarantees are required, such that existing approaches for GP-based learning for cooperative control are generally data-inefficient.

In this section, we address this weakness of current methods by proposing distributed learning-based feedback linearizing tracking control laws for leader-follower consensus of multi-agent systems. We focus here on systems with a single control input for simplicity of exposition, but the extension to systems with multiple control inputs is straightforward. To this end, we first present a novel, fully distributed GP approach for predicting unknown dynamics, which straightforwardly extends centralized approaches of the form (5.1) to arbitrary computation graphs by employing dynamic average consensus algorithms [213]. Based on the probabilistic uniform error bounds for GP regression presented in Chapter 2, we derive explicit, distributed regression error bounds for the proposed method, which are, to the best of our knowledge, the first of their kind for fully distributed GP predictions. We then make use of the proposed distributed learning approach to predict partially unknown dynamics shared by all agents. Subsequently, we present a novel cooperative tracking control law, which incorporates the proposed distributed learning approach, and show that the designed control law achieves tracking consensus with guaranteed tracking accuracy.

The remainder of this section is structured as follows. After the problem statement is

provided in Section 5.2.1, a novel method for computing distributed GP predictions is introduced in Section 5.2.2. Section 5.2.3 presents the proposed cooperative tracking control law employing distributed predictions and provides an analysis of the tracking error. Finally, in Section 5.2.4, numerical simulations are given to demonstrate the effectiveness of the proposed approaches.

5.2.1. Problem Setting

In order to demonstrate the effectiveness of the GP model aggregation approach for distributed control problems, we extend the example system (3.53) from Section 3.2 to the multi-agent scenario. For this purpose, we consider N possibly heterogeneous follower agents, for simplicity referred to as agents in the following, whose dynamics can be described by single-input systems in the controllable canonical form

$$\dot{x}_{m,1} = x_{m,2}, \qquad \cdots \qquad \dot{x}_{m,d} = \beth_m(\boldsymbol{x}_m) + g_m(\boldsymbol{x}_m)u_m + f(\boldsymbol{x}_m), \tag{5.34}$$

for $m = 1, \ldots, M$, where $\boldsymbol{x}_m(t) = [x_{m,1}(t)\ x_{m,2}(t)\ \ldots\ x_{m,d_x}(t)]^T \in \mathbb{R}^{d_x}$ denotes the state of agent m, $\boldsymbol{x}_m(0) = \boldsymbol{x}_m^0$ is the initial state, $u_m : \mathbb{R}_{0,+} \to \mathbb{U} \subset \mathbb{R}$ is the control input for agent m, $\beth_m : \mathbb{R}^{d_x} \to \mathbb{R}$ and $g_m : \mathbb{R}^{d_x} \to \mathbb{R}$ are known functions that describe the individual dynamics of agent m, and $f : \mathbb{R}^{d_x} \to \mathbb{R}$ represents an unknown nonlinearity shared by all agents such as, e.g., hydrodynamic forces due to ocean currents affecting the dynamics of a fleet of underwater vehicles [14].

In order to ensure the global controllability of each agent, we require that each function $g_m(\cdot)$ satisfies Assumption 3.7, which is a standard prerequisite for the design of feedback linearizing control laws [113, Defintion 13.1] and ensures that each agent's dynamics exhibits a relative degree d_x. Thereby, global controllability is ensured, and the existence of internal dynamics is excluded. Although this assumption restricts the considered system class, the focus of this work is on distributed learning for cooperative tracking control. Therefore, we leave the extension to larger system classes for future research.

The goal of the agents is to follow a virtual leader with state $\boldsymbol{x}_l(t) \in \mathbb{R}^d$, which satisfies the following properties.

Assumption 5.4. *The leader state $\boldsymbol{x}_l(t)$ describes a reference trajectory for the agents, which has the form*

$$\dot{x}_{l,1} = x_{l,2}, \qquad \dot{x}_{l,2} = x_{l,3}, \qquad \cdots \qquad \dot{x}_{l,d_x} = r_{\text{ref}} \tag{5.35}$$

where $r_{\text{ref}} : \mathbb{R}_+ \to \mathbb{R}$ is a continuous reference signal, such that $\boldsymbol{x}_l(t) \in \mathbb{S}$ for all $t \geq 0$ and some compact set $\mathbb{S} \subset \mathbb{R}^{d_x}$, and $|r_{\text{ref}}(t)| \leq \bar{r}$ for some constant $\bar{r} \in \mathbb{R}_+$.

This assumption effectively defines the leader as a reference trajectory with the structure defined in (3.3). Since the reference signal $r_{\text{ref}}(\cdot)$ is a design choice, the assumption of continuity is not restrictive in practice. Moreover, practical problems usually involve tasks in compact state spaces, such that the assumption of a bounded reference is typically not an issue. Therefore, this assumption can frequently be found in literature, e.g., [32, 34], in order to ensure that the agents are capable of following the leader.

As multi-agent systems may comprise a large number of agents in real-world applications, we consider a restricted communication between agents. This communication topology is

described by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots, M\}$ denotes the set of agents, which correspond to the vertices of the graph, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. An edge $(m, m') \in \mathcal{E}$ represents that agent m' can receive information from agent m, and vice versa. The edges of a weighted graph can be compactly represented through the weighted adjacency matrix $\boldsymbol{A}^{\mathrm{ad}} \in \mathbb{R}^{M \times M}$, with elements $A^{\mathrm{ad}}_{mm'} > 0$ if $(m', m) \in \mathcal{E}$. Throughout this section, we assume a fixed topology, i.e., $\boldsymbol{A}^{\mathrm{ad}}$ is constant, and a self-connectivity element $A^{\mathrm{ad}}_{mm} = 0$. Moreover, we assume that only a few agents can communicate with the leader as expressed by the diagonal matrix $\boldsymbol{B}^l = \mathrm{diag}(b^l_1, \dots, b^l_M) \in \mathbb{R}^{M \times M}$, whose diagonal elements satisfy $b^l_m = 1$ if agent m has access to the leader state, and $b_m = 0$ otherwise. In order to allow a cooperative tracking control design, we require the following common assumptions on the communication topology.

Assumption 5.5. *The communication graph $\mathcal{G}$ among the agents is undirected and connected. Moreover, at least one agent is connected to the leader, i.e., there exists $m = 1, \dots, M$ such that $b_m = 1$.*

Assumption 5.6. *The weighted adjacency matrix $\boldsymbol{A}^{\mathrm{ad}} \in \mathbb{R}^{M \times M}$ of the graph $\mathcal{G}$ is weight balanced, i.e., $\mathbf{1}^T \boldsymbol{A}^{\mathrm{ad}} = \boldsymbol{A}^{\mathrm{ad}} \mathbf{1} = \mathbf{1}$.*

Assumption 5.5 guarantees that information about the leader state is directly transmitted to some agents and propagates through the network to all agents since they have at least indirect access through paths starting at the leader. This assumption is a crucial prerequisite to track the leader with all agents [214]. Finally, Assumption 5.6 ensures that information of each agent is equally treated in the network [215].

In order to be able to infer a model of the unmodeled nonlinearity $f(\cdot)$ exhibiting a probabilistically uniformly bounded model error, we exemplarily consider training data satisfying the requirement of Proposition 2.2 as stated in the following.

Assumption 5.7. *Each agent m has a training data set*

$$\mathbb{D}_m = \{(\boldsymbol{z}^{(n)}_m = \boldsymbol{x}^{(n)}_m, y^{(n)}_m = f(\boldsymbol{x}^{(n)}_m) + \epsilon^{(n)})\}_{n=1}^{N_m}, \tag{5.36}$$

such that the observation noise $\epsilon^{(n)}$ in each $\mathbb{D}_m$ individually satisfies the requirements of Assumption 2.1.

This assumption reflects the fact that agents are often able to collect data on their own, without sharing data amongst them. While it does not allow the splitting of a large data set into M subsets based on the observations y_m, assigning samples using the measured states $\boldsymbol{x}_m$ is admissible. Hence, a wide range of assignment strategies is admissible, e.g., based on proximity to the agents' initial states $\boldsymbol{x}_m(0)$ and a random distribution of data to the agents.

Based on these assumptions, we consider the problem of designing a distributed, feedback linearizing controller of the form

$$u_m(t) = \frac{1}{g_m(\boldsymbol{x}_m(t))}(-\beth_m(\boldsymbol{x}_m(t)) - \hat{f}_m(\boldsymbol{x}_m(t)) + \pi_{\mathrm{lin}}(\boldsymbol{\varepsilon}_m(t))), \quad m = 1, \dots, M, \tag{5.37}$$

where $\hat{f}_m : \mathbb{R}^{d_x} \to \mathbb{R}$ is a model of the unknown nonlinearity $f(\cdot)$ obtained from data, which will be specified later, and $\pi_{\mathrm{lin}} : \mathbb{R}^{d_x} \to \mathbb{R}$ is a linear control law depending on the local

consensus error[7]

$$\boldsymbol{\varepsilon}_m(t) = -\sum_{i=1}^{M} A_{m,i}^{\mathrm{ad}}(\boldsymbol{x}_m(t) - \boldsymbol{x}_i(t)) - b_m(\boldsymbol{x}_m(t) - \boldsymbol{x}_l(t)). \tag{5.38}$$

Remark 5.5. *We consider a common control law $\pi_{\mathrm{lin}}(\cdot)$ in all agents merely for notational simplicity. An extension to heterogeneous linear control laws $\pi_{\mathrm{lin},m}(\cdot)$, $m = 1, \ldots, M$, is straightforward.*

The goal of the distributed, feedback linearizing controllers is to track the state of the leader $\boldsymbol{x}_l(t)$ with the agent states $\boldsymbol{x}_m(t)$. While it would be straightforward to employ a model $\hat{f}_m(\cdot)$ based merely on the local data set $\mathbb{D}_m$ in the cooperative controller (5.37) and prove a probabilistically bounded tracking error as defined in Definition 3.1 [210], such an approach crucially relies on training data being evenly-distributed among the agents. In practice, this requirement can possibly lead to poor tracking performance as each agent typically has only data from a small portion of the state space, which we demonstrate in Section 5.2.4. In order to overcome this issue, we consider the problem of designing a cooperative control law of the form (5.37) with distributedly computed models $\hat{f}_m(\cdot)$, such that the tracking accuracy is improved and a probabilistically bounded tracking error is guaranteed.

5.2.2. Consensus-Based Aggregation of Gaussian Process Predictions

In order to address the problem of a distributed computation of models $\hat{f}_m(\cdot)$, we exploit the structure of GP aggregation schemes

$$\tilde{\mu}(\boldsymbol{z}) = \psi_\mu \left(\sum_{m \in \mathbb{M}} \omega_m \boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{z}), \sigma_m^2(\boldsymbol{z})) \right), \tag{5.1 revisited}$$

where we have $\boldsymbol{z} = \boldsymbol{x}$ due to form of the training data stated in Assumption 5.7. This expression requires a centralized evaluation since all GP predictions are needed, but we can formulate the argument of $\psi_\mu(\cdot)$ as a dynamic average $\boldsymbol{p}_{\mathrm{av}}(t) = \sum_{m=1}^{M} \boldsymbol{p}_m(t)/M$ of the local reference signals

$$\boldsymbol{p}_m(t) = \omega_m M \boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{x}(t)), \sigma_m^2(\boldsymbol{x}(t))) \tag{5.39}$$

when time-varying inputs $\boldsymbol{x}(t)$ are considered. The dynamic average $\boldsymbol{p}_{\mathrm{av}}(t)$ can straightforwardly be determined in a distributed fashion using consensus algorithms such as the first-order method

$$\dot{\boldsymbol{\chi}}_m = -k_p \sum_{i=1}^{M} A_{mi}^{\mathrm{ad}}(\boldsymbol{\chi}_m - \boldsymbol{\chi}_i) + \dot{\boldsymbol{p}}_m, \tag{5.40a}$$

$$\boldsymbol{\chi}_m(0) = \omega_m M \boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{x}(0)), \sigma_m^2(\boldsymbol{x}(0))), \tag{5.40b}$$

[7] Since each agent has to obtain information about the reference trajectory through the leader, the local tracking control laws cannot use it as a feedforward term and are consequently time-independent in this section.

where $\boldsymbol{\chi}_m(t)$ denotes the local consensus state of agent m and $k_p \in \mathbb{R}_+$ is a constant controlling the consensus convergence rate. We leave the extension to more sophisticated consensus algorithms open for future research. The local consensus state $\boldsymbol{\chi}_m(t)$ provides each agent with a local estimate of the dynamic average consensus value $\boldsymbol{p}_{\mathrm{av}}(t)$. Therefore, each agent can directly approximate the aggregated mean (5.1) via

$$\hat{\mu}_m(\boldsymbol{x}(t)) = \psi_\mu\left(\boldsymbol{\chi}_m(t)\right). \tag{5.41}$$

Due to the strong theoretical foundations of consensus algorithms, the approximated aggregation $\hat{\mu}_m(\boldsymbol{x}(t))$ inherits the probabilistic uniform error bounds Theorem 5.3 and Theorem 5.4 from centralized aggregation schemes. In order to show this, we define the Laplacian matrix of the graph $\mathcal{G}$ as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}^{\mathrm{ad}}$, where $\boldsymbol{D}$ is a diagonal matrix with elements $D_{mm} = \sum_{i=1}^{M} A^{\mathrm{ad}}_{mi}$. Moreover, the eigenvalues $\lambda_1(\boldsymbol{L}), \ldots, \lambda_M(\boldsymbol{L})$ of the Laplacian matrix $\boldsymbol{L}$ are ordered by magnitude, i.e., $\lambda_1(\boldsymbol{L}) = 0 < \lambda_2(\boldsymbol{L}) \leq \cdots \leq \lambda_M(\boldsymbol{L})$. Based on these definitions, we extend Theorem 5.3 to the distributed aggregation (5.41) as shown in the following proposition.

Proposition 5.1. *Consider an unknown function $f(\cdot)$, a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ with Lipschitz continuous, stationary kernel satisfying Assumption 2.6, and training data sets $\mathbb{D}_m$, such that Assumption 5.7 holds. Moreover, consider an aggregation scheme* (5.1)*, which meets the conditions of Assumption 5.3 and is defined using continuous maps $\psi_\mu(\cdot)$ and $\boldsymbol{\psi}_\omega(\cdot,\cdot)$ with Lipschitz constants L_{ψ_μ} and L_{ψ_ω}, respectively. Assume that the agents can communicate according to a topology satisfying Assumptions 5.5 and 5.6 and that Assumption 2.7 and* (2.75) *hold for each individual GP model. If $\boldsymbol{x}(t) \in \mathbb{S}$ for all $t \in \mathbb{R}_{0,+}$ and there exists a finite $F \in \mathbb{R}_+$ such that*

$$\sup_{t' \geq 0} \|\dot{\boldsymbol{x}}(t')\| < F, \tag{5.42}$$

then, for $\delta \in (0,1)$ satisfying (2.74)*, it holds that*

$$\mathbb{P}\left(|f(\boldsymbol{x}(t)) - \hat{\mu}_m(\boldsymbol{x}(t))| \leq \tilde{\eta}(\boldsymbol{x}(t)), \quad \forall t \in \mathbb{R}_{0,+}, m = 1, \ldots, M\right) \geq 1 - \delta, \tag{5.43}$$

where

$$\tilde{\eta}(\boldsymbol{x}(t)) = \eta(\boldsymbol{x}(t)) + \tilde{\eta}_{\mathrm{tr}}(t) + \frac{L_{\tilde{\mu}} F}{\lambda_2(\boldsymbol{L}) k_p}, \tag{5.44}$$

$$\eta(\boldsymbol{x}) = 2\beta \sum_{m=1}^{M} w_m(\boldsymbol{x}) \sigma_m(\boldsymbol{x}), \tag{5.45}$$

$$\tilde{\eta}_{\mathrm{tr}}(t) = L_{\psi_\mu} \mathrm{e}^{-\lambda_2(\boldsymbol{L}) k_p t} \left\| \left(\boldsymbol{I}_M - \frac{1}{M} \mathbf{1}\mathbf{1}^T \right) \boldsymbol{p}(0) \right\| \tag{5.46}$$

for β defined in (5.20)*, $L_{\tilde{\mu}}$ denoting the Lipschitz constant of the aggregated mean $\tilde{\mu}(\cdot)$, and $\boldsymbol{p}(0) = [\boldsymbol{p}_1^T(0) \ \cdots \ \boldsymbol{p}_N^T(0)]^T$.*

Proof. In order to prove this theorem, we apply the triangle inequality to the left-hand side of (5.43), such that we obtain

$$|f(\boldsymbol{x}(t)) - \hat{\mu}_m(\boldsymbol{x}(t))| \leq |f(\boldsymbol{x}(t)) - \tilde{\mu}(\boldsymbol{x}(t))| + |\tilde{\mu}(\boldsymbol{x}(t)) - \psi_\mu(\boldsymbol{\chi}_m(t))|.$$

Since the assumptions of Theorems 2.2 and 5.3 are satisfied, it follows that the first summand satisfies

$$|f(\boldsymbol{x}(t)) - \tilde{\mu}(\boldsymbol{x}(t))| \leq \eta(\boldsymbol{x}(t)) \tag{5.47}$$

with probability of at least $1 - \delta$. By exploiting the Lipschitz continuity of $\psi_\mu(\cdot)$, we obtain

$$|\tilde{\mu}(\boldsymbol{x}(t)) - \psi_\mu(\boldsymbol{\chi}_m(t))| \leq L_{\psi_\mu} \left\| \boldsymbol{\chi}_m(t) - \sum_{m=1}^{M} w_m \boldsymbol{\psi}_\omega \left(\mu_m(\boldsymbol{x}(t)), \sigma_m^2(\boldsymbol{x}(t)) \right) \right\|.$$

This bound corresponds to the consensus error, which is well-known to be bounded for connected communication graphs with weight-balanced weighted adjacency matrix $\boldsymbol{A}^{\mathrm{ad}}$, see, e.g., [213]. More precisely, if $\|\dot{\boldsymbol{p}}(t)\| = \|[\dot{\boldsymbol{p}}_1^T(t) \ \cdots \ \dot{\boldsymbol{p}}_M^T(t)]^T\|$ is bounded for all $t \geq 0$, we obtain the consensus error bound

$$\begin{aligned} \left\| \boldsymbol{\chi}_m(t) - \sum_{m=1}^{M} w_m \boldsymbol{\psi}_\omega \left(\mu_m(\boldsymbol{x}(t)), \sigma_m^2(\boldsymbol{x}(t)) \right) \right\| \leq \\ \mathrm{e}^{-\lambda_2(\boldsymbol{L}) k_p t} \left\| \left(\boldsymbol{I} - \frac{1}{N} \boldsymbol{1}\boldsymbol{1}^T \right) \boldsymbol{p}(0) \right\| + \frac{1}{\lambda_2(\boldsymbol{L}) k_p} \sup_{0 \leq t' \leq t} \|\dot{\boldsymbol{p}}(t')\|, \end{aligned} \tag{5.48}$$

where $\lambda_2(\boldsymbol{L})$ is the smallest nonzero eigenvalue of $\boldsymbol{L}$. This condition is satisfied due to the definition of $\boldsymbol{p}_n(t)$ in (5.39), the Lipschitz continuity of $\boldsymbol{\psi}_\omega(\cdot)$ guaranteed by assumption, the Lipschitz continuity of $\mu_m(\cdot)$ and $\sigma_m^2(\cdot)$ guaranteed by Lemma 2.2 and Corollary 2.6, respectively, and the assumed boundedness of $\dot{\boldsymbol{x}}(t)$. Therefore, we have $\|\dot{\boldsymbol{p}}(t)\| \leq L_{\tilde{\mu}} F / L_{\psi_\mu}$, where $L_{\tilde{\mu}}$ denotes the Lipschitz constant of $\tilde{\mu}(\cdot)$, such that (5.48) yields

$$|\tilde{\mu}(\boldsymbol{x}(t)) - \psi_\mu(\boldsymbol{\chi}_m(t))| \leq \tilde{\eta}_{\mathrm{tr}}(t) + \frac{L_{\tilde{\mu}} F}{\lambda_2(\boldsymbol{L}) k_p}, \tag{5.49}$$

where $\tilde{\eta}_{\mathrm{tr}}(\cdot)$ is defined in (5.46). Finally, the summation of (5.47) and (5.49) concludes the proof. □

While global Lipschitz continuity is a restrictive assumption, violated, e.g., by the gPoE aggregation, local Lipschitz continuity of $\psi_\mu(\cdot)$ and $\boldsymbol{\psi}_\omega(\cdot, \cdot)$ on their relevant input domains is sufficient for Proposition 5.1. Common aggregation schemes such as gPoE usually satisfy this property since the posterior variance $\sigma_m^2(\cdot)$ is guaranteed to be positive and bounded for GPs trained using finite data sets. Therefore, this assumption is not restrictive in practice. Similarly, the assumption on the boundedness of the temporal derivative $\dot{\boldsymbol{x}}(t)$ is not a severe restriction since it merely requires $\boldsymbol{x}(\cdot)$ to evolve continuously over time. When $\boldsymbol{x}(t)$ is the state of a control system, this assumption crucially depends on the specific control law. We will show in Section 5.2.3 that the proposed cooperative tracking control law guarantees (5.42) and admits the straightforward computation of the constant F.

Based on these unrestrictive assumptions, Proposition 5.1 decouples the different components of the error bound in an intuitive way. In addition to the error bound resulting from a centralized aggregation $\eta(\cdot)$, it considers the transient behavior of the consensus algorithm in $\tilde{\eta}_{\mathrm{tr}}(\cdot)$, and bounds the stationary consensus error using $L_{\tilde{\mu}} F / (\lambda_2(\boldsymbol{L}) k_p)$. While the transient error bound $\tilde{\eta}_{\mathrm{tr}}(\cdot)$ is almost negligible in practice due to the exponential decrease, the stationary error bound $L_{\tilde{\mu}} F / (\lambda_2(\boldsymbol{L}) k_p)$ can become large if F is large. In order to reduce this component of the bound, the connectivity of the communication graph $\mathcal{G}$ can be increased,

such that $\lambda_2(\boldsymbol{L})$ grows [215]. Analogously, one can increase the consensus gain k_p, such that an arbitrarily small stationary consensus error can be guaranteed in principle. Due to [213, Theorem 2], it is in fact straightforward to see that $\tilde{\eta}(\boldsymbol{x}(t))$ converges to the centralized prediction error bound $\eta(\boldsymbol{x}(t))$ if $\boldsymbol{x}(\cdot)$ converges to a constant value.

Even though (5.40) admits a straightforward derivation of error bounds, it is impractical for an implementation as it requires the derivatives $\dot{\boldsymbol{p}}_m(t)$ and consequently $\dot{\boldsymbol{x}}(t)$, which can be a strong restriction. This can be overcome via a simple change of variables $\boldsymbol{\chi}_m = \boldsymbol{p}_m - \tilde{\boldsymbol{\chi}}_m$, which directly yields the new consensus system

$$\dot{\tilde{\boldsymbol{\chi}}}_m = k_p \sum_{i=1}^{M} A_{mi}^{\mathrm{Ad}} (\boldsymbol{p}_m - \boldsymbol{p}_i - \tilde{\boldsymbol{\chi}}_m + \tilde{\boldsymbol{\chi}}_i), \tag{5.50}$$

with initial state $\tilde{\boldsymbol{\chi}}_m(0) = \boldsymbol{0}$. Based on the new consensus state $\tilde{\boldsymbol{\chi}}_m(t)$, the prediction can be computed using $\hat{\mu}_m(\boldsymbol{x}(t)) = \psi_\mu(\boldsymbol{p}_m(t) - \tilde{\boldsymbol{\chi}}_m(t))$. Since merely a linear change of variables is employed for adapting the consensus algorithm, it is straightforward to see that Proposition 5.1 immediately extends to the consensus scheme (5.50). Thereby, this scheme combines practical applicability and strong theoretical guarantees.

5.2.3. Cooperative Tracking Control using Distributed Gaussian Processes

In order to demonstrate the effectiveness of the distributed aggregation scheme (5.41), we employ it in a feedback linearizing controller to compensate for unmodeled nonlinearities. Before starting with the derivation of this cooperative tracking control law, we represent the multi-agent system in a compact form to simplify notation. The dynamics (5.34) of all agents can be jointly described by

$$\dot{\boldsymbol{x}}^1 = \boldsymbol{x}^2, \qquad \cdots \qquad \dot{\boldsymbol{x}}^d = \beth(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x}), \tag{5.51}$$

where

$$\beth(\boldsymbol{x}) = \begin{bmatrix} \beth_1(\boldsymbol{x}_1) \\ \vdots \\ \beth_M(\boldsymbol{x}_M) \end{bmatrix}, \quad \boldsymbol{G}(\boldsymbol{x}) = \begin{bmatrix} g_1(\boldsymbol{x}_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & g_M(\boldsymbol{x}_M) \end{bmatrix}, \quad \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_M) \end{bmatrix}, \tag{5.52}$$

and $\boldsymbol{x}^j = [x_{1,j} \ \dots \ x_{M,j}]^T$, $j = 1, \dots, d_x$, $\boldsymbol{x} = [\boldsymbol{x}_1^T \ \dots \ \boldsymbol{x}_M^T]^T$, $\boldsymbol{u} = [u_1 \ \dots \ u_M]^T$. Similarly, the leader dynamics (5.35) can be augmented, which yields

$$\dot{\boldsymbol{x}}_l^1 = \boldsymbol{x}_l^2, \qquad \dot{\boldsymbol{x}}_l^2 = \boldsymbol{x}_l^3, \qquad \cdots \qquad \dot{\boldsymbol{x}}_l^{d_x} = r_{\mathrm{ref}} \boldsymbol{1}, \tag{5.53}$$

and we define the joint consensus error in the j-th dimension as $\boldsymbol{\varepsilon}^j = [\varepsilon_{1,j} \ \varepsilon_{2,j} \ \dots \ \varepsilon_{M,j}]^T \in \mathbb{R}^M$. Using this notation, we introduce the filtered state $\boldsymbol{\nu} = [\nu_1 \ \cdots \ \nu_M]^T$ in analogy to standard feedback linearizing control as $\boldsymbol{\nu} = \boldsymbol{\mathcal{E}}[\tilde{\theta}_1 \ \cdots \ \tilde{\theta}_{d_x - 1} \ 1]^T \in \mathbb{R}^M$, where $\boldsymbol{\mathcal{E}} = [\boldsymbol{\varepsilon}^1 \ \cdots \ \boldsymbol{\varepsilon}^{d_x}] \in \mathbb{R}^{M \times d_x}$ and $\tilde{\theta}_i$, $i = 1, \dots, d_x - 1$, are filter coefficients defining a Hurwitz polynomial. It can easily be checked that the local filtered states ν_m can be computed purely based on the local consensus error $\boldsymbol{\varepsilon}_m$, such that we can define the linear control law $\pi_{\mathrm{lin}}(\cdot)$ in (5.37) as

$$\pi_{\mathrm{lin}}(\boldsymbol{\varepsilon}_m) = k_c \nu_n + \sum_{i=1}^{d_x - 1} \tilde{\theta}_i \varepsilon_{m,i+1}, \tag{5.54}$$

where $k_c \in \mathbb{R}_+$ is a constant control gain. In order to completely specify the feedback linearizing control law (5.37), it remains to define the compensation $\hat{f}_m(\cdot)$ of the unmodeled nonlinearity $f(\cdot)$. For this purpose, we employ the distributed GP prediction scheme proposed in Section 5.2.2, yielding the control law

$$u_m(t) = \frac{1}{g_m(\boldsymbol{x}_m(t))} \left(-\beth_m(\boldsymbol{x}_m(t)) - \psi_\mu(\boldsymbol{\chi}_m(t)) + \pi(\boldsymbol{\varepsilon}_m(t))\right), \tag{5.55a}$$

$$\dot{\boldsymbol{\chi}}_m = -k_p \sum_{i=1}^{M} A_{mi}^{\mathrm{ad}}(\boldsymbol{\chi}_m - \boldsymbol{\chi}_i) + \dot{\tilde{\boldsymbol{p}}}_m, \tag{5.55b}$$

$$\tilde{\boldsymbol{p}}_m(t) = w_m M \boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{x}_m(t)), \sigma_m^2(\boldsymbol{x}_m(t))), \tag{5.55c}$$

$$\tilde{\boldsymbol{\chi}}_m(0) = w_m M \boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{x}_n(0)), \sigma_m^2(\boldsymbol{x}_m(0))). \tag{5.55d}$$

For proving a probabilistically bounded tracking error of the multi-agent system controlled by (5.55), inspired by [214], we define a Lyapunov function

$$V(\boldsymbol{\nu}, \boldsymbol{\mathcal{E}}_1) = \frac{1}{2}\boldsymbol{\nu}^T\boldsymbol{\nu} + \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\mathcal{E}}_1 \boldsymbol{P} \boldsymbol{\mathcal{E}}_1^T\right), \tag{5.56}$$

where $\boldsymbol{\mathcal{E}}_1 = \begin{bmatrix}\boldsymbol{\varepsilon}^1 & \boldsymbol{\varepsilon}^2 & \cdots & \boldsymbol{\varepsilon}^{d_x-1}\end{bmatrix} \in \mathbb{R}^{M\times(d_x-1)}$ and $\boldsymbol{P} \in \mathbb{R}^{(d_x-1)\times(d_x-1)}$ is a positive definite, symmetric matrix. The dynamics of the filtered state $\boldsymbol{\nu}$ can be described by

$$\dot{\boldsymbol{\nu}} = -\tilde{\boldsymbol{L}}\Big(\beth(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x}) - r_{\mathrm{ref}}\mathbf{1}\Big) + \begin{bmatrix}\sum_{i=1}^{d_x-1}\tilde{\theta}_i\varepsilon_{1,i+1}\\ \vdots \\ \sum_{i=1}^{d_x-1}\tilde{\theta}_i\varepsilon_{M,i+1}\end{bmatrix}, \tag{5.57}$$

where $\tilde{\boldsymbol{L}} = \boldsymbol{L} + \boldsymbol{B}^l$. From the definition of the filtered state $\boldsymbol{\nu}$, it follows that

$$\dot{\boldsymbol{\mathcal{E}}}_1 = \boldsymbol{\mathcal{E}}_1\tilde{\boldsymbol{\Theta}}^T + \boldsymbol{\nu}\boldsymbol{l}^T, \tag{5.58}$$

where $\boldsymbol{l} = [0\ 0\ \cdots\ 1]^T \in \mathbb{R}^{d_x-1}$, and

$$\tilde{\boldsymbol{\Theta}} = \left[\begin{array}{c|c} \boldsymbol{0}_{(d_x-2)\times 1} & \boldsymbol{I}_{d_x-2} \\ \hline -\tilde{\theta}_1 & -\tilde{\theta}_2 \cdots -\tilde{\theta}_{d_x-1}\end{array}\right]. \tag{5.59}$$

Since the second summand of the Lyapunov function (5.56) depends on $\boldsymbol{\mathcal{E}}_1$, (5.58) allows us to relate the decrease of the Lyapunov function along system trajectories to the weights of the linear control law (5.54) described by $\tilde{\boldsymbol{\Theta}}$. Analogously to the augmented Laplacian $\tilde{\boldsymbol{L}}$, the matrix $\tilde{\boldsymbol{\Theta}}$ can be employed to define a positive definite matrix. For this definition, we employ the fact that $\tilde{\boldsymbol{\Theta}}$ is a Hurwitz matrix, such that $\boldsymbol{P}$ is chosen to be the solution to the Lyapunov equation

$$\tilde{\boldsymbol{\Theta}}^T\boldsymbol{P} + \boldsymbol{P}\tilde{\boldsymbol{\Theta}} = -\boldsymbol{Q}, \tag{5.60}$$

where $\boldsymbol{Q}$ is an arbitrary, positive definite, symmetric matrix.

Before starting with the stability analysis of the multi-agent system controlled by (5.55), we introduce the following lemma, which extends the prediction error bound in Proposition 5.1 by expressing the impact of the usage of local states $\boldsymbol{x}_m(t)$ in (5.55c) in terms of $\boldsymbol{\nu}$ and $\boldsymbol{\mathcal{E}}_1$.

Lemma 5.1. *Consider an unknown function $f(\cdot)$, a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ with Lipschitz continuous, stationary kernel satisfying Assumption 2.6, and training data sets $\mathbb{D}_m$, such that Assumption 5.7 holds. Moreover, consider an aggregation scheme (5.1), which meets the conditions of Assumption 5.3 and is defined using continuous maps $\psi_\mu(\cdot)$ and $\boldsymbol{\psi}_\omega(\cdot,\cdot)$ with Lipschitz constants L_{ψ_μ} and L_{ψ_ω}, respectively. Assume that the agents can communicate according to a topology satisfying Assumptions 5.5 and 5.6 and that Assumption 2.7 and (2.75) hold for each individual GP model. If $\boldsymbol{x}(t) \in \mathbb{S}$ for all $t \in \mathbb{R}_{0,+}$ and there exists a finite $F \in \mathbb{R}_+$ such that*

$$\max_{m=1,\dots,M} \sup_{t' \geq 0} \|\dot{\boldsymbol{x}}_m(t')\| < F, \tag{5.61}$$

then it holds with probability of at least $1-\delta$ for all $t \in \mathbb{R}_{0,+}$, $\boldsymbol{x}(t) \in \mathbb{S}^N = \mathbb{S} \times \cdots \times \mathbb{S}$ that

$$\|\boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{\psi}_\mu(\boldsymbol{\chi}(t))\| \leq \sqrt{M}\tilde{\eta}(\boldsymbol{x}_l(t)) + \frac{\left(\sqrt{M}L_{\tilde{\mu}} + L_f\right)}{\underline{\lambda}(\tilde{\boldsymbol{L}})} \left(\|\boldsymbol{\nu}(t)\| + (1 + \|\tilde{\boldsymbol{\Theta}}\|_{\mathrm{F}})\|\boldsymbol{\mathcal{E}}_1(t)\|_{\mathrm{F}}\right), \tag{5.62}$$

where $\boldsymbol{\psi}_\mu(\boldsymbol{\chi}(t)) = \left[\psi_\mu(\boldsymbol{\chi}_1^T(t)) \ \cdots \ \psi_\mu(\boldsymbol{\chi}_M^T(t))\right]^T$.

Proof. In order to prove this lemma, we first bound the difference between $\psi_\mu(\boldsymbol{\chi}_n(t))$ and the corresponding centralized aggregation, which yields analogously to Proposition 5.1

$$|\psi_\mu(\boldsymbol{\chi}_m(t)) - \check{\mu}_m(\boldsymbol{x}(t))| \leq \tilde{\eta}_{\mathrm{tr}}(t) + \frac{L_{\tilde{\mu}} F}{\lambda_2(\boldsymbol{L}) k_p}, \tag{5.63}$$

where

$$\check{\mu}_m(\boldsymbol{x}(t)) = \psi_\mu \left(\sum_{m=1}^{M} w_m \boldsymbol{\psi}_\omega \left(\mu_m(\boldsymbol{x}_m(t)), \sigma_m^2(\boldsymbol{x}_m(t)) \right) \right). \tag{5.64}$$

Due to Lipschitz continuity of $\psi_\mu(\cdot)$, $\boldsymbol{\psi}_\omega(\cdot,\cdot)$, $\mu_m(\cdot)$ and $\sigma_m^2(\cdot)$, we can bound the distance to the aggregated prediction for $\boldsymbol{x}_l(t)$ by

$$|\check{\mu}_m(\boldsymbol{x}(t)) - \tilde{\mu}(\boldsymbol{x}_l(t))|^2 \leq L_{\psi_\mu}^2 L_{\psi_\omega}^2 \sum_{i=1}^{M} (L_{\mu_i}^2 + L_{\sigma_i^2}^2) \|\boldsymbol{x}_l - \boldsymbol{x}_i\|^2. \tag{5.65}$$

For obtaining a bound in terms of $\boldsymbol{\mathcal{E}}(t)$ and $\boldsymbol{\nu}(t)$, we make use of the Cauchy-Schwarz and the triangle inequality to get

$$\|\check{\boldsymbol{\mu}}(\boldsymbol{x}(t)) - \tilde{\boldsymbol{\mu}}(\boldsymbol{x}_l(t))\|^2 \leq M L_{\tilde{\mu}}^2 \sum_{i=1}^{M} \|\boldsymbol{x}_l(t) - \boldsymbol{x}_i(t)\|^2, \tag{5.66}$$

where $\check{\boldsymbol{\mu}}(\boldsymbol{x}(t))$ and $\tilde{\boldsymbol{\mu}}(\boldsymbol{x}_l(t))$ are the concatenations of $\check{\mu}_m(\boldsymbol{x}(t))$ and $\tilde{\mu}(\boldsymbol{x}_l(t))$, respectively. By reordering the summands, it can be seen that $\sum_{i=1}^{M} \|\boldsymbol{x}_l(t) - \boldsymbol{x}_i(t)\|^2 = \sum_{i=1}^{d_x} \|\boldsymbol{x}_l^i(t) - \boldsymbol{x}^i(t)\|^2$, such that we have

$$\|\check{\boldsymbol{\mu}}(\boldsymbol{x}(t)) - \tilde{\boldsymbol{\mu}}(\boldsymbol{x}_l(t))\| \leq \frac{\sqrt{M} L_{\tilde{\mu}}}{\underline{\lambda}(\tilde{\boldsymbol{L}})} \left(\|\boldsymbol{\nu}(t)\| + (1 + \|\tilde{\boldsymbol{\Theta}}\|_{\mathrm{F}})\|\boldsymbol{\mathcal{E}}_1(t)\|_{\mathrm{F}}\right) \tag{5.67}$$

due to $\boldsymbol{\varepsilon}^j(t) = -\tilde{\boldsymbol{L}}(\boldsymbol{x}^j(t) - \boldsymbol{x}_l^j(t))$, $\sum_{i=1}^{d_x} \|\boldsymbol{\varepsilon}^i(t)\|^2 = \|\boldsymbol{\varepsilon}^1(t)\|^2 + \|\dot{\boldsymbol{\mathcal{E}}}_1(t)\|_{\mathrm{F}}^2$ and (5.58). By applying the triangle inequality, we finally obtain

$$\|\boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{\psi}_\mu(\boldsymbol{\chi}(t))\| \leq \|\boldsymbol{f}(\boldsymbol{x}(t)) - \tilde{\boldsymbol{\mu}}(\boldsymbol{x}_l(t))\| + \sqrt{M}\tilde{\eta}_{\mathrm{tr}}(t) + \frac{\sqrt{M}L_{\tilde{\mu}}F}{\lambda_2(\boldsymbol{L})k_p} \tag{5.68}$$

$$+ \frac{\sqrt{M}L_{\tilde{\mu}}}{\underline{\lambda}(\tilde{\boldsymbol{L}})}\left(\|\boldsymbol{\nu}(t)\| + (1 + \|\tilde{\boldsymbol{\Theta}}\|_{\mathrm{F}})\|\boldsymbol{\mathcal{E}}_1(t)\|_{\mathrm{F}}\right), \tag{5.69}$$

which directly yields the result due to Theorems 2.2 and 5.3 and a linearization of $\boldsymbol{f}(\cdot)$ around $\boldsymbol{x}_l(t)$. □

Lemma 5.1 theoretically justifies our approach of employing the local agent states for computing the local predictions: the smaller the consensus error is, the lower the impact on the prediction error bound with local states. This is particularly beneficial since the feedback control $\pi_{\mathrm{lin}}(\cdot)$ dominates the control law (5.55) for large consensus errors regardless of the prediction error, such that a small consensus error can typically be ensured. In fact, due to the linear relationship between consensus error and prediction error bound, it is even possible to recover Proposition 5.1 in the limit of vanishing consensus errors.

Note that Lemma 5.1 requires a bound F for the agents' state derivative, which depends on the control law (5.55) and consequently on the distributed predictions. Due to the boundedness of GP predictions, we can provide a closed-form bound F as shown in the following lemma.

Lemma 5.2. *Consider a multi-agent system* (5.34) *with M agents satisfying Assumption 3.7 and an unmodeled nonlinearity $f(\cdot)$ satisfying Assumption 2.6. Consider a leader of the form* (5.35)*. Moreover, assume that a control law* (5.55) *with a distributed prediction is employed, which meets the conditions of Assumption 5.3 and is defined using continuous maps $\psi_\mu(\cdot)$ and $\boldsymbol{\psi}_\omega(\cdot,\cdot)$ with Lipschitz constants L_{ψ_μ} and L_{ψ_ω}, respectively. If the state of every agent remains in a compact set $\mathbb{S}$ for all $t \in \mathbb{R}_{0,+}$, then we have*

$$\max_{m=1,\dots,M} \sup_{t' \geq 0} \|\dot{\boldsymbol{x}}_m(t')\| \leq F \tag{5.70}$$

for

$$F = \bar{f} + \bar{\mu} + \left(1 + (k_c+1)\sqrt{\|\tilde{\boldsymbol{\Theta}}\|_F^2 - d_x + 3}\right) \max_{\boldsymbol{x},\boldsymbol{x}' \in \mathbb{S}} \|\boldsymbol{x} - \boldsymbol{x}'\| \tag{5.71}$$

where $\bar{f} = \max_{\boldsymbol{x}\in\mathbb{S}} |f(\boldsymbol{x})|$, and $\bar{\mu} = \max_{\boldsymbol{x}_m \in \mathbb{S}, m=1,\dots,M} |\psi_\mu(\sum_{m=1}^M \omega_m \boldsymbol{\psi}_\omega(\mu_m(\boldsymbol{x}_m), \sigma_m^2(\boldsymbol{x}_m)))|$.

Proof. By substituting the control law (5.55a) into (5.34), we obtain

$$\dot{x}_{n,d} = \boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{\psi}_\mu(\boldsymbol{\chi}(t)) + \pi_{\mathrm{lin}}(\boldsymbol{\varepsilon}_n). \tag{5.72}$$

Due to the compactness of $\mathbb{S}$ and the linearity of $\pi_{\mathrm{lin}}(\cdot)$ in $\boldsymbol{\varepsilon}_n$, we have

$$|\pi_{\mathrm{lin}}(\boldsymbol{\varepsilon}_n)| \leq (k_c+1)\sqrt{\|\tilde{\boldsymbol{\Theta}}\|_F^2 - d_x + 3} \max_{\boldsymbol{x},\boldsymbol{x}' \in \mathbb{S}} \|\boldsymbol{x} - \boldsymbol{x}'\|. \tag{5.73}$$

Due to continuity of $\boldsymbol{\psi}_\omega(\cdot,\cdot)$, $\psi_\mu(\cdot)$, $\mu_m(\cdot)$, and $\sigma_m^2(\cdot)$, it follows that $\psi_\mu(\boldsymbol{\chi}_m(t))$ is bounded by $\bar{\mu}$ on the compact domain $\mathbb{S}$. Since $f(\cdot)$ is continuous, there exists a finite upper bound $\bar{f}$ on the compact set $\mathbb{S}$. Hence, it directly follows that

$$|f(\boldsymbol{x}_m(t)) - \psi_\mu(\boldsymbol{\chi}_m(t))| \leq \bar{f} + \bar{\mu}. \tag{5.74}$$

Exploiting the compactness of $\mathbb{S}$ one more time, we finally obtain $\sup_{t\geq 0} \|\dot{\boldsymbol{x}}_m\| \leq F$ for F defined in (5.71), such that Lemma 5.1 holds. □

Since the derivative state not only depends on the distributed prediction but also on the linear control law $\pi_{\text{lin}}(\cdot)$, the bound (5.71) depends linearly on the control gain k_c. This introduces a linear dependency of the prediction error bound in Lemma 5.1 on k_c due to (5.44), which can be compensated by the reciprocal dependency on the prediction gain k_p. Therefore, the prediction gain k_p should be chosen greater than the control gain k_c, which intuitively resembles the fact that convergence of the prediction is necessary before a consensus of the agent states can be achieved.

Using these auxiliary results, we quantify the tracking error bound for multi-agent systems controlled by (5.55) as shown in the following.

Theorem 5.5. *Consider an unknown function $f(\cdot)$, a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$ with Lipschitz continuous, stationary kernel satisfying Assumption 2.6, and training data sets $\mathbb{D}_m$, such that Assumption 5.7 holds. Moreover, consider an aggregation scheme (5.1), which meets the conditions of Assumption 5.3 and is defined using continuous maps $\psi_\mu(\cdot)$ and $\boldsymbol{\psi}_\omega(\cdot,\cdot)$ with Lipschitz constants L_{ψ_μ} and L_{ψ_ω}, respectively. Assume that the agents can communicate according to a topology satisfying Assumptions 5.5 and 5.6 and that Assumption 2.7 and (2.75) hold for each individual GP model. Choose a control gain $k_c \in \mathbb{R}_+$ such that $\boldsymbol{\Upsilon} \succ \mathbf{0}$, where*

$$\boldsymbol{\Upsilon} = \begin{bmatrix} k_c \underline{\lambda}(\tilde{\boldsymbol{L}}) - \iota & -\frac{\iota}{2}(1 + \|\tilde{\boldsymbol{\Theta}}\|_{\text{F}}) - \frac{1}{2}\bar{\lambda}(\boldsymbol{P}) \\ -\frac{\iota}{2}(1 + \|\tilde{\boldsymbol{\Theta}}\|_{\text{F}}) - \frac{1}{2}\bar{\lambda}(\boldsymbol{P}) & \frac{1}{2}\underline{\lambda}(\boldsymbol{Q}) \end{bmatrix}, \tag{5.75}$$

$$\iota = \|\boldsymbol{I}_N - \tilde{\boldsymbol{L}}\|\|\tilde{\boldsymbol{\theta}}\| + \frac{(\sqrt{M} L_{\tilde{\mu}} + L_f)\bar{\lambda}(\tilde{\boldsymbol{L}})}{\underline{\lambda}(\tilde{\boldsymbol{L}})} \tag{5.76}$$

for $\boldsymbol{P}$, $\boldsymbol{Q}$ defined in (5.60) such that $\underline{\lambda}(\boldsymbol{P}) \geq 1$. Then, the tracking error $\boldsymbol{e} = [\boldsymbol{e}_1^T \ \cdots \ \boldsymbol{e}_M^T]^T$ admits a probabilistic bound $\upsilon(\cdot)$, if $\mathbb{B}_{\upsilon(t)}(\boldsymbol{x}_l(t)) \subset \mathbb{S}$ holds for all $t \in \mathbb{R}_{0,+}$, where $\upsilon(\cdot)$ is defined through

$$\upsilon(t) = \frac{\sqrt{M}\bar{\lambda}(\tilde{\boldsymbol{L}})\sqrt{\bar{\lambda}(\boldsymbol{P})}(1 + \|\tilde{\boldsymbol{\Theta}}\|_{\text{F}})}{\underline{\lambda}(\boldsymbol{\Upsilon})\underline{\lambda}(\tilde{\boldsymbol{L}})} \left(\sup_{0\leq t'\leq t} \tilde{\eta}(\boldsymbol{x}_l(t')) + \bar{r} \right). \tag{5.77}$$

Proof. We prove this theorem by showing that the temporal derivative of the Lyapunov function (5.56) is decreasing except for a small ball around $\boldsymbol{\varepsilon} = \mathbf{0}$. Let $V_1(\boldsymbol{\nu}) = \frac{1}{2}\boldsymbol{\nu}^T\boldsymbol{\nu}$ denote the first summand of (5.56). Due to (5.57), the temporal derivative of $V_1(\cdot)$ is given by

$$\dot{V}_1(\boldsymbol{\nu}) = \boldsymbol{\nu}^T \left(-\tilde{\boldsymbol{L}} \left(\boldsymbol{\beth}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x}) - \dot{\boldsymbol{x}}_l^{d_x} \right) + \begin{bmatrix} \sum_{i=1}^{d_x-1} \tilde{\theta}_i \varepsilon_{1,i+1} \\ \vdots \\ \sum_{i=1}^{d_x-1} \tilde{\theta}_i \varepsilon_{M,i+1} \end{bmatrix} \right). \tag{5.78}$$

Substituting the control law (5.55a) yields

$$\dot{V}_1(\boldsymbol{\nu}) = -k_c \boldsymbol{\nu}^T \tilde{\boldsymbol{L}} \boldsymbol{\nu} - \boldsymbol{\nu}^T \tilde{\boldsymbol{L}} \left(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{\psi}_\mu(\boldsymbol{\chi}) - \dot{\boldsymbol{x}}_l^{d_x} \right) + \boldsymbol{\nu}^T \left(\boldsymbol{I}_N - \tilde{\boldsymbol{L}} \right) \begin{bmatrix} \sum_{i=1}^{d_x-1} \tilde{\theta}_i \varepsilon_{1,i+1} \\ \vdots \\ \sum_{i=1}^{d_x-1} \tilde{\theta}_i \varepsilon_{M,i+1} \end{bmatrix}. \tag{5.79}$$

It can be easily seen that

$$\begin{bmatrix} \sum_{i=1}^{d_x-1} \tilde{\theta}_i \varepsilon_{1,i+1} \\ \vdots \\ \sum_{i=1}^{d_x-1} \tilde{\theta}_i \varepsilon_{M,i+1} \end{bmatrix} = \dot{\boldsymbol{\mathcal{E}}}_1 \tilde{\boldsymbol{\theta}} \tag{5.80}$$

with $\tilde{\boldsymbol{\theta}} = [\tilde{\theta}_1 \ \cdots \ \tilde{\theta}_{d_x-1}]^T$. Therefore, we have due to (5.58) that

$$\boldsymbol{\nu}^T \left(\boldsymbol{I}_N - \tilde{\boldsymbol{L}} \right) \begin{bmatrix} \sum_{i=1}^{d_x-1} \tilde{\theta}_i \varepsilon_{1,i+1} \\ \vdots \\ \sum_{i=1}^{d_x-1} \tilde{\theta}_i \varepsilon_{M,i+1} \end{bmatrix} = \boldsymbol{\nu}^T \left(\boldsymbol{I}_N - \tilde{\boldsymbol{L}} \right) \left(\boldsymbol{\nu} \boldsymbol{l}^T + \boldsymbol{\mathcal{E}}_1 \tilde{\boldsymbol{\Theta}}^T \right) \tilde{\boldsymbol{\theta}}, \tag{5.81}$$

such that we can bound $\dot{V}_1(\boldsymbol{\nu})$ by

$$\begin{aligned} \dot{V}_1(\boldsymbol{\nu}) \leq & \left(-k_c \underline{\lambda}(\tilde{\boldsymbol{L}}) + \bar{\sigma}(\boldsymbol{I}_N - \tilde{\boldsymbol{L}}) \|\tilde{\boldsymbol{\theta}}\| \right) \|\boldsymbol{\nu}\|^2 + \|\boldsymbol{I}_N - \tilde{\boldsymbol{L}}\| \|\tilde{\boldsymbol{\theta}}\| \|\tilde{\boldsymbol{\Theta}}\|_F \|\boldsymbol{\nu}\| \|\boldsymbol{\mathcal{E}}_1\|_F \\ & + \bar{\lambda}(\tilde{\boldsymbol{L}}) \left(\|\boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{\psi}_\mu(\boldsymbol{\chi}(t))\| + \left\| \dot{\boldsymbol{x}}_l^{d_x} \right\| \right) \|\boldsymbol{\nu}\|. \end{aligned} \tag{5.82}$$

Boundedness of $\|\boldsymbol{x}_l^{d_x}\|$ holds by Assumption 5.4, such that $\|\dot{\boldsymbol{x}}_l^{d_x}\| \leq \sqrt{M}\bar{r}$. For bounding $\|\boldsymbol{f}(\boldsymbol{x}(t)) - \boldsymbol{\psi}_\mu(\boldsymbol{\chi}(t))\|$ we employ Lemma 5.1 and Lemma 5.2, such that we can substitute (5.62) in (5.82) to get

$$\dot{V}_1(\boldsymbol{\nu}) \leq \left(-k_c \underline{\lambda}(\tilde{\boldsymbol{L}}) + \iota \right) \|\boldsymbol{\nu}\|^2 + \iota(1 + \|\tilde{\boldsymbol{\Theta}}\|_{\mathrm{F}}) \|\boldsymbol{\nu}\| \|\boldsymbol{\mathcal{E}}_1\|_F + \bar{\lambda}(\tilde{\boldsymbol{L}}) \sqrt{M} \left(\tilde{\eta}(\boldsymbol{x}_l)\| + \bar{r} \right) \|\boldsymbol{\nu}\|, \tag{5.83}$$

where ι is defined in (5.76). For bounding the temporal derivative of the second summand of (5.56) denoted as $V_2(\boldsymbol{\mathcal{E}}_1) = \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\mathcal{E}}_1 \boldsymbol{P} \boldsymbol{\mathcal{E}}_1^T \right)$ we proceed similarly, such that it follows from (5.58) and (5.60) that

$$\dot{V}_2(\boldsymbol{\mathcal{E}}_1) \leq -\frac{1}{2} \underline{\lambda}(\boldsymbol{Q}) \|\boldsymbol{\mathcal{E}}_1\|_F^2 + \bar{\lambda}(\boldsymbol{P}) \|\boldsymbol{\nu}\| \|\boldsymbol{\mathcal{E}}_1\|_F. \tag{5.84}$$

Combining (5.83) and (5.84), and writing it in a quadratic form yields

$$\dot{V}(\boldsymbol{\nu}, \boldsymbol{\mathcal{E}}_1) \leq - \begin{bmatrix} \|\boldsymbol{\nu}\| & \|\boldsymbol{\mathcal{E}}_1\|_F \end{bmatrix} \boldsymbol{\Upsilon} \begin{bmatrix} \|\boldsymbol{\nu}\| \\ \|\boldsymbol{\mathcal{E}}_1\|_F \end{bmatrix} + \begin{bmatrix} \sqrt{M} \bar{\lambda}(\tilde{\boldsymbol{L}}) \left(\tilde{\eta}(\boldsymbol{x}_l) + \bar{r} \right) & 0 \end{bmatrix} \begin{bmatrix} \|\boldsymbol{\nu}\| \\ \|\boldsymbol{\mathcal{E}}_1\|_F \end{bmatrix}, \tag{5.85}$$

where $\boldsymbol{\Upsilon}$ is defined in (5.75). By employing Sylvester's criterion, it is straightforward to see that there exist a k_c, such that $\boldsymbol{\Upsilon}$ is positive definite. Then, we have

$$\dot{V}(\boldsymbol{\nu}, \boldsymbol{\mathcal{E}}_1) \leq 0, \quad \text{for all } \boldsymbol{\nu}, \boldsymbol{\mathcal{E}}_1 : \left\| \begin{bmatrix} \|\boldsymbol{\nu}\| & \|\boldsymbol{\mathcal{E}}_1\|_F \end{bmatrix} \right\| \leq \frac{\sqrt{M} \bar{\lambda}(\tilde{\boldsymbol{L}}) \left(\tilde{\eta}(\boldsymbol{x}_l) + \bar{r} \right)}{\underline{\lambda}(\boldsymbol{\Upsilon})}. \tag{5.86}$$

As $V(\cdot)$ has a quadratic form and $\underline{\lambda}(\boldsymbol{P}) \geq 1$, (5.86) implies that

$$\left\| \begin{bmatrix} \|\boldsymbol{\nu}(t)\| & \|\boldsymbol{\mathcal{E}}_1(t)\|_F \end{bmatrix} \right\| \leq \frac{\sqrt{\bar{\lambda}(\boldsymbol{P})}\sqrt{M}\bar{\lambda}(\tilde{\boldsymbol{L}})\left(\tilde{\eta}(\boldsymbol{x}_l(t)) + \bar{r}\right)}{\underline{\lambda}(\boldsymbol{\Upsilon})} \tag{5.87}$$

for sufficiently small initial values $\boldsymbol{\nu}(0)$, $\boldsymbol{\mathcal{E}}_1(0)$. Due to $\sum_{i=1}^{d_x} \|\boldsymbol{\varepsilon}^i\|^2 = \|\boldsymbol{\varepsilon}^1\|^2 + \|\dot{\boldsymbol{\mathcal{E}}}_1\|_{\mathrm{F}}^2$ and (5.58), the total consensus error can be bounded in terms of $\boldsymbol{\nu}$ and $\boldsymbol{\mathcal{E}}_1$, which leads to

$$\|\boldsymbol{\varepsilon}(t)\| \leq \frac{\sqrt{\bar{\lambda}(\boldsymbol{P})}\sqrt{M}\bar{\lambda}(\tilde{\boldsymbol{L}})(1 + \|\tilde{\boldsymbol{\Theta}}\|_F)\left(\tilde{\eta}(\boldsymbol{x}_l(t)) + \bar{r}\right)}{\underline{\lambda}(\boldsymbol{\Upsilon})}. \tag{5.88}$$

Finally, the cooperative tracking error $\boldsymbol{\varepsilon}$ satisfies the identity $\boldsymbol{\varepsilon}^i = -\tilde{\boldsymbol{L}}(\boldsymbol{x}^i - \boldsymbol{x}_l^i)$, such that $\|\boldsymbol{x} - \boldsymbol{x}_l\| \leq \|\boldsymbol{\varepsilon}\|/\underline{\lambda}(\tilde{\boldsymbol{L}})$. □

Theorem 5.5 exhibits several intuitive properties. A high connectivity of the graph $\mathcal{G}$ augmented by the leader node, as measured through the eigenvalues of $\tilde{\boldsymbol{L}}$, implies comparatively large eigenvalues of $\boldsymbol{\Upsilon}$. This has the effect that a high connectivity allows a lower control gain k_c for ensuring positive definiteness of $\boldsymbol{\Upsilon}$. Moreover, since the ultimate tracking error bound (5.77) is reciprocal to $\underline{\lambda}(\boldsymbol{\Upsilon})$, it can be reduced by increasing the connectivity of the graph $\mathcal{G}$ augmented by the leader node. In addition to an increase in connectivity, a large control gain k_c is the main parameter for guaranteeing small tracking errors. In fact, arbitrarily small ultimate tracking error bounds $\upsilon(t)$ can be achieved through a suitable value of k_c. In order to see this, note that $\underline{\lambda}(\boldsymbol{Q})$ can be chosen arbitrarily large as positive definiteness of $\boldsymbol{\Upsilon}$ can always be ensured through a suitable control gain k_c. Moreover, it is trivial to check that $\underline{\lambda}(\boldsymbol{\Upsilon}) \to \frac{1}{2}\underline{\lambda}(\boldsymbol{Q})$ for $k_c \to \infty$. Since the ultimate bound is reciprocal to $\underline{\lambda}(\boldsymbol{\Upsilon})$, this implies that arbitrarily small tracking errors can be guaranteed.

Remark 5.6. *Theorem 5.5 extends the ideas of Corollary 3.2 to multi-agent systems. Therefore, the discussion in Remark 3.1 transfers to Theorem 5.5, such that we can investigate the effective behavior of the tracking error bound for small time intervals using the heuristic*

$$\tilde{\upsilon}(t) = \frac{\sqrt{M}\bar{\lambda}(\tilde{\boldsymbol{L}})\sqrt{\bar{\lambda}(\boldsymbol{P})}(1 + \|\tilde{\boldsymbol{\Theta}}\|_{\mathrm{F}})}{\underline{\lambda}(\boldsymbol{\Upsilon})\underline{\lambda}(\tilde{\boldsymbol{L}})}\left(\tilde{\eta}(\boldsymbol{x}_l(t)) + \bar{r}\right). \tag{5.89}$$

5.2.4. Numerical Evaluation

We evaluate our proposed approach in two simulations. First, we illustrate the prediction errors and error bounds of the proposed distributed GP approach. Afterward, we demonstrate the effectiveness of employing distributed predictions in a cooperative control scheme applied to a system with unmodeled nonlinearities.

Distributed Predictions for Dynamical Systems

In this section, we investigate the performance of the distributed GP approach proposed in (5.40), (5.41) for learning the nonlinear function

$$f(\boldsymbol{x}) = \sin(x_1) + \frac{1}{2(1 + \exp(\frac{x_2}{10}))}. \tag{5.90}$$

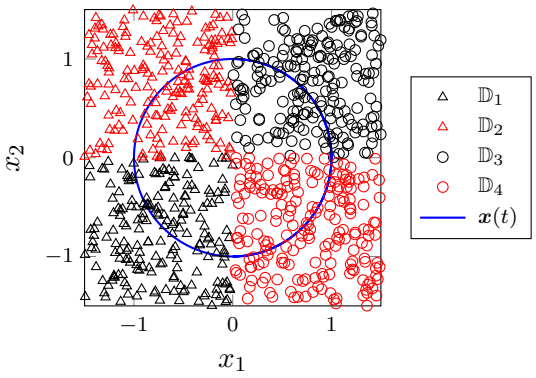

Figure 5.9.: Each Gaussian process is trained with data $\mathbb{D}_n$ from a single quadrant, while the trajectory $\boldsymbol{x}(\cdot)$ passes through all quadrants.

We consider a system with $M = 4$ GP models, each of which is based on a SE kernel (2.18) with signal standard deviation $\sigma_f = 0.5$ and length scales $l_i = 0.5$, $i = 1, 2$. Training data is uniformly distributed on the domain $[-1, 1] \times [-1, 1]$, and training targets are perturbed by zero mean Gaussian noise with $\sigma_{\text{on}} = 0.1$. Each of the GP models is trained using $N = 500$ training samples from a single quadrant, as illustrated in Fig. 5.9. For the distributed aggregation of the predictions, we employ the PoE scheme (5.6) and assume a circular communication graph described by the adjacency matrix

$$\boldsymbol{A}^{\text{ad}} = \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}. \tag{5.91}$$

The predictions are aggregated along the trajectory $\boldsymbol{x}(t) = \boldsymbol{x}_l(t)$ for $r_{\text{ref}}(t) = -\sin(t)$ using a prediction consensus gain $k_p = 1000$. The prediction error is uniformly bounded with $\delta = 0.1$ and $\tau = 0.01$, and the Lipschitz constants for $\psi_\mu(\cdot)$ and $\boldsymbol{\psi}_\omega(\cdot, \cdot)$ required for (5.44) are numerically approximated along $\boldsymbol{x}(t)$.

We compare the prediction error $|\hat{\mu}_m(\boldsymbol{x}(t)) - f(\boldsymbol{x}(t))|$ and error bound $\tilde{\eta}(\boldsymbol{x}(t))$ of the proposed method to the prediction error $|\mu_m(\boldsymbol{x}(t)) - f(\boldsymbol{x}(t))|$ and error bounds $\eta_{\text{loc}}(\boldsymbol{x}(t))$ of the standard approach employing only local predictions. As depicted in Fig. 5.10, the distributed predictions yield errors, which are almost identical to the best available individual prediction. While the individual predictions achieve these small errors only for a small period of time, the distributed predictions constantly maintain this high accuracy. A similar behavior can be observed for the uniform prediction error bounds, where the distributed GP approach guarantees an almost constant prediction accuracy, while the local error bounds strongly vary over time. This clearly demonstrates the advantages of distributed predictions.

Cooperative Tracking Control with Unmodeled Nonlinearities

In order to demonstrate the efficiency of the cooperative control law proposed in (5.55), we extend the previously introduced simulation setting to the control of a multi-agent system. For this purpose, we define $\beth_m(\boldsymbol{x}) = 0$ and $g_m(\boldsymbol{x}) = 1$ for $m = 1, \ldots, 4$ and $\boldsymbol{x} \in \mathbb{R}^2$. Moreover, we choose the diagonal matrix $\boldsymbol{B}^l = \text{diag}(1, 0, 1, 0)$, such that two follower agents are connected to the leader. In the feedback linearizing control law, we employ $\tilde{\theta} = \frac{7}{4}$,

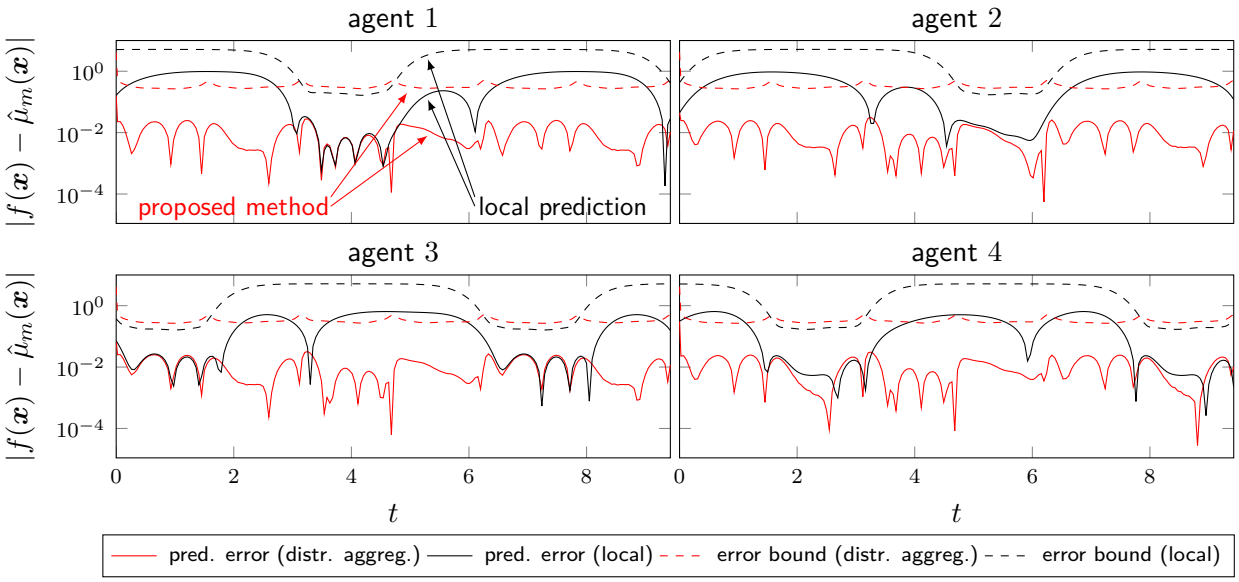

Figure 5.10.: The prediction errors $|\hat{\mu}_m(\boldsymbol{x}(t)) - f(\boldsymbol{x}(t))|$ observed when using the proposed distributed prediction aggregation method are small for the whole trajectory $\boldsymbol{x}(t)$, while the errors $|\mu_m(\boldsymbol{x}(t)) - f(\boldsymbol{x}(t))|$ resulting from the standard approach based on local predictions are only small for states $\boldsymbol{x}(t)$ close to the training data of the corresponding model. The theoretical error bounds for distributed predictions $\tilde{\eta}(\boldsymbol{x}(t))$ and individual predictions $\eta_{\text{loc}}(\boldsymbol{x}(t))$ exhibit the same behavior.

$k_c = 1000$, and use $k_p = 50000$ as gain for the prediction consensus. Finally, we determine the heuristic tracking error bound $\tilde{v}(\boldsymbol{x}(t))$ based on $Q = 600$.

We compare the simulation results of our proposed cooperative tracking control law employing distributed GP predictions to the same control law using only the individual predictions of the agent, no compensation of the unknown nonlinearity, and an exact model of $f(\cdot)$. As illustrated in Fig. 5.11, employing only the local model of each agent in the control law yields an improvement compared to the absence of any model. However, it performs significantly worse than the proposed control law with distributed predictions, which even achieves tracking errors almost identical to those of the control law with exact knowledge of $f(\cdot)$. Even though the corresponding tracking error bound heuristics $\tilde{v}(\boldsymbol{x}(t))$, which can be obtained for local predictions and exact model knowledge through a straightforward adaptation of [214], are conservative, they analogously reflect this behavior. These results underline the improvement in control performance resulting from the application of distributed predictions in cooperative tracking control.

5.3. Discussion

As we demonstrate in this chapter, the aggregation of GP models is an effective approach for addressing many problems in control systems. This is due to the straightforward extension of uniform error bounds for exact GP regression for many aggregation schemes, which allows the application of aggregated GP predictions instead of exact GP models when performance guarantees are required. When employing GP aggregations for online learning using iter-

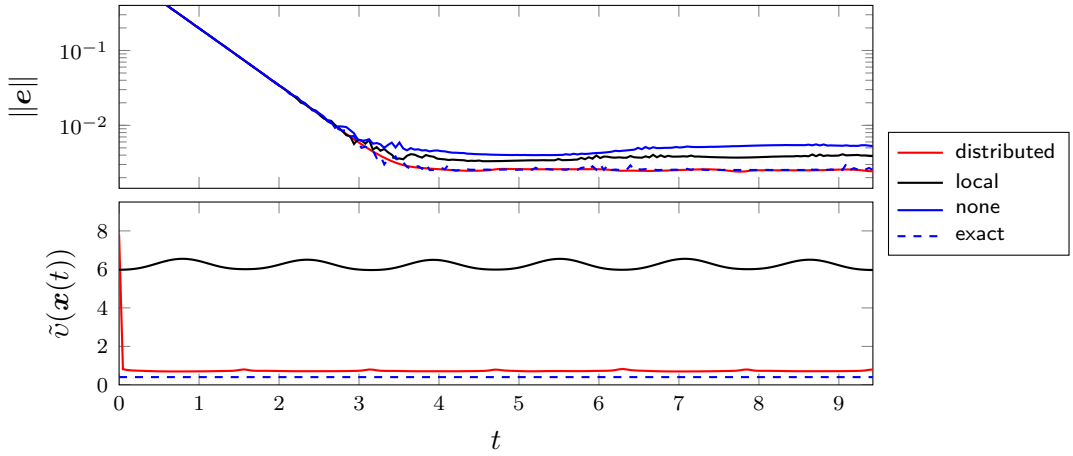

Figure 5.11.: Tracking errors $\|\boldsymbol{e}(t)\|$ and probabilistic error bound heuristic $\tilde{v}(t)$ for cooperative control with distributed learning are significantly smaller than with local learning and without learning.

atively constructed trees, as proposed in Section 5.1, they allow computationally efficient model updates and predictions. Thereby, learning in control loops at high rates with accuracy guarantees is enabled. Even though the computational efficiency achieved with the proposed method is often sufficient for the practical realization of online learning control, the necessary computation times are growing logarithmically. Hence, they will always exceed constraints on the computation time eventually, such that the method is not directly applicable to continual learning [216]. Moreover, the achievable uniform error bounds can be easily lower bounded due to the restriction to data sets with constant size in each local model. While we have found these limitations to be not practically relevant for our considered applications due to their relatively short execution time, they certainly need to be kept in mind when applying LoG-GPs in long-lasting experiments.

While LoG-GPs, as proposed in Section 5.1, address practically relevant issues for implementing GP-based learning for real-world control systems, they remain to have critical limitations. Most importantly, the memory complexity remains high. In fact, the results for learning from more than 500000 training samples using LoG-GPs require more than 50GB of RAM when implemented using MATLAB. Even though this requirement can be met in an academic setting for implementing online learning controllers, it probably exceeds any reasonable computing infrastructure available for controllers in commercial applications. In addition, suitable kernel hyperparameters are required a priori for achieving a high regression performance, which is at least a controversial assumption considering online learning problems. This requirement can be partially mitigated through gradient steps for the hyperparameters in each update iteration as demonstrated in Section 5.1.6, but we have found that a decent initial guess remains necessary to ensure a competitive regression performance. Finally, the simultaneous activity of multiple models ensured using (5.15) ensures the continuity of aggregated predictions, but the Lipschitz constant of aggregated mean functions can be high if few data samples are available at the boundary between local GP models. While this issue can often be mitigated by adding training samples to all active models during the model update step, an increased computational effort is required. Since these are only a few example limitations experienced when working with LoG-GPs, it is clear that many application-specific problems still need to be addressed. Therefore, the need for GP approx-

imations fitting the needs of specific control applications definitely remains an interesting problem for future research.

This is similarly true for the proposed distributed aggregation method in Section 5.2, which can merely be seen as the first step towards an extension of GP regression to multi-agent systems considering control-theoretic requirements. Our method provides a significant improvement over the state-of-the-art approach of employing individual GP model predictions in each agent, but it strongly relies on the convergence of each agent to the leader state. It seems reasonable that the proposed consensus-based distributed GP aggregation can be extended to approaches such as formation control and flocking, but a direct extension might potentially require an additional consensus state for each agent, which would significantly increase the communication load in the network. Furthermore, no improvement of the uniform error bound over the best local model is possible with the employed MoE and gPoE aggregation schemes, such that it remains unclear how efficient distributed online learning strategies can be designed. Therefore, this chapter clearly demonstrates the potential of GP model aggregations for learning in control systems, but additional research might be required to apply it to specific problems.

Architectures for Practical Control with Gaussian Processes

6.

When faced with the problem of realizing a learning-based control law in practice, the previous chapters provide a theoretical and algorithmic foundation for the design of specific components, but they do not give suggestions on the architectural composition in a real-world implementation. The design of such architectures is related to applied control problems, for which we often employ architectures composed of multiple cascaded control loops in practice [1]. This enables the consideration of both practical and theoretical needs in the implementation of sub-systems, e.g., by using different sampling rates, such that slowly evolving dynamics can be controlled with computationally more demanding control techniques, while parametric controllers are used for quickly changing processes.

For employing GP models in real-world control loops, such considerations must also be taken into account to address practical deviations from the previously considered theoretical settings. For example, the assumed noise-free state measurements are often not available since sensors are noisy and observers are frequently employed to estimate signals. Moreover, the computation time of GP model updates and predictions often keeps occasionally exceeding the sampling times of modern control loops despite the application of GP approximations such as LoG-GPs. Finally, small autonomous systems do often not only have computational limitations, but also a small memory. Therefore, they are not capable of storing large, memory-demanding GP models. Some of these challenges can be straightforwardly addressed, e.g., by adding a separate, slowly adapting layer for learning to the control architecture [217], and not all of them are always present, e.g., when control loops run at sufficiently low sampling rates to allow the direct integration of GP approximations [27, 191]. However, there exist many cases when this is not possible, such that a thorough development and analysis of implementation architectures for GP-based control is necessary.

In this chapter, we formalize three different architectures, demonstrate their practical applicability, and analyze the effect of design choices on previously derived guarantees. We start by considering a popular architecture currently used in experiments [26, 27, 28] in Section 6.1, which corresponds to the direct implementation of GP-based control laws. We refer to this approach as synchronous architecture in the following. Under the assumption of negligible computation times, the only practical deviation in this setting corresponds to the unavailability of noise-free state measurements. Therefore, this scenario allows us to straightforwardly analyze the effect of state measurement disturbances on derived tracking error bounds, which we show to be bounded. The practical effectiveness of the synchronous architecture is demonstrated in experiments with a robotic manipulator. Since the computation time of GP model evaluations restricts the sampling time of controllers in this architecture, we propose an asynchronous strategy for model evaluations in Section 6.2. In this novel architecture, the model-based control law holds the GP prediction until a new one is available, such that pure error feedback components of the controller can run at in-

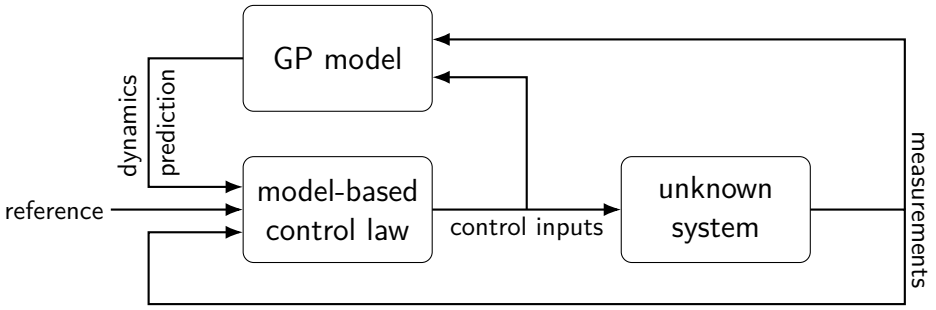

Figure 6.1.: Illustration of the synchronous online learning control architecture with GP models. The GP model is directly included in the control loop, such that it is evaluated at the same rate as the model-based control law.

dependent and higher update rates. We exemplarily investigate the effect of the delayed predictions on the derived tracking accuracy guarantees and demonstrate the applicability in a human-robot interaction experiment. Additionally, we address the high memory requirements of GP models by proposing a networked learning architecture in Section 6.3. This architecture exploits a reachability analysis to iteratively determine irrelevant data, which is sent to a cloud storage, such that local memory constraints can be satisfied. We show that the reduction of local training data does not affect the control performance guarantees and demonstrate the adherence to resource constraints in a robotic rehabilitation simulation. Finally, the results of this chapter are discussed in Section 6.4.

6.1. Synchronous Online Learning from Disturbed State Measurements

Currently, the predominantly used architecture for practical GP-based control relies on a synchronous evaluation of the GP model and the pure error feedback components of the control law [26, 27, 28] as illustrated in Fig. 6.1. Since the GP model is employed exactly like a parametric model, it resembles the natural extension of classical control architectures. This requires the sufficiently fast evaluation of the GP model to achieve the high sample rates of typical control loops, such that frequently GP approximations [28] or a time-varying data set of constant size are applied [26, 27]. However, the synchronous architecture effectively coincides with the idealized setting considered in the derivation of theoretical results. Therefore, it allows the straightforward evaluation of theoretical guarantees in experiments, which explains its popularity.

The major difference in the practical application of the synchronous architecture to the commonly considered theoretical setting lies in the absence of exact state measurements. Inaccuracies in state measurements can result from a variety of sources ranging from quantization errors in sensors [218] to errors caused by numerical differentiation [219] and observers [220]. Since disturbed measurements are a problem not limited to control applications, noisy states have been investigated in different scenarios. In early works, Gaussian distributed states are employed for the approximate propagation of uncertainty in discrete-time, GP state space models [221, 222]. A key idea in these approaches is the linearization around a nominal state, which can also be used for the development of approaches for dealing

with noise on training inputs [169, 223].

While these approaches require a modification of GP regression to deal with disturbed training inputs, we prove that even with the standard GP model, disturbed state measurements merely require a marginal modification of the uniform error bound derived in Corollary 2.3. Thereby, we show that our guarantees for GP-based control extend to real-world scenarios. In simulations, we investigate the dependency of the tracking error for an online learning GP-based controller on the magnitude of disturbances, which reveals an almost linear relationship between them. Therefore, this controller can safely be implemented on systems with decent measurement noise levels using the synchronous architectures, which we demonstrate in a real-world experiment with a robotic manipulator.

The remainder of this section is structured as follows. The problem of online learning control with GP models trained using data with disturbed state measurements is described in Section 6.1.1. In Section 6.1.2, we prove the boundedness of the tracking error despite disturbed state measurements. The dependency of the tracking accuracy on the disturbance magnitude is illustrated in Section 6.1.3 before the real-world applicability of the synchronous learning-based control architecture is demonstrated in an experiment in Section 6.1.4.

6.1.1. Problem Setting

For investigating the synchronous learning-based control architecture and the dependency of the employed GP models on state measurement disturbances, we consider again the setting in Section 3.1. Therefore, the system dynamics are given by

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}(u + f(\boldsymbol{x})), \tag{3.2 revisited}$$

where $f : \mathbb{X} \to \mathbb{R}$ is an unknown, scalar perturbation of the linear system. While the considered dynamics remain unchanged, we assume that the true state $\boldsymbol{x}(t)$ cannot be measured, but instead, only the disturbed signal

$$\tilde{\boldsymbol{x}}(t) = \boldsymbol{x}(t) + \boldsymbol{\epsilon}_x(t) \tag{6.1}$$

is available, where $\boldsymbol{\epsilon}_x : \mathbb{R}_{0,+} \to \mathbb{R}^{d_x}$ is an unknown disturbance. The disturbance function $\boldsymbol{\epsilon}_x(\cdot)$ can be the realization of a stochastic process representing measurement noise, or a deterministic function caused by, e.g., unobserved latent dynamics. In order to limit the effect of disturbed state measurement on the control performance, we assume bounded disturbances $\boldsymbol{\epsilon}_x(t)$ as formalized in the following.

Assumption 6.1. *The state measurement disturbance $\boldsymbol{\epsilon}_x : \mathbb{R}_{0,+} \to \mathbb{R}^{d_x}$ is bounded by a constant $\bar{\epsilon}_x$, i.e., $\|\boldsymbol{\epsilon}_x(t)\| \leq \bar{\epsilon}_x$ for all $t \in \mathbb{R}_{0,+}$.*

While this assumption requires bounded disturbances $\boldsymbol{\epsilon}_x(t)$, they can be arbitrarily large. Therefore, this assumption admits practically relevant effects, such as measurement noise and disturbances caused by sensor offsets, such that it does not severely restrict the applicability of the derived theory to real-world applications.

Due to the restriction to disturbed measurements $\tilde{\boldsymbol{x}}(t)$, the control law (3.4) cannot be directly employed, but instead, we have to use

$$u(t) = \boldsymbol{\theta}^T(\tilde{\boldsymbol{x}}(t) - \boldsymbol{x}_{\mathrm{ref}}(t)) + r_{\mathrm{ref}}(t) - \hat{f}(\tilde{\boldsymbol{x}}(t)). \tag{6.2}$$

In order to infer the model $\hat{f} : \mathbb{R}^{d_x} \to \mathbb{R}$ of the unknown input perturbation $f(\cdot)$, we employ a time-triggered online learning scheme with sampling time $T_s \in \mathbb{R}_+$ as presented in Section 4.3.2. This leads to a time-varying data set

$$\mathbb{D}(t) = \left\{ \tilde{\boldsymbol{x}}^{(n)} = \boldsymbol{x}(nT_s) + \boldsymbol{\epsilon}_x(t^{(n)}), y^{(n)} = f(\boldsymbol{x}^{(n)}) + \epsilon_y^{(n)} \right\}_{n=1}^{\left\lfloor \frac{t}{T_s} \right\rfloor}, \tag{6.3}$$

which contains disturbed state measurements $\tilde{\boldsymbol{x}}^{(n)}$ in contrast to the data set (4.81) originally considered in Section 4.3.2. For a consistent analysis, we also assume bounded target noise denoted as $\epsilon_y^{(n)}$ for clarity of exposition, i.e., we require Assumption 2.3. If similar sensors or techniques as for state measurements are employed for the target observations, this is a natural requirement. Using the data set $\mathbb{D}(t)$, we can update a GP model online and use the mean function $\mu(\cdot)$ as model $\hat{f}(\cdot)$.

In a practical implementation, performing these computations takes time. Due to the consideration of the synchronous learning-based control architecture, we assume that the necessary computation time is small in comparison to the sampling time of the control system, such that we can ignore it in practice. This is formally stated in the following assumption.

Assumption 6.2. *The computation times of GP model updates and evaluations are negligible.*

Using GP approximations such as the proposed LoG-GP method in Section 5.1, the computation times required for model updates and evaluations can be reduced to a few milliseconds with suitable hardware. Therefore, this assumption is not restrictive for moderately fast and slow control loops operating at sampling rates lower a few hundred Hz.

Based on this assumption and the control law (6.2), we consider the problem of extending the previously derived tracking error bound in Theorem 4.4 to the synchronous learning-based control architecture with noisy state measurements. This requires the derivation of an upper bound for the solution $\boldsymbol{e}(\cdot)$ of the closed-loop error dynamics given by

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})\boldsymbol{e} + \boldsymbol{b}(f(\boldsymbol{x}) - \hat{f}(\tilde{\boldsymbol{x}})) + \boldsymbol{b}\boldsymbol{\theta}^T\boldsymbol{\epsilon}_x, \tag{6.4}$$

where $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \boldsymbol{A} - \boldsymbol{b}\boldsymbol{\theta}^T$.

6.1.2. Learning Control with Disturbed State Measurements

Since the training data set $\mathbb{D}(t)$ defined in (6.3) does not have noise-free training inputs as considered for the derivation of uniform error bounds in Chapter 2, our previous results cannot be directly used to analyze the synchronous learning-based control architecture with disturbed state measurements $\tilde{\boldsymbol{x}}(t)$. Therefore, we first extend the uniform error bound in Corollary 2.3 to the considered setting before we investigate the tracking error itself.

For the extension of Corollary 2.3, the restriction to bounded disturbances $\boldsymbol{\epsilon}_x(t)$ is crucial, since it directly allows us to project the disturbance of training inputs to the targets for continuous functions $f(\cdot)$. This is exploited for the derivation of the following result.

Lemma 6.1. *Consider an unknown, Lipschitz continuous function $f(\cdot) \in \mathcal{H}_k^{\mathbb{S}}$ satisfying Assumption 2.4 and a prior Gaussian process $\mathcal{GP}(0, k(\cdot,\cdot))$. Moreover, assume that the*

noise $\boldsymbol{\epsilon}_x(t^{(n)})$, $\epsilon_y^{(n)}$, $n = 1, \ldots, N$, *satisfies Assumption 2.3 and Assumption 6.1. Then, the posterior mean function* $\mu(\cdot)$ *defined in* (2.26) *admits a uniform error bound*

$$\eta(\boldsymbol{x}) = \beta\sigma(\boldsymbol{x}) \tag{6.5}$$

for

$$\beta = \Gamma + \frac{\sqrt{N}(\bar{\epsilon}_y + L_f\bar{\epsilon}_x)}{\sigma_{\mathrm{on}}}\sqrt{\|\boldsymbol{K}(\boldsymbol{K} + \sigma_{\mathrm{on}}^2\boldsymbol{I}_N)^{-1}\|} \tag{6.6}$$

with probability 1 *on the compact set* $\mathbb{S} \subset \mathbb{R}^{d_x}$.

Proof. In order to derive a uniform error bound with noise on the training inputs, we project the noise $\boldsymbol{\epsilon}_x^{(n)}$ on the training targets $y^{(n)}$. Due to the Lipschitz continuity of $f(\cdot)$, for each $\boldsymbol{\epsilon}_x^{(n)}$, there exists a value $\tilde{\epsilon}_y^{(n)}$ with

$$|\tilde{\epsilon}_y^{(n)}| \leq L_f\|\boldsymbol{\epsilon}_x(t^{(n)})\| + |\epsilon_y^{(n)}| \tag{6.7}$$

such that

$$f\left(\boldsymbol{x}^{(n)}\right) + \epsilon_y^{(n)} = f(\tilde{\boldsymbol{x}}^{(n)}) + \tilde{\epsilon}_y^{(n)}. \tag{6.8}$$

This allows us to equivalently consider training pairs $(\boldsymbol{x}^{(n)}, f(\tilde{\boldsymbol{x}}^n) + \tilde{\epsilon}_y^{(n)})$, which exhibit measurement noise only on the training targets. Therefore, we can directly employ Corollary 2.3 to obtain a uniform error bound since $\tilde{\epsilon}_y^{(n)} \leq L_f\bar{\epsilon}_x + \bar{\epsilon}_y$. □

While this lemma requires the restriction to Lipschitz continuous functions, this is not a severe limitation since a bounded RKHS norm Γ as required for RKHS-based uniform error bound directly yields Lipschitz constants for many kernels $k(\cdot,\cdot)$, [224]. Therefore, this assumption does not pose a relevant additional restriction but enables the straightforward projection of training input disturbances $\boldsymbol{\epsilon}_x(t^{(n)})$ on the training targets. Since this projection is generally biased, a growing data set is not guaranteed to reduce the error bound, and a stationary learning error can remain. This is reflected by the $\sqrt{N}$ factor in (6.6), which causes a comparatively fast-growing scaling factor β in (6.5). Therefore, the choice of suitable training inputs as discussed in Section 4.2 becomes crucial with disturbed state measurements $\tilde{\boldsymbol{x}}(t)$ in order to achieve a small posterior standard deviation $\sigma(\cdot)$ where necessary.

Since the uniform error bound in Lemma 6.1 keeps the same structure as the bounds considered in previous chapters, the tracking error analysis presented in Section 4.3.2 can be straightforwardly extended to the synchronous learning architecture with disturbed state measurements $\tilde{\boldsymbol{x}}(t)$. This is shown in the following proposition.

Proposition 6.1. *Consider a system* (3.2) *satisfying Assumption 3.1, to which a control law* (6.2) *is applied to track a reference trajectory with bounded derivative* $\dot{\boldsymbol{x}}_{\mathrm{ref}}(\cdot)$. *Assume that a Gaussian process* $\mathcal{GP}(0, k(\cdot,\cdot))$ *with stationary, Lipschitz continuous kernel* $k(\cdot,\cdot)$ *is employed to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *of the unknown input perturbation* $f(\cdot)$ *satisfying Assumption 2.4 online from a data set* (6.3), *for which Assumptions 2.3 and 6.1 hold. If the computation times satisfy Assumption 6.2 and* $\mathbb{B}_{\bar{v}}(\boldsymbol{x}_{\mathrm{ref}}(t)) \in \mathbb{S}$ *holds for all* $t \in [0, T]$, $T \in \mathbb{R}_{0,+}$, *where*

$$\bar{v} = -\frac{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}\left(L_\eta F T_s + \eta_{T_s} + (L_\mu + \|\boldsymbol{b}\boldsymbol{\theta}^T\|)\bar{\epsilon}_x\right) \tag{6.9}$$

for η_{T_s} defined in (4.85), then, the tracking error is bounded by $\upsilon(\cdot) = \bar{\upsilon}$ during the time interval $[0, T]$ for initial states $\mathbb{X}_0 = \{\boldsymbol{x}_{\mathrm{ref}}(0)\}$.

Proof. Due to the Lipschitz continuity of the kernel $k(\cdot, \cdot)$, it follows from Lemma 2.2 that the GP mean $\mu(\cdot)$ is Lipschitz continuous. Therefore, we obtain

$$|f(\boldsymbol{x}) - \mu(\tilde{\boldsymbol{x}})| \leq |f(\boldsymbol{x}) - \mu(\boldsymbol{x})| + L_\mu \bar{\epsilon}_x. \tag{6.10}$$

Since the assumptions of Lemma 6.1 are satisfied, this directly implies

$$|f(\boldsymbol{x}) - \mu(\tilde{\boldsymbol{x}})| \leq \eta(\boldsymbol{x}) + L_\mu \bar{\epsilon}_x. \tag{6.11}$$

Since we can additionally bound the effect of the state measurement disturbance $\boldsymbol{\epsilon}_x(\cdot)$ on the control law (6.2) by

$$|\boldsymbol{b}\boldsymbol{\theta}^T \boldsymbol{\epsilon}_x(t)| \leq \|\boldsymbol{b}\boldsymbol{\theta}^T\| \bar{\epsilon}_x \tag{6.12}$$

due to Assumption 6.1, we obtain

$$\|\boldsymbol{e}(t)\| \leq \|\boldsymbol{U}\| \|\boldsymbol{U}^{-1}\boldsymbol{b}\| \int_0^t \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))(t-t')} \left(\eta(\boldsymbol{x}(t')) + (L_\mu + \|\boldsymbol{b}\boldsymbol{\theta}^T\|) \bar{\epsilon}_x \right) \mathrm{d}t'. \tag{6.13}$$

analogously to (4.89). By bounding $\eta(\boldsymbol{x}(t))$ for time-triggered data measurements defined in (6.3) using (4.90), the result directly follows from the proof of Theorem 4.4. □

Due to the structure of the tracking error bound (6.9), Proposition 6.1 can be interpreted as the analog of Theorem 4.4 for disturbed state measurements $\tilde{\boldsymbol{x}}(t)$. A direct comparison reveals merely one critical difference: The measurement disturbance $\boldsymbol{\epsilon}_x(\cdot)$ has a direct impact on the feedback control, such that we have an additional term $(L_\mu + \|\boldsymbol{b}\boldsymbol{\theta}^T\|)\bar{\epsilon}_x$ in (6.9). This term prevents the reduction of the tracking error bound $\bar{\upsilon}$ to arbitrarily small values by simply increasing the control gains $\boldsymbol{\theta}$ since an increase of $-\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))$ is accompanied by an increase of $\|\boldsymbol{b}\boldsymbol{\theta}^T\|$. Therefore, a trade-off between noise amplification and reduction of the effect of unmodeled nonlinearities is required, which is a well-known necessity in the practical application of control laws. While the GP model is also affected by the measurement disturbance $\boldsymbol{\epsilon}_x(\cdot)$, the Lipschitz constant L_μ is not particularly affected by state measurement disturbance $\boldsymbol{\epsilon}_x(\cdot)$. Hence, an improvement of the model accuracy and, consequently, a decrease of the uniform error bound η_{T_s} almost immediately reduces the tracking error. Thereby, the GP model enables the reduction of the tracking errors to values which would not be possible for a controller without nonlinearity compensation regardless of the control gains $\boldsymbol{\theta}$. This clearly motivates the practical application of GP models in real-world problems, which can be straightforwardly achieved using the synchronous learning-based control architecture.

6.1.3. Numerical Evaluation

To investigate the performance of GP-based control with disturbed state measurements, we revisit the setting in Section 4.3.5. Therefore, our closed-loop dynamics are described by

$$\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \begin{bmatrix} 0 & 1 \\ -k_c\tilde{\theta} & -k_c - \tilde{\theta} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.80 revisited}$$

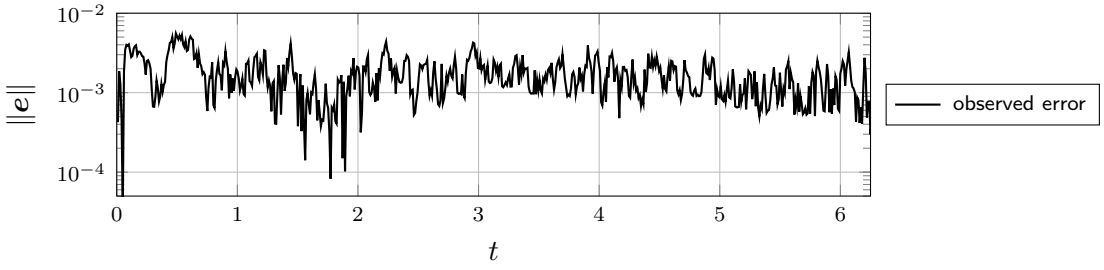

Figure 6.2.: Examplary evolution of the tracking error norm $\|\boldsymbol{e}(\cdot)\|$ with time-triggered online learning from states with measurement disturbance bounded by $\bar{\epsilon}_x = 0.01$.

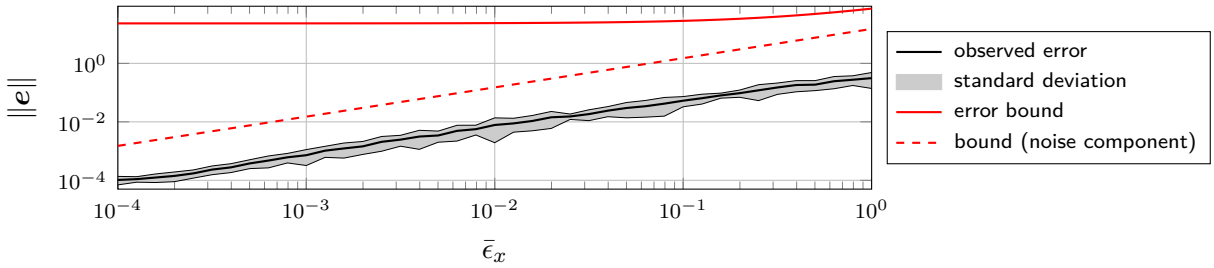

Figure 6.3.: Maximum tracking error and its bound $\bar{v}$ in dependency of the state measurement noise bound $\bar{\epsilon}_x$ for time-triggered online learning.

with a nonlinear input perturbation $f(\boldsymbol{x}) = 1 - \sin(2x_1) + {}^{1}/_{(1+\exp(-x_2))}$. The control gains are set to $k_c = 10$ and $\tilde{\theta} = 5$, a sinusoidal trajectory with period time 2π is tracked, and training data is generated and processed online with a sample time of $T_s = 0.01$. In order to straighten the evaluation, we set $\bar{\epsilon}_x = \bar{\epsilon}_y$ and sample disturbances from uniform distributions $\epsilon_y \sim \mathcal{U}([-\bar{\epsilon}_y, \bar{\epsilon}_y])$, $\boldsymbol{\epsilon}_x \sim \mathcal{U}([-\bar{\epsilon}_x, \bar{\epsilon}_x]^2)$. The RKHS norm is approximated as proposed in [74] and multiplied with 1.5 to obtain the upper bound Γ required for the GP uniform error bound. Moreover, the direct dependency of the uniform error bound is partially avoided by choosing the observation noise standard deviation parameter σ_{on} in GP regression as $\sigma_{\text{on}} = \bar{\epsilon}_x(L_f + 1)\sqrt{N(T)}$.

An exemplary evolution of the tracking error realized by the learning-based synchronous control architecture in this setting is illustrated for $\bar{\epsilon}_x = 0.01$ in Fig. 6.2. Due to the disturbed state measurements, the tracking error exhibits a clearly visible noisy behavior, but it maintains a low value on average in the depicted period of the reference trajectory. When investigating the dependency of the maximum tracking error on the magnitude of the measurement disturbance $\bar{\epsilon}_x$, we average over 10 runs to account for the randomness caused by the uniformly sampled measurement disturbances. The resulting curve, which is depicted in Fig. 6.3, shows an increasing behavior closely following the growth rate of the noise component

$$\bar{v}_{\text{noise}} = -\frac{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\text{cl}}(\boldsymbol{\theta}))}(L_\mu + \|\boldsymbol{\theta}\|)\bar{\epsilon}_x \tag{6.14}$$

of the tracking error bound (6.9). Since this relationship is linear in $\bar{\epsilon}_x$, a small measurement disturbance bound $\bar{\epsilon}_x$ is practically almost negligible, while large values also lead to a low tracking accuracy in the simulations. Therefore, it is possible to safely apply GP models in

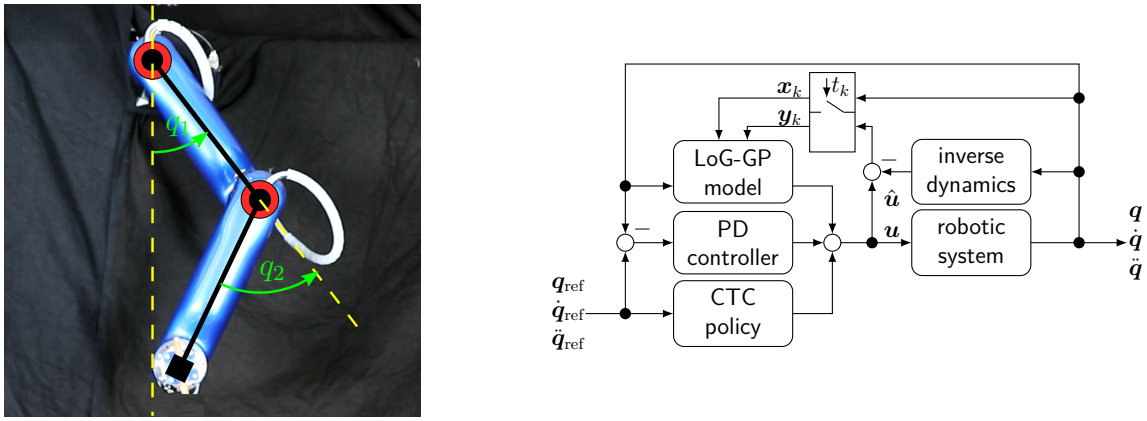

Figure 6.4.: Left: the robotic manipulator CARBO used for demonstrating the practical applicability of the synchronous online learning architecture in control problems. The graphic is adopted from [27]. Right: schematic of the used synchronous learning-based control architecture. A combination of CTC policy, PD controller, and GP-based compensation of unknown nonlinearities is employed for control, while simultaneously, a mapping between joint angles, velocities, and accelerations to torques is learned.

practical control problems without a significant impact on derived theoretical guarantees, as long as existing state measurement disturbances are small.

6.1.4. Experimental Demonstration in Control of Robotic Manipulators

We demonstrate the real-world applicability of the synchronous learning-based control architecture by implementing it on the CARBO robotic manipulator illustrated on the left side of Fig. 6.4. For this purpose, we employ the augmented computed torque control (CTC) law proposed in [34] to track a sinusoidal reference trajectory. This leads to the architecture illustrated on the right side of Fig. 6.4, which employs a combination of CTC policy, PD controller, and GP-based compensation of unknown nonlinearities. The CTC policy is defined using approximate robot parameters following the standard approach in robotics control [225]. We define the PD controller with proportional gain $\boldsymbol{k}_p = [4\ 3]^T$ and differential gain $\boldsymbol{k}_d = [20\ 20]^T$. The model is learned online using LoG-GPs with $\bar{N} = 50$ as presented in Section 5.1 from joint angles, velocities, and accelerations as training inputs and torque errors as training targets. The hyperparameters of the LoG-GP model are tuned offline using log-likelihood maximization from a previously recorded data set generated by tracking the considered reference trajectory. Since the learning-based controller is implemented using a synchronized architecture, the CTC policy, the PD controller, and the LoG-GP model updates and evaluations are all computed at a rate of 200Hz.

An example trajectory of the two joint angles and velocities of the CARBO robotic manipulator are depicted together with the corresponding reference trajectories in Fig. 6.5. While the measured joint angles look smooth, bounded measurement noise is clearly visible in the angular velocities due to numerical differentiation. Therefore, the CARBO robot control problem reflects the problem setting formulated in Section 6.1.1. Due to the very small

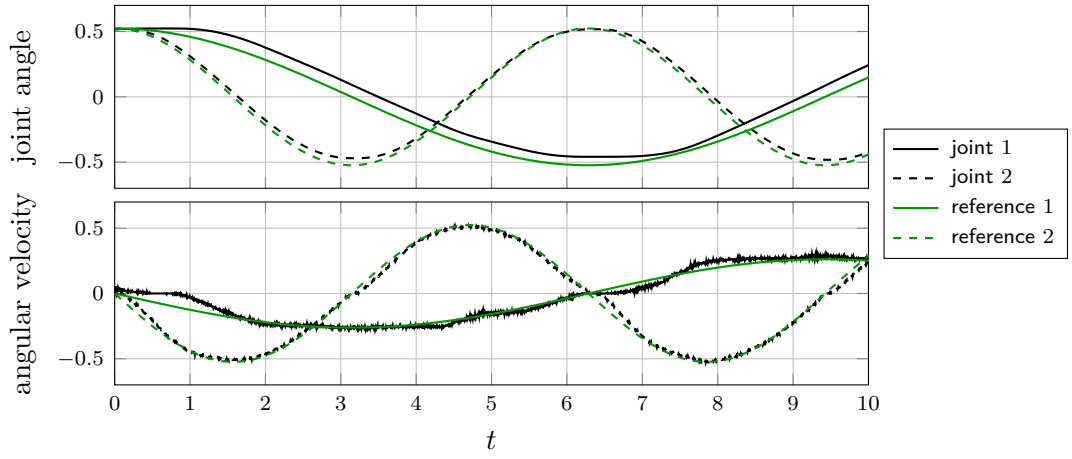

Figure 6.5.: Measured joint angles and angular velocities together with their reference values for the CARBO manipulator controlled by a CTC controller augmented by an online learned LoG-GP model. Even though measurement noise in the velocity measurements is clearly visible, the tracking of the reference trajectory is approximately achieved.

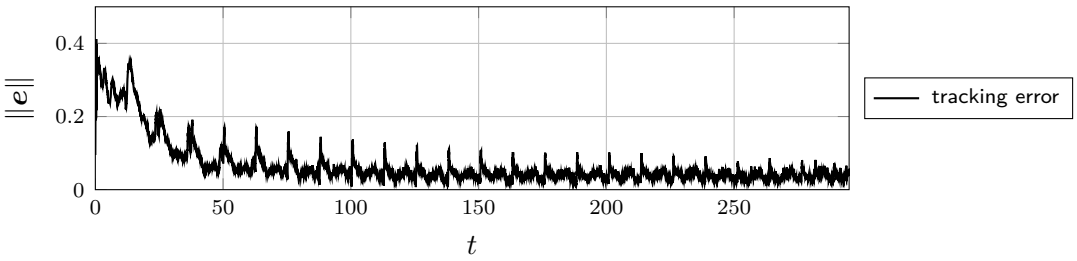

Figure 6.6.: Due to the online learning of LoG-GP models for control, the tracking error of the CARBO manipulator reduces over time, but the state measurement noise prevents its vanishment.

control gains $\boldsymbol{k}_p$ and $\boldsymbol{k}_d$, a remaining tracking error is visually perceivable for the angles and velocities of both joints, but the synchronous online learning control architecture is generally capable of steering the robot along the reference trajectory.

While the sampling rate of 200Hz limits the achievable tracking accuracy using the online learning paradigm, the repetitive behavior of sinusoidal reference trajectories effectively allows a periodic improvement, similar as discussed in Section 4.3.4 for episodic learning. This effect can indeed be observed when running the experiment for a longer time, as illustrated in Fig. 6.6. In the beginning, the imprecise LoG-GP model is not capable of compensating for the unknown nonlinearity well. After $\approx$ 50s, which corresponds to 4 periods, the model accuracy has significantly improved, and decent tracking performance can already be achieved. While the tracking error keeps getting smaller over time afterward, the decrease in every period shrinks. However, due to the state measurement disturbances, it does not vanish but instead converges to a small value.

Despite this apparent lower bound on the achievable tracking error through the GP-based compensation of unknown nonlinearities, the synchronous learning-based control architecture enables an eventual improvement over a controller without a GP model but significantly

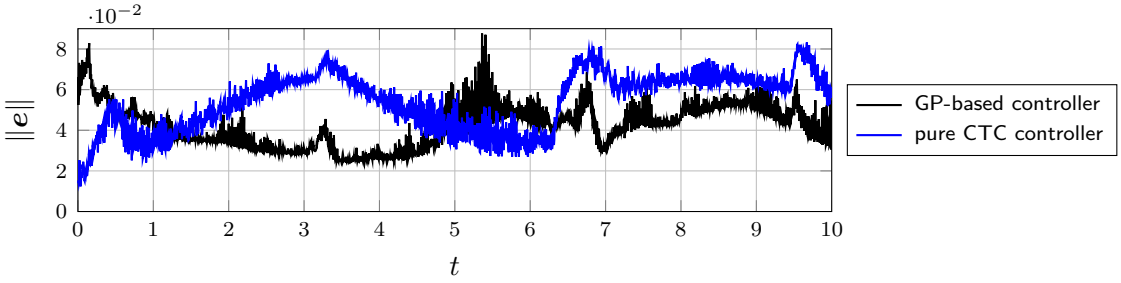

Figure 6.7.: When learning a model online using LoG-GPs, the augmented CTC controller achieves a better tracking accuracy than a pure CTC controller with significantly higher gains after a certain number of periods.

higher control gains, as depicted in Fig. 6.7. This figure depicts the tracking error after 22 periods of online learning in comparison to a control law without a GP model but with increased proportional gains $\boldsymbol{k}_p = [40\ 30]^T$. While there are short intervals of time during which the tracking error with the online learning control law is larger, the controller without learning exhibits a lower tracking accuracy on average. Therefore, our experimental evaluation demonstrates the practical applicability and effectiveness of the synchronous learning-based control architecture.

6.2. Asynchronous Online Learning with Computational Delays

While the synchronous learning control architecture provides a straightforward approach for evaluating theoretical results in real-world experiments, it crucially requires the computation of GP model updates and evaluations at the same rate as other parts of the control loop. This is a challenging and limiting requirement since GP models are computationally demanding. Even though it is possible to reduce their complexity, it remains questionable if the rates of modern control loops, which often require sampling times as low as 0.1ms, can ever be achieved. Moreover, the computation time for different operations of GP models strongly differs as updates generally exhibit a significantly higher complexity than predictions. In addition, GP approximations such as LoG-GPs can cause occasional spikes in the computation times. This can be catastrophic in control loops if GP model computations lead to the delayed application of control inputs.

This is a problem not limited to GP models but can also be found with other learning techniques. For example, when employing deep learning [17] in control, model updates take a considerable amount of time. In order to enable online updates despite this limitation, they can be executed in separately spawned processes [226], such that they run asynchronously. Since GP models additionally suffer from comparatively slow model evaluations, this idea can be extended to compute GP predictions asynchronously as fast as possible [227]. While these examples demonstrate the awareness of the problem, the existing literature does not go beyond conceptual solutions. Thus, the formalization of a clear architecture addressing the problem of high computation times often coming along with learned models is an open issue. Moreover, theoretical foundations for the validity of such proposed asynchronous learning approaches are missing.

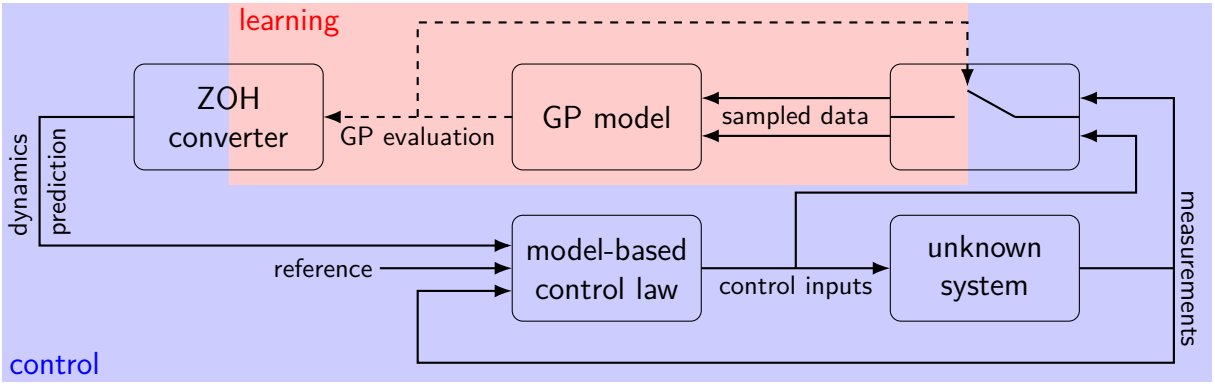

Figure 6.8.: Illustration of the asynchronous online learning control architecture with GP models. The necessary computations for updating and evaluating GP models are decoupled from the control loop through a zero-order hold converter and a sampling approach. This prevents large computation times in the GP model from slowing down the remaining control loop.

We address this gap by proposing the online learning control architecture depicted in Fig. 6.8. This architecture decouples the necessary computations for GP models from the remaining control loop in order to allow high-frequency updates for the pure error feedback components. In order to achieve this, GP model evaluations are kept constant through a zero-order hold (ZOH) converter until a new prediction has been computed. Moreover, new test and training inputs are only admitted when the previous update and evaluation computations have been finished, such that potential older data is discarded. While this can lead to delayed model evaluations, we exemplarily show that the effect of this delay on the tracking error can be bounded. Numerical simulations and real-world experiments involving human participants demonstrate the practical effectiveness of the proposed asynchronous online learning architecture.

The remainder of this section is structured as follows. In Section 6.2.1, the problem setting with non-negligible computational delays is formalized. The tracking error bound for linear systems with unknown input perturbation is adapted to the asynchronous setting in Section 6.2.2. We numerically investigate the effect of the computational delay on the tracking accuracy achieved by the asynchronous online learning architecture in Section 6.2.3. In Section 6.2.4, which is based on [40], we finally demonstrate the practical robustness and effectiveness of this architecture in a real-world experiment that employs human participants as unknown disturbances.

6.2.1. Problem Setting

In order to analyze the tracking error guaranteed when using the asynchronous online learning control architecture, we exemplarily consider the setting in Section 3.1, such that the system dynamics can be described by

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}(u + f(\boldsymbol{x})), \qquad \text{(3.2 revisited)}$$

where $f : \mathbb{X} \to \mathbb{R}$ is an unknown, scalar perturbation of the linear system. For compensating this nonlinearity using a control law, we learn a GP model online from a data set

$$\mathbb{D}(t) = \left\{ \boldsymbol{x}^{(n)} = \boldsymbol{x}(nT_s), y^{(n)} = f(\boldsymbol{x}^{(n)}) + \epsilon^{(n)} \right\}_{n=1}^{\left\lfloor \frac{t}{T_s} \right\rfloor}, \qquad \text{(4.81 revisited)}$$

generated using a time-triggered sampling scheme. Since the computation of GP model updates and evaluations can take a considerable amount of time, the GP mean function $\mu(\boldsymbol{x}(t))$ for a state $\boldsymbol{x}(t)$ is generally not available at time t, but at a later time $t + T_m^c > t$, where $T_m^c \in \mathbb{R}_{0,+}$ denotes the m-th computation time. Moreover, we consider a single computing unit, such that no other operations can be performed with the GP model during the interval $[t, t + T_m^c)$. Due to the asynchronous architecture, this requires us to adapt the control law (3.4) to obtain

$$u(t) = -\boldsymbol{\theta}^T(\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{ref}}(t)) + r_{\mathrm{ref}}(t) - \mu\left(\boldsymbol{x}\left(\sum_{m=1}^{M(t)} T_m^c\right)\right), \tag{6.15}$$

where

$$M(t) = \max_{M \in \mathbb{N}} M - 1 \tag{6.16}$$

$$\text{such that } \sum_{m=1}^{M(t)} T_n^c \leq t \tag{6.17}$$

denotes the previous number of numerical operations, e.g., model evaluations, performed with the GP model. The sum $\sum_{m=1}^{M(t)} T_m^c$ in (6.15) defines the time at which the computation of the currently used prediction has been started. Therefore, GP model evaluations are only available at discrete time instances, whereas the linear feedback components are continuously evaluated. This corresponds to a practical scenario in which the control loop is operated at sampling rates that are significantly higher than admissible for online learning with GPs.

In order to quantify the effect of non-negligible computation times T_m^c on the tracking accuracy in the asynchronous control architecture, we make the following assumptions

Assumption 6.3. *While GP model updates and evaluations are computed, no other operations can be executed with the GP model. The execution time $T_m^c \in \mathbb{R}_+$ of these $m \in \mathbb{N}$ operations is upper bounded by a constant $\bar{T}^c \in \mathbb{R}_+$, i.e., $T_m^c \leq \bar{T}^c$ for all $m \in \mathbb{N}$.*

Assumption 6.4. *The sampling time for online data generation and GP model updates is larger than the computation time, i.e., $T_s > \bar{T}^c$.*

Assumption 6.3 ensures that the GP model evaluation is available after a finite amount of time, which is a natural requirement for any algorithm. Since this assumption does not pose an upper bound on the admissible computation times, it is not restrictive in practice. Assumption 6.4 allows that all training samples $(\boldsymbol{x}^{(n)}, y^{(n)})$ can be processed and used for GP model updates since it ensures that there is at least one time instant t, at which numerical operations are performed with the GP model, between two sample times $t^{(n)}$, $t^{(n+1)}$. If this assumption is not satisfied, data can be dropped, which effectively leads to a reduced sampling time T_s in the time-triggered data generation scheme. Therefore, the presented subsequent analysis can simply be executed using the effective sampling time, such that Assumption 6.4 is not restrictive.

Based on these assumptions, we consider the problem of extending the tracking accuracy guarantees derived in Theorem 4.4 to the asynchronous online learning architecture. Therefore, we need to derive an upper bound for the solutions $\boldsymbol{e}(t)$ of the closed-loop error dynamics

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})\boldsymbol{e} + \boldsymbol{b}\left(f(\boldsymbol{x}) - \mu\left(\boldsymbol{x}\left(\sum_{m=1}^{M} T_m^c\right)\right)\right), \tag{6.18}$$

where $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \boldsymbol{A} - \boldsymbol{b}\boldsymbol{\theta}^T$.

6.2.2. Accuracy Guarantees with Delayed Predictions

Since the asynchronous evaluation and update of GP models merely lead to a bounded delay in the nonlinearity compensation of the control law (6.15), we can directly employ the derived bound on the state derivative $\dot{\boldsymbol{x}}(t)$ in Lemma 5.2 to quantify the caused model inaccuracy. This approach is employed in the following proposition.

Proposition 6.2. *Consider a system* (3.2) *satisfying Assumption 3.1, to which a control law* (6.15) *is applied to track a reference trajectory with bounded derivative* $\dot{\boldsymbol{x}}_{\mathrm{ref}}(\cdot)$. *Assume that a Gaussian process* $\mathcal{GP}(0, k(\cdot,\cdot))$ *with stationary kernel* $k(\cdot,\cdot)$ *is employed to learn a model* $\hat{f}(\cdot) = \mu(\cdot)$ *of the Lipschitz continuous input perturbation* $f(\cdot)$ *online from a data set* (4.81)*, such that Assumption 3.2 holds on a compact set* $\mathbb{S} \subset \mathbb{X}$ *with a uniform error bound* $\eta(\cdot)$ *which admits a Lipschitz constant* L_η. *If the computation times* T_m^c *and the sampling times* T_s *satisfy Assumptions 6.3 and 6.4, respectively, and* $\mathbb{B}_{\bar{\upsilon}}(\boldsymbol{x}_{\mathrm{ref}}(t)) \in \mathbb{S}$ *holds for all* $t \in [0, T]$, $T \in \mathbb{R}_{0,+}$, *where*

$$\bar{\upsilon} = -\frac{\|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\|}{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))}\left((L_\eta T_s + 2L_f\bar{T}^c)F + \eta_{T_s}\right), \tag{6.19}$$

then, a probabilistic tracking error bound $\upsilon(\cdot) = \bar{\upsilon}$ *is ensured during the time interval* $[0, T]$ *for initial states* $\mathbb{X}_0 = \{\boldsymbol{x}_{\mathrm{ref}}(0)\}$.

Proof. It is straightforward to see that the definition of $M(t)$ in (6.16) and (6.17) together with Assumption 6.3 implies that

$$t - \sum_{m=1}^{M(t)} T_m^c \leq 2\bar{T}^c. \tag{6.20}$$

Therefore, it follows from a slight adaptation of Lemma 5.2 to the control law (6.15) that

$$\left| f\left(\boldsymbol{x}\left(\sum_{m=1}^{M(t)} T_m^c\right)\right) - f(\boldsymbol{x}(t))\right| \leq 2L_f F\bar{T}^c. \tag{6.21}$$

This directly yields

$$\|\boldsymbol{e}(t)\| \leq \|\boldsymbol{U}\|\|\boldsymbol{U}^{-1}\boldsymbol{b}\| \int_0^t \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))(t-t')}\left(\eta\left(\sum_{m=1}^{M(t)} T_m^c\right) + 2L_f F\bar{T}^c\right)\mathrm{d}t'. \tag{6.22}$$

analogously to (4.89). Due to Assumption 6.4, at most one training sample is added between two computation operations, such that

$$\sum_{m=1}^{M(t)} T_m^c - t^{(n)} \leq T_s \tag{6.23}$$

holds for the sampling times $t^{(n)}$. This implies

$$\|\boldsymbol{e}(t)\| \leq \|\boldsymbol{U}\| \|\boldsymbol{U}^{-1}\boldsymbol{b}\| \int_0^t \mathrm{e}^{\bar{\lambda}(\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}))(t-t')} \left(\eta\left(t^{(n)}\right) + L_\eta F T_s + 2L_f F \bar{T}^c\right) \mathrm{d}t', \tag{6.24}$$

such that the result directly follows from the proof of Theorem 4.4. □

This proposition shows that the computational delays T_m^c have a similar impact on the tracking accuracy guarantees in the asynchronous online learning control architecture as the sampling time for model updates T_s. Moreover, we can observe that for a vanishing upper bound $\bar{T}^c$, (6.19) asymptotically converges to the tracking error bound for the theoretical setting derived in Theorem 4.4. This intuitively corresponds to the fact that $\bar{T}^c \to 0$ implies negligible computation times, which is a key requirement for synchronous learning as stated in Assumption 6.2. Therefore, Proposition 6.2 theoretically shows that the proposed asynchronous architecture illustrated in Fig. 6.8 is a natural extension of synchronous learning-based control implementations and provides a theoretical foundation for its application in real-world control problems.

6.2.3. Numerical Evaluation

We exemplarily investigate the behavior of the proposed asynchronous online learning control architecture for varying computational delay bounds $\bar{T}^c$ in the example setting from Section 4.3.5. Hence, we consider a linear dynamical system with closed-loop dynamics described by

$$\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \begin{bmatrix} 0 & 1 \\ -k_c\tilde{\theta} & -k_c - \tilde{\theta} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.80 revisited}$$

and a nonlinear input perturbation $f(\boldsymbol{x}) = 1 - \sin(2x_1) + {}^1\!/_{(1+\exp(-x_2))}$. The control gains are set to $k_c = 10$ and $\tilde{\theta} = 5$, and training data is generated and processed online with a sample time of $T_s = 0.01$. In order to slightly simplify the simulation, we use constant computational delays $T_m^c = \bar{T}^c$ for all $m \in \mathbb{N}$.

An example trajectory of the tracking error for a computational delay of $\bar{T}^c = 0.001$ is depicted in Fig. 6.9. Due to the small computational delay, the control performance is barely deteriorated, such that a high tracking accuracy can be observed. This behavior can also be seen in Fig. 6.10, where the dependency of the tracking error averaged over 10 random simulations is visualized in dependency on the computation time $\bar{T}^c$. When $\bar{T}^c$ is large, the effective sampling time is reduced, such that a performance deterioration by a factor ≈ 10 is visible. When reducing the computation time $\bar{T}^c$ below the nominal sampling time $T_s = 0.01$, this effect quickly disappears, and practically no performance difference is visible in the range $\bar{T}^c \in [0.001, 0.0001]$. While at a conservative level, this behavior can also be observed in the probabilistic tracking error bound $\bar{v}$ defined in (6.19).

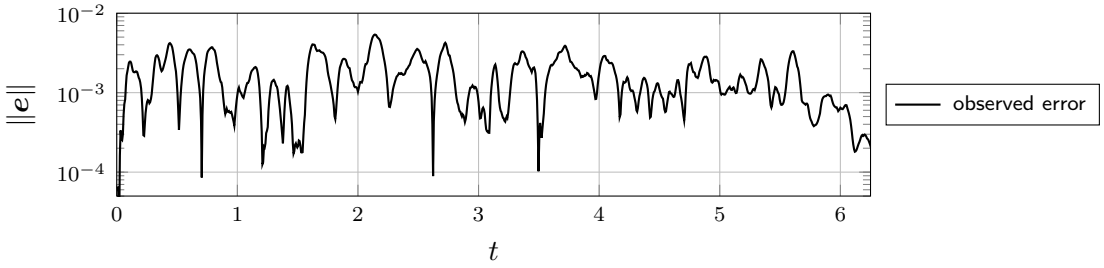

Figure 6.9.: Examplary evolution of the tracking error norm $\|\boldsymbol{e}(\cdot)\|$ with time-triggered online learning using the asynchronous learning control architecture for computational delay $\bar{T}^c = 0.001$.

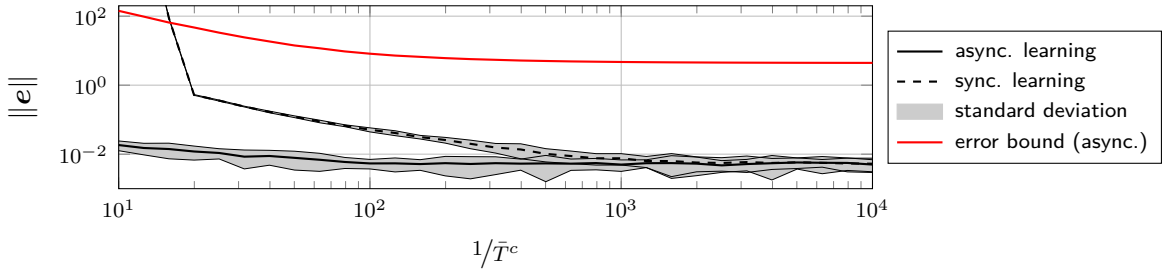

Figure 6.10.: Maximum tracking error and its bound $\bar{v}$ in dependency of the computational delay $\bar{T}^c$ for time-triggered online learning with the asynchronous learning control architecture.

The advantages of the asynchronous learning control architecture become apparent in a direct comparison with the synchronous integration of GPs, which is also depicted in Fig. 6.10. When the computational delay is small, the performance of both architectures is almost identical. However, a significant reduction of the tracking error can be seen for computational delays $\bar{T}^c \approx T_s$. In fact, for large computation times $\bar{T}^c$, the synchronous architecture seems to cause an unstable closed loop due to the delayed application of the linear feedback components. This clearly demonstrates the practical advantages of the proposed asynchronous online learning control architecture.

6.2.4. Experimental Demonstration in Human-Robot Interaction Scenario

We demonstrate the straightforward applicability of the asynchronous online learning control architecture in a human-robot interaction experiment, where the human imposes an unknown perturbation on the nominal robot dynamics. Here, human-induced dynamics are chosen due to their inherently non-parametric and unknown behavior, which requires expressive online-learning methods for a successful control adaptation. First, the experimental setup is explained in detail before the proposed asynchronous online learning architecture is compared to a manually tuned controller in a user study. Thereby, we demonstrate its ability to learn the dynamics of different human operators online and adapt the controller accordingly.

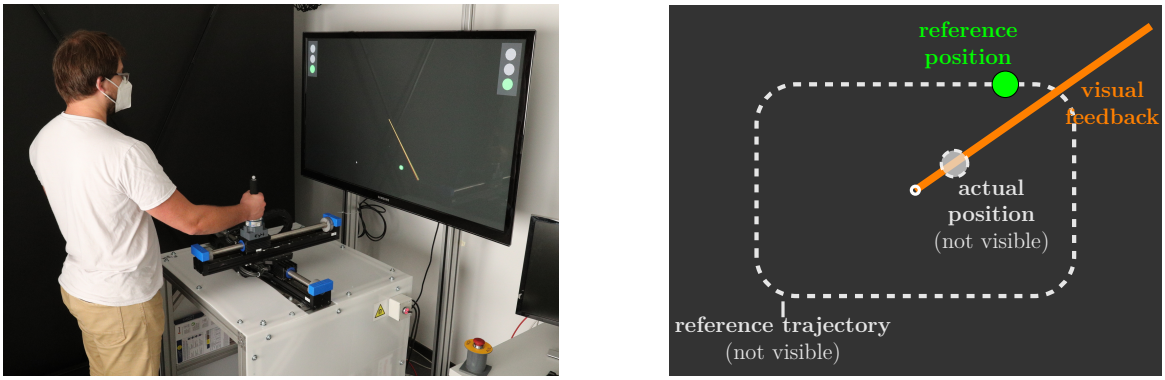

Figure 6.11.: Left: example photo of a participant performing the experiment task with the linear axis setup. Right: depiction of the task design of the experimental user study. The rounded rectangle drawn with the gray dashed line represents the reference trajectory of the green circle, which the participant is instructed to track. Instead of the actual current position, depicted by the gray circle, the subject can only see the polar angle shown by the orange line.

Experiment Setup and Task Design

The experiments are executed on a manipulandum with two degrees of freedom, which consists of two orthogonally mounted single rail stages (Copley Controls Thrustube Module), each driven by linear servo motors. Both rail stages are equipped with optical encoders that measure the position of a cart on the upper rail with 1 µm precision. Additionally, a six degree of freedom force-torque sensor (JR3-75M25) is mounted below the handle, through which the human interacts with the system, to measure forces in the horizontal plane. The force-torque sensor is strictly required to operate the device and is not used for the computation of control policies. Additionally, an inherent output force limitation is integrated as a safety measure, which guarantees that the interaction force applied to the human remains in safe regions, therefore, enabling experiments with aggressively tuned controllers. For control, we employ the CTC-based approach illustrated on the right side of Fig. 6.4 in two configurations: the full controller with a GP model and a version without learning. The decoupling between the control loop and numerical operations on the GP model is implemented by running them in separate processes connected via a UDP communication. The GP model is learned online using LoG-GPs with $\bar{N} = 50$ data points per local model. The hyperparameters of the local GP models are adapted online via log-likelihood maximization, which is executed using one step of the RPROP algorithm [228] every time a sample is added to a local model. This method is employed since it has been demonstrated to exhibit lower computational complexity compared to other gradient-based optimization schemes [229]. This approach allows us to run the control loop at a rate of 4 kHz, while we sample training data for model updates at a rate of 200 Hz.

The task itself is designed as follows: Standing in front of the apparatus and facing the screen, the subjects are instructed to track a green dot by moving the handle on top of the cart as illustrated on the left side of Fig. 6.11. In addition, the participants are informed that different controllers will support them during task execution. The provided visual feedback is artificially modified, therefore, limiting the participants' ability to successfully perform the task. Specifically, the subjects do not see their current position in the task space en-

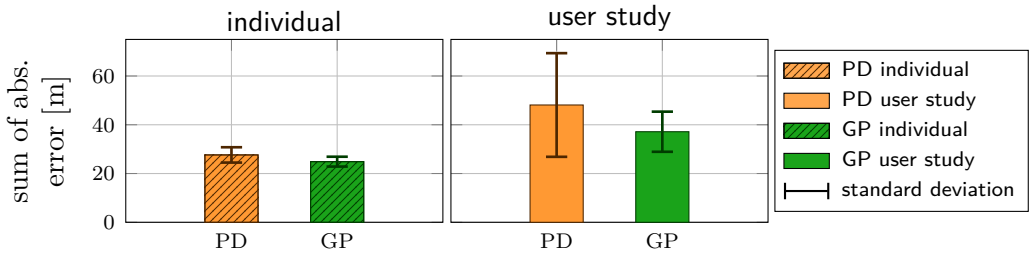

Figure 6.12.: Mean and standard deviation of the summed absolute error for one surrogate participant (left) and the complete user study (right).

tirely, but instead, only the angle from the origin is visualized through a pointer. However, their tracking performance is evaluated on the Cartesian position error despite the limited feedback. The task design and the visual feedback are depicted on the right side of Fig. 6.11. Each run of the experiment begins at the same starting position for the green circle and consists of five repetitions of the reference trajectory.

The complete experimental procedure is split into two parts; first, a training phase, followed by a test phase. During the initial training phase, the participants get accustomed to the task by performing one experiment run with both the GP-based and non-learning controller. Subsequently, the test phase begins, which consists of four experiment runs per controller. At every run, a random controller variation is selected. If, during any trial, the workspace limit is reached, the device shuts down as a safety precaution, and the run is evaluated as a failure. The failed runs are not repeated subsequently.

Comparison to Tuned PD Controller

In order to demonstrate the benefits and flexibility of the proposed learning control architecture, we conducted a user study with nine healthy, right-handed participants and compared to different controller variations. First, a so-called tuned PD variation is evaluated, which uses a combination of CTC and PD control law without online learning in the control loop. Due to the lack of a better procedure for adaptation in human-robot interaction, the gains of the PD controller are tuned manually to balance the applied forces and the resulting tracking performance. As practical considerations prevent a tuning with all participants, one individual is chosen instead. However, since the PD controller needs to be safe for all users, cautious tuning is preferred, which tends to result in lower control gains. The best trade-off is obtained for the parametrization

$$\boldsymbol{k}_{p,\text{tuned}} = \begin{bmatrix} 35 \\ 35 \end{bmatrix}, \quad \boldsymbol{k}_{d,\text{tuned}} = \begin{bmatrix} 3.5 \\ 3.5 \end{bmatrix}. \tag{6.25}$$

Second, a GP variation is investigated, which uses the proposed asynchronous online learning control architecture with the same CTC policy as in the PD variation. Here, small gains

$$\boldsymbol{k}_p = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{K}_d = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \tag{6.26}$$

are chosen for the PD control parametrization.

The effect of online learning on the control performance, as measured through the mean and standard deviation of the summed absolute error, is significant, as illustrated in Fig. 6.12.

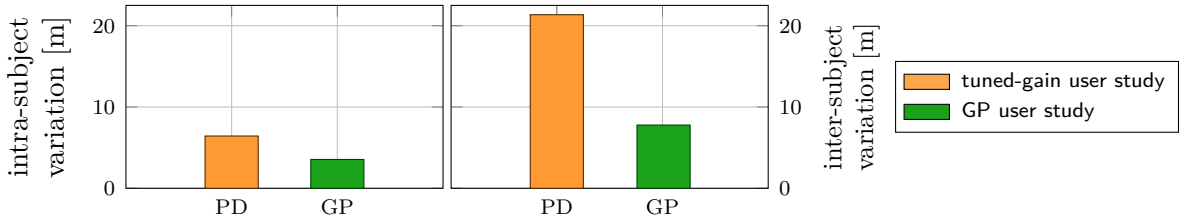

Figure 6.13.: Left: Average of intra-subject standard deviation in tracking error. Right: Inter-subject standard deviation of participant-specific average tracking error.

Table 6.1.: Tree structure parameters and computation times for a LoG-GP model at the end of the experiments of one participant.

	PRED. TIME	UPD. TIME	TREE DEPTH	NUM. LEAFS	POINTS/LEAF
AV.	0.16ms	1.56ms	59.5	294	25.65
MAX.	1.07ms	13.69ms	60	295	49

On the left-hand side, it can be seen that for the individual, the two controllers perform comparably well with regard to tracking performance, with the GP controller leading to slightly better tracking. However, the observed difference in tracking error is insignificant since it lies within the statistical variation and can be attributed to the cautious tuning of the PD controller. Therefore, the PD gains are appropriately tuned to ensure a fair comparison. When deploying the tuned PD controller to previously unobserved individuals and comparing the performance to the learning-based GP controller in a user study, as shown in Fig. 6.12 on the right, it can be seen that the tuned PD controller performs significantly worse. Similarly, the GP controller, on average, results in higher tracking errors in the user study than for the surrogate individual. However, the increase in mean tracking error is larger for the tuned PD controller, with a substantial growth in the standard deviation. Therefore, the tuned PD controller leads to inconsistent tracking results, which can be attributed to the different levels of task proficiency of the participants.

This becomes even clearer when looking at the intra- and inter-subject variation of the tracking error as depicted in Fig. 6.13. While on the left-hand side of Fig. 6.13, each participant exhibits a similar variation of the tracking error among the experiment runs for both controllers, the variation of the tracking error between different subjects is significantly larger for the tuned PD controller as seen on the right side. This is due to the inability of the tuned PD control law to adapt to the large variation in the induced dynamics exhibited by the participant pool. In contrast, the GP controller is able to compensate for the complex and varying dynamics caused by the different participants.

While this beneficial control performance can be attributed to online learning control with GP models in general, the strength of the asynchronous architecture becomes clear at a timing and structural analysis of the LoG-GP models as depicted in Table 6.1 for one exemplary participant. Since $\approx$7500 training samples are generated during each experiment, the tree becomes quite large. In combination with the nature of the observed task, this causes a strong variation of the tree depth, with some branches exhibiting a depth of 60. Due to the dependency of the complexity of model updates and predictions on the tree depth, this variation directly transfers to varying computation times. For the model evaluations, this does not cause any problems since the maximum prediction time remains below 5ms, which

is the sampling time corresponding to the update rate of 200 Hz. In contrast, a few ($\approx$1.42%) update times exceed this threshold. However, due to the asynchronous architecture, these violations are no problem in practice since the control loop is not slowed down, and thus, no catastrophic deterioration of the control performance occurs. Hence, this experiment demonstrates the robustness of the asynchronous online learning architecture against real-time violations in GP model updates and predictions.

6.3. Networked Online Learning under Resource Constraints

While the learning-based control architectures in Sections 6.1 and 6.2 enable the relatively straightforward transfer of theoretical guarantees to real-world applications, their design is strongly tailored to this purpose. This has the effect that the implementation of these architectures is generally resource intensive, e.g., powerful computing units and large memories are required. In practice, many reasons exist why providing these resources is undesirable and challenging. For example, weight and space limitations can prevent the installation of powerful computing devices on autonomous robots, and large local data storages may be too costly for industrial products. Therefore, there is a need for fundamentally different architectures which are tailored toward resource constraints arising in real-world applications.

A flexible way to achieve this is by exploiting the increasing possibilities arising from cloud computing for networked learning, similarly as proposed for control [230, 231]. Cloud computing generally offers massive computational and memory resources through remote servers. These servers can be accessed via network connections and are usually shared by multiple users [232]. However, cloud computing is not just the simple time-sharing of a remote resource as in classic server-client architectures. Without noticing it, a client in cloud computing often communicates with many servers simultaneously, which themselves can exchange information with each other [233]. This ensures that the computational load can be efficiently distributed among computation units, which allows us to abstract clouds as a powerful computing service.

We exploit this service by proposing a cloud-based online learning control architecture, which is capable of ensuring the satisfaction of memory and computational constraints on local devices while retaining theoretical guarantees. Since realistic communication network restrictions such as time delays and limited bandwidth prevent the full externalization of the online model inference to the cloud, we employ it only partially for determining and storing GP models. Our approach, which is illustrated in Fig. 6.14, transmits data to the remote computing system, where a LoG-GP model is maintained and iteratively updated. This LoG-GP model can ensure a bounded tracking error when used in model-based control. We use this accuracy information to determine the local GP models which need to be communicated back to the resource-constrained system via a sampling-based reachability analysis. Thereby, only a small amount of data remains on the local device, such that predictions and model updates can be computed locally without relevant delays. Moreover, we ensure the timely availability of required data on the local system using an effective transmission scheme. This scheme provides insight into fundamental trade-offs between bandwidth, time delays, local memory, and achievable tracking error. The effectiveness of the developed networked online learning architecture is demonstrated in simulations of a robotic exoskeleton as an example

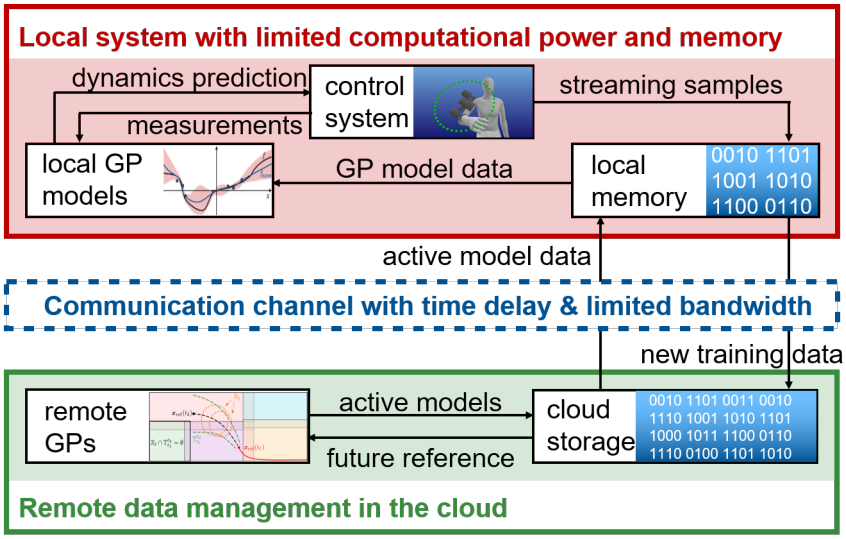

Figure 6.14.: Overview of the proposed networked online learning architecture: The LoG-GP predicts the unknown dynamics, e.g., of a wearable robot, for a measured state. For computing these predictions, it can only access GP model data in the local memory. Measurements of the system are continuously stored in the local memory and regularly sent to the cloud, where necessary models for a future reference trajectory are determined using a sampling-based approach, and corresponding data is sent to the local memory.

of a wearable robotics application.

The remainder of this section is structured as follows. In Section 6.3.1, the problem setting is formalized together with the considered resource constraints. The sampling-based reachability analysis for determining relevant local GP models is presented in Section 6.3.2. In Section 6.3.3, a delay-aware transmission scheme for irrelevant training data is proposed. Finally, the obedience of the proposed architecture to resource constraints is numerically demonstrated in Section 6.3.4.

6.3.1. Problem Setting

Similar to the previous sections, we investigate the networked online learning control architecture by exemplarily considering the setting in Section 3.1. This means that our system dynamics are described by

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}(u + f(\boldsymbol{x})), \tag{3.2 revisited}$$

where $f : \mathbb{X} \to \mathbb{R}$ is an unknown, scalar perturbation of the linear system. In order to track a given reference trajectory (3.3) with this system, we employ a control law

$$u(t) = -\boldsymbol{\theta}^T(\boldsymbol{x}(t) - \boldsymbol{x}_{\mathrm{ref}}(t)) + r_{\mathrm{ref}}(t) - \hat{f}(\boldsymbol{x}(t)), \tag{3.4 revisited}$$

where $\hat{f} : \mathbb{X} \to \mathbb{R}$ is a model of the unknown nonlinear perturbation $f(\cdot)$. We use LoG-GPs as proposed in Section 5.1 to learn this model $\hat{f}(\cdot)$ in a time-triggered fashion from a data set

$$\mathbb{D}(t) = \left\{\boldsymbol{x}^{(n)} = \boldsymbol{x}(nT_s), y^{(n)} = f(\boldsymbol{x}^{(n)}) + \epsilon^{(n)}\right\}_{n=1}^{\lfloor \frac{t}{T_s} \rfloor}. \tag{4.81 revisited}$$

Since this data set continuously grows over time, the LoG-GP model must be permanently updated. In order to ensure that these updates can indeed be executed online, they must be computed at higher rates than training samples are generated. Under the assumption of negligible computation times for predictions, this requirement can be formalized as the computational constraint

$$T_m^c \leq T_s \qquad \forall m \in \mathbb{N}, \tag{6.27}$$

where T_m^c denote the computation time of the m-th model update and T_s is the sampling time of the data set (4.81). Additionally, the continuous stream of data generated by event-triggered sampling leads to a steadily growing size of the data set $\mathbb{D}(t)$. Therefore, the amount of generated data will eventually reach the memory limitations, which are unavoidable in all real-world systems. Formally, this can be modeled via the memory constraint

$$|\mathbb{D}_{\mathrm{loc}}(t)| \leq \bar{N}_{\mathrm{mem}}, \tag{6.28}$$

where $\mathbb{D}_{\mathrm{loc}}(t)$ denotes the data set stored in the memory of the technical system and $\bar{N}_{\mathrm{mem}} \in \mathbb{N}$ represents the memory limitations. Since this restriction can crucially limit the achievable control performance due to the derived results in Chapter 4, we consider that data can be transferred to a cloud via a network connection, effectively extending the overall memory capacity. The available memory in the cloud is usually significantly larger than on the local system, such that we assume it to be infinite for simplicity. However, the data transfer between the cloud and the local system takes non-negligible time in practice due to effects such as network delays $T_d \in \mathbb{R}_+$ and finite bandwidth $B_{\mathrm{com}} \in \mathbb{R}_+$. Therefore, data sent to the cloud cannot be immediately accessed by the local system, but the time T_{access} between requesting data $\mathbb{D}$ and using it has to satisfy the network constraint

$$T_{\mathrm{access}} \geq \frac{|\mathbb{D}|}{B_{\mathrm{com}}} + T_d. \tag{6.29}$$

In order to develop a networked online learning control architecture under these restrictions, we make the following assumption on the closed-loop system without constraints.

Assumption 6.5. *The closed-loop system*

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta})\boldsymbol{e} + \boldsymbol{b}(f(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x})), \tag{6.30}$$

where $\boldsymbol{A}_{\mathrm{cl}}(\boldsymbol{\theta}) = \boldsymbol{A} - \boldsymbol{b}\boldsymbol{\theta}^T$ *and* $\tilde{\mu}(\cdot)$ *denotes the aggregated mean function of the LoG-GP model defined in* (5.1)*, admits a probabilistic tracking error bound* $\upsilon : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$.

If the conditions of Theorem 5.3 or Theorem 5.4 are satisfied, this assumption immediately follows from Theorem 3.1. Therefore, Assumption 6.5 is not restrictive and merely used to streamline the presentation in this section, such that we can focus on the problem of developing a networked online learning control architecture for inferring a highly accurate LoG-GP model of the unknown dynamics $f(\cdot)$ under computational, memory and network constraints.

6.3.2. Reachability-Based Local Model Selection

Since the time delay T_d prevents externalizing the online learning, we propose the networked online learning approach outlined in Fig. 6.14, which performs inference locally but transfers unnecessary data to the cloud. In order to transmit data to the cloud without performance loss, we exploit the modular structure of LoG-GPs whose aggregated mean function (5.1) can be expressed as

$$\tilde{\mu}(\boldsymbol{x}) = \psi_\mu \left(\sum_{l \in \mathbb{M}} \omega_l(\boldsymbol{x}) \boldsymbol{\psi}_\omega(\mu_l(\boldsymbol{x}), \sigma_l^2(\boldsymbol{x})) \right), \tag{6.31}$$

where $\mathbb{M}$ denotes the set of local GP models in the leaf nodes and $\psi_\mu : \mathbb{R}^{d_\psi} \to \mathbb{R}$, $\boldsymbol{\psi}_\omega : \mathbb{R}^2 \to \mathbb{R}^{d_\psi}$ are nonlinear functions. In this equation, we emphasize the state dependency of the weights $\omega_l : \mathbb{R}^{d_x} \to \mathbb{R}_{0,+}$ because they define the regions in which a local model l is active ($\omega_l(\boldsymbol{x}) \neq 0$) and consequently is necessary for the computation of the aggregation $\tilde{\mu}(\cdot)$. This property can be straightforwardly combined with the probabilistic tracking error bound ensured by Assumption 6.5, such that a sampling-based reachability analysis can be used to determine necessary models for computing (6.31).

In detail, the set $\mathbb{W}$ of potentially active models during a time window $\mathbb{I} = [t_1, t_2]$, $t_1, t_2 \in \mathbb{R}$, $t_2 > t_1$, is defined through the intersections between active regions $\mathbb{X}_l = \{\boldsymbol{x} : \omega_l(\boldsymbol{x}) > 0\}$ of local models $l \in \mathbb{M}$ and the tube

$$\mathbb{T}_{t_1}^{t_2} = \{\boldsymbol{x} \in \mathbb{R}^{d_x} : \exists t \in \mathbb{I}, \boldsymbol{x} \in \mathbb{B}_{\upsilon(t)}(\boldsymbol{x}_{\mathrm{ref}}(t))\} \tag{6.32}$$

based on balls

$$\mathbb{B}_{\upsilon(t)}(\boldsymbol{x}_{\mathrm{ref}}(t)) = \{\boldsymbol{x} \in \mathbb{R}^{d_x} : \|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{ref}}(t)\| \leq \upsilon(t)\} \tag{6.33}$$

with radius given by Assumption 6.5. Therefore, the set $\mathbb{W}$ is formally defined as

$$\mathbb{W} = \{l \in \mathbb{M} : \mathbb{X}_l \cap \mathbb{T}_{t_1}^{t_2} \neq \emptyset\}, \tag{6.34}$$

which is illustrated in Fig. 6.15. Since the computation of the intersections $\mathbb{X}_l \cap \mathbb{T}_{t_1}^{t_2}$ requires an explicit representation of the active regions $\mathbb{X}_l$ of local models $l \in \mathbb{M}$, which LoG-GPs do not provide, the definition of $\mathbb{W}$ in (6.34) cannot be directly used in practice. We follow a different idea exploiting the implicit representation of the active regions $\mathbb{X}_l$ via the weights $\omega_l(\cdot)$, which allows us to directly compute the set of active models

$$\mathbb{W}_{\boldsymbol{x}} = \{l \in \mathbb{M} : \omega_l(\boldsymbol{x}) > 0\} \tag{6.35}$$

for a given state $\boldsymbol{x}$. Therefore, we can alternatively represent the set of potentially active models during the time window $\mathbb{I}$ via

$$\mathbb{W} = \bigcup\nolimits_{t \in \mathbb{I}} \bigcup\nolimits_{\boldsymbol{x} \in \mathbb{B}_{\upsilon(t)}(\boldsymbol{x}_{\mathrm{ref}}(t))} \mathbb{W}_{\boldsymbol{x}}. \tag{6.36}$$

By approximating the unions over uncountable sets via discretization and random sampling as outlined in Algorithm 6.1, we can over-approximate the set $\mathbb{W}$ via $\hat{\mathbb{W}}$ and obtain

$$\tilde{\mu}_{\hat{\mathbb{W}}}(\boldsymbol{x}) = \psi_\mu \left(\sum_{l \in \hat{\mathbb{W}}} \omega_l(\boldsymbol{x}) \boldsymbol{\psi}_\omega(\mu_l(\boldsymbol{x}), \sigma_l^2(\boldsymbol{x})) \right). \tag{6.37}$$

If sufficiently many samples are used in Algorithm 6.1, this approximation yields identical predictions as shown in the following result.

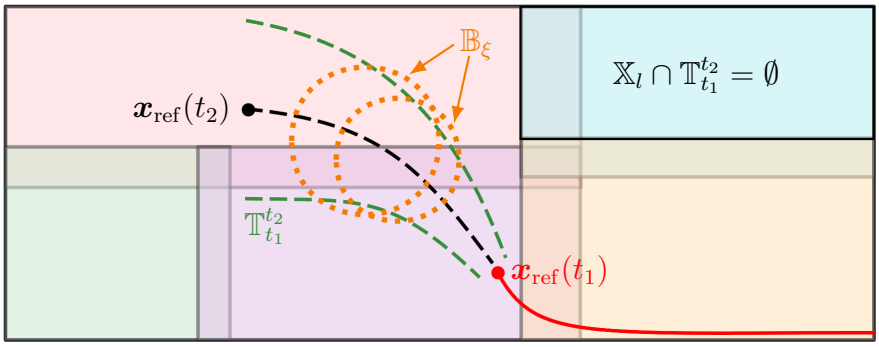

Figure 6.15.: A local model $l \in \mathbb{M}$ is inactive if its active region $\mathbb{X}_l$ does not intersect with the tube $\mathbb{T}_{t_1}^{t_2}$ induced by the tracking error bound $v(t)$ as illustrated for the region in the top right. The set of active models $\hat{\mathbb{W}}$ is found by over-approximating the tube $\mathbb{T}_{t_1}^{t_2}$ with balls $\mathbb{B}_\xi$, from which random samples $\boldsymbol{x}^{(i)}$ are drawn to determine the active models $\mathbb{W}_{\boldsymbol{x}^{(i)}}$ at these states.

Algorithm 6.1. Determining active models

$\hat{\mathbb{W}} \leftarrow \emptyset$
compute ζ using (6.38)
for $j = 0 : \lceil \frac{t_2 - t_1}{T_{\text{dis}}} \rceil$ **do**
  for $i = 1 : N_s$ **do**
    Determine active models $\mathbb{W}_{\boldsymbol{x}^{(i)}}$ for input $\boldsymbol{x}^{(i)} \sim \mathcal{U}(\mathbb{B}_{\zeta_j}(\boldsymbol{x}_{\text{ref}}(t)))$
    $\hat{\mathbb{W}} \leftarrow \hat{\mathbb{W}} \cup \mathbb{W}_{\boldsymbol{x}^{(i)}}$
  end for
end for

Theorem 6.1. *Consider a dynamical system* (3.2)*, to which a controller* (3.4) *is applied to a Lipschitz continuous reference trajectory* (3.3)*, such that Assumption 6.5 is satisfied. Choose*

$$\zeta_j = 2\underline{\zeta} + L_{\boldsymbol{x}_{\text{ref}}} \frac{T_{\text{dis}}}{2} + v(t_1 + jT_{\text{dis}}) \tag{6.38}$$

for constants $\underline{\zeta}, T_{\text{dis}} \in \mathbb{R}_+$*. Then, for* $N_s \in \mathbb{N}$ *random samples and* $N_{\mathbb{I}} = \lceil (t_2 - t_1)/T_{\text{dis}} \rceil$ *time discretization steps, it holds that*

$$\mathbb{P}\left(\tilde{\mu}_{\hat{\mathbb{W}}}(\boldsymbol{x}(t)) = \tilde{\mu}(\boldsymbol{x}(t)),\ \forall t \in [t_1, t_2]\right) \geq 1 - |\mathbb{M}| N_{\mathbb{I}} \left(1 - \frac{\min\{r_{\min}^{d_x}, c^{d_x}\}}{\bar{\zeta}^{d_x}}\right)^{N_s}, \tag{6.39}$$

where $r_{\min}$ *denotes the radius of the largest ball contained in the smallest active region of a leaf node* $l \in \mathbb{M}$ *and* $\bar{\zeta} = \max_{j=0,\ldots,N_{\mathbb{I}}} \zeta_j$*.*

Proof. Due to Assumption 6.5, at each time t, the tracking error $\boldsymbol{e}$ is bounded by $v(t)$. Since $\boldsymbol{x}_{\text{ref}}(\cdot)$ is assumed to be Lipschitz continuous, this implies

$$\|\boldsymbol{e}(t)\| \leq \tilde{\zeta}_j, \qquad \forall t \in \left[t_1 + \frac{2j-1}{2} T_{\text{dis}}, t_1 + \frac{2j+1}{2} T_{\text{dis}}\right] \tag{6.40}$$

for $\tilde{\zeta}_j = {}^{L_{\boldsymbol{x}_{\mathrm{ref}}} T_{\mathrm{dis}}}/_2 + \upsilon(t_1 + jT_{\mathrm{dis}})$. Consequently, we have

$$\mathbb{T}_{t_1}^{t_2} \subset \bigcup_{j=1}^{N_{\mathbb{I}}} \mathbb{B}_{\tilde{\zeta}_j}(\boldsymbol{x}_{\mathrm{ref}}(t_1 + jT_{\mathrm{dis}})). \tag{6.41}$$

Therefore, it remains to show that the set of active models for time $t_1 + jT_{\mathrm{dis}}$ defined as

$$\mathbb{W}_{t_1+jT_{\mathrm{dis}}} = \bigcup_{\boldsymbol{x} \in \mathbb{B}_{\tilde{\zeta}_j}} \mathbb{W}_{\boldsymbol{x}} \tag{6.42}$$

is overapproximated by Algorithm 6.1. For this purpose, choose any model $l \in \mathbb{W}_{t+jT_{\mathrm{dis}}}$. Then, the intersection between the active region $\mathbb{X}_l$ of this model and the ball $\mathbb{B}_{\zeta_j}$ has a volume of at least $\pi^{d_x/2}(\min\{r_{\min}, \underline{\zeta}\})^{d_x}/\Gamma(\frac{d_x}{2}+1)$, where $\Gamma : \mathbb{R}_+ \to \mathbb{R}_+$ denotes Euler's gamma function [234]. Therefore, the probability of a sample $\boldsymbol{x}^{(i)} \sim \mathcal{U}(\mathbb{B}_{\zeta_j})$ being in the active region of model l can be bounded by

$$P(\omega_l(\boldsymbol{x}^{(i)}) > 0 | l \in \mathbb{W}_{t_1+jT_{\mathrm{dis}}}) \geq \min\{r_{\min}^{d_x}, \underline{\zeta}^{d_x}\}/\zeta_j^{d_x}. \tag{6.43}$$

The probability of none of the N_s samples falling into the active region $\mathbb{X}_l$ is consequently upper bounded by $(1 - P(\omega_l(\boldsymbol{x}^{(i)}) > 0 | l \in \mathbb{W}_{t_1+jT_{\mathrm{dis}}}))^{N_s}$, such that (6.39) follows from the union bound over all $N_{\mathbb{I}}$ time steps and all models $l \in \mathbb{M}$. □

Since this theorem ensures that (6.31) and (6.37) are identical with probability greater than (6.39), it ensures that using $\tilde{\mu}_{\hat{\mathbb{W}}}(\cdot)$ as model in the control law (3.4) yields no reduction in control performance with high probability. Therefore, it allows us to determine irrelevant data for a time interval $\mathbb{I}$, which we exploit in the following section for transmitting data to the cloud, thereby reducing the local memory occupation.

6.3.3. Delay-Aware Local Model Transmission

Due to the non-negligible time required for a data transfer, the transmission to and from the cloud must be carefully scheduled in order to ensure that the necessary data is always available locally. For simplicity, we consider that data is transmitted at regularly spaced time instances jT_{trans}, $j = \mathbb{N}$, such that each time interval $\mathbb{I}_j = [(j-1)T_{\mathrm{trans}}, jT_{\mathrm{trans}}]$ has a length of $T_{\mathrm{trans}} \in \mathbb{R}_+$. During each time interval $\mathbb{I}_j$, we propose the transmission scheme illustrated in Fig. 6.16, where the idea is that the memory is divided into two parts. During each interval $\mathbb{I}_j$, half of the memory is used for updating the local data set with data from the cloud, while the other half contains the data set $\mathbb{D}_j$ necessary for computing the mean predictions $\tilde{\mu}_{\hat{\mathbb{W}}}(\cdot)$ during time interval $\mathbb{I}_j$ according to the potentially active models $\hat{\mathbb{W}}_j$. To identify the data for updating the local memory, the data set $\mathbb{D}_{j-1}$ from the previous interval $\mathbb{I}_{j-1}$, which contains newly measured training samples as well as data from the cloud, is sent to the cloud. Once this transmission has been completed, the cloud contains the complete data set $\mathbb{D}_{(j-1)T_{\mathrm{trans}}}$ obtained until time $(j-1)T_{\mathrm{trans}}$, such that Algorithm 6.1 can be employed to determine the possibly active models $\hat{\mathbb{W}}_{j+1}$ for the next time interval $\mathbb{I}_{j+1}$ in the cloud. The corresponding data set $\mathbb{D}_{j+1}$ is sent to the local memory, such that it is available for $t \geq (j+1)T_{\mathrm{trans}}$.

It is straightforward to see that this transmission scheme can ensure the satisfaction of the network constraint (6.29) for a fixed data set $\mathbb{D}_j$, if $T_{\mathrm{access}} = {}^{T_{\mathrm{trans}}}/_2$ is sufficiently large. However, due to the online generation of data during system operation, it generally cannot

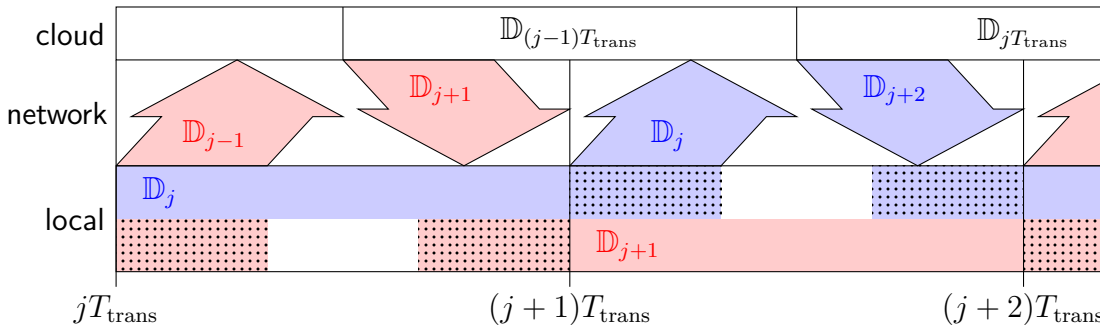

Figure 6.16.: During each interval $\mathbb{I}_j = [jT_{\text{trans}}, (j+1)T_{\text{trans}}]$, the previously necessary data $\mathbb{D}_{j-1}$ is sent to the cloud and the data $\mathbb{D}_{j+1}$ for the next interval $\mathbb{I}_{j+1}$ is fetched. While these data sets occupy memory during the interval $\mathbb{I}_j$, parts of $\mathbb{D}_{j-1}$ and $\mathbb{D}_{j+1}$ are in transmission and not available on the local system. Therefore, these data sets cannot be used for prediction, which is highlighted through the dotted pattern. The data in the cloud is updated with incoming transmissions, such that it contains the complete data set $\mathbb{D}_{(j-1)T_{\text{trans}}}$ up to the end of the previous interval $j-1$.

Algorithm 6.2. Data transfer scheme: cloud

1: **for** $n = 1, \ldots, \infty$ **do**
2: **if** $nT_s \geq jT_{\text{trans}}$ **then**
3: $j \leftarrow j+1$
4: Receive data set $\mathbb{D}_{j-1}$
5: Determine active model set $\hat{\mathbb{W}}_{j+1}$ using Algorithm 6.1
6: generate data set $\mathbb{D}_{j+1}$ according to $\hat{\mathbb{W}}_{j+1}$ and transmit it
7: **end if**
8: **end for**

be ensured that the data sets $\mathbb{D}_j$ have a bounded size, such that the fixed time T_{trans} might eventually not be sufficient to finish the transmission within the time interval $\mathbb{I}_j$. Therefore, the learning with data generated online during system operation has to be stopped eventually at some interval $\mathbb{I}_{\bar{N}_{\text{com}}}$, $\bar{N}_{\text{com}} \in \mathbb{N}$ in order to upper bound the size of all sets $\mathbb{D}_j$. This leads to the data transfer scheme outlined in Algorithm 6.2 for the cloud and in Algorithm 6.3 for the local system, for which it is straightforward to prove the satisfaction of the network constraint (6.29) as shown in the following result.

Lemma 6.2. *Choose* $T_{\text{trans}} \geq \frac{\bar{N}_{\text{mem}}}{B} + 2T_d$ *and* $\bar{N}_{\text{com}} \in \mathbb{N}$ *such that the memory constraint* (6.28) *is satisfied. Then, Algorithms 6.2 and 6.3 ensure the satisfaction of the network constraint* (6.29).

Proof. Satisfaction of the memory constraint (6.28) implies that the transmission of $\mathbb{D}_j$, $j \in \mathbb{N}$, can be achieved with time $\frac{\bar{N}_{\text{mem}}}{2B_{\text{com}}} + T_d$ since it can contain at most half of the memory capacity $\bar{N}_{\text{mem}}$. Since two sets need to be transmitted, the overall transmission time per time interval $\mathbb{I}_j$ can be bounded by

$$\bar{T}_{\text{trans}} \leq \frac{\bar{N}_{\text{mem}}}{B_{\text{com}}} + 2T_d. \tag{6.44}$$

Algorithm 6.3. Data transfer scheme: local system

```
for n = 1, ..., ∞ do
    if nT_s ≤ N̄_com T_trans then
        sample data pair (x^(n), y^(n)) and update D_t^loc
    end if
    if nT_s ≥ jT_trans then
        j ← j + 1
        transmit updated data set D_{j−1}
        delete data set D_{j−1}
        receive necessary data set D_{j+1} for next interval
    end if
end for
```

Due to the construction of the transmission interval length T_{trans}, this immediately implies the satisfaction of the network constraint (6.29). □

In order to apply this lemma in a real-world system, it remains to develop an approach for enforcing the memory constraint (6.28) by choosing a suitable value of $\bar{N}_{\text{comm}}$. In practice, this value can be selected online using heuristics such that learning can be stopped, e.g., when the number of active models exceeds a threshold. Moreover, when the reference $\boldsymbol{x}_{\text{ref}}(\cdot)$ is periodic, we can determine $\bar{N}_{\text{comm}}$ based on the data sets from previous periods, as shown in the following theorem.

Theorem 6.2. *Assume the reference trajectory is periodic with period $T_p = cT_{\text{trans}}$ for $T_{\text{trans}} \geq {}^{\bar{N}_{\text{mem}}}/_{B_{\text{com}}} + 2T_d$ and $c \in \mathbb{N}$. Let*

$$\bar{N}_{\text{com}} = c + \min_{|\mathbb{D}_j| > \frac{\bar{N}_{\text{mem}} - 2\bar{m}}{2}} j, \qquad \bar{m} = \max_{j \in \mathbb{N}} |\mathbb{D}_{(j+c)}| - |\hat{\mathbb{D}}_j| \leq \left\lceil \frac{T_p}{T_s} \right\rceil. \tag{6.45}$$

Then, Algorithms 6.2 and 6.3 ensure the satisfaction of the memory constraint (6.28) *and network constraint* (6.29)*.*

Proof. Since the cardinality of $\mathbb{D}_{j+c}$ can be bounded by $|\mathbb{D}_{j+c}| \leq |\mathbb{D}_j| + \bar{m}$, memory constraints are satisfied as long as $|\mathbb{D}_j| \leq \bar{N}_{\text{mem}}/2 - \bar{c}$. Therefore, $\bar{N}_{\text{com}}$ as defined in (6.45) ensures that the memory constraint (6.28) is satisfied, which implies the satisfaction of the network constraint (6.29) due to Lemma 6.2. □

This theorem allows us to determine online when to stop adding new training samples to the LoG-GP by checking if $|\mathbb{D}_j| > \frac{\bar{N}_{\text{mem}}}{2} - \bar{m}$, which can be performed with low complexity and can be directly implemented. Moreover, it provides valuable insight into the interrelations between achievable tracking accuracy, memory constraint $\bar{N}_{\text{mem}}$, time delay T_d and limited bandwidth B_{com}. In order to see this, note that the data set size $|\mathbb{D}_j|$ usually grows almost linearly with the interval length T_{trans}. Since an increase in bandwidth B_{com} admits smaller T_{trans}, learning can continue up to higher values of $\bar{N}_{\text{com}}$ in general. Therefore, a higher data density can be achieved, which in turn yields a lower GP variance as shown in Chapter 4, thus, guaranteeing a smaller tracking error. In contrast, an increase in local memory $\bar{N}_{\text{mem}}$ admits larger data set sizes $|\mathbb{D}_j|$, but in turn requires longer intervals T_{trans}, such that the achievable data density and consequently the tracking accuracy are barely affected. Finally, a reduction

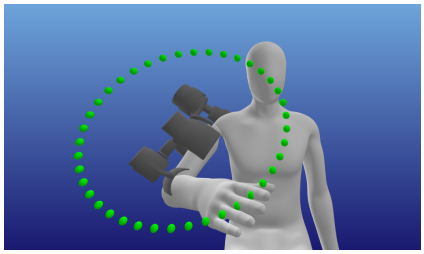

Figure 6.17.: Visualization of the upper-limb human-exoskeleton simulation and trajectory tracking task. The green circles depict discrete points along the elliptic reference trajectory, which must be followed with the hand.

of the delay T_d allows smaller values of T_{trans} and thereby also leads to an improvement in achievable control performance. Therefore, available bandwidth B_{com} for data transmission and time delay T_d are crucial for the achievable tracking accuracy when using the networked online learning control law, while finite local memory $\bar{N}_{\text{mem}}$ only has secondary relevance to enable implementation of the transmission scheme using Algorithm 6.2 and 6.3. This insight can be beneficially used for the design of autonomous systems in practice since it allows a reduction of local memory when sufficient bandwidth for data transmission is available.

6.3.4. Numerical Evaluation in Exoskeleton Control

In order to evaluate the applicability of the proposed networked online learning control architecture for resource-constrained systems[1], we employ it for the control of an upper-limb exoskeleton assisting a user in tracking a reference trajectory, which is simulated in Julia [235], a modern programming language for accelerating physics simulations. Since the exoskeleton is intended to be used in a portable manner, this scenario resembles an example of a wearable robotic system with memory and computational constraints. These constraints are particularly challenging for the control of the exoskeleton as human user data is required in practice to infer models allowing for personalized assistance.

For the simulation, we assume a rigid kinematic coupling between the human and exoskeleton arm, which allows the modeling of both as one kinematic chain consisting of four degrees of freedom. The exoskeleton model is based on the design described in [236], whilst the model parameters for the human are chosen according to anthropometric tables [237]. Here, the reference is set to 70 kg and 1.75 m. As illustrated in Fig. 6.17, the goal is to track an elliptic trajectory with the hand of the human by employing the learning-based feedback linearizing control law (3.76), such that the tracking error bound $\upsilon(\cdot)$ can be determined using Corollary 3.2. Each period of the ellipse takes $T_p = 6$s, the simulation runs at 1kHz, and we consider a memory constraint of $\bar{N}_{\text{mem}} = 4000$ data pairs for the local memory. Streaming data for online learning is generated with noise standard deviation $\sigma_{\text{on}} = 0.05$ at a sampling rate of 100Hz, i.e., $T_s = 10$ms. Each local GP model can contain a maximum of $\bar{N} = 100$ training points, and the hyperparameters are set to $\sigma_f = 1$, $l_i = 1/l_i = 3$ for inputs corresponding to joint angles/angular velocities. Algorithm 6.1 is run with temporal discretization $T_{\text{dis}} = 10$ms and $N_s = 1000$ random samples. Finally, the control gains are set to $k_c = 400$ and $\tilde{\theta} = 1$.

[1]Open-source code conceptually demonstrating the proposed method is available at `https://gitlab.lrz.de/online-GPs/cloud-GPs`.

Table 6.2.: Bandwidth B, time delay T_d, state measurement standard deviation $\sigma_{\boldsymbol{x}}$ and resulting time when learning is stopped T_s

	LOW BANDWIDTH	MEDIUM BANDWIDTH	HIGH BANDWIDTH	LARGE DELAY
B [samples/s]	1500	3000	10000	10000
T_d [s]	0.1	0.1	0.1	1.0
T_s [s]	44.67	60.04	—	30.81

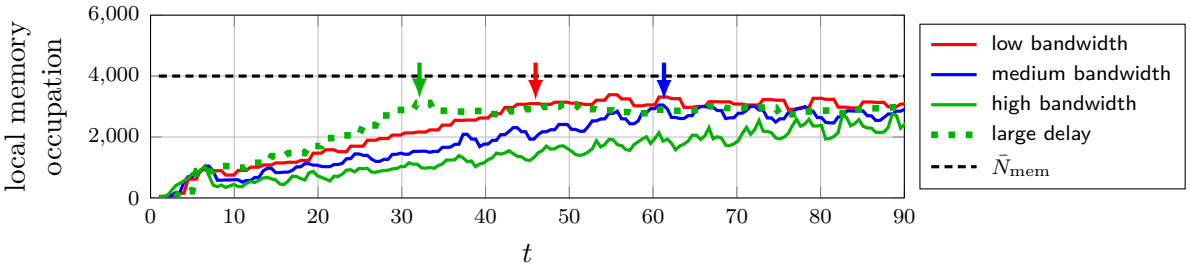

Figure 6.18.: The higher the bandwidth B, the longer the LoG-GP can learn before the number of training pairs in the local memory reaches the limitations. Large time delay T_d causes a significantly earlier stopping of learning, as indicated by the arrows.

In order to investigate the dependency of the tracking accuracy and memory occupation on the network bandwidth B and time delay T_d, we compare networked LoG-GP controllers under different simulation conditions as outlined in Table 6.2. Moreover, we employ a LoG-GP without memory constraints, i.e., $\bar{N}_{\text{mem}} = \infty$, as a baseline to illustrate the absence of a performance loss of the networked LoG-GP when a sufficiently high bandwidth is available. The average update time for the LoG-GP is 0.3ms $< T_s$ in all simulations, and the resulting curves for the evolution of the local memory occupation are depicted in Fig. 6.18. Since the LoG-GP has low accuracy during the first period, the tracking error bound $\upsilon(\cdot)$ is large during the first 6s, such that all data is required on the local system. After this period, the different curves exhibit the behavior discussed in Section 6.3.3: the lower the bandwidth B_{com}, the faster the local memory consumption grows. Moreover, an increase in time delay T_d causes a significantly faster growing memory occupation. Due to the limited local memory, this leads to an early stop in learning at the times depicted in Table 6.2, after which the memory occupation stagnates.

The stagnation has an immediate effect on the evolution of the tracking error, as illustrated in Fig. 6.19. While online updates are executed, the feedback linearizing control law (3.76) with networked LoG-GP model achieves the same improvement in tracking accuracy as with the unconstrained LoG-GP. After the updates stop, the tracking performance ceases to improve and effectively remains constant. Due to the continual learning of the networked LoG-GP with a high bandwidth connection, the evolution of the tracking error is visually identical to the curve resulting from the LoG-GP without memory constraint. This clearly demonstrates that the proposed approach allows a transfer of data to the cloud without any loss in performance when sufficient transmission bandwidth is available. Moreover, even when online learning has to be stopped early, it still yields a significant improvement in tracking accuracy compared to the baseline case without model learning, where a stationary

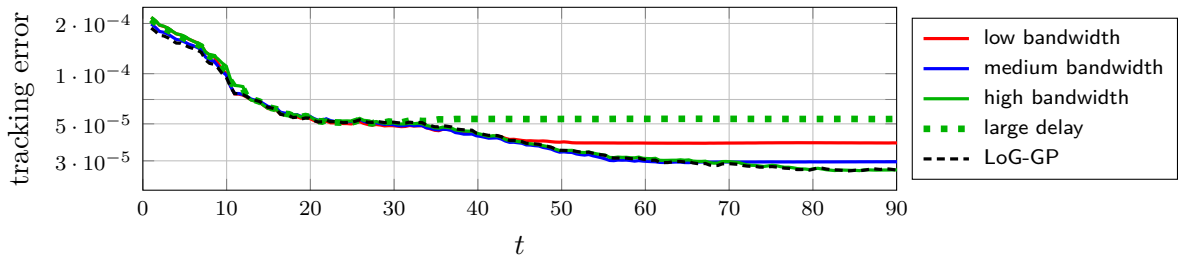

Figure 6.19.: When the memory limitation is reached and the learning process stops, the tracking error stagnates. Since higher bandwidths B allow learning for a longer time, larger values of B yield lower tracking errors eventually. While learning continues, networked LoG-GPs ensure the same tracking accuracy as LoG-GPs without memory constraints. Overall, online learning significantly improves the tracking accuracy over the baseline case without model learning, which is not depicted since it permanently exceeds $2 \cdot 10^{-2}$.

error of $\approx 2 \cdot 10^{-2}$ rad has been observed. This strongly underlines the practical advantages of the networked online learning control architecture by enabling the inference of GP models for control despite resource constraints.

6.4. Discussion

In this chapter, we discuss, to the best of our knowledge, for the first time, different architectures for implementing learning-based control with GP models. While the synchronous architecture discussed in Section 6.1 resembles merely the direct implementation of theoretical approaches for integrating GP models into control loops, it can allow a fast realization of learning-based controllers due to its conceptual simplicity. Since the GP model is confronted with state measurement disturbances in real-world experiments, the achievable control performance in practice is lower than in a theoretical analysis without this effect. We demonstrate that our derived guarantees can be adapted to this more realistic scenario as shown through Proposition 6.1, but the purpose of this analysis is a conceptual demonstration of the flexibility of our theory. Therefore, there remain many open questions, such as an analysis of the asymptotic behavior and the effect of GP-based learning on the optimal control gains with disturbed state measurements, which we do not address because the focus of this chapter lies on the architectures for learning-based control. However, our results should serve as a motivation to deepen the research in this area in order to provide a theoretical understanding of effects observed in real-world experiments and potentially enable new approaches in applied control [238].

While the synchronous architecture can be straightforwardly implemented, executing it on a real device can be challenging. As demonstrated in Section 6.1.4, it is possible to run it with 200Hz on the CARBO robotic manipulator, but higher sampling rates are yet to be achieved. Moreover, an increase of the sampling time occasionally resulted in unstable behavior of the robot. This example illustratively demonstrates the weakness of the synchronous architecture caused by the direct coupling of the learning and control components. The asynchronous architecture presented in Section 6.2 effectively overcomes this issue and allows the robust integration of GP models into control loops. Due to the decoupling of learn-

ing and control, different types of GP approximations can be seamlessly exchanged without the necessity to perform any changes in the control loop itself. This significantly simplifies the development process, which is an additional practical benefit. At the same time, the decoupling complicates the theoretical analysis since it requires the consideration of delayed model evaluations. The required analysis is rather straightforward for time-triggered sampling schemes as demonstrated through Proposition 6.2, but it is, in fact, also possible to adapt event-triggered strategies to this scenario [239]. Therefore, the asynchronous online learning control architecture offers an effective way to implement control laws with GP models in practice but potentially requires a slight adaptation of the theoretically derived guarantees.

Since the synchronous and asynchronous architectures directly run on local devices, they require considerable computational and memory resources on the controller hardware. This limitation is used as the motivation for the networked architecture proposed in Section 6.3, which offloads requirements to the cloud. Due to the necessary coordination of multiple coupled processes on physically separated computing units, the implementation of this architecture is generally challenging. However, it also provides the potential for significant improvements over purely local architectures since the cloud infrastructure can enable the application of computationally demanding methods. The presented realization of the networked architecture in Section 6.3 should only be understood as an initial framework in this context, which still admits plenty of opportunities for improvement. This begins at the simple realization of continual learning by avoiding the termination of data generation, which can already be achieved by transmitting only the most important data samples back to the local devices as selected, e.g., through the insights gained from Chapter 4. Moreover, advanced ideas like hierarchical structures can be envisioned, which combine accurate GP models with large transmission delays from the cloud with imprecise local models with low latency. Hence, the networked online learning control architecture exhibits a huge potential for future research.

Conclusion and Outlook on Future Research Directions

7.

Gaussian process regression has received growing attention in the control community over the last decade. On the one hand, many researchers focus on control-theoretical properties of closed-loop systems containing GP models, such that formal performance guarantees for a variety of control laws exist today. On the other hand, there is a great interest in the execution of GP-based control laws on real-world systems, which is frequently realized through a fallback to GP approximations, whose theoretically relevant properties for control are often unknown. Due to a usually control-oriented approach, both theoretical and applied research additionally often fail to capture and exploit the data-driven nature of GP models, such that they are employed as a static model.

In this thesis, we aim to overcome these limitations of existing research by developing a framework that covers the whole process from data generation, over control performance guarantees, to the implementation of GP-based control laws on real-world systems. This is achieved through the following key contributions.

7.1. Summary of the Contributions

In Chapter 2, we derive a Bayesian uniform error bound for GP regression. Since the evaluation of this bound requires a Lipschitz constant of the function to be learned, we exploit the GP distribution to derive a probabilistic Lipschitz constant. Moreover, we demonstrate that all other required parameters of the Bayesian uniform error bound can be easily computed, which enables its straightforward usage in control.

Based on the GP error bound, we analyze the tracking accuracy for two classes of control systems employing GP models in Chapter 3. We start with the illustrative example of linear systems with an unknown input perturbation, which is compensated using a GP model. We show that a tracking error bound for this system class can be formulated as a dynamical system whose input depends on the model accuracy along the reference trajectory. In order to investigate the tracking accuracy for a broader class of dynamics, we leverage ideas from Lyapunov stability theory. This allows us to derive a relatively tight error bound by expressing it as an optimization problem, while the linearization around a reference trajectory allows a closed-form expression, which can be interpreted as an extension of the result for linear systems. As this analysis merely requires GP-based control laws to nominally stabilize the system defined through the GP mean function, it can be applied to a broad range of control design techniques.

In order to prove the promise of improving control performance with an increasing number of training samples, we develop data generation strategies during closed-loop control for achieving arbitrarily small tracking errors in Chapter 4. For this purpose, we analyze the

decay of the posterior GP variance and correspondingly uniform error bounds with growing data sets. This analysis yields a novel, kernel-specific data density measure, directly relating uniform error bounds to the available training data. Using this measure, we straightforwardly derive tracking error convergence rates for growing data sets if a GP model is used to compensate for input perturbations in linear dynamical systems. Moreover, we demonstrate that the necessary accuracy of GP models for achieving desired tracking error bounds is generally state-dependent in nonlinear control systems. We exploit the gained knowledge about the role of training data for the control performance to develop three strategies for learning during closed-loop operation. Firstly, an episodic approach is developed, which relies on an iteration between rolling out a GP-based control law and updating it offline with sampled data. For this strategy, we prove that sufficiently fast sampling of training data guarantees an arbitrarily high tracking accuracy after a finite number of episodes. Secondly, the time-triggered sampling of data is proposed, for which we show a vanishing tracking error in the limit of continuous model updates, i.e., zero sampling time. Finally, an event-triggered data generation strategy is developed, which samples data points on a per-need basis to keep the tracking error below a desired threshold. We demonstrate that this event trigger can generally not ensure arbitrary tracking accuracies but derive bounds for the admissible desired errors.

In order to be capable of realizing these online learning schemes in practice, we develop a computationally efficient GP approximation in Chapter 5, which inherits uniform error bounds from exact GP regression. Our method, called LoG-GP, iteratively constructs a tree whose leaf nodes contain locally active GP models. We show that the construction of this tree and, thereby, also the computation of model updates has a logarithmic computational complexity. For determining predictions, we aggregated multiple individual GP model evaluations. Our method ensures that only a small number of GP models needs to be evaluated simultaneously, such that predictions can be determined with a squared logarithmic complexity. The employed aggregation schemes for GP models effectively correspond to a weighted summation of mean functions, such that uniform error bounds from exact GP regression directly extend. In addition to online learning, we show that aggregation schemes also enable scalable, distributed GP models in multi-agent systems. We propose a consensus-based aggregation of GP models, such that the strong theoretical foundations of consensus algorithms straightforwardly allow to extend prediction error bounds. This distributed model is employed in a cooperative tracking control law, for which the model accuracy guarantees allow the derivation of tracking error bounds.

Finally, we address the problem of the implementation of GP-based control laws in real-world applications in Chapter 6. Since state measurements of real systems are commonly disturbed, e.g., by sensor noise, we first show that this deviation from our previous theory does not invalidate our derived guarantees. This allows us to develop and analyze two architectures for learning-based control, which address specific limitations of GP-based controllers. We begin with an asynchronous architecture in which the sampling rates of GP models and the remaining control loop are decoupled. This allows the less frequent computation of GP predictions and thereby accounts for their comparatively high computational complexity, even when approximations are employed. Moreover, we propose a networked online learning architecture with LoG-GP models, which enables learning on systems with resource constraints such as memory limitations. The idea for achieving this relies on the fact that only a few models are usually necessary to evaluate a LoG-GP model. Therefore,

we can perform a reachability analysis using previously derived tracking error bounds to determine relevant models. Irrelevant models are sent to the cloud, such that local resource constraints can be satisfied without sacrificing control performance. The real-world applicability of the proposed online learning control architectures is demonstrated in simulations and robotics experiments.

7.2. Implications of Derived Results

The contributions of this thesis have a direct impact on the challenges in learning-based control with GP models. These implications are discussed in the following.

Challenge 1 is addressed by the uniform error bounds for GP regression. These bounds are shown to be valid in two fundamentally different settings. In Section 2.2, the considered set of unknown functions is restricted by an upper bound on the RKHS norm. This allows us the derivation of uniform error bounds for various assumptions on the observation noise and is exploited to bound the effect of disturbed training inputs on the regression error in Section 6.1.2. In Section 2.3, a Bayesian perspective is taken, which admits straightforwardly computable error bounds for GP regression. Moreover, this approach allows to provide accuracy guarantees for individual GP model components, e.g., when kernels with an additive structure are employed for regression.

Challenge 2 is addressed by the derivation of tracking error bounds for two classes of control laws in Chapter 3. While our time-varying bounds for linear systems are limited to a relatively narrow class of linear control laws, the derived results for controllers designed using the certainty equivalence principle cover many practically found control design techniques. Therefore, existing proofs for performance guarantees of many GP-based control laws can be considered as special cases of our results.

Challenge 3 is met by the development of a kernel-specific measure for the density of training data in GP models, which establishes a direct relationship between the training data and our derived control performance guarantees. Our analysis clearly shows that data can be of different importance for performance certificates, which offers a new degree of freedom in the design of learning-based control laws. By proposing three different techniques for learning during closed-loop control, we demonstrate how this new flexibility can be exploited to improve the tracking accuracy through learning with GPs.

Challenge 4 is addressed by the introduction of GP model aggregation for enabling online learning and learning in multi-agent systems. We theoretically analyze the computational complexity of the proposed LoG-GP method and demonstrate its capability for online learning in numerical simulations and real-world experiments. Moreover, the inheritance of uniform error bounds from exact GP models is formally proven. This proof is extended for the consensus-based aggregation, such that scalable GP models in multi-agent systems are enabled.

Challenge 5 is met by the development of online learning control architectures tailored towards the restrictions on real-world systems. The proposed asynchronous architecture ensures high robustness against time-varying computation times of GP model evaluations and is demonstrated to work excellently in a physical human-robot interaction scenario experimentally evaluated on 9 participants. The networked control architecture alleviates the high memory requirements of GP-based online learning while leaving performance guarantees unaffected.

7.3. Future Directions

While this thesis addresses many crucial challenges to enable the theoretically supported application of GP-based control laws on real-world systems, several important open problems remain to be solved in future works.

GP-based observer and filter design Even though state measurements are a common requirement for many control design techniques, they are often not available in practice. This makes it crucial to estimate the system states from the available sensor measurements, which requires filters and observers frequently developed using model-based design techniques. Since accurate models are not available when employing GP-based control laws, this necessitates the development of learning-based state estimators. While initial steps have been taken towards the GP-based observer and filter design [240, 241], these results are tailored to specific problems. Moreover, fundamental problems such as suitable training data generation schemes have not been investigated. Therefore, the field of GP-based observer and filter design remains a relevant and challenging problem for future research.

Learning-based control with state constraints Due to the tracking accuracy guarantees, our derived results can enable the satisfaction of state constraints together with a planner providing suitable reference trajectories. However, this might not be an optimal solution, such that the problem of designing learning-based controllers for ensuring state constraints is far larger. Often referred to as safety, the satisfaction of state constraints in learning-based control systems has experienced massive attention in the last few years, but methods capable of permanently preventing constraint violations with high probability are limited to specific constraint formulations. GP models can play a crucial role in these problems, and the principled strength of probabilistic models for deriving flexible safety guarantees has been demonstrated [242]. Nevertheless, the development of a complete theoretical and practical framework for GP-based control with state constraints remains an open problem.

Life-long learning with GPs In order to learn models in everyday life situations, it is probably not sufficient to simply continue updating the same GP model. For example, environmental conditions often change suddenly and spontaneously, and physical contacts can cause instantaneous modifications of system dynamics. Thus, there is a need for a form of knowledge representation that selects models, decides upon the necessity of new models, and merges old ones when situations are considered sufficiently similar. GP models are particularly suitable for such a knowledge representation since their probabilistic nature can

enable the detection of sudden situation changes, and models can be easily updated using new data. Therefore, the connection of GP models with suitable knowledge representations has great potential to pave the way from machine learning to true artificial intelligence in physical systems.

A. Appendix

A.1. Fundamental Results from Linear Algebra

In this thesis, we use several identities from linear algebra to reformulate matrix inverses. While the employed equalities can be found in many reference books on linear algebra, we follow the notation and presentation of [243]. The most important result is known as the Woodbury matrix inversion lemma.

Lemma A.1 ([243, Corollary 3.9.8])**.** *Let* $\boldsymbol{A} \in \mathbb{R}^{d_1 \times d_1}$, $\boldsymbol{B} \in \mathbb{R}^{d_1 \times d_2}$, $\boldsymbol{C} \in \mathbb{R}^{d_2 \times d_1}$ *and* $\boldsymbol{D} \in \mathbb{R}^{d_2 \times d_2}$ *be arbitrary matrices. If* $\boldsymbol{A}$, $\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$, *and* $\boldsymbol{D}$ *are non-singular, then,* $\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$ *is nonsingular and*

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}\boldsymbol{C})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{D}^{-1} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}. \tag{A.1}$$

Based on the Woodbury matrix inversion lemma, several important equalities for special cases can be straightforwardly shown. One of them is the matrix push through identity.

Corollary A.1 ([243, Fact 3.20.6])**.** *Let* $\boldsymbol{A} \in \mathbb{R}^{d_1 \times d_2}$ *and* $\boldsymbol{B} \in \mathbb{R}^{d_2 \times d_1}$ *be arbitrary matrices. Assume that* $\boldsymbol{I} + \boldsymbol{A}\boldsymbol{B}$ *is non-singular, then,*

$$(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{B})^{-1}\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{I} + \boldsymbol{B}\boldsymbol{A})^{-1}. \tag{A.2}$$

In addition to the matrix push through identity, the Sherman-Woodbury-Morrison formula is a direct consequence of the Woodbury matrix inversion lemma.

Corollary A.2. *[243, Fact 3.21.3] Let* $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ *and* $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^{d}$ *be an arbitrary matrix and vectors, respectively. If* $\boldsymbol{A}$ *is non-singular, then,*

$$(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}. \tag{A.3}$$

A.2. Lyapunov Stability Theory

Even though we do not directly employ results from Lyapunov stability theory, the proofs follow similar ideas, and many concepts are adopted in this thesis. Therefore, a brief introduction to Lyapunov's direct method for stability analysis is provided here, which closely follows the presentation in [113]. Lyapunov's direct method is concerned with autonomous dynamical systems

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \tag{A.4}$$

where $\boldsymbol{f} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ is a continuous function. An equilibrium of an autonomous system is defined as a state $\boldsymbol{x} \in \mathbb{R}^{d_x}$, such that $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$. Without loss of generality, we assume that the origin is an equilibrium in the following, i.e., $\boldsymbol{f}(\mathbf{0}) = \mathbf{0}$. Note that this assumption does not pose a relevant restriction since a change of variables can always be used to ensure it.

Lyapunov stability theory investigates the asymptotic behavior of solutions $\boldsymbol{x}(\cdot)$ of dynamical systems (A.4) in the neighborhood of equilibria. Important types of behaviors are formalized using the concept of stability as defined in the following.

Definition A.1 ([113, Definition 4.1])**.** *The equilibrium point $\boldsymbol{x} = \mathbf{0}$ of* (A.4) *is*

- *stable if, for each $c_1 \geq 0$, there is $c_2 > 0$ such that*

$$\|\boldsymbol{x}(0)\| < c_2 \qquad \Rightarrow \qquad \|\boldsymbol{x}(t)\| \leq c_1, \ \forall t \in \mathbb{R}_{0,+}; \tag{A.5}$$

- *asymptotically stable if it is stable and c_2 can be chosen such that*

$$\|\boldsymbol{x}(0)\| < c_2 \qquad \Rightarrow \qquad \lim_{t \to 0} \boldsymbol{x}(t) = \mathbf{0}; \tag{A.6}$$

- *unstable if it is not stable.*

Since we generally cannot determine closed-form solutions for nonlinear dynamical systems (A.4), directly using the conditions in Definition A.1 is not possible. Therefore, the behavior of a proxy function $V : \mathbb{R}^{d_x} \to \mathbb{R}_{0,+}$, the so called Lyapunov candidate, is investigated. The theoretical foundation of this approach, which is commonly referred to as Lyapunov's direct method, is given by the following theorem.

Theorem A.1 ([113, Theorem 4.1])**.** *Let $\boldsymbol{x} = \mathbf{0}$ be an equilibrium point of* (A.4) *and $\mathbb{S} \subset \mathbb{R}^{d_x}$ be a domain containing $\boldsymbol{x} = \mathbf{0}$. Let $V : \mathbb{R}^{d_x} \to \mathbb{R}_{0,+}$ be a continuously differentiable function such that*

$$V(\mathbf{0}) = \mathbf{0}, \qquad V(\boldsymbol{x}) > 0, \forall \boldsymbol{x} \in \mathbb{R}^{d_x} \setminus \{\mathbf{0}\} \tag{A.7}$$

$$\dot{V}(\boldsymbol{x}) \leq 0, \forall \boldsymbol{x} \in \mathbb{S}. \tag{A.8}$$

Then, $\boldsymbol{x} = \mathbf{0}$ is stable. Moreover, if

$$\dot{V}(\boldsymbol{x}) < 0, \forall \boldsymbol{x} \in \mathbb{S} \setminus \{\mathbf{0}\}, \tag{A.9}$$

then, $\boldsymbol{x} = \mathbf{0}$ is asymptotically stable.

Once a Lyapunov candidate has been chosen, this theorem provides stability conditions that can be directly evaluated. However, the results are only qualitative due to the definition of the stability concepts.

In order to obtain quantitative guarantees for the evolution of the system state $\boldsymbol{x}(t)$ in the proximity of an equilibrium, comparison functions can be employed. We define them as follows.

Definition A.2 ([113, Definition 4.2])**.** *A continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$.*

In our introduction of a Lyapunov candidate in Definition 3.2, we employ class $\mathcal{K}$ functions to ensure $V(\mathbf{0}) = 0$ using the requirement

$$\alpha_1(\|\boldsymbol{x}\|) \leq V(\boldsymbol{x}) \leq \alpha_2(\|\boldsymbol{x}\|) \quad \forall \boldsymbol{x} \in \mathbb{R}^{d_x}. \tag{3.35 revisited}$$

Moreover, they allow us to quantify the impact of model errors on the tracking accuracy guarantees. In Lyapunov stability theory, they are similarly used to derive convergence rates for state trajectory $\boldsymbol{x}(\cdot)$ to the equilibrium $\mathbf{0}$. This is exemplarily shown using the following theorem, which is a marginal adaptation of [113, Theorem 4.8].

Theorem A.2. *Let $\boldsymbol{x} = \mathbf{0}$ be an equilibrium point of Eq. (A.4) and $\mathbb{S} \subset \mathbb{R}^{d_x}$ be a domain containing $\boldsymbol{x} = \mathbf{0}$. Moreover, assume there exists a Lyapunov candidate $V : \mathbb{R}^{d_x} \to \mathbb{R}_{0,+}$ and class $\mathcal{K}$ functions $\alpha_1, \alpha_2, \alpha_3 : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$, such that*

$$\nabla^T V(\boldsymbol{x}) \boldsymbol{f}(\boldsymbol{x}) \leq -\alpha_3(\boldsymbol{x}) \tag{A.10}$$

and (3.35) hold for all $\boldsymbol{x} \in \mathbb{S}$. Then, the equilibrium $\boldsymbol{x} = \mathbf{0}$ is asymptotically stable. Moreover, if there exists a Euclidean ball around the origin, which is contained in $\mathbb{S}$, the state trajectories satisfy

$$\|\boldsymbol{x}(t)\| \leq \tilde{\alpha}(\|\boldsymbol{x}(0)\|, t) \tag{A.11}$$

in a neighborhood of the equilibrium, where $\tilde{\boldsymbol{\alpha}} : \mathbb{R}^{d_x} \times \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$ is a continuous function, which is increasing in its first argument, decreasing in its second argument and $\tilde{\alpha}(0, t) = 0$, for all $t \in \mathbb{R}_{0,+}$, $\lim_{t \to \infty} \tilde{\alpha}(\|\boldsymbol{x}\|, t) = 0$ for all $\boldsymbol{x} \in \mathbb{R}^{d_x}$.

Notation

Acronyms and Abbreviations

ARD	automatic relevance determination
GP	Gaussian process
i.i.d.	independently and identically distributed
RKHS	reproducing kernel Hilbert space
SE	squared exponential

Conventions & Operators

$\boldsymbol{z}, \boldsymbol{Z}$	lower/upper case bold symbols for vectors/matrices
$\boldsymbol{z}_i, z_{ij}$	i-th column/ element in column i and row j of a matrix $\boldsymbol{Z}$
z_i	i-th element of a vector $\boldsymbol{z}$
d_z	dimension of a vector $\boldsymbol{z} \in \mathbb{R}^{d_z}$
$\mathbf{0}$	vector/matrix of zeros of proper dimensions
$\mathbf{1}_N \in \mathbb{R}^N$	vector containing only ones
$(0, 1)$	open unit interval
$[0, 1]$	closed unit interval
$a \vert b$	random variable a conditioned on b
$.^T$	transpose of a vector or matrix
$.^{-1}$	inverse of a scalar or matrix
$.^{(n)}$	n-th training data point
$\dot{z}$	derivative of z with respect to time
$\vert \cdot \vert$	absolute value of a real number or cardinality of a set
$\Vert \cdot \Vert$	Euclidean norm of a vector or spectral norm of a matrix

$\|\cdot\|_k$	norm of a function in the reproducing kernel Hilbert space attached to a kernel
$\|\cdot\|_\infty$	infinity norm of a function
∇	Nabla operator
$<\cdot,\cdot>$	inner product
$\circ$	Hadamard product
arg max	arguments of the maximum of a function
arg min	arguments of the minimum of a function
$\det(\cdot)$	determinant of a matrix
$\mathrm{diag}(\cdot)$	operator, which arranges the elements of its argument into a diagonal matrix
$\mathbb{E}[\cdot]$	expectation operator, which returns the mean value of its argument
$\exp(\cdot), e^{\cdot}$	exponential function of a scalar or matrix
$\mathcal{F}[\cdot]$	Fourier transform of a function
$f(\cdot)$	shorthand notation to denote a function $f : \mathbb{R}^d \to \mathbb{R}$
$\boldsymbol{f}(\boldsymbol{Z}), \boldsymbol{F}(\boldsymbol{Z})$	concatenation of all evaluations of a scalar function $f : \mathbb{R}^d \to \mathbb{R}$/ vector field $\boldsymbol{f} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ at inputs $\boldsymbol{z}_i$
$\cdot \sim \mathcal{GP}(\cdot,\cdot)$	random function follows a Gaussian process distribution with mean and covariance function
$I(\cdot,\cdot)$	mutual information between two random variables
$\boldsymbol{I}_N \in \mathbb{R}^{N\times N}$	identity matrix
inf	infimum of a function
L_f	Lipschitz constant of function $f(\cdot)$
$\log(\cdot)$	natural logarithm of a positive scalar
$M(\tau, \mathbb{S})$	τ-covering number of a set $\mathbb{S}$
max	maximum of a function
min	minimum of a function
$\cdot \sim \mathcal{N}(\cdot,\cdot)$	random variable follows Gaussian distribution with mean and variance
$N_k(\varrho, \mathbb{S})$	ϱ-packing number of a set $\mathbb{S}$ with respect to a kernel
$\mathcal{O}(\cdot)$	big O notation describing the asymptotic behavior of functions

$\mathcal{O}_p(\cdot)$	big O in probability notation describing the probabilistic asymptotic behavior of functions
$\mathbb{P}(\cdot)$	probability of an event
$p(\cdot)$	probability density function
$\cdot \sim \mathcal{U}(\cdot, \cdot)$	uniform distribution
sup	supremum of a function
$\underline{\lambda}(\cdot), \bar{\lambda}(\cdot)$	minimal and maximal real parts of the eigenvalues of a matrix
$\cdot \sim \chi^2_N$	random variable follows a chi-square distribution with N degrees of freedom

Sets & Spaces

$\mathbb{A}_m$	graph branch for leaf m
$\mathbb{B}_r(\boldsymbol{c}) \subset \mathbb{R}^d$	Euclidean ball with radius r and center $\boldsymbol{c}$
$\mathbb{D}$	training data set
$\mathbb{D}_{\text{loc}}$	data set stored in the memory of the technical system
$\mathbb{D}_N$	training data set with N data pairs
$\mathbb{H}_0$	linear span of feature functions defined by a kernel
$\mathbb{H}_k$	reproducing kernel Hilbert space attached to a kernel
$\overline{\mathbb{H}}$	completion of $\mathbb{H}$
$\mathbb{I}$	time interval
$\mathbb{K}_h(\boldsymbol{z})$	subset of $\mathbb{D}$, which excludes training samples with a negative effect on the considered bound
$\mathcal{L}_p$	function space consisting of all p-fold integrable functions
$\mathcal{G}$	(undirected) graph
$\mathcal{V}$	graph vertices
$\mathcal{E}$	graph edges
$\mathbb{M}$	set of leaves in a graph
$\mathbb{N}, \mathbb{N}_+$	set of non-negative/ positive integer numbers
$\mathbb{R}, \mathbb{R}_+, \mathbb{R}_{0,+}$	set of real/ postive real/ non-negative real numbers

$\mathbb{S} \subset \mathbb{R}^d$	compact subset of $\mathbb{R}^d$
$\mathbb{T}$	tube
$\mathbb{U}$	set of control inputs
$\mathbb{W}$	set of potentially active models during a time window
$\mathbb{X}$	set of states
$\mathbb{X}_0$	set of initial states
$\mathbb{X}_l$	set active subsets of input data

Functions

$f : \mathbb{R}^d \to \mathbb{R}$	general scalar function, refer to definition
$\boldsymbol{f} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$	general vector field, refer to definition
$\boldsymbol{f} : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^{d_x}$	general dynamics
$\boldsymbol{G} : \mathbb{X} \to \mathbb{U}$	scaling of the control input for a control affine system
$g : \mathbb{X} \to \mathbb{R}$	scaling of the control input for a single input control affine system
$H : \mathbb{R} \to \{0, 1\}$	unit step function
$h : \mathbb{R}^d \to \mathbb{R}_+$	density measure
$k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{0,+}$	positive definite kernel
$\boldsymbol{k} : \mathbb{R}^d \to \mathbb{R}^N$	kernel vector with elements $k(\cdot, \boldsymbol{z}^{(n)})$ for $N \in \mathbb{N}$ training samples $\boldsymbol{z}^{(n)} \in \mathbb{R}^{d_z}$, $n = 1, \ldots, N$
$N_{\text{trig}} : \mathbb{R}_{0,+} \to \mathbb{N}$	number of triggering instances up to time t
$\boldsymbol{p}_{\mathbf{av}} : \mathbb{R}_{0,+} \to \mathbb{R}^d$	dynamic average of local reference signals
$\boldsymbol{p}_m : \mathbb{R}_{0,+} \to \mathbb{R}^d$	local reference signal (for agent m)
$\boldsymbol{p}^n : \mathbb{R}^{d_1} \to [0, 1]^{d_2}$	vector-valued conditional probability
$V : \mathbb{R}^d \to \mathbb{R}_{0,+}$	generic Lyapunov function
$\boldsymbol{x}_{\text{ref}} : \mathbb{R}_{0,+} \to \mathbb{X}$	reference trajectory
$\boldsymbol{x}_l : \mathbb{R}_{0,+} \to \mathbb{X}$	leader state/ reference trajectory for agents
$\boldsymbol{\epsilon}_{\boldsymbol{x}} : \mathbb{R}_{0,+} \to \mathbb{R}^d$	measurement disturbance of state $\boldsymbol{x}$
$\eta : \mathbb{R}^d \to \mathbb{R}_{0,+}$	(Gaussian process) error bound

$\tilde{\eta}_{\text{loc}} : \mathbb{R}^d \to \mathbb{R}_{0,+}$	individual/local (Gaussian process) model error bound
$\tilde{\eta}_{\text{tr}} : \mathbb{R}^d \to \mathbb{R}_{0,+}$	(Gaussian process) transient consensus error bound
$\mu : \mathbb{R}^d \to \mathbb{R}$	posterior Gaussian process mean function, model of unknown function
$\tilde{\mu} : \mathbb{R}^d \to \mathbb{R}$	aggregated posterior Gaussian process mean function, model of unknown function
$\check{\boldsymbol{\mu}} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$	exact GP aggregation based on local information
$\xi : \mathbb{R}_{0,+} \to \mathbb{R}$	solution of a differential equation defining the tracking error bound
$\boldsymbol{\mu} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$	posterior mean functions of multiple GPs concatenated as vector
$\mu^0 : \mathbb{R}^d \to \mathbb{R}$	prior Gaussian process mean function
$\mu^N : \mathbb{R}^d \to \mathbb{R}$	posterior Gaussian process mean function given N training samples
$\pi_{\text{lin}} : \mathbb{X} \to \mathbb{U}$	linear control law
$\boldsymbol{\pi} : \mathbb{X} \times \mathbb{R}_{0,+} \to \mathbb{U}$	control law
$\sigma^2 : \mathbb{R}^d \to \mathbb{R}_{0,+}$	posterior Gaussian process variance function with unspecified number of training samples
$\tilde{\sigma}^2 : \mathbb{R}^d \to \mathbb{R}_{0,+}$	aggregated posterior Gaussian process variance function with unspecified number of training samples
$\boldsymbol{\sigma}^2 : \mathbb{R}^{d_1} \to \mathbb{R}_{0,+}^{d_2}$	posterior variance functions of multiple GPs concatenated as vector
$(\sigma^N)^2 : \mathbb{R}^d \to \mathbb{R}_{0,+}$	posterior Gaussian process variance function given N training samples
$\upsilon : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$	tracking error bound
$\boldsymbol{\phi} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$	finite dimensional feature vector
$\boldsymbol{\Phi} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$	concatenation of finite-dimensional feature vectors
$\boldsymbol{\phi} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$	finite dimensional feature vector
$\boldsymbol{\chi} : \mathbb{R}_{0,+} \to \mathbb{R}^d$	consensus state for distributed GP aggregation
$\psi : \mathbb{R}^{d_\psi} \to \mathbb{R}$	scalar function for model aggregation
$\boldsymbol{\psi}_\omega : \mathbb{R}^2 \to \mathbb{R}^{d_\psi}$	vector-valued map of individual model mean and variance for aggregation

Variables

$\boldsymbol{A}$	state transition matrix
$\boldsymbol{A}_{cl}$	closed-loop system state transition matrix
$\boldsymbol{A}^{\text{ad}}$	weighted adjacency matrix
B_{com}	finite bandwidth
$\boldsymbol{B}^{l}$	augmentation of leader states in graph Laplacian
$\boldsymbol{b}$	input vector
c	general constant, refer to definition
$\boldsymbol{e}$	tracking error
F	bound on $\|\dot{\boldsymbol{x}}(t)\|$
h^m	graph depth of leaf m
h	fill distance
i	general integer, refer to definition
j	general integer, refer to definition
$\boldsymbol{K}$	gram matrix of kernel $k(\cdot,\cdot)$
k_c	scalar gain
N	number of training data points
$\bar{N}$	data capacity limit
$\bar{N}_{\text{mem}}$	memory capacity limit
n	general integer, refer to definition
$\boldsymbol{L}$	graph Laplacian
$\boldsymbol{L}^{\boldsymbol{K}}$	lower triangular matrix in Cholesky factorization
$\tilde{\boldsymbol{L}}$	augmented Laplacian
L_f	Hölder/Lipschitz continuity coefficient for function $f(\cdot)$
l_i	kernel length scale
M	number of nodes in a graph
p_f	Hölder continuity order for function $f(\cdot)$
$\boldsymbol{P}$	solution of the continuous-time Lyapunov equation

$\boldsymbol{Q}$	positive definite (weighting) matrix
T_{access}	time for accessing data
$\bar{T}^c$	upper bound for execution time
T_d	network delay time
T_{dis}	temporal discretization time
T_m^c	execution time for node m
$\bar{T}_m^c$	computational delay bound for node m
T_p	period time
T_s	sampling time
T_{trans}	data transmision time
t	time
$\boldsymbol{U}$	matrix obtained via eigendecomposition $\boldsymbol{A}_{cl} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$
$u, \boldsymbol{u}$	scalar/ multi-dimensional control input
$r, \boldsymbol{r}$	scalar/ multi-dimensional reference signal
$\boldsymbol{w}$	weight for linear regression
$x, \boldsymbol{x}$	scalar/ multi-dimensional state
y	training target
$\boldsymbol{y}$	concatenation of training targets
$\boldsymbol{Z}$	concatenation of training inputs
$\boldsymbol{z}$	(training) input
$\boldsymbol{\alpha}$	solution vector of a linear system of equations
β	scaling factor for posterior standard deviation in uniform error bounds
$\beth_m$	control input for agent m
Γ	bound on norm in the reproducing kernel Hilbert space
γ_N	maximum information gain after N training samples
δ	violation probability
$\boldsymbol{\epsilon}$	observation noise
$\bar{\epsilon}_x$	input disturbance bound

$\bar{\epsilon}_y$	output disturbance bound
$\tilde{\epsilon}$	total (input and output) disturbance
$\boldsymbol{\varepsilon}_m$	consensus error of agent m
$\boldsymbol{\varepsilon}_m^j$	joint consensus error in j-th dimension of agent m
$\boldsymbol{\mathcal{E}}_m$	joint consensus error in all dimensions of agent m
ι	parameter of the consensus tracking error bound
$\boldsymbol{\theta}$	control gain vector
$\tilde{\boldsymbol{\theta}}$	hurwit gain vector
$\boldsymbol{\vartheta}$	hyperparameter vector
$\boldsymbol{\mu}^0$	mean data vector
$\boldsymbol{\mu}_w, \boldsymbol{\mu}_w^N$	posterior mean of weights $\boldsymbol{w}$
$\boldsymbol{\mu}_w^0$	prior mean of weight $\boldsymbol{w}$
$\boldsymbol{\nu}$	filtered state for cooperative tracking control
ρ	log ratio for assessing data density
$\boldsymbol{\Sigma}_w, \boldsymbol{\Sigma}_w^N$	posterior variance of weights $\boldsymbol{w}$
$\boldsymbol{\Sigma}_w^0$	prior variance of weight $\boldsymbol{w}$
σ_f^2	kernel signal variance
σ_{on}^2	observation noise variance
$\tilde{\sigma}_{\text{on}}^2$	scaling constant of sub-Gaussian distribution assumed for the observation noise
τ	virtual grid constant
$\bar{\upsilon}$	constant tracking error bound
$\boldsymbol{\Upsilon}$	positive definite matrix in consensus tracking error bound

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 2nd ed. New York, NY: John Wiley & Sons, 2005.

[2] N. H. A. Hamid, M. M. Kamal, and F. H. Yahaya, "Application of PID Controller in Controlling Refrigerator Temperature," in *Proceedings of the International Colloquium on Signal Processing and Its Applications*, 2009, pp. 378–384.

[3] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A Survey of Distributed Optimization and Control Algorithms for Electric Power Systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[4] K. J. Astrom, "Theory and Applications of Adaptive Control," *Automatica*, vol. 19, no. 5, pp. 471–486, 1983.

[5] K. J. Åström and B. Wittenmark, "A Survey of Adaptive Control Applications," in *Proceedings of the IEEE Conference on Decision and Control*, 1995, pp. 649–654.

[6] Qin, S. Joe and Badgwell, Thomas A, "A Survey of Industrial Model Predictive Control Rechnology," *Control Engineering Practice*, vol. 11, pp. 733–764, 2003.

[7] K. J. Åström and P. R. Kumar, "Control: A perspective," *Automatica*, vol. 50, pp. 3–43, 2014.

[8] P. Maciejasz, J. Eschweiler, K. Gerlach-Hahn, A. Jansen-Troy, and S. Leonhardt, "A Survey on Robotic Devices for Upper Limb Rehabilitation," *Journal of Neuroengineering and Rehabilitation*, vol. 11, no. 3, pp. 1–29, 2014.

[9] R. Gassert and V. Dietz, "Rehabilitation Robots for the Treatment of Sensorimotor Deficits: A Neurophysiological Perspective," *Journal of NeuroEngineering and Rehabilitation*, vol. 15, no. 1, pp. 1–15, 2018.

[10] H. Yu, S. Huang, G. Chen, Y. Pan, and Z. Guo, "Human-Robot Interaction Control of Rehabilitation Robots with Series Elastic Actuators," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1089–1100, 2015.

[11] D. Zhang, T. H. Guan, F. Widjaja, and W. T. Ang, "Functional Electrical Stimulation in Rehabilitation Engineering: A Survey," in *Proceedings of the International Convention on Rehabilitation Engineering and Assistive Technology in Conjunction with 1st Tan Tock Seng Hospital Neurorehabilitation Meeting*, 2007, pp. 221–226.

[12] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010, pp. 1642–1648.

[13] M. Guiggiani, *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*, 2nd ed. Cham, Switzerland: Springer International Publishing, 2018.

[14] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, 2011.

[15] B. H. Wang, D. B. Wang, Z. A. Ali, B. Ting Ting, and H. Wang, "An Overview of Various Kinds of Wind Effects on Unmanned Aerial Vehicle," *Measurement and Control*, vol. 52, no. 7-8, pp. 731–739, 2019.

[16] F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. Van den Hof, "Systems & Control for the Future of Humanity, Research Agenda: Current and Future Roles, Impact and Grand Challenges," *Annual Reviews in Control*, vol. 43, pp. 1–64, 2017.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-Level Control through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," in *Proceedings of the International Conference on Learning Representations*, 2016.

[21] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of Real-World Reinforcement Learning," in *ICML Workshop on Real-Life Reinforcement Learning*, 2019. [Online]. Available: http://arxiv.org/abs/1904.12901

[22] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press, 2006.

[23] M. P. Deisenroth, "Efficient Reinforcement Learning using Gaussian Processes," Ph.D. dissertation, Karlsruher Institut für Technologie, 2009.

[24] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model Learning with Local Gaussian Process Regression," *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[25] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal Kernels," *Journal of Machine Learning Research*, vol. 7, pp. 2651–2667, 2006.

[26] M. K. Helwa, A. Heins, and A. P. Schoellig, "Provably Robust Learning-Based Approach for High-Accuracy Tracking Control of Lagrangian Systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1587–1594, 2019.

[27] J. Umlauft, T. Beckers, A. Capone, A. Lederer, and S. Hirche, "Smart Forgetting for Safe Online Learning with Gaussian Processes," in *Proceedings of the Conference on Learning for Dynamics and Control*, 2020, pp. 160–169.

[28] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-Driven Model Predictive Control for Trajectory Tracking With a Robotic Arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[29] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust Constrained Learning-based NMPC enabling Reliable Mobile Robot Path Tracking," *International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547–1563, 2016.

[30] A. Gahlawat, P. Zhao, A. Patterson, N. Hovakimyan, and E. A. Theodorou, "L1-GP: L1 Adaptive Control with Bayesian Learning," in *Learning for Dynamics & Control*, 2020, pp. 1–15.

[31] J. Umlauft and S. Hirche, "Feedback Linearization based on Gaussian Processes with Event-Triggered Online Learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 10, pp. 4154–4169, 2019.

[32] M. Greeff and A. P. Schoellig, "Exploiting Differential Flatness for Robust Learning-based Tracking Control using Gaussian Processes," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1121–1126, 2021.

[33] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela, "Bayesian Nonparametric Adaptive Control using Gaussian Processes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 537–550, 2015.

[34] T. Beckers, D. Kulić, and S. Hirche, "Stable Gaussian Process based Tracking Control of Euler-Lagrange Systems," *Automatica*, vol. 103, no. 23, pp. 390–397, 2019.

[35] A. Lederer, J. Umlauft, and S. Hirche, "Uniform Error Bounds for Gaussian Process Regression with Application to Safe Control," in *Advances in Neural Information Processing Systems*, 2019, pp. 659–669.

[36] ——, "Episodic Gaussian Process-Based Learning Control with Vanishing Tracking Errors," 2023. [Online]. Available: http://arxiv.org/abs/2307.04415

[37] A. Lederer, A. Capone, J. Umlauft, and S. Hirche, "How Training Data Impacts Performance in Learning-based Control," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 905–910, 2021.

[38] A. Lederer, A. Ordonez Conejo, K. Maier, W. Xiao, J. Umlauft, and S. Hirche, "Gaussian Process-Based Real-Time Learning for Safety Critical Applications," in *International Conference on Machine Learning*, 2021, pp. 6055–6064.

[39] A. Lederer, Z. Yang, J. Jiao, and S. Hirche, "Cooperative Control of Uncertain Multi-Agent Systems via Distributed Gaussian Processes," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 3091–3098, 2023.

[40] S. Tesfazgi, A. Lederer, J. F. Kunz, A. J. Ordóñez Conejo, and S. Hirche, "Model-Based Robot Control with Gaussian Process Online Learning: An Experimental Demonstration," To appear in *Proceedings of the IFAC World Congress*, 2021. [Online]. Available: https://arxiv.org/pdf/2110.00481.pdf

[41] A. Lederer, M. Zhang, S. Tesfazgi, and S. Hirche, "Networked Online Learning for Control of Safety-Critical Resource-Constrained Systems based on Gaussian Processes," in *Proceedings of the IEEE Conference on Control Technology and Applications*, 2022, pp. 1285–1292.

[42] N. Cressie, *Statistics for Spatial Data*, 2nd ed. Hoboken, New Jersey: John Wiley & Sons, 2015.

[43] N. Wiener, *Extrapolation, Interpolation and Smoothing of Stationary Time Series*. Cambridge, Massachusetts: MIT Press, 1949.

[44] A. N. Kolmogorov, "Interpolation und Extrapolation von stationären zufälligen Folgen," *Izvestiya Akademii Nauk SSSR*, vol. 5, no. 1, pp. 3–14, 1941.

[45] R. J. Adler, *An Introduction to Continuity, Extrema, and Related Topics for General Gaussian Processes*. Institute of Mathematical Statistics, 1990.

[46] Z. Ghahramani, "Probabilistic Machine Learning and Artificial Intelligence," *Nature*, vol. 27, pp. 452–459, 2015.

[47] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer Science+Business Media, 2006.

[48] R. A. Freeman and P. V. Kokotovic, *Robust Nonlinear Control Design*, 1st ed. Birkhäuser Boston, 1996.

[49] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, Massachusetts: The MIT Press, 2002.

[50] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse Spectrum Gaussian Process Regression," *Journal of Machine Learning Research*, vol. 11, pp. 1865–1881, 2010.

[51] A. Gijsberts and G. Metta, "Real-Time Model Learning using Incremental Sparse Spectrum Gaussian Process Regression," *Neural Networks*, vol. 41, pp. 59–69, 2013.

[52] D. K. Duvenaud, "Automatic Model Construction with Gaussian Processes," Ph.D. dissertation, University of Cambridge, 2014.

[53] G. Evangelisti and S. Hirche, "Physically Consistent Learning of Conservative Lagrangian Systems with Gaussian Processes," in *Proceedings of the IEEE Conference on Decision and Control*, 2022, pp. 4078–4085.

[54] M. A. Álvarez, L. Rosasco, and N. D. Lawrence, "Kernels for Vector-valued Functions: A Review," *Foundations and Trends in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2011.

[55] A. Lederer, A. Capone, T. Beckers, J. Umlauft, and S. Hirche, "The Impact of Data on the Stability of Learning-Based Control," in *Proceedings of the Conference on Learning for Dynamics and Control*, vol. 144, 2021, pp. 623–635.

[56] J. Kirschner, M. Mutný, N. Hiller, R. Ischebeck, and A. Krause, "Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 5959–5971.

[57] S. Sundararajan and S. S. Keerthi, "Predictive Approaches for Choosing Hyperparameters in Gaussian Processes," *Neural Computation*, vol. 13, no. 5, pp. 1103–1118, 2001.

[58] A. van der Vaart and H. van Zanten, "Information Rates of Nonparametric Gaussian Process Methods," *Journal of Machine Learning Research*, vol. 12, pp. 2095–2119, 2011.

[59] Z. M. Wu and R. Schaback, "Local Error Estimates for Radial Basis Function Interpolation of Scattered Data," *IMA Journal of Numerical Analysis*, vol. 13, no. 1, pp. 13–27, 1993.

[60] R. Schaback, "Improved Error Bounds for Scattered Data Interpolation by Radial Basis Functions," *Mathematics of Computation*, vol. 68, no. 225, pp. 201–217, 2002.

[61] H. Wendland, *Scattered Data Approximation*. Cambridge University Press, 2004.

[62] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, "Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences," pp. 1–64, 2018. [Online]. Available: http://arxiv.org/abs/1807.02582

[63] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press, 2004.

[64] S. Mendelson, "Improving the Sample Complexity using Global Data," *IEEE Transactions on Information Theory*, vol. 48, no. 7, pp. 1977–1991, 2002.

[65] L. Shi, "Learning Theory Estimates for Coefficient-based Regularized Regression," *Applied and Computational Harmonic Analysis*, vol. 34, no. 2, pp. 252–265, 2013.

[66] E. T. Maddalena, P. Scharnhorst, and C. N. Jones, "Deterministic Error Bounds for Kernel-based Learning Techniques under Bounded Noise," *Automatica*, vol. 134, p. 109896, 2021.

[67] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, 2012.

[68] S. R. Chowdhury and A. Gopalan, "On Kernelized Multi-armed Bandits," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 844–853.

[69] C. Fiedler, C. W. Scherer, and S. Trimpe, "Practical and Rigorous Uncertainty Bounds for Gaussian Process Regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 7439–7447.

[70] B. Laurent and P. Massart, "Adaptive Estimation of a Quadratic Functional by Model Selection," *The Annals of Statistics*, vol. 28, no. 5, pp. 1302–1338, 2000.

[71] J. Mercer, "Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 209, no. 441-458, pp. 415–446, 1909.

[72] N. Aronszajn, "Theory of Reproducing Kernels," *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

[73] I. Steinwart, "Reproducing Kernel Hilbert Spaces cannot contain all Continuous Functions on a Compact Metric Space," pp. 1–2, 2020. [Online]. Available: http://arxiv.org/abs/2002.03171

[74] P. Scharnhorst, E. T. Maddalena, Y. Jiang, and C. N. Jones, "Robust Uncertainty Bounds in Reproducing Kernel Hilbert Spaces: A Convex Optimization Approach," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2848–2861, 2023.

[75] A. Berlinet and C. Thomas-Agnan, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer US, 2004.

[76] B. C. Csaji and B. Horvath, "Nonparametric, Nonasymptotic Confidence Bands With Paley-Wiener Kernels for Band-Limited Functions," *IEEE Control Systems Letters*, vol. 6, pp. 3355–3360, 2022.

[77] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge, UK: Cambridge University Press, 2018.

[78] A. K. Akametalu, S. Kaynama, J. F. Fisac, M. N. Zeilinger, J. H. Gillula, and C. J. Tomlin, "Reachability-based Safe Learning with Gaussian Processes," in *Proceedings of the IEEE Conference on Decision and Control*, 2014, pp. 1424–1431.

[79] Y. Wang, C. Ocampo-Martinez, and V. Puig, "Robust Model Predictive Control based on Gaussian Processes: Application to Drinking Water Networks," in *Proceedings of the European Control Conference*, 2015, pp. 3292–3297.

[80] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2019.

[81] V. Dhiman, M. J. Khojasteh, M. Franceschetti, and N. Atanasov, "Control Barriers in Bayesian Learning of System Dynamics," *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 214–229, 2023.

[82] A. Capone, A. Lederer, and S. Hirche, "Gaussian Process Uniform Error Bounds with Unknown Hyperparameters for Safety-Critical Applications," in *Proceedings of the International Conference on Machine Learning*, 2022, pp. 2609–2624.

[83] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate Uncertainties for Deep Learning using Calibrated Regression," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 4369–4377.

[84] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. New York, NY: Cambridge University Press, 2013.

[85] M. Omainska, J. Yamauchi, A. Lederer, S. Hirche, and M. Fujita, "Rigid Motion Gaussian Processes With SE(3) Kernel and Application to Visual Pursuit Control," *IEEE Control Systems Letters*, vol. 7, pp. 2665–2670, 2023.

[86] S. Curi, F. Berkenkamp, and A. Krause, "Efficient Model-Based Reinforcement Learning through Optimistic Policy Search and Planning," in *Advances in Neural Information Processing Systems*, 2020.

[87] R. M. Dudley, "The Sizes of Compact Subsets of Hilbert Space and Continuity of Gaussian Processes," *Journal of Functional Analysis*, vol. 1, no. 3, pp. 290–330, 1967.

[88] S. Grünewälder, J.-Y. Audibert, M. Opper, and J. Shawe-Taylor, "Regret Bounds for Gaussian Process Bandit Problems," *Journal of Machine Learning Research*, vol. 9, pp. 273–280, 2010.

[89] M. Talagrand, "Sharper Bounds for Gaussian and Empirical Processes," *The Annals of Probability*, vol. 22, no. 1, pp. 28–76, 1994.

[90] S. Ghosal and A. Roy, "Posterior Consistency of Gaussian Process Prior for Nonparametric Binary Regression," *The Annals of Statistics*, vol. 34, no. 5, pp. 2413–2429, 2006.

[91] S. Särkkä, "Linear Operators and Stochastic Partial Differential Equations in Gaussian Process Regression," in *Proceedings of the International Conference on Artificial Neural Networks and Machine Learning*, 2011, pp. 151–158.

[92] L. P. Swiler, M. Gulian, A. L. Frankel, C. Safta, and J. D. Jakeman, "A Survey of Constrained Gaussian Process Regression: Approaches and Implementation Challenges," *Journal of Machine Learning for Modeling and Computing*, vol. 1, no. 2, pp. 119–156, 2020.

[93] J. Umlauft, A. Lederer, and S. Hirche, "Learning Stable Gaussian Process State Space Models," in *Proceedings of the American Control Conference*, 2017, pp. 1499–1504.

[94] W. Xiao, A. Lederer, and S. Hirche, "Learning Stable Nonparametric Dynamical Systems with Gaussian Process Regression," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1194–1199, 2020.

[95] E. Schulz, M. Speekenbrink, and A. Krause, "A Tutorial on Gaussian Process Regression: Modelling, Exploring, and Exploiting Functions," *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.

[96] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe Controller Optimization for Quadrotors with Gaussian Processes," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 491–496.

[97] C. Fiedler, C. W. Scherer, and S. Trimpe, "Learning Functions and Uncertainty Sets Using Geometrically Constrained Kernel Regression," in *Proceedings of the IEEE Conference on Decision and Control*, 2022, pp. 2141–2146.

[98] J. Kocijan, *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Springer International Publishing, 2016.

[99] M. Liu, G. Chowdhary, B. Castra Da Silva, S. Y. Liu, and J. P. How, "Gaussian Processes for Learning and Control: A Tutorial with Examples," *IEEE Control Systems*, vol. 38, no. 5, pp. 53–86, 2018.

[100] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively Learning Gaussian Process Dynamics," in *Learning for Dynamics and Control*, 2020, pp. 1–11.

[101] A. Capone, G. Noske, J. Umlauft, T. Beckers, A. Lederer, and S. Hirche, "Localized Active Learning of Gaussian Process State Space Models," in *Learning for Dynamics and Control*, 2020, pp. 490–499.

[102] J. Umlauft, L. Pöhler, and S. Hirche, "An Uncertainty-Based Control Lyapunov Approach for Control-Affine Systems Modeled by Gaussian Process," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 483–488, 2018.

[103] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious Model Predictive Control using Gaussian Process Regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2020.

[104] F. Berkenkamp and A. P. Schoellig, "Safe and Robust Learning Control with Gaussian Processes," in *Proceedings of the European Control Conference*, 2015, pp. 2496–2501.

[105] A. von Rohr, M. Neumann-Brosig, and S. Trimpe, "Probabilistic Robust Linear Quadratic Regulators with Gaussian Processes," in *Proceedings of the Conference on Learning for Dynamics and Control*, 2021, pp. 324–335.

[106] J. Umlauft, T. Beckers, and S. Hirche, "Scenario-based Optimal Control for Gaussian Process State Space Models," in *Proceedings of the European Control Conference*, 2018.

[107] A. Marco, P. Hennig, S. Schaal, and S. Trimpe, "On the Design of LQR Kernels for Efficient Controller Learning," in *Proceedings of the IEEE Conference on Decision and Control*, 2017, pp. 5193–5200.

[108] E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig, "Gaussian Process Based Predictive Control for Periodic Error Correction," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 110–121, 2016.

[109] C. Fiedler, C. W. Scherer, and S. Trimpe, "Learning-enhanced Robust Controller Synthesis with Rigorous Statistical and Control-theoretic Guarantees," in *Proceedings of the IEEE Conference on Decision and Control*, 2021, pp. 5122–5129.

[110] A. Capone and S. Hirche, "Backstepping for Partially Unknown Nonlinear Systems Using Gaussian Processes," *IEEE Control Systems Letters*, vol. 3, no. 2, pp. 416–421, 2019.

[111] J. Lunze, *Regelungstechnik 2: Mehrgößensysteme, Digitale Reglung*, 8th ed. Heidelberg, Germany: Springer Vieweg, 2014.

[112] L. Perko, *Differential Equations and Dynamical Systems*, 3rd ed. Springer, 2006.

[113] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2002.

[114] D. Sun, M. J. Khojasteh, S. Shekhar, and C. Fan, "Uncertainty-aware Safe Exploratory Planning using Gaussian Process and Neural Control Contraction Metric," in *Proceedings of the Conference on Learning for Dynamics and Control*, 2021, pp. 728–741.

[115] F. Castaneda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, "Gaussian Process-based Min-norm Stabilizing Controller for Control-Affine Systems with Uncertain Input Effects and Dynamics," in *Proceedings of the American Control Conference*, 2021, pp. 3683–3690.

[116] G. S. Lima, S. Trimpe, and W. M. Bessa, "Sliding Mode Control with Gaussian Process Regression for Underwater Robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 99, no. 3-4, pp. 487–498, 2020.

[117] H. Mania, S. Tu, and B. Recht, "Certainty Equivalence is Efficient for Linear Quadratic Control," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 154–10 164.

[118] M. Korda and I. Mezic, "Optimal Construction of Koopman Eigenfunctions for Prediction and Control," *IEEE Transactions on Automatic Control*, vol. 65, no. 12, pp. 5114–5129, 2020.

[119] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian Optimization with Safety Constraints: Safe Automatic Parameter Tuning in Robotics," *Machine Learning*, pp. 1–35, 2021.

[120] A. Lederer, A. Capone, and S. Hirche, "Parameter Optimization for Learning-based Control of Control-Affine Systems," in *Proceedings of the Conference on Learning for Dynamics and Control*, vol. 120, 2020, pp. 465–475.

[121] Y. Pan and E. A. Theodorou, "Data-driven Differential Dynamic Programming using Gaussian Processes," in *Proceedings of the American Control Conference*, 2015, pp. 4467–4472.

[122] F. Castañeda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, "Pointwise Feasibility of Gaussian Process-based Safety-Critical Control under Model Uncertainty," in *Proceedings of the IEEE Conference on Decision and Control*, 2021, pp. 6762–6769.

[123] A. Lederer, A. Begzadić, N. Das, and S. Hirche, "Learning-Based Control of Elastic Joint Robots via Control Barrier Functions," To appear in *Proceedings of the IFAC World Congress*, 2023. [Online]. Available: http://arxiv.org/abs/2212.00478

[124] P. Jagtap, G. J. Pappas, and M. Zamani, "Control Barrier Functions for Unknown Nonlinear Systems using Gaussian Processes," in *Proceedings of the IEEE Conference on Decision and Control*, 2020, pp. 3699–3704.

[125] A. Krause, A. Singh, and C. Guestrin, "Near-optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies," *Journal of Machine Learning Research*, vol. 9, pp. 235–284, 2008.

[126] P. Hennig and C. J. Schuler, "Entropy Search for Information-efficient Global Optimization," *Journal of Machine Learning Research*, vol. 13, pp. 1809–1837, 2012.

[127] M. L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, 1999.

[128] R. Beatson, O. Davydov, and J. Levesley, "Error Bounds for Anisotropic RBF Interpolation," *Journal of Approximation Theory*, vol. 162, no. 3, pp. 512–527, 2010.

[129] M. Scheuerer, R. Schaback, and M. Schlather, "Interpolation of Spatial Data - A Stochastic or a Deterministic Problem?" *European Journal of Applied Mathematics*, vol. 24, no. 4, pp. 601–629, 2013.

[130] S. Shekhar and T. Javidi, "Gaussian Process Bandits with Adaptive Discretization," *Electronic Journal of Statistics*, vol. 12, pp. 3829–3874, 2018.

[131] M. Opper and F. Vivarelli, "General Bounds on Bayes Errors for Regression with Gaussian Processes," *Advances in Neural Information Processing Systems*, pp. 302–308, 1999.

[132] C. K. I. Williams and F. Vivarelli, "Upper and Lower Bounds on the Learning Curve for Gaussian Processes," *Machine Learning*, vol. 40, pp. 77–102, 2000.

[133] W. Wang and B. Haaland, "Controlling Sources of Inaccuracy in Stochastic Kriging," *Technometrics*, pp. 1–13, 2018.

[134] S. Gershgorin, "Über die Abgrenzung der Eigenwerte einer Matrix," *Bulletin de l'Academie des Sciences de l'URSS. Classe des sciences mathematiques et na*, no. 6, pp. 749–754, 1931.

[135] F. Vivarelli, "Studies on the Generalisation of Gaussian Processes and Bayesian Neural Networks," Ph.D. dissertation, Aston University, 1998.

[136] A. Lederer, J. Umlauft, and S. Hirche, "Posterior Variance Analysis of Gaussian Processes with Application to Average Learning Curves," 2019. [Online]. Available: http://arxiv.org/abs/1906.01404

[137] ——, "Uniform Error and Posterior Variance Bounds for Gaussian Process Regression with Application to Safe Control," 2021. [Online]. Available: http://arxiv.org/abs/2101.05328

[138] M. Greeff, A. W. Hall, and A. P. Schoellig, "Learning a Stability Filter for Uncertain Differentially Flat Systems using Gaussian Processes," in *Proceedings of the IEEE Conference on Decision and Control*, 2021, pp. 789–794.

[139] A. Capone, A. Lederer, J. Umlauft, and S. Hirche, "Data Selection for Multi-Task Learning under Dynamic Constraints," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 959–964, 2021.

[140] A. Capone, A. Lederer, and S. Hirche, "Confidence Regions for Predictions of Online Learning-Based Control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 983–988, 2020.

[141] A. Capone and S. Hirche, "Anticipating the Long-term Effect of Online Learning in Control," in *Proceedings of the American Control Conference*, 2020, pp. 3865–3872.

[142] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Local Gaussian Process Regression for Real Time Online Model Learning and Control," in *Advances in neural information processing systems*, 2009, pp. 1193–1200.

[143] T. N. Hoang, Q. M. Hoang, K. H. Low, and J. How, "Collective Online Learning of Gaussian Processes in Massive Multi-Agent Systems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7850–7857, 2019.

[144] T. Beckers, L. J. Colombo, S. Hirche, and G. J. Pappas, "Online Learning-Based Trajectory Tracking for Underactuated Vehicles with Uncertain Dynamics," *IEEE Control Systems Letters*, vol. 6, pp. 2090–2095, 2022.

[145] F. Solowjow, D. Baumann, J. Garcke, and S. Trimpe, "Event-Triggered Learning for Resource-Efficient Networked Control," in *Proceedings of the American Control Conference*, 2018, pp. 6506–6512.

[146] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An Introduction to Event-triggered and Self-triggered Control," in *Proceedings of the IEEE Conference on Decision and Control*, 2012, pp. 3270–3285.

[147] J. Jiao, A. Capone, and S. Hirche, "Backstepping Tracking Control Using Gaussian Processes with Event-Triggered Online Learning," *IEEE Control Systems Letters*, vol. 6, pp. 3176–3181, 2022.

[148] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR Tuning based on Gaussian Process Global Optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 270–277.

[149] K. P. Wabersich and M. N. Zeilinger, "Cautious Bayesian MPC: Regret Analysis and Bounds on the Number of Unsafe Learning Episodes," *IEEE Transactions on Automatic Control*, vol. 68, no. 8, pp. 4896–4903, 2023.

[150] F. Castañeda, J. J. Choi, W. Jung, B. Zhang, C. J. Tomlin, and K. Sreenath, "Probabilistic Safe Online Learning with Control Barrier Functions," 2022. [Online]. Available: http://arxiv.org/abs/2208.10733

[151] A. G. Bottero, C. E. Luis, J. Vinogradska, F. Berkenkamp, and J. Peters, "Information-Theoretic Safe Exploration with Gaussian Processes," in *Advances in Neural Information Processing Systems*, 2022, pp. 1–13.

[152] M. Capotondi, G. Turrisi, C. Gaz, V. Modugno, G. Oriolo, and A. De Luca, "An Online Learning Procedure for Feedback Linearization Control without Torque Measurements," in *Proceedings of the Conference on Robot Learning*, 2019.

[153] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian Process Meets Big Data: A Review of Scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[154] M. S. Fadali and A. Visioli, *Digital Control Engineering Analysis and Design Second Edition*. Waltham, MA: Academic Press, 2012.

[155] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*. Princeton, New Jersey: Princeton University Press, 2009.

[156] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[157] M. P. Deisenroth and J. W. Ng, "Distributed Gaussian Processes," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 1481–1490.

[158] V. Tresp, "Mixtures of Gaussian Processes," *Advances in Neural Information Processing Systems*, 2001.

[159] S. Masoudnia and R. Ebrahimpour, "Mixture of Experts: A Literature Survey," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 275–293, 2014.

[160] Z. Liu, L. Zhou, H. Leung, and H. P. Shum, "Kinect Posture Reconstruction based on a Local Mixture of Gaussian Process Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 11, pp. 2437–2450, 2016.

[161] Y. Cao and D. J. Fleet, "Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions," pp. 1–5, 2014. [Online]. Available: http://arxiv.org/abs/1410.7827

[162] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. M. Allen, V. D. Lam, A. Bewley, and A. Shah, "Learning to Drive in a Day," in *Proceedings of the International Conference on Robotics and Automation*, 2019, pp. 8248–8254.

[163] O. Andersson, M. Wzorek, and P. Doherty, "Deep Learning Quadcopter Control via Risk-aware Active Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017, pp. 3812–3818.

[164] D. Nguyen-Tuong and J. Peters, "Incremental Sparsification for Real-time Online Model Learning," *Journal of Machine Learning Research*, vol. 9, pp. 557–564, 2010.

[165] S. Lee, H. Choi, and K. Min, "Reduction of Engine Emissions via a Real-Time Engine Combustion Control with an EGR Rate Estimation Model," *International Journal of Automotive Technology*, vol. 18, no. 4, pp. 571–578, 2017.

[166] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 1094–1099.

[167] J. Schreiter, D. Nguyen-Tuong, and M. Toussaint, "Efficient Sparsification for Gaussian Process Regression," *Neurocomputing*, vol. 192, no. May, pp. 29–37, 2016.

[168] M. F. Huber, "Recursive Gaussian Process: On-line Regression and Learning," *Pattern Recognition Letters*, vol. 45, no. 1, pp. 85–91, 2014.

[169] H. Bijl, T. B. Schön, J.-W. van Wingerden, and M. Verhaegen, "System Identification through Online Sparse Gaussian Process Regression with Input Noise," *IFAC Journal of Systems and Control*, vol. 2, pp. 1–11, 2017.

[170] T. D. Bui, C. V. Nguyen, and R. E. Turner, "Streaming Sparse Gaussian Process Approximations," in *Advances in Neural Information Processing Systems*, 2017, pp. 3300–3308.

[171] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian Processes for Big Data," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2013.

[172] M. Mutný and A. Krause, "Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features," in *Advances in Neural Information Processing Systems*, 2018, pp. 9005–9016.

[173] E. Snelson and Z. Ghahramani, "Local and Global Sparse Gaussian Process Approximations," *Journal of Machine Learning Research*, vol. 2, pp. 524–531, 2007.

[174] A. G. Wilson and H. Nickisch, "Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)," *Proceedings of the International Conference on Machine Learning, ICML 2015*, vol. 3, pp. 1775–1784, 2015.

[175] G. Pleiss, J. R. Gardner, K. Q. Weinberger, and A. G. Wilson, "Constant-time Predictive Distributions for Gaussian Processes," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 6575–6584.

[176] L. Csató and M. Opper, "Sparse On-line Gaussian Processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, 2002.

[177] A. Koppel, "Consistent Online Gaussian Process Regression Without the Sample Complexity Bottleneck," in *Proceedings of the American Control Conference*, 2019, pp. 3512–3518.

[178] H. Bijl, J. W. van Wingerden, T. B. Schön, and M. Verhaegen, "Online Sparse Gaussian Process Regression using FITC and PITC Approximations," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 703–708, 2015.

[179] T. Le, K. Nguyen, V. Nguyen, T. D. Nguyen, and D. Phung, "GoGP: Fast Online Regression with Gaussian Processes," in *Proceedings of the IEEE International Conference on Data Mining*, 2017, pp. 257–266.

[180] A. Ranganathan and M.-h. Yang, "Online Sparse Matrix Gaussian Process Regression and Vision Applications," in *Proceedings of the European Conference on Computer Vision*, 2008, pp. 468–482.

[181] A. Ranganathan, M. H. Yang, and J. Ho, "Online Sparse Gaussian Process Regression and its Applications," *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 391–404, 2011.

[182] J. Harrison, A. Sharma, and M. Pavone, "Meta-Learning Priors for Efficient Online Bayesian Regression," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2018.

[183] R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori, "Incremental Semiparametric Inverse Dynamics Learning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 544–550.

[184] A. Rahimi and B. Recht, "Random Features for Large-scale Kernel Machines," in *Advances in Neural Information Processing Systems*, 2008, pp. 1–8.

[185] Q. Lu, G. Karanikolas, Y. Shen, and G. B. Giannakis, "Ensemble Gaussian Processes with Spectral Features for Online Interactive Learning with Scalability," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2020, pp. 1910–1920.

[186] E. Angelis, P. Wenk, B. Schölkopf, S. Bauer, and A. Krause, "SLEIPNIR: Deterministic and Provably Accurate Feature Expansion for Gaussian Process Regression with Derivatives," 2020. [Online]. Available: http://arxiv.org/abs/2003.02658

[187] Y. Gal and R. Turner, "Improving the Gaussian Process Sparse Spectrum Approximation by Representing Uncertainty in Frequency Inputs," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 655–664.

[188] M. van der Wilk, "Sparse Gaussian Process Approximations and Applications," Ph.D. dissertation, University of Cambridge, 2018.

[189] M. K. Titsias, "Variational Model Selection for Sparse Gaussian Process Regression," University of Manchester, Tech. Rep., 2009.

[190] C. A. Cheng and B. Boots, "Incremental Variational Sparse Gaussian Process Regression," in *Advances in Neural Information Processing Systems*, 2016, pp. 4410–4418.

[191] F. Meier and S. Schaal, "Drifting Gaussian Processes with Varying Neighborhood Sizes for Online Model Learning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 264–269.

[192] S. M. Omohundro, "Five Balltree Construction Algorithms," *Science*, vol. 51, no. 1, pp. 1–22, 1989.

[193] J. W. Ng and M. P. Deisenroth, "Hierarchical Mixture-of-Experts Model for Large-Scale Gaussian Process Regression," 2014. [Online]. Available: http://arxiv.org/abs/1412.3078

[194] V. Tresp, "A Bayesisan Committee Machine," *Neural Computation*, vol. 12, pp. 2719–2741, 2000.

[195] D. Rullière, N. Durrande, F. Bachoc, and C. Chevalier, "Nested Kriging Predictions for Datasets with a Large Number of Observations," *Statistics and Computing*, vol. 28, no. 4, pp. 849–867, 2018.

[196] H. Liu, J. Cai, Y. Wang, and Y. S. Ong, "Generalized Robust Bayesian Committee Machine for Large-scale Gaussian Process Regression," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 3131–3140.

[197] L. Devroye, "Universal Limit Laws for Depths in Random Trees," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 409–432, 1998.

[198] A. Lederer, K. Maier, J. Umlauft, A. J. Ordonez Conejo, W. Xiao, and S. Hirche, "Real-Time Regression with Dividing Local Gaussian Processes," 2020. [Online]. Available: https://arxiv.org/pdf/2006.09446.pdf

[199] N. Terry and Y. Choe, "Splitting Gaussian Process Regression for Computationally-Efficient Regression," *Plos One*, vol. 16, no. 8, p. e0256470, 2021.

[200] A. Douzal-chouakria, E. Gaussier, E. Dimert, A. Douzal-chouakria, E. Gaussier, and E. D. Pr, "Prédictions d'activité dans les réseaux sociaux en ligne," in *4ième conférence sur les modèles et l'analyse des réseaux : Approches mathématiques et informatiques*, 2013, p. 16.

[201] D. Dua and C. Graff, "UCI Machine Learning Repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[202] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep Kernel Learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 51, 2016, pp. 370–378.

[203] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "GPflow: A Gaussian Process Library using TensorFlow," *Journal of Machine Learning Research*, vol. 18, no. 40, pp. 1–6, 2017.

[204] M. Yazdanian and A. Mehrizi-Sani, "Distributed Control Techniques in Microgrids," *IEEE Transactions on Smart Grid*, vol. 5, no. 6, pp. 2901–2909, 2014.

[205] A. Alam, B. Besselink, V. Turri, J. Martensson, and K. H. Johansson, "Heavy-duty Vehicle Platooning for Sustainable Freight Transportation: A Cooperative Method to Enhance Safety and Efficiency," *IEEE Control Systems Magazine*, vol. 35, no. 6, pp. 34–56, 2015.

[206] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized Flocking in Mobile Robot Swarms," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 97–120, 2008.

[207] Z. Yuan and M. Zhu, "Communication-aware Distributed Gaussian Process Regression Algorithms for Real-time Machine Learning," in *Proceedings of the American Control Conference*, 2020, pp. 2197–2202.

[208] P. Budde gen. Dohmann, A. Lederer, M. Dißemond, and S. Hirche, "Distributed Bayesian Online Learning for Cooperative Manipulation," in *Proceedings of the IEEE Conference on Decision and Control*, 2021, pp. 2888–2895.

[209] V.-A. Le and T. X. Nghiem, "Gaussian Process Based Distributed Model Predictive Control for Multi-agent Systems using Sequential Convex Programming and ADMM," in *Proceedings of the IEEE Conference on Control Technology and Applications*, 2020, pp. 31–36.

[210] T. Beckers, S. Hirche, and L. Colombo, "Safe Online Learning-based Formation Control of Multi-Agent Systems with Gaussian Processes," in *Proceedings of the IEEE Conference on Decision and Control*, 2021, pp. 2197–2202.

[211] T. Beckers, G. J. Pappas, and L. J. Colombo, "Learning Rigidity-based Flocking Control using Gaussian Processes with Probabilistic Stability Guarantees," in *Proceedings of the IEEE Conference on Decision and Control*, 2022, pp. 7254–7259.

[212] Z. Yang, S. Sosnowski, Q. Liu, J. Jiao, A. Lederer, and S. Hirche, "Distributed Learning Consensus Control for Unknown Nonlinear Multi-Agent Systems based on Gaussian Processes," in *Proceedings of the IEEE Conference on Decision and Control*, 2021, pp. 4406–4411.

[213] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez, "Tutorial on Dynamic Average Consensus: The Problem, its Applications, and the Algorithms," *IEEE Control Systems Magazine*, vol. 39, no. 3, pp. 40–72, 2019.

[214] H. Zhang and F. L. Lewis, "Adaptive Cooperative Tracking Control of Higher-order Nonlinear Systems with Unknown Dynamics," *Automatica*, vol. 48, no. 7, pp. 1432–1439, 2012.

[215] R. Olfati-Saber and R. M. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[216] S. Thrun, "A Lifelong Learning Perspective for Mobile Robot Control," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 1994, pp. 23–30.

[217] M. Khosravi, C. König, M. Maier, R. S. Smith, J. Lygeros, and A. Rupenyan, "Safety-Aware Cascade Controller Tuning Using Constrained Bayesian Optimization," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 2, pp. 2128–2138, 2023.

[218] S. J. Orfanidis, *Introduction to Signal Processing*. Pearson Education, 2016.

[219] M. Mboup, C. Join, and M. Fliess, "A Revised Look at Numerical Differentiation with an Application to Nonlinear Feedback Control," in *Proceedings of the Mediterranean Conference on Control and Automation*, 2007.

[220] A. Radke and Z. Gao, "A Survey of State and Disturbance Observers for Practitioners," in *Proceedings of the American Control Conference*, 2006, pp. 5183–5188.

[221] A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian Process Priors with Uncertain Inputs - Application to Multiple-step Ahead Time Series Forecasting," *Advances in Neural Information Processing Systems*, pp. 545–552, 2003.

[222] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search," in *Proceedings of the International Conference on Machine Learning*, 2011, pp. 465–472.

[223] A. McHutchon and C. E. Rasmussen, "Gaussian Process Training with Input Noise," in *Advances in Neural Information Processing Systems*, 2011, pp. 1–9.

[224] I. Steinwart and A. Christmann, *Support Vector Machines*. New York, NY: Springer Science+Business Media, 2008.

[225] R. M. Murray, Z. Li, and S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[226] M. Lutter, C. Ritter, and J. Peters, "Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning," *Proceedings of the International Conference on Learning Representations*, pp. 1–17, 2019.

[227] B. Wilcox and M. C. Yip, "SOLAR-GP: Sparse Online Locally Adaptive Regression Using Gaussian Processes for Bayesian Robot Model Learning and Control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2832–2839, 2020.

[228] M. Riedmiller and H. Braun, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," in *Proceedings of the IEEE International Conference on Neural Networks*, 1993, pp. 586–591.

[229] M. Blum and M. Riedmiller, "Optimization of Gaussian Process Hyperparameters using RPROP," in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013, pp. 339–344.

[230] G. Hu, W. P. Tay, and Y. Wen, "Cloud Robotics: Architecture, Challenges and Applications," *IEEE Network*, vol. 26, no. 3, pp. 21–28, 2012.

[231] Y. Xia, "Cloud Control Systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 2, pp. 134–142, 2015.

[232] L. Qian, Z. Luo, Y. Du, L. Guo, X. Ave, and X. District, "Cloud Computing: An Overview," in *Proceedings of the Conference on Cloud Computing*, 2009, pp. 626–631.

[233] B. Hayes, "Cloud Computing," *Communications of the ACM*, vol. 51, no. 7, pp. 9–11, 2008.

[234] M. Abramowitz and I. A. Segun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York: Dover, 1965.

[235] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A Fresh Approach to Numerical Computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[236] E. Trigili, S. Crea, M. Moise, A. Baldoni, M. Cempini, G. Ercolini, D. Marconi, F. Posteraro, M. Carrozza, and N. Vitiello, "Design and Experimental Characterization of a Shoulder-Elbow Exoskeleton with Compliant Joints for Post-Stroke Rehabilitation," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 4, pp. 1485–1496, 2020.

[237] R. Drillis, R. Contini, and M. Bluestein, "Body Segment Parameters: A Survey of Measurement Techniques," *Artificial Limbs*, vol. 8, pp. 44–66, 1964.

[238] T. Beckers, L. J. Colombo, M. Morari, and G. J. Pappas, "Learning-based Balancing of Model-based and Feedback Control for Second-order Mechanical Systems," in *Proceedings of the IEEE Conference on Decision and Control*, 2022, pp. 4667–4673.

[239] X. Dai, A. Lederer, Z. Yang, and S. Hirche, "Can Learning Deteriorate Control? Analyzing Computational Delays in Gaussian Process-Based Event-Triggered Online Learning," in *Proceedings of the Conference on Learning for Dynamics and Control*, 2023, pp. 445–457.

[240] M. Buisson-Fenet, V. Morgenthaler, S. Trimpe, and F. Di Meglio, "Joint State and Dynamics Estimation with High-gain Observers and Gaussian Process Models," *IEEE Control Systems Letters*, vol. 5, no. 5, pp. 1627–1632, 2020.

[241] A. J. Ordóñez-Conejo, A. Lederer, and S. Hirche, "Adaptive Low-Pass Filtering using Sliding Window Gaussian Processes," in *Proceedings of the European Control Conference*, 2022, pp. 2234–2240.

[242] S. Curi, A. Lederer, S. Hirche, and A. Krause, "Safe Reinforcement Learning via Confidence-Based Filters," in *Proceedings of the IEEE Conference on Decision and Control*, 2022, pp. 3409–3415.

[243] D. S. Bernstein, *Scalar, Vector, and Matrix Mathematics: Theory, Facts, and Formulas*, revised and expanded ed. Princeton, New Jersey: Princeton University Press, 2018.