Gaussian Processes in Reinforcement Learning: Stability Analysis and Efficient Value Propagation

Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation von Julia Vinogradska aus Odessa, Ukraine
Tag der Einreichung: 19. September 2017, Tag der Prüfung: 29. November 2017
Darmstadt — D 17

1. Gutachten: Prof. Jan Peters
2. Gutachten: Prof. Carl Rasmussen

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Intelligente Autonome Systeme

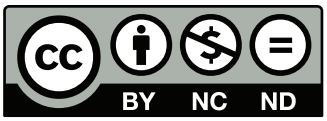
Gaussian Processes in Reinforcement Learning: Stability Analysis and Efficient Value Propagation

Genehmigte Dissertation von Julia Vinogradska aus Odessa, Ukraine

1. Gutachten: Prof. Jan Peters
2. Gutachten: Prof. Carl Rasmussen

Tag der Einreichung: 19. September 2017
Tag der Prüfung: 29. November 2017

Darmstadt — D 17



Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 19. September 2017

(J. Vinogradska)

Abstract

Control of nonlinear systems on continuous domains is a challenging task for various reasons. For robust and accurate control of complex systems a precise model of the system dynamics is essential. Building such highly precise dynamics models from physical knowledge often requires substantial manual effort and poses a great challenge in industrial applications. Acquiring a model automatically from system measurements employing regression techniques allows to decrease manual effort and, thus, poses an interesting alternative to knowledge-based modeling. Based on such a learned dynamics model, an approximately optimal controller can be inferred automatically. Such approaches are the subject of model-based reinforcement learning (RL) and learn optimal control from interactions with the system. Especially when probabilistic dynamics models such as Gaussian processes are employed, model-based RL has been tremendously successful and has attracted much attention from both the control and machine learning communities. However, several problems need to be solved to facilitate widespread deployment of model-based RL for learning control in real world scenarios. In this thesis, we address two current limitations of model-based RL that are indispensable prerequisites for widespread deployment of model-based RL in real world tasks.

In many real world applications a poor controller can cause severe damage to the system or even put the safety of humans at risk. Thus, it is essential to ensure that the controlled system behaves as desired. While this question has been studied extensively in classical control, stability of closed-loop control systems with dynamics given as a Gaussian process has not been considered yet. We propose an automatic tool to compute regions of the state space where the desired behavior of the system can be guaranteed. We consider dynamics given as the mean of a GP as well as the full GP posterior distribution. In the first case, the proposed tool constructs regions of the state space, such that the trajectories starting in this region converge to the target state. From this asymptotic result, we follow statements for finite time horizons and stability under the presence of disturbances. In the second case the system dynamics is given as a GP posterior distribution. Thus, computation of multi-step-ahead predictions requires averaging over all plausible dynamics models given the observations. A a consequence, multi-step-ahead predictions become analytically intractable. We propose an approximation based on numerical quadrature that can handle complex state distributions, e.g., with multiple modes and provides upper bounds for the approximation error. Exploiting these error bounds, we present an automatic tool to compute stability regions. In these regions of the state space, our tool guarantees that for a finite time horizon the system behaves as desired with a given probability. Furthermore, we analyze asymptotic behavior of closed-loop control systems with dynamics given as a GP posterior distribution. In this case we show that for some common choices of the prior, the system has a unique stationary distribution to which the system state converges irrespective of the starting state.

Another major challenge of RL for real world control applications is to minimize interactions with the system required for learning. While RL approaches based on GP dynamics models have demonstrated great data efficiency, the average amount of required system interactions can

further be reduced. To achieve this goal, we propose to employ the numerical quadrature based approximation to propagate the value of a state. To show how this approximation can further increase data efficiency, we employ it in the two main classes of model-based RL: policy search and value iteration. In policy search, the state distribution must be computed to evaluate the expected long-term reward for a policy. The proposed numerical quadrature based approximation substantially improves estimates of the expected long-term reward and its gradients. As a result, data efficiency is significantly increased.

For the value function based approaches for policy learning, the value propagation step is completely characterized by the Bellman equation. However, this equation is intractable for nonlinear dynamics. In this case, we propose a projection-based value iteration approach. We employ numerical quadrature to facilitate projection of the value function onto a linear feature space. Suitable features for value function representation are learned online without manual effort. This feature learning is constructed such that upper bounds for the projection error can be obtained. The proposed value iteration approach learns globally optimal policies and significantly benefits from the introduced highly accurate approximations.

Zusammenfassung

Die Regelung nichtlinearer Systeme ist aus vielerlei Gründen eine technische Herausforderung. Um eine robuste und präzise Regelung zu erreichen, ist ein Modell der Systemdynamik essentiell. Die Herleitung eines akkuraten Dynamikmodells aus physikalischem Wissen verursacht häufig sehr hohen Arbeitsaufwand und ist eine Herausforderung in industriellen Anwendungen. Die automatische Berechnung eines Modells aus Messungen mit Hilfe von Regression erlaubt es, den Arbeitsaufwand deutlich zu verringern und ist daher eine sinnvolle Alternative zu klassischer, wissensbasierter Modellierung. Basierend auf solch einem gelernten Modell kann ein näherungsweise optimaler Regler automatisch bestimmt werden. Solche Ansätze sind Gegenstand des modellbasierten Reinforcement Learnings (RL) und lernen optimale Regelung eines Systems aus Interaktionen mit ebendiesem. Gerade in Kombination mit probabilistischen Dynamikmodellen wie z.B. Gaußschen Prozessen, hat sich modellbasiertes RL als außerordentlich erfolgreich erwiesen und viel Aufmerksamkeit der Experten sowohl in Regelungstechnik als auch in Machine Learning erregt. Allerdings müssen noch einige Probleme gelöst werden, um modellbasiertes RL massentauglich für reale praktische Probleme zu machen. In der vorliegenden Arbeit adressieren wir zwei der aktuellen Einschränkungen von RL, deren Lösung eine Vorraussetzung für die breite Anwendung solcher Methoden in der Praxis ist.

In vielen Anwendungen kann ein schlecht eingestellter Regler das System beschädigen oder sogar Menschen in Gefahr bringen. Daher ist es essentiell, das wunschgemäße Systemverhalten abzusichern. Während diese Fragesellung in der klassichen Regelungstechnik schon ausgiebig studiert wurde, ist die Stabilität von geschlossenen Regelkreisen mit Gaußschen Prozessen als Dynamikmodell bislang noch nicht betrachtet worden. Wir stellen einen Ansatz vor, um automatisiert Bereiche des Zustandsraums zu berechnen, in denen die Konvergenz zum Zielzustand garantiert ist. Dabei betrachten wir Dynamiken, die als Mittelwert sowie als volle a-posteriori Verteilung eines Gaußschen Prozesses gegeben sind. Im ersten Fall berechnet unser Ansatz Bereiche des Zustandsraums, von denen aus die Systemtrajektorien zum Zielzustand konvergieren. Aus diesem asymptotischen Ergebnis leiten wir Aussagen her für endliche Zeithorizonte sowie über die Stabilität des Systems wenn Störungen auftreten. Im zweiten Fall nehmen wir als Dynamikmodell die a-posteriori Verteilung eines Gaußschen Prozesses an. Daher muss für die Vorhersage des Systemzustands nach mehreren Zeitschritten über alle plausiblen Dynamikmodelle gegeben unsere Beobachtungen gemittelt werden. Folglich sind solche Mehrschrittvorhersagen nicht analytisch lösbar. Wir schlagen daher eine Approximation basierend auf numerischer Quadratur vor, die komplexe Zustandsverteilungen, z.B. mit mehrereren Modi, abbilden kann. Zusätzlich kann bei dieser Approximation der Fehler nach oben hin beschränkt werden. Mit Hilfe dieser Fehlerschranken berechnet unser Ansatz automatisiert stabile Bereiche des Systems. In diesen Bereichen ist das wunschgemäße Verhalten des Systems mit einer gegebenen Wahrscheinlichkeit garantiert. Desweiteren analysieren wir das asymptotische Verhalten von geschlossenen Regelkreisen mit einer GP a-posteriori Vetreilung über Dynamiken. Wir zeigen, dass eine eindeutige, stationäre Verteilung existiert, zu der der Systemzustand unabhängig von der Wahl des Startpunkts konvergiert.

Eine weitere große Herausforderung für RL in der Praxis ist, dass die Anzahl der benötigten Systeminteraktionen möglichst gering sein soll. Obwohl sich RL mit Gaußschen Prozessen als Dynamikmodell als sehr dateneffizient erwiesen hat, kann die durschnittliche Anzahl der benötigten Systeminteraktionen noch weiter reduziert werden. Um dieses Ziel zu erreichen, schlagen wir den Einsatz der oben erwähnten, auf numerischer Quadratur basierenden Approximation vor, um den Wert eines Zustands zu propagieren. Wir zeigen, wie diese Approximation die Dateneffizienz erhöhen kann, indem wir es in den zwei Hauptklassen von modellbasiertem RL anwenden: Policy Search und Value Iteration. In Policy Search muss die Zustandsverteilung berechnet werden, um die erwartete Langzeitbelohnung eines Reglers zu bestimmen. Die vorgeschlagene Approximation auf Basis numerischer Quadratur verbessert signifikant die Schätzung der erwarteten Langzeitbelohnung und ihres Gradienten. Dies führt zu einer deutlich gesteigerten Dateneffizienz.

Bei den Ansätzen, die zum Lernen des optimalen Reglers eine Value Funktion verwenden, ist die Value Weiterverbreitung durch die Bellman Gleichung charakterisiert. Allerdings ist diese Gleichung für nichtlineare Dynamiken analytisch nicht lösbar. Für dieses Szenario schlagen wir einen projektionsbasierten Ansatz zur Value Iteration vor. Wir verwenden numerische Quadratur, um die Projektion der Value Funktion auf einen linearen Merkmalsraum zu ermöglichen. Für die Darstellung der Value Funktion geeinete Merkmale werden dabei online und ohne manuellen Aufwand gelernt. Dieses Lernen der Merkmale ist so konstruiert, dass obere Schranken für den Projektionsfehler hergeleitet werden können. Der vorgeschlagene Ansatz zur Value Iteration lernt global optimale Regler und profitiert deutlich von der vorgestellten hochpräzisen Approximation.

Acknowledgements

First and foremost, I would like to thank my doctoral advisor, Prof. Jan Peters for his excellent supervision. I have benefited from his overview and experience in the fields of reinforcement learning and control. Thank you, Jan, for guiding me through the stormy waters of research, all the inspiring discussions we had and for your patience.

I would like to thank Prof. Carl Rasmussen for reviewing this thesis and agreeing to be external committee member. Your input is greatly appreciated. I would also like to thank the other members of the thesis committee, Professors Roth, von Stryk, Kersting and Binnig, without whose time investment a thesis defense would not have been possible.

I am deeply grateful to my supervisor, Dr. Bastian Bischoff, for guiding me through this PhD journey, for his enduring patience and confidence in me. The technical discussions with him substantially improved the quality of my work and helped me to stay focused. Bastian, no words seem to be enough to describe how grateful I am. Thank you for asking the hard questions, for always being there, thank you for all.

I wish to thank my colleagues at the Bosch Center for Artificial Intelligence. The positive atmosphere and mutual support in this group is outstanding. Dear colleagues, thank you for making me feel like at home from the first day. I am thankful to Dr. Yasser Jadidi, Dr. Heiner Markert and Dr. Mathias Bürger for making my thesis at Bosch possible and giving me the best conditions I could have hoped for. I would like to thank Dr. Duy Nguyen-Tuong for his support and his advice, Dr. Martin Schiegg for proofreading parts of this thesis and Dr. Christian Daniel for his helpful advice.

While working on this thesis, I had the fortune to supervise the undergraduate work of Jan Achterhold, Torsten Koller, Anne Romer, Henner Schmidt. I thank Anne and Henner for their support on the stability experiments and Torsten and Jan for their contributions to the experiments on policy search.

Finally, I sincerely thank my friends and family for the unwavering support and encouragement during these years. I am deeply grateful to my parents for teaching me to do things “the right way” only and their unconditional love and support.

Contents

List of Symbols

The following tables give an overview of the symbols used in this thesis. Symbols that only pertain to a specific section of the thesis are defined where they are used.

Formatting

x	scalar
$\boldsymbol{x}$	vector
x_j	j-th entry of $\boldsymbol{x}$
$\boldsymbol{x}^\intercal$	transpose of vector x
X	matrix
$X^\intercal$	transpose of matrix X

Symbols

$\boldsymbol{x}$	state
$\boldsymbol{u}$	control signal/action
$\pi(\cdot)$	policy
$\boldsymbol{f}(\cdot,\cdot)$	forward dynamics $\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$
$r(\cdot)$	reward function $r : \boldsymbol{x} \mapsto r(\boldsymbol{x}) \in \mathbb{R}$
$\hat{\boldsymbol{x}}$	extended state including control signal, $\hat{\boldsymbol{x}} = (\boldsymbol{x}, \pi(\boldsymbol{x}))^\intercal$
t	time step
$\boldsymbol{x}_t$	system state at time t
$\boldsymbol{x}_d$	desired state
$F[\cdot]$	propagate one step forward, maps state distribution $p(\boldsymbol{x}_t)$ to next state distribution $p(\boldsymbol{x}_{t+1})$
ξ_i	i-th quadrature node
w_i	i-th quadrature weight
$\boldsymbol{\alpha}$	weight vector
ϕ	basis function/feature
Φ	quadrature weight propagation matrix
$\mathscr{D}$	dataset
$\mathbb{R}$	real numbers
$\mathscr{B}(\mathbb{R})$	Borel sets of $\mathbb{R}$
$\mathcal{B}_r(\boldsymbol{x})$	full metric ball around $\boldsymbol{x}$ with radius r
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance Σ
$k(\cdot,\cdot)$	kernel function
$k(\boldsymbol{x}_*, X)$	vector of kernel evaluations between $\boldsymbol{x}_*$ and the rows of X
$k(X, X)$	covariance matrix between the rows of X
I	identity matrix
$p(\cdot)$	probability distribution

$\mathrm{diag}(\boldsymbol{w})$	square matrix with the entries of $\boldsymbol{w}$ on the diagonal
$\mathbb{E}_{\boldsymbol{x}}[\cdot]$	expectation with respect to $\boldsymbol{x}$
$\frac{\partial \cdot}{\partial x}$	partial derivative with respect to x
$\mathcal{C}(X)$	continuous functions on the domain X
$\langle \cdot, \cdot \rangle$	inner product
$\cdot \leftarrow \cdot$	is assigned the value of (in pseudocode algorithms)
$\sim$	is distributed according to
$\approx$	is approximately equal to
$\in$	is member of
$\cup, \bigcup$	set union
$\sqcup$	disjoint set union
$\overset{!}{<}$	shall be less than

1 Introduction

In 1968, Donald Knuth characterized computers as: "These machines have no common sense; they have not yet learned to *think*, and they do exactly as they are told, no more and no less."[1]. This statement on the nature of computation has been valid for many decades – in fact, when taken literately it still holds today. However, the way we tell computers what to do has changed. In the early days of digital computation, programming has been procedure oriented and tasks were solved by carefully crafting precise, task specific instructions for the machine to execute. A different approach is taken in machine learning, where knowledge is not encoded in the program itself but instead derived automatically from data. This allows the programmer to focus on specifying the goal instead of the procedure and letting the computer then infer the solution from data.

Machine learning is not a new field of computer science, but has already seen several hype cycles. Recently, the increase in computational resources and available data combined with algorithmical advancements have led to its renaissance and widespread deployment. While most progress has been made in supervised learning and many supervised learning techniques are the new gold standard in multiple applications, the field of reinforcement learning (RL) has also advanced significantly. It has been tremendously successful in control tasks (Deisenroth et al., 2015) or solving games, e.g., Atari (Mnih et al., 2015) or Go (Silver et al., 2016), achieving superhuman performance. The MIT technology review included reinforcement learning in their list of top 10 technologies of 2017. The potential of RL is enormous: for the first time since the advent of computer science it seems possible to make machines *think*, i.e., to learn desirable behavior from interaction with the environment. Reinforcement learning could revolutionize automation, helping us to solve problems more efficiently on a large scale. It could also allow us to tackle complex problems that were hitherto out of reach, because their solution is not easily broken down to a list of executable instructions.

1.1 Motivation

While reinforcement learning techniques have matured significantly and have incredible potential, widespread deployment of RL in real world tasks has yet to come. Several issues have to be solved to make RL techniques a viable approach for real world applications as, e.g., control on continuous domains. In this thesis, we aim to address some of the current limitations of reinforcement learning for control.

Control of nonlinear systems on continuous domains is a challenging task for various reasons. In general, a (model-based) control task can be subdivided in three subproblems: (i) find a suitable dynamics model, (ii) design a controller based on this model and (iii) analyze the controller performance to derive guarantees. Each of these three problems is challenging on its own and general standard approaches do not exist. In the following, we will discuss these three sub-tasks and how RL can help to solve them.

[1] Donald Knuth in the preface to *The Art of Computer Programming*, 1968.

Let's begin with finding suitable dynamics models. Classical control methods assume a detailed understanding of the system dynamics and, thus, rely heavily on expert knowledge. Building a highly precise dynamics model from physical knowledge often requires substantial manual effort and poses a great challenge in industrial applications, as this approach is system specific and presupposes permanent availability of sufficiently many experts. Acquiring a model automatically from system measurements employing regression techniques allows to decrease manual effort and, thus, poses an interesting alternative to knowledge-based modeling. However, classical as well as learned *deterministic* models suffer from one major drawback. The obtained dynamics model is by construction only an estimate of the underlying dynamics and inherently approximate. A deterministic model, however, cannot reflect uncertainty of the model prediction. Once the dynamics model is obtained, this information is lost and typically neglected in further considerations, e.g., when designing a controller. This loss of information can cause severe problems in controller design, e.g., controllers exploiting the model behavior in highly uncertain regions, where model predictions are almost arbitrary, and, thus, failing on the real system. Gaussian processes (GPs) (Rasmussen and Williams, 2005) offer a flexible, fully Bayesian, kernel-based regression framework that is a preferred choice for learning dynamics models from data. Instead of compromising on a single deterministic model, GPs infer a distribution over plausible models given the observed data. Overall, GPs are a powerful tool to accurately model system dynamics from observed data, drastically decreasing the required manual effort.

Synthesizing (optimal) controllers for nonlinear dynamics is a challenging problem and in the case of initially unknown dynamics the core task in reinforcement learning. Learning a GP dynamics model and based on this model a suitable controller has been successfully demonstrated e.g., in (Deisenroth et al., 2015; Bischoff et al., 2013; Kupcsik et al., 2013). Employing GP dynamics models, these methods benefit from Bayesian averaging over all plausible models. However, when the full GP posterior distribution is considered, multi-step-ahead predictions become intractable. To fully enjoy the advantages of the Bayesian model, it is crucial to accurately approximate these predictions. As propagation of the value is at the center of all RL approaches, this approximation is the key to successfully learn high precision controllers while minimizing the number of required system interactions. At the same time, value propagation is performed numerous times during learning, requiring the chosen approximation to be highly efficient to ensure computational feasibility.

Finally, having learned a GP forward dynamics and a controller, the performance of the controller must be analyzed to ensure proper functionality. This step may seem less impactful than actually learning control, but it is indispensable for most applications, especially when poor quality control is a safety risk or could cause damage to the system. In these cases, it is crucial to know a region of the state space (a *region of attraction (ROA)*), where the controller is guaranteed to work as desired. This kind of guarantees has been extensively studied in classic control theory since the 19th century. Stability can easily be analyzed for linear dynamics, but for general, nonlinear dynamics stability analysis is a challenging task that involves massive manual effort and where no standard approach exists. For learned models and controllers, stability has not been studied yet. The lack of stability guarantees might be the single largest deficiency of RL for control that has prevented its widespread deployment on real world tasks.

In this thesis, we will focus on the two major challenges discussed above: finding an efficient method for approximate value propagation through GP dynamics and analyzing the stability of closed-loop control with GP forward dynamics models.

1.2 Contributions

The goal of this thesis is to build a solid foundation that is necessary for widespread deployment of reinforcement learning for control tasks in real world scenarios. We will focus on two key challenges that typically arise in practice, (i) deriving stability guarantees for learned controllers and GP dynamics models and (ii) further improve data efficiency of learning control. Thus, the contribution of this thesis is fourfold:

- We develop a highly efficient and accurate approximation for multi-step-ahead predictions with GP forward dynamics. This approximation can handle complex state distributions with multiple modes and upper bounds for the approximation error can be obtained.
- We develop an automatic tool to compute stability guarantees for given GP forward dynamics model and given controller. Furthermore, we analyze asymptotic behavior of closed-loop control systems with a GP posterior distribution over dynamics models.
- We employ the developed approximate inference technique to the two main classes of model-based RL:
  - policy search
  - and value iteration.

  In both cases this approximation enables accurate and efficient value propagation and helps to further increase data efficiency.

In the following sections, we discuss these contributions in more detail. This discussion is organized in two parts: the first deals with stability analysis and the second concentrates on the contributions to policy learning. Different aspects of the proposed approximation of multi-step-ahead predictions are important for stability analysis and policy learning. Thus, we don't discuss this contribution on its own, but cover it in the context of stability analysis and learning control.

1.2.1 Stability Guarantees for Gaussian Process Forward Dynamics

Learning control based on a Gaussian process forward model has become a viable alternative to the classical approaches, where a model and controller is derived analytically based on expert knowledge about the system and first principles. However, in most real-world applications proper functioning of the controller is crucial to ensure safety or prevent damage of the system. For example, one might be interested in finding a state space region where it can be guaranteed that the controller will work as desired. This kind of guarantees has been studied extensively in classical control theory. For nonlinear system dynamics, stability guarantees can be obtained by finding a Lyapunov function. However, finding such functions is typically very challenging and no general approach exists.

For system dynamics modeled as a Gaussian process, stability analysis has not been considered yet. We propose an automatic tool, that computes a stability region of the closed-loop-control system given a Gaussian process dynamics model. This automatic stability prover can handle dynamics given as the mean of a GP or the full GP posterior.

If the system dynamics is given as the mean of a GP, finding the controller's region of attraction is a classical problem from nonlinear control theory. However, as a nonparametric model a GP introduces a large amount of nonlinear terms that render the classical methods impractical. Furthermore, classical control methods typically exploit problem specific properties, e.g. a certain parametric form of the dynamics, to find a suitable Lyapunov function. For such a general regressor as GPs this approach is, thus, infeasible. Instead, we propose an algorithmic approach that constructs a region of attraction based on the study of invariant sets. From this asymptotic stability statement, we follow some results for finite time horizons and stability of the system under the presence of disturbances.

If the full GP posterior distribution is considered, there are several challenges that need to be overcome to enable stability guarantees. Firstly, when averaging over all plausible dynamics models given the data, multi-step-ahead predictions become intractable and must be approximated. The state distribution can be very complex, e.g. with multiple modes, and, thus, requires a powerful and expressive approximation method. Additionally, this approximation has to allow for error analysis, as we want to give reliable theoretical guarantees. We propose an approximation based on numerical quadrature, that matches these requirements and approximates the state distribution at any time step as a mixture of Gaussian basis functions. This approximation enables finite time statements about the success probability of the controller, that can be computed automatically.

Finally, we analyze behavior of the closed-loop control system with full GP posterior distribution as dynamics model. Relying on Markov chain theory and exploiting some basic properties of Gaussian processes, we show that such systems converge to a unique, stationary distribution irrespective of the starting state.

1.2.2 Efficient Value Propagation for Gaussian Process Forward Dynamics

Model-based reinforcement learning approaches that employ a Gaussian process as dynamics model are particularly appealing, as they offer a large variety of advantages over model-free approaches as, e.g., high accuracy and data efficiency. However, the use of GPs as dynamics models renders multi-step-ahead predictions intractable and complicates value propagation. We propose the use of a numerical quadrature based approximation, as it provides an accurate estimate that can capture complex distributions. We study how this high precision approximation can be employed in two main classes of model-based RL algorithms: policy search and value iteration.

In policy search, a parametric form for the controller is chosen and the goal of learning is to infer optimal parameters for the controller from interactions with the system. More precisely, the objective is to maximize the expected long-term reward given an immediate reward function and an initial state distribution. Employing a GP dynamics model and choosing a smooth parametric form for the policy, the optimization objective depends differentiably on the policy parameters due to smoothness of the GP. Thus, the optimal parameters can be found using gradient based optimization. However, as the multi-step-ahead predictions become intractable when employing a GP dynamics model, the expected long-term cost and its gradients must be approximated as well.

Employing highly precise approximations helps to significantly improve the gradient estimates and, thus, making learning more robust with respect to the (possibly poor) initial parameters. The proposed nuQuPS framework benefits from this advanced approximation and further increases data efficiency, successfully learning near-optimal policies reliably in a few episodes. Furthermore, the accurate approximation allows nuQuPS to naturally handle starting state distributions that assign a positive probability to state space regions that expose qualitatively different behavior, e.g. a high variance Gaussian or a uniform distribution over a region.

Value function based approaches do not presuppose any parametric form of the policy. In contrast to policy search where parameters are chosen to optimize the system's behavior for those trajectories likely under the given starting state distribution and dynamics model, value function based approaches are global, aiming to infer optimal actions for all states. To compute the value function when the dynamics is modeled as a Gaussian process, one has to average the value of any state over all plausible dynamics given the observed data. However, the value function is in general analytically intractable even for one particular dynamics function. Thus, the value function must be approximated. We propose a framework that learns a GP dynamics model and a value function, which is used to infer an optimal policy in an episodic setting. To represent the value function our approach employs a projection-based linear approximation and learns suitable features online. For value propagation, the value function is projected onto the linear feature space. To enable this projection, we propose a numerical quadrature based approximation of the local value propagation. Furthermore, upper bounds for the approximation error caused by numerical quadrature and the projection error can be derived.

1.3 Outline of the Thesis

This thesis is organized in five chapters. Each chapter is self-contained and can be read independently of each other. However, the chapters 2, 3 and 4 have some overlap as they all consider the same closed-loop control setting with a Gaussian process dynamics model. To make each chapter self-contained, a brief recap of Gaussian process regression is included at the beginning of these chapters and, thus, appears multiply in this thesis. Nonetheless, we recommend reading the chapters in order and skipping the brief summary of GP regression in Chapters 3 and 4.

Figure 1.1 illustrates the outline and relation of the thesis chapters. Chapter 2 addresses the stability of closed-loop control systems with Gaussian process forward dynamics models. We provide an automatic tool to find stable regions of the system for the case of dynamics given as the mean of a GP and the full GP posterior. For the deterministic case of GP mean dynamics, the analysis yields a statement on asymptotic stability, which we use to follow some results for finite time horizons and the system behavior under the presence of disturbances. To analyze stability considering the full GP posterior for the system dynamics, we propose an approximation for multi-step-ahead predictions based on numerical quadrature. Employing this approximation, we provide a tool to compute stability regions of the closed-loop control system for finite time horizons. Finally, we analyze asymptotic behavior of closed-loop control system with GP dynamics and show that such systems have a unique, invariant distribution to which the state converges irrespective of the initial conditions.

The approximation for multi-step-ahead predictions proposed in Chapter 2 is a powerful technique that can handle complex distributions. Motivated by this result, we study how this approx-

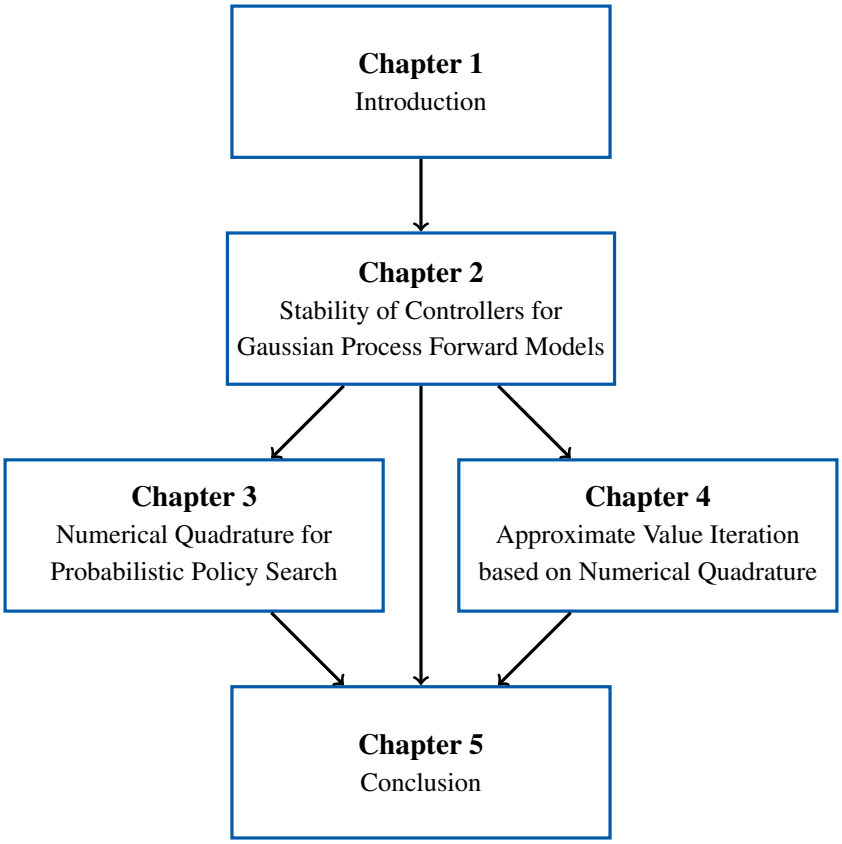


Figure 1.1.: This diagram shows the structure of the thesis. Chapter 1 motivates the content of this thesis. Chapter 2 develops an automatic stability analysis tool for closed-loop control structures with GP forward dynamics models. To enable stability statements for dynamics given as a GP posterior distribution, a highly accurate approximation method for multi-step-ahead predictions is proposed. In Chapters 3 and 4, we show that this method is highly efficient for value propagation and improves reliability and data efficiency of the major classes of model-based RL approaches. Chapter 5 summarizes our findings and gives and discusses open questions and future research directions.

imation method can be employed for policy learning. In Chapter 3, we propose a policy search approach that relies on this numerical quadrature approximation to estimate the expected long-term reward. We show that this increased accuracy improves data efficiency, decreasing volatility of the learning success due to parameter initialization. Furthermore, we propose a novel policy search approach that is inspired by the result regarding convergence of the system to a stationary distribution. This approach searches policy parameters to directly optimize the system's stationary distribution. In Chapter 4, we consider value iteration based on GP dynamics models. Here, we propose an approach that employs a numerical quadrature based approximation for the projection of the value function onto a linear feature space and learns suitable features online.

Chapter 5 summarizes our findings and discusses possible directions for future work.

2 Stability of Controllers for Gaussian Process Forward Models

In industrial control applications, a key issue is the validation of the obtained controller. This validation must convince certification organizations that the controller is safe to apply. In most real-world scenarios testing controller performance is not sufficient, as the domains are typically continuous and statistical statements require an infeasible amount of test runs. Thus, mathematical proofs for the stability of the controlled system must be derived to obtain reliable guarantees. In classical control, such guarantees have been studied extensively. However, these classical methods do not apply straightforwardly to nonparametric models, are task-specific and, thus, involve manual effort. In this chapter, we propose an approach to validate learned controllers. We analyze stability of closed-loop control systems with forward dynamics given as a Gaussian process. Here, a major challenge is that for learned models and controllers, no properties of the specific system can be exploited to obtain the guarantees. Instead, we focus on an automatic tool to compute the stability statements that does not involve manual effort.

2.1 Introduction

Learning control based on Gaussian process (GP) forward models has become a viable approach in the machine learning and control theory communities. Many successful applications impressively demonstrate the efficiency of this approach (Deisenroth et al., 2015; Pan and Theodorou, 2014; Klenske et al., 2013; Maciejowski and Yang, 2013; Nguyen-Tuong and Peters, 2011; Engel et al., 2006; Kocijan et al., 2004). In contrast to classic control theory methods, learning control does not presuppose a detailed understanding of the underlying dynamics but tries to infer the required information from data. Thus, relatively little expert knowledge about the system dynamics is required and fewer assumptions, such as a parametric form and parameter estimates, must be made. Employing Gaussian processes as forward models for learning control is particularly appealing as they incorporate uncertainty about the system dynamics. GPs infer a distribution over all plausible models given the observed data instead of one (possibly erroneous) model and, thus, avoid severe modeling errors. Furthermore, obtaining a task solution by a learning process can significantly decrease the involved manual effort.

Unfortunately, performance guarantees rarely exist for arbitrary system dynamics and policies learned from data. An important and well-established type of performance guarantee is *(asymptotic) stability*. A *stability region* in the state space ensures, that all trajectories starting in this region converge to the target. Classic control theory offers a rich variety of stability analysis, e.g., for linear, nonlinear, and stochastic systems (Khalil, 2014; Khasminskii and Milstein, 2011; Skogestad and Postlethwaite, 2005; Kushner, 1967). In a preliminary study (Vinogradska et al., 2016), we introduced a tool to analyze the stability of learned policies for closed-loop control systems with transition dynamics given as a GP. This tool handled two types of dynamics: dynamics given as (i)

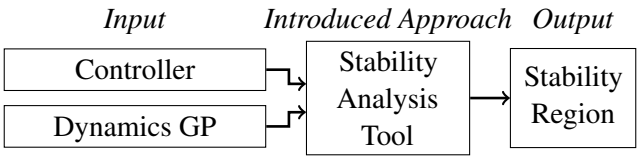


Figure 2.1.: We present a tool to analyze stability for a given controller and Gaussian process (GP) dynamics model. Our tool analytically derives a stability region where convergence to the goal is guaranteed. Thus, we can provide stability guarantees for many of the existing GP based learning control approaches.

the mean of a GP, and (ii) the full GP distribution. While the first case results in a deterministic closed-loop system, uncertainty is present in the second case. As propagating distributions through a GP is analytically intractable, we proposed a novel approach for approximate multi-step-ahead predictions based on numerical quadrature. Finally, we were able to analyze asymptotic stability of the deterministic dynamics in case (i), but in case (ii) only finite time horizons were considered. Furthermore, the results were limited to GPs with squared exponential kernel.

In this chapter, we provide a solid foundation for stability analysis of learned dynamics and present a stability analysis tool. For our detailed study on stability of closed-loop control systems, we consider dynamics given as (i) the mean of a GP, and (ii) the full GP distribution. In both cases, we analyze finite as well as infinite time horizons. For dynamics given as the mean of a GP, we obtain a region of starting points in the state space such that the target state is reached up to a given tolerance at the (finite) time horizon. For infinite time horizons, we construct a stability region in the state space. Furthermore, we study the behavior of dynamics given by the mean of a GP when disturbances are present. Here, we derive criteria for the disturbance such that the closed-loop control system remains stable. For full GP dynamics, we propose an algorithm to find a region of starting points in the state space such that the probability of the state to be in the target region at the (finite) time horizon is at least a given minimum probability. As the GP predictive distribution at any query point is Gaussian, the probability to transition to a point arbitrarily far from the current point is greater than zero and, thus, the system will eventually leave any compact set. Hence, for full GP dynamics the notions of deterministic analysis do not apply. Here, we extend the previous results (Vinogradska et al., 2016) substantially and analyze asymptotic behavior of closed-loop control systems with full GP dynamics. Employing methods from Markov chain theory, we show that for many choices of the prior the system converges to a unique, invariant limiting distribution.

This chapter lays a foundation for theoretical stability analysis for control of probabilistic models learned from data. The proposed approaches provide stability guarantees for many existing learning control approaches based on GPs. The chapter is organized as follows: first, we briefly review related work and introduce the problem to be addressed. Section 2.2 provides necessary background for the stability analysis. In Section 2.3, we analyze closed-loop control systems with dynamics given as the mean of a GP. Subsequently, we study the case when the dynamics are given as the full GP distribution in Section 2.4. Section 2.5 provides empirical evaluations on benchmark control tasks. A conclusion is given in Section 2.6.

2.1.1 Related Work

To date, only very few special cases of system dynamics and policies learned from data have been analyzed with respect to stability. For example, Perkins and Barto (2003) and Nakanishi et al. (2002) analyze the scenario of an agent switching between given safe controllers. Kim and Ng (2005) monitor stability while learning control for the special case of a linear controller and linear dynamics. In (Anderson et al., 2007), stability is analyzed for neural network dynamics employing classical methods for linear time invariant systems. Recently, stability properties of autoregressive systems with dynamics given as the mean of a GP were analyzed by Beckers and Hirche (2016), who derive an upper bound on the number of equilibrium points for squared exponential and some special cases of polynomial covariance functions. Furthermore, the authors compute an invariant set and show the boundedness of all system trajectories. However, no control inputs are considered in this work and no asymptotic stability results are given. Furthermore, the full GP distribution is not considered in (Beckers and Hirche, 2016). To the best of our knowledge, stability analysis for closed-loop control with probabilistic models, such as GPs, and arbitrary policies has not been addressed so far.

In classic control theory, the first formal analysis of closed-loop dynamics dates back to the 19th century (Routh, 1877; Hurwitz, 1895; Lyapunov, 1892). Lyapunov's approach allows to analyze stability for nonlinear deterministic systems $\dot{x} = f(x)$. It studies the system behavior around *equilibrium points*, i.e., points, where all derivatives $\dot{x}$ of the state x vanish and the system comes to a hold. An equilibrium point x_e is *stable*, if for every $\varepsilon > 0$ a $\delta > 0$ exists such that $\|x(t) - x_e\| < \epsilon$ for every solution $x(t)$ of the differential equation with $\|x(t_0) - x_e\| < \delta$ and $t \geq t_0$. The equilibrium point is *asymptotically stable*, if it is stable and δ can be chosen such that $\|x(t) - x_e\| \to 0$ as $t \to \infty$, see (Khalil, 2014). One approach to proving stability of x_e is to find a *Lyapunov function*, i.e., a non-negative function, that vanishes only at x_e and decreases along system trajectories. This approach can be applied to nonlinear dynamics. However, finding a Lyapunov function is typically challenging as there exists no general method for this task.

Additionally to the difficulty of finding a suitable Lyapunov function, classical Lyapunov approaches are highly challenging as proving nonnegativity of a nonlinear function is in general intractable. In (Parrilo, 2000), a relaxation of the positive definiteness problem for polynomials was introduced: a polynomial is clearly nonnegative, if it can be written as a sum of squares (SOS) of polynomials. Fortunately, checking whether a polynomial can be written as SOS can be formulated as a semidefinite program (SDP) and can be solved in polynomial time. Thus, for polynomial system dynamics one can search for polynomial Lyapunov functions conveniently by solving the corresponding SDP. This SOS relaxation has been successfully applied to a number of nonlinear stability problems as finding the region of attraction of a system (Chesi, 2004; Majumdar et al., 2014), robust analysis (Topcu et al., 2010b) or controller design (Moore and Tedrake, 2014). While there are some limitations to these methods (Majumdar et al., 2014; Ahmadi and Parrilo, 2011), SOS approaches are of high practical importance. However, SOS methods can only be applied directly to polynomial dynamics. Nonpolynomial dynamics must be recast as polynomial systems by an (automated) procedure (Papachristodoulou and Prajna, 2005). This recasting procedure introduces new variables and constraints for each nonpolynomial term and can, thus, significantly increase the size of the SDP that must be solved. Especially for nonparametric and nonpolynomial

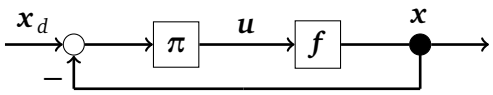


Figure 2.2.: A closed-loop control structure with controller π and the system dynamics f. We study stability for two types of dynamics: (i) the mean of the GP and (ii) the full GP predictive distribution.

models as, e.g., a GP with squared exponential or Matérn kernels, recasting often leads to SDPs that are computationally infeasible (Topcu et al., 2010a).

For consideration of uncertainty in the dynamics, stochastic differential equations (SDEs) have been introduced (Adomian, 1983). SDEs are differential equations where some terms are stochastic processes. In the presence of uncertainty, x_e is *stable in probability*, if for every $\varepsilon > 0$ and $p > 0$, there exists $\delta > 0$ such that $\mathrm{P}\{\|x(t) - x_e\| > \varepsilon\} < p$ for $t > t_0$ and solutions $x(t)$ with $\|x(t_0) - x_e\| < \delta$. Furthermore, x_e is *asymptotically stable in probability*, if it is stable in probability and $\mathrm{P}\{\|x(t) - x_e\| > \varepsilon\} \to 0$ for a suitable δ, see (Khasminskii and Milstein, 2011). Stability follows from the existence of a *supermartingale*, a stochastic Lyapunov function analogue. However, supermartingales exist only if the noise vanishes at the equilibrium point. Relaxing the supermartingale criterion allows for stability statements for a finite time horizon (Steinhardt and Tedrake, 2012; Kushner, 1966). Kushner's (1966) approach requires a supermartingale to be found manually, which is challenging in most cases. Steinhardt and Tedrake (2012) address this difficulty by constructing a supermartingale via SOS programming. This approach is suitable for systems with polynomial dynamics and shares some of the difficulties of SOS approaches as described above.

When controlling a system, uncertainty may arise from modeling inaccuracies, the presence of (external) disturbances and the lack of data, as some system parameters are not known in advance but only during operation. Robustness of nonlinear systems to structured or unstructured uncertainty has been considered in classical control theory, e.g., via the small gain theorem (Zhou and Doyle, 1998). However, these approaches typically lead to an infinite number of nonlinear matrix inequalities and cannot be solved analytically. Another approach is to take uncertainty into account when designing a controller. Thus, a controller must be suitable for a family of systems that is specified, e.g., by bounds for model parameters or for nonparametric uncertainties as bounds for the operator norm of the unknown dynamics. Robust control (Skogestad and Postlethwaite, 2005; Zhou and Doyle, 1998) designs a single controller that is provably stable for all systems in the specified family – often at cost of overall controller performance. Adaptive control (Narendra and Annaswamy, 2012; Tao, 2003) instead adjusts control parameters online in order to achieve prespecified performance, which can be computationally demanding. Both schemes require stability analysis of the dynamics system, e.g., via Lyapunov functions as described above.

2.1.2 Problem Statement

The goal of this chapter is to provide an automated tool to analyze stability of a closed-loop structure when the dynamics model is given as a GP, see Figure 2.1. As inputs, our tool expects a control policy and a GP dynamics model. It checks the stability of the corresponding closed-

loop structure and returns a stability region. Trajectories starting in this region are guaranteed to converge to the target (in probability). Subsequently, we discuss the different components of the tool in more detail.

We consider controllers which depend only on the current state and are differentiable with respect to the state. The dynamics model is given as a GP with stationary covariance function. This covariance function is assumed to have bounded derivatives with respect to all input dimensions and bounded prior mean. We consider a discrete-time system $\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ with $\boldsymbol{x}_t \in \mathbb{R}^D$, $\boldsymbol{u}_t \in \mathbb{R}^F$, $t = 1, 2, \ldots$ and a controller $\boldsymbol{\pi}\colon \mathbb{R}^D \to \mathbb{R}^F$, whose objective is to move the system to a desired state $\boldsymbol{x}_d$, see Figure 2.2. In this chapter, we study two possible cases for the dynamics $\boldsymbol{f}$: (i) the mean of a GP and (ii) the full GP predictive distribution. Note, that in the second case, distributions have to be propagated through the GP resulting in non-Gaussian state distributions.

2.2 Preliminaries

In this section, we introduce basic concepts for the proposed stability analysis. First, we briefly review Gaussian process regression, as GPs are employed to describe the considered dynamics. Second, we recap numerical quadrature, as it is the basis for the uncertainty propagation method proposed in Section 2.4.

2.2.1 Gaussian Process Regression

Given noisy observations $\mathcal{D} = \{(\boldsymbol{z}^i, y^i = f(\boldsymbol{z}^i) + \varepsilon^i) \mid 1 \leq i \leq N\}$, where $\varepsilon^i \sim \mathcal{N}(0, \sigma_n^2)$, the prior on the values of f is $\mathcal{N}(0, K(Z,Z) + \sigma_n^2 I)$. The covariance matrix $K(Z,Z)$ is defined by the choice of covariance function k as $[K(Z,Z)]_{ij} = k(\boldsymbol{z}^i, \boldsymbol{z}^j)$. One commonly employed covariance function is the squared exponential

$$k(\boldsymbol{z}, \boldsymbol{w}) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{w})^\intercal \Lambda^{-1} (\boldsymbol{z} - \boldsymbol{w})\right),$$

with signal variance σ_f^2 and squared lengthscales $\Lambda = \operatorname{diag}(l_1^2, \ldots, l_{D+F}^2)$ for all input dimensions. Given a query point $\boldsymbol{z}_*$, the conditional probability of $f(\boldsymbol{z}_*)$ is

$$f(\boldsymbol{z}_*) \mid \mathcal{D} \sim \mathcal{N}\big(\boldsymbol{k}(\boldsymbol{z}_*, Z)\boldsymbol{\beta},\ k(\boldsymbol{z}_*, \boldsymbol{z}_*) - \boldsymbol{k}(\boldsymbol{z}_*, Z)(K(Z,Z) + \sigma_n^2 I)^{-1} \boldsymbol{k}(Z, \boldsymbol{z}_*)\big) \tag{2.1}$$

with $\boldsymbol{\beta} = (K(Z,Z) + \sigma_n^2 I)^{-1} \boldsymbol{y}$. The hyperparameters, e.g. $\sigma_n^2, \sigma_f^2, \Lambda$ for the squared exponential kernel, are estimated by maximizing the log marginal likelihood of the data (Rasmussen and Williams, 2005).

In this thesis, $\boldsymbol{f}$ models system dynamics. It takes state-action pairs $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{u})^\intercal$ and outputs successor states $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$. As these outputs are multivariate, we train conditionally independent GPs for each output dimension. We write $\sigma_{n,m}^2$, $\sigma_{f,m}^2$, Λ_m for the GP hyperparameters in output dimension m and k_m for the corresponding covariance function.

2.2.2 Numerical Quadrature

Numerical quadrature approximates the value of an integral

$$\int_a^b f(\boldsymbol{x}) d\boldsymbol{x} \approx \sum_{i=1}^{p} w_i f(\boldsymbol{\xi}_i)$$

given a finite number p of function evaluations. A widely used class of quadrature rules are interpolatory quadrature rules, which integrate all polynomials up to a certain degree exactly. In this chapter, we employ Gaussian quadrature rules, where the evaluation points $\xi_1, \ldots, \xi_p$ are chosen to be the roots of certain polynomials from orthogonal polynomial families. They achieve the highest accuracy possible for univariate interpolatory formulæ (Süli and Mayers, 2003). In general, interpolatory quadrature rules are well suited for smooth functions and the approximation error depends on the magnitude of the higher order derivatives of the function.

For multivariate integrals, the quadrature problem is significantly harder. While many formulæ for the univariate case can straightforwardly be generalized to multivariate integrals, they often suffer from the curse of dimensionality. However, quadrature methods that scale better and are feasible for up to 20 dimensions have been developed. These methods can be divided in two classes: monomial rules and sparse grid methods. Both types are exact for all monomials up to a chosen degree and the number of nodes scales polynomially with the degree and number of dimensions. Monomial rules are derived by directly solving a polynomial system while sparse grid rules are obtained by combining lower dimensional Gaussian quadrature rules to achieve exactness on all monomials up to the chosen degree. An overview of these approaches can be found in (Skrainka and Judd, 2011).

2.3 Stability of GP Mean Dynamics

A wide variety of dynamics can be modeled well by the mean of a GP (Micchelli et al., 2006) due to the universal approximation property of many kernels. Thus, stability of systems with GP mean forward dynamics is an interesting problem. Please note that stability of closed-loop control systems with dynamics given as the mean of a GP is a classical problem from nonlinear control with a particular choice of dynamics function. Classical approaches to solve this problem have been discussed in Section 2.1.1 and range from direct Lyapunov approaches to the more recent SOS approach to construct polynomial Lyapunov functions. However, for GP mean dynamics, finding a Lyapunov function directly seems even more challenging than for a classical ODE derived from expert knowledge about the physical system. On the other hand, constructing a polynomial Lyapunov function as the solution of an SOS problem is typically computationally infeasible for these dynamics systems, as the nonparametric nature of GP mean functions results in a large number of nonpolynomial terms that significantly increase the complexity of the underlying semidefinite program (see Section 2.1.1). Thus, we propose an alternative approach to address this problem.

In this section, we provide tools to check the stability of a closed-loop control structure with a given differentiable Markovian control policy π and GP mean dynamics. For this class of forward dynamics, the next state $\boldsymbol{x}_{t+1}$ is given as the mean of the GP predictive distribution at the inputs $\boldsymbol{x}_t, \boldsymbol{u}_t = \pi(\boldsymbol{x}_t)$. Firstly, we consider infinite time horizons. In this case, we attempt to find a region of starting points in the state space, such that trajectories starting in this region are guaranteed to converge to the desired point $\boldsymbol{x}_d$ as $t \to \infty$. To obtain this result, we analyze the sensitivity of the GP predictive mean to variations in the input space and derive upper bounds for the distance of the predictions at two different points. This analysis can be used to find a region around the target point $\boldsymbol{x}_d$, where the next state is closer to $\boldsymbol{x}_d$ than the point before. It follows straightforwardly, that the full metric ball of maximal radius, which lies completely inside this region, is a stability

region. All trajectories that start inside of this metric ball converge to the target point as $t \to \infty$. Furthermore, we will show that in this metric ball, the distance of the current state to the target decreases exponentially. We will exploit this statement to derive some finite time horizon and robustness results. For finite time horizons, we aim to find a region in the state space, such that the state does not deviate more than a given tolerance from the target when the time horizon is reached. The robustness analysis deals with asymptotic stability of the GP mean dynamics when disturbances are present. We derive conditions for these disturbances, such that convergence to the target state $\boldsymbol{x}_d$ is still guaranteed.

In the following, we will briefly review the notion of stability employed for the analysis and derive the main result of this section: an algorithm to find a stability region. After proving correctness of this algorithm, we will derive statements on finite time stability of the GP mean dynamics and on infinite time stability of the disturbed system.

2.3.1 Stability Notion

If the system dynamics $\boldsymbol{f}$ is given by the mean of a GP, the resulting closed-loop structure is deterministic. To assess the quality of a controller π, which aims to stabilize the system at the reference point $\boldsymbol{x}^d$, we propose an algorithm to find a stability region of the closed-loop system.

Definition 1. *The reference point $\boldsymbol{x}_d$ is stable, if for every $\varepsilon > 0$ there exists $\delta > 0$, such that $\|\boldsymbol{x}_t - \boldsymbol{x}_d\| < \varepsilon$ for $t = 1, 2, \ldots$ and $\|\boldsymbol{x}_0 - \boldsymbol{x}_d\| < \delta$. If $\boldsymbol{x}_d$ is stable and there exists a $\delta_0 > 0$ such that $\|\boldsymbol{x}_t - \boldsymbol{x}_d\| \to 0$ for $t \to \infty$, $\|\boldsymbol{x}_0 - \boldsymbol{x}_d\| < \delta_0$, $\boldsymbol{x}_d$ is asymptotically stable. A subset X^c of the state space is a stability region, if $\|\boldsymbol{x}_t - \boldsymbol{x}_d\| \to 0$ as $t \to \infty$ for all $\boldsymbol{x}_0 \in X^c$.*

Please note that π is implicit in this definition, as the states $\boldsymbol{x}_t$ for $t = 1, 2, \ldots$ depend on π via $\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \pi(\boldsymbol{x}_t))$. This definition matches Lyapunov's stability notion (Khalil, 2014). We will present an algorithm that checks asymptotic stability of $\boldsymbol{x}_d$ by constructing a stability region X^c.

When the time horizon considered is finite, we are interested in finding all starting states, such that the distance to the target is at most a given $\varepsilon > 0$.

Definition 2. *A subset X^c of the state space is an (ε, T)-stability region, if $\|\boldsymbol{x}_t - \boldsymbol{x}_d\| < \varepsilon$ holds for all $\boldsymbol{x}_0 \in X^c$ and $t \geq T$.*

In the following, we propose an algorithm to verify asymptotic stability of $\boldsymbol{x}_d$ as in Definition 1 and prove its correctness. Subsequently, we show how to find finite time statements as in Definition 2.

2.3.2 Algorithm Sketch

In this section, we derive an algorithm, that can find a stability region of a closed-loop control structure with GP mean dynamics. To find a stability region, we analyze how the distance between the current state and the target evolves. The basic idea of the algorithm involves *positively invariant sets*, i.e., sets that once entered, the system will never leave again (Blanchini, 1999). More precisely, we aim to find all $\boldsymbol{x}$, such that

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| < \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\| \tag{2.2}$$

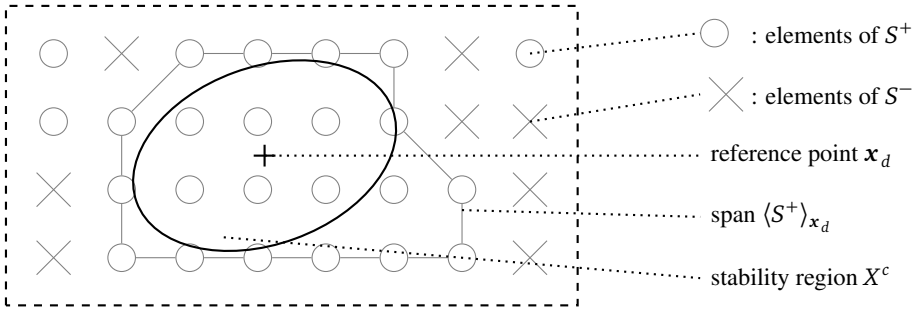


Figure 2.3.: Basic idea of Algorithm 1 to construct a stability region: for a finite set of states S, employ upper bound from Lemma 1 to check whether the successor state is closer to $\boldsymbol{x}_d$, obtaining sets S^+ and S^-. Return metric ball of maximal radius that fits into the span $\langle S^+ \rangle_{\boldsymbol{x}_d}$ as stability region X^c. For $\boldsymbol{x}^{(0)} \in X^c$, the controlled system never leaves X^c and converges to $\boldsymbol{x}_d$ as $t \to \infty$.

for $\boldsymbol{x}^t = \boldsymbol{x}$ and a fixed $0 < \gamma < 1$. Assume there is a region containing $\boldsymbol{x}_d$ where Eq. (2.2) holds. For all states in this region, the distance to the target point decreases in one time step. If it is possible to fit a full metric ball centered at the target point $\boldsymbol{x}_d$ in this region, then all trajectories starting in the ball will never leave it. In addition, the distance of the current state to the reference point $\boldsymbol{x}_d$ will decrease in every step by at least factor γ. Thus, it follows immediately from the existence of such a ball that the target point is asymptotically stable. However, finding a region in closed form where Eq. (2.2) holds, is usually intractable. Instead, we follow an algorithmic approach to construct a stability region.

The proposed algorithm (Alg. 1) finds a region where Eq. (2.2) holds if one exists. It employs an upper bound for $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\|$, that depends only on $\boldsymbol{x}_t$. With this upper bound we can check whether Eq. (2.2) holds for a (finite) set of points. However, to find a continuous state space region where Eq. (2.2) holds, this upper bound must allow for generalization from discrete points. More precisely, we need an upper bound of $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\|$ that depends on $\boldsymbol{x}_t$ smoothly, thus, having bounded gradients, and can be handled conveniently. We employ an upper bound derived from sensitivity analysis of the GP mean to variations in the input space. This bound depends on $\boldsymbol{x}_t$ in a way that can be exploited to compute a finite set of points S, e.g., a grid that is sufficient to consider as follows: For any grid point, we check if Eq. (2.2) holds, obtaining the set S^+ with the grid points that fulfill Eq. (2.2) and the rest in S^-. Let $\langle S^+ \rangle_{\boldsymbol{x}_d}$ be the polygon spanned by the connected component of S^+, which contains $\boldsymbol{x}_d$. We can choose S such that Eq. (2.2) holds for all points in $\langle S^+ \rangle_{\boldsymbol{x}_d}$, see Figure 2.3. Algorithm 1 gives an overview, Sec. 2.3.3 provides technical details and proves correctness of the approach.

2.3.3 Correctness of the Algorithm

In this section, we elaborate on the computation steps of Algorithm 1 and prove that the returned state space region X^c is a stability region of the closed-loop system in the sense of Definition 1. Fundamental to Algorithm 1 is the upper bound for $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\|$, which can be obtained as follows. We denote $\hat{\boldsymbol{x}} := (\boldsymbol{x}, \pi(\boldsymbol{x}))^\intercal$ and recall that $\boldsymbol{f}(\hat{\boldsymbol{x}})$ is the GP predictive mean at $\hat{\boldsymbol{x}}$, see Eq. (2.1).

Algorithm 1 Stability region X^c for GP mean dynamics

Input: dynamics GP $\boldsymbol{f}$, control policy π, $\boldsymbol{x}_d$, γ
Output: stability region X^c

```
Construct grid S with Lemma 2, S⁺ ← ∅, X^c ← ∅
for x_0 ∈ S do
    Compute upper bound C(x_0, x_d, π) ≥ ||x_1 − x_d|| with Lemma 1
    if C < γ ||x_0 − x_d|| then S⁺ ← S⁺ ∪ {x} fi
od
if ⟨S⁺⟩_{x_d} exists then
    Fit metric ball B(x_d) ⊆ ⟨S⁺⟩_{x_d}
    X^c ← B(x_d) fi
return X^c
```

Lemma 1. *Let $\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^d \in \mathbb{R}^{D+F}$ and $B \in \mathbb{R}^{D\times D}$ be a positive definite matrix. The distance of GP predictive means $\boldsymbol{f}(\hat{\boldsymbol{x}})$ and $\boldsymbol{f}(\hat{\boldsymbol{x}}^d)$ in the metrics induced by B is bounded by*

$$\|\boldsymbol{f}(\hat{\boldsymbol{x}}) - \boldsymbol{f}(\hat{\boldsymbol{x}}^d)\|_B^2 \leq (\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}^d)^\intercal M(\boldsymbol{x}, \boldsymbol{x}_d, \pi)(\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}^d) =: C(\boldsymbol{x}, \boldsymbol{x}_d, \pi)$$

with a symmetric matrix M. This matrix can be constructed explicitly and depends on $\boldsymbol{x}, \boldsymbol{x}_d$ and the policy π.

Proof. This statement is obviously true for $\boldsymbol{x} = \boldsymbol{x}_d$, so let $\boldsymbol{x} \neq \boldsymbol{x}_d$. Evaluating

$$\|\boldsymbol{f}(\hat{\boldsymbol{x}}) - \boldsymbol{f}(\hat{\boldsymbol{x}}^d)\|_B^2 = \sum_{m,m'=1}^{D} \sum_{i,k=1}^{N} b_{mm'}(k_m(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^i) - k_m(\hat{\boldsymbol{x}}^d, \hat{\boldsymbol{x}}^i))(k_{m'}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^k) - k_{m'}(\hat{\boldsymbol{x}}^d, \hat{\boldsymbol{x}}^k))\beta_{mi}\beta_{m'k} \tag{2.3}$$

we realize the need for an upper bound of $k_m(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^i) - k_m(\hat{\boldsymbol{x}}^d, \hat{\boldsymbol{x}}^i)$. Recall that the covariance functions k_m are differentiable with bounded derivatives with respect to $\boldsymbol{x}$ and also, that the control policy π is differentiable with respect to the state $\boldsymbol{x}$. Thus, we can integrate the gradient field of k_m along a curve $\boldsymbol{\tau}$ from $\hat{\boldsymbol{x}}^d$ to $\hat{\boldsymbol{x}}$. As this path integral does not depend on the particular curve $\boldsymbol{\tau}$, we may choose $\boldsymbol{\tau} = \boldsymbol{\tau}^{D+F} \dots \boldsymbol{\tau}^1$ as the curve along the edges of the hypercube defined by $\hat{\boldsymbol{x}}^d$ and $\hat{\boldsymbol{x}}$, i.e., $\boldsymbol{\tau}_p^j(t) = \hat{x}_p$ if $p \leq j-1$, $\boldsymbol{\tau}_p^j(r) = \hat{x}_p^d + r(\hat{x}_p - \hat{x}_p^d)$ with $r \in [0;1]$ if $p = j$, and $\boldsymbol{\tau}_p^j(t) = \hat{x}_p^d$ otherwise. This definition yields

$$k_m(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^i) - k_m(\hat{\boldsymbol{x}}^d, \hat{\boldsymbol{x}}^i) = \sum_{j=1}^{D+F} \int_{\hat{x}_j^d}^{\hat{x}_j} \frac{\partial k_m(\boldsymbol{\chi}, \hat{\boldsymbol{x}}^i)}{\partial \chi_j} \bigg|_{\boldsymbol{\chi}=\boldsymbol{\tau}^j} d\chi_j \tag{2.4}$$

and we compute the partial derivatives $\partial k_m(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^d)/\partial \hat{x}_j$ in all state-action space dimensions $1 \leq j \leq D+F$.

We rewrite the sum in Eq. (2.3) by substituting Eq. (2.4) and the partial derivatives of the covariance functions. To find an upper bound for this sum, we estimate the occurring integrals by the product of integration interval length and an upper or lower mean value according to the

sign of the respective summand. The necessary upper and lower mean values can be obtained via Riemannian upper and lower sums or the upper and lower bounds for the partial derivatives of k with respect to $\boldsymbol{x}$. Sorting the summands by products of integration interval lengths $(\hat{x}_j - \hat{x}_j^d)(\hat{x}_p - \hat{x}_j^d)$, this sum can be rewritten as a quadratic form. The entries of M can be chosen to form a symmetric matrix by making $M_{pj} = M_{jp}$ half of the coefficient of $(\hat{x}_j - \hat{x}_j^d)(\hat{x}_p - \hat{x}_p^d)$. □

For any point in the state action space, we can find an upper bound for the distance of its prediction and the target point. Note that this distance estimated by Lemma 1 depends heavily on the eigenvalues of $M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})$. This fact is exploited to compute a grid S, which constitutes the first step of Algorithm 1. As M is constructed as a symmetric matrix, the eigenvalue problem is well conditioned, i.e., when M is perturbed, the change in the eigenvalues of M is at most as large as the perturbation.

Lemma 2. *Let $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^D$ and $M(\boldsymbol{x}, \boldsymbol{x}_d, \pi)$, B be as defined in Lemma 1. If $\|\boldsymbol{f}(\hat{\boldsymbol{x}}) - \boldsymbol{f}(\hat{\boldsymbol{x}}_d)\|_B < c\|\boldsymbol{x} - \boldsymbol{x}_d\|$ for $c < 1$, there exist Δ_j such that $\|\boldsymbol{f}(\hat{\boldsymbol{z}}) - \boldsymbol{f}(\hat{\boldsymbol{x}}_d)\|_B < c\|\boldsymbol{z} - \boldsymbol{x}_d\|$, for all $\boldsymbol{z}$ with $|\hat{z}_j - \hat{x}_j| < \Delta_j$, $1 \leq j \leq D + F$.*

Proof. Solving for the eigenvalues of $M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})$, the set $Q_{\boldsymbol{x}} := \{v \in \mathbb{R}^{D+F} \mid (v - \hat{\boldsymbol{x}}^d)^\intercal M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})(v - \hat{\boldsymbol{x}}^d) < a\}$ can be determined. It is symmetric to the axes defined by the eigenvectors of $M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})$ and meets them at $\pm a^{-1}\sqrt{\lambda}$. For $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{\Delta}$ we want to ensure $\boldsymbol{z} \in Q_{\boldsymbol{z}}$, if $\boldsymbol{x} \in Q_{\boldsymbol{x}}$. Thus, we estimate how much the eigenvalues of $M(\boldsymbol{z}, \boldsymbol{x}_d, \boldsymbol{\pi})$ differ from those of $M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})$. As $M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})$ is symmetric, the eigenvalue problem has condition $\kappa(\lambda, M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})) = 1$ for any eigenvalue λ. Thus, $\left|{}^{\partial\lambda}/_{\partial\hat{x}_j}\right| \leq \left\|{}^{\partial M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})}/_{\partial\hat{x}_j}\right\|$. Computing $\left\|{}^{\partial M(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{\pi})}/_{\partial\hat{x}_j}\right\|$ or upper bounds for this expression allows solving for all Δ_j. □

We are now able to compute a grid S, such that it is sufficient to check Eq. (2.2) for all grid points to retrieve a (continuous) stability region. While the grid width may become small, there is a lower bound for it. As the GP falls back to the prior far away from training data, the entries of M are bounded for bounded mean priors and, being continuous, Lipschitz. Thus, a lower bound exists.

Theorem 1. *The region X^c returned by Algorithm 1 is a stability region. All trajectories starting in X^c move closer to the desired point $\boldsymbol{x}_d$ in each step. Convergence to $\boldsymbol{x}_d$ is guaranteed for all points in X^c as $t \to \infty$.*

Proof. We exploit Lemma 2 to compute a grid S as the first step in Algorithm 1. Employing Lemma 1, the algorithm checks for all points in S whether Eq. (2.2) holds, obtaining the sets S^+ and S^-, respectively. Lemma 2 ensures that Eq. (2.2) also holds for all points inside the polygon $\langle S^+ \rangle_{\boldsymbol{x}_d}$. For every point in $\langle S^+ \rangle_{\boldsymbol{x}_d}$, the next state is closer to the target point $\boldsymbol{x}_d$. Fitting a full metric ball $\mathcal{B}(\boldsymbol{x}_d)$ in $\langle S^+ \rangle_{\boldsymbol{x}_d}$, we recover an invariant set X^c. More precisely, the trajectories starting in X^c move closer to $\boldsymbol{x}_d$ with every time step. □

Theorem 1 provides an asymptotic stability result for closed-loop control systems with dynamics given as the mean of a GP and can, thus, can be applied to a broad class of dynamics systems. However, the procedure is based on a grid in the state space and as a consequence suffers from the curse of dimensionality. This limitation is not surprising, as it is well known that estimating stability regions of high degree is NP-hard (Ahmadi et al., 2013). However, several approaches

can help to scale the proposed method to higher dimensions. Instead of computing a global grid width $\boldsymbol{\Delta}$ as in Lemma 2, the proof can straightforwardly be adapted to compute local grid widths. Such local grid widths depend on the magnitude of the derivatives of the GP mean with respect to the inputs and, thus, can be substantially larger than the global one, e.g., when moving away from training data or where the dynamics is very slow. With this technique, the number of grid points can be greatly reduced. Furthermore, as we are constructing an inner approximation to the full stability region, the order of the grid points processed by Algorithm 1 can be modified to avoid redundant computations. When starting at the equilibrium point and expanding the considered grid points spherically according to the chosen distance metrics, computation can be stopped as soon as the first grid point is tested to lie outside the stability region. This first negative test point limits the stability region and all grid points with the same or greater distance to $\boldsymbol{x}_d$ need not be considered anymore.

2.3.4 Finite Time Horizons and Robustness

In this section, we will employ the proposed Algorithm 1 to derive results for finite time horizons and stability in the presence of disturbances.

Finite Time GP Mean Stability

Above, we proposed an algorithm to check asymptotic stability of the target $\boldsymbol{x}_d$ and to find a stability region, if one exists. In this section, we are interested in finding all starting states which deviate from the target by at most a given tolerance $\varepsilon > 0$ after $T < \infty$ timesteps. State space regions that fulfill this criterion were introduced in Definition 2 as (ε, T)-*stability regions*.

The main result of this section follows from the infinite time horizon results of Sec. 2.3.3. Recall that Algorithm 1 computes a state space region X^c and $\gamma < 1$, such that

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| < \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\|$$

for all $\boldsymbol{x}_t \in X^c$. We can exploit this result to find a state space region $X^c_{\varepsilon,T}$, such that $\|\boldsymbol{x}_T - \boldsymbol{x}_d\| < \varepsilon$ for all $\boldsymbol{x}_0 \in X^c_{\varepsilon,T}$.

Lemma 3. *Let $\varepsilon > 0$, $T < \infty$ and $X^c \subseteq X, \gamma < 1$ such that $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| < \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\|$ for all $t = 1, 2, \ldots$ and $\boldsymbol{x}_0 \in X^c$. Then $\|\boldsymbol{x}_T - \boldsymbol{x}_d\| < \varepsilon$ for all $\boldsymbol{x}_0 \in X^c$ with $\|\boldsymbol{x}_0 - \boldsymbol{x}_d\| < \varepsilon/\gamma^T$.*

Proof. Choosing $\boldsymbol{x}_0 \in X^c$ and iteratively employing $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| < \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\|$ we get

$$\|\boldsymbol{x}_T - \boldsymbol{x}_d\| < \gamma^T \|\boldsymbol{x}_0 - \boldsymbol{x}_d\| \overset{!}{<} \varepsilon.$$

Solving for ε, we obtain $\|\boldsymbol{x}_0 - \boldsymbol{x}_d\| < \varepsilon/\gamma^T$ and, thus, $X^c_{\varepsilon,T} := \mathcal{B}_{\varepsilon/\gamma^T}(\boldsymbol{x}_d)$ is an (ε, T)-stability region. □

Lemma 3 enables finite time horizon statements. For a given tolerance ε we can compute a region $X^c_{\varepsilon,T}$ such that $\|\boldsymbol{x}_t - \boldsymbol{x}_d\| < \varepsilon$ for all $t \geq T$ if $\boldsymbol{x}_0 \in X^c_{\varepsilon,T}$. Such stability statements are relevant for many applications as controllers are often designed for finite system runtimes.

In this section, we will analyze the behavior of closed-loop control systems with dynamics given as the mean of a GP when disturbances are present. More precisely, we assume that the next state is given as

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{r}_t \tag{2.5}$$

with the disturbance $\boldsymbol{r}_t < \infty$ at time t, that does not depend on the system state. We aim to find conditions for $\boldsymbol{r}_t < \infty$ such that $\boldsymbol{x}_t$ still converges to $\boldsymbol{x}_d$ as $t \to \infty$. Let $R_t \coloneqq \|\boldsymbol{r}_t\|$ be the magnitude of the disturbance at time t. Based on Equation (2.2) we derive an upper bound for R_t. Recall that Algorithm 1 returns a stability region X^c, which is a full metric ball and let ρ be the radius of this ball. Convergence to the target state is guaranteed, as long as the system does not leave X^c and $R_t \to 0$ as $t \to \infty$.

Theorem 2. *Let X^c be a full metric ball of radius ρ that is a stability region of the undisturbed closed-loop control system with GP mean dynamics. For $\boldsymbol{x}_0 \in X^c$, the system in Eq. (2.5) converges to the target state $\boldsymbol{x}_d$, if*

$$R_t \leq \rho - \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\| \tag{2.6}$$

for all t and there exists some T_0 such that

$$R_t < (c - \gamma)\|\boldsymbol{x}_t - \boldsymbol{x}_d\| \tag{2.7}$$

with a constant $c < 1$ holds for all $t \geq T_0$.

Proof. Firstly, we will show that $\boldsymbol{x}_{t+1} \in X^c$, if $\boldsymbol{x}_t \in X^c$ and Eq. (2.6) holds. Employing Equation (2.5), we get

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| = \|\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{r}_t - \boldsymbol{x}_d\| \leq \|\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) - \boldsymbol{x}_d\| + R_t \tag{2.8}$$

and assuming $\boldsymbol{x}_t \in X^c$, it follows

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| \leq \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\| + R_t. \tag{2.9}$$

It follows $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| \leq \rho$, if $R_t \leq \rho - \gamma\|\boldsymbol{x}_t - \boldsymbol{x}_d\|$ and, thus, $\boldsymbol{x}_{t+1} \in X^c$. By induction, $\boldsymbol{x}_t \in X^c$ for all $t > 0$, if $\boldsymbol{x}_0 \in X^c$ and Equation (2.6) holds for all t.

If the disturbance $\boldsymbol{r}_t$ vanishes after a finite number of time steps, it is sufficient to ensure Equation (2.6) to show convergence to the target state. After this finite number of disturbed steps, the system state will still lie in X^c and for all time steps to come, the dynamics will resemble the non-disturbed dynamics. Convergence is then guaranteed by Theorem 1.

Next, we will show that $\boldsymbol{x}_t$ converges to $\boldsymbol{x}_d$, if the disturbance $\boldsymbol{r}_t$ decreases sufficiently fast, as stated in Equation (2.7). For $\boldsymbol{x}_t \in X^c$, Equation (2.9) holds. To ensure convergence to $\boldsymbol{x}_d$, we aim to decrease the distance to the target state with every time step. Thus,

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| \leq \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\| + R_t \overset{!}{<} c\, \|\boldsymbol{x}_t - \boldsymbol{x}_d\| \tag{2.10}$$

for a constant $c < 1$. Solving for R_t, we obtain Eq. (2.7). □

This theorem provides a criterion for the disturbance such that the closed-loop control system will still converge to the target state. However, in practice, Equations (2.6) and (2.7) may be difficult to verify, as they involve $\|\boldsymbol{x}_t - \boldsymbol{x}_d\|$ for all t. We derive a closed-form upper bound for the magnitude R_t of the disturbance $\boldsymbol{r}_t$.

Lemma 4. *The Inequality (2.6) holds for all t, if*

$$R_t \leq \rho - \left(\gamma^t \|\boldsymbol{x}_0 - \boldsymbol{x}_d\| + \sum_{k=1}^{t-1} \gamma^k R_k \right). \tag{2.11}$$

The inequality (2.7) holds for $t \geq T_0$, if

$$R_t \leq (c^{t+1} - \gamma c^t) \|\boldsymbol{x}_{T_0} - \boldsymbol{x}_d\|. \tag{2.12}$$

Proof. Both inequalities can be obtained by iteratively applying Eqs. (2.6) and (2.7).

$$\begin{aligned} \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| &\leq \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\| + R_t \\ &\leq \gamma \left(\gamma \|\boldsymbol{x}_{t-1} - \boldsymbol{x}_d\| + R_{t-1}\right) + R_t \\ &\leq \gamma^t \|\boldsymbol{x}_0 - \boldsymbol{x}_d\| + \sum_{k=1}^{t-1} \gamma^k R_k + R_t \overset{!}{\leq} \rho \end{aligned}$$

and analogously

$$\begin{aligned} \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_d\| &\leq \gamma \|\boldsymbol{x}_t - \boldsymbol{x}_d\| + R_t \\ &< \gamma (c \|\boldsymbol{x}_{t-1} - \boldsymbol{x}_d\|) + R_t \\ &< \gamma c^t \|\boldsymbol{x}_{T_0} - \boldsymbol{x}_d\| + R_t \overset{!}{\leq} c^{t+1} \|\boldsymbol{x}_{T_0} - \boldsymbol{x}_d\|. \end{aligned}$$

□

Employing Lemma 4, we can analyze stability for disturbed system trajectories, if the magnitude of the disturbance (or an upper bound for the magnitude) is known for every time step.

2.4 Stochastic Stability of GP Dynamics

In this section, we study closed-loop systems with dynamics given as a full GP distribution. For any query point $\boldsymbol{x}_t$ a GP predicts the next state $\boldsymbol{x}_{t+1}$ to be normally distributed. If, however, $\boldsymbol{x}_t$ is not a point, but a distribution, the integral

$$p(\boldsymbol{x}_{t+1}) = \int_{\mathbb{R}^D} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t)\, p(\boldsymbol{x}_t)\, d\boldsymbol{x}_t \tag{2.13}$$

determines the next state distribution. Note that $p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t)$ is Gaussian with respect to $\boldsymbol{x}_{t+1}$. The next state distribution $p(\boldsymbol{x}_{t+1})$, however, is not Gaussian, even if $p(\boldsymbol{x}_t)$ is. Generally, $p(\boldsymbol{x}_{t+1})$ is

analytically intractable and only approximations, e.g., via moment matching (Quiñonero-Candela et al., 2003) or linearization (Ko and Fox, 2008), can be computed. These methods suffer from severe inaccuracies in many cases, e.g., they cannot handle distributions with multiple modes. Also, no pointwise approximation error bounds are available. Thus, these methods are unsuitable for stability analysis. In this thesis, we propose numerical quadrature (Sec. 2.4.2) to approximate $p(\boldsymbol{x}_{t+1})$, instead. This technique yields significantly better results, e.g., it can handle distributions with multiple modes. In addition, error analysis is readily available (Wasowicz, 2006; Masjed-Jamei, 2014) and can be employed to derive stability guarantees for a finite time horizon.

We will also analyze the closed-loop system behaviour for infinite time horizons. The proposed numerical quadrature approximation converges to a stationary limiting distribution. Unfortunately, the error bounds based on quadrature error analysis do not apply when infinitely many timesteps are taken. However, we will show that closed-loop control systems with GP dynamics expose the same infinite time horizon behavior – they converge to a stationary distribution.

In the following, we will discuss *finite time stochastic stability*, introduce an algorithm to find a stability region based on numerical quadrature, and prove its correctness. Subsequently, we will elaborate on the construction of good quadrature rules for multi-step-ahead predictions. Finally, we will analyze the system behavior for infinite time horizons.

2.4.1 Stability Notion

Consider a deterministic system, that is locally, but not globally, asymptotically stable. Adding noise to the system may render the target point unstable. Especially when noise is unbounded (e.g., Gaussian), all trajectories will eventually leave any ball around the target point with probability one. For this reason, in the study of SDEs, other stability notions than Lyapunov's, are common. In particular, bounding the probability to drift away from the target over a finite time T is desirable, as in the following definition (Kushner, 1966).

Definition 3. *Let Q_1, Q_2 be subsets of the state space X, with Q_2 open and $Q_1 \subset Q_2 \subset X$. The system is finite time stable with respect to $Q_1, Q_2, 1-\lambda, T$, if $\boldsymbol{x}_0 \in Q_1$ implies $P\{\boldsymbol{x}_t \in Q_2\} \geq 1-\lambda$ for all $t \leq T$.*

However, we are interested in finding a set Q_s of initial conditions, such that the goal Q is reached within time T with a desired probability (cf. Steinhardt and Tedrake 2012).

Definition 4. *The set Q_s is a stability region with respect to the target region Q, time horizon T and success probability $1-\lambda$, if $P\{\boldsymbol{x}_T \in Q\} \geq 1-\lambda$ holds for all $\boldsymbol{x}_0 \in Q_s$ with $\lambda > 0$ and the target region $Q \subset X$.*

This definition focuses on reaching a target Q after time T, whereas Definition 3 bounds the exit probability from a region Q_2 within time $0 \leq t \leq T$. The methods proposed in this chapter can be employed to analyze stability in the sense of both definitions. In the following, we will present an algorithm to find a stability region according to Definition 4.

2.4.2 Algorithm Sketch

Algorithm 2 Stability region for GP dynamics

Input: dynamics GP $\boldsymbol{f}$, control policy π, time horizon T, target region Q, approximation error tolerance e_{tol}, desired success probability $1-\lambda$
Output: stability region X^c

Construct grid S with Lemma 6
$S^+ \leftarrow \emptyset$
Compute quadrature $\boldsymbol{w}, \boldsymbol{\xi}$ based on Eq. (2.18) and Lemma 5, such that $\|\epsilon_T\|_{\mathcal{C}(X)} < e_{\text{tol}}$
$\boldsymbol{\phi} \leftarrow \left(\boldsymbol{f}(\xi_1, \pi(\xi_1)), \ldots, \boldsymbol{f}(\xi_N, \pi(\xi_N))\right)^\intercal$
$\boldsymbol{m} \leftarrow (\int_Q \phi_1(\boldsymbol{x}) d\boldsymbol{x}, \ldots, \int_Q \phi_N(\boldsymbol{x}) d\boldsymbol{x})^\intercal$
for $\boldsymbol{x} \in S$ **do**
  $p(\boldsymbol{x}_1) \leftarrow \boldsymbol{f}(\boldsymbol{x}, \pi(\boldsymbol{x}))$
  $\boldsymbol{x}_T \leftarrow \boldsymbol{\alpha}_T \boldsymbol{\phi}$
  $P_{min}\{\boldsymbol{x}_T \in Q\} \leftarrow \boldsymbol{\alpha}_T \boldsymbol{m} - \text{vol}(Q)\|\epsilon_T\|_{\mathcal{C}(X)}$
  if $P_{min}\{\boldsymbol{x}_T \in Q\} > 1-\lambda$ **then** $S^+ \leftarrow S^+ \sqcup \{\boldsymbol{x}\}$ **fi**
od
return $X^c \leftarrow \langle S^+ \rangle$

We will now discuss how to find a stability region as in Definition 4 for a closed-loop system with GP dynamics. To analyze system behavior, the capability to compute next state distributions is crucial. As discussed previously, Eq. (2.13) is analytically intractable and approximation methods must be employed. We propose numerical quadrature to approximately propagate distributions. We will show that numerical quadrature approximates state distributions as Gaussian mixture models. Fortunately, computation of multi-step-ahead predictions becomes convenient and fast even for long time horizons. However, relying on approximations of the state distribution is not sufficient for stability guarantees. The error introduced by approximation and error propagation must be bounded to recover reliable statements on the probability for $\boldsymbol{x}_T$ to be in a set Q. This error bound $\epsilon_t(\boldsymbol{x})$ at time t can be obtained following one of the available quadrature error analyses, e.g., by Masjed-Jamei (2014).

With numerical quadrature and quadrature error analysis, we can compute a lower bound for the *success probability*, i.e., the probability for $\boldsymbol{x}_T$ to be in the target set Q. A priori, this enables checking success probabilities for a finite set of points. Fortunately, as in the case of GP mean dynamics, the statement can be generalized to continuous regions. Our algorithm constructs a grid in the state space and computes success probabilities for all grid points. We prove that a stability region can be inferred from these grid point results.

In the following, we elaborate on numerical quadrature as approximate inference method and, studying quadrature error propagation, prove the correctness of the proposed approach. The presented tool is outlined in Algorithm 2.

In most applications, especially when the states have physical interpretations, the state space is bounded. For this reason, we assume $X = [a_1, b_1] \times \cdots \times [a_D, b_D]$ and solve

$$F[p(\boldsymbol{x}_t)] := \int_X p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t. \tag{2.14}$$

We propose numerical quadrature to approximate this integral. We choose a composed Gaussian product quadrature rule, which will be detailed in Section 2.4.4. For the rest of this section, it is sufficient to note that our quadrature rule provides a set of evaluation points $\mathbb{X}$ and positive weights w_n for all nodes $\boldsymbol{\xi}_n \in \mathbb{X}$. Integral (2.14) is then approximated by

$$F[p(\boldsymbol{x}_t)] \approx \sum_{\boldsymbol{\xi}_n \in \mathbb{X}} w_n p(\boldsymbol{x}_{t+1} | \, \boldsymbol{x}_t = \boldsymbol{\xi}_n) p(\boldsymbol{x}_t = \boldsymbol{\xi}_n), \tag{2.15}$$

resulting in a weighted sum of Gaussian distributions. The approximate state distribution at time $t+1$ can be given by

$$p(\boldsymbol{x}_{t+1}) \approx \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_{t+1} \tag{2.16}$$

with $\alpha_{t+1,n} := w_n p(\boldsymbol{x}_t = \boldsymbol{\xi}_n)$, $\phi_n(\boldsymbol{x}) := p(\boldsymbol{x}_{t+1} = \boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}_n)$. Note that the Gaussian basis functions $\phi_n(\boldsymbol{x})$ do not change over time, so the state distribution at time t is represented by the weight vector $\boldsymbol{\alpha}_t$. To propagate any distribution multiple steps through the GP, the basis functions ϕ_n must be calculated only once and the task reduces to sequential updates of the weight vector $\boldsymbol{\alpha}$. As $p(\boldsymbol{x}_t) \approx \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_t$, the weight vector $\boldsymbol{\alpha}_{t+1}$ is given by

$$\boldsymbol{\alpha}_{t+1} = \operatorname{diag}(\boldsymbol{w}) \Phi \boldsymbol{\alpha}_t = (\operatorname{diag}(\boldsymbol{w}) \Phi)^t \boldsymbol{\alpha}_1 \tag{2.17}$$

with the matrix Φ, $\Phi_{ij} = \phi_j(\boldsymbol{\xi}_i)$ with $1 \leq i, j \leq n$, which contains the basis function values at all grid points. In practice, it is helpful to normalize the matrix $\operatorname{diag}(\boldsymbol{w})\Phi$ such that each column sums to 1. In this case, unit vectors will be mapped to unit vectors. This ensures that our approximate state distribution is in fact a probability density, i.e., integrates to 1.

Our algorithm aims to find a stability region Q_s, i.e., a region where the success probability is at least $1 - \lambda$ for a given time horizon T, target region Q, and $\lambda > 0$. Computing $\boldsymbol{m} = \int_Q \boldsymbol{\phi}(\boldsymbol{x}) d\boldsymbol{x}$, the probability for $\boldsymbol{x}_T$ to be in Q is approximately $\boldsymbol{m}^\intercal \boldsymbol{\alpha}_T$.

2.4.3 Correctness of the Algorithm

We will now show that the region returned by Algorithm 2 is a stability region as in Definition 4. In Section 2.4.2, we proposed numerical quadrature to approximate GP predictions when the input is a distribution. However, to obtain stability guarantees it is not sufficient to consider approximate solutions. For a lower bound on the success probability, additional knowledge about the approximation error and how it is propagated through the dynamics GP is essential.

When approximate inference steps are cascaded for multi-step-ahead predictions, errors are introduced and propagated through the dynamics. Typically, error propagation fortifies initial errors with every time step. The difference of approximation and true distribution after a finite time is bounded. Fortunately, numerical quadrature allows to control this error bound by refining the used quadrature rule.

The derivation of an upper bound for the approximation error after a finite number of time steps relies heavily on quadrature error analysis, i.e., upper bounds for the errors that are made in one time step when numerical quadrature is employed. Several error analyses for numerical quadrature are available (Masjed-Jamei, 2014; Davis et al., 2014; Wasowicz, 2006; Evans, 1994) with the classic analyses relying on higher order derivatives of the function to be integrated. We will employ the error analysis by Masjed-Jamei (2014) that requires first order derivatives only. More precisely (Masjed-Jamei 2014, Theorem 2.1), for a differentiable function f with $\nu_1(x) \leq f'(x) \leq \nu_2(x)$ with continuous functions ν_1, ν_2 the quadrature error is upper and lower bounded

$$m \leq \sum_{i=1}^{N} w_i f(\xi_i) - \int_a^b f(x)dx \leq M \tag{2.18}$$

with the bounds

$$m = \sum_{i=0}^{N} \int_{\xi_i}^{\xi_{i+1}} \left(\nu_1(t) \frac{\nu_i(t) + |\nu_i(t)|}{2} + \nu_2(t) \frac{\nu_i(t) - |\nu_i(t)|}{2} \right) dt$$

$$M = \sum_{i=0}^{N} \int_{\xi_i}^{\xi_{i+1}} \left(\nu_1(t) \frac{\nu_i(t) - |\nu_i(t)|}{2} + \nu_2(t) \frac{\nu_i(t) + |\nu_i(t)|}{2} \right) dt$$

for $\xi_0 = a$, $\xi_{N+1} = b$ and $\nu_i = t - \left(a + \sum_{j=1}^{i} w_i\right)$, $i = 0, \ldots, n$. In the following, we employ this result to derive an error bound for the pointwise approximation error of the proposed multi-step-ahead predictions based on numerical quadrature.

Lemma 5. *Let $\boldsymbol{x}_0 \in X$ and ϵ_T be the pointwise approximation error $\epsilon_T(\boldsymbol{x}) = p(\boldsymbol{x}_T = \boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{x})^\intercal \boldsymbol{\alpha}_T$ at time T. There exists an upper bound for $\|\epsilon_T\|_{\mathcal{C}(X)}$ that can be controlled by the choice of quadrature rule.*

Proof. The statement obviously holds for T=0 and for T=1, as $p(\boldsymbol{x}_1)$ is Gaussian and computed exactly. If $T \geq 2$, an error is introduced as $p(\boldsymbol{x}_t)$ cannot be computed analytically for $t \geq 2$. Employing numerical quadrature to compute $p(\boldsymbol{x}_2)$, we obtain $p(\boldsymbol{x}_2) = \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_2 + \epsilon_2$ with our approximation $\boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_2$ and the unknown initial quadrature error $\epsilon_2(\boldsymbol{x})$. However, the error analysis by Masjed-Jamei (2014, Theorem 2.1) gives us a bound for $\|\epsilon_2\|_{\mathcal{C}(X)} := \max_{\boldsymbol{x} \in X} |\epsilon_2(\boldsymbol{x})|$. When propagating further, an unknown, bounded error term and an approximation term must be handled. For the approximation term, we get $F[\boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_t] = \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_{t+1} + \boldsymbol{\epsilon}^\intercal \boldsymbol{\alpha}_t$ with quadrature error bounds $\boldsymbol{\epsilon} = (\epsilon_i(\boldsymbol{x}))_i$ for the integrals (2.14), when setting $p(\boldsymbol{x}_t) = \phi_i$. The error term $\epsilon_t(\boldsymbol{x})$ is propagated to $F[\epsilon_t]$ with $\max_{\boldsymbol{x} \in X} |F[\epsilon_t]| \leq C \|\epsilon_t\|_{\mathcal{C}(X)}$ and $C = \max_{\boldsymbol{x} \in X} \int p(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t) d\boldsymbol{x}_t$. Finally, the error at time T is bounded by $\|\epsilon_T\|_{\mathcal{C}(X)} \leq \|\boldsymbol{\epsilon}\|^\intercal \boldsymbol{\alpha}_{T-1} + \sum_{t=2}^{T-2} C^t \|\boldsymbol{\epsilon}\|^\intercal \boldsymbol{\alpha}_t + C^{t-1} \|\epsilon_2\|$, where $\boldsymbol{\epsilon}$ and ϵ_2 can be made arbitrarily small by refining the quadrature rule. □

Lemma 5 theoretically enables stability guarantees for finite time horizons T. However, as with any approximation of the distribution (2.14), this error bound grows exponentially with T, while quadrature error decreases polynomially with function evaluations. This drawback limits real-word application of Lemma 5 to small T, but we found that typically, approximation behaves far better than the worst-case bound.

For any starting point, we can now compute the approximate state distribution at time T and a lower bound on the success probability, i.e., the probability for $\boldsymbol{x}_T$ to be in the target region Q. It remains to show that it is sufficient to compute the success probability for a discrete set of starting states and generalize to the underlying continuous state space region.

Lemma 6. *Let $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^D$ be starting states and $T \in \mathbb{N}$. If $P\{\boldsymbol{x}_T \in Q\} \geq 1-\lambda$, there exist Δ_j such that $P\{\boldsymbol{z}_T \in Q\} \geq 1-\lambda$ for all $\boldsymbol{z}$ with $|z_j - x_j| < \Delta_j$, $1 \leq j \leq D$.*

Proof. Firstly, we note that the absolute error bound does not depend on the starting state. The state distribution $p(\boldsymbol{x}_1)$ is the GP prediction at the query point $\boldsymbol{x}_0$. Thus, for $T = 1$ the claim follows from Lemma 2. When $t \geq 2$, only an approximation of the state distribution is known. The mixture model weights at time $t = 2$ depend only on the values of $p(\boldsymbol{x}_1)$ at the set of evaluation points $\mathbb{X}$. Thus, the lemma follows for $T = 2$. If $t > 2$, the difference of the approximate distributions for the starting points $\boldsymbol{x}$ and $\boldsymbol{z}$ is

$$\boldsymbol{\phi}^\intercal(\boldsymbol{\alpha}_{\boldsymbol{x},t} - \boldsymbol{\alpha}_{\boldsymbol{z},1}) = \boldsymbol{\phi}^\intercal(\mathrm{diag}(\boldsymbol{w})\Phi)^{t-2}(\boldsymbol{\alpha}_{\boldsymbol{x},1} - \boldsymbol{\alpha}_{\boldsymbol{z},2}).$$

Thus, the difference in approximate probability mass is linear in $(\boldsymbol{\alpha}_{\boldsymbol{x},2} - \boldsymbol{\alpha}_{\boldsymbol{z},2})$. This fact concludes the proof. □

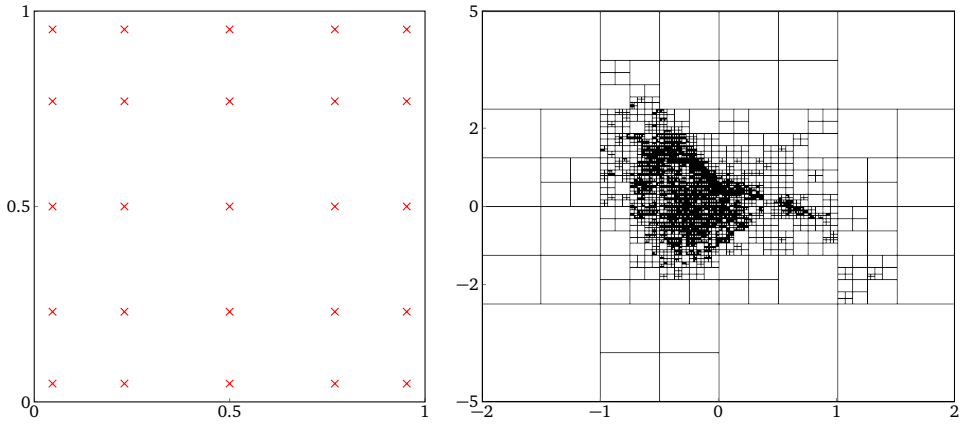
Finally, we are able to prove the main result of this section, the correctness of Algorithm 2.

Theorem 3. *The region X^c returned by Algorithm 2 is a stability region in the sense of Definition 4. For all starting points $\boldsymbol{x}_0 \in X^c$, the probability mass in the target region Q at time T is at least $1-\lambda$.*

Proof. For the first step in Algorithm 2 we exploit Lemma 6 to construct a grid S. For S, stability follows for the region around the grid points S^+ with success probability greater than $1-\lambda$. Secondly, employing Lemma 5 we determine a quadrature rule that ensures the pointwise error at time T to be at most e_{tol}. Finally, we compute approximate success probabilities for all grid points, as in Sec. 2.4.2. Subtracting the maximum error mass $e_{\mathrm{tol}}\,\mathrm{vol}(Q)$ in target region Q, we obtain all grid points S^+, which have a success probability of at least $1-\lambda$. □

2.4.4 Construction of Numerical Quadratures for Uncertainty Propagation

In the sections above, we introduced numerical quadrature to approximate GP predictions at uncertain inputs and showed that this approximate inference method can be used to derive finite time horizon stability guarantees. The presented theoretical results are valid for any choice of quadrature rule that allows for quadrature error analysis. However, in practice, the choice of a suitable quadrature rule is crucial to the applicability of the proposed algorithm. Thus, in the



(a) Product Quadrature **(b)** Mountain Car

Figure 2.4.: Construction of suitable quadrature rules. Plot (a) shows the nodes of a Gaussian product quadrature rule on the unit square. Here, the quadrature rule for the unit square was constructed as an outer product of the Gaussian quadrature rule with 5 nodes. The state space partition for the mountain car system obtained with Algorithm 3 is shown in (b). For each rectangle in this partition, a Gaussian product quadrature such as (a) is employed.

following, we will elaborate on how to find "good" quadrature rules for a given GP dynamics and control policy.

Gaussian product quadrature extends univariate Gaussian quadrature to a multivariate rule using a product grid of evaluation points. This construction has some desirable properties, such as positive quadrature weights and readily available quadrature nodes. However, being an outer product of one-dimensional rules, it suffers from the curse of dimensionality. In this chapter, we apply numerical quadrature to integrals as in Equation (2.14). Typically, the system trajectories are not uniformly spread over the state space. Instead, they are concentrated in a significantly smaller region. We exploit this observation to improve the efficiency of product grid quadrature and cope with the curse of dimensionality. For this purpose, we partition the state space $X = X_1 \sqcup \cdots \sqcup X_L$ and apply a Gaussian product quadrature to each obtained subregion X_l. The next state distribution, cf. Equation (2.14), can be written as

$$F[p(\boldsymbol{x}_t)] := \int_X p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t = \sum_{l=1}^{L} \int_{X_l} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t$$

and, applying a numerical quadrature rule with nodes $\mathbb{X}_l$ for each integral, we get

$$F[p(\boldsymbol{x}_t)] \approx \sum_{l=1}^{L} \sum_{\xi_n \in \mathbb{X}_l} w_{nl} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t = \xi_n) p(\boldsymbol{x}_t = \xi_n).$$

Setting $\mathbb{X} = \mathbb{X}_1 \sqcup \cdots \sqcup \mathbb{X}_L$, we recover Equation (2.15). Thus, composed quadrature rules can be handled just as a single Gaussian product rule. We exploit this to construct quadrature rules which

Algorithm 3 Construction of composed quadrature rules

Input: dynamics GP $\boldsymbol{f}\colon (\boldsymbol{x}_t, \boldsymbol{u}_t) \mapsto \boldsymbol{x}_{t+1}$, control policy $\pi\colon \boldsymbol{x} \mapsto \pi_{\boldsymbol{\theta}}(\boldsymbol{x})$ with parameters $\boldsymbol{\theta}$, state space X, maximum partition size $L_{\max}$

Output: composed quadrature rule with nodes $\mathbb{X}$ and weight vector $\boldsymbol{w}$

Initialize partition $X = X_1 \sqcup \cdots \sqcup X_s$ and quadrature $\mathbb{X} = \mathbb{X}_1 \sqcup \cdots \sqcup \mathbb{X}_s$ with $s < L_{\max}$
while $s < L_{\max}$ **do**
  Sample starting state $\boldsymbol{x}_0$, policy parameters $\boldsymbol{\theta}^*$
  Compute mean rollout $\boldsymbol{\tau}_0 = \boldsymbol{x}_0, \ldots, \boldsymbol{\tau}_H$ with dynamics GP $\boldsymbol{f}$ and policy $\pi_{\boldsymbol{\theta}^*}$
  for $l = 1, \ldots, s$ **do**
    if $\mathrm{vol}(X_l) / |\mathbb{X}_l| \min_{\boldsymbol{\tau}_i \in X_l} \mathrm{Var}(\boldsymbol{\tau}_i) < 1$ **then** subdivide X_l, add nodes to $\mathbb{X}$ **fi**
  od
od
return quadrature nodes $\mathbb{X}$ and weights $\boldsymbol{w}$

are efficient for the computation of next state distributions for our particular GP dynamics model and controller. In other words, we aim to find a partition $X = X_1 \sqcup \cdots \sqcup X_L$ of the state space, such that integral (2.14) is approximated well by the resulting quadrature rule. For this purpose, we maintain a partition of the state space, sample mean trajectories $\boldsymbol{\tau} = \boldsymbol{\tau}_0, \ldots, \boldsymbol{\tau}_H$ and subdivide regions X_l that were visited and fulfill a certain criterion. More precisely, we subdivide X_l if X_l does not contain enough quadrature nodes to integrate predictive distributions from the dynamics GP well. To estimate whether X_l should be divided, we introduce

$$\rho_l(\boldsymbol{\tau}) \coloneqq \frac{\mathrm{vol}(X_l)}{|\mathbb{X}_l| \min_{\boldsymbol{\tau}_i \in X_l} \mathrm{Var}(\boldsymbol{\tau}_i)}, \tag{2.19}$$

where $\mathrm{Var}(\boldsymbol{\tau}_i)$ denotes the variance of the GP predictive distribution at the point $\boldsymbol{\tau}_i$, and subdivide X_l if $\rho_l(\boldsymbol{\tau})$ is greater than 1. This criterion relates the higher order derivatives of GP predictions which fall inside X_l with the quadrature node density in X_l. Constructing a composed quadrature rule with this approach will concentrate most quadrature nodes in state space regions that are visited frequently when following system trajectories. Algorithm 3 summarizes our approach and Figure 2.4 illustrates the constructed quadrature rules.

2.4.5 GP Dynamics with Infinite Time Horizons

In Section 2.4.2, we proposed numerical quadrature to approximate next state distributions for closed-loop control systems, where the dynamics is given as a full Gaussian process. For this approximate inference, pointwise error bounds are available. Based on this error analysis, we were able to derive stability guarantees for finite time horizons. In this section, we analyze the case when the time horizon is infinite. Unfortunately, error bounds as in Section 2.4.3 cannot be given in this case. As we will see, the numerical quadrature approximation converges to a stationary limit distribution, i.e., this distribution does not depend on the starting state, as t goes to infinity. As the quadrature error cannot be bounded in this case, there is no guarantee that the system dynamics will converge to the same distribution. However, we will show that closed-loop control systems

with GP dynamics also converge to a stationary distribution as $t \to \infty$ for many choices of the prior. We will first analyze the behavior of the proposed numerical quadrature approximation and subsequently study asymptotic behavior of closed-loop control systems with GP dynamics.

Asymptotic Behavior of NQ Approximated GP Dynamics

In this section we study the properties of numerical quadrature as approximate inference when the time horizon is infinite. Recall that with numerical quadrature the state distribution is approximated by

$$p(\boldsymbol{x}_t) \approx \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_t \tag{2.16 revisited}$$

with Gaussian basis functions $\boldsymbol{\phi} = (\phi_1, \dots, \phi_N)^\intercal$, that do not change over time, and the weight vector $\boldsymbol{\alpha}_t$ at time t. As stated in Equation (2.17), the weight vector $\boldsymbol{\alpha}_t$ is updated by

$$\boldsymbol{\alpha}_{t+1} = \operatorname{diag}(\boldsymbol{w})\Phi\boldsymbol{\alpha}_t \tag{2.17 revisited}$$

with the positive matrix $\operatorname{diag}(\boldsymbol{w})\Phi$ as in Section 2.4.2. More precisely, $\operatorname{diag}(\boldsymbol{w})\Phi$ is a left stochastic matrix. Thus, any column vector $\boldsymbol{\beta}$ with norm $\|\boldsymbol{\beta}\|_1 = 1$ is mapped to a unit vector, i.e. $\|\operatorname{diag}(\boldsymbol{w})\Phi\boldsymbol{\beta}\|_1 = 1$. In this case, Equation (2.17) represents a power iteration which leads to the following theorem.

Theorem 4. *Let $(\mathbb{X}, \boldsymbol{w})$ be a quadrature rule with positive weights $\boldsymbol{w}$ on the state space X and $\boldsymbol{f} : (\boldsymbol{x}_t, \boldsymbol{u}_t) \mapsto \boldsymbol{x}_{t+1}$ be a dynamics GP. The approximate state distribution $p(\boldsymbol{x}_t) \approx \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_t$ obtained by cascading approximate one step ahead predictions with the quadrature $(\mathbb{X}, \boldsymbol{w})$ converges to a unique stationary distribution, independent of the starting distribution, as $t \to \infty$.*

Proof. Consider Equation (2.17). The matrix $\operatorname{diag}(\boldsymbol{w})\Phi$ has norm one, as each of its columns sums to one. Thus, as the starting weights $\boldsymbol{\alpha}_0$ form a stochastic vector, Equation (2.17) takes the form of a power iteration, see (Golub and Van Loan, 2013, Section 7.3). By the Perron-Frobenius theorem the largest magnitude eigenvalue λ_1 of $\operatorname{diag}(\boldsymbol{w})\Phi$ is positive and simple. More precisely, the eigenvalue of $\operatorname{diag}(\boldsymbol{w})\Phi$ with the highest absolute value is $\lambda_1 = 1$. The corresponding dominant eigenvector v_1 is positive. No other eigenvectors of $\operatorname{diag}(\boldsymbol{w})\Phi$ are non-negative. Thus, the power iteration in Equation (2.17) converges to the dominant eigenvector, if $\boldsymbol{\alpha}_0$ has a nonzero projection on the eigenspace of $\lambda_1 = 1$. As $\boldsymbol{\alpha}_0$ is a non-negative unit vector and v_1 is positive, the projection to the dominant eigenspace is ${v_1}^\intercal \boldsymbol{\alpha}_0 > 0$. Thus, the numerical quadrature approximation of $p(\boldsymbol{x}_t)$ converges to $\boldsymbol{\phi}^\intercal v_1$. □

This theorem shows that the numerical quadrature approximation to the system state converges to a stationary distribution. The limiting distribution corresponds to the dominant eigenvector of the iteration matrix $\operatorname{diag}(\boldsymbol{w})\Phi$ and does not depend on the starting state. However, no bound for the approximation error can be obtained for the limit $t \to \infty$. Thus, in contrast to finite time horizon analysis, no statement about the true system state follows from this result. In the following, we will study asymptotic behavior of closed-loop control systems with GP dynamics to see whether the (exact) system state converges to a limiting distribution as well.

We have seen that employing the numerical quadrature approximation for next state distributions, closed-loop control systems with GP dynamics converge to a limiting distribution. In this section, we will show that these closed-loop control systems expose the same behavior even if no approximations are employed. More precisely, a closed-loop control system with dynamics given as a GP will finally converge to a limiting distribution that is independent of the starting state for many common choices of the prior. To obtain this result, we apply Markov Chain theory as in (Meyn and Tweedie, 2009).

Consider a Markov chain $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \ldots\}$ with the random variables $\boldsymbol{\Sigma}_t$ taking values in $\mathbb{R}^D$ and a *Markov kernel* $P\colon \mathbb{R}^D \times \mathscr{B}(\mathbb{R}^D) \to \mathbb{R}$, i.e., $P(\cdot, A)$ is a non-negative measurable function on $\mathbb{R}^D$, $P(\boldsymbol{x}, \cdot)$ is a probability measure on $\mathscr{B}(\mathbb{R}^D)$ and

$$\mathbb{P}_\mu[\boldsymbol{\Sigma}_0 \in A_0, \cdots, \boldsymbol{\Sigma}_n \in A_n] = \int_{\boldsymbol{y}_0 \in A_0} \cdots \int_{\boldsymbol{y}_{n-1} \in A_{n-1}} \mu(d\boldsymbol{y}_0) P(\boldsymbol{y}_0, d\boldsymbol{y}_1) \cdots P(\boldsymbol{y}_{n-1}, A_n) \tag{2.20}$$

for any initial distribution μ and $n \in \mathbb{N}$. We will employ some definitions and results from (Meyn and Tweedie, 2009) to prove our claim.

Definition 5. *The following definitions will help describing Markov chains and the sets they (re-)enter.*

1. *For $A \in \mathscr{B}(\mathbb{R}^D)$ and $\Sigma_0 \in A$, the first return time τ_A is given as*

$$\tau_A := \min\{n \geq 1 \colon \boldsymbol{\Sigma}_n \in A\}.$$

   We also define the return time probability $L(\boldsymbol{x}, A) := \mathbb{P}_{\boldsymbol{x}}(\tau_A < \infty)$.

2. *We call $\boldsymbol{\Sigma}$ φ-irreducible, if there exists a measure φ on $\mathscr{B}(\mathbb{R}^D)$, such that $\varphi(A) > 0$ implies $L(\boldsymbol{x}, A) > 0$ for all $\boldsymbol{x} \in X$ and $A \in \mathcal{B}(\mathbb{R}^D)$. ψ is a maximal irreducibility measure, if $\boldsymbol{\Sigma}$ is ψ-irreducible and $\varphi \prec \psi$, i.e., $\psi(A) = 0 \Rightarrow \varphi(A) = 0$, for all irreducibility measures φ of $\boldsymbol{\Sigma}$.*

3. *We say that the set $C \in \mathscr{B}(\mathbb{R}^D)$ is small, if there exists a probability measure ν, such that $P^n(\boldsymbol{x}, A) \geq \delta\, \nu(A)$ with some integer n, $\delta > 0$ for all $\boldsymbol{x} \in C$ and all $A \in \mathcal{B}(\mathbb{R}^D)$.*

For finite or countable state spaces, a Markov kernel defines the probability to move from one state to another. The state space partitions into *communication classes*, such that all states in one class can be reached with positive probability from any other state in the same class. If there is only one communication class, the Markov chain is called irreducible – its analysis cannot be further reduced to the consideration of multiple chains with smaller state spaces. The communication structure has an immense effect on the long term behavior of a Markov chain. However, when considering general state spaces (e.g., continuous as in our case), the probability to reach one single state will be zero. The concept of communication is thus generalized to Borel sets via irreducibility measures. Every irreducible Markov chain has maximum irreducibility measures, see Theorem 5. While these are not unique, all maximum irreducibility measures of a Markov chain are equivalent,

i.e., they have the same null sets. Small sets play an important role in the analysis of Markov chains on general state spaces, as they behave analogously to single states in discrete state spaces. Sets which consist of one single point, are small. However, it can be shown that for irreducible Markov chains, any set that will be entered with positive probability contains a non-trivial small set (Meyn and Tweedie, 2009, Theorem 5.2.2).

Theorem 5. *Let $\boldsymbol{\Sigma}$ be φ-irreducible. Then there exists a maximal irreducibility measure ψ, which is equivalent to*

$$\psi'(A) \coloneqq \int_X \varphi(d\boldsymbol{y}) K_{a\frac{1}{2}}(\boldsymbol{y}, A)$$

with the transition kernel $K_{a\frac{1}{2}}(\boldsymbol{y}, A) = \sum_{n=1}^{\infty} P^n(\boldsymbol{x}, A) 2^{-(n+1)}$.

Proof. The proof is given in (Meyn and Tweedie, 2009, Proposition 4.2.2). □

We aim to analyze the asymptotic behavior of Markov chains that are defined by our closed-loop control structure with GP dynamics. One type of asymptotic behavior that is well-known from Markov chains on discrete state spaces is periodicity. The notion of periodicity can be extended to Markov chains on general state spaces as follows.

Definition 6. *Let $\boldsymbol{\Sigma}$ be irreducible with maximal irreducibility measure ψ. The period of $\boldsymbol{\Sigma}$ is the largest integer $d > 0$, such that there exist non-empty, pairwise disjoint sets $X_1, \cdots, X_d \subseteq \mathbb{R}^D$ with*

$$P(\boldsymbol{x}, X_{i+1}) = 1 \text{ for all } \boldsymbol{x} \in X_i, 1 \leq i \leq d-1 \text{ and } P(\boldsymbol{x}, X_1) = 1 \text{ for } \boldsymbol{x} \in X_d$$

and

$$\psi\left(\bigcup_{i=1}^{d} X_i\right) = 1.$$

If $d = 1$ we call $\boldsymbol{\Sigma}$ aperiodic, if $d > 1$ the Markov chain $\boldsymbol{\Sigma}$ is periodic.

Next, we apply the definitions above to establish some properties of closed-loop control systems with GP dynamics. Fortunately, the topology of $\mathbb{R}^D$ and the Gaussian state predictions obtained from a GP largely facilitate the analysis.

Lemma 7. *The Markov chain $\boldsymbol{\Sigma}$ on $\mathbb{R}^D$ defined by a GP forward dynamics is irreducible with respect to the Lebesgue measure λ and aperiodic. Also, all compact sets are small.*

Proof. The transition probabilities are Gaussian, i.e.,

$$P(\boldsymbol{x}, \cdot) = \mathcal{N}(\boldsymbol{m}(\boldsymbol{x}), S(\boldsymbol{x}))$$

and the Markov kernel has a Gaussian density $q \colon \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$. Thus, $q(\boldsymbol{x}, \boldsymbol{y}) > 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D$ and for any $A \subseteq \mathbb{R}^D$ with $\lambda(A) > 0$ it holds

$$P(\boldsymbol{x}, A) = \int_A q(\boldsymbol{x}, \boldsymbol{y}) \lambda(d\boldsymbol{y}) > 0,$$

so $\boldsymbol{\Sigma}$ is irreducible with respect to the Lebesgue measure λ.

Assume $\boldsymbol{\Sigma}$ has period $d > 1$, i.e., $P(\boldsymbol{x}, X_i) = 1$ for $\boldsymbol{x} \in X_{i-1}$, then $1 = P(\boldsymbol{x}, X_i) = \int_{X_i} q(\boldsymbol{x}, \boldsymbol{y})\lambda(d\boldsymbol{y})$ and, as $q > 0$, it follows that the complement $\overline{X}_i$ of X_i is a λ-null set, i.e., $\lambda(\overline{X}_i) = 0$. However, as X_i, X_j are disjoint for $j \neq i$ it holds $X_j \subseteq \overline{X}_i$. It follows $\lambda(X_j) = 0$ for $j \neq i$. Analogously as for X_i, $\lambda(\overline{X}_j) = 0$. As λ is a non-trivial measure, this is a contradiction. Thus, $d = 1$ and $\boldsymbol{\Sigma}$ is aperiodic.

Finally, as P has a continuous, positive density, there is a lower bound $\delta > 0$ for this density on any compact set. Thus, all compact sets are small. □

Above, we showed that the Markov chains considered are aperiodic. The Markov chain can still re-enter sets infinitely often with positive probability. A stronger form of this property is Harris recurrence. As we will see later, Harris recurrence along with some other properties implies the existence and uniqueness of an invariant state distribution.

Definition 7. *Let $A \in \mathcal{B}(\mathbb{R}^D)$ and consider the event*

$$\{\boldsymbol{\Sigma} \in A \text{ infinitely often}\} := \bigcap_{N=1}^{\infty} \bigcup_{k=N}^{\infty} \{\boldsymbol{\Sigma}_k \in A\}.$$

The set A is called Harris recurrent, if

$$Q(\boldsymbol{x}, A) := \mathbb{P}_{\boldsymbol{x}}\{\boldsymbol{\Sigma} \in A \text{ infinitely often}\} = 1.$$

The Markov chain $\boldsymbol{\Sigma}$ is Harris recurrent, if it is ψ-irreducible and $A \in \mathcal{B}(X)$ is Harris recurrent if $\psi(A) > 0$.

We say that the measure μ is an invariant measure for the Markov kernel P, if

$$\mu(A) = \int_X \mu(d\boldsymbol{x})P(\boldsymbol{x}, A),$$

or shortly $\mu P = \mu$.

The Markov chain $\boldsymbol{\Sigma}$ is positive Harris recurrent, if it is Harris recurrent and admits an invariant probability measure.

Finally, we can state some conditions for the existence and uniqueness of an invariant state distribution as well as convergence of the system state to this distribution (in total variation).

Theorem 6. *If the chain $\boldsymbol{\Sigma}$ is φ-irreducible, aperiodic and positive Harris recurrent, then it admits a unique invariant probability measure μ^* and for all probability measures ν it holds*

$$\|\nu P^n - \mu^*\|_{TV} := \sup_{A \in \mathcal{B}(\mathbb{R}^D)} |\mu^*(A) - \nu(A)| \to 0$$

as $n \to \infty$.

Proof. The proof can be found in (Meyn and Tweedie, 2009, Theorem 13.0.1). □

To employ this theorem to our case of closed-loop control systems with GP dynamics, it remains to establish Harris recurrence. However, it can be challenging to verify the definition directly. Fortunately, drift criteria can be employed. In the following, we will write $Pf := \int P(\boldsymbol{x}, d\boldsymbol{y}) f(\boldsymbol{y})$ for any function $f : \mathbb{R}^D \to \mathbb{R}$ for notational convenience.

Theorem 7. *Let P be a φ-irreducible and aperiodic Markov kernel. If there exists a function $V : \mathbb{R}^D \to [0, \infty)$ with*

$$PV \leq V + c\mathbb{1}_A \tag{2.21}$$

for a small set A, a constant $c < \infty$, and the sets $\{\boldsymbol{x} \mid V(\boldsymbol{x}) \leq r\}$ are small for all $r \in \mathbb{R}$, then P is Harris recurrent. P is positive Harris recurrent, if Eq. (2.21) can be modified to

$$PV \leq V - \varepsilon + c\mathbb{1}_A \tag{2.22}$$

for some $\varepsilon > 0$.

Proof. The proof can be found in (Meyn and Tweedie, 2009, Theorems 9.1.8, 11.0.1). □

Such a function V is called *drift function*. It is the stochastic analogue to a Lyapunov function. Inequality (2.21) states that whenever the system state is outside of the small set A, V is expected to decrease in the following step. Thus, the system tends towards the minimum of V, which is a recurrent behavior. The speed of convergence can be incorporated in a drift criterion as well.

Definition 8. *The Markov Kernel P has geometric drift towards a set A, if there exists a function $V : \mathbb{R}^D \to [1, \infty]$, finite for some $\boldsymbol{x}$, and $\beta < 1, c < \infty$, such that*

$$PV \leq \beta V + c\mathbb{1}_A. \tag{2.23}$$

We say that a φ-irreducible Markov Kernel P with invariant probability measure μ^* is V-geometrically ergodic on a set A for a function $V : \mathbb{R}^D \to [1, \infty]$, if

$$\sup_{|f| \leq V} \left| \int f \, d(P^n(\boldsymbol{x}, \cdot) - \mu^*) \right| \leq cV(\boldsymbol{x})\rho^n \tag{2.24}$$

for all $\boldsymbol{x} \in A$, a constant $c < \infty$ and $\rho < 1$ and $V(\boldsymbol{x}) < \infty$ for all $\boldsymbol{x} \in A$.

We will employ geometric drift functions to prove the convergence of Markov chains generated by closed-loop control systems with GP forward dynamics at an exponential convergence speed, irrespective of the starting state. Studying the drift function $V(\boldsymbol{x}) = 1 + \|\boldsymbol{x}\|^2$, the following criterion for geometric ergodicity can be derived.

Theorem 8. *A Markov kernel P with Gaussian transition probabilites has V-geometric drift towards a compact set if and only if*

$$\limsup_{\|\boldsymbol{x}\| \to \infty} \frac{\|\boldsymbol{m}(\boldsymbol{x})\|^2 + \operatorname{tr}(S(\boldsymbol{x}))}{\|\boldsymbol{x}\|^2} < 1 \tag{2.25}$$

with the mean map $\boldsymbol{m}(\boldsymbol{x})$ and variance map $S(\boldsymbol{x})$. It follows that P has a unique invariant probability measure μ^* and is V-geometrically ergodic.

Proof. The proof is given in (Hansen, 2003, Lemma 3.2, Corollary 3.3). □

Finally, we can prove the main result of this section, the convergence of many closed-loop control systems with GP dynamics towards a unique, invariant limiting distribution that is independent of the starting state.

Theorem 9. *Any discrete-time closed-loop control system with state space $\mathbb{R}^D$, where the forward dynamics is given as GP with zero-mean prior and stationary covariance function, is geometrically ergodic, i.e., the system has a unique invariant probability measure and the Markov chains generated by the system converge to its stationary distribution with exponential speed, see Equation (2.24).*

Proof. We have to verify Eq. (2.25). As the GP predictions fall back to the zero mean prior far away from the training data, $\|\boldsymbol{m}(\boldsymbol{x})\|^2 + \operatorname{tr}(S(\boldsymbol{x}))$ converges to $\sum_{k=1}^{D} \sigma_{k,f}^2$ as $\|\boldsymbol{x}\| \to \infty$. Thus,

$$\limsup_{\|\boldsymbol{x}\|\to\infty} \frac{\|\boldsymbol{m}(\boldsymbol{x})\|^2 + \operatorname{tr}(S(\boldsymbol{x}))}{\|\boldsymbol{x}\|^2} = 0 < 1,$$

which concludes the proof. □

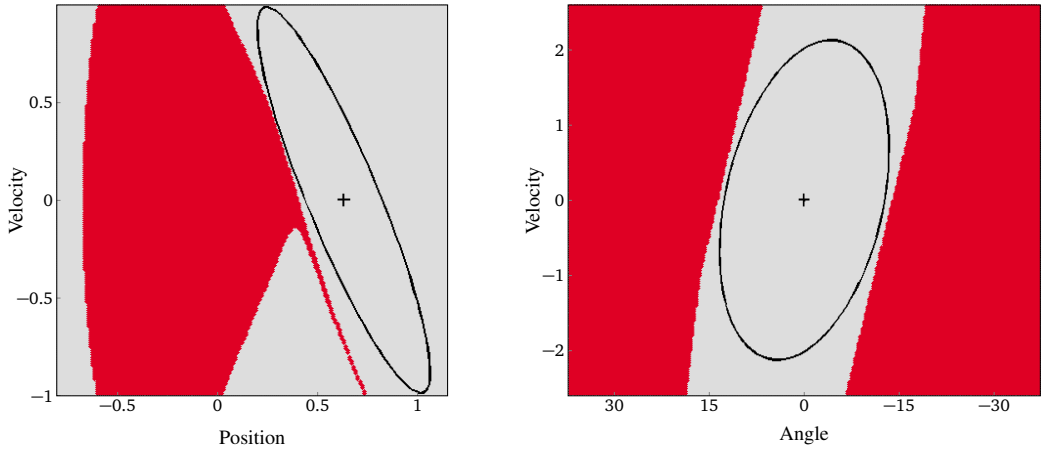
This theorem gives us a strong result about the asymptotic behavior of closed-loop control systems with dynamics given as a GP with stationary covariance function and zero mean prior. This result applies to many learning control approaches in the literature.

2.5 Empirical Evaluation

In this section, we evaluate the previously obtained theoretical results on two benchmark tasks: mountain car and inverted pendulum. Moreover, the performance of the proposed uncertainty propagation is compared to the state-of-the-art moment matching approach and Monte Carlo sampling. We begin with a brief introduction of the test-beds.

Mountain Car. A car with limited engine power has to reach a desired point in the mountainscape (Sutton and Barto, 1998). The state space has two dimensions: position and velocity of the car. We analyze stability of a PD-controller $\pi((x, \dot{x})^\intercal) = K_p x + K_d \dot{x}$. The gains are chosen as $K_p = 25$ and $K_d = 1$ and the control signal is limited to $u_{\max} = 4$. The GP dynamics model was trained on 250 data points from trajectories with random starting points and control gains.

Inverted Pendulum. In the inverted pendulum task, the goal is to bring the pendulum to an upright position with limited torque (see Doya, 2000) and balance it there. The system state has two dimensions: pendulum angle and velocity. We evaluate stability of a PD-controller with $K_p = 6$, $K_d = 3$ and control limit $u_{\max} = 1.2$. The dynamics GP was trained on 200 points from rollouts with random starting points and control gains.



(a) Mountain Car **(b)** Inverted Pendulum

Figure 2.5.: Stability regions for GP mean dynamics on the two benchmark tasks. This figure shows empirically obtained stability regions in gray, points that did not converge in red. The ellipsoid indicates the stability region returned by Algorithm 1.

Cart-Pole. In the cart-pole domain (Deisenroth et al., 2015), a cart with an attached free-swinging pendulum is running on a track of limited length. The goal is to swing the pendulum up and balance it, with the cart in the middle of the track. The state space has four dimensions: position of the cart x, velocity of the cart $\dot{x}$, angle of the pendulum ϑ and the angular velocity $\dot{\vartheta}$. A horizontal force with $u_{\max} = 10$ can be applied to the cart. To demonstrate the proposed approach we analyze stability of the cart-pole system for a PD-controller with randomly chosen gains. The dynamics GP was trained on 250 points from rollouts with sampled starting state and control gains.

2.5.1 Stability of GP Predictive Mean Dynamics

To evaluate the presented theory on stability of the closed-loop control system with GP mean dynamics, a stability region is determined as described in Section 2.3. We compare this region to the true stability region, empirically obtained as follows. A grid on the state space is defined and rollouts from every grid point are computed. After a sufficiently long time (1000s), we check whether the state has converged to the target point. Thus, we empirically determine a region of starting points, where the system converges to the desired state. Figure 2.5 shows the obtained regions for the mountain car and pendulum systems. In both cases the theoretically obtained stability region, which is marked by an ellipsoid, is a subset of the empirically determined region. This effect is due to our analysis yielding a full metric ball centered around the target point, although the full stability region is not necessarily convex. Also, trajectories which first move away from the target point, but finally converge to it, are not considered in the presented theory. Instead, all trajectories starting in the theoretically obtained stability region move towards the target point contractively.

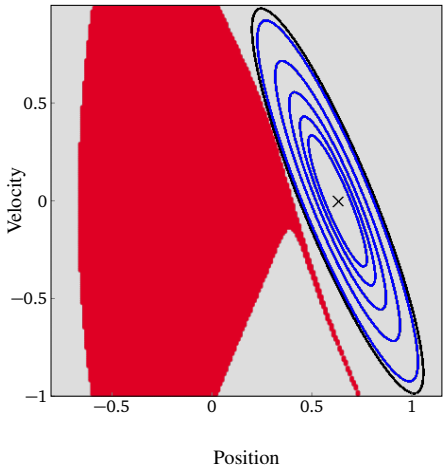


Figure 2.6.: Finite time stability of GP mean dynamics. The black ellipsoid marks the (infinite time) stability region obtained with Algorithm 1. The blue ellipsoids mark the (ε, T)-stability regions with $\varepsilon = 0.05$ and $T = 10, 20, 30, 40, 50$ timesteps (inner to outer ellipsoid). For any starting point inside of one blue ellipsoid, the distance to the target is less than ε after T timesteps.

Next, we compute finite time (ε, T)-stability regions as in Lemma 3. Figure 2.6 shows the obtained regions for $\varepsilon = 0.05$ and different choices of T. By construction, these are full metric balls around the reference point and lie completely inside of the stability region obtained with Algorithm 1.

Finally, we consider robustness of the GP mean dynamics. We introduce disturbances and apply the convergence criteria from Theorem 2 and Lemma 4. We sample the starting time and duration as well as the magnitude of the disturbance. The direction of the disturbance is then sampled independently for every time step where a disturbance is present. The results are shown in Figure 2.7. Note that robustness results such as Theorem 2 and Lemma 4 must consider the worst case scenario. Thus, there are trajectories where the robustness criteria are not able to guarantee convergence, although they converge to the target state.

2.5.2 Numerical Quadrature Uncertainty Propagation

The key to stability analysis for closed-loop systems with dynamics given as full GP distribution is the prediction at uncertain inputs. We compare the performance of the presented approximate inference to the state-of-the-art moment matching (MM) method and Monte Carlo (MC). Consider the following scenario: in the mountain car domain, we position the car on the right slope with some positive velocity. Furthermore, we introduce small Gaussian uncertainty about the starting state. We employ a constant control signal, that is too small to bring the car up directly. We compute rollouts, propagating state distributions through the GP with (i) the presented numerical quadrature (NQ), (ii) MM as in (Deisenroth, 2010), and (iii) MC. The resulting distributions for a time horizon of 3s are shown in Figure 2.8. The MM approximation differs significantly from MC

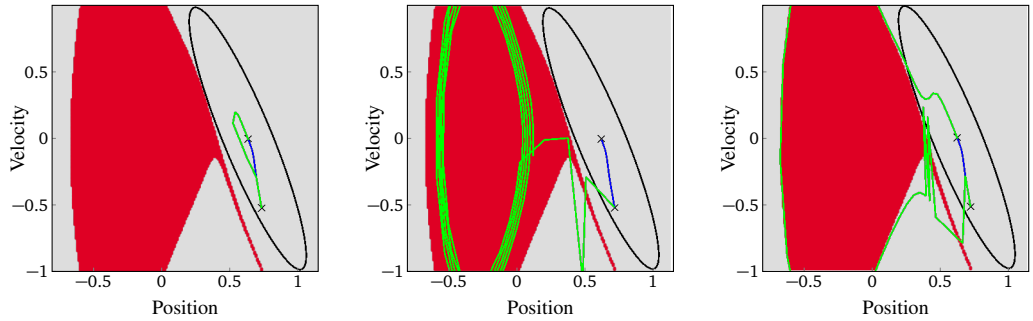


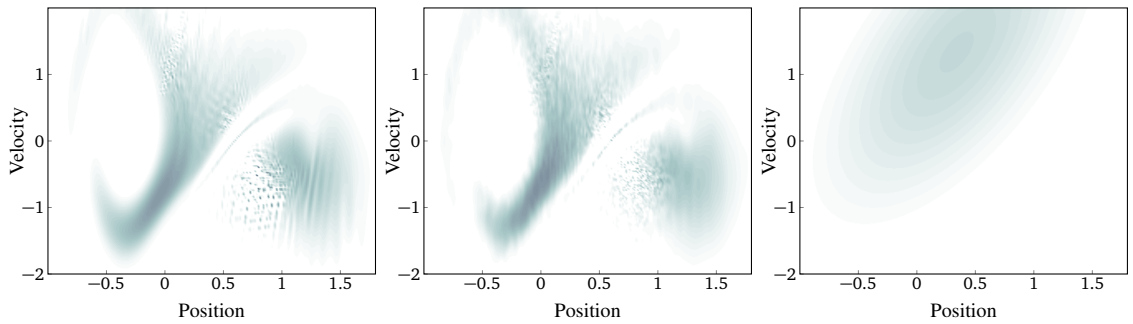
Figure 2.7.: Robustness of GP mean dynamics. Disturbances were sampled for a finite time period. The undisturbed trajectory is colored blue, the distrubed trajectories are colored green. Our robustness criterion from Theorem 2 and Lemma 4 guarantees the convergence of the disturbed trajectory in the left picture. For both other pictures, our criterion cannot guarantee convergence of the disturbed trajectory to the target. As such criteria must be based on worst case estimates of the disturbance, some disturbed trajectories still converge to the target (as in the right picture).

and NQ results, concentrating most of the probability mass, where the MC approximation has very low probability density. The NQ result closely matches the distribution obtained with MC.

2.5.3 Stability of Gaussian Process Dynamics

Employing numerical quadrature, we determine success probabilities for the two test-beds and a time horizon of 100s. To obtain the stability region for a particular choice of λ, all pixels whose color corresponds to a value higher then or equal to λ must be selected. As Figure 2.9 shows, the obtained stability regions match the empirical MC results. The error bound from Lemma 5 can demand for extremely fine quadrature rules to obtain a stability region estimate that matches the true stability region closely. Please note that quadrature rules with less nodes also allow for a guaranteed stability region, however this region can be significantly smaller than the true stability region. For long time horizons, the requirements to obtain a very accurate inner approximation of the stability region can become computationally infeasible. However, we found that the real-world results are substantially better than this worst-case bound. We also experienced computation time ($\approx$ 120s) for NQ to be a fraction of the time required for long time MC predictions. Of course, this will not hold for systems with many state dimensions and our particular setup of product quadrature rules, as these rules suffer from the curse of dimensionality. However, there are various approaches to overcome this drawback of NQ (Heiss and Winschel, 2008; Novak and Ritter, 1996; Xiao and Gimbutas, 2010; Ryu and Boyd, 2015) and our analysis holds for arbitrary quadrature rules.

To show how the proposed approach can be scaled to higher dimensions, we compute success probabilities for the four-dimensional cart-pole system. We employ Algorithm 3 to construct a suitable quadrature rule. However, the quadrature rule CN:3-1 (Stroud, 1971; code from Burkardt, 2014) is applied to each subregion instead of the Gaussian product rule as for the previous examples. This quadrature rule has 8 nodes as opposed to 16 nodes for the Gaussian product rule of the same exactness. Please note also that all necessary computations for the proposed approach can be



(a) Our Numerical Quadrature **(b)** Monte Carlo Sampling **(c)** Moment Matching

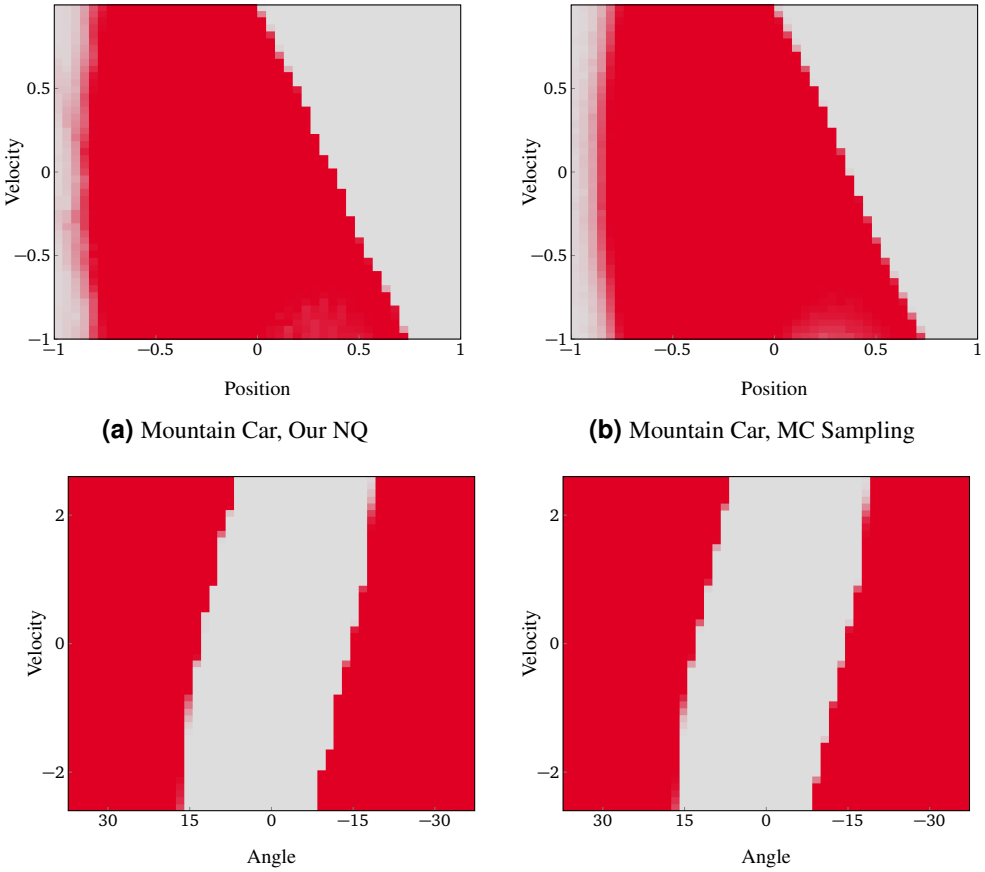
Figure 2.8.: This figure illustrates multi-step-ahead prediction when the input is a distribution. Starting with a normally distributed state centered around the inflection point of the right slope in the mountain car domain, we compute a rollout incorporating uncertainty. The plots show the state distribution after T=30 time steps obtained with our numerical quadrature approach (a), moment matching (c) and the reference Monte Carlo sampling result (b).

executed in parallel. Thus, we conduct these computations on a GPU, which leads to a significant speedup and overall computation time comparable to the 2D examples ($\approx$ 140s). The result is shown in Figure 2.10. As for the other benchmark problems, the obtained stability region closely matches the true stability region of the system.

To evaluate the results on convergence of GP dynamics to a stationary distribution, we considered the dynamics shown in the left plot of Figure 2.11. The GP mean dynamics has two attractors and their regions of attraction partition the interval $[-1; 1]$. However, when considering the uncertainty, the system has an invariant probability measure (shown in Figure 2.11 on the right) to which it converges irrespective of the starting point.

2.6 Conclusion

Gaussian Processes provide a flexible, non-parametric, Bayesian approach to modeling unknown system dynamics from observations. Thus, GP dynamics are a viable approach for learning model-based control. There has been remarkable research in this field that advanced these methods to become an appealing alternative to classic control. For widespread real-world application of learning control based on GP forward dynamics models, performance guarantees are crucial, especially in safety-critical applications. However, stability analysis of closed-loop control with GP forward dynamics learned from data has barely been analyzed so far. In this chapter, we laid a foundation for stability analysis of closed-loop control with GP dynamics. Subsequently, we conclude the chapter with a short summary of the main contributions and a brief outlook on possible future work in this direction.



(a) Mountain Car, Our NQ

(b) Mountain Car, MC Sampling

(c) Pendulum, Our NQ

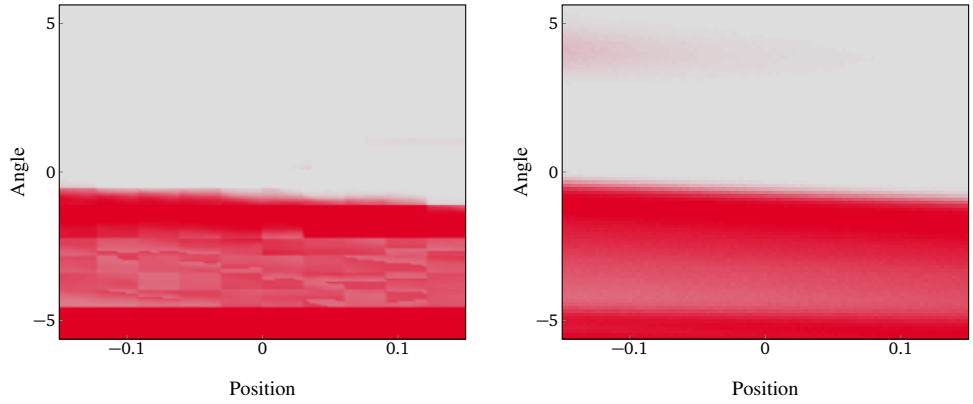
(d) Pendulum, MC Sampling

Figure 2.9.: Success probabilities for dynamics given as GP full distribution and a time horizon of 100s. The color scale encodes the success probability from zero (red) to one (gray). The mountain car ((a) and (b)) and inverted pendulum ((c) and (d)) results were obtained with the proposed numerical quadrature (NQ) and Monte Carlo (MC).

2.6.1 Summary of Contributions

In this chapter, we analyzed stability of closed-loop control systems with Gaussian process forward model dynamics. We considered two possible types of system dynamics: (i) the mean and (ii) the full GP predictive distribution.

In the first case, we studied asymptotic stability as well as finite time horizons and robustness to disturbances. We presented an algorithm to construct a region in the state space such that trajectories starting inside this region are guaranteed to converge to the target point and stay there for all times. For finite time horizons, we showed how to find a state space region such that the target state will be reached at time horizon up to a certain tolerance. Studying robustness, we derived a criterion for disturbances such that the system remains asymptotically stable. The



(a) Cart-Pole, Our NQ

(b) Cart-Pole, MC Sampling

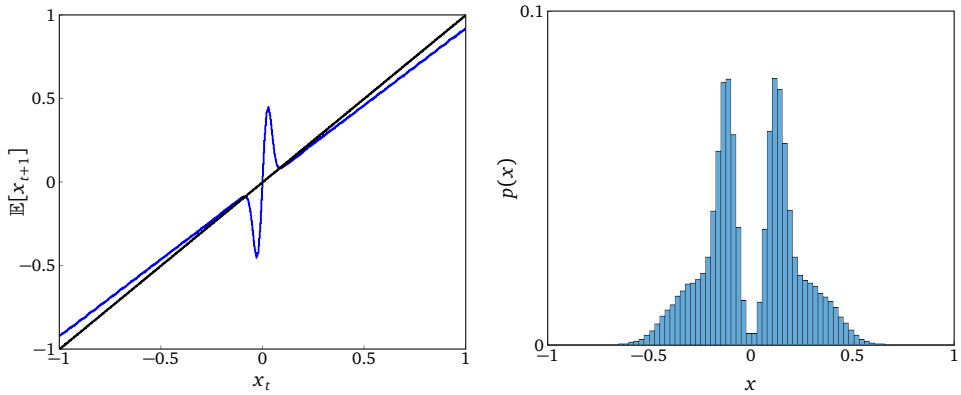
Figure 2.10.: Success probabilities for the cart-pole system. The dynamics are given as GP full distribution and the time horizon is 100s. The color scale encodes the success probability from zero (red) to one (gray). The plots show the results for $\dot{x} = 0, \dot{\theta} = 0$ obtained with numerical quadrature (a) and Monte Carlo sampling (b).

theoretical results have been evaluated on benchmark problems and compared to empirically obtained results.

In the second case, we introduced a novel approach based on numerical quadrature to approximately propagate uncertainties through a GP. In contrast to other state-of-the-art methods, our approach can model complex distributions with multiple modes. Evaluation results closely match the true distribution approximated by extensive sampling. We used the introduced approximate inference method to derive finite-time stability guarantees based on quadrature error analysis. Empirical Monte Carlo results confirm our theoretical results on the two benchmark problems. Furthermore, we considered the system behavior when the time horizon is infinite. We showed that, applying numerical quadrature to propagate distributions through the GP, the system state converges to a limiting distribution that does not depend on the starting state. Motivated by this result, we studied the system behavior for infinite time horizons without applying any approximate inference steps. We succeeded to show that closed-loop control systems with dynamics given as full GP distribution converge to a unique and invariant limiting distribution that does not depend on the starting state for many choices of the covariance function. Overall, the proposed methods provide stability guarantees for many existing learning control approaches based on GPs.

2.6.2 Discussion and Next Steps

While the proposed methods apply to many interesting examples of learning control in the literature, the analysis could be extended in several directions. Firstly, for closed-loop control with GP dynamics, we considered GPs with stationary covariance functions and zero mean prior. To include other choices of the prior, e.g., nonstationary covariance functions and unbounded mean priors, novel criteria to ensure Harris recurrence must be found. Another interesting question is how



(a) Forward Dynamics Function **(b)** Invariant Limiting Distribution

Figure 2.11.: Asymptotic behavior of closed-loop systems with GP dynamics. Plot (a) shows the mean dynamics function with the mean $m(x)$ of the GP prediction at the query point x on the y axis. When considering the mean dynamics, there are two fixpoints $x_{(1)} = -0.1$ and $x_{(2)} = 0.1$. For any starting point $-1 < x < 0$, the mean trajectory converges to $x_{(1)}$ and analogously for all $0 < x < 1$ the trajectories converge to $x_{(2)}$. Now we consider the full GP dynamics and take the uncertainty into account. Plot (b) shows a histogram of the probability density after 10000 time steps when starting at $x = -0.1$, obtained with Monte Carlo. This result matches the histogram obtained when starting at $x = 0.1$ or at any other point. Thus, the system converges to a unique, invariant limiting distribution.

the stability of GP dynamics is affected by disturbances in the input space. While some results could straightforwardly be followed for dynamics given as the mean of a GP, the task seems a lot more challenging when uncertainty is included. Many of the subtle, intricate arguments applied to show convergence of the system to a limiting distribution must be carefully reconsidered when disturbances are present.

Secondly, the obtained results are stability results for closed-loop control systems with dynamics given as a GP. The GP training data have a huge impact on the quality of the GP dynamics model when compared to the physical system the data was collected from. In this direction, criteria to evaluate the quality of a learned model would further support the application of learning control for real-world problems. Towards this goal, PAC Bayesian bounds for the approximation error could be employed to obtain a stability statement for the true dynamics without assuming correctness of the model.

Another interesting research direction is encouraged by the presented result on asymptotic stability of closed-loop control systems with GP dynamics. In this chapter we proved the existence of a limiting distribution the system will converge to. This distribution is unique and invariant. However, little more is known about the limiting distribution. Further research in this direction could help for a better understanding of the behavior of learned dynamics and also be directly

employed in learning control – possibly by optimizing the limiting distribution itself instead of local approximations as, e.g., system trajectories.

3 Numerical Quadrature for Probabilistic Policy Search

Control of technical systems is a core task in engineering and has a long tradition, with the first formal analysis dating back to James Clerk Maxwell who analyzed control of the steam engine. In classical control, dynamics models are derived from physics' first principles and controllers are then obtained analytically. If expert knowledge is difficult to obtain or simply not available, classical control approaches cannot be applied. Reinforcement learning solves sequential decision problems and is, thus, closely related to optimal control. However, RL has traditionally been separate from the field of classical control and takes a different approach to obtain a controller. Instead of analytically deriving a model and control structure, RL learns a control policy from interactions with the system. Thus, RL presents a feasible alternative to manual development of task specific solutions based on expert knowledge. A major advantage of learning control is that it can substantially decrease manual effort required to obtain a task solution.

In real world scenarios system interactions are often costly and, thus, the number of system interactions required to learn the controller should be minimized. Model-based RL, especially when combined with a Gaussian process dynamics model, has been tremendously successful in learning control while minimizing manual effort and system interactions. In policy search, the policy is obtained by maximizing the expected long-term reward. To evaluate this expected long-term reward, the state evolution must be computed. We have seen in Chapter 2 that the state distribution is analytically intractable for a Gaussian process forward model. As this propagation step is at the core of policy search, the employed approximation technique significantly affects policy learning. In Chapter 2 we introduced a highly accurate approximation method for multi-step-ahead predictions in GP dynamics models. In this chapter, we employ this approximation method in policy search based on GP dynamics models and show how it can further increase data efficiency.

3.1 Introduction

Learning control has become a viable approach in both the machine learning and control community. Many successful applications impressively demonstrate the advantages of learning control (Deisenroth et al., 2015; Pan and Theodorou, 2014; Klenske et al., 2013; Maciejowski and Yang, 2013; Nguyen-Tuong and Peters, 2011; Engel et al., 2006; Kocijan et al., 2004). In contrast to classical control methods, learning control does not presuppose a detailed understanding of the underlying dynamics but tries to infer the required information from data. Thus, relatively little expert knowledge about the system dynamics is required and fewer assumptions, such as a parametric form and parameter estimates, must be made.

For real-world applications of learning control, it is desirable to minimize the system interaction time. Thus, approaches that explicitly learn a dynamics model are often preferred, as model-free methods can require a prohibitive amount of system interactions (Atkeson and Santamaria, 1997;

Kuvayev, 1997; Moore and Atkeson, 1993; Sutton, 1990). However, one drawback of model-based methods is that modeling errors can derail learning, as the inherently approximate and frequently highly erroneous model is implicitly assumed to approximate the real dynamics sufficiently well (Schneider, 1996). Thus, solutions to the approximate control problem might result in policies that do not solve the control task for the true system dynamics. This model bias can have severe consequences especially when few data is available and, thus, the model predictions are highly uncertain. Hence, employing Gaussian processes (GPs) as forward models for learning control is particularly appealing as they incorporate uncertainty about the system dynamics estimate. GPs infer a distribution over all plausible models given the observed data instead of compromising on an approximate model and, thus, avoid severe modeling errors. Another major advantage of Gaussian process forward models is that stability analyses for such closed-loop control systems are available (Beckers and Hirche, 2016; Vinogradska et al., 2016), including automatic tools (Vinogradska et al., 2016) which do not require any expert knowledge.

One difficulty of learning control based on a GP forward model is that the state distribution when applying a policy for several discrete time steps is analytically intractable. Thus, to evaluate a policy, approximations for multi-step-ahead predictions are required. Common choices include Monte Carlo sampling (Girard et al., 2002), linearization of the posterior mean (Ko and Fox, 2008) and moment matching (Quiñonero-Candela et al., 2003). While Monte Carlo methods suffer from slow convergence (Ghavamzadeh and Engel, 2007) and introduce the need for small learning rates due to noisy numerical gradients, the deterministic linearization and moment matching methods provide analytical gradients of the multi-step-ahead predictions. However, such methods approximate the state distribution as a Gaussian. As a consequence their expressivity is limited, e.g., moment matching and linearization cannot handle state distributions with multiple modes. Thus, such approximations suffer from severe inaccuracies, especially when the state distribution differs significantly from a Gaussian.

Another major drawback of Gaussian approximations for multi-step-ahead predictions is that learning is limited to optimizing locally around one trajectory. This problem arises from the limited expressivity of Gaussians. When a distribution with high variance is propagated through nonlinear dynamics, the predictive distribution is typically highly complex and cannot be represented sufficiently well by a Gaussian. Thus, approximating predictions at uncertain inputs by a Gaussian is feasible only for state distributions with low variance. As a workaround (Deisenroth et al., 2014), multiple starting points that are representative of all starting states must be hand-selected to learn a suitable policy. The trajectories starting at the selected points must then be optimized simultaneously. As a result, the learned policy is highly dependent on the chosen starting points, suffers from bad generalization between those points and optimization is prone to local optima.

An alternative method to approximate multi-step-ahead predictions (Vinogradska et al., 2016) employs numerical quadrature and provides analytical expressions for the state distribution. It approximates even complex distributions with multiple modes accurately and, additionally, bounds for the approximation error can be obtained. A comparison of numerical quadrature and moment matching for long-term predictions is shown in Figure 3.2.

In this chapter, we propose nuQuPS, a model-based policy search method that employs Gaussian processes as forward dynamics model and relies on numerical quadrature (Vinogradska et al., 2016) to approximate multi-step-ahead predictions. The use of numerical quadrature increases data-efficiency and improves the quality of the learned policies significantly. Furthermore, numerical

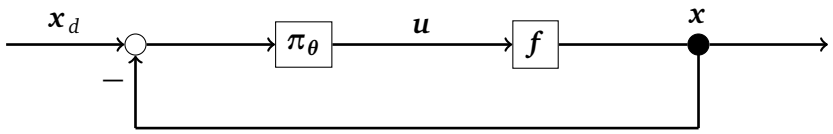


Figure 3.1.: A closed-loop control structure with controller π_θ, system dynamics f and target state x_d. We model the system dynamics f as a Gaussian process and assume that the policy π_θ is parameterized by θ. The proposed algorithm learns f and θ from interactions with the real system.

quadrature enables the use of arbitrary distributions for the start state. Thus, it is possible to learn a policy, e.g., for all starting points in a certain region by choosing a uniform starting state distribution. Overall, the proposed algorithm is capable of learning global policies with unprecedented data-efficiency, while no manual effort or expert knowledge are required.

The chapter will be organized as follows: first, we specify the considered problem. In Section 3.1.2, we briefly review related work. Section 3.2 introduces the proposed algorithm, which is evaluated on multiple benchmark tasks in Section 3.4. A conclusion summarizes and discusses the provided results (Section 3.5).

3.1.1 Problem Statement

In this chapter, we aim to learn to control a previously unknown dynamics system. We consider discrete-time dynamics

$$x_{t+1} = f(x_t, u_t) + \varepsilon \tag{3.1}$$

with $x_t, x_{t+1} \in \mathbb{R}^D$, $u_t \in \mathbb{R}^F$, unknown f and i.i.d. Gaussian measurement noise $\varepsilon \in \mathcal{N}(0, \Sigma_\varepsilon)$. Figure 3.1 shows such a closed-loop control setting. Given a reward function $r : x_t \mapsto r(x_t) \in \mathbb{R}_+$, the goal is to find a policy $\pi : x \mapsto \pi(x)$ that maximizes the expected reward up to time horizon T, when choosing $u_t := \pi(x_t)$ for $t = 1, \ldots, T$. In policy search, we assume that the policy is parameterized by the parameter vector θ and write $\pi_\theta(x_t) := \pi(\theta, x)$. The objective is then to find a parameter vector θ^* that maximizes the expected long-term reward

$$R_\pi(\theta) = \sum_{t=0}^{T} \mathbb{E}_{x_t}[r(x_t)]. \tag{3.2}$$

We rely on Gaussian processes to model the system dynamics f from observations of the system behavior, as will be detailed in Section 3.2.1. To compute the expected long-term reward (3.2) for a particular policy, the state distributions $p(x_1), \ldots, p(x_T)$ must be determined. As with most nonlinear dynamics models, for our GP forward dynamics prediction of the next state becomes intractable when the input is a distribution. Thus, multi-step-ahead predictions must be approximated.

In our setting, it is desirable to learn a policy that is suitable not only for one starting state, but can control the system (3.1) reliably for a continuous region of starting states. More precisely,

we aim to learn policies suitable for a given a starting state distribution $p(\boldsymbol{x}_0)$. We assume that $p(\boldsymbol{x}_0)$ is piecewise differentiable, but make no other assumptions e.g., that $p(\boldsymbol{x}_0)$ is Gaussian or has low variance. Thus, to approximate $p(\boldsymbol{x}_1), \ldots, p(\boldsymbol{x}_T)$ we propose the use of high performance numerical quadrature as in (Vinogradska et al., 2016), see Section 3.2.4. To maximize the expected long-term reward (3.2), we perform gradient ascent. In Section 3.2.5 we will derive analytical expressions for $\partial R_\pi / \partial \boldsymbol{\theta}$.

3.1.2 Related Work

The problem of deriving control laws when uncertainty is present has been considered in classic control theory for many years. In controller design, uncertainty may arise from modeling inaccuracies, the presence of (external) disturbances and the lack of data, as some system parameters are not known in advance but only during operation. Thus, a controller must be designed for a family of systems that is specified, e.g., by bounds for model parameters or for nonparametric uncertainties as bounds for the operator norm of the unknown dynamics. Robust control (Skogestad and Postlethwaite, 2005; Zhou and Doyle, 1998) designs a single controller that is provably stable for all systems in the specified family – often at cost of overall controller performance. Adaptive control (Narendra and Annaswamy, 2012; Tao, 2003) instead adjusts control parameters online in order to achieve prespecified performance, which can be computationally demanding. These methods rely on parametric dynamics models, which must be specified by an expert for each problem. In addition, both schemes require stability analysis of the dynamics system, e.g., via manually designed Lyapunov functions, which can be extremely challenging for complex, nonlinear systems.

Nonparametric system dynamics models are very appealing due to their high flexibility. They learn a dynamics model from data instead of relying on expert knowledge to pick a sufficiently accurate parametric form suitable for the dynamics. Nonparametric regression methods that learn the system dynamics from data have been considered in e.g., (Deisenroth et al., 2015; Kocijan et al., 2004; Kupcsik et al., 2013; Pan and Theodorou, 2014; Schneider, 1996). In (Schneider, 1996), locally weighted Bayesian regression has been employed to model the system dynamics and uncertainty was treated as noise. To learn a policy, stochastic dynamic programming was applied on the discretized state space. The approaches (Deisenroth et al., 2015; Kocijan et al., 2004; Kupcsik et al., 2013; Pan and Theodorou, 2014) model the forward dynamics as a Gaussian process.

Gaussian processes as forward models allow to incorporate uncertainty about the system dynamics without the need to discretize the state space. GPs have also been employed to model the system dynamics in (Deisenroth et al., 2015; Rasmussen and Kuss, 2004; Rottmann and Burgard, 2009; Deisenroth et al., 2009; Bischoff et al., 2013; Kupcsik et al., 2013). The approaches (Rasmussen and Kuss, 2004; Rottmann and Burgard, 2009; Deisenroth et al., 2009; Bischoff et al., 2013) learn global value functions that are subsequently used to derive policies. For example, the PVI algorithm (Bischoff et al., 2013) learns the system dynamics and the value function, which are both modeled as GPs, in an episodic setting. While these value iteration based approaches provide great flexibility as no assumptions on the policy are made, maintaining a model for the global value function can be computationally demanding in large state spaces.

In contrast to such GP based fitted value iteration methods, PILCO (Deisenroth et al., 2015) and GPREPS (Kupcsik et al., 2013) are policy search approaches that rely on GPs as forward dynamics

models. The PILCO algorithm (Deisenroth et al., 2015) employs GPs as forward dynamics models and conducts a search in the policy space, performing gradient ascent on the expected reward in an episodic setting. PILCO is particularly appealing, as analytical gradients of the expected reward with respect to the policy parameters are available. These analytical gradients enable scaling to high-dimensional problems and allow for highly flexible policies with many parameters.

The GP based value iteration approaches as well as the policy search PILCO and GPREPS approaches benefit from Bayesian averaging over all plausible models by incorporating the uncertainty provided by the GP dynamics models. However, propagating uncertainty through a Gaussian process is analytically intractable even for simple, Gaussian input distributions. All of the methods mentioned above rely on the moment matching approximation (Quiñonero-Candela et al., 2003) or on sampling to propagate distributions through a GP. This moment matching approximation is only applicable to Gaussian input distributions and provides a good estimate of the next state distribution only if the variance of the input distribution is low. Unfortunately, these requirements typically do not hold during learning.

3.2 Numerical Quadrature Based Policy Search

We introduce nuQuPS, a model-based policy search approach with GPs as dynamics models and numerical quadrature for long-term predictions. First, we briefly recap Gaussian process regression which will be employed to model the system dynamics. Section 3.2.2 provides an overview of the nuQuPS algorithm. The choice of reward function and policy parametrization are discussed in Section 3.2.3. Subsequently, we elaborate on the proposed approximation for long-term predictions based on numerical quadrature. Finally, we compute analytic expressions for the gradients of the expected long-term reward with respect to the policy parameters $\boldsymbol{\theta}$ when numerical quadrature is used to propagate uncertainties.

3.2.1 Gaussian Process Regression

In the following, we will briefly recap Gaussian process regression as it will be used to learn the system dynamics from observed data.

Given noisy observations $\mathscr{D} = \{(\boldsymbol{z}^i, y^i = f(\boldsymbol{z}^i) + \varepsilon^i) \mid 1 \leq i \leq N\}$, where $\varepsilon^i \sim \mathcal{N}(0, \sigma_n^2)$, the prior on the values of f is $\mathcal{N}(0, K(Z,Z) + \sigma_n^2 I)$. The covariance matrix $K(Z,Z)$ is defined by the choice of covariance function k as $[K(Z,Z)]_{ij} = k(\boldsymbol{z}^i, \boldsymbol{z}^j)$. In this paper, we employ the squared exponential covariance function

$$k(\boldsymbol{z}, \boldsymbol{w}) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{w})^\intercal \Lambda^{-1} (\boldsymbol{z} - \boldsymbol{w})\right),$$

with signal variance σ_f^2 and squared lengthscales $\Lambda = \operatorname{diag}(l_1^2, \ldots, l_{D+F}^2)$ for all input dimensions. Given a query point $\boldsymbol{z}_*$, the conditional probability of $f(\boldsymbol{z}_*)$ is

$$f(\boldsymbol{z}_*) \mid \mathscr{D} \sim \mathcal{N}\big(\boldsymbol{k}(\boldsymbol{z}_*, Z)\boldsymbol{\beta},\ k(\boldsymbol{z}_*, \boldsymbol{z}_*) - \boldsymbol{k}(\boldsymbol{z}_*, Z)(K(Z,Z) + \sigma_n^2 I)^{-1} \boldsymbol{k}(Z, \boldsymbol{z}_*)\big) \tag{3.3}$$

with $\boldsymbol{\beta} = (K(Z,Z) + \sigma_n^2 I)^{-1} \boldsymbol{y}$. The hyperparameters, e.g., $\sigma_n^2, \sigma_f^2, \Lambda$ for the squared exponential kernel, are estimated by maximizing the log marginal likelihood of the data (Rasmussen and Williams, 2005).

Algorithm 4 Numerical Quadrature based Probabilistic Policy Search (nuQuPS)

Input: start distribution $p(\boldsymbol{x}_0)$, time horizon T, reward function r
Output: policy $\pi_{\boldsymbol{\theta}^*}$ that maximizes $R_\pi(\boldsymbol{\theta})$ (see Eq. (3.2))

```
sample initial parameters θ ~ N(0, I)
D ← ∅
while not converged do
    Sample x̃_0 ~ p(x_0)
    Compute rollout x̃_0, ..., x̃_T from x̃_0 employing π_θ
    D ← D ∪ {(x̃_0, π_θ(x̃_0)), x̃_1, ), ..., (x̃_{T-1}, π_θ(x̃_{T-1})), x̃_T, )}, train GP g on D
    Construct quadrature rule suited for g with Algorithm 5
    Compute p(x_1), ..., p(x_T) approximately with numerical quadrature (see Sec. 3.2.4)
    R_π(θ) ← Σ_{t=0}^{T} E_{x_t}[r(x_t)]
    θ* ← argmax_θ R_π(θ) via gradient ascent, ∂R_π(θ)/∂θ as in Sec. 3.2.5
    θ ← θ*
end while
return π_{θ*}
```

In this thesis, we employ a Gaussian process $\boldsymbol{g}$ to model system dynamics. It takes state-action pairs $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{u})^\intercal$ and outputs differences to successor states, i.e., $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{u}_t)$. As these outputs are multivariate, we train conditionally independent GPs for each output dimension. We write $\sigma^2_{n,m}$, $\sigma^2_{f,m}, \Lambda_m$ for the GP hyperparameters in output dimension m and k_m for the corresponding covariance function.

Please note that learning a GP as forward dynamics model from observations in this setting introduces a bias to the model as the GP models measurement noise on the outputs but the training inputs are assumed to be noise free. One approach to handle input noise is to employ heteroscedastic GPs (Kersting et al., 2007) and treat the input noise as non-constant noise on the output. In (McHutchon and Rasmussen, 2011) the specific heteroscedastic noise structure introduced by noise on the inputs is explicitly taken into account. This noise structure leads to an additional term in the predictive mean and variance of the GP. As these additional terms are differentiable with respect to the policy parameters, this noisy input GP can be used to model the system dynamics instead of a standard GP model. For the sake of simplicity, we adhere to a simple GP model although the following derivations can be generalized to noisy input GPs.

3.2.2 Learning Policies from Scratch: nuQuPS

The proposed algorithm proceeds in an episodic setting. In the beginning, $\boldsymbol{\theta}$ is initialized randomly. The currently known best policy $\pi_{\boldsymbol{\theta}}$ is employed on the real system to get new information about the system dynamics. A starting point is sampled from the starting state distribution $p(\boldsymbol{x}_0)$ and a state-action trajectory $(\widetilde{\boldsymbol{x}}_0, \pi_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_0)), \ldots, (\widetilde{\boldsymbol{x}}_{T-1}, \pi_{\boldsymbol{\theta}}(\widetilde{\boldsymbol{x}}_{T-1})), \widetilde{\boldsymbol{x}}_T$ is observed. This rollout data is used to update the GP dynamics model $\boldsymbol{g}$. With the updated dynamics model, the current policy is evaluated according to Eq. (3.2). For this policy evaluation, GP predictions at the uncertain inputs $p(\boldsymbol{x}_0), \ldots, p(\boldsymbol{x}_{T-1})$ must be computed. We employ numerical quadrature as described in Section 3.2.4 to propagate uncertainties through the GP. To ensure highly accurate

approximate predictions and maintain computational efficiency in high dimensional state spaces, we tailor the quadrature rule to the GP dynamics $\boldsymbol{g}$ as described in Section 3.2.6 and Algorithm 5. To improve the policy $\pi_{\boldsymbol{\theta}}$, the parameters $\boldsymbol{\theta}$ are updated to maximize $R_\pi(\boldsymbol{\theta})$ via gradient ascent. Closed-form expressions for $\partial R_\pi / \partial \boldsymbol{\theta}$ are given in Section 3.2.5. The current parameters $\boldsymbol{\theta}$ are set to the obtained optimal parameters $\boldsymbol{\theta}^*$, which can then be used for a rollout in the next episode. Algorithm 4 summarizes this approach.

3.2.3 Choice of Policy Parametrization and Reward Function

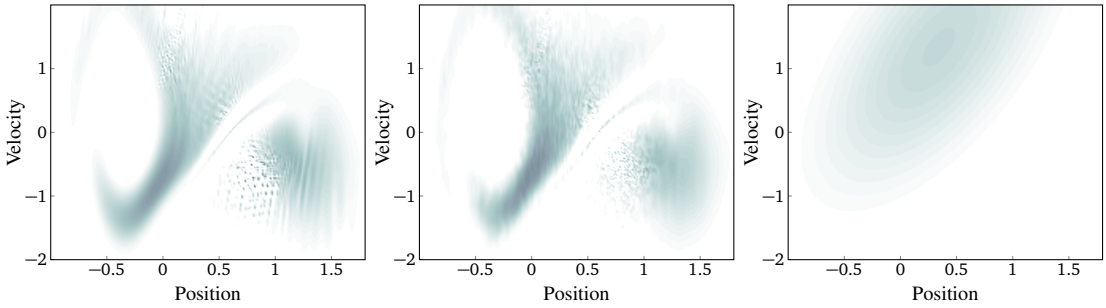
In the following, we will elaborate on the possible choices of reward function and parametric form for the policy. In both cases, the restrictions result from our goal to provide analytical gradients to speed up policy search and make it scalable to high-dimensional problems. Fortunately, these restrictions allow for fairly flexible choices for both the policy and the reward function.

The reward function is used to determine the expected long-term cost of a policy. To evaluate a policy, the expected reward for all state distributions $p(\boldsymbol{x}_1), \dots, p(\boldsymbol{x}_T)$ when following this policy must be computed. Thus, the reward function r must be chosen such that $\mathbb{E}_{\boldsymbol{x}_t}[r(\boldsymbol{x}_t)]$ is analytically tractable for all $t = 0, \dots, T$. As we will see in Section 3.2.4, the state distribution is approximated by a Gaussian mixture model when applying numerical quadrature to propagate uncertainties. Due to linearity of the expectation, we conclude that r must be chosen such that expectations with respect to Gaussian distributions are analytically tractable. This holds, e.g., for all polynomials, Gaussians or mixtures of Gaussians. A straightforward choice for the reward function is to penalize the distance of the state to the target state $\boldsymbol{x}_d$, e.g.,

$$r(\boldsymbol{x}) = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_d)^\intercal \Sigma_r (\boldsymbol{x} - \boldsymbol{x}_d)\right), \tag{3.4}$$

where Σ_r defines width and orientation of the reward function.

For the policy, we assumed that it is parameterized by a parameter vector $\boldsymbol{\theta}$. To optimize the policy, we aim to perform a gradient ascent employing analytic gradients of the expected long-term reward with respect to $\boldsymbol{\theta}$. Thus, the parametric form of the policy must be chosen such that it allows for analytic gradient computation. Additionally, in many real-world applications, the magnitude of the control signal is limited, e.g., by a constant $\boldsymbol{u}_{\max} \geq \boldsymbol{\pi}(\boldsymbol{x}) \geq -\boldsymbol{u}_{\max}$. These constraints must be incorporated in the parametric form. In (Deisenroth et al., 2015), it was proposed to apply a squashing function such as, e.g., $\sigma(\boldsymbol{x}) = \sin(\boldsymbol{x})$ or $\sigma(\boldsymbol{x}) = 9/8 \sin(\boldsymbol{x}) + 1/8 \sin(3\boldsymbol{x})$. Also, analytic gradients of the GP prediction with respect to $\boldsymbol{\theta}$ were provided for (i) squashed linear policies $\pi(\boldsymbol{x}) = u_{\max}\sigma(M\boldsymbol{x} + b)$ and (ii) squashed RBF policies $\pi(\boldsymbol{x}) = \sigma(k(C, \boldsymbol{x})^\intercal \boldsymbol{\beta})$. Following (Deisenroth et al., 2015), we squash the policies to limit the magnitude. Applying numerical quadrature for multi-step-ahead predictions, we will derive analytic policy gradients based on the gradients given in (Deisenroth et al., 2015). Thus, nuQuPS can handle squashed linear and squashed RBF policies. Additionally, our numerical quadrature approach allows to use sums of squashed linear policies, which corresponds to a Fourier series expansion of an arbitrary, bounded function.



(a) Numerical Quadrature **(b)** Monte Carlo Sampling **(c)** Moment Matching

Figure 3.2.: This figure illustrates multi-step-ahead prediction when the input is a distribution. Starting with a normally distributed state centered around the inflection point of the right slope in the mountain car domain, we approximate the state distribution at $T = 30$. The plots show the approximate state distribution obtained with numerical quadrature (a), moment matching (c) and the reference Monte Carlo sampling result, computed with 10^5 samples (b). As can be seen, the state distribution significantly differs from a Gaussian. Furthermore, moment matching does not match the first two moments of the state distribution.

3.2.4 Numerical Quadrature for Uncertainty Propagation

To evaluate a policy, multi-step-ahead predictions for a given GP dynamics $\boldsymbol{g}$ must be computed, see Eq. (3.2). Given a query point $(\boldsymbol{x}_t, \boldsymbol{u}_t)$, the GP predicts the next state $\boldsymbol{x}_{t+1}$ to be normally distributed as stated in Eq. (3.3). When the input $\boldsymbol{x}_t$ is uncertain, the next state distribution is given as

$$p(\boldsymbol{x}_{t+1}) = \int p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t. \tag{3.5}$$

In general, this distribution is not Gaussian even if $p(\boldsymbol{x}_t)$ is. Furthermore, for most choices of covariance function, as e.g., for the widely employed squared exponential, $p(\boldsymbol{x}_{t+1})$ is analytically intractable and must be approximated.

When $p(\boldsymbol{x}_t)$ is Gaussian, the first two moments of $p(\boldsymbol{x}_{t+1})$ can be computed in closed-form, which gives rise to the moment matching approximation (Quiñonero-Candela et al., 2003). The predictive distribution $p(\boldsymbol{x}_{t+1})$ is then approximated by a Gaussian with the computed moments. However, this approximation has some major drawbacks. First, approximating the state distribution by a Gaussian can lead to severe inaccuracies when the true state distribution is more complex, e.g., has multiple modes. Unfortunately, this is often the case when the system dynamics is highly nonlinear. Second, the first two moments of the predictive distribution can only be computed analytically when the input distribution is Gaussian. Starting with a Gaussian state distribution $p(\boldsymbol{x}_0)$, the state distribution $p(\boldsymbol{x}_1)$ will be approximated with a Gaussian. To compute $p(\boldsymbol{x}_2)$, the approximation of $p(\boldsymbol{x}_1)$ will be used as the model input. However, as the true distribution $p(\boldsymbol{x}_1)$ is not Gaussian, the moment matching approximation will not capture the first two moments of $p(\boldsymbol{x}_2)$ correctly. As a result, when cascading multiple moment matching steps, the computed

moments do not match the true moments of the state distribution after only two steps. Typically, the computed moments will drift further away from the true moments with every time step. Figure 3.2 (cf. (Vinogradska et al., 2016)) illustrates the described problems, that can occur when applying moment matching for long-term predictions.

As an alternative to moment matching, in (Vinogradska et al., 2016) numerical quadrature was proposed to approximate the GP predictive distribution. Numerical quadrature approximates the value of an integral

$$\int_a^b f(\boldsymbol{x})d\boldsymbol{x} \approx \sum_{i=1}^{p} w_i f(\boldsymbol{\xi}_i)$$

given a finite number p of function evaluations. A widely used class of quadrature rules are interpolatory quadrature rules, which integrate all polynomials up to a certain degree exactly. In this chapter, we employ Gaussian quadrature rules, where the evaluation points $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_p$ are chosen to be the roots of certain polynomials from orthogonal polynomial families. They achieve the highest accuracy possible for univariate interpolatory formulæ (Süli and Mayers, 2003). For multivariate integrals, the quadrature problem is significantly harder. While many formulæ for the univariate case can straightforwardly be generalized to multivariate integrals, they often suffer from the curse of dimensionality. However, quadrature methods that scale better and are feasible for up to 20 dimensions have been developed. See (Skrainka and Judd, 2011) for an overview. In (Vinogradska et al., 2016), it has been shown that numerical quadrature accurately approximates the state distribution and can be computed efficiently, see also Figure 3.2. However, no policy learning was considered in (Vinogradska et al., 2016). We follow this approach and approximate the integral from Eq. (3.5) with numerical quadrature.

When learning to control a physical system, the state space is bounded in most cases. Thus, we assume $\boldsymbol{x} \in X = [a_1, b_1] \times \cdots \times [a_D, b_D]$. Given the state distribution $p(\boldsymbol{x}_t)$, the next state is determined by

$$F[p(\boldsymbol{x}_t)] := \int_X p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t. \tag{3.6}$$

We apply numerical quadrature to approximate this integral. We choose a composed Gaussian product quadrature rule, which will be detailed in Section 3.2.6. For now, it is sufficient to note that our quadrature rule provides a set of evaluation points $\mathbb{X}$ and positive weights w_n for all nodes $\boldsymbol{\xi}_n \in \mathbb{X}$. Integral (3.6) is then approximated by

$$F[p(\boldsymbol{x}_t)] \approx \sum_{\boldsymbol{\xi}_n \in \mathbb{X}} w_n p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t = \boldsymbol{\xi}_n) p(\boldsymbol{x}_t = \boldsymbol{\xi}_n), \tag{3.7}$$

which results in a weighted sum of Gaussian distributions. The approximate state distribution at time $t+1$ can be written

$$p(\boldsymbol{x}_{t+1}) \approx \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_{t+1} \tag{3.8}$$

with $\alpha_{t+1,n} := w_n p(\boldsymbol{x}_t = \boldsymbol{\xi}_n)$ and $\phi_n(\boldsymbol{x}) := p(\boldsymbol{x}_{t+1} = \boldsymbol{x} \mid \boldsymbol{x}_t = \boldsymbol{\xi}_n)$. Note that the Gaussian basis functions $\phi_n(\boldsymbol{x})$ do not change over time, so the state distribution at time t is represented

by the weight vector $\boldsymbol{\alpha}_t$. To propagate any distribution multiple steps through the GP, the basis functions ϕ_n must be calculated only once and the task reduces to sequential updates of the weight vector $\boldsymbol{\alpha}$. As $p(\boldsymbol{x}_t) \approx \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}_t$, the weight vector $\boldsymbol{\alpha}_{t+1}$ is given by

$$\boldsymbol{\alpha}_{t+1} = \operatorname{diag}(\boldsymbol{w})\Phi\boldsymbol{\alpha}_t = (\operatorname{diag}(\boldsymbol{w})\Phi)^t\boldsymbol{\alpha}_1 \tag{3.9}$$

with the matrix Φ, $\Phi_{ij} = \phi_j(\boldsymbol{\xi}_i)$ with $1 \leq i, j \leq n$, which contains the basis function values at all grid points. In practice, it is helpful to normalize the matrix $\operatorname{diag}(\boldsymbol{w})\Phi$ such that each column sums to 1. In this case, unit vectors will be mapped to unit vectors. This ensures that the approximate state distribution is in fact a probability density, i.e., integrates to 1. Note that the columns of Φ can be computed independently and that, in general, Φ is very sparse. Thus, the computation of Φ and the multiplication in Eq. 3.9 are very well suited for parallel computation, e.g., on a GPU.

To evaluate a policy, we compute

$$R_\pi(\boldsymbol{\theta}) = \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{x}_t}[r(\boldsymbol{x}_t)] \tag{3.2 revisited}$$

and due to linearity of the expectation the numerical quadrature approximation results in

$$R_\pi(\boldsymbol{\theta}) \approx \sum_{t=0}^{T}\sum_{n=1}^{N} \alpha_{t,n}\mathbb{E}_{\boldsymbol{x}\sim\phi_n}[r(\boldsymbol{x})] =: \widetilde{R}_\pi(\boldsymbol{\theta}). \tag{3.10}$$

Thus, the expected reward in any time step is composed of the rewards for the state to be distributed as one of the Gaussians ϕ_n. When choosing r, hence, one must ensure that expectations with respect to a normally distributed state are analytically tractable, as detailed in Section 3.2.3.

3.2.5 Analytic Reward Gradients

When approximating predictions at uncertain inputs with numerical quadrature, the expected long-term reward $\widetilde{R}_\pi(\boldsymbol{\theta})$ for following policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ can be computed as given in Eq. (3.10). To improve the policy, we aim to maximize $\widetilde{R}_\pi(\boldsymbol{\theta})$ via gradient ascent. In the following, we will derive a closed-form expression for the gradient $\partial\widetilde{R}_\pi(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$.

$$\frac{\partial\widetilde{R}_\pi(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} = \frac{\partial}{\partial\boldsymbol{\theta}}\sum_{t=0}^{T}\sum_{n=1}^{N}\alpha_{t,n}\mathbb{E}_{\boldsymbol{x}\sim\phi_n}[r(\boldsymbol{x})] \tag{3.11}$$

$$= \sum_{t=0}^{T}\sum_{n=1}^{N}\frac{\partial\alpha_{t,n}}{\partial\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{x}\sim\phi_n}[r(\boldsymbol{x})] + \alpha_{t,n}\frac{\partial}{\partial\boldsymbol{\theta}}\left(\mathbb{E}_{\boldsymbol{x}\sim\phi_n}[r(\boldsymbol{x})]\right) \tag{3.12}$$

The gradient $\partial/\partial\boldsymbol{\theta}\,\mathbb{E}_{\boldsymbol{x}\sim\phi_n}[r(\boldsymbol{x})]$ was given in (Deisenroth et al., 2015) for squashed linear and squashed RBF policies. For the first term, we compute $\partial\alpha_{t,n}/\partial\boldsymbol{\theta}$ as follows. Applying Eq. (3.9), we get

$$\frac{\partial\boldsymbol{\alpha}_t}{\partial\boldsymbol{\theta}} = \frac{\partial}{\partial\boldsymbol{\theta}}\left((\operatorname{diag}(\boldsymbol{w})\Phi)^t\boldsymbol{\alpha}_1\right). \tag{3.13}$$

The derivative of a matrix power A^k with respect to the parameter b can be computed as $\partial A^k(b)/\partial b = \sum_{l=0}^{k-1} A^l(b)\frac{\partial A(b)}{\partial b}A^{k-l-1}(b)$. Note also that $\partial \boldsymbol{\alpha}_1/\partial \boldsymbol{\theta} = 0$ and, thus,

$$\frac{\partial \boldsymbol{\alpha}_t}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}}\left((\mathrm{diag}(\boldsymbol{w})\Phi)^t \boldsymbol{\alpha}_1\right) = \frac{\partial (\mathrm{diag}(\boldsymbol{w})\Phi)^t}{\partial \boldsymbol{\theta}}\boldsymbol{\alpha}_1 \tag{3.14}$$

$$= \sum_{l=0}^{t-1}(\mathrm{diag}(\boldsymbol{w})\Phi)^l \frac{\partial (\mathrm{diag}(\boldsymbol{w})\Phi)}{\partial \boldsymbol{\theta}}(\mathrm{diag}(\boldsymbol{w})\Phi)^{t-l-1}\boldsymbol{\alpha}_1. \tag{3.15}$$

Finally, it remains to compute the derivatives of the entries of $\mathrm{diag}(\boldsymbol{w})\Phi$ with respect to $\boldsymbol{\theta}$. As $\boldsymbol{w}$ are the quadrature weights, which do not depend on $\boldsymbol{\theta}$, we compute the derivatives of Φ with respect to $\boldsymbol{\theta}$

$$\frac{\partial \Phi_{ij}}{\partial \boldsymbol{\theta}} = \frac{\partial \phi_j(\xi_i)}{\partial \boldsymbol{\theta}} = \frac{\partial p(\boldsymbol{x}_{t+1} = \xi_j \mid \boldsymbol{x}_t = \xi_i)}{\partial \boldsymbol{\theta}}. \tag{3.16}$$

Let $\boldsymbol{\mu}_j$ be the mean of ϕ_j and Σ_j its covariance matrix. Note that Σ_j is diagonal, as ϕ_j is the GP predictive distribution at the point ξ_j and, thus, the different state dimensions do not covary.

The derivatives $\partial \boldsymbol{\mu}_j/\partial \boldsymbol{\theta}$ and $\partial \Sigma_j/\partial \boldsymbol{\theta}$ were provided in (Deisenroth et al., 2015) for squashed linear and squashed RBF policies. Thus, it remains to compute $\partial \phi_j(\xi_i)/\partial \boldsymbol{\mu}_j$ and $\partial \phi_j(\xi_i)/\partial \Sigma_j$ for diagonal Σ_j.

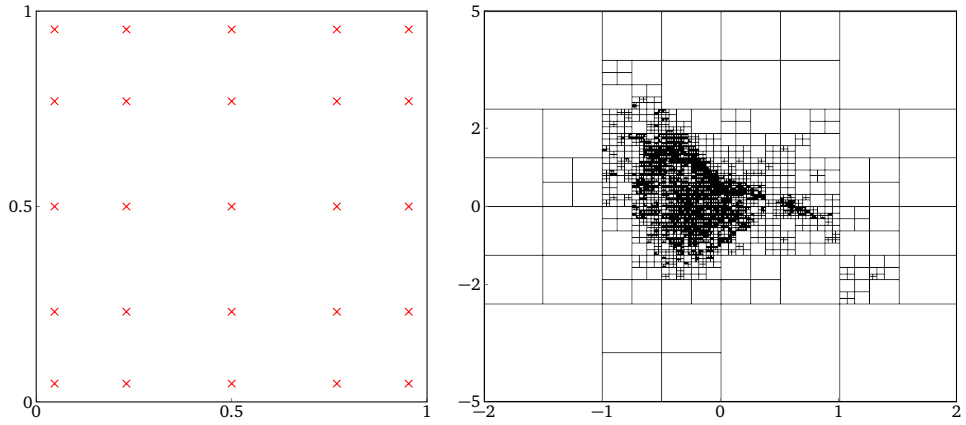
$$\frac{\partial \phi_j(\xi_i)}{\partial \boldsymbol{\mu}_j} = \frac{\partial}{\partial \boldsymbol{\mu}_j}\left((2\pi)^D \det(\Sigma_j)\right)^{-\frac{1}{2}} \exp\left((\boldsymbol{\mu}_j - \xi_i)^\intercal \Sigma_j^{-1}(\boldsymbol{\mu}_j - \xi_i)\right) \tag{3.17}$$

$$= -\phi_j(\xi_i)\Sigma_j^{-1}(\boldsymbol{\mu}_j - \xi_i) \tag{3.18}$$

$$\frac{\partial \phi_j(\xi_i)}{\partial \Sigma_j} = \frac{\partial}{\partial \Sigma_j}\left((2\pi)^D \det(\Sigma_j)\right)^{-\frac{1}{2}} \exp\left((\boldsymbol{\mu}_j - \xi_i)^\intercal \Sigma_j^{-1}(\boldsymbol{\mu}_j - \xi_i)\right) \tag{3.19}$$

$$= \frac{\partial \left((2\pi)^D \det(\Sigma_j)\right)^{-\frac{1}{2}}}{\partial \Sigma_j} \exp\left((\boldsymbol{\mu}_j - \xi_i)^\intercal \Sigma_j^{-1}(\boldsymbol{\mu}_j - \xi_i)\right)$$
$$+ \left((2\pi)^D \det(\Sigma_j)\right)^{-\frac{1}{2}} \frac{\partial}{\partial \Sigma_j} \exp\left((\boldsymbol{\mu}_j - \xi_i)^\intercal \Sigma_j^{-1}(\boldsymbol{\mu}_j - \xi_i)\right)$$
$$= -\frac{1}{2}\phi_j(\xi_i)\det(\Sigma_j)^{-1}\,\mathrm{diag}\left(\left(\prod_{\substack{m=1,\\ m\neq m'}}^{D} \Sigma_{j,mm}\right)_{m'=1,\ldots,D}\right)$$
$$+ \frac{1}{2}\phi_j(\xi_i)\Sigma_j^{-2}\,\mathrm{diag}(\boldsymbol{\mu}_j - \xi_i)^2 \tag{3.20}$$

Combining these gradients with the ones provided in (Deisenroth et al., 2015) via chain rule, we get $\partial \Phi_{ij}/\partial \boldsymbol{\theta}$. Substituting the result in Eq. (3.15), we compute $\partial \boldsymbol{\alpha}_t/\partial \boldsymbol{\theta}$. Finally, returning to Eq. (3.12) we have computed all components and get a closed-form expression for $\partial \widetilde{R}_\pi(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$.



(a) Product Quadrature **(b)** Mountain Car

Figure 3.3.: Construction of suitable quadrature rules. Plot (a) shows the nodes of a Gaussian product quadrature rule on the unit square. Here, the quadrature rule for the unit square was constructed as an outer product of the Gaussian quadrature rule with 5 nodes. A state space partition with approximately 4000 rectangles for the mountain car system obtained with Algorithm 5 is shown in (b). For each rectangle in this partition, a Gaussian product quadrature such as (a) is employed.

3.2.6 Construction of Quadrature Rules

Our proposed policy search approach nuQuPS employs numerical quadrature to approximate GP predictions when the input is a distribution. The derived expressions for the approximate distribution and the gradients of the expected long-term reward are valid for any choice of quadrature rule. However, in practice, the choice of a suitable quadrature rule is crucial to the accuracy of the approximate long-term predictions and, thus, to the learning success of nuQuPS. Thus, in the following, we will elaborate on how to construct "good" quadrature rules for a given GP dynamics model. During learning, nuQuPS constructs a quadrature rule in every episode, that is suited to the current GP dynamics model.

Gaussian product quadrature extends univariate Gaussian quadrature to a multivariate rule using a product grid of evaluation points. This construction has some desirable properties, such as positive quadrature weights and readily available quadrature nodes. However, being an outer product of one-dimensional rules, it suffers from the curse of dimensionality. In this chapter, we apply numerical quadrature to integrals as in Equation (3.6) and address the mentioned drawback as follows. Typically, the system trajectories are not uniformly spread over the state space. Instead, they are concentrated in a significantly smaller region. We exploit this observation to improve the efficiency of product grid quadrature and cope with the curse of dimensionality. For this purpose, we partition the state space $X = X_1 \sqcup \cdots \sqcup X_L$ and apply a Gaussian product quadrature to each obtained subregion X_l. The next state distribution, cf. Eq. (3.6), can be written as

$$F[p(\boldsymbol{x}_t)] \coloneqq \int_X p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t = \sum_{l=1}^{L} \int_{X_l} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t) p(\boldsymbol{x}_t) d\boldsymbol{x}_t$$

and, applying a numerical quadrature rule with nodes $\mathbb{X}_l$ for each integral, we get

$$F[p(\boldsymbol{x}_t)] \approx \sum_{l=1}^{L} \sum_{\xi_n \in \mathbb{X}_l} w_{nl} p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t = \xi_n) p(\boldsymbol{x}_t = \xi_n).$$

Setting $\mathbb{X} = \mathbb{X}_1 \sqcup \cdots \sqcup \mathbb{X}_L$, we recover Eq. (3.7). Thus, composed quadrature rules can be handled just as a single Gaussian product rule. We exploit this fact to construct quadrature rules which are efficient for the computation of next state distributions for our particular GP dynamics model and subregion of policy parameter space. In other words, we aim to find a partition $X = X_1 \sqcup \cdots \sqcup X_L$ of the state space, such that integral (3.6) is approximated well by the resulting quadrature rule for all policies $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ that will be evaluated during gradient ascent in the current episode. For this purpose, we maintain a partition of the state space, sample trajectories and subdivide regions X_l that were visited and fulfill a certain criterion which we introduce below.

To sample representative system trajectories, we perturb the currently known best policy parameters $\boldsymbol{\theta}^*$ with white noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \Sigma_{\boldsymbol{\epsilon}})$. We sample starting states $\boldsymbol{\tau}_0$ from $p(\boldsymbol{x}_0)$ and compute a rollout $\boldsymbol{\tau}_0, \ldots, \boldsymbol{\tau}_T$ and one rollout from the target state $\boldsymbol{x}_d$ with each sampled policy parameter vector $\boldsymbol{\theta}$. The noise variance $\Sigma_{\boldsymbol{\epsilon}}$ determines how far we expect to differ from the initial policy during gradient ascent. Thus, we start with a high variance and decrease it with the number of episodes.

We subdivide the region X_l, if it does not contain enough quadrature nodes to integrate predictive distributions from the dynamics GP well. To estimate whether X_l should be divided, we introduce

$$\rho_l(\boldsymbol{\tau}) := \frac{\operatorname{vol}(X_l)}{|\mathbb{X}_l| \min_{\boldsymbol{\tau}_i \in X_l} \operatorname{Var}(\boldsymbol{\tau}_i)} \tag{3.21}$$

and subdivide X_l if $\rho_l(\boldsymbol{\tau})$ is greater than 1. This criterion relates the higher order derivatives of GP predictions which fall inside X_l with the quadrature node density in X_l.

Constructing a composed quadrature rule with this approach will concentrate most quadrature nodes in state space regions that are visited frequently when following system trajectories. Algorithm 5 summarizes our approach and Figure 3.3 illustrates the constructed quadrature rules.

3.3 Numerical Quadrature for Probabilistic Policy Search with Infinite Time Horizons

In control, many problems are not equipped with a finite time scale and controllers are typically designed to optimize the long-term behavior of the system. Thus, in most practical applications we are interested in controllers that run forever, i.e., the goal is to learn controllers for infinite time horizons. For nuQuPS, however, we considered only finite time horizons so far, as the state distribution is analytically intractable and must be approximated separately for every time step. Thus, we constrained ourselves to take a finite number of steps into account for the expected long-term reward.

In this section, we explore the possibility to consider infinite time horizons. As we have seen in Section 2.4, there exists a unique limiting state distribution for any closed-loop control system with Gaussian process forward dynamics with constant mean prior and squared exponential kernel.

Algorithm 5 Construction of composed quadrature rules

Input: dynamics GP $\boldsymbol{g}: (\boldsymbol{x}_t, \boldsymbol{u}_t) \mapsto \boldsymbol{x}_{t+1}$, start distribution $p(\boldsymbol{x}_0)$, current best policy $\pi_{\boldsymbol{\theta}^*}$, state space X, maximum partition size $L_{\max}$, policy noise variance $\Sigma_{\boldsymbol{\epsilon}}$

Output: composed quadrature rule with nodes $\mathbb{X}$ and weight vector $\boldsymbol{w}$

Initialize partition $X = X_1 \sqcup \cdots \sqcup X_s$ and quadrature $\mathbb{X} = \mathbb{X}_1 \sqcup \cdots \sqcup \mathbb{X}_s$ with $s < L_{\max}$
while $s < L_{\max}$ **do**
  Sample starting state $\boldsymbol{\tau}_0 \sim p(\boldsymbol{x}_0)$ and policy parameters $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}^*, \Sigma_{\boldsymbol{\epsilon}})$
  Compute rollouts $\boldsymbol{\tau}_0, \ldots, \boldsymbol{\tau}_T$ and $\boldsymbol{x}_d = \boldsymbol{\tau}'_0, \ldots, \boldsymbol{\tau}'_T$ with dynamics GP $\boldsymbol{g}$ and policy $\pi_{\boldsymbol{\theta}}$
  for $l = 1, \ldots, s$ **do**
    if $\frac{\operatorname{vol}(X_l)}{|\mathbb{X}_l| \min_{\boldsymbol{\tau}_i, \boldsymbol{\tau}'_i \in X_l} \operatorname{Var}(\boldsymbol{\tau}_i)} < 1$ **then** subdivide X_l, add nodes to $\mathbb{X}$ **fi**
  od
od
return quadrature nodes $\mathbb{X}$ and weights $\boldsymbol{w}$

The system state will converge to this stationary distribution irrespective of the initial starting state (certain or uncertain). The stationary distribution depends on the policy $\pi_{\boldsymbol{\theta}}$ and, thus, on its parameters $\boldsymbol{\theta}$. Let us write $p_{*,\boldsymbol{\theta}}(\boldsymbol{x})$ for the probability density of the stationary distribution. The question we explore in the following is whether we can optimize the system's stationary distribution with respect to the policy parameters $\boldsymbol{\theta}$. More precisely, we want to find a parameter vector $\boldsymbol{\theta}_*$, such that the expected reward with respect to the limiting distribution $p_{*,\boldsymbol{\theta}_*}(\boldsymbol{x})$ is maximal, i.e.,

$$\boldsymbol{\theta}_* = \max_{\boldsymbol{\theta}} R_\pi^\infty(\boldsymbol{\theta}) \coloneqq \max_{\boldsymbol{\theta}} \int r(\boldsymbol{x}) p_{*,\boldsymbol{\theta}}(\boldsymbol{x}) d\boldsymbol{x}. \tag{3.22}$$

To optimize the system's stationary distribution, we need to be able to compute its probability density $p_{*,\boldsymbol{\theta}}$ for any given parameter vector $\boldsymbol{\theta}$. The stationary distribution is the unique solution of

$$p_{*,\boldsymbol{\theta}}(\boldsymbol{x}_{t+1}) = \int p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t)) p_{*,\boldsymbol{\theta}}(\boldsymbol{x}_t) d\boldsymbol{x}_t \tag{3.23}$$

with the GP predictive distribution $p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t))$. This equation is a homogeneous Fredholm equation of the second kind and this type of equation has been studied extensively for various integral kernels. Unfortunately, it is analytically intractable for our integral kernel $p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t))$ and, thus, we need to approximate its solution $p_{*,\boldsymbol{\theta}}$. We will follow the well-known Nyström method (Jerri, 1999), which approximates the solution of Fredholm equations of the second kind via numerical quadrature. Applying numerical quadrature to the integral on the right hand side

$$p_{*,\boldsymbol{\theta}}(\boldsymbol{x}_{t+1}) = \int p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t)) p_{*,\boldsymbol{\theta}}(\boldsymbol{x}_t) d\boldsymbol{x}_t \tag{3.24}$$

$$\approx \sum_{i=1}^{N} w_i p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t = \xi_i, \pi(\boldsymbol{x}_t = \xi_i)) p_{*,\boldsymbol{\theta}}(\xi_i), \tag{3.25}$$

we obtain an approximation of $p_{*,\boldsymbol{\theta}}$ as a finite mixture of Gaussians. Additionally, the Nyström method enforces that $p_{*,\boldsymbol{\theta}}$ is a distribution by normalizing the solution to have mass 1. Note that the mixture components coincide with the basis functions $\phi_1, \cdots, \phi_N$ as in Section 3.2.4 and we can write

$$p_{*,\boldsymbol{\theta}}(\boldsymbol{x}) \approx \boldsymbol{\phi}^\intercal \boldsymbol{\alpha}^\infty \tag{3.26}$$

with the Gaussian basis functions $\phi_i(\boldsymbol{x}) = p(\boldsymbol{x}_{t+1} = \boldsymbol{x} \mid \boldsymbol{x}_t = \xi_i, \pi_{\boldsymbol{\theta}}(\xi_i))$. However, the mixture weights are unknown, as they involve the probability density of the stationary distribution $\alpha_i^\infty = w_i p_{*,\boldsymbol{\theta}}(\xi_i)$. To find the weights, we collocate $\boldsymbol{x}_{t+1}$ in Eq. (3.24) with the quadrature nodes $\xi_1, \cdots, \xi_N$ and obtain a linear equation system of size $N \times N$

$$(\operatorname{diag}(\boldsymbol{w})\Phi - I)\boldsymbol{\alpha}^\infty = 0 \tag{3.27}$$

with the matrix $\Phi_{i,j} = \phi_j(\xi_i)$. To ensure that $p_{*,\boldsymbol{\theta}}$ is a probability density, we normalize the columns of $\operatorname{diag}(\boldsymbol{w})\Phi$ to sum to 1. The solution $\boldsymbol{\alpha}^\infty$ of this equation is the eigenvector of $\operatorname{diag}(\boldsymbol{w})\Phi$ corresponding to the eigenvalue $\lambda = 1$. As $\operatorname{diag}(\boldsymbol{w})\Phi$ is a stochastic matrix, i.e., each of its columns sums to 1, of which we can solve for $\boldsymbol{\alpha}^\infty$. Furthermore, by the Perron-Frobenius theorem the eigenvalue $\lambda = 1$ is simple and, thus, there exists one dominant eigenvector $\boldsymbol{\alpha}^\infty$.

Above, we have seen how the Nystöm method can be employed to approximate the stationary distribution of the considered closed-loop control system. Our approximation (3.26) of the stationary distribution depends on the policy parameters $\boldsymbol{\theta}$, as the basis functions $\phi_1, \cdots, \phi_N$ and the matrix Φ change, when we change $\boldsymbol{\theta}$. How does a change in $\boldsymbol{\theta}$ influence the expected reward of the stationary distribution? Substituting our approximation (3.26) for $p_{*,\boldsymbol{\theta}}$, we get

$$R_\pi^\infty(\boldsymbol{\theta}) = \int r(\boldsymbol{x}) p_{*,\boldsymbol{\theta}} d\boldsymbol{x} \approx \sum_{i=1}^N \alpha_i^\infty \int r(\boldsymbol{x})\phi_i(\boldsymbol{x}) d\boldsymbol{x} = \sum_{i=1}^N \alpha_i^\infty \mathbb{E}_{\boldsymbol{x}\sim\phi_i}[r(\boldsymbol{x})]. \tag{3.28}$$

As already mentioned in Section 3.2.5, the analytic gradients of $\mathbb{E}_{\boldsymbol{x}\sim\phi_i}[r(\boldsymbol{x})]$ with respect to $\boldsymbol{\theta}$ were provided in (Deisenroth et al., 2015) for squashed linear and squashed RBF policies. The α_i^∞ in Eq. (3.28) depend on $\boldsymbol{\theta}$, as they are the entries of the eigenvector of $\operatorname{diag}(\boldsymbol{w})\Phi$. A priori, it seems that this fact renders our policy search approach infeasible, as the dominant eigenvectors of general matrices do not depend differentiably on the matrix parameters, e.g., the number and multiplicity of eigenvalues might change with $\boldsymbol{\theta}$. However, in our case the matrix $\operatorname{diag}(\boldsymbol{w})\Phi$ is positive and the Perron-Frobenius theorem guarantees that the multiplicity of $\lambda = 1$ is one and, thus, the existence of $\boldsymbol{\alpha}^\infty$ for any choice of $\boldsymbol{\theta}$. Fortunately, the dominant eigenvector of positive matrices is differentiable with respect to the parameters of the matrix entries (Deutsch and Neumann, 1985) as

$$\frac{\partial \boldsymbol{\alpha}^\infty}{\partial \boldsymbol{\theta}} = (I - \operatorname{diag}(\boldsymbol{w})\Phi)^\# \left(\frac{\partial \operatorname{diag}(\boldsymbol{w})\Phi}{\partial \boldsymbol{\theta}}\right)\boldsymbol{\alpha}^\infty \tag{3.29}$$

with the generalized group inverse $(I - \operatorname{diag}(\boldsymbol{w})\Phi)^\#$ of $I - \operatorname{diag}(\boldsymbol{w})\Phi$. Thus, we can compute the gradients of $R_\pi^\infty(\boldsymbol{\theta})$ with respect to the policy parameters $\boldsymbol{\theta}$ and perform gradient ascent to solve Equation (3.22).

Algorithm 6 Numerical Quadrature for Probabilistic Policy Search with Infinite Time Horizons

Input: dynamics GP $\boldsymbol{g} : (\boldsymbol{x}_t, \boldsymbol{u}_t) \mapsto \boldsymbol{x}_{t+1}$, initial policy $\pi_{\boldsymbol{\theta}}$, quadrature rule $\boldsymbol{w}, \mathbb{X}$
Output: policy $\pi_{\boldsymbol{\theta}_*}$ that maximizes $R_\pi^\infty(\boldsymbol{\theta})$

Initialize $\boldsymbol{\alpha_0}$ randomly and normalize $\boldsymbol{\alpha}_0 \leftarrow |\boldsymbol{\alpha}_0| / \|\boldsymbol{\alpha}_0\|_1$
Perform gradient ascent to compute $\boldsymbol{\theta}_*$
while not converged **do**
  $\phi_i \leftarrow \boldsymbol{g}(\xi_i, \pi(\xi_i))$ for $i = 1, \cdots, N$
  $\Phi_{i,j} \leftarrow \phi_j(\xi_i)$ for $i, j = 1, \cdots, N$
  Normalize columns $(\mathrm{diag}(\boldsymbol{w})\Phi)_{i,j} \leftarrow (\mathrm{diag}(\boldsymbol{w})\Phi)_{i,j} / \sum_{k=1}^N w_k \phi_j(\xi_k)$
  Compute dominant eigenvector $\boldsymbol{\alpha}_{\boldsymbol{\theta}}^\infty$ via power iteration (3.30) starting from $\boldsymbol{\alpha}_0$
  $\boldsymbol{\alpha_0} \leftarrow \boldsymbol{\alpha}_{\boldsymbol{\theta}\infty}$
  Estimate gradient $\frac{\partial \boldsymbol{\alpha}_{\boldsymbol{\theta}}^\infty}{\partial \boldsymbol{\theta}}$ numerically performing power iterations starting from $\boldsymbol{\alpha}_0$
  Use $\frac{\partial \boldsymbol{\alpha}_{\boldsymbol{\theta}}^\infty}{\partial \boldsymbol{\theta}}$ and gradient from (Deisenroth et al., 2015) to compute $\frac{\partial R_\pi^\infty(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
  make gradient ascent step
end while
$\boldsymbol{\theta}_* \leftarrow \boldsymbol{\theta}$

In practice, the matrix $\mathrm{diag}(\boldsymbol{w})\Phi$ is typically very large and the computation of its dominant eigenvector as well as the generalized group inverse of $I - \mathrm{diag}(\boldsymbol{w})\Phi$ is challenging and time consuming. For a gradient ascent scheme, where the gradients must be computed hundreds of times, the time required for these computations is prohibitive. As the matrix $\mathrm{diag}(\boldsymbol{w})\Phi$ is positive, the Perron-Frobenius theorem guarantees that the power iteration

$$\boldsymbol{\alpha}_{t+1} = \mathrm{diag}(\boldsymbol{w})\Phi\boldsymbol{\alpha}_t \tag{3.30}$$

with arbitrary $\boldsymbol{\alpha}_0$ converges to $\boldsymbol{\alpha}^\infty$. This iteration can be executed efficiently on a GPU, as the matrix $\mathrm{diag}(\boldsymbol{w})\Phi$ needs to be transferred to the GPU memory only once and matrix-vector multiplications can be highly parallelized. Furthermore, power iteration can be applied to estimate the gradient of $\boldsymbol{\alpha}^\infty$ with respect to $\boldsymbol{\theta}$ numerically. To increase efficiency, the start vector for the power iteration to compute the dominant eigenvector for $\boldsymbol{\theta} + h$ can be set to the dominant eigenvector for $\boldsymbol{\theta}$, which has been previously computed. In this case, the vector typically converges in a few iterations.

Algorithm 6 summarizes our policy search approach for infinite time horizons. The approach can straightforwardly be applied in an episodic setting to learn controller and system dynamics simultaneously.

3.4 Empirical Evaluation on Typical Benchmark Problems

We evaluate the proposed algorithm on two benchmark tasks: mountain car and cart-pole swing up and hold. We compare our results with other model-based approaches in multiple experiments. First, we briefly introduce the two test-beds.

Mountain Car. In the mountain-car domain (see, e.g., (Moore and Atkeson, 1995; Sutton and Barto, 1998)), a car starts at some point in a valley landscape and has to reach a certain point on the hill to the right side of the valley and stay there. However, the car's engine is not powerful enough to reach the goal directly from all starting positions. From the points in the valley, the car has to first drive in the opposite direction and gain momentum to reach the goal. The state space has two dimensions: position and velocity of the car, the control signal was limited to $u_{\max} = 4$.

Cart-Pole. In the cart-pole domain (Deisenroth et al., 2015), a cart with an attached free-swinging pendulum is running on a track of limited length. The goal is to swing the pendulum up and balance it, with the cart in the middle of the track. The state space has four dimensions: position of the cart x, velocity of the cart $\dot{x}$, angle of the pendulum ϑ and the angular velocity $\dot{\vartheta}$. A horizontal force with $u_{\max} = 10$ can be applied to the cart.

To evaluate the proposed algorithm (nuQuPS), we perform several experiments on the two test-beds. First, we examine how the approximation of multi-step-ahead predictions affects learning, comparing moment matching to the proposed numerical quadrature on the mountain car task. Second, we perform the cart-pole swing up and balance task. Third, we demonstrate how nuQuPS is capable of learning global policies.

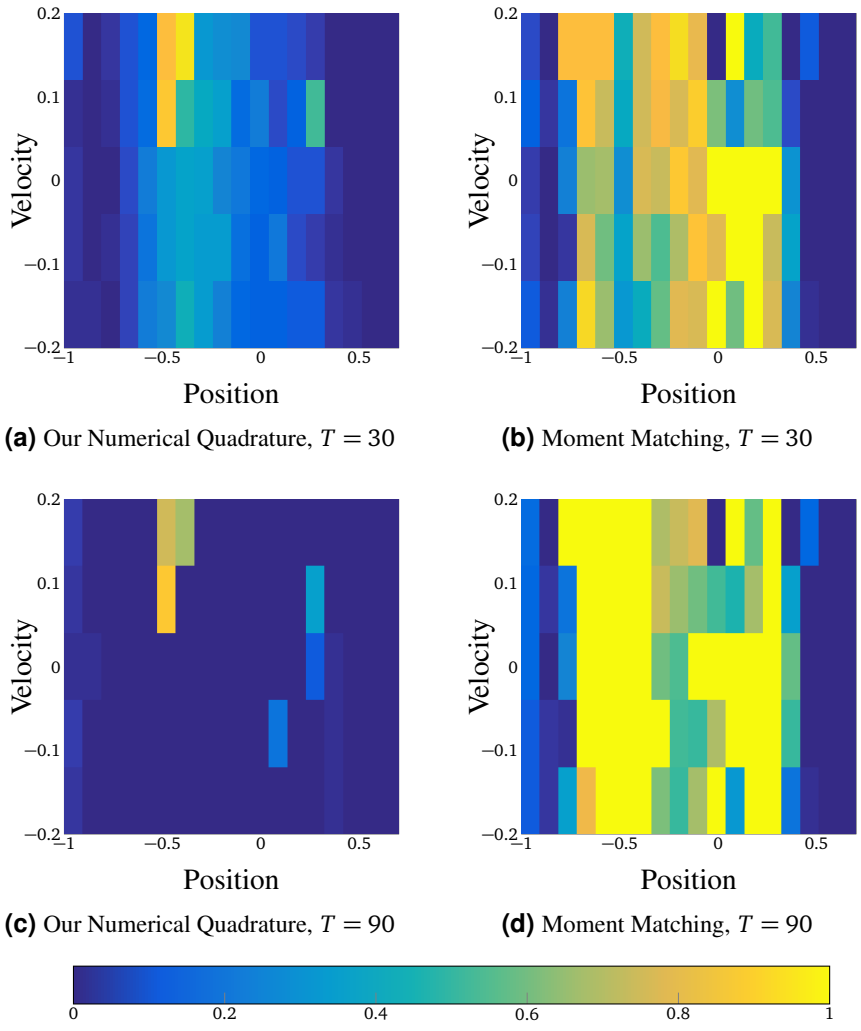
3.4.1 Learning Control in the Mountain Car Domain

In this experiment, we want to evaluate how the proposed numerical quadrature approximate inference performs in comparison to moment matching and how these approximation methods affect learning. For this purpose, we start with a given, pre-trained GP dynamics model for the mountain car domain. We want to test how well we can learn policies for starting states concentrated in different regions of the state space. Thus, we build a grid in the state space region $[-1, 1] \times [-0.2, 0.2]$ and conduct a policy search for each grid point, choosing $p(\boldsymbol{x}_0)$ as a Gaussian centered at the corresponding grid point. To evaluate the learned policies, we test them on the given GP dynamics. From each starting distribution, we compute 100 rollouts employing the learned policy. We average the distance of the final rollout point to the target state over these rollouts. We color each cell according to the computed average distance, which gives a map of the learning success.

We perform this experiment computing the expected long-term reward and its gradient with respect to the policy parameters with (i) numerical quadrature and (ii) moment matching. For the start distributions, we choose the covariance matrix $0.0025I$. For every starting distribution, we set the time horizon $T = 30$ and learn a squashed linear policy $\boldsymbol{\pi}(\boldsymbol{x}) = u_{\max}\sigma(M\boldsymbol{x} + b)$, cf. Section 3.2.3.

Figure 3.4, plots (a) and (b), show the results. As can be seen, numerical quadrature provides a more accurate estimate of the expected reward and its gradient and, thus, is more reliable in learning a successful policy. However, maximizing the expected long-term reward is a non-convex optimization task and, thus, optimization does not find a successful policy in all cases.

Note, that solving the mountain car task with a squashed linear policy requires to maintain a delicate balance between gaining momentum for the car to drive up the hill and slowing the car down at the right time to hold it at the target. To evaluate whether we succeeded to learn policies that are capable of both accelerating the car sufficiently and hold it, as opposed to the



(a) Our Numerical Quadrature, $T = 30$

(b) Moment Matching, $T = 30$

(c) Our Numerical Quadrature, $T = 90$

(d) Moment Matching, $T = 90$

Figure 3.4.: Comparison of the numerical quadrature and moment matching approximations for policy search. Given a pre-trained GP dynamics model for the mountain car domain, a squashed linear policy was learned starting in each cell. Policy values and gradients were computed with numerical quadrature (plots (a) and (c)) and moment matching (plots (b) and (d)). With the learned policy, rollouts from the corresponding cell were computed with time horizons $T = 30$ and $T = 90$. The cells are colored according to the average distance of the rollout final state to the target state.

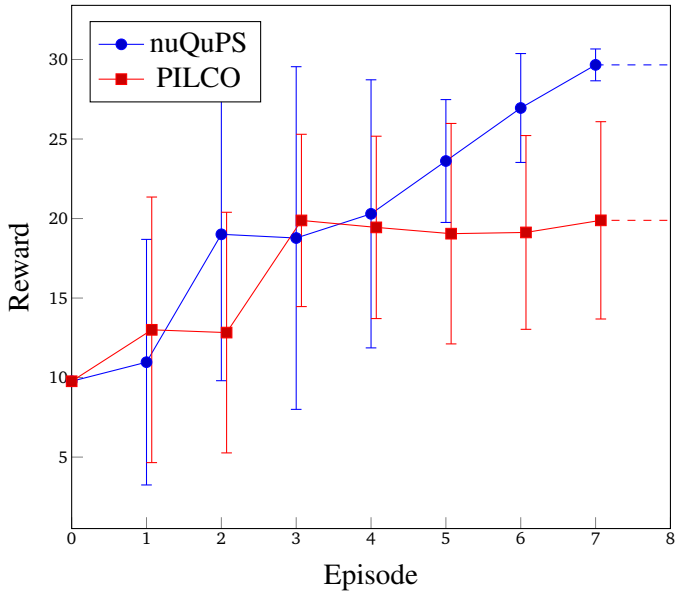


Figure 3.5.: Average reward with standard deviation as a function of system interaction time for the cart-pole swing up and hold task. One episode equals 4 seconds of system interaction time. A squashed linear policy was learned with our nuQuPS approach (blue) and PILCO (red). The results were averaged over five trials with different initial policies.

local optimum of accelerating the car as much as possible and overshoot, we also test the learned policies with a larger time horizon $T = 90$. The results are shown in Figure 3.4, plots (c) and (d). As can be seen, in most cases numerical quadrature finds a policy that can both accelerate and hold the car. In contrast, moment matching typically fails at this task.

3.4.2 Learning Control in the Cart-Pole Domain

To evaluate overall performance of the proposed nuQuPS, we learn to swing up and balance the pendulum in the cart-pole domain. Starting with the cart standing in the middle of the tracks, pendulum down, we perform five experiments with different initial policies. The starting state variance was set to 10^{-6} and the time horizon to $T = 40$. We learn squashed linear policies for this task. Note that a linear policy cannot swing up and then balance the pendulum on the cart. However, there are squashed linear policies that are capable of this task. Figure 3.5 shows the average reward per episode obtained with nuQuPS in comparison to the PILCO algorithm. The policies from each episode were evaluated on 100 rollouts, where the starting point was drawn from the start distribution $p(\boldsymbol{x}_0)$.

3.4.3 Learning Global Policies for the Mountain Car Task

In this experiment, our goal is to learn a policy that can successfully solve the mountain car task for a continuous region of starting points. One major advantage of nuQuPS is that it can handle

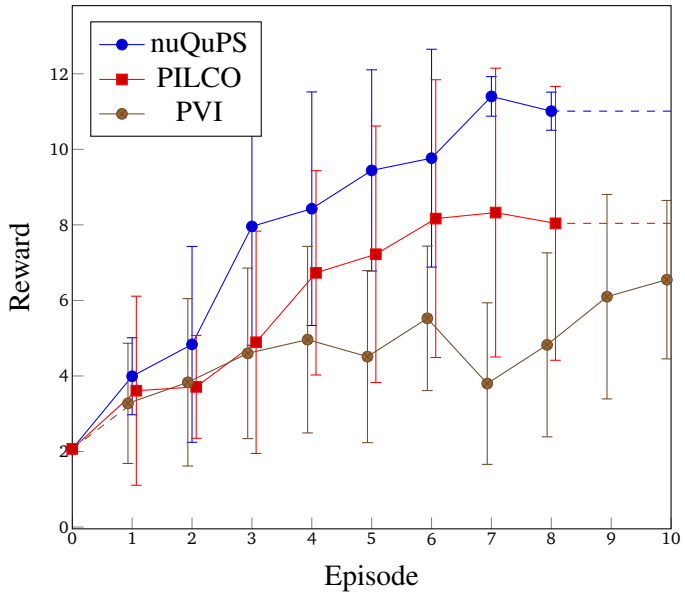


Figure 3.6.: Average reward with standard deviation as a function of system interaction time for the mountain car benchmark. One Episode equals 3 seconds of system interaction time. The task was to learn a policy for all car starting positions in $[-1, 1] \times [-0.1, 0.1]$. For nuQuPS, a uniform distribution over this region was chosen as the starting state. For PILCO, we chose 10 starting points on a regular grid in the starting region. PVI obtained rollouts with the starting point sampled from the uniform starting distribution at the end of each episode. After learning, a squashed, linear policy was fitted with a least squares approach.

arbitrary starting state distributions. For our task, we choose the starting state to be uniformly distributed on $[-1, -0.1] \times [1, 0.1]$.

We compare the learning success with two other model-based approaches: probabilistic value iteration (PVI) (Bischoff et al., 2013) and PILCO (Deisenroth et al., 2015). PVI learns a global value function and, thus, can handle our uniform starting state distribution. For the rollout performed at the end of an episode, the starting point is sampled from $p(\boldsymbol{x}_0)$. As a value iteration approach, PVI does not assume a certain parametric form of the policy, which enables highly flexible and complex policies. However, to compute the learned optimal actions, an argmax must be performed for every time step, which is computationally challenging and often impractical. To overcome this drawback and also make PVI comparable to the other algorithms, after learning, we fit a squashed linear policy with least squares.

To learn a policy for all starting states in $[-1, -0.1] \times [1, 0.1]$ with PILCO, we follow the approach in (Deisenroth et al., 2014). We distribute ten starting points over $[-1, -0.1] \times [1, 0.1]$ and maximize the mean expected long-term reward of these points with PILCO.

The results are shown in Figure 3.6. For each policy, the obtained reward was averaged over 3000 rollouts with starting points sampled from the initial state distribution. We perform five trials with different initial policies for the three approaches. As can be seen, nuQuPS outperforms PILCO and PVI on this task. We found that PILCO is prone to local optima, which leads to a high variance in learning success. PVI, on the other hand, shows slower convergence, as learning a global value function is a significantly harder task than finding policy parameters for a given starting state distribution. In addition, in PVI the support points must be chosen, where the value function will be computed and used as training data for the value function model. This selection of support points is a nontrivial task and can highly affect the learning success.

3.5 Conclusion

Learning control is promising as it allows to drastically reduce the amount of expert knowledge, that is required otherwise. Model-based approaches are particularly appealing since they are highly data efficient, especially when probabilistic models are employed to account for the inherently uncertain dynamics estimates. However, long-term predictions with such models typically become intractable, introducing the need for approximations. In this chapter, we show that the accuracy of the chosen approximation method significantly affects learning and propose numerical quadrature to approximate long-term predictions. We conclude the chapter with a short summary of the main contributions and a brief outlook on possible future work in this direction.

3.5.1 Summary of Contributions

In this chapter, we introduced nuQuPS, a model-based policy search approach, that makes use of Gaussian processes as dynamics models. To propagate uncertainties through the GP dynamics model, nuQuPS employs numerical quadrature. Numerical quadrature for approximate inference can model complex distributions, e.g., with multiple modes. The numerical quadrature approximation can be parallelized straightforwardly and, thus, be computed efficiently. Furthermore, we provided analytic gradients that can be used for policy search. With these analytic gradients, policy improvement can be performed with any gradient based optimization scheme.

Due to its flexibility, numerical quadrature provides highly accurate approximations even for complex distributions, e.g., when the input distribution has high variance. This high accuracy significantly speeds up learning and results in enhanced robustness and data efficiency. Additionally, numerical quadrature can handle arbitrary, non-Gaussian starting state distributions, e.g., a uniform distribution over all possible starting states. As a result, nuQuPS provides a principled way to learn policies, that are suitable, e.g., for all feasible starting states. Thus, nuQuPS combines the ability to learn global policies and remains scalable to high-dimensional problems.

Combining Bayesian averaging with high-performance uncertainty propagation, nuQuPS achieves unprecedented data-efficiency on all tested tasks. It is highly robust to different choices of initial policy. Extensive evaluation on two simulated test-beds demonstrate nuQuPSs superior performance.

3.5.2 Discussion and Next Steps

The proposed nuQuPS algorithm allows to robustly learn global policies with unprecedented data-efficiency. However, there remain several open questions. To learn a policy, nuQuPS proceeds in an episodic setting, acquiring more data from system interactions, updating its dynamics model and re-optimizing the policy to suit the updated dynamics model. In all tests we performed, this procedure converged after a few episodes. However, as for most policy search approaches on continuous state and action spaces, there is no guarantee that nuQuPS will converge. In nuQuPS, maximizing the expected long-term reward is a non-convex optimization task and, thus, no guarantees can be given. However, even when assuming that the global optimum is found in every episode, convergence of nuQuPS remains an open problem.

Second, our nuQuPS approach, as most previous policy search approaches, optimizes the policy for the case that there are no external disturbances. Robust control, however, considers the problem of learning a policy that is successful even when external disturbances are present. To solve this task, assumptions e.g., on the parametric form of the disturbance are made and stability analysis is employed. Recently, stability analysis for learned GP dynamics and policies was provided (Vinogradska et al., 2016). This stability analysis combined with a high performance policy search approach as the proposed nuQuPS could open the door for learning robust control.

Finally, numerical quadrature has proven to provide highly accurate approximate multi-step-ahead predictions, that greatly speed up policy learning. These high-quality approximations could as well support other Bayesian model-based learning approaches. Especially for value iteration based on GP dynamics models (Bischoff et al., 2013; Rasmussen and Kuss, 2004; Deisenroth et al., 2009), numerical quadrature could be promising to cope with the intractability of the model. On the other hand, numerical quadrature is known to scale badly with the number of state dimensions. A thorough analysis of the practical implications of scaling numerical quadrature is neeeded to enable the use of NQ as a general, highly accurate approximate inference method for Bayesian models.

4 Approximate Value Iteration based on Numerical Quadrature

In Chapter 3, we showed that data efficiency in policy search based on Gaussian process dynamics models can be increased, when a highly accurate approximation for multi-step-ahead predictions is employed. A different approach to learning control policies is taken by value function based methods. These approaches do not make any assumptions about the policy. Instead, they learn a value function that indicates how desirable a state is in the long run for the specific task. The optimal control policy is then determined by greedily maximizing the value function. In value iteration, the value function is computed iteratively as the fixed point of its characteristic equation. As opposed to policy search, value iteration is a global approach that tries to infer optimal control for any system state. When the system dynamics is modeled as a Gaussian process with nonlinear kernel, the value function (and its iterative form) is analytically intractable. Thus, the value propagation step must be approximated just as in policy search. However, in policy search the state is simulated forward employing the dynamics model, while value iteration employs the model to back up a state's value with every iteration step. In this chapter, we propose an approximate value iteration approach that employs a highly accurate approximation for the value propagation step. With this highly accurate approximate value propagation, globally optimal control can be learned with high data efficiency.

4.1 Introduction

Learning control has become a viable approach in both the machine learning and control community. Many successful applications impressively demonstrate the advantages of learning control (Deisenroth et al., 2015; Pan and Theodorou, 2014; Klenske et al., 2013; Maciejowski and Yang, 2013; Nguyen-Tuong and Peters, 2011; Engel et al., 2006; Kocijan et al., 2004). In contrast to classical control methods, learning control does not presuppose a detailed understanding of the underlying dynamics but tries to infer the required information from data. Thus, relatively little expert knowledge about the system dynamics is required and fewer assumptions, such as a parametric form and parameter estimates, must be made.

For real-world applications of learning control, it is desirable to minimize the system interaction time. Thus, approaches that explicitly learn a dynamics model are often preferred, as model-free methods can require a prohibitive amount of system interactions (Atkeson and Santamaria, 1997; Kuvayev, 1997; Moore and Atkeson, 1993; Sutton, 1990). However, one drawback of model-based methods is that modeling errors can derail learning, as the inherently approximate and frequently highly erroneous model is implicitly assumed to approximate the real dynamics sufficiently well (Schneider, 1996). Thus, solutions to the approximate control problem might result in policies that do not solve the control task for the true system dynamics. This model bias can have severe consequences especially when few data is available and, thus, the model predictions are highly

uncertain. Hence, employing Gaussian processes (GPs) as forward models for learning control is particularly appealing as they incorporate uncertainty about the system dynamics estimate. GPs infer a distribution over all plausible models given the observed data instead of compromising on an approximate model and, thus, avoid severe modeling errors. Another major advantage of Gaussian process forward models is that stability analyses for such closed-loop control systems are available (Beckers and Hirche, 2016; Vinogradska et al., 2016), including automatic tools (Vinogradska et al., 2016) which do not require any expert knowledge.

A well-studied class of approaches to policy learning are value function based methods. These methods compute the value function, that maps each state to the expected long-term reward when starting in this state and following an optimal policy. The optimal policy can then be inferred by greedily optimizing the value function. Variations of this approach include iterative computation of the value function (value iteration) and alternating between computation of the value function for a fixed policy and improving this policy based on the value function (policy iteration), see e.g., (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998). For finite state and action spaces, these approaches solve the optimal control task exactly and yield globally optimal policies. However, for continuous state and action spaces analytical solutions are only known for linear dynamics with quadratic cost and Gaussian noise (Bertsekas, 2005).

In this chapter, we introduce AVINQ (Approximate Value Iteration based on Numerical Quadrature), a probabilistic value iteration approach that approximates the value function iterates by projection onto a linear space of learned features. The dynamics is modeled as a GP and learned in an episodic setting. To enable projection of the (analytically intractable) value function iterates, numerical quadrature is employed. The proposed approach can learn globally optimal control from scratch. Employing GPs as forward dynamics models, AVINQ accurately handles the model uncertainty and is highly data efficient. It requires no manual effort or problem specific knowledge. Furthermore, as the quadrature error can be upper bounded, under some assumptions it can be shown that AVINQ converges to a globally optimal policy.

The chapter will be organized as follows: first, we give an overview of value iteration approaches and specify the considered problem. In Section 4.2.3, we briefly review related work. Section 4.3 introduces the proposed algorithm, which is evaluated on multiple benchmark tasks in Section 4.4. A conclusion summarizes and discusses the provided results (Section 4.5).

4.2 Preliminaries

We begin with a description of the problem we consider in this chapter. Subsequently, we recap some fundamental concepts of reinforcement learning, especially the class of value function based approaches. Finally, we briefly review related work.

4.2.1 Problem Statement

In this paper, we aim to learn to control a previously unknown dynamics system. We consider discrete-time dynamics

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\varepsilon} \tag{4.1}$$

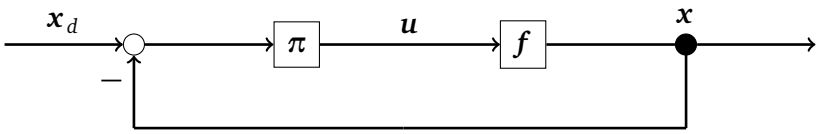


Figure 4.1.: A closed-loop control structure with controller π, system dynamics f and target state x_d. We model the system dynamics f as a Gaussian process. The proposed algorithm learns f from interactions with the real system, computes the optimal value function and infers the policy π by greedily optimizing this value function.

with $x_t, x_{t+1} \in \mathbb{R}^D$, $u_t \in \mathbb{R}^F$, unknown f and i.i.d. Gaussian measurement noise $\varepsilon \in \mathcal{N}(0, \Sigma_\varepsilon)$. Figure 4.1 shows such a closed-loop control setting. Given a reward function $r : x_t \mapsto r(x_t) \in \mathbb{R}_+$, the goal is to find a policy $\pi : x \mapsto \pi(x)$ that maximizes the expected reward up to time horizon T, when choosing $u_t := \pi(x_t)$ for $t = 1, \ldots, T$. We make no further assumptions (such as, e.g., a parametric form) about π. The system dynamics f will be learned from observed data and modeled as a Gaussian process (GP). We aim to minimize the system interactions necessary to learn the optimal policy π.

In the following, we will briefly recap Gaussian process regression. Given noisy observations $\mathcal{D} = \{(z^i, y^i = f(z^i) + \varepsilon^i) \mid 1 \leq i \leq N\}$, where $\varepsilon^i \sim \mathcal{N}(0, \sigma_n^2)$, the prior on the values of f is $\mathcal{N}(0, K(Z,Z) + \sigma_n^2 I)$. The covariance matrix $K(Z,Z)$ is defined by the choice of covariance function k as $[K(Z,Z)]_{ij} = k(z^i, z^j)$. We employ the squared exponential covariance function

$$k(z, w) = \sigma_f^2 \exp\left(-\frac{1}{2}(z - w)^\intercal \Lambda^{-1}(z - w)\right),$$

with signal variance σ_f^2 and squared lengthscales $\Lambda = \operatorname{diag}(l_1^2, \ldots, l_{D+F}^2)$ for all input dimensions. Given a query point z_*, the conditional probability of $f(z_*)$ is

$$f(z_*) \mid \mathcal{D} \sim \mathcal{N}\big(k(z_*, Z)\beta, k(z_*, z_*) - k(z_*, Z)(K(Z,Z) + \sigma_n^2 I)^{-1} k(Z, z_*)\big) \tag{4.2}$$

with $\beta = (K(Z,Z) + \sigma_n^2 I)^{-1} y$. The hyperparameters, e.g., $\sigma_n^2, \sigma_f^2, \Lambda$ for the squared exponential kernel, are estimated by maximizing the log marginal likelihood of the data (Rasmussen and Williams, 2005).

In this thesis, we employ a Gaussian process g to model system dynamics. It takes state-action pairs $z = (x, u)^\intercal$ and outputs differences to successor states, i.e., $x_{t+1} = x_t + g(x_t, u_t)$. As these outputs are multivariate, we train conditionally independent GPs for each output dimension. We write $\sigma_{n,m}^2$, $\sigma_{f,m}^2, \Lambda_m$ for the GP hyperparameters in output dimension m and k_m for the corresponding covariance function.

We will employ the GP dynamics model learned from observed data to infer an optimal policy π. In the following, we will briefly recap value function based approaches.

4.2.2 MDPs and Value Iteration

In reinforcement learning, an agent chooses actions in an environment and recieves immediate rewards for the visited states, aiming to maximize the cumulated long-term reward. A common

assumption is that the next system state depends only on the current state and the chosen action. This setting can be described as a *Markov Decision Process (MDP)*. Formally, an MDP consists of a set of states S, a set of actions A, a state transition p that maps state-action pairs $(\boldsymbol{x}, \boldsymbol{u})$ to the probability $p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u})$ of the successor state $\boldsymbol{x}'$, the immediate reward function $r : S \to \mathbb{R}$ and a discount factor $0 < \gamma < 1$.

The goal in an MDP is to find a policy $\pi : \boldsymbol{x} \mapsto \boldsymbol{u}$, that maximizes the expected long-term reward. For any policy π, the *Value Function* $V_\pi : S \to \mathbb{R}$ rates how desirable a state $\boldsymbol{x}$ is in the long run when following the policy π. More precisely, it is defined as

$$V_\pi(\boldsymbol{x}_0) = \sum_{t=0}^{T} \gamma^t \mathbb{E}_{\boldsymbol{x}_t}[r(\boldsymbol{x}_t)] \tag{4.3}$$

$$= r(\boldsymbol{x}_0) + \gamma \int_S p(\boldsymbol{x}_1 \mid \boldsymbol{x}_0, \pi(\boldsymbol{x}_0)) V_\pi(\boldsymbol{x}_1) d\boldsymbol{x}_1 \tag{4.4}$$

for $T \leq \infty$, i.e., the expected long-term reward for choosing the actions $\boldsymbol{u} = \pi(\boldsymbol{x})$ in the considered MDP. The optimal value function V^* is defined as

$$V^*(\boldsymbol{x}) = \max_\pi V_\pi(\boldsymbol{x}) \tag{4.5}$$

$$= \max_{\boldsymbol{a} \in A} \int_S p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u}) \Big(r(\boldsymbol{x}) + \gamma V^*(\boldsymbol{x}') \Big) d\boldsymbol{x}' \tag{4.6}$$

and given the optimal value function V^*, the optimal policy can be obtained as

$$\pi^*(\boldsymbol{x}) = \arg\max_{\boldsymbol{u} \in A} \int_S p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u}) V^*(\boldsymbol{x}') d\boldsymbol{x}'. \tag{4.7}$$

The optimal value function can be computed for each $\boldsymbol{x} \in S$ as given in Eq. (4.6). Instead of trying to solve this equation directly, an iterative approach can be taken. Setting $V^0(\boldsymbol{x}) = 0$ the equation

$$V^{k+1}(\boldsymbol{x}) = \max_{\boldsymbol{u} \in A} \int_S p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u}) \Big(r(\boldsymbol{x}) + \gamma V^k(\boldsymbol{x}') \Big) d\boldsymbol{x}' \tag{4.8}$$

can be iterated and is guaranteed to converge to V^*. This approach is known as *Value Iteration*. Variations of the value iteration approach include *Policy Iteration*, where Eq. (4.8) is modified to solve for V_π iteratively and alternated with a policy improvement step that greedily optimizes the policy.

For finite state and action spaces, Equations (4.6) and (4.8) can be solved analytically. However, for continuous state and action spaces, these equations are only tractable if the dynamics is linear. For nonlinear dynamics, Eq. (4.8) can be solved approximately by choosing a suitable function approximator for V^k. Under some assumptions, approximate value iteration is guaranteed to converge to the true value function (Bertsekas and Tsitsiklis, 1996). One well studied class of approximate value iteration schemes represents the functions V^k as a linear combination of basis functions $\phi_1^k, \ldots, \phi_N^k$, i.e., $V^k(\boldsymbol{x}) \approx \sum_{i=1}^N \alpha_i^k \phi_i^k(\boldsymbol{x})$. These approximations have some desirable properties, e.g., the value function representation is easy to interpret and to handle. The choice of suitable basis functions $\phi_1^k, \ldots, \phi_N^k$ significantly affects the quality of approximation. Thus, finding suitable basis functions is crucial for approximate value iteration schemes. Typically, the basis functions are problem specific and must be hand-engineered for each task.

4.2.3 Related Work

The problem of deriving control laws when uncertainty is present has been considered in classic control theory for many years. In controller design, uncertainty may arise from modeling inaccuracies, the presence of (external) disturbances and the lack of data, as some system parameters are not known in advance but only during operation. Thus, a controller must be designed for a family of systems that is specified, e.g., by bounds for model parameters or for nonparametric uncertainties as bounds for the operator norm of the unknown dynamics. Robust control (Skogestad and Postlethwaite, 2005; Zhou and Doyle, 1998) designs a single controller that is provably stable for all systems in the specified family – often at cost of overall controller performance. Adaptive control (Narendra and Annaswamy, 2012; Tao, 2003) instead adjusts control parameters online in order to achieve prespecified performance, which can be computationally demanding. These methods rely on parametric dynamics models, which must be specified by an expert for each problem. In addition, both schemes require stability analysis of the dynamics system, e.g., via manually designed Lyapunov functions, which can be extremely challenging for complex, nonlinear systems.

Nonparametric system dynamics models are very appealing due to their high flexibility. They learn a dynamics model from data instead of relying on expert knowledge to pick a sufficiently accurate parametric form suitable for the dynamics. Nonparametric regression methods that learn the system dynamics from data have been considered, e.g., in (Deisenroth et al., 2015; Kocijan et al., 2004; Kupcsik et al., 2013; Pan and Theodorou, 2014; Schneider, 1996). In (Schneider, 1996), locally weighted Bayesian regression has been employed to model the system dynamics and uncertainty was treated as noise. To learn a policy, stochastic dynamic programming was applied on the discretized state space. The approaches (Deisenroth et al., 2015; Kocijan et al., 2004; Kupcsik et al., 2013; Pan and Theodorou, 2014) model the forward dynamics as a Gaussian process.

Gaussian processes as forward models allow to incorporate uncertainty about the system dynamics without the need to discretize the state space. GPs have also been employed to model the system dynamics in (Deisenroth et al., 2015; Rasmussen and Kuss, 2004; Rottmann and Burgard, 2009; Deisenroth et al., 2009; Bischoff et al., 2013; Kupcsik et al., 2013). The approaches PILCO (Deisenroth et al., 2015) and GPREPS (Kupcsik et al., 2013) are policy search approaches that rely on GPs as forward dynamics models. The PILCO algorithm (Deisenroth et al., 2015) employs GPs as forward dynamics models and conducts a search in the policy space, performing gradient ascent on the expected reward in an episodic setting. However, policy search approaches assume a parametric form of the policy to be given and typically optimize locally around a trajectory. Obtaining globally optimal policies with policy search is challenging and involves manual effort, e.g., choosing representative trajectories to be optimized simultaneously.

In contrast to policy search methods, value function based approaches do not impose limitations on the policy and learn globally optimal policies. Model-free approaches based on a value function (Riedmiller, 2005; Mnih et al., 2015; Lagoudakis and Parr, 2003) are a viable choice for small, finite state spaces or when the system is easily accessible and system interactions are not costly. However, in real-world scenarios, e.g., control of robotic systems, the required number of samples renders these methods impractical. Value function based approaches that learn a GP dynamics model were presented in (Deisenroth et al., 2009; Bischoff et al., 2013; Rasmussen and Kuss, 2004). The GPRL approach (Rasmussen and Kuss, 2004) assumes the

dynamics to be given as a GP and computes the value function at a fixed set of support points, that are used to train a noise-free GP to represent the value function. This approach is extended to learn the system dynamics and automatically select support points for the value function by the PVI algorithm (Bischoff et al., 2013). GPDP (Deisenroth et al., 2009) also employs GPs to represent the value function for either a given dynamics model or a learned GP forward dynamics. However, the approximation error of these approaches cannot be estimated straightforwardly as they rely on moment matching (Girard et al., 2002) as approximate inference which does not provide upper bounds for the error. Furthermore, the learning success is highly dependent on many hyperparameters as, e.g., the support points or the hyperparameters of the employed selection criteria, that must be hand-tuned. To the best of our knowledge, no approach to learn a globally optimal policy from scratch with high data efficiency, that does not require manual effort and allows for convergence analysis has been considered to date.

4.3 AVINQ

The AVINQ algorithm learns globally optimal control from interactions with the dynamics system. At the same time, we aim to minimize the amount of system interactions required for learning, while still assuming no prior knowledge about the system dynamics. In the following, we introduce AVINQ and, subsequently, analyze some convergence properties of the proposed algorithm.

4.3.1 Algorithm Sketch

The proposed algorithm proceeds in an episodic setting. In the beginning, a starting point is sampled from the initial state distribution $p(\boldsymbol{x}_0)$ and a rollout $(\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{x}_1), \cdots, (\boldsymbol{x}_{T-1}, \boldsymbol{u}_{T-1}, \boldsymbol{x}_T)$ with randomly chosen actions $\boldsymbol{u}_0, \cdots, \boldsymbol{u}_{T-1}$ is computed. An initial dynamics model $\boldsymbol{g}_0$ is trained on these observations. Based on this dynamics model, an approximation $\widetilde{V}^*$ of the optimal value function V^* is computed employing an approximate value iteration scheme. With this approximate optimal value function, the optimal policy given the dynamics model can be computed for any $\boldsymbol{x}$. A new rollout is computed with the currently optimal policy. This completes the episode. New episodes are computed until the optimal value function (and, thus, the optimal policy) converges. Algorithm 7 shows a high level overview of the proposed approach. The approximate value iteration scheme and our approach to learn basis functions for value function approximation are detailed in the following sections.

Approximate Value Iteration for GP Dynamics

To infer an optimal value function from the learned GP dynamics model, we propose an approximate value iteration approach. In value iteration, the Bellman Equation (4.6) is not solved directly, but iterated as

$$V^{t+1}(\boldsymbol{x}) = \max_{\boldsymbol{u} \in A} \int_S p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u}) \Big(r(\boldsymbol{x}) + \gamma V^t(\boldsymbol{x}') \Big) d\boldsymbol{x}', \tag{4.9}$$

while setting $V^0(\boldsymbol{x}) = 0$. We write A for the Bellman integral operator, that maps V^t to the right hand side of Eq. (4.9), i.e., $AV^t := V^{t+1}$. This iteration continues until $t = T$, if $T < \infty$ or

Algorithm 7 High level overview of AVINQ

Input: time horizon $T \leq \infty$, discount factor γ, reward function r
Output: approximation $\widetilde{V}^*$ of optimal value function V^*
  Sample $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$ and random actions $\boldsymbol{u}_0, \cdots, \boldsymbol{u}_{T-1}$
  Observe $\mathscr{D} = \{(\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{x}_1), \cdots, (\boldsymbol{x}_{T-1}, \boldsymbol{u}_{T-1}, \boldsymbol{x}_T)\}$
  $\widetilde{V}^0 \leftarrow 0$, $e \leftarrow 1$
  while not converged **do**
    Train GP dynamics model $\boldsymbol{g}_e$ on $\mathscr{D}$
    Compute $\widetilde{V}_e^1, \widetilde{V}_e^2, \cdots \widetilde{V}_e^*$ as in Alg. 8
    $\pi_e(\boldsymbol{x}) \leftarrow \arg\max_{\boldsymbol{u} \in A} \int p(\boldsymbol{x}' | \boldsymbol{x}, \boldsymbol{u}) \widetilde{V}_e^*(\boldsymbol{x}') d\boldsymbol{x}'$
    Sample $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$
    Compute $(\boldsymbol{x}_0, \pi_e(\boldsymbol{x}_0), \boldsymbol{x}_1), \cdots, (\boldsymbol{x}_{T-1}, \pi_e(\boldsymbol{x}_{T-1}), \boldsymbol{x}_T)$
    $\mathscr{D} \leftarrow \mathscr{D} \cup \{(\boldsymbol{x}_0, \pi_e(\boldsymbol{x}_0), \boldsymbol{x}_1), \cdots, (\boldsymbol{x}_{T-1}, \pi_e(\boldsymbol{x}_{T-1}), \boldsymbol{x}_T)\}$
    $e \leftarrow e + 1$
  end while

until the value function converges, if $T = \infty$. In our case, $p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u})$ is given as the prediction of the dynamics GP and is, thus, normally distributed. However, Eq. (4.9) cannot be solved in closed form for all but the linear kernel. Thus, we will solve this equation approximately, choosing a linear representation $\widetilde{V}^t(\boldsymbol{x}) = \sum_{i=1}^{N_t} \alpha^t \phi_i^t(\boldsymbol{x})$ of the value function iterate. In particular, we choose Gaussian basis functions as weighted sums of Gaussians (also called radial basis function networks) have the universal approximation property. With this choice of approximator, the integral in Eq. (4.9) becomes

$$R(\boldsymbol{x}, \boldsymbol{u}) := \int_S p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u}) \Big(r(\boldsymbol{x}) + \gamma V^t(\boldsymbol{x}') \Big) d\boldsymbol{x}' = r(\boldsymbol{x}) + \gamma \sum_{i=1}^{N} \alpha_i^k \int_S p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u}) \phi_i^k(\boldsymbol{x}') d\boldsymbol{x}'. \tag{4.10}$$

The integral of the product of Gaussians can be computed as

$$\int \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{s}, S) \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{y}, Y) d\boldsymbol{x} = (2\pi)^{-\frac{D}{2}} \det(Z)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} \boldsymbol{z}^\intercal Z^{-1} \boldsymbol{z} \right) \tag{4.11}$$

with $\boldsymbol{z} = \boldsymbol{s} - \boldsymbol{y}$ and $Z = S + Y$. As $p(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u})$ and $\phi_i^k(\boldsymbol{x}')$ are both Gaussian, we can apply this formula and as a result obtain an analytical solution to Eq. (4.10). However, the maximum of $R(\boldsymbol{x}, \boldsymbol{u})$ with respect to $\boldsymbol{u}$ remains analytically intractable. For any fixed $\boldsymbol{x}$, we can maximize $R(\boldsymbol{x}, \boldsymbol{u})$ with respect to $\boldsymbol{u}$ via gradient ascent. Thus, we can solve the right hand side of the Bellman Equation (4.9) for a finite number of $\boldsymbol{x}$, if we replace V^k by $\widetilde{V}^k$. To continue our approximate value iteration scheme, we need to find an approximation $\widetilde{V}^{t+1}$ of V^{t+1} of the form $\widetilde{V}^{t+1} = \sum_{i=1}^{N_{t+1}} \alpha^{t+1} \phi_i^{t+1}$. We will introduce an algorithm to find the basis functions ϕ_i^{t+1} in the following section. For now, lets assume that $\phi_1^{t+1}, \cdots \phi_{N_t}^{t+1}$ are given. It remains to find the weights $\alpha_1^{t+1}, \cdots, \alpha_{N_{t+1}}^{t+1}$ that minimize the distance of our approximation $\widetilde{V}^{t+1}$ to the true result of the Bellman equation $V^{t+1} \approx A\widetilde{V}^t$. We measure the distance of two scalar functions in L_2-norm, i.e., $\|f_1 - f_2\|_{L_2}^2 = \int (f_1(\boldsymbol{x}) - f_2(\boldsymbol{x}))^2 d\boldsymbol{x}$. In this case, the weights $\alpha_1^{t+1}, \cdots, \alpha_{N_{t+1}}^{t+1}$ of the best

Algorithm 8 Value Function Approximation with NQ

Input: GP dynamics $\boldsymbol{g}_e$, $\widetilde{V}^t = \sum_{i=1}^{N_t} \alpha_i^t \phi_i^t$, optional: basis function set $B = \{\phi_1^{t+1}, \cdots, \phi_{N_{t+1}}^{t+1}\}$ given by expert
Output: $\widetilde{V}^{t+1} = \sum_{i=1}^{N_{t+1}} \alpha_i^{t+1} \phi_i^{t+1}$
If B not given, compute with Alg. 9
$M_{i,j} \leftarrow \langle \phi_i^{t+1}, \phi_j^{t+1} \rangle_{L_2}$ for $i, j = 1, \cdots, N_{t+1}$
Get nodes $\xi_1, \cdots, \xi_K$ and weights $w_1, \cdots, w_K$ from NQ
$b_i^{t+1} \leftarrow \sum_{k=1}^{K} w_k \phi_i^{t+1}(\xi_k) A\widetilde{V}^t(\xi_k)$ for $i = 1, \cdots, N_{t+1}$
$\boldsymbol{\alpha}^{t+1} \leftarrow M^{-1} \boldsymbol{b}^{t+1}$

approximation of V^{t+1} in the linear subspace of L_2 spanned by the basis functions $\phi_1^{t+1}, \cdots, \phi_{N_{t+1}}^{t+1}$ can be obtained straightforwardly by projection. More precisely, the weight vector $\boldsymbol{\alpha}^{t+1}$ is the solution to the linear equation system

$$M\boldsymbol{\alpha}^{t+1} = \boldsymbol{b}^{t+1} \tag{4.12}$$

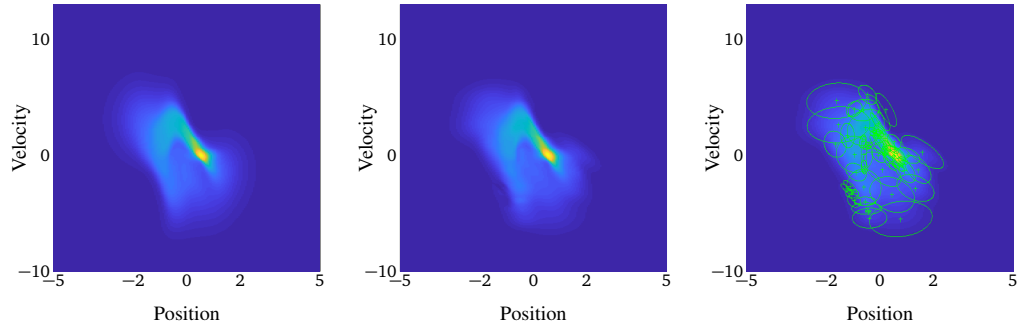
with the mass matrix $M_{i,j} = \langle \phi_i^{t+1}, \phi_j^{t+1} \rangle_{L_2} = \int \phi_i^{t+1}(\boldsymbol{x}) \phi_j^{t+1}(\boldsymbol{x}) d\boldsymbol{x}$ and the projections $b_i^{t+1} = \langle \phi_i^{t+1}, V^{t+1} \rangle_{L_2} = \int \phi_i^{t+1}(\boldsymbol{x}) V^{t+1}(\boldsymbol{x}) d\boldsymbol{x}$. The entries of M can be computed employing Eq. (4.11). Unfortunately, the entries of $\boldsymbol{b}^{t+1}$ are intractable, as they would require a closed form expression for V^{t+1}, which we try to approximate. However, as noted above we can compute the value of V^{t+1} at any point $\boldsymbol{s}$. Thus, we propose to approximate the integrals in $\boldsymbol{b}^{t+1}$ with numerical quadrature. Numerical quadrature estimates the value of an integral by a weighted, finite sum of function evaluations at certain points (also called quadrature nodes). The nodes and weights are given by the quadrature rule. With this approximation, we can estimate $\boldsymbol{b}^{t+1}$ with a finite number of values of V^{t+1}. Solving Equation (4.12), we obtain the weights for our approximation $\widetilde{V}^{t+1}$ of V^{t+1}. We write Π_B for the projection operator onto the linear subspace of L_2 spanned by the set of basis functions B, i.e., $\widetilde{V}^t = \Pi_B[V^t]$. Algorithm 8 summarizes the proposed projection approach.

If suitable basis functions for the considered problem are known a priori, e.g., given by an expert, they can be directly employed with the algorithm described above. If no suitable basis functions are given, which we believe is the most frequent case, a set of suitable basis functions can be learned iteratively as we will describe in the following section.

Learning Features for Value Function Approximation

A set of suitable basis functions for value function approximation is crucial for the success of any approximate value iteration scheme. In most cases, the choice of basis functions is nontrivial and involves manual effort. Subsequently, we introduce an algorithm to find suitable basis functions online during value iteration.

The proposed approximate value iteration scheme minimizes the L_2-error of the approximate solution. However, convergence guarantees for approximate value iteration approaches can typically be given, if upper bounds for the L_∞-error are known. Thus, we aim to find good basis functions



(a) Exact $A\widetilde{V}^{t-1}$ given $\widetilde{V}^{t-1}$ **(b)** Our Approximation $\widetilde{V}^t$ **(c)** $\widetilde{V}^t$ with Learned Basis Functions

Figure 4.2.: Finding suitable basis function for value function approximation with Alg. 9. Subplot (a) shows the exact value function iterate $A\widetilde{V}^{t-1}$ computed at each pixel. Plot (b) shows the projection onto the space spanned by the obtained basis functions, i.e., our approximation $\widetilde{V}^t$. In subplot (c), the obtained basis functions are indicated by 1σ-ellipsoids.

to decrease the L_2-error of the approximation while being able to reach any defined L_∞-error bound. More precisely, the goal is for any value function iterate $V^t(\boldsymbol{x})$ and given $\varepsilon > 0$ to find a set $B = \{\phi_1^t, \cdots, \phi_{N_t}^t\}$ of Gaussian basis functions, such that the projection $\widetilde{V}^t$ of V^t onto the linear space spanned by B (see Eq. (4.12)) fulfills

$$\|\widetilde{V}^t - V^t\|_{L_\infty} = \max_{\boldsymbol{x}\in S} |\widetilde{V}^t(\boldsymbol{x}) - V^t(\boldsymbol{x})| < \varepsilon. \tag{4.13}$$

We propose an iterative approach, that adds basis function after basis function until the criterion (4.13) is met. We start with $B = \{\}$ and define the projection error $\rho_B(\boldsymbol{x}) = |\Pi_B[V^t(\boldsymbol{x})] - V^t(\boldsymbol{x})|$. Note that we can evaluate ρ_B at any point $\boldsymbol{x} \in S$, although we cannot express V^t in closed form. We follow a greedy approach with respect to L_∞-error. Given the set $B = \{\phi_1^t, \cdots, \phi_l^t\}$ of previously added basis functions, we find a local maximum $\boldsymbol{x}_*$ of ρ_B via gradient ascent. As this optimization is non-convex, we cannot expect to find the global maximum of ρ_B. However, we perform gradient ascent from multiple starting points to make sure that we estimate the scale of ρ_B properly. If $\rho_B(\boldsymbol{x}_*) > \varepsilon$, we add another Gaussian basis function ϕ_{l+1}^t with mean $\boldsymbol{x}_*$ and covariance Σ_*. To estimate the covariance matrix Σ_*, we compute the Hessian of ρ_B at $\boldsymbol{x}_*$. It is well-known that the Hessian of the log of a normal density equals the negative inverse of its covariance matrix, i.e.,

$$\begin{aligned} -\log\left(\phi_{l+1}^t(\boldsymbol{x})\right) &= \frac{D}{2}\log(2\pi) + \frac{1}{2}\log(\det(\Sigma_*)) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_*)^\intercal \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{x}_*) \\ -\frac{\partial^2 \log\left(\phi_{l+1}^t(\boldsymbol{x})\right)}{\partial x_i \partial x_j} &= \left(\Sigma_*^{-1}\right)_{i,j} \end{aligned} \tag{4.14}$$

and following the chain rule

$$-\frac{\partial^2 \log\left(\phi_{l+1}^t\right)}{\partial x_i \partial x_j} = \left(\phi_{l+1}^t(\boldsymbol{x}_*)\right)^{-2} \frac{\partial \phi_{l+1}^t}{\partial \boldsymbol{x}_i} \frac{\partial \phi_{l+1}^t}{\partial \boldsymbol{x}_j} - \left(\phi_{l+1}^t(\boldsymbol{x}_*)\right)^{-1} \frac{\partial^2 \phi_{l+1}^t(\boldsymbol{x})}{\partial x_i \partial x_j}. \tag{4.15}$$

Algorithm 9 Find Basis Functions to Approximate V^t

Input: function handle V^t, tolerance ε
Output: basis function set B, s.t. $\|\Pi_B[V^t] - V^t\|_{L_\infty} < \varepsilon$

```
B ← {}, l ← 0, Ṽ^t = 0
Set function handle ρ_B(x) ← |Ṽ^t(x) − V^t(x)|
while ‖Ṽ^t − V^t‖_{L_∞} ≈ ρ_B(x_*) > ε do
    x_* ← max_{x∈S} ρ_B(x)
    Compute Σ_* with Eq. (4.15)
    φ^t_{l+1} ← N(x_*, Σ_*)
    B ← B ∪ {φ^t_{l+1}}
    Ṽ^t ← Π_B[V^t]
    Set function handle ρ_B ← |Ṽ^t(x) − V^t(s)|
end while
```

Thus, setting the Hessian of ϕ^t_{l+1} equal to the Hessian of ρ_B at $\boldsymbol{x}_*$, we can compute the covariance matrix Σ_* with Eq. (4.14) and (4.15). Please note that this estimate is very accurate, if ρ_B is locally close to a Gaussian shape (and exact, if ρ_B is Gaussian).

We add this new basis function to B and update ρ_B. Subsequently, the search for the maximum of ρ_B begins as above. This procedure is repeated until no more states $\boldsymbol{x}_*$ with $\rho_B(\boldsymbol{x}_*) > \varepsilon$ are found. Please note that this algorithm decreases the L_2-error of the approximation in every step. As the solution of an integral equation with a smooth kernel, the value function is continuous. Thus, the L_∞-error of our approximation also decreases with the number of basis functions (though not necessarily monotonously). Finally, it is important to note that this approach guarantees that the L_∞-error is smaller than ε only under the assumption that the global maximum of ρ_B is found. In our experiments, we found ρ_B well-behaved and did not encounter problems to find its global maximum.

Algorithm 9 shows the proposed approach for finding basis functions to approximate the value function iterates. Figure 4.2 illustrates our approach to find basis functions. In the following, we will analyze the convergence of AVINQ .

4.3.2 Convergence Analysis

Much research has been conducted towards convergence guarantees for value iteration and approximate value iteration approaches. While guarantees can be given straightforwardly for value iteration approaches on finite state and action spaces (Sutton and Barto, 1998), this task is a lot more challenging for approximate value iteration approaches on both finite or infinite state and action spaces. Guarantees that approximate value iteration will converge to a globally optimal policy can be given assuming the dynamics model is correct, if there are upper bounds for the L_∞-error of the approximate value function iterates (Bertsekas and Tsitsiklis, 1996). In (Bertsekas and Tsitsiklis, 1996), it is shown that

$$\limsup_{t\to\infty} \|V^* - V^t\|_{L_\infty} \leq \frac{2\gamma}{(1-\gamma)^2}\varepsilon, \tag{4.16}$$

if the approximation error is uniformly bounded, i.e., $\|\widetilde{V}^t - V^t\|_{L_\infty} < \varepsilon$ for all $t = 1, 2, \ldots$ and an $\varepsilon > 0$. To compute $\widetilde{V}^t$, we employ numerical quadrature to approximately project V^t onto to the space spanned by the basis functions. As upper bounds for the quadrature error can be given and our feature learning guarantees a certain precision, this condition holds for AVINQ given the true dynamics. In an episodic setting, however, there is no guarantee that the dynamics model will converge globally to the true dynamics. If the employed dynamics model incorporates uncertainty, exploration is encouraged as in regions of high uncertainty the expected reward averages over many states. Thus, uncertain states will be preferred over states with known low value. However, without explicit exploration, no firm convergence guarantees can be given.

4.4 Evaluation

We evaluate the proposed algorithm on the mountain car benchmark task and compare our results with other approaches in multiple experiments.

In the mountain-car domain (see, e.g., (Moore and Atkeson, 1995; Sutton and Barto, 1998)), a car starts at some point in a valley landscape and has to reach a certain point on the hill to the right side of the valley and stay there. However, the car's engine is not powerful enough to reach the goal directly from all starting positions. From the points in the valley, the car has to first drive in the opposite direction and gain momentum to reach the goal. The state space has two dimensions: position and velocity of the car, the magnitude of the control signal is limited to $u_{\max} = 4$.

We learn a globally optimal policy for the mountain car benchmark with AVINQ . As discount factor we choose $\gamma = 0.95$. The basis functions are learned with Algorithm 9, such that the approximation error is smaller than $\varepsilon = 0.07$. This is reached with up to 60 Gaussian basis functions in all iteration steps and episodes.

For the projection onto the linear space spanned by the basis functions, we employ adaptive Gauss-Kronrod quadrature for the integrals $\int \phi_i^t(\boldsymbol{x}) V^t(\boldsymbol{x}) d\boldsymbol{x}$. This adaptive quadrature scheme subdivides the integration area and estimates the quadrature error on each subregion by comparison of a higher and a lower order quadrature rule. As the Gauss-Kronrod quadrature rule is nested, i.e., the higher order rules contain all nodes of the lower order rules, this scheme is very efficient. To further increase the precision and efficiency of the quadrature, we perform a coordinate transformation, such that the basis function ϕ_i^t becomes axis aligned in the new coordinate system. This coordinate transformation concentrates the mass close to the center of the integration area making sure that the adaptive scheme does not stop prematurely. Furthermore, we normalize the height of the integrand $\phi_i^t(\boldsymbol{x}) V^t(\boldsymbol{x})$ to increase numerical stability and avoid cancellation effects. It is also well known that the mass matrix M tends to be badly conditioned, if the basis functions are not normalized with respect to L_2-norm. After normalizing the basis functions to solve Eq. 4.12 we found the solution of the linear system to be highly robust. Figure 4.3 shows the resulting value function after five episodes. As can be seen, the obtained value function captures the dynamics of the mountain car system very well. In Figure 4.4 the obtained discounted reward during learning is compared to PILCO and NFQ. The results were averaged over multiple runs with different initializations of the random number generator. AVINQ outperforms all tested algorithms. Please note also that PILCO does not learn a global policy, while the other approaches do.

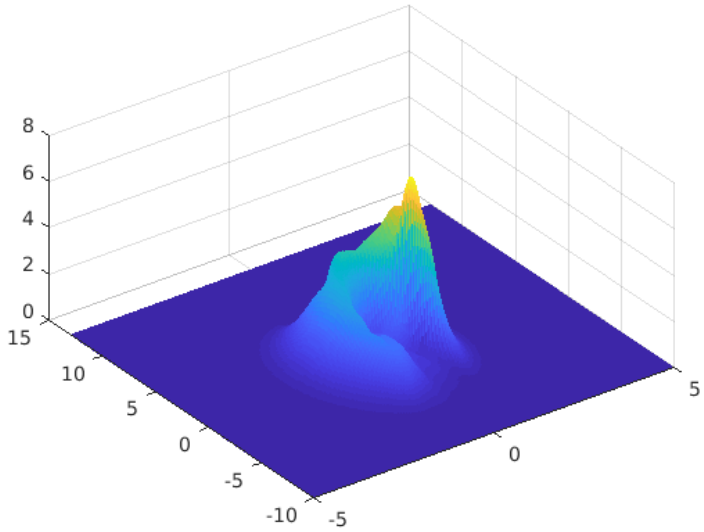


Figure 4.3.: Obtained value function for the mountain car task after five episodes.

4.5 Conclusion

For closed-loop control of initially unknown systems, learning approaches have proven to provide a viable alternative to classical control, that reduces the amount of required expert knowledge. Learning control approaches that build a model in order to infer good policies are advantageous as they typically require less data from interaction with the system. Especially when probabilistic dynamics models are employed, the data efficiency can be increased drastically, as the task-inherent uncertainty is explicitly considered. In this chapter, we introduced a value iteration approach that relies on a Gaussian process as forward dynamics model and handles uncertainty of the dynamics estimate. We conclude this chapter with a short summary of the proposed approach and a brief discussion of possible future work in this direction.

4.5.1 Summary of Contributions

In this chapter we introduced AVINQ, an approach to learn globally optimal control on continuous state and action domains from scratch, while maintaining high data efficiency. AVINQ employs Gaussian processes as forward dynamics models, which are learned in an episodic setting. To overcome intractability of the Bellman equation, AVINQ approximates the value function by projection onto a linear feature space.

The weights of the projected value function can be computed by solving a linear equation system. However, the right hand side of this equation system is composed of projections of the value function onto the feature spaces. These projections are intractable, as they require a closed form expression for the (intractable) value function. To enable these projections, numerical quadrature is employed. This allows to evaluate the value function at a finite number of points, which is feasible.

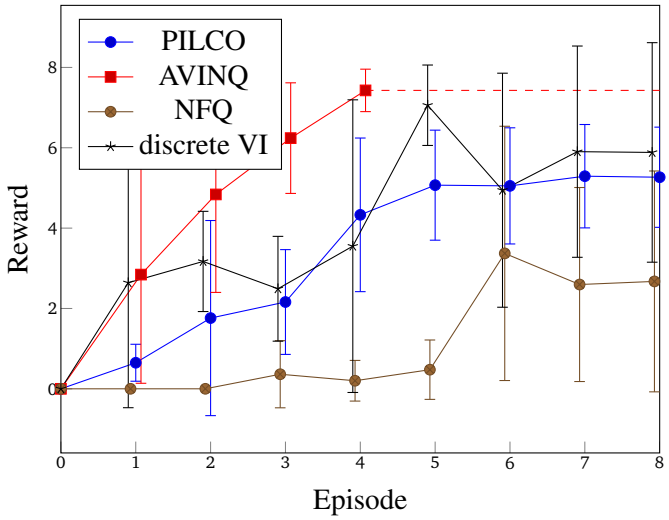


Figure 4.4.: Average reward as function of system interaction time for the mountain car task. One episode equals 3 seconds of system interaction time.

The quality of the obtained value function approximation depends strongly on the choice of features the value function is projected on. To achieve a highly accurate approximation and at the same time minimize manual tuning effort, we proposed a principled approach to find suitable features for the value function at hand. The introduced approach allows to control the approximation error to meet any given tolerance.

AVINQ benefits from Bayesian averaging over all plausible models by incorporating the uncertainty provided by the GP dynamics models. Furthermore, it requires no manual effort or problem specific knowledge. Upper bounds for the errors introduced by numerical quadrature and projection can be given.

4.5.2 Discussion and Next Steps

The challenging task of learning globally optimal control for nonlinear dynamics from noisy observations of the system behavior can be tackled by the proposed AVINQ algorithm. However, several open questions need to be solved to enable widespread deployment of AVINQ to real world problems.

Above, we assumed that the system stated can be perfectly measured during the rollouts. While we can handle measurement noise, AVINQ still depends on full (noisy) knowledge of the state. In many real world applications, however, the system state is not fully accessible, e.g., only some state dimensions can be observed. For these applications, the setting is described by partially observable MDPs (POMDPs) rather than the discussed MDPs. Multiple approaches can be considered to extend AVINQ to a POMDP setting. In a POMDP setting, the forward dynamics cannot be modeled by a Gaussian process sufficiently well. Instead, one of the various forms of latent variable models could be chosen to handle the incompleteness of knowledge. Furthermore, the lack of information

about the full system state introduces uncertainty to the value function. This value uncertainty needs to be taken into account to successfully learn control in a POMDP setting.

After learning is completed, the resulting policy can be analyzed and stability guarantees can be computed automatically (Vinogradska et al., 2016). These guarantees are crucial for real world applications, where a poor policy could cause damage to the system or precise control might be safety critical. In these cases, it is desirable to not only give stability guarantees after learning, but also to ensure safety during learning. Thus, safe exploration strategies need to be developed and incorporated into AVINQ.

5 Conclusion

Learning control has become a viable alternative to the derivation of control laws via classical control theory methods. Especially when combined with a dynamics model that explicitly incorporates uncertainty of the model predictions, these approaches have been tremendously successful and have attracted much attention in both the machine learning and control communities. Despite their many advantages, widespread deployment of these approaches in real world applications has yet to come. This thesis has made several contributions to overcome some of the limitations of learning control and move towards the goal of learning control in real world scenarios. In the following, we summarize the results of the thesis and, subsequently, discuss some open problems and possible future research directions.

5.1 Summary of Contributions

In this thesis, we addressed two major shortcomings of reinforcement learning to facilitate widespread deployment of learning control in real world applications.

Firstly, we provided an automatic tool to analyze stability of closed-loop control systems given a controller and a GP forward dynamics model. For the dynamics given as the mean of a GP, we presented an approach based on invariant sets to compute a stability region, i.e., a region of the state space, such that all trajectories starting in this region converge to the target state. From this asymptotic statement, we derived a result for finite time horizons and a criterion for stability in the presence of disturbances. When the full GP posterior distribution is considered for the dynamics model, multi-step-ahead predictions become intractable and must be approximated. We proposed an approximation based on numerical quadrature that can handle complex distributions with multiple modes and provides upper bounds for the approximation error. Exploiting these approximation error bounds, we provided a tool to compute stability regions for finite time horizons. This tool can compute a region of the state space, such that all trajectories starting from this region converge to the target with at least a certain guaranteed minimum probability. The automatic stability provers for the dynamics given as both GP mean and full GP posterior distribution achieve meaningful results in our experiments on standard benchmark problems. Furthermore, we analyzed the asymptotic behavior of closed-loop control systems with the dynamics given as a GP posterior distribution. Combining Markov chain theory with some basic properties of GPs, we gained an interesting insight: for certain, very common choices of kernel and mean prior these systems have a unique stationary distribution to which the system converges irrespective of the starting state.

Secondly, we employed the approximation developed in the first part of this thesis to make policy learning more robust and increase data efficiency. Value propagation is at the core of all reinforcement learning approaches. However, with Gaussian processes as dynamics models value propagation becomes intractable and must be approximated. We considered the two major classes of model-based reinforcement learning: policy search and value iteration. For both cases, we provided highly accurate approximate value propagation methods. In policy search, the expected

long-term reward is maximized with respect to the policy parameters. As the state distribution is intractable, we employed a numerical quadrature based approximation to approximate the state and, thus, the expected long-term reward. This approximation is highly precise, allows for analytical policy gradients and can be computed efficiently in parallel. The proposed policy search approach can also handle arbitrary starting state distributions in a principled way and without requiring manual effort. It outperforms all tested approaches on our benchmark problems. Considering infinite time horizons and exploiting our result from the first part of this thesis on the existence and uniqueness of a stationary distribution, we presented a policy search approach to directly optimize this distribution.

Value function based approaches determine an optimal policy from the value function. For nonlinear dynamics the optimal value function is in general intractable. Employing GPs as forward dynamics models, we proposed an approach that projects the value propagation steps onto a linear feature space. To enable this projection, we employed an approximation based on numerical quadrature. The presented approach additionally learns suitable features for value function representation online. Upper bounds for the projection error and the approximation error introduced by numerical quadrature can be obtained. The proposed value iteration approach successfully learns a globally optimal policy in a few seconds of system interaction time. Overall, we showed that robustness and data efficiency of learning control can be increased substantially when highly precise approximations are employed moving one step towards widespread deployment of learning control for real world tasks.

5.2 Open Problems

While this thesis proposed some steps to facilitate widespread deployment of learning control to real world tasks, several open questions remain on the way to this goal. In the following, we discuss possible future research directions.

Elimination of the Model Assumption. The automatic stability provers presented in this thesis make a statement about the behavior of the system as described by the model. However, a model is per definition approximate and may not perfectly capture the behavior of the real system. Interpreting the obtained stability statements as statements about the real behavior of the physical system implicitly assumes that the model is the ground truth. This assumption is not specific to the proposed approaches, but also common in classical control theory. However, it is desirable to obtain statements about the real system behavior. As our model is built on observations of the real system, there is at least some information we have about the true behavior. One possible approach would be to combine the proposed stability statements with PAC-Bayesian analysis (van der Vaart and van Zanten, 2011; Seeger, 2002) for the convergence of the model learning. Employing generalization error bounds for the model and extending the proposed stability analysis is a promising direction to obtain statements about the behavior of the true physical system.

Learning Robust Controllers. The behavior of physical systems is often not static but varies over time. Changes in the system dynamics may appear on different time scales, e.g., there are long-term effects as wear and tear and short-term effects as ambient temperature. These deviations from the learned dynamics model are typically not considered in learning control. Thus, such

changes in the dynamics can cause poor performance of the learned controller. A straightforward approach to handle long-term effects is to gather new observations and learn a new controller when the dynamics changes. However, this approach cannot cope with short-term effects on the dynamics model. Another way to address this issue it to optimize controllers to perform well on a class of dynamics models. For example, one could assume a maximum deviation of the dynamics from the initially learned model. To learn a policy that is suitable for the initially observed system as well as for the altered system behavior, the expected long-term reward must be considered for all models in the specified class. This significantly increases computational complexity and requires new methods to make this task computationally feasible.

Guaranteeing Safety During Learning. In this thesis we proposed methods to analyze the stability of a closed-loop control system after the dynamics and controller is learned. While this brings us one step closer to learning control for safety critical applications in the real world, safety of the learning itself has not been considered to date. For applications where human safety may be at risk or poor controllers could cause severe damage, learning approaches cannot be applied if one cannot ensure safety of any controller that is used, including the controllers during learning. The proposed stability analysis tools could be applied during learning to obtain a *safe region* where we can guarantee satisfactory performance of the controller. To guarantee safety during learning, the rollouts must then be limited to never leave the currently known safe regions. This requirement substantially limits exploration and might cause premature convergence with a model that does not completely capture the dynamics and, thus, may result in a poor controller. Hence, a new strategy to gain informative observations while respecting the safety limitations is necessary. For example, safe active learning approaches (Schreiter et al., 2015) could be combined with stability analysis and learning control.

Partially Observable Systems. The state of a physical system is in many cases not completely observable. For example, some state dimensions may be impractical to measure as sensors are expensive or not available. In this case, one only has partial information about the system state. To handle such systems, temporal information can be included to model the dynamics as Gaussian process latent variable model. Even one step ahead predictions are analytically intractable in this case and must be approximated. Typically, a great challenge for these models is to capture possible multimodality of the output distribution. One approach to tackle this problem is to employ a numerical quadrature based approximation to such models, enabling complex predictive distributions as an output. Furthermore, stability analysis of closed-loop control systems with GPLVM dynamics models has not been considered yet. While this is a challenging task, it seems promising to apply the methods proposed in this thesis to such models under the assumption that the dynamics model is smooth.

Convergence of Episodic Learning. In this thesis, we argued that reinforcement learning can significantly decrease manual effort required to solve real world control tasks. For practical applications without supervision, it is essential that learning is reliable and the task solution will eventually be learned successfully. However, both proposed policy learning approaches in this thesis involve gradient based optimization of nonconvex functions, which is only a local technique and will generally only find local optima. A combination with global search methods (Omidvar

and Li, 2011) could increase learning robustness within each episode. Arguably, the question whether the episodic setting of gathering data with the current policy, updating the model and learning an optimal policy given the model will eventually converge to an optimal policy for the true system dynamics. Such statements can typically be given only if the approach includes an efficient exploration strategy (Grande et al., 2014). The policy search approach proposed in this thesis does not explicitly include exploration and, thus, its convergence cannot be guaranteed. While we did not encounter divergence in any of our experiments, incorporation of an explicit exploration strategy into the learning framework and at the same time preserving the sample efficiency is a desirable goal.

Scaling Numerical Quadrature. In this thesis, we employed numerical quadrature to approximate multi-step-ahead predictions in Gaussian process forward models. Numerical quadrature provides highly accurate estimates of the state distribution and upper bounds for the approximation error can be obtained. However, it is well known that scaling numerical quadrature to high dimensional integrals is challenging. For univariate integrals, a family of quadrature rules (the Gaussian quadrature rules) exists, that reaches the maximum possible accuracy for smooth functions given the number of evaluation points. No such canonical family of optimal rules exists for multivariate integrals. Instead, optimal rules are known only in some cases, i.e., a set of geometries and exactnesses, and several principled (in general suboptimal) methods to construct multivariate rules from univariate quadrature have been studied. These approaches converge with a higher convergence rate than Monte Carlo approximations, however with growing number of dimensions their suboptimal construction renders them inferior to Monte Carlo sampling, which has a dimension-independent convergence rate and is typically the only feasible solution for very high dimensional problems. The exact number of dimensions, where numerical quadrature breaks down and becomes inferior to Monte Carlo integration is not known precisely. Several promising approaches to help scaling numerical quadrature have been proposed, as e.g., exploiting the problem's manifold structure, adaptive subdivision of the integration area, random projection onto lower dimensional spaces and many more. Thus, an important question that has not been answered yet is how many dimensions can be handled with numerical quadrature and any of these scaling approaches. A thorough analysis of these concepts and a series of experiments should be conducted to answer this question. For the analysis, we propose to pick a low dimensional problem as, e.g., the cart-pole system, and artificially increase the number of state dimensions. This dimensionality increase can be achieved by adding some (linearly independent) transformations of the state dimensions as, e.g., sine and cosine of the pole angle. Thus, the number of state dimensions increases, while the problem manifold maintains its (low) number of dimensions. Applying numerical quadrature combined with some approaches to scaling should give us an upper bound on how many dimensions are currently feasible with numerical quadrature.

Furthermore, one can think of hybrid approaches that combine numerical quadrature with some computationally lighter approaches (such as moment matching) to approximate the state distribution. While no theoretical guarantees can be given for such methods, they are promising for learning control, as they can provide more accurate estimates of the state distribution while requiring less computational resources than a highly precise numerical quadrature with guaranteed low approximation error.

5.3 Publications Included in this Thesis

Excerpts of the research presented in this thesis have led to the following publications.

J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, A. Romer, H. Schmidt, and J. Peters. Stability of controllers for gaussian process forward models. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, pages 545–554, 2016

J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, and J. Peters. Stability of controllers for gaussian process dynamics. *Journal of Machine Learning Research*, 18, 2017a

J. Vinogradska, B. Bischoff, and J. Peters. Numerical quadrature for probabilistic policy search. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017c

J. Vinogradska, B. Bischoff, and J. Peters. Approximate value iteration based on numerical quadrature. *IEEE Robotics and Automation Letters*, 2017b

Bibliography

G. Adomian. *Stochastic systems*. Mathematics in Science and Engineering. Elsevier Science, 1983.

A. A. Ahmadi and P. A. Parrilo. Converse results on existence of sum of squares lyapunov functions. In *Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6516–6521, Dec 2011.

A. A. Ahmadi, A. Majumdar, and R. Tedrake. Complexity of ten decision problems in continuous time dynamical systems. In *Proceedings of the 2013 American Control Conference*, pages 6376–6381, 2013.

C. W. Anderson, P. M. Young, M. R. Buehner, J. N. Knight, K. A. Bush, and D. C. Hittle. Robust reinforcement learning control using integral quadratic constraints for recurrent neural networks. *IEEE Transactions on Neural Networks*, 18(4):993–1002, July 2007.

C. G. Atkeson and J. C. Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 3557–3564. IEEE, 1997.

T. Beckers and S. Hirche. Stability of gaussian process state space models. In *Proceedings of the European Control Conference*. IEEE, 2016.

D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Number vol. 1 in Athena Scientific optimization and computation series. Athena Scientific, 2005.

D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-dynamic Programming*. Anthropological Field Studies. Athena Scientific, 1996.

B. Bischoff, D. Nguyen-Tuong, H. Markert, and A. Knoll. Learning control under uncertainty: A probabilistic value-iteration approach. In *Proceedings of the 21st European Symposium on Artificial Neural Networks, ESANN 2013*, 2013.

F. Blanchini. Set invariance in control. *Automatica*, 35(11):1747 – 1767, 1999.

J. Burkardt. Stroud – numerical integration in m dimensions. `https://people.sc.fsu.edu/~jburkardt/m_src/stroud/stroud.html`, 2014.

G. Chesi. Estimating the domain of attraction for uncertain polynomial systems. *Automatica*, 40 (11):1981–1986, 2004.

P.J. Davis, P. Rabinowitz, and W. Rheinbolt. *Methods of Numerical Integration*. Computer Science and Applied Mathematics. Elsevier Science, 2014.

M.P. Deisenroth. *Efficient Reinforcement Learning Using Gaussian Processes*. Karlsruhe series on intelligent sensor actuator systems. KIT Scientific Publ., 2010.

M.P. Deisenroth, C.E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

M.P. Deisenroth, P. Englert, J. Peters, and D. Fox. Multi-task policy search for robotics. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*, pages 3876–3881. IEEE, 2014.

M.P. Deisenroth, D. Fox, and C.E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2): 408–423, 2015.

E. Deutsch and M. Neumann. On the first and second order derivatives of the perron vector. *Linear Algebra and its Applications*, 71(Supplement C):57 – 76, 1985.

K. Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12:219–245, 2000.

Y. Engel, P. Szabo, and D. Volkinshtein. Learning to control an octopus arm with gaussian process temporal difference methods. In Y. Weiss, B. Schölkopf, and J.C. Platt, editors, *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 347–354. MIT Press, 2006.

G.A. Evans. The estimation of errors in numerical quadrature. *International Journal of Mathematical Education in Science and Technology*, 25(5):727–744, 1994.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages 457–464. MIT Press, 2007. ISBN 0-262-19568-2.

A. Girard, C. E. Rasmussen, J. Quiñonero Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pages 529–536, 2002.

G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

Robert Grande, Thomas Walsh, and Jonathan How. Sample efficient reinforcement learning with gaussian processes. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1332–1340, 2014.

N.R. Hansen. Geometric ergodicity of discrete-time approximations to multivariate diffusions. *Bernoulli*, 9(4):725–743, 08 2003.

F. Heiss and V. Winschel. Likelihood approximation by numerical integration on sparse grids. *Journal of Econometrics*, 144(1):62 – 80, 2008.

A. Hurwitz. Ueber die Bedingungen, unter welchen eine Gleichung nur Wurzeln mit negativen reellen Theilen besitzt. *Mathematische Annalen*, 46(2):273–284, 1895.

A. Jerri. *Introduction to Integral Equations with Applications*. A Wiley-Interscience publication. Wiley, 1999.

K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning (ICML 2007)*, pages 393–400. ACM, 2007.

H.K. Khalil. *Nonlinear control*. Prentice Hall, 2014.

R. Khasminskii and G.N. Milstein. *Stochastic Stability of Differential Equations*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2011.

H.J. Kim and A.Y. Ng. Stable adaptive control with online learning. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, pages 977–984. MIT Press, 2005.

E.D. Klenske, M.N. Zeilinger, B. Schölkopf, and P. Hennig. Nonparametric dynamics estimation for time periodic systems. In *Proceedings of the 51st Annual Allerton Conference on Communication, Control, and Computing*, pages 486–493. IEEE, 2013.

J. Ko and D. Fox. GP-BayesFilters: Bayesian filtering using gaussian process prediction and observation models. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008.

J. Kocijan, R. Murray-Smith, C.E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *Proceedings of the American Control Conference, (ACC 2004).*, volume 3, pages 2214–2219. IEEE, 2004.

A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI 2013)*, 2013.

H.J. Kushner. Finite time stochastic stability and the analysis of tracking systems. *IEEE Transactions on Automatic Control*, 11(2):219–227, 1966.

H.J. Kushner. *Stochastic Stability and Control*. Mathematics in science and engineering. Academic Press, 1967.

L. Kuvayev. Model-based reinforcement learning with an approximate, learned model. In *Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*, 1997.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4(Dec):1107–1149, 2003.

A.M. Lyapunov. *General Problem of the Stability Of Motion*. Doctoral dissertation, Univesity of Kharkov, 1892. Englisch Translation by A.T. Fuller, Taylor & Francis, London 1992.

J.M. Maciejowski and X. Yang. Fault tolerant control using gaussian processes and model predictive control. In *Proceeedings of the 2013 Conference on Control and Fault-Tolerant Systems (SysTol 2013)*, pages 1–12. IEEE, 2013.

A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control and verification of high-dimensional systems with dsos and sdsos programming. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, pages 394–401, 2014.

M. Masjed-Jamei. New error bounds for gauss-legendre quadrature rules. *Filomat*, 28(6):1281–1293, 2014.

A. McHutchon and C. E. Rasmussen. Gaussian process training with input noise. In *Advances in Neural Information Processing Systems 25 (NIPS 2011)*, pages 1341–1349, 2011.

S. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

C. A. Micchelli, Y. Xu, and H. Zhang. Universal kernels. *Journal of Machine Learning Research*, 7:2651–2667, December 2006.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.

A. W. Moore and C. G. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21(3):199–233, 1995.

J. Moore and R. Tedrake. Adaptive control design for underactuated systems using sums-of-squares optimization. In *Proceedings of the 2014 American Control Conference*, pages 721–728, June 2014.

J. Nakanishi, J.A. Farrell, and S. Schaal. A locally weighted learning composite adaptive controller with structure adaptation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.*, pages 882–889 vol.1, 2002.

K.S. Narendra and A.M. Annaswamy. *Stable Adaptive Systems*. Dover Books on Electrical Engineering. Dover Publications, 2012.

D. Nguyen-Tuong and J. Peters. Model learning in robotics: a survey. *Cognitive Processing*, (4), 2011.

E. Novak and K. Ritter. High dimensional integration of smooth functions over cubes. *Numerische Mathematik*, 75(1):79–97, 1996.

M. N. Omidvar and X. Li. *A Comparative Study of CMA-ES on Large Scale Global Optimisation*, pages 303–312. Springer Berlin Heidelberg, 2011.

Y. Pan and E. Theodorou. Probabilistic differential dynamic programming. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 1907–1915. Curran Associates, Inc., 2014.

A. Papachristodoulou and S. Prajna. *Analysis of Non-polynomial Systems Using the Sum of Squares Decomposition*, pages 23–43. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

P.A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, 2000.

T.J. Perkins and A.G. Barto. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3:803–832, March 2003.

J. Quiñonero-Candela, A. Girard, J. Larsen, and C.E. Rasmussen. Propagation of uncertainty in bayesian kernel models - application to multiple-step ahead forecasting. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 701–704, vol. 2, 2003.

C.E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, pages 751–759. MIT Press, 2004.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

M. Riedmiller. Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, pages 317–328, Berlin, Heidelberg, 2005. Springer-Verlag.

A. Rottmann and W. Burgard. Adaptive autonomous control using online value iteration with gaussian processes. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 2106–2111. IEEE, 2009.

E.J. Routh. *A Treatise on the Stability of a Given State of Motion: Particularly Steady Motion*. Macmillan and Company, 1877.

E.K. Ryu and S.P. Boyd. Extensions of gauss quadrature via linear programming. *Foundations of Computational Mathematics*, 15(4):953–971, 2015.

J. Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. In *Advances in Neural Information Processing Systems 9 (NIPS 1996)*, 1996.

J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint. Safe exploration for active learning with gaussian processes. In *Proceedings of the 2015 European Conference on Machine Learning and Knowledge Discovery in Databases*, ECMLPKDD'15, pages 133–149. Springer, 2015.

M. Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *Journal of Machine Learning Research*, 3(Oct):233–269, 2002.

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*, 529(7587):484–489, 2016.

S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, 2005.

B.S. Skrainka and K.L. Judd. High performance quadrature rules: How numerical integration affects a popular model of product differentiation. *Available at SSRN 1870703*, 2011.

J. Steinhardt and R. Tedrake. Finite-time regional verification of stochastic nonlinear systems. In H.F. Durrant-Whyte, N. Roy, and P. Abbeel, editors, *Robotics: Science and Systems VII*, pages 321–328. MIT Press, 2012.

A.H. Stroud. *Approximate calculation of multiple integrals*. Prentice-Hall series in automatic computation. Prentice-Hall, 1971.

E. Süli and D.F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

R.S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning (ICML 1990)*, pages 216–224. Morgan Kaufmann, 1990.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

G. Tao. *Adaptive Control Design and Analysis*. John Wiley & Sons, Inc., 2003.

U. Topcu, A. Packard, P. Seiler, and G. Balas. Help on sos [ask the experts]. *IEEE Control Systems*, 30(4):18–23, 2010a.

U. Topcu, A. K. Packard, P. Seiler, and G. J. Balas. Robust region-of-attraction estimation. *IEEE Transactions on Automatic Control*, 55(1):137–142, 2010b.

A. van der Vaart and H. van Zanten. Information rates of nonparametric gaussian process methods. *Journal of Machine Learning Research*, 12:2095–2119, July 2011.

J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, A. Romer, H. Schmidt, and J. Peters. Stability of controllers for gaussian process forward models. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, pages 545–554, 2016.

J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, and J. Peters. Stability of controllers for gaussian process dynamics. *Journal of Machine Learning Research*, 18, 2017a.

J. Vinogradska, B. Bischoff, and J. Peters. Approximate value iteration based on numerical quadrature. *IEEE Robotics and Automation Letters*, 2017b.

J. Vinogradska, B. Bischoff, and J. Peters. Numerical quadrature for probabilistic policy search. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017c.

J. Vinogradska, B. Bischoff, and J. Peters. Approximate value iteration based on numerical quadrature, 2017d. Patent DE 102017218811.1 (applied).

J. Vinogradska, B. Bischoff, and J. Peters. Numerical quadrature for probabilistic policy search, 2017e. Patent DE 102017211209.3 (applied).

J. Vinogradska, B. Bischoff, and J. Peters. Numerical quadrature for probabilistic policy search with infinite time horizons, 2017f. Patent DE 102017218813.8 (applied).

S. Wasowicz. On error bounds for gauss-legendre and lobatto quadrature rules. *Journal of Inequalities in Pure & Applied Mathematics*, 7(3):Paper No. 84, 7 p., 2006.

H. Xiao and Z. Gimbutas. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Computers & Mathematics with Applications*, 59(2):663 – 676, 2010.

K. Zhou and J.C. Doyle. *Essentials of Robust Control*. Prentice Hall Modular Series for Eng. Prentice Hall, 1998.

A Curriculum Vitæ

Education

09/2014 –08/2017	PhD student at Bosch Center for Artificial Intelligence and TU Darmstadt *Advisors:* Prof. Jan Peters and Dr. Bastian Bischoff *Thesis Title:* Gaussian Processes in Reinforcement Learning: Stability Analysis and Efficient Value Propagation
10/2011 – 08/2014	Master of Science in mathematics with minor in computer science at Universität Stuttgart *Advisor:* Prof. Volker Diekert *Thesis Title:* Automorphismen von Graphgruppen (engl.: Automorphisms of Graph Groups) *Final Grade:* 1.2 (among 5% best of year)
10/2008 – 09/2011	Bachelor of Science in mathematics with minor in computer science at Universität Stuttgart *Advisor:* Prof. Michael Eisermann *Thesis Title:* Effiziente Darstellungen von Permutationsgruppen (engl.: Efficient Representations of Permutation Groups) *Final Grade:* 1.9 (among 5% best of year)
09/1999 – 07/2008	Abitur (German secondary school diploma qualifying for university admission) *Final Grade:* 1.0 (among 5% best of year)

Working Experience

10/2009 – 09/2013	Teaching Assistant and Research Assistant in Advanced Mathematics at Universität Stuttgart

Miscellaneous

Languages:	German (native), Russian (native), English (fluent), French (basic)
Computer Skills:	Matlab (used most), C/C++ (basic), LaTeX (good)
Awards:	Stiftung Werner von Siemens Ring Jungwissenschaftler 2017 (Junior Scientist of the year 2017)

B Publications

Excerpts of the research presented in this thesis have led to the following publications.

B.1 Conference and Journal Publications

J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, A. Romer, H. Schmidt, and J. Peters. Stability of controllers for gaussian process forward models. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, pages 545–554, 2016

J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, and J. Peters. Stability of controllers for gaussian process dynamics. *Journal of Machine Learning Research*, 18, 2017a

J. Vinogradska, B. Bischoff, and J. Peters. Numerical quadrature for probabilistic policy search. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017c

J. Vinogradska, B. Bischoff, and J. Peters. Approximate value iteration based on numerical quadrature. *IEEE Robotics and Automation Letters*, 2017b

B.2 Patents

J. Vinogradska, B. Bischoff, and J. Peters. Numerical quadrature for probabilistic policy search, 2017e. Patent DE 102017211209.3 (applied)

J. Vinogradska, B. Bischoff, and J. Peters. Numerical quadrature for probabilistic policy search with infinite time horizons, 2017f. Patent DE 102017218813.8 (applied)

J. Vinogradska, B. Bischoff, and J. Peters. Approximate value iteration based on numerical quadrature, 2017d. Patent DE 102017218811.1 (applied)